UNIVERSIDAD Ð CIENFUEGOS
INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA

Universidad de Cienfuegos

"Carlos Rafael Rodríguez"

Facultad de Ingeniería

Título:

Repositorio Digital de Plantas Medicinales

en la Universidad de Cienfuegos.

Trabajo de diploma para optar por el título de

Ingeniera Informática

Autora:

Elizabeth Medina Muñoz

Tutor:

MSc. Richard Darian Sánchez Rivero

MSc. Caridad Josefa Rivero Casanova

Cienfuegos, Cuba

Curso: 2024

AGRADECIMIENTOS

Hoy es un día especial para mí y quiero agradecerles a todas las personas que me acompañaron en estos 4 años de mi vida.

Gracias a mi mami por estar siempre en los momentos difíciles, por tu apoyo incondicional en todo momento, mi amiga, gracias por confiar en mí, gracias por todo tu amor.

Gracias a mi papi por su apoyo, por enseñarme a nunca rendirme, gracias por estar cuando te necesite, gracias por todo tu amor.

Gracias a ambos por existir, este es mi regalo para ustedes, los AMO.

Gracias a mi tía Mary y a mi Abu Fredis por todo su apoyo siempre.

Gracias a mis hermanos por todo su apoyo.

A mi novio gracias por quererme, gracias por la paciencia y la fuerza que me has dado en esta etapa final, te amo.

Gracias a mi tutor Richard por todo el tiempo dedicado, gracias por confiar en mí.

Gracias a mis compañeros de estudio quienes me ayudaron y apoyaron en algunos momentos, con los que compartí una bella etapa de mi vida.

Gracias a toda mi familia y amigos por estar presente toda esta etapa, gracias por su apoyo en todo momento, los quiero a todos.

DEDICATORIA

A mi madre y a mi padre que son mi inspiración, fortaleza y lo más importante de mi vida.

RESUMEN

En la actualidad, la cantidad de información disponible representa un desafío común debido a su gran volumen. La Facultad de Agronomía de la Universidad de Cienfuegos enfrenta dificultades en la gestión de esta información, ya que carece de un software adecuado que facilite la recopilación y organización de datos, lo que dificulta la obtención de la información necesaria. El objetivo de este proyecto es crear un repositorio digital sobre plantas medicinales que permita almacenar, gestionar y acceder a esta información de manera efectiva.

Para llevar a cabo esta tarea, se ha utilizado Django, un framework web avanzado que promueve el desarrollo rápido y un diseño limpio y práctico. Desarrollado por expertos en programación, Django se encarga de muchas de las complejidades del desarrollo web, permitiéndole concentrarse en la creación de su aplicación sin tener que empezar desde cero.

Palabras claves: Repositorio, información, almacenamiento, gestión y aplicación web.

Abstract

At present, the amount of information available represents a common challenge due to its sheer volume. The Faculty of Agronomy of the University of Cienfuegos faces difficulties in the management of this information, since it lacks adequate software that facilitates the collection and organization of data, which makes it difficult to obtain the necessary information. The aim of this project is to create a digital repository on medicinal plants that allows this information to be stored, managed and accessed effectively.

To carry out this task, Django has been used, an advanced web framework that promotes rapid development and a clean and practical design. Developed by programming experts, Django takes care of many of the complexities of web development, allowing you to focus on building your app without having to start from scratch.

Keywords: Repository, information, storage, management and web application.

Índice

Índice de tablas

Índice de figuras

Introducción

Internet, surgido en la década de 1970, experimentó un crecimiento gradual hasta principios de los años 90, momento en el que se liberalizó su acceso, lo que provocó un aumento significativo en el número de usuarios y en la diversidad y volumen de información intercambiada. Posteriormente, se produjo un avance aún más significativo con la llegada de la World Wide Web. Los primeros usuarios de Internet, principalmente académicos e investigadores de universidades y otras instituciones de investigación, rápidamente comenzaron a explorar las posibilidades del nuevo entorno para facilitar la comunicación de ideas y datos, aprovechando herramientas como el correo electrónico, FTP (para transferencia de archivos) y telnet (para conexión a computadoras remotas). [1]

En la actualidad, estamos inmersos en un mundo dinámico y en constante evolución, impulsado principalmente por los avances tecnológicos, conocidos como Tecnologías de la Información y las Comunicaciones (TIC). Estas tecnologías han permeado todos los ámbitos de la sociedad, provocando transformaciones significativas en la manera en que se gestionan, organizan y acceden a la información, tanto en su forma impresa como digital. El desarrollo de las TIC ha dado lugar al surgimiento del movimiento de Acceso Abierto (AA), el cual brinda la oportunidad de acceder a documentos científicos de manera gratuita y sin restricciones. Este movimiento está en constante expansión, gracias al crecimiento de iniciativas y proyectos que se adhieren al principio fundamental del Acceso Abierto.[2]

En la última década, el número de repositorios institucionales ha experimentado un aumento significativo debido a la rápida expansión de la información digital. Esta tendencia se refleja en el Ranking Web de Repositorios del Mundo, donde se han registrado 2.283 repositorios institucionales a nivel global. España, por ejemplo, cuenta con 72 repositorios institucionales, lo que demuestra el notable crecimiento en este ámbito. A principios del siglo XXI, surgió una tendencia global a establecer repositorios digitales institucionales en universidades. Estos repositorios tenían como objetivo centralizar y brindar acceso gratuito en línea a las investigaciones y trabajos académicos realizados por los estudiantes de cada institución.[3]

La idea convencional de las bibliotecas tradicionales se basa en ofrecer materiales físicos como libros, revistas, tesis y artículos a través de servicios de préstamo y consulta. Estas prácticas a menudo restringían el acceso a los documentos debido a la falta de copias disponibles o a la presencia de materiales desactualizados. Sin embargo, con el avance de la tecnología informática, la expansión de Internet y la disminución de los costos asociados con la adquisición de recursos y servicios, se ha impulsado en las últimas dos décadas el desarrollo y la implementación de Bibliotecas Digitales (BD). En otras

palabras, se ha comenzado a automatizar las bibliotecas tradicionales para adaptarse a un entorno digital en constante crecimiento y evolución.[3]

El movimiento de acceso abierto busca la difusión gratuita y sin restricciones de la producción científica, es decir, los resultados de la investigación. Esta iniciativa se está expandiendo al ámbito educativo, incluyendo los recursos educativos abiertos (OER), cuyo valor ha aumentado con la implementación del Espacio Europeo de Educación Superior (EEES) y un nuevo enfoque pedagógico que fomenta el uso compartido de recursos entre docentes y estudiantes. Los OERs surgieron en 2001 con el programa OpenCourseWare (OCW) del Massachusetts Institute of Technology (MIT) y desde entonces han ganado interés. En estos once años, los recursos educativos abiertos han evolucionado en dos etapas: una inicial centrada en el acceso a los contenidos y otra actual preocupada por su integración en las prácticas educativas. [1]

En 2006, un estudio realizado por la Asociación de Bibliotecas de Investigación (ARL) reveló que el 78% de las 87 bibliotecas universitarias encuestadas ya tenían en funcionamiento un repositorio digital o planeaban instalarlo entre 2006 y 2007. Se observó un aumento significativo en el número de bibliotecas universitarias en América del Norte que implementaron software para repositorios a partir de 2004, como DSpace, desarrollado por el MIT y Hewlett-Packard, y Eprints, creado por la Universidad de Southampton. [1]

En la actualidad, las universidades recopilan una gran cantidad de datos e información que son fundamentales para su funcionamiento y para la toma de decisiones estratégicas. Esta información abarca desde datos académicos y administrativos hasta investigaciones y proyectos llevados a cabo por la institución y sus miembros. En este contexto, la importancia de contar con un sistema eficiente para el almacenamiento y gestión de estos datos se ha vuelto cada vez más destacada. En este proyecto se describe la creación de un repositorio digital de plantas medicinales para que la universidad disponga de todos sus datos de manera centralizada.

Situación problemática:

Se ha identificado que los archivos existentes se encuentran desorganizados y dispersos. Esta falta de estructura y orden dificulta la gestión eficiente de la información, generando problemas en la búsqueda, accesibilidad y mantenimiento de los archivos.

Problema a resolver:
¿Cómo gestionar los documentos académicos relacionados con las plantas medicinales en la Universidad de Cienfuegos?

Objeto de estudio:
La gestión de documentos académicos de la Universidad de Cienfuegos.

Campo de acción:
La gestión de documentos académicos, relacionados con las plantas medicinales en la Universidad de Cienfuegos.

Objetivo general:
Desarrollar un repositorio digital sobre las plantas medicinales en la Universidad de Cienfuegos, que permita el almacenamiento, gestión y acceso de la información.

Objetivos específicos:

- Analizar los referentes teórico-metodológicos del sistema de gestión universitaria, asociados a la gestión de documentos académicos.
- Diseñar la arquitectura del repositorio digital de plantas medicinales para la Universidad de Cienfuegos.
- Implementar el sistema de repositorio digital de plantas medicinales para la Universidad de Cienfuegos.
- Evaluar el desempeño del repositorio digital de plantas medicinales, a través de pruebas funcionales y casos de pruebas.

Tareas de investigación:

- Entrevistas al personal revisión de la literatura científica sobre las plantas medicinales, incluyendo su composición, formación y funciones.
- Investigación sobre las tecnologías y herramientas disponibles para la creación de repositorios digitales.
- Modelación de datos necesarios para almacenar la información sobre las plantas medicinales, como el nombre, la ubicación geográfica.
- Creación de las vistas y plantillas necesarias para permitir la creación, edición, eliminación y visualización de la información sobre las plantas medicinales.
- Implementación de la funcionalidad de búsqueda para permitir a los usuarios filtrar la información sobre las plantas medicinales.
- Realización de pruebas de validación para verificar el correcto funcionamiento del repositorio digital de plantas medicinales.

Idea a defender:
El repositorio digital permitirá la recopilación, organización sobre las plantas medicinales, facilitando así el acceso a la información.

Métodos del nivel teórico:

- Método Analítico-Sintético: Posibilitó realizar un análisis de las distintas partes que afectan el objeto de estudio y sintetizar los elementos más significativos.

- Método Histórico-Lógico: Para la realización de la investigación se hizo necesario estudiar la evolución del problema y la existencia de metodologías, procedimientos y sistemas informáticos similares al que se pretende elaborar; determinando cuáles son las tendencias actuales para el desarrollo de un repositorio digital.

Métodos del nivel empírico:

- Observación: El repositorio digital se utiliza para mejorar la búsqueda de información sobre las plantas medicinales al proporcionar una plataforma centralizada y organizada para almacenar y acceder a datos relacionados con su distribución, composición y otros aspectos relevantes.

- Entrevista: Se utilizó para obtener información acerca de cómo se desarrolla el proceso de gestión de la información de las plantas medicinales.

Capítulo 1 Fundamentos teórico-metodológicos:

Este capítulo sienta las bases teóricas de la investigación al explorar los fundamentos conceptuales relacionados con su enfoque y alcance. Analiza sistemas existentes en el campo relevante y describe los lenguajes y herramientas que sustentan el trabajo.

Capítulo 2 Descripción de la propuesta de solución:

El capítulo describe la propuesta de diseño y los entregables generados durante la cuarta fase de la metodología de desarrollo de software Agile Iniciad Procesos (AUP) adaptada para el repositorio digital de plantas medicinales. Esta fase abarca las etapas de análisis, diseño e implementación. Se presentan los diagramas de casos de uso, los diagramas de clases, los prototipos y otros artefactos utilizados para definir los requisitos del sistema, diseñar su arquitectura y desarrollar su implementación.

Capítulo 3 Valoración de la viabilidad de la propuesta de solución:

El capítulo evalúa la viabilidad del repositorio digital de plantas medicinales propuesto mediante la validación de los requisitos del sistema y el diseño arquitectónico. Se realizan pruebas funcionales para verificar que el sistema cumple con los requisitos especificados. También se realizan pruebas unitarias para garantizar que los componentes individuales del sistema funcionan correctamente. Además, se realizan pruebas de aceptación con los usuarios finales para obtener su opinión y garantizar que el sistema cumple con sus necesidades y expectativas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El capítulo presenta los fundamentos teóricos y conceptuales para el desarrollo de un repositorio digital de plantas medicinales. Se revisa la literatura sobre repositorios digitales, sistemas de gestión de información y prácticas de gestión de datos en el campo de las plantas medicinales. Además, se describen los principios y teorías clave relacionados con la organización, el almacenamiento y el acceso eficiente a la información. Se explican los métodos y técnicas utilizados para recopilar y analizar los requisitos de los usuarios y las mejores prácticas para garantizar la calidad y la interoperabilidad de los datos en el repositorio digital.

1.2 Descripción del dominio del problema

La gestión efectiva de la información sobre las plantas medicinales es crucial para garantizar una agricultura sostenible y de calidad. Las plantas medicinales son una valiosa fuente de compuestos bioactivos con propiedades terapéuticas. Al almacenar, organizar y preservar la información sobre estas plantas, podemos facilitar la investigación, el desarrollo y la utilización de sus beneficios medicinales.

La ausencia de un repositorio digital dedicado a las plantas medicinales dificulta la investigación y el desarrollo en este campo. Sin una plataforma centralizada para almacenar y organizar la información sobre las plantas medicinales, los investigadores y los profesionales de la salud enfrentan desafíos para acceder a datos completos y actualizados. Esta carencia obstaculiza la toma de decisiones informadas y aumenta el riesgo de duplicar esfuerzos y perder información valiosa. Por lo tanto, es esencial establecer un repositorio digital que facilite la gestión, el almacenamiento y el acceso eficiente a la información sobre las plantas medicinales.

Un repositorio digital de plantas medicinales sería una herramienta invaluable para investigadores y profesionales de la salud. Al reunir y organizar información de manera centralizada, dicho repositorio facilitaría el acceso a datos completos y actualizados sobre las plantas medicinales. Esto agilizaría la investigación, promovería la colaboración y reduciría el riesgo de perder información valiosa. Además, un repositorio digital permitiría a las instituciones tomar decisiones informadas basadas en evidencia, impulsando así el avance en el campo de la medicina herbal.

