Desarrollo De Una Aplicación Web Para La Administración De Inventario De Hardware Y Licencias En El Laboratorio María Salomé S.A.S

Jesús David Montaño Zapata

Director

Mg. Carlos Alberto Mejía Rodríguez

Informe final de prácticas para optar por el título de Ingeniero de Sistemas

Universidad Popular Del Cesar

Ingeniería De Sistemas

Aguachica, Cesar

2023

Nota de aceptación

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Carlos Alberto Mejía Rodríguez
Director

_____
Lina Marcela Arévalo Vergel
Jurado

_____
Danny Jhoan Ríos Barona
Jurado

Aguachica, Día_____ Mes_____ Año_____

Dedicatoria

Este proyecto va dedicado en primer lugar, a mi familia, quienes han sido mi apoyo incondicional y han creído en mí en cada etapa de esta travesía universitaria. Posteriormente, a los docentes, cuya paciencia y sabiduría han sido fundamentales para mí desarrollo intelectual, profesional y humano. También a los amigos que encontré y compañeros de clase, quienes compartieron risas, desafíos y conocimientos, haciendo que esta experiencia fuera más enriquecedora. Por último, a todos aquellos que, de una u otra manera, me brindaron su aliento y confianza, inspirándome a seguir adelante incluso en los momentos más difíciles.

Este proyecto es fruto del esfuerzo, del hambre de conocimiento y el anhelo de aplicar lo aprendido, es por ello que les dedico este trabajo, porque sin todas esas voces de aliento, de guía y de consuelo en algunos momentos, no hubiese llegado hasta este punto.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todos aquellos que contribuyeron de manera invaluable al éxito de este proyecto. Sin su valioso respaldo, este logro no habría sido posible.

En primer lugar, extiendo mi gratitud al ingeniero Carlos Mejía, mi director de proyecto, cuya orientación experta y consejos desempeñaron un papel fundamental en el desarrollo de este informe. Sus conocimientos han sido la piedra angular que ha dado forma a cada aspecto de este proyecto.

A mi familia, les agradezco profundamente por su comprensión, paciencia y apoyo incondicional a lo largo de esta extensa travesía académica. Su respaldo emocional ha sido el combustible que me ha impulsado a superar obstáculos y perseverar.

Finalmente, quiero expresar mi agradecimiento a la Universidad Popular del Cesar Seccional Aguachica por proporcionarme la oportunidad de aprender y crecer como profesional. Estoy agradecido por la plataforma educativa que me han brindado y por el ambiente propicio para el desarrollo de habilidades y conocimientos.

Tabla de contenido

Lista de tablas

Lista de figuras

IDENTIFICACIÓN DE LA PRÁCTICA Y RESPONSABILIDAD INSTITUCIONAL.

Practica académica: DESARROLLO DE UNA APLICACIÓN WEB PARA LA ADMINISTRACIÓN DE INVENTARIO DE HARDWARE Y LICENCIAS EN EL LABORATORIO MARÍA SALOMÉ S.A.S.

Institución: Laboratorio María Salomé S.A.S.

Responsable: Wbeimar Cardona Ramírez

Línea: Ingeniería del Software

Campo de Aplicación: Sistemas de información empresarial

1. Introducción

En la era digital actual, la eficiente gestión de inventario de hardware y licencias se ha vuelto crucial para optimizar los procesos en entornos empresariales. El Laboratorio María Salomé S.A.S, consciente de la importancia de mantener un control preciso de sus recursos tecnológicos, ha decidido embarcarse en el desarrollo de una aplicación web dedicada a la administración de su inventario.

La creciente complejidad de los entornos tecnológicos y la necesidad de asegurar un uso eficiente de los recursos motivan la creación de esta aplicación. La falta de herramientas especializadas ha llevado a situaciones donde la administración de activos se vuelve manual, propensa a errores y carente de una visión integral. La experiencia práctica en este ámbito ha resaltado la urgencia de contar con una solución que permita un seguimiento efectivo de hardware y licencias.

Se realizará una revisión exhaustiva de la literatura relacionada con la gestión de inventario, tecnologías web y prácticas recomendadas en el desarrollo de aplicaciones empresariales. La metodología de desarrollo ágil permitirá una adaptación continua a las necesidades del Laboratorio María Salomé S.A.S.

Se espera que este proyecto proporcione al Laboratorio María Salomé S.A.S una herramienta efectiva para el seguimiento y administración de su inventario, mejorando la eficiencia operativa y optimizando el uso de sus recursos tecnológicos. A lo largo del desarrollo, se pretende demostrar que una aplicación web personalizada puede ser una inversión valiosa para la gestión integral de activos.

2. Planteamiento del problema

2.1 Descripción del problema

La empresa Laboratorio María Salomé S.A.S, se enfrenta a desafíos significativos en la gestión de su inventario de hardware, licencias y garantías, con consecuencias directas en la eficiencia operativa de la compañía.

La carencia de una solución integral para la administración de estos elementos complica la asignación, seguimiento y control de los equipos de cómputo, generando retrasos en la distribución, dificultades en el mantenimiento y obstáculos en la gestión de eventos y reparaciones. Esta problemática ha sido ampliamente documentada por estudios especializados en gestión de inventarios, como el realizado por la revista "Journal of Operations Management" (Browning, 2023).

La falta de un sistema de información que relacione adecuadamente cada computadora con sus respectivas licencias de software representa un desafío crítico. Esta carencia obstaculiza la supervisión precisa de las licencias, aumentando el riesgo de incumplimiento normativo y posibles implicaciones legales. La "International Journal of Information Management" (Dwivedi, 2002), ha subrayado la importancia de sistemas de información efectivos en la gestión de licencias.

La ausencia de un acceso rápido a las garantías de daño de los equipos añade un desafío adicional. La carencia de un sistema eficiente para rastrear y registrar garantías provoca demoras en la resolución de problemas de hardware, según investigaciones de la "Association for Computing Machinery" (Machinery, 2022). Estas demoras pueden resultar en tiempos prolongados de inactividad y costos adicionales.

Para abordar estas problemáticas, es imperativo desarrollar una Aplicación Web que permita una gestión integral y efectiva del inventario de hardware y licencias. Esta solución contribuirá a optimizar los procesos internos, mejorar la toma de decisiones y reducir los riesgos asociados a la gestión de activos tecnológicos. Estos principios están respaldados por las mejores prácticas definidas por el "Information Systems Audit and Control Association" (ISACA, 2022), que destaca la importancia de soluciones tecnológicas para la gestión eficiente de activos.

El desarrollo de la aplicación conlleva la aplicación e integración de tecnologías actuales y relevantes de desarrollo. Esta iniciativa permitirá al autor aplicar los conocimientos adquiridos a lo largo del proceso formativo en el programa de Ingeniería de Sistemas de la Universidad Popular del Cesar, Seccional Aguachica. Se pondrán en práctica habilidades en el desarrollo web, diseño de bases de datos, implementación de interfaces de usuario, entre otras, alineadas con las tendencias definidas por la "Association for Computing Machinery" (Machinery, 2022)y la "International Association of Software Architects" (Iasa Global, 2021)

Este planteamiento del problema proporciona una base sólida para la ejecución de la propuesta, respaldada por investigaciones y principios reconocidos en el ámbito de la gestión de activos y sistemas de información.

3. Justificación

En un entorno empresarial en constante evolución y dinámico, la capacidad de adaptación a los cambios tecnológicos se ha convertido en un factor determinante para el éxito. En este contexto digital, destacar en el mercado implica la implementación de soluciones tecnológicas que impulsen la productividad y posicionamiento de las empresas.

La creación de un aplicativo web se justifica plenamente, ya que brindará soluciones tecnológicas fundamentales para optimizar la gestión de inventario de hardware, licencias y garantías en la empresa Laboratorio María Salomé S.A.S. Este proyecto se justifica en la necesidad de abordar estos desafíos mediante el desarrollo de aplicaciones web utilizando tecnologías clave como Java, Django, Python, etc.

Según (Acosta, 2021), “la demanda creciente de portales web ha impulsado la comunicación constante con los usuarios”. Un aplicativo web se presenta como una herramienta esencial para mejorar la eficiencia operativa, ofreciendo a los clientes una plataforma que permite acceso fácil y rápido a información crucial sobre productos y servicios. Además, las aplicaciones web proporcionan un medio efectivo para la interacción constante con los usuarios, según lo respaldado por el artículo "The Role of Web Applications in Business: A Review" publicado en "International Journal of Computer Applications" (Kumari Kusuma, 2017).

Desde la perspectiva del desarrollo de backend, el proyecto se basa en los principios destacados por Vallejo (2022). La aplicación web requerirá conocimientos sólidos en lenguajes como Java, Django y Python, así como en la gestión eficiente de bases de datos y API, alineándose con las mejores prácticas definidas en el "Journal of Web Engineering" (Lübeck, Alemania, 2021).

Este proyecto no solo aborda desafíos actuales en la gestión de inventarios y desarrollo de aplicaciones web, sino que también se enriquece con el respaldo de investigaciones y tendencias recientes, garantizando su relevancia académica y su contribución al avance del conocimiento en el área de desarrollo tecnológico.

4. Objetivos

4.1 Objetivo General

Desarrollar de una aplicación web para la administración de inventario de hardware y licencias en el laboratorio María Salomé S.A.S.

4.2 Objetivos específicos

- Definir los requerimientos específicos de la aplicación para la gestión de inventario, licencias y garantías en colaboración con los usuarios y la empresa.
- Diseñar la arquitectura y la estructura de la aplicación, incluyendo la base de datos y la interfaz de usuario siguiendo los requerimientos.
- Desarrollar la aplicación web con funcionalidades de gestión de inventario por usuario y tipo de hardware.
- Desplegar la aplicación en el entorno de producción de Laboratorio María Salomé S.A.S, y realizar las pruebas correspondientes.

5. Marco Referencial

5.1 Marco de Antecedentes

Revisar estudios académicos o científicos apoya al investigador a establecer fundamentos sólidos para escoger la opción más adecuada en cuanto a métodos, herramientas o dispositivos a emplear, contribuyendo así a enriquecer su experiencia para resolver el problema, responder a la pregunta de investigación y alcanzar los propósitos definidos.

5.1.1 Antecedentes Internacionales

En el ámbito internacional se encuentra el trabajo realizado por (Sánchez, 2021), titulado "Academic experience in rapid development of web information systems with Python and Django", el cual tuvo como objetivo principal el desarrollo de sistemas de información web mediante el uso del marco de trabajo Django del lenguaje de programación Python con su patrón de desarrollo Modelo-Plantilla-Vista (MTV), Los proyectos que se presentan son sistemas web prototipo con operaciones para crear, leer, actualizar y eliminar (CRUD, en inglés) registros de la base de datos. Se concluye que Python es un lenguaje de programación adecuado para un fácil y rápido dominio, Este proyecto se centró en la gestión de inventario y procesos de despacho de la empresa Chori Rico.

Python es un lenguaje de programación libre, de alto nivel y multiplataforma inventado por Guido Van Rossum en 1989 (Carrion, 2018); es decir, Python funciona sin costo es Windows, Unix, Linux, y otros sistemas operativos con una sintaxis más simple y elegante que la de otros lenguajes de programación basándose en lo siguiente:

Django es un marco de trabajo (framework) para el desarrollo de aplicaciones web usando Python. Django considera algunas funcionalidades listas para usar para facilitar el

desarrollo de aplicaciones web. Como resultado, no es necesario escribir todo el código ni usar tiempo para buscar errores de código en el framework. Es decir, mediante Django, el desarrollo de sistemas de información web puede ser rápido, seguro, escalable y también fáciles de mantener (carrion, 2015). Django representa un marco de trabajo para el desarrollo rápido de sistemas de información web con Python.

Por otro lado (López, 2015) presentan un trabajo de grado titulado "Aplicación web desarrollada en Python usando el Framework Django" donde se desarrolló de una aplicación web con características multimedia a lo largo de todas sus fases, desde el análisis de requisitos hasta su puesta en funcionamiento. El objeto concreto de la aplicación será el de asistir a los usuarios en la creación de collages fotográficos con las imágenes que ellos aporten de acuerdo a unas plantillas predefinidas. La aplicación se desarrolló en el lenguaje de programación Python haciendo uso del framework para desarrollo web Django.

Una última investigación que demuestra la agilidad y facilidad que nos brinda Python junto con su framework para el desarrollo web es la realizada por Ruby Carolina y Andrés Felipe (Andres, 2021) Descrito en su artículo titulado "Desarrollo de un aplicativo web en Python mostrando la necesidad de la experiencia UI basada en el Front - End según lo estipulado por la norma ISO 25010 a través de la metodología SCRUM.".

5.1.2 Antecedentes nacionales

En el ámbito nacional se encuentra el trabajo realizado por (Díaz Marín, 2016), titulado "Aplicativo web para gestionar el préstamo de herramientas de trabajo", el cual tuvo como objetivo principal la adopción de un aplicativo web para gestionar el préstamo de herramientas de trabajo tomando como base un laboratorio de una universidad, con la que se intenta mejorar la forma en la cual se realiza el proceso de préstamo de equipos e inventario del mismo y que no

solo brinde una posible solución al problema que se presenta en la institución, sino que también se puede modificar de tal manera que sirva para cualquier organización que requiera hacer préstamo de elementos.

Por otro lado (Dávila Bernal, 2020) Presenta su trabajo de grado titulado “Desarrollo de un prototipo de sistema de facturación e inventarios para tiendas minoristas de ropa que mediante redes neuronales mejore el control de inventarios” donde se propone el desarrollo de un software de facturación POS con una red neuronal artificial implementada para calcular la demanda de los productos, disminuyendo así la incertidumbre que esto genera y por consiguiente reducir el riesgo de tomar decisiones desacertadas en la gestión de inventarios. Además, se implementará un modelo matemático EOQ que utilizará la demanda predicha para calcular la cantidad optima de productos a pedir enfocado a minimizar los costos generados por el pedido y la manutención de productos.

Una última investigación hace referencia a una aplicación de gestión, en donde (Alvarez, 2012) demuestra con su proyecto de grado titulado “Desarrollo De Un Módulo Para El Control De Préstamos De Los Escenarios Deportivos En La Universidad Francisco De Paula Santander Ocaña” donde se propone la solución a la falta de una herramienta que permita el control de los escenarios deportivos en la universidad para facilitar la gestión de los procesos dentro de la oficina que permita mantener la información organizada sin riesgos de perder sus integridad y que permita la generación de reportes de forma rápida y eficaz.

5.1.3 Antecedentes regionales

En el ámbito nacional se encuentra el trabajo realizado por (Clavijo, 2018) que lleva por titulo “Implementación de un sistema de información ERP (enterprise resource planning) que permita los procesos de facturación, inventarios, clientes, proveedores y cartera en el almacén Frenautos 2001 en Aguachica-Cesar” la elección de este proyecto como punto de referencia se basa en la calidad de su investigación y la aplicación efectiva de la metodología de Programación Extrema (XP), garantizando así la solidez y eficiencia del aplicativo desarrollado con la finalidad de que el usuario realice sus funciones de forma más efectiva y práctica. La aplicación implementada presenta una interfaz de fácil manejo y como resultado satisface las necesidades de los requisitos funcionales del cliente.

Por otro lado, el proyecto que lleva por título “Implementación de un sistema de información en entorno Web para el manejo de inventario, facturación y procesos con proveedores de la empresa dentales de oriente en Aguachica, Cesar.” Elaborado por (Palomino Heider, 2018) Nos muestra una aplicación exitosa de metodologías de investigación y desarrollo de software, especialmente el enfoque de Ingeniería Web (IWeb), ha demostrado ser efectiva en la construcción de un sistema integral para el manejo de inventario, facturación y procesos con proveedores.

La adaptabilidad de la interfaz de usuario dinámica y fácil de manejar, junto con la sistematización de procesos administrativos y la disponibilidad de servicios en línea, refuerzan la importancia de integrar avances tecnológicos en nuestro propio proyecto con Django.

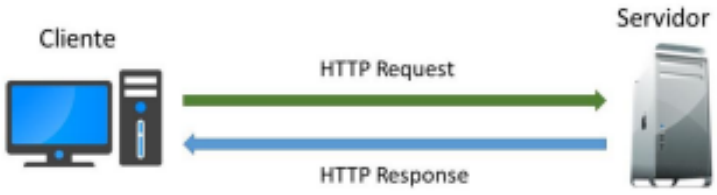
5.2 Marco Teórico

5.2.1 Aplicación Web

Una aplicación web, es una aplicación distribuida, la cual es accedida por los clientes finales mediante la conexión con un servidor vía web por una determinada red, ya sea Internet o una Intranet. Es posible en término general, describirla como un programa informático que es ejecutado en un navegador del cliente, debido a que la codificación de la misma es con algún lenguaje soportado por éstos, como, por ejemplo, HTML con JavaScript, entre otros. Dichos navegadores serán capaces de renderizar la aplicación para mostrarla a los usuarios y que puedan interactuar con ella. (Universidad de Alcalá, 2022)

La comunicación existente entre los servidores y los clientes finales se realiza mediante el protocolo HTTP (Hypertext Transfer Protocol). Dicho protocolo de la capa de aplicación sigue el esquema petición – respuesta entre los clientes y servidores, el cual, permite la transferencia de información en la World Wide Web. La forma que tiene el usuario de realizar peticiones al servidor es mediante las URLS (Wikipedia, 2017). A continuación, se muestra un resumen de un intercambio de peticiones entre un servidor y un cliente:

Figura 1 ***Diagrama petición – respuesta HTTP.***



Nota. Figura tomada de (Vergara, 2017)

5.2.2 Framework Web

Es posible definir un framework web como un conjunto de componentes software que construyen un diseño reutilizable que facilita y agiliza el desarrollo de una aplicación Web robusta. Se puede considerar como una aplicación web incompleta y configurable a la que se le pueden añadir nuevas funcionalidades para conseguir el comportamiento deseado por el grupo de programadores. Estas herramientas y funcionalidades pueden ser: librerías de clases, funciones, scripts, etc. (Vergara, 2017)

Como objetivos fundamentales de los frameworks en general, se tienen:

- Acelerar el proceso de desarrollo, ya que el programador de la aplicación no necesita plantearse una estructura global de ella, debido a que el framework le proporciona un esqueleto predefinido sobre el que tendrá que trabajar.
- Reutilizar código ya existente y por lo tanto evitar reescribir un mismo fragmento de código.
- Promover buenas prácticas de desarrollo como el uso de patrones de diseño.
- Es más fácil disponer de herramientas adaptadas al framework concreto para agilizar y facilitar el desarrollo.
- Permitir tener una mayor fiabilidad de las páginas de la aplicación web ya que el Framework ha sido previamente sometido a diferentes pruebas.

Existen diferentes tipos de Frameworks Web orientados a distintos lenguajes de programación, funcionalidades, etc. A continuación, se muestra en una tabla los Frameworks más conocidos, junto con una breve descripción de los mismos:

Tabla 1 ***Tabla de Frameworks de programación Web.***

Framework	Lenguaje	Descripción breve
Spring MVC	Java	Sigue el modelo MVC para aplicaciones de Internet y aplicaciones de seguridad. Ofrece amplia gama de servicios.
JSF	Java	Framework Java que no tiene dependencias externas, posee muchas características.
Struts 2	Java	Framework Java para aplicaciones web con Java EE.
Django	Python	Framework Python que promueve el desarrollo rápido y el diseño limpio.
Pylons	Python	Framework MVC de código abierto que enfatiza la flexibilidad y el desarrollo rápido. Usa el estándar WSGI.
Ruby on Rails	Ruby	Basado en Ruby, orientado al desarrollo de aplicaciones Web.
CakePHP	PHP	Framework MVC para PHP de desarrollo rápido.

Nota. tabla tomada de (Vergara, 2017)

A finales de la década de los 90 el diseño de una aplicación web se programaba mayoritariamente mediante un estándar muy popular llamado Common Gateway Interfaces (CGI) (ionos, 2020). Cuando se programaba una aplicación en CGI, el programador tenía que estar al control de todos los aspectos. El programador tenía que encargarse de la correcta comunicación entre sus ficheros HTML y las bases de datos y una vez terminada esta comunicación establecer el fin de la misma. Desde un punto de vista práctico, este método podía ser manejable si el programador estaba trabajando con poco flujo de información.

5.2.3 Framework Django

Django es un framework de desarrollo web de código abierto, escrito en Python. Es un framework sencillo de instalar, que posee una documentación muy completa y extensa, y que viene con un Sistema Gestor de Bases de Datos ya incorporado (sqlite3), a pesar de que no es el único que puede utilizar, el funcionamiento desde la aplicación es el mismo en cualquiera, ya que emplea un sistema de modelos que indica a los Gestores lo que deben hacer. En este momento, el entorno emplea python3, por lo que, al ir actualizando, la versión del lenguaje de programación, los métodos y librerías que se utilicen no se quedan obsoletos. Esto ayudara a la hora de tener que realizar futuros mantenimientos en las aplicaciones desarrolladas en Django (django, 2005).

El equipo responsable de la producción y mantenimiento de numerosos portales de noticias, The World Online, mostraba ciertas restricciones de tiempo para añadir nuevas características a las aplicaciones Web y, además, había una intensa demanda de periodismo. De la necesidad de crear un gran número de entradas correctas en tan poco tiempo, surgió la aparición del framework Web Django, donde lo fueron modificando constantemente, para irlo adaptando a sus necesidades, durante 2 años. Por lo tanto, sigue los principios DRY (Don’t Repeat Yourself; No Te Repitas) para evitar escribir código duplicado e invertir el menor esfuerzo posible para ahorrar tiempo en el desarrollo de la aplicación.

Este Framework Web, fue nombrado de esta manera en alusión al guitarrista de Jazz gitano Django Reinhardt (1910 – 1953) de 1930 hasta principios de los años 50. El cual es considerado uno de los mejores guitarristas de ese estilo.

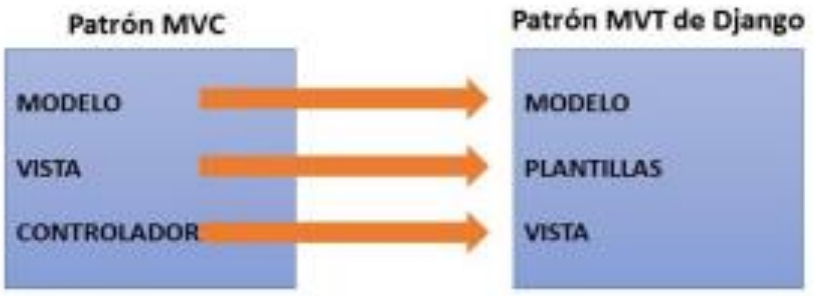
5.2.3.1 Patrón Modelo – Vista – Controlador en Django

En cuanto a la arquitectura de Django, cabe comentar que fue diseñado para que exista un acoplamiento débil entre las distintas partes de la aplicación y una clara separación de las mismas. Por lo que, como se puede deducir, es posible realizar cambios en una parte específica de la aplicación sin afectar a las demás piezas de la misma. por lo tanto, tiene una implementación especial del Modelo – Vista – Controlador.

- **Modelos:** los modelos van a corresponder con la parte de la aplicación que define la estructura de la base de datos y se encarga de la comunicación con ella. Se define que estructura seguirá toda la información que se maneje en la aplicación.
- **Vistas:** las vistas van a corresponder a la parte interfaz del usuario, con el código que elige que datos pedirle o mostrarle al usuario en cada momento.
- **Controladores:** corresponde a la parte de la aplicación que elige que vistas ejecutar en respuesta a las acciones o peticiones del usuario.

En las aplicaciones Django, debido a que el Controlador es manejado por el mismo Framework, y la parte más significativa se la llevan los modelos (Models), vistas (Views) y plantillas (Templates), Django es conocido también como un Framework MTV (Models – Views – Templates). Se pasa a describir posteriormente este significado.

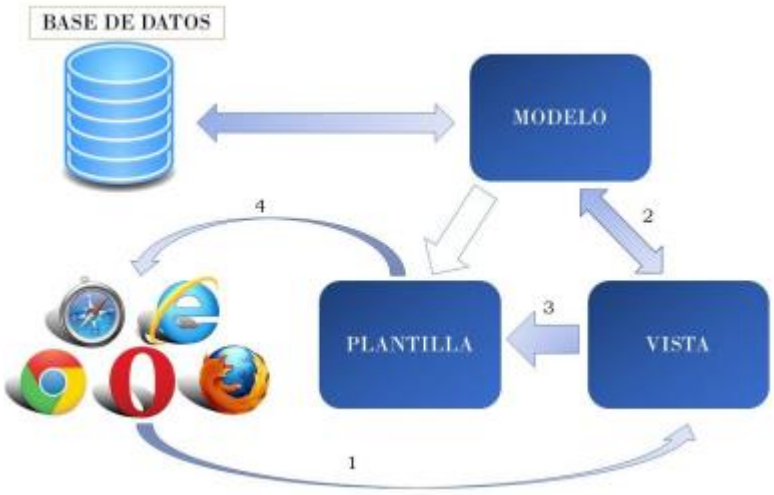
Figura 2 ***Modelo vista controlador***



MVC de arquitectura Tipo 1 (Walls, 2005)

En Django, se conoce al Controlador como “vista” y a la vista como “plantilla”. Ya que en su implementación la vista describe el dato que se muestra al usuario de la aplicación. Por lo que se centran más en qué dato ve y no cómo lo ve. Por lo tanto, lo que decide entonces qué dato ve el usuario es la vista, y cómo lo ve se lo delega a la plantilla. La vista, es la capa de la lógica de negocios, donde se encuentra la inteligencia que accede al modelo y la delega a la plantilla adecuada. Por último, el modelo, no cambia con respecto a la explicación del patrón Modelo – Vista – Controlador convencional, sigue siendo la capa de acceso a la base de datos, que contiene, cómo acceder a ellos, cuál es su comportamiento, como validarlos y las relaciones entre los datos. Se pasa a mostrar un esquema de la colaboración existente entre los distintos componentes del patrón:

Figura 3 Diagrama colaboración capa MTV



Fuente: Trabajo de grado titulado “Estudio del Framework de Desarrollo Web Django”

- El cliente desde su navegador realiza una solicitud de un servicio web de la aplicación, por lo que se pone en contacto con la capa de la vista.
- La capa de la vista necesita tener contacto directo con la del modelo ya que es de ella de donde cogerá los datos que solicita el usuario. Por lo que entra en acción la

capa del modelo. Ésta última capa, gracias a las consultas a la base de datos devuelve los objetos a la capa de la vista, que ella ya se encargará de filtrar y devolver explícitamente los que el usuario ha solicitado en la petición web.

