Benemérita Universidad Autónoma de Puebla

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

“APLICACIÓN WEB PARA LA ENSEÑANZA Y PRÁCTICA DEL INGLÉS MEDIANTE LA INTERACCIÓN CON HISTORIAS CORTAS”

TESIS PRESENTADA PARA OBTENER EL GRADO DE LICENCIATURA EN INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

Jesús Alejandro Serrano Graciano

DIRECTORA DE TESIS:

Dra. Meliza Contreras González

PUEBLA, PUE. NOVIEMBRE 2023

AGRADECIMIENTOS

Primeramente, agradezco a Dios por estar presente en mi vida,

Por colocarme en una familia cuyo apoyo es incondicional,

A mi padre y madre por ser mi mejor escuela y permitirme estudiar,

A mis hermanos: Gustavo y Belén, por su confianza en mí,

A mi ejemplar asesora de Tesis Meliza por su inspiración,

Y a mis maestros por su motivación

PREFACIO

La elaboración de la presente tesis surgió del interés personal de usar la tecnología como una herramienta para ayudar a mejorar en inglés. Durante el transcurso de mi carrera en la licenciatura en Ingeniería en Ciencias de la Computación, siempre mostré un especial interés en la programación, me gustó la programación web porque puede correr en diferentes dispositivos y sistemas operativos, como mi objetivo es llegar a toda la gente que sea posible considero que viene perfecto para este proyecto. De esta manera, el principal objetivo de esta investigación consistió en desarrollar una aplicación web con la que, diversos usuarios pudiesen ser capaces de mejorar sus habilidades de inglés mediante la interacción con historias cortas, para hacer esto más atractivo al usuario implementé tecnologías modernas como sistemas de recomendación para recomendar diferentes historias a diferentes usuarios según sus posibles gustos y módulos en tiempo real para mejorar la experiencia del mismo.

Para el desarrollo de la aplicación fue importante analizar otras similares e identificar en qué aspectos se pueden mejorar o implementar nuevos según sea el caso, todo esto tomando en cuenta el nivel de satisfacción de los usuarios.

El software se llevó a cabo mediante una metodología ágil basada en iteraciones, donde en cada iteración incluye una nueva funcionalidad del sistema.

Para la realización de la aplicación se utilizó Visual Studio Code, el cual contiene diversas extensiones para el manejo de lenguajes como HTML, JavaScript y Python entre otros.

También se utilizaron herramientas de programación como JQuery, Bootstrap 5, Django y MySQL.

CONTENIDO

Contenido

ÍNDICE DE FIGURAS

GLOSARIO Y ACRÓNIMOS

A

Algoritmo	Serie de reglas o instrucciones que permiten llegar a un resultado siguiendo una secuencia de pasos.
AJAX	Asynchronous JavaScript and XML, conjunto de técnicas que permiten que las aplicaciones web funcionen de forma asíncrona, enviando y recuperando datos del servidor sin necesidad de recargar toda la página.

C

Clave foránea	Columna o conjunto de columnas en una tabla cuyos valores corresponden a los valores de la clave primaria de otra tabla.
Clave primaria	Columna o conjunto de columnas en una tabla cuyos valores identifican de forma exclusiva una fila de la tabla.
Cliente	Ordenador o software que accede a un servidor y recupera información de este, generalmente siempre es el navegador.
Código abierto	Software mantenido y desarrollado mediante una colaboración abierta, generalmente no tiene costo.

D

Diseño responsivo	Tipo de desarrollo web que permite la óptima visualización en distintos tipos de dispositivos.

J

JQuery	Biblioteca de JavaScript con diversas características que permiten el control de eventos, manipulación de HTML e interactuar con Ajax de una manera simple.

Q

Query params	Query parameters, son parámetros de tipo clave-valor al final de una URL usados para proporcionar información adicional.

S

Servidor	Programa informático que ofrece un servicio especial a otros programas denominados clientes.

U

URL	Uniform Resource Locator, dirección dada a un recurso único en la web.

INTRODUCCIÓN

Importancia del aprendizaje de idiomas en la actualidad

Actualmente en el mercado laboral es muy importante hablar un segundo idioma, en el caso de ciencias de la computación y en países como México, muchas empresas buscan gente capacitada con habilidades lingüísticas en inglés por diversos factores. Por ejemplo, empresas extranjeras que necesitan un lenguaje en común para comunicarse internamente u otras que buscan satisfacer a los clientes que se comuniquen en este inglés. Hablar un segundo idioma se ha convertido en una habilidad esencial en el mundo laboral moderno, y el inglés es uno de los idiomas más demandados en muchas industrias.

Aplicaciones como herramientas de aprendizaje

Utilizar las bondades de la computación con el objetivo de crear una aplicación para el aprendizaje no es un tema nuevo, la tecnología siempre ha intentado mejorar la vida del ser humano, y el ámbito educativo no es la excepción. En la actualidad, existen diversas aplicaciones móviles y web que se han convertido en herramientas de aprendizaje muy populares, gracias a su facilidad de uso y su capacidad de adaptarse a diferentes estilos de aprendizaje.

Estas aplicaciones pueden abarcar diferentes áreas del conocimiento, desde el aprendizaje de idiomas, pasando por las ciencias, las matemáticas, la historia, entre otros. Lo interesante de estas herramientas es que suelen presentar la información de una manera interactiva, dinámica y visualmente atractiva, lo que facilita la retención de la información por parte del estudiante.

Además, muchas de estas aplicaciones cuentan con sistemas de seguimiento y evaluación, que permiten al usuario conocer su progreso y desempeño en tiempo real. De esta manera, el estudiante puede identificar sus fortalezas y debilidades, y trabajar en ellas para mejorar su aprendizaje.

Otro aspecto importante de estas herramientas es su accesibilidad. En muchos casos, estas aplicaciones pueden ser utilizadas desde cualquier dispositivo con acceso a internet, lo que permite a los estudiantes acceder a ellas en cualquier momento y lugar, sin necesidad de estar en un aula o tener un horario establecido.

Las aplicaciones como herramientas de aprendizaje son una opción cada vez más popular y efectiva para el estudio y la adquisición de conocimientos. Su uso puede mejorar el rendimiento académico de los estudiantes, gracias a su interactividad, su capacidad de seguimiento y evaluación, y su accesibilidad.

Ámbito

Podemos encontrar diversas aplicaciones en sitios como Google PlayStore o App Store, sin embargo, los principales problemas son la compatibilidad entre aplicaciones pues en ocasiones algunas pueden ser mejor o peor dependiendo de la plataforma que se utilice.

La intención de la aplicación es brindar acceso a dos principales tipos de usuario, esto es, el usuario estándar y el administrador, siendo el primero destinado a estudiantes y el segundo a profesores, dicho esto, los profesores pueden agregar contenido para que los estudiantes practiquen y evalúen sus habilidades.

Descripción del problema

El manejo de una lengua extranjera, especialmente el inglés se ha vuelto imprescindible para abrirse paso en muchas industrias debido a la gran cantidad de personas que hablan esta lengua. En el área de la computación el inglés es un lenguaje indispensable, basta con leer cualquier documentación de lenguajes de programación, frameworks o bibliotecas para darse cuenta de que todo está escrito en inglés, sin embargo, según diversas fuentes, incluyendo Papora, AmazingTalker y El Financiero, solo el 5% de la población en México habla inglés (Papora, 2022; AmazingTalker,

2022.; El Financiero, 2021). Este problema puede ser causado por diversos factores sociales o económicos.

Algunos de los principales problemas que surgen en el aprendizaje de un idioma son las cuotas de inscripción muy altas, horarios poco flexibles, falta de acceso a la educación, cursos demasiado difíciles de seguir y falta de interés. Por esta razón los aprendices del idioma deben invertir considerablemente en cuotas de inscripción, transportes y tiempo lo que puede orillarlos a desertar de sus estudios.

La elección de desarrollar la aplicación en formato web se basó en la necesidad de ofrecer una solución accesible y disponible para un gran número de usuarios. En la actualidad, la mayoría de la población cuenta con un smartphone y acceso a internet, lo que permite utilizar una aplicación web desde casi cualquier navegador. Además, este formato permite ofrecer una experiencia de usuario homogénea, independientemente del sistema operativo utilizado en el dispositivo del usuario.

Conscientes de los desafíos que enfrentan quienes buscan mejorar su nivel de inglés, y de la importancia de este idioma en el mundo de la tecnología, surge la idea de crear una aplicación web que pueda ayudar a las personas a aprender inglés de una manera más efectiva. La aplicación utiliza la lectura de historias cortas seleccionadas por técnicas de inteligencia artificial, de manera gratuita, para hacer que el aprendizaje sea más agradable y accesible para todos.

Objetivo general

Proporcionar una aplicación web interactiva para el aprendizaje de inglés, basada en historias cortas, que mantenga el interés del usuario y mejore sus habilidades lingüísticas, a través del uso de inteligencia artificial para personalizar el contenido y adaptarlo a las necesidades individuales de cada usuario.

Objetivos específicos

- Diseñar una interfaz de usuario intuitiva y amigable

- Desarrollar un sistema de recomendación personalizado que sugiera historias cortas adaptadas al contenido de estas y a las preferencias del usuario.
- Diseñar y aplicar estrategias de evaluación y retroalimentación efectivas para medir el progreso del usuario y ofrecer sugerencias de mejora.
- Desarrollar y aplicar ejercicios interactivos que utilicen herramientas de reconocimiento de voz para mejorar la pronunciación y ejercicios de corrección ortográfica para fortalecer las habilidades de escritura del usuario.

Metodología de desarrollo

Para alcanzar los objetivos establecidos, se llevaron a cabo las siguientes actividades en el diseño del sistema:

1. Investigación de aplicaciones existentes: Se realizó un estudio exhaustivo de aplicaciones similares con el objetivo de identificar las mejores prácticas y características que satisfacen las necesidades y preferencias de los usuarios.
2. Selección de la metodología de desarrollo de software: Se llevó a cabo una revisión de diversas metodologías de desarrollo de software con el fin de elegir la más adecuada para el proyecto y sus necesidades.
3. Evaluación de frameworks: Se investigaron y evaluaron diferentes frameworks, considerando aspectos como el conocimiento previo en materias relacionadas y el dominio del lenguaje de programación.
4. Proceso de desarrollo iterativo: Se adoptó un enfoque iterativo en la realización de la aplicación, que consistió en las etapas de investigación, planificación, desarrollo y pruebas.

La finalización de este proyecto ofrece una herramienta de aprendizaje accesible para una amplia gama de personas que dispongan de un dispositivo con acceso a internet. Además, esta aplicación también puede ser aprovechada por profesores y educadores del idioma, brindándoles un instrumento adicional para enriquecer sus clases y mejorar la experiencia de enseñanza.

CAPÍTULO I

ESTADO DEL ARTE

1.1 Estado del Arte

Gracias a los avances tecnológicos, el mundo de la computación ofrece diferentes soluciones a diversos desafíos. Entre ellos, un desafío común es el aprendizaje de idiomas pues ya sea por necesidad o interés personal, aprender un segundo idioma es una tarea que puede llegar a complicarse por diversos factores como la falta de disponibilidad, los recursos económicos o el nivel de motivación.

Es por lo anterior que resulta de gran relevancia tratar los últimos avances tecnológicos en aplicaciones, respecto al aprendizaje de idiomas sean aplicaciones web o móviles.

Dicho lo anterior, resulta crucial abordar los últimos avances tecnológicos en el campo de las aplicaciones de aprendizaje de idiomas, ya sean estas web o móviles. Estas innovadoras herramientas tecnológicas han revolucionado la forma en que las personas pueden acceder y adquirir conocimientos en idiomas extranjeros.

1.1.1 Cake

Cake es una aplicación móvil ideada para el aprendizaje de inglés y coreano que se encuentra disponible en Google Play Store y App Store, su principal característica es la forma en que alienta la pronunciación, pues su método consiste en mostrar contenido audiovisual al usuario con el objetivo de hacer que este repita una frase proveniente del material. La figura 1.1.1.1 ilustra su funcionamiento.

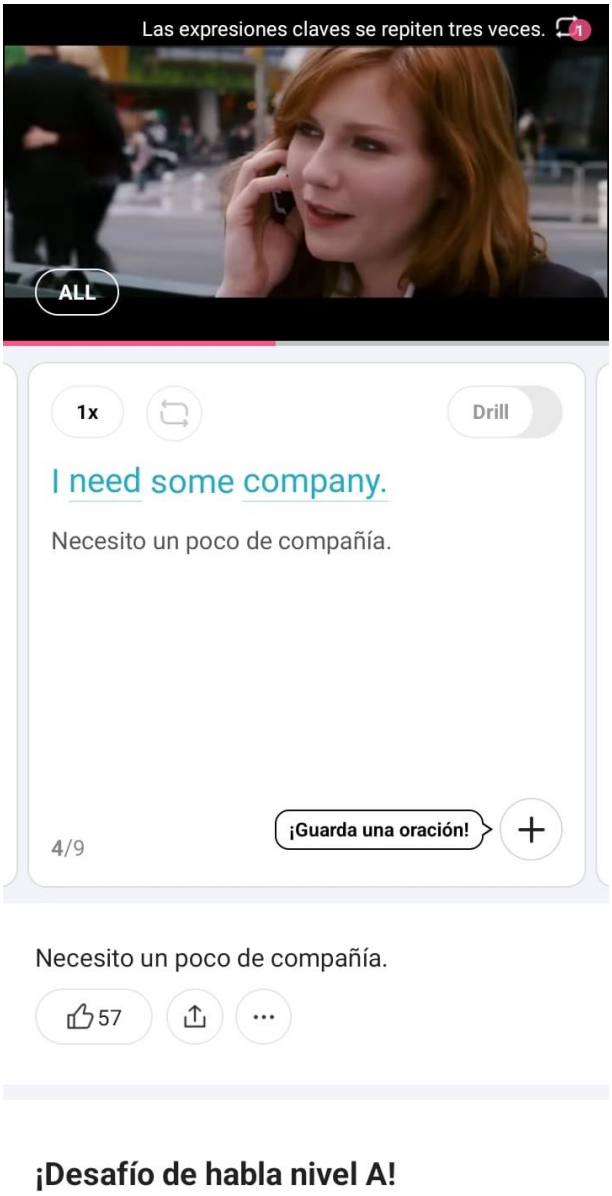

Figura 1.1.1.1. Ilustración gráfica de la aplicación "Cake" para aprender idiomas.

