ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

PROTOTIPO DE DESCRIPCIÓN DE FIGURAS GEOMÉTRICAS PARA NO VIDENTES.

REALIZAR UN PDF INCLUSIVO, DONDE CONSTE LA DESCRIPCIÓN DE FIGURAS OBTENIDAS DE UN ARCHIVO LATEX USANDO UN LLM

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

CARLOS JOSTIN LEÓN GALEAS

jostinleon18@gmail.com - carlos.leon01@epn.edu.ec

DIRECTOR: PhD. ANA MARÍA ZAMBRANO VIZUETE

ana.zambrano@epn.edu.ec

DMQ, agosto 2024

CERTIFICACIONES

Yo, Carlos Jostin León Galeas declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

CARLOS JOSTIN LEÓN GALEAS

Certifico que el presente trabajo de integración curricular fue desarrollado por Carlos Jostin León Galeas, bajo mi supervisión.

_____

ANA MARÍA ZAMBRANO VIZUETE
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

CARLOS JOSTIN LEÓN GALEAS

ANA MARÍA ZAMBRANO VIZUETE

DEDICATORIA

A mis padres por darme siempre su apoyo incondicional, por cada uno de los consejos, ánimos y enseñanzas que me dieron a lo largo de toda mi vida y brindarme todo lo necesario para lograr conseguir este objetivo.

A mis tíos por recibirme en su hogar como uno más de sus hijos, y brindarme su ayuda y las facilidades para poderme desarrollar plenamente en el ámbito educativo y personal.

A mi hermano y mis primas por siempre ser ese apoyo emocional que llegué a necesitar para seguir avanzando en mi formación profesional.

A todos mis amigos que hice a lo largo de esta hermosa travesía, y espero que nuestros caminos se junten en algún punto de nuestras vidas.

- Carlos León

AGRADECIMIENTO

Agradezco a Dios por haberme llevado por este camino y darme salud y fuerza para seguir adelante día a día. A su vez, no me cansaré de agradecer a mi mamá y a mi papá por apoyarme siempre, y que sus enseñanzas formaron la persona que soy ahora.

Agradezco al personal docente de la EPN que formaron parte de mi formación profesional y quienes me brindaron su consejo o su experiencia para mejorar como persona. Especialmente agradezco a la PhD. Anita Zambrano por confiar en mí al proponerme trabajar en este TIC y guiarme a través de todo el proyecto. A su vez, agradezco al profe Marco Molina por enseñarme que con paciencia y dedicación de ambas partes es posible aprender lo propuesto; al profe Julio Caiza por confiar en mi para colaborar en sus proyectos, que me dieron mucha experiencia complementaria en mi formación; y a la profe Anita Rodríguez por su paciencia y dedicación que demuestra al dar sus clases.

Agradezco a todos mis amigos y compañeros de clase que hice a lo largo de mi formación profesional, porque de alguna u otra forma influyeron en mi desempeño a través de esta gran aventura. Especialmente agradezco a Gaby por ser esa amiga incondicional que, a pesar de no estar siempre juntos, me animó siempre a luchar y seguir adelante hasta conseguir mis objetivos. También agradezco a José por ser ese amigo que me acompañó en casi todas las asignaturas de mi formación, por ser el que me salvó muchas veces de las materias complicadas, y me animó a no rendirme jamás a pesar de las dificultades que se presentaban.

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo aborda el desarrollo de una aplicación web que genera archivos PDF inclusivos, específicamente diseñados para mejorar la accesibilidad de figuras en formatos SVG y TikZ para personas con discapacidad visual. La iniciativa surge de la necesidad de proporcionar descripciones textuales detalladas que acompañen a las imágenes en documentos científicos, educativos y técnicos, facilitando así el acceso a la información visualmente representada. La aplicación utiliza un modelo de lenguaje grande (*LLM*) para analizar y describir figuras complejas, integrando estas descripciones como etiquetas de texto alternativo en el código LaTeX original. Este proceso se realiza automáticamente, asegurando precisión y consistencia en las descripciones.

Además, se implementan tecnologías como Web Speech API y gTTS para proporcionar una experiencia auditiva, permitiendo a los usuarios recibir retroalimentación sonora mientras navegan por la aplicación. La aplicación web también soporta navegación por teclado y ofrece diferentes velocidades de lectura para adaptarse a las preferencias de los usuarios. El uso del framework Flask y la metodología Kanban en el desarrollo del proyecto aseguraron una gestión ágil y eficiente, facilitando la implementación y pruebas del sistema.

Los resultados demuestran que la aplicación es efectiva en la creación de documentos accesibles, mejorando significativamente la inclusión digital de personas con discapacidad visual. Esta herramienta promueve una mayor equidad en el acceso a la información, contribuyendo al avance hacia una sociedad más inclusiva. La integración de IA y tecnologías del habla subraya el potencial de las soluciones innovadoras para abordar los problemas de accesibilidad de los contenidos digitales.

PALABRAS CLAVE: PDF etiquetado, accesibilidad digital, herramienta con IA, PDF inclusivo, usuario no vidente, Text To Speech.

ABSTRACT

This study details the development of a web application designed to create inclusive PDFs, aimed at enhancing the accessibility of figures in SVG and TikZ formats for individuals with visual impairments. The initiative addresses the critical need for providing detailed textual descriptions alongside images in scientific, educational, and technical documents, thereby facilitating access to visually represented information. The application employs a Large Language Model (LLM) to analyze and describe complex figures, integrating these descriptions as alternative text tags within the original LaTeX code. This process is automated, ensuring accuracy and consistency in the descriptions provided.

Additionally, technologies such as the Web Speech API and gTTS are implemented to offer an auditory experience, allowing users to receive audio feedback while navigating the application. The web application also supports keyboard navigation and offers variable reading speeds to accommodate user preferences. The use of the Flask framework and the Kanban methodology in project development ensured agile and efficient management, facilitating the implementation and testing phases of the system.

The results indicate that the application is effective in producing accessible documents, significantly enhancing digital inclusion for individuals with visual impairments. This tool promotes greater equity in access to information, contributing to the broader goal of a more inclusive society. The integration of IA and speech technologies underscores the potential for innovative solutions in addressing accessibility challenges in digital content.

KEYWORDS: Tagged PDF, Digital Accessibility, AI Tool, Inclusive PDF, Visually Impaired User, Text To Speech

1 INTRODUCCIÓN

En el mundo actual, el avance tecnológico ha llevado a una mayor conciencia sobre la importancia de la inclusión y accesibilidad en todos los aspectos de la sociedad. En este contexto, es crucial desarrollar aplicaciones y herramientas que fomenten la accesibilidad desde un punto de vista ético y social. Surge así la necesidad de abordar el mejoramiento de documentos digitales accesibles, como los archivos PDF, los cuales desempeñan un papel fundamental en la comunicación e intercambio de información en ámbitos tan diversos como la educación y el mundo empresarial.

Una de las problemáticas más apremiantes es la dificultad que enfrentan las personas con discapacidad visual para acceder a la información contenida en imágenes, por ejemplo en formato SVG y TikZ, formatos ampliamente utilizados en documentos científicos, educativos y técnicos para representar gráficos y diagramas complejos. Sin descripciones textuales adecuadas, estas imágenes se vuelven inaccesibles, privando a los usuarios no videntes de información crucial. Este desafío no solo limita la autonomía y el acceso a la información de estas personas, sino que también crea barreras significativas en su educación y desarrollo profesional. La falta de descripciones adecuadas en estos formatos contribuye a una brecha digital que impide la plena participación e inclusión de las personas con discapacidad visual en la sociedad.

A lo largo del presente Trabajo de Integración Curricular (TIC) se explora el desarrollo de una aplicación web para generar archivos PDF inclusivos, con un enfoque específico en mejorar las descripciones de figuras en formato SVG y TikZ. Este enfoque surge de la necesidad apremiante de facilitar el acceso a la información para personas con discapacidad visual, un grupo cuya integración digital ha sido históricamente desafiante y olvidado. Al abordar esta problemática, se busca no solo cumplir con estándares de accesibilidad, sino también promover una sociedad más equitativa y accesible.

Con el objetivo de lograr este fin, se establecen metas concretas que abarcan desde el análisis de herramientas y tecnologías pertinentes hasta la implementación y prueba de la aplicación. Se hace énfasis en la importancia de desarrollar un flujo de trabajo eficiente que permita añadir etiquetas de texto alternativo al código LaTeX, un paso fundamental para la generación de documentos PDF inclusivos. Este proceso incluye la automatización de la generación de descripciones utilizando *LLM*[1] (*Large Language Model*) que puedan interpretar y describir imágenes complejas de manera precisa [1].

El proyecto delimita claramente su alcance, identificando las funciones clave de la aplicación y los criterios que guiarán su desarrollo. Se destaca la necesidad de integrar

[1] LLM es un tipo de programa de inteligencia artificial que puede reconocer, interpretar y generar texto en lenguaje humano.

tecnologías como un *LLM* para la generación automática de descripciones de figuras en formato SVG y TikZ, así como el uso de herramientas como la *Web Speech API* y *gTTS* para garantizar la accesibilidad auditiva de los documentos generados. Estas tecnologías se seleccionan no solo por su capacidad técnica, sino también por su potencial para transformar radicalmente la accesibilidad digital.

El Marco Teórico proporciona un contexto esencial para comprender la importancia y viabilidad del TIC, abordando desde la definición de un PDF inclusivo hasta la descripción de las herramientas y tecnologías específicas que se emplearán en el desarrollo del prototipo. Se hace énfasis en la relevancia de la metodología Kanban como un enfoque eficaz para la gestión ágil de proyectos, lo que se traduce en una mayor eficiencia y organización durante el proceso de desarrollo. Esta metodología permitirá una adaptación flexible y una mejora continua en el desarrollo de la aplicación, asegurando que se cumplan todos los objetivos planteados.

1.1 OBJETIVO GENERAL

Desarrollar una aplicación web para la generación de un PDF inclusivo, con respecto a mejorar las descripciones de figuras SVG y TikZ, a partir de código LaTeX usando un *LLM*.

1.2 OBJETIVOS ESPECÍFICOS

1. Analizar herramientas y tecnologías necesarias para el desarrollo de la aplicación web, y para el manejo de texto alternativo de las figuras SVG y TikZ.
2. Diseñar el flujo de trabajo para agregar las etiquetas de texto alternativo al código LaTeX dentro de la aplicación.
3. Implementar la aplicación web en base al diseño planteado.
4. Probar el funcionamiento general de la aplicación.

1.3 ALCANCE

En este TIC se plantea el desarrollo de una aplicación web destinada a la generación de un PDF inclusivo en lo que respecta a mejorar la descripción de figuras SVG y TikZ. La aplicación permitirá cargar una carpeta que deberá contener un archivo LaTeX, y el resto de los archivos necesarios para generar correctamente el PDF (por ejemplo, imágenes en diferentes formatos, paquetes personalizados, etc.). A partir de esta carpeta, se generará un nuevo archivo *.tex* que incluirá las etiquetas de texto alternativo que describen de mejor manera las figuras en formato SVG y TikZ. Se planea usar un *LLM* para obtener la

descripción mejorada de las figuras, y la respuesta proporcionada se incluirá en las etiquetas de texto alternativo.

La aplicación generará un PDF inclusivo en cuanto a la descripción de figuras a partir del archivo LaTeX etiquetado. Este PDF generado contendrá las descripciones de las figuras como texto alternativo, el cual deberá ser leído por el lector de PDF Adobe Acrobat Reader. Por último, la aplicación producirá un audio mediante un *TTS*[2] (Text To Speech) del PDF, lo cual permitirá al usuario escuchar el contenido del PDF inclusivo directamente desde la aplicación.

A continuación, se destacan algunas funcionalidades que tendrá la aplicación web:

a. Narración integrada y manejo por teclado:

La aplicación web contará con un narrador activado por defecto, que proporcionará retroalimentación auditiva mientras el usuario con discapacidad visual navega por la aplicación mediante el tabulador. El usuario tendrá la opción de desactivar el narrador o cambiar su velocidad de habla, pudiendo elegir entre: normal (x1.2), rápida (x1.5) y muy rápida (x2). La aplicación también incluirá atajos de teclado que permitirán navegar directamente a las opciones deseadas, de modo que el usuario pueda desplazarse en el orden que prefiera y evitar el orden predeterminado.

b. Selección de la carpeta desde el sistema de archivos:

La aplicación permite seleccionar una carpeta del sistema de archivos, la cual debe contener un único archivo LaTeX, además del resto de archivos necesarios para generar correctamente el PDF. Para el caso de los usuarios con discapacidad visual, será necesario que utilicen un lector de pantalla en su computador para navegar en su sistema de archivos y seleccionar la carpeta que desean subir. La aplicación proporcionará una retroalimentación auditiva una vez que se complete correctamente la carga de los archivos de la carpeta, o si la carpeta no cumple con los parámetros establecidos (contener un único archivo LaTeX).

c. Descargar el contenido generado:

Una vez cargada la carpeta y procesado el archivo LaTeX, la aplicación generará un nuevo archivo .tex con las etiquetas de texto alternativo, el PDF inclusivo en cuanto a la descripción de figuras, y el audio del PDF. El usuario podrá elegir cada sección del PDF que desee escuchar. La aplicación permitirá al usuario descargar cada uno de los contenidos mencionados.

[2] TTS es una tecnología que convierte texto a voz y en aplicaciones permite leer en voz alta un texto dado.

1.4 MARCO TEÓRICO

En este apartado se detallan los conceptos más relevantes necesarios para entender lo que implica generar un PDF inclusivo, tener noción de las herramientas y tecnologías a utilizar para la implementación del prototipo, y la metodología que se utilizará para gestionar este TIC.

1.4.1 PDF INCLUSIVO

Un PDF inclusivo es un documento digital que contiene metadatos, como texto alternativo en imágenes y el uso de etiquetas para identificar la estructura del documento, permitiendo que las tecnologías asistidas, como los lectores de pantalla, faciliten la navegación [2]. Estos metadatos incluyen información detallada que describe cada elemento del documento, asegurando que el contenido sea interpretado adecuadamente por las tecnologías asistidas. Principalmente, la accesibilidad en los documentos PDF se logra mediante la adición de etiquetas que definen la estructura del documento y proporcionan descripciones útiles de su contenido.