- El siguiente paso es que la capa de la vista envíe a la capa de la plantilla los datos requeridos por el usuario, tras previo procesamiento de los mismos. En este paso, ha actuado indirectamente el modelo, ya que es quien proporcionó los datos a la vista.
- En último lugar, la capa de la plantilla es la encargada de cómo se deben mostrar los datos al usuario. Por lo que, con los datos recibidos de la vista, los incluye en su respectiva plantilla para que los navegadores puedan procesarlo y el usuario visualice correctamente la información.

Con estos cuatro pasos, se ha pretendido mostrar al lector una visión global de como intervienen las distintas capas de este patrón tras una solicitud del usuario de la aplicación hasta que le devuelve los objetos requeridos.

Estas consideraciones anteriores marcan una diferencia notable entre Django y otros frameworks web como puede ser Ruby on Rails, ya que como se ha comentado ofrece una modificación particular sobre el patrón Modelo – Vista – Controlador.

Mapeador Objeto-Relacional (ORM), el ORM en Django va a ser el que nos permita interactuar constantemente con la base de datos. Django se va a encargar de traducir nuestras operaciones sobre los objetos, que se ejecutaran sobre las tablas de la base de datos. Esto permitirá al programador más flexibilidad ya que no tendrá que centrarse en estas acciones, sino que Django detectara que el programador está modificando un dato y se lo hará saber a la base de datos (Alcántara, 2021).

Panel de administración de fácil manejo, en Django una de las principales ventajas es la posibilidad de utilizar su panel de administrador para la página web en desarrollo. Mediante el panel de administración, el administrador de la página tendrá fácil acceso a toda la información circulando por su página o páginas vinculadas a este panel.

Desde este punto podrá además controlar de una forma muy sencilla toda la información que contienen las bases de datos de la página, permitiendo la posibilidad de actualizar, borrar o crear cualquier objeto de contenido en la página. Proporciona también una forma sencilla de controlar a los usuarios de la página web, pudiendo modificar desde aquí los permisos de acceso de cada uno de ellos (django, 2005).

Figura 4 ***Panel de Administración***

Fuente: Elaboración propia.

Bibliotecas incorporadas con funciones de gran utilidad, Django está dotado de una gran librería (escrita en Python) con funcionalidades dirigidas para ayudar al programador a ahorrar tiempo en ejecutar ciertas tareas. Entre la gran variedad de librerías que existen podemos destacar funciones como el envío de correos, leer datos de páginas web o también trabajar con archivos comprimidos (Pythontutorial, 2023).

5.2.3.2 Requisitos de Django

Para poder realizar una aplicación web, usando Django, es necesario conocer los siguientes puntos:

- Como Django es un Framework para Python, se debe de tener instalado Python en el computador.
- En cuanto a la base de datos subyacente, Django viene con SQLite3 por defecto, pero es posible disponer de otro motor de base de datos como: MySQL, PostgreSQL y Oracle.
- En cuanto al servidor Web donde correrá Django, se pueden diferenciar dos partes: o Para la parte de desarrollo de la aplicación, Django aporta un servidor propio. o Para la parte de producción, se puede montar sobre otros servidores como Apache, entre otros.

5.3 Marco Legal

5.3.1 Normas Internacionales

En el contexto del desarrollo de software, se reconoce la importancia fundamental de garantizar la calidad en aspectos como funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad y satisfacción. Este marco legal se centra en la adopción de estándares internacionales, específicamente la norma ISO-9126, que proporciona una guía exhaustiva para evaluar la calidad del software (Robayo, 2023).

De acuerdo con las observaciones de Domínguez (2016), se destaca la madurez y flexibilidad de la norma ISO-9126, posicionándola como la elección preferida en comparación con otras metodologías. Su aplicación se organizará en cuatro etapas: identificación de acciones y usuarios, definición de atributos y métricas de calidad, establecimiento de escenarios de calidad, diseño y ejecución de la evolución, así como la documentación y análisis detallado de los resultados obtenidos.

En el panorama digital actual, la seguridad de la información emerge como un componente crucial. La norma ISO 27001 se erige como una guía esencial que abarca diversas actividades y acciones para llevar a cabo análisis de riesgos. Esto incluye la delimitación del alcance del proyecto, la identificación de activos (infraestructura, software, talento humano, etc.), la identificación de amenazas y vulnerabilidades, la evaluación del riesgo y la documentación minuciosa de los análisis y resultados (Icariatechnology, 2023).

El cumplimiento de este marco legal es obligatorio para todas las entidades y profesionales vinculados al desarrollo de software. Auditorías periódicas serán realizadas para

verificar el cumplimiento de las normativas establecidas. La falta de conformidad podría acarrear sanciones y medidas correctivas.

Se insta a los actores del desarrollo de software a mantener una perspectiva de mejora continua. La adaptación y evolución constante de las prácticas y estándares, en concordancia con los avances tecnológicos y las necesidades cambiantes del mercado, son fundamentales para garantizar la vigencia y relevancia de este marco legal.

5.3.2 Normas Nacionales

En Colombia, la Ley de Protección de Datos Personales (Ley Estatutaria 1581, 2012), establece en su artículo 2° principios y disposiciones aplicables a quienes gestionen bases de datos en las que se registren datos personales y sensibles, bien sea que se ubiquen en el territorio o fuera de este y que le sea extensiva esta norma; entre estos principios se encuentran el de legalidad en materia de tratamiento de datos; veracidad o calidad de la información, libertad, acceso y circulación restringida y el de seguridad, por lo que quienes realicen el tratamiento de los datos deben procurar las medidas técnicas, administrativas y humanas necesarias para garantizar la integridad de los datos.

5.4 Marco conceptual

Aplicación web, la aplicación web se define como un conjunto de programas y recursos diseñados para operar a través de navegadores web. Su arquitectura se centra en la accesibilidad y la interactividad, proporcionando interfaces intuitivas que permiten a los usuarios interactuar con la aplicación de manera eficiente (Amazon, 2023).

Backend, también conocido como el lado del servidor, representa la infraestructura y la lógica de negocio que respalda la aplicación web. Django, un marco de desarrollo web basado en Python, destaca por su versatilidad y eficiencia en la construcción de la capa backend. Este se encarga de gestionar la interacción con la base de datos, la autenticación y la lógica subyacente (Jesus, 2023)

Frontend, el frontend, o lado del cliente, se enfoca en la presentación de la información y la interacción directa con los usuarios. Tecnologías como HTML, CSS y JavaScript son esenciales para la construcción de interfaces de usuario atractivas y funcionales. La colaboración entre frontend y backend garantiza una experiencia de usuario coherente y fluida (Ferry, 2014).

Desarrollo ágil de software, el desarrollo ágil de software se erige como un enfoque colaborativo y adaptable que busca maximizar la satisfacción del cliente a través de entregas incrementales y frecuentes. SCRUM y eXtreme Programming (XP) son metodologías ágiles ampliamente adoptadas. SCRUM se centra en la gestión eficiente del proyecto mediante Sprint y roles definidos, mientras que XP promueve prácticas como la programación en parejas y la retroalimentación constante (desarrollodesoftware, 2020).

Base de Datos, la gestión de datos es esencial en cualquier aplicación web. La elección y diseño de la base de datos impactan directamente en el rendimiento y la escalabilidad del

sistema. Tecnologías como MySQL, PostgreSQL o MongoDB se utilizan para almacenar, organizar y recuperar información de manera eficiente y estructurada (Oracle, 2024).

Python, es un lenguaje de programación de código abierto y multiplataforma. Con el pasar del tiempo Python ha ganado popularidad debido a sus facilidades tanto de uso como de aprendizaje, así mismo, por la amplia variedad de posibilidades que ofrece al permitir trabajar en múltiples ramas del conocimiento como la inteligencia artificial, machine learning, deep learning entre muchos otras.

Django, Es un framework de código abierto y escrito en Python que está basado en el patrón de diseño MTV (Model Template View) para el desarrollo de páginas webs, aplicaciones y APIs (con Django REST Framework).

Django REST Framework, Django Rest Framework es una aplicación Django que permite la reutilización de una gran cantidad de código, fue utilizado en este proyecto dado que permite construir proyectos software bajo la arquitectura REST, Por otro lado, Django REST Framework trae incluido una interfaz administrativa que permite realizar pruebas y peticiones sobre el protocolo HTTP.

PostgreSQL, Es un sistema de gestión de bases de datos relacional orientado a objetos y es de código abierto.

Las bases de datos PostgreSQL se han convertido en unas de las más utilizadas por los usuarios gracias a que ofrecen una serie de ventajas como son: la disponibilidad multiplataforma, la fácil configuración, sistemas de fiabilidad, control de concurrencias entre muchas otras.

GitHub, Es una plataforma que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando un sistema de control de

versiones, llamado Git, su principal objetivo en este proyecto es llevar registro de los cambios realizados durante el desarrollo del aplicativo Web.

5.4.1 Metodología

En la elección de la metodología de investigación para este proyecto, se ha optado principalmente un enfoque aplicado, fundamentado en la intención de "aplicar" los conocimientos teóricos adquiridos a través de la práctica y la experiencia concreta durante el desarrollo de la aplicación web utilizando Django. Este enfoque aplicado surge con el propósito de abordar problemas específicos, principalmente la comprensión detallada de las características y requisitos del aplicativo.

La investigación aplicada se orienta hacia la resolución efectiva de estos problemas, para enriquecer aún más la comprensión de la aplicación y percepciones no cuantificables únicamente mediante datos numéricos, se incorporará un enfoque cualitativo en la metodología.

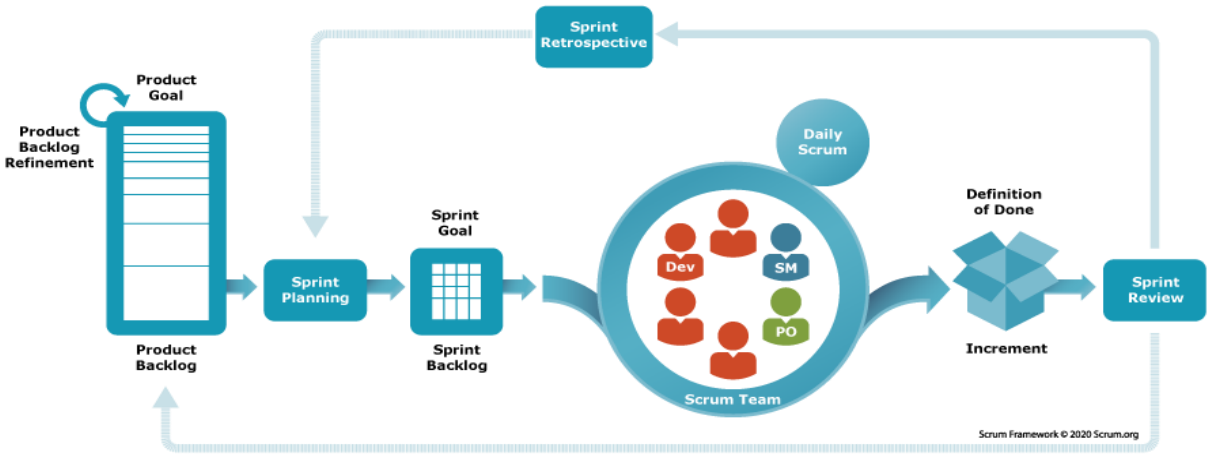
La inclusión de un enfoque cualitativo se justifica por la necesidad de explorar a fondo las experiencias, percepciones y necesidades de los usuarios finales, así como obtener una mejor comprensión de las dinámicas y desafíos específicos que pueden surgir durante la implementación y el uso del aplicativo web. Este enfoque permitirá recopilar información detallada sobre la usabilidad, la experiencia del usuario y cualquier factor subjetivo que pueda influir en el éxito del proyecto.

Aspectos clave como la selección de frameworks, gestores de base de datos y otras tecnologías se abordarán de manera fundamentada tanto en la realidad concreta del proyecto como en la comprensión cualitativa de las necesidades y preferencias de los usuarios.

5.4.1.1 Scrum

Para el desarrollo del proyecto, como se muestra en la figura 1, se usó el marco de trabajo SCRUM, es un marco ligero que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptables para problemas complejos. (Schwaber, 2020). Scrum se basa en el empirismo y el pensamiento Lean. El empirismo afirma que el conocimiento proviene de la experiencia y la toma de decisiones basadas en lo que se observa. El pensamiento Lean reduce los desperdicios y se centra en lo esencial (Schwaber, Scrum.org, 2024).

Figura 5 ***Marco de trabajo SCRUM***



Tomada del sitio Scrum.org (Schwaber, Scrum.org, 2024)

Teniendo en cuenta lo anterior, el tamaño del proyecto y la finalidad del aplicativo web en el presente trabajo se abordará la metodología de desarrollo ágil XP, por un lado, porque permite la integración de las partes del aplicativo (Frontend y Backend) tras haber sido desarrollado por un solo estudiante y ponerlo a prueba, también porque es la empresa quien proporcionarán información relevante sobre las funcionalidades requeridas por la aplicación, esto

comprenderían las historias de usuario que se desglosa en las tareas a realizar por parte del desarrollador o grupo de desarrolladores.

Teniendo en cuenta esto se determinan las fases para alcanzar los objetivos del proyecto:

Fase de Planificación, en la cual se determina un cronograma, equipo y forma de trabajo, elegir las herramientas y tecnologías que apoyarán el trabajo con Django para el desarrollo del Aplicativo Web y conocer los requerimientos del mismo.

Fase de Gestión, definir unas fechas de revisión y socialización de avances con el cliente a fin de mantener una comunicación y participación constante, gestionar la carga de trabajo, revisar tanto los procesos realizados en la Programación Extrema como los avances efectuados en el aplicativo, por ejemplo, estudiar las historias de usuario y definir sobre cuál de ellas se trabajará en un tiempo determinado, etc. (Raebum, 2022)

Fase de Diseño, se definirá en esta fase la herramienta de simulación del de los módulos para la aplicación, también el modelo de la base de datos para su correcto funcionamiento.

Fase de Codificación, se inicia el desarrollo de los módulos y funciones del proyecto, aplicando todos los requerimientos solicitados por el cliente, trabajando con Django.

Fase de Pruebas, se realiza la validación de los resultados obtenidos en las pruebas, se aplican las mejoras, se analiza la información final y se describen resultados claros respecto a funcionalidad, mantenimiento, usabilidad, etc.

6. Resultados

6.1 Fase de Planificación

Como marco de trabajo se empleó Scrum, que con la filosofía de agilismo motivo a realizar reuniones semanales, en las que se planteaban las diferentes funcionalidades y actividades. Cada iteración se considera un incremento que dará como resultado un producto de software completo y que cumpla con los requerimientos planteados.

Para cualquier desarrollo de software se hace evidente la necesidad de tener una planificación clara del producto que se quiere obtener, sin embargo, al hacer uso de la metodología Scrum dicha planificación debe ser flexible y adaptable, es decir, durante el desarrollo de la práctica se tomaron en cuenta las etapas planteadas por la metodología, pero se adaptaron a las necesidades y requerimientos del proyecto.

6.1.1 Análisis de requerimientos

El análisis de requisitos es el primer paso para un desarrollo de software de alta calidad, dado que, se establecen las bases del software que se comenzará a desarrollar. Se investiga el problema al que se le quiere dar solución y se plantean las pautas de manera clara del sistema que cumplirá con dichos requerimientos.

Estos son los requerimientos dados por el Jefe de TI de la empresa Laboratorio María Salomé y en los cuales se rigen para el desarrollo del aplicativo cumpliendo con todos y cada uno de ellos.

Tabla 2 Requisitos funcionales para la aplicación.

Requisitos Funcionales
La aplicación debe permitir manejar el inventario por usuario y tipos de Hardware.
La aplicación debe permitir registrar los eventos relacionados a cada activo.
La aplicación debe permitir generar actas de entregas de los activos asignados a los usuarios.
La aplicación debe permitir almacenar actas digitales.
La aplicación deber permitir registrar garantías y fechas de vencimiento.
La aplicación debe mostrar un aviso de las licencias vencidas.
La aplicación debe permitir asignar licencias a los activos que las requieran.

Fuente: elaboración propia

Tabla 3 Requisitos no funcionales para la aplicación.

Requisitos no Funcionales
La aplicación debe ser intuitiva y fácil de usar.
La aplicación debe ser segura.
La aplicación debe ser escalable y capaz de sufrir modificaciones.

Fuente: elaboración propia

6.1.2 Recursos de Software

Los recursos de software a utilizar para el desarrollo de la aplicación se basarán en el uso del código abierto, utilizando herramientas de software respaldadas por comunidades de desarrollo que permiten realizar propuestas empresariales a bajo costo. Entre ellos se considera al lenguaje Python, ya que es un código abierto y es apto para todas las plataformas.

La siguiente tabla describe otras herramientas de código abierto a utilizar para el desarrollo de la propuesta:

Tabla 4 ***Recursos para el desarrollo del software***

Tipo de recursos	Nombre
Lenguaje de programación	Python
Framework	Django
Framework de diseño de las vistas	Bootstrap
Motor de la base de datos	PostgreSQL
Herramienta de versionamiento	Git

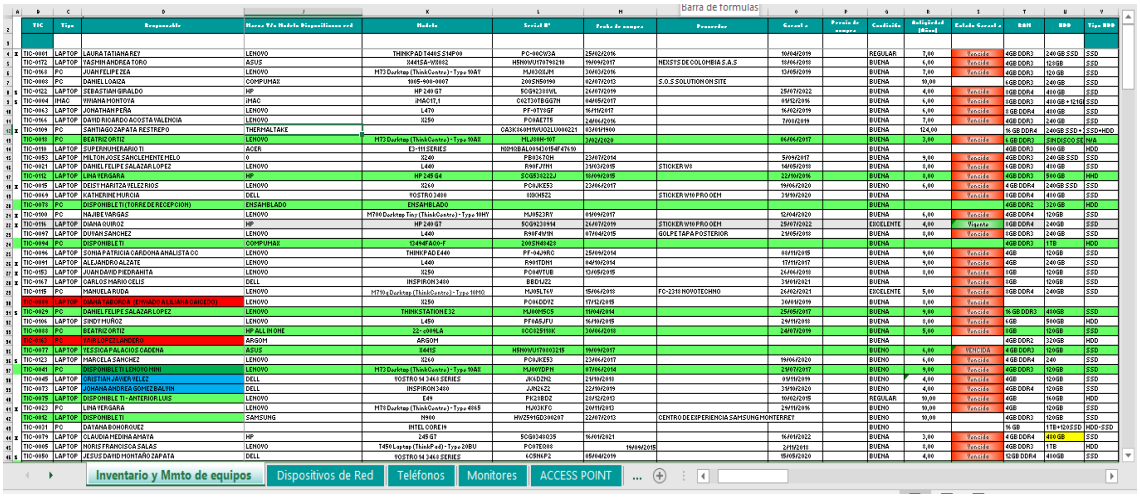
Fuente: elaboración propia

6.1.3 Recopilación de datos

Inicialmente para la recopilación de información, por gestiones del tutor de la práctica se pudo acceder al archivo de la base de datos en formato EXCEL.

Dicho archivo sirvió como fundamento para el planteamiento de la Base de Datos, posteriormente se planteó un cuestionario para alinear el desarrollo con las necesidades de la empresa.

Figura 6 **Archivo gestión de activos**



Fuente: elaboración propia

Cuestionario

El objetivo de la entrevista es analizar a fondo las necesidades, procesos y procedimientos relacionados con la administración de inventario de hardware y licencias en el Laboratorio María Salomé S.A.S. La información recopilada será fundamental para el desarrollo de la aplicación web. La aplicación se centrará en la gestión y seguimiento de activos de hardware y licencias de software en el laboratorio.

¿Actualmente, el Laboratorio María Salomé S.A.S cuenta con un sistema o herramienta para gestionar y rastrear su inventario de hardware y licencias de software?

a. En caso afirmativo, ¿qué sistema o herramienta utilizan actualmente para esta gestión?

¿Cuáles son los procedimientos utilizados para añadir nuevos activos de hardware al inventario? ¿Y para agregar nuevas licencias de software?

¿Cómo se asigna un estado a los activos de hardware en su laboratorio? (Ejemplo de estados: disponible, en uso, en reparación, obsoleto)

¿Cómo se asigna un estado a las licencias de software?

¿Tienen un proceso de seguimiento y registro de detalles de mantenimiento para los activos de hardware? Si es así, ¿cómo se lleva a cabo?

¿Existe la posibilidad de agregar comentarios o notas a los activos de hardware y licencias de software? ¿Cómo se gestiona esta información adicional?

¿El laboratorio recibe notificaciones o alertas en caso de vencimiento de licencias de software?

6.2 Fase de Gestión

6.2.1 Pruebas de Verificación

Estas pruebas son de una alta prioridad para cada etapa del proceso que se quiere desarrollar, permitiendo la validación y correcciones pertinentes en cada uno de los módulos establecidos para el sistema.

Scrum Master: Ing. Wbeimar Cardona

Development Team: Jesus David Montaño

Para hacer más fácil el desarrollo, las historias de usuario se encuentran divididas en etapas:

Login/logout

Roles/permisos

Actas

CRUD inventario

Registro/control.

A continuación, se muestran las historias de usuario

6.2.1.1 Historias de Usuarios

Tabla 5 *Historia de usuario No.1 Login/logout*

Historia de Usuario			
Numero:	1	**Usuario:**	Administrador
Nombre de Historia:	Inicio Sesión		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Login/logout
Descripción:	Como usuario administrador, podrá ingresar al sistema con un nombre de usuario y clave única.		
Observaciones:	Como administrador quiero poder cambiar mi contraseña para tener, mayor seguridad de los datos registrados.		

Fuente: elaboración propia.

Tabla 6 *Historia de usuario No.2 Login/logout*

Historia de Usuario			
Numero:	2	**Usuario:**	Administrador
Nombre de Historia:	Creación de usuarios		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Login/logout
Descripción:	Como usuario administrador, podrá crear usuarios en el sistema con un nombre de usuario y clave única.		
Observaciones:	como administrador quiero poder crear usuarios para tener, mayor seguridad de los datos registrados.		

Fuente: elaboración propia.

Tabla 7 *Historia de usuario No.3 Roles/permisos*

Historia de Usuario			
Numero:	3	**Usuario:**	Administrador
Nombre de Historia:	Administración de permisos		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Roles/permisos
Descripción:	Como usuario administrador, podrá asignar permisos especiales a usuarios específicos o grupos de usuarios.		
Observaciones:	como administrador quiero poder tener el control de los permisos de cada usuario o grupos de usuarios para tener mayor seguridad y control de la aplicación.		

Fuente: elaboración propia.

Tabla 8 *Historia de usuario No.4 CRUD inventario*

Historia de Usuario			
Numero:	4	**Usuario:**	Administrador
Nombre de Historia:	Agregar al Inventario		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	CRUD inventario
Descripción:	Como administrador, quiero poder agregar nuevos equipos al inventario.		
Observaciones:	como administrador quiero poder mantener un inventario organizado de los activos.		

Fuente: elaboración propia.

Tabla 9 *Historia de usuario No.5 CRUD inventario*

Historia de Usuario			
Numero:	5	**Usuario:**	Administrador
Nombre de Historia:	Editar activos en Inventario		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	CRUD inventario
Descripción:	Como administrador, quiero poder editar los equipos al inventario.		
Observaciones:	Como administrador quiero poder mantener un inventario correcto de los activos.		

Fuente: elaboración propia.

Tabla 10 *Historia de usuario No.6 CRUD inventario*

Historia de Usuario			
Numero:	6	**Usuario:**	Administrador
Nombre de Historia:	Eliminar activos al Inventario		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	CRUD inventario
Descripción:	Como administrador, quiero poder eliminar activos del inventario.		
Observaciones:	como administrador quiero poder mantener un inventario actualizado de los activos.		

Fuente: elaboración propia.

Tabla 11 *Historia de usuario No.7 Registro/control*

Historia de usuario			
Numero:	7	**Usuario:**	Administrador
Nombre de Historia:	Asignación de activos		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	Como administrador, quiero poder asignar activos a usuarios específicos.		
Observaciones:	como administrador quiero poder asignar y liberar los activos en inventario a los usuarios.		

Fuente: elaboración propia.

Tabla 12 *Historia de usuario No. 8 Registro/control*

Historia de usuario			
Numero:	8	**Usuario:**	Administrador
Nombre de Historia:	Registro de Eventos.		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	Media
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	Como administrador quiero registrar eventos relacionados con los equipos.		
Observaciones:	El administrador puede registrar eventos como: reparaciones o renovaciones de licencias, para llevar un registro histórico de las actividades y cambios en los equipos.		

Fuente: elaboración propia.

Tabla 13 *Historia de usuario No. 9 Actas*

Historia de usuario			
Numero:	9	**Usuario:**	Administrador
Nombre de Historia:	Generar actas de entregas		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Actas
Descripción:	Generan actas de entregas de activos, asignados a los usuarios.		
Observaciones:	Podrán documentar de manera formal la asignación de equipos a usuarios y tener un registro de aceptación.		

Fuente: elaboración propia.

Tabla 14 *Historia de usuario No. 10 Actas*

Historia de usuario			
Numero:	10	**Usuario:**	Administrador
Nombre de Historia:	Almacenamiento de Actas Digitales		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	1	**Etapa asignada:**	Actas
Descripción:	Adjuntar y almacenar actas digitales firmadas por los usuarios		
Observaciones:	Podrá tener un registro electrónico de las actas de entrega		

Fuente: elaboración propia.

Tabla 15 *Historia de usuario No. 11 Actas*

Historia de usuario			
Numero:	11	**Usuario:**	Administrador
Nombre de Historia:	Eliminar Actas Digitales		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	1	**Etapa asignada:**	Actas
Descripción:	Como administrador podrá eliminar actas digitales almacenadas.		
Observaciones:	Podrá dar de baja las actas que ya no sean necesarias de almacenar.		

Fuente: elaboración propia.

Tabla 16 *Historia de usuario No. 12 Registro/control*

Historia de usuario			
Numero:	12	**Usuario:**	Administrador
Nombre de Historia:	Registro y Control de Garantías		
Prioridad en Negocio:	Media	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	Como Administrador, quiero registrar y consultar las garantías de los equipos		
Observaciones:	Se garantizara un rápido acceso a la información de garantía y seguimiento de los vencimientos.		

Fuente: elaboración propia.

Tabla 17 *Historia de usuario No. 13 Registro/control*

Historia de usuario			
Numero:	13	**Usuario:**	Administrador
Nombre de Historia:	Registro de Licencias		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	El administrador podrá registrar y gestionar licencias de software, asociadas a los computadores.		
Observaciones:	Se llevara un control que avisara de licencias en uso y legalmente funcionales.		

Fuente: elaboración propia.

Tabla 18 *Historia de usuario No. 14 Registro/control*

Historia de usuario			
Numero:	14	**Usuario:**	Administrador
Nombre de Historia:	Filtrado de Licencias		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	Como administrador podrá consultar las licencias mediante un filtrado intuitivo.		
Observaciones:	Como administrador quiero poder consultar las licencias por tipos de manera fácil y rápida.		

Fuente: elaboración propia.

Tabla 19 *Historia de usuario No. 15 Registro/control*

Historia de usuario			
Numero:	15	**Usuario:**	Usuario
Nombre de Historia:	Filtrado de usuarios		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	Como administrador podrá consultar los usuarios mediante un filtrado intuitivo.		
Observaciones:	Como administrador quiero poder consultar los usuarios por área de manera fácil y rápida.		

Fuente: elaboración propia.

Tabla 20 *Historia de usuario No. 16 Registro/control*

Historia de usuario			
Numero:	16	**Usuario:**	Usuario
Nombre de Historia:	Filtrado de activos		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	Como administrador podrá consultar los activos mediante un filtrado intuitivo.		
Observaciones:	Como administrador quiero poder consultar los activos por tipos de manera fácil y rápida.		
Validación:	El administrador puede seleccionar el tipo de activo en el filtro.		

Fuente: elaboración propia.

Tabla 21 *Historia de usuario No. 17 Registro/control*

Historia de usuario			
Numero:	17	**Usuario:**	Administrador
Nombre de Historia:	Historial de acciones		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	Como administrador podrá ver un historial de las acciones recientes realizadas en el aplicativo		
Observaciones:	El administrador podrá tener acceso a un historial de acciones realizadas en el aplicativo		

Fuente: elaboración propia.

Tabla 22 *Historia de usuario No. 18 Registro/control*

Historia de usuario			
Numero:	18	**Usuario:**	Administrador
Nombre de Historia:	Seguridad		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	Que la seguridad para los datos sea segura y confiable		
Observaciones:	Seguridad a la exposición no autorizada de información.		

Fuente: elaboración propia.

Tabla 23 *Historia de usuario No. 19 Registro/control*

Historia de usuario			
Numero:	19	**Usuario:**	Administrador
Nombre de Historia:	Escalabilidad		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	2	**Etapa asignada:**	Registro/control
Descripción:	la aplicación deberá ser escalable y capaz de manejar un crecimiento significativo en la cantidad de datos .		
Observaciones:	Se asegurara que la aplicación pueda crecer con las necesidades de la organización sin problemas de rendimiento		

Fuente: elaboración propia.

Tabla 24 *Historia de usuario No. 20 Pruebas del sistema*

Historia de usuario			
Numero:	20	**Usuario:**	Administrador, Desarrollador
Nombre de Historia:	Pruebas del sistema		
Prioridad en Negocio:	Alta	**Riesgo de Desarrollo:**	**Alta**
Puntos Estimados:	10	**Etapa asignada:**	Pruebas del sistema
Descripción:	Evaluar el funcionamiento de el Aplicativo Web a través de pruebas de calidad.		

Fuente: elaboración propia.

Tabla 21 *Historias de usuario ordenadas por prioridad*

Etapas	Historia de usuario	Prioridad	Puntos estimados
Login/logout	Inicio de Sesión	Alta	5
Login/logout	Creación de usuarios	Alta	2
Roles/permisos	Administración de Permisos	Alta	5
CRUD inventario	Agregar al inventario	Alta	3
CRUD inventario	Editar activos	Alta	2
CRUD inventario	Eliminar activos	Alta	2
Registro/control	Asignación de activos	Alta	2
Registro/control	Registro de eventos	Media	2
Actas	Generación de actas	Alta	20
Actas	Almacenamiento de actas	Alta	1
Actas	Eliminar Actas Digitales	Alta	1
Registro/control	Registro y control de garantías	Alta	2
Registro/control	Registro de licencias	Alta	2
Registro/control	Filtrado de licencias	Alta	5
Registro/control	Filtrado de usuarios	Alta	2
Registro/control	Filtrado de activos	Alta	2
Registro/control	Historial de acciones	Alta	2
Registro/control	Seguridad	Alta	10
Registro/control	Escalabilidad	Alta	15

Fuente: elaboración propia.

6.2.2 Estimación del Backlog

Para llegar a una estimación del tiempo de desarrollo del sistema se tuvo en consideración el tiempo límite del proyecto (16 semanas), la disponibilidad de tiempo del desarrollador y así se obtuvo la planificación de cada Sprint. La siguiente tabla indica los resultados obtenidos:

Tabla 25 *Criterios de estimación*

Tamaño de Sprint	Horas por día	Horas por semana	Total de horas por Sprint
2 semanas(10 dias)	3	15	30

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Para la estimación de cada Sprint se usó la técnica de Estimación Aproximada que consiste en debatir la cantidad de historias de usuario que pueden realizarse en cada sprint. Se obtuvieron las siguientes estimaciones:

Tabla 26 *Estimación del Sprint N1.*

Etapa	Historia De Usuario	Prioridad	Puntos Estimados	Tiempo Estimado
Login/Logout	Inicio de Sesion	Alta	5	4 Dias
Login/Logout	Creación de usuario	Alta	2	2 dias
Roles/permisos	Administración de permisos	Alta	5	4 dias
Total de dias del Sprint				10 dias

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Tabla 27 *Estimación del Sprint N2.*

Etapa	Historia De Usuario	Prioridad	Puntos Estimados	Tiempo Estimado
CRUD/inventario	Agregar al inventario	Alta	3	6 Dias
CRUD/inventario	Editar activos	Alta	2	2 dias
CRUD/inventario	Eliminar activos	Alta	2	2 dias
Total de dias del Sprint				10 dias

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia.

Tabla 28 *Estimación del Sprint N3.*

Etapa	Historia De Usuario	Prioridad	Puntos Estimados	Tiempo Estimado
Registro/control	Asignación de activos	Alta	3	2 Dias

Registro/control	Registro de eventos	Media	2	2 dias
Actas	Generación de actas	Alta	5	6 dias
	Total de dias del Sprint			10 dias

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Tabla 29 *Estimación del Sprint N4.*

Etapa	**Historia De Usuario**	**Prioridad**	**Puntos Estimados**	**Tiempo Estimado**
Actas	Almacenamiento de actas	Alta	5	3 Dias
Actas	Eliminar Actas Digitales	Alta	3	2 dias
Registro/control	Registro, control de garantías	Alta	6	5 dias
	Total de dias del Sprint			10 dias

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Tabla 30 *Estimación del Sprint N5.*

Etapa	**Historia De Usuario**	**Prioridad**	**Puntos Estimados**	**Tiempo Estimado**
Registro/control	Registro de licencias	Alta	3	3 Dias
Registro/control	Filtrado de licencias	Alta	2	2 dias
Registro/control	Filtrado de usuarios	Alta	5	5 dias
	Total de dias del Sprint			10 dias

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Tabla 31 *Estimación del Sprint N6.*

Etapa	Historia De Usuario	Prioridad	Puntos Estimados	Tiempo Estimado
Registro/control	Filtrado de activos	Alta	2	2 Dias
Registro/control	Historial de acciones	Alta	5	2 dias
Registro/control	Seguridad	Alta	10	6 dias
	Total de dias del Sprint			10 dias

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

6.2.3 Definición de los Sprint

Para el desarrollo de cada Sprint se han planificado revisiones y entregables para validar los avances correspondientes del sistema. Dentro de la Planificación del Sprint se establece el Objetivo del Sprint (Sprint Goal) y el Sprint Backlog correspondiente, de este último se optó por usar la herramienta de Taskboard.

Tabla 32 *Taskboard al finalizar el Sprint N° 1*

Objetivo:	Desarollar interfaz y permisos de usuarios		
Inicio:	2/10/2023	**Fin**	13/10/2023
Sprint	**Pendiente**	**En curso**	**Finalizado**
1			Inicio de Sesión
1			Creación de usuarios
1			Administración de Permisos
	Agregar al inventario		
	Editar activos		
	Eliminar activos		
	Asignación de activos		
	Registro de eventos		
	Generación de actas		
	Almacenamiento de actas		
	Eliminar Actas Digitales		
	Registro y control de garantías		
	Registro de licencias		
	Filtrado de licencias		
	Filtrado de usuarios		
	Filtrado de activos		

	Historial de acciones		
	Seguridad		

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Tabla 33 *Taskboard al finalizar el Sprint N° 2*

Objetivo:	Desarollar funcionalidades CRUD		
Inicio:	16/10/2023	**Fin**	27/10/2023
Sprint	**Pendiente**	**En curso**	**Finalizado**
1			Inicio de Sesión
1			Creación de usuarios
1			Administración de Permisos
2			Agregar al inventario
2			Editar activos
2			Eliminar activos
	Asignación de activos		
	Registro de eventos		
	Generación de actas		
	Almacenamiento de actas		
	Eliminar Actas Digitales		
	Registro y control de garantías		
	Registro de licencias		
	Filtrado de licencias		
	Filtrado de usuarios		
	Filtrado de activos		
	Historial de acciones		
	Seguridad		

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Tabla 34 *Taskboard al finalizar el Sprint N° 3*

Objetivo:	Desarollar funcionalidades de requerimientos		
Inicio:	30/10/2023	**Fin**	10/11/2023
Sprint	**Pendiente**	**En curso**	**Finalizado**
1			Inicio de Sesión
1			Creación de usuarios
1			Administración de Permisos
2			Agregar al inventario
2			Editar activos
2			Eliminar activos
3			Asignación de activos
3			Registro de eventos
3			Generación de actas
	Almacenamiento de actas		
	Eliminar Actas Digitales		
	Registro y control de garantías		
	Registro de licencias		
	Filtrado de licencias		
	Filtrado de usuarios		
	Filtrado de activos		
	Historial de acciones		
	Seguridad		

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Tabla 35 *Taskboard al finalizar el Sprint N° 4*

Objetivo:	Desarollar CRUD actas		
Inicio:	13/11/2023	**Fin**	24/11/2023
Sprint	**Pendiente**	**En curso**	**Finalizado**
1			Inicio de Sesión
1			Creación de usuarios
1			Administración de Permisos
2			Agregar al inventario
2			Editar activos
2			Eliminar activos
3			Asignación de activos
3			Registro de eventos
3			Generación de actas
4			Almacenamiento de actas
4			Eliminar Actas Digitales
4			Registro y control de garantías
	Registro de licencias		
	Filtrado de licencias		
	Filtrado de usuarios		
	Filtrado de activos		
	Historial de acciones		
	Seguridad		

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Tabla 36 *Taskboard al finalizar el Sprint N° 5*

Objetivo:	Desarollar filtros de vistas		
Inicio:	27/11/2023	**Fin**	8/12/2023
Sprint	**Pendiente**	**En curso**	**Finalizado**
1			Inicio de Sesión
1			Creación de usuarios
1			Administración de Permisos
2			Agregar al inventario
2			Editar activos
2			Eliminar activos
3			Asignación de activos
3			Registro de eventos
3			Generación de actas
4			Almacenamiento de actas
4			Eliminar Actas Digitales
4			Registro y control de garantías
5			Registro de licencias
5			Filtrado de licencias
5			Filtrado de usuarios
	Filtrado de activos		
	Historial de acciones		
	Seguridad		

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

Tabla 37 *Taskboard al finalizar el Sprint N° 6*

Objetivo:	Desarollar apartado de acciones recientes y seguridad de la App		
Inicio:	11/12/2023	**Fin**	22/12/2023
Sprint	**Pendiente**	**En curso**	**Finalizado**
1			Inicio de Sesión
1			Creación de usuarios
1			Administración de Permisos
2			Agregar al inventario
2			Editar activos
2			Eliminar activos
3			Asignación de activos
3			Registro de eventos
3			Generación de actas
4			Almacenamiento de actas
4			Eliminar Actas Digitales
4			Registro y control de garantías
5			Registro de licencias
5			Filtrado de licencias
5			Filtrado de usuarios
6			Filtrado de activos
6			Historial de acciones
6			Seguridad

Nota. En esta tabla se presentan de forma detallada la estimación del Sprint establecidos en base a la aplicación de la metodología Scrum y previa evaluación de las metas planteadas dentro del proyecto. La elaboración es propia

6.3 Fase de diseño

La fase de diseño en el desarrollo de nuestra aplicación web en Django, dedicada a la gestión de inventario de equipos de cómputo, es un paso crucial para transformar los requisitos y funcionalidades en una interfaz intuitiva y fácil de usar. En este proceso, nos enfocamos en diseñar una experiencia de usuario eficiente y atractiva, donde la accesibilidad y la usabilidad se fusionen para ofrecer un entorno de gestión de inventario que no solo cumple con las necesidades técnicas, sino que también se adapta de manera natural a las interacciones diarias de los usuarios.

A través de una cuidadosa atención a los detalles visuales, la disposición de elementos y la navegación fluida, aspiramos a crear una aplicación que optimice la eficiencia operativa y mejore la experiencia general del usuario, elevando así la calidad y el impacto de nuestra solución.

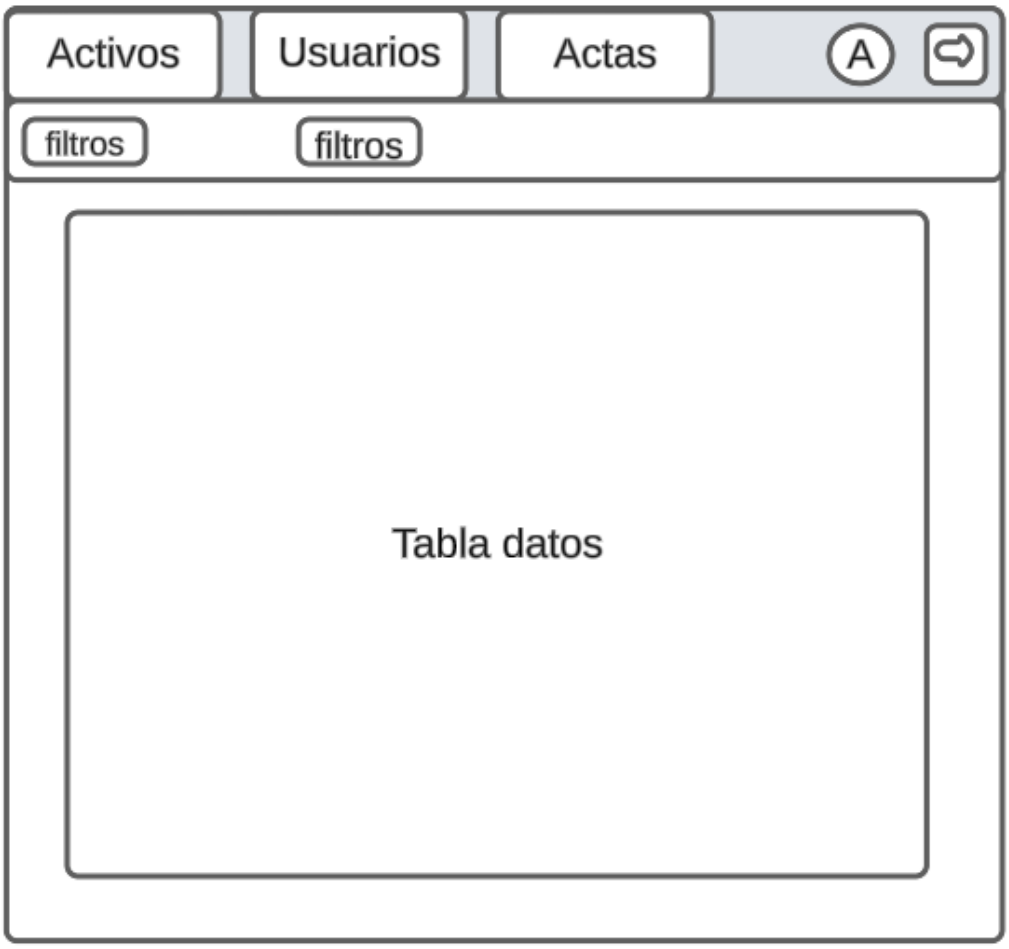
Los siguientes fueron las primeras propuestas de diseño base para la aplicación:

Figura 7 *Primera propuesta de diseño de barra de navegación*

Fuente: elaboración propia.

Para la visualización completa de la aplicación se sugirió una plantilla base donde se llamen las demás opciones mostradas en la barra de navegación.

Figura 8 *Plantilla base de la aplicación*



Fuente: elaboración propia.

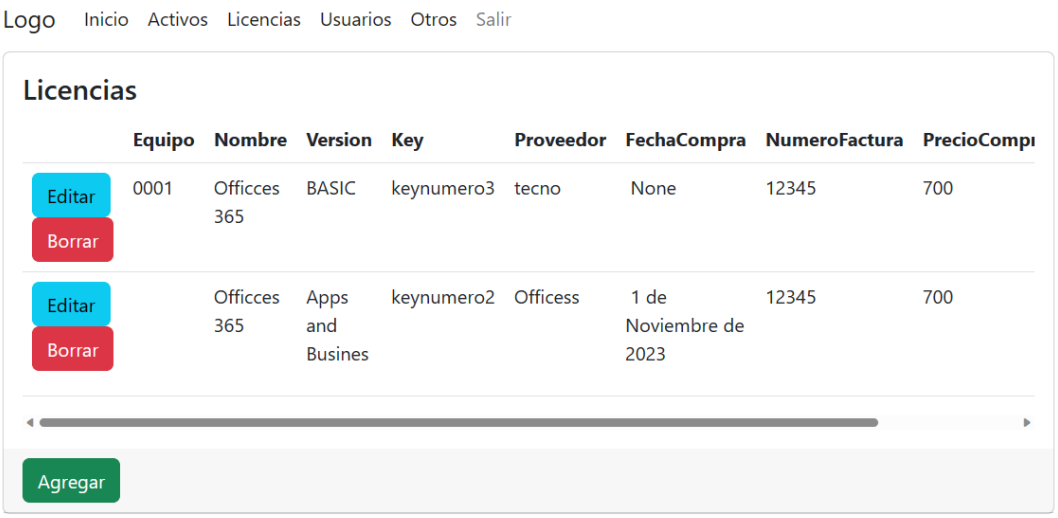
Se pensó en un diseño que los usuarios puedan aprender a utilizar la aplicación de manera rápida y efectiva. A través de una disposición lógica de elementos, una navegación clara y funciones fácilmente accesibles, aspiramos a proporcionar una experiencia de usuario que no solo sea eficiente en términos operativos, sino que también permita a los usuarios familiarizarse y dominar la aplicación de manera rápida y sin complicaciones. Aquí se muestran las modificaciones que sufrió el diseño de la vista principal:

Figura 9 *Esqueleto base para el diseño de los modulos de la aplicacion*

Fuente: elaboración propia.

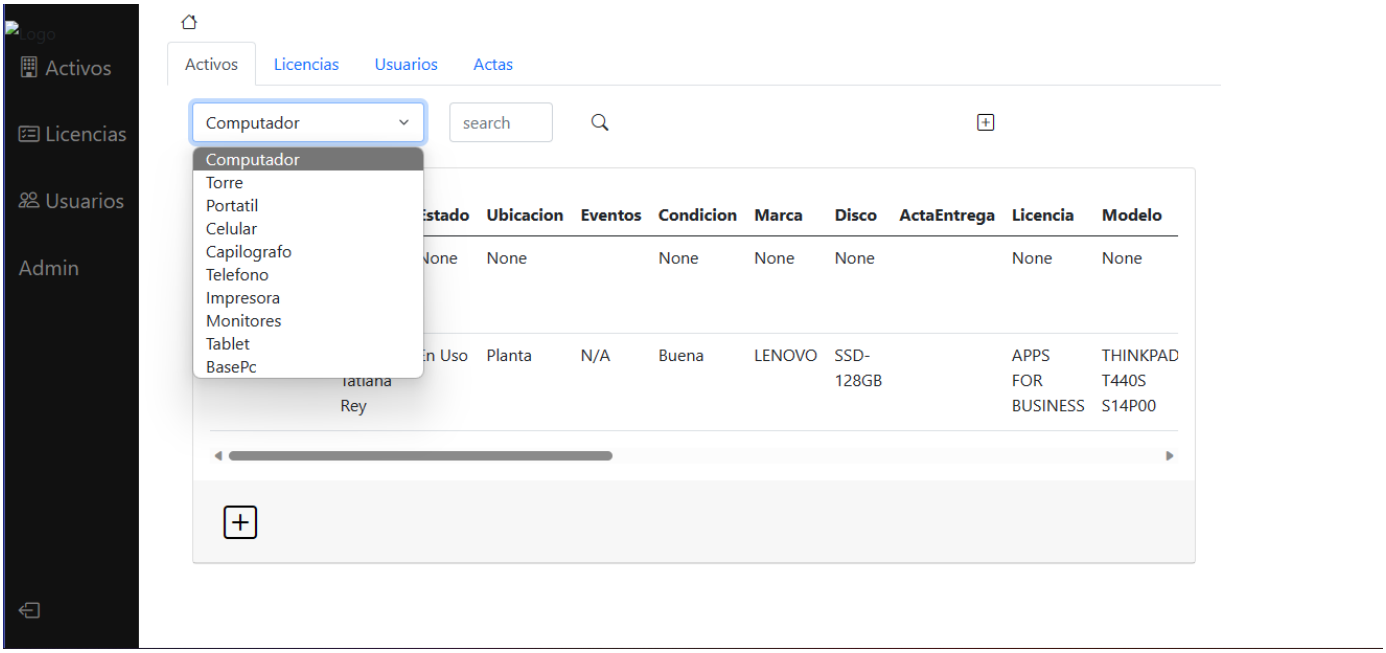
Se integró la siguiente tabla facilitar la adopción y utilización de la aplicación, contribuyendo así a una gestión de inventario más efectiva y a una mayor satisfacción del usuario.

Figura 10 *Modulo principal*



Se llevaron a cabo modificaciones estratégicas en el diseño con la intención expresa de optimizar la experiencia del usuario y aumentar la eficiencia operativa.

Figura 11 *Vista principal de la aplicación.*



Fuente: elaboración propia.

En el punto nueve, titulado “manual de usuario de la aplicación” podrá ver el resultado final del diseño desarrollado para la aplicación.

6.4 Fase De Codificación

Antes de proceder a la explicación de las distintas partes de una aplicación Django, en este capítulo se pretende mostrar al lector los diferentes archivos generados automáticamente al crear una nueva aplicación con este Framework. El objetivo principal es que pueda conocerlos para que cuando sean mencionados en la descripción profunda de cada parte sepa a cuál de ellos se refiere.

6.4.1 Proyecto Django

Dando más claridad a las características lógicas y visuales de las funcionalidades y servicios que se ofrecen en la plataforma y el cómo interactúa con el usuario, se mostrarán a continuación

diferentes imágenes y textos explicativos sobre cada paso a paso que se a realizado en la aplicación Web con Django.

Como nota importante para el desarrollo de esta documentación cabe mostrar las versiones que se usarán para ello.

• Python 3.12

• Django 5.0

1. Crear Una Aplicación Web.

Django trabaja sobre Python, por lo que será necesario tenerlo instalado previamente donde se vaya a generar la aplicación.

Para instalar Django, es necesario ejecutar el siguiente comando:

```
C:\Users\Rober\Documents\GestorCines>pip install django==5.0
Collecting django==5.0
Downloading Django-5.0-py2.py3-none-any.whl (6.9MB)
100% |################################| 6.9MB 130kB/s
```

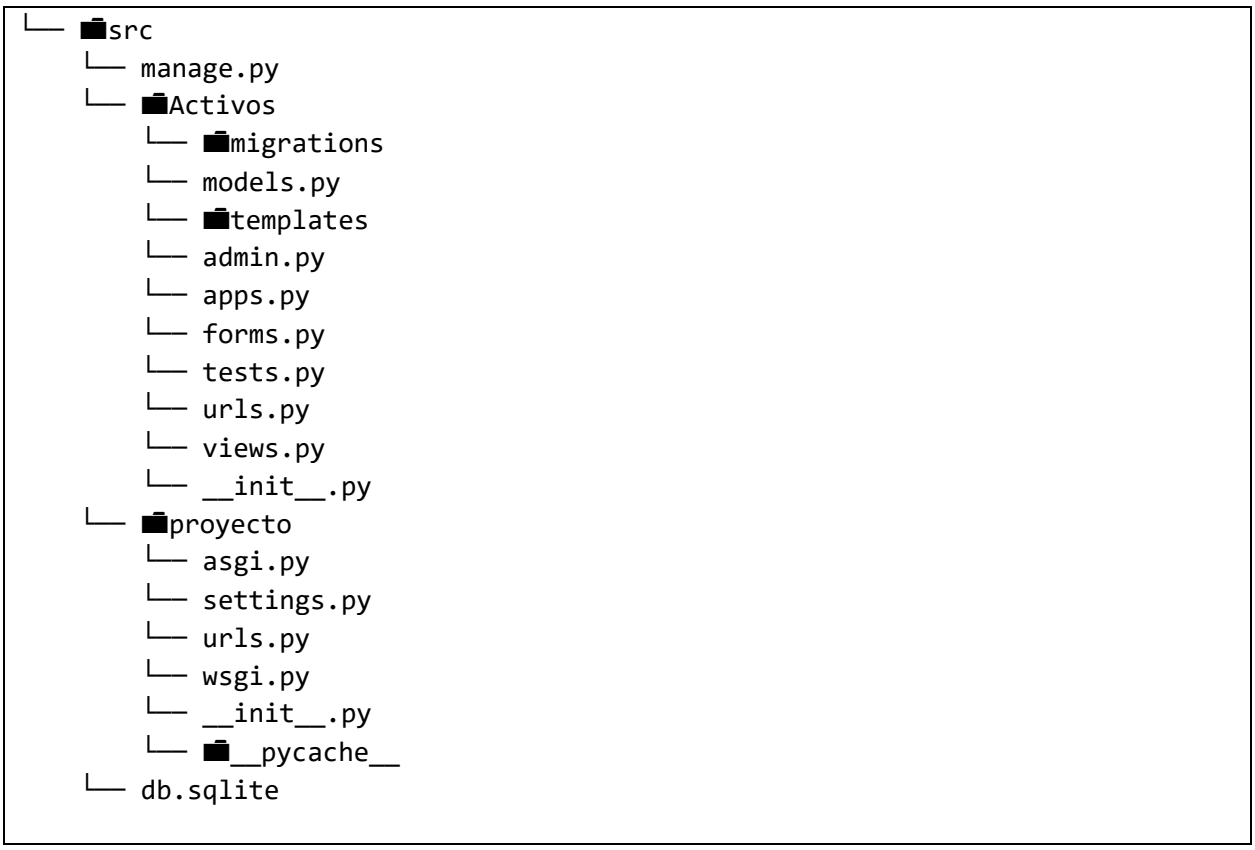
Una vez que se ha instalado correctamente la versión deseada de Django mediante pip, se pasa a crear el nuevo proyecto.

```
PS C:\Users\DELL\OneDrive\Documentos\Python\proyecto\aplicacion>django-admin.py
startproject proyecto
```

Cuando se ha generado el proyecto correctamente está todo preparado para generar la aplicación:

```
PS C:\Users\DELL\OneDrive\Documentos\Python\proyecto\aplicacion>Python manage.py
startapp Activos
```

Con el proyecto y la aplicación en Django creadas se genera una estructura de archivos y directorios los cuales se pretende describir brevemente cada uno de ellos.



Pasamos a comentar cada archivo o carpeta que se aprecia en la imagen anterior:

src/, Esta primera carpeta es el contenedor del proyecto en la cual residirán los archivos y directorios que éste usará. El nombre de este directorio se puede modificar en cualquier momento sin tener que variar ningún parámetro adicional.

manage.py, manage.py es una línea de comandos mediante la cual se puede interactuar con el proyecto de diferentes maneras como pueden ser: crear aplicaciones, levantar el servidor de desarrollo, posibilidad de llamar a ejecutar migraciones, entre otras. Es parecido a django-admin utilizado para crear el proyecto.

src/proyectos, Es el paquete Python del proyecto. En este directorio se permite importar cualquier herramienta al proyecto. El nombre de esta carpeta generada sí que es importante para Django ya que será utilizado para importar librerías en él.

proyecto/settings.py, Este archivo contiene la configuración del proyecto Django. No es necesario cambiar las configuraciones si no se necesitan, ya que Django asigna un valor por defecto para cada una de ellas. Más adelante se verá cómo se modifica este archivo para alterar la funcionalidad del sitio web creado.

proyecto/urls.py, Se puede describir como una tabla de contenidos del sitio Web en Django. En él se encuentran las declaraciones URL para el proyecto.

proyecto/wsgi.py, Es el archivo encargado para la compatibilidad con el servidor Web. Se puede considerar como un punto de entrada para que los servidores Web que son compatibles con WSGI, puedan servir el proyecto.

Activos/migrations, En este directorio es donde se almacenarán todas las migraciones que se realizarán en el proyecto. Quedan registradas con el objetivo de poder analizar los cambios que se han ido produciendo en la parte del modelo de la aplicación.

Activos/admin.py, En el archivo admin.py es donde se definirán los modelos a registrar para el sitio de administración de Django junto con otra información adicional.

Activos/models.py, En este archivo Python será donde quedarán definidos los modelos a utilizar en la aplicación. Se declararán los atributos de cada uno, así como las relaciones entre distintos ellos y otra información adicional.

Activos/tests.py, En test.py se almacenarán las pruebas a realizar para la aplicación donde resida dicho archivo.

db.sqlite, Se corresponde con la base de datos SQLite3 generada automáticamente en los pasos anteriores. Es posible reemplazar esta base de datos por otra como por ejemplo PostgreSQL.

2. Servidores Web En Django

En cuanto a los servidores donde se alojarán las aplicaciones Web, es posible diferenciar dos apartados:

Fase de desarrollo, En la fase de desarrollo, el equipo implementa la aplicación utilizando un servidor escrito en Python con Django para simplificar el proceso y centrarse en el desarrollo. El servidor tiene ventajas como ser ligero y reiniciarse automáticamente al detectar cambios en la aplicación. Para poner en marcha el servidor de desarrollo, solo es necesario ejecutar un comando específico.

```
PS C:\Users\DELL\OneDrive\Documentos\Python\proyecto\aplicacion> python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
febrero 03, 2024 - 15:17:30
Django version 3.2.8, using settings 'aplicacion.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Fase de producción, Para esta última etapa, Django aporta distintas posibilidades sobre los servidores web a utilizar ya que soporta la especificación WSCGI. Se puede lanzar sobre Apache, SCGI, entre otros.

3. Capa Del Modelo En Django

Antes de definir la capa del modelo en Django, es relevante destacar que las aplicaciones web modernas están vinculadas a una lógica que suele estar asociada a una base de datos. Estas aplicaciones se conectan con el servidor de base de datos correspondiente para recuperar y mostrar datos de manera amigable para el usuario. La capa del modelo en Django es una parte del patrón Modelo-Vista-Controlador, denominado Models-Templates-Views en este framework. Esta capa gestiona el acceso a la base de datos de la aplicación, almacenando información sobre cómo se validan, acceden y comportan los datos, así como las relaciones entre ellos. En la siguiente sección, se detallarán las distintas partes de esta capa.

4. Migraciones

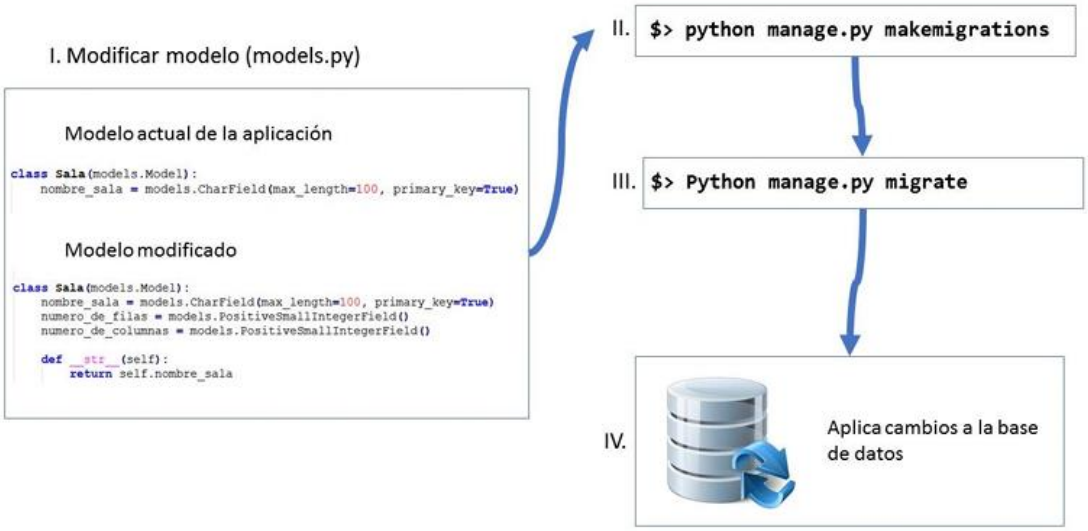
Durante el desarrollo de una aplicación Django, es posible realizar modificaciones en el archivo models.py, ajustando los modelos según sea necesario, como añadiendo nuevos campos, eliminando existentes o estableciendo nuevas relaciones entre clases. Estas operaciones se reflejan mediante las migraciones de Django, las cuales gestionan y trabajan con el esquema de la base de datos de manera abstracta para el programador.

Las migraciones evitan que el programador tenga que realizar manualmente cambios en la base de datos, ya que Django se encarga de ello. Además, se destaca la presencia de un sistema de control de versiones que registra los cambios realizados en los modelos, permitiendo retroceder si es necesario.

Se disponen de distintos comandos para interactuar con las migraciones, como Migrate para sincronizar modelos y tablas, Makemigrations para crear nuevas migraciones, Sqlmigrate para visualizar scripts SQL, y Showmigrations para ver el historial de migraciones. Este enfoque eficiente de Django facilita a los desarrolladores realizar cambios en los modelos sin afectar la

base de datos existente, ahorrando tiempo y líneas de código. A continuación, se detallarán los pasos a seguir al realizar cambios en un modelo específico.

Figura 12 *Proceso de modificación de un modelo junto con migraciones.*



Nota. Imagen tomada de “ESTUDIO DEL FRAMEWORK DE DESARROLLO WEB DJANGO”

4. Capa De Vista

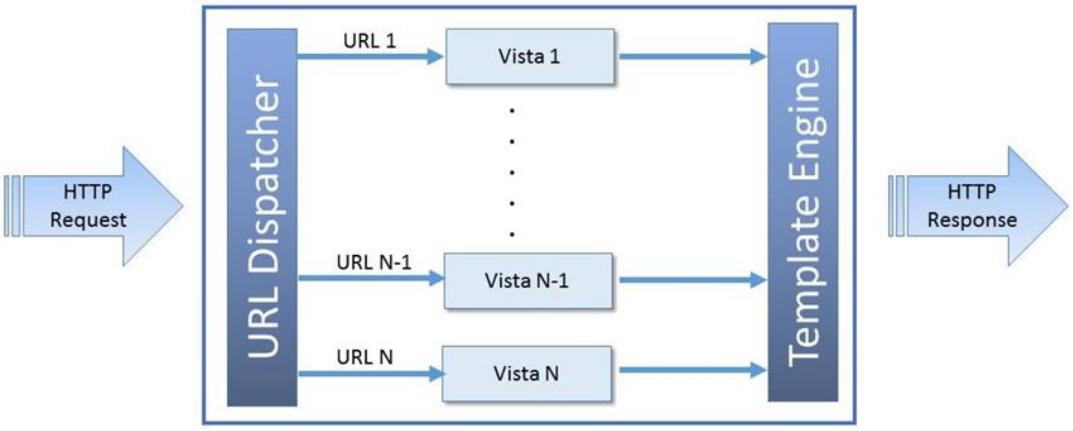
En Django, después de definir la capa del modelo que organiza los datos en la base de datos, entra en acción la capa de la vista. Esta capa muestra la información al usuario, procesando lo que el modelo tiene guardado y pasándolo a la capa, que se encarga de la presentación final.

Las vistas en Django son funciones basadas en Python que gestionan las solicitudes web del usuario y devuelven respuestas adecuadas. Pueden entregar distintos tipos de datos, como páginas HTML, listas de elementos o imágenes, proporcionando dinamismo a las páginas web.

Las funciones vistas que se encuentran en el archivo views.py, contienen la lógica para manejar las solicitudes y enviar respuestas. La relación entre las URLs y las funciones vistas asegura que el usuario reciba la información solicitada al interactuar con la aplicación.

Se pasa a mostrar un diagrama para entender como la capa de la vista, maneja interiormente las peticiones HTTP que se producen en una aplicación:

Figura 13 *Manejo de solicitudes y respuestas HTTP por la capa de la vista.*



Nota. Imagen tomada de “ESTUDIO DEL FRAMEWORK DE DESARROLLO WEB DJANGO”

- Se recibe la petición del usuario de la aplicación.
- Dicha dirección URL es manejada por el mapeador de URLs que dispone la capa de la vista, la cual realizará el mapeo entre la URL recibida y la vista que tiene asociada en el URLconf.
- La vista se pasa al motor de plantillas, el cual, se comentará en el siguiente capítulo.
- Se devuelve al cliente una respuesta HttpResponse.

5. Capa De Plantillas

La capa de plantillas en Django se encarga de presentar los datos al usuario final en el navegador. Recibe la información de la capa de la vista, que a su vez la obtiene del modelo.

Las plantillas en Django definen cómo se mostrará la información y pueden incluir contenido estático y partes dinámicas. Utilizan el motor de plantillas DTL (Django Template

Language) que permite generar diferentes formatos de texto. Las plantillas reciben los datos a través de contextos, que son diccionarios en Python que son proporcionadas por las vistas.

Para configurar los motores de plantillas en Django, se utiliza el archivo settings.py del proyecto. Se pueden especificar detalles como lo son backend de plantillas, directorios de búsqueda, y opciones específicas. Django facilita una API para cargar y renderizar plantillas almacenadas dentro del sistema de archivos, independientemente del backend utilizado.

En resumen, la capa de plantillas contribuye a crear aplicaciones web amigables y prácticas, permitiendo al programador gestionar la presentación de datos de manera eficiente.

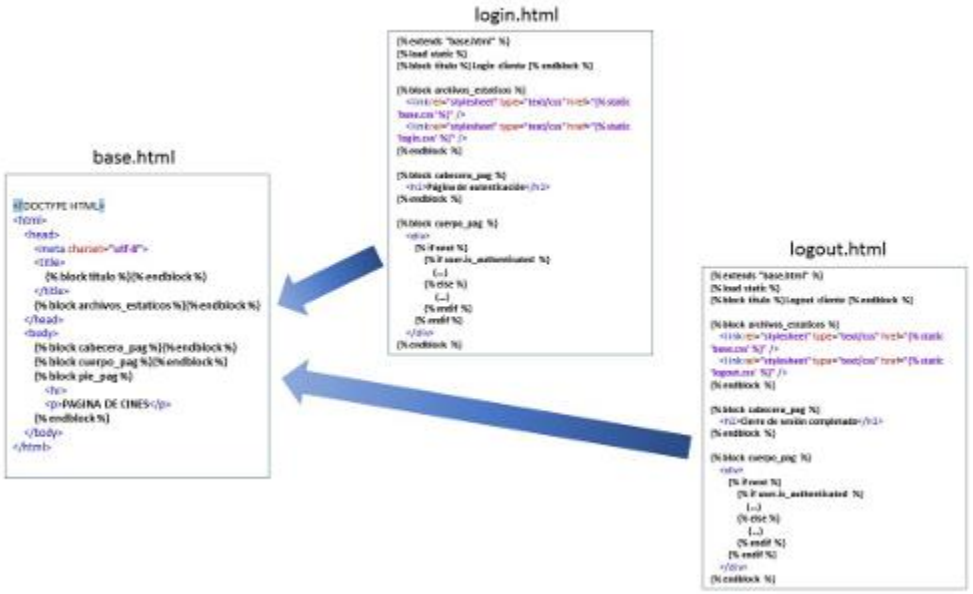
5.1 Herencia De Plantillas

Django simplifica el desarrollo web al evitar la duplicidad de código mediante un robusto sistema de herencia para las plantillas. Se recomienda crear una plantilla base con código y una estructura común. Las plantillas hijas heredan de la base y contienen su propio contenido.

Para dividir el contenido, se utilizan bloques definidos en la plantilla base con las etiquetas "{% block nombre_bloque %}" y "{% endblock %}". Las plantillas hijas empiezan con "{% extends plantilla_padre.html %}" para indicar a Django que cargue la plantilla base. Django detecta las etiquetas "block" y reemplaza el contenido de los bloques con el de las plantillas hijas.

Las plantillas hijas no necesitan definir todos los bloques de la base; en ese caso, se muestra el contenido predeterminado de la plantilla padre. Este sistema de herencia permite una gestión eficiente y sin redundancias en el desarrollo de aplicaciones en Django.

Figura 14 *Ejemplo de herencia de plantillas.*



Nota. Imagen tomada de “ESTUDIO DEL FRAMEWORK DE DESARROLLO WEB DJANGO”

6. Formularios En Aplicación Web

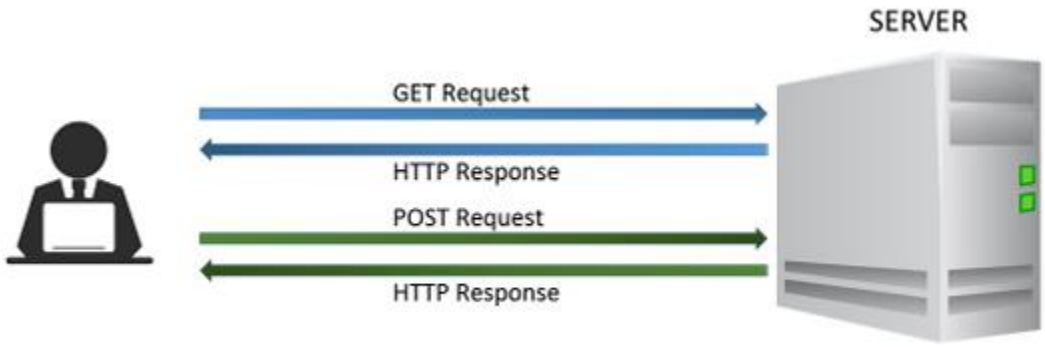
En el ámbito de la programación web, los formularios desempeñan un papel crucial al posibilitar la interacción del usuario con la aplicación. Estos permiten al usuario realizar diversas acciones, como ingresar texto o seleccionar entre varias opciones. Los formularios HTML son herramientas que facilitan la creación de objetos en las tablas de la base de datos y la obtención de información de la misma. El siguiente apartado explorará la manera en que Django se encarga de manejar los formularios web.

En un formulario web, se emplean estos métodos HTTP, GET y POST, según su propósito específico. El método GET envía los datos de manera visible en la URL, siendo adecuado para formularios de búsqueda web, pero no recomendado para formularios de inicio de sesión debido a posibles problemas de seguridad.

En cambio, el método POST envía datos de manera invisible al usuario y se utiliza para solicitudes que implican cambios en el sistema, como iniciar sesión o enviar formularios con gran cantidad de datos. Es crucial considerar la seguridad de la aplicación al elegir el método HTTP, ya que esto evita posibles amenazas de usuarios malintencionados que intenten comprometer el sistema o suplantar la identidad de otros usuarios debido a errores en la gestión de los métodos de formularios.

Para simplificar, se presenta un diagrama de las peticiones GET y POST entre el usuario y el servidor, ilustrando el intercambio de mensajes que tiene lugar.

Figura 15 *Diagrama de petición – respuesta de métodos GET y POST de HTTP.*



Nota. Imagen tomada de “ESTUDIO DEL FRAMEWORK DE DESARROLLO WEB DJANGO”

7. Usuarios En Django

En las aplicaciones web, la gestión de usuarios es esencial para garantizar la seguridad y controlar el acceso a distintas partes. Django incorpora un sistema de autenticación predeterminado que permite administrar cuentas, accesos y sesiones de usuarios, incluyendo la creación de grupos con permisos específicos.

El framework ofrece herramientas genéricas para la gestión de usuarios, pero aspectos específicos como la fortaleza de contraseñas o el número de intentos de inicio de sesión pueden requerir implementaciones adicionales de terceros, como django-axes para el control de intentos fallidos de inicio de sesión.

Django proporciona un módulo 'django.contrib.auth' con modelos, vistas y plantillas predefinidas para simplificar el trabajo del programador. Este sistema se configura automáticamente al generar el proyecto y se encuentra en el archivo settings.py.

Aunque Django ofrece un sistema completo, es posible crear un sistema de autenticación personalizado, ya que algunos aspectos son extensibles y reemplazables.

7.1 Tipos usuarios en Django

Es posible tener en una aplicación numerosos tipos de usuarios debido a los permisos que tengan asignados cada uno. Pero de una forma general, en Django, nos podemos encontrar con estos tres tipos de usuarios:

User, Este tipo de usuarios son una instancia de la clase django.contrib.auth.models.User. Esta clase se puede considerar como la clase de la que parten los otros tipos de usuarios, ya que se pueden entender como implementaciones especiales de la misma. En los próximos apartados se comentarán particularidades de esta clase, así como sus atributos y métodos.

AnonymousUser, Será una instancia de la clase jango.contrib.auth.models.AnonymousUser. Dicha instancia una implementación particular de la clase django.contrib.auth.models.User. Por ejemplo, algunas particularidades que posee dicha clase son: o No tiene nombre de usuario ni contraseña, por lo que el campo username será siempre nulo y el método set_password() y check_password() no los tendrá implementados. o No pertenece a ningún grupo y no posee ningún permiso especial. Por lo que los campos groups y user_permissions serán vacíos y is_active, is_staff y is_superuser estarán a False.

Superuser, es como el primer caso, User, pero tiene unas características especiales. Los campos is_staff y is_superuser los tiene a True, por lo que es lo que le marca la diferencia respecto a otro usuario de la aplicación. Este tipo de usuarios son los que tendrán permisos para acceder a la parte de administración de Django.

7.2 Proceso de autenticación en petición Web

Para que un usuario pueda iniciar sesión y cerrar sesión, con el backend de autenticación por defecto, existen dos formas para ello. La primera es mediante una vista personalizada, recoger los parámetros username y password del correspondiente formulario y, posteriormente, llamar al método authenticate. A dicho método será necesario pasarle el objeto request junto con el usuario y la contraseña. Esto se realiza para comprobar que si no devuelve None, se pueda llamar al método login() pasándole request y el resultado de la llamada a authenticate. Ambos métodos pertenecen al paquete ‘django.contrib.auth’.

7.3 Sesiones de los usuarios

En lo que respecta a la gestión de usuarios, un aspecto crucial involucra el uso de cookies y sesiones de la aplicación. Una cookie es información enviada por un sitio web y almacenada en

el navegador del cliente para mantener datos del usuario. Gracias a ellas, un usuario no necesita ingresar sus credenciales en cada página, ya que se almacena una cookie al iniciar sesión.

Django aborda la complejidad de las sesiones proporcionando un entorno de sesiones al programador. Para usarlo, se debe agregar 'django.contrib.sessions.middleware.SessionMiddleware' al proyecto y tener 'django.contrib.sessions' en INSTALLED_APPS, configuración incluida por defecto en la creación de un proyecto Django.

Django almacena los datos de la sesión en el servidor, específicamente en la tabla "django_session" de la base de datos, donde se guarda una versión codificada del identificador de sesión (hash). La gestión de las sesiones se realiza automáticamente, almacenando el identificador del usuario al autenticarse y eliminando los datos de la sesión al cerrarla.

Es crucial para la seguridad de la aplicación asegurarse de que, al cerrar sesión, los datos de la sesión se eliminen por completo para evitar riesgos de seguridad. Django maneja esto internamente al utilizar las vistas LoginView y LogoutView. Para proporcionar máxima seguridad, Django evita almacenar información de la sesión en la URL y prefiere el uso de cookies para evitar posibles vulnerabilidades.

7.4 Limitar accesos a usuarios registrados.

En cuanto a la restricción de acceso a usuarios registrados, Django ofrece el decorador login_required(), ubicado en 'django.contrib.auth.decorators'. Este se utiliza en vistas que requieren autenticación previa y redirige a usuarios no identificados a la página de inicio de sesión predeterminada. Para usuarios identificados, ejecuta la vista normalmente. Los parámetros opcionales son redirect_field_name (para especificar el campo de redirección) y login_url (para indicar la página de inicio de sesión personalizada).

En el caso de vistas basadas en clases, Django proporciona el Mixin django.contrib.auth.mixins.LoginRequiredMixin. Si un usuario no registrado intenta acceder, se redirige a la página de inicio de sesión o muestra un error HTTP 403 si raise_exception de AccessMixin está en True.

A continuación, se presenta un ejemplo de una vista restringida exclusivamente para usuarios registrados en la aplicación:

```
from django.contrib.auth.decorators import login_required

@login_required
def sesion_detalle(request, identificador):
# resto de código de la vista
```

Además de limitar el acceso a usuarios registrados en diferentes secciones de la aplicación, también se puede restringir el acceso si un usuario específico no cumple con un requisito particular o basándose en sus permisos. Para el primer caso, Django proporciona el decorador user_passes_test o, en su lugar, el Mixin UserPassesTestMixin para vistas basadas en clases. Esto permite establecer requisitos que el usuario debe cumplir para ejecutar la vista o denegarle el acceso. En el segundo caso, Django ofrece el decorador permission_required y el Mixin PermissionRequiredMixin, donde se deben incluir los permisos necesarios para que el usuario acceda a esa parte de la aplicación.

Otra forma de diferenciar entre usuarios registrados y no registrados es al mostrar información en las plantillas de la aplicación. Es posible verificar si un usuario está registrado mediante request.user.is_authenticated.