La desventaja con esta aplicación es que tiene una cantidad limitada de las lecciones que puedes hacer en un lapso de tiempo, si un usuario desea realizar más tiene la opción de ver anuncios para desbloquearlas o comprar una suscripción. Se trata de una aplicación que facilita el aprendizaje del inglés a través de archivos de audio, los cuales posibilitan la práctica de conversaciones

y la comprensión de vocabulario en diversos idiomas. Es la aplicación ideal para aquellos que buscan perfeccionar su pronunciación en inglés.

1.1.2 LearnEnglish Sounds Right

LearnEnglish Sounds Right es una aplicación móvil de British Council que se encuentra disponible en Google Play Store y App Store, es una aplicación que se centra en presentar una tabla de símbolos y sonidos de vocales, diptongos y consonantes.

Su uso es bastante simple, se pulsa un fonema y este emite el sonido de cómo debería pronunciarse, además, presionar el símbolo de flecha hacia abajo disponible en cada fonema muestra tres palabras de ejemplo con ese sonido, las cuales también pueden ser pulsadas para escuchar la pronunciación de una palabra. La figura 1.1.2.1 muestra un ejemplo de la aplicación.

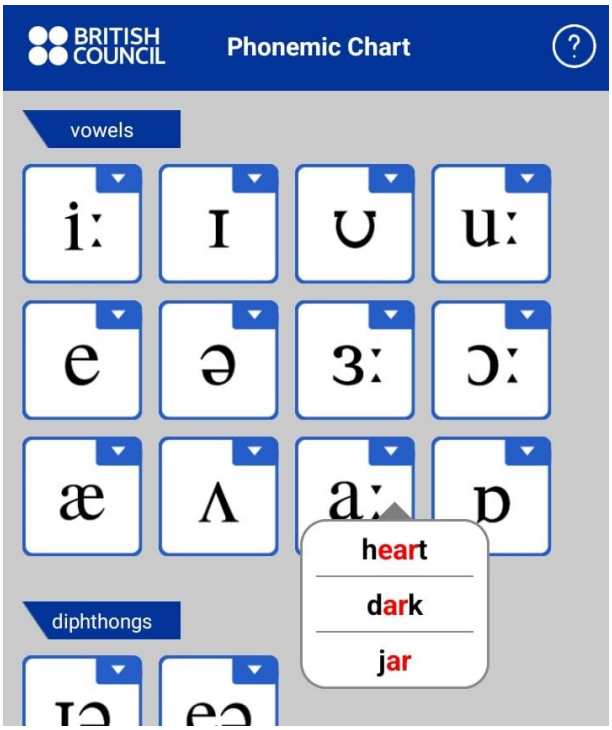

Figura 1.1.2.1. Aplicación "LearnEnglish Sounds Right". Muestra una parte de la tabla de pronunciación del alfabeto fonético, se puede apreciar un fonema con tres palabras de ejemplo.

Esta aplicación es muy útil para conocer el sonido de algunos fonemas, sin embargo, tiene solo ese propósito por lo que no es la mejor opción si se busca aprender y practicar inglés con ejemplos de muchas palabras o frases cotidianas.

1.1.3 Duolingo

Duolingo es otra aplicación tanto web como móvil que facilita el aprendizaje de múltiples idiomas, cuenta con un sistema en base a niveles, los cuales son necesarios para desbloquear unidades. Estudiar con Duolingo resulta entretenido, y estudios respaldan su efectividad. Acumulas puntos en las lecciones para desbloquear niveles adicionales a medida que perfeccionas tus destrezas para comunicarte en situaciones de la vida cotidiana.

Usarla es sencillo, cuenta con una interfaz de usuario agradable e intuitiva y con una pequeña mascota que te motiva para seguir avanzando. Entre los ejercicios que contiene en sus lecciones se encuentran preguntas y respuestas, repetir palabras y algunas lecturas. En la figura 1.1.3.1 se puede observar la pantalla principal de la aplicación.

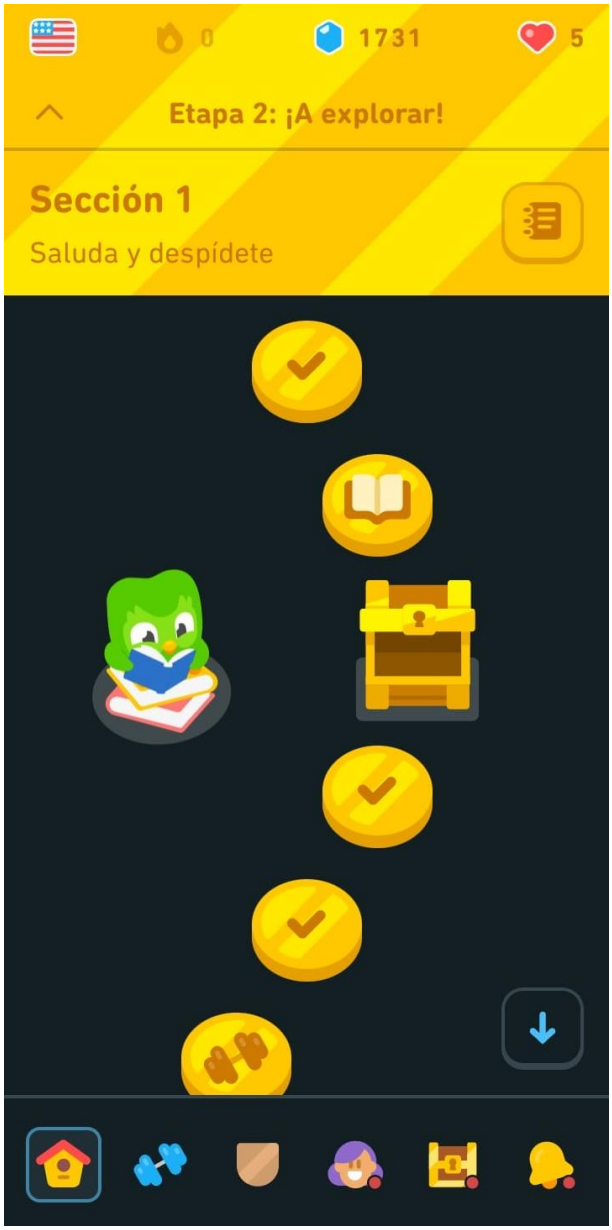

Figura 1.1.3.1. Pantalla principal de la aplicación "Duolingo".

Duolingo es una de las aplicaciones para aprender idiomas más populares que existen, sin embargo, hay algunas características que podrían mejorarse de acuerdo a algunos usuarios. Algunas de estas características son no poder elegir qué contenido se desea aprender o una cantidad excesiva de notificaciones, pero la principal desventaja es que la aplicación cuenta con un sistema de vidas que van disminuyendo con los errores del usuario y cuando

estas llegan a cero, se debe esperar a que estas se recarguen con el tiempo o comprar una suscripción para obtener vidas ilimitadamente.

1.2 Justificación

Considerando el análisis de las aplicaciones anteriores en el área de la enseñanza de idiomas, es posible darse cuenta de que cada aplicación aborda la problemática desde su punto de vista como LearnEnglish Sounds Right que se enfoca en mostrar al usuario la correcta pronunciación de fonemas con algunos ejemplos o Cake y Duolingo que se enfocan más en enseñar algunas frases comunes con su respectiva traducción y pronunciación, brindando un poco más de información como el contexto donde esas frases pueden ser utilizadas.

Todas estas aplicaciones ofrecen contenido de calidad e implementan diversas herramientas tecnológicas a fin de facilitar el aprendizaje del idioma inglés, sin embargo, algunas de estas cuentan con características que pueden ocasionar pérdida de interés por parte de los aprendices:

- Una aplicación muy básica puede resultar poco atractiva si el estudiante está buscando algo más completo.
- Demasiadas interrupciones debido a anuncios o limitantes de uso pueden causar pérdida de concentración.
- No ser capaz de elegir los temas para aprender puede llegar a ser molesto para un usuario que busca aprender temas de su interés.

Es por todo lo antes mencionado que la solución propuesta en el presente trabajo es utilizar herramientas de la tecnología como inteligencia artificial, minería de datos y otros algoritmos para lograr que esta aplicación permita ayudar a sus usuarios a aprender y practicar inglés de manera entretenida e intuitiva.

La implementación de propuestas tecnológicas para satisfacer las necesidades de la mayoría de los usuarios es un desafío constante. Sin embargo, gracias a los notables avances en el desarrollo de software y hardware en los

últimos años, se ha vuelto cada vez más viable y efectivo ofrecer soluciones innovadoras que aborden estas necesidades de manera eficiente. Estos avances tecnológicos han permitido la creación de plataformas y aplicaciones altamente funcionales, interactivas y accesibles, capaces de brindar experiencias de usuario enriquecedoras y adaptadas a las demandas actuales.

Debido a esto, el presente proyecto busca aprovechar herramientas tecnológicas para desarrollar una aplicación web interactiva destinada al aprendizaje del idioma inglés. El objetivo es satisfacer las necesidades de los usuarios proporcionando una aplicación accesible y de calidad que promueva un proceso de aprendizaje efectivo y atractivo, permitiendo mejorar sus habilidades lingüísticas de manera dinámica y personalizada al interactuar con historias cortas y resolver los ejercicios correspondientes.

CAPÍTULO II

MARCO TEÓRICO

Dado que el enfoque de esta tesis se dirige hacia el desarrollo de una aplicación web para la enseñanza del idioma inglés a través de diversas herramientas tecnológicas, es importante presentar los conceptos clave que sustentan y enriquecen la comprensión de este proyecto.

En ese sentido, en esta sección se abordarán los conceptos teóricos fundamentales que establecen las bases necesarias para la correcta implementación de los procesos de análisis, diseño, desarrollo y pruebas. Al familiarizarse con estos conceptos, se podrá apreciar de manera precisa el enfoque y la relevancia de esta tesis, permitiendo una interpretación más sólida y enriquecedora.

2.1 Metodologías para el desarrollo de software

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos organizativos diseñados con el propósito de organizar equipos de trabajo para desarrollar un software informático de la mejor manera posible.

A la hora de crear productos o soluciones para un cliente o mercado específico, es necesario considerar factores como los costes, la planificación, la dificultad, el equipo de trabajo disponible, los lenguajes utilizados, etc. Estos elementos se integran en una metodología de desarrollo facilitando la organización del trabajo de la forma más ordenada posible.

La aplicación de una metodología de desarrollo de software posibilita la disminución de la complejidad, la organización eficiente de tareas, la agilización del proceso y la mejora del resultado final de las aplicaciones en desarrollo

En la actualidad, las metodologías de desarrollo de software se pueden clasificar en dos grandes grupos: las ágiles y las tradicionales.

2.1.1 Metodologías de desarrollo de software ágiles

En la actualidad, las metodologías de desarrollo de software ágiles son ampliamente utilizadas debido a su alta flexibilidad y agilidad. Los equipos que las utilizan experimentan mayor eficiencia y productividad debido a que saben lo que deben hacer en cada momento. Además, permiten adaptar el software a las necesidades emergentes, facilitando así la creación de aplicaciones más eficientes y flexibles.

Las metodologías ágiles se basan en la metodología incremental, en la que se agregan nuevas funcionalidades a la aplicación final en cada ciclo de desarrollo. En las metodologías ágiles los ciclos son más cortos y rápidos, por lo que se agregan pequeñas funcionalidades en lugar de grandes cambios.

En este tipo de metodologías el equipo se reúne frecuentemente para discutir novedades, además, el proyecto se construye poco a poco mientras el cliente puede solicitar nuevos requerimientos o correcciones durante el desarrollo.

Algunas de las principales metodologías ágiles son: Kanban, Scrum, Lean y Programación extrema (XP).

2.1.1.1 Metodología Scrum

La metodología Scrum es una metodología para la gestión de proyectos complejos en los que se necesita un resultado a corto plazo en un entorno cambiante, estos requisitos de logran a través de sprints. Un sprint es un intervalo de tiempo corto y fijo en el que un equipo de scrum se dedica a cumplir una cantidad de trabajo establecida. Los protocolos clave en los que un equipo de scrum puede participar son:

- **Organización del backlog:** El propietario del producto se encarga de dirigir el producto conforme su visión y la de los clientes. Por tanto, realiza mantenimiento de esta lista utilizando los 4comentarios de los usuarios y el equipo de

desarrollo para ayudar a priorizar, mantenerla limpia y trabajar sobre ella en cualquier momento.

- **Planificación de sprint:** El equipo determina el alcance durante el sprint actual y se define el objetivo de este. Posteriormente, se añaden historias de usuario acorde al sprint desde el backlog del producto. Al finalizar la reunión. cada miembro del equipo de scrum debe tener en claro qué se puede entregar en el sprint y cómo se puede entregar el incremento.
- **Sprint:** Es el periodo de tiempo en el que un equipo trabaja de forma conjunta para realizar un incremento. La duración de un sprint puede cambiar dependiendo de la complejidad del proyecto, aunque lo más común es mantener de 2 semanas a un mes y todos los eventos, desde la planeación hasta la retrospectiva, tienen lugar durante el sprint.
- **Scrum diario o reunión rápida:** Se trata de una reunión de corta duración con la finalidad de que todo el equipo esté coordinado en torno al objetivo del sprint y cuenten con un plan para las próximas 24 horas. Generalmente cada miembro del equipo responde a las preguntas, ¿qué hice ayer?, ¿qué tengo planeado para hoy?, ¿hay algún obstáculo? La reunión rápida constituye el momento para plantear cualquier preocupación relacionada con el logro de los objetivos del sprint o para informar sobre posibles obstáculos detectados.
- **Revisión de sprint:** El equipo se reúne en una sesión informal para mostrar a las partes interesadas y a sus compañeros los elementos del backlog que están “finalizados”. El propietario del producto es quien decide si se lanza o no el incremento.

- **Retrospectiva del sprint:** En este punto, el equipo se reúne para analizar y documentar el funcionamiento adecuado o inadecuado del sprint, el proyecto, las herramientas u algún otro aspecto. El objetivo de esta reunión es que el equipo se centre principalmente en lo que salió bien y en lo que debe mejorarse para el siguiente sprint, antes que en lo que salió mal.

2.1.1.2 Historias de usuario

Una historia de usuario es una descripción general e informal de una función de software redactada desde la perspectiva del usuario final. Su objetivo es expresar cómo la función proporcionará valor al cliente.

Podemos ver las historias de usuario como oraciones informales para llevar a cabo un requisito de la aplicación, por lo general estas siguen la estructura “Como [rol], [quiero] [para].”, donde “rol” es el autor de la historia y no está limitado a un usuario de la aplicación, sino que también puede ser un miembro del proyecto, mientras que “quiero” es la intención u objetivo de la historia y “para” el beneficio general o la problemática que se resuelve.

2.1.1.3 Tareas con ayuda de las historias de usuario

Una vez que una historia de usuario es aprobada se debe planificar como una tarea sencilla y en el caso de que una tarea sea compleja puede incluir subtareas con el propósito de mantener una organización clara y eficiente.

Existen diversas aplicaciones web o de escritorio especializadas en la gestión del desarrollo de software y cada una plantea diferentes necesidades, no obstante, la gran mayoría toman en cuenta los siguientes rubros:

- **Nombre de la tarea:** es un nombre descriptivo de la tarea a realizar.
- **Responsable:** es el nombre de la persona responsable de cumplir con la implementación de la tarea.
- **Prioridad:** indica el valor de importancia que se le da a una tarea, el valor de prioridad puede ser muy bajo, bajo, medio, alto o muy alto en donde las prioridades bajas, a diferencia de las altas, no representan una repercusión significativa en la aplicación si es que no se llegan a cumplir.
    - **Prioridad alta:** estas tareas son indispensables pues están directamente relacionadas con el funcionamiento principal de la aplicación, es recomendable hacer estas tareas primero.
    - **Prioridad media:** denota una importancia media en el proyecto, aunque son importantes no son críticas para el funcionamiento de este y postergarlas no causaría un impacto que pueda comprometer la aplicación.
    - **Prioridad baja:** señalan un nivel de baja urgencia y son altamente flexibles en el sentido de que usualmente se pueden adelantar o retrasar, cumplir una tarea con prioridad baja es importante para satisfacer a los usuarios y tener un producto de mejor calidad, sin embargo, la aplicación puede funcionar correctamente sin estas.
    - **Prioridad muy alta:** son las tareas de mayor importancia, esto quiere decir que se deben ejecutar lo más pronto posible y no realizarlas a tiempo podría comprometer severamente el flujo del proyecto.

- **Prioridad muy baja:** son las tareas de menor importancia por lo que, de considerarse necesario, podrían atrasarse o hasta omitirse para dar lugar a la implementación de tareas más importantes.

- **Descripción:** una descripción a detalle de lo que se desea realizar, puede tener tantas líneas como sean necesarias pues su objetico es garantizar que el desarrollador entienda perfectamente lo que se debe realizar.
- **Duración:** la duración de tiempo aproximada para terminar una tarea.

Cuando una tarea cuenta con una complejidad considerable es recomendable dividirla en subtareas, una subtarea contiene los mismos aspectos de una tarea en general.

2.2 Base de datos

Una base es una recopilación organizada de datos estructurados que son almacenados electrónicamente en un sistema informático, puede almacenar distintos tipos de información como texto, números, videos, imágenes y archivos. Para mantener todo en orden y ejecutar operaciones como acceder, modificar y almacenar datos, se emplea un software llamado sistema de administración de bases de datos (DBMS). En el contexto de la informática, el término "base de datos" puede referirse tanto al software encargado de gestionarla como al sistema en su conjunto, o incluso a las aplicaciones conectadas a esa base de datos.

2.2.1 Tipos de bases de datos

Existen diferentes tipos de bases de datos, cada una se adapta mejor a diversas necesidades. Por ejemplo, las bases de datos relacionales son conocidas por su capacidad de acceder a información estructurada de una manera eficaz y flexible, mientras que las bases de datos NoSQL se destacan por su capacidad

para manejar datos no estructurados de aplicaciones complejas. También están las bases de datos orientadas a objetos o las bases de datos orientadas a grafos.

2.2.2 Base de datos relacional

Resulta importante mencionar que, así como existe el término DBMS, a un sistema de bases de datos relacional también se le conoce como RDBMS, los cuáles son utilizados para la administración de bases de datos relacionales. En particular, las bases de datos relacionales son un modelo de base de datos, es decir, siguen un conjunto de reglas que determinan la organización, manipulación y almacenamiento de los datos. Algunos de los RDBMS más utilizados son MySQL, PostgreSQL, Oracle Database, MariaDB y Microsoft SQL Server.

Dentro de un modelo de base de datos relacional, cada conjunto de información se organiza en una tabla sus características o atributos se utilizan como columnas y cada instancia de datos se registra como una fila. En la tabla de la figura 2.2.1 podemos observar el ejemplo de una tabla para guardar películas.

año	género	título	duración
2008	Acción	Batman: el caballero de la noche	152
2002	Acción	El hombre araña	121
2010	Ciencia ficción	El origen	148

Figura 2.2.1. Tabla sencilla para el almacenamiento de películas. Ejemplo con el tipo de datos relacional o estructurado, la tabla "películas" tiene atributos como "año", "género", "título" y "duración" representados en forma de columnas, mientras que los registros guardados son las propias filas de la tabla.

En bases de datos, una relación indica la asociación existente entre entidades o tablas, estas son útiles para mantener una organización óptima y eficiente de los datos, en las figuras 2.2.2 a 2.2.4 podemos ver tablas

representadas como entidades ejemplificando los diferentes tipos de relaciones y su funcionamiento.

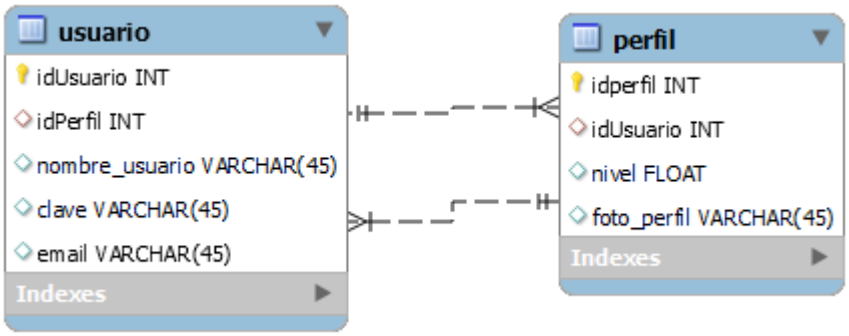

Figura 2.2.2. Relación uno a uno. Una relación uno a uno indica que una fila en una tabla sólo puede tener relación con otra fila de otra tabla, es utilizada principalmente para separar información sensible de una tabla y así aumentar la seguridad de la información. En este ejemplo se muestra como la tabla usuario se relaciona directamente con la tabla perfil y viceversa.

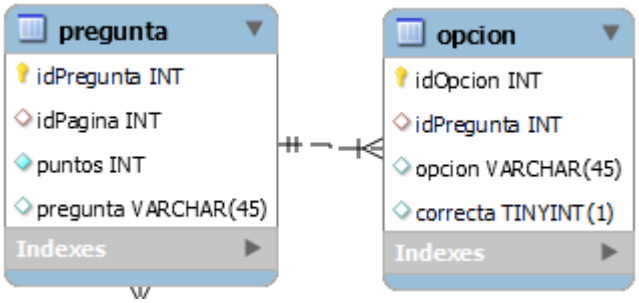

Figura 2.2.3. Relación uno a muchos. En este tipo de relación una fila de una tabla puede tener relación con muchas filas de otra tabla, no obstante, cada fila de la segunda tabla sólo puede coincidir con una fila de la primera. En este ejemplo se representa el funcionamiento de una pregunta de opción múltiple, donde una pregunta puede tener varias opciones y una opción pertenece a una pregunta.

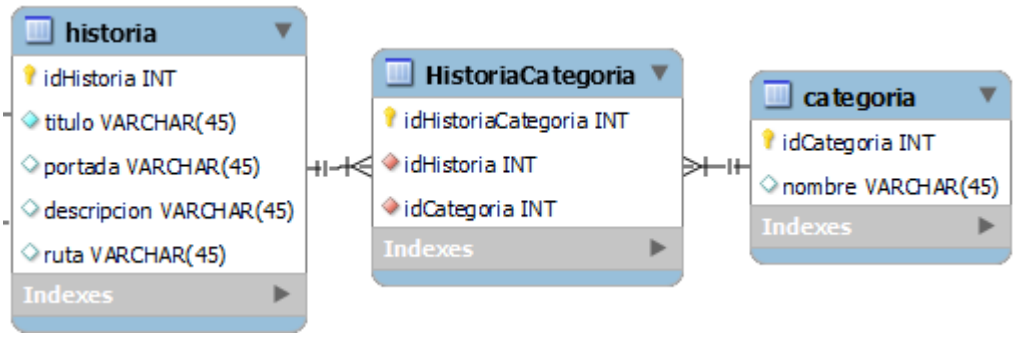

Figura 2.2.4. Relación muchos a muchos. En este tipo de relación una fila de una tabla puede tener relación con muchas filas de otra tabla y viceversa, esto se logra a partir de una tabla intermediaria que contiene dos columnas, donde cada una tiene relación muchos a uno con una tabla. En este ejemplo se muestran las tablas historia y categoría, donde una historia puede tener muchas categorías al igual que una categoría puede tener muchas historias.

2.2.2.1 SQL

El Structured Query Language (SQL) es un lenguaje de programación estandarizado para la interacción con los sistemas de gestión de bases de datos relacionales (RDBMS). El lenguaje SQL consta de palabras reservadas como SELECT, FROM, WHERE, AND, OR y LIKE entre muchas otras, que permiten obtener información de manera sencilla e intuitiva.

En el siguiente fragmento de código se observa una consulta SQL, seguido de la figura 2.2.5 en donde se aprecia el resultado.

```
SELECT id, story_id, user_profile_id, score_percentage FROM `score
` WHERE score_percentage >= 75;
```

←T→				id	story_id	user_profile_id	score_percentage
☐	Editar	Copiar	Borrar	4	3	1	100
☐	Editar	Copiar	Borrar	6	3	1	100
☐	Editar	Copiar	Borrar	10	4	1	100
☐	Editar	Copiar	Borrar	11	6	1	100
☐	Editar	Copiar	Borrar	12	6	1	75
☐	Editar	Copiar	Borrar	13	8	1	100

Figura 2.2.5. Resultado de una consulta SQL.

2.2.3 Base de datos no relacional

Las bases de datos relacionales también son conocidas como NoSQL y son aquellas que no siguen el modelo tabular de filas y columnas.

Estas bases de datos utilizan un modelado de almacenamiento acorde a diferentes tipos, algunos ejemplos son clave-valor y documentos.

2.2.3.1 Clave-valor

En las bases de datos clave-valor, cada dato está asociado a una clave única utilizada para acceder a la información de un elemento y por lo general los datos son almacenados como objetos de tipo binario. Se caracterizan por su rapidez para acceder a los datos y son ampliamente utilizadas en la administración de sesiones o almacenamiento en caché.

Algunos ejemplos de bases de datos clave-valor incluyen Redis y BigTable.

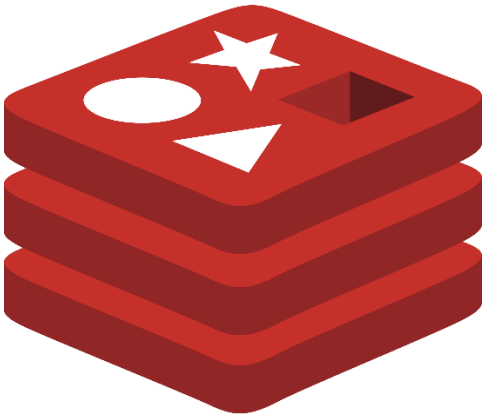

Figura 2.2.6. Logo de la base de datos no relacional "Redis".

2.2.3.2 Documentales

Las bases de datos de documentos funcionan almacenando la información en documentos, donde cada documento puede almacenar diferentes tipos de datos. Los registros en una base de datos documental suelen tener formato XML o JSON. Se caracterizan por su amplia flexibilidad y son esto las hace frecuentemente utilizadas en sistemas susceptibles a cambios.

Algunos ejemplos de bases de datos documentales son MongoDB y CouchDB.

```
mongosh mongodb+srv://<credentials>@cluster0.g2cycis.mongodb.net/
Atlas atlas-w72ye2-shard-0 [primary] admin> use unicorns_db
switched to db unicorns_db
Atlas atlas-w72ye2-shard-0 [primary] unicorns_db> db.createCollection("unicorns")
{ ok: 1 }
Atlas atlas-w72ye2-shard-0 [primary] unicorns_db> db.unicorns.find()

Atlas atlas-w72ye2-shard-0 [primary] unicorns_db> db.unicorns.insert({color: "pink"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("656029eaae5ddb7f449d9d19") }
}
Atlas atlas-w72ye2-shard-0 [primary] unicorns_db> db.unicorns.find()
[ { _id: ObjectId("656029eaae5ddb7f449d9d19"), color: 'pink' } ]
Atlas atlas-w72ye2-shard-0 [primary] unicorns_db>
```

Figura 2.2.7. Ejecución de comandos para manipulación de datos en MongoDB.

2.3 Patrón de diseño MVC

El MVC (Modelo-Vista-Controlador), es un patrón de diseño de software enfocado en la implementación de interfaces de usuario, gestión de datos y control lógico que divide las responsabilidades en tres componentes:

- **Modelo:** responsable de la gestión de datos y lógica de negocios.
- **Vista:** se encarga del diseño y la presentación visual.
- **Controlador:** gestiona la dirección de los comandos hacia modelos y vistas.

En la figura 2.3.1 podemos observar un ejemplo.

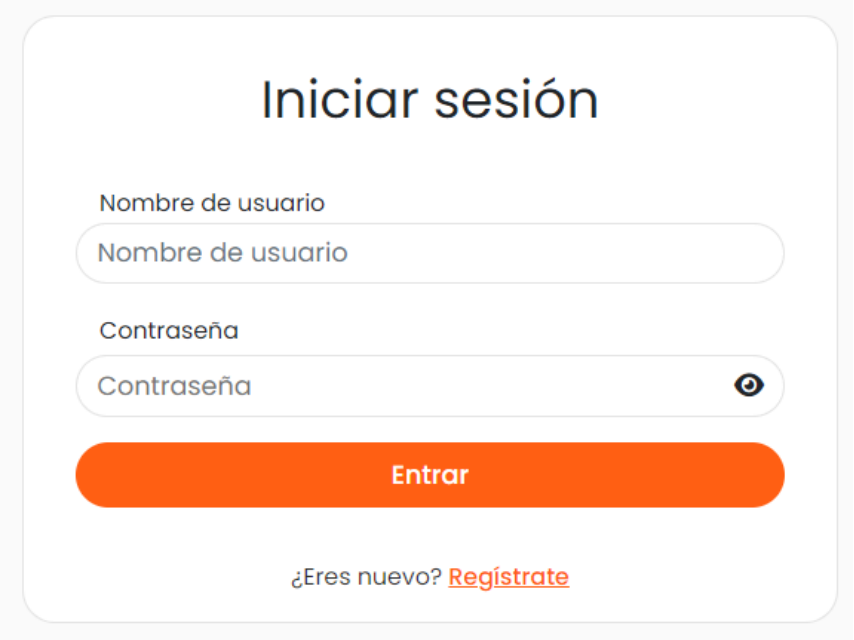

Figura 2.3.1. Ejemplo de modelo MVC con una pantalla de inicio de sesión. La vista es lo que observamos, al presionar el botón de "Entrar", el controlador se comunica con el modelo el cual le indica si la combinación de credenciales es correcta y actúa en consecuencia, puede enviar a la vista un mensaje de "Contraseña incorrecta", como también puede modificar el modelo con un dato de "último inicio de sesión" y enviar otra vista para acceder al menú principal de la aplicación.

2.4 Django y su patrón de diseño MTV

Django es un framework web de alto nivel, de código abierto y escrito en Python diseñado para permitir a los desarrolladores concentrarse en la creación de aplicaciones y evitar perder tiempo en redundancias.

Django sigue el patrón de diseño MTV (Modelo-Plantilla-Vista), una variación del MVC, en Django cada uno de estos tiene diferentes responsabilidades:

- **Modelo:** responsable de manejar los datos, al igual que en MVC.
- **Vista:** es la función que se llama para una URL en particular se encarga de enviar los datos a una plantilla.
- **Plantilla:** se encarga de mostrar la información recibida de una vista a través de HTML, CSS, etc.

En las figuras 2.4.1 a 2.4.4 se ilustra el proceso de MVT con el framework Django.

```python
# Tag
class Tag(models.Model):
    class Meta:
        db_table = prefix + 'tag'

    # keys
    id = models.AutoField(primary_key=True)
    # fields
    name1 = models.CharField(max_length=100)
    name2 = models.CharField(max_length=100)

    def __str__(self) -> str:
        return self.name1
```

Figura 2.4.1. Modelo en Django. Es responsable de manejar los datos, tales como la creación y modificación de tablas o registros en una base de datos.

```python
def index(request):
    if request.method == 'GET':
        tags = Tag.objects.all().order_by('name1')
        context = {
            'tags': tags
        }
        return render(request, 'admin/tags.html', context)
```

Figura 2.4.2. Vista en Django. Indica la lógica a realizar antes de renderizar una plantilla, en este ejemplo se observa como la vista se comunica con el modelo y devuelve como resultado una plantilla 'admin/tags.html' con la información necesaria para que esta renderice la información.

```html
<table class="table">
  <thead>
    <tr>
      <th class="fs-5 fw-bold">#</th>
      <th class="fs-5 fw-bold">English name</th>
      <th class="fs-5 fw-bold">Spanish name</th>
      <th class="fs-5 fw-bold">Actions</th>
    </tr>
  </thead>
  <tbody>
    {% for tag in tags %}
    <tr id="tag_{{ tag.id }}">
      <td class="counter">{{ forloop.counter }}</td>
      <td>{{ tag.name1 }}</td>
      <td>{{ tag.name2 }}</td>
      <td>
        <a class="btn" type="button" role="button" title="Edit tag" data
          data-bs-target="#modal_edit_tag_{{ tag.id }}">
          <span style="color: #77b9fc; font-size: 0.9rem;"><i class="f
        </a>
        <a class="btn" type="button" role="button" title="Delete tag" da
          data-bs-target="#modal_delete_tag_{{ tag.id }}">
          <span style="color: #f53126; font-size: 0.9rem;"><i class="f
        </a>
      </td>
    </tr>
    <!-- Modal Edit Tag -->
    <div class="modal fade" id="modal_edit_tag_{{ tag.id }}" tabindex="-
    </div>
    <!-- End of Modal Edit Tag -->

    <!-- Modal Delete Tag -->
    <div class="modal fade" id="modal_delete_tag_{{tag.id}}" tabindex="-
    </div>
    <!-- End of Modal Delete Tag -->
    {% endfor %}
  </tbody>
</table>
```

Figura 2.4.3. Plantilla (HTML) en Django. Muestra la información que recibe de una vista, en este ejemplo se observa la creación de una tabla que itera los registros de "tags" mediante una etiqueta.

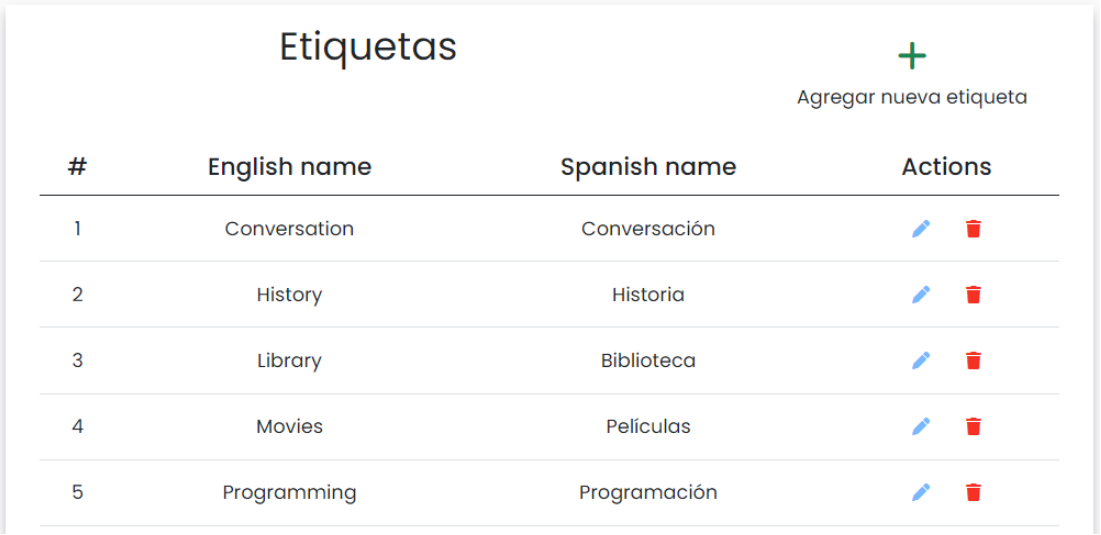

Figura 2.4.4. Plantilla (Visualización) en Django. Muestra la información que recibe de una vista, en este ejemplo se muestra el resultado de utilizar las etiquetas de Django para renderizar datos de una manera sencilla.

2.5 HTTP

HTTP proveniente de sus siglas en inglés que significan “Hypertext Transfer Protocol”, es un protocolo que facilita la solicitud y transferencia de datos y recursos. HTTP es el protocolo base de cualquier intercambio de información en la web y sigue una estructura cliente-servidor. En este contexto, el cliente, por lo general un navegador web, inicia una solicitud de datos; de esta manera una página web se compone de la combinación de varios subdocumentos recibidos que pueden incluir especificaciones de estilo (CSS), texto, imágenes, videos, scripts, entre otros elementos.

En este protocolo, el cliente y el servidor se comunican a través de mensajes individuales el uno con el otro. Los mensajes enviados por un cliente reciben el nombre de solicitudes o peticiones mientras que los mensajes que el servidor envía son llamados respuestas, es el cliente el que siempre inicia una petición y el servidor responde.

En la figura 2.5.1 podemos observar de manera gráfica el flujo que sigue HTTP.

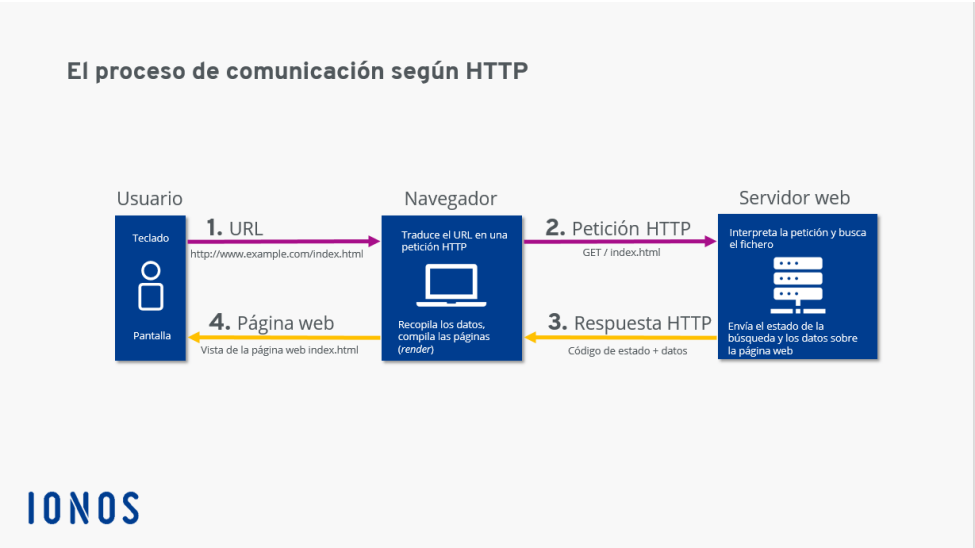

Figura 2.5.1. Esquema del proceso de comunicación mediante HTTP, recuperado el 15 de noviembre de 2023, https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/protocolo-http/

2.6 WebSocket

El protocolo de WebSocket permite una comunicación bidireccional entre cliente y servidor a través de una conexión TCP. Este proceso inicia con un paso crucial conocido como “apretón de manos” (handshake) donde el cliente envía una solicitud HTTP al servidor, esperando una respuesta que permita establecer un canal de conexión bidireccional.

En la figura 2.6.1 podemos observar de manera gráfica el flujo que sigue el protocolo WebSocket.

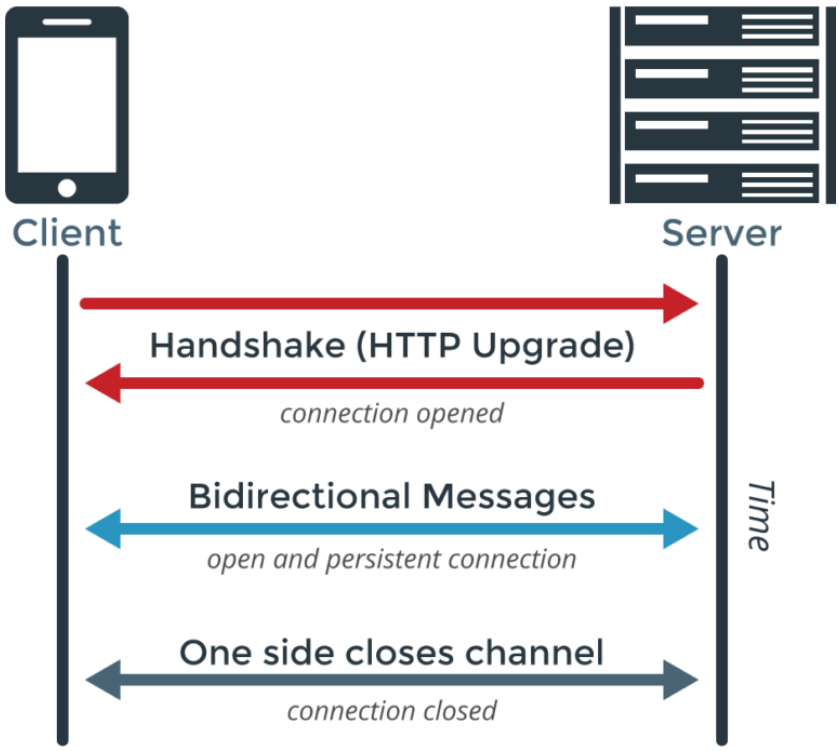

Figura 2.6.1. Protocolo de enlace WebSocket, recuperado el 15 de noviembre de 2023, https://www.dotcom-monitor.com/blog/es/websocket-application-monitoring/

Esta tecnología revoluciona la forma en que los navegadores web y aplicaciones interactúan con los servidores, pues permite que el cliente reciba mensajes del servidor sin necesidad de realizar peticiones. Esta capacidad es fundamental para aplicaciones en tiempo real, donde se requiere una actualización constante de la información.

Algunos ejemplos de WebSockets incluyen aplicaciones de mensajería, videollamadas, plataformas con chats en vivo y videojuegos online.

2.7 Diagrama de casos de uso

El diagrama de casos de uso, perteneciente a la categoría de diagramas de comportamiento de UML, se emplea frecuentemente en el análisis de diversos

sistemas ya que facilita la representación visual de diferentes roles y su interacción dentro del sistema.

Para asegurar una comprensión general de un diagrama de casos de uso, se utilizan elementos estandarizados en su elaboración. Los tres elementos principales son:

- **Actor:** no está limitado a una persona, puede ser un sistema, rol, etc. Se representa mediante el dibujo de una figura humana.
- **Sistema:** tiene forma de rectángulo y representa el sistema de caso de uso.
- **Caso de uso:** se muestra en forma de elipse, suele incluir un breve texto describiendo el proceso en cuestión.

La relación entre elementos se representa mediante líneas de unión denominadas asociaciones.

Una línea recta entre el actor y el caso de uso, se indica una relación directa entre el actor y el proceso descrito en la elipse. En contraste, una línea discontinua señala una relación entre casos de uso distintos. Para diferenciar entre los dos tipos de asociación, se utiliza una palabra clave denominada "estereotipo" en UML, que se coloca entre dos pares de paréntesis angulares.

La relación de dependencia entre los casos de uso se expresa mediante la punta de una flecha. Además, existe otra relación llamada generalización, que resulta útil al describir comportamientos generales. Esta relación se emplea eficazmente cuando se trabaja mediante roles en niveles, ya que permite indicar que todo lo que pueda realizar un elemento A, también lo puede realizar un elemento B. Se representa mediante una línea continua con una flecha partiendo del elemento B con dirección al elemento A.

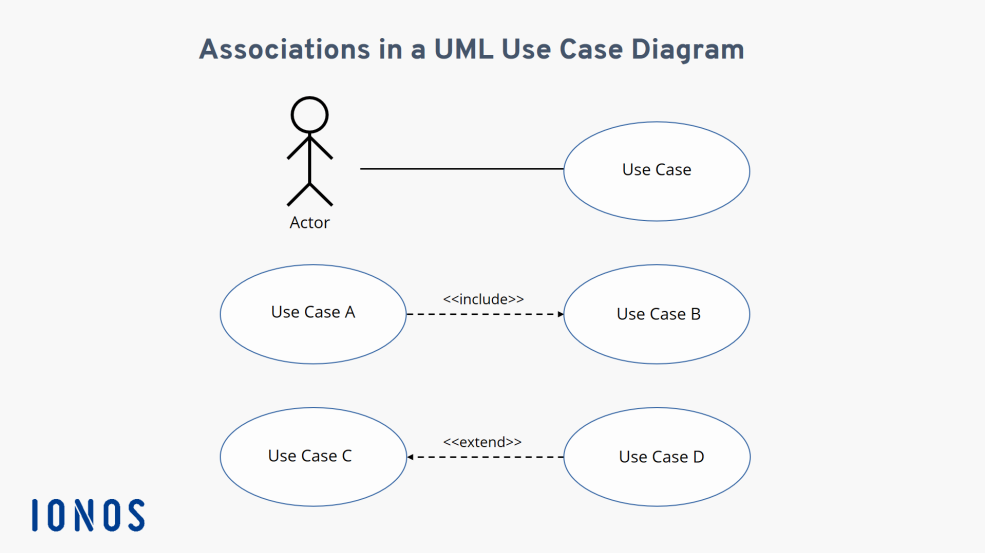

Figura 2.7.1. Esquema de la relación entre actor y caso de uso, así como entre diferentes casos de uso, recuperado el 15 de noviembre de 2023, https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/

Existen dos relaciones estereotipo fundamentales, <<include>> y <<extend>>, podemos simplemente ver estas relaciones como un “puede” y “debe” respectivamente, tomando de referencia la figura 2.7.1 tenemos que:

- **Include:** indica que una vez ejecutado el caso A, debe haber un caso B que sea llamado.
- **Extend:** indica que una vez que se presente el caso C, se crea un caso D que es posible pero no obligatorio de ejecutar.

2.8 Sistemas de recomendación

Un sistema de recomendación constituye una herramienta que establece un conjunto de criterios y evaluaciones provenientes de datos de usuarios con el objetivo de realizar predicciones sobre elementos que podrían llegar a ser útiles o valiosos para el usuario. Funcionan recopilando y analizando los datos que el

usuario proporciona de manera directa o indirecta a fin de utilizar esta información para recomendar contenido.

2.8.1 Sistemas de recomendación basados en contenido

Los sistemas de recomendación basados en contenido son aquellos que ofrecen sugerencias de elementos a los usuarios que guardan similitud con aquellos que han demostrado preferir en el pasado. Por ejemplo, podrían proponer películas que compartan actores, director o género similar, noticias con contenido relacionado, o personas que tengan amigos en común.

2.8.2 Sistemas de recomendación basados en filtrado colaborativo

Las técnicas de recomendación que utilizan el filtrado colaborativo se fundamentan en la noción de la "sabiduría de la multitud" para proponer sugerencias de elementos. Estos sistemas parten de la premisa básica de que los usuarios con preferencias similares en el pasado seguirán teniéndolas en el futuro.

El filtrado colaborativo basado en usuarios identifica a usuarios con gustos similares mediante el análisis de sus patrones de valoración, posteriormente siguiere los elementos que usuarios con gustos similares han valorado positivamente.

CAPÍTULO III

ANÁLISIS Y DISEÑO

En este capítulo se contempla el análisis de la aplicación web considerando la metodología ágil de desarrollo de software: Scrum, tomando en cuenta los objetivos planteados. El primer paso es definir las historias de usuario para obtener las tareas necesarias y los requisitos para implementar cada una de estas. Es de suma importancia identificar las tareas que pueden hacerse independientemente de otras y las que requieren de otros módulos para funcionar, para iniciar se van a priorizar las historias que puedan realizarse independientemente y/o sean fundamentales para el funcionamiento de la aplicación.

3.1 Definir historias de usuario

Siguiendo la metodología ágil Scrum, el primer paso fue definir las historias de usuario, a continuación, se muestran algunas de las historias de usuario que se obtuvieron:

Historia 1: Como desarrollador principiante, quiero utilizar herramientas acordes a mi nivel de conocimientos para poder avanzar lo más eficientemente posible.

Historia 2: Como desarrollador de la aplicación, quiero tomar en cuenta el consejo de mis profesores de tener una aplicación consistente, por lo que quiero utilizar una pequeña paleta de colores, o en su caso una paleta con colores no muy diferentes.

Historia 3: Como desarrollador de la aplicación, quiero mantener un diseño web responsivo, con el objetivo de que la aplicación sea accesible desde diferentes dispositivos.

Historia 4: Como usuario de la aplicación y aficionado a la lectura, quiero que el aprendizaje sea mediante historias cortas, así que me gustaría contar con algo parecido a una biblioteca donde puedas elegir el libro que quieras para aprender del contenido que tú quieras.

Historia 5: Como aprendiz del idioma con problemas de pronunciación, quiero ejercicios de pronunciación con el objetivo de mejorar mi habilidad para hablar en voz alta.

Historia 6: Como creador de contenido, quiero un modo administrador que me permita administrar contenido fácilmente durante toda la aplicación.

Historia 7: Como desarrollador de la aplicación, quiero ofrecer a mis usuarios una manera de traducción diferente de la típica forma de colocar texto en español e inglés uno encima de otro.

Historia 8: Como usuario de aplicaciones de aprendizaje, me gustaría llevar un registro de cómo va mi progreso en diferentes áreas relativas al inglés para saber en cuáles debo enfocarme para mejorar mi nivel.

Historia 9: Como usuario de aplicaciones de inglés, quiero tener una opción para guardar el contenido que me guste y así poder visualizarlo en algún otro momento.

Historia 10: Como responsable de la aplicación, quiero que el contenido mostrado sea personalizable para que mis usuarios tengan una mejor experiencia.

Historia 11: Como responsable de la aplicación, quiero que mis usuarios puedan guardar pequeñas notas sobre lo que están aprendiendo, pues no muchas aplicaciones cuentan con esta característica.

Historia 12: Como miembro del staff de la aplicación, quiero que los usuarios me reporten si las historias contienen errores para así poder corregirlas lo más pronto posible.

3.2 Creación de un diagrama de casos de uso

Un diagrama de casos de uso es útil para analizar un sistema con el objetivo de facilitar una representación visual.

El funcionamiento puede definirse mediante una serie de declaraciones.

Un usuario “estudiante” puede crear una cuenta, iniciar sesión, tomar notas, desbloquear historias, marcar que le gustan historias, reportar historias y contestar las mismas.

Las funciones para desbloquear, gustar, reportar y contestar requieren la creación de una historia.

El rol “staff” puede hacer todo lo que el “estudiante”, además de crear historias y administrar reportes.

El rol “administrador” puede hacer lo que el “staff” y también puede administrar los sistemas de recomendación y las cuentas que los estudiantes hayan creado.

El actor de nombre “Servidor” indica algunas funciones de la aplicación como recomendar historias, devolver retroalimentación de un ejercicio o enviar una calificación cuando el estudiante envíe sus respuestas.

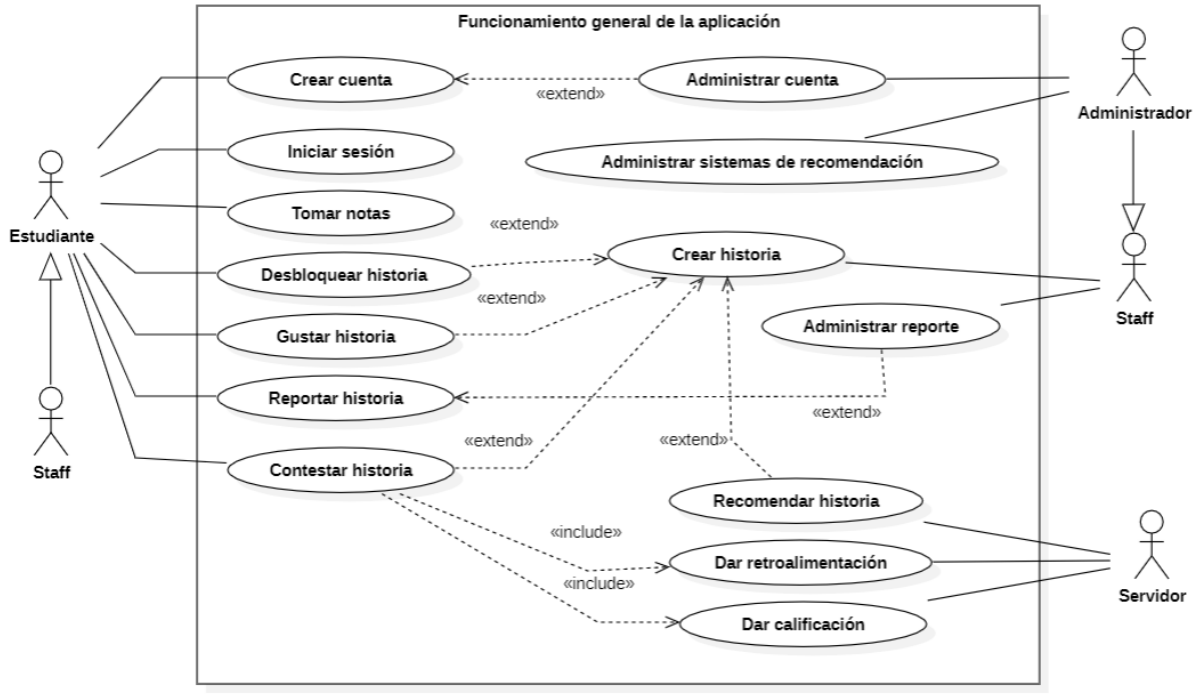

Figura 3.2.1. Diagrama de casos de uso del funcionamiento general de la aplicación.

3.3 Planificación de tareas

Una vez definidas las historias de usuario, se procedió a abstraer la información más relevante entre de cada una de ellas, esto dio como resultado la planificación de las tareas mencionadas en esta sección.

Las siguientes tareas no contienen el rubro “responsable”, esto es con el fin de evitar redundancia pues la responsabilidad de implementar todas las tareas recae en el autor de la presente tesis. De la misma manera se evitó poner el requerimiento de un diseño responsivo para cada página, pues se entiende que la aplicación completa será responsiva y sólo se menciona este tema en dado caso de ser necesario debido a cumplir con requerimientos específicos.

Tarea 1

Duración: 2 semanas

Inicio: 8 de agosto de 2022

Fin: 22 de agosto de 2022

Tarea 1	
Nombre	Elección de tecnologías de acuerdo a su curva de aprendizaje.
Prioridad	Muy alta.
Descripción	Investigar sobre diferentes frameworks para la creación de aplicaciones web. Probar al menos tres frameworks para el desarrollo de aplicaciones web, implementando una aplicación sencilla en cada uno de estos e investigar sus principales ventajas y desventajas. Por último, elegir aquel que cuente con una fácil curva de aprendizaje y que a su vez sea compatible con otras tecnologías con las que ya se tiene experiencia.

Tarea 2

Duración: 2 semanas

Inicio: 22 de agosto de 2022

Fin: 5 de septiembre de 2022

Tarea 2	
Nombre	Implementar páginas y métodos para la autenticación.
Prioridad	Muy alta.
Descripción	Crear las páginas para iniciar sesión y registrarse, e implementar lo necesario para la funcionalidad de estas dos páginas, así como un botón que permita cerrar la sesión.

Tarea 3

Duración: 2 semanas

Inicio: 5 de septiembre de 2022

Fin: 19 de septiembre de 2022

Tarea 3	
Nombre	Creación de una plantilla principal para mostrar el contenido de la aplicación.
Prioridad	Media.
Descripción	Diseñar una plantilla general para mostrar todo el contenido de la aplicación, esta plantilla incluye una barra de navegación que mostrará diferentes botones si el usuario ha iniciado sesión o no. • Sesión iniciada: enlace al inicio con el logo de la aplicación y un botón para cerrar la sesión, además se necesita una barra lateral que se pueda expandir o contraer manual y responsivamente y en esta habrá más enlaces hacia diferentes contenidos, los contenidos de administración sólo estarán disponibles para usuarios con permisos de administrador. • Sesión no iniciada: incluir un enlace con el logo de la aplicación para ver la página de presentación, además se necesitan dos más para mostrar la página de registro y la de inicio de sesión. Cuando el ancho de la pantalla sea de 500px o menor, la barra lateral deberá ocultarse completamente y en su lugar mostrar un botón en la barra de navegación, cuya función

	será abrir la barra lateral nuevamente con el objetivo de mostrar los enlaces a diferentes partes de la aplicación, la barra lateral debe contar con una forma de volver a cerrarse. En el caso de que la sesión no esté iniciada y el tamaño de la pantalla sea pequeño, colapsar los enlaces en la barra de navegación la cuál de igual manera se abrirá y cerrará con un botón, este comportamiento debe ocurrir cuando se considere necesario por lo que no es forzoso hacerlo cuando la pantalla sea menor a 500px.

Tarea 4

Duración: 3 semanas

Inicio: 12 de septiembre de 2022

Fin: 3 de octubre de 2022

Tarea 4	
Nombre	Implementar la creación y administración de historias y etiquetas.
Prioridad	Alta.
Descripción	Implementar las funciones básicas para listar, crear, editar y eliminar historias y etiquetas. Cada historia puede tener muchas etiquetas y a su vez cada etiqueta puede tener muchas historias.

Tarea 4.1	
Nombre	Implementar la administración de etiquetas.
Prioridad	Alta.
Descripción	Implementar las funciones básicas para listar, crear, editar y eliminar etiquetas. Las etiquetas deben tener un nombre en inglés y un nombre en español, cada etiqueta puede tener una o muchas historias. **Adicional:**

	Estas funciones sólo estarán disponibles para usuarios con roles de Staff o Administrador.

Tarea 4.2	
Nombre	Implementar la administración de historias.
Prioridad	Alta.
Descripción	Implementar las funciones básicas para listar, crear, editar y eliminar historias. Cada historia debe tener un título en inglés, un título en español, una portada, una descripción en inglés, una descripción en español, una cantidad de experiencia necesaria para desbloquear la historia, un número de “likes”, una ruta para acceder a esta historia, la fecha y hora de creación de la historia y la fecha y hora de actualización de la historia, además puede contener una o muchas etiquetas. El valor de la ruta es un slug del título en inglés de la historia.

Tarea 4.3	
Nombre	Creación de la página “Galería de historias” para ver todas las historias.

Prioridad	Alta.
Descripción	Agregar a la barra lateral un enlace que redireccione hacia una página para visualizar todas las historias. Esta página debe mostrar todas las historias que se encuentren en la base de datos. Las historias deben ser presentadas en una cuadrícula mostrando su imagen de portada y debajo de estas, el nombre de la historia en inglés. Si un usuario presiona click en alguna de estas historias, lo redireccionará a una página donde podrá ver la información de la historia.

Tarea 5

Duración: 2 semanas

Inicio: 3 de octubre de 2022

Fin: 17 de octubre de 2022

Tarea 5	
Nombre	Agregar páginas a las historias.
Prioridad	Alta.
Descripción	Hacer los cambios que sean necesarios para lograr lo siguiente: Una historia puede tener muchas páginas y una página debe pertenecer a una historia, si una historia es borrada las páginas deberán hacerlo también. Una página puede tener un subtítulo en español y un subtítulo en inglés, debe tener un tipo de página y su fecha de creación. El tipo de página es un número entero. **Adicional:** Estas funciones sólo estarán disponibles para usuarios con roles de Staff o Administrador.

Tarea 5.1	
Nombre	Crear vista para agregar páginas.

Prioridad	Alta.
Descripción	Implementar un botón en la parte de administración de historias que redireccione a una nueva página llamada “Ver páginas”, esta página debe permitir crear una nueva página y administrar las ya existentes con botones para editarlas y eliminarlas. Las páginas ya existentes deben ser mostradas en una lista y ordenadamente iniciando por las que se crearon primero. Para crear una nueva página el usuario debe elegir entre tres plantillas: “#1: Plantilla general”, “#2: Video y ejercicios”, “#3: Diseño libre”; el tipo de elección que el usuario haga se usará para rellenar el atributo “tipo de página” siendo 1, 2 y 3 respectivamente. Las plantillas deben ser enlaces hacia un sitio en donde se rellenará la información necesaria para agregar una nueva página a la historia actual. Estos enlaces deben ser mostrados de forma llamativa, se adjuntan referencias en la figura T5.1.

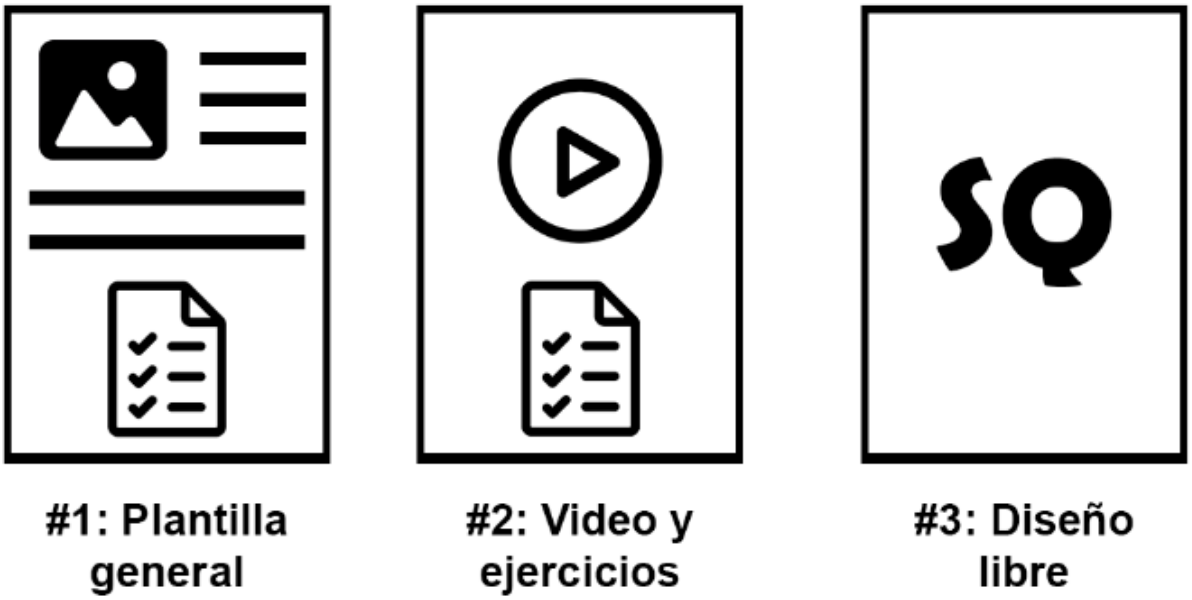

Figura 3.3.1. Referencia para seleccionar un tipo de plantilla.

Tarea 6

Duración: 3 semanas

Inicio: 10 de octubre de 2022

Fin: 31 de octubre de 2022

Tarea 6	
Nombre	Creación de efecto para la traducción de texto.
Prioridad	Muy alta.
Descripción	Crear una forma para traducir los textos. Al colocar el cursor sobre un texto que tenga traducción, este deberá cambiar su tipo de automático a “pointer”, además el texto se resaltará en un color amarillo rgb(253, 253, 164). Al realizar un click sobre un texto que tenga traducción, cada palabra del texto debe realizar una animación igual que una hoja de papel que rota sobre el eje Y, revelando detrás de este, el texto traducido de inglés a español y viceversa. **Consideraciones:** Los textos se deben comportar como un texto normal, esto significa que una traducción no debe crear una nueva línea “\n”. Tomar en cuenta que no todas las traducciones tienen la misma cantidad de palabras, ejemplo: “Action movies” – “Películas de acción”, no deben de quedar espacios vacíos entre palabras.

	Una vez que un texto sea traducido, permanecerá en ese estado hasta que se vuelta a ejecutar la función para volver a traducir.

Tarea 7

Duración: 3 semanas

Inicio: 31 de octubre de 2022

Fin: 21 de noviembre de 2022

Tarea 7	
Nombre	Creación de listas de guardado y favoritos.
Prioridad	Media.
Descripción	Implementar los cambios necesarios en la base de datos para hacer posible que un usuario marque historias como favoritas o las guarde para verlas posteriormente. Estos modelos deben contar con la fecha y hora en que fueron guardados. El usuario debe tener la opción de guardar o marcar como favorita una historia desde la vista de información de la historia. Agregar un link en la barra de navegación para visualizar las historias guardadas y otro para las favoritas, estas deben mostrarse en la misma plantilla que la utilizada en la galería de historias y deben visualizarse ordenadamente por fecha mostrando primero las agregadas recientemente. **Consideraciones:** El usuario debe poder visualizar en la información de la historia si ésta ya está marcada como favorita o guardada. Como precaución, se debe evitar que el usuario abuse de esta funcionalidad, por lo que, debe haber un límite de

	“cambios”. Por ejemplo, que no pueda guardar o marcar como favorita una historia más de 20 veces en un lapso de 10 minutos.

Tarea 8

Duración: 1 semana

Inicio: 7 de noviembre de 2022

Fin: 14 de noviembre de 2022

Tarea 8	
Nombre	Creación de contenido: Diálogo.
Prioridad	Alta.
Descripción	Una historia puede tener muchas páginas, de la misma manera, una página puede tener muchos diálogos. Al momento de crear una página debe aparecer una opción para agregar un nuevo diálogo. Los campos a llenar para crear un diálogo son nombre, texto en inglés, texto en español, número de elemento y color. **Consideraciones:** El número de elemento debe aumentar conforme se agreguen elementos y este número debe servir para mostrarlos ordenadamente en la página. Este contenido debe estar disponible en las plantillas 1 y 3.

Tarea 9

Duración: 1 semana

Inicio: 14 de noviembre de 2022

Fin: 21 de noviembre de 2022

Tarea 9	
Nombre	Creación de ejercicio: Repetir frase.
Prioridad	Alta.
Descripción	Una página puede tener muchos ejercicios, entre ellos, el ejercicio de Repetir frase. Al momento de crear una página debe aparecer una opción para agregar un nuevo ejercicio de Repetir frase. Este ejercicio requiere un texto en inglés, un texto en español, número de elemento y una casilla para saber si se desea mostrar el texto en inglés a la hora de contestarlo. **Consideraciones:** El número de elemento debe aumentar conforme se agreguen elementos y este número debe servir para mostrarlos ordenadamente en la página. Este ejercicio debe estar disponible en todas las plantillas.

Tarea 10

Duración: 1 semana

Inicio: 21 de noviembre de 2022

Fin: 28 de noviembre de 2022

Tarea 10	
Nombre	Diseño de la página para mostrar la información de una historia.
Prioridad	Muy alta.
Descripción	Crear una página para visualizar la información de una historia, esta página se muestra cuando el usuario abre alguna historia de la galería de historias. Implementar un diseño para mostrar todos los detalles de la historia, estos son título, imagen de portada, etiquetas y descripción. Colocar dos botones de tal manera que se vean relevantes, un botón es para continuar contestando la historia y otro para empezar una nueva. **Consideraciones:** La url de la información de la historia debe apoyarse del valor de su slug, ejemplo: /story-info/slug-de-la-historia **Adicional:** Implementar la funcionalidad de traducir texto en todos los textos que lo requieran.

Tarea 11

Duración: 1 semana

Inicio: 28 de noviembre de 2022

Fin: 5 de diciembre de 2022

Tarea 11	
Nombre	Implementar la página para contestar las páginas de una historia de acuerdo a su plantilla.
Prioridad	Muy alta.
Descripción	Esta página debe activarse cuando el usuario de pulse en el botón “Empezar” para empezar a contestar la historia. En la parte superior debe tener el número de la página actual y el número total de páginas que la historia tiene, ejemplo, “1/2” donde “1” indica la página actual y “2” el número de páginas total. Posteriormente se debe renderizar el contenido de la página de acuerdo a su plantilla. Al final debe haber dos botones: • El primer botón es para dejar de contestar y al presionarse debe salir una ventana de confirmación que si es aceptada va a redireccionar al usuario a la página de la información de la historia. • El segundo botón envía y evalúa las respuestas si se está en la última página y de lo contrario sólo las envía. Las funciones de estos botones y sus detalles se realizarán posteriormente.

Tarea 12

Duración: 1 semana

Inicio: 5 de diciembre de 2022

Fin: 12 de diciembre de 2022

Tarea 12	
Nombre	Renderizar los diálogos para contestar páginas de una historia.
Prioridad	Muy alta.
Descripción	Renderizar los diálogos en el lugar que les corresponda de acuerdo a su número de elemento y plantilla dentro de la página para contestar la historia. Deben visualizarse como diálogos entre diferentes personas, es decir, el nombre de la persona seguido de dos puntos y su contenido. Los nombres deben ir en negritas y con el color que fueron guardados.

Tarea 13

Duración: 1 semana

Inicio: 12 de diciembre de 2022

Fin: 19 de diciembre de 2022

Tarea 13	
Nombre	Renderizar los ejercicios de repetir frase para contestar páginas de una historia.
Prioridad	Muy alta.
Descripción	Renderizar los ejercicios de repetir frase en el lugar que les corresponda de acuerdo a su número de elemento y plantilla dentro de la página para contestar la historia. El usuario debe contar con dos botones que podrá presionar para escuchar el contenido en inglés que se introdujo, uno debe pronunciar el contenido a velocidad normal y otro con una velocidad menor. También debe haber una barra de deslizamiento para controlar el volumen de la voz. Adicionalmente debe haber otro botón que permita repetir el texto en inglés que escuchó y guarde su respuesta para una posterior evaluación. El contenido del ejercicio debe de ser traducible siempre y cuando el creador del ejercicio lo haya indicado, de lo contrario, la función de traducción para la parte de inglés mostrará el mensaje “Pulsa el botón para escuchar la frase” en lugar de la traducción.

	Ejemplo con la casilla para mostrar texto activada: *“Aguacate” / “Avocado”* **Ejemplo con la casilla para mostrar texto desactivada:** *“Aguacate” / “Pulsa el botón para escuchar la frase”* **Consideraciones:** Si el editor decide no mostrar el texto en inglés este no debe enviarse desde el servidor, pues el usuario podría encontrar una forma de leerlo y hacer trampa.

Tarea 13.1	
Nombre	Implementar la lógica para contestar los ejercicios de repetir frase.
Prioridad	Muy alta.
Descripción	Utilizar el botón de “enviar respuestas” o ”enviar y evaluar respuestas” para calificar todos los ejercicios del tipo “Repetir frase”. Implementar un algoritmo que califique de manera justa, encontrando un balance proporcional a lo que el usuario contestó bien y mal. El servidor debe devolver retroalimentación de las respuestas una vez termine de calificar.

	Sólo se debe permitir el envío de las respuestas una vez, si el usuario vuelve a recargar la página debe de mostrarse la retroalimentación mencionada y no volver a aceptar más respuestas para su evaluación.

Tarea 14

Duración: 3 semanas

Inicio: 19 de diciembre de 2022

Fin: 9 de enero de 2023

Tarea 14	
Nombre	Creación del modelo para almacenar puntajes en la base de datos.
Prioridad	Alta.
Descripción	Crear un modelo para guardar los puntajes del usuario, debe tener el perfil del usuario que hizo el puntaje, la historia a la que pertenece, la fecha y hora en que fue realizado, el puntaje que el usuario obtuvo, el puntaje límite que se podía obtener, el porcentaje total, el porcentaje en escritura, el porcentaje en comprensión y el porcentaje en pronunciación. • Se debe establecer un valor en puntos para cada ejercicio. • Contestar perfectamente un ejercicio otorgará todos sus puntos posibles, así mismo una respuesta parcial debe obtener puntos parciales y no puede haber puntos negativos. • Cada ejercicio suma puntos en al menos una categoría ya sea escritura, comprensión y/o pronunciación; si el usuario contestó perfectamente se suma un 1 y de lo contrario se elige un valor que se considere justo acorde a la respuesta proporcionada, siendo 0 el valor mínimo permitido.

	El **puntaje límite** es la cantidad máxima de puntos que se pueden obtener. El **puntaje obtenido** es la sumatoria de todos los puntos que el usuario obtuvo contestando. El **porcentaje total** es la división del puntaje obtenido sobre el puntaje límite. El **porcentaje en escritura** es la cantidad total de puntos en ejercicios de escritura sobre el total de ejercicios de escritura, de no haber ejercicios será 1. El **porcentaje en comprensión** es la cantidad total de puntos en ejercicios de comprensión sobre el total de ejercicios de comprensión, de no haber ejercicios será 1. El **porcentaje en pronunciación** es la cantidad total de puntos en ejercicios de pronunciación sobre el total de ejercicios de pronunciación, de no haber ejercicios será 1. **Consideraciones:** Notar que los puntos por respuesta y los puntos por categoría no son los mismos. Calificar al usuario con un poco de tolerancia para no desmotivarlo con bajas calificaciones si es que llega a tener errores.

Tarea 14.1	
Nombre	Implementación de puntajes en forma de gráficas.

Prioridad	Alta.
Descripción	Dentro de la página de información de la historia, debajo de los botones para empezar o continuar, mostrar la mejor puntuación del usuario en forma de gráficas. Las gráficas requeridas son “Escritura”, “Comprensión” y “Pronunciación”, mismas que deben presentarse como barras de progreso rectas con los nombres mencionados cuya traducción es “Writing”, “Comprehension” y “Speaking” respectivamente. Adicionalmente se requiere una barra de progreso circular para el porcentaje de puntos la cual en el centro debe contener una calificación del desempeño del estudiante. La calificación debe ser una letra donde S+ indica un desempeño sobresaliente y posteriormente S, S-, A+, A, A-, B+, B, B-, C+, C y C- indicando un mayor o un menor desempeño respectivamente. Todas las gráficas deben tener una animación de progreso donde lleguen hasta el número correspondiente de puntos. **Consideraciones:** El mejor puntaje es el puntaje más antiguo con mayor valor en “porcentaje total”.

Tarea 14.2

Nombre	Implementación de puntajes en forma de tabla.
Prioridad	Alta.
Descripción	Dentro de la página de información de la historia, debajo de las gráficas de la mejor puntuación, mostrar una tabla con las mejores puntuaciones del usuario. La tabla debe llevar por título “Tabla de puntuaciones” / “Scores table” y las columnas “Posición”, “Puntaje” y “Fecha”. Se deben ordenar los datos de mayor a menor valor en “porcentaje total”, en la columna de posición debe ir el número de posición del puntaje empezando por “#1”, “#2” y “#3” sucesivamente, en la columna de puntaje debe ir el porcentaje total y en la columna de la fecha debe ir la fecha y hora del puntaje en un formato similar a “14/06/2022 7:25 p.m.”

Tarea 14.3	
Nombre	Incorporación del ejercicio repetir frase a estadísticas.
Prioridad	Alta.
Descripción	Modificar la evaluación de estos ejercicios para crear las estadísticas correspondientes de la historia. Este ejercicio aumenta las estadísticas de pronunciación.

Tarea 15

Duración: 1 semana

Inicio: 9 de enero de 2023

Fin: 16 de enero de 2023

Tarea 15	
Nombre	Incorporar una página de contenido no encontrado
Prioridad	Media
Descripción	Implementar una vista de “Página no encontrada” y su funcionalidad. El usuario debe ser redirigido a esta página cuando intente acceder a contenido inexistente.

Tarea 16

Duración: 1 semana

Inicio: 16 de enero de 2023

Fin: 23 de enero de 2023

Tarea 16	
Nombre	Implementar sistema de nivel y experiencia
Prioridad	Alta
Descripción	Modificar lo necesario para obtener el siguiente comportamiento. Al finalizar una historia esta debe de sumar puntos de experiencia al usuario, estos puntos servirán para subir de nivel y desbloquear nuevas historias. Implementar un algoritmo para asignar un nivel con base a la experiencia acumulada del usuario, el nivel debe ser un número entero. Cada historia tiene una cantidad de experiencia para desbloquearse, en la galería de historias sólo deben aparecer las historias que un usuario esté autorizado para contestar, es decir, que su experiencia sea mayor o igual a la requerida. Si el usuario intenta acceder a una historia y no cumple los requisitos, este debe ser redireccionado a la galería de historias. **Consideraciones:**

	Subir de nivel debe significar un reto para el usuario, sin embargo, hay que tomar en cuenta el riesgo de desmotivar al usuario si hace mucho esfuerzo y no sube, por lo que se debe implementar un algoritmo para alcanzar el siguiente nivel controladamente sin que sea demasiado fácil o demasiado difícil.

Tarea 17

Duración: 2 semanas

Inicio: 23 de enero de 2023

Fin: 6 de febrero de 2023

Tarea 17	
Nombre	Agregar una página para ver el perfil del usuario
Prioridad	Media
Descripción	Crear una página para ver el perfil de un usuario. Un usuario debe tener una foto de un avatar predefinido, misma que se debe mostrar en un círculo dentro de una barra de progreso circular, esta barra de progreso debe indicar el nivel del usuario. También debe mostrarse el nombre del usuario y enlaces apuntando a la colección de historias favoritas y a la colección de historias guardadas. Debajo crear otra sección para mostrar las estadísticas del usuario mostradas en forma de barras de progreso circular, similar a como se muestra en la página de información de la historia, estas barras de progreso deben ser escritura, comprensión y pronunciación. El valor para cada barra de progreso será el promedio en su respectiva estadística de las mejores y únicas historias contestadas por el usuario. Posteriormente, implementar una sección más que muestre una tabla con los mejores puntajes que el usuario tiene registrado por cada historia, es decir, ordenando

	primero aquellos con un valor alto en la columna “porcentaje total”.

Tarea 17.1	
Nombre	Modificación de enlaces en la barra de navegación
Prioridad	Media
Descripción	Modificar el enlace a la página del perfil de usuario para lo siguiente. El acceso debe ser mediante un link en la barra de navegación que tendrá el mismo estilo que la barra de progreso circular de la página mencionada, pero en un tamaño pequeño que le permita verse estético en la barra de navegación. Este enlace también debe funcionar como menú desplegable en donde se encontrarán opciones como “Favoritos”, “Guardados” y “Cerrar sesión”, moviendo estos enlaces localizados en la barra de navegación como opciones dentro del menú.

Tarea 18

Duración: 1 semana

Inicio: 6 de febrero de 2023

Fin: 13 de febrero de 2023

Tarea 18	
Nombre	Creación de ejercicio: Revisar ortografía.
Prioridad	Muy alta.
Descripción	Una página puede tener muchos ejercicios, entre ellos, el ejercicio de Repetir ortografía. Al momento de crear una página debe aparecer una opción para agregar un nuevo ejercicio de Revisar ortografía. Este ejercicio requiere un número de elemento, un texto en inglés, un texto en español y otro texto en inglés, pero con errores de ortografía. **Consideraciones:** El número de elemento debe aumentar conforme se agreguen elementos y este número debe servir para mostrarlos ordenadamente en la página. Este ejercicio debe estar disponible en todas las plantillas.

Tarea 19

Duración: 2 semanas

Inicio: 13 de febrero de 2023

Fin: 27 de febrero de 2023

Tarea 19	
Nombre	Renderizar los ejercicios de revisar ortografía para contestar páginas de una historia.
Prioridad	Muy alta.
Descripción	Renderizar los ejercicios de revisar ortografía en el lugar que les corresponda de acuerdo a su número de elemento y plantilla dentro de la página para contestar la historia. En este ejercicio se debe poder visualizar un texto erróneo en inglés con su traducción correcta al español, el usuario debe contar con un campo de texto para poder escribir el texto en inglés correctamente. Adicionalmente, se deben poner etiquetas que ayuden al usuario a identificar qué es lo que tiene que escribir. **Ejemplo:** *“Fix the text: A loong time ago…” / “Traducción: Hace mucho tiempo…”*

Tarea 19.1	
Nombre	Implementar la lógica para contestar los ejercicios de revisar ortografía.

Prioridad	Muy alta.
Descripción	Utilizar el botón de “enviar respuestas” o ”enviar y evaluar respuestas” para calificar todos los ejercicios del tipo “Revisar ortografía”. Implementar un algoritmo que califique de manera justa, encontrando un balance proporcional a lo que el usuario contestó bien y mal. El servidor debe devolver retroalimentación de las respuestas una vez termine de calificar. Sólo se debe permitir el envío de las respuestas una vez, si el usuario vuelve a recargar la página debe de mostrarse la retroalimentación mencionada y no volver a aceptar más respuestas para su evaluación.

Tarea 19.2	
Nombre	Incorporación del ejercicio revisar ortografía a estadísticas.
Prioridad	Muy alta.
Descripción	Modificar la evaluación de estos ejercicios para crear las estadísticas correspondientes de la historia. Este ejercicio aumenta las estadísticas de escritura.

Tarea 20

Duración: 1 semana

Inicio: 27 de febrero de 2023

Fin: 6 de marzo de 2023

Tarea 20	
Nombre	Creación de ejercicio: Opción múltiple.
Prioridad	Muy alta.
Descripción	Una página puede tener muchos ejercicios, entre ellos, el ejercicio de Opción múltiple. Al momento de crear una página debe aparecer una opción para agregar un nuevo ejercicio de Opción múltiple. Este ejercicio requiere un número de elemento, un texto en inglés, un texto en español, una casilla “mostrar aleatorio” y una o muchas opciones de las cuales sólo una puede ser correcta. Una opción debe tener un número de opción un texto en inglés, un texto en español y un campo para saber si es la opción correcta. **Consideraciones:** Los números de elemento y de opción deben aumentar conforme se agreguen elementos y su función es mostrarlos ordenadamente en la página. Este ejercicio debe estar disponible en todas las plantillas.

Tarea 21

Duración: 2 semanas

Inicio: 6 de marzo de 2023

Fin: 20 de marzo de 2023

Tarea 21	
Nombre	Renderizar los ejercicios de opción múltiple para contestar páginas de una historia.
Prioridad	Muy alta.
Descripción	Renderizar los ejercicios de opción múltiple en el lugar que les corresponda de acuerdo a su número de elemento y plantilla dentro de la página para contestar la historia. En este ejercicio se debe poder visualizar un texto en inglés con su traducción al español y una lista de todas sus opciones de las cuáles sólo puede elegir una. Si la casilla para mostrar en aleatorio está activada, las opciones deben mostrarse en orden aleatorio.

Tarea 21.1	
Nombre	Implementar la lógica para contestar los ejercicios de opción múltiple.
Prioridad	Muy alta.
Descripción	Utilizar el botón de “enviar respuestas” o ”enviar y evaluar respuestas” para calificar todos los ejercicios del tipo “Opción múltiple”.

	El servidor debe devolver retroalimentación de las respuestas una vez termine de calificar, es decir, se debe indicar la respuesta correcta. Sólo se debe permitir el envío de las respuestas una vez, si el usuario vuelve a recargar la página debe de mostrarse la retroalimentación mencionada y no volver a aceptar más respuestas para su evaluación.

Tarea 21.2	
Nombre	Incorporación del ejercicio revisar ortografía a estadísticas.
Prioridad	Muy alta.
Descripción	Modificar la evaluación de estos ejercicios para crear las estadísticas correspondientes de la historia. Este ejercicio aumenta las estadísticas de comprensión.

Tarea 22

Duración: 1 semana

Inicio: 20 de marzo de 2023

Fin: 27 de marzo de 2023

Tarea 22	
Nombre	Creación de contenido y renderizado: Imagen.
Prioridad	Muy alta.
Descripción	Una página puede tener muchos contenidos, entre ellos, el contenido de Imagen. Al momento de crear una página debe aparecer una opción para agregar un nuevo contenido de Imagen. Este contenido requiere un número de elemento y un campo para guardar la imagen, además se debe renderizar de acuerdo a su plantilla. **Consideraciones:** El número de elemento debe aumentar conforme se agreguen elementos y su función es mostrarlos ordenadamente en la página. Tener en cuenta que el archivo de imagen se debe borrar del almacenamiento cuando se desee eliminar este contenido, página o historia a la que pertenece. Este contenido debe estar disponible en todas las plantillas.

Tarea 23

Duración: 1 semana

Inicio: 27 de marzo de 2023

Fin: 3 de abril de 2023

Tarea 23	
Nombre	Creación de contenido y renderizado: Texto.
Prioridad	Muy alta.
Descripción	Una página puede tener muchos contenidos, entre ellos, el contenido de Texto. Al momento de crear una página debe aparecer una opción para agregar un nuevo contenido de Texto. Este contenido requiere un número de elemento, un texto en inglés y un texto en español, además se debe renderizar de acuerdo a su plantilla. **Consideraciones:** El número de elemento debe aumentar conforme se agreguen elementos y su función es mostrarlos ordenadamente en la página. El texto debe ser traducible al momento de renderizar. Este contenido debe estar disponible en todas las plantillas.

Tarea 24

Duración: 1 semana

Inicio: 3 de abril de 2023

Fin: 10 de abril de 2023

Tarea 24	
Nombre	Creación de contenido y renderizado: Video.
Prioridad	Muy alta.
Descripción	Una página puede tener muchos contenidos, entre ellos, el contenido de Video. Al momento de crear una página debe aparecer una opción para agregar un nuevo contenido de Video. Este contenido requiere un número de elemento y una dirección URL a un video de Youtube. **Consideraciones:** El número de elemento debe aumentar conforme se agreguen elementos y su función es mostrarlos ordenadamente en la página. Este contenido debe estar disponible en la plantilla de “video y ejercicios” y en la de “diseño libre”. La plantilla de “video y ejercicios” sólo debe permitir un video.

Tarea 25

Duración: 1 semana

Inicio: 10 de abril de 2023

Fin: 17 de abril de 2023

Tarea 25	
Nombre	Creación de contenido y renderizado: Audio.
Prioridad	Muy alta.
Descripción	Una página puede tener muchos contenidos, entre ellos, el contenido de Audio. Al momento de crear una página debe aparecer una opción para agregar un nuevo contenido de Audio. Este contenido requiere un número de elemento, un archivo de audio, una etiqueta para nombrar el audio, una descripción en inglés, una descripción en español y una casilla para saber si se desea mostrar la descripción al usuario. **Consideraciones:** El número de elemento debe aumentar conforme se agreguen elementos y su función es mostrarlos ordenadamente en la página. La etiqueta para nombrar el audio es útil solo en modo edición para identificar el audio, no se debe mostrar al usuario y debe implementarse una forma para advertir al editor de esta característica.

	El texto debe ser traducible al momento de renderizar. La descripción al momento de renderizar debe ser traducible y sólo mostrarse si el editor lo indicó. Tener en cuenta que el archivo de audio se debe borrar del almacenamiento cuando se desee eliminar este contenido, página o historia a la que pertenece. Este contenido debe estar disponible en la plantilla de diseño libre.

Tarea 26

Duración: 2 semanas

Inicio: 17 de abril de 2023

Fin: 8 de mayo de 2023

Tarea 26	
Nombre	Implementar un sistema de recomendación de historias en base a contenido.
Prioridad	Media.
Descripción	Realizar un sistema de recomendación en base a contenido. Investigar e implementar las funciones necesarias para crear un sistema de recomendación que funcione en base a contenido, es decir, que recomiende a un usuario historias similares a otras. **Consideraciones:** Se recomienda utilizar los datos de la descripción y las etiquetas.

Tarea 27

Duración: 3 semanas

Inicio: 8 de mayo de 2023

Fin: 29 de mayo de 2023

Tarea 27	
Nombre	Implementar un sistema de recomendación de historias basado en un filtro colaborativo usuario a usuario.
Prioridad	Media.
Descripción	Realizar un sistema de recomendación colaborativo usuario a usuario. Investigar e implementar las funciones necesarias para crear un sistema de recomendación colaborativo que funcione de usuario a usuario, es decir, que recomiende historias basadas en su interacción y la de otros usuarios con interacciones similares. **Consideraciones:** Se recomienda utilizar los datos de las historias favoritas.