1.3 Descripción de los sistemas existentes

DSpace: DSpace es un software de código abierto desarrollado por las bibliotecas del MIT y Hewlett Packard Labs. Es un sistema de repositorio digital que permite capturar y describir documentos digitales.[4]

Open Journal Systems: Open Journal Systems (OJS) es una solución de código abierto para la gestión y publicación de revistas académicas en línea. OJS es un sistema de

gestión y publicación de revistas altamente flexible, operado por un editor, que puede descargarse gratuitamente e instalarse en un servidor web local.[5]

EPrints: EPrints es un sofware gratuito y de código abierto para la creación y gestión de repositorios digitales de acceso abierto adscriptos al estándar OAI-PMH.[6]

Este trabajo no implementará DSpace, ni Open Journal Systems, ni EPrints ya que presentará un método más simple para desarrollar un sistema de repositorio utilizando el framework Django.

1.4 Metodología de desarrollo de software

Una metodología de desarrollo de software es un conjunto de prácticas, procesos y técnicas utilizadas para diseñar, desarrollar y mantener software de alta calidad. Las metodologías proporcionan un marco de trabajo que guía el proceso de desarrollo de principio a fin y hay diferentes enfoques y filosofías de metodologías que se utilizan según los requerimientos del proyecto.

Las metodologías ágiles son un conjunto de metodologías de desarrollo de software que se enfocan en la entrega rápida y constante de software funcional en la cual se centran en la colaboración, la comunicación y la flexibilidad para adaptarse a los cambios en los requisitos del proyecto. En este caso utilizamos la metodología AUP (Agile Unified Process).

1.4.1 Metodología AUP

El AUP (Agile Unified Process) es una metodología ágil de desarrollo de software que se enfoca en la entrega rápida y constante de software a través de ciclos de desarrollo iterativos e incrementales. Esta metodología es una variante del Proceso Unificado de Desarrollo de Software (UP), que combina elementos de UP y metodologías ágiles como Scrum y XP.

El AUP se enfoca en la entrega temprana y constante de software funcional, en lugar de entregar todo el software al final del proyecto. Para lograr esto, el AUP utiliza ciclos iterativos y entregas incrementales, lo que permite al equipo de desarrollo recibir comentarios tempranos y frecuentes de los usuarios finales y realizar ajustes en consecuencia. También enfatiza la colaboración y la comunicación entre los miembros del equipo y los clientes. Los miembros del equipo trabajan juntos en un entorno colaborativo, y los clientes son considerados parte del equipo de desarrollo, lo que les permite proporcionar comentarios y realizar ajustes en tiempo real.[7]

1.4.2 Escenarios para la disciplina Requisitos.

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (CUN, DPN, MC) y existen tres formas de encapsular los requisitos (CUS, HU, DRP), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

Escenario No 1:

Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

Escenario No 2:

Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

Escenario No 3:

Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

Escenario No 4:

Proyectos que no modelen negocio solo pueden modelar el sistema con HU. 2.4 Características por escenarios.

Teniendo en cuenta los escenarios de la disciplina de requisitos de software de la metodología AUP variación UCF, se selecciona el escenario No.4 ya que aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio bien definido.

1.5 Arquitectura de software

La arquitectura del software se refiere al diseño y estructura de un sistema de software. Es un conjunto de decisiones y patrones de diseño que definen cómo se organizarán los componentes de software y cómo interactuarán entre sí para lograr los objetivos del sistema. Se enfoca en aspectos como la estructura de los componentes, su interacción y comunicación, la distribución de la funcionalidad en diferentes capas o módulos, la gestión de datos y la seguridad del sistema.

Es importante porque proporciona una visión clara y coherente del sistema de software, lo que facilita la comprensión y el mantenimiento del sistema a lo largo del tiempo. Una

buena arquitectura de software también permite la escalabilidad y la adaptabilidad del sistema a medida que cambian las necesidades del negocio o los requisitos del usuario.[8]

Además, la arquitectura del software también puede influir en la calidad del software, como la eficiencia, la fiabilidad, la seguridad y la facilidad de uso. Por lo tanto, una arquitectura bien diseñada puede contribuir significativamente al éxito del proyecto de software.

Los principios arquitectónicos y los patrones de diseño que sustentan los repositorios de documentos son ampliamente aplicables. Los estilos y patrones arquitectónicos pueden adaptarse a otros sistemas y expresarse mediante abstracciones que permiten a los ingenieros de software describir la arquitectura de manera consistente y predecible. Esto facilita el diseño e implementación de sistemas robustos y escalables.[9]

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

- **Descomposición Modular:** El software se estructura en grupos funcionales muy acoplados.
- **Cliente-servidor:** El software reparte su carga de cómputo en dos partes independientes, pero sin reparto claro de funciones.
- **Arquitectura de tres niveles:** Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.[10]

Otras arquitecturas menos conocidas son:

- Orientada a servicios (SOA del inglés Service-Oriented Architecture).
- Dirigida por eventos.
- Máquinas virtuales.
- Modelo Vista Controlador.
- Entre pares.
- En pipeline.

- Arquitectura de microservicios (MSA del inglés MicroServices Architecture). Algunos consideran que es una especialización de una forma de implementar SOA. [11]

1.5.1 Estilos arquitectónicos

Los estilos arquitectónicos categorizan los sistemas según los siguientes elementos:

- Componentes: Módulos funcionales del sistema.
- Conectores: Mecanismos de comunicación y coordinación entre componentes.
- Restricciones: Reglas que gobiernan la integración de componentes.
- Modelos semánticos: Representaciones que describen las propiedades generales del sistema en función de las propiedades de sus componentes.

Estos elementos proporcionan un marco para describir y analizar la arquitectura del sistema, facilitando el diseño e implementación de sistemas sólidos y escalables. De estos, se puede mencionar que:

- Sirven para sintetizar estructuras de soluciones.
- Pocos estilos abstractos encapsulan una enorme variedad de configuraciones concretas.
- Definen los patrones posibles de las aplicaciones.
- Permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales.

Al diseñar sistemas distribuidos, es crucial equilibrar el rendimiento, la confiabilidad, la seguridad y la facilidad de administración. No existe un enfoque único, lo que ha llevado al surgimiento de varios estilos arquitectónicos distribuidos. La elección del estilo arquitectónico apropiado depende de los requisitos no funcionales específicos del sistema. El objetivo es seleccionar un estilo que aborde y respalde eficazmente estos requisitos para garantizar un sistema distribuido robusto y bien equilibrado.[12]

En esta sección se describen cinco estilos arquitectónicos:

1. **Arquitectura maestro-esclavo,** que se usa en sistemas de tiempo real en los que se requiere garantía de tiempos de respuesta de interacción.
2. **Arquitectura cliente-servidor de dos niveles**, que se usa para sistemas cliente-servidor simple, y en situaciones donde es importante centralizar el sistema por

razones de seguridad. En tales casos, la comunicación entre el cliente y el servidor por lo general está encriptada.

3. **Arquitectura cliente-servidor multinivel**, que se usa cuando existe un enorme volumen de transacciones a procesar por el servidor.

4. **Arquitectura de componentes distribuidos**, que se usa cuando es necesario combinar los recursos de diferentes sistemas y bases de datos, o como un modelo de implementación para sistemas cliente-servidor multinivel.

5. **Arquitectura peer-to-peer** (entre pares o punto a punto, o par a par), que se usa cuando los clientes intercambian de manera local la información almacenada, y el papel del servidor es presentar a los clientes entre sí. También puede usarse cuando se deba elaborar un amplio número de cálculos independientes.[13]

1.6 Tecnologías y herramientas para el desarrollo de la solución

Para crear soluciones tecnológicas exitosas e innovadoras, es fundamental contar con herramientas y tecnologías adecuadas. La elección de las herramientas y tecnologías correctas es esencial para garantizar la eficiencia, la escalabilidad, la seguridad y una buena experiencia de usuario. Los desarrolladores y las empresas disponen de una gran variedad de opciones, que incluyen lenguajes de programación, plataformas de desarrollo y herramientas de colaboración. Analizar y evaluar estas opciones permite tomar decisiones acertadas que allanen el camino para el éxito en el desarrollo de soluciones tecnológicas competitivas.

1.6.1 Tecnología del lado del servidor.

Django

Django es un framework web Python de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático. Construido por desarrolladores experimentados, se encarga de gran parte de las complicaciones del desarrollo web, para que pueda centrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto. [14]

Python

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes.[15]

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto y gratuito. Es considerado uno de los sistemas de gestión de bases de datos más poderosos y avanzados disponibles en la actualidad.[16]

1.6.2 Tecnología del lado del cliente.

Bootstrap

El Bootstrap es un framework de diseño web que proporciona un conjunto de herramientas y componentes para crear páginas web responsivas y con un diseño moderno y atractivo. Bootstrap fue desarrollado por Twitter y ahora es mantenido por la comunidad de desarrolladores de software de código abierto. [17]

JavaScript

Es un lenguaje de programación diseñado en un principio para añadir interactividad a las páginas webs y crear aplicaciones web. A pesar de la similitud en el nombre, no está relacionado con Java. Se emplea en el desarrollo de páginas web para tareas como cambiar automáticamente la fecha de una página, hacer que una página aparezca en una ventana emergente al hacer clic en un enlace o que un texto o imagen cambien al pasar el ratón por encima. También suele emplearse para hacer encuestas y formularios. Se ejecuta en el ordenador del visitante a la web, por lo que no requiere descargas constantes desde el sitio web.[18]

HTML (Lenguaje de Marca de Hipertexto)

HTML no es un lenguaje de programación; es un lenguaje de marcado que define la estructura de tu contenido. HTML consiste en una serie de elementos que usarás para encerrar diferentes partes del contenido para que se vean o comporten de una determinada manera. Las etiquetas de encierre pueden hacer de una palabra o una imagen un hipervínculo a otro sitio, se pueden cambiar palabras a cursiva, agrandar o achicar la letra, etc.[19]

CSS (Hojas de Estilos en Cascadas)

CSS es el lenguaje que utilizamos para dar estilo a un documento HTML.CSS describe cómo deben mostrarse los elementos HTML.[20]

1.6.3 Herramientas CASE (Ingeniería de Software Asistida por Computadora).

Visual Studio Code

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución (como C++, C#, Java, Python, PHP, Go, .NET).[21]

1.6.4 Herramientas para el diseño.

Enterprise Architect

Enterprise Architect es una herramienta gráfica multi-usuario diseñada para ayudar a su equipo a construir sistemas robustos y fáciles de mantener. Incorporando reporting integrado y documentación de alta calidad.[22]

Draw.io

Draw.io es un software utilizado para diseñar diagramas de forma gratuita y offline, aunque también tiene una versión completamente funcional en el navegador web, y, además, facilita la integración con múltiples plataformas y programas. Esta herramienta permite realizar cualquier tipo de diagrama de flujo, diagramas de procesos, organigramas, así como diagramas de red, UML, mapas conceptuales y otros elementos necesarios para realizar un diseño.[23]

Pencil Project

Pencil Project es una herramienta de escritorio gratuita y de código libre para **crear diagramas.** Tiene una interfaz intuitiva y permite colaborar en tiempo real. Pencil Project es una herramienta de prototipado de aplicaciones web que permite crear wireframes de webs y aplicaciones móviles y dispone de algunas bibliotecas de símbolos para simular interfaces.[24]

1.7 Conclusiones parciales

Se realizó un análisis del dominio del problema para un mejor entendimiento de la investigación. El repositorio propuesto aprovechará la metodología de desarrollo del software AUP (Análisis, Diseño, Programación y Pruebas) para garantizar un desarrollo estructurado y sistemático. El estudio de las herramientas a la propuesta de solución arrojó de que estas no son las ideas para este caso en particular, lo que justifica el desarrollo e implementación de la propuesta de solución. Se seleccionaron las tecnologías y herramientas adecuadas para garantizar la eficiencia y escalabilidad del repositorio.

CAPÍTULO 2: DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

2.1 Introducción

El capítulo describe la propuesta de diseño y los entregables generados durante las fases uno y dos a través del flujo de trabajo de la modelación de la metodología de desarrollo de software Agile Unified Procesos (AUP) adaptada para el repositorio digital de plantas medicinales. Esta fase abarca las etapas de análisis, diseño e implementación. Se presentan los diagramas de historias de usuario, los diagramas de clases, los prototipos y otros artefactos utilizados para definir los requisitos del sistema, diseñar su arquitectura y desarrollar su implementación.

2.2 Concepción general del sistema

El sistema digital desarrollado para investigadores y profesionales de la Facultad de Agronomía de la Universidad de Cienfuegos permite recopilar, organizar y almacenar información integral sobre las plantas medicinales en una plataforma centralizada. Facilita el acceso seguro y confiable a datos detallados sobre las plantas medicinales, incluyendo su distribución geográfica, propiedades farmacológicas y usos tradicionales. El sistema incorpora metadatos y garantiza la compartición segura y accesible de la información con la comunidad científica y otros usuarios autorizados.

2.3 Requisitos

En proyectos informáticos, los requisitos son las especificaciones que deben cumplirse para garantizar el éxito del proyecto. Estos requisitos se dividen en dos categorías principales:

Requisitos funcionales: Describen las acciones, funciones y funcionalidades específicas que debe realizar el sistema informático.

Requisitos no funcionales: Establecen restricciones y cualidades generales del sistema, como rendimiento, seguridad, usabilidad y mantenibilidad.