```
{% if user.is_authenticated %}

<p>Ya estás autenticado en la web.</p>

{% else %}
```

```
<p>Introduce tus credenciales para loguearte como cliente</p>

{% endif %}
```

8. Sitio de Administración de Django

Teniendo en cuenta que esta aplicación es desarrollada para la empresa Laboratorio María Salomé y su funcionalidad no será muy dada al público y más a el área de tecnología lo cual es muy importante que esta aplicación pueda disponer de un sitio de administración en el cual se lleven las labores de control y seguimiento de las tareas de los usuarios encargados de la gestión de los activos de la empresa.

Django se centra en el desarrollo rápido de aplicaciones web y simplifica la administración mediante una interfaz HTML para los administradores. Esta interfaz, presente en 'django.contrib.admin', se basa en la lectura de metadatos de modelos para facilitar la modificación del contenido. Diseñada para administradores no técnicos, proporciona un entorno claro y limpio, eliminando tareas comunes de desarrollo como el control de inicio de sesión, la presentación de formularios y la validación de datos, ahorrando tiempo a los programadores.

Además, la interfaz asegura que solo usuarios con los permisos adecuados accedan, evitando problemas potenciales. Al estar incluida en 'django.contrib', específicamente en 'django.contrib.admin', siendo una aplicación Django, cuenta con sus propios modelos, plantillas y vistas. Para acceder a ella desde una aplicación, es necesario integrarla en el URLConf, y toda la configuración necesaria se detalla en este capítulo.

8.1 Preparación de entorno administrador

La implementación del sitio del administrador es muy sencilla, solo es necesario tener en cuenta lo siguientes puntos:

- Lo primero que es necesario realizar, es añadir ‘django.contrib.amdin’ dentro de settings.py en el apartado de aplicaciones instaladas: INSTALLED_APPS. Con ello se podrá utilizar dicha aplicación.

- Otra configuración necesaria, es definir una expresión regular para determinar cuándo se debe de dirigir a la interfaz de los administradores. Para ello se deberá de incluir en el URLConf una instancia de AdminSite.
- Para poder acceder a ella por primera vez es un requisito tener una cuenta de un usuario que sea Superuser. Para ello hay que crear un usuario con dichas características.
- Como se ha comentado en la introducción, en la interfaz de administración es posible realizar cambios en algunos modelos con el fin de tener un mayor control de los mismos, pudiendo crear modificar y eliminar objetos en las tablas. Es posible elegir que modelos se desean modificar registrándolos en la página de administración. Ya que cada modelo tiene una clase ModelAdmin con las funcionalidades propias que tendrán en el sitio.

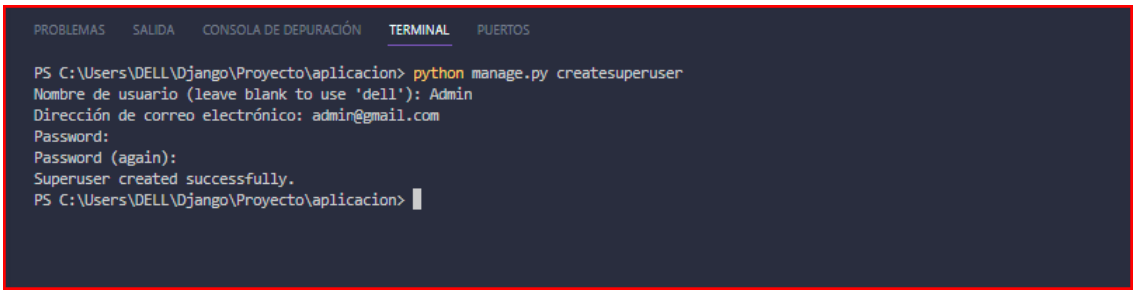
8.2 Creación SuperUser

Es necesario crear un usuario como primera medida, ya que la base de datos inicialmente no contiene información.

Este primer usuario es llamado SuperUser quien será el administrador y tendrá todos los permisos para crear, eliminar y dar permisos a los demás usuarios.

Figura 16 *Crear superuser*

Datos: Nombre de usuario, Dirección de correo electrónico, Pasword



Fuente: elaboración propia

La respuesta es: “Superuser created successfully”, Súper usuario creado exitosamente.

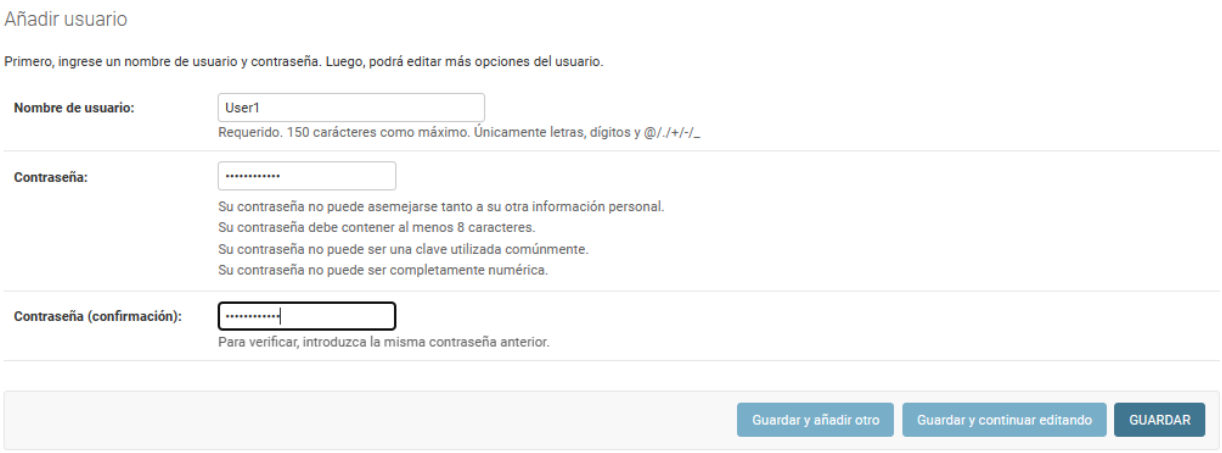
Para el registrar un usuario se deben proporcionar un usuario y una contraseña la cual debe contener mínimo una minúscula, una mayúscula, un digito, un carácter especial y su tamaño debe estar entre 8-15.

Figura 17 *Usuario*

Fuente: elaboración propia

En el panel de administración, encuentras el modelo de usuario, que generalmente se llama "User" o "Usuarios". Al hacer clic en él, se abre la sección dedicada a la gestión de usuarios.

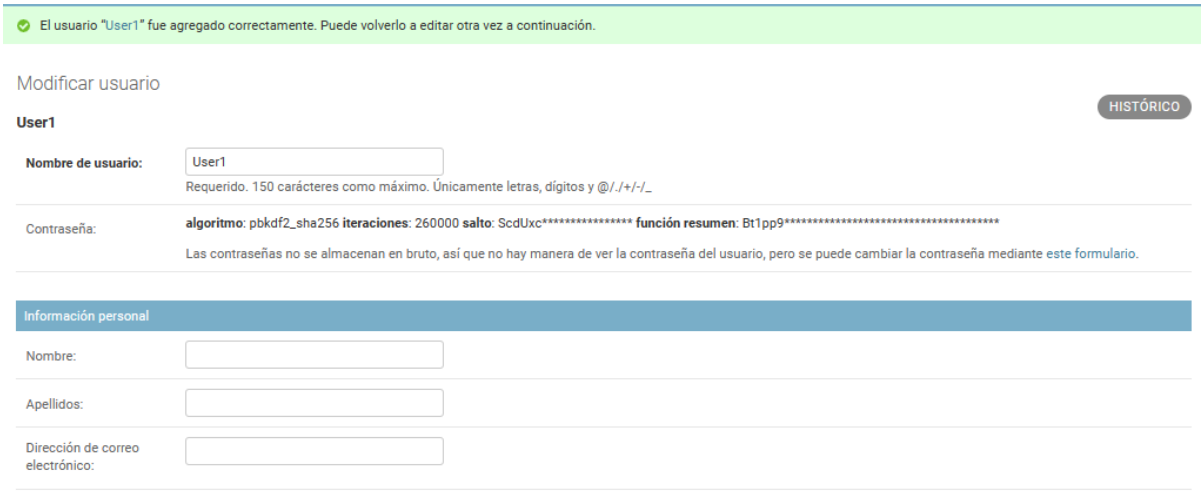
Figura 18 *Crear usuario*



Fuente: elaboración propia

Dentro del formulario, puedes ingresar la información del nuevo usuario, como nombre de usuario, y contraseña.

Figura 19 *Confirmación*



Fuente: elaboración propia

La respuesta es: El usuario “User1” fue agregado correctamente.

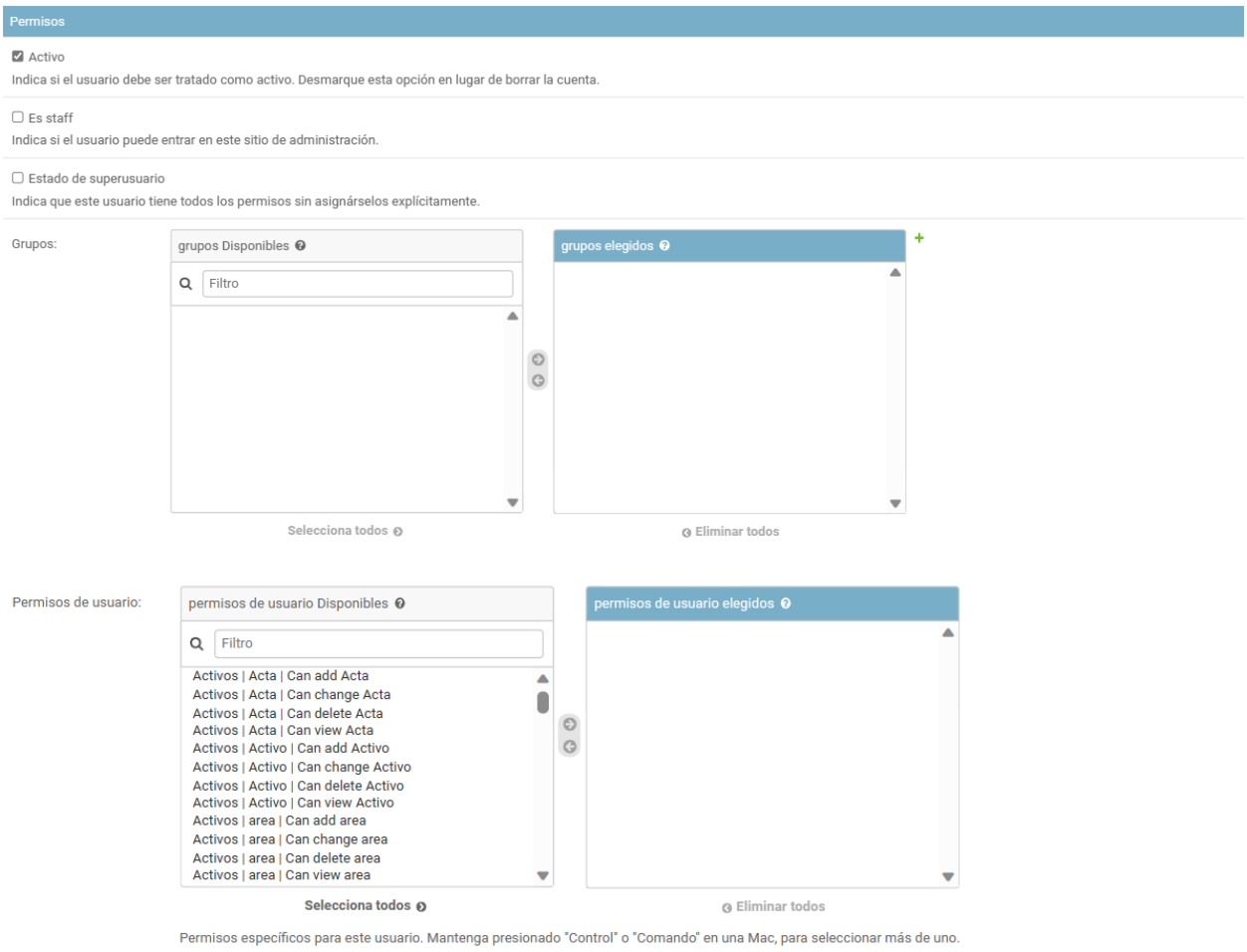
Por defecto cada vez que se haga un registro los usuarios obtiene su rol en el sistema como User, si se desea asignarle permisos deberá editar el usuario.

8.3 Gestión de permisos

La gestión de permisos en Django es crucial para garantizar la seguridad y la integridad de la aplicación según la necesidad de cada usuario.

En la vista de detalle o editar el usuario, encontramos una sección dedicada a los permisos. Aquí, puedes asignar o revocar permisos específicos según las necesidades del mismo. Django proporciona una interfaz muy intuitiva y fácil de usar para esta gestión de permisos.

Figura 20 *Modulo Permisos*



Fuente: elaboración propia

En este módulo tenemos diferentes clases de permisos que le podemos asignar a el usuario, se pueden asignar a un grupo con permisos ya asignados por el administrador, o se pueden asignar permisos específicos según la necesidad que presente el usuario, una vez que hayas realizado los cambios necesarios, guardas la información del usuario.

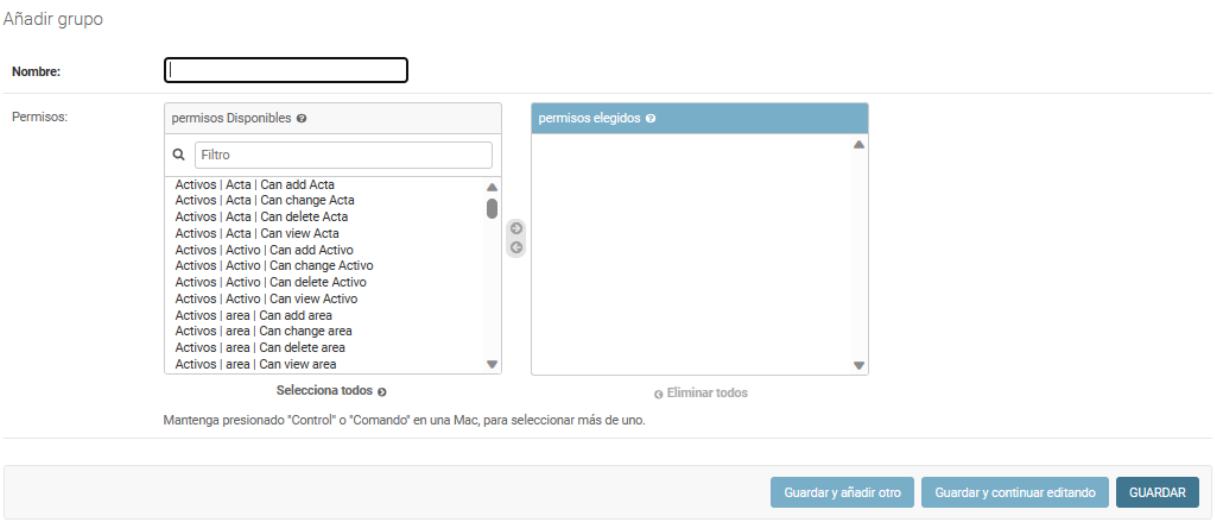
El siguiente modulo al que se tiene acceso como administrador es Grupos que es fundamental para organizar los usuarios en categorías o roles específicos, facilitando asi la asignación de permisos de manera más eficiente

Grupos

En el modelo de usuario de Django, puedes crear grupos para representar roles específicos en tu aplicación. Por ejemplo, podrías tener grupos como "Administradores", "Editores" o "Usuarios Normales".

Para la creación de grupos se ingresa al módulo que se muestra en la Figura 10.

Figura 21 *Crear grupos*



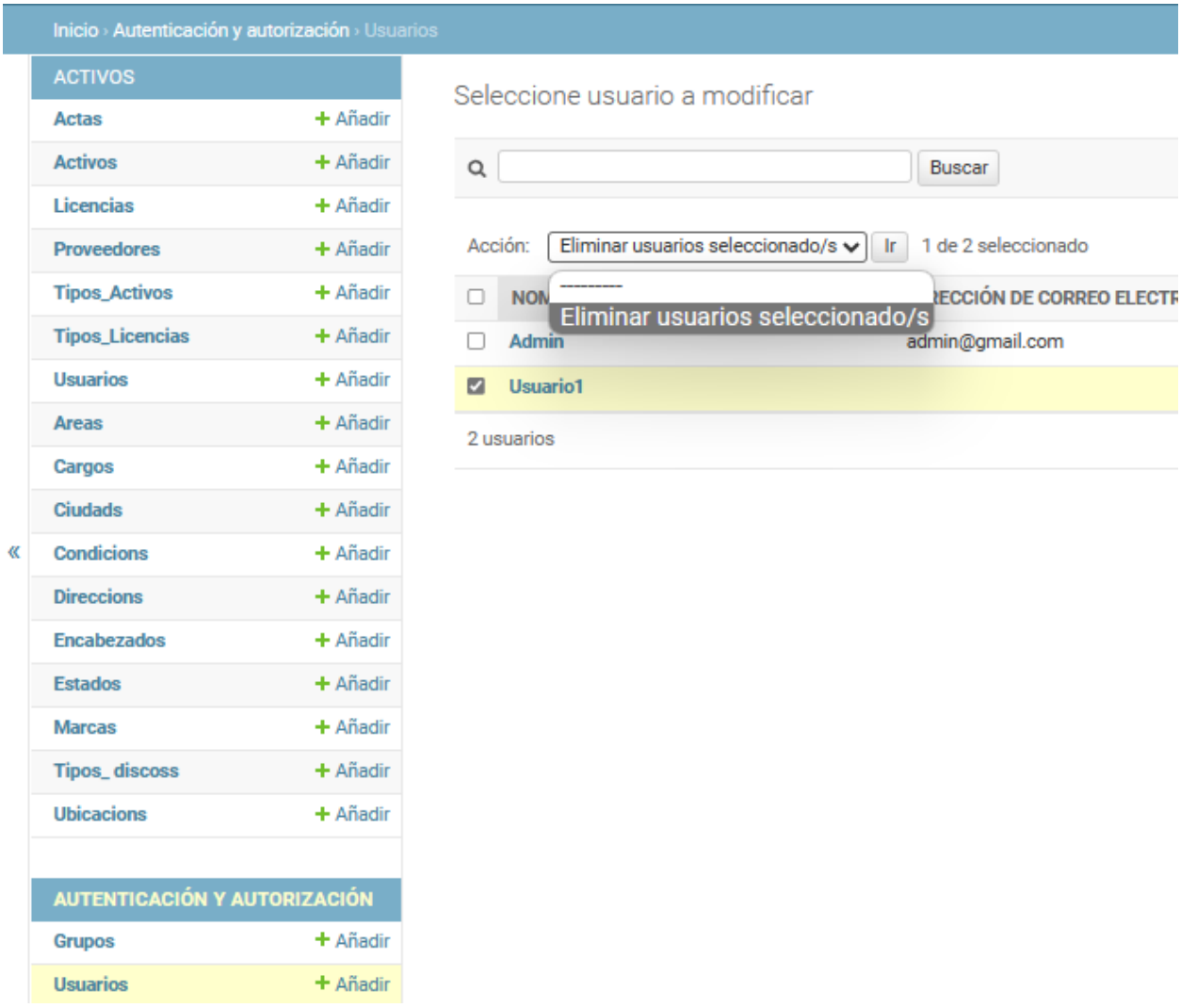
Fuente: elaboración propia

8.4 Eliminar un usuario

Eliminar un usuario o muchos al mismo tiempo es una tarea fácil de realizar gracias a el administrador de Django ya que nos permite seleccionar uno o muchos usuarios, al agregar la acción eliminar y en el botón “ir”, luego de confirmar la acción podremos aplicarla de una manera fácil y rápida.

Ruta:localhost:8000/ admin/Activos/usuario/40/delete/

Figura 22 *Eliminar usuario*



Fuente: elaboración propia

De esta manera habremos eliminado el o los usuarios deseados.

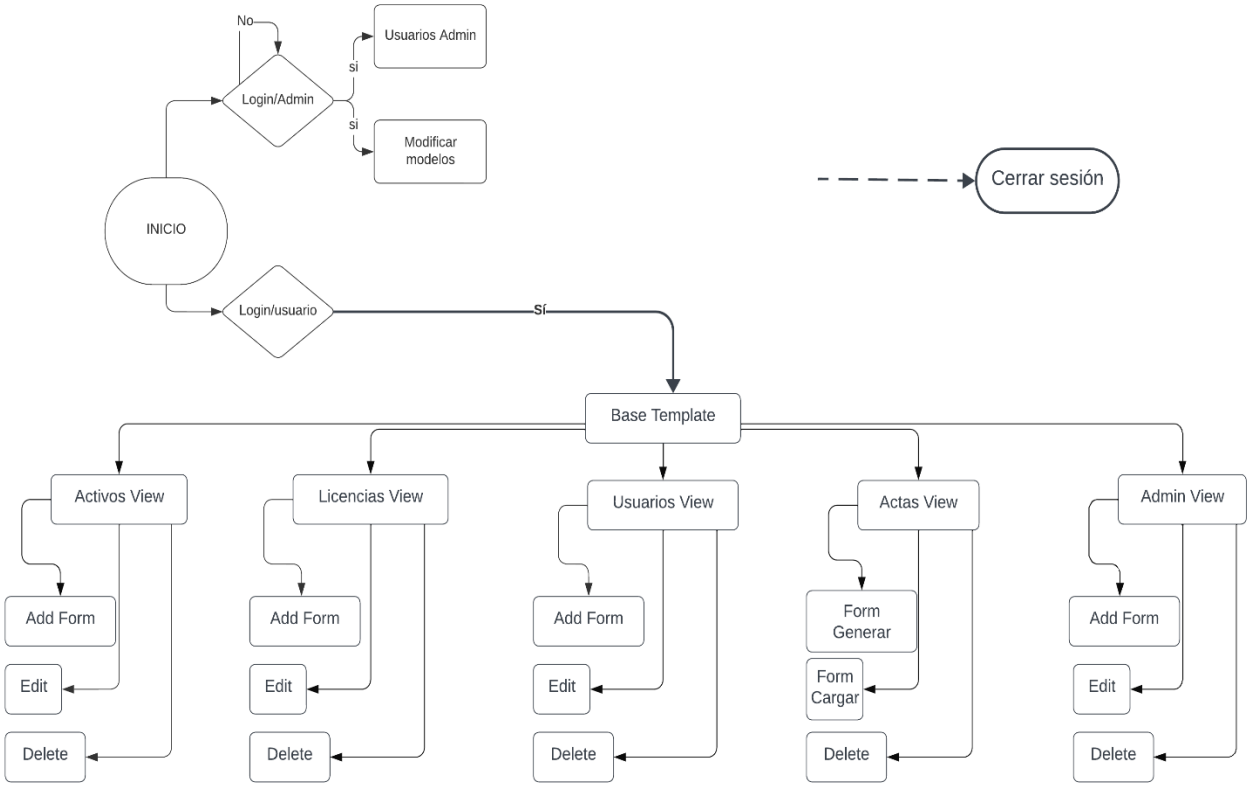
8.5 Implementación De La Aplicación

En este apartado se va a describir el proceso de implementación de la aplicación desarrollada siguiendo con los requisitos entregados por el jefe de TI del Laboratorio María Salome

8.5.1 Flujo De La Aplicación

En este apartado de pretender mostrar de una manera visual, legible y amigable para el lector, el flujo de la aplicación de Gestión de activos que se a desarrollado. Para ello se muestra a continuación un diagrama de flujo de la misma.

Figura 23 *Flujo De Aplicación Desarrollada*



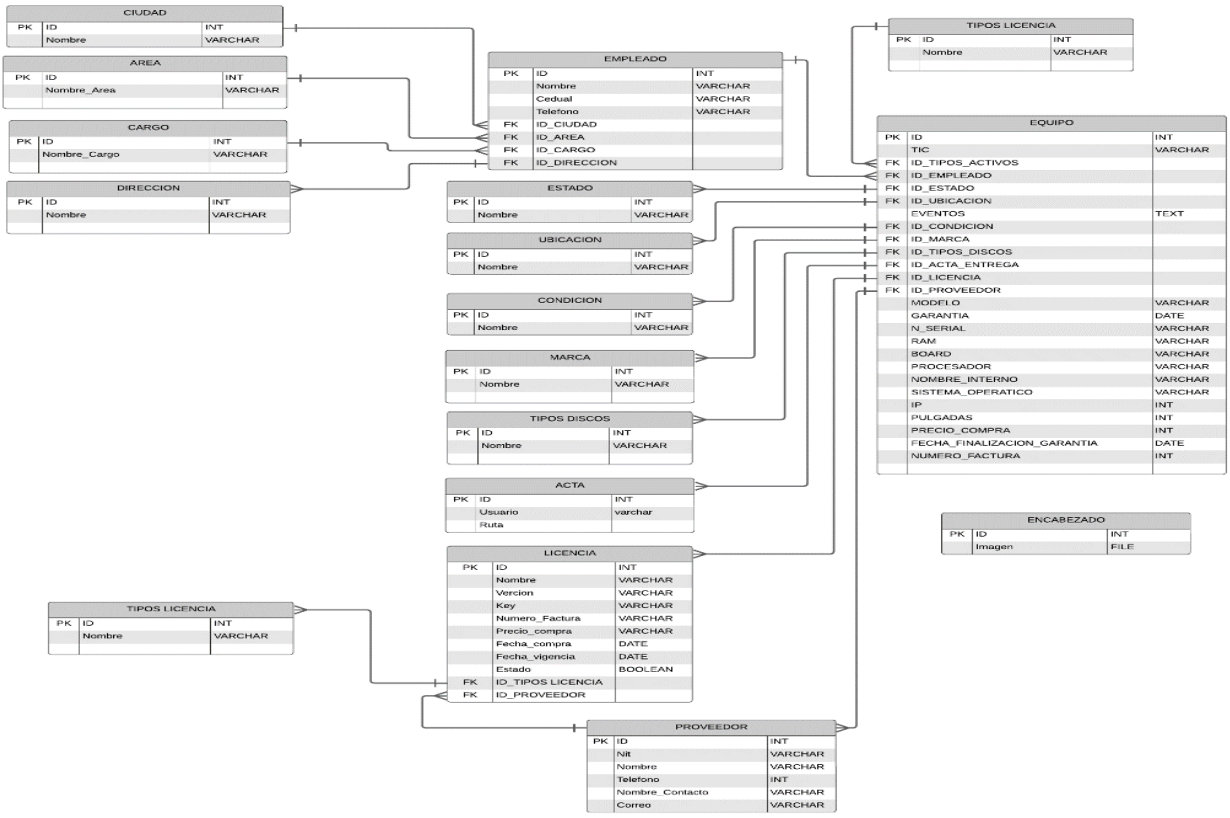
Nota. Elaboración propia

En el diagrama anterior se puede apreciar de una forma reducida el flujo de la aplicación que puede seguir un determinado usuario entre las distintas páginas de la misma. Por motivos de legibilidad, no se ha incluido una unión a ‘Cerrar sesión’ desde cada pantalla. Pero se da por hecho de que el usuario en cualquier momento puede cerrar la sesión en la aplicación (siempre que la haya iniciado anteriormente). Es posible distinguir claramente las dos partes de la aplicación: la parte del usuario y la parte de administración.

8.5.2 Desarrollo del modelo

En este inicio, se procederá a elaborar el diseño del modelo de la aplicación. Es importante señalar que, en un principio, se ha empleado el motor de base de datos SQLite3, el cual es el predeterminado en Django. En esta sección, se crearán las diversas clases que se asociarán con las tablas en la base de datos. Para lograrlo, se requiere realizar un análisis previo de los diferentes tipos de objetos presentes en la aplicación, junto con los atributos que estos contendrán. Los modelos que se implementarán son los siguientes:

Figura 24 *Modelado de la base de datos de la aplicación desarrollada.*



Nota. Elaboración propia

Una vez que se tiene claro los modelos a utilizar junto con sus campos y relaciones, es hora de implementarlo en el archivo models.py.

Previamente se deben realizar la siguiente importación:

```
from django.db import models
```

Es necesario realizarla ya que cada modelo es una subclase de models.Model.

Se representarán los objetos previamente mencionados como clases Python, subclases de 'django.db.models.Model'. Cada campo de estas clases corresponderá a una variable de la clase Model, representando un campo en la tabla de la base de datos.

Estos campos serán instancias de la clase Field para indicar el tipo de dato que contendrán, recibiendo parámetros para cumplir con los requisitos. Además, se definirá el método __str__() para cada clase, devolviendo una representación Unicode del objeto.

Para cada modelo, se definirá una clase interna Meta según sea necesario, proporcionando metadatos específicos como opciones de ordenamiento, nombres en singular y plural, entre otros. Las opciones Meta incluyen 'ordering', 'verbose_name' y 'verbose_name_plural'.

8.5.2.1 Tipos de relaciones en los modelos Django

En Django, los tipos de relaciones en los modelos permiten establecer conexiones entre las tablas de la base de datos. Aquí hay una breve explicación de los tipos de relaciones que se han utilizado entre los modelos:

ForeignKey (Muchos a Uno):

Se utiliza para establecer una relación muchos a uno entre dos modelos.

Este tipo de relación se ha utilizado para vincular los siguientes modelos:

- Activo con Usuario, Estado, Proveedor, Acta, Marca, Condición, Ubicación y con Tipos_activos.
- Acta con Usuario.
- Usuario con Dirección, Area, Cargo y con Ciudad.

- Licencia con proveedor y con tipos_licencia.

La consecuencia de haber utilizado ForeignKey sobre dichos campos es que Django crea un índice sobre ellos de forma automática en su tabla de la base de datos.

ManyToManyField (Muchos a Muchos):

Permite establecer una relación muchos a muchos entre dos modelos.

Este tipo de relación se ha utilizado para vincular los siguientes modelos:

- Activo con Licencia

Se ha empleado este tipo de relación ya que, por ejemplo, un activo puede tener muchas licencias asociadas y una licencia puede estar en muchos activos. con el objetivo de poder extender dicha clase para añadirla nuevos campos adicionales que son requeridos por los requisitos del problema. Estas relaciones proporcionan la capacidad de modelar la complejidad de las relaciones entre los objetos en una aplicación Django de manera eficiente y sencilla.

En lo que respecta a las migraciones, es importante destacar que cada vez que se efectuó una modificación en el modelo, se ejecutaron los comandos "makemigrations" y "migrate" en ese mismo orden.

A continuación, se presenta en secuencia los comandos y los resultados de una de las primeras migraciones en la aplicación:

```
PS C:\Users\DELL\Documentos\Python\proyecto\aplicacion> python manage.py makemigrations
Migrations for 'aplicacion': aplicacion\migrations\0001_initial.py
- Create model Activo
- Create model Acta
- Create model Usuario
- Create model Proveedor
```

```
- Create model Licencia
- Create model Tipos_Licencias
- Create model Tipos_Activos
- Create model Ciudad
- Create model Ubicacion
- Create model Direccion
- Create model Area
- Create model Estado
- Create model Marca
- Create model Cargo
- Create model Condicion


PS C:\Users\DELL\Documentos\Python\proyecto\aplicacion> python manage.py migrate
Operations to perform:
  Apply all migrations: Activos, admin, auth, contenttypes, sessions
Running migrations:
  Applying aplicacion.0001_initial... OK
```

8.5.3 Implementación de vistas y formularios

En este apartado se van a describir las vistas y los formularios que se han creado para el desarrollo de la aplicación. Como se ha comentado a lo largo del desarrollo del Trabajo de Fin de Grado, las vistas serán las encargadas de qué datos mostrar a los clientes. Es por ello, que va a existir una vista para cada una de las partes que tiene la aplicación, con el objetivo de que cada una de ellas

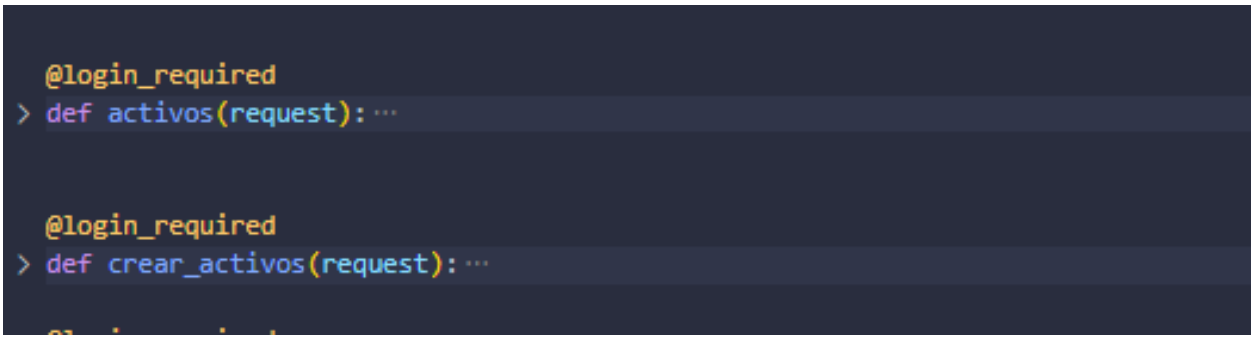
cumpla con los requisitos iniciales. Dichas funciones Python se han implementado en el archivo views.py. Lo primero que es necesario importar a dicho archivo son los modelos que se han declarado anteriormente, junto con los formularios que se crearán a continuación, entre otras clases necesarias. Cabe comentar que se han definido funciones para las vistas, las cuales, recibe como parámetro obligatorio un objeto HttpRequest. En dicho objeto viaja la información referente a la solicitud realizada por el usuario, por ejemplo, si el método empleado es GET o POST, entre otra información posible. Pueden recibir opcionalmente más parámetros, que generalmente, será recogido de la URL, el cual es la misión del URLConf. Estos parámetros opcionales se comentarán en la descripción posterior de las vistas.

En cuanto a los decoradores posibles que pueden tener las vistas, se ha utilizado el @login_required, la cual realiza lo siguiente:

- Si el usuario está logueado en el sistema, ejecuta la vista normal.
- Si el usuario no se encuentra actualmente logueado en la aplicación, redirigirá a la página de inicio de sesión y pasa como argumento la ruta a la cual el usuario se dirigía, para que una vez que inicie correctamente sesión, sea redirigido a ella.

En el archivo de configuración settings.py del proyecto, se va a definir lo siguiente:

Figura 25 *Autenticación de Usuario*



Fuente: Elaboración propia

Con ello se establece que la URL de login para la aplicación, es la que se le pasa a continuación mediante una URL con nombre. Con ellos correctamente señalado Django sabrá a que URL se debe de dirigir cuando un inicio de sesión sea requerido.

Vistas más importantes implementadas en la aplicación:

Activos

Objetivo: Mostrar los Activos vigentes de la empresa con sus respectivas características

Notas a destacar:

- Se emplean filtro de búsqueda para hacer más fácil y amigable las consultas sobre tipos de activos o activos por Usuarios.
- Se emplean campos como estados para tener una fácil visualización de los equipos que estén en un estado definición como, por ejemplo: Disponible, En uso, En reparación.
- Se emplea una funcionalidad de poder eliminar registros por algún tipo de circunstancia.
- Dentro de esta vista se desarrolla la funcionalidad donde se muestran los tipos de licencias disponibles y podremos seleccionar más de una licencia y será asignada al Activo.
- Dentro de esta vista se desarrolla la funcionalidad de poder asignar un Usuario al Activo.

Crear activos

Objetivo: Mostrar un formulario con los campos sugeridos por los requerimientos para el adecuado registro de un activo en la base de datos.

Notas a destacar:

- Luego de crear el activo la vista nos redirigirá a la vista principal en este caso sería Activos.

Editar activos

Objetivo: Cargar todos los datos del activo seleccionado con el fin de poder modificar cualquier campo antes registrado del mismo.

Licencias

Objetivo: Como lo anterior, devuelve una tabla con los datos de las características de cada licencia con un diseño amigable e intuitivo para quien lo opera.

Notas a destacar:

- Se emplean filtro de búsqueda para hacer más fácil y amigable las consultas sobre tipos de licencias.

Crear licencias

Objetivo: Mostrar un formulario con los campos sugeridos por los requerimientos para el adecuado registro de licencias en la base de datos.

Notas a destacar:

- Luego de crear la licencia la vista nos redirigirá a la vista principal en este caso sería Licencias.

Usuarios

Objetivo: Mostrar los usuarios ingresados en el sistema con sus respectivos datos.

Notas a destacar:

- Se emplean filtro de búsqueda para hacer más fácil y amigable las consultas sobre tipos de licencias.

Crear usuarios

Objetivo: Mostrar un formulario con los campos sugeridos por los requerimientos para el adecuado registro de Usuarios en la base de datos.

Notas a destacar:

- Luego de crear el Usuario la vista nos redirigirá a la vista principal en este caso sería Usuarios.

Acta

Objetivo: Nos muestra dos formularios intuitivos y fáciles de manejar para la generación y almacenamiento de actas.

Notas a destacar:

- Se hace uso de librerías como **reportlab** para la generación del archivo PDF, asemejando a la estructura de actas de entrega entregada por el Jefe de TI de la empresa beneficiaria de la aplicación Web en Django.
- Se implementa un formulario donde solo escogiendo un usuario de la lista desplegable nos genera un archivo PDF que corresponde al acta de entrega del usuario donde se pintan los activos y las características solicitadas en los requerimientos de cada uno de los usuarios (empleados de la compañía).
- Se muestra un formulario donde luego de haber firmado el acta(PDF) y teniendo en cuenta que esta firma podría tardar un tiempo y acumularse diferentes actas, la vista nos muestra el listado de usuarios y un botón que desplegará el gestor de archivos predeterminado, donde podremos subir un acta y será asociado al usuario seleccionado.
- Se muestra una tabla con dos campos fundamentales del proceso que son el nombre del usuario y el link del archivo PDF asociado a ese usuario.

8.5.4 Desarrollo del URL Conf

Usando las vistas creadas previamente, se establecerá en el archivo *urls.py*. Este archivo es el encargado de vincular las direcciónes URL y las vistas, junto con otra información extra. Además de las importaciones estándar de Django, es crucial agregar las vistas definidas en *views.py*.

Cada vez que creamos una instancia de *url()* dentro de la variable *patterns*, le asignamos un nombre. Esto nos permite acceder a estas URLs desde otras partes de la aplicación.

Quedando la variable *urlpatterns* final de la siguiente manera:

Figura 26 *Variable urlpatterns*

```
urlpatterns=[

    path('',views.activos, name='activos'),
    path('logout/', exit ,name="exit"),


    path('activos', views.activos, name='activos'),
    path('activos/crear',views.crear_activos, name="crear_activos"),
    path('activos/editar.html/<int:id>',views.editar_activos,name="editar_activos"),

    path('licencias', views.licencias, name='licencias'),
    path('licencias/crear',views.crear_licencias, name="crear_licencias"),
    path('licencias/editar.html/<int:id>',views.editar_licencias,name="editar_licencias"),

    path('usuarios', views.usuarios, name='usuarios'),
    path('usuarios/crear',views.crear_usuarios, name="crear_usuarios"),
    path('usuarios/editar.html/<int:id>',views.editar_usuarios,name="editar_usuarios"),


    path('listas/crear_ciudades', views.crear_ciudades, name='crear_ciudades'),
    path('listas/crear_tipos', views.crear_tipos_activos, name='crear_tipos_activos'),
    path('listas/crear_ubicacion', views.crear_ubicacion, name='crear_ubicacion'),
    path('listas/crear_direccion', views.crear_direccion, name='crear_direccion'),
    path('listas/crear_area', views.crear_area, name='crear_area'),
    path('listas/crear_cargo', views.crear_cargo, name='crear_cargo'),
    path('listas/crear_proveedor', views.crear_proveedor, name='crear_proveedor'),

    path('actas', views.generar_actas, name='actas'),

]
```

Fuente: Elaboración propia

8.5.5 Implementación de las plantillas

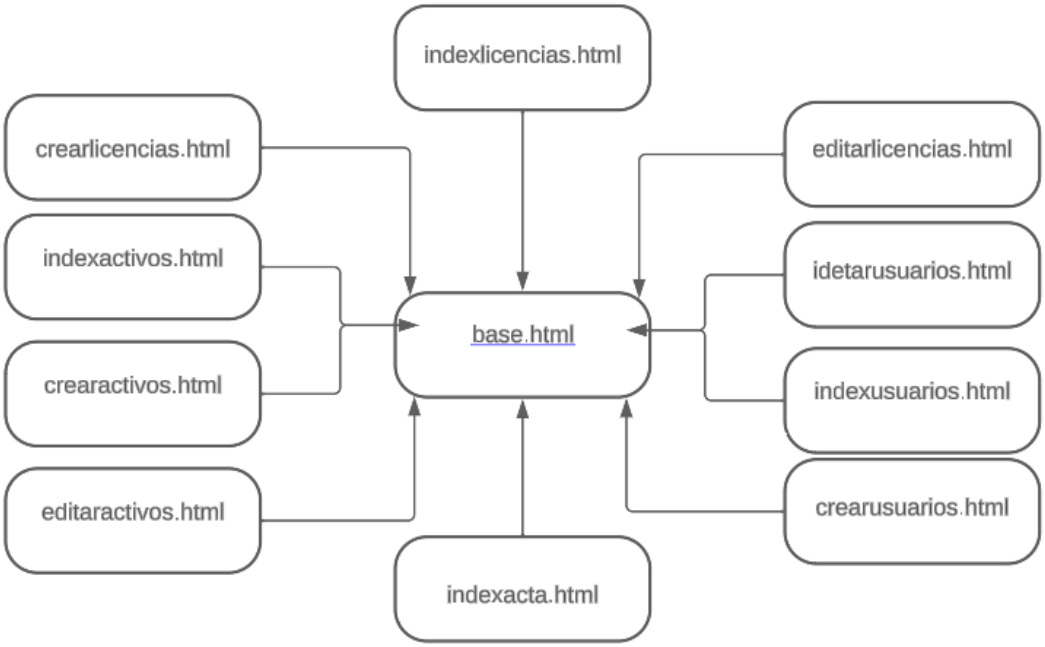
Para la implementación de las plantillas, se ha creado un nuevo directorio en la carpeta raíz del proyecto: Activos/templates. En este directorio será donde se alojen las distintas plantillas generadas para la aplicación.

Para indicar a Django que tiene que cargar las plantillas del sistema de archivos, es necesario mostrarlo en el archivo settings.py.

Con el objetivo de reducir al máximo el número de líneas de código Se ha utilizado la herencia de plantillas. Para ello, se ha creado una plantilla base, que es el padre de todas, la cual definirá la parte común que será la barra de navegación (sidenav) y el resto de la página, así como distintos

bloques para que las plantillas hijas lo rellenen según sus necesidades. Se muestra un diagrama con el uso de las plantillas que se han utilizado en la aplicación actual:

Figura 27 *Diagrama de herencia de plantillas.*



Fuente: Elaboración propia

Para poder realizar dicha herencia de plantillas, es necesario en cada una de las plantillas hijas, introducir la siguiente línea en primer lugar:

```
{% extends "baseventanas.html" %}
```

Con este código anterior se indica al motor de plantillas de Django que la plantilla actual hereda de base.html, y por lo tanto podrá sobrescribir los bloques definidos en ella, aparte de implementar los suyos propios.

Para las plantillas creadas en este apartado, se han incluido las tecnologías CSS3, JavaScript y Bootstrap. Se han introducido con la finalidad de poder darle un mejor estilo a las páginas HTML generadas tras el renderizado de las plantillas y que se muestren de una forma más agradable y amigable al usuario.

8.5.6 Configuración del sitio de administración de Django

Como anteriormente se ha mensionado, Django brinda una interfaz de administrador que es cargada por defecto. En la aplicación que se a desarrollado, se a implementado dicha interfaz para un mayor control sobre la base de datos de la aplicación.

Para hacer uso de esta interfaz de administración, lo primero que hay que introducir es la siguiente línea de código en el archivo *URLConf*.

```
path('admin/', admin.site.urls),
```

Con esa línea se introducen todas las *URLs* del sitio de administración que trae por defecto Django.

Como anteriormente se menciono en el apartado 2 del capitulo 7, se crea un superusuario que tendrá permisos de administrador para acceder a esta parte.

Cabe resaltar que el usuario administrador será el único que podrá logearse en esta parte de la aplicación, será el encargado de crear usuario con permisos definidos según la necesidad presentada.

Introduciendo las credenciales del superusuario creado previamente, se muestra la siguiente pantalla:

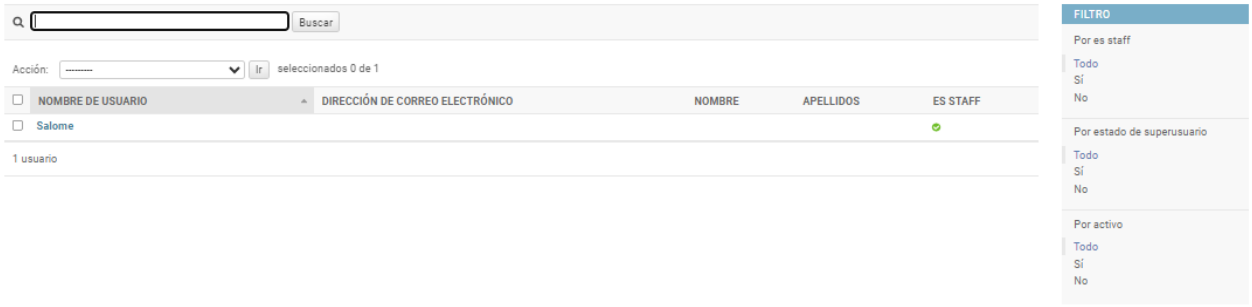
Figura 28 *Inicio del sitio de administrador de Django.*

Fuente: Elaboración propia

En ella es donde el administrador tiene todo el control tanto de la base de datos de la aplicación como de los usuarios registrados en ella.

Para verificar que el superusuario está correctamente creado se dara clic en ‘Usuario’, y se podrá ver que aparece correctamente el superusuario creado:

Figura 29 *Comprobación superusuario en 'Usuarios' del sitio de administración.*



Fuente: Elaboración propia

Una vez que se tiene levantado correctamente el sitio de administración, se va a trabajar sobre el archivo admin.py, en el cual, se incluirá toda la configuración que se desee añadir al mismo. En este archivo, se va indicar los modelos que se desee manipular desde el sitio de administración y se crearán subclases de ModelAdmin, mediante las cuales se permiten configurar diversas opciones para un determinado modelo.

Los modelos que se van a registrar para el sitio de administración son los siguientes:

I. Activo
II. Acta
III. Usuario
IV. Proveedor
V. Licencia
VI. Tipos_Licencias
VII. Tipos_Activos
VIII. Ciudad

IX. Ubicacion

X. Direccion

XI. Area

XII. Estado

XIII. Marca

XIV. Cargo

XV. Condicion

Para cada una de ellas se ha implementado una subclase de ModelAdmin, con la finalidad de editar las opciones facilitadas por Django.

Se muestra la configuración necesaria para realizar lo comentado en este punto del trabajo para un determinado modelo:

Figura 30 *Subclase de ModeloAdmin*

```python
from django.contrib import admin
from .models import *
# Register your models here.

class ActivoAdmin(admin.ModelAdmin):
    list_display = ('id','tic','usuario','condicion','garantia','nombre_equipo')

admin.site.register(Activo, ActivoAdmin)
```

Fuente: Elaboración propia

9. Manual de usuario de la aplicación

En este capítulo se va a mostrar y dar a entender al lector un manual de usuario de la aplicación. Una vez finalizada la lectura del manual, podrá dar uso a la aplicación conociendo exactamente las diferentes opciones expuestas en cada página.

9.1 Ingresar al sistema

Ya registrados los usuarios, se accede al sistema por medio del Login o sistema de autenticación.

Datos: Username, Password

Ruta:localhost:8000/accounts/login/?next=/activos

Figura 31 *Inicio de sesión*

Fuente: Elaboración propia

Como anteriormente se ha mencionado Django proporciona un sistema de autenticación robusto que maneja la creación, almacenamiento y verificación de contraseñas de manera segura. @login_required trabaja en conjunto con este sistema para garantizar la seguridad de la

autenticación, Ayuda a prevenir que usuarios anónimos (no autenticados) accedan a contenido o realicen acciones que deberían estar reservadas para usuarios autenticados.

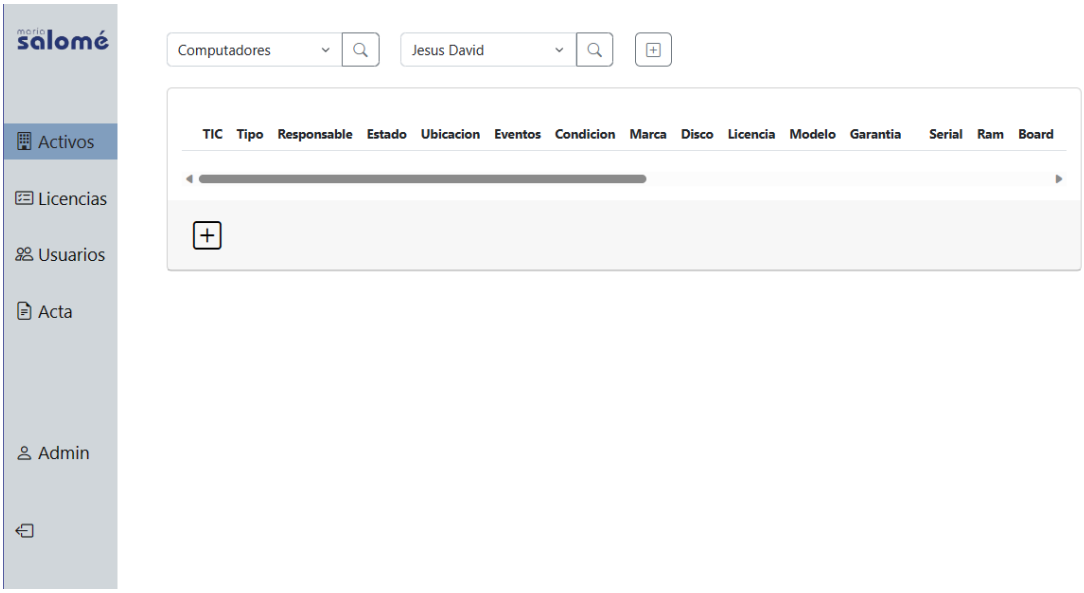
9.2 Homepage

La Homepage es la página principal e inicial a la hora de acceder al aplicativo, donde se maneja todo el dashboard.

Tiene cinco categorías: Activos, Licencias, Usuarios, Acta, Admin, cada una cuenta con un botón que permite su visualización como se muestra en la siguiente imagen:

Ruta: localhost:8000/activos

Figura 32 *Homepage*



Fuente: Elaboración propia

9.3 Activos

9.3.1 Registrar activos

Dentro de la vista activos tendremos dos caminos para ingresar al formulario de registra de activos: Guarda los datos enviados en el formulario si son válidos, contiene las características del activo que se desea crear. En este caso, se incluyen los siguientes campos:

- tic: Código único para el activo.
- Tipos activos: Tipo de activo al que pertenece.
- usuario: Usuario responsable del activo.
- estado: Estado actual del activo.
- ubicación: Ubicación física del activo.
- evento: Descripción de eventos relacionados con el activo.
- condición: Condición actual del activo.
- marca: Marca del activo.
- discos: Tipo de discos asociados al activo.
- licencia: Licencia asociada al activo.
- proveedor: Proveedor del activo.
- modelo: Modelo del activo.
- garantía: Fecha de vencimiento de la garantía.
- serial: Número de serie del activo.
- ram: Capacidad de RAM del activo.
- board: Información sobre la placa base del activo.
- procesador: Tipo de procesador del activo.
- Nombre equipo: Nombre asignado al activo.

- Sistema operativo: Sistema operativo instalado en el activo.
- ip: Dirección IP asignada al activo.
- pulgadas: Tamaño de pantalla del activo.
- Precio compra: Precio de compra del activo.
- Fecha compra: Fecha de compra del activo.
- Numero factura: Número de factura asociado a la compra del activo.

Ruta: localhost:8000/activos/crear.html

Se envía una solicitud de tipo POST. Esta solicitud está destinada a crear un nuevo producto.

Figura 33 *Registro de activos*

Fuente: Elaboración propia

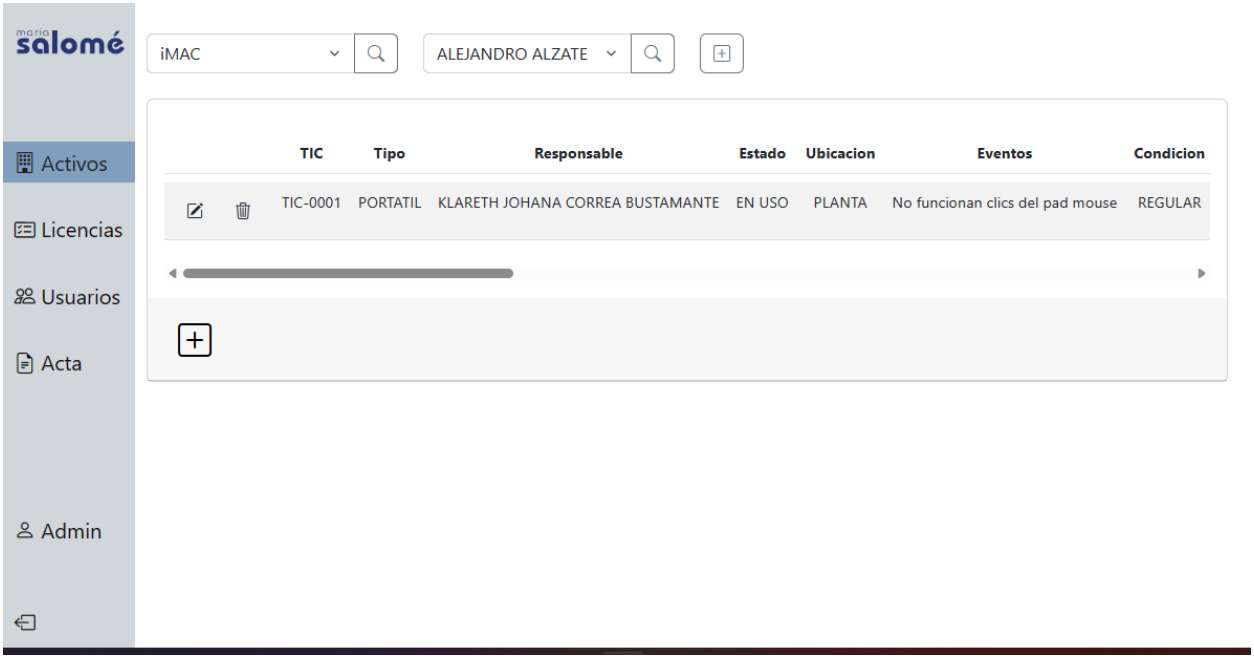
Luego de tener pre registrados algunos datos como Marca, Condición, Tipo, etc.

Se va a facilitar el proceso de registro de cada uno de los activos, luego de terminar el registro veremos los activos de la siguiente forma.

En caso de intentar registrar un activo con el mismo número TIC será denegado, devolviéndonos a la vista de registrar activos.

Ruta localhost:8000/activos.html

Figura 34 *Activos registrados*



Fuente: Elaboración propia

En la parte superior derecha se implementaron dos filtros aplicables a la tabla de activos, donde podremos filtrar por tipo de activo o por usuario, esto facilita la búsqueda de tipos de activos y la búsqueda de activos a nombre de un usuario.

9.3.2 Editar Activos

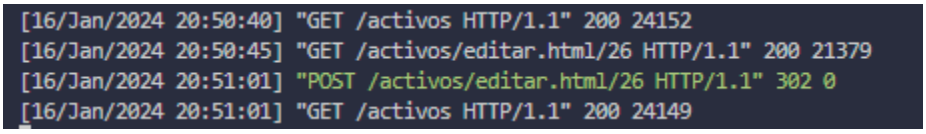
Cada uno de los activos puede ser editado en cualquiera de sus campos, en la parte derecha dentro de la tabla, se encuentra el botón editar donde nos cargara un formulario con los datos del activo cargados, donde podremos ajustarlos según la necesidad.

Argumento request: La solicitud HTTP.

Tipo de petición: se envía la petición POST con el ID del activo a editar.

Ruta: localhost:8000/activos/editar.html

Figura 35 *Edición activos*



Fuente: Elaboración propia

Luego de terminar la edición se da click en el botón de Enviar información, la página cargara de nuevo a la vista principal de activos.

9.3.3 Eliminar activos

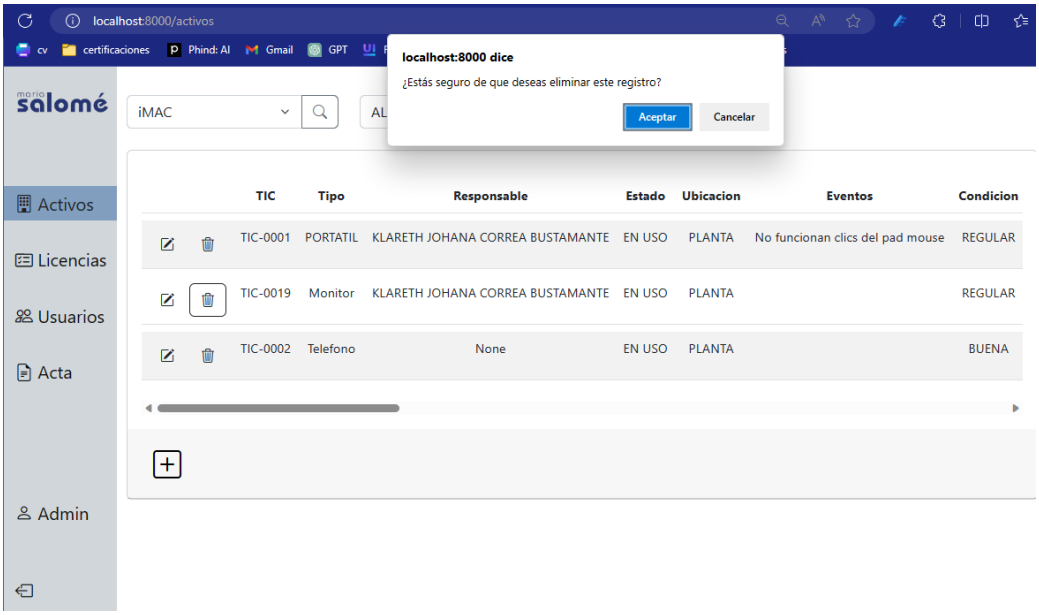
Para el caso de eliminar un activo, en cada fila de activos tendremos un botón para ejecutar y luego de la confirmación de la acción se eliminará el activo de forma definitiva.

Argumento request: La solicitud HTTP.

Tipo de petición: se envía la petición DELETE con el ID del activo a eliminar

Ruta: localhost:8000/activos

Figura 36 *Eliminación de activos*



Fuente: Elaboración propia

9.4 Licencias

9.4.1Registrar Licencia

Dentro de la vista licencias tendremos dos caminos para ingresar al formulario de registro de activos:

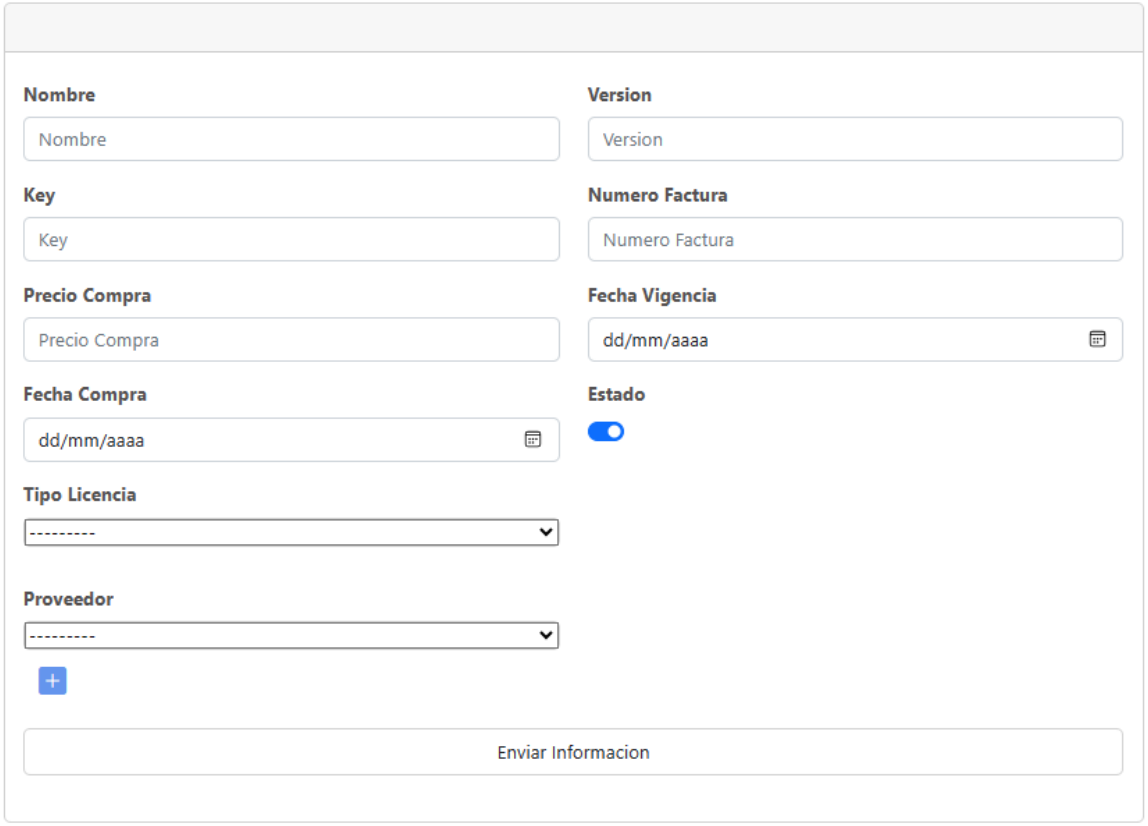
Guarda los datos enviados en el formulario si son válidos, contiene las características de la licencia que se desea crear. En este caso, se incluyen los siguientes campos:

- proveedor: Proveedor de la licencia.
- Tipos licencias: Tipo de licencia al que pertenece.
- nombre: Un nombre identificador de la licencia.
- version: versión de la licencia.
- key: clave única que identifica y activa la licencia.
- Fecha compra: Fecha de compra de la licencia.
- Numero factura: Número de factura asociado a la compra de la licencia.
- Precio compra: Precio de compra de la licencia.
- fecha vigencia: Fecha de vigencia de la licencia.
- estado: Estado actual de la licencia.

Ruta: localhost: 8000/licencias/crear.html

Se envía una solicitud de tipo POST. Esta solicitud está destinada a crear una nueva licencia.

Figura 37 *Registro de licencias*



Fuente: Elaboración propia

El sistema estará verificando que se envié los datos necesarios para registrar el producto, de lo contrario devolverá un mensaje con el tipo de error.

9.4.2 Editar Licencia

Cada una de las licencias puede ser editada en cualquiera de sus campos. en la parte derecha dentro de la tabla, se encuentra el botón editar donde nos cargara un formulario con los datos de la licencia, donde podremos ajustarlos según la necesidad.

Argumento request: La solicitud HTTP.

Tipo de petición: se envía la petición POST con el ID de la licencia a editar.

Ruta: localhost: 8000/licencias/editar.html

Figura 38 *Edición de licencias*

```
[17/Jan/2024 19:38:49] "GET /licencias/editar.html/9 HTTP/1.1" 200 10453
[17/Jan/2024 19:38:54] "POST /licencias/editar.html/9 HTTP/1.1" 302 0
[17/Jan/2024 19:38:54] "GET /licencias HTTP/1.1" 200 15121
```

Fuente: Elaboración propia

9.4.3 Eliminar Licencias

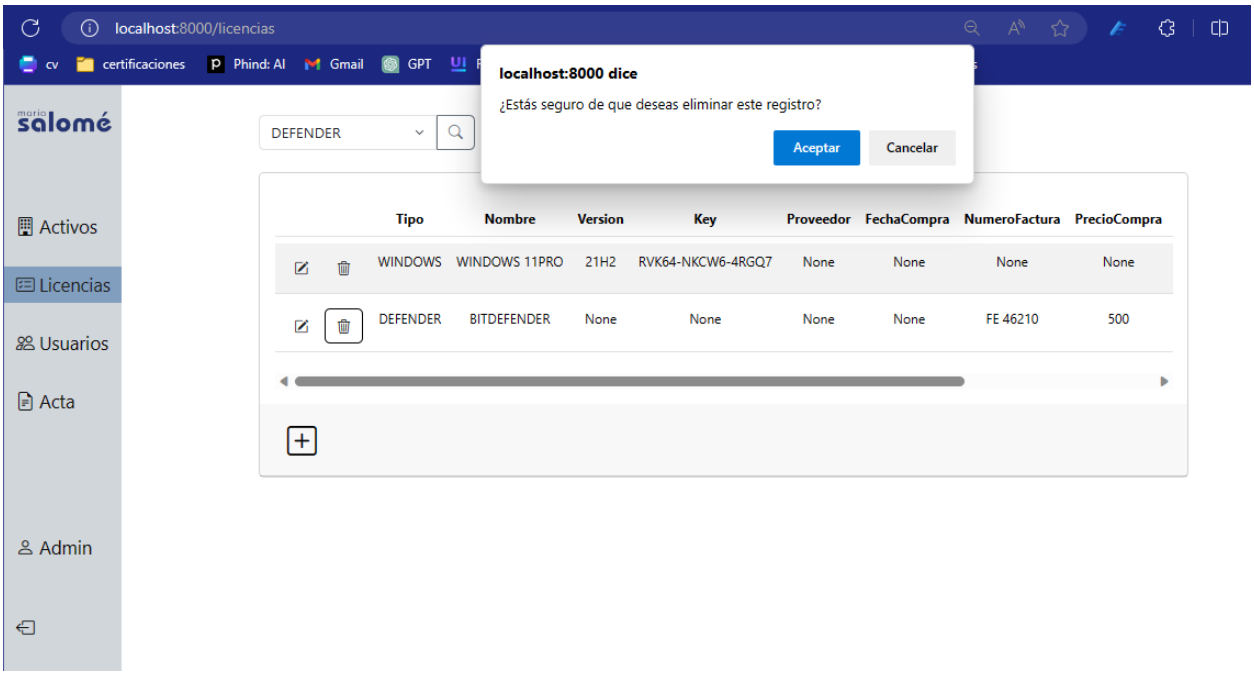
Para el caso de eliminar una licencia, en cada fila de la tabla licencias tendremos un botón para ejecutar la eliminación y luego de la confirmación de la acción se eliminará la licencia de forma definitiva.

Argumento request: La solicitud HTTP.

Tipo de petición: se envía la petición DELETE con el ID de la licencia a eliminar

Ruta: localhost:8000/licencias

Figura 39 *Eliminación de licencias*



Fuente: Elaboración propia

9.5 Usuarios

9.5.1 Registrar Usuarios

Dentro de la vista Usuarios tendremos dos caminos para ingresar al formulario de registro de usuarios, que en este caso son los empleados a los que se les van a asignar como responsables de los activos entregados para sus funciones:

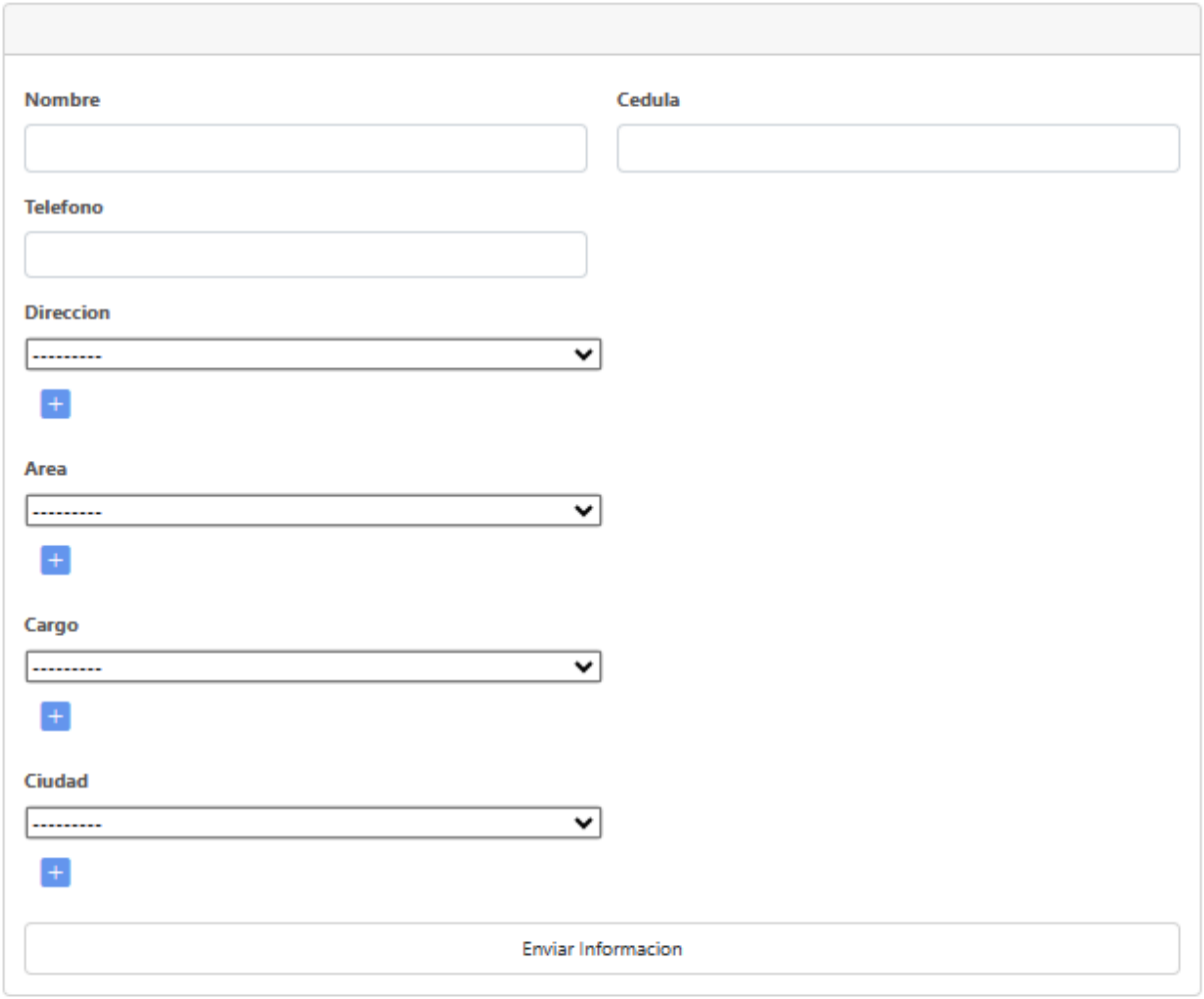
Guarda los datos enviados en el formulario si son válidos, contiene las características del usuario que se desea crear. En este caso, se incluyen los siguientes campos:

- Nombre: Un nombre identificador de la licencia.
- Cedula: Número de identificación.
- Teléfono: Número de teléfono.
- Dirección: Nombre de la dirección a la que pertenece.
- Área: Nombre del área al que pertenece.
- Cargo: Nombre del cargo que ejerce.
- Ciudad: Nombre de la ciudad donde esta erradicado.

Ruta: localhost: 8000/usuarios/crear.html

Se envía una solicitud de tipo POST. Esta solicitud está destinada a crear un nuevo Usuario.

Figura 40 *Registro de usuarios*



Fuente: Elaboración propia

9.5.2 Editar Usuarios

Cada uno de los usuarios puede ser editado en cualquiera de sus campos. en la parte derecha dentro de la tabla, se encuentra el botón editar donde nos cargara un formulario con los datos del usuario a editar, podremos ajustarlos según la necesidad.

Argumento request: La solicitud HTTP.

Tipo de petición: se envía la petición POST con el ID del usuario a editar.

Ruta: localhost:8000/usuarios/editar.html

Figura 41 *Edición de usuarios*

```
[17/Jan/2024 20:14:47] "GET /usuarios/editar.html/7 HTTP/1.1" 200 11278
[17/Jan/2024 20:14:52] "POST /usuarios/editar.html/7 HTTP/1.1" 302 0
[17/Jan/2024 20:14:52] "GET /usuarios HTTP/1.1" 200 43464
```

Fuente: Elaboración propia

9.5.3 Eliminar Usuarios

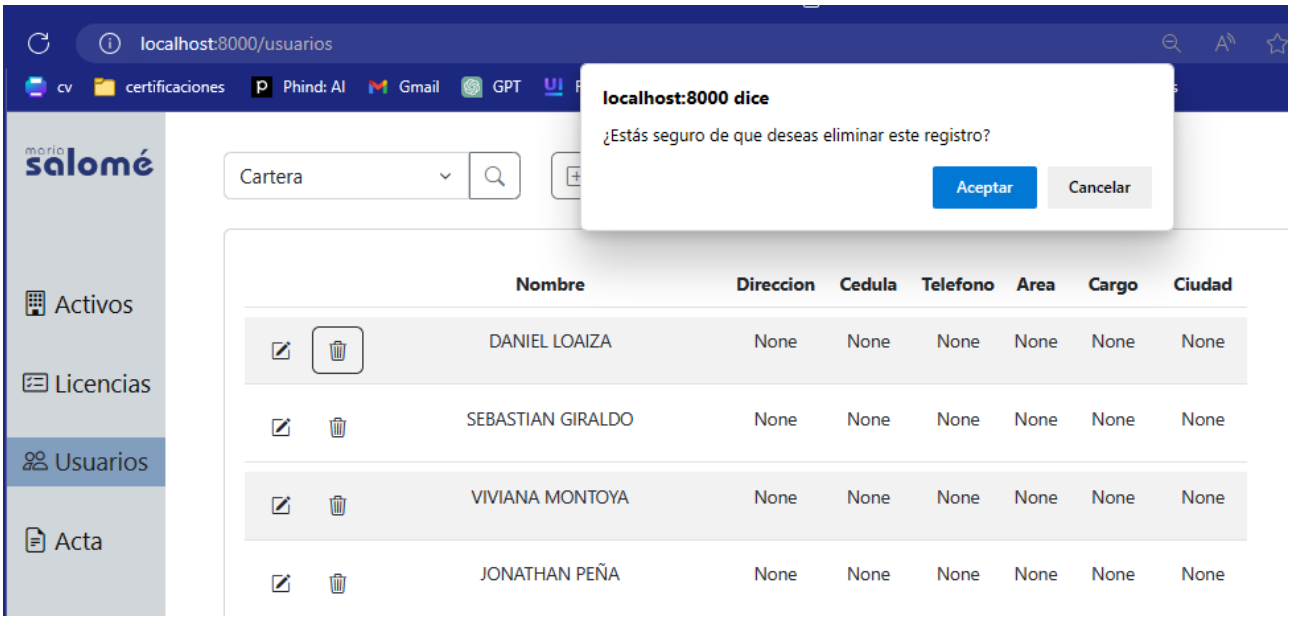
Para el caso de eliminar un usuario, en cada fila de la tabla usuarios tendremos un botón para ejecutar la eliminación y luego de la confirmación de la acción se eliminará el usuario de forma definitiva.

Argumento request: La solicitud HTTP.

Tipo de petición: se envía la petición DELETE con el ID del usuario a eliminar

Ruta: localhost:8000/usuarios

Figura 42 *Eliminación de usuarios*

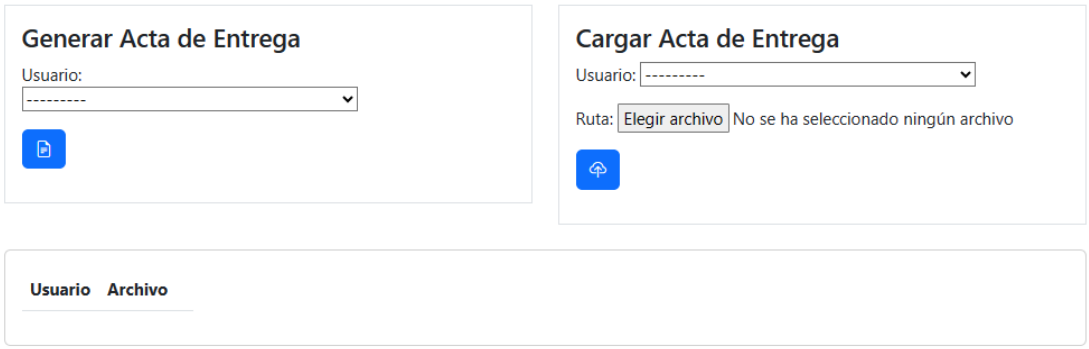


Fuente: Elaboración propia

9.6 Actas

Vista Actas, en la parte inferior tendremos listadas las actas cargadas con su usuario relacionado.

Figura 43 *Modulo de Actas*



Fuente: Elaboración propia

9.6.1 Generar Actas de Entrega

Para generar un acta de entrega tendremos que tener los usuarios relacionados con los activos de los cuales sean responsables, para esto seguiremos los siguientes pasos:

9.6.1.1Asignar un Activo a un Usuario:

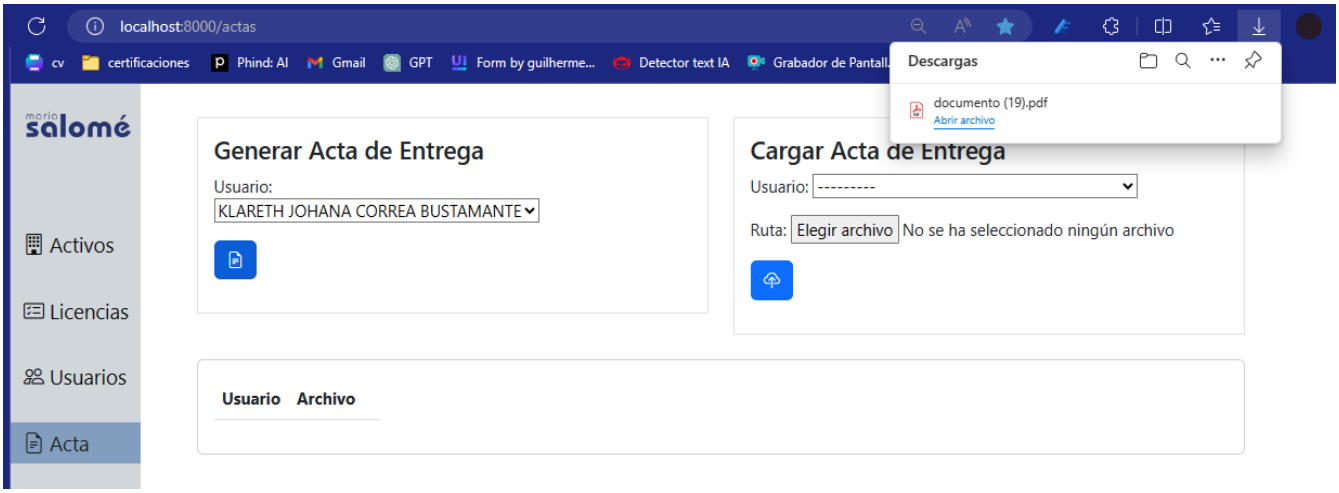
Cuando se está creando un activo encontramos el campo Usuario dentro del formulario, donde nos mostrara una lista de usuarios registrados a los cuales podremos asignarles activos según la necesidad de cada uno, en la vista de Activos utilizando el filtro de usuarios podremos validar que el usuario tenga todos los activos necesarios. Luego de esto iremos a la vista Actas y en el módulo Generar Actas de Entrega seleccionaremos el Usuario solicitado, y daremos en el botón con icono de archivo, esto va a generar un archivo PDF que incluye información detallada sobre los activos asignados a ese usuario.

Se configura la respuesta HTTP para descargar el PDF creado.

Tipo de petición: se envía la petición POST con el ID del usuario a generar el acta.

Ruta: localhost:8000/actas

Figura 44 *Generación de actas*



Fuente: Elaboración propia

Librerías Utilizadas

ReportLab, Esta librería se utiliza para la creación y generación de archivos PDF. Proporciona un conjunto de herramientas para la creación de documentos PDF de manera programática.

io.BytesIO, La clase BytesIO se utiliza para crear un flujo de bytes en memoria, que actúa como un búfer para almacenar el contenido del PDF antes de guardarlo o enviarlo como respuesta HTTP.

Datetime, Se utiliza para obtener la fecha actual de creación del PDF.

9.6.2 Generación del PDF

Se utiliza **ReportLab** para crear un documento PDF.

Se establecen márgenes, se añade un encabezado con la fecha actual y la imagen del encabezado institucional.

Se agregan detalles sobre el destinatario (nombre y cédula).

Se añade un cuerpo de texto que indica la entrega de activos y la responsabilidad del destinatario.

Se incluye una lista de activos asignados al destinatario, con detalles como el tipo de activo, marca, número de serie, etc.

Se agregan observaciones sobre el cuidado y responsabilidades del destinatario respecto a los activos.

Generación de Detalles Adicionales, Se añaden detalles adicionales en negrita en una sección especial del documento.

Firma y Cargo, Se incluye un espacio para la firma y el cargo del emisor del documento.

Figura 45 *Ejemplo de acta*

Laboratorio María Salomé	ENTREGA DE EQUIPOS DE CÓMPUTO Y/O ACCESORIOS	CÓDIGO	F-DE-TI-04
		EMISIÓN	07-Nov-2023
		VERSIÓN	01

Medellín, 17 de enero de 2024

Señor(a)

KLARETH JOHANA CORREA BUSTAMANTE C.C 1234567891

A continuación le estamos haciendo entrega de los siguientes activos, los cuales estaran bajo su completa responsabilidad y custodia.

DETALLES

- PORTATIL LENOVO S/N PC-00CW3A TIC-0001 Eventos: No funcionan clics del pad mouse

OBSERVACIONES:

- Vigilar con frecuencia que no le falte el activo o lo tenga en mal estado, ya que usted es la única persona responsable.
- Si observa que le quitan o le cambian su activo, informe de inmediato a su jefe para que se hagan los cargos y descargos correspondientes y la anotación del estado del mismo.
- Si considera que su activo requiere algún mantenimiento para colocarlo en mejores condiciones, informe a su jefe.
- En el momento de su desvinculación de la empresa es indespensable la presentación de un paz y salvo expedido por su jefe, quien temporalmente asumirá la responsabilidad del mismo hasta entregarlos al nuevo empleado.
- Si al momento de la devolución del activo este presenta deterioro por manipulación como partes rotas, quebradas u otro tipo de avería, el empleado debe asumir el valor de los repuestos o reparación
- Exija una copia de la carta donde lo responsabilizan de su activo, para que este al tanto su contenido.
- Fuera del Laboratorio Maria Salome, lo que suceda con el equipo es completamente responsabilidad. En caso de daño o perdida usted deberá asumir el costo de reparacion o reposición.

Diana Marcela Quiroz
Coordinadora Administrativa

KLARETH JOHANA CORREA BUSTAMANTE
Asistente - Cartera

Fuente: Elaboración propia

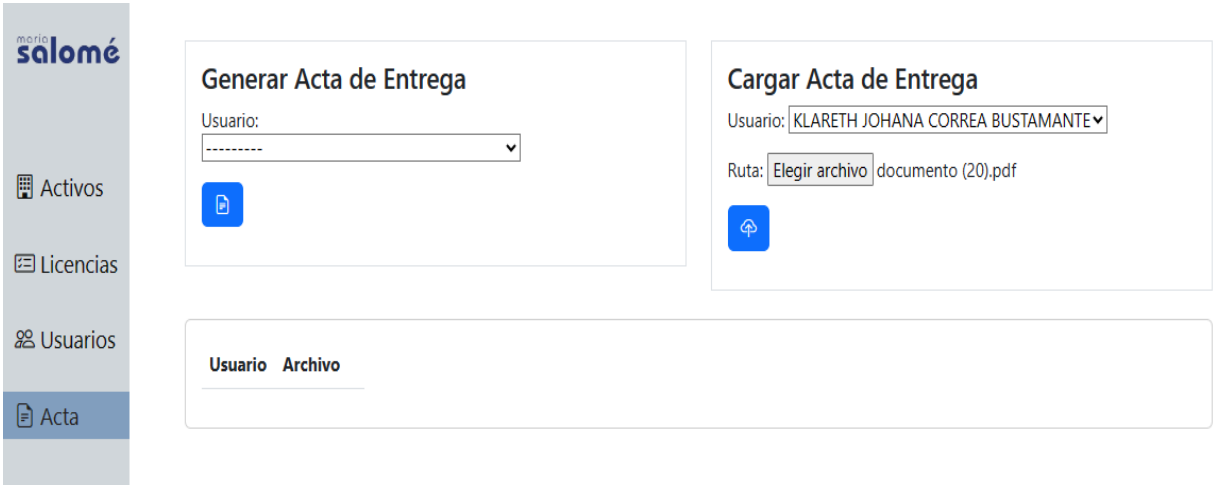
9.6.2 Cargar Acta de Entrega

Para el proceso de Cargar un acta se hace independiente ya que se necesitan firmadas por el usuario relacionado.

Luego de tener el acta firmada, en el apartado de Cargar Acta de Entrega encontraremos el mismo listado de usuarios donde se seleccionará el usuario relacionado con el acta que vamos a cargar y mediante el botón de cargar escogeremos el archivo en el administrador de archivos que se abrirá y le daremos en aceptar.

El programa verifica la solicitud POST y si el formulario es válido, guardara el formulario en la base de datos asociando el usuario seleccionado con el acta.

Figura 46 *Cargar acta*



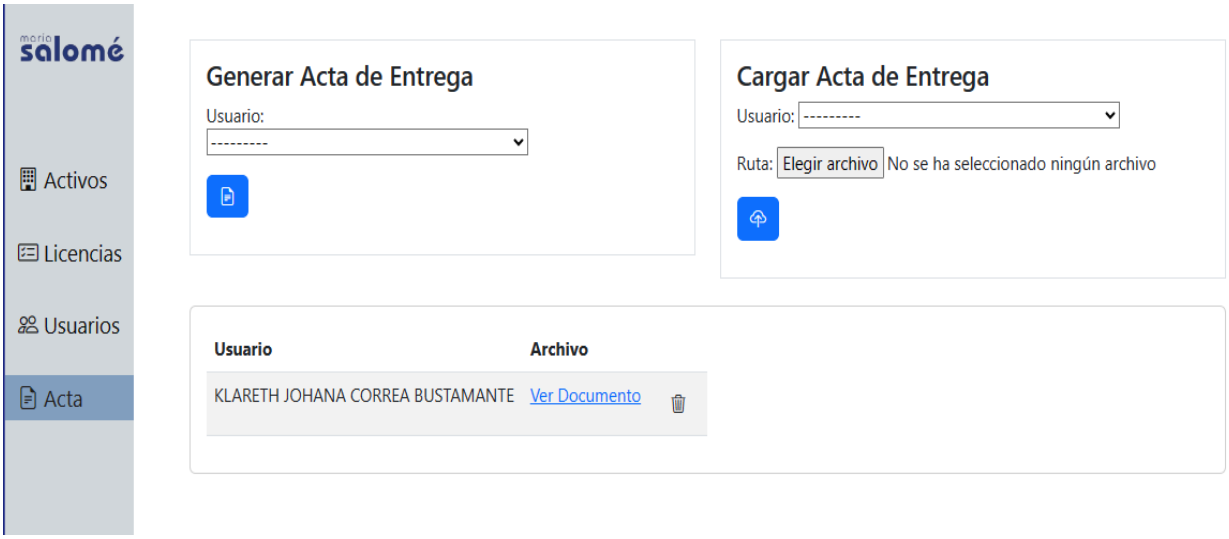
Fuente: Elaboración propia

Se configura la respuesta HTTP para guardar el PDF cargado.

Tipo de petición: se envía la petición POST con el ID del usuario, y el documento seleccionado.

Ruta: localhost:8000/actas

Figura 47 *Actas cargadas*



Fuente: Elaboración propia

En la tabla de actas podremos tener un fácil acceso a descargar las actas de cada uno de los usuarios mediante el link “Ver Documento”.

9.6.3 Eliminar Actas

Se buscar el registro de acta en la base de datos utilizando el ID obtenido. Si se encuentra, se elimina y se responde con un JSON indicando que la eliminación fue exitosa.

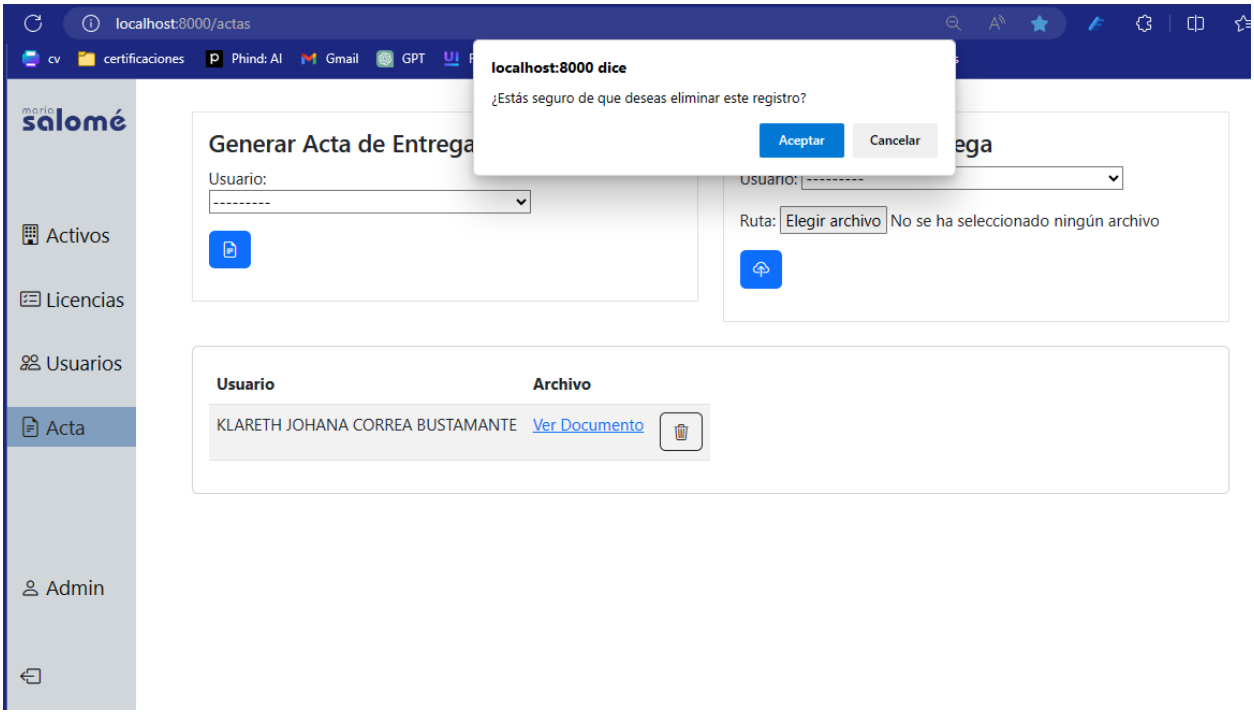
En caso de que el registro no sea encontrado, se responde con un JSON indicando que el registro no fue encontrado.

Se configura la respuesta HTTP para Eliminar el PDF cargado.

Tipo de petición: se envía la petición DELETE con el ID del usuario, y el ID del documento seleccionado.

Ruta: localhost:8000/actas

Figura 48 *Eliminación de actas*



Fuente: Elaboración propia

De esta forma tendremos control de las actas que ya no sean necesarias almacenar.

Conclusiones

Durante la ejecución de este proyecto, se ha alcanzado el desarrollo exitoso de una aplicación web mediante el uso de Django, un robusto framework respaldado por el motor de Python, y aprovechando PostgreSQL como un eficiente gestor de base de datos que asegura escalabilidad y funcionalidad. Esta aplicación ha sido diseñada para gestionar activos empresariales y administrar las relaciones entre los activos asignados y los usuarios responsables de su manejo. El acceso a la aplicación se realiza a través de un sistema de inicio de sesión seguro, respaldado por una sólida encriptación de claves.

La elección estratégica de tecnologías clave, como Django, PostgreSQL y Bootstrap, ha permitido la creación de una plataforma intuitiva y de fácil uso. La adopción de la metodología ágil Scrum ha facilitado la organización y planificación del desarrollo a través de sprints, proporcionando una visión clara de los objetivos y asegurando la entrega constante de mejoras funcionales en la aplicación. Además, el énfasis en la colaboración y la comunicación efectiva dentro del equipo de desarrollo ha sido esencial para garantizar el éxito del proyecto.

En cuanto a la arquitectura de la aplicación, la elección de Django como framework ha demostrado ser sólida, brindando una base estructurada para la creación de rutas, middleware y el manejo eficiente de peticiones HTTP. El ORM proporcionado por Django ha facilitado la modelación de entidades de manera sencilla y la realización eficiente de consultas a la base de datos. La implementación del sistema de gestión de usuarios con inicio de sesión ha sido un componente crucial para garantizar la seguridad y privacidad de la información en la aplicación. A través de un adecuado proceso de autenticación y autorización, se ha logrado proteger los recursos, asegurando que solo los usuarios autorizados tengan acceso a funcionalidades específicas.

Este enfoque ha sido esencial para crear una aplicación segura, confiable y escalable en el manejo.

Recomendaciones

A partir de la valiosa experiencia obtenida durante la ejecución de este proyecto, se derivan recomendaciones fundamentales para futuras iniciativas similares:

Mantener la metodología Scrum

Es esencial continuar empleando Scrum o alguna otra metodología ágil. La estructura organizativa por sprints y la colaboración eficaz dentro del equipo han demostrado ser pilares fundamentales para el éxito del proyecto y la entrega incremental de valor.

Pruebas exhaustivas

Garantizar la implementación de pruebas unitarias y de integración a lo largo de todo el ciclo de desarrollo. Las pruebas desempeñan un papel crucial al validar el correcto funcionamiento de la aplicación y al detectar posibles errores de manera temprana, facilitando así su corrección y evitando complicaciones en etapas posteriores.

Continuar la mejora del rendimiento

Aunque se ha logrado una aplicación funcional y eficiente, es imperativo seguir optimizando el código y las consultas a la base de datos. Este enfoque garantiza una experiencia de usuario fluida y eficaz, especialmente en entornos de alta concurrencia.

Reforzar la seguridad

La seguridad de los datos y la protección de la información de los usuarios son aspectos críticos. Es esencial seguir las mejores prácticas de seguridad, como el almacenamiento seguro de contraseñas y el uso de tokens de autenticación, para prevenir vulnerabilidades y ataques.

Documentación completa y actualizada

Mantener una documentación detallada y actualizada de la aplicación, que abarque la arquitectura, el diseño de la API, los modelos de datos y las decisiones de diseño. Esto facilitará el

mantenimiento, las futuras expansiones del proyecto y la integración eficiente de nuevos miembros al equipo.

Explorar tecnologías complementarias

Investigar y considerar el uso de tecnologías complementarias que puedan mejorar la funcionalidad y la experiencia del usuario.

Monitoreo y análisis

Implementar un sistema de monitoreo para la aplicación en producción, permitiendo identificar problemas de rendimiento y uso. Tomar acciones proactivas para mejorar la experiencia del usuario y realizar análisis de datos para obtener información relevante sobre el comportamiento de los usuarios y el rendimiento de la aplicación.

Retroalimentación y mejora continua

Fomentar la retroalimentación del cliente y de los usuarios finales para comprender sus necesidades y expectativas. Utilizar esta información para realizar mejoras continuas en la aplicación y agregar nuevas funcionalidades que añadan valor a los usuarios.

En síntesis, la implementación de estas recomendaciones no solo contribuirá al crecimiento constante del equipo de desarrollo, sino que también asegurará la entrega de soluciones de alta calidad que satisfagan las necesidades cambiantes de nuestros clientes y usuarios.

Bibliografía

Acosta, L. &. (2021). *scielo.* Obtenido de http://scielo.sld.cu/pdf/rus/v14n2/2218-3620-rus-14-02-237.pdf

Alcántara, V. (2021). *freecodecamp*. Obtenido de https://www.freecodecamp.org/espanol/news/para-que-se-utiliza-django-de-python-5-razones-claves-por-las-que-uso-el-framework-django-para-proyectos/

Alvarez, O. l. (2012). *repositorioinstitucional.ufpso.edu.co.* Obtenido de https://repositorioinstitucional.ufpso.edu.co/xmlui/bitstream/handle/20.500.14167/1230/Cuerpo%20del%20trabajo%20Ligdy%20MIlena%20Orozco%20Alvarez%20190607_removed.pdf?sequence=1&isAllowed=y

Amazon. (2023). *Aws.Amazon.* Obtenido de https://aws.amazon.com/es/what-is/web-application/

Andres, D. R. (2021). *repository.ucc.edu.co.* Obtenido de https://repository.ucc.edu.co/entities/publication/e18cd7a5-6bc8-452b-a763-1e4ec516cea1

Browning, S. d. (2023). *onlinelibrary.wiley*. Obtenido de https://onlinelibrary.wiley.com/journal/18731317

carrion, S. (2015). Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django. *Scielo*.

Carrion, S. (2018). Analyzing best practices on Web development frameworks: The lift approach. *ScienceDirect*.

Clavijo, L. P. (2018). *IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN ERP (Enterprise Resource Planning) QUE PERMITA LOS PROCESOS DE FACTURACIÓN,*

INVENTARIOS, CLIENTES, PROVEEDORES Y CARTERA EN EL ALMACÉN FRENAUTOS 2001 EN AGUACHICA-CESAR.

Dávila Bernal, N. A. (2020). *repository.unab.edu.co*. Obtenido de https://repository.unab.edu.co/handle/20.500.12749/12050

desarrollodesoftware. (2020). *desarrollodesoftware*. Obtenido de https://desarrollodesoftware.com.co/desarrollo-de-software/desarrollo-agil-de-software/

developer.mozilla.org. (2022). Obtenido de https://developer.mozilla.org/es/docs/Learn/Server-side/Django

Díaz Marín, D. J. (2016). *repositorio.utp.edu.co*. Obtenido de https://repositorio.utp.edu.co/server/api/core/bitstreams/d7095695-d26a-4ac1-972c-15fb3e5a7482/content

django. (2005). *www.djangoproject.com.* Obtenido de https://www.djangoproject.com/

Dwivedi, D. Y. (2002). *ScienceDirect*. Obtenido de https://www.sciencedirect.com/journal/international-journal-of-information-management

Ebac. (2023). *Ebac*. Obtenido de https://ebac.mx/blog/frameworks#:~:text=El%20objetivo%20principal%20de%20un%20framework%20es%20ahorrar,errores%20y%20a%20crear%20un%20c%C3%B3digo%20m%C3%A1s%20limpio.

Ferry, A. (2014). *openaccess.* Obtenido de https://openaccess.uoc.edu/bitstream/10609/141486/1/Tecnologias%20y%20herramientas%20para%20el%20desarrollo%20web_Modulo1_Introduccion%20al%20frontend%20y%20backend.pdf

Foundation, D. S. (2005). *EcuRed*. Obtenido de https://www.ecured.cu/Django#:~:text=La%20meta%20fundamental%20de%20Django%20es%20facilitar%20la,principio%20de%20DRY%20%28del%20ingl%C3%A9s%20Don%27t%20Repeat%20Yourself%29.

Iasa Global. (2021). *iasaglobal*. Obtenido de https://iasaglobal.org/

Icariatechnology. (2023). *icariatechnology.* Obtenido de https://icariatechnology.com/la-calidad-del-software-y-sus-estandares-mas-importantes/

instituto tecnologico metropolitano. (2019). *redaly*. Recuperado el 22 de 08 de 2023, de https://www.redalyc.org/journal/3442/344263272003/html/

ionos. (2020). *ionos.es*. Obtenido de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/common-gateway-interface/

ISACA. (2022). *Information System Audit and Control Association*. Obtenido de https://www.isaca.org/

Jesus. (2023). *Explorando Django: Tu Introducción al Desarrollo Web con Python*. Obtenido de Dongee: https://www.dongee.com/tutoriales/que-es-django-en-programacion/

Kumari Kusuma, K. R. (2017). Estimación robusta de la dirección de llegada mediante métodos subespaciales. *Revista Internacional de Aplicaciones Informáticas*, 159.

Ley Estatutaria 1581. (2012). *(DO. 48.587).*

López, R. (2015). *Universidad Politecnica de Valencia.* Obtenido de : http://hdl.handle.net/10251/54577

Lübeck, Alemania. (2021). Open Journal of Cloud Computing.

Machinery. (2022). *ACM*. Obtenido de https://www.acm.org/)

Martinez, S. R. (2020). *repository.unad.edu.co.*

Oracle. (2024). *Oracle*. Obtenido de https://www.oracle.com/es/database/what-is-data-management/

Palomino Heider, J. W. (2018). *IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN EN ENTORNO WEB PARA EL MANEJO DE INVENTARIO, FACTURACIÓN Y PROCESOS CON PROVEEDORES DE LA EMPRESA DENTALES DE ORIENTE EN AGUACHICA, CESAR.*

Pereira, A. D. (2019). *repository.udistrital.edu.co.* Obtenido de https://repository.udistrital.edu.co/bitstream/handle/11349/24811/DiazPereiraAlejandro2020.pdf?sequence=3

Proyectos ágiles. (2022). *Qué es SCRUM*. Obtenido de https://proyectosagiles.org/que-es-scrum/

Pythontutorial. (2023). *Pythontutorial*. Obtenido de https://www.pythontutorial.net/django-tutorial/django-orm/

Raebum, A. (2022). *asana*. Obtenido de https://asana.com/es/resources/extreme-programming-xp

Rico, M. A. (2006). *Udlap*. Obtenido de http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo2.pdf

Robayo, D. J. (2023). *repositorio.* Obtenido de https://repositorio.uide.edu.ec/bitstream/37000/6348/1/2299-Texto%20del%20art%c3%adculo-12602-1-10-20230909.pdf

Sánchez, A. C. (2021). *Scielo.* Obtenido de https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-50062021000500085

Schwaber, K. (2020). Obtenido de https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf

Schwaber, K. (2024). *Scrum.org*. Obtenido de https://www.scrum.org/learning-series/what-is-scrum/

Sommerville. (2005). *Ingeniería del software.* Pearson Educación.

Sommerville, I. (2011). *Ingeniería de Software.* México: Pearson Educación.

superintendencia de industria y comercio. (2019). *superintendencia de industria y comercio* . Recuperado el 22 de 8 de 2023, de https://sigi.sic.gov.co/SIGI/files/mod_documentos/documentos/GS01-P15/versiones/GS01-P15%20PROCEDIMIENTO%20GESTION%20DE%20INFRAESTRUCTURA_copia_no_controlada.pdf

Universidad de Alcalá. (2022). *Arquitectura y Diseño de Sistemas Web y C/S.* Obtenido de https://www1.uah.es/estudios/asignaturas/descarga_fichero.asp?CodAsig=780041&CodPlan=G581&Anno=2022-23

Vergara, R. C. (2017). *Biblioteca Digital Universidad de Alcalá.* Obtenido de https://ebuah.uah.es/dspace/handle/10017/32018

Walls. (2005). *Manning Publication*.

Wikipedia. (2017). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto

Zhou, Z.-H. (s.f.). Fronteras de las Ciencias de la Computación. *Springer*. Obtenido de https://www.springer.com/journal/11704