Tarea 28

Duración: 2 semanas

Inicio: 29 de mayo de 2023

Fin: 12 de junio de 2023

Tarea 28	
Nombre	Creación de modelo para colecciones de flashcards y flashcards.
Prioridad	Baja.
Descripción	Las “tarjetas de estudio” también conocidas como “flashcards” tienen la finalidad de brindar una experiencia amigable al usuario, permitiéndole hacer notas para llevar un registro de sus conocimientos. Un usuario puede tener muchas colecciones de flashcards, a su vez una colección de flashcards puede tener muchas flashcards. Crear una sección para que un usuario pueda crear colecciones de flashcards y flashcards. Una colección de flashcards debe tener un color, nombre, descripción, fecha de creación y fecha de actualización. Una “flashcard” debe tener un color, un texto frontal, un texto posterior, fecha de creación y fecha de actualización. **Consideraciones:**

	Un usuario sólo puede acceder a sus propias tarjetas de estudio, dicho de otra manera, no puede acceder a tarjetas de otros usuarios.

Tarea 28.1	
Nombre	Página para colecciones de flashcards y flashcards.
Prioridad	Baja.
Descripción	Agregar un link en la barra de navegación para acceder a la página donde se podrán administrar colecciones y flashcards. El usuario debe poder ver una lista con todas las colecciones que ha creado, donde se muestra su título y descripción con el color de fondo que se guardó. El usuario puede hacer click en alguna colección de flashcards para ver todas sus flashcards o crear nuevas. Los flashcards también deben mostrarse con el color de fondo que el usuario elija, mostrando primero el texto frontal y su texto posterior al ser presionadas.

Tarea 29

Duración: 3 semanas

Inicio: 12 de junio de 2023

Fin: 3 de julio de 2023

Tarea 29	
Nombre	Modificación de permisos y gestión de usuarios.
Prioridad	Alta.
Descripción	Modificar la aplicación para restringir accesos de la siguiente forma. Se requieren dos principales roles, super usuario y staff, un usuario con permisos de staff puede agregar, modificar y eliminar historias y etiquetas mientras que un usuario con el rol de super usuario tiene acceso a más recursos.

Tarea 29.1	
Nombre	Crear una sección para administrar usuarios.
Prioridad	Alta.
Descripción	Crear una nueva página para administrar usuarios, esta página sólo debe ser accesible para usuarios con rol de super usuario. Esta sección debe permitir ver todos los usuarios registrados y sus roles disponibles, el super usuario debe poder agregar o eliminar roles y eliminar usuarios.

Tarea 29.2	
Nombre	Crear una sección para administrar los sistemas de recomendación.
Prioridad	Alta.
Descripción	Crear una nueva página para administrar los sistemas de recomendación, esta página sólo debe ser accesible para usuarios con rol de super usuario. Esta sección debe permitir configurar valores esenciales para el funcionamiento de los sistemas de recomendación e incluir un botón para actualizar manualmente cada uno de los sistemas establecidos. Sistema de recomendación basado en contenido: opción de actualizar automáticamente cada cierta cantidad de segundos y opción para actualizar automáticamente cada que hay cambios en las historias. Sistema de recomendación usuario a usuario: opción de actualizar automáticamente cada cierta cantidad de segundos. **Consideraciones:** Cuando la aplicación inicie por primera vez debe crear los registros correspondientes para guardar la configuración por defecto.

Tarea 30

Duración: 6 semanas

Inicio: 3 de julio de 2023

Fin: 14 de agosto de 2023

Tarea 30	
Nombre	Creación de un sistema administrador de reportes para identificar posibles problemas en las historias.
Prioridad	Baja.
Descripción	Crear un sistema en tiempo real que permita enviar reportes de una historia. Agregar un botón en la sección de información de cada historia que permita enviar un reporte de una historia. Cuando un usuario envíe el reporte debe aparecer un mensaje a modo de notificación con el texto “Gracias por sus comentarios”. **Consideraciones:** Todas las solicitudes deben ser AJAX con el motivo de actualizar únicamente las partes necesarias del código una vez enviada la solicitud y obtenida una respuesta.

Tarea 30.1	
Nombre	Creación del modelo para almacenar reportes en la base de datos.

Prioridad	Baja.
Descripción	Un reporte debe pertenecer a una historia, almacenar el perfil del usuario que hace el reporte, tener una descripción, un estado de reporte, una fecha y hora en que el reporte ha sido creado, una fecha y hora en la que el reporte ha sido actualizado, una fecha y hora en que el reporte ha sido arreglado y adicionalmente puede pertenecer a una página de una historia. El estado del reporte debe tomar como valor “no leído”, “leído”, “en progreso” o “arreglado”.

Tarea 30.2	
Nombre	Creación de una sección para administrar los reportes.
Prioridad	Baja.
Descripción	Agregar un enlace en la barra de navegación que redireccione hacia una página para administrar los reportes, esta página debe estar disponible para usuarios con rol de staff o super usuario. Esta página debe: Contener un botón para activar o desactivar el funcionamiento en tiempo real. Contener un botón para actualizar el contenido manualmente. Tener un filtro de los reportes por su estado.

	Los reportes deben visualizarse en forma de una tabla mostrando una casilla para seleccionar el reporte, la historia a la que pertenece, los primeros 20 caracteres de la descripción y la fecha de emisión del reporte. La fecha de emisión del reporte debe estar en formato “número-del-día nombre-recortado-del-mes”, este efecto se puede conseguir con el filtro |date:”d M” de Django, por ejemplo, “14 Ago” o “03 Sep”. Los reportes no leídos deben diferenciarse del resto poniéndolos con letras en negritas, al presionar click en un reporte, este debe abrirse y debe cambiar su estado a “leído” si previamente se encontraba en “no leído”. La tabla de reportes debe contener un botón en la parte superior que abra un menú contextual con las opciones “Marcar como no leído”, “Marcar como leído”, “Marcar en revisión”, “Marcar como arreglado” y “Eliminar”, estos botones realizarán la acción adecuada para todos los reportes que se encuentren seleccionados al momento de enviar la solicitud. **Adicional:** Se recomienda poner como tamaño de alto de la tabla 60vh, si el contenido de los reportes excede este tamaño, deben mostrarse en un scroll.

Tarea 30.3	
Nombre	Implementación cuando un reporte es abierto.
Prioridad	Baja.
Descripción	Cuando un reporte es abierto, el contenido de la tabla debe ser reemplazado por el contenido del reporte, mostrando en la parte superior el nombre de la historia con el prefijo “Historia: ” y un ícono de estado para saber el estado del reporte. Debajo de esto debe mostrar las fechas completas de creación y actualización con el prefijo “Emitido: ” para la fecha de creación y con el prefijo “Modificado: ” para la fecha de actualización. El reporte abierto debe mostrar la página a la que pertenece y de no contener alguna mostrar el texto “No especificada”. Por último, mostrar la descripción del reporte. **Ejemplo:** “Emitido: 14 de agosto de 2023 a las 19:31”. “Modificado: 23 de agosto de 2023 a las 21:00”. **Adicional:** Cuando el menú contextual es abierto, si se selecciona “Marcar como no leído” u alguna otra opción, debe afectar únicamente al reporte abierto. Si el contenido del reporte excede el tamaño del alto de la tabla, debe mostrarse con un scroll.

Tarea 30.4	
Nombre	Cargar información de un reporte mediante query params en la url.
Prioridad	Baja.
Descripción	Agregar y eliminar automáticamente los query params que sean necesarios para cubrir los siguientes casos. Al abrir un reporte se debe actualizar la url actual con query params, si se recarga la página los query params sirven para no perder el reporte que se tenía abierto, pues deberán cargar el reporte correspondiente con estos. **Consideraciones:** Este comportamiento debe lograrse utilizando JavaScript.

CAPÍTULO IV

IMPLEMENTACIÓN

En este apartado se muestra el proceso seguido para la realización de la aplicación, realizando las tareas meticulosamente diseñadas, producto de las historias de usuario y siguiendo la filosofía de la metodología ágil Scrum.

A su vez, a lo largo de este capítulo se detallan algunos de los principales problemas que surgieron durante la realización de la aplicación, junto con las soluciones desarrolladas para superarlos.

4.1 Elección de tecnologías

La primera tarea consta de elegir las tecnologías adecuadas, en ese entonces no tenía mucha experiencia en el diseño web así que mi estrategia fue probar diferentes frameworks de diseño web, entonces elegir aquél más apropiado conforme a los requisitos establecidos de la aplicación.

Decidí hacer pequeñas aplicaciones con Node.js, Spring Boot y Django, sin embargo, decidí utilizar Django por su fácil curva de aprendizaje y la ventaja de estar desarrollado en Python, pues este lenguaje consta con una amplia variedad de librerías para inteligencia artificial y ciencia de datos, mismas que me ayudarían en un futuro a crear los sistemas de recomendación.

4.2 Autenticación

Django provee funcionalidades que ayudan a guardar los datos de un usuario en el navegador, por lo que, a diferencia de otros framework, no se necesita generar y guardar un token manualmente en la memoria del navegador, en las figuras 4.2.1 y 4.2.2 observamos las páginas de inicio de sesión y registro respectivamente.

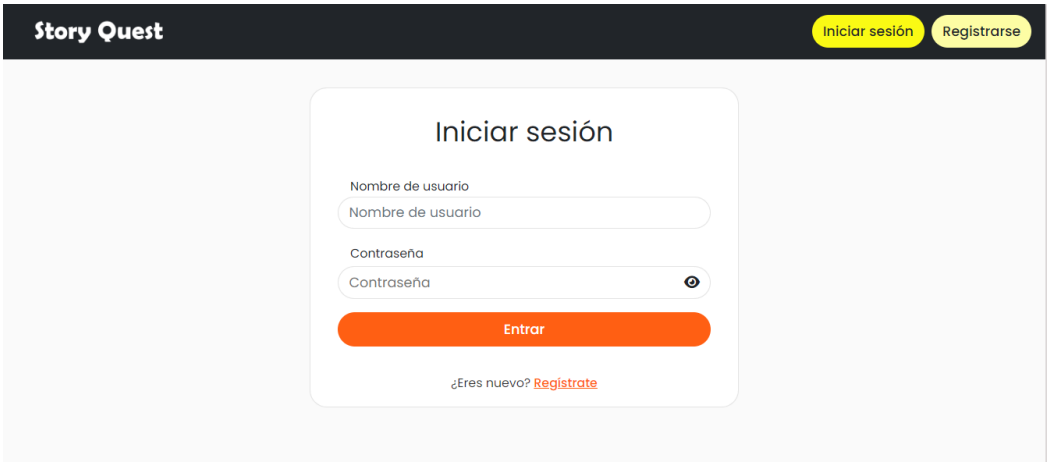

Figura 4.2.1. Página de inicio de sesión.

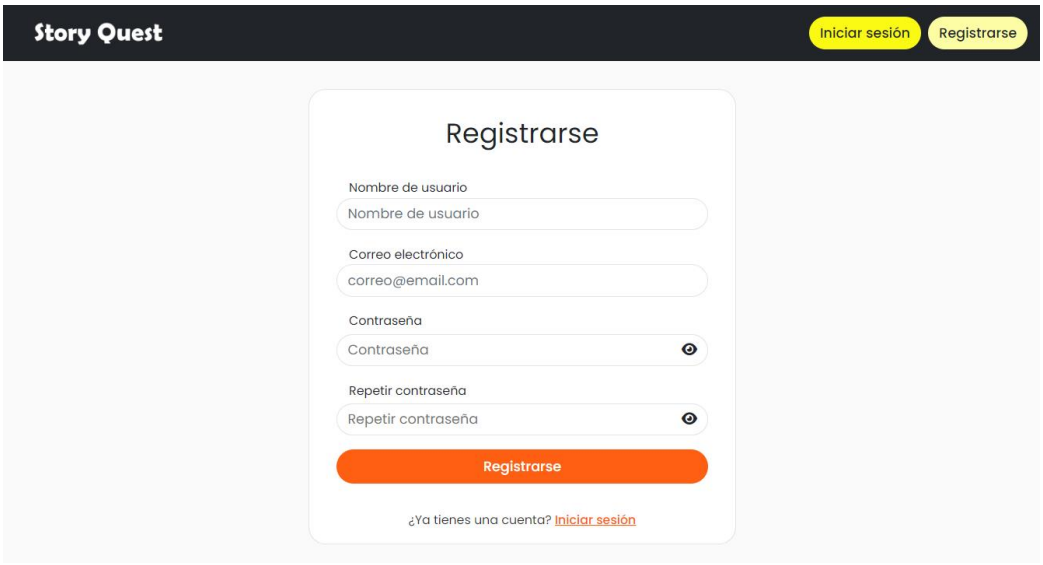

Figura 4.2.2. Página de registro.

No hubo dificultades al crear estas páginas pues siguen el flujo básico de Django, primero se creó una URL y se le asignó una vista, que a su vez contiene la lógica para mandar los datos necesarios a la plantilla que se encarga de mostrarnos de manera gráfica la página.

El siguiente fragmento de código muestra la creación de diversas rutas en Django, se crea estableciendo una dirección y una función para ejecutar cuando se reciba una solicitud apuntando.

```
from django.contrib import admin
from django.urls import path, include
from rest_framework.authtoken import views
from main.views.auth_views import Login, Logout
from main.views.views import test_not_found

from django.contrib.staticfiles.urls import staticfiles_urlpatterns

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls')),
    path('api/redirect/', include(('main.urls', 'main'))),
    path('login/', Login.as_view(), name='login'),
    path('logout/', Logout.as_view(), name='logout'),
    path('api_generate_token/', views.obtain_auth_token),
    path('404/', test_not_found),
]

urlpatterns += staticfiles_urlpatterns()

handler404 = 'main.views.views.not_found_page'
```

En el siguiente ejemplo se muestra la clase “Login” encargada del inicio de sesión, esta vista recoge los datos ‘username’ y ‘password’ para autenticar al usuario y renderiza la plantilla ‘no-logged/login.html’ en caso de un inicio de sesión exitoso.

```
class Login(FormView):
    template_name = 'no-logged/login.html'
    form_class = AuthenticationForm
    success_url = reverse_lazy('index')

    @method_decorator(csrf_protect)
    @method_decorator(never_cache)
    def dispatch(self, request, *args, **kwargs):
        if request.user.is_authenticated:
            return HttpResponseRedirