DISEÑO DE UN SISTEMA DE INFORMACIÓN PARA LA REORGANIZACIÓN Y CONSULTA EFICIENTE DE TRABAJOS DE GRADO DEL DEPARTAMENTO DE TECNOLOGÍA

Autor

Diego Alexander Suárez Burgos

Licenciatura en electrónica

Facultad de Ciencia y Tecnología

Universidad Pedagógica Nacional

2025

DISEÑO DE UN SISTEMA DE INFORMACIÓN PARA LA REORGANIZACIÓN Y CONSULTA EFICIENTE DE TRABAJOS DE GRADO DEL DEPARTAMENTO DE TECNOLOGÍA

Autor

Diego Alexander Suárez Burgos

Trabajo de grado para obtener el título de Licenciado en Electrónica

Asesor:

Jimmy William Ramírez Cano

Licenciatura en Electrónica

Facultad de Ciencia y Tecnología

Universidad Pedagógica Nacional

Bogotá, Colombia

2025

AGRADECIMIENTOS

A mis padres y a mi hermana, por su apoyo incondicional, por acompañarme con amor, paciencia y fortaleza en los momentos más difíciles y por creer en mí, incluso cuando yo mismo dudaba. Gracias por ser mi refugio y mi motor para seguir adelante en cada etapa de este camino. A la Universidad Pedagógica Nacional, por brindarme una formación integral, no solo como futuro licenciado, sino también como ser humano crítico, comprometido y consciente de su labor educativa en la sociedad.

Y a mi asesor, Jimmy Ramírez, por su guía, paciencia y acompañamiento en este proceso que en algún momento sentí imposible, pero que gracias a su orientación logré transformar en un trabajo de grado sólido y significativo.

Tabla de contenido

1. INTRODUCCIÓN 9

2. PROBLEMA / NECESIDAD 10

3. JUSTIFICACIÓN 11

4. OBJETIVOS 13

4.1 OBJETIVO GENERAL 13

4.2 OBJETIVOS ESPECÍFICOS 14

5. ANTECEDENTES 14

6. MARCO TEÓRICO 18

6.1 ASPECTO ACADÉMICO 19

6.1.1 PROYECTO DE TRABAJO DE GRADO 19
6.1.2 LÍNEAS DE INVESTIGACIÓN 19
6.1.3 SISTEMA DE INFORMACIÓN 20
6.1.4 GESTIÓN DEL CONOCIMIENTO 21
6.1.5 BÚSQUEDA DE INFORMACIÓN 22
6.1.6 ACCESO ABIERTO 23
6.1.7 PROCESAMIENTO DE LENGUAJE NATURAL (PNL) 23
6.1.8 EXPRESIONES REGULARES 24

6.2 ASPECTO TECNOLÓGICO 24

6.2.1 PYTHON 25
6.2.2 FRAMEWORK 25
6.2.3 DJANGO 26
6.2.4 REPOSITORIO DIGITAL 27
6.2.5 ORM (OBJECT-RELATIONAL MAPPING) 27

6.2.6 BASE DE DATOS 28
6.2.7 OPEN SOURCE 29
6.2.8 ANÁLISIS Y EXTRACCIÓN DE PDF (PYMUPDF) 30
6.2.9 INTERFAZ DE USUARIO (UI) 31
6.2.10 PIPENV 32
7. METODOLOGÍA 32
7.1 FASE 1 - ANÁLISIS Y DEFINICIÓN DE REQUERIMIENTOS 33
7.2 FASE 2 - DISEÑO DEL SISTEMA Y DEL SOFTWARE: 34
7.3 FASE 3 - IMPLEMENTACIÓN Y PRUEBA DE UNIDADES: 34
7.4 FASE 4 - INTEGRACIÓN Y PRUEBA DEL SISTEMA: 35
7.5 FASE 5 - FUNCIONAMIENTO Y MANTENIMIENTO: 35
8. RESULTADOS 35
8.1 EJECUCIÓN FASE 1 - ANÁLISIS Y DEFINICIÓN DE REQUERIMIENTOS 35
DIAGRAMA DE CASOS DE USO 36
ANÁLISIS Y COMPARACIÓN DE DJANGO CON OTRAS PLATAFORMAS 37
LÍNEAS DE INVESTIGACIÓN 41
8.2 EJECUCIÓN FASE 2 - DISEÑO DEL SISTEMA Y DEL SOFTWARE 43
8.2.1 UML DIAGRAMA DE CLASES 44
..... 44
EL DIAGRAMA MTV DE DJANGO 46
CREACIÓN DEL APLICATIVO WEB HASTA LA CREACIÓN DEL SUPERUSUARIO 49
CREACIÓN DE UNA PRIMERA INTERFAZ 62
8.3 FASE 3: IMPLEMENTACIÓN DEL SISTEMA 66
8.3.1 EXTRACCIÓN DE INFORMACIÓN DESDE PDFS 66
8.3.2 EXPRESIONES REGULARES, PATRONES DEL CÓDIGO FINAL 78

8.3.2.1 CONFIGURACIÓN DE DJANGO Y PLANTILLAS USADAS PARA LAS INTERFACES (HTMLS) 81
8.4 FASE 4: PRUEBAS Y MANTENIMIENTO 90
8.4.1 FUNCIONAMIENTO DEL SISTEMA 90
8.4.2 RESTRICCIONES DE ADMINISTRADOR A USUARIO 90
8.4.3 FILTROS DE BÚSQUEDA 91
8.4.4 APLICACIÓN WEB DEL REPOSITORIO DIGITAL 92
9. CONCLUSIONES 93
10. REFERENCIAS BIBLIOGRÁFICAS 97

ÍNDICE DE FIGURAS

Figura 1. Ciclo de vida del software. Fuente. González, 2014, SISTEMA DE INFORMACIÓN PARA CONSULTAR Y ADMINISTRAR LOS DOCUMENTOS DE LOS TRABAJOS DE GRADO DE LA LICENCIATURA EN ELECTRÓNICA. (pag 18) 33

Figura 2. Diagrama UML de casos de uso elaborado en Lucid chart. Fuente. Elaboración propia 36

Figura 3. Diagrama de clases UML. Fuente. Elaboración propia 44

Figura 4. Diagrama MTV de Django. Fuente. Condori, 2012, Revista Boliviana de Ciencia y Tecnología. 47

Figura 5. Comprobación de versión de Python y pip en terminal de Windows. Fuente: Elaboración propia 49

Figura 6. Mensaje final cuando se instala Django. Fuente: Elaboración propia 50

Figura 7. Instalación de pipenv en consola. Fuente: Elaboración propia..... 50

Figura 8. Creación del proyecto Django y comprobación de su versión. Fuente: Elaboración propia..... 51

Figura 9. Activación del ambiente virtual. Fuente: Elaboración propia 52

Figura 10. Archivos creados en la carpeta del proyecto. Fuente: Elaboración propia..... 53

Figura 11. Activación del servidor. Fuente: Elaboración propia 54

Figura 12. Interfaz de Django que muestra que todo está bien configurado. Fuente: Elaboración propia .. 55

Figura 13. Nombramiento del proyecto y verificación del ambiente virtual. Fuente: Elaboración propia.. 56

Figura 14. Contenido de la carpeta del proyecto con el comando **tree.** Fuente: Elaboración propia 57

Figura 15. Inclusión del nombre del proyecto en el archivo settings.py. Fuente: Elaboración propia 57

Figura 16. Creación del superusuario colocando correo y contraseña. Fuente: Elaboración propia 61

Figura 17. Clase **Documento** en models.py. Fuente: Elaboración propia 62

Figura 18. Vistas primarias del archivo views.py. Fuente: Elaboración propia..... 63

Figura 19. Rutas enlazadas en archivo urls.py. Fuente: Elaboración propia 65

Figura 20. Archivo HTML que muestra los detalles del documento. Fuente: Elaboración propia 66

Figura 21. Prueba en el Documento 1 con PyMuPDF. Fuente. Elaboración propia..... 69

Figura 22. Prueba en el Documento 2 con PyMuPDF. Fuente. Elaboración propia..... 70

Figura 23. Prueba del documento 3 con PyMuPDF. Fuente. Elaboración propia 71

Figura 24. Prueba del documento 1 con **spacy** y **pdfplumber**. Fuente. Elaboración propia 73

Figura 25. Prueba del documento 2 con spacy y pdfplumber. Fuente. Elaboración propia..... 73

Figura 26. Prueba del documento 3 con spacy y pdfplumber. Fuente. Elaboración propia 74

Figura 27. Bibliotecas importadas al código final. Fuente. Elaboración propia 76

Figura 28. Función extraer_texto del código final. Fuente. Elaboración propia 77

Figura 29. Página principal del proyecto. Fuente. Elaboración propia 85

Figura 30. Referencias de un documento de la licenciatura en diseño tecnológico. Fuente. Tomado de Zambrano, 2022, Desarrollo de competencias digitales a través de la gamificación como estrategia en estudiantes de grado noveno (pag 102) 87

Figura 31. Texto de la página 102 del documento escogido. Fuente. Elaboración propia 87

Figura 32. Conexión de plantillas en el navegador. Fuente. Elaboración propia 89

ÍNDICE DE TABLAS

Tabla 1. Comparación entre Django, Omeka, Dspace, Greenstone, Esprints 45

1. INTRODUCCIÓN

Este proyecto se enfoca en la elaboración de un sistema de reorganización y consulta eficiente de información para un repositorio digital. El trabajo es producto de la investigación titulada "Análisis de la producción de conocimiento en la formación de Licenciados en Diseño Tecnológico y Electrónica de la Universidad Pedagógica Nacional. Una mirada desde los trabajos de grado entre 2017 y 2022" del grupo de investigación Educación y regionalización en CTel – GIER aprobado con el código DTE-64-23. Para el desarrollo del sistema, se utilizó tanto la información de los trabajos de grado como la identificación de las líneas de investigación del Departamento de Tecnología, con el fin de incorporar estos trabajos en la aplicación.

Los repositorios universitarios son una especie de cofre de información en el que poseen un catálogo extenso de tesis de grado, trabajos de investigación, artículos académicos, informes técnicos, presentaciones, materiales de cursos, entre otros. Estos elementos están disponibles para toda la comunidad educativa y para el público en general, por tanto, se pueden acceder y consultar fácilmente desde la página web de cada repositorio.

La Universidad Pedagógica cuenta con un repositorio para su comunidad académica en el que alojan todo este tipo de documentos académicos. Sin embargo, los estudiantes del Departamento de Tecnología enfrentan dificultades al buscar y acceder a información para identificar su problema de investigación. Aunque la universidad cuenta con una extensa colección de documentos digitales, la búsqueda de información se ve obstaculizada por la carencia de herramientas que faciliten una consulta eficiente, abarcando especialmente, lo relacionado con los trabajos de grado. La búsqueda y selección de antecedentes para proyectos de grado resulta poco efectiva, generando una pérdida significativa de tiempo para los estudiantes. Además, algunos de ellos enfrentan dificultades al no tener un tema central claro para sus trabajos de grado, lo que los lleva a invertir tiempo adicional en la exploración de temas y la búsqueda de

inspiración.

Debido a esto, se pretende crear una forma alterna para consultar y reorganizar la información preponderante de los trabajos de grado del Departamento de Tecnología en específico, a través de un repositorio digital que reúna la información de trabajos de grado, organizadas por líneas de investigación durante los últimos 5 años, para ello se trabajó en conjunto con el semillero Educación y regionalización en CTe|I – GIER. El repositorio se crea con el fin de favorecer la consulta e identificación de antecedentes de los trabajos de grado para los trabajos del DTE. Esto favorece la proyección de nuevos trabajos de grado y dar continuidad a investigaciones ya sustentadas. Esto se logró a través de un framework de Python llamado Django que sirve para diseñar aplicativos webs de una manera sencilla y rápida, debido a que tiene un conjunto de herramientas y bibliotecas que facilitan el desarrollo. Además, es posible programar en Python para que incluya una extracción de datos importantes de los trabajos de grado y se logre reorganizar toda la información relevante del documento dentro de una tabla, haciendo. Este proceso implica utilizar la biblioteca PyMuPDF para acceder al contenido de los documentos y, seguidamente, aplicar expresiones regulares mediante regex, las cuales permiten identificar y extraer automáticamente la información relevante a través de patrones. Posteriormente, estos datos se organizan de manera estructurada en una tabla.

2. PROBLEMA / NECESIDAD

El problema de investigación se centra en la ineficiencia y dificultades que enfrentan los estudiantes del Departamento de Tecnología de la Universidad Pedagógica Nacional al buscar y acceder a información relevante en el actual repositorio de trabajos de grado. A pesar de que la universidad alberga una gran cantidad de documentos digitales, la búsqueda de información no es eficiente. Esto se debe a la falta de herramientas que permitan una consulta ágil y efectiva, especialmente en los trabajos de grado, por ello la

búsqueda y selección de información para elaborar los antecedentes resulta poco eficiente. Esta problemática se traduce en una pérdida de tiempo considerable, ya que los estudiantes deben realizar una búsqueda de numerosos documentos para encontrar antecedentes relevantes para sus proyectos. Incluso hay casos en los que los estudiantes no tienen un tema central claro para desarrollar sus proyectos, lo que los lleva a invertir un tiempo considerable en la búsqueda de inspiración y en la exploración de temas diversos.

Los estudiantes que enfrentan dificultades para encontrar información relevante para sus proyectos académicos pueden sentirse desmotivados y desorientados. Esto puede llevar a una disminución en la calidad de sus trabajos y afectar negativamente su enfoque y compromiso con la investigación. Además, esto puede obstaculizar la continuidad de los trabajos de investigación, lo que limita la capacidad de los estudiantes para construir sobre investigaciones previas y avanzar en áreas de interés específicas. Además, afecta a la proyección de nuevos trabajos de grado acordes con las líneas de investigación del Departamento de Tecnología. Esto no solo tiene implicaciones en el presente, sino que también puede tener consecuencias a largo plazo en sus trayectorias académicas y profesionales, ya que se enfrentan a dificultades para desarrollar proyectos de investigación de manera efectiva y contribuir al campo de la Tecnología además del desarrollo del DTE y sus programas.

3. JUSTIFICACIÓN

Es importante establecer un sistema de información que mejore la eficiencia en la búsqueda y acceso a información crucial de los trabajos de grado presentados por los estudiantes del Departamento de Tecnología. [2] Hacer que la búsqueda y el acceso a la información sea más eficiente es clave para que los estudiantes podamos desarrollar nuestras investigaciones de forma más organizada y efectiva. La idea es que la herramienta propuesta no solo nos ayude a encontrar lo que necesitamos más rápido, sino que también nos motive y haga sentir más involucrados con nuestros proyectos académicos. Cuando es más fácil acceder

a la información, es mucho más probable que nos comprometamos y trabajemos con mayor entusiasmo en nuestras investigaciones.

La información que poseen los documentos son experiencias compartidas para toda la comunidad educativa, no solo en el ámbito investigativo, donde se centra el abordaje de una problemática y su solución, sino que conlleva un proceso del paso a paso de las tareas realizadas por los autores de los documentos de grado y las dificultades que ha tenido durante todo el proceso. [3] El conocimiento no es algo que se pueda tocar o guardar en una caja, más bien, su gestión no es un producto final, sino un proceso constante. Se trata de cómo conectamos la información con la experiencia, y de cómo ponemos ese resultado al alcance de todos en una organización para que puedan hacer su trabajo de forma más eficiente. Hay muchas formas de definirlo, pero una que me parece bastante clara es la de Doculabs, que lo entiende como la manera en que se aprovecha la experiencia colectiva dentro de la organización y se comparte con las personas adecuadas en el momento oportuno. En otras palabras, es una forma de administrar y compartir el conocimiento que todos los miembros aportan. La literatura académica respalda la importancia de implementar soluciones que mejoren la eficiencia en el acceso a la información, ya que esto no solo impacta directamente en la calidad de la formación académica, sino que también tiene implicaciones significativas en la productividad estudiantil y la contribución al conocimiento en el campo de la tecnología.

La falta de acceso eficiente a antecedentes relevantes obstaculiza la construcción progresiva del conocimiento en el campo de la tecnología. [4] La continuidad de investigaciones es fundamental para avanzar en áreas de interés y fortalecer el desarrollo académico, cuando no se cuenta con acceso a los antecedentes, este proceso se ve afectado y hace que limite las posibilidades de profundizar o ampliar el

conocimiento existente. Esto resulta complicado, a la hora de continuar una investigación, si no se puede

acceder fácilmente a la información fundamental de los trabajos de grado previos, ya que esa información es la que permite establecer vínculos y apoyar nuevas propuestas investigativas. La continuidad de investigaciones anteriores es clave para avanzar en áreas de interés y fortalecer el desarrollo académico. Cuando no se cuenta con acceso a los antecedentes, este proceso se ve afectado y limita las posibilidades de profundizar o ampliar el conocimiento existente. Resulta complicado continuar una investigación si no se puede acceder fácilmente a la información fundamental de los trabajos de grado previos, ya que esa información es la que permite establecer vínculos y apoyar nuevas propuestas investigativas.

Este proyecto cumple con los anteriores requerimientos, ya que ofrece una consulta eficiente de información preponderante y la reorganización de esa información acorde con las líneas de investigación que favorezcan la identificación de antecedentes para la continuidad de los trabajos de grado o ya sea para la proyección de una nueva investigación. [4] Para que una investigación sea de calidad, es indispensable contar con los recursos adecuados, sin ellos, es difícil alcanzar buenos resultados. En este contexto, la propuesta de un sistema de información busca ser una herramienta útil para los estudiantes, especialmente para aquellos que hoy en día deben invertir mucho tiempo buscando información en el repositorio de la UPN. Este sistema facilitará una búsqueda más rápida y precisa de los trabajos de grado del departamento de tecnología, lo que no solo ahorra tiempo, sino que también mejora el proceso de elaboración de nuevos proyectos académicos. Esto no quiere decir que los estudiantes no deban hacer una lectura exhaustiva del documento, sino que logren identificar fácilmente aspectos importantes de lo que están consultando y puedan aprovechar esa información para el desarrollo de sus proyectos de investigación.

4. OBJETIVOS

4.1 Objetivo general

Diseñar e implementar un aplicativo digital que integre metadatos relevantes y enlaces de descarga a los trabajos de grado realizados por los estudiantes del Departamento de Tecnología de la Universidad Pedagógica Nacional durante los últimos cinco años, haciendo uso del framework Django.

4.2 Objetivos específicos

- Aplicar la información suministrada por el grupo de investigación Educación y Regionalización en CTeI – GIER acerca de los trabajos de grado y las líneas de investigación identificadas para diseñar una primera interfaz del repositorio digital.

- Implementar un sistema automatizado que permita extraer información clave de los trabajos de grado, como palabras clave, objetivos, metodología, título, contacto, etc.

- Reorganizar la información extraída de los trabajos de grado en un formato claro y sistematizado que facilite a los estudiantes del Departamento de Tecnología la búsqueda rápida de información mejorando así la accesibilidad de antecedentes relevantes.

5. ANTECEDENTES

Desde 2008, algunas instituciones educativas han adoptado el uso de repositorios en línea para almacenar y gestionar información de consulta y material didáctico. Estos repositorios también han servido como una herramienta educativa valiosa, que, con el tiempo, su utilidad se ha extendido debido a los beneficios que ofrecen. En la actualidad, la mayoría de las universidades han desarrollado sus propios repositorios digitales para alojar y gestionar una gran variedad de información académica y científica.

Asimismo, se mostraron algunos de los proyectos relacionados con el objeto de estudio, los cuales emplean el framework Django como herramienta principal para su desarrollo.

En 2016, Luz Helena González Sandoval desarrolló un trabajo de grado llamado “Sistema de información para consultar y administrar los documentos de los trabajos de grado de la Licenciatura en Electrónica”, en el cual examinó cuatro sistemas de gestión de repositorios institucionales de código abierto: Omeka, DSpace, Greenstone y EPrints. Su objetivo fue determinar cuál de estas plataformas se ajustaba mejor a los requerimientos definidos por la coordinación de la Licenciatura en Electrónica, con la finalidad de implementar un sistema que facilitara la organización y consulta de los trabajos de grado. Utilizó la metodología en cascada, lo que le permitió organizar paso a paso el desarrollo del software, gracias a esto, ofrece una base técnica útil que puede orientar a quienes quieran seguir investigando o creando sistemas parecidos, además, por la forma en que estructuró el proceso y enfocó su propuesta, se convierte en un buen punto de partida para pensar en nuevas herramientas que respondan mejor a las necesidades actuales de estudiantes y docentes.

[6] En 2015, Diego Fernando Caicedo y Germán Céspedes realizaron un proyecto en la Facultad de Ciencias Básicas e Ingenierías de la Universidad de los Llanos, en el que desarrollaron un repositorio digital, el cual utilizaron metodologías ágiles como SCRUM y XP, además trabajaron con el framework Django. La idea principal del proyecto era guardar y organizar los documentos que se generan durante los procesos de acreditación y en los planes de mejoramiento. Uno de los puntos importantes que destacaron fue que, debido a las características del programa y lo que sugiere el CNA, no era tan adecuado crear un modelo de datos rígido. En su lugar, optaron por una forma de organizar archivos más flexibles, lo que les permitió implementar un sistema web funcional que facilitara la administración de toda esa documentación.

[14] En 2017, María Barrera y Mario Barros, estudiantes de la Universidad Estatal de Milagro en Ecuador, presentaron un trabajo de grado titulado “Análisis de los procesos de divulgación científica de los docentes investigadores para el diseño de un repositorio web en la Universidad Estatal de Milagro”. Su objetivo principal era mejorar la forma en que se administra y difunde la producción científica dentro de la

universidad, sobre todo ante las nuevas exigencias que el estado ecuatoriano ha impuesto a las instituciones de educación superior. Como parte de la propuesta, desarrollaron una solución tecnológica basada en herramientas web de código abierto, estableciendo los requerimientos necesarios para crear un repositorio institucional que sirviera para almacenar y organizar los artículos científicos generados por los docentes investigadores.

[7] En 2021, Camilo Andrés Basabe Chacón llevó a cabo un estudio titulado Análisis de las tendencias de los proyectos de grado del Departamento de Educación Musical de la Universidad Pedagógica Nacional, en el que se examina la estructura, el enfoque y la proyección investigativa de los trabajos presentados durante el periodo 2016-2017. Este estudio ofrece una visión amplia de cómo ha cambiado la línea de investigación en dicho periodo, y resulta especialmente importante comprender tanto los aspectos teóricos como prácticos de la investigación en el ámbito artístico. Entre los hallazgos más relevantes, se destaca que el 74% de los trabajos de grado se centraron en la creación de materiales, mientras que un 16% abordaron reflexiones teóricas y el 10% restante se enfocó en aplicaciones prácticas, lo cual evidencia una tendencia marcada hacia la producción de recursos didácticos en este campo.

La revisión detallada de los antecedentes relacionados con el trabajo propuesto por el grupo "Educación y Regionalización en CTeI" ha sido abordada con especial atención a su línea de investigación sobre la formación de docentes. Estos antecedentes se integran de manera significativa en el presente estudio, proporcionando un marco sólido que contextualiza la relevancia de nuestra investigación.