Para que un PDF sea considerado inclusivo, debe cumplir con ciertos estándares de accesibilidad, como los establecidos por la Iniciativa de Accesibilidad Web (WAI) del W3C [3], en particular las Directrices de Accesibilidad para contenido Web (WCAG). Entre estos requisitos se encuentran la inclusión de texto alternativo para todas las imágenes, la correcta jerarquización de títulos y subtítulos, y la definición clara de las tablas y listas. Además, el PDF debe incluir un orden lógico de lectura y una estructura que sea reconocida por los lectores de pantalla.

Las etiquetas en un PDF inclusivo sirven para identificar la estructura del documento, el propósito de cada elemento, e incluir texto alternativo a recursos gráficos, como imágenes. Estas etiquetas permiten a las tecnologías asistidas interpretar correctamente el contenido del documento y presentarlo de manera comprensible para los usuarios. Por tanto, el PDF inclusivo mejora la experiencia de los usuarios y amplía el acceso a la información a más personas.

En la actualidad existen algunas limitaciones en la creación y distribución de un PDF inclusivo. La primera de ellas es la necesidad de usar herramientas y software específicos que no siempre están disponibles de forma gratuita. Además, la creación de documentos accesibles requiere conocimiento técnico sobre todo lo que implica, el cumplimiento de las normativas de accesibilidad y la capacidad de aplicar estos conocimientos de manera efectiva.

Otra limitación es la incompatibilidad de los lectores de PDF a todas las características de accesibilidad. Por esta razón, el PDF inclusivo generado por la aplicación se garantiza que será compatible con Adobe Acrobat Reader. La variabilidad en cómo será leído por este lector de PDF, dependerá de la configuración previa del usuario y de la versión de Adobe Acrobat Reader que se esté utilizando.

1.4.2 HERRAMIENTAS Y TECNOLOGÍAS PARA LA IMPLEMENTACIÓN DEL PROTOTIPO

En este subcapítulo se expondrán las diferentes herramientas y tecnologías utilizadas para el desarrollo e implementación de la aplicación web, detallando características y beneficios. Empezando con Flask [4], un microframework web de Python para el desarrollo del *back-end* de la aplicación, destacando su fácil integración con las tecnologías web estándar como HTML, CSS y JavaScript. Se continúa con la descripción de *Web Speech API* [5] como tecnología para la síntesis de voz en un sitio web. Posteriormente, se analiza la librería gTTS [6], que ofrece capacidades de síntesis de voz utilizando las ventajas del servicio de Google. Asimismo, se examina la biblioteca latex2text [7] como una forma para la extracción de texto escrito en LaTeX a texto plano, y el uso del paquete pdfcomment [8] para agregar anotaciones o comentarios a los PDFs. Finalmente, se aborda el framework Bootstrap [9] para tener un diseño responsivo en el *front-end* de la aplicación, y su fácil compatibilidad con Flask.

1.4.2.1 Framework Flask [4]

Flask es un microframework que ha sido diseñado para facilitar el desarrollo de aplicaciones web en Python bajo el patrón MVC (Modelo-Vista-Controlador). El término micro no indica una limitación, sino que se refiere a su núcleo sólido con los servicios básicos, pero es altamente extensible porque permite la adición de paquetes según surjan las necesidades al desarrollador. Esta estructura ligera y flexible hace de *Flask* un framework ideal para proyectos de distintos tamaños.

La fortaleza de utilizar *Flask* para desarrollar una aplicación web radica en la posibilidad de utilizar la gran variedad de bibliotecas existentes en Python, ampliando así la funcionalidad de la aplicación sin tener que empezar desde cero. Esto, sumado al uso de entornos virtuales para instalar estas bibliotecas y dependencias del proyecto sin afectar al sistema global, evita conflictos de versiones con otras aplicaciones.

El desarrollo del *back-end* se realiza en un archivo con extensión ".py" donde se configuran y gestionan las rutas, se crean las instancias del servidor y se integran las diversas funcionalidades de las bibliotecas importadas en la aplicación. Flask se totalmente compatible con HTML, CSS y JavaScript, lo que permite implementar todo el *front-end* con estas tecnologías sin problemas. Además, el motor de plantillas *Jinja2* integrado en *Flask* facilita la creación de páginas HTML dinámicas, manteniendo separada la lógica de la aplicación, de la presentación.

1.4.2.2 Web Speech API [5]

Web Speech API permite la síntesis y el reconocimiento de voz en las aplicaciones web. Específicamente para la síntesis de voz, esta API permite que las aplicaciones web conviertan texto en habla, posibilitando la inclusión mensajes que se lean en voz alta cuando se realizan acciones o se navega a través de la aplicación [6]. Esta funcionalidad podría mejorar la interacción del usuario y proporcionar una experiencia más dinámica y accesible.

El uso de *Web Speech API* contribuye a la creación de aplicaciones web más inclusivas, donde se pueda facilitar un narrador para aquellos usuarios con discapacidad visual, dislexia, dificultades de lectura o para aquellos que prefieren recibir retroalimentación de forma auditiva. Proporcionar estas alternativas auditivas a la información visual, podría conseguir que las aplicaciones se vuelvan más accesibles, asegurando que más personas puedan interactuar y hacer uso de los servicios proporcionados en la aplicación.

Web Speech API permite configurar y personalizar la síntesis de voz, logrando retroalimentación auditiva acorde a las necesidades y preferencias de cada usuario. Las voces disponibles varían según el navegador utilizado y pueden personalizarse en velocidad (rate), tono (pitch) y volumen (volume). La programación de esta herramienta se realiza en *JavaScript*, lo que facilita su integración en una amplia variedad de aplicaciones web.

Mediante *Web Speech API* es posible agregar funciones de síntesis de voz en aplicaciones web, sin la necesidad de buscar e instalar bibliotecas externas. Sumado a esto la personalización de la voz podría ofrecer una experiencia adaptable y agradable para cada usuario.

1.4.2.3 gTTS (Google Text to Speech) [6]

La librería *gTTS* utiliza Python y permite la conversión de texto en audio utilizando el servicio de síntesis de voz de Google. Esta librería se conecta a la API de Google para generar archivos de audio a partir de cadenas de texto, proporcionando una manera sencilla y eficaz de integrar capacidades de síntesis de voz en aplicaciones Python. Al aprovechar la avanzada tecnología de síntesis de voz de Google, este TTS puede producir voces naturales y claras en varios idiomas.

El uso de *gTTS*, al igual que *Web Speech API*, puede mejorar la accesibilidad de recursos en internet. Un ejemplo de esto, para *gTTS*, es poder generar un audio a partir del texto extraído de un PDF, permitiendo que usuarios con discapacidades visuales o de lectura puedan comprender la información contenida en el documento. Esta funcionalidad es valiosa en contextos educativos y profesionales, porque la información generalmente está contenida en un documento digital (PDF).

La librería gTTS ofrece la posibilidad de especificar el idioma del texto a narrar, el tipo de voz (según su disponibilidad en el idioma) y la velocidad del habla. Estas opciones permiten que la síntesis de voz sea adaptable a las necesidades y preferencias de los usuarios. Además, gTTS permite guardar el audio generado en diferentes formatos, lo que facilita su uso en diversas plataformas y dispositivos. La programación de esta herramienta se realiza en *Python*, lo que la hace una herramienta de fácil integración en aplicaciones web sobre *Python*, tales como *Flask*.

1.4.2.4 Biblioteca latex2text - Simple Latex to Text Converter [7]

La biblioteca *latex2text* permite la extracción de texto escrito en lenguaje de marcado LaTeX a texto plano. Esta librería hace que el contenido escrito en LaTeX sea accesible y utilizable en contextos donde el formato LaTeX no es adecuado, o se lo requiere presentar o procesar en otros formatos.

El uso de *latex2text*, puede ser aplicado para extraer el contenido de un documento de forma fiel al escrito por el autor y no desde el formato PDF, que generalmente el orden del contenido puede variar según la disposición que mejor se adapte visualmente al espacio disponible en el PDF. Esta extracción precisa es esencial en contextos académicos donde la exactitud del texto, especialmente en artículos científicos y matemáticos, es crítica.

Uno de los retos que se deben enfrentar cuando se usa *latex2text* es la extracción de las etiquetas de referencia (*ref*) en figuras, tablas, secciones y ecuaciones, porque estas no se traducen automáticamente a sus correspondientes números, lo que puede llevar a una pérdida de contexto en el texto plano extraído. Estas referencias son ampliamente utilizadas en documentos académicos y científicos para conectar partes del contenido, y su ausencia podría implicar que el texto sea menos comprensible.

1.4.2.5 Paquete pdfcomment [8]

Pdfcomment es un paquete de LaTeX que permite agregar anotaciones y comentarios en los documentos PDF a partir de código LaTeX. Dependiendo de la etiqueta o comando utilizado del paquete, se pueden incluir anotaciones de distinto tipo que pueden ser visibles como notas emergentes, cuadros de texto, o comentarios resaltados. La visibilidad o compatibilidad de estas anotaciones varía de acuerdo con el visor de PDF utilizado. Este paquete fue desarrollado siguiendo como referencia el visor Adobe Acrobat Reader.

De las etiquetas o comandos disponibles en el paquete *pdfcomment*, se destaca aquel llamado *pdftooltip*, la cual se utiliza para agregar descripciones emergentes (*tooltips*) en un documento PDF. Un *tooltip* es un cuadro de texto que aparece cuando el cursor se coloca sobre un área específica del documento. Este puede ser útil para proporcionar información adicional, definiciones, o aclaraciones sin sobrecargar el texto principal.

La utilidad principal de *pdftooltip* en LaTeX radica en su capacidad para mantener el texto original sin cambios visuales, pero permite proporcionar información adicional solo donde sea necesario, lo que mejora la accesibilidad al ofrecer información adicional sin cambiar el contenido original del documento. Sin embargo, esta funcionalidad depende de la compatibilidad del visor PDF, ya que no todos los visores soportan *tooltips*, y además, los *tooltips* generalmente solo admiten texto plano, sin permitir la inclusión de imágenes, tablas o formatos complejos dentro del texto emergente.

1.4.2.6 Bootstrap [9]

Bootstrap es un framework de *front-end* de código abierto que facilita el desarrollo de sitios y aplicaciones web con diseños responsivos y modernos. Este framework proporciona una colección de herramientas y componentes CSS y JavaScript preconstruidos, como botones, formularios, y menús de selección, que permiten a los desarrolladores crear interfaces de usuario cohesivas y estéticamente agradables sin tener que escribir le código desde cero.

El uso de *Bootstrap* contribuye a la creación de diseños que se adaptan de forma automática a diferentes tamaños de pantalla, es decir, facilita la creación de un *front-end* con diseño responsivo. Esto ayuda al desarrollador a mantener organizado el contenido de una aplicación, independientemente del dispositivo desde el que se acceda, ya sea un smartphone o monitores de escritorio de grandes dimensiones.

Una de las ventajas de Bootstrap es la preexistencia de componentes y estilos estándar, lo cual permite al desarrollador enfocarse en la funcionalidad de su proyecto, sin limitar la personalización de los componentes de Bootstrap. Al ser un framework ampliamente adoptado, cuenta con una gran cantidad de documentación para resolver problemas y facilitar la implementación de los componentes en los proyectos.

Integrar Bootstrap en una aplicación Flask es sencillo; Flask se encarga del *back-end*, gestionando rutas, bases de datos y lógica del servidor, mientras que Bootstrap se utilizar para el *front-end*, manejando el diseño y la presentación visual [4].

1.4.3 METODOLOGÍA KANBAN [10]

La metodología Kanban es una técnica de gestión de proyecto que se originó a partir del sistema de producción de Toyota en 1940. El término Kanban se refiere a representar visualmente el progreso en las tareas de un proyecto, de tal forma de lograr gestionar el flujo de tareas y asegurar la continuidad del proyecto general. Este método ha sido ampliamente adaptado en diversas industrias más allá de la manufactura, incluyendo el desarrollo de software.

La implementación de Kanban va de la mano con la creación de un tablero, donde se ubicarán los tres tipos de tareas: pendientes, en progreso y completadas. Con esto, visualmente se puede apreciar las tareas que han sido completadas, y aquellas que requieren atención para continuar con el flujo de trabajo. El tablero puede ser físico, mediante la utilización de tarjetas y columnas en un pizarrón, o digital, a través de herramientas o aplicaciones de gestión de proyectos.

La implementación de Kanban también tiene retos que se deben afrontar. Uno de ellos es la disciplina de los involucrados en el proyecto para actualizar regularmente el tablero, y respetar los límites de las actividades en curso. Otro reto puede ser la comprensión y el compromiso que deben tener los involucrados para aplicar los principios y las prácticas de Kanban.

2 METODOLOGÍA

En este capítulo se detalla el desarrollo de la aplicación web Latex2AIP (LaTeX to Accessible Inclusive PDF), abordando aspectos clave como el Estado del Arte correspondiente a la generación de PDFs inclusivos a partir de LaTeX; luego se plantea el tablero Kanban con el avance en las actividades de cada fase; después se detallan las funcionalidades que se consideraron para el diseño de la aplicación, y se culmina con los detalles relevantes en la implementación del prototipo web.

En el Apartado 2.1 se detalla sobre la problemática que se busca combatir, y se muestran proyectos existentes que también buscan combatir esta problemática. En el Apartado 2.2 se plantea el tablero Kanban con las actividades realizadas hasta la Fase Teórica, y las que faltan por realizar en el resto de las fases. En el Apartado 2.3 se detallan los Requerimientos Funcionales y No Funcionales del prototipo web, se muestra el diagrama de Casos de Uso de la aplicación, y se plantean directrices seguidas para el diseño de la interfaz, incluyendo un "*mockup"* del prototipo. Para culminar este capítulo, en el Apartado 2.4 se actualiza el tablero Kanban con las actividades completadas en la fase de diseño, y se describen las funcionalidades más importantes implementadas en la aplicación web.

2.1 ESTADO DEL ARTE

La lectura de un PDF es una tarea sencilla para personas videntes, debido a que a más de la información visible (contenido textual), suelen contar con imágenes y figuras que aportan con información adicional al documento. En el caso de personas con discapacidad visual, no cuentan con la información adicional que se busca transmitir a través de la figura, y que podría ser muy importante para entender y comprender el contenido del PDF.

La información que suelen tener las figuras en los artículos científicos, y que puede ser accedida por una persona no vidente, es un *caption* generado por el escritor del artículo. Este *caption* suele ser una breve descripción (subtítulo) de lo que representa la figura, más no una descripción con detalles sobre el contenido visual de la figura, que suele estar contenido en lo que se conoce como texto alternativo (*alt text*).

Según el estudio realizado a más de 300 figuras en publicaciones relacionadas con la informática [11], se encontró que el texto alternativo y los subtítulos carecen de calidad descriptiva, con muchas figuras teniendo un texto alternativo poco informativo. Este estudio identifica los desafíos a los que se enfrentan los autores al elaborar *alt text* para figuras

complejas, destacando así la necesidad de mejorar de alguna forma la calidad del texto alternativo para las figuras.

Para contrarrestar los desafíos en la generación de texto alternativo para figuras, se plantea hacer uso de un *LLM*, tal y como menciona Gerd Kortemeyer en [12]. En este último artículo, se sugiere el uso de un *LLM* para generar HTML accesible a partir de código LaTeX, esto se puede extrapolar al uso de un *LLM* que permita mejorar las descripciones de las figuras e incluirlas como texto alternativo al código LaTeX, y así lograr generar un PDF inclusivo.

Generar un PDF inclusivo a partir de LaTeX no es una tarea simple, debido a que el código LaTeX no soporta la creación de formatos de documentos estructurados, específicamente "tagged PDF" requerido para hacer accesible el contenido a usuarios con discapacidad visual [13]. Debido a la falta de este soporte surgieron varios proyectos o soluciones, como los mostrados en la Tabla 2.1, con el objetivo de ayudar a las personas no videntes a acceder al contenido de artículos científicos.

Tabla 2.1. Proyectos dirigidos a ayudar a las personas no videntes a acceder al contenido de artículos científicos

Proyecto	Descripción
Latex-access [14]	Conjunto de scripts y configuraciones que mejoran la accesibilidad de los documentos LaTeX. Incluye herramientas para agregar etiquetas de accesibilidad y metadatos, y para asegurarse de que el contenido sea legible por los lectores de pantalla. Este proyecto ha sido diseñado para facilitar el uso de LaTeX para matemáticos y científicos ciegos al proporcionar una traducción en tiempo real de líneas de código LaTeX a braille. Además, traduce la línea actual al habla inglesa.
Tagpdf [15]	Paquete que permite estructurar los documentos LaTeX para mejorar la accesibilidad de los archivos PDF generados. Incluye la adición manual de etiquetas en el contenido y la estructura, que son esenciales para la navegación con lectores de pantalla.

Axessibility [16]	Paquete diseñado para mejorar la accesibilidad de las fórmulas matemáticas en archivos PDF. Permite que las fórmulas en los documentos PDF generados con LaTeX sean accesibles para tecnologías de asistencia, como lectores de pantalla. Axessibility genera automáticamente comentarios de cada formula en el PDF, estos contienen el código LaTeX original de la formula.
AGAP (Automated Generation of Accessible PDF) [17]	Extensión de ALAP (Accessible LaTeX based Authoring and Presentation) que automatiza y facilita el proceso de generación de PDF accesibles a partir de código LaTeX. ALAP es un sistema de escritura y presentación de documentos LaTeX que ha sido diseñado para ser accesible para personas con discapacidades visuales. AGAP detecta las violaciones de la accesibilidad y proporciona instrucciones sobre cómo solucionarlas en tiempo de compilación. AGAP permite la interacción mediante síntesis de voz y atajos de teclado.

El presente TIC también busca ayudar a las personas con discapacidad visual a acceder al contenido de artículos científicos, mediante el desarrollo de la aplicación Latex2AIP que genera un PDF con mejoras en las descripciones de figuras SVG y TikZ. Con Latex2AIP se logra procesar el documento LaTeX completo y se consigue generar un audio de todo su contenido, e incluso seccionado, a diferencia de [14] que reproduce en voz alta la línea actual, y por tanto, el usuario está limitado a ir navegando línea a línea por el documento, lo cual puede ser aburrido y tedioso para el usuario. Además, Latex2AIP permite cambiar el idioma del audio entre español, inglés o portugués, mientras que [14] se limita a reproducir el contenido de la línea actual en el idioma inglés. Latex2AIP modifica de forma automática el código LaTeX para incluir las etiquetas de texto alternativo en las figuras, mientras que [15] proporciona etiquetas que el usuario debe agregar manualmente si desea incluir texto alternativo. Latex2AIP consigue extraer el contenido del documento directamente desde el código LaTeX, por lo tanto no es necesario hacer que el código de las ecuaciones se incluya como texto alternativo, tal y como ocurre en [16], porque

directamente al extraer el contenido del código LaTeX se consigue extraer el código de las ecuaciones. Por otra parte, Latex2AIP permite la interacción mediante atajos de teclado y un narrador que guía la navegación por la aplicación, similar a lo que ocurre en [17], pero Latex2AIP no limita al usuario a utilizar ALAP[3] para la generación de un PDF accesible, el usuario simplemente se debe subir la carpeta con los archivos necesarios para generar el PDF y al procesar tiene acceso al archivo LaTeX etiquetado, al PDF, y a los audios de este documento.

Para el desarrollo de Latex2AIP, una aplicación web orientada a usuarios con discapacidad visual, se deben considerar los principios de diseño universal [18] de tal forma que se logre una usabilidad óptima y fácil adaptabilidad por parte del usuario. Teniendo en cuenta esto, la inclusión de un narrador, navegación por teclado y atajos, deberán permitir que el usuario navegue con facilidad a través de todas las funcionalidades de la aplicación. Por otra parte, la disposición de los elementos debe estar en un orden lógico y sin sobrecarga de opciones en pantalla, para así evitar que el usuario pierda el orden de navegación y se afecte la lógica de la aplicación [19].

Al implementar sintetizadores de voz, es posible guiar a un usuario a través de los componentes de una aplicación web, proporcionándole una retroalimentación más personalizada (en el contexto de la aplicación) de la que hubiese obtenido usando un lector de pantalla cualquiera. Por tal motivo, se tiene plantea la necesidad de incluir un narrador en la aplicación.

2.2 TABLERO KANBAN

Para llevar un orden y avance adecuado en las actividades necesarias para la implementación de este TIC, se planteó un tablero Kanban utilizando la herramienta *Notion* [20]. Para las tareas que se encuentran en el tablero mostrado en la Figura 2.1. se detalla la fase en la que se realiza dicha actividad, y el riesgo correspondiente dividido en tres niveles: *High*, *Medium*, y *Low*.

Como se observa en la Figura 2.1, hay un total de 4 actividades por realizar en la fase de diseño, y el cumplimiento adecuado con dichas actividades permitirá continuar a la fase de implementación de la aplicación web ‘Latex2AIP’.

[3] ALAP es un sistema de escritura y presentación de documentos LaTeX que ha sido diseñado para usuarios no videntes.

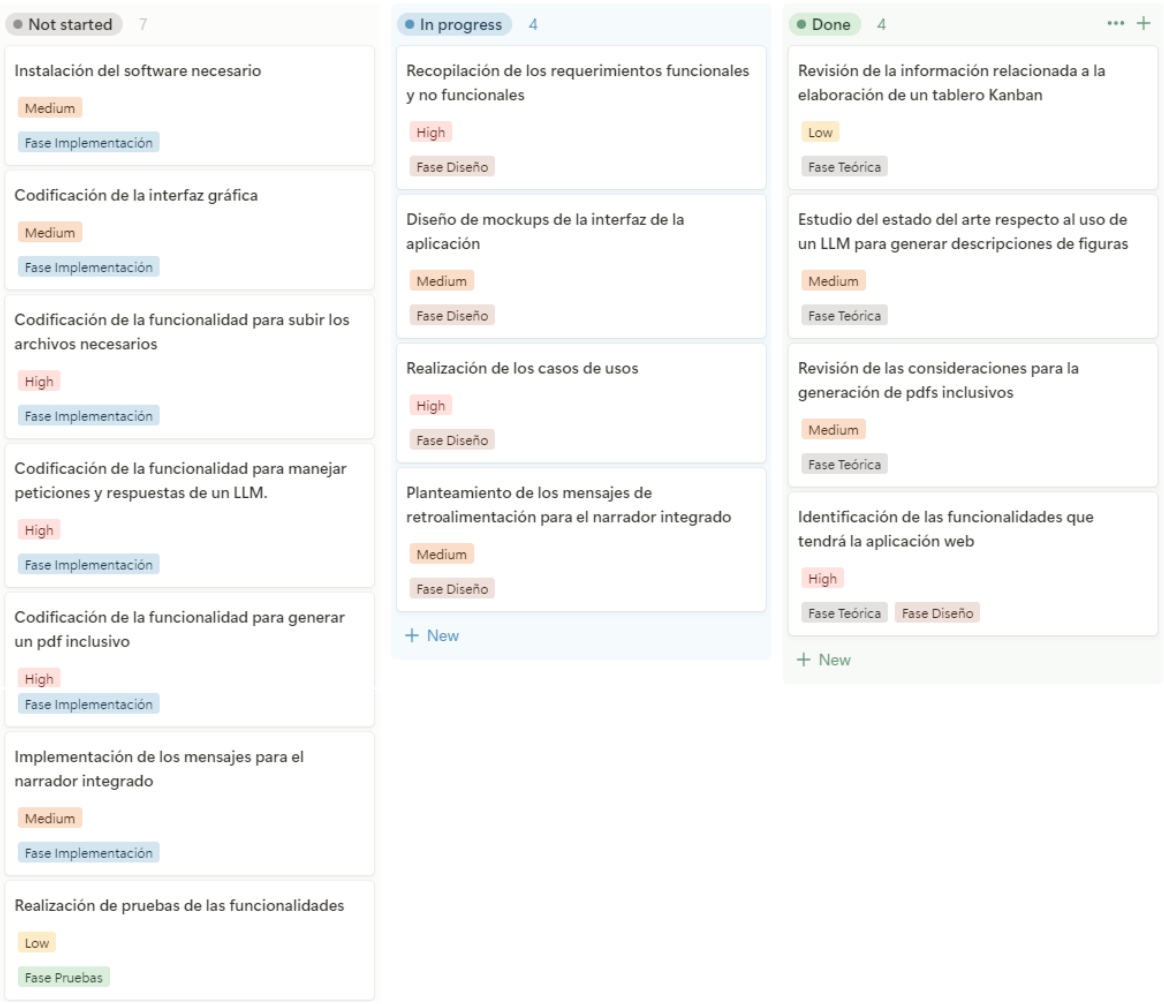

Figura 2.1. Planteamiento del Tablero Kanban durante la fase de diseño.

2.3 DISEÑO

En este apartado, se detallan los aspectos más importantes para el diseño adecuado de la aplicación web, empezando por listar los Requerimientos Funcionales y Requerimientos No Funcionales. Luego se muestra el Diagrama de Casos de Uso, y se culmina con algunas directrices para el diseño de la interfaz gráfica del prototipo.

2.3.1 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

La recopilación de requerimientos funcionales (RF) y no funcionales (RNF) es fundamental para el desarrollo correcto de un software, y que la versión final cumpla con las características planteadas en los requerimientos. En este apartado se detallan estos requerimientos divididos en funcionales y no funcionales.

2.3.1.1 Requerimientos funcionales (RF)

Los requerimientos funcionales establecen todas las características y todo lo que debe hacer el software final, de tal forma que logre cumplir con los objetivos planteados para el desarrollo de este TIC.

Los RF se detallan a continuación:

- RF1. Se debe permitir al usuario navegar por todas las funcionalidades de la aplicación mediante teclado, con su respectiva retroalimentación auditiva. Además, se debe incluir atajos de teclado que permitan al usuario saltarse entre las funcionalidades que requiera realizar.
- RF2. La aplicación debe permitir al usuario cargar una carpeta que contenga todos los archivos necesarios para generar el pdf inclusivo.
- RF3. La aplicación debe poder conectarse a un *LLM* para la obtención de las descripciones de figuras SVG y TikZ.
- RF4. La aplicación debe poder insertar, como texto alternativo o *tags*, las descripciones obtenidas desde un *LLM* en la correspondiente figura.
- RF5. La aplicación debe generar un archivo PDF que incluya las nuevas descripciones, como texto alternativo, en las figuras respectivas.
- RF6. La aplicación debe poder generar un audio, usando un TTS, de todo el contenido del documento LaTeX o de una sección específica.
- RF7. La aplicación debe permitir al usuario cambiar el idioma del audio a generar, entre español, inglés o portugués.
- RF8. Se debe permitir al usuario descargar el archivo LaTeX que incluye los tags, el PDF con las nuevas descripciones en las figuras, y el audio generado del documento o las secciones.

2.3.1.2 Requerimientos no funcionales (RNF)

Los requerimientos no funcionales establecen características que complementan los establecidos en los RF y que permiten ofrecer un software de calidad. Los RNF se detallan a continuación:

- RNF1. Se debe ofrecer retroalimentación auditiva que guie a los usuarios a través de todas las funcionalidades de la aplicación web.
- RNF2. La retroalimentación auditiva debe poder ser desactivada por los usuarios que no lo necesiten o no lo deseen, y podrán cambiar la velocidad de habla de dicha retroalimentación.
- RNF3. La retroalimentación auditiva deberá ser inmediata a medida que el usuario cambie o navegue entre las opciones de la aplicación.
- RNF4. Las preferencias en la velocidad de la retroalimentación, el idioma del audio a generar, y la necesidad o no del narrador, deberá ser almacenada mediante una cookie. Esto conseguirá que un usuario que procese varios archivos no deba volver a elegir estas configuraciones.
- RNF5. Los archivos subidos por el usuario no serán almacenados en una Base de Datos, y se descartarán del servidor una vez se complete la generación del pdf.
- RNF6. El audio generado por la aplicación comenzará a reproducirse automáticamente una vez esté listo el archivo de audio.
- RNF7. La aplicación deberá ser compatible con los navegadores webs más comunes.

2.3.2 DIAGRAMA DE CASOS DE USO DE LA APLICACIÓN

Mediante el diagrama de casos de uso se busca ejemplificar y describir el comportamiento de la aplicación desde el punto de vista del usuario o actor. Para realizar este diagrama se hizo uso de la aplicación en línea *Draw.io*.

En la Figura 2.2 es observable este diagrama, donde se definen dos actores, el usuario (que puede ser vidente o no vidente) y el *LLM*. El usuario es aquel que tendrá a su disposición todas las funcionalidades de la aplicación hasta generar un PDF inclusivo, entre estas funcionalidades está el procesamiento del documento LaTeX, la generación del audio del documento completo o por secciones, cambiar las preferencias en el narrador integrado, y descargar los archivos generados (LaTeX procesado, PDF inclusivo o audio). El *LLM* es un actor externo a la aplicación que interviene en el caso de uso “procesar documento LaTeX”, esto debido a que cuando el usuario presiona el botón para procesar el documento se hace la petición al *LLM* para generar las descripciones de las figuras SVG y TikZ.

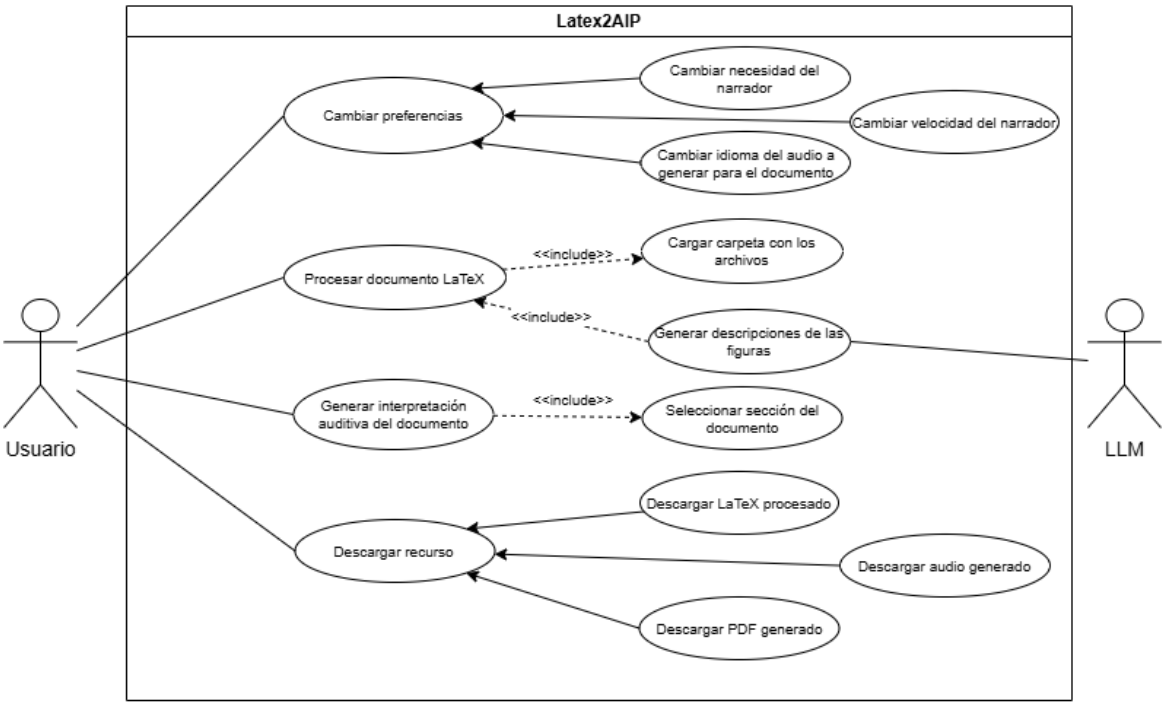

Figura 2.2. Diagrama de casos de uso de la aplicación.

Se debe recalcar que para el caso de uso ‘Cagar carpeta con los archivos’, el usuario no vidente deberá activar su lector de pantalla integrado en su PC, de tal forma que pueda navegar por su sistema de archivos hasta encontrar la carpeta que desea subir. Una vez que elija subir una carpeta, recibirá una retroalimentación auditiva indicando cuantos archivos se cargaron, o por qué no se cargaron.

2.3.3 DISEÑO DE LA INTERFAZ

Para desarrollar una aplicación inclusiva y accesible, es esencial diseñar tanto una interfaz gráfica como una interfaz no gráfica para usuarios no videntes. La interfaz no gráfica es crucial para personas con discapacidad visual, ya que les permite interactuar con la aplicación a través de la navegación por teclado y lectores de pantalla, asegurando así una experiencia de usuario fluida y efectiva. Dado que la principal prioridad es garantizar la accesibilidad y usabilidad para todos, la interfaz gráfica puede ser simplificada, enfocándose en una presentación clara y directa que no complique innecesariamente la complejidad de uso de la aplicación. Al adoptar este enfoque, se asegura que los usuarios no videntes tengan una experiencia de navegación optimizada y se evita sobrecargar la interfaz gráfica, manteniendo la aplicación accesible y fácil de usar para todos.

2.3.3.1 Diseño de interfaz gráfica

El enfoque principal en el diseño de la aplicación *Latex2AIP* es ser accesible para personas no videntes, pero también se busca que la interfaz gráfica permita a los usuarios videntes acceder a la funcionalidad principal de la aplicación, la cuál es obtener el PDF inclusivo con las descripciones mejoradas de las figuras SVG y TikZ. En el segundo caso, se podría contribuir para que los artículos científicos publicados sean en un PDF con descripciones mejoradas en las figuras.

Los sitios webs actuales utilizan muchas de las tecnologías propuestas por WCAG [4], de tal forma de garantizar que los lectores de pantalla sean admitidos en la mayoría de los navegadores. Sin embargo, el funcionamiento lineal de los lectores de pantalla presenta un desafío para que los usuarios con discapacidad visual naveguen y descubran el contenido de forma sencilla. Para abordar esta problemática, se proponen algunas directrices para desarrollar interfaces gráficas accesibles [21]:

- Utilizar etiquetas HTML semánticas para definir claramente los elementos estructurales, facilitando así la navegación mediante teclado.
- Mantener un diseño visual limpio y sin desorden, evitando incluir elementos innecesarios que compliquen la navegación por teclado del usuario.

El diseño final de la aplicación *Latex2AIP* seguirá estas directrices, de tal forma de proveer estas opciones de accesibilidad a los usuarios no videntes, y así la aplicación pueda ser usada por este grupo de usuarios.

2.3.3.2 Diseño de interfaz no gráfica

El diseño de la interfaz no gráfica se lo realiza pensando en el grupo de usuarios no videntes, de esta forma para la aplicación *Latex2AIP* se plantean los siguientes puntos que describen la utilidad de este tipo de interfaz:

- Sincronizar la retroalimentación auditiva de todas las funcionalidades de la aplicación, de esta forma se busca que las indicaciones que recibe el usuario no vidente se actualicen inmediatamente después de que realice una acción en la aplicación.
- Hay que asegurar que cada una de las funcionalidades de la página sea accesible mediante el orden lógico de tabulación, y la existencia de atajos de teclado para las funcionalidades más importantes.

- Implementar atajos de teclados fáciles de utilizar, o que sean combinaciones con teclas fácilmente identificables, como la F y J que regularmente vienen con una marca distintiva en los teclados.
- Evitar que el flujo lógico de la aplicación provea demasiadas opciones al usuario, y lograr que las opciones a realizar o configurar se muestren en un orden lineal (una tras otra).

2.3.3.3 Mockup de la aplicación web

Luego de lo mencionado en los Apartados 2.3.3.1 y 2.3.3.2, se implementarán las mencionadas consideraciones para el diseño de la interfaz gráfica y no gráfica, y así alcanzar el objetivo de que *Latex2AIP* resulte en una aplicación que pueda ser usada fácilmente por usuarios videntes y no videntes. Por otra parte, se realizó un bosquejo de la interfaz gráfica de la aplicación usando la herramienta Draw.io. En la Figura 2.3 se muestra el mockup que permitirá implementar todos los RF y RNF planteados en el Apartado 2.3.1.

Figura 2.3. Bosquejo de la interfaz gráfica de la aplicación web.

Cabe destacar que en el bosquejo de la Figura 2.3, se evidencian las etiquetas HTML semánticas que definen la estructura de las secciones de la aplicación. Con esto se busca mantener una estructura adecuada del contenido para conseguir que la navegación por teclado sea sencilla.

2.4 IMPLEMENTACIÓN

2.4.1 ACTUALIZACIÓN DEL TABLERO KANBAN

En este punto se actualizó el tablero Kanban para visualizar el avance en las actividades. Como se muestra en la Figura 2.4 todas las actividades planificadas para la fase de diseño han sido completadas, y ahora las actividades en progreso son aquellas para la fase de implementación. Para esta nueva fase se plantean 6 actividades principales, que van desde la instalación del software necesario para la aplicación, pasando por la codificación de las funcionalidades, hasta la implementación de los mensajes para el narrador.

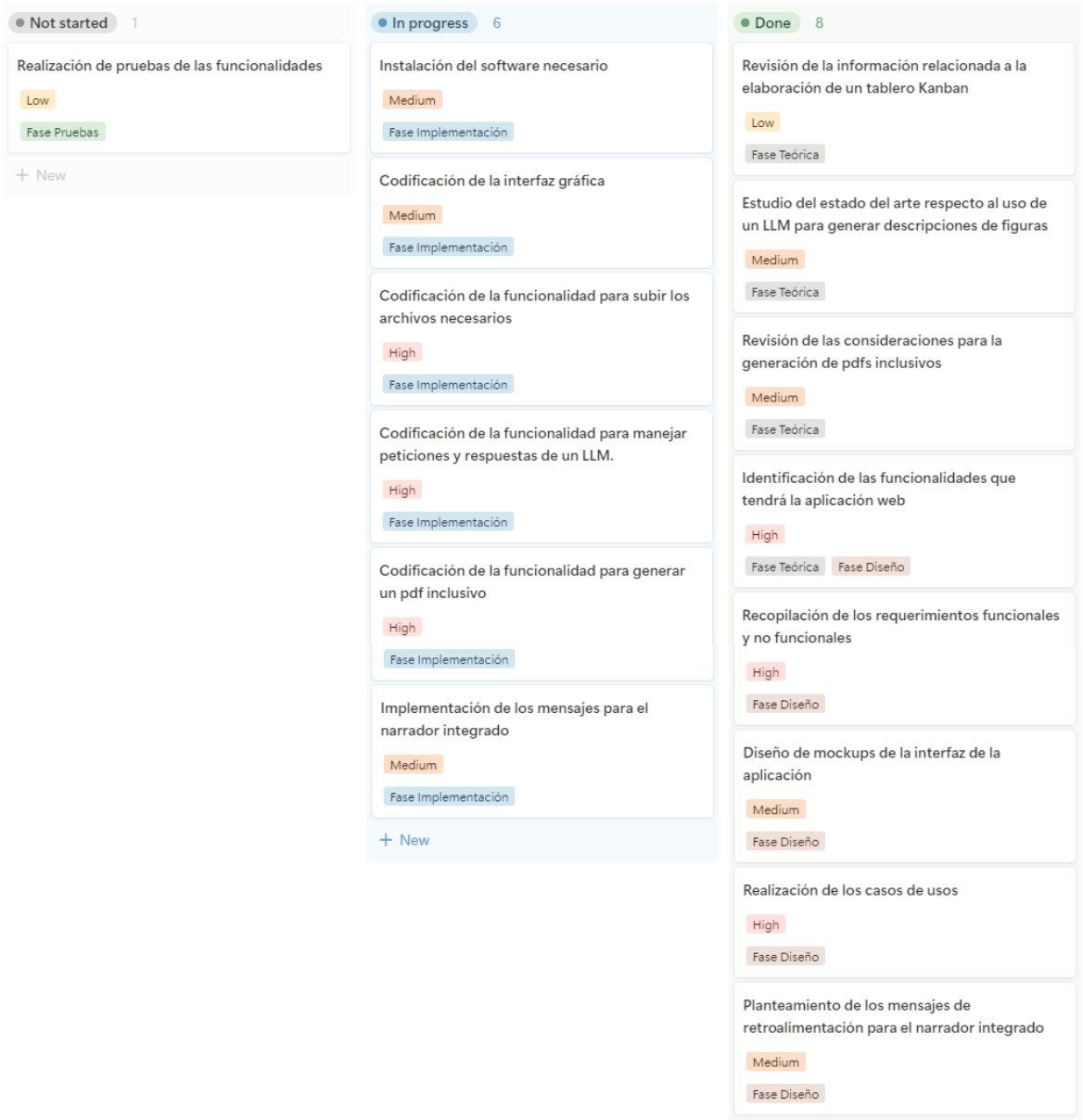

Figura 2.4. Actualización del Tablero Kanban durante la fase de implementación.

2.4.2 INSTALACIÓN DE FLASK Y CONFIGURACIÓN DEL AMBIENTE VIRTUAL

En el presente apartado se explicará a breves rasgos la configuración realizada del ambiente virtual, y la instalación del framework Flask para el desarrollo de la aplicación web en Python. Además, se explica la forma de instalación de las bibliotecas o dependencias en el ambiente virtual.

2.4.2.1 Configuración del ambiente virtual

El uso de un ambiente virtual permite instalar todas las dependencias necesarias para un proyecto y que estas se encuentren aisladas del resto de proyectos en un mismo PC. Para empezar con la creación de dicho entorno, primero se debe seleccionar una carpeta en la que se tendrá todo el proyecto. Posteriormente, ubicado en la ruta de la carpeta seleccionada se usa el Código 2.1 en la consola del equipo. Luego este comando creará el ambiente virtual llamado 'venv' y generará una carpeta con este nombre en la ruta utilizada.

```
1  python -m venv venv
```

Código 2.1: Comando para la creación del ambiente virtual.

Para empezar a trabajar en este ambiente virtual, e instalar todas las dependencias necesarias, se debe activar el entorno. Para el caso de equipos Windows, se deben usar los comandos mostrados en el Código 2.2. Luego de estos comandos, antes de la ruta del proyecto aparecerá (venv), indicando que el entorno está activado.

```
1  Set-ExecutionPolicy RemoteSigned -Scope Process
2  venv\Scripts\activate
```

Código 2.2: Comandos para la activación del ambiente virtual en Windows.

2.4.2.2 Instalación del framework Flask

Una vez que se tiene el ambiente virtual se procede con la instalación del framework Flask. Para esto, se debe ubicar en la ruta del proyecto y activar el ambiente virtual, como se mostró previamente. Posteriormente, se usa el Código 2.3 en la consola del equipo, ubicado en la ruta del ambiente virtual activado.

```
1  pip install flask
```

Código 2.3: Comando para la instalación del framework Flask.

Luego de este comando, en la ruta del proyecto se creará la carpeta src, la cual contendrá el archivo Python que manejará el *back-end* de la aplicación web. Para más detalles sobre cómo configurar las carpetas en el *back-end*, las rutas de la aplicación, la conexión con

una Base de Datos (BD) y en general, toda la configuración posible en el framework Flask, se puede consultar en [5].

2.4.2.3 Instalación de dependencias en el ambiente virtual

Como el proyecto ha sido implementado en una PC Windows, la instalación de dependencias y bibliotecas Python, generalmente se realiza mediante el comando "pip install" seguido del nombre de la dependencia que se quiera instalar. El procedimiento para la instalación de las dependencias o bibliotecas es muy similar al utilizado en el Apartado 2.4.2.2.

2.4.3 FUNCIONALIDAD PARA CARGAR LOS ARCHIVOS AL SERVIDOR

Para realizar la funcionalidad de carga de los archivos al servidor, es necesario codificar en el lado del *front-end* (funcionalidad en el cliente), y en el lado del *back-end* (manejo de los archivos en el servidor). Para el lado del *front-end* se muestra el elemento HTML que permite la carga de la carpeta desde el sistema de archivos, y para el lado del *back-end* se muestra la forma y ruta de almacenaje de los archivos contenidos en la carpeta.

2.4.3.1 Codificación en Front-End

Para permitir que el cliente suba su carpeta de archivos al servidor, se hace uso del elemento HTML *input* del tipo "*file*", configurado para que se puedan enviar múltiples archivos (*multiple*) contenidos en directorios o carpetas (*webkitdirectory*) del sistema de ficheros del usuario. El estilo de este elemento es "*form-control*" el cual adopta uno definido en Bootstrap. En el Código 2.4 se muestra el elemento HTML previamente descrito.

```
1  <input class="form-control" type="file" name="folder" id="filesInput"
   webkitdirectory multiple />
```

Código 2.4: Elemento HTML para subir una carpeta.

Este elemento se encuentra dentro del formulario que se envía como petición *post* al servidor en la ruta "*/process_file*", de tal forma que su contenido (los archivos de la carpeta subida por el usuario) se enviarán en dicha petición.

2.4.3.2 Codificación en Back-End

Para almacenar los archivos en el servidor se ha elegido la ruta 'src/static/files'. En el Código 2.5 se muestra la parte de la función "process_files()" que define el comportamiento de la aplicación a las peticiones *post* en la ruta "*/process_file*". En la línea 4 se obtiene el listado de los archivos contenidos en la carpeta que sube el usuario desde el front-end. De la línea 10 hasta 13, se recorre el listado de archivos subidos, se establece el nombre que

originalmente tenía, se define la ruta específica a cada archivo según su nombre, y se guarda el archivo en dicha ruta.

```
1 @app.route('/process_file', methods=['POST'])
2 def process_file():
3     if 'folder' in request.files:
4         folder = request.files.getlist('folder')
5 
6         # Ruta en el servidor donde se guardan los archivos
7         ruta_destino = os.path.join('src', 'static', 'files')
8 
9         # Copiar los archivos de la carpeta subida a la ruta destino
10        for archivo in folder:
11            nombre_archivo =
  secure_filename(quitar_prefijo_carpeta(archivo.filename))
12            ruta_destino_archivo = os.path.join(ruta_destino,
  nombre_archivo)
13            archivo.save(ruta_destino_archivo)
```

Código 2.5: Manejo de los archivos enviados por el cliente.

2.4.4 FUNCIONALIDAD PARA MANEJAR LA PETICIÓN Y RESPUESTA DEL *LLM*

El *back-end* es el encargado de realizar la petición y de adaptar la respuesta del *LLM* a las funciones de la aplicación. Para manejar las peticiones se creó una función llamada "*getLLMresponse()*", y para manipular la respuesta del *LLM* se creó la función llamada "*tagTexFile()*". Para el presente TIC, el *LLM* utilizado para obtener las descripciones mejoradas de las figuras fue desarrollado por el Sr. Mateo Viteri en su TIC llamado "DESARROLLO DE UN SERVICIO PARA GENERAR DESCRIPCIONES DE FIGURAS PRESENTES EN ARCHIVOS LATEX MEDIANTE EL USO DEL PROCESAMIENTO TEXTUAL DE UN MODELO DE LENGUAJE LARGO".

Del *LLM* elegido se conoce que está a la espera de un archivo zip que contenga como mínimo el código .tex y los demás archivos necesarios para generar el documento. Esta *LLM* detectará las figuras SVG y TikZ, procesará el contexto del documento y el código de las figuras para empezar a generar la descripción mejorada. Como respuesta se obtiene un *JavaScript Object Notation* (json), con la descripción original de la figura, el *label*, la descripción mejorada, entre otra información.

En el Código 2.6 se encuentra la función "*getLLMresponse()*" que se encarga de hacer la petición al *LLM* enviando el archivo zip 'peticion.zip' que se creó en la función "*process_file()*", la cual maneja los archivos enviados por el cliente. En la línea 4 se hace la petición *post* al *LLM*, hacia la url establecida y enviando el archivo zip definido.

```
def getLLMresponse():
    url = 'http://172.19.0.2:8001/upload'
    files = {'file': open('src/static/files/peticion.zip', 'rb')}
    response = requests.post(url, files=files)
    return response
```

Código 2.6: Manejo de la petición al LLM.

En el Código 2.7 se encuentra una parte de la función "*tagTexFile()*", esta se encarga de llamar a la función "*getLLMresponse()*", y luego gestiona la respuesta dada por el *LLM*. De la línea 5 a 14 se valida la obtención de un archivo *json* válido, con las descripciones de las figuras. En la línea 17 se crea un diccionario con el *label* que identifica la figura y su descripción final generada. A partir del diccionario se procesa el código LaTeX para incluir como texto alternativo las descripciones mejoradas en las figuras respectivas, esa parte del código, así como el código completo se encuentra en el **Anexo I**.

```
def tagTexFile(input_text):
    response = getLLMresponse()

    # Verifica que la respuesta sea exitosa
    if response.status_code == 200:
        try:
            # Intenta interpretar la respuesta como JSON
            json_response = response.json()
        except ValueError:
            json_response = []
            print("La respuesta no es un JSON válido")
    else:
        json_response = []
        print(f"Error en la solicitud: {response.status_code}")

    # Crear un diccionario para acceder rápidamente a la descripción basada en el label
    figure_descriptions = {fig['label']: fig['description_final'].replace('\n', ' ') for fig in json_response}
```

Código 2.7: Manejo de la respuesta del LLM.

2.4.4.1 Etiquetado de las figuras

El código LaTeX se procesa recorriendo línea a línea en busca de alguna coincidencia con algún *label* del diccionario previamente mencionado en el Código 2.7. Cuando se detecta una coincidencia se busca el *caption* correspondiente a dicha figura, y se incluye la etiqueta '*\pdftooltip*' al texto del *caption*. De esta forma se incluye el texto alternativo en el *caption* de la figura. En la Figura 2.5 se muestra el cambio, que realiza automáticamente la aplicación, en el código LaTeX para lograr etiquetar una figura.

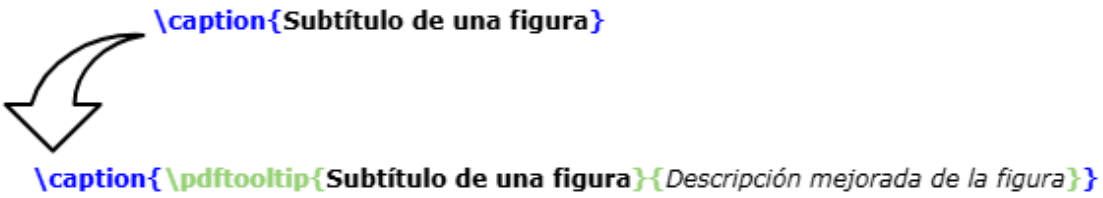

Figura 2.5. Ejemplo de etiquetado de una figura.

La Figura 2.5 indica la estructura de la etiqueta '*caption*' en color azul, y el cambio que se produce luego de incluir la etiqueta '*pdftooltip*' para agregar la descripción mejorada como texto alternativo. El texto en negrita corresponde a la descripción o subtítulo visible que tienen las figuras, y el texto en cursiva corresponde al texto alternativo; en el caso particular de este TIC, el texto alternativo será la descripción mejorada de las figuras, que podrá ser accedido mediante un lector de PDF compatible con accesibilidad. De forma visual es posible apreciar este texto alternativo abriendo el PDF en Adobe Acrobat Reader, y ubicando el cursor sobre el subtítulo de la figura con el texto alternativo. En la Figura 2.6 se aprecia esta característica mencionada.

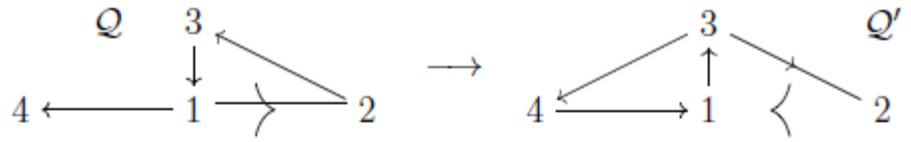

Figure 1: Mutación Quiver en el vértice 1.

La figura muestra una mutación del quiver en el vértice 1, que transforma el quiver de la izquierda en el quiver de la derecha. La mutación del quiver es una transformación de un quiver que cambia la orientación de las flechas en un único vértice. En este caso, el vértice 1 es el vertice mutado. El quiver de la izquierda, denominado Q, tiene nodos denominados 1, 2, 3 y 4 y flechas entre ellos que representan el quiver original. El quiver de la derecha, denominado Q prima, tiene los mismos nodos y flechas que el primer quiver, pero con la orientación de las flechas del vértice 1 invertida. Además, se añade una nueva flecha entre los nodos 3 y 4. La mutación del quiver se indica mediante una flecha que apunta del quiver de la izquierda al quiver de la derecha. La figura sirve como ejemplo de una mutación de quiver, que es un concepto fundamental en la teoría de álgebras de clústers. Las mutaciones de quiver se utilizan para transformar un quiver en otro cambiando la orientación de determinadas flechas. Este proceso puede aplicarse repetidamente, dando lugar a una secuencia de quivers y mutaciones que pueden utilizarse para estudiar las propiedades de las álgebras de clústers.

Figura 2.6. Ejemplo del texto alternativo en una figura.

Hacer que la descripción mejorada de las figuras sea texto alternativo garantiza que el documento PDF no se altere visualmente en nada con respecto al original (generado del código LaTeX sin las etiquetas en las figuras). De esta forma se genera un PDF inclusivo para personas no videntes, o para aquellos que necesiten esta descripción más detallada; dando origen a una herramienta útil que facilita el acceso a la información a este grupo históricamente ignorado y olvidado.

2.4.5 FUNCIONALIDAD PARA GENERAR EL PDF INCLUSIVO

La función que maneja la respuesta del *LLM* "*tagTexFile()*" genera un *string* como salida, el cual corresponde al código LaTeX etiquetado con las descripciones mejoradas de las figuras SVG y TiKz. Este *string* se recibe como parámetro de entrada de la función "*compileLatexToPdf()*", la cual se encarga de generar el archivo PDF. En el Código 2.8 se muestra la función previamente mencionada, donde se puede notar que:

- En la línea 3 y 4 se crea un archivo LaTeX llamado '*taggedLatex.tex*' con el parámetro de entrada, el cual corresponde al código LaTeX etiquetado.
- De la línea 7 a 22 se ejecutan los comandos en consola necesarios para compilar el archivo LaTeX y generar correctamente el PDF.
- De la línea 25 a 30 se cambia el nombre del PDF generado a '*taggedPDF.pdf*' y se lo ubica en un lugar accesible para que esté disponible su envío al cliente, o permitir que el usuario lo descargue sin problemas.

```
1  def compileLatexToPdf(tex_content):
2      # Crear un archivo .tex con el código Latex etiquetado
3      with open('src/static/files/taggedLatex.tex', 'w', encoding="utf-
   8") as f:
4          f.write(tex_content)
5
6      # Compilar el archivo LaTeX con pdflatex y bibtex
7      try:
8          # Primera compilación con pdflatex
9          subprocess.run(['pdflatex', '-interaction=nonstopmode', '-
   shell-escape', 'taggedLatex.tex'])
10
11         # Ejecutar bibtex si hay un archivo .aux
12         if os.path.isfile('taggedLatex.aux'):
13             subprocess.run(['bibtex', 'taggedLatex'])
14
15         # Segunda compilación con pdflatex
16         subprocess.run(['pdflatex', '-interaction=nonstopmode', '-
   shell-escape', 'taggedLatex.tex'])
17
18         # Tercera compilación con pdflatex
```

```
        run_command(['pdflatex', '-interaction=nonstopmode', '-shell-escape', 'taggedLatex.tex'])

    except subprocess.CalledProcessError as e:
        print("Error al ejecutar pdflatex o bibtex:", e)

    # Mover el PDF generado a un lugar accesible
    try:
        os.replace('src/static/files/taggedLatex.pdf', 'src/static/files/taggedPDF.pdf')
    except OSError as e:
        print("Error al mover el archivo PDF:", e)
        return None
    return 'src/static/files/taggedPDF.pdf'
```

Código 2.8: Generación del PDF a partir del código LaTeX etiquetado.

2.4.6 FUNCIONALIDAD PARA IMPLEMENTAR LOS MENSAJES DEL NARRADOR INTEGRADO

Para implementar el narrador integrado, la aplicación cuenta con una función llamada "*alertVoice()*", esta recibe como parámetro de entrada el mensaje que se leerá en voz alta con el narrador. La función primero verifica si el narrador está activado, luego cancela cualquier otro mensaje que esté siendo leído, posteriormente sobre el objeto SpeechSynthesisUtterance se configura el mensaje y la personalización de la rapidez, tono, volumen y voz (hombre o mujer). Después de configurar y personalizar el objeto *SpeechSynthesisUtterance*, se usa la función "*window.speechSynthesis.speak()*" para hacer que el mensaje se lea en voz alta. Todo lo anteriormente descrito se puede encontrar en el Código 2.9.

```
function alertVoice(message) {
    if (conSonidos === 1) {
        // Cancelar cualquier mensaje de voz en curso
        window.speechSynthesis.cancel();

        // Crear un nuevo objeto SpeechSynthesisUtterance
        var utterance = new SpeechSynthesisUtterance(message);

        // Configurar la velocidad de lectura (rango de 0.1 a 10)
        utterance.rate = rapidez;

        // Configurar el tono de la voz (rango de 0.1 a 2)
        utterance.pitch = 0.2;

        // Configurar el volumen de la voz (rango de 0 a 1)
        utterance.volume = 1.0;

        // Obtener la voz por defecto del navegador
        var voices = window.speechSynthesis.getVoices();
        utterance.voice = voices[0]; // Establecer la voz
```

```
21
22          // Lanzar la síntesis de voz y se empieza a escuchar el
  mensaje en voz alta.
23          window.speechSynthesis.speak(utterance);
24      }
25  }
```

Código 2.9: Función para implementar mensajes narrados en voz alta.

Esta función está implementada en JavaScript, y es ejecutada por prácticamente todos los elementos en el *front-end*, los cuales definen el mensaje que se debe leer. Generalmente el mensaje corresponde a las instrucciones que guían a un usuario no vidente a través de la interfaz. Sin embargo, esta función también se usa para dar las instrucciones iniciales en la aplicación, y para el mensaje que detalla todos los atajos de teclado disponibles.

2.4.7 CODIFICACIÓN DE LA INTERFAZ

De acuerdo con lo que se definió en la Etapa de Diseño, y lo descrito en el Apartado 2.3.3, hay dos enfoques aplicados para la codificación de la interfaz. El primero es netamente lo visual de la interfaz gráfica, definiendo un diseño visual limpio y ordenado. El segundo se enfoca en brindar las facilidades de navegación y la retroalimentación auditiva necesaria para los usuarios no videntes.

2.4.7.1 Interfaz gráfica

Partiendo del Mockup presentado en el Apartado 2.3.3.3, se implementaron todas las funcionalidades de la aplicación. Se estructuraron todos los elementos o etiquetas HTML de forma ordenada, de tal forma que se consigue un diseño que facilita la navegación mediante teclado (tabulador). Se utilizó Hojas de Estilos en Cascada (CSS) para mejorar la disposición visual de los elementos y así lograr una interfaz gráfica limpia y ordenada. Para perfeccionar el aspecto visual de los elementos y conseguir un diseño responsivo y moderno, se utilizaron los diseños de Bootstrap, para el *switch*, *selects*, *input* y *buttons*, Luego de todo lo previamente mencionado, el resultado de la interfaz gráfica se muestra en la Figura 2.7.

En la Figura 2.8 se divide en secciones la interfaz gráfica y las funcionalidades de la aplicación. A continuación, se explica a más detalle las secciones y las funcionalidades que hay en cada una:

- La sección de preferencias del usuario incluye: activar o desactivar el narrador, selección de la velocidad del narrador (normal, rápida, muy rápida), selección del idioma del PDF (español, inglés, portugués). Cuando el usuario recarga la aplicación estas preferencias conservan el estado seleccionado.

- La sección de archivos y procesamiento es la más importante porque permite al usuario cargar su carpeta con los archivos correspondientes, automáticamente se muestra el código LaTeX que será etiquetado, y permite al usuario procesar su documento. El procesamiento incluye el etiquetado del código LaTeX y la generación del PDF inclusivo. En esta misma sección se cargará el código LaTeX etiquetado, y en la sección del PDF se mostrará el PDF inclusivo.

- La sección de audios permite al usuario seleccionar la sección del documento que quiere generar y escuchar en audio, puede cambiar su velocidad de reproducción y descargar el audio generado. Las secciones del documento se detectan automáticamente por la aplicación, e incluye la opción de todo el contenido o las figuras con descripciones mejoradas. En caso de que el usuario haya seleccionado mal el idioma del PDF no necesita volver a procesar el archivo, simplemente debe cambiar la opción en la sección de preferencias y presionar en generar audio.

- La sección de descarga incluye los botones para descargar el archivo LaTeX etiquetado y el PDF inclusivo. También se tiene un botón para procesar otro archivo desde el principio.

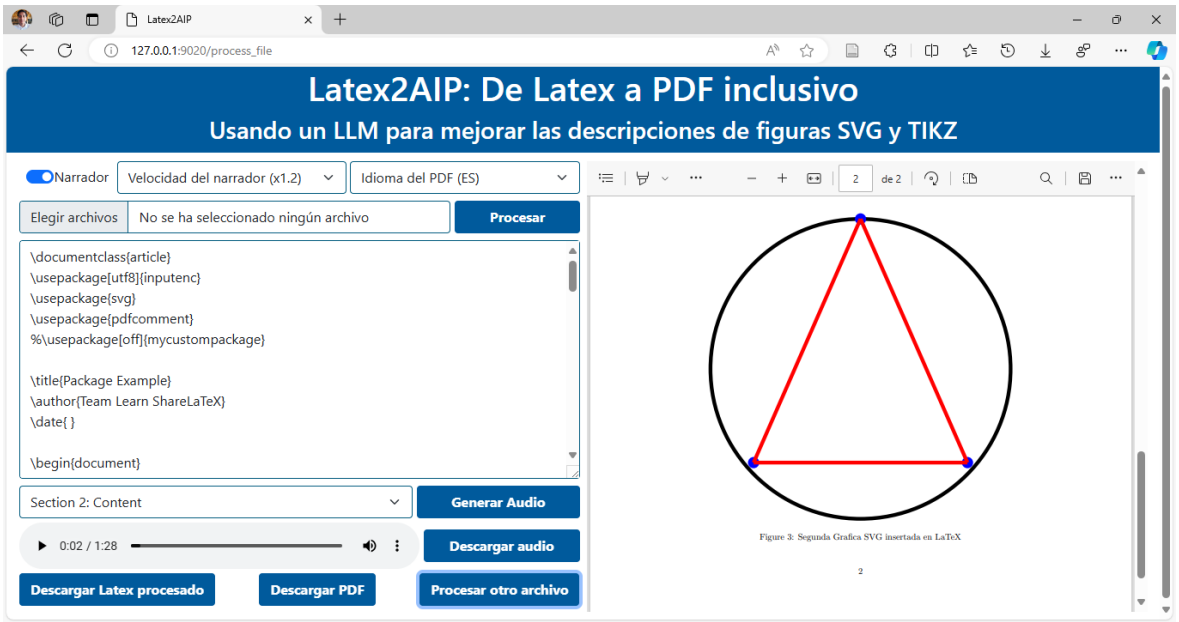

Figura 2.7. Interfaz gráfica de Latex2AIP.

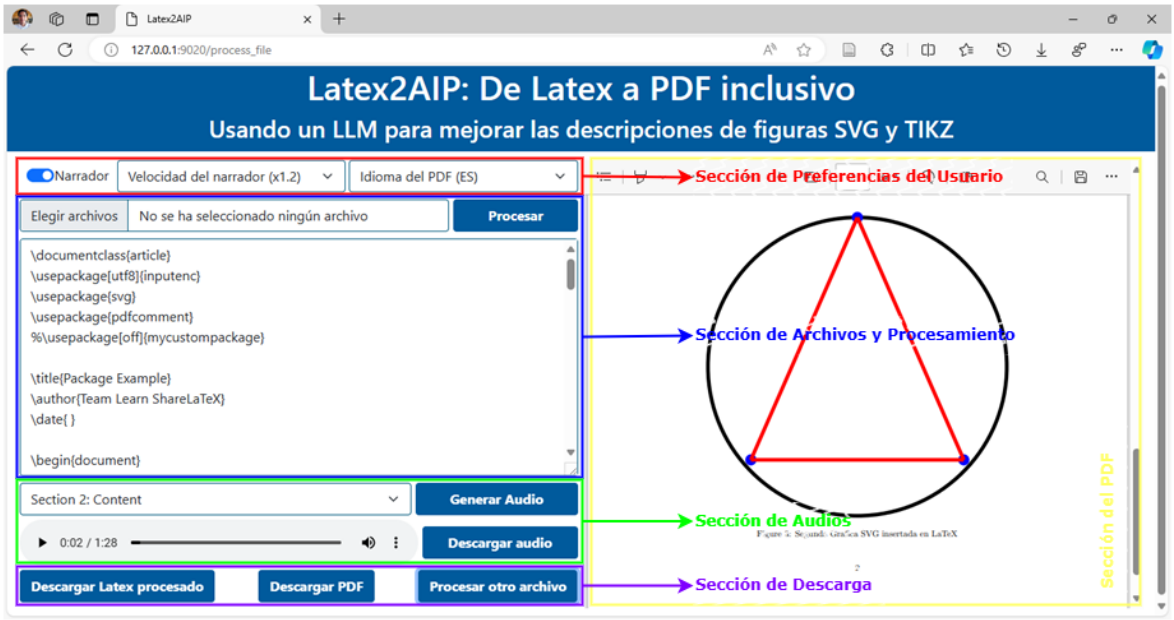

Figura 2.8. Interfaz gráfica seccionada de Latex2AIP.

2.4.7.2 Interfaz no gráfica

En esta interfaz no gráfica se incluyen las funciones que facilitan la navegación del usuario no vidente. A continuación se detallan estas funciones:

- Mediante lo expuesto en el Apartado 2.4.6 se logra proporcionar una retroalimentación auditiva de la navegación por la aplicación, logrando que sea de forma inmediata cuando el usuario realice una acción o interactúe en Latex2AIP.
- Se configuró la navegación por la aplicación mediante el tabulador de forma secuencial y lógica, se probará su efectividad en la Fase de Pruebas.
- Se implementaron atajos de teclado para que el usuario pueda navegar directamente a la opción que desee modificar. El listado de atajos disponibles se detalla a continuación:
    - Shift + s para activar o desactivar narrador.
    - Shift + d para elegir la velocidad del narrador.
    - Shift + f para subir la carpeta de archivos.
    - Shift + g para elegir el idioma del PDF.
    - Shift + h para silenciar el mensaje actualmente narrado.
    - Shift + j para procesar los archivos cargados.
    - Shift + k para escuchar el listado de atajos del teclado.
    - Shift + l para empezar de nuevo a elegir los archivos.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan los resultados más relevantes del presente TIC, abarcando aspectos clave como la definición del entorno de pruebas y la validación de los Requerimientos Funcionales y No Funcionales. Posteriormente, se exponen las conclusiones del proyecto, seguidas de recomendaciones para aquellos interesados en realizar un proyecto similar, y se concluye con consideraciones para trabajos futuros.

En el Apartado 3.1, se describen en detalle los resultados relevantes, incluyendo la actualización del tablero Kanban, la definición del entorno de pruebas, el diseño y discusión de la encuesta, los resultados de las pruebas, y la validación de los Requerimientos Funcionales y No Funcionales. En el Apartado 3.2, se presentan las Conclusiones derivadas de la realización de este TIC. El Apartado 3.3 ofrece Recomendaciones útiles para quienes deseen emprender un proyecto similar y se aborda consideraciones para el trabajo futuro, con el objetivo de mejorar el prototipo de la aplicación web.

3.1 RESULTADOS Y DISCUSIÓN

3.1.1 ACTUALIZACIÓN DEL TABLERO KANBAN

En este punto, como se observa en la Figura 3.1, todas las actividades vinculadas a la fase de implementación han sido completadas, y solo se encuentra “en progreso” la actividad relacionada a la realización de Pruebas. Cabe destacar que no hay actividades “por hacer” en el tablero Kanban.

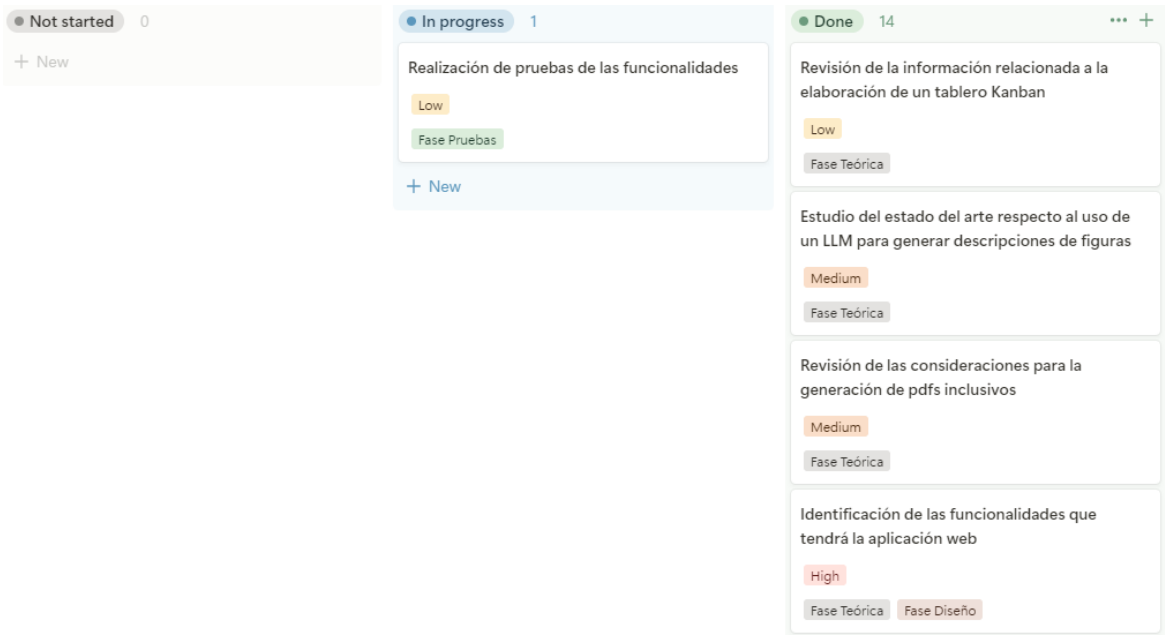

Figura 3.1. Actualización del tablero Kanban en la Fase de Resultados

3.1.2 DEFINICIÓN DE ENTORNO DE PRUEBAS

Para verificar el correcto funcionamiento del prototipo de aplicación web denominado Latex2AIP, se establecieron dos escenarios de pruebas controlados con el objetivo de evaluar el cumplimiento de los Requerimientos Funcionales y No Funcionales, así como de identificar errores en la aplicación. El primer escenario, denominado *"Alpha Testing",* involucró a cinco usuarios sin discapacidad visual, quienes usaron vendas de tela para simular la falta de visión. El segundo escenario, denominado *"Beta Testing",* contó con la participación de cinco usuarios con discapacidad visual real, contactados a través de la fundación 'Asociación de Invidentes Milton Vedado'. En cada escenario de pruebas se consideraron los siguientes puntos:

- Antes de que los usuarios realizaran las pruebas en la aplicación, recibieron una capacitación sobre las funcionalidades disponibles y las instrucciones generales para la navegación en la aplicación.
- Las pruebas se realizaron de forma presencial, utilizando el computador en el cual se desarrolló la aplicación.
- Al finalizar las pruebas en el prototipo, se solicitó a los usuarios que proporcionaran retroalimentación sobre la navegación y los recursos de salida obtenidos (audios y PDFs).
- El *"Alpha Testing"* se llevó a cabo antes del *"Beta Testing",* lo que permitió corregir algunos problemas en el prototipo antes de iniciar el segundo escenario de pruebas.
- Para verificar la mejora en las descripciones de las figuras SVG y TikZ, se realizó una encuesta en la que los usuarios pudieron escuchar las nuevas descripciones generadas para seis figuras y decidir si estas habían mejorado en comparación con las descripciones originales. Para más detalles sobre la encuesta, diríjase al apartado 3.1.3 Diseño de la Encuesta.

Para el escenario *"Beta Testing"* se tienen las siguientes consideraciones adicionales:

- Los usuarios considerados para las pruebas tienen distintos niveles de discapacidad visual: 2 participantes con Baja Visión, 1 participante con Visión Parcial y 2 participantes con Ceguera Total.

- Los usuarios utilizaron audífonos de casco al navegar por la aplicación, de modo que podían escuchar perfectamente la retroalimentación auditiva incluida en la aplicación y el audio generado para el PDF.
- Las opiniones de los usuarios de este escenario de pruebas sobre las nuevas descripciones para las figuras incluidas en el PDF se anotaron como retroalimentación adicional a la encuesta sobre la mejoría de las descripciones. Para más detalles sobre la esta retroalimentación adicional, diríjase al apartado 3.1.4 Discusión de la Encuesta.

3.1.3 DISEÑO DE LA ENCUESTA

Para comprobar la mejoría en las descripciones de las figuras SVG y TikZ se realizó una encuesta en línea que constaba de lo siguiente:

- Un video para cada una de las 6 figuras, donde se podía apreciar la imagen y un audio generado con la tecnología utilizada en Latex2AIP. Este audio se adaptó para que se escuche en primera instancia la descripción original, y posteriormente la descripción generada por un LLM. Se trabajó en idioma inglés para 3 figuras y el resto en idioma español.
- Un contenido textual para cada figura, donde se detallaba lo mencionado a través del audio en el video, es decir, se tenía la descripción original y la descripción generada.
- Dos opciones para elegir, mejoró o no mejoró la descripción generada respecto a la descripción original de la figura.

Es importante mencionar que mediante esta encuesta no se buscaba determinar el nivel de mejoría en las descripciones de las figuras porque este TIC se enfoca en generar un PDF que incluya mejores descripciones en las figuras SVG y TikZ, el nivel de mejoría en las descripciones puede variar mucho dependiendo del LLM utilizado para generarlas. Este TIC busca sentar bases para la generación de un PDF inclusivo en lo que respecta a mejorar las descripciones de figuras SVG y TikZ, y con el avance de la tecnología y de la Inteligencia Artificial sería posible mejorar el nivel de las descripciones y ampliar su alcance a otros formatos de figuras.

3.1.4 DISCUSIÓN DE LA ENCUESTA

En el Anexo II, se encuentran los resultados detallados para cada figura, y a modo de análisis se tiene lo siguiente:

- Indiscutiblemente la descripción generada mejoró la descripción original para cada figura. Estos resultados reafirman lo mencionado en el Apartado 2.1 Estado del Arte, donde se describen los inconvenientes del *caption* o subtítulo de la figura, dado que se limita a lo que considere el autor relevante mencionar para dicha figura, y no a otorgar una descripción detallada del contenido visual de esta.
- El nivel de mejoría no se determinó a través de la encuesta, pero es evidente que había descripciones fácilmente entendibles, y otras más complejas de entender, incluso para personas videntes. Por lo cual, resalta la importancia de elegir adecuadamente un LLM que genere mejores descripciones de las figuras.

Los usuarios no videntes, pertenecientes al segundo escenario de pruebas, dieron sus opiniones sobre las descripciones de las figuras disponibles en la encuesta. En el Anexo III se adjunta el formulario de consentimiento firmado por la vicepresidenta de la fundación 'Asociación de Invidentes Milton Vedado', que permite compartir los resultados extraídos de las pruebas con los integrantes de la fundación (sin revelar información personal). A continuación, se detallan 2 de las opiniones más destacadas:

- "La descripción generada es excelente, (tanto) por la descripción de los detalles visuales y la contextualización que brinda sobre la figura".
- "A pesar de no conocer el contexto del documento en el que estaba la figura, (la descripción generada) es de gran ayuda".

3.1.5 RESULTADOS DEL ENTORNO DE PRUEBAS

Del primer escenario de pruebas *"Alpha Testing"* se obtuvieron varias observaciones, las cuales se solventaron previo a realizar el segundo escenario de pruebas. En la Tabla 3.1 se detalla la observación dada por los usuarios y la corrección implementada en la aplicación.

Tabla 3.1. Observaciones obtenidas en el escenario *"Alpha Testing",* y su corrección.

Observación	Corrección
Mejorar la fluidez entre los mensajes de retroalimentación de la aplicación.	Se cambió la forma en la que los mensajes de retroalimentación son leídos, de tal forma que cuando se navega entre las opciones de la aplicación, el mensaje de retroalimentación se lee inmediatamente, y no se manda a la cola de mensajes.

Incluir una forma de conocer los atajos de teclado disponibles en la aplicación.	Se incluyó en el mensaje de bienvenida de la aplicación la información sobre el atajo de teclado que permite escuchar el listado total de los atajos disponibles.
Permitir interrumpir el mensaje de retroalimentación actual.	Se incluyó un atajo de teclado para cancelar el mensaje que se esté leyendo en voz alta actualmente.

Una vez solventadas las observaciones del primer escenario de pruebas, se procedió a realizar el segundo escenario de pruebas *"Beta Testing"* donde se obtuvieron nuevas observaciones o mejoras a implementar en la aplicación. En la Tabla 3.2 se detalla la observación dada por los usuarios y la corrección implementada en la aplicación.

Tabla 3.2. Observaciones obtenidas en el escenario *"Beta Testing",* y su corrección.

Observación	Corrección
Mejorar la rapidez de los mensajes de retroalimentación del narrador integrado.	Se cambió la velocidad normal (por defecto) del narrador a x1.2, y se incluyó la opción para cambiar el nivel a Rápida (x1.5) y Muy Rápida (x2).
Incluir un mensaje de retroalimentación cuando se presiona una de las teclas configuradas para navegar o interactuar con la aplicación.	Se establecieron nuevos mensajes de retroalimentación (leídos en voz alta) para cuando se presionan las teclas que permiten al usuario navegar o interactuar. Por ejemplo, cuando se presiona la tecla Enter para procesar la carpeta, se escucha el mensaje "Enter".
Incluir la forma de repetir el último mensaje de retroalimentación dado por el narrador.	Se almacena en una variable el último mensaje leído por el narrador integrado, y se incluyó un atajo de teclado para repetir este último mensaje.

3.1.3 VALIDACIÓN DE REQUERIMIENTOS FUNCIONALES

Para validar los Requerimientos Funcionales se tomó en cuenta la retroalimentación bridada por los usuarios, de ambos escenarios de pruebas, luego de finalizar las pruebas al prototipo. En la Tabla 3.3 se encuentra el estado y las observaciones de los RF.

Tabla 3.3. Validación de los RF.

ID	Detalles	Estado	Observación
RF1	Navegación completa por la aplicación mediante teclado con retroalimentación auditiva y atajos para funcionalidades específicas	Validado	Se estableció un orden concreto de navegación, mediante el tabulador, por todas las opciones de la aplicación. También se incluyeron atajos de teclado para cada funcionalidad.
RF2	Carga de carpetas con archivos necesarios para generar un PDF inclusivo	Validado	Se restringió que el sistema de archivos muestre solamente carpetas para cargar, y se permite la carga siempre y cuando haya un archivo .tex.
RF3	Conexión a un *LLM* para obtener descripciones de figuras SVG y TikZ.	Validado	Se utilizó un *LLM* que recibe un archivo .zip con los archivos subidos y responde con las descripciones de todas las figuras SVG y TikZ que se encuentran en el documento LaTeX.
RF4	Inserción de descripciones obtenidas de LLM como texto alternativo o *tags* en las figuras.	Validado	Se incluyen las descripciones como un *tag* en el *caption* de la figura. Mediante Adobe Acrobat Reader se puede apreciar visualmente el contenido del *tag*.
RF5	Generación de un PDF con nuevas descripciones como texto alternativo en las figuras.	Validado	Se muestra visualmente el PDF generado en la aplicación. Se incluyó una opción para descargar dicho PDF.
RF6	Generación de audio del contenido del documento LaTeX o de secciones específicas mediante TTS.	Validado	Se permite seleccionar la sección del documento para generar el audio, incluyendo

			una opción para todo el contenido del documento.
RF7	Opción de cambiar el idioma del audio generado entre español, inglés o portugués	Validado	Se permite cambiar el idioma del audio, antes o después de cargar y procesar los archivos.
RF8	Descarga del archivo LaTeX con tags, PDF con descripciones nuevas y el audio del documento o secciones.	Validado	Se incluyeron botones para descargar cada uno de los archivos generados. Es posible acceder a ellos mediante atajos de teclado.

3.1.4 VALIDACIÓN DE REQUERIMIENTOS NO FUNCIONALES

Para validar los Requerimientos No Funcionales también se tomó en cuenta la retroalimentación bridada por los usuarios, de ambos escenarios de pruebas, luego de finalizar las pruebas al prototipo. En la Tabla 3.4 se encuentra el estado y las observaciones de los RNF.

Tabla 3.4. Validación de los RNF.

ID	Detalles	Estado	Observación
RNF1	Retroalimentación auditiva para guiar a los usuarios en todas las funcionalidades de la aplicación web.	Validado	Se utilizó Web Speech API para proporcionar mensajes en voz alta en todos los elementos y componentes de la aplicación.
RNF2	Opción para desactivar la retroalimentación auditiva y ajustar la velocidad de habla.	Validado	Se incluyó un Switch para activar o desactivar los mensajes narrados, y se habilitaron 3 velocidades para el narrador (normal, media y alta).
RNF3	Retroalimentación auditiva inmediata durante la navegación por la aplicación.	Validado	Cada nuevo mensaje por narrar interrumpe el mensaje que se esté narrando actualmente.
RNF4	Almacenamiento de preferencias de velocidad de retroalimentación, idioma de	Validado	Se utilizaron *'sessions'* para el almacenamiento de estas preferencias de usuario, y se

	audio y necesidad de narrador mediante cookies.		mantienen incluso cuando el usuario actualice la página.
RNF5	No se almacenarán archivos subidos en una base de datos; se descartarán tras la generación del PDF.	Validado	Mediante el uso de comandos, se automatizó la eliminación de los archivos subidos por el usuario luego de generar el PDF.
RNF6	Reproducción automática del audio generado.	Validado	Se activó la opción *"autoplay"* para lograr que se reproduzca el audio una vez se genere correctamente el archivo.
RNF7	Compatibilidad con los navegadores web más comunes.	Validado	Se revisó que los elementos usados en la creación de la aplicación sean compatibles con los navegadores web actuales.

3.1.5 CIERRE DEL TABLERO KANBAN

Una vez que se completaron las pruebas de funcionamiento, y se corrigieron los errores detectados, el tablero Kanban quedó como se muestra en la Figura 3.2. En esta figura se evidencia que se completaron las 15 tareas principales planificadas para el desarrollo de este TIC.

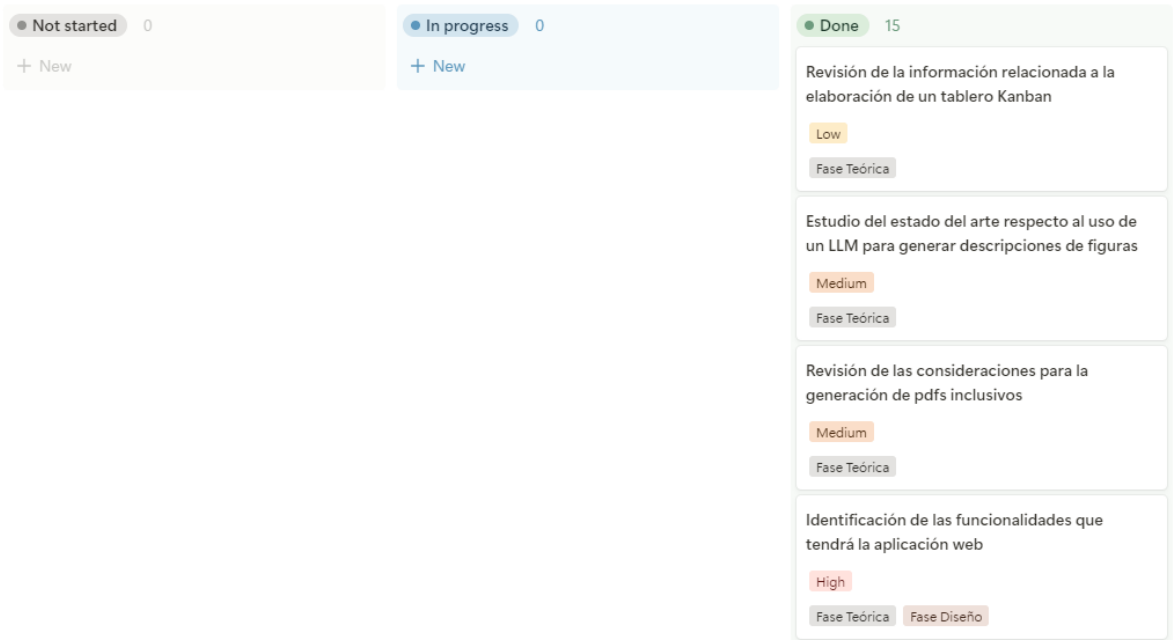

Figura 3.2. Cierre del tablero Kanban.

3.2 CONCLUSIONES

La aplicación web planteada en este TIC fue desarrollada mediante 4 fases fundamentales: Teórica, Diseño, Implementación y Pruebas; y en base de este proceso se desprenden las siguientes conclusiones:

- Al finalizar el presente TIC, se logró desarrollar una aplicación web para la generación de PDFs inclusivos, mejorando las descripciones de figuras SVG y TikZ a partir de código LaTeX mediante el uso de un LLM. Con esta aplicación, se ha alcanzado un hito significativo en la promoción de la accesibilidad digital, proporcionando una herramienta que mejora el acceso a la información técnica y científica, contenida en figuras, para personas con discapacidad visual. Proporcionar un PDF con descripciones mejoradas de figuras empodera a los usuarios no videntes, permitiéndoles participar de manera más plena y equitativa en entornos educativos y profesionales. Esta iniciativa promueve una mayor autonomía y desarrollo profesional para personas con discapacidad visual, disminuyendo brechas digitales y contribuyendo a una sociedad más justa y accesible para todos.

- A lo largo del proyecto, se llevó a cabo un análisis exhaustivo de diversas tecnologías, seleccionando aquellas que mejor se ajustaban a los requerimientos del proyecto. Se optó por utilizar Flask para el desarrollo del back-end de la aplicación, debido a su fácil integración con tecnologías web estándar. Se escogió gTTS para garantizar la accesibilidad auditiva de los documentos generados. Además, se incorporó la librería latex2text para la extracción de texto LaTeX a texto plano y el framework Bootstrap para lograr un diseño responsivo en el front-end. Este análisis detallado y la elección estratégica de herramientas y tecnologías fueron fundamentales para sentar las bases del desarrollo exitoso de la aplicación, asegurando su funcionalidad y accesibilidad.

- Un paso muy importante para la generación de un PDF inclusivo fue diseñar el flujo de trabajo para agregar etiquetas de texto alternativo al código LaTeX dentro de la aplicación, específicamente para las figuras SVG y TikZ. Durante el desarrollo, se estableció un proceso eficiente donde el usuario carga una carpeta con los archivos necesarios, y el back-end realiza una solicitud al LLM para obtener nuevas descripciones mejoradas de las figuras. Estas descripciones se integran automáticamente en el archivo .tex generado por la aplicación, utilizando la etiqueta /pdftooltip dentro del caption correspondiente a cada figura. Este archivo LaTeX

procesado se utiliza luego para generar el PDF inclusivo, asegurando que las figuras estén mejor descritas para facilitar su comprensión por parte de personas con discapacidad visual.

- En este proyecto se cumplió satisfactoriamente con el desarrollo e implementación de una aplicación web para la generación de PDF inclusivos. Desde el diseño inicial, donde se establecieron claramente los Requerimientos Funcionales y No Funcionales, hasta la implementación práctica utilizando Flask para el back-end y *Bootstrap* para el *front-end*, se ha asegurado que la aplicación cumpla con estándares de accesibilidad y funcionalidad. Siguiendo la metodología Kanban, el proceso de diseño incluyó la elaboración detallada de un Diagrama de Casos de Uso y directrices para la interfaz gráfica, lo cual guió la implementación efectiva de las funcionalidades clave, como la integración de descripciones mejoradas de figuras SVG y TikZ mediante el uso de un LLM y la generación automática de etiquetas de texto alternativo en documentos LaTeX, asegurando así la creación de documentos PDF accesibles y de alta calidad.

- Mediante la realización exitosa de escenarios de pruebas controlados se pudo verificar el funcionamiento del prototipo de la aplicación web Latex2AIP. Estos escenarios, denominados "Alpha Testing" y "Beta Testing", involucraron tanto a usuarios simulando la falta de visión como a usuarios con discapacidad visual, quienes proporcionaron valiosa retroalimentación sobre la navegación y la usabilidad de la aplicación. Los resultados de las pruebas demostraron que tanto los Requerimientos Funcionales como No Funcionales fueron validados correctamente, asegurando que la aplicación cumpliera con estándares de accesibilidad y funcionalidad. Además, la encuesta realizada confirmó la mejora significativa en las descripciones de figuras SVG y TikZ generadas por la aplicación, validando así el éxito del proyecto en proporcionar una herramienta efectiva para la creación de PDFs inclusivos y accesibles.

3.3 RECOMENDACIONES

A lo largo del desarrollo de este TIC se presentaron desafíos e inconvenientes que fueron resueltos, en base a esto nacen las siguientes recomendaciones:

- Para el desarrollo de la funcionalidad que permite generar el audio del contenido del documento, es recomendable escoger aquel *TTS* que tenga las mejores

opciones de personalización, de tal forma que se pueda permitir al usuario generar el audio según sus gustos personales o sus necesidades. Por esta razón, el momento de escoger un *TTS* es crucial para determinar todas las opciones de personalización que se deben implementar en la aplicación web, tales como: cambiar el idioma, la velocidad, el tono, las palabras por minutos, etc.

- Se recomienda seleccionar un *LLM* que tenga una capacidad robusta para interpretar y describir figuras con precisión, lo cual es crucial para generar PDFs inclusivos con descripciones de alta calidad. Además, debido al rápido avance de la tecnología, es esencial actualizar regularmente el *LLM* utilizado para asegurarse de que las descripciones de las figuras se mantengan precisas y relevantes, mejorando así continuamente la accesibilidad y la inclusión digital de los documentos generados.

- Al momento de escribir el código del *front-end* de la aplicación web, es recomendable estructurar adecuadamente el código HTML para asegurar que la ubicación visual de los elementos facilite la navegación correcta a través de las opciones de la aplicación web. Una estructura bien organizada del HTML no solo mejora la accesibilidad para usuarios con discapacidad visual, que podrían depender de lectores de pantalla, sino que también proporciona una experiencia de usuario más intuitiva y eficiente para todos los usuarios.

- Se recomienda realizar un análisis exhaustivo de las limitaciones de la biblioteca latex2text en la extracción de texto, ya que presenta problemas específicos, como el manejo inadecuado de la etiqueta \href, lo que ha requerido el desarrollo de funciones adicionales de preprocesamiento del código LaTeX. Además de este problema, podrían existir otras limitaciones con comandos personalizados y entornos complejos. Por lo tanto, es aconsejable buscar una biblioteca alternativa más robusta y precisa para reducir la necesidad de preprocesamiento manual y mejorar la eficiencia y calidad de la conversión de LaTeX a texto plano, facilitando así el desarrollo de documentos inclusivos y accesibles.

- Para aquellos que son nuevos en el uso de Flask, es recomendable usar el libro *"Flask Web Development"* de Miguel Grinberg. Este libro es una guía completa y práctica para aprender y desarrollar aplicaciones web en Python utilizando Flask. *"Flask Web Development"* cubre desde los conceptos básicos hasta temas avanzados, incluyendo la configuración de entornos de desarrollo, la creación y manejo de bases de datos, la autenticación de usuarios, y el despliegue de

aplicaciones. La claridad y la profundidad con las que se abordan los temas hacen de este libro una herramienta invaluable para principiantes y desarrolladores intermedios que deseen construir aplicaciones web robustas y escalables con Flask.

Gracias a la experiencia adquirida durante el desarrollo de la aplicación, se derivan las siguientes consideraciones para el trabajo futuro con el objetivo de mejorar el prototipo:

- Se podría desarrollar un método que traduzca las ecuaciones en formato LaTeX a un lenguaje natural, facilitando así la comprensión del contenido auditivo generado por la aplicación. Esto permitiría que los usuarios sin conocimientos previos de LaTeX puedan acceder a la información sin dificultad, mejorando significativamente la accesibilidad de la aplicación.
- Se podría diseñar un sistema alternativo para la carga de archivos en la aplicación, eliminando la necesidad de utilizar un lector de pantalla externo. En su lugar, se podría fortalecer al narrador integrado en la aplicación web para que guíe a los usuarios a través del nuevo proceso de carga de archivos. Esto no solo simplificaría el uso de la aplicación, sino que también ofrecería una experiencia de usuario más fluida y coherente.
- Actualmente, la aplicación permite la generación de audios por secciones. En futuras mejoras, se podría implementar un seccionamiento más detallado, permitiendo la generación de audios por subsecciones, gráficos, ecuaciones, y otros elementos específicos. Este nivel de detalle permitiría a los usuarios acceder de manera más precisa y eficiente a la información que necesitan, mejorando así la usabilidad y funcionalidad de la aplicación.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] «¿Qué son los modelos LLM? | Grandes modelos de lenguaje». Accedido: 23 de julio de 2024. [En línea]. Disponible en: https://www.cloudflare.com/es-es/learning/ai/what-is-large-language-model/

[2] «An introduction to tagged PDF files: internals and the challenges of accessibility». Accedido: 23 de julio de 2024. [En línea]. Disponible en: https://www.overleaf.com/learn/latex/An_introduction_to_tagged_PDF_files%3A_internals_and_the_challenges_of_accessibility

[3] W. W. A. Initiative (WAI), «Introducción a la Accesibilidad Web», Web Accessibility Initiative (WAI). Accedido: 23 de julio de 2024. [En línea]. Disponible en: https://www.w3.org/WAI/fundamentals/accessibility-intro/es

[4] M. Grinberg, *Flask web development*, First edition. Sebastopol, CA: O'Reilly, 2014.

[5] «Uso de la Web Speech API - Referencia de la API Web | MDN». Accedido: 23 de mayo de 2024. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API

[6] «gTTS — gTTS documentation». Accedido: 23 de julio de 2024. [En línea]. Disponible en: https://gtts.readthedocs.io/en/latest/index.html

[7] «latex2text — Simple Latex to Text Converter — pylatexenc 3.0alpha000021 documentation». Accedido: 23 de julio de 2024. [En línea]. Disponible en: https://pylatexenc.readthedocs.io/en/latest/latex2text/

[8] J. Kleber, «A user-friendly interface to PDF annotations».

[9] M. O. contributors Jacob Thornton, and Bootstrap, «About». Accedido: 23 de julio de 2024. [En línea]. Disponible en: https://getbootstrap.com/docs/5.3/about/overview/

[10] Universidad Nacional de Colombia, M. D. Arango Serna, L. F. Campuzano Zapata, Ingeniero de Producción. Rymel S.A.S., J. A. Zapata Cortes, y Universidad Nacional de Colombia, «Mejoramiento de procesos de manufactura utilizando Kanban», *Rev. Ing. Univ. Medellín*, vol. 14, n.º 27, pp. 221-234, 2015, doi: 10.22395/rium.v14n27a13.

[11] C. Williams, L. De Greef, E. Harris, L. Findlater, A. Pavel, y C. Bennett, «Toward supporting quality alt text in computing publications», en *Proceedings of the 19th International Web for All Conference*, Lyon France: ACM, abr. 2022, pp. 1-12. doi: 10.1145/3493612.3520449.

[12] G. Kortemeyer, «Using artificial-intelligence tools to make LaTeX content accessible to blind readers», 18 de junio de 2023, *arXiv*: arXiv:2306.02480. Accedido: 11 de octubre de 2023. [En línea]. Disponible en: http://arxiv.org/abs/2306.02480

[13] F. Mittelbach y C. Rowley, «LaTeX Tagged PDF—A blueprint for a large project», *TUGboat*, vol. 41, n.º 3, pp. 292-298, 2020, doi: 10.47397/tb/41-3/tb129mitt-tagpdf.

[14] «latex-access, providing easy on-the-fly braille and speech access to LaTeX documents». Accedido: 17 de junio de 2024. [En línea]. Disponible en: https://latex-access.sourceforge.net/

[15] «CTAN: Package tagpdf». Accedido: 17 de junio de 2024. [En línea]. Disponible en: https://www.ctan.org/pkg/tagpdf

[16] D. Ahmetovic *et al.*, «Axessibility: a LaTeX Package for Mathematical Formulae Accessibility in PDF Documents», en *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*, Galway Ireland: ACM, oct. 2018, pp. 352-354. doi: 10.1145/3234695.3241029.

[17] S. Zulfiqar, S. Arooj, U. Hayat, S. Shahid, y A. Karim, «Automated Generation of Accessible PDF», *22nd Int. ACM SIGACCESS Conf. Comput. Access.*, pp. 1-3, oct. 2020, doi: 10.1145/3373625.3418045.

[18] N. Menzi-Çetin, E. Alemdağ, H. Tüzün, y M. Yıldız, «Evaluation of a university website's usability for visually impaired students», *Univers. Access Inf. Soc.*, vol. 16, n.º 1, pp. 151-160, mar. 2017, doi: 10.1007/s10209-015-0430-3.

[19] E. R. de Souza y S. Freitas, «Dosvox Usability: recommendations for improving interaction of blind people with the web using the system», *Work*, vol. 41, pp. 3443-3448, ene. 2012, doi: 10.3233/WOR-2012-0622-3443.

[20] «Gestión obras y reformas: Control total con plantilla Notion». Accedido: 24 de junio de 2024. [En línea]. Disponible en: https://ovacen.com/gestion-obras-reformas-proyectos-notion/

[21] J. Bargas-Avila y K. Opwis, «Beyond web content accessibility guidelines: Design of enhanced text user interfaces for blind internet users», *Int. J. Hum.-Comput. Stud.*, vol. 66, pp. 257-270, abr. 2008, doi: 10.1016/j.ijhcs.2007.10.006.

5 ANEXOS

Luego de completar el desarrollo y escritura del presente TIC, se adjunta la información complementaria pulsado el siguiente enlace:

https://epnecuador-my.sharepoint.com/:f:/g/personal/carlos_leon01_epn_edu_ec/Eta08K-Z6HlNkw-mLFmhkNMBuVg_gjfsmFidXH2w-uWqww?e=dZUfW6

La estructura de los anexos es la siguiente:

ANEXO I. Código del prototipo de aplicación web *Latex2AIP.*

ANEXO II. Resultados de la encuesta.

ANEXO III. Formulario de aceptación de divulgación de resultados de las pruebas.