UNIVERSIDAD Đ CIENFUEGOS
INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA

Universidad de Cienfuegos

“Carlos Rafael Rodríguez”

Facultad de Ingeniería

Título:

Repositorio Digital de Costras Biológicas del Suelo

Trabajo de diploma para optar por el título de

Ingeniero Informático

Autor:

Omar Luis Gutiérrez Pérez

Tutor:

MSc. Richard Darian Sánchez Rivero

Cienfuegos, Cuba

Curso: 2023

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad Cienfuegos los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

_____ _____

(Autor) (Autor)

_____ _____

(Tutor) (Cotutor)

AGRADECIMIENTOS

Quiero expresar mi profundo agradecimiento a todas las personas que han contribuido de manera significativa en la realización de esta tesis, brindándome su apoyo, orientación y aliento a lo largo de este arduo proceso.

En primer lugar, deseo expresar mi más sincero agradecimiento a mi tutor MSc. Richard Darian Sánchez Rivero, por su invaluable guía y conocimientos expertos. Su compromiso y dedicación fueron fundamentales para el desarrollo de esta investigación. Agradezco sinceramente su tiempo, paciencia y sus valiosas sugerencias, las cuales han enriquecido enormemente este trabajo.

También quiero agradecer a todos los profesores y personal académico del Departamento de Informática de la Facultad de Ingeniería de la Universidad de Cienfuegos quienes han compartido sus conocimientos y experiencias a lo largo de mi formación académica. Sus enseñanzas y apoyo han sido fundamentales para mi crecimiento profesional.

Mi más sincero agradecimiento también va dirigido a mi familia y amigos, quienes han estado a mi lado en todo momento, brindándome su amor, comprensión y apoyo incondicional. Su aliento y palabras de aliento me han dado fuerzas para superar obstáculos y alcanzar mis metas académicas.

DEDICATORIA

Dedico este trabajo a aquellas personas que han sido el motor y la inspiración en mi camino hacia la realización de esta tesis.

En primer lugar, dedico este trabajo a mi madre, Mariela Gutiérrez Pérez. Su amor incondicional, apoyo constante y sacrificios han sido la base de mi educación y desarrollo, a mis abuelos en especial por apoyarme día a dio, a mi esposa y demás familiares allegados por apoyarme y darme las fuerzas por seguir adelante Gracias por creer en mí y por ser mi fuente de fortaleza en cada paso que he dado.

RESUMEN

En la actualidad la gestión de la información es un problema muy común por los grandes volúmenes de información. La facultad de Agronomía de la Universidad de Cienfuegos presenta dificultades en este tipo de trabajos por no poseer un software informático capaz de facilitarle esta tarea de recopilar y organizar esta información por lo q surgen dificultades a la hora de obtener la información deseada. El objetivo de esta investigación es desarrollar repositorio digital sobre las costras biológicas del suelo, que permita su almacenamiento, gestión y acceso de dicha información.
Para el desarrollo de este trabajo se utilizó Django un framework web 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.

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

Abstract

Nowadays, information information management is a very common problem due to the large volumes of information. large volumes of information. The Faculty of Agronomy of the University of Cienfuegos presents difficulties in this type of work because it does not have computer software capable of facilitating the task of compiling and organizing this information. information, so difficulties arise when it comes to obtaining the desired information. desired information. The objective of this research is to develop digital repository on the biological soil crusts, which allows its storage, management and storage, management and access to this information.

For the Django, a high-level web framework that encourages rapid development and clean and pragmatic design, was used for the development of this work. pragmatic design. Built by experienced developers, it takes care of much of the complications of web development much of the hassle of web development, so you can focus on writing your application without writing your application without reinventing the wheel.

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

II | Índice:

IV | Índice de tabla:

v | Índice de figura:

Introducción

Actualmente vivimos en un mundo en constante cambio, favorecido principalmente por los avances tecnológicos - conocidos como Tecnologías de la Información y las Comunicaciones (TIC's)- que se han integrado a todos los sectores de la sociedad generando profundos cambios en el modo de gestionar el almacenamiento, la organización y el acceso a la información tanto impresa como digital. Los avances de las TIC's propiciaron el nacimiento del movimiento de Acceso Abierto (AA). Este movimiento ofrece la posibilidad de consultar un documento científico de forma libre y gratuita; se encuentra en continuo crecimiento a través del incremento de iniciativas y proyectos que surgen bajo el concepto de AA.[1]

La concepción que se tiene de bibliotecas tradicionales es ofrecer documentos (libros, revistas, tesis, artículos, etc.) en soportes físicos a través de servicios de préstamos y consultas. Esas políticas limitaban de alguna manera con el acceso a los documentos por la falta de ejemplares o la existencia de ejemplares desactualizados. No obstante, junto al incremento de los recursos informáticos, la Internet y el descenso de los costos para adquirir esos recursos y servicios relacionados, potenciaron en los últimos 20 años el diseño y la creación de las Bibliotecas Digitales (BD), es decir, se inició la automatización de las bibliotecas tradicionales, con un crecimiento sostenido y en constante evolución.[2]

Internet, desarrollado a partir de los años 70, creció paulatinamente hasta principios de los años 90 cuando se le liberalizó el acceso, provocando un salto en el número de usuarios y en el tipo y cantidad de información que se intercambiaba (el siguiente gran salto, mucho más importante, ocurrió unos años después cuando se introdujo la World Wide Web). Los primeros usuarios de Internet –autores-investigadores de universidades y de otras instituciones de investigación– no tardaron en experimentar con el nuevo entorno para agilizar el intercambio de ideas y datos, utilizando para ello las nuevas herramientas como correo electrónico y los protocolos FTP (para la transferencia de ficheros) y telnet (para la conexión a ordenadores remotos).[3]

El movimiento por el acceso abierto (open access) persigue la difusión libre y gratuita de la producción científica, es decir, los contenidos derivados de la investigación. Esta iniciativa se está ampliando al material docente, a los denominados recursos educativos en abierto (OER), el valor de los cuales se ha visto acrecentado por el Espacio Europeo de Educación Superior (EEES) y, en especial, por un nuevo modelo pedagógico que promueve que docentes y estudiantes usen, reutilicen y compartan recursos con el resto de la comunidad educativa. Los OERs tienen sus inicios en 2001, cuando el Massachusetts Institute of Technology (MIT) creó el programa OpenCourseWare (OCW) y desde entonces el interés por ellos ha ido aumentando. En estos once años los contenidos abiertos para la educación han seguido dos fases de desarrollo: la inicial, focalizada en proporcionar acceso a los contenidos y la actual, más preocupada por el uso por su incorporación en las prácticas educativas.[4]

En 2006, el estudio de la ARL mostró que el 78% de las 87 bibliotecas universitarias que participaban en la encuesta tenían un repositorio ya en marcha o lo tendría instalado entre 2006 y 2007. Entre las bibliotecas universitarias norteamericanas, se detectó un incremento notable del número de instalaciones de software para repositorios a partir de 2004 como, por

ejemplo, Dspace, desarrollado por el MIT y Hewlett-Packard y Eprints, por la University of Southampton (Association of Research Libraries, 2006).[3]

El desarrollo de una costra en la superficie del suelo reduce drásticamente la intensidad de infiltración, provoca la disminución del almacenaje del agua en el suelo, desencadena la escorrentía superficial y por tanto la erosión. Por otra parte los fenómenos de encostrado pueden impedir la germinación de las semillas y el desarrollo de los cultivos.[5]

La formación de una costra superficial es el resultado de la ruptura de los agregados y de la reorganización de diversos componentes en el seno de la misma, obteniéndose una estructura que contrasta con la organización de los materiales en un suelo no encostrado. Por analogía con la descripción de los horizontes y los procesos de edafogénesis en el seno de un perfil, la descripción de las características morfológicas de una costra ha sido utilizada para explicar el comportamiento de la misma, e interpretar los procesos que llevan a su formación. Esta idea se encuentra ya en los estudios pioneros de los fenómenos de encostrado y por esta razón desde entonces se describieron una cantidad importante de características morfológicas que pueden relacionarse con el proceso de formación de una costra.[5]

En la actualidad, las universidades generan grandes cantidades de datos e información que son cruciales para su funcionamiento y para la toma de decisiones estratégicas. Esta información puede incluir desde datos académicos y administrativos hasta investigaciones y proyectos realizados por la institución y sus miembros. En este contexto, la necesidad de contar con un sistema eficaz para el almacenamiento y gestión de esta información se ha vuelto cada vez más relevante. En este trabajo se presenta la creación de un repositorio de costras biológicas del suelo para la que universidad tenga todos sus datos.

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:
Como gestionar la información de documentos académicos relacionados con las costras biológicas del suelo.

Objeto de estudio:
Las herramientas para gestionar repositorios de ficheros.

Campo de acción:
Programas pare gestionar repositorios de ficheros

Objetivo general:
Desarrollar repositorio digital sobre las costras biológicas del suelo, que permita el almacenamiento, gestión, acceso de la información.

Objetivos específicos:

- Diseñar la arquitectura del repositorio digital de costras biológicas del suelo, incluyendo la selección de la plataforma y herramientas adecuadas.
- Implementar el sistema de repositorio digital de costras biológicas del suelo en la Universidad.
- Evaluar el desempeño del repositorio digital de costras biológicas del suelo, a través de pruebas de usabilidad y eficiencia, y realizar las mejoras necesarias.

Tareas de investigación:

- Entrevistas al personal revisión de la literatura científica sobre las costras biológicas del suelo, incluyendo su composición, formación, funciones y su impacto en la calidad del suelo y la productividad agrícola.
- Investigación sobre las tecnologías y herramientas disponibles para la creación de repositorios digitales, incluyendo la selección de la plataforma y las herramientas adecuadas para la creación del repositorio digital de costras biológicas del suelo.
- Modelación de datos necesarios para almacenar la información sobre las costras biológicas del suelo, 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 costras biológicas del suelo.
- Implementación de la funcionalidad de búsqueda para permitir a los usuarios buscar y filtrar la información sobre las costras biológicas del suelo.
- Realización de pruebas de validación para verificar el correcto funcionamiento del repositorio digital de costras biológicas del suelo.

Idea a defender:

Con la creación de un repositorio digital para la gestión de la información sobre las costras biológicas del suelo es fundamental para mejorar la capacidad de la institución para llevar a cabo investigaciones y proyectos relacionados con la agricultura y la ecología del suelo. El repositorio digital permitirá la recopilación, organización y acceso eficiente a la información sobre las costras biológicas del suelo, lo que facilitará la toma de decisiones informadas y mejorará la productividad agrícola y la calidad del suelo.

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: Se empleó para identificar características de los fenómenos naturales o sociales relacionados con las costras biológicas del suelo, como su distribución, composición y estructura. De igual manera se empleó para identificar las carencias más significativas en este proceso para el guardado de la información.

- Entrevista: Se utilizó para obtener información acerca de cómo se desarrolla el proceso de gestión de la información de las costras biológicas del suelo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se lleva a cabo la fundamentación teórica del tema que se va desarrollar. Se realiza un estudio de los sistemas existentes relacionados con el campo de acción y el objeto de estudio. También se detallan los lenguajes y herramientas a utilizar en las que se basa el trabajo.

1.2 Descripción del dominio del problema

La gestión eficiente de la información sobre las costras biológicas del suelo es fundamental para lograr una producción agrícola sostenible y de alta calidad. Estas son una capa protectora que se forma en la superficie y desempeñan un papel crítico en la retención de agua y nutrientes en el suelo, así como en la prevención de la erosión y la desertificación.
La falta de un repositorio digital que recopile y organice la información sobre las costras biológicas del suelo en un solo lugar representa un desafío significativo para la realización de investigaciones y proyectos relacionados con la agricultura y la ecología del suelo. Sin una herramienta eficiente para la gestión de la información sobre las costras biológicas del suelo, la toma de decisiones informadas se ve obstaculizada y se corre el riesgo de perder datos valiosos. Por lo tanto, es fundamental contar con un repositorio digital que permita la gestión, almacenamiento y acceso eficiente a la información sobre las costras biológicas del suelo.
Un repositorio digital de información sobre las costras biológicas del suelo permitiría la recopilación, organización y acceso eficiente a la información valiosa para la realización de investigaciones y proyectos relacionados con la agricultura y la ecología del suelo. Además, un repositorio digital facilita la toma de decisiones informadas y reduce el riesgo de pérdida de datos valiosos. La disponibilidad de información centralizada y fácilmente accesible puede mejorar significativamente la capacidad de una institución para llevar a cabo investigaciones y proyectos en el campo de la agricultura y la ecología del suelo.

1.3 Metodología de desarrollo de software

1.3.1 Metodología AUP

AUP define 7 disciplinas (4 ingenieriles y 3 de gestión de proyectos), las disciplinas son: [6]

1. Modelo. El objetivo de esta disciplina es entender el negocio de la organización, el problema de dominio que se abordan en el proyecto, y determinar una solución viable para resolver el problema de dominio. Agrupa los flujos de trabajos de Modelado de negocio, Requisitos y Análisis y Diseño.

2. Implementación. El objetivo de esta disciplina es transformar su modelo (s) en código ejecutable y realizar un nivel básico de las pruebas, en particular, la unidad de pruebas.

3. Prueba. El objetivo de esta disciplina consiste en realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificando que se cumplan los requisitos.

4. Despliegue. El objetivo de esta disciplina es la prestación y ejecución del sistema y que el mismo este a disposición de los usuarios finales.

5. Gestión de configuración. El objetivo de esta disciplina es la gestión de acceso a herramientas de su proyecto. Esto incluye no sólo el seguimiento de las versiones con el tiempo, sino también el control y gestión del cambio para ellos.

6. Gestión de proyectos. El objetivo de esta disciplina es dirigir las actividades que se lleva a cabo en el proyecto. Esto incluye la gestión de riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos, etc.), coordinación con el personal y los sistemas fuera del alcance del proyecto para asegurarse de que es entregado a tiempo y dentro del presupuesto.

7. Entorno. El objetivo de esta disciplina es apoyar el resto de los esfuerzos por garantizar que el proceso sea el adecuado, la orientación (normas y directrices), y herramientas (hardware, software, etc.) estén disponibles para el equipo según sea necesario.

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso Narrativos(CUN), Diagramas de Procesos de Negocio(DPN) o Modelo Conceptual(MC)) y existen tres formas de encapsular los requisitos (Casos de Uso Sintéticos(CUS), Historias de Usuario(HU), Documentos de Requisitos del Proyecto (DRP)), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:[6]

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.

Decidimos usar el esenario No 4 donde modelamos un sistema con Historias de Usuarri (HU).

1.4 Arquitectura de software

La arquitectura de software de un sistema informático comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos. La arquitectura no es el software operacional. Más bien, es la representación que capacita al ingeniero del software para:[7]

- Analizar la efectividad del desafío para la consecución de los requisitos fijados.
- Considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el desafío es relativamente fácil.
- Reducir los riesgos asociados a la construcción del software.

La arquitectura de software es importante por las 3 razones siguientes:[7]

- Facilitan la comunicación entre todas las partes (partícipes) interesadas en el desarrollo de un sistema basado en computadora.
- Destaca decisiones tempranas de desafío que tendrán un profundo impacto en todo el trabajo de ingeniería del software que sigue, y es tan importante en el éxito final del sistema como una entidad operacional.
- Constituye un modelo relativamente pequeño e intelectualmente comprensible de cómo debe estar estructurado el sistema y cómo trabajan juntos sus componentes.

El modelo arquitectónico y los patrones arquitectónicos contenidos dentro son transferibles. Esto es, los estilos y patrones de arquitectura pueden ser aplicados en el desafío de otros sistemas y representados a través de un conjunto de abstracciones que facilitan a los ingenieros del software la descripción de la arquitectura de un modo predecible.[7]

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:[7]

- 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.

Otras arquitecturas menos conocidas son:[7]

- Modelo Vista Controlador.
- En pipeline.
- Entre pares.
- Orientada a servicios (SOA del inglés Service-Oriented Architecture).
- Arquitectura de microservicios (MSA del inglés MicroServices Architecture). Algunos consideran que es una especialización de una forma de implementar SOA.
- Dirigida por eventos.

- Máquinas virtuales.

1.4.1 Estilos arquitectónicos

Cada estilo arquitectónico describe una categoría del sistema que contiene: un conjunto de componentes, que realiza una función requerida por el sistema, un conjunto de conectores que posibilitan la comunicación, la coordinación y la cooperación entre los componentes; restricciones que definen como se puede integrar los componentes que forman el sistema; y modelos semánticos que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes. De estos, se puede mencionar que: [8]

- 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.

Los diseñadores de sistemas distribuidos deben organizar sus diseños de sistema para encontrar un equilibrio entre rendimiento, confiabilidad, seguridad y manejabilidad del sistema. No hay un modelo universal de organización de sistemas adecuado a todas las circunstancias, así que han surgido varios estilos arquitectónicos distribuidos. Cuando diseñe una aplicación distribuida, deberá elegir un estilo arquitectónico que soporte los requerimientos no funcionales críticos de su sistema. [8]

En esta sección se describen cinco estilos arquitectónicos:[8]

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.

1.5 Tecnologías y herramientas para el desarrollo de la solución

En la actualidad, las tecnologías y herramientas para el desarrollo de soluciones tecnológicas se han convertido en una parte fundamental del proceso de creación de productos y servicios innovadores. La elección de las herramientas y tecnologías adecuadas puede marcar una gran diferencia en el éxito de una solución, ya sea en términos de eficiencia, escalabilidad, seguridad o experiencia de usuario. Desde lenguajes de programación hasta plataformas de desarrollo y herramientas de colaboración en equipo, existen diversas opciones disponibles para los desarrolladores y empresas que buscan crear soluciones tecnológicas avanzadas y competitivas. En este sentido, es importante conocer y evaluar las diferentes opciones disponibles para tomar decisiones informadas y lograr el éxito en el desarrollo de soluciones tecnológicas.

Los sistemas de repositorio de documentos se han desarrollado activamente durante décadas desde que la tecnología web y se han hecho muy populares en la red. Muchas universidades e institutos han construido sus repositorios en Internet para documentos y datos de investigación producidos en sus propias sedes. El concepto "repositorio de documentos" también incluye sitios para compartir código, como GitHub, y archivos de documentos como arXive. Para sitios como GitHub y arXive, es necesario que los usuarios puedan cargar sus archivos.[9]

A medida que los repositorios de documentos han ido floreciendo en web, han aparecido muchos artículos sobre cómo construir repositorios de documentos. DSpace se lanzó por primera vez en 2002 y desde entonces ha sido un recurso de materiales digitales de investigación para el trabajo académico que tiene lugar en el MIT y que ahora puede ser reconocido como pionero de un tipo típico de repositorios de universidades e institutos.[9]

En este trabajo no vamos a implementar Dspace porque vamos a presentar una forma sencilla de desarrollar un pequeño sistema de repositorios utilizando el framework Django, a continuación, dejamos detalladas las herramientas utilizadas:

1.5.1Django

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.[10]

1.5.2 Bootstrap

Bootstrap es un producto de código abierto creado por Otto y Jacob Thornton quienes, cuando este fue lanzado inicialmente, eran ambos empleados de Twitter. Había una necesidad de estandarizar los conjuntos de herramientas front-end de ingenieros en toda la compañía. Desde que Bootstrap fue lanzado en agosto de 2011, ha tenido tanto éxito en su popularidad que ha evolucionado de ser un proyecto totalmente CSS impulsada por incluir una gran cantidad de plugins JavaScript e iconos que van de la mano con las formas y los botones que permite un sensible diseño web y cuenta con un robusto sistema grid de 12 columnas. Uno de los aspectos más destacados es la herramienta de construcción en el sitio web de Bootstrap, donde se puede personalizar los componentes para adaptarse a sus necesidades y elegir qué CSS y JavaScript se desea incluir en su sitio. Todos de esto permite el desarrollo front-end web para ser catapultado hacia adelante, a partir de un estable base de diseño y desarrollo a futuro. Este framework front-end se ha posesionado como uno de los más robustos y populares del mercado, gracias a su versatilidad y funcionalidad.[11]

1.5.3 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.[12]

1.5.4 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.[13]

1.5.5 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).[14]

1.5.6 HTML

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.[15]

1.5.7 CSS

CSS es el lenguaje que utilizamos para dar estilo a un documento HTML.CSS describe cómo deben mostrarse los elementos HTML.[16]

1.5.8 SQLite3

SQLite es una biblioteca en proceso que implementa un motor de base de datos SQL transaccional autónomo, sin servidor, sin configuración. El código de SQLite es de dominio público y, por lo tanto, es de uso gratuito para cualquier propósito, comercial o privado. SQLite es la base de datos más implementada en el mundo con más aplicaciones de las que podemos contar, incluidos varios proyectos de alto perfil.[17]

1.5.9 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.[18]

1.5.10 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.[19]

1.6 Conclusiones parciales

En este capítulo se presenta la descripción del dominio del problema relacionado con la agricultura y la ecología del suelo, y se destaca la importancia de la gestión de la información sobre las costras biológicas del suelo para mejorar la calidad del suelo y la producción agrícola. Se establece la misión y visión de la investigación, que busca desarrollar un repositorio digital para la gestión eficiente de la información sobre las costras bilógicas del suelo. Se mencionan las tecnologías y herramientas para el desarrollo de la solución. Se destaca que la falta de un repositorio digital que recopile y organice la información sobre las costras bilógicas del suelo en un solo lugar dificulta la realización de investigaciones y proyectos relacionados con la agricultura y la ecología del suelo.

Capítulo 2: DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

2.1 Introducción

Este capítulo está dedicado al proceso de desarrollo del dominio utilizando los flujos de trabajo de la metodología AUP. Se realiza el análisis del proceso del repositorio. Se identifican los requisitos funcionales y no funcionales. Se analizan las historias de usuario, la descripción de la arquitectura

2.2 Concepción general del sistema

El sistema desarrollado permite a los editores de la Facultad de Agronomía de la Universidad de Cienfuegos la recopilación, organización y almacenamiento de datos relacionados con las costras biológicas del suelo en una plataforma digital. Este proceso implica proporcionar información detallada sobre las costras biológicas del suelo a través de un sistema eficiente y seguro. Además, es importante considerar la integración de metadatos y la posibilidad de compartir la información de manera accesible y segura para el personal de la institución y otros interesados.

2.3 Requisitos

Los requisitos en un proyecto informático son las características, funcionalidades y restricciones que deben cumplirse para que el proyecto sea exitoso. Existen diferentes tipos de requisitos en un proyecto informático, que pueden clasificarse en dos categorías principales: requisitos funcionales y requisitos no funcionales. Los requisitos funcionales establecen qué tareas debe ser capaz de realizar el sistema o software que se está desarrollando. Estos requisitos suelen describir las acciones específicas que el sistema debe poder llevar a cabo. Los requisitos funcionales definen los comportamientos y las funcionalidades específicas que el sistema debe ofrecer.

Por otro lado, los requisitos no funcionales abarcan aspectos que no están directamente relacionados con las funcionalidades del sistema, pero que son igualmente importantes para el éxito del proyecto. Estos requisitos pueden incluir aspectos como el rendimiento, la seguridad, la usabilidad, la escalabilidad, la compatibilidad, la accesibilidad, entre otros. Los requisitos no funcionales se centran en aspectos más generales que afectan al sistema en su conjunto, y pueden tener un impacto significativo en su implementación y en la experiencia de los usuarios.

2.3.1 Requisitos funcionales

Los requisitos funcionales identificados para el sistema de repositorio digital de costras bilógicas del suelo 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 PDFs.	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	Agregar descripción un PDFs.	El sistema permite que los usuarios con rol administrador y editor agregar la descripción del PDFs.	Alta	Alta
RF 8	Editar PDFs.	El sistema permite que los usuarios con rol administrador y editor editar el título, el año, el	Media	Baja

		autor, el país, fuente de publicidad y seleccionar otro PDF.		
RF 9	Mostrar una lista de PDFs.	El sistema permite que los usuarios con rol investigador visualizar listado de los PDFs.	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 PDF.	El sistema permite que los usuarios con rol investigador puedan visualizar la descripción del archivo deseado.	Alta	Baja
RF 12	Realizar búsqueda avanzada de PDFs.	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 q desea buscar.	Alta	Media
RF 17	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 18	Editar descripción	El sistema permite que los usuarios con rol administrador y editor editar la descripción del PDFs deseado.	Alta	Baja

Tabla 1. Requisitos funcionales.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales identificados para el sistema de repositorio digital de costras bilógicas del suelo 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, esta debe cumplir con las siguientes características:

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 costras bilógicas del suelo. 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 usuarios son las técnicas utilizadas para especificar los requisitos del software, son equivalentes a los casos de uso en el proceso unificado y constituyen la base para las pruebas funcionales. En total se definieron 26 historias de usuarios, a continuación, se muestra la historia de usuario de mayor criticidad, “Agregar descripción un PDFs”:

Número:7	**Requisito:** Agregar descripción un PDFs
Programador: Omar Luis Gutiérrez Pérez	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:12 horas**
Riesgo en desarrollo:	**Tiempo Real:**
Descripción: El componente permitirá entrar el resumen y la descripción del archivo deseado luego seleccionará el archivo al que representa esta descripción. **Pdf:** Campo de selección donde se selecciona el archivo PDF al que corresponde dicha descripción. **Description:** Campo de texto donde se añade la descripción del archivo PDF seleccionado anteriormente.	
Observaciones: Al entrar la descripción se necesita seleccionar el archivo al q pertenece dicha descripción.	

Prototipo elemental de interfaz gráfica de usuario:

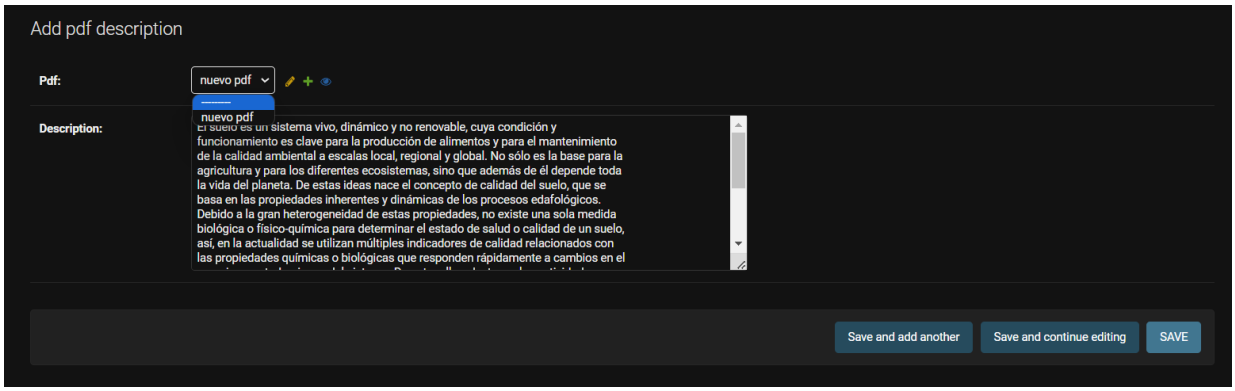

Figura 1.Añadir descripción

Tabla 2.Historia de usuario: Agregar descripción a un PDFs

2.5 Descripción de la arquitectura

La arquitectura del software constituye una especificación de las principales ideas del diseño proporcionando una descripción más detallada de cómo realizar dicho sistema.[20]

2.5.1 Patrón de arquitectura Modelo Vista Template(Plantilla)

La propuesta de solución está basada en el patrón Modelo-Vista-Template, el cual describe la forma de organizar el código de la aplicación:

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.

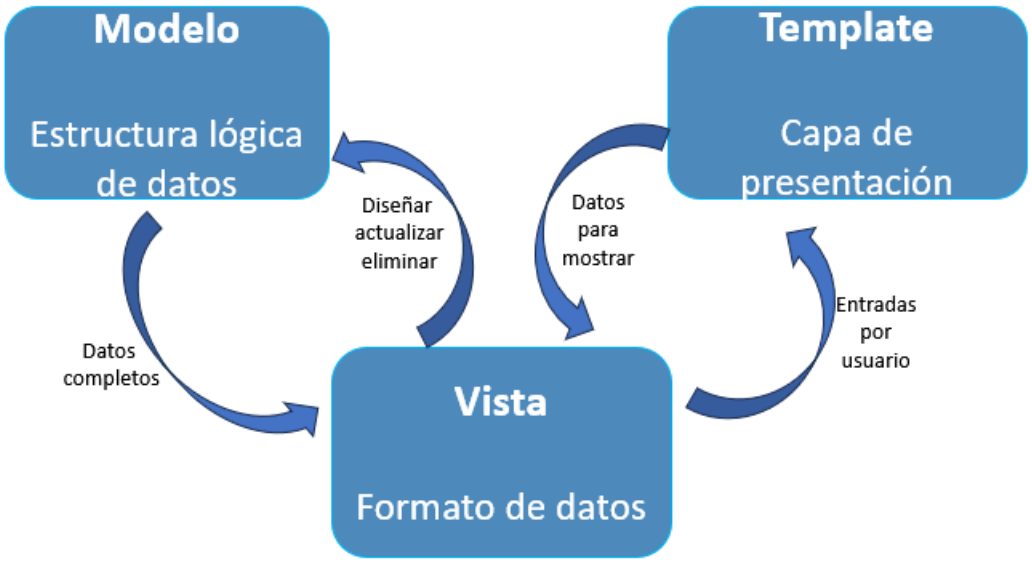

Figura 2. Patron de arquitectura Modelo, Vista, Template.

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

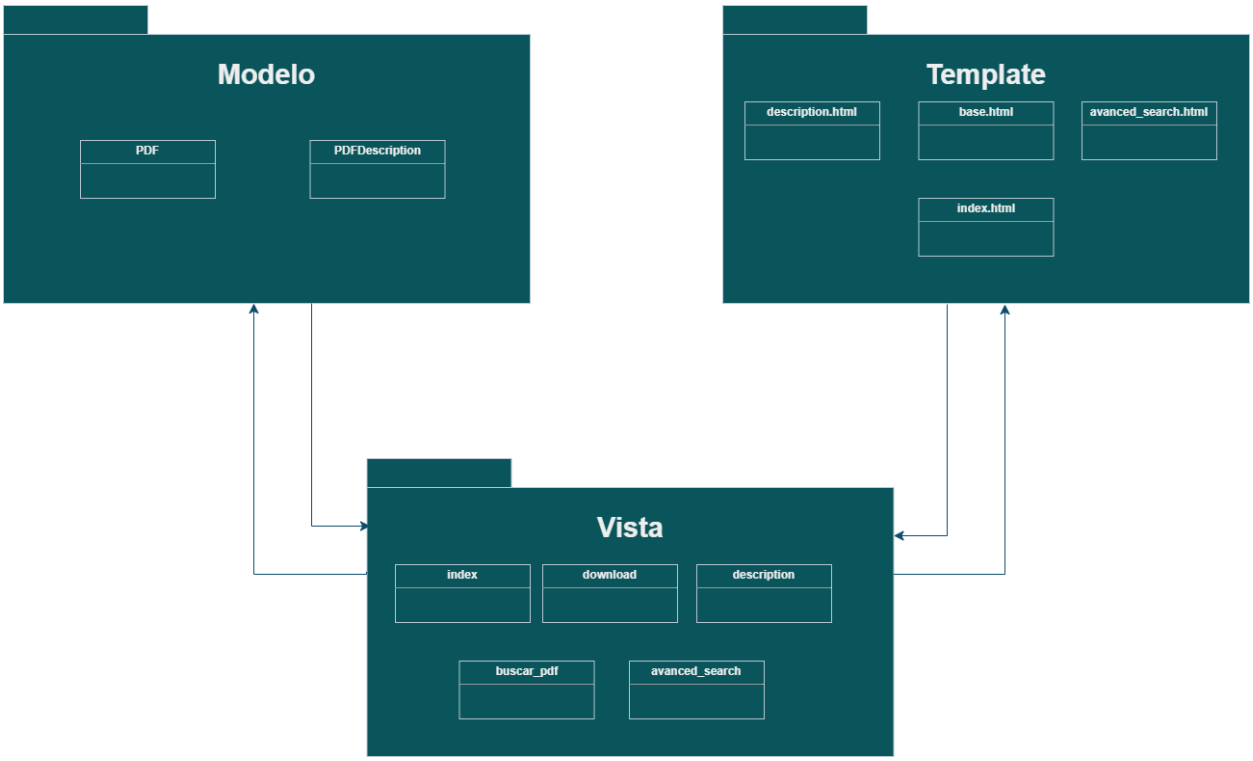

Figura 3.Aplicación de la Arquitectura a la Propuesta de Solución.

2.6 Diagrama de clases diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases. Tiene el propósito de describir las clases que conforman el modelo de un determinado sistema. Dado el carácter de refinamiento iterativo que caracteriza un desarrollo orientado a objetos, el diagrama de clase va a ser creado y refinado durante las fases de análisis y diseño, estando presente como guía en la implementación del sistema.

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

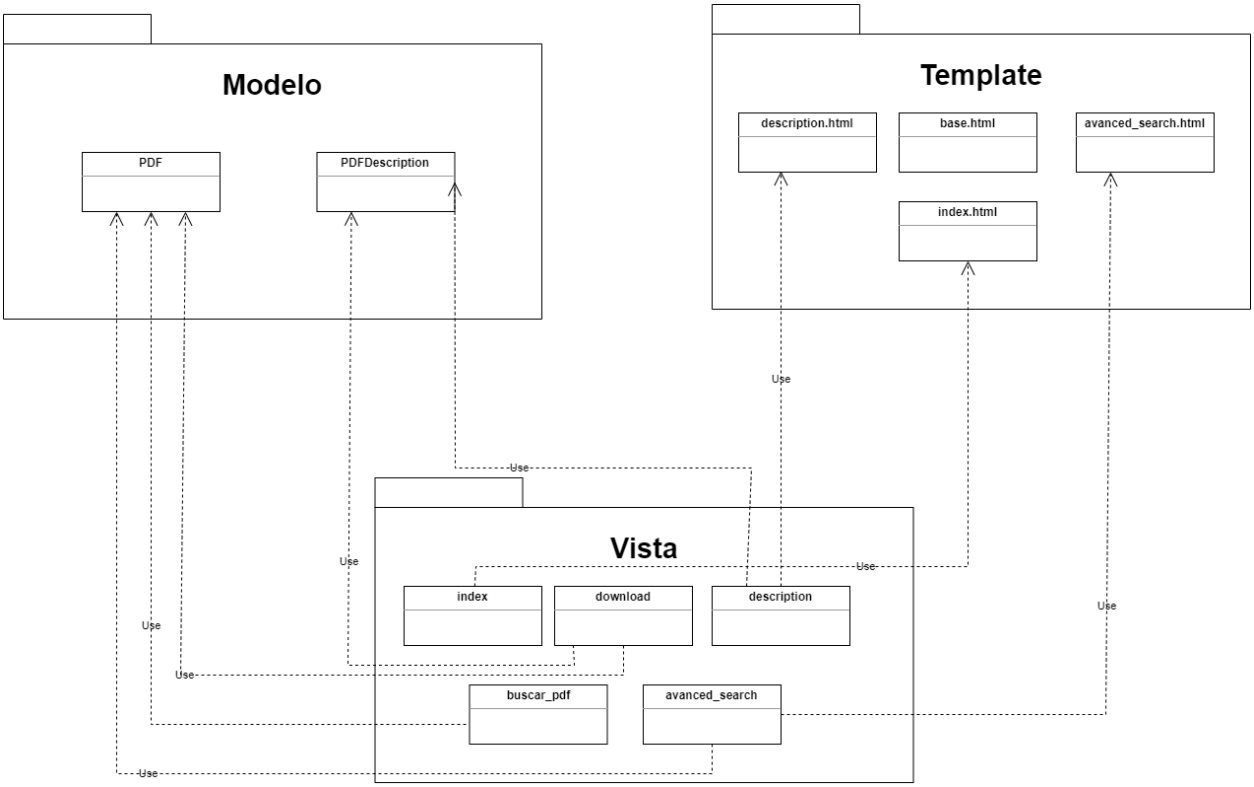

Figura 4.Diagrama de clases diseño.

2.7 Patrones de diseño

Los patrones de diseño son aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. [21]

Patrones GRASP: es un acrónimo de General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para sugerir la importancia de aprehender (grasping en inglés) estos principios para diseñar con éxito el software orientado a objetos. Los patrones de asignación de responsabilidades GRASP dan la medida de un refinamiento del diseño, los cuales se describen a continuación:

Experto: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Se evidencia en la solución en las clases entidades, que son las que cuentan con la información necesaria para cumplir la responsabilidad sobre los elementos del negocio, crearlos y modificarlos, así como las clases repositorio.

Creador: El patrón creador permite identificar quién debe ser el responsable de la instanciación de nuevos objetos o clases. Este patrón se utilizó para identificar qué 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.

Patrones GoF: Los patrones que se presentan proceden de Design Patterns, un libro básico y muy popular que presenta 23 patrones que son útiles durante el diseño de objetos. Puesto que el libro fue escrito por cuatro autores, estos patrones se conocen como los patrones de la "pandilla de los cuatro" o patrones "GoF". Los patrones GoF describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación entre clases y la formación de estructuras de mayor complejidad. Nos permiten crear grupos de objetos para ayudarnos a realizar tareas complejas.

Patrón Decorator: Este método pertenece a la clase base.html, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. Este archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en al resto de las plantillas ({% extends 'base.html' %}). Este procedimiento es una implementación del patrón Decorator.

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.

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.

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.[21]

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.[21]

A continuación, se muestra el modelo de datos:

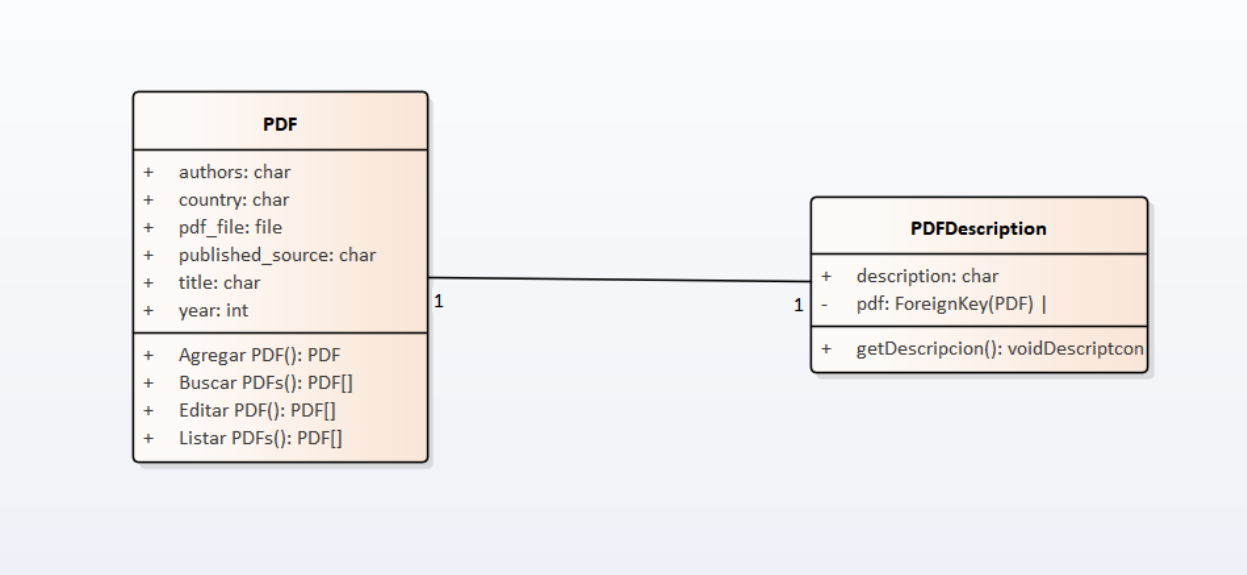

Figura 5.Modelo de Datos.

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 el presente capítulo se ha descrito la solución propuesta a través de los requisitos del sistema, las historias de usuario, se describe la arquitectura y el patrón utilizado (Modelo-Vista-Template). Se analizan los patrones de diseño, los modelos de datos y los estándares de codificación empleados.

Capítulo 3: VALIDACIÓN DE LA VIABILIDAD DE LA PROPUESTA DE SOLUCIÓN

3.1 Introducción

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 al sistema informático Repositorio Digital De Las Costras Biológicas Del Suelo.

3.2 Técnicas de validación de requisitos

Con el objetivo de ratificar que los requisitos del software obtenidos definen el sistema que el cliente desea se llevó a cabo un proceso de validación de los mismos, para el cual se emplearon las siguientes técnicas: [21]

Diseño de prototipos web: Se crearon prototipos interactivos del sistema, los cuales se presentaron al cliente con el objetivo de ofrecer una visión preliminar de cómo sería el producto final. A través de la interacción con los prototipos, el cliente pudo evaluar si satisfacían sus necesidades y requerimientos. Estos prototipos fueron aprobados por el cliente, lo que indica que se logró una adecuada comprensión de los requisitos y una alineación con sus expectativas.
Generación de casos de prueba: Como parte del proceso de validación de los requisitos funcionales del sistema fueron diseñados casos de pruebas para cada historia de usuario.

3.3 Métricas aplicadas a los requisitos

Evaluación de los requisitos a través de revisiones: Se realizaron exhaustivas revisiones de cada uno de los requisitos por parte del equipo de desarrollo y del cliente. Durante las revisiones internas, se identificaron y corrigieron no conformidades técnicas y ortográficas. Además, se llevaron a cabo dos revisiones con el cliente. En la primera reunión, el cliente expresó su satisfacción con el trabajo realizado por el equipo de desarrollo, aunque se realizaron ajustes a algunos requisitos funcionales para su mejora. En la segunda reunión, se añadieron detalles adicionales a los requisitos funcionales y se dio la aprobación final a todos ellos.

Creación de prototipos web: Se diseñaron prototipos interactivos del sistema, los cuales se presentaron al cliente para que pudiera tener una visión preliminar de cómo sería el producto final. A través de la interacción con estos prototipos, el cliente pudo evaluar si cumplían con sus

necesidades y requerimientos. Los prototipos fueron aprobados por el cliente, lo que indica que se logró una comprensión adecuada de los requisitos y una alineación con sus expectativas.

Desarrollo de casos de prueba: Como parte del proceso de validación de los requisitos funcionales del sistema, se generaron casos de prueba específicos para cada historia de usuario. Estos casos de prueba permitieron verificar y comprobar que el sistema cumplía con los requisitos establecidos

3.4 Métricas aplicadas a los requisitos

Con el propósito de evaluar la calidad de la especificación de los requisitos, se aplicó una métrica llamada Calidad de la especificación (CE). Para determinar la comprensibilidad y precisión de los requisitos, se calculó el total de requisitos de la especificación de la siguiente manera:

Nr: Número total de requisitos de especificación.
Nf: Cantidad de requisitos funcionales.
Nnf: Cantidad de requisitos no funcionales.

Realizando la sustitución de valores para el sistema en cuestión, se obtiene:

Nr = Nf + Nnf
Nr =15 + 6
Nr = 21

Para evaluar la Especificidad de los Requisitos (ER) y medir la ambigüedad en ellos, se utiliza la siguiente fórmula:
ER = Nui / Nr

Donde Nui es el número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas. Un valor cercano a 1 en ER indica una menor ambigüedad. En el caso de los requisitos obtenidos para el proyecto, solo se presentó contradicción en una interpretación. Sustituyendo las variables, se tiene:

ER = 20 / 21
ER = 0.95238

Este resultado refleja una baja ambigüedad en los requisitos, ya que el 96.87% de ellos fue entendido de manera unánime por los revisores. Los requisitos que presentaron ambigüedad fueron modificados y validados para asegurar su correcta comprensión.

3.5 Caja blanca

La prueba de caja blanca, denominada a veces “prueba de caja de cristal” es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales; y ejerciten las estructuras internas de datos para asegurar su validez. [21]

Para comprobar que las funciones internas del sistema se ejecutan correctamente se le realizaron pruebas al código de las principales funcionalidades del sistema. Para ello se empleó la técnica de **la prueba del camino básico**, el cual consiste en garantizar que se pueda probar al menos una vez cada una de las sentencias del programa, partiendo de la obtención de la medida de la complejidad de un procedimiento o algoritmo y la obtención de un conjunto básico de caminos de ejecución de este, los cuales son utilizados para obtener los casos de prueba. [21]
Seguidamente se muestra el proceso de pruebas realizado a la funcionalidad buscar_pdf, específicamente de la views.py por ser de mayor criticidad en el sistema. Se comienza por analizar el código y enumerar las instrucciones:

```
def buscar_pdf(request):
    if request.method == 'GET':                                                    #1
        query = request.GET.get('q')                                               #2

        if query:                                                                  #3
            resultados = PDF.objects.filter(Q(title__icontains=query))             #4
            if resultados:                                                         #5
                context = {'resultados': resultados}                               #6
            else:                                                                  #7
                context = {'mensaje': 'No se encontraron archivos que coincidan con la búsqueda.'}  #8
        else:                                                                      #9
            context = {}                                                           #10

        return render(request, 'index.html', context)                              #11
```

Figura 6.Código del Método buscar_pdf.

Luego se debe construir el grafo de flujo correspondiente al código de la función como se muestra en la Figura:

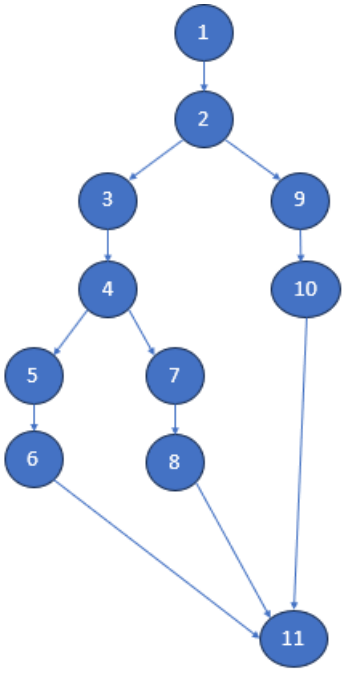

Figura 7.Grafo de Flujo

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:

1. V (G) = a - n + 2, siendo a el número de arcos o aristas del grafo y n el número de nodos.
2. V (G) = r, siendo r el número de regiones cerradas del grafo.
3. 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:

1. V(G) = a – n + 2
   V(G) = 12 – 11 + 2
   V(G) = 3

2. V(G) = 3

3. V(G) = c + 1
   V(G) = 2 + 1
   V(G) = 3

El conjunto de caminos básico sería:

Camino básico 1: 1-2-3-4-5-6-11
Camino básico 2: 1-2-3-4-7-8-11
Camino básico 3: 1-2-9-10-11

Se definen los casos de prueba para cada uno de los caminos básicos obtenidos. A continuación, se muestra el resultado de las pruebas aplicadas a los caminos básicos 1 y 2.

Descripción: Se verificará que la búsqueda deseada coincida con un archivo en la base de datos		
Condición de ejecución:	**Entrada:**	**Resultados esperados:**
La variable querty no debe de estar vacía, para ello se debe hacer clic en el cuadro de buscar y escribir el nombre del documento deseado.	context: (muestre el archivo deseado)	Que se muestre en pantalla el documento entrado por parámetros.
Resultado obtenido: Satisfactorio.		

Tabla 3.Resultado de las Pruebas Aplicadas a los Caminos Básicos 1

Descripción: Se verificará que la búsqueda deseada no coincida con un archivo en la base de datos		
Condición de ejecución:	**Entrada:**	**Resultados esperados:**
La variable querty no debe de estar vacía, para ello se debe hacer clic en el cuadro de buscar y escribir el nombre del documento deseado.	context: (No se encontraron archivos que coincidan con la búsqueda.)	Que muestre en pantalla un mensaje de que no se encontraron archivos que coincidan con la búsqueda.
Resultado obtenido: Satisfactorio.		

Tabla 4.Resultado de las Pruebas Aplicadas a los Caminos Básicos 2

Se logró el cálculo de la complejidad ciclomática para los algoritmos de la aplicación. La misma visualiza la forma que se emplea para obtener dicho cálculo con las estructuras especificadas. Los resultados obtenidos en el cálculo de la complejidad ciclomática definen el número de caminos independientes dentro de un fragmento de código y la cantidad de casos de pruebas diseñados.

3.6 Caja negra

Las pruebas de caja negra, también denominada “prueba de comportamiento”, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los detectados por los métodos de caja blanca. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación.[21]

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.

Dentro del método de Caja Negra la técnica de la **Partición de Equivalencia** es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores.[21]

Comprobamos que las funcionalidades del sistema se realizan de forma correcta y respondan a las necesidades del cliente. Se realizaron un total de 3 iteraciones de pruebas de caja negra, obteniéndose los siguientes resultados:

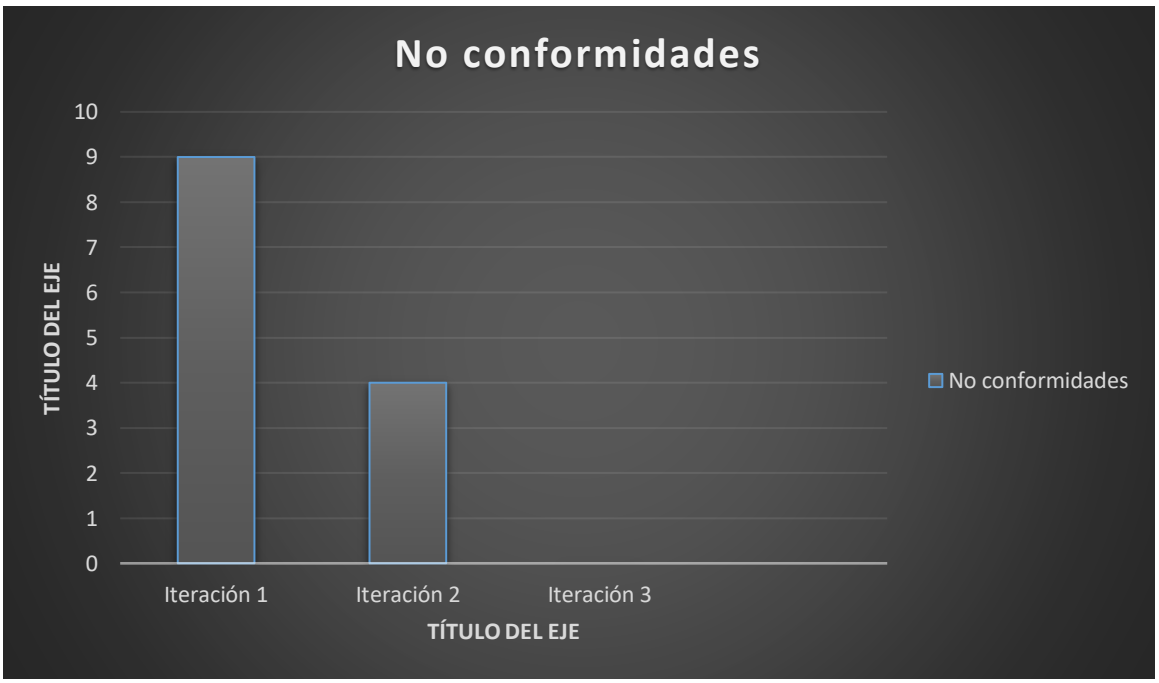

Figura 8.Resultados de las Pruebas de Caja Negra

En una primera iteración se detectaron un total de 9 no conformidades (NC), 2 NC errores de validación, 3 NC funcional, 2 NC de correspondencia entre la aplicación y el documento, 1 NC de errores ortográficos y 1 NC de funcionalidad. En la segunda iteración se obtuvieron 4 no conformidades donde aún preexistían errores de validación y se detectaron además problemas de escritura. Y finalmente, en una tercera iteración se obtuvieron resultados satisfactorios al no detectarse no conformidades.

3.7 Pruebas unitarias

Las pruebas de aceptación son especificadas por el cliente, se centran en las características y funcionalidades generales del sistema que son visibles. Estas pruebas derivan de las HU que se han implementado como parte de la liberación del software. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección y toma de decisiones acerca de estas pruebas. [21]

Al comprobar la aceptación del cliente con la solución desarrollada se realizaron un total de 3 iteraciones, obteniéndose los resultados siguientes:

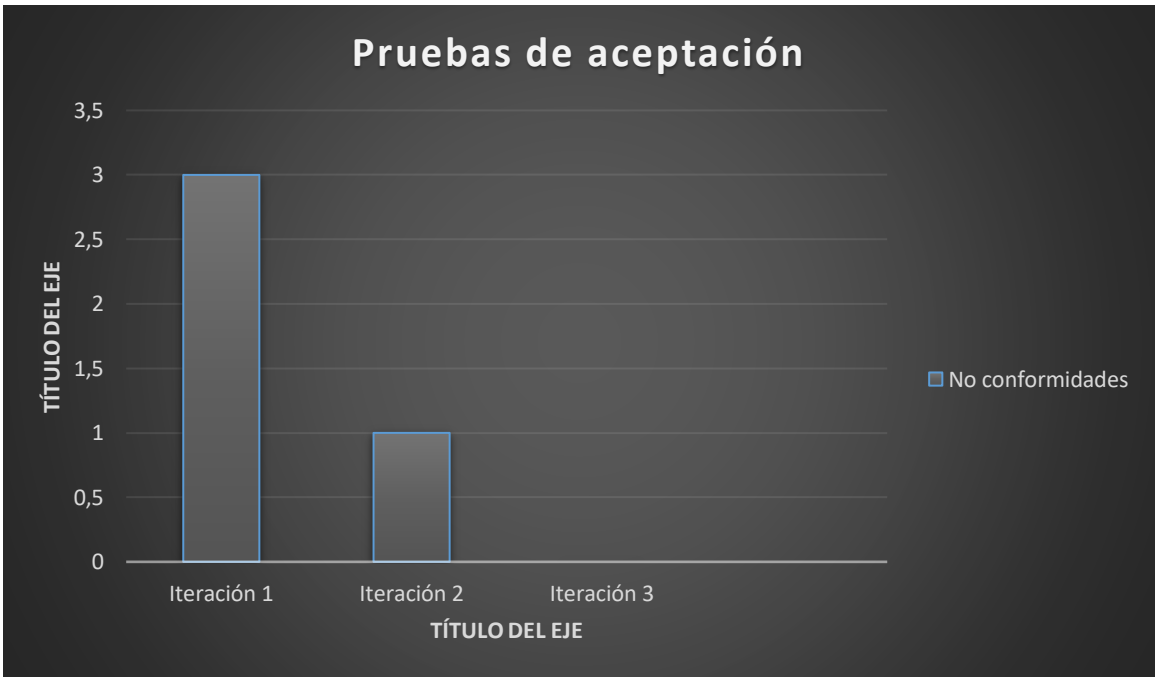

Figura 9.Pruebas de aceptación.

En una primera iteración se detectaron un total de 3 no conformidades (NC) 2 de errores ortográficos y una de error de concordancia. En la segunda iteración se obtuvo 1 (NC) de error ortográfico. Y en la tercera iteración se obtuvieron resultados satisfactorios al no detectarse no conformidades.

3.8 Validación de la solución

Un aspecto crucial en el control de calidad del desarrollo de software son las pruebas y, dentro de estas, las pruebas funcionales, en las cuales se hace una verificación dinámica del comportamiento de un sistema, basada en la observación de un conjunto seleccionado de ejecuciones controladas o casos de prueba.[22]

Para hacer pruebas funcionales se requiere una planificación que consiste en definir los aspectos a chequear y la forma de verificar su correcto funcionamiento, punto en el cual adquieren sentido los casos de prueba. [22]

A continuación, se ejemplifican los casos de prueba.

Caso de prueba: Iniciar sesión: Vista que valida que los campos nombre de usuario y contraseña no estén vacíos, con el fin de permitir el ingreso al sistema.

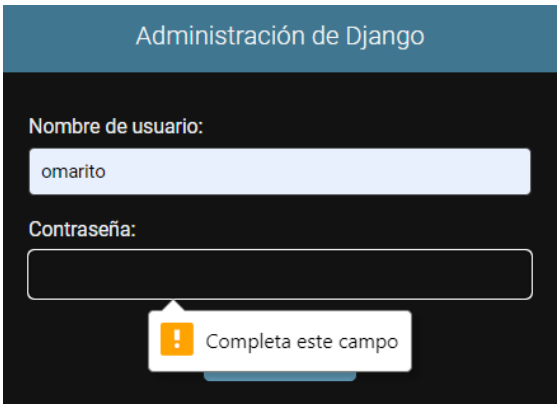

Figura 10.Iniciar sesión, completa campo de contraseña.

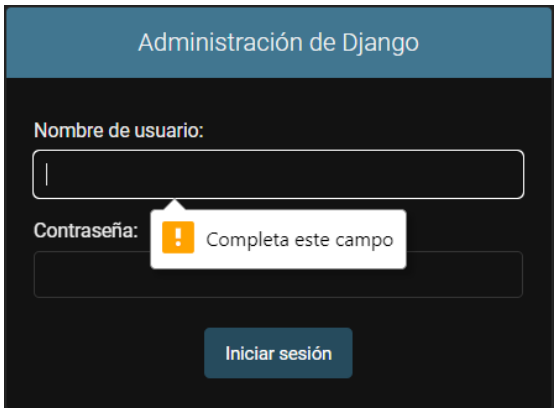

Figura 11.Iniciar sesión, completa campo de nombre de usuario.

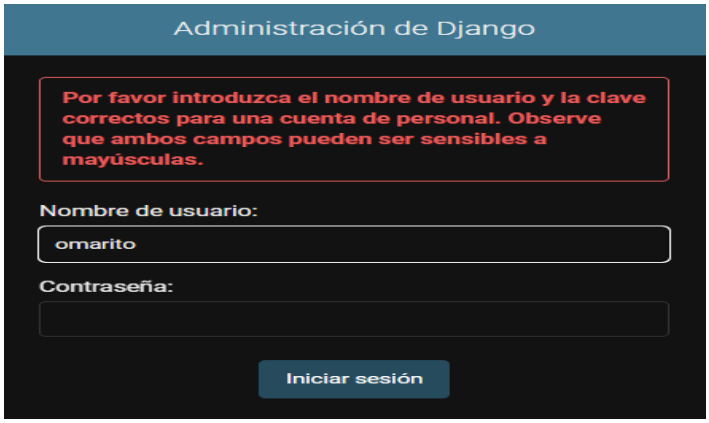

Figura 12.Iniciar sesión, campo de usuario o contraseña incorrecto.

Validaciones:

La validación se hace cuando se dé clic en el botón Iniciar sesión.

Si la autenticación no tuvo éxito saldrá el mensaje “Por favor introduzca el nombre de usuario y la clave correctos para una cuenta de personal. Observe que ambos campos pueden ser sensibles a mayúsculas.”

Caso de prueba: Añadir usuario: Vista que valida los datos requeridos de los nuevos usuarios con el fin que los usuarios se creen correctamente.

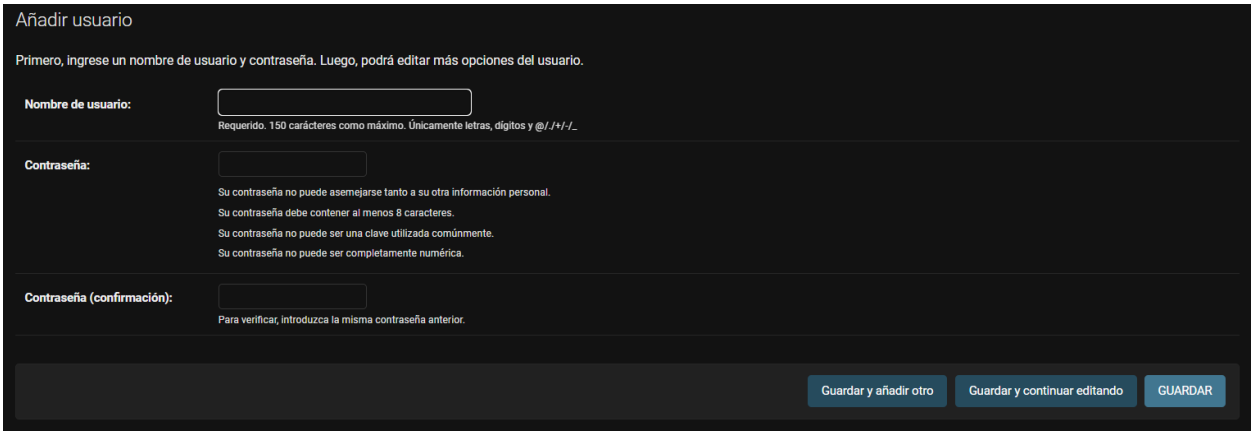

Figura 13.Caso de prueba: Añadir usuario.

Validaciones:

- 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.

Caso de prueba: Modificar usuario:

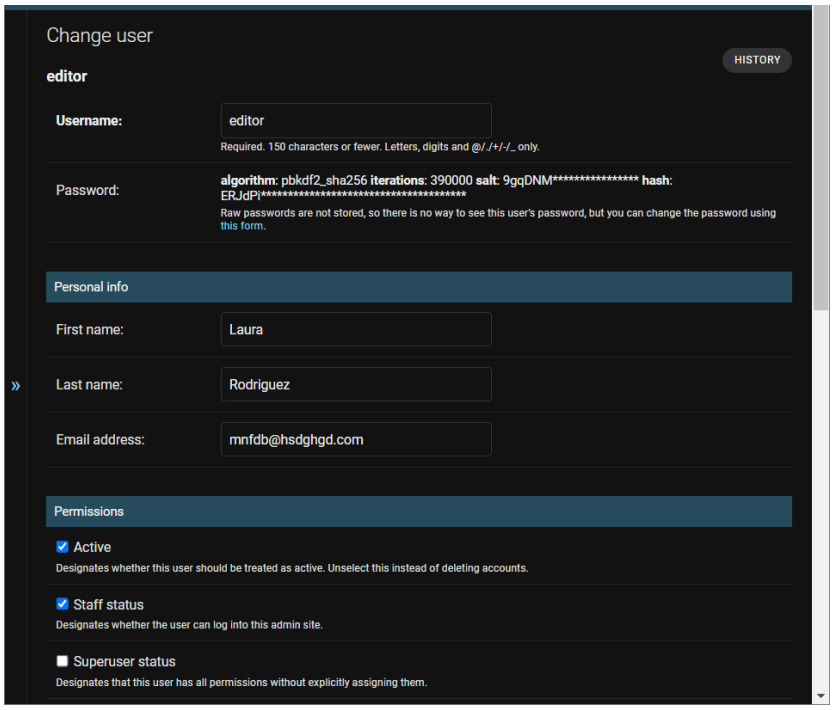

Figura 14.Caso de prueba: Modificar usuario.

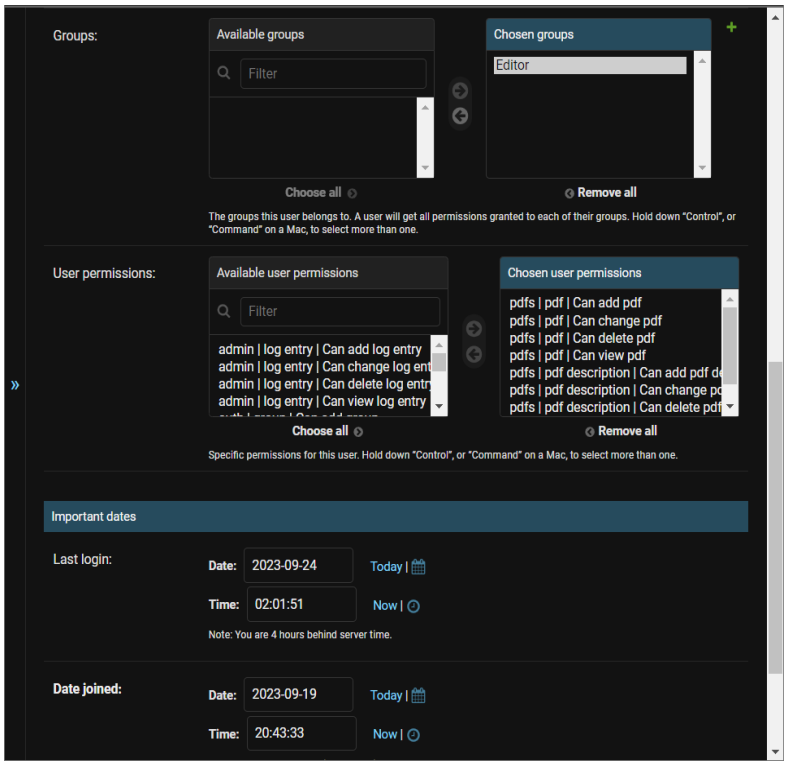

Figura 15.Caso de prueba: Modificar usuario.

Validaciones:

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.

Caso de prueba: Agregar PDFs:

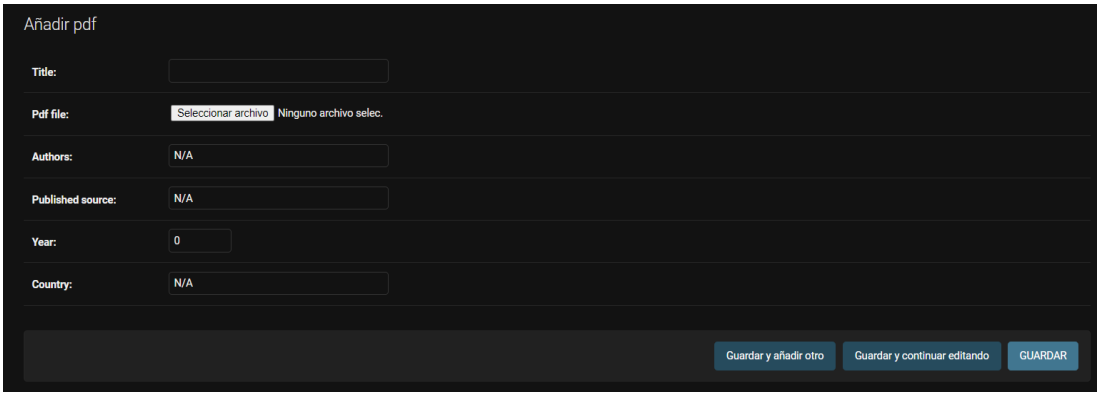

Figura 16.Caso de prueba: Agregar PDFs.

Validaciones:

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.

El administrador o editor no podrá dejar la casilla del Title vacía ni podrá guardar esta información si no ha seleccionado ningún archivo.

Caso de prueba: Cambiar contraseña:

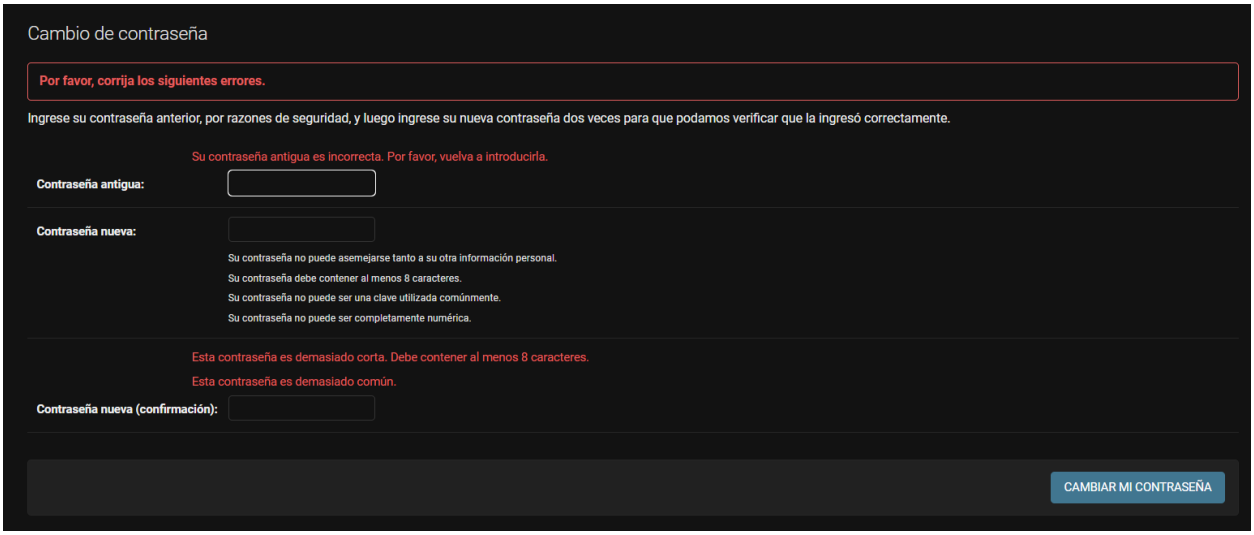

Figura 17.Caso de prueba: Cambiar contraseña.

Validaciones:

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.

En la contraseña antigua debes poner la antigua contraseña si no te va a dar un error de que la contraseña es incorrecta, en el campo de contraseña nueva debes de poner la nueva contraseña donde debes de cumplir una serie de requisitos como Su contraseña no puede asemejarse tanto a su otra información personal, su contraseña debe contener al menos 8 caracteres, su contraseña no puede ser una clave utilizada comúnmente, su contraseña no puede ser completamente numérica.

3.9 Conclusiones

Se utilizaron técnicas de validación de requisitos, como el diseño de prototipos, generación de casos de prueba y revisiones exhaustivas, para obtener requisitos claros y alineados con las expectativas del cliente, lo cual es fundamental para el éxito del desarrollo de software.
En el desarrollo del software, se llevaron a cabo pruebas de validación y verificación utilizando tanto pruebas de caja blanca como pruebas de caja negra. Las pruebas de caja blanca evaluaron el funcionamiento interno del sistema, identificando caminos básicos a través del cálculo de la complejidad ciclomática. Por otro lado, las pruebas de caja negra se enfocaron en

evaluar los requisitos funcionales y detectar posibles errores en la interfaz, rendimiento y funcionalidades.

Las actividades de validación de requisitos, así como las pruebas de caja blanca y caja negra, desempeñaron un papel importante en el proceso de desarrollo del software al garantizar la calidad, la claridad de los requisitos y el cumplimiento de los mismos. Estas técnicas contribuyeron a obtener un sistema funcional y confiable.

Conclusiones generales

Se implemento software q incluye el desarrollo de un repositorio digital de las costras biológicas del suelo generando importantes beneficios en la centralización de la información que permite el almacenamiento, gestión, acceso de la información.

Para la implementación del software se utilizó Django que es un framework de desarrollo web de alto nivel y de código abierto, escrito en Python. Proporciona un conjunto de herramientas y funciones que facilitan la creación de aplicaciones web

Se llevaron a cabo pruebas de caja blanca y caja negra para evaluar tanto el funcionamiento interno del sistema como los requisitos funcionales. Estas pruebas contribuyeron a detectar posibles errores, garantizar el correcto funcionamiento de las funciones internas y verificar el cumplimiento de los requisitos del software.

Referencias

[1] W. U. Amado Noreña, «Cybertesis Perú, análisis del repositorio institucional de la Universidad Nacional Mayor de San Marcos», *Repos. Tesis - UNMSM*, 2010, Accedido: 29 de septiembre de 2023. [En línea]. Disponible en: https://cybertesis.unmsm.edu.pe/handle/20.500.12672/13778

[2] J. Texier, «Los repositorios institucionales y las bibliotecas digitales: una somera revisión bibliográfica y su relación en la educación superior», *11th Lat. Am. Caribb. Conf. Eng. Technol. - 2013*, p. 8, 2013.

[3] A. Keefer, «Los repositorios digitales universitarios y los autores.», *An. Doc.*, vol. 10, pp. 205-214, 2007.

[4] G. Santos-Hermosa, N. Ferran-Ferrer, y E. Abadal, «Recursos educativos abiertos: repositorios y uso», *El Prof. Inf.*, vol. 21, n.º 2, pp. 136-145, ene. 2012, doi: 10.3145/epi.2012.mar.03.

[5] T. Castro y M. Mercedes, «Génesis y clasificación de costras superficiales en suelos de cultivo».

[6] «Si-01 Metodologia AUP.pdf».

[7] R. S. Pressman, «Ingenieria del Software. Un Enfoque Practico».

[8] I. Sommerville, «Ingenieria de Software».

[9] Department of Fisheries Distribution and Management, National Fisheries University, Shimonoseki, Yamaguchi, Japan, K. Kajitori, y K. Aoki, «Implementation of a Simple Document Repository System», *Int. J. Mod. Educ. Comput. Sci.*, vol. 8, n.º 9, pp. 12-19, sep. 2016, doi: 10.5815/ijmecs.2016.09.02.

[10] «Django», Django Project. Accedido: 21 de junio de 2023. [En línea]. Disponible en: https://www.djangoproject.com/

[11] C. I. Y. Portoviejo, «100378542-3 DELGADO MORALES CRISTIAN ANDRÉS».

[12] «¿Qué es Python? - Explicación del lenguaje Python - AWS», Amazon Web Services, Inc. Accedido: 21 de junio de 2023. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/python/

[13] «Qué es el JavaScript - Definición, significado y ejemplos». Accedido: 21 de junio de 2023. [En línea]. Disponible en: https://www.arimetrics.com/glosario-digital/javascript

[14] «Documentation for Visual Studio Code». Accedido: 21 de junio de 2023. [En línea]. Disponible en: https://code.visualstudio.com/docs

[15] «Conceptos básicos de HTML - Aprende desarrollo web | MDN». Accedido: 21 de junio de 2023. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics

[16] «CSS Tutorial». Accedido: 21 de junio de 2023. [En línea]. Disponible en: https://www.w3schools.com/css/

[17] «Acerca de SQLite». Accedido: 21 de junio de 2023. [En línea]. Disponible en: https://www.sqlite.org/about.html

[18] «ramblainf.com/enterprise-architect#:~:text=Enterprise Architect es una herramienta,con gran nivel de fiabilidad.» Accedido: 21 de junio de 2023. [En línea]. Disponible en: https://www.ramblainf.com/enterprise-architect

[19] K. Team, «¿Qué es Draw.io? | KeepCoding Bootcamps». Accedido: 25 de octubre de 2023. [En línea]. Disponible en: https://keepcoding.io/blog/que-es-drawio/

[20] «Sistema para la gestión del proceso de trabajos de.pdf».

[21] L. Romero Romero y D. Nasco Castrillón, «Sistema para la gestión del proceso de trabajos de diplomas en la Facultad 3», p. 61.

[22] R. L. Matos Peraza, «Sistema de información sobre el ingreso a la Educación Superior del CRD en Cienfuegos.», Informática, Carlos Rafael Rodríguez, Cienfuegos, Cuba, 2020.