En particular, los antecedentes revisados aportan valiosas perspectivas sobre la definición y delimitación de las líneas de investigación. Se identificó que la formación de docentes, tal como se ha investigado en trabajos anteriores, no solo se limita a aspectos pedagógicos, sino que también aborda la comprensión más amplia de las "necesidades" en la educación actual. Estos antecedentes han enriquecido la comprensión al destacar

la interconexión entre la formación de docentes y las necesidades educativas, proporcionando así un fundamento sólido para nuestra propia investigación. Al situar nuestro trabajo en este contexto, buscamos contribuir a la literatura existente al expandir la comprensión de las necesidades educativas en el ámbito de la formación de docentes.

Esta revisión fortalece la coherencia y relevancia de nuestro estudio en el marco de las investigaciones previas realizadas por el grupo "Educación y Regionalización en CTeI", consolidando así la contribución de nuestro trabajo a la comprensión más amplia de las dinámicas educativas y sus necesidades subyacentes

Durante el año 2013, se impulsaron proyectos centrados en la formación docente desde un enfoque investigativo, con el propósito de fortalecer la cultura en Ciencia, Tecnología e Innovación (CTeI) entre la población infantil y juvenil de Cundinamarca y Bogotá. Esta iniciativa incluyó la capacitación de docentes en el desarrollo de proyectos escolares liderados por estudiantes y maestros, así como acciones para divulgar y apropiar socialmente el conocimiento en municipios no certificados y en todas las localidades de la ciudad.

Entre 2014 y 2016, se continuó con esta línea de trabajo a través del proyecto Formación en Ciencia, Tecnología e Innovación en la comunidad educativa de las instituciones educativas de los municipios no certificados de Cundinamarca. Esta investigación tuvo como finalidad promover prácticas investigativas en docentes y directivos, logrando recopilar más de mil experiencias de investigación realizadas por maestros en distintas zonas del departamento.

[1] En el año 2017 se llevó a cabo el proyecto Producción y gestión de conocimiento en la investigación educativa en contexto en Escuelas Normales Superiores de Cundinamarca, cuyo propósito fue analizar cómo se desarrollaba la producción investigativa en el ciclo complementario de estas instituciones entre los años 2010 y 2015. Los hallazgos del estudio mostraron una relación entre los procesos de formación en investigación, la práctica pedagógica y las actividades de extensión. A partir de estos resultados, en 2018 se

formuló un nuevo proyecto denominado "La extensión como eje en la formación de maestros en Escuelas Normales Superiores de Cundinamarca durante el periodo 2013-2018", enfocado en identificar las dinámicas establecidas entre estas instituciones de formación de docentes y las comunidades locales, con el fin de comprender el papel de la extensión en la preparación profesional de los futuros maestros.

[1] En el 2021, el grupo orientó su labor hacia una mirada contextualizada y situada de los procesos de formación en investigación en los niveles de educación básica y media. Se trabajó colaborativamente con docentes egresados de la Universidad Pedagógica Nacional, con el propósito de identificar las diversas propuestas y estrategias pedagógicas que se han implementado en el desarrollo de proyectos escolares. Esto permitió explorar las prácticas en las escuelas de Cundinamarca, como indagar en los posibles vínculos con la formación inicial que los docentes recibieron en la UPN.

[1] Actualmente, el grupo "Educación y Regionalización en CTeI" adelanta el proyecto denominado “Aproximación a los discursos, perspectivas y enfoques de la educación en tecnología en el marco de la formación de maestros en Colombia”, proyecto que aporta insumos importantes alrededor de los discursos que orientan la educación en tecnología en los programas de formación de maestros en el país y las perspectivas epistemológicas de educación en tecnología que subyacen en la política nacional e internacional. Los resultados de este trabajo permitirán caracterizar los enfoques del desarrollo de la educación en tecnología que se evidencian en los programas de formación de maestros en el país, además de proporcionar al programa un referente importante frente a las discusiones relacionadas con la pertinencia curricular y los procesos de acreditación

6. MARCO TEÓRICO

El desarrollo de un repositorio digital académico requiere de la comprensión y articulación de diversos conceptos clave que sustentan tanto su diseño como su implementación. En este sentido, el presente marco

teórico tiene como finalidad establecer los fundamentos teóricos y técnicos que dan soporte al proyecto, así como los elementos que intervienen alrededor del proyecto establecido.

6.1 Aspecto Académico

6.1.1 Proyecto de trabajo de grado

El Proyecto de Trabajo de Grado (PTG) es un documento que los estudiantes presentan ante Comité Curricular de cualquier programa de la Universidad Pedagógica Nacional. Este documento representa la propuesta inicial que el estudiante debe presentar ante el Comité Curricular del programa, con el propósito de obtener autorización para cursar el espacio académico Trabajo de Grado I.

En dicho proyecto, el estudiante debe plantear de forma clara, precisa y sintetizada la temática que desea abordar en su trabajo de grado, así como los propósitos que orientarán su desarrollo. Además, es indispensable que esta propuesta cuente con el respaldo de un docente de la Licenciatura, quien no solo avala el contenido del documento, sino que también asume formalmente la dirección del trabajo de grado, comprometiéndose a acompañar al estudiante durante el proceso de investigación y redacción.

Este paso marca el inicio formal de una etapa clave en la trayectoria académica del estudiante, ya que define el enfoque del trabajo de grado y establece las bases metodológicas y temáticas sobre las cuales se desarrollará. (Dulzaides & Molina, 2004)

6.1.2 Líneas de investigación

Las líneas de investigación constituyen estructuras temáticas que orientan y organizan la labor investigativa dentro de una institución académica. Estas líneas, se configuran a partir de un conjunto de objetivos,

principios metodológicos y criterios científicos enfocados en el análisis y solución de problemas concretos en un campo del conocimiento determinado y se usan para favorecer la generación de nuevo conocimiento mediante una planificación coherente y sistemática del trabajo investigativo.

Las líneas de investigación son estructuras flexibles que pueden ser modificadas, fortalecidas o incluso ser descartables, dependiendo de las valoraciones periódicas que puedan adaptarse a los cambios en el entorno científico o social, lo que hace mantener su esencia. Las líneas de investigación, en un concepto amplio, funcionan como marcos organizadores que ubican y dan coherencia al trabajo investigativo, permitiendo una continuación en la generación de conocimiento por parte de individuos, colectivos o instituciones. (Celi, 2018)

6.1.3 Sistema de información

Un sistema de información puede entenderse como un conjunto estructurado de elementos, entre los que se incluyen el hardware, el software, los datos, los procedimientos y el capital humano. Este se diseña para recolectar, almacenar, procesar y distribuir información útil dentro de una organización. Estos sistemas tienen como propósito principal facilitar la toma de decisiones, optimizar la gestión y respaldar las actividades operativas y estratégicas de la entidad donde se implementan.

Desde una perspectiva más técnica, un sistema de información opera a partir de datos organizados, los cuales son procesados según criterios previamente establecidos para generar información pertinente y oportuna, el flujo de información, puede ser evaluado mediante mecanismos de retroalimentación que permiten ajustar el sistema en función de la calidad y relevancia de los resultados obtenidos.

Autores como Andreu, Ricart y Valor (1991), citados por Trasobares (2003), definen un sistema de información como:

Conjunto formal de procesos que, operando sobre una colección de datos estructurada de acuerdo a las necesidades de la empresa, recopila, elabora y distribuye selectivamente la información necesaria para la operación de dicha empresa y para las actividades de dirección y control correspondientes, apoyando, al menos en parte, los procesos de toma de decisiones necesarios para desempeñar funciones de negocio de la empresa de acuerdo con su estrategia. (Trasobares, 2003, p. 1)

Los procesos tienen como finalidad recopilar, transformar y distribuir selectivamente la información requerida tanto para el funcionamiento operativo como para la dirección y el control institucional, apoyando de manera directa la toma de decisiones estratégicas.

La eficacia de un sistema de información reside en su capacidad para integrar de manera armónica sus diversos componentes, contribuyendo así a un manejo más eficiente de los recursos informativos y al fortalecimiento de los procesos administrativos y académicos en los que se aplica. (Trasobares, 2003)

6.1.4 Gestión del conocimiento

Actualmente, el conocimiento se entiende como un recurso fundamental que facilita la comprensión del entorno y también proporciona capacidad de intervención. La gestión del conocimiento se centra en mejorar el uso de estos recursos creando un entorno adecuado que facilite su circulación. En realidad, lo que se rige no es el conocimiento, sino las condiciones que posibilitan su creación y transmisión.

La gestión del conocimiento utiliza herramientas específicas que permiten optimizar el uso y circulación de la información, cuyo propósito es identificar, recopilar, sistematizar y compartir el conocimiento explícito y tácito. Esta práctica aborda cómo establecer procedimientos efectivos que garanticen el acceso a la información necesaria para todos los miembros de la agencia. También fomenta la colaboración entre equipos, la circulación activa del conocimiento y su uso práctico en la toma de decisiones, con el objetivo

de construir una cultura organizacional que promueva el aprendizaje continuo, la innovación y la mejora continua, reconociendo el conocimiento como un recurso esencial para el crecimiento sostenible a largo plazo y el logro de objetivos a largo plazo

Un sistema de información se considera eficaz cuando es capaz de articular una variedad de elementos, permitiendo un manejo efectivo de los recursos de información. Esta articulación fortalece las tareas administrativas y los procesos académicos en los que se implementa el sistema. (Canals, 2003)

6.1.5 Búsqueda de información

Al iniciar cualquier proceso de búsqueda de información, es fundamental reflexionar sobre el propósito específico que se le dará a dicha información. No es lo mismo indagar para resolver una duda cotidiana, desarrollar una actividad académica, o emprender una investigación con mayor nivel de profundidad, ésto permite orientar de manera más precisa los esfuerzos y recursos destinados a la búsqueda.

Uno de los principales desafíos radica en la sobreabundancia de resultados que pueden generarse con una simple consulta, dado que resulta prácticamente imposible revisar todos los contenidos disponibles, se hace necesario delimitar tanto la cantidad como el tipo de documentos requeridos. De esta manera, se optimiza el tiempo invertido y se mejora la calidad del proceso de recuperación de información.

Para enfrentar esta situación de forma eficaz, se recurre a lo que se conoce como estrategia de búsqueda, entendida como el conjunto organizado de procedimientos y acciones que un usuario pone en práctica con el objetivo de localizar información pertinente. Esta estrategia se fundamenta en cinco etapas esenciales: como son la definición de la búsqueda, preparación de la búsqueda, selección de las fuentes documentales, elaboración de la ecuación de búsqueda y la recuperación de la información. (Biblioteca Central UNAM,

2021).

6.1.6 Acceso abierto

Aunque un recurso esté disponible en línea, no necesariamente puede utilizarse libremente, para que su uso sea legítimo dentro del contexto del Acceso Abierto. Este debe estar acompañado de una licencia adecuada, como las licencias Creative Commons, que especifique explícitamente los permisos otorgados, siempre bajo la condición de reconocer la autoría original para evitar el plagio.

Cuando se accede a un recurso en Internet que no cuenta con una licencia de uso explícita, este se considera protegido por derechos de autor, lo que impide su utilización legal en otros contextos, como su integración en un Objeto Educativo Abierto (OEA), según advierten Ramírez Montoya y García-Peñalvo (2015) citados por el mismo García-Peñalvo (2017) en “Mitos y Realidades del Acceso Abierto”.

Es importante diferenciar entre estar disponible en línea y estar abierto al uso: el simple acceso digital no implica que sea abierto o gratuito, y que algo sea gratuito no significa automáticamente que esté abierto al uso, modificación o redistribución.

Los contenidos verdaderamente abiertos permiten a cualquier persona utilizarlos, adaptarlos y compartirlos libremente, siempre que se respeten las condiciones establecidas por la licencia con la que fueron publicados. (García, 2017)

6.1.7 Procesamiento de lenguaje natural (PNL)

Una de las funciones esenciales dentro del campo de la Inteligencia Artificial (IA) es el tratamiento de lenguajes naturales mediante herramientas computacionales. En este contexto, los lenguajes de programación desempeñan un rol importante, ya que actúan como el puente entre el lenguaje humano y su procesamiento por parte de las máquinas. El Procesamiento de Lenguaje Natural (PLN) se basa en emplear

lenguas humanas para establecer una comunicación con el computador, el cual debe ser capaz de interpretar correctamente las frases que se le suministren.

El lenguaje natural en sistemas informáticos facilita el diseño de programas que pueden ejecutar tareas lingüísticas o que permiten crear modelos para analizar cómo los seres humanos procesan el lenguaje. No obstante, el empleo del lenguaje natural en la interacción entre personas y máquinas constituye tanto una ventaja como un desafío frente a otros métodos de comunicación, debido a la complejidad de la interpretación de dichos lenguajes. (Vásquez, Quispe & Huayna, 2009)

6.1.8 Expresiones regulares

Las expresiones regulares son herramientas sumamente valiosas para quienes se dedican a la programación, independientemente de su género, su aplicación permite resolver de forma eficiente diversos problemas dentro de los algoritmos de los programas, aumentando así la potencia y versatilidad de las soluciones desarrolladas.

Aunque en un principio puedan parecer complejas, las expresiones regulares se vuelven comprensibles una vez que se entiende su funcionamiento. Por ello, se anima a los programadores a seguir profundizando en su estudio y dominio.

Las expresiones regulares utilizadas en Perl tienen su origen en una herramienta denominada regex, desarrollada por el pionero Henry Spencer (https://en.wikipedia.org/wiki/Henry_Spencer). A partir de este trabajo se creó la biblioteca PCRE (Perl Compatible Regular Expressions), la cual es ampliamente empleada en herramientas modernas de programación. (Sguerra, 2006)

6.2 Aspecto tecnológico

6.2.1 Python

Python se considera un lenguaje de programación moderno y avanzado, con soporte para la programación orientada a objetos, su versatilidad le permite ser útil en múltiples contextos como el desarrollo de software empresarial, académico, videojuegos, aplicaciones web y, más recientemente, en áreas emergentes como la inteligencia artificial. Además, cuenta con un extenso respaldo por parte de la comunidad y una gran variedad de bibliotecas que fortalecen tanto su uso básico como su implementación a gran escala.

Python es un lenguaje de propósito general, interpretado y clasificado como de alto nivel, lo que lo hace accesible y potente a la vez. Su desarrollo fue iniciado por Guido van Rossum y su primera aparición pública se dio en 1991. Una de sus principales características es la simplicidad de su sintaxis, lo cual favorece tanto el aprendizaje como la escritura de código eficiente. Gracias a estas características, se emplea en una gran diversidad de proyectos, como aplicaciones científicas, automatización de procesos, páginas web y desarrollo de software en general. (Supelano, 2023)

6.2.2 Framework

Dentro del ámbito del desarrollo de software, un framework se entiende como una estructura previamente establecida que facilita la organización y construcción de nuevos proyectos, en términos de programación, se trata de un conjunto de herramientas y funciones reutilizables que automatizan tareas habituales como la creación de objetos o la conexión a bases de datos. El entorno proporciona una base robusta para construir aplicaciones específicas, permitiendo a los desarrolladores evitar la repetición de tareas rutinarias y enfocarse más en el análisis de los requerimientos del sistema.

Los frameworks permiten escalar fácilmente las aplicaciones, facilitando su administración conforme

aumentan en complejidad y tamaño con esto promueven buenas prácticas que garantizan coherencia en el código y colaboración efectiva entre distintos programadores, lo cual contribuye a un mantenimiento más ágil a largo plazo. Algunos frameworks destacados son Django y Flask para aplicaciones web con Python, Ruby on Rails para Ruby, y Angular y React para crear interfaces en JavaScript. estos entornos de trabajo resultan fundamentales porque proporcionan una arquitectura firme que agiliza el desarrollo y permite que los programadores se enfoquen en la lógica propia de sus soluciones. (Villalobos, Sánchez & Gutiérrez, 2010)

6.2.3 Django

Django es un framework de desarrollo web basado en la arquitectura Modelo Vista Controlador (MVC). Esta arquitectura facilita un trabajo ágil y eficiente, además de la reutilización de código. La interfaz de usuario está separada de la gestión de datos y la lógica de negocio, lo que simplifica el proceso de desarrollo.

La seguridad es un aspecto importante de Django, ya que incluye mecanismos integrados que ayudan a prevenir vulnerabilidades como ataques de SQL, CSRF (falsificación de solicitud entre sitios) y XSS (scripts entre sitios). El framework proporciona un conjunto de métodos y conceptos automatizados que simplifican su interpretación mediante ajustes predefinidos, reduciendo así la necesidad de una configuración manual repetitiva.

Django se utiliza en el desarrollo de diversas aplicaciones web, desde pequeños sitios web hasta proyectos grandes y complejos. Su diseño robusto, sus capacidades de gestión de bases de datos y sus funciones de seguridad integradas, lo convierten en una opción confiable para proyectos que requieren un desarrollo ágil y una buena organización. Algunos ejemplos destacados de sitios web creados con Django incluyen Instagram, Pinterest y el Washington Times. (Barrera & Barros, 2017)

6.2.4 Repositorio Digital

Un repositorio digital es una plataforma en línea que almacena, organiza y comparte recursos digitales, como documentos, imágenes o datos, garantizando su acceso, preservación y visibilidad. Se determinó que los repositorios institucionales digitales han surgido como un torrente de innovación para las Instituciones de Educación Superior, al utilizar y promover herramientas tecnológicas que, como gérmenes, engendran transformaciones significativas en la gestión y difusión del saber.

Los repositorios institucionales digitales se están posicionando como una de las principales innovaciones dentro de las instituciones de educación superior (IES), ya que no solo implementan herramientas tecnológicas, sino que también impulsan transformaciones importantes en la forma en que se gestiona y se utiliza la información. Al revisar diferentes repositorios digitales, se puede ver que comparten varias características clave, como el tipo de software que usan, la estructura de metadatos, el uso de identificadores normalizados, la forma en que presentan los recursos, la accesibilidad, el manejo de derechos de autor, los mecanismos para subir contenido, los motores de búsqueda y las estrategias de preservación. Sin embargo, también presentan diferencias notables en temas como la interoperabilidad, el proceso de digitalización, la opción de hacer depósitos remotos, el nivel de visibilidad que tienen, el tipo de licencias que aplican y las formas en que difunden la información. (Keefer, 2008)

6.2.5 ORM (Object-Relational Mapping)

Según César (2021), el término ORM (Object Relational Mapping) hace referencia a una técnica de programación que posibilita la interacción con sistemas de gestión de bases de datos relacionales (RDMS), tales como SQL Server, PostgreSQL, MySQL u Oracle, ésto permite representar los datos almacenados en

dichas bases como objetos dentro de un lenguaje de programación orientado a objetos.

De acuerdo con ESIC (2018), un ORM constituye un modelo de programación que convierte las tablas de una base de datos en estructuras que facilitan las tareas habituales de los desarrolladores. En vez de redactar consultas SQL manuales, los programadores pueden usar el ORM para llevar a cabo acciones como insertar, modificar, eliminar o consultar datos utilizando objetos y métodos propios del lenguaje de programación.

El ORM opera mediante objetos en vez de sentencias SQL explícitas, se usa como central para facilitar la conexión entre aplicaciones basadas en la orientación a objetos y las bases de datos relacionales, posibilitando que los desarrolladores trabajen con los datos utilizando clases y objetos dentro del lenguaje que emplean.

Los desarrolladores no necesitan escribir instrucciones SQL de forma manual, ya que el ORM les permite ejecutar acciones como insertar, actualizar, eliminar o consultar datos utilizando estructuras de

programaciones orientadas a objetos. Este sistema traduce automáticamente dichas acciones en consultas SQL facilitando el proceso de desarrollo y mejorando la claridad del código fuente.

6.2.6 Base de datos

Para evitar ambigüedades asociadas a expresiones genéricas como "fuentes de información" o "repositorios de datos", se opta por emplear el término "bases de datos". Estas se definen como colecciones de información estructurada y dispuesta con un fin específico. Asimismo, pueden interpretarse como archivos compuestos por datos relacionados entre sí, recolectados con el objetivo de satisfacer las necesidades informativas de un grupo particular de usuarios. Cada elemento almacenado en la base representa una unidad de información que contiene datos básicos, los cuales describen atributos específicos de una entidad.

Las bases de datos consisten en agrupaciones sistemáticas de datos estructurados cuyo propósito es permitir un almacenamiento eficiente y facilitar su acceso. Pueden ser tanto digitales como físicas, y suelen organizarse mediante archivos o tablas vinculadas que contienen información específica. Entre sus principales atributos destacan la estructura ordenada, la capacidad de recuperar datos rápidamente, la integridad que asegura la precisión, la interrelación entre tablas para manejar información compleja, mecanismos de seguridad para limitar el acceso, la posibilidad de crecer según las necesidades, y la estabilidad garantizada mediante transacciones y respaldos. Gracias a estas características, las bases de datos resultan fundamentales en diversos ámbitos —desde plataformas empresariales hasta aplicaciones móviles y sitios web—, ya que permiten organizar grandes volúmenes de información, optimizar los procesos internos y respaldar decisiones basadas en datos confiables y accesibles. (Rivera, 1994)

6.2.7 Open Source

En sus inicios, el término "código abierto" se utilizaba específicamente para referirse al software open source (OSS), este tipo de software se caracteriza por tener su código fuente disponible para cualquier persona, permitiendo que cualquiera lo examine, lo modifique y lo comparta libremente según sus necesidades.

El desarrollo de software open source se realiza de forma colaborativa y sin un control centralizado, basándose en el trabajo conjunto y la revisión entre miembros de la comunidad, la metodología suele ser más accesible económicamente, más adaptable y con mayor vida útil que los programas propietarios, ya que no depende de una sola entidad para su evolución. Con el tiempo, el open source pasó de ser una simple técnica de desarrollo a convertirse en un movimiento que promueve un modelo descentralizado de creación y solución de problemas más allá del software.

Actualmente, numerosos proyectos open source se encuentran disponibles en plataformas como GitHub, donde cualquier usuario puede colaborar o explorar el código. Los ejemplos que más se destacan del software de código abierto incluyen Linux, Ansible y Kubernetes. Empresas como Red Hat aplican este enfoque colaborativo para desarrollar soluciones tecnológicas empresariales, sus desarrolladores están involucrados en múltiples proyectos open source en todos los niveles del ecosistema tecnológico. La estrategia consiste en partir de soluciones existentes que respondan, parcial o totalmente, a las demandas de sus clientes, y luego mejorar dichos sistemas con nuevas características, mayor seguridad y parches que fortalezcan su funcionamiento.

Finalmente, todas las mejoras implementadas por Red Hat se integran nuevamente en los proyectos open source originales, lo que garantiza que la comunidad en general se beneficie de estas innovaciones. A medida que los usuarios utilizan el software, ofrecen retroalimentación valiosa, informan sobre fallos y proponen nuevas funcionalidades, lo que orienta el desarrollo continuo del software dentro de la empresa. (Red Hat, 2023)

6.2.8 Análisis y extracción de PDF (PyMuPDF)

PyMuPDF es una biblioteca con licencia semiabierta que ofrece una interfaz de programación de aplicaciones (API) en Python para el manejo y procesamiento de documentos en formatos como PDF, XPS, OpenXPS, CBZ, PUB y FB2 (libros electrónicos), así como alrededor de diez formatos de imagen ampliamente utilizados, se caracteriza por su alta velocidad y eficiencia, lo que la convierte en una opción ideal para dispositivos con recursos limitados, como los teléfonos inteligentes, se destaca por su fiabilidad y excelente capacidad de representación de documentos.

La API proporciona funcionalidades tanto básicas como avanzadas para la conversión y manipulación de documentos, entre las cuales se incluyen: conversión de archivos PDF a imágenes PNG, visualización y acceso a metadatos, manejo de contornos de documentos, renderización de páginas en formatos rasterizados o vectoriales (como SVG), búsqueda de texto dentro del documento, extracción de texto e imágenes, modificación y creación de páginas, eliminación de páginas no deseadas e incluso incrustación de datos. PyMuPDF es compatible con múltiples sistemas operativos, incluyendo macOS, Linux y Windows, lo que amplía su aplicabilidad en diversos entornos de desarrollo. (FileFormat Products, 2020)

6.2.9 Interfaz de Usuario (UI)

Es común confundir los términos de interfaz de usuario (UI) y experiencia de usuario (UX). La UI se refiere a los componentes visuales de una aplicación, mientras que la UX se enfoca en cómo se siente y percibe la interacción del usuario con una herramienta. Al diseñar una interfaz, no solo es importante que sea visualmente atractiva; también se deben considerar aspectos que a menudo se pasan por alto, ya que descuidar estos detalles puede perjudicar seriamente la experiencia del usuario. Antes de que la apariencia visual influya, los usuarios valoran más cómo se sienten al usar la herramienta y su nivel de frustración.

Conectados a esa experiencia, los diseñadores deben enfocarse en crear herramientas que sean fáciles de usar y convenientes para los usuarios. Según Jakob Nielsen (2012), la usabilidad es una característica clave de un software que no se limita a una sola dimensión, sino que incluye varios componentes que deben satisfacer cinco condiciones importantes:

El sistema debe tener una interfaz sencilla que facilite su uso, permitiendo al usuario interactuar con él de forma rápida y natural [Aprendizaje]

Se debe garantizar un rendimiento que le permita al usuario incrementar su productividad al implementarlo [Eficiencia].

El uso debe ser fácil de recordar, de modo que el usuario pueda retomar su manejo sin dificultades incluso después de un tiempo sin utilizarlo [Memorable].

El sistema debe estar diseñado para que los errores sean poco frecuentes y, si llegan a ocurrir, el usuario pueda corregirlos con facilidad [Acierto].

Y como última condición, debe ofrecer una experiencia agradable que mantenga al usuario con una sensación de comodidad al utilizarlo [Satisfacción]. (Ramírez, 2017)

6.2.10 Pipenv

Pipenv es una herramienta que nos ayuda a gestionar entornos virtuales en Python. Combina en una sola interfaz lo que normalmente haríamos por separado con pip, virtualenv y Pipfile. Pipenv nos permite crear y gestionar automáticamente entornos virtuales para cada proyecto, y también gestiona la configuración de las librerías que utilizamos en un archivo llamado Pipfile. Genera un archivo Pipfile.lock, que garantiza que el servicio funcione de la misma manera en todas las computadoras, con el mismo conjunto de paquetes instalados. Pipenv ofrece compatibilidad total y soporte completo para los principales sistemas operativos, incluyendo Linux, macOS y Windows. Pipenv facilita la creación de un entorno virtual del sistema

operativo, evitando conflictos con otras instalaciones de Python y protegiendo la integridad del proyecto. (Packaging Authority (PyPA), 2020)

7. METODOLOGÍA

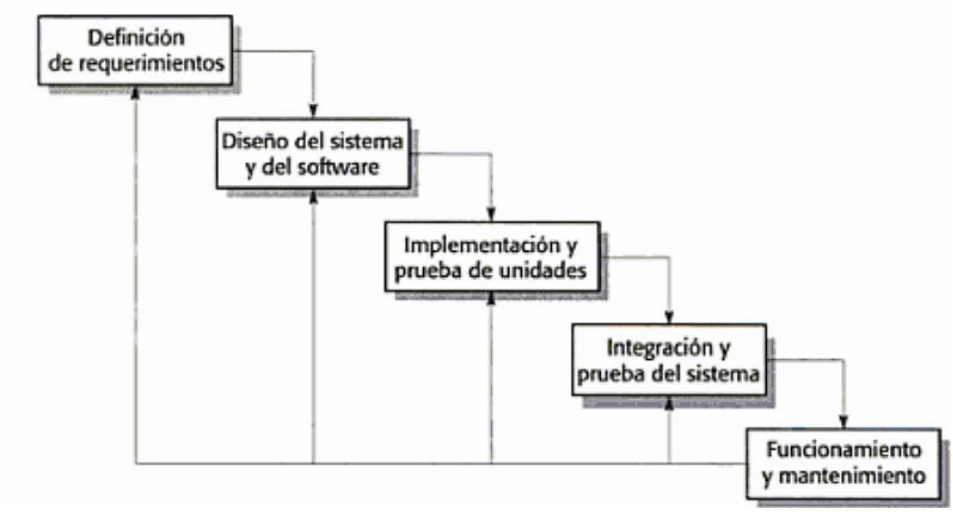


Figura 1. Ciclo de vida del software. Fuente. González, 2014, SISTEMA DE INFORMACIÓN PARA CONSULTAR Y ADMINISTRAR LOS DOCUMENTOS DE LOS TRABAJOS DE GRADO DE LA LICENCIATURA EN ELECTRÓNICA. (pag 18)

La figura 1 muestra la metodología en cascada aplicada al desarrollo software en cinco fases: análisis de requerimientos y herramientas (Fase 1), diseño del sistema y del software (Fase 2), implementación y prueba de unidades (Fase 3), integración y prueba del sistema (Fase 4), y funcionamiento y mantenimiento (Fase 5). Esta última fase se muestra con flechas que regresan a las etapas anteriores, lo cual representa que, durante el uso del sistema, pueden encontrarse errores que requieran algún ajuste en las otras fases.

Fases de la Figura 1

7.1 Fase 1 - Análisis y definición de requerimientos: En esta fase se realizó un análisis de los requisitos del proyecto, para ello se establecieron los criterios para la incluir los trabajos de grado, se definieron los campos de información a extraer de cada documento. También, se evaluó la secuencia de pasos a seguir, así como los requerimientos necesarios para el desarrollo del proyecto. Se emplearon diagramas UML que permitieron una representación estructurada del sistema. Se realizó un análisis de herramientas utilizadas, comparándolas con otras alternativas. Se presentaron las líneas de investigación que se implementaron en el proyecto. Estas fueron definidas en el trabajo de

investigación del semillero Educación y Regionalización en CTeI – GIER.

7.2 Fase 2 - Diseño del sistema y del software: Con los requerimientos definidos, se procedió a crear un sistema para la organización de los trabajos de grado dentro del repositorio, cómo se presentarán al usuario y de qué manera se llevará a cabo la extracción de la información más relevante. Para ello, se usó los diagramas UML como herramienta de modelado, incluyendo el diagrama de casos de uso y el diagrama de clases, que permitirán planear las funcionalidades y la estructura del sistema. Además, se desarrolló un diagrama de funcionamiento que muestre la relación entre el controlador, el modelo y la vista, en concordancia con la arquitectura MVC implementada en Django.

También se realizó un análisis del framework Django, abarcando desde la instalación, hasta la implementación final de la aplicación web. Como parte de este proceso, se diseñaron las interfaces de usuario amigables, y se llevó a cabo la creación del repositorio digital, integrando todos los componentes mencionados de manera coherente y funcional.

7.3 Fase 3 - Implementación y prueba de unidades: En esta fase se procedió con la implementación del repositorio digital, considerando cada trabajo de grado como una unidad funcional dentro del sistema. Para ello, se desarrolló un software encargado de la gestión, organización y análisis de los documentos, aprovechando el uso de la herramienta de Django y con la ventaja que tiene de presentación e interacción con bases de datos.

Una parte fundamental del sistema es la extracción automatizada de información académica, por lo cual se implementó y probó distintos extractores de texto. Esto con el fin de identificar secciones clave de los trabajos de grado: El título, autor, director, palabras clave, metodología, contenidos, conclusiones, fuentes e identificación de la línea de investigación. Durante este proceso se realizaron múltiples pruebas con distintos tipos de documentos, documentando los errores y ajustes necesarios

para optimizar cada extractor.

7.4 Fase 4 - Integración y prueba del sistema: Se integraron todas las partes del sistema y se probó como un sistema completo para asegurar el cumplimiento de los requerimientos. Con esto, se comprueba que cada elemento funcione correctamente. Durante este proceso, los trabajos de grado procesados son almacenados en el repositorio digital y su información estructurada se organizó en una tabla de datos. Esta facilitó la consulta por parte de los usuarios. El repositorio digital también contiene campos de búsqueda que permiten filtrar, clasificar y analizar los contenidos académicos de manera eficiente. Se hizo este proceso con el fin de comprobar que el sistema está funcionando correctamente y hacer los cambios correspondientes

7.5 Fase 5 - Funcionamiento y mantenimiento: El sistema se pone en funcionamiento y necesita de mantenimiento. Esto se realiza cuando se deben corregir errores que no se descubrieron en etapas anteriores o cuando surgen nuevos requerimientos para mejorar el sistema. Debido a que el framework Django posee una estructura flexible, es fácil identificar los posibles errores que este pueda traer por lo tanto no se requiere de mucho tiempo para realizar este paso.

8. RESULTADOS

A continuación, se presentan los resultados obtenidos a lo largo del desarrollo del proyecto, organizados en las diferentes fases que reflejan el proceso de diseño, implementación y validación del repositorio digital. Cada fase evidencia el cumplimiento de los objetivos planteados, mostrando los avances técnicos, metodológicos y funcionales alcanzados, así como las herramientas y enfoques utilizados para garantizar la eficacia del sistema.

8.1 Ejecución Fase 1 - Análisis y definición de requerimientos

Empezando por la primera fase que se muestra en la figura 1, se representa de manera clara y estructurada los actores involucrados y sus interacciones con el sistema, se elaboró un diagrama de casos de uso, una herramienta de modelado UML que facilita la visualización de las funcionalidades del sistema desde la perspectiva del usuario. Este diagrama permitió identificar y organizar las acciones principales que se llevarán a cabo dentro del repositorio

Diagrama de casos de uso

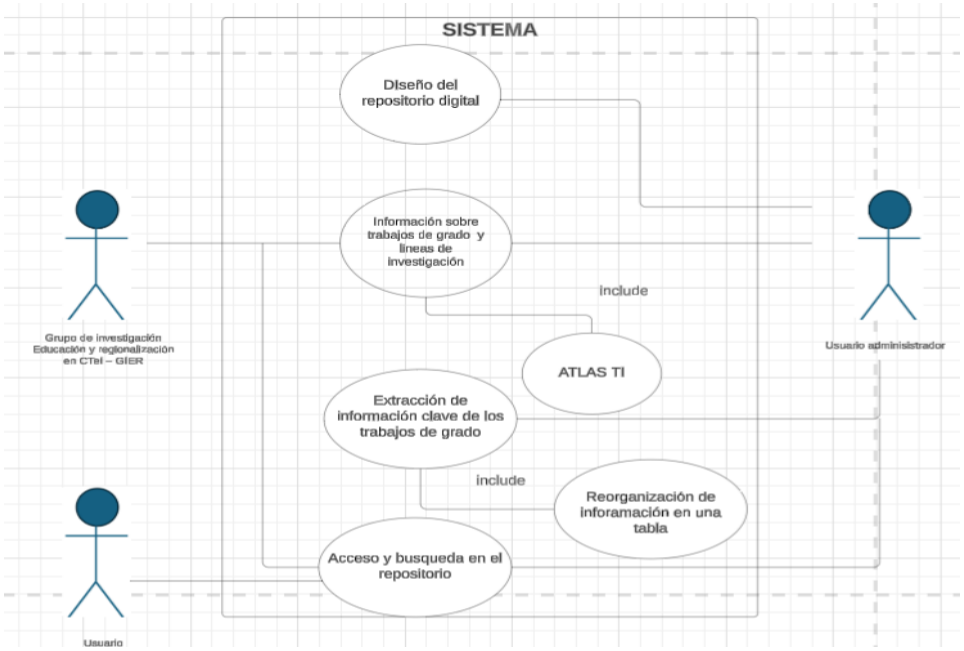


Figura 2. Diagrama UML de casos de uso elaborado en Lucid chart. Fuente. Elaboración propia

El diagrama de la figura 2 muestra tres actores: Grupo de investigación GIER, Usuario y Usuario administrador; y cinco casos de uso principales que componen el sistema a construir: diseño del repositorio digital, carga de información sobre trabajos de grado, extracción de información, reorganización en tablas, y búsqueda en el repositorio. La herramienta ATLAS.ti se incluye como apoyo para la extracción.

La figura 2 se desarrolló con el propósito de representar la arquitectura del sistema planteado. A partir de los requerimientos identificados, se determinaron los casos de uso y se asignaron roles específicos a cada actor.

En el diagrama se distinguen tres actores principales: el Grupo de investigación Educación y Regionalización en CTeI – GIER, el Usuario y el Usuario administrador. El grupo de investigación es el encargado de proporcionar los trabajos de grado y definir las líneas de investigación. El usuario final representa a quienes consultan el repositorio, mientras que el administrador es el encargado de la gestión del contenido.

Los casos de uso se agrupan en torno a procesos clave del sistema. En primer lugar, el diseño del repositorio digital, que establece la estructura que es diseñada por el usuario administrador. Luego, la carga y organización de información sobre trabajos de grado y líneas de investigación, que permite clasificar el contenido por categorías. Después se encuentra la extracción de metadatos de los documentos PDF, donde se incluye la herramienta ATLAS.ti suministrada por el Grupo de investigación Educación y Regionalización en CTeI – GIER. Este análisis da paso a la reorganización de información en una tabla para una mejor visualización. Por último, se incluye el acceso y búsqueda en el repositorio, que constituye el punto de interacción directa del usuario con el sistema.

La figura 2 fue útil para la toma de decisiones, la definición de los flujos de información y la participación de los actores identificados, guiando así el desarrollo coherente del repositorio digital.

Análisis y comparación de Django con otras plataformas

Para el diseño del sistema de información se usó del framework Django para la creación de la estructura del repositorio digital. Cabe recalcar que no se descartaron otras opciones que puedieron resultar ser beneficiosas para la construcción de este proyecto como Omeka, Dspace, Greenstone y Eprints, aplicaciones

web open source que se analizaron. Es de resaltar que para escoger cuál fue la mejor para la realización del repositorio digital. Django cuenta con un método de gestión de datos llamado ORM (Mapeo Objeto-Relacional), que logra la interacción con la base de datos mediante objetos de Python, sin necesidad de escribir sentencias SQL directamente (Vamsi, Lokesh, Reddy & Swetha, 2021).

A través de estos modelos se define la estructura de las tablas y se pueden realizar operaciones como crear, consultar, modificar y eliminar registros, lo cual resulta esencial para construir un repositorio. Su acceso permite que cualquier persona interesada pueda ingresar al sistema y buscar información relevante para sus investigaciones. Por otro lado, trabajar con Django suele ser una gran experiencia para desarrolladores con poca experiencia, gracias a la cantidad de funciones integradas que ofrece y a su compatibilidad con bibliotecas populares como NumPy, Pandas o herramientas de procesamiento de lenguaje natural como NLTK y spaCy. Una de las cualidades más destacadas de Django es su escalabilidad, se entiende como la capacidad del sistema para adaptarse a una gran carga de trabajo sin perder rendimiento. Esto lo hace especialmente adecuado para aplicaciones de gran magnitud o que requieren más tiempo de desarrollo. (Ghimire, 2020).

En la tabla 1, se presenta una comparación entre las plataformas Django, Omeka, Dspace, Greenstone y Eprints a partir de algunos aspectos importantes.

Tabla 1.

Comparación entre Django, Omeka, Dspace, Greenstone, Esprints

Criterio	Django	Omeka	DSpace	Greenstone	Eprints
Lenguaje de programación	Python	PHP	Java	C++	Perl

Tipo de herramienta	Framework web complete	CMS orientado a patrimonio cultural	Plataforma de repositorio institucional	Software para bibliotecas digitales	Plataforma para publicaciones académicas
Personalización	Código complete modificable, diseño y lógica completamente adaptables	Personalización mediante plugins y plantillas	Personalización limitada a configuraciones internas	Plantillas fijas con opciones técnicas	Personalización limitada mediante configuración
Interfaz de usuario	Diseñada a medida con HTML/CSS/JS o frameworks modernos	Plantillas prediseñadas	Interfaz rígida basada en estándares bibliotecarios|	Interfaz básica y anticuada	Interfaz funcional pero poco atractiva
Compatibilidad con IA/NLP	Compatible con bibliotecas como NLTK, SpaCy, OpenAI, PymuPDF	No compatible	No compatible	No compatible	No compatible
Escalabilidad	Escalable a entornos web móviles o API REST	Enfocado a uso web estático	Escalable, pero con requerimientos técnicos complejos	Escalabilidad limitada	Escalabilidad limitada

Nota. Elaboración propia con base en Django Software Foundation, 2025; University Libraries, 2025; Confluence, 2025; Greenstone Wiki, 2023; Greenstone Wiki, 2023.

El desarrollo de un repositorio digital requiere una plataforma que no solo ofrezca técnica, sino también flexibilidad, escalabilidad y compatibilidad con tecnologías modernas de procesamiento de información. En

este sentido, luego de haber comparado Django con otras opciones como se muestra en la tabla 1, llegué a la conclusión que se adapta como la opción más conveniente frente a otras plataformas como Omeka, DSpace, Greenstone o EPrints, por varias razones, dentro de las que se destacan:

- Django está construido sobre Python, uno de los lenguajes más populares y potentes del momento, usado en inteligencia artificial, análisis de datos, automatización y desarrollo web. Python permite integrar bibliotecas como SpaCy, NLTK o PyMuPDF, lo que facilita tareas de extracción semántica o análisis de lenguaje natural, lo cual es fundamental para organizar trabajos de grado por temáticas y líneas de investigación. Por el contrario, plataformas como Omeka (PHP), DSpace (Java), Greenstone (C++) y EPrints (Perl) utilizan lenguajes menos orientados a la IA moderna o con menor soporte en análisis textual.

- Django no es una plataforma cerrada sino un framework de desarrollo web completo, lo que significa que ofrece al desarrollador control total sobre la estructura, el flujo de datos, y las funcionalidades del repositorio. En cambio, Omeka y Greenstone son CMS y bibliotecas digitales con estructuras rígidas prediseñadas, mientras que DSpace y EPrints están orientados exclusivamente a contextos académicos muy específicos, con menor margen de adaptación fuera de ese entorno

- Django permite una personalización tanto a nivel visual como funcional. Gracias al modelo MTV (Modelo-Template-Vista), los desarrolladores pueden construir desde cero las vistas, la lógica de negocio y las estructuras de datos. Esto sobresale de las demás plataformas, que dependen en gran parte de configuraciones preestablecidas, plantillas fijas o plugins limitados, lo que dificulta la integración de herramientas externas o la adaptación a necesidades específicas del usuario final (González-Pérez et al., 2020).

- Django permite diseñar interfaces modernas utilizando HTML, CSS, JavaScript adaptándose a cualquier

tipo de dispositivo. Las plataformas como DSpace o Greenstone, en cambio, poseen interfaces más anticuadas, menos amigables para el usuario final y con pocas opciones de rediseño sin intervención técnica

- A diferencia de las otras plataformas, Django permite integrar bibliotecas de inteligencia artificial o procesamiento de lenguaje natural como OpenAI, SpaCy, GPT, PyMuPDF o Tesseract. Esto lo convierte en una herramienta ideal para analizar, categorizar y extraer automáticamente datos clave de los documentos, lo cual lo convierte en la mejor opción para este proyecto. Las otras plataformas no fueron diseñadas para este tipo de integración y, en muchos casos, ni siquiera ofrecen soporte directo para Python
- Django cuenta con una comunidad global muy activa, documentación extensa y actualizaciones frecuentes, lo que garantiza estabilidad a largo plazo. También al estar respaldado por el lenguaje Python, donde los problemas técnicos pueden ser difíciles se puede encontrar información clave para la resolución de problemas (Django Software Foundation, 2025).

Líneas de investigación

Es de resaltar que este proyecto proviene de la investigación "Análisis de la producción de conocimiento en la formación de Licenciados en Diseño Tecnológico y Electrónica de la Universidad Pedagógica Nacional. Una mirada desde los trabajos de grado entre 2017 y 2022", desarrollado por el grupo de investigación Educación y Regionalización en CTeI – GIER.

En ese contexto, se analizaron un total de **201 trabajos de grado**, de los cuales **85 corresponden a la Licenciatura en Electrónica** y **116 a la Licenciatura en Diseño Tecnológico** del periodo del 2017 al 2022. A través de un riguroso análisis documental y el uso del software **Atlas.TI**, se establecieron líneas de investigación específicas para cada programa, las cuales sirvieron como categorías analíticas que orientan

la organización, interpretación y proyección de futuros trabajos de grado.

El análisis, basado en una metodología cualitativa y apoyado en el software **Atlas.TI**, En este proceso, los códigos obtenidos de los trabajos fueron reagrupados mediante un procedimiento de **codificación axial**, lo que permitió identificar categorías centrales. Estas categorías fueron consideradas como **líneas de investigación** propias de cada programa, ya que agrupan conjuntos de temas afines bajo una estructura analítica. Es importante señalar que, aunque la categoría "Metodologías" aparece en el listado, no se considera una línea de investigación, sino que da cuenta de los enfoques y métodos utilizados en los trabajos. Por ejemplo, en la Licenciatura en Electrónica, la línea "Análisis – Técnica" está compuesta por seis **categorías generales**: Arte Rupestre, Análisis Radio - Sutatenza, Comparación de Modelos, Diseño de material educativo, Encuestas y Proceso Técnico – Producción. (Educación y Regionalización en CTeI – GIER, 2024)

Para **la Licenciatura en Electrónica**, se identificaron las siguientes líneas de investigación:

- Análisis Técnica
- EduTech
- Experiencias ETIAE/MTIAE
- Metodologías
- Productos – Prototipos – Tecnológicos
- Sistemas de Control
- Tecnologías Digitales

De forma similar, las demás líneas como EduTech, Productos – Prototipos – Tecnológicos, Sistemas de Control o Tecnologías Digitales, también agrupan diferentes **categorías generales** diferentes. Este mismo

modelo de organización se replica para los trabajos de las licenciaturas en Diseño Tecnológico y Tecnología. Toda esta estructura se encuentra documentada en el Informe Final del Proyecto de Investigación

Para **la Licenciatura en Diseño Tecnológico**, las líneas definidas fueron:

- Análisis Técnica
- Diseño de Prototipos
- Educación en y con tecnología
- Experiencias ETIAE/MTIAE
- Herramientas Digitales
- LDT
- Metodologías
- Monografías
- Propuesta Disciplinar

Estas líneas de investigación no solo organizan los trabajos de grado según sus enfoques temáticos y metodológicos, sino que también permiten fortalecer los semilleros de investigación y dar continuidad a procesos investigativos que enriquecen la formación docente en el Departamento de Tecnología. (Educación y Regionalización en CTeI – GIER, 2024)

Aunque el informe desarrolla con mayor profundidad las licenciaturas en Diseño Tecnológico y Electrónica, se **reconoce que la Licenciatura en Tecnología comparte estas mismas líneas de investigación**, ya que hace parte del mismo contexto institucional y académico del Departamento de Tecnología.

8.2 Ejecución Fase 2 - Diseño del sistema y del software

8.2.1 UML Diagrama de clases

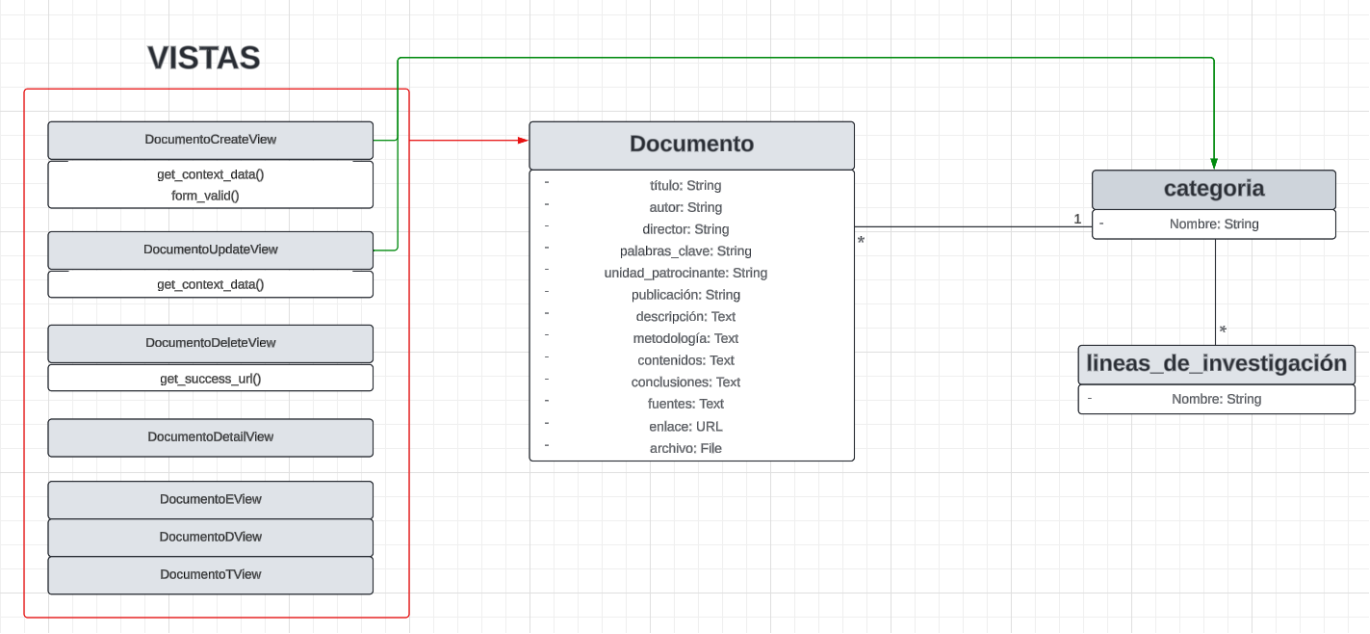


Figura 3. Diagrama de clases UML. Fuente. Elaboración propia

La Figura 3 representa el diagrama de clases del sistema, en el cual se modela la estructura del repositorio digital. Se destaca la clase Documento como clase principal, que centraliza los atributos esenciales de cada trabajo de grado, permitiendo su almacenamiento y consulta de manera ordenada y estructurada.

El sistema está centrado en una sola clase principal: **Documento**. Esta contiene algunos atributos esenciales que describen gran parte de un trabajo de grado. Esta clase encapsula tanto la información descriptiva (título, autor, metodología, etc.), como los componentes técnicos de cada documento (archivo y enlace).

La clase **Documento** se relaciona con categoría mediante una asociación de uno a muchos, permitiendo clasificar cada trabajo de grado en una categoría específica. A su vez, cada **categoría** está vinculada a múltiples **líneas_de_investigación**, lo que facilita la organización temática y el filtrado de documentos.

En cuanto a la interacción del sistema, las clases de la sección **Vistas** (DocumentoCreateView, DocumentoUpdateView, DocumentoDeleteView, DocumentoDetailView, DocumentoEView,

DocumentoDView y DocumentoTView) representan los controladores que gestionan la creación, modificación, eliminación, visualización y filtrado de documentos. Estas vistas dependen de la clase Documento para acceder y manipular su información, y en el caso de las vistas de creación y edición, también interactúan con las clases categoría y líneas_de_investigación para cargar opciones y establecer relaciones.

La clase **Documento** representa cada uno de los trabajos de grado almacenados en el sistema. Este modelo fue diseñado con el objetivo de estructurar la información relevante de manera clara y ordenada. Entre sus atributos principales se encuentran:

título: corresponde al nombre del trabajo de grado.

autor: campo de texto que puede contener uno o dos nombres, dependiendo de cuantos estudiantes realizaron el documento. Se optó por mantener este atributo como texto para simplificar la base de datos, ya que siempre hay uno o dos autores, y no es necesario normalizar esta información.

director: persona encargada de la dirección académica del trabajo.

palabras_clave, unidad_patrocinante, publicación: metadatos que complementan la descripción del trabajo.

descripción, metodología, contenidos, conclusiones, fuentes: campos de texto que permiten almacenar de forma estructurada las secciones clave de cada documento.

archivo: permite el acceso al trabajo original, desde el almacenamiento local para la extracción de metadatos.

enlace del repositorio: permite el acceso al trabajo original desde el repositorio de la universidad.

Este diseño mostrado en la figura 3, permite organizar, consultar y clasificar eficientemente los documentos

dentro del repositorio digital. Al concentrar la información en una sola clase, se logra un modelo de datos **más simple**, fácil de mantener y flexible para los requerimientos del sistema.

Asimismo, Documento se relaciona con las clases Categoría y Líneas de Investigación, que permiten clasificar los trabajos según el programa académico y su enfoque temático. La categoría refleja los tres programas del Departamento de Tecnología: Licenciatura en Electrónica, Diseño Tecnológico y Tecnología, y sus respectivas líneas de investigación como se muestra en la sesión 8.1.3.

Se consideró y evaluó la posibilidad de modelar a los autores como una clase aparte, sin embargo, se concluyó que no era necesario incluir esto al sistema, debido a que:

- Cada documento tiene un máximo de **dos autores**, los cuales pueden almacenarse conjuntamente en un solo campo.
- El sistema no requiere realizar búsquedas independientes de los autores.
- Se priorizó un diseño limpio y flexible que facilite su implementación y mantenimiento.

Esta clase **Documento** se implementó en el código, en el apartado de models.py.

El diagrama MTV de Django

Según Condori Ayala (2012), aunque Django suele identificarse como un framework basado en el patrón de arquitectura MVC (Modelo-Vista-Controlador), en realidad adopta una estructura particular conocida como MTV: Modelo, Template (Plantilla) y Vista. Esta organización mantiene similitudes conceptuales con MVC, pero adapta la función y la terminología a su propio enfoque de desarrollo.

Django puede considerarse parte de la tercera generación de tecnologías para el desarrollo web, ya que no se diseñó estrictamente para seguir un patrón tradicional, sino con la finalidad de ofrecer una solución práctica y altamente funcional para construir aplicaciones web eficientes.

Comprender cómo se articulan los componentes de Django resulta esencial para dominar el framework. En este sentido, es útil establecer una analogía con el patrón MVC tradicional:

- **Modelo** en Django cumple la misma función que es de representar y gestionar los datos.
- **Vista** en Django corresponde al componente que en MVC se denomina "Controlador".
- **Plantilla (Template)** en Django corresponde a la presentación al usuario que en MVC se conoce como "Vista".

Ya entendiendo esto se articula un diagrama para entender mejor el funcionamiento de MTV:

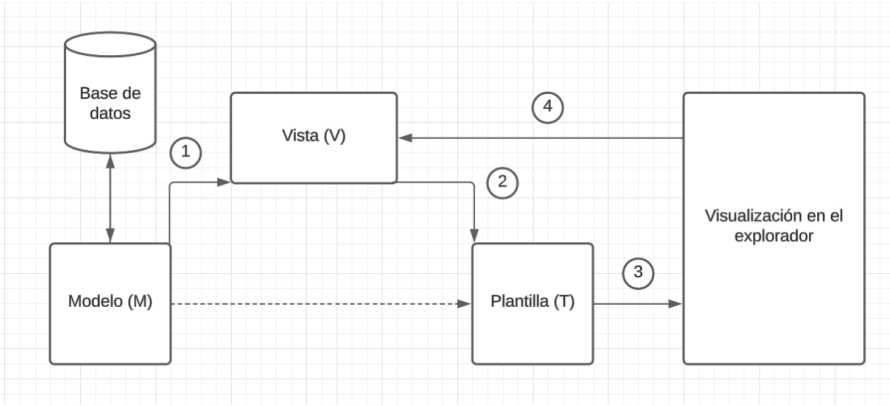


Figura 4. Diagrama MTV de Django. Fuente. Condori, 2012, Revista Boliviana de Ciencia y Tecnología. Adaptado de: http://www.revistasbolivianas.ciencia.bo/scielo.php?pid=S1997-40442012000200016&script=sci_arttext&tlng=es

La figura 4 muestra el flujo de información en una aplicación Django basada en el patrón MTV. El modelo accede y gestiona los datos desde la base de datos; la vista procesa la lógica del sistema y comunica la información al usuario; y la plantilla se encarga de mostrar los datos de forma visual para su presentación en el navegador.

Las flechas numeradas indican: (1) la comunicación del modelo hacia la vista, (2) el paso de datos de la vista a la plantilla, (3) el envío del contenido renderizado al navegador y (4) la respuesta directa de la vista

al explorador cuando no se requiere renderizado adicional.

El navegador realiza una solicitud, la vista procesa esta solicitud y se comunica con el modelo para obtener los datos requeridos. Después la vista delega a la plantilla la responsabilidad de presentar esa información y por último la plantilla genera la respuesta que será enviada de nuevo al navegador.

8.2.1.1 MODELO (MODELS.PY)

El modelo en Django representa la estructura de los datos que serán almacenados en la base de datos. Estos pueden incluir métodos que permiten manipular o procesar la información. Esta abstracción permite definir y controlar la lógica relacionada con los datos de forma clara y concisa. (Condori Ayala, 2012)

8.2.1.2 VISTA

Las vistas en Django se implementan como funciones o clases en Python y su propósito principal es determinar qué datos se deben mostrar y cómo se deben procesar. Gracias al ORM (Object-Relational Mapping) integrado, es posible realizar consultas a la base de datos utilizando únicamente código Python, sin necesidad de escribir instrucciones SQL directamente. Las vistas también pueden encargarse de tareas como es el envió de correo electrónico, la validación de formularios, la autenticación de usuarios o la interacción con servicios externos. Es importante destacar que las vistas no manejan el aspecto visual de la información solo se encarga del proceso de los datos, esa responsabilidad recae en las plantillas. (Condori Ayala, 2012)

8.2.1.3 LAS PLANTILLAS (HTML)

Las plantillas constituyen la capa encargada de la presentación. Usualmente se trata de archivos HTML con etiquetas y sintaxis propia de Django, lo que permite insertar contenido dinámico generado por las vistas. Aunque están orientadas principalmente a la generación de HTML, también pueden utilizarse para producir otros tipos de archivos como XML, JSON, CSS, JavaScript o incluso CSV. Su función esencial es

transformar los datos entregados por la vista en una representación visual adecuada para el usuario. (Condori Ayala, 2012)

Creación del aplicativo web hasta la creación del superusuario

El desarrollo del repositorio digital se inició con la preparación del entorno de trabajo como ya se mencionó, se usó el framework Django. La elección de Django responde a su arquitectura basada en el patrón Modelo–Vista–Controlador (MVC) que se muestra en la figura 4, su integración con bases de datos relacionales como SQLite y su amplia documentación web. Iniciamos con la instalación de herramientas y la ejecución de varios comandos en Windows para empezar a trabajar con el entorno virtual.

1- Se decidió comenzar con la instalación del lenguaje de programación de Python a través de la página web de Anaconda. Luego, se procedió instaló Spyder para ejecutar comandos en Python desde el enlace https://anaconda.org/anaconda/spyder.

2- Se verificó la instalación de Python desde la terminal de Windows. Inicialmente, no se detectó correctamente, por lo que fue necesario descargar e instalar la última versión desde su página oficial https://www.python.org/downloads/. Posteriormente, se muestra la versión instalada mediante la consola como se comprueba en la siguiente imagen.

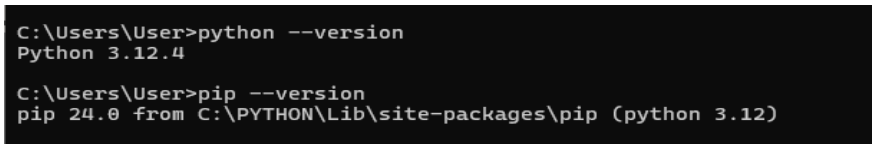


Figura 5. Comprobación de versión de Python y pip en terminal de Windows. Fuente: Elaboración propia

Como se puede observar en la figura 5, la versión de Python que se descargo es la 3.12 y aparece en el sistema para validar que funcione en Spyder, se habilitó la opción para añadirlo a las variables de entorno.

3- Se instalo Django desde la terminal de Windows usando el comando ***python -m pip install django***

el cual debe salir un mensaje al final como el siguiente:

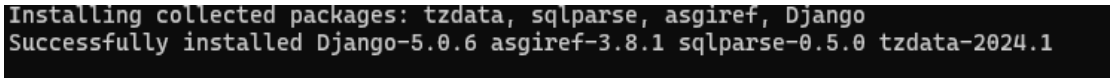


Figura 6. Mensaje final cuando se instala Django. Fuente: Elaboración propia

La Figura 6 muestra la instalación exitosa de Django y sus dependencias. Se confirma la instalación de Django en su versión 5.0.6, junto con los paquetes internos necesarios para su funcionamiento: asgiref (3.8.1), sqlparse (0.5.0) y tzdata (2024.1), lo que indica que el entorno está correctamente configurado para comenzar el desarrollo del proyecto web.

4- Se instaló **Pipenv** desde la consola de Windows como parte de la preparación del entorno de desarrollo del proyecto. Esta herramienta se eligió porque permite manejar de forma más sencilla los entornos virtuales y las dependencias que necesita el proyecto. Al usar **Pipenv**, se generan dos archivos importantes: **Pipfile**, donde se registran las librerías instaladas, y **Pipfile.lock**, que garantiza que todos los que trabajen en el proyecto usen las mismas versiones de esas librerías, por ejemplo, Python 2 puede seguir siendo ejecutada en una nueva versión como Python 3. Esto ayuda a que el proyecto sea más estable y funcione igual en cualquier equipo.

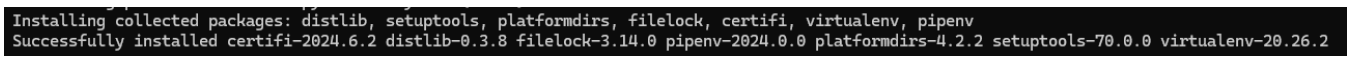


Figura 7. Instalación de pipenv en consola. Fuente: Elaboración propia

La Figura muestra la instalación automática de varios paquetes esenciales que conforman el entorno de trabajo de Pipenv. Entre ellos se encuentran: distlib, setuptools, platformdirs, filelock, virtualenv y pipenv. Estos paquetes permiten la creación y gestión de entornos virtuales de forma controlada, asegurando la compatibilidad entre dependencias durante el desarrollo del proyecto.

Se usa el comando ***-m pip install pipenv*** para la instalación y debe mostrar como en la figura 7 después de

finalizar la instalación.

5- Cuando haya finalizado la instalación creé una carpeta llamada **PROYECTOSDJANGO** donde alojé otra carpeta llamada **PROYECTOREPOSITORIO** que es donde se va ejecutar el entorno virtual. La ruta que se escogió fue la carpeta de documentos y de ahí se copió la ruta para usarla, después se abre la terminal de Windows en administrador y se usó los comandos ***cd*** y ***dir*** para direccionarme a la carpeta donde se va instalar el entorno virtual, luego de que se encuentre la ruta especifica del proyecto, se ejecuta el comando **pip install Django** como se muestra en la siguiente imagen.

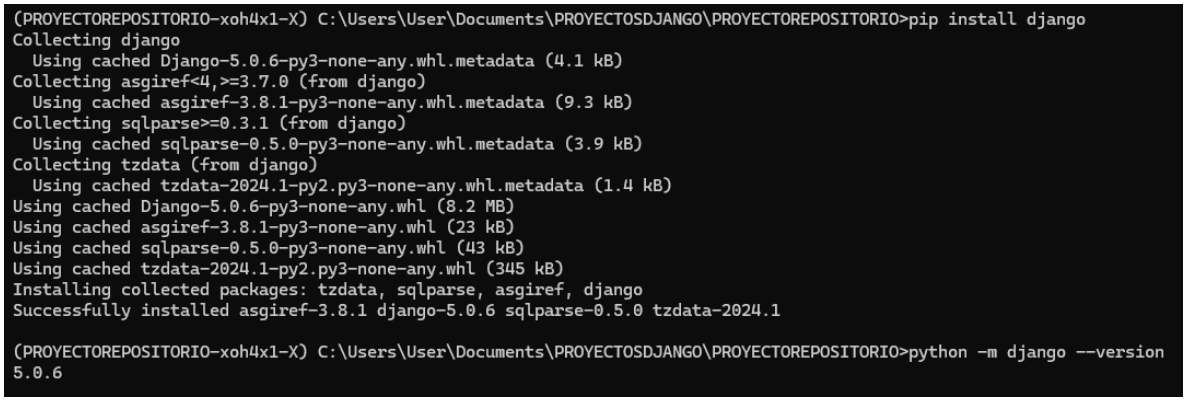


Figura 8. Creación del proyecto Django y comprobación de su versión. Fuente: Elaboración propia

En la figura 8 se observa la ejecución del comando pip install django **dentro del entorno virtual activado**, seguido de la verificación de versión con python -m django --version, confirmando la instalación correcta de Django 5.0.6 junto a sus dependencias: asgiref, sqlparse y tzdata.

También se ilustra el proceso de instalación de Django dentro de un entorno virtual previamente activado, debido a la instalación de pipenv como se evidencia en la figura 7, lo cual permite aislar las dependencias del proyecto. Luego, se comprueba la correcta instalación mediante el comando ***python -m django --version***, que confirma la versión 5.0.6 del framework.

6- Después de haber rectificado que todo va correctamente por consola como se observa en la figura 8,

me dirigí a la carpeta del proyecto y activé el ambiente virtual usando el comando ***pipenv shell***

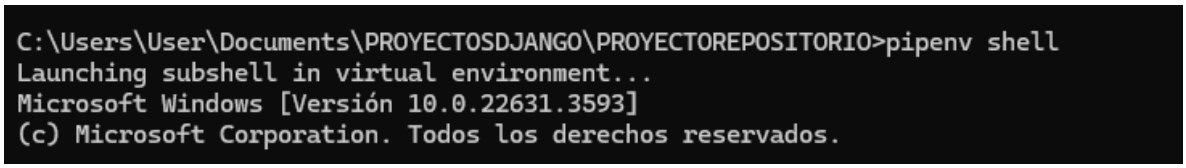


Figura 9. Activación del ambiente virtual. Fuente: Elaboración propia

En la Figura 9 se muestra la activación del entorno virtual mediante el comando pipenv shell. Luego empieza cargar el ambiente virtual con el mensaje "Launching subshell in virtual environment…", y se activa con mensaje final de "…Todos los derechos reservados.".

También, se evidencia el uso del comando ***pipenv shell*** dentro del directorio del proyecto. Al ejecutarse, inicia una subterminal (subshell) que activa el entorno virtual creado previamente. Esto permite instalar paquetes, trabajar dentro de un espacio controlado y en mi caso ejecutar el servidor Django, lo cual es una buena práctica en el desarrollo de aplicaciones web.

7- Se inicia creando la estructura del proyecto usando el comando ***django-admin starproject config*** este comando crea unos archivos y directorios necesarios para la preparación de un nuevo sitio web. Se aseguró que los archivos se han creado correctamente, revisando la ruta donde se creó la carpeta del proyecto para asegurarse de que estaban los archivos correctamente. La siguiente figura muestra la estructura inicial del proyecto Django después de haber sido creado correctamente, destacando la carpeta de configuración (config) y el archivo principal (manage.py). Donde config tiene cinco archivos esenciales para la configuración del proyecto.

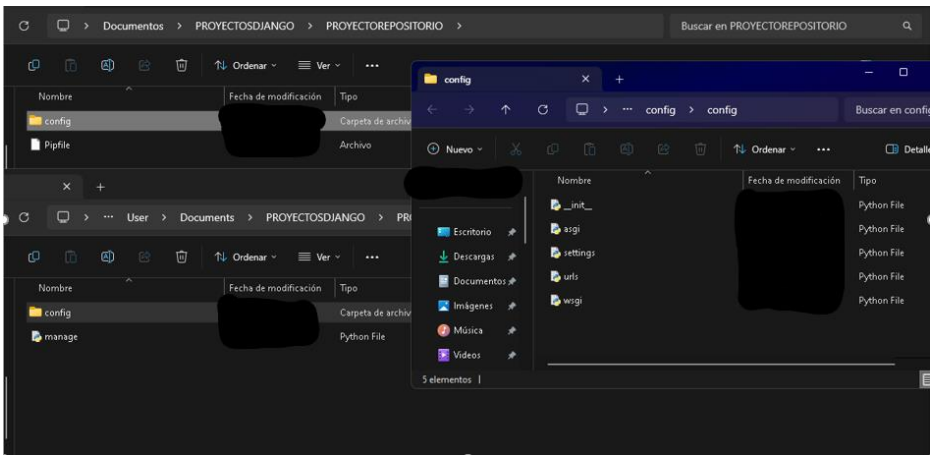


Figura 10. Archivos creados en la carpeta del proyecto. Fuente: Elaboración propia

La figura 10 muestra las carpetas que han sido creadas, La primera carpeta llamada "PROYECTOREPOSITORIO", en donde se ubican los archivos config, que fue generado por Django al ejecutar el comando django-admin startproject config, y Pipfile que fue generado por el entorno virtual.

La segunda carpeta son los archivos de config en los que se encuentra otra carpeta config y el archivo manage.py.

El contenido de la carpeta config, como se muestra en la figura 10, contiene archivos esenciales para la configuración del proyecto, como:

- __init__.py: indica que el directorio es un paquete de Python.
- settings.py: archivo donde se define la configuración del proyecto-
- urls.py: contiene las rutas o URLS del sitio web.
- wsgi.py y asgi.py: archivos para ejecutar la aplicación en servidores WSGI o ASGI.

Esta organización es parte de la estructura básica que Django genera automáticamente y sirve como punto de partida para el desarrollo de cualquier aplicación web estructurada.

8- Para evitar problemas al iniciar el servidor, se ajustó la ubicación de los archivos **config** y

manage.py, asegurando que estuvieran en el directorio adecuado para inicializar el servidor. Se usó el comando ***python manage.py runserver,*** después de habilitar el entorno virtual como se muestra en la figura 9, para correr el servidor, como se muestra a continuación:

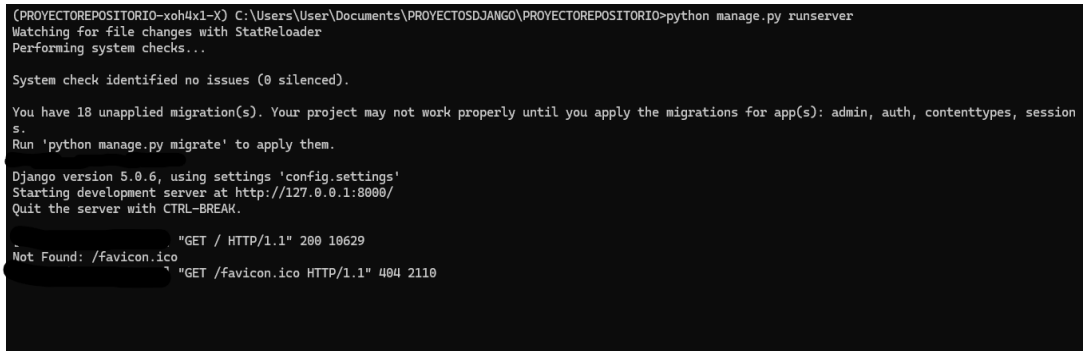


Figura 11. Activación del servidor. Fuente: Elaboración propia

La Figura 11 muestra la ejecución del comando python manage.py runserver dentro de la ruta del proyecto con el entorno virtual activado. En la terminal se indica que hay 18 migraciones pendientes por aplicar, lo cual se refleja en el mensaje “You have 18 unapplied migrations”. Además, se confirma que el servidor de desarrollo de Django se ha iniciado correctamente y está disponible en la dirección http://127.0.0.1:8000/. Finalmente, aparece la instrucción “Quit the server with CTRL-BREAK”, que indica que se puede detener la ejecución del servidor presionando CTRL+C.

Se activa el servidor, pero muestra que se tienen 18 migraciones pendientes, esto quiere decir que hay cambios en los modelos y que toca aplicarlos en la base de datos, sin embargo, no es un problema para mostrar que si funciona el servidor por lo que aplicaremos las migraciones en pasos siguientes

9- Ya terminado este paso, después de habilitar el servidor como se muestra en la figura 11, se abrió el navegador y coloca la ruta por defecto http://127.0.0.1:8000/ para ver que funciona correctamente el servidor de Django

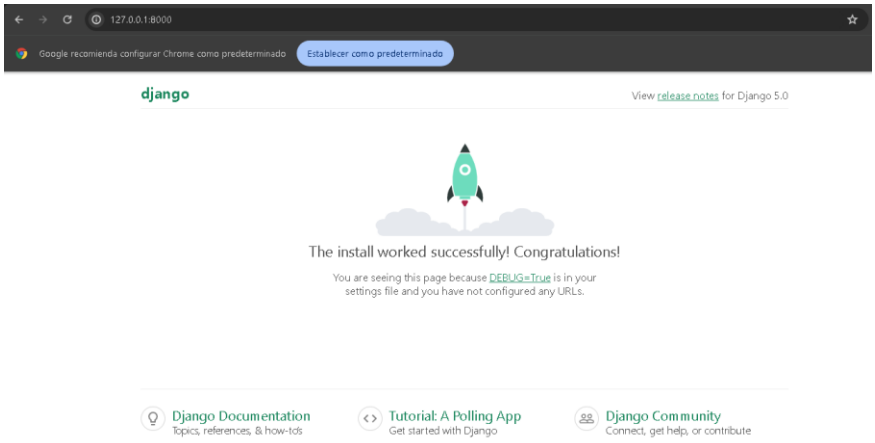


Figura 12. Interfaz de Django que muestra que todo está bien configurado. Fuente: Elaboración propia

La figura 12 muestra “The install worked successfully! Congratulations!”, acompañado de una imagen ilustrativa de un cohete despegando, que significa que el servidor se ha iniciado correctamente, lo cual indica que no hay errores en la configuración del proyecto hasta este punto.

10- Gracias a esto, podemos acceder a la aplicación web a través del navegador utilizando la dirección proporcionada. Ahora es posible continuar con el desarrollo y la configuración de nuevas funcionalidades dentro del entorno de desarrollo local de Django.

Cuando ejecutamos el servidor a veces se presenta el error antes mencionado “You have 18 unapplied migration(s)…”, entonces para solucionar este error salimos primero del servidor en consola usando ***CTRL+C*** y después ejecutamos un comando para aplicar las migraciones que es ***python manage.py migrate*** con esto no volvió a mostrar el error después de ejecutar el servidor ***pyhton manage.py runserver***, sino que se muestra lo de la figura 12 donde evidencia que el servidor sigue funcionando correctamente.

8.2.1.4 PROYECTO Y APLICACIÓN EN DJANGO

En Django, se utilizan los conceptos de "proyecto" y "aplicación" para organizar el código y las funcionalidades del sitio web. Entender esta estructura es esencial para trabajar de manera eficiente con

Django.

Proyecto: Un proyecto en Django es un conjunto de configuraciones y aplicaciones para un sitio web. Es el contenedor principal que abarca todas las aplicaciones necesarias para que el sitio web funcione. Al iniciar un nuevo proyecto de Django, se crea una estructura básica de directorios y archivos que incluyen configuraciones globales para el proyecto.

Aplicación: Una aplicación en Django es un componente de software que realiza una tarea específica. Cada aplicación es responsable de una funcionalidad específica, como la gestión de usuarios, la administración de pagos, o la visualización de productos. Esto permite dividir el proyecto en partes más manejables y organizadas. https://docs.djangoproject.com/en/5.2/intro/tutorial01/

11- Ahora se colocó un nombre al proyecto, como se muestra a continuación:

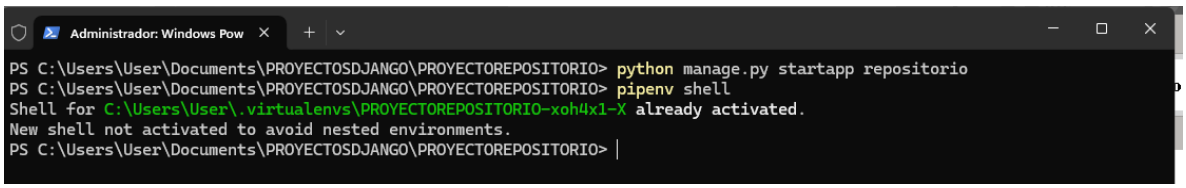


Figura 13. Nombramiento del proyecto y verificación del ambiente virtual. Fuente: Elaboración propia

La figura 13 muestra dos comandos utilizados en consola dentro de la ruta del proyecto, uno es ***python manage.py starapp repositorio*** donde "repositorio" es el nombre del proyecto y ***pipenv shell*** que activa el entorno virtual previamente creado.

Otro inconveniente común al iniciar el servidor, es que el ambiente virtual parecía desactivarse. Sin embargo, al verificar con ***pipenv shell***, se confirmó que seguía activo. Este es un error habitual en Windows, pero no afecta la funcionalidad del entorno.

12- Se comprobó de que se haya cambiado el nombre de nuestra aplicación con el comando ***tree***

```
PS C:\Users\User\Documents\PROYECTOSDJANGO\PROYECTOREPOSITORIO> tree
Listado de rutas de carpetas
El número de serie del volumen es 087B-518A
C:.
├───config
└───repositorio
    └───migrations
PS C:\Users\User\Documents\PROYECTOSDJANGO\PROYECTOREPOSITORIO>
```

*Figura 14. Contenido de la carpeta del proyecto con el comando **tree.** Fuente: Elaboración propia*

Cuando se usa el comando ***tree*** como se muestra en la figura 14, se despliega el contenido de la carpeta donde se encuentra nuestro proyecto y muestra una nueva carpeta llamada "repositorio" que es el nombre que se colocó al proyecto

13- Dentro de la carpeta **config** de la aplicación se crearon varios archivos, el primero a configurar es el archivo **settings.py** donde se agregó un parámetro para el proyecto de Django, debido a que el programa aún no sabe que la aplicación existe. Se puede editar usando bloc de notas, pero para mayor comodidad, se usó un programa llamado Spyder de Anaconda que sirve para ejecutar código del lenguaje Python y ayuda a identificar errores de código a medida que se va editando.

Dentro de **settings.py** hay una variable llamada **INSTALLED_APPS** dentro de esta variable añadiremos el nombre de nuestra aplicación.

```
29
30
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'repositorio'
41  ]
```

Figura 15. Inclusión del nombre del proyecto en el archivo settings.py. Fuente: Elaboración propia

Como se muestra en la figura 15 dentro del archivo settings.py se agregó el nombre del proyecto en la línea 40, dentro de la lista “INSTALLED_APPS”. Esta inclusión permite que Django reconozca y registre el nombre del proyecto llamado ‘repositorio’.

14- Dentro de los parámetros a configurar de Django tenemos lo que son las URLs, views, models y plantillas que son las encargadas de darle forma al proyecto y se pueden editar dentro del aplicativo de SPYDER.

Según Django Software Foundation (s.f.), para estructurar las rutas dentro de una aplicación en Django, se crea un módulo en Python conocido como “urlconf” (configuración de **URLs**). Este módulo actúa como un puente entre las rutas definidas en la URL y las funciones que se encargan de procesar esas solicitudes, conocidas como vistas (views). En esencia, es un archivo donde se define cómo deben responderse las diferentes direcciones web que hacen parte del proyecto.

La configuración de URLs puede ser tan simple o compleja como el proyecto lo requiera, permitiendo incluso añadir otras configuraciones de rutas para mantener una estructura organizada. Al estar basado en código Python, este sistema ofrece flexibilidad para construir rutas de forma dinámica. Django facilita la traducción de URLs según el idioma activo del sitio, lo cual es especialmente útil en proyectos multilingües.

Las vistas (**views.py**), en el contexto del desarrollo web con Django, remiten la función de vista que corresponde a un componente fundamental que se encarga de recibir una solicitud web y generar una respuesta adecuada. Esta respuesta puede adoptar múltiples formas, como el contenido HTML de una página, una redirección, un mensaje de error (como un 404), un archivo XML, una imagen u otro tipo de dato. Lo importante es que la vista contiene la lógica necesaria para determinar qué respuesta devolver según la solicitud recibida. Por convención y para mantener una estructura organizada, es habitual que estas funciones se almacenen en un archivo denominado **views.py**, localizado dentro del directorio del proyecto

o de una aplicación específica, que en nuestro caso de la carpeta ‘repositorio’.

Los modelos (**models.py**) en Django, representan la estructura central para definir y organizar los datos de una aplicación, puede considerarse como la fuente principal y confiable de información, ya que describe los campos y comportamientos importantes de los datos que se desean almacenar. Cada modelo se corresponde con una tabla específica en la base de datos.

Desde una perspectiva técnica, un modelo en Django es una clase escrita en Python que hereda de **django.db.models.Model**. Cada atributo definido en esta clase equivale a un campo en la tabla de la base de datos, permitiendo mapear directamente la estructura de los datos.

Django utiliza un sistema de **plantillas** para generar HTML dinámicamente de manera sencilla. Una plantilla combina partes estáticas de HTML con una sintaxis especial que permite insertar contenido dinámico. El enfoque más común es el uso del lenguaje de plantillas de Django (DTL), aunque también se puede usar Jinja2 u otros motores de plantillas externos. Django permite configurar uno o varios motores de plantillas, e incluso ninguno si no se requieren.

La API de Django para plantillas define el proceso de carga y renderización, permitiendo encontrar y compilar plantillas, luego representarla con datos de contexto para generar la salida final. Aunque DTL tiene ciertas limitaciones y particularidades, es una opción recomendada por su integración nativa, especialmente para aplicaciones reutilizables como las de django.contrib.admin. El soporte tanto para los motores de plantillas como para DTL, se encuentra en el espacio de nombres django.template. (Django Software Foundation, s.f.)

Ya entendido esta parte, se fueron realizando ajustes según las necesidades específicas del proyecto

15- Para comenzar el proyecto se inicia la terminal de Windows en modo administrador y luego se usan siempre 3 comandos de manera ordenada

- ***cd (ruta del proyecto)*** (ubicar la ruta del proyecto)

- ***pipenv Shell*** (activar el ambiente virtual)

- ***Python manage.py runserver*** (inicia el servidor)

Si se hace un cambio en models.py antes de darle ***python manage.py runserver*** le damos:

- ***Python manage.py makemigrations*** (crea archivos de migración)

- ***Python manage.py migrate*** (aplica las migraciones pendientes a la base de datos)

- ***Python runserver*** (inicia el servidor)

Ya con estos comandos se puede observar los cambios realizados desde la ruta http://127.0.0.1:8000/

8.2.1.5 PUBLICAR EL REPOSITORIO EN GITHUB

Para llevar un registro ordenado de los cambios realizados en el desarrollo del proyecto, se utilizó Git como sistema de control de versiones y se subió el código a GitHub como respaldo. A continuación, se describen los pasos realizados:

i. Se instaló Git y se configuraron el nombre de usuario y el correo electrónico:

   git config --global user.name "TRABAJOS_DE_GRADO_CYT"

   git config --global user.email "dasuarezb@hotmail.com"

ii. Se activó el ambiente virtual con ***pipenv shell*** dentro de la ruta del proyecto.

iii. Se inicializó un repositorio de Git en la carpeta del proyecto con ***git init***.

iv. Se añadieron todos los archivos al repositorio con ***git add -A***.

v. Se realizó el primer commit con ***git commit -m "commit inicial"***.

vi. Con lo anterior, se crea un repositorio en la página de GitHub con el correo se puso por comando

https://github.com/DiegoSuarez01/repositorio.git

Ahora se enlaza el proyecto local con el repositorio remoto de GitHub a través de los siguientes pasos:

vii. Se creó un repositorio en GitHub y se vinculó con el repositorio local mediante ***git remote add origin https://github.com/DiegoSuarez01/repositorio.git***

viii. Comprobación de que el repositorio se encuentra enlazado con ***git remote -v***

ix. Se subieron los cambios al repositorio remoto en GitHub con ***git push -u origin master***

x. Esto abrió una pestaña emergente para que pudiera ingresar la cuenta de administrador y acceder al repositorio en GitHub. Luego, se ingresó con el usuario y contraseña. Es de anotar que el proceso salió correctamente

8.2.1.6 CREANDO SUPERUSUARIO

Al crear un superusuario para acceder al panel de administración y gestionar modelos y datos sin necesidad de escribir consultas SQL o usar la consola de Django

```
PS C:\Users\User\Documents\PROYECTOSDJANGO\PROYECTOREPOSITORIO> python manage.py createsuperuser
Username (leave blank to use 'user'): repositorio
Email address: dasuarezb@hotmail.com
Password:
Password (again):
Superuser created successfully.
PS C:\Users\User\Documents\PROYECTOSDJANGO\PROYECTOREPOSITORIO>
```

Figura 16. Creación del superusuario colocando correo y contraseña. Fuente: Elaboración propia

La Figura 16 muestra la creación exitosa de un superusuario en Django con el comando ***python manage.py createsuperuser***, ejecutado desde la consola en la ruta del proyecto. Durante el proceso, se solicitó un nombre de usuario (repositorio), una dirección de correo electrónico y una contraseña (ingresada dos veces para su confirmación). Al finalizar, se mostró el mensaje “Superuser created successfully”, lo que indica que el usuario administrador ha sido creado correctamente y podrá acceder a la interfaz de administración del sistema.

Por seguridad la consola no muestra el **password,** como se observa en la figura 16, sale en negro. En este caso lo que se escribe no se muestra, de igual manera el super usuario ya fue creado y se puede acceder al panel de administrador desde el enlace http://127.0.0.1:8000/admin

Creación de una primera interfaz

Para dar inicio a la construcción de la interfaz gráfica del repositorio digital, fue necesario implementar una vista básica en Django que permitiera visualizar una lista de documentos y tres enlaces donde el primero fuera para cargar los documentos, otro donde se devuelve a la lista de documentos, y la última donde lo direcciona al archivo. A continuación, se describe el procedimiento realizado para lograr esta primera interfaz funcional.

Primero, se definió el modelo mencionado en el diagrama de clases de la fase 2 llamado **Documento** dentro del archivo **models.py** de la aplicación correspondiente. Este modelo representa cada archivo de trabajo de grado y contiene los campos de información que se mostraran. Sin embargo, no todos se van añadir a la plantilla debido a que es un primer borrador de prueba.

```python
class Documento(models.Model):
    CATEGORIAS = [
        ('electronica', 'Electrónica'),
        ('diseno', 'Diseño Tecnológico'),
        ('tecnologia', 'Tecnología'),
    ]
    #INFORMACIÓN GENERAL
    titulo = models.CharField(max_length=255, blank=True, null=True)
    autor = models.CharField(max_length=255, blank=True, null=True)
    director = models.CharField(max_length=255, blank=True, null=True)
    palabras_clave = models.CharField(max_length=255, blank=True, null=True)
    unidad_patrocinante = models.CharField(max_length=255, blank=True, null=True)
    fecha_publicacion = models.CharField(max_length=255, blank=True, null=True)
    #SECCIONES DEL DOCUMENTO
    descripcion = models.TextField(blank=True, null=True)
    metodologia = models.TextField(blank=True, null=True)
    contenidos = models.TextField(blank=True, null=True)
    conclusiones = models.TextField(blank=True, null=True)
    lineas_investigacion = models.TextField(blank=True, null=True)
    fuentes = models.TextField(blank=True, null=True)
    enlace = models.URLField(blank=True, null=True)
    archivo = models.FileField(upload_to='documentos/', max_length=255, blank=True, null=True)

    categoria = models.CharField(max_length=50, choices=CATEGORIAS, default='electronica')

    def __str__(self):
        return self.titulo if self.titulo else "Documento sin título"
```

*Figura 17. Clase **Documento** en models.py. Fuente: Elaboración propia*

En la figura 17 se muestra la clase oficial definida en el archivo models.py, esta clase fue previamente definida como se muestra en la figura 3, la cual se introducen las "**categorías**", que representan las ubicaciones donde se almacenarán los archivos y las "**líneas_investigación**" para clasificar los documentos. Se definieron diversos atributos de tipo texto con el fin de permitir ingresar información adicional al momento de cargar un documento. Esta información complementaria será posteriormente utilizada para ser mostrada en la tabla de detalles de cada documento ya que debía mostrar algo para no dejar el espacio en blanco.

8.2.1.7 VISTA DETALLADA DE LOS DOCUMENTOS

Además de listar todos los documentos cargados, se implementó una vista de detalle que permite al usuario consultar información más específica sobre cada trabajo de grado almacenado en el sistema. Esta vista muestra atributos como el título del documento, el nombre del autor, la metodología empleada y un enlace de descarga directa del archivo.

Para lograrlo, se crearon unas nuevas vistas en el archivo views.py, las cuales reciben una plantilla distinta para mostrar la interfaz:

```python
from django.shortcuts import render
from django.views.generic import CreateView, ListView, DetailView
from .models import Documento
from django.urls import reverse_lazy

# Vista para crear documentos
class DocumentoCreateView(CreateView):
    model = Documento
    fields = ['titulo', 'autor', 'metodologia', 'archivo']
    template_name = 'documento_form.html'
    success_url = reverse_lazy('documento_list')

# Vista para listar documentos
class DocumentoListView(ListView):
    model = Documento
    template_name = 'documento_list.html'
    context_object_name = 'documentos'

# Vista para ver el detalle de un documento
class DocumentoDetailView(DetailView):
    model = Documento
    template_name = 'documento_detalle.html'
```

Figura 18. Vistas primarias del archivo views.py. Fuente: Elaboración propia

Como ya se mencionó en la sesión 8.1.5 en el paso 14, views guarda la lógica del sistema de lo que debe mostrarse al usuario cuando se accede a una URL, en la figura 18 muestra que se usaron vistas basada en clases, donde:

DocumentoCreateView:

Uso ***model=Documento*** *para* crear un formulario donde va asignar los valores mostrados en ***fields***, no se usaron todos los atributos porque se hace una primera prueba del funcionamiento. Dentro del ***template_name*** se aloja la plantilla ***documento_form***, donde se va a escribir los datos que vamos a mostrar, ya cuando carguemos los datos se usa la variable ***success_url*** para redireccionar a la plantilla ***documento_list***.

DocumentoListView:

Muestra una lista de todos los documentos guardados en la base de datos. Donde ***context_object_name*** define el nombre de la variable que se usó en el HTML para recorrer los objetos.

DocumentoDetailView:

Muestra los detalles del documento seleccionado redirigiéndolo a la plantilla ***documento_detalle***

8.2.1.8 CONFIGURACIÓN DE LA URL

Estas vistas fueron conectadas en **urls.py** mediante funciones **path**, que enlazan rutas específicas del navegador con las vistas correspondientes.

```python
from django.contrib import admin
from django.urls import path
from django.conf import settings
from django.conf.urls.static import static
from repositorio.views import DocumentoCreateView, DocumentoListView, DocumentoDetailView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('crear/', DocumentoCreateView.as_view(), name='documento_crear'),
    path('documentos/', DocumentoListView.as_view(), name='documento_list'),
    path('documento/<int:pk>/', DocumentoDetailView.as_view(), name='documento_detalle'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Figura 19. Rutas enlazadas en archivo urls.py. Fuente: Elaboración propia

Se importa las vistas usadas anteriormente para poder acceder a ellas desde una URL, como se muestra en la figura 19, se guarda los enlaces dentro de una lista llamada “urlpatterns”.

Al acceder a ***/documentos/***, se ejecuta la vista que presenta el listado completo, mientras que en ***/crear/*** se accede al formulario de carga. Cada documento puede explorarse en detalle con **/documento/<int:pk>/**, donde **<int:pk>** representa su identificador en la base de datos, y por último ***admin/*** es para acceder al panel de administrador que ofrece Django.

8.2.1.9 Plantilla de detalles del repositorio

Para presentar los detalles completos de cada documento, se usó la vista **DocumentoDetailView**, que permite obtener automáticamente la información de un documento específico utilizando su clave primaria (**pk**). Esta vista está enlazada en urls.py mediante una ruta dinámica, y renderiza una plantilla HTML (**documento_detalle.html**) que organiza los datos en una tabla. Esta también incluye un enlace de descarga directo al archivo PDF cargado, sin embargo, más adelante se explica porque se elimina esta opción.

```
<head>
    <meta charset="UTF-8">
    <title>Detalle del Documento</title>
</head>
<body>

<ul>
    <li><a href="{% url 'documento_crear' %}">Cargar Documento</a></li>
    <li><a href="{% url 'documento_list' %}">Lista de Documentos</a></li>
</ul>

<h1>Detalle del Documento</h1>

<table border="1" cellpadding="10">
    <tr>
        <th>Título</th>
        <th>Autor</th>
        <th>Metodología</th>
        <th>Archivo</th>
    </tr>
    <tr>
        <td>{{ object.titulo }}</td>
        <td>{{ object.autor }}</td>
        <td>{{ object.metodologia }}</td>
        <td>
            {{ object.archivo }}
            {% if object.archivo %}
                <a href="{{ object.archivo.url }}" download>Descargar</a>
            {% else %}
                No disponible
            {% endif %}
        </td>
    </tr>
</table>

</body>
</html>
```

Figura 20. Archivo HTML que muestra los detalles del documento. Fuente: Elaboración propia

En la figura 20 muestra una plantilla, en la cual tenemos dos enlaces que direccionan a la URL de cada plantilla, después va el título de la plantilla y la respectiva tabla con los datos del archivo y por último el botón de descargar

Así es como se establece la conexión entre los distintos archivos necesarios en un proyecto desarrollado con Django. Es importante tener en cuenta que todas las plantillas HTML deben almacenarse dentro de una carpeta denominada **templates**, ya que Django la reconoce como la ubicación predeterminada para los archivos de interfaz. Hay que mantener la coherencia entre las configuraciones Cualquier cambio realizado en una parte del sistema (como una vista, plantilla o URL) debe reflejarse correctamente en las demás para que funcione adecuadamente la aplicación.

8.3 FASE 3: IMPLEMENTACIÓN DEL SISTEMA

El desarrollo del repositorio digital de trabajos de grado, consistió en la exploración e implementación de herramientas que se usan para la extracción automática de información desde documentos en formato PDF. Este proceso resultó esencial para realizar pruebas del funcionamiento de cada una de las herramientas

8.3.1 EXTRACCIÓN DE INFORMACIÓN DESDE PDFs

Durante esta etapa, se evaluaron diversas bibliotecas de Python con el fin de identificar la más adecuada en términos de eficiencia, precisión y compatibilidad con los tipos de documentos disponibles. Se presentaron algunas dificultades durante las pruebas, pero finalmente se logró encontrar una solución. Se mostrará el proceso que llevé a cabo con cada una de las herramientas analizadas.

- Primer intento - PyMuPDF: Biblioteca eficiente para leer y extraer contenido de archivos PDF. Se logró extraer los títulos, pero los resultados no fueron óptimos.

- Segundo intento – spaCy y PDFPlumber: En este caso se incorporó el procesamiento de lenguaje natural a través de la biblioteca spaCy y PDFPlumber, con el fin de identificar patrones más precisos dentro del texto. Sin embargo, la falta de estructura lógica en algunos documentos, impidió lograr una extracción limpia y segmentada, afectando especialmente la detección de secciones como metodología o conclusiones.

- Tercer intento - NLTK: Posteriormente se empleó NLTK (Natural Language Toolkit) para procesar el texto extraído con técnicas más avanzadas de segmentación, tokenización y detección de frases clave. Aunque mejoró el procesamiento de texto, no se tuvo una precisión que se necesitaba.

- Cuarto intento – PyMuPDF con expresiones regulares: Se logró una extracción precisa de la información haciendo uso de funciones integradas de PyMuPDF y expresiones regulares

8.3.1.1 PRIMER INTENTO (PYMUPDF)

Es importante automatizar la extracción de información porque permite reducir el tiempo requerido para categorizar los trabajos de grado según sus líneas de investigación. También contribuye a una mejor organización del repositorio digital, facilitando la gestión de los documentos y optimizando los procesos de búsqueda mediante filtros personalizados. Esta automatización mejora la eficiencia del sistema, tanto para quienes lo administran como para los usuarios que consultan los trabajos disponibles.

Empezamos analizando PyMuPDF. Es una biblioteca de Python para leer, analizar, modificar y extraer datos de archivos PDF, XPS y ePub (pag 32). Es rápida, ligera y muy eficiente para manipular documentos digitales; perfecta para implementarla en el proyecto del repositorio y extraer información esencial.

Se instaló la biblioteca a través de la consola de comandos de Windows usando el comando ***pip install pymupdf***, también fue necesario instalarlo desde la consola de Spyder, ya que se requería hacer pruebas del funcionamiento antes de implementarla en el proyecto. La intención de este procedimiento es verificar los

resultados directamente desde Spyder para luego integrarlos en la aplicación y realizar pruebas desde el servidor tras subir el trabajo de grado.

***import fitz* :** Biblioteca de PyMuPDF

***import re* :** Biblioteca de expresiones regulares (regex)

***import tkinter as tk* :** Biblioteca de interfaz gráfica

***from tkinter import filedialog* :** Importa desde tkinter "filedialog" que se usó para abrir archivos.

Donde **re** (regex) fue utilizada para implementar expresiones regulares, una herramienta que permite realizar búsquedas por patrones dentro del texto. Esta funcionalidad resultó necesaria para localizar elementos específicos como nombres o secciones del documento que siguen una estructura fija. En el Anexo 1 se presenta el código completo, el cual ofrece más detalles sobre la lógica empleada y su funcionamiento. Es importante aclarar que **no se detallará el código de cada una de las pruebas individuales**, con el fin de evitar redundancias innecesarias en la explicación. **Únicamente se explicará el código correspondiente a la última prueba**, ya que esta integra y combina algunos elementos de las pruebas realizadas.

Se realizó un método de prueba que consistió en aplicar los extractores de información sobre tres documentos iguales, denominados Documento 1, Documento 2 y Documento 3, con el objetivo de evaluar la precisión de cada extractor. Los documentos son:

Documento 1: Valbuena, J. O. (2017). *Análisis de la emergencia de la educación en Tecnología en Colombia (1994-2015).* Recuperado de: http://hdl.handle.net/20.500.12209/9555.

Documento 2: Torres, C. & Salazar, J. F. (2017). *Análisis histórico, técnico y educativo de las Escuelas Radiofónicas de Acción Cultural Popular (ACPO) de 1947 a 1967*. Recuperado de: http://hdl.handle.net/20.500.12209/9554.

Documento 3: Valencia, J. C. (2017). *Comparación cualitativa de programas de generación de modelos 3D para documentar y digitalizar un yacimiento arqueológico con arte rupestre.* Recuperado de: http://hdl.handle.net/20.500.12209/9556.

En el primer intento, se utilizó la biblioteca PyMuPDF (también conocida como fitz) para extraer el contenido textual de los documentos.

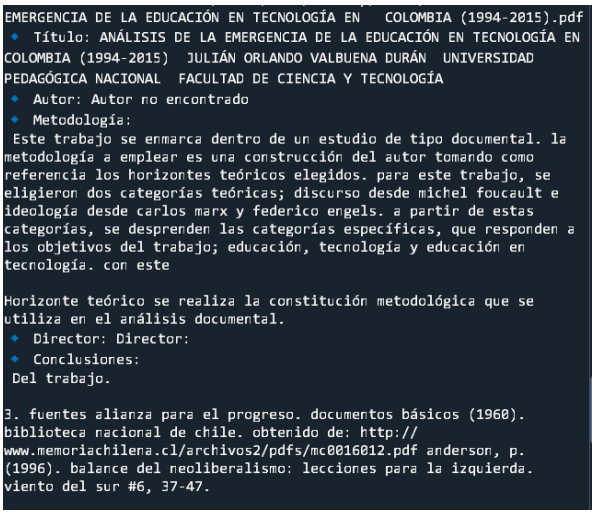


Figura 21. Prueba en el Documento 1 con PyMuPDF. Fuente. Elaboración propia

En la Figura 21, se evidencia los resultados del Documento 1, en el cual el código obtiene buenos resultados al identificar el título del trabajo, basándose en la lógica de que en muchos documentos académicos, las primeras líneas suelen contener el título. Sin embargo, los saltos de línea presentes en los archivos PDF no siempre reflejan una estructura coherente, lo que dificulta delimitar con precisión el bloque de texto correspondiente al título completo.

El campo autor no logra identificar correctamente los nombres, a pesar de implementar búsquedas mediante patrones comunes como “autor:”, “autores:” o “presentado por:”, los cuales son usualmente empleados en los trabajos de grado. No obstante, estos patrones no son reconocidos de manera efectiva.

En cuanto a la metodología, aunque se logra extraer un fragmento relacionado, este aparece de forma

desorganizada, ya que se toma únicamente el primer resultado que coincide con la palabra clave sin un análisis estructural del contenido.

Respecto al director, la función de búsqueda retorna una línea que contiene coincidencias léxicas (como "director"), pero no necesariamente un nombre completo o claramente identificado.

Finalmente, para las conclusiones, el código extrae el contenido directamente a partir de la aparición de la palabra "conclusiones", pero lo hace interpretando esta palabra como una frase dentro del texto, y no como un título de sección, lo que genera problemas en la precisión de la información extraída.

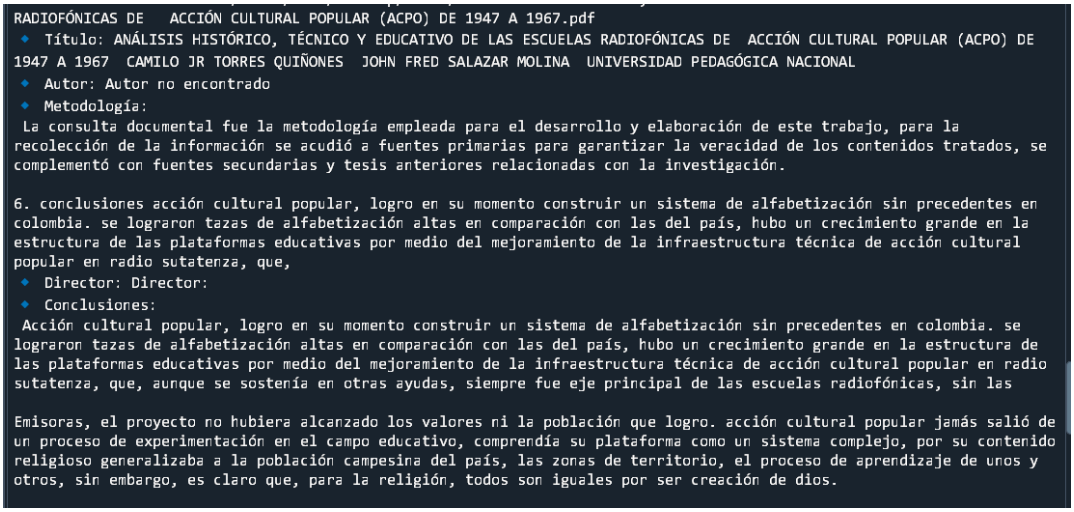


Figura 22. Prueba en el Documento 2 con PyMuPDF. Fuente. Elaboración propia

En el documento 2 que se muestra en la figura 22, ocurre lo mismo con título, autor y director. En metodología toma bien el párrafo, pero no delimita hasta dónde va la información y con conclusiones, no hubo problemas. Esto quiere decir que aún sigue con problemas en encontrar con precisión la información, no delimita bien el contenido y está tomando información errónea por lo que se prosiguió con el siguiente documento.

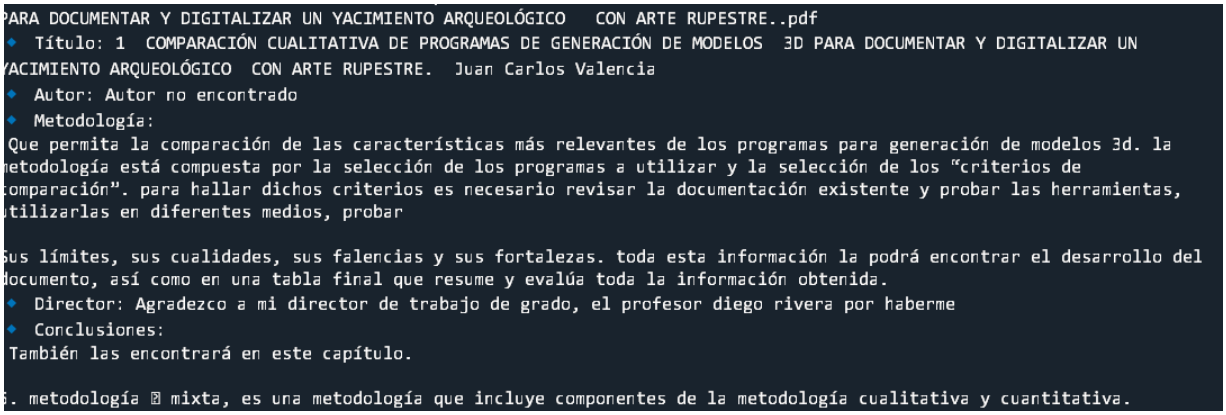


Figura 23. Prueba del documento 3 con PyMuPDF. Fuente. Elaboración propia

En la Figura 23, se presenta el documento 3, en el que tanto el título como el autor continúan sin ser correctamente identificados. Sin embargo, en cuanto al campo director, se obtiene un resultado, que no corresponde al valor esperado. La función detecta una coincidencia en la presencia de la palabra clave, pero extrae una frase que no pertenece a solo el nombre del director. Esto ocurre porque el documento contiene múltiples coincidencias similares y el sistema no es capaz de determinar cuál de ellas es la más relevante.

Con la metodología, el comportamiento es similar. Se extrae un párrafo que contiene la palabra clave, pero no se corresponde con la sección metodológica esperada. La función no logra determinar cuál coincidencia es la más importante de las que encontró

En el caso de las conclusiones, el resultado obtenido es solo una frase relacionada, pero no representa el bloque real de conclusiones del trabajo. En consecuencia, los resultados globales de extracción para este documento tampoco son satisfactorios.

Por lo tanto, se descartó esta opción y se buscó una mejor. En este sentido, se buscó asesoría con el profesor David Martinez el día 17 de octubre del 2024 donde se le comentó acerca de la dificultad. Durante la reunión fue sugerido usar bibliotecas que tuvieran Procesamiento de Lenguaje Natural (PLN) para que brindara la capacidad de interpretar el lenguaje humano de una manera más precisa. Con ello, se puede conseguir que

el análisis del texto sea más eficiente y coherente.

8.3.1.2 SEGUNDO INTENTO – SPACY Y PDFPLUMBER

Para implementar técnicas de Procesamiento de Lenguaje Natural (PLN) y mejorar la extracción de texto desde archivos PDF, fue necesario instalar dos bibliotecas fundamentales: spaCy, orientada al análisis lingüístico y pdfplumber, especializada en la lectura estructurada de archivos PDF.

Se inició instalando estas dos bibliotecas a través de la consola de Spyder usando los comandos ***pip install spacy*** y ***pip install pdfplumber***. Una vez instaladas, para que spaCy funcione correctamente con textos en español, se descargó el modelo lingüístico correspondiente utilizando el siguiente comando: ***python -m spacy download es_core_news_sm***. Este modelo permite a spaCy identificar entidades, estructuras gramaticales y patrones del lenguaje en **español**, lo cual fue fundamental para mejorar la precisión en la detección de títulos, autores, metodologías y otras secciones del texto.

Para la implementación importamos las librerías al código, con los siguientes comandos:

import pdfplumber

import spacy

Luego, usando la misma dinámica anterior, se utilizaron los mismos tres documentos se empleó la misma lógica de extracción que se había implementado previamente con PyMuPDF. Esto con el fin de comparar resultados y ver el comportamiento de las nuevas herramientas. En el anexo 2 se encuentra el código utilizado

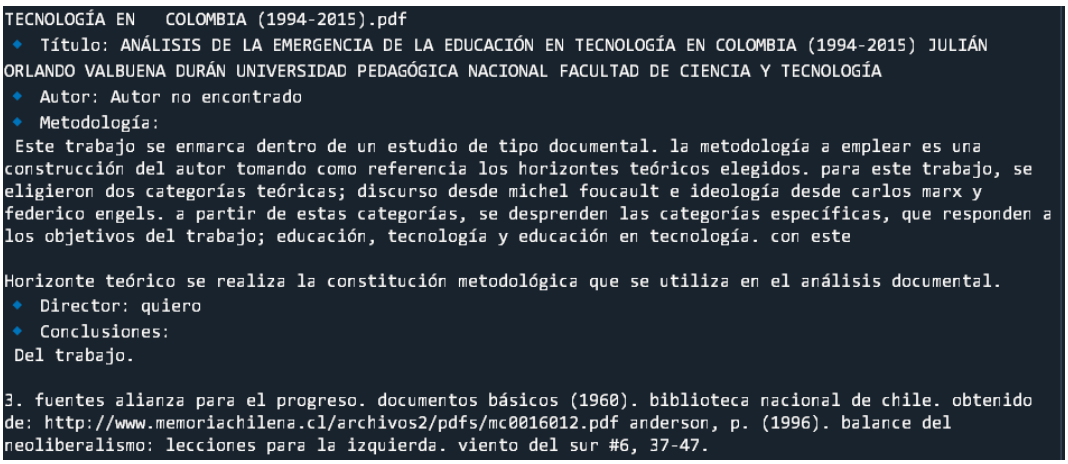


*Figura 24. Prueba del documento 1 con **spacy** y **pdfplumber**. Fuente. Elaboración propia*

En el documento 1, como se muestra en la Figura 24, los resultados se mantuvieron similares a los obtenidos en la prueba anterior, con excepción del campo correspondiente al director. En este caso, el sistema extrajo un fragmento contenido dentro de una frase, pero no logró identificar correctamente el nombre del director. Esto evidencia que la herramienta aún no identifica adecuadamente entre coincidencias textuales y la información específica que se desea extraer.

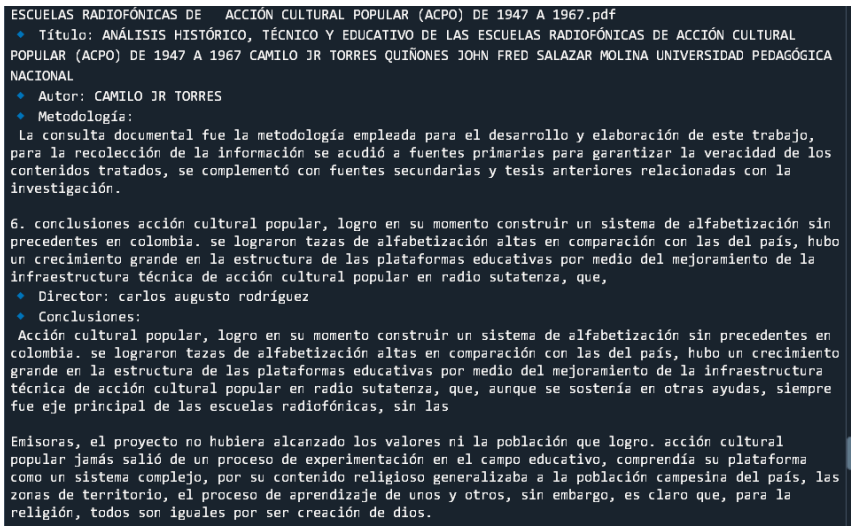


Figura 25. Prueba del documento 2 con spacy y pdfplumber. Fuente. Elaboración propia

En el segundo documento, como se evidencia en la Figura 25, observamos que tuvo una mejora

considerable, pero en metodología sigue sin saber cuánta información tomar. Como se muestra en la figura 22, tampoco hubo resultados en el documento 2 y cuando debería parar, en el autor, solo se tomó un nombre, pero con respecto a lo demás, se extrajo correctamente la información.

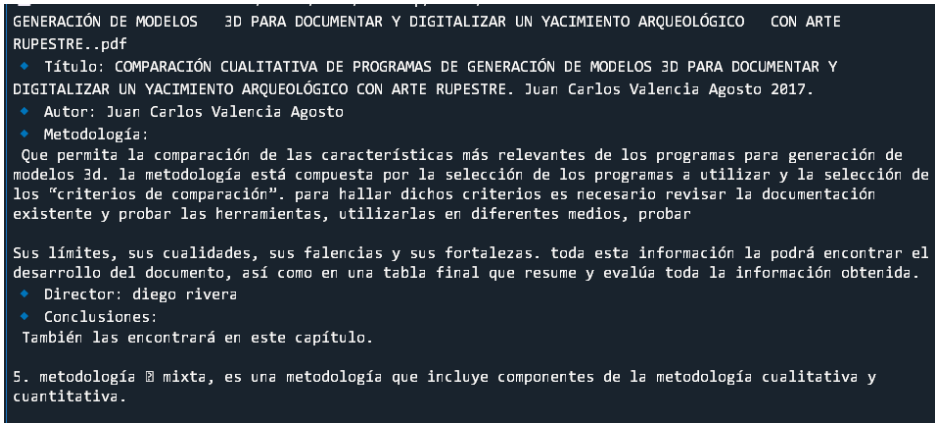


Figura 26. Prueba del documento 3 con spacy y pdfplumber. Fuente. Elaboración propia

Para el tercer documento, como se muestra en la Figura 26, se evidencian cambios en los resultados de los campos autor y director. En este caso, la extracción fue parcialmente exitosa ya que se logró identificar nombres en ambos campos, pero se observa que en el nombre del autor se incluyó el mes "Agosto", lo cual indica que la herramienta aún presenta dificultades para diferenciar adecuadamente entre nombres propios y otros factores textuales que comparten una estructura similar. Aunque el resultado representa una mejora con respecto a la figura 23, no se puede considerar completamente preciso.

Las herramientas muestran un desempeño superior en comparación con la utilizada anteriormente, pero aún no alcanzan el nivel de precisión necesario para extraer con exactitud la información solicitada. Sigue presentando errores en la delimitación de secciones específicas del documento, lo que evidencia la necesidad de seguir ajustando los métodos de procesamiento y segmentación del texto.

8.3.1.3 TERCER INTENTO – NLTK

Para este nuevo intento se optó por una alternativa basada en Procesamiento de Lenguaje Natural (PLN),

utilizando la biblioteca NLTK (Natural Language Toolkit). Esta herramienta cuenta con funcionalidades para la segmentación de texto, análisis gramatical y detección de palabras clave, lo que la convierte en una opción prometedora. Debido a sus características, se decidió incorporarla al sistema con el objetivo de mejorar la calidad y precisión en la extracción de información académica.

Se probaron los tres documentos que se usaron en las pruebas anteriores. No obstante, los resultados obtenidos no reflejan la precisión esperada. En el caso del documento 3, el autor fue identificado correctamente, sin embargo, en el documento 2 se extrajo un nombre que no corresponde. En general, en los documentos se detecta únicamente la primera coincidencia encontrada en el texto, lo que no garantiza que se esté extrayendo la información realmente deseada. Por esto, se concluye que aún persisten dificultades en cuanto a la precisión, razón por la cual se decidió explorar otras alternativas de extracción más eficientes. En el anexo 3 se adjunta el código usado para la extracción de la información con esta biblioteca.

Otra alternativa explorada fue el uso de bibliotecas orientadas al análisis de imágenes, **como PyTesseract y PDF2Image**, con el objetivo de extraer información en los casos en los que las herramientas anteriores no lograban detectar texto correctamente. Esta opción se consideró en caso de que los documentos hayan sido escaneados. Sin embargo, durante las pruebas realizadas no se obtuvo ningún resultado satisfactorio, ya que estas herramientas están optimizadas principalmente para textos presentes en libros, revistas, artículos y otras producciones literarias digitalizadas mediante escáner, lo cual no corresponde al tipo de documentos analizados en este proyecto.

8.3.1.4 CUARTO INTENTO- PYMUPDF CON EXPRESIONES REGULARES

En este último intento, se identificó que una de las principales dificultades de las herramientas previamente implementadas, como se muestra en la figura 21, 22 y 23, es la falta de precisión en la extracción de

información, ya que en muchos casos recuperaban bloques de texto irrelevantes o simplemente devolvían la primera coincidencia encontrada, sin un análisis profundo de los resultados encontrados. Debido a esta situación, se tomó la decisión de retomar el uso de expresiones regulares, aplicándolas sobre la primera herramienta utilizada (PyMuPDF), dado que fue la más explorada a lo largo del proyecto y resultó ser más comprensible y manejable para el desarrollo.

El uso de expresiones regulares y funciones propias de PyMuPDF permitió definir patrones específicos de búsqueda, lo que mejoró considerablemente la capacidad de extraer los metadatos de los trabajos de grado. Esta estrategia ofreció resultados más satisfactorios y coherentes en la mayoría de los documentos analizados. Se dejó el código completo usado en el anexo 4 para su consulta. A continuación, se presentará el proceso detallado de construcción del código con esta aproximación.

A continuación, se importarán las bibliotecas necesarias para realizar las pruebas:

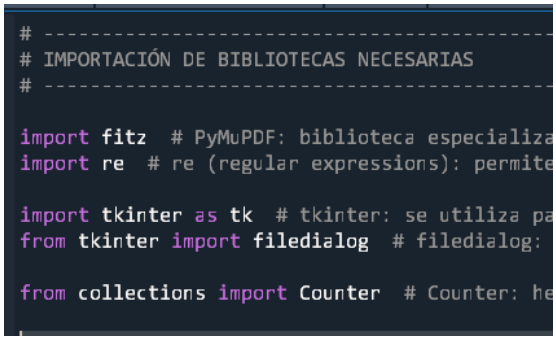


Figura 27. Bibliotecas importadas al código final. Fuente. Elaboración propia

En la figura 27 se muestra cinco bibliotecas: **fitz,** que es la biblioteca de **PyMuPDF** para la extracción de texto; **re** que son las expresiones regulares que vamos a utilizar para limitar la información; **tkinter**, se usó para crear una interfaz pequeña donde pueda seleccionar PDFs de manera ágil, reemplazando una función en la que tenía que poner la ruta y el nombre del archivo cada vez que ejecutaba el código; y **Counter,** es usada para contar la frecuencia de una palabra en un texto que será implementada después.

```python
def extraer_texto(pdf_path):
    # Abrir el archivo PDF
    doc = fitz.open(pdf_path)
    num_paginas = len(doc)

    # Extraer el texto de cada página
    texto = ""
    for pagina in doc:
        texto += pagina.get_text()

    # Devolver el texto y el número de páginas
    return texto, num_paginas, pdf_path
```

Figura 28. Función extraer_texto del código final. Fuente. Elaboración propia

La figura 28 muestra la primera función **extraer_texto,** la cual se encarga de abrir el archivo y extraer el texto de cada página. Esto con el propósito de extraer ciertas partes del documento en las primeras páginas y al final devuelve el **texto**, **num_paginas** que se usa para contar cuantas páginas de texto hay y la última **pdf_path** que devuelve el archivo.

A partir de este punto, se implementó un conjunto de funciones utilizadas para la extracción de información desde los documentos. El código se encuentra estructurado en dos extractores principales: uno diseñado para trabajar con documentos que no siguen el formato RAE y otro específico para aquellos que sí lo utilizan. Para determinar a cuál extractor debe dirigirse cada archivo, se desarrolló un procesador que analiza el contenido en busca de patrones característicos del formato RAE. Si se detecta una coincidencia, el documento es enviado al extractor correspondiente, de lo contrario, se redirige al extractor alternativo. Esta organización permite mantener el código organizado y evita que la extracción se vea afectada por las diferencias en el formato de los documentos. Luego, al final, se incluye una función que muestra los resultados por consola y se hace uso de **Tkinter**, una herramienta para la creación de interfaces gráficas, la cual facilita la selección de los documentos con los que se desea trabajar, para evitar la necesidad de ingresar manualmente la ruta de cada documento. Es de aclarar que la explicación de cada función se deja en el anexo 5. Con esto, se da cumplimiento al **segundo objetivo** específico “Implementar un sistema automatizado que permita extraer información clave de los trabajos de grado, como palabras clave,

objetivos, metodología, título, contacto, etc.”.

Finalmente, se logró una extracción precisa gracias a las expresiones regulares y a las funciones propias de la biblioteca PyMuPDF, las cuales permiten identificar patrones textuales directamente en el contenido de los archivos PDF. Aunque se evaluó inicialmente el uso de bibliotecas de procesamiento de lenguaje natural como spaCy o NLTK, estas mostraron limitaciones frente a documentos académicos con formatos irregulares y estructuras no estandarizadas, donde los segmentos resultaban inconsistentes. En cambio, las expresiones regulares permitieron definir reglas específicas ajustadas al dominio del problema (por ejemplo, detección de títulos, autores y metodología) sin depender de modelos entrenados previamente, lo que resultó más eficiente y preciso en este contexto. Estos patrones fueron refinados a partir de las pruebas, logrando un alto nivel de exactitud incluso en documentos sin una estructura definida.

8.3.2 Expresiones regulares, patrones del código final

Las expresiones regulares (regex) son patrones utilizados para buscar, extraer o reemplazar texto dentro de cadenas. Estas permiten identificar estructuras específicas como títulos, metodologías, conclusiones, etc. Resulta especialmente útil en el procesamiento de documentos que no tienen una estructura definida. En este proyecto, las expresiones regulares fueron clave para detectar secciones relevantes (como la metodología, introducción, o conclusiones) dentro de trabajos de grado en formato PDF, ya que la información extraída no sigue un formato uniforme, como se ha evidenciado en las pruebas (8.3.1), estos formatos presentan saltos de línea en lugares indebidos y fragmentos entre páginas que se incluyen dentro del contenido, generando así casos irregulares. Además, de la gran cantidad de documentos, se presenta problemas con llevar un mismo patrón para todos lo cual conlleva a incluir opciones dentro de estas expresiones, excepto para los documentos que tienen un formato RAE incluido.

Una aplicación concreta de estas expresiones puede observarse en la función

extraer_secciones_sin_formato_rae, la cual utiliza patrones regex para localizar etiquetas como “Metodología”, “Conclusiones” o “Resumen”, incluso si aparecen con distintas variantes ortográficas, mayúsculas o con una numeración. Esto emplea expresiones para dividir el texto en párrafos, limpiar caracteres innecesarios y filtrar contenido irrelevante. Esta estrategia permite extraer información precisa incluso en documentos sin un formato académico estandarizado como el RAE.

Una de las expresiones más sencillas que se implementó en el código, fue la sección de conclusiones, debido a que la mayoría de documentos tenían el mismo título y finalizaban igual. La expresión usada fue la siguiente:

rf"\s*(\d+\s*\.\s*\d*\s*)?(Conclusiones|Conclusi[oó]n|CONCLUSIONES|CONCLUSI[OÓ]N|Conclusiones\sy\srecomendaciones)\s*\n+([\s\S]*?){cierre}"

donde:

- “r” es como inicia la expresión.
- “f” es una string cruda que sirve para insertar variable.
- “\s*” aquí puede iniciar con un espacio o más el patrón o puede que no tenga espacios es por ello que se coloca este espacio.
- “*(\d+\s*\.\s*\d*\s*)?”* donde “\d” es cualquier número, puede haber uno o más, “\.” se usa para encontrar un punto en específico, y “\d*” busca un número, varios números o puede que ninguno, encuentra cualquier coincidencia con una numeración al inicio como hay dos dígitos entonces puede iniciar con un “1.1” o solo un digito “1.”. Esto se agrupó y al final se colocó un signo de interrogación (?) que significa que esa agrupación puede aparecer en la búsqueda, pero puede que no, sin embargo, debido a que en algunos documentos esta numeración aparece en la misma línea o está directamente unida al título de la sección, es conveniente incluir esta agrupación para hacer la expresión regular más

precisa.

- La siguiente agrupación incluye las posibles coincidencias del nombre de la sección que se desean detectar en el texto, como “Conclusiones”, “Conclusión” o variantes con mayúsculas. Para manejar correctamente palabras con y sin tilde, se emplean llaves [] en expresiones como [oó], lo que permite que el patrón coincida tanto con la letra sin tilde (o) como con la letra con tilde (ó). Esto no es necesario colocarlo ya que se puede incluir la tilde dentro de la palabra, pero haciendo pruebas no lo detectaba. Por ello, se decidió dejar las llaves.
- “\n+” busca un salto de línea o más.
- “([\s\S]*?)” es una expresión utilizada en **conjunto** para leer cualquier carácter seguido de cualquier espacio, letra número y saltos de línea.
- “*?” no son combinados el * significa que se puede repetir varias veces o puede que ninguna y el signo de interrogación es que posiblemente se encuentre esa situación.
- “{cierre}” es una lista que se incluye a la expresión para poder limitar la información extraída.

cierres = [r"(?=\n\s*\n)", r"(?=\.\s*\n)",]

Estos patrones de cierre indican dónde debe detenerse la extracción del contenido. Es decir, cuando encuentre dos saltos de línea (\n\n) o un punto seguido de salto de línea, se considera que ahí posiblemente termina la sección.

- “?=” es una expresión regular que indica que debe detener la extracción en la segunda expresión regular
- “\n\s*\n” hace un salto de línea, puede haber un espacio, más de uno, o ninguno y otro salto de línea.
- “\.” Significa que hay un punto y lo que le sigue es un punto un espacio, más de uno, o ninguno y un salto de línea, lo que significa un punto y aparte

Otra de las expresiones regulares usadas introduce nuevas expresiones, las cuales se describirán a continuación:

r"(?i)publicaci[oó]n\s*:?\s*(.+)"

- "?i" se usa para que busque la palabra en mayúsculas y minúsculas
- ".+" el punto no significa que haya explícitamente un punto sino un carácter es decir que significa que hay uno o más caracteres.

Para finalizar se encuentra otra expresión en la que se encuentra nuevas expresiones:

"(?i)Palabras(?:\s+Claves?)?\s*:?\s*\n*([\s\S]+?)(?=\n\s*(?:\[\d+\])

- "?:" es algo opcional que puede capturarse o no
- "s?" esto indica que puede ir la "s" o puede que no

Estas expresiones fueron aplicadas en gran parte del código, por lo que fue necesario explicar su funcionamiento y ponerlas a prueba. Con ello se obtuvieron resultados satisfactorios en la mayoría de los documentos analizados.

8.3.2.1 Configuración de Django y Plantillas usadas para las interfaces (HTMLs)

Durante la configuración de Django, fue necesario crear dos carpetas, una llamada "media" que se encuentra en la ruta donde instalamos el pipenv, como se muestra en la Figura 7 el proceso de instalación de pipenv. La carpeta contiene los documentos PDF los cuales se usaron para extraer los metadatos.

8.3.2.2 SETTINGS.PY

Para que Django reconozca esta carpeta, fue necesario añadir estas dos variables en el archivo settings.py:

MEDIA_URL = '/media/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

Y esto se añadió al archivo urls, justo después de la lista urlpatterns usando:

+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

La otra carpeta creada fue “templates” que es donde se aloja las plantillas HTML. Estas se usaron para crear las vistas del repositorio digital, como la página de inicio, las listas de documento, los detalles de los documentos entre otros.

Para que Django reconozca esta carpeta fue necesario incluir la ruta dentro del diccionario de TEMPLATES que se configura en el archivo settings.py, donde debe incluir estos dos parámetros:

- 'BACKEND': 'django.template.backends.django.DjangoTemplates',

- 'DIRS': [BASE_DIR / 'templates'],

Otra configuración importante realizada en el archivo settings.py fue la inclusión del nombre del proyecto dentro de la lista INSTALLED_APPS. Como se muestra en la Figura 15, se agregó la cadena 'repositorio', lo cual permitió que Django reconociera y activara la aplicación principal del sistema. Esta acción es importante para que se puedan ejecutar correctamente las vistas, modelos, formularios y demás componentes definidos dentro del proyecto.

8.3.2.3 VIEWS.PY

Para el desarrollo del repositorio digital, se implementaron vistas tanto basadas en funciones (Function-Based Views - FBV) como vistas basadas en clases (Class-Based Views - CBV), dependiendo del tipo de operación y la necesidad de reutilización de código. (Django Software Foundation, 2025)

Dentro de este archivo se incluyó funciones y clases para el funcionamiento del repositorio, como son:

- **Funciones de búsqueda**: Sirven para encontrar algunos atributos de la clase documento, como se

muestra en la figura 3, del diagrama de clases realizado, lo que se ingresa por texto puede encontrar coincidencias por título, autor, director, metodología y licenciatura

- **Funciones de validación**: Son aquellas funciones donde se valida si los parámetros extraídos han sido almacenados.
- **Funciones de extracción**: Las funciones de extracción fueron explicadas en el anexo 5, sin embargo, como fue incluido en el archivo views.py, se quitó las impresiones por consola "print" y la biblioteca tkinter, que ya no eran necesarios para este apartado.

Se utilizaron vistas basadas en clases, aprovechando las herramientas que Django ofrece como ListView, DetailView, CreateView, entre otras. Estas vistas permiten reducir la cantidad de código repetitivo y facilitan la implementación de formularios y controladores estándar.

Entre ellas se encuentran:

- **Clase DocumentoUpdateView**: aquí se aloja la función relacionada con la edición de información.
- **Clase DocumentoEliminarView**: Se encuentra la función para eliminar algún documento.
- **Clase DocumentoCreateView**: Se aloja el formulario donde se cargan los documentos para cada licenciatura.
- C**lase DocumentoEView, DocumentoDView y DocumentoTView**: Aquí se incluyen las vistas relacionadas con cada licenciatura.
- **Clase DocumentoDetailView**: Aquí se encuentra la función donde se observa los detalles de cada documento.

8.3.2.4 URLS.PY

En un proyecto Django, el archivo urls.py cumple la función de asociar las rutas web (URLs) con las vistas

del sistema, permitiendo que los usuarios accedan a las diferentes secciones del repositorio digital mediante enlaces o direcciones específicas.

Dentro de la lista de **urlpatterns** se incluyen las rutas web, que se van a utilizar para las vistas basadas en clases y en funciones. Cada vista tiene su ruta especifica, como novedad se encuentra la ruta de administrador "***path('admin/', admin.site.urls),***" y la ruta de detalles del documento "***path('documento/detalle/<int:pk>/',***" donde, int:pk es un indicador para cada documento cargado a la base de datos, en este caso el indicador es un numero entero.

8.3.2.5 BASE.HTML

El archivo base.html cumple la función de plantilla principal o "plantilla madre" dentro del sistema de plantillas de Django. Este archivo define la estructura general de la interfaz del repositorio digital, y sirve como punto de partida para las demás páginas del sitio.

Contiene elementos comunes a todas las vistas; el encabezado (header) donde se encuentra el título principal; el menú de navegación, donde se ubican botones para poder retornar a la página principal o cargar un archivo; los estilos CSS, que se usan para darle un estilo a cada elemento; y las referencias a archivos estáticos (como imágenes, íconos o archivos JavaScript). Gracias a esta estructura, se evita repetir el mismo código en cada plantilla individual.

Para esta plantilla HTML, se añadió una barra inferior que incluye varios elementos: el primero es una barra de búsqueda, que permite al usuario buscar por título, autor, director, licenciatura, metodología, descripción, contenidos y conclusiones; el segundo es un botón de filtros de búsqueda donde se puede escoger la licenciatura, sus respectivas líneas de investigación y el año de publicación; el tercero es un botón que redirige a la página principal del repositorio; y por último, se añadió un botón para cargar un documento y alojarlo donde corresponda.

8.3.2.6 PRINCIPAL.HTML

Esta plantilla es la página principal del proyecto donde se pueden navegar por los diferentes trabajos de grado de cada programa del Departamento de Tecnología.

Figura 29. Página principal del proyecto. Fuente. Elaboración propia

La Figura 29 muestra la interfaz de la página principal. Esta contiene elementos estéticos como lo es el logo de la universidad y del Departamento de Tecnología. Además, contiene iconos que representan cada licenciatura y la barra inferior es lo que se obtuvo como resultado de base.html.

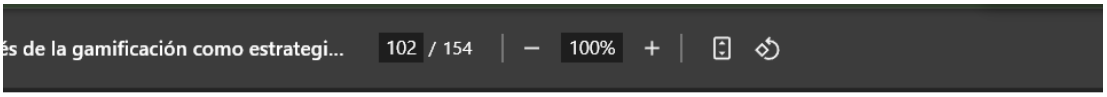
La idea es que por medio de botones me redirige a otra nueva plantilla. En la figura 29 se evidencia las tres licenciaturas del departamento de tecnología, es allí donde puedo seleccionar una y me va a dirigir a la lista de documentos que contiene esa licenciatura.

8.3.2.7 DOCUMENTO_DETALLE.HTML

Dentro de esta plantilla se aloja la extracción de metadatos de cada documento. Se tenía planeado conectar el repositorio de la universidad con el repositorio digital, para que se sincronizara con los trabajos que se iban cargando al sistema. El jueves 20 de febrero del 2025, en reunión con la directora del Departamento de Tecnologia, Patricia Tellez Lopez y con mi asesor Jimmy Ramirez, para hablar del tema, se nos informó

que esa información debía ser hablada con el director de la biblioteca, encargados de la gestión. El 21 de febrero del 2025, tuvimos una reunión con el señor Alejandro Toro, encargado de la Biblioteca Central de la Universidad, nos informó que, por temas de políticas institucionales y legales, no era posible establecer una conexión directa con la base de datos de la Universidad con el repositorio realizado. También, mencionó que no era permitido tener una descarga directa del archivo de forma local. Por ello se colocó un link que redireccionara al repositorio de la universidad y allí poder descargar el archivo. Es por esto que en el espacio de archivo se optó por dejar la descripción "Si desea consultar el documento completo el enlace lo direcciona al repositorio de la Universidad Pedagógica Nacional para su respectiva consulta y descarga.".

Ya habiendo aclarado esto, la plantilla aloja la información extraída, con su enlace de descarga que lo redirige al repositorio de la UPN, además se encuentran dos botones donde se puede eliminar y editar el documento en caso de que una información sea errónea. Durante la extracción de información se presentaron varias inconsistencias, una de ellas fue que algunos documentos no poseen explícitamente la información que se quiere buscar o simplemente no la contienen, como por ejemplo la metodología en documentos que hablan de algún tema en específico o los contenidos en algunos documentos que no tienen tabla de contenidos. Otra de las dificultades que se presentaron fue, el tipo de texto que se estaba extrayendo, como se muestra a continuación:



Desarrollo de competencias Digitales

Referencias

Alessandro, B. (2018). Digital skills and competence, and digital and online learning. TurinEuropean Training Foundation., 72. Recuperado de: https://www.etf.europa.eu/sites/default/files/2018-10/DSC and DOL_0.pdf

Alsawaier, R. S. (2018). The effect of gamification on motivation and engagement. International Journal of Information and Learning Technology, 35(1), 56–79. Recuperado de: https://doi.org/10.1108/IJILT-02-2017-0009

Ašeriškis, D., Blažauskas, T., & Damaševičius, R. (2017). UAREI : A model for formal description and visual representation / software gamification. Dyna, 84, 326–334. Recuperado de: https://www.redalyc.org/jatsRepo/496/49650910038/index.html

Amauris R., J. (2020). Fortalecimiento de las competencias tecnológicas de la información de los maestros de secundaria en los politécnicos de República Dominicana. Memorias Coloquio Doctoral HIU, 8va. Edición (pp. 140-150). Humboldt International University. Recuperado de

Figura 30. Referencias de un documento de la licenciatura en diseño tecnológico. Fuente. Tomado de Zambrano, 2022, Desarrollo de competencias digitales a través de la gamificación como estrategia en estudiantes de grado noveno (pag 102)

En la figura 30 se muestra un fragmento de uno de los documentos utilizados para la extracción, donde se muestra claramente que la pagina 102 trae el título “Referencias” y que cuenta con las fuentes correspondientes a esa sesión.

```
Texto de la página 102:

'Desarrollo de competencias Digitales
\nDyna, 84, 326-334. \nRecuperado de: https://www.redalyc.org/jatsRepo/496/49650910038/index.html
\nAmauris R., J. (2020). Fortalecimiento de las competencias tecnológicas de la información de los
\nmaestros de secundaria en los politécnicos de República Dominicana. Memorias \nColoquio Doctoral
HIU, 8va. Edición (pp. 140-150). Humboldt International University. \nRecuperado de \nhttps://
www.researchgate.net/profile/Manuel_Prieto3/publication/
343263983_MEMORI\nAS_DEL_COLOQUIO_DOCTORAL_HIU/links/5f205046299bf1720d6add41/MEMOR\nIAS-DEL-
COLOQUIO-DOCTORAL-HIU.pdf#page=151 \nAmezcua A., T. A., & Amezcua A., P. A. (2018). La
gamificación como estrategia de \nmotivación en el aula. En Gamificación en Iberoamérica (pp.
137-146). Recuperado de \nhttp://www.academia.edu/download/57490842/
Gamificacion_2octubre2018.pdf#page=13\n7 \nArdíla-Muñoz, J. Y. (2019). Supuestos teóricos para la
gamificación de la educación superior. \nMagis, Revista Internacional de Investigación en
Educación, 12(24), 71-84. Recuperado \nde: https://revistas.javeriana.edu.co/index.php/MAGIS/
article/view/25494 \nArufe-Giráldez, V. (2019). Fortnite EF, un nuevo juego deportivo para el aula
```

Figura 31. Texto de la página 102 del documento escogido. Fuente. Elaboración propia

Se usó una línea de código para extraer la información con los saltos de línea y caracteres invisibles para ver cómo se está extrayendo el texto, la línea es print(repr(texto[:3000])). En la figura 31 se muestra el resultado de cómo se está extrayendo la página 102 del documento. Se encontró que empieza la información en “Dyna” algo que podemos notar en la figura 30, señalado de color azul, podemos concluir que no está identificando todo lo que hay antes de ese fragmento. Por lo tanto, la extracción de fuentes de este documento no se extrajo en su totalidad.

Hay algunos documentos que tienen este tipo de fallos por lo que fue necesario añadir la opción de editar, para los documentos que se van cargando al sistema y añadir la información faltante manualmente. Algunos de los ejemplos serán añadidos en el anexo 6 para su visualización.

La tabla de detalles presenta la información correspondiente a los atributos definidos en la clase Documento, como se observa en la figura 3. Al momento de cargar un documento al sistema, parte de esta información es extraída automáticamente. Sin embargo, algunos campos, como la licenciatura a la que pertenece el trabajo, **la línea de investigación asociada** y el enlace al repositorio institucional de la Universidad Pedagógica Nacional, deben ser ingresados manualmente. Con esto, se da cumplimiento al **primer objetivo** específico “Aplicar la información suministrada por el grupo de investigación Educación y Regionalización en CTeI – GIER acerca de los trabajos de grado y las líneas de investigación identificadas para diseñar una primera interfaz del repositorio digital”.

8.3.2.8 NAVEGACIÓN DEL SISTEMA POR MEDIO DE LAS PLANTILLAS

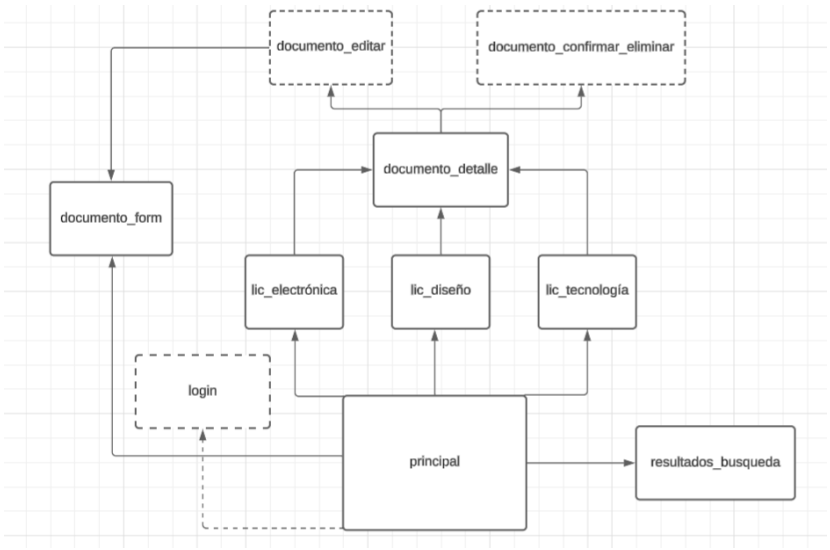


Figura 32. Conexión de plantillas en el navegador. Fuente. Elaboración propia

La figura 32 muestra la conexión que hay entre las plantillas donde la interfaz inicial es **"principal.html"** y de ahí se desprende las ramas de las plantillas. Desde ahí se puede acceder a las plantillas de: "documento_form", que se puede acceder desde el botón "cargar archivo", donde se puede cargar un trabajo de grado; "lic_electrónica", "lic_diseño" y "lic_tecnología, aquí se poder observar la lista de los trabajos de grado de cada licenciatura; "resultados_busqueda" , el cual se puede acceder desde la barra de búsqueda que hay en la barra inferior, haciendo cualquier búsqueda, dirigiéndose a la plantilla mostrando todos los resultados con o sin filtros aplicados; Por último, se accede a la plantilla "login" que sirve para iniciar sesión como administrador y poder eliminar, editar o cargar un archivo, por tal motivo se ven líneas punteadas en algunos cuadros.

Desde la lista de trabajos y desde los resultados de búsqueda, se puede acceder a la plantilla "documento_detalle", que se muestra la tabla con la información extraída del trabajo de grado seleccionado. Para poder editar o eliminar los trabajos de grado, se puede desde la plantilla "documento_detalle".

8.4 FASE 4: PRUEBAS Y MANTENIMIENTO

Durante esta fase se hizo la prueba del funcionamiento del repositorio, para poder realizar los cambios necesarios dentro del sistema

8.4.1 Funcionamiento del sistema

Una vez finalizado el desarrollo del repositorio digital, se procedió a cargar los documentos académicos, asignándoles su respectiva licenciatura, línea de investigación y enlace al repositorio institucional de la Universidad Pedagógica Nacional (UPN). Al cargar un nuevo documento, el sistema redirige automáticamente a la página principal presentada en la figura 28, donde se encuentran los programas del Departamento de Tecnología. Los trabajos almacenados pueden ser consultados mediante la selección de una licenciatura específica, lo que permite visualizar una lista de los trabajos correspondientes. Al seleccionar un trabajo de la lista, el sistema redirige a una plantilla detallada que presenta la información extraída en una tabla.

Desde la barra inferior de navegación, el usuario puede volver a la página principal mediante el botón designado. El sistema cuenta con una barra de búsqueda ubicada en la parte inferior izquierda, la cual permite realizar consultas específicas. Para facilitar la búsqueda, se ofrecen filtros que permiten hacer la búsqueda más sencilla. Al hacer clic en el botón de búsqueda, el sistema redirige a una plantilla que muestra los resultados obtenidos. En esta vista se presenta información clave como el título (que actúa como enlace a los detalles del documento), el autor o autores, el director, el año de publicación, la metodología, la licenciatura correspondiente y las líneas de investigación asociadas. En el Anexo 7 se incluirá un video demostrativo del funcionamiento completo del sistema.

8.4.2 Restricciones de administrador a usuario

Para garantizar la seguridad y el control del contenido, al momento de publicar el repositorio digital de

manera abierta al público, se optó por restringir ciertas funcionalidades que no deben estar disponibles para todos los usuarios. Específicamente, se limitaron las acciones de cargar, eliminar y editar documentos, ya que estas operaciones deben estar bajo la supervisión y autorización de administradores o personal encargado. Django nos ofrece herramientas integradas para los administradores. Para este proyecto se implementó un sistema de inicio de sesión para que únicamente el administrador tenga permisos de acceso para estas funciones.

8.4.3 Filtros de búsqueda

Los filtros de búsqueda implementados en el sistema facilitan la localización de la información contenida en los trabajos de grado. Uno de los filtros que se implementaron, es la posibilidad de escoger la licenciatura, el segundo filtro depende del primero ya que son las líneas de investigación que van por cada licenciatura y el ultimo filtro es por año de publicación.

La barra de búsqueda, ubicada dentro de la plantilla principal base.html, permite realizar consultas por diferentes parámetros, como palabras clave, título, autor, licenciatura o línea de investigación. Una vez el usuario realice la búsqueda, se redirige a la plantilla "resultados_busqueda.html". Esta vista muestra una visualización resumida de los documentos que coinciden con la búsqueda, además de la licenciatura y la línea o las líneas de investigación que corresponde ese documento. Esta clasificación fue previamente realizada, lo que permitió dejar una organización clara por línea de investigación correspondiente a cada trabajo de grado. Por tanto, se da cumplimiento con el **tercer y último objetivo específico** "Reorganizar la información extraída de los trabajos de grado en un formato claro y sistematizado que facilite a los estudiantes del Departamento de Tecnología la búsqueda rápida de información mejorando así la accesibilidad de antecedentes relevantes". Con ello se da cumplimiento al objetivo general "Diseñar e implementar un aplicativo digital que integre metadatos relevantes y enlaces de descarga a los trabajos de grado realizados por los estudiantes del Departamento de Tecnología de la Universidad Pedagógica

Nacional durante los últimos cinco años, haciendo uso del framework Django" y dando cumplimiento al objetivo general que consiste en "Diseñar e implementar un aplicativo digital que integre metadatos relevantes y enlaces de descarga a los trabajos de grado realizados por los estudiantes del Departamento de Tecnología de la Universidad Pedagógica Nacional durante los últimos cinco años, haciendo uso del framework Django".

8.4.4 Aplicación web del repositorio digital

Con todo listo, se actualizó el repositorio que previamente creamos en GitHub, en la sección 8.2.1.5, haciendo uso de la terminal de comandos de Windows, se dirigió a la ruta del proyecto por medio del comando cd, estando ahí, se usaron los siguientes comandos,

***git add -A*:** Se añade los archivos modificados

git commit -m "Cambios": Para dar una descripción de los cambios realizados

git push origin master: Sube los cambios al repositorio publicado en GitHub

Luego se realizó, un registro en la página de **Render.com** que nos permitió desplegar el aplicativo Web. Durante el proceso, se configuró colocando el nombre, "repositoriodte". Luego se estableció conexión con GitHub y se seleccionó el repositorio. En Star Command se colocó el comando ***gunicorn config.wsgi:application ,*** y finalmente se inició el servidor Web, quedando disponible la aplicación en línea para su acceso y funcionamiento.

La base de datos funciona de manera distinta, Render necesita de un almacenamiento externo para guardar los trabajos de grado, debido a que al reconectar el servidor se borra la información por completo. Por tal motivo es necesario crear una base de datos PostgreSQL y configurarla en el archivo **settings**, se puede consultar el archivo en el anexo 4. Una de las optimizaciones que se hizo en el código fue extraer la información desde el "enlace", sin necesidad de subir el archivo de manera local. La configuración se realizo

desde el archivo **views** añadiendo un elif en la línea 264. Lo demás estará funcionando con normalidad y el servidor queda activo. Sí ningún usuario o el administrador interactúa durante 15 minutos con el servidor, se cierra la aplicación, pero cualquiera que ingrese al link puede activarlo y esperar 2 o 3 minutos mientras se inicia nuevamente la aplicación. Para su consulta ingresa a https://repositoriodte.onrender.com

9. Conclusiones

- La tabla de detalles presenta la información correspondiente a los atributos definidos en la clase Documento, como se muestra en la figura 3. Cuando se carga un documento al sistema, varios de estos datos son extraídos de forma automática mediante el sistema de extracción implementado. No obstante, existen campos que requieren ser diligenciados manualmente, como la licenciatura a la que pertenece el trabajo, las líneas de investigación, las cuales fueron suministradas por el grupo de investigación Educación y Regionalización en CTeI – GIER, y el enlace al repositorio institucional de la Universidad Pedagógica Nacional. Este procedimiento permitió complementar la primera construcción del repositorio, mostrando los detalles de los documentos. De esta manera, se dio cumplimiento al primer objetivo específico: “Aplicar la información suministrada por el grupo de investigación Educación y Regionalización en CTeI – GIER acerca de los trabajos de grado y las líneas de investigación identificadas para diseñar una primera interfaz del repositorio digital”. En la sección 8.3.3.6
- En el código del extractor de información, se implementó un sistema divido en dos partes principales: uno adaptado para trabajar con documentos que no siguen el formato RAE y otro específicamente diseñado para aquellos que sí lo utilizan. Esta organización permitió mantener el código limpio y estructurado, evitando que las diferencias en los formatos interfieran en el proceso de extracción. Con esto, se dio cumplimiento al segundo objetivo específico del proyecto: “Implementar un sistema automatizado que permita extraer información clave de los trabajos de grado, como palabras clave,

objetivos, metodología, título, contacto, etc.”. En la sección 8.3.1.4.

- La integración de filtros de búsqueda en el sistema, facilitó significativamente la localización de la información contenida en los trabajos de grado. Se incorporaron tres filtros principales: por programa, por línea de investigación dependiendo el programa seleccionado y por año de publicación. Además, se integró una barra de búsqueda dentro de la plantilla base.html que permite realizar consultas por palabras clave, título, autor, director y metodología, mostrando los resultados en la plantilla resultados_busqueda.html, donde se presenta una vista resumida de los documentos relacionados. Esta funcionalidad cumple con el tercer objetivo específico del proyecto, que consiste en reorganizar la información extraída de los trabajos de grado en un formato claro y sistematizado que permitiera a los estudiantes del Departamento de Tecnología realizar búsquedas rápidas y mejorar el acceso a antecedentes relevantes.
- La implementación del repositorio digital del Departamento de Tecnología de la Universidad Pedagógica Nacional representa un avance significativo en la gestión y socialización del conocimiento generado por los estudiantes en sus trabajos de grado. Esta herramienta digital no solo permite centralizar la información académica, sino que además facilita su acceso, organización y consulta de antecedentes para futuros proyectos.
- Gracias a este proyecto, mejoró la visibilidad de los trabajos de grado del Departamento de Tecnología, haciendo uso de filtros por línea de investigación, año y licenciatura, para fomentar una consulta eficiente, análisis y reutilización del conocimiento producido en el ámbito universitario.
- Una de los metas principales de este proyecto fue ofrecer una plataforma que apoye a futuras investigaciones. El sistema permite organizar los trabajos de grado por líneas de investigación de las tres licenciaturas (Electrónica, Diseño Tecnológico y Tecnología), brindando a los estudiantes de

semestres iniciales una guía sobre los enfoques, metodologías y problemas abordados anteriormente. El trabajo acumulado impulsa la construcción de nuevas propuestas investigativas con base en antecedentes claros, generando una continuidad en los procesos académicos.

- Otro aspecto clave del desarrollo fue, la automatización de la extracción de información desde archivos PDF, lo que permite cargar metadatos relevantes como título, autor, director, año de publicación, descripción, metodología, contenidos, conclusiones y líneas de investigación, de forma semiautomatizada. Esta funcionalidad reduce el tiempo de carga manual y promueve la eficiencia en la digitación de los formatos, lo cual es importante para un correcto funcionamiento del sistema de búsqueda y clasificación. A pesar de que algunos campos aún requieren intervención humana (como la categoría, las líneas de investigación y el enlace institucional), la estructura del sistema permite futuras mejoras que incluyan validación automática o asistencia mediante técnicas de expresiones regulares.
- El repositorio fue pensado para facilitar la experiencia de los usuarios, estudiantes o docentes. El sistema cuenta con filtros funcionales por línea de investigación, categoría y año, además de una barra de búsqueda eficiente. Esto representa una mejora significativa frente al acceso tradicional a documentos o dispersos en el repositorio de la UPN.
- El sistema desarrollado deja abiertas las puertas para futuras ampliaciones debido a su publicación en GITHUB. Por un lado, puede usarse para futuros proyectos sobre repositorios digitales haciendo uso de Django y SQLite que garantizan un entorno estable, flexible y de bajo costo, apropiado para una institución educativa con recursos limitados. Por otro, puede usarse como ejemplo para repositorios de acceso abierto como Redalyc o la Biblioteca Digital Colombiana.
- Este repositorio digital contribuye a consolidar la identidad del Departamento de Tecnología como un departamento académico comprometido con la sistematización y divulgación del conocimiento. Sirve

como apoyo para la sistematización y enfoque de los trabajos de grado pertenecientes a este departamento. Fomenta el sentido de pertenencia y orgullo institucional al ver sus producciones documentadas y disponibles para consulta.

- Este trabajo de grado permitió poner en práctica saberes adquiridos a lo largo del programa académico en áreas como informática, diseño de bases de datos, modelado de sistemas, lógica de software, y fundamentos pedagógicos. Así, el proceso de desarrollo del sistema sirvió como evidencia concreta de mi proceso formativo demostrando la posibilidad de aplicar lo aprendido a proyectos con impacto real en la comunidad académica.

- A lo largo del desarrollo del repositorio digital se aplicaron conocimientos en áreas clave como la estructuración de bases de datos, la programación web con Django, y la aplicación de técnicas de extracción automática de información a partir de archivos PDF. Sin embargo, el proceso no estuvo exento de dificultades que promovieron mi experiencia formativa. Una de las principales fue lidiar con la variabilidad de formatos de los documentos, lo que exigió implementar expresiones regulares avanzadas y herramientas o funciones integradas en fitz (PyMuPDF), para lograr resultados precisos. También se presentaron obstáculos en temas de que el sistema violaba algunas de las políticas institucionales, en especial al integrar una descarga de archivo local, por lo que fue necesario solicitar un cambio de objetivo general al programa. En la parte visual, se requirió un trabajo constante para mantener una interfaz coherente, adaptable y funcional, especialmente al incluir filtros, vistas diferenciadas por licenciatura y mecanismos de búsqueda avanzada. Sin contar, con el manejo individual de todas las fases del proyecto lo cual tomo tiempo y habilidades de resolución de problemas. Pese a todas estas dificultades fueron superadas con éxito, permitiendo no solo la culminación del sistema, sino también el fortalecimiento de competencias técnicas, investigativas y de programación fundamentales para la formación profesional

10.REFERENCIAS BIBLIOGRÁFICAS

[1] Educación y regionalización en CTeI – GIER (2019). Análisis de la producción de conocimiento en la formación de Licenciados en Diseño Tecnológico y Electrónica de la Universidad Pedagógica Nacional. Una mirada desde los trabajos de grado entre 2017 y 2022

[2] Creswell, J. W. (2014). Research design: Qualitative, quantitative, and mixed methods approaches. Thousand Oaks, CA: Sage Publications

[3] García Pérez, A. (2001). La gestión de documentos electrónicos como respuesta a las nuevas condiciones del entorno de información. Acimed, 9(3), 190-200.X

[4] Ariza, M. B. (2007). Abordaje hermenéutico de la investigación cualitativa. Teorías, procesos, técnicas. U. Cooperativa de Colombia.

[5] Tuckman, B. W., & Hinkle, J. S. (1986). An experimental study of the physical, temporal, and methodological aspects of role playing. The Journal of Experimental Education, 54(1), 37-47.

[6] González, L. (2014). Sistema de información para consultar y administrar los documentos de los trabajos de grado de la Licenciatura en Electrónica. Trabajo de grado para optar al título de Licenciada en Electrónica, Departamento de Tecnología, Facultad de Ciencia y Tecnología, Universidad Pedagógica Nacional, Bogotá, Colombia. http://repository.pedagogica.edu.co/handle/20.500.12209/1967

[7] Caicedo Mosquera, D. F., & Cespedes Yela, G. A. (2015). Sistema de información web para el repositorio de los documentos generados en el proceso de auto evaluación y el plan de mejoramiento de los programas. Tomado de

https://repositorio.unillanos.edu.co/bitstream/handle/001/316/Paper%203.pdf?sequence=4

[8] Basabe Chacón, C. A. (2021). Análisis de las tendencias de los proyectos de grado del Departamento de Educación Musical de la UPN: hacia la configuración de la Línea de Investigación Creación, una mirada entre 2016 Y 2017. Tomado de http://upnblib.pedagogica.edu.co/handle/20.500.12209/17134

[9] Dulzaides Iglesias, M. E., & Molina Gómez, A. M. (2004). Análisis documental y de información: dos componentes de un mismo proceso. Acimed, 12(2), 1-1.

[10] Celi Luis (2018). Líneas y Áreas de Investigación de la Escuela Politécnica Nacional. Tomado de https://www.epn.edu.ec/wp-content/uploads/2018/11/%C3%81reas-y-L%C3%ADneas-de-Investigaci%C3%B3n.pdf

[11] Trasobares, A. H. (2003). Los sistemas de información: evolución y desarrollo. Proyecto social: Revista de relaciones laborales, (10), 149-165. Tomado de https://dialnet.unirioja.es/servlet/articulo?codigo=793097

[12] Supelano, L. F. (2023). Resolución de problemas geométricos y algebraicos a través de la programación usando el lenguaje Python en la Institución Educativa Departamental Rafael Pombo Sopó. Recuperado de: http://hdl.handle.net/20.500.12209/18934.

[13] Barrera Rea, M. G., & Barros Naranjo, M. W. (2017). *Análisis de los procesos de divulgación científica de los docentes investigadores para el diseño de un Repositorio WEB en la Universidad Estatal de Milagro* (Bachelor's thesis). Tomado de https://repositorio.unemi.edu.ec/handle/123456789/3828

[14] Keefer, A. (2008). Los repositorios digitales universitarios y los autores. *Anales de Documentación*, *10*, 205–214. Recuperado a partir de

https://revistas.um.es/analesdoc/article/view/1151

[15] Macias Carpio, F. A. (2023). *Estudio Comparativo entre SQL Directo y un ORM* (Bachelor's thesis, Babahoyo: UTB-FAFI. 2023). Tomado de http://dspace.utb.edu.ec/handle/49000/14779

[16] Rivera, M. D. C. G. (1994). La base de datos. Importancia y aplicación en educación. Perfiles educativos, (65). Tomado de https://www.redalyc.org/pdf/132/13206506.pdf

[17] Villalobos, G. M., Sánchez, G. D. C., & Gutiérrez, D. A. B. (2010). Diseño de framework web para el desarrollo dinámico de aplicaciones. *Scientia et technica*, *16*(44), 178-183. Tomado de https://www.redalyc.org/pdf/849/84917316032.pdf

[18] Vamsi, K. M., Lokesh, P., Reddy, K. N., & Swetha, P. (2021). Visualization of real world enterprise data using python Django framework. In IOP Conference Series: Materials Science and Engineering (Vol. 1042, No. 1, p. 012019). IOP Publishing.

[19] Ghimire, D. (2020). Comparative study on Python web frameworks: Flask and Django.

[20] Canals, A. (2003). La gestión del conocimiento. Tomado de http://148.202.167.116:8080/xmlui/handle/123456789/3300

[21] Biblioteca Central UNAM. La búsqueda de información. Tomado de https://www.bibliotecacentral.unam.mx/index.php/desarrollo-de-capacidades-informativas-digitales-y-comunicacionales/la-busqueda-de-informacion

[22] Red Hat (2023). ¿Qué es el open source?. Tomado de https://www.redhat.com/es/topics/open-source/what-is-open-source#:~:text=Originalmente%2C%20la%20expresi%C3%B3n%20open%20source,la%20forma%20que%20consideren%20conveniente

[23] García-Peñalvo, F. J. (2017). Mitos y realidades del acceso abierto. Education in the Knowledge Society, 18(1), 7-20. Tomado de https://www.redalyc.org/pdf/5355/535554765001.pdf

[24] Vásquez, A. C., Quispe, J. P., & Huayna, A. M. (2009). Procesamiento de lenguaje natural. Revista de investigación de Sistemas e Informática, 6(2), 45-54. Tomado de https://www.academia.edu/download/77493941/5121.pdf

[25] Sguerra, M. D. (2006). Las expresiones regulares. *INVENTUM*, *1*(1), 31-37. Tomado de https://revistas.uniminuto.edu/index.php/Inventum/article/download/145/146

[26] FileFormat Products. (n.d.). *PyMuPDF: Librería Python para trabajar con PDFs*. Tomado de https://products.fileformat.com/es/pdf/python/pymupdf/

[27] Ramírez-Acosta, K. (2017). Interfaz y experiencia de usuario: parámetros importantes para un diseño efectivo. *Revista tecnología en marcha*, *30*, 49-54. Tomado de https://www.scielo.sa.cr/scielo.php?script=sci_arttext&pid=S0379-39822017000500049

[28] Royce, W. (1997). Gestión de proyectos de ingeniería de software (2.a ed.). Los Ángeles: With R. Thayer & Ed Yourdon. Tomado de https://www.ride.org.mx/index.php/RIDE/article/view/905/2938

[29] Python Packaging Authority (PyPA). (2020). *Pipenv: Python Development Workflow for Humans*. Proyecto fundado por Kenneth Reitz. https://pipenv.pypa.io

[30] Valbuena, J. O. (2017). Análisis de la emergencia de la educación en Tecnología en Colombia (1994-2015). Recuperado de: http://hdl.handle.net/20.500.12209/9555.

[31] Torres, C. & Salazar, J. F. (2017). Análisis histórico, técnico y educativo de las Escuelas Radiofónicas de Acción Cultural Popular (ACPO) de 1947 a 1967. Recuperado de:

http://hdl.handle.net/20.500.12209/9554.

[32] Valencia, J. C. (2017). Comparación cualitativa de programas de generación de modelos 3D para documentar y digitalizar un yacimiento arqueológico con arte rupestre. Recuperado de: http://hdl.handle.net/20.500.12209/9556

ANEXOS

Anexo 1. Código de la primera prueba de PyMuPDF.

https://github.com/DiegoSuarez01/anexos/blob/main/anexo%201.py

Anexo 2. Código del segundo intento con spacy y pdfplumber.

https://github.com/DiegoSuarez01/anexos/blob/main/anexo%202.py

Anexo 3. Código del tercer intento con NLTK.

https://github.com/DiegoSuarez01/anexos/blob/main/prueba%203.py

Anexo 4. Código completo del extractor de información.

https://github.com/DiegoSuarez01/anexos/blob/main/rae2.py#L14

- Enlace al repositorio con todos los archivos. https://github.com/DiegoSuarez01/repositorio.git

Anexo 5. Explicación de las funciones que se usaron para el extractor de información. ANEXO 5

Anexo 6. Ejemplos de formatos que fueron fallidos para la extracción. Anexo 6

Anexo 7. Video explicativo del funcionamiento del repositorio digital. https://youtu.be/MUyVduVcv1Q