2.3.1 Requisitos funcionales

Los requisitos funcionales identificados para el sistema de repositorio digital de plantas medicinales son los siguientes:

N°	Nombre	Descripción	Prioridad	Complejidad
RF 1	Iniciar sesión.	El usuario introduce sus datos, el sistema lo comprueba y si son válidos, el usuario queda autenticado con el nivel de privilegios asignados; si los datos no son válidos el sistema muestra un mensaje de error.	Alta	Alta

RF 2	Crear usuario.	El sistema permite que los usuarios con rol administrador creen nuevos usuarios con el nombre, usuario y rol.	Alta	Media
RF 3	Modificar usuario.	El sistema permite que los usuarios con rol administrador modifiquen los usuarios ya existentes.	Alta	Baja
RF 4	Eliminar usuario.	El sistema permite que los usuarios con rol administrador eliminen cuentas de usuarios existentes.	Alta	Media
RF 5	Listar usuario	El sistema permite que los usuarios con rol administrador visualicen el listado de usuarios existentes.	Baja	Media
RF 6	Agregar Artículo científico.	El sistema permite que los usuarios con rol administrador y editor agregar el título del archivo, agregar el archivo en la base de datos, agregar el autor, el año, el país y la fuente de publicidad para que aparezca en la interfaz de investigador.	Alta	Alta
RF 7	Editar Artículo científico.	El sistema permite que los usuarios con rol administrador y editor editar el título, el año, el autor, el país, fuente de publicidad y seleccionar otro Artículo científico.	Media	Baja
RF 8	Eliminar Artículo científico	El sistema permite que los usuarios con rol administrador eliminar Artículo científico.	Alta	Media
RF 9	Mostrar una lista de Artículo científico.	El sistema permite que los usuarios con rol investigador visualizar listado de los Artículo científico.	Alta	Baja
RF 10	Permitir la descarga de archivos PDF.	El sistema permite que los usuarios con rol investigador puedan descargar el archivo deseado.	Alta	Baja

RF 11	Mostrar la descripción de un Artículo científico.	El sistema permite que los usuarios con rol investigador puedan visualizar la descripción del archivo deseado.	Alta	Baja
RF 12	Realizar búsqueda avanzada de Artículo científico.	El sistema permite que los usuarios con rol investigador puedan realizar una búsqueda avanzada de él o de los archivos deseados según el título, autor, año o país documento que desea buscar.	Alta	Media
RF 13	Cerrar sesión	El sistema permita que los usuarios con el rol de administrador y editor puedan cerrar sesión para salir del sistema.	Alta	Baja
RF 14	Cambio de contraseña.	El sistema permitirá cambiar la contraseña del usuario autenticado actualmente.	Alta	Baja

Tabla 1 Requisitos Funcionales

2.3.2 Requisitos no funcionales

Los requisitos no funcionales identificados para el sistema de repositorio digital de plantas medicinales son los siguientes:

- **Requisito de interfaz o apariencia externa:** La interfaz debe ser sencilla, intuitiva, amigable y mantener el formato en vistas similares. Debe ser legible y de fácil uso para el usuario.
- **Requisito de software:** Se necesita que la computadora posea un navegador web.
- **Requisito de hardware:** El sistema requiere de una computadora que haga la función de servidor.
- **Requisito de seguridad:** El sistema debe ser seguro y proteger los archivos PDF de accesos no autorizados.
- **Requisito de diseño responsivo:** La aplicación debe seguir los estándares de diseño responsivo para verse correctamente en dispositivos móviles y de escritorio.
- **Requisito de gestión de errores:** El sistema debe gestionar correctamente los errores y proporcionar mensajes de error claros al usuario cuando sea necesario.

Estos requisitos funcionales y no funcionales son fundamentales para el desarrollo exitoso del sistema de repositorio digital de plantas medicinales. Aseguran que el

sistema cumpla con las funcionalidades esperadas, así como con los estándares de rendimiento, seguridad y usabilidad necesarios.

2.4 Historias de Usuario

Las historias de usuario son una técnica empleada para definir los requisitos del software de forma clara y concisa. Son similares a los casos de uso en el Proceso Unificado y sirven como base para probar la funcionalidad del software. En total se definieron 14 historias de usuarios, a continuación, se muestra la historia de usuario,
"Agregar Artículo científico.":

Número:6	**Requisito:** Agregar Artículo científico.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:10 horas**
Riesgo en desarrollo:	**Tiempo Real:**
Descripción: El componente permitirá que usuario rol administrador y editor podrá poner el título del archivo, seleccionar al archivo deseado, poner el autor. La fuente de publicidad, el año y el país. **Title:** Campo de texto donde el usuario entrara el nombre del archivo deseado. **Pdf file:** Campo de selección donde podrás seleccionar el archivo deseado. **Authors:** Campo de texto donde el usuario entrara el autor del archivo deseado. **Published source:** Campo de texto donde el usuario entrara la fuente de publicidad del archivo deseado. **Year:** Campo de texto donde el usuario entrara el año del archivo deseado. **Country:** Campo de texto donde el usuario entrara el nombre del país del archivo deseado.	
Observaciones: El administrador o editor no podrá dejar la casilla del Title vacía ni podrá guardar esta información si no ha seleccionado ningún archivo.	
Prototipo elemental de interfaz gráfica de usuario:	

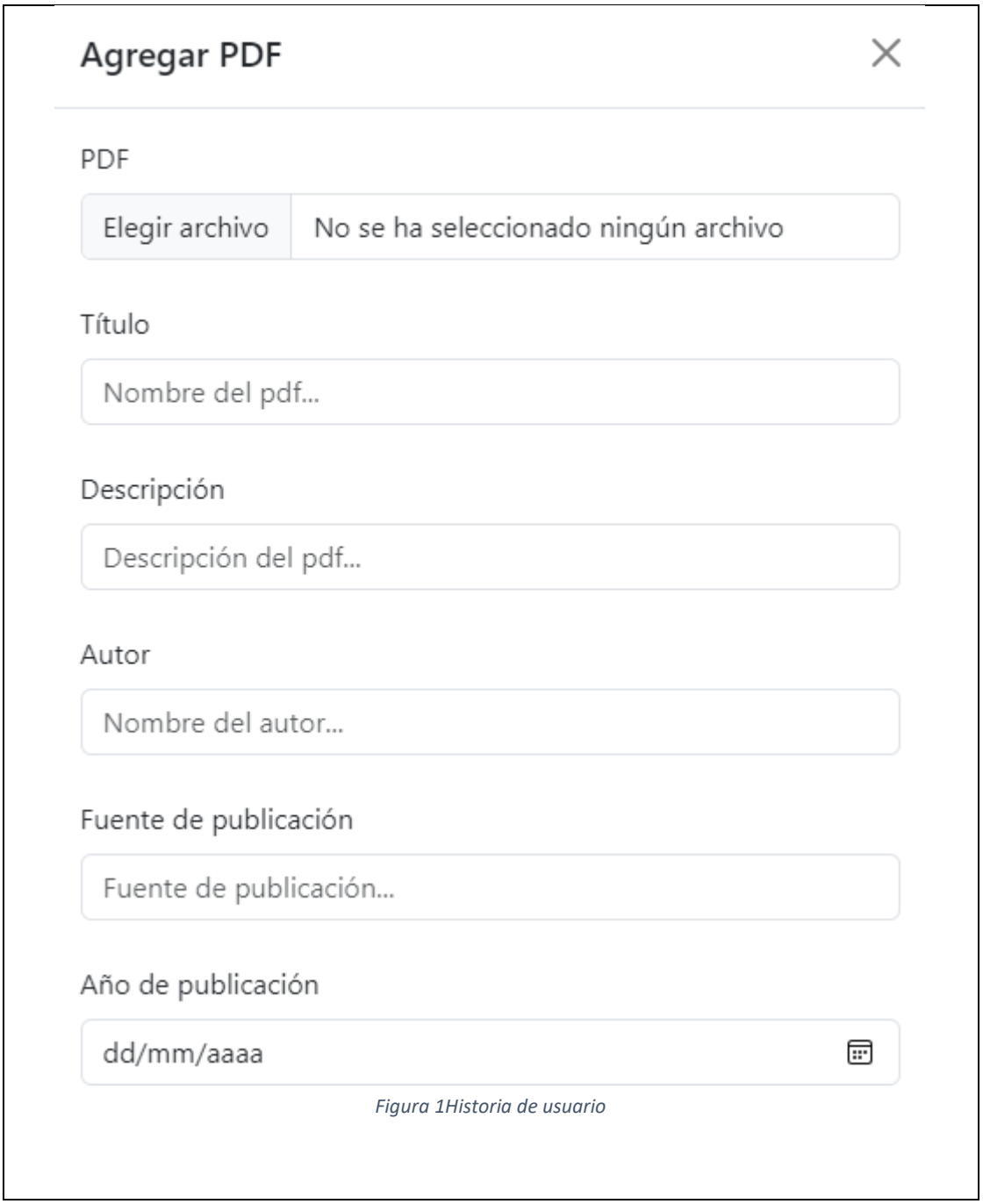


Figura 1Historia de usuario

2.5 Descripción de la arquitectura

La arquitectura del software detalla los elementos esenciales del diseño del sistema, ofreciendo una descripción más detallada de su implementación.

2.5.1 Patrón de arquitectura Modelo Vista Plantilla

La solución propuesta se basa en el patrón Modelo-Vista-Plantilla, que estructura el código de la aplicación de manera específica:

Modelo: Este componente maneja todo lo relacionado con la base de datos. Define la estructura de los datos y se encarga de las operaciones de lectura y escritura, la validación de los datos y las relaciones entre las tablas.

Vista: En Django, la vista es el lugar donde se implementa la lógica de negocio. Toma un Web request y devuelve un Web response. La vista decide qué datos se deben mostrar y cómo mostrarlos y luego se los pasa a la plantilla.

Plantilla: Este componente se encarga de la presentación de los datos. Es donde se define cómo se deben mostrar los datos en la interfaz de usuario. Django tiene su propio lenguaje de plantillas que permite la inserción de datos dinámicos en HTML. [25]

La siguiente imagen muestra como los tres componentes del patrón arquitectónico se relacionan, evidenciando que la Vista es la capa intermedia entre la Plantilla y el Modelo:

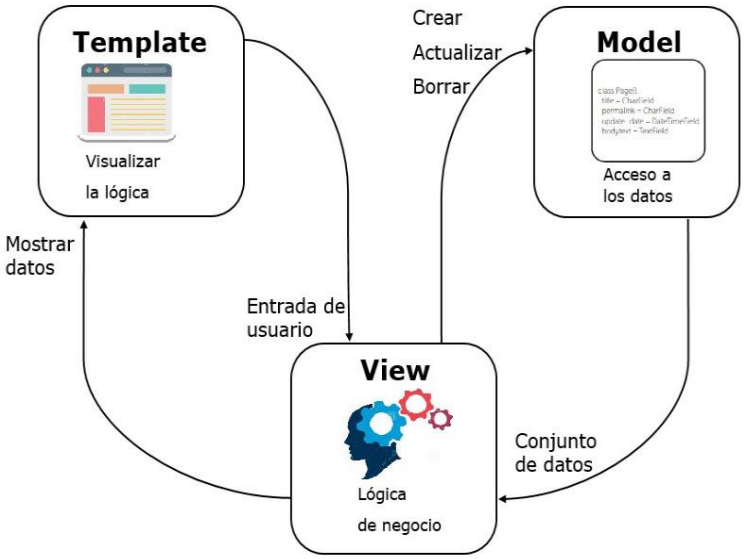


Figura 2. Patron de arquitectura Modelo, Vista, Template.

A continuación, se muestra la arquitectura de la aplicación a la solución propuesta:

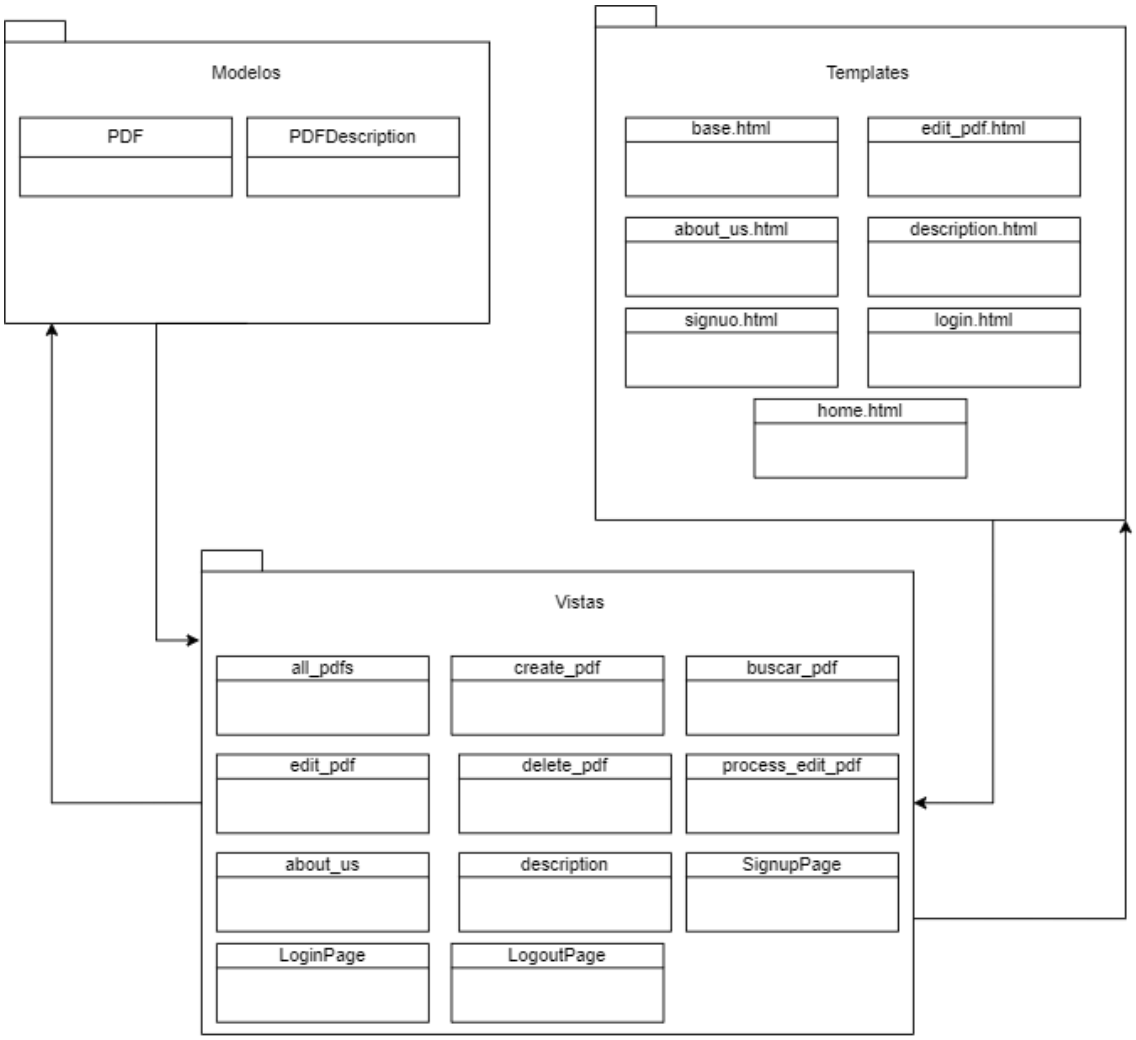


Figura 3.Aplicación de la Arquitectura a la Propuesta de Solución.

2.6 Diagrama de clases diseño

Un diagrama de clases es un diagrama estático que representa la estructura de un sistema mediante la visualización de sus clases. Describe las clases que forman el modelo del sistema y sus relaciones. Es una herramienta iterativa que se crea y refina durante el análisis, el diseño y la implementación del sistema, guiando el desarrollo del mismo.[26]

A continuación, se muestra el diagrama de clases de la propuesta de solución:

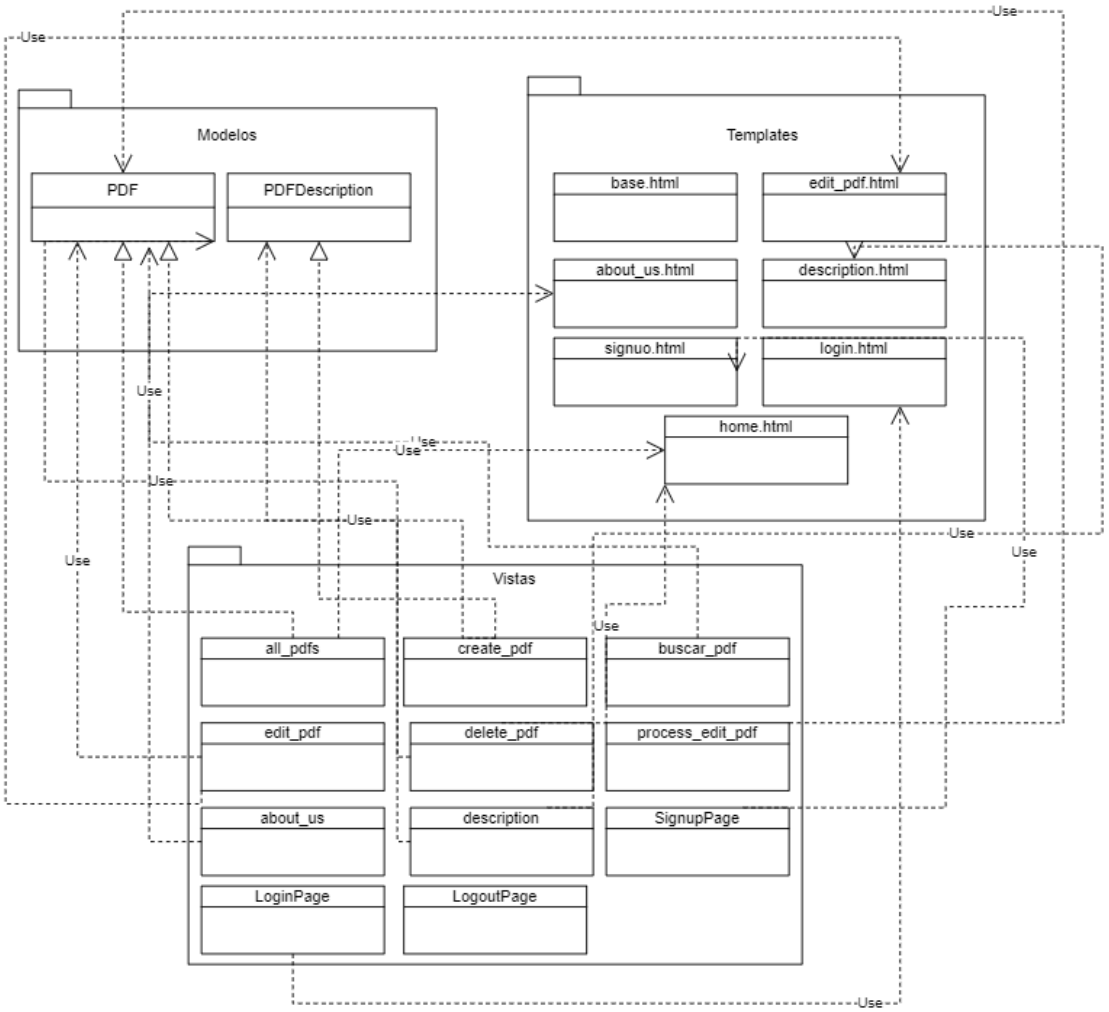


Figura 4.Diagrama de clases diseño.

2.7 Patrones de diseño

Los patrones de diseño proporcionan esquemas para estructurar y relacionar componentes de software, ayudando a crear sistemas robustos y mantenibles. [27]

Patrones GRASP: (patrones generales de software para asignar responsabilidades): son principios que guían la asignación de responsabilidades a clases y objetos en el diseño orientado a objetos. Ayudan a crear diseños más limpios, mantenibles y reutilizables. Estos patrones incluyen:

- **Experto:** Asignar responsabilidades a las clases que tienen la información necesaria para realizarlas.
- **Creador:** Este patrón ayuda a determinar quién es responsable de crear nuevos objetos o clases. Se utilizó para identificar qué la clase A debe crear elementos de una clase B, apoyándose en que la clase A debería: contener, agregar, registrar, utilizar y tener los datos de inicialización de la clase B.
- **Bajo acoplamiento:** Asignar una responsabilidad para mantener bajo acoplamiento. El grado de acoplamiento no puede considerarse aisladamente de

otros principios como Experto y Alta Cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño.

- **Alta Cohesión:** Asignar una responsabilidad de modo que la cohesión siga siendo alta. Las clases controladoras contienen los patrones de bajo acoplamiento y alta cohesión nivelados, pues permite el uso de los componentes de forma individual, evidenciando el bajo acoplamiento, así como la dependencia entre ellos o alta cohesión.[28]

Patrones GoF: Los patrones de diseño GoF son un conjunto de principios que guían la organización y conexión de objetos en el desarrollo de software. Estos patrones, popularizados por el libro "Design Patterns" escrito por Gang of Four (GoF), brindan soluciones reutilizables para problemas comunes de diseño. Al agrupar las relaciones entre clases y su composición, los patrones GoF permiten crear estructuras intrincadas que simplifican tareas complejas de software.[29]

Patrón Decorator: nos permite componer nuevos objetos a partir de objetos ya existentes sin utilizar directamente la herencia.

Patrón de Repositorio (Repository): Es un patrón para abstraer el acceso a los datos y proporcionar una capa de abstracción entre el modelo y el almacenamiento de datos. Esto te permite cambiar la fuente de datos sin afectar la lógica de negocio en las vistas y templates.[31]

Mediator (Mediador): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto. Se evidencia en las urls.py, las cuales son mediadoras entre las clases Modelo, Vista y Template.[32]

2.8 Modelo de datos

Las bases de datos necesitan una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. La estructura tiene que ser desarrollada para la necesidad de las aplicaciones que la usarán. La puesta en práctica de la base de datos es el paso final en el desarrollo de aplicaciones. Se conforman con los requisitos del proceso del negocio, que es la primera abstracción de la vista de la base de datos.

Un modelo de datos se puede definir como un conjunto de conceptos, reglas y convenciones bien definidos que permiten aplicar una serie de abstracciones a fin de describir y manipular los datos de un cierto mundo real que se desea almacenar en la base de datos. El modelo relacional se caracteriza a muy grandes rasgos por disponer que toda la información debe estar contenida en tablas, y las relaciones entre datos deben ser representadas explícitamente en esos mismos datos.

A continuación, se muestra el modelo de datos:

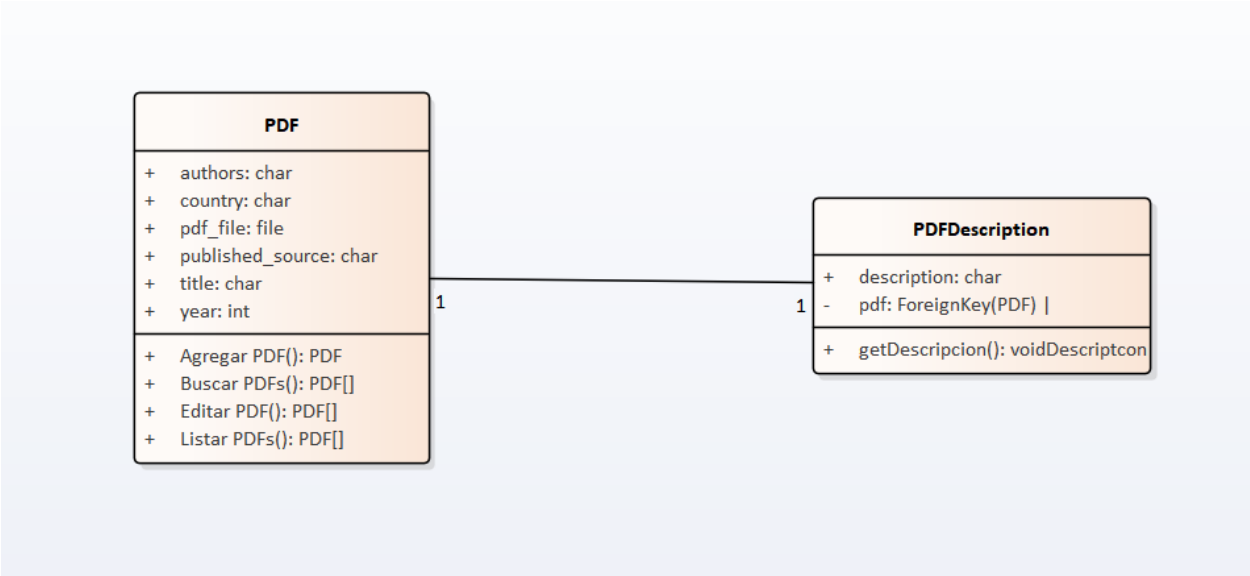


Figura 5.Modelo de Datos.

2.9 Estándares de codificación empleado

Convenciones de nomenclatura

General

- Se exceptúan el uso de las tildes y la letra ñ.
- El nombre de todas las variables comenzará con letra minúscula y si este está compuesto por varias palabras, estarán separados por guion bajo. Ejemplo: “published_source”.

Clases

- El nombre de las clases comenzará con minúsculas si este está compuesto por varias palabras, todas las palabras internas estarán separados por guion bajo. Ejemplo: ‘’buscar_pdf’’.
- Intentar mantener los nombres de las clases descriptivos y simples. Usar palabras completas, evitar acrónimos y abreviaturas, a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML.

Nombres de variables

- No se utilizarán nombres de variables que puedan ser ambiguos.

2.10 Conclusiones

En este capítulo, se ha presentado la solución propuesta para el desarrollo del repositorio de plantas medicinales. Se detallaron los requisitos, historias de usuario, arquitectura y patrón de diseño utilizado. Además, se analizaron los patrones de diseño,

modelos de datos y estándares de codificación empleados a lo largo del proceso. Toda esta información sienta las bases sólidas para avanzar a las siguientes etapas del proyecto, como las pruebas y el despliegue final del sistema.

CAPÍTULO 3: VALORACIÓN DE LA VIABILIDAD DE LA PROPUESTA DE SOLUCIÓN

En el presente capítulo se valora la viabilidad de la propuesta de solución mediante la validación de los requisitos y el diseño propuesto, además de la realización de pruebas funcionales, unitarias y de aceptación a la plataforma de Repositorio Digital de Plantas Medicinales.

3.1 Pruebas de Software

Niveles de pruebas

Los niveles de prueba organizan y administran actividades para verificar y validar los componentes de un producto. Incluyen prueba de unidad (verifica el funcionamiento de partes individuales), prueba de integración (verifica la interacción entre componentes), prueba de sistema (evalúa el cumplimiento de los requisitos) y prueba de aceptación (verifica la funcionalidad según las necesidades del cliente). Cada nivel utiliza diferentes tipos y técnicas de pruebas para su realización. Para los efectos de la presente investigación se propone la siguiente estrategia de pruebas[33]:

Tipo de Prueba	**Método de prueba**	**Técnica de prueba**
Unitarias	Caja blanca	Camino básico
Funcionalidad	Caja negra	Partición de equivalencia

Tabla 2 Estrategia de prueba aplicada a la solución

Pruebas de Caja Blanca

La prueba de caja blanca utiliza la estructura de control del diseño procedimental para diseñar casos de prueba. Este enfoque garantiza el ejercicio de todos los caminos independientes de cada módulo, las decisiones lógicas, los bucles en sus límites operacionales y las estructuras internas de datos. Además, se emplea la técnica de prueba del camino básico para asegurar que cada sentencia del programa se pruebe al menos una vez, utilizando la medida de complejidad de un procedimiento o algoritmo para obtener un conjunto básico de caminos de ejecución y casos de prueba.

A continuación, se muestra el proceso de pruebas realizado a la funcionalidad login_view, específicamente a la controladora tbUser en donde su funcionalidad consiste en iniciar sesión en la plataforma. Primeramente, se comienza por analizar el código y enumerar las instrucciones[34]:

```python
#Funcion para Loguear mis usuarios
def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(request, username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect(request.GET.get('next', 'inicio'))
            else:
                messages.error(request, 'El usuario o la contraseña son incorrectos.')
                form = AuthenticationForm(request)
    else:
        form = AuthenticationForm(request)
    context ={
        'form': form
    }

    return render(request, 'api/login.html',context)
```

Figura 6. Función login_view

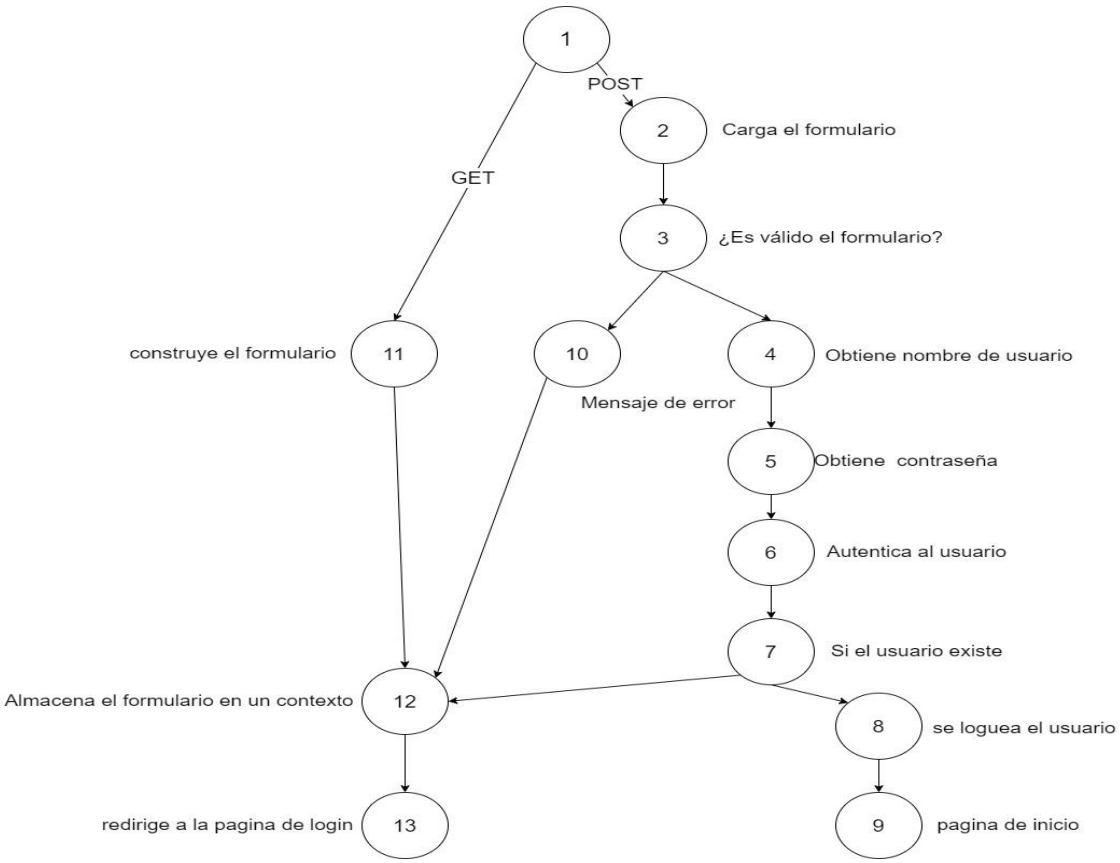


Figura 7. Grafo Resultante

Luego se determina la complejidad ciclomática V (G) del grafo resultante, la cual es un indicador del número de caminos independientes que existen en un grafo, es decir, es cualquier camino dentro del código que introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición. La complejidad ciclomática puede ser calculada de 3 formas

- V (G) = a - n + 2, siendo a el número de arcos o aristas del grafo y n el número de nodos.
- V (G) = r, siendo r el número de regiones cerradas del grafo.
- V (G) = c + 1, siendo c el número de nodos de condición.

Realizando los cálculos correspondientes se obtiene por cualquiera de las variantes el siguiente resultado:

V (G) = a - n + 2

V (G) = 14 – 13 +2

V(G) = 3

Una vez calculada la complejidad ciclomática de un fragmento de código, se puede determinar el riesgo que supone utilizando los rangos definidos en la siguiente tabla:

Complejidad Ciclomática	Evaluación del Riesgo
1-10	Programa Simple, sin mucho riesgo
11-20	Más complejo, riesgo moderado
21-50	Complejo, Programa de alto riesgo
50	Programa no testeable, Muy alto riesgo

La complejidad ciclomática del código, que es 3, lo ubica en una categoría de simplicidad. Esto significa que el programa es fácil de comprender, mantener y probar, lo que reduce el riesgo de errores y facilita la garantía de calidad y estabilidad del software.[35]

3.1.2 Pruebas de Caja negra

Las pruebas de caja negra se enfocan en los requisitos funcionales del software, permitiendo a los ingenieros obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos. Este enfoque es complementario a las técnicas de caja blanca y busca descubrir diferentes tipos de errores, incluyendo funciones incorrectas o incompletas, errores de interfaz, problemas en estructuras de datos o

accesos a bases de datos externas, errores de rendimiento, y problemas de inicialización y terminación.

Para desarrollar las pruebas de caja negra existen varias técnicas, entre ellas se encuentran:

• **Técnica de la Partición de Equivalencia:** esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

• **Técnica del Análisis de Valores Límites:** esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables

• **Técnica de Grafos de Causa-Efecto:** es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

La técnica de Partición de equivalencia es una técnica de prueba que se utiliza para reducir el número de casos de prueba necesarios para probar un rango de valores de entrada. Para aplicar la prueba de partición de equivalencia a un caso de uso Agregar PDF, podríamos dividir cada campo en rangos válidos e inválidos.[36]

Identificación de particiones de equivalencia:

PDF

- Partición válida: Agregar archivo
- Partición no válida: No agregar archivo

Título

- Partición válida: Título no vacío
- Partición no válida: Título vacío

Descripción

- Partición válida: Descripción no vacía
- Partición no válida: Descripción vacía

Autor

- Partición válida: Autor no vacío
- Partición no válida: Autor vacío

Fuente de publicación

- Partición válida: Fuente de publicación no vacía
- Partición no válida: Fuente de publicación vacía

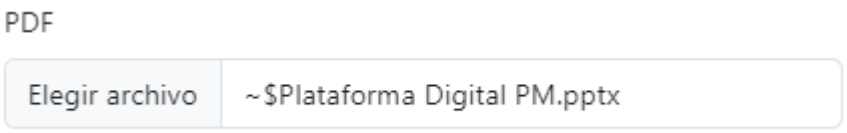
Año de publicación

- Partición válida: Fecha no vacía
- Partición no válida: Fecha vacía

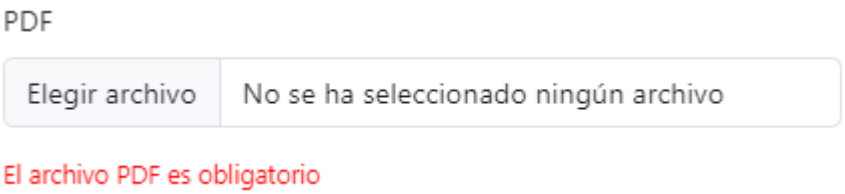
Casos de Prueba:

- **PDf:**

  - Caso de prueba 1 (Válido): Agregar archivo



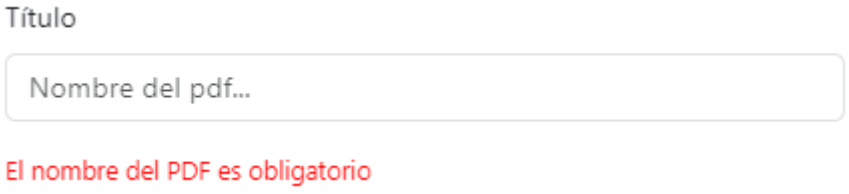
  - Caso de prueba 2 (No Válido): No agregar archivo



- **Título**

  - Caso de prueba 3 (Válido): Título válido que no está registrado en la base de datos

  - Caso de prueba 4 (No Válido): Título vacío



- **Descripción**

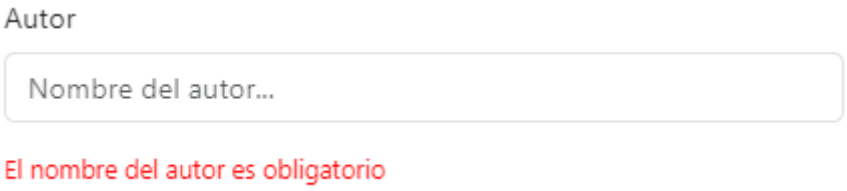
  - Caso de prueba 5 (Válido): Descripción no vacía.

  - Caso de prueba 6 (No Válido): Descripción vacía

- **Autor**

  - Caso de prueba 7(Válido): Nombre de autor no vacío

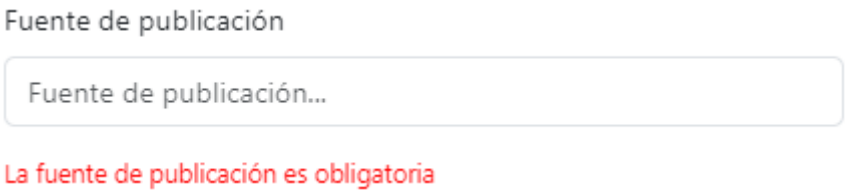
  - Caso de prueba 8 (No Válido): Nombre del autor vacío



- **Fuente de publicación**

  - Caso de prueba 9 (Válido): Fuente de publicación no vacía

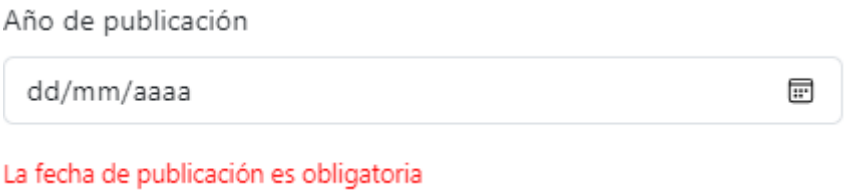
  - Caso de prueba 10 (No Válido): Fuente de publicación vacía



- **Año de publicación**

  - Caso de prueba 11 (Válido): Año de publicación no vacía

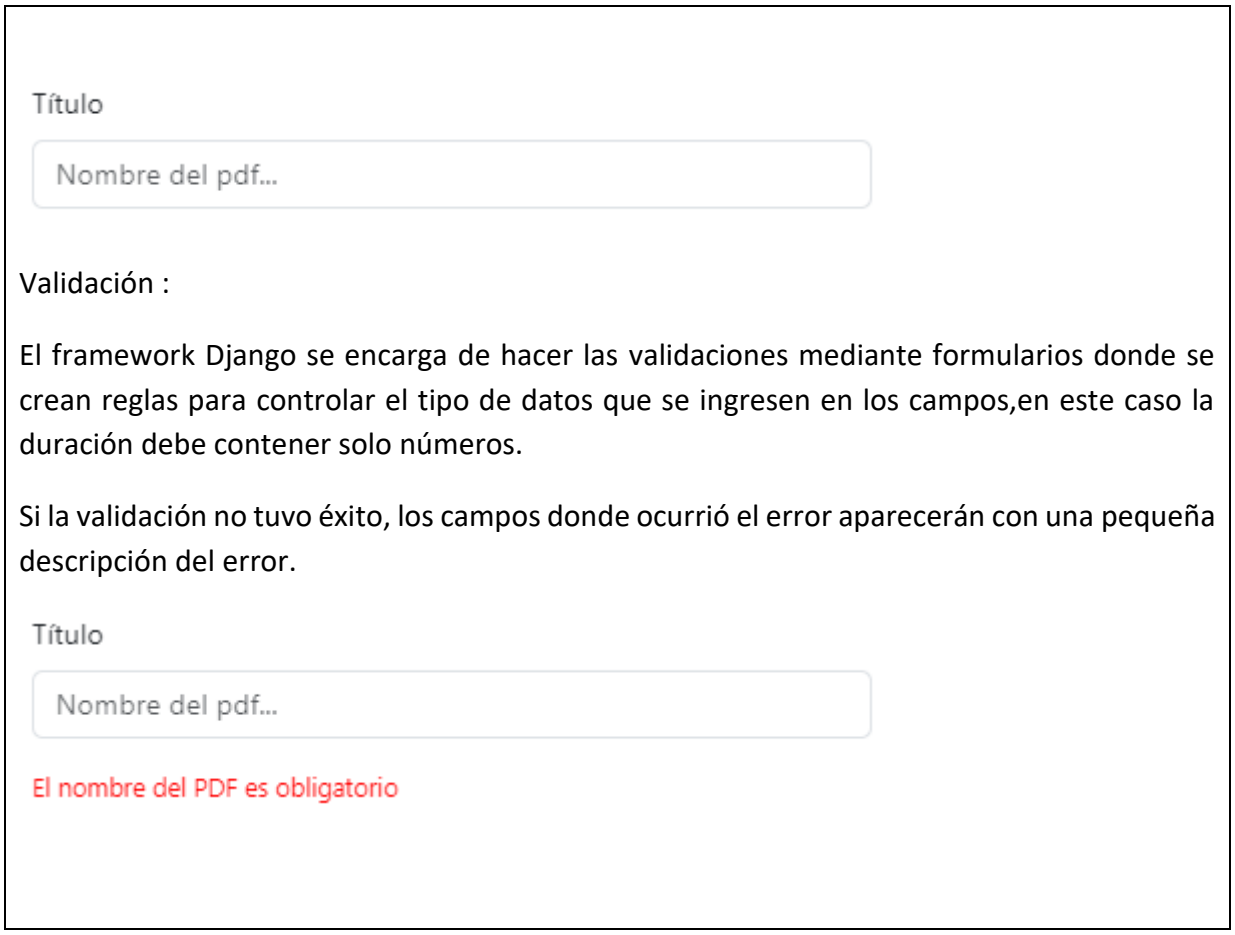
  - Caso de prueba 12 (Válido): Año de publicación vacía



3.2 Diseño de pruebas funcionales

Las pruebas son un aspecto crucial en el control de calidad del desarrollo de software y, dentro de estas, las pruebas funcionales, en las cuales se verifica de forma dinámica el comportamiento de un sistema, basada en la observación de un conjunto seleccionado de ejecuciones controladas o casos de prueba. Las pruebas funcionales son las que se aplican al producto final, permitiendo detectar los puntos del producto que no cumplen sus especificaciones, es decir, que no funcionan correctamente.[37]

Requisito funcional	Agregar Artículo científico
Resumen	El Sistema le permite al usuario Agregar Artículo científico
Pruebas funcionales del sistema	
Sección para insertar los datos de un Artículo científico	
Es la interfaz donde el usuario Agrega los Artículos científicos que va a tener la plataforma.	



Conclusiones

Las pruebas de software realizadas han sido cruciales para asegurar la calidad del producto. Las pruebas de Caja Negra se han centrado en verificar el funcionamiento de las funciones del software desde el punto de vista del usuario, asegurando que la entrada se procesa correctamente y se generan resultados precisos. Esto garantiza la integridad de la información que se maneja. Por otro lado, las pruebas de Caja Blanca se han dirigido a validar la lógica interna del software, detectando posibles errores o desviaciones en el código. Estas pruebas permiten una corrección temprana de los problemas que se puedan encontrar. Las pruebas funcionales, como parte integral de la validación, simulan el uso del software en escenarios específicos, evaluando su comportamiento de acuerdo a las especificaciones definidas. Esto permite identificar las áreas donde el software no funciona correctamente y se desvía de lo previsto, asegurando un producto final de alta calidad.

Conclusiones generales

A través del desarrollo del presente proyecto de diploma, se logró crear un Repositorio de plantas medicinales para la Universidad Carlos Rafael Rodríguez. Este trabajo implicó:

Se obtuvo una especificación completa de los procesos de gestión de documentos académicos en la Universidad de Cienfuegos, y se identificaron las principales problemáticas.

Se diseñó un repositorio digital siguiendo el flujo de trabajo de la metodología AUP, lo cual sirvió como guías para la implementación.

Se implementó un repositorio digital de plantas medicinales en la Universidad de Cienfuegos que garantiza que la información se mantenga accesible y actualizada a largo plazo, facilitando la investigación, el aprendizaje y la consulta para la comunidad universitaria.

Las técnicas de validación realizadas permitieron confirmar el correcto funcionamiento y la satisfacción de los usuarios, lo cual aportó significativamente a la calidad del aporte práctico de la investigación.

Recomendaciones

- Continuar el estudio con el objetivo de añadir nuevas funcionalidades al sistema.
- Desarrollar una aplicación móvil que permita el acceso a los artículos científicos con funciones similares a la versión web.

Referencias Bibliografícas

[1] A. Keefer, «LOS REPOSITORIOS DIGITALES UNIVERSITARIOS Y LOS AUTORES», *An. Doc.*, 2007.

[2] C. B. Ortí y U. de Valencia, «LAS TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN (T.I.C.)».

[3] I. I. L. Miranda y D. M. M. Arroyo, «DISEÑO DE REPOSITORIO DIGITAL PARA USO EN INSTITUCIONES DE NIVEL SUPERIOR».

[4] «DSpace», *Wikipedia, la enciclopedia libre*. 14 de abril de 2024. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=DSpace&oldid=159446181

[5] «Open Journal Systems (OJS) - Software», Public Knowledge Project. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://pkp.sfu.ca/software/ojs/

[6] «EPrints», *Wikipedia, la enciclopedia libre*. 19 de febrero de 2024. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=EPrints&oldid=158297299

[7] «Agile Unified Process - EcuRed». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://www.ecured.cu/Agile_Unified_Process

[8] «Arquitectura de software: Qué es y qué tipos existen | OpenWebinars», OpenWebinars.net. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/

[9] W. C. huaman, «Los 10 patrones comunes de arquitectura de software», Medium. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://medium.com/@maniakhitoccori/los-10-patrones-comunes-de-arquitectura-de-software-d8b9047edf0b

[10] «Arquitectura de software», *Wikipedia, la enciclopedia libre*. 13 de agosto de 2024. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Arquitectura_de_software&oldid=161852918

[11] «SOA frente a microservicios: diferencia entre estilos arquitectónicos - AWS», Amazon Web Services, Inc. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/compare/the-difference-between-soa-microservices/

[12] «Los estilos arquitectónicos de software: un enfoque para el diseño de sistemas». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://es.linkedin.com/pulse/los-estilos-arquitect%C3%B3nicos-de-software-un-enfoque-el-rojas-mogoll%C3%B3n-ni0te

[13] I. Sommerville, *Ingenieria de Software*, 7ma ED.

[14] «Django», Django Project. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://www.djangoproject.com/

[15] «Welcome to Python.org», Python.org. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://www.python.org/

[16] P. G. D. Group, «PostgreSQL», PostgreSQL. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://www.postgresql.org/

[17] M. O. contributors Jacob Thornton, and Bootstrap, «Introduction». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://getbootstrap.com/docs/5.0/getting-started/introduction/

[18] «JavaScript | MDN». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript

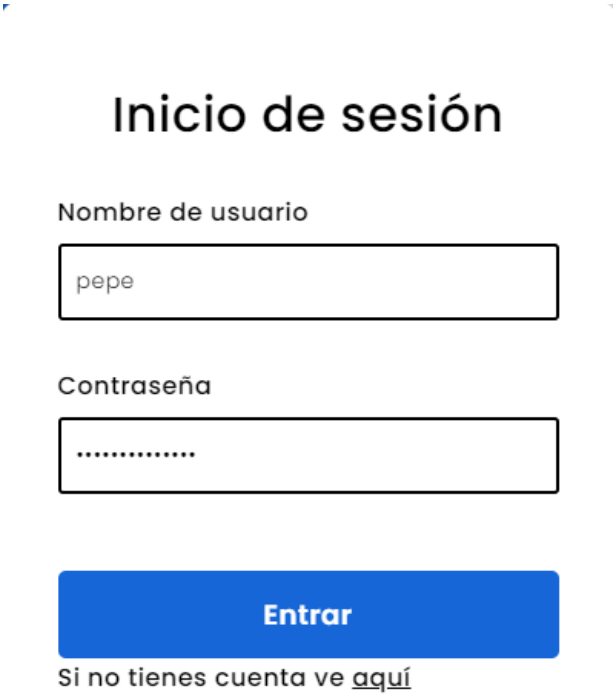
[19] «HTML: Lenguaje de etiquetas de hipertexto | MDN». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTML

[20] «CSS | MDN». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/CSS
[21] «Visual Studio Code - Code Editing. Redefined». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://code.visualstudio.com/
[22] «Full Lifecycle Modeling for Business, Software and Systems | Sparx Systems». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://sparxsystems.com/products/ea/
[23] «draw.io». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://app.diagrams.net/
[24] «Home - Pencil Project». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://pencil.evolus.vn/
[25] «Patrón MVT: Modelo-Vista-Template | Curso de Django | Hektor Profe». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://docs.hektorprofe.net/django/web-personal/patron-mvt-modelo-vista-template/
[26] «Tutorial de diagrama de clases UML», Lucidchart. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml
[27] «Patrones de diseño / Design patterns». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://refactoring.guru/es/design-patterns
[28] R. C. Mora, «Patrones de GRASP», Adictos al trabajo. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://adictosaltrabajo.com/2003/12/22/grasp/
[29] «Patrones Gof - EcuRed». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://www.ecured.cu/Patrones_Gof

[31] «Patrón Repositorio (Repository Pattern) y Unidad de Trabajo (Unit Of Work) en ASP.NET Core WebApi 3.0», DEV Community. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://dev.to/ebarrioscode/patron-repositorio-repository-pattern-y-unidad-de-trabajo-unit-of-work-en-asp-net-core-webapi-3-0-5goj
[32] «Mediador | Marco de Desarrollo de la Junta de Andalucía». Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/194
[33] «Pruebas de software», *Wikipedia, la enciclopedia libre*. 22 de agosto de 2024. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Pruebas_de_software&oldid=162020754
[34] «Caja blanca (sistemas)», *Wikipedia, la enciclopedia libre*. 11 de noviembre de 2023. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Caja_blanca_(sistemas)&oldid=155259843
[35] «Complejidad ciclomática», *Wikipedia, la enciclopedia libre*. 23 de abril de 2024. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Complejidad_ciclom%C3%A1tica&oldid=159641415
[36] «Caja negra (sistemas)», *Wikipedia, la enciclopedia libre*. 17 de noviembre de 2023. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Caja_negra_(sistemas)&oldid=155443936
[37] «¿Qué son las pruebas funcionales? Tipos, ejemplos, lista de comprobación y aplicación», https://www.zaptest.com/es. Accedido: 1 de septiembre de 2024. [En línea]. Disponible en: https://www.zaptest.com/es/que-son-las-pruebas-funcionales-tipos-ejemplos-lista-de-comprobacion-y-aplicacion

Anexos

HU: Iniciar sesión

Número:1	**Requisito:** Iniciar sesión.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:10 horas**
Riesgo en desarrollo: Bajo	**Tiempo Real:**
Descripción: El componente permitirá que usuario introduzca sus datos, el sistema lo comprueba y si son válidos, el usuario queda autenticado con el nivel de privilegios asignados; si los datos no son válidos el sistema muestra un mensaje de error. **Username:** Campo de texto donde se añade el usuario de la persona que va a entrar a la parte administrativa. **Password:** Campo de texto donde se añade la contraseña del usuario entrado anteriormente.	
Observaciones: Si el usuario introduce la información de forma correcta, el sistema entra a la parte administrativa de la aplicación.	
Prototipo elemental de interfaz gráfica de usuario: 	

HU: Añadir usuario

Número:2	**Requisito:** Añadir usuario.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:9 horas**
Riesgo en desarrollo: Medio	**Tiempo Real:**

Descripción: El componente permitirá que usuario rol administrador crear un nuevo usuario.

Username: Campo de texto donde se añade el usuario de la persona que va a entrar a la parte administrativa.

Password: Campo de texto donde se añade la contraseña del usuario entrado anteriormente.

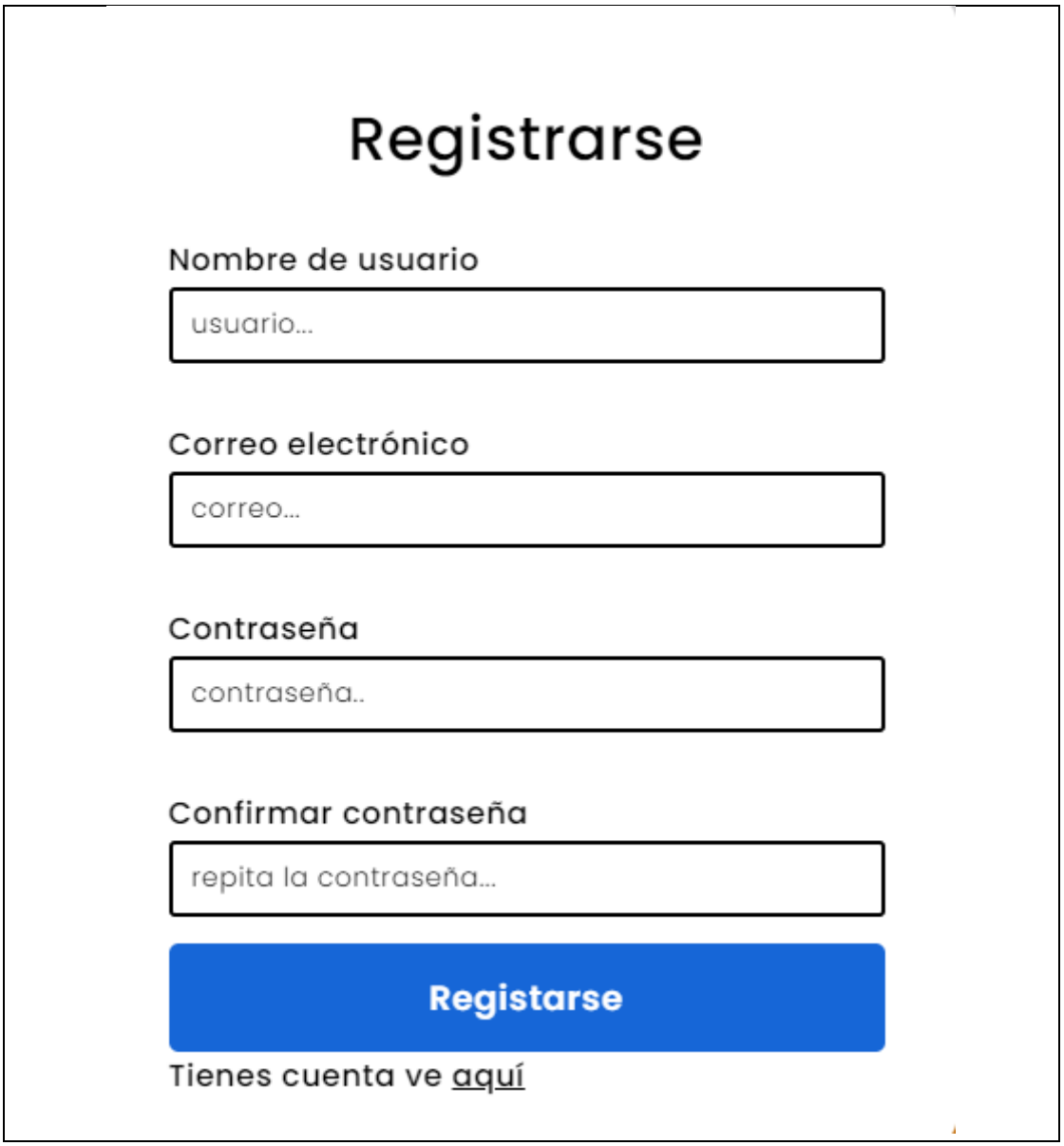
Password confirmation: Campo de texto donde confirmas la contraseña entrada anterior.

Observaciones: El administrador introduce un usuario con los siguientes requisitos Obligatorio. 150 caracteres o menos. Sólo letras, dígitos y @/./+/-/_.

En la contraseña debe cumplir los siguientes requisitos:

- Tu contraseña no puede ser demasiado parecida a tus otros datos personales.
- Tu contraseña debe contener al menos 8 caracteres.
- Su contraseña no puede ser una contraseña de uso común.
- Tu contraseña no puede ser totalmente numérica.

Prototipo elemental de interfaz gráfica de usuario:



HU: Modificar usuario

Número:3	**Requisito:** Modificar usuario.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:8 horas**
Riesgo en desarrollo: Medio	**Tiempo Real:**
Descripción: El componente permitirá que usuario rol administrador pueda modificar el usuario y darle los permisos deseado a cada usuario que va a tener acceso para modificar el sistema.	

Username: Campo de texto donde se añade el usuario de la persona que va a entrar a la parte administrativa.

Password: Campo de selección donde seleccionas un formulario para poder cambiar la contraseña del usuario al que se está modificando.

Personal info: Aquí podrás entrar la información personal del usuario.

- **First name:** Campo de texto donde se añade el nombre del usuario.
- **Last name:** Campo de texto donde se añade el apellido del usuario.
- **Email address:** Campo de texto donde se añade correo del usuario.

Permissions: En este apartado podrás entra los permisos del usuario.

- **Active:** Campo de selección donde se designa si este usuario debe ser tratado como activo. Deseleccione esta opción en lugar de eliminar cuentas.
- **Staff status:** Campo de selección donde se indica si el usuario puede iniciar sesión en este sitio de administración.
- **Superuser status:** Campo de selección donde se designa que este usuario tiene todos los permisos sin asignarlos explícitamente.

Groups: Aquí podrás poner en que grupo va a estar el usuario y al cual no quieres que pertenezca.

- **Available groups:** Campo de selección donde seleccionas dentro de los grupos disponibles el grupo al que pertenece el usuario.
- **Chosen groups:** Campo de selección donde ves los grupos elegidos y te da la opción para quitar el grupo que desees al q no pertenezca el usuario.

User permissions: Aquí podrás poner y quitar los permisos deseados para el usuario.

- **Available user permissions:** Campo de selección donde seleccionas los permisos q va a tener el usuario.
- **Chosen user permissions:** Campo de selección donde seleccionas podrás quitar permisos q tiene el usuario.

Important dates: En este apartado tendrás las fechas importantes.

Last login: Campo de selección donde podrás entrar el ultimo acceso.

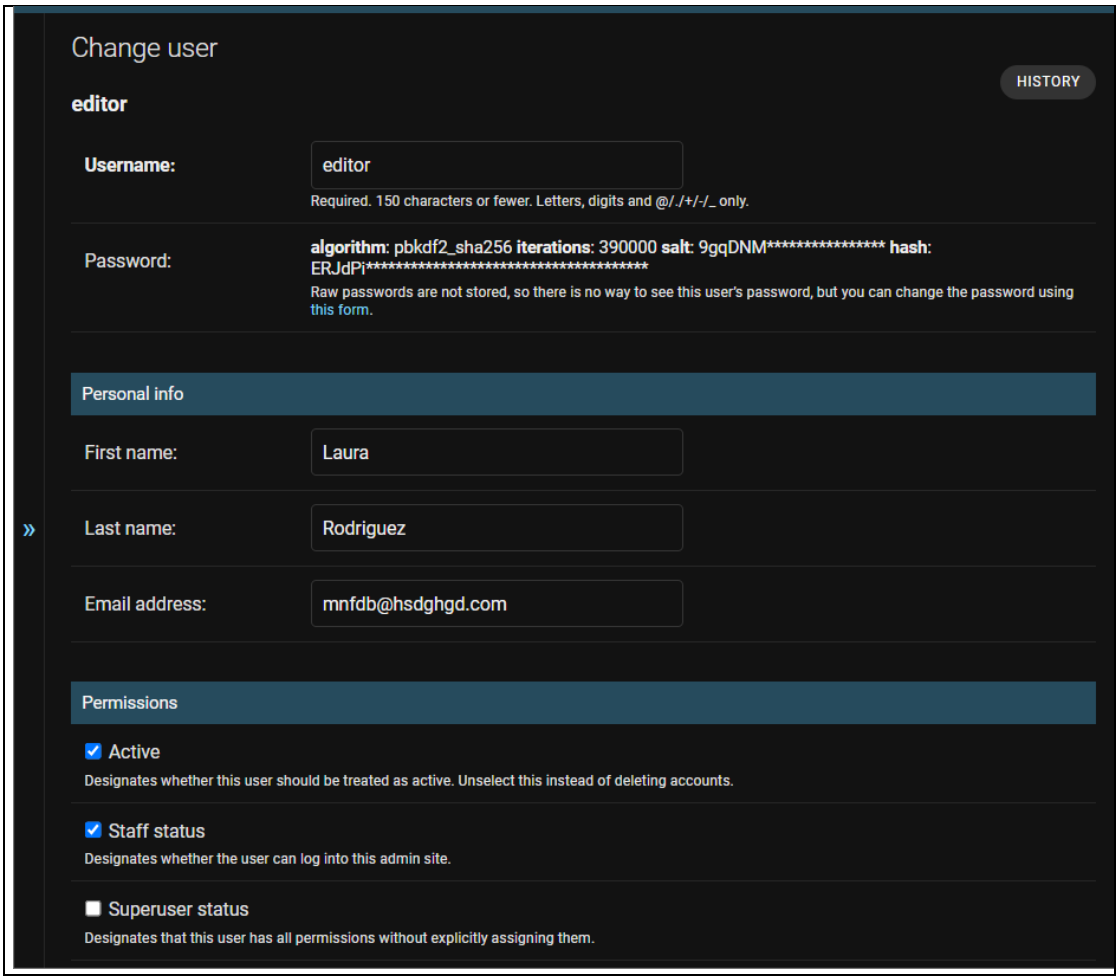
- **Date:** Campo de selección donde entraras la fecha.
- **Time:** Campo de selección donde entras la hora.

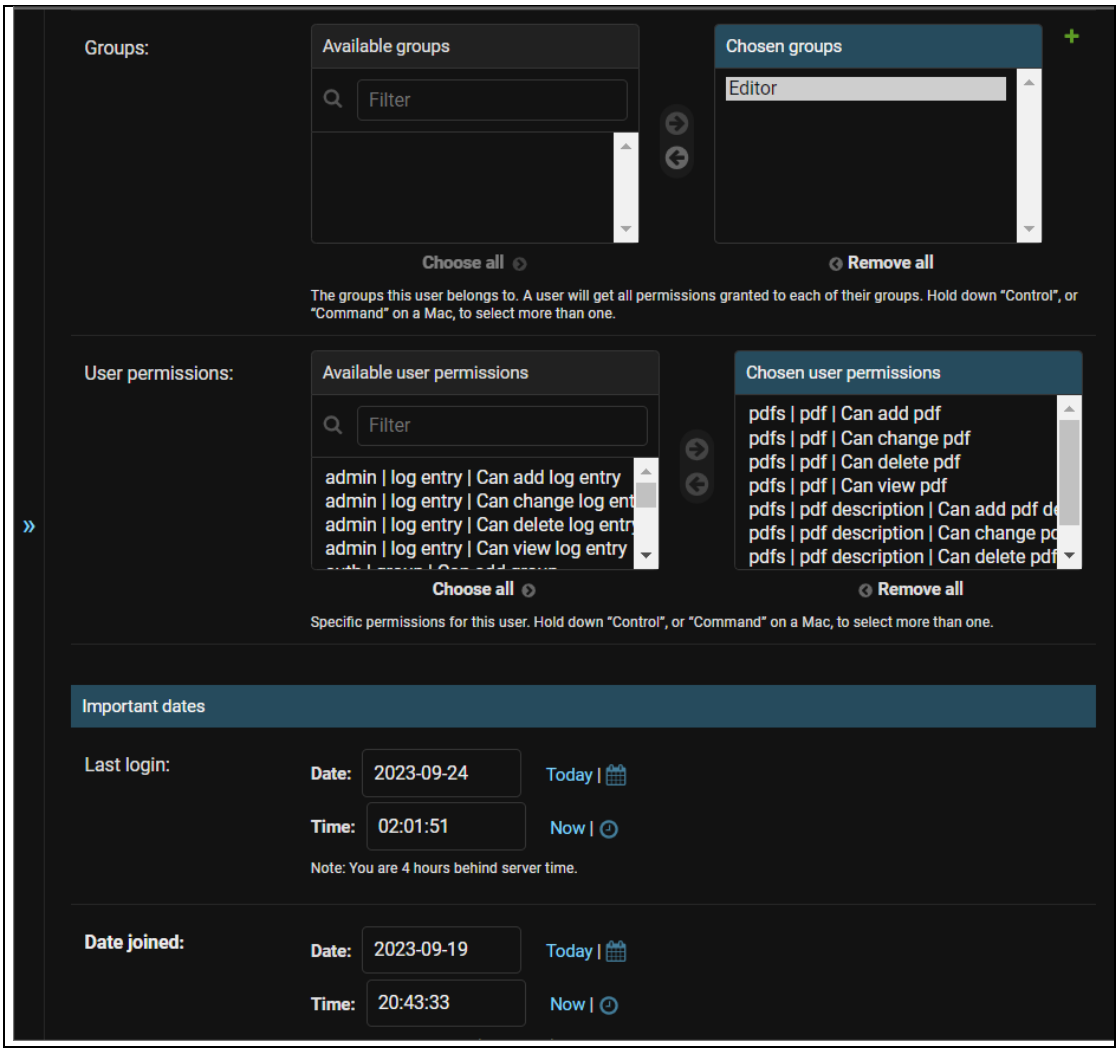
Date joined: Campo de selección donde podrás entrar la fecha de incorporación.

- **Date:** Campo de selección donde entraras la fecha.
- **Time:** Campo de selección donde entras la hora.

Observaciones: El administrador introduce y modifica del usuario deseado el usuario, la contraseñas, los permisos, los grupos y las fechas importantes deseadas.

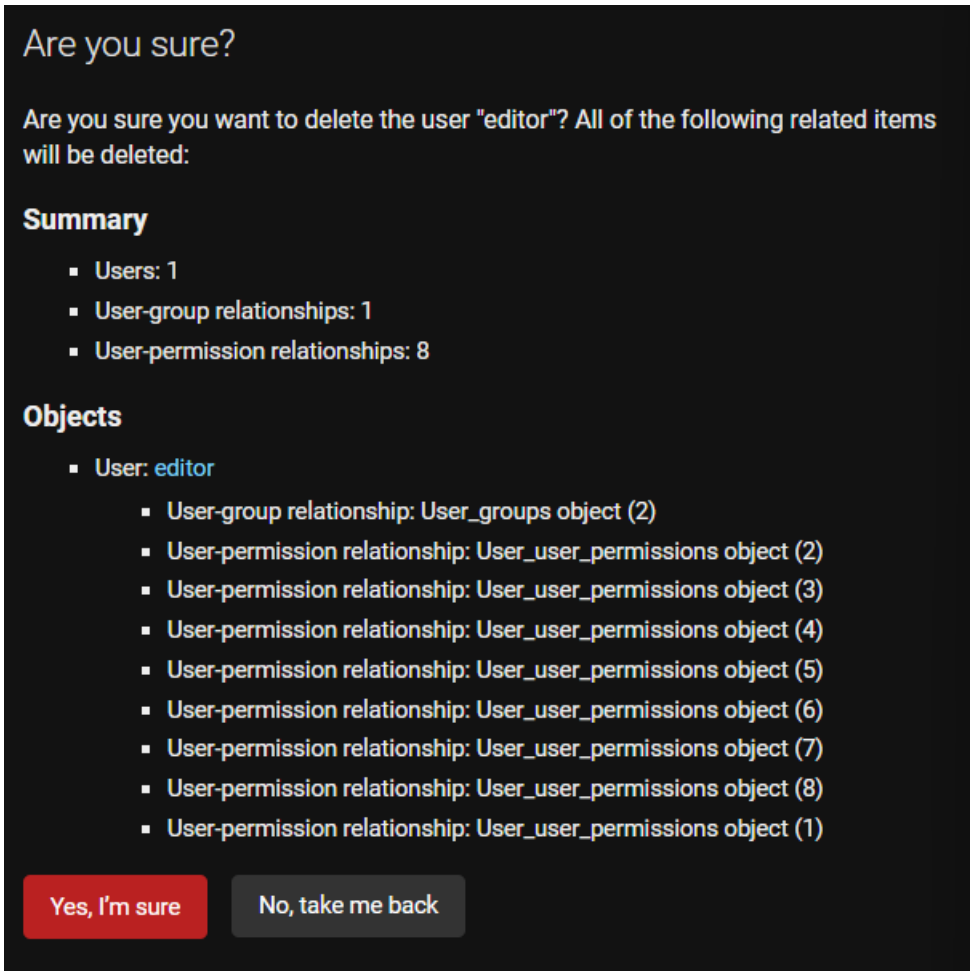
Prototipo elemental de interfaz gráfica de usuario:


Change user
HISTORY
editor
Username:
editor
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
Password:
algorithm: pbkdf2_sha256 iterations: 390000 salt: 9gqDNM********** hash: ERJdPi**********
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.
Personal info
First name:
Laura
Last name:
Rodriguez
Email address:
mnfdb@hsdghgd.com
Permissions
Active
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
Staff status
Designates whether the user can log into this admin site.
Superuser status
Designates that this user has all permissions without explicitly assigning them.



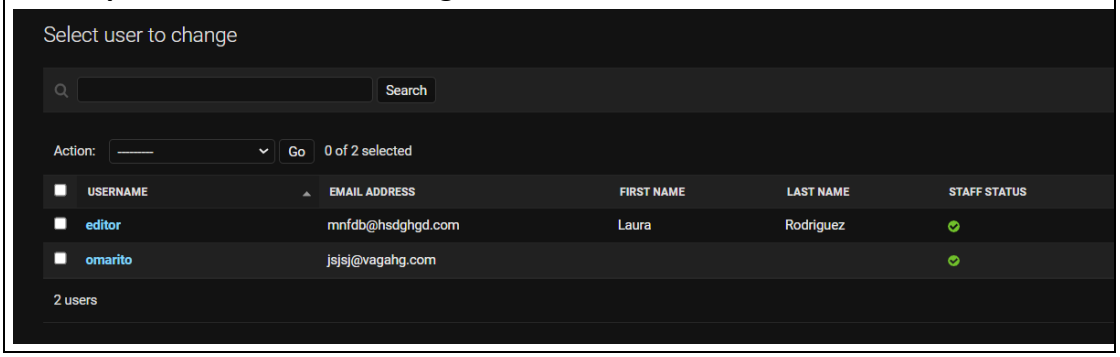
HU: Eliminar usuario

Número:4	**Requisito:** Eliminar usuario.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:10 horas**
Riesgo en desarrollo: Bajo	**Tiempo Real:**
Descripción: El componente permitirá que usuario rol administrador pueda eliminar el usuario deseado. **Yes, I´m sure:** Campo de selección donde seleccionas que si estás seguro de eliminar el usuario. **No, Take me back:** Campo de selección donde seleccionas que no quieres eliminar el usuario u te lleva de vuelta a donde están todos los usuarios.	

Observaciones: Si el administrador selecciona q si está seguro de eliminar el usuario este lo elimina de forma permanente, pero si marca la opción de no este lo regresa a la acción de modificar usuario.
Prototipo elemental de interfaz gráfica de usuario: 

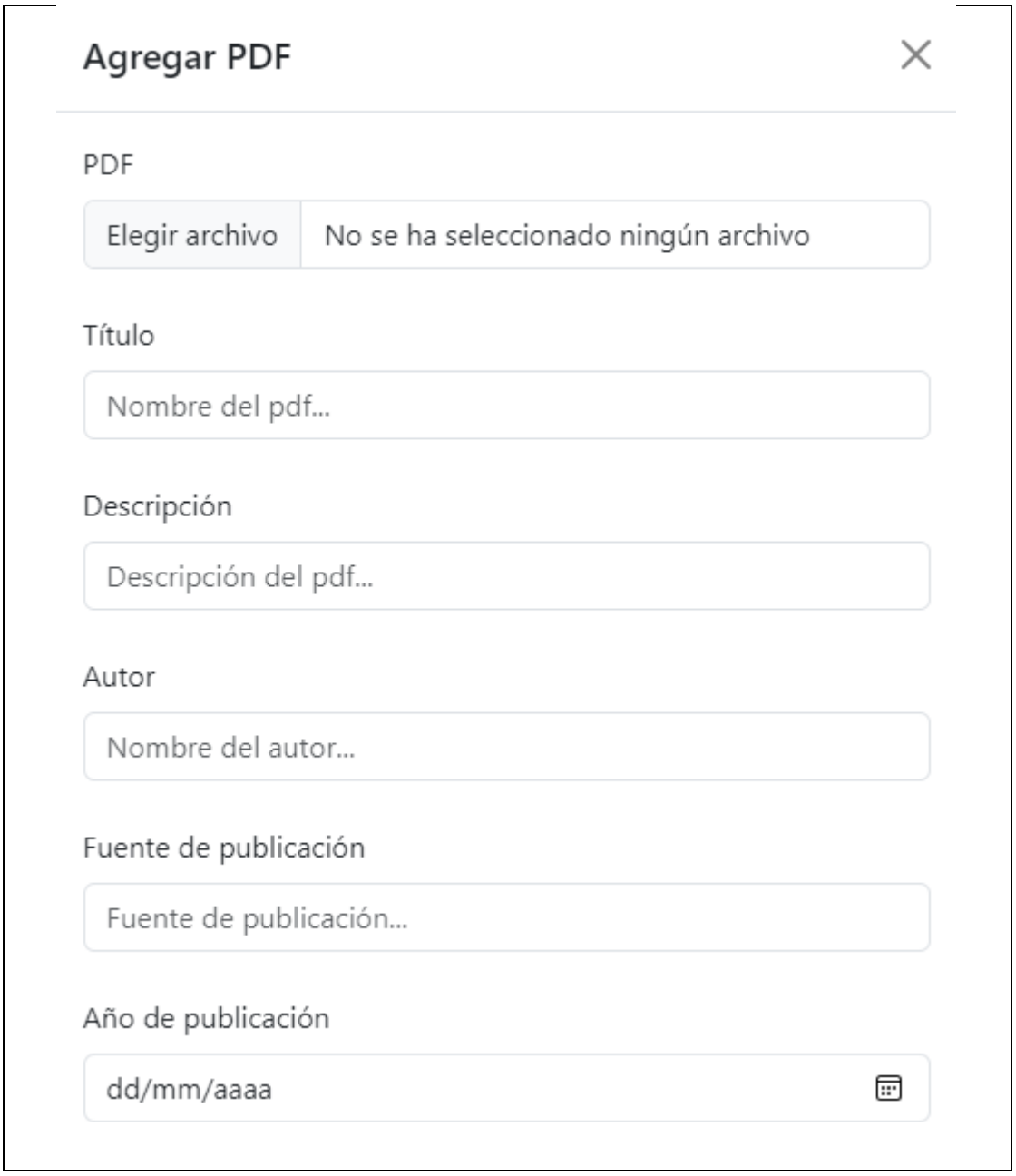
HU: Listar usuario

Número:5	**Requisito:** Listar usuario.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Baja	**Tiempo Estimado:10 horas**
Riesgo en desarrollo: Bajo	**Tiempo Real:**
Descripción: El componente permitirá que usuario rol administrador pueda seleccionar el usuario deseado. **Search:** Campo de texto donde pones el nombre del usuario q deseas buscar. **Action:** Campo de selección donde seleccionas la acción deseada.	

Observaciones: Si el administrador selecciona en la casilla al usuario deseado y en acción realiza la acción deseada a ese usuario seleccionado en caso de no tener ningún usuario seleccionado el sistema te manda un mensaje de q debes tener algún usuario seleccionado pata poder realizar alguna acción.
Prototipo elemental de interfaz gráfica de usuario: 

HU: Agregar Artículo científico

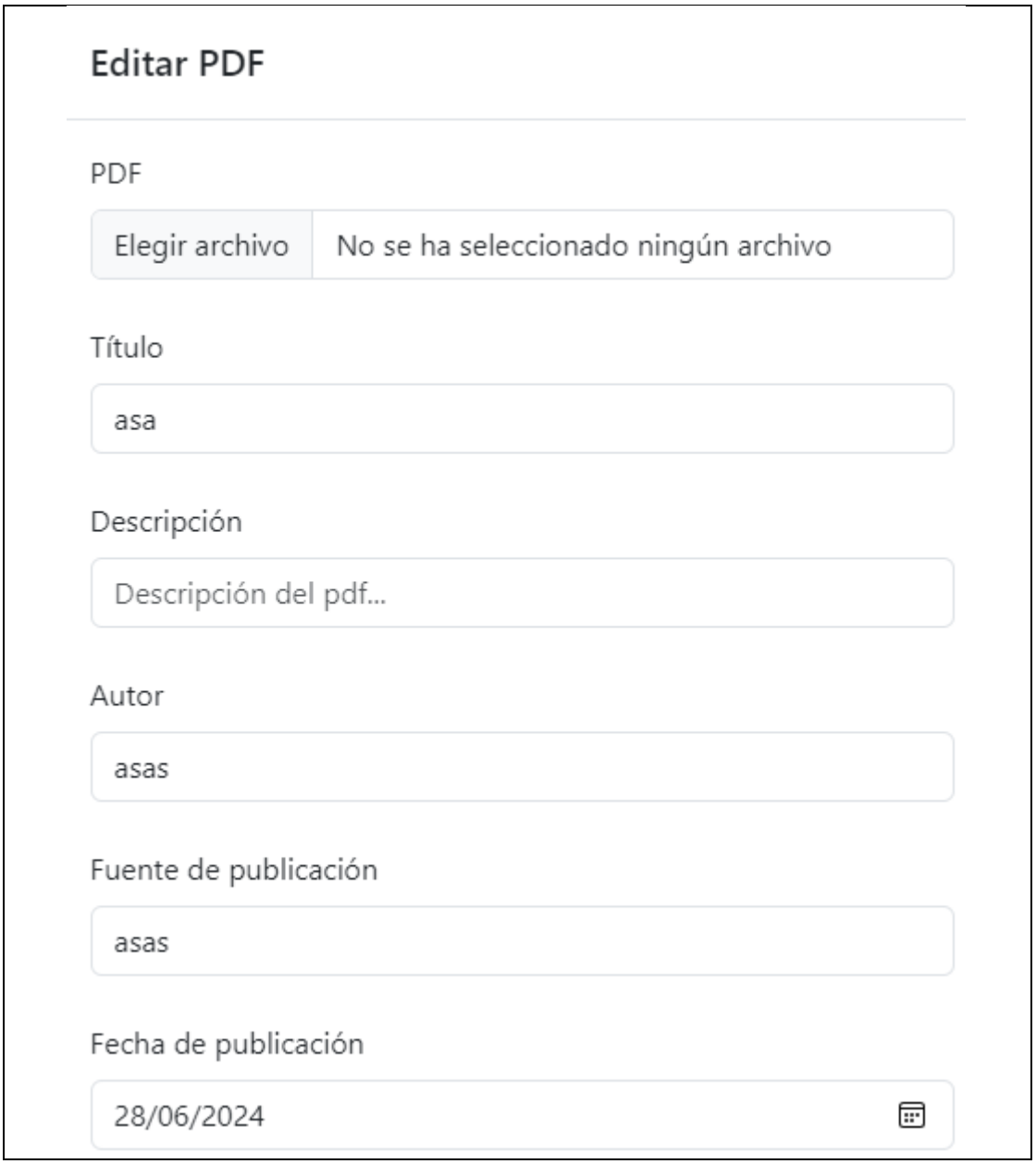
Número:6	**Requisito:** Agregar Artículo científico.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:10 horas**
Riesgo en desarrollo:	**Tiempo Real:**
Descripción: El componente permitirá que usuario rol administrador y editor podrá poner el título del archivo, seleccionar al archivo deseado, poner el autor. La fuente de publicidad, el año y el país. **Title:** Campo de texto donde el usuario entrara el nombre del archivo deseado. **Pdf file:** Campo de selección donde podrás seleccionar el archivo deseado. **Authors:** Campo de texto donde el usuario entrara el autor del archivo deseado. **Published source:** Campo de texto donde el usuario entrara la fuente de publicidad del archivo deseado. **Year:** Campo de texto donde el usuario entrara el año del archivo deseado. **Country:** Campo de texto donde el usuario entrara el nombre del país del archivo deseado.	
Observaciones: El administrador o editor no podrá dejar la casilla del Title vacía ni podrá guardar esta información si no ha seleccionado ningún archivo.	
Prototipo elemental de interfaz gráfica de usuario:	



HU: Modificar Artículo científico

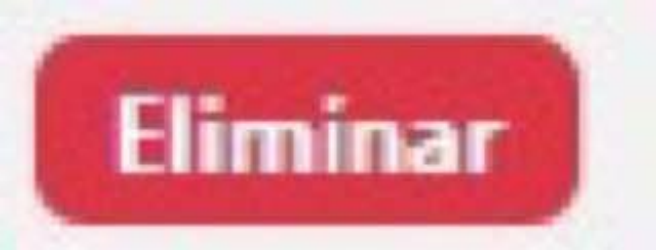
Número:7	**Requisito:** Modificar artículo científico.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Media	**Tiempo Estimado:10 horas**
Riesgo en desarrollo: Bajo	**Tiempo Real:**

Descripción: El componente permitirá que usuario rol administrador y editor pueda modificar el título, el año, el autor, el país, fuente de publicidad y seleccionar otro PDF. **Title:** Campo de texto donde el usuario entrara el nombre del archivo q desea editar. **Pdf file:** Campo de selección donde seleccionas el archivo PDF que deseas cambiar por el actual. **Authors:** Campo de texto donde el usuario editar el autor del archivo deseado. **Published source:** Campo de texto donde el usuario entrara la fuente de publicidad del archivo que desea editar. **Year:** Campo de texto donde el usuario edita el año del archivo deseado. **Country:** Campo de texto donde el usuario entrara el nombre del país del archivo editado.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

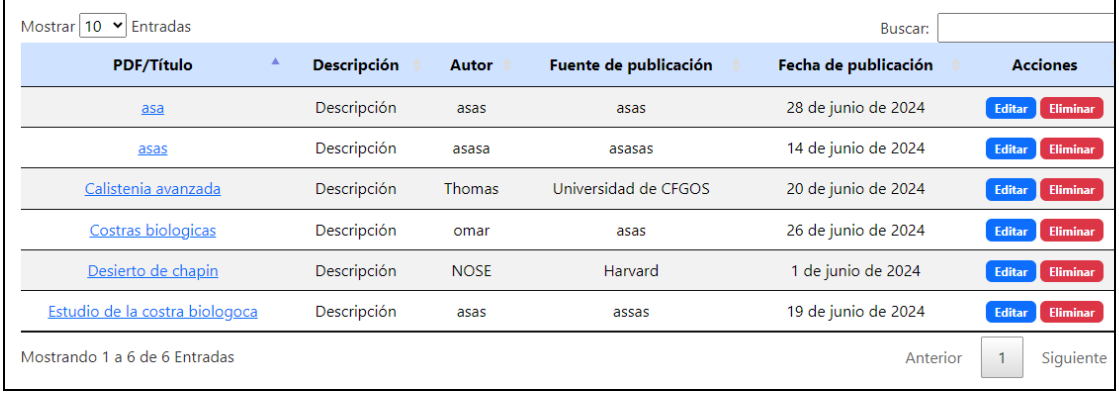


HU: Elimina Artículo científico

Número:8	**Requisito:** Eliminar Artículo científico.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:10 horas**
Riesgo en desarrollo: Bajo	**Tiempo Real:**
Descripción: El componente permitirá que usuario rol administrador pueda eliminar el PDF deseado.	

Al presionar el botón eliminar saldrá un anuncio donde te permitirá elegir **SI**, si deseas eliminar o **NO** sino quieres eliminar el PDF.
Observaciones: Si el administrador selecciona q si está seguro de eliminar el PDF este lo elimina de forma permanente, pero si marca la opción de no este lo regresa a la acción de modificar PDF.
Prototipo elemental de interfaz gráfica de usuario: 

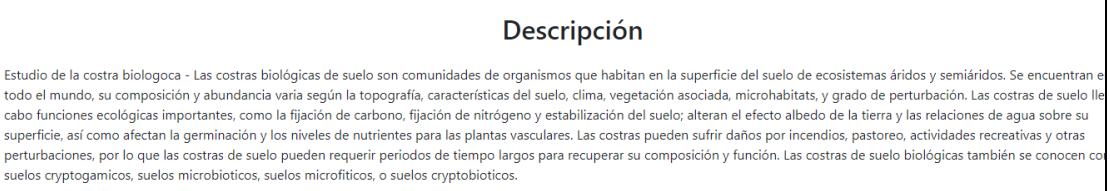
HU: Mostrar una lista de Artículos científicos

Número:9	**Requisito:** Mostrar una lista de artículos científicos.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:12 horas**
Riesgo en desarrollo:	**Tiempo Real:**
Descripción: El componente mostrar una lista de los archivos subidos a la aplicación.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

HU: Permitir descarga de archivos Artículo científico

Número:10	**Requisito:** Permitir la descarga de archivos artículo científico.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:12 horas**
Riesgo en desarrollo:	**Tiempo Real:**
Descripción: El componente permitirá descargar el archivo deseado **Descargar:** Campo de selección donde seleccionas si deseas descargar ese documento.	
Observaciones: Al darle en la opción de descargar el sistema te descarga el archivo deseado.	
Prototipo elemental de interfaz gráfica de usuario:	

HU: Mostrar descripción de un Artículo científico

Número:11	**Requisito:** Mostrar descripción un artículo científico
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:12 horas**
Riesgo en desarrollo:	**Tiempo Real:**
Descripción: El componente permitirá mostrar la descripción del archivo deseado. **Descargar:** Campo de selección donde seleccionas si deseas descargar ese documento.	
Observaciones: Te muestra la descripción del archivo deseado y te da la opción de descarga para que lo tenga guardado en su dispositivo.	
Prototipo elemental de interfaz gráfica de usuario: 	

HU: Permitir la búsqueda avanzada de los archivos artículo científico

Número:12	**Requisito:** Permitir la búsqueda avanzada de artículo científico
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:12 horas**
Riesgo en desarrollo:	**Tiempo Real:**
Descripción: El componente permitirá eliminar la descripción deseada.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

HU: Cerrar sesión

Número:13	**Requisito:** Cerrar sesión.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**
Prioridad: Media	**Tiempo Estimado:12 horas**
Riesgo en desarrollo:	**Tiempo Real:**
Descripción: El componente permitirá eliminar la descripción deseada.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

HU: Cambio de contraseña

Número:14	**Requisito:** Cambio de contraseña.
Programador: Elizabeth Medina Muñoz.	**Iteración Asignada:1**

Prioridad: Media	**Tiempo Estimado:12 horas**
Riesgo en desarrollo:	**Tiempo Real:**
Descripción: El componente permitirá cambiar la contraseña del usuario autenticado actualmente. **Contraseña antigua:** Campo de escritura donde pondrás la contraseña antigua. **Contraseña nueva:** Campo de escritura donde pondrás la contraseña nueva. **Contraseña nueva (confirmación):** Campo de escritura donde pondrás la misma contraseña escrita anteriormente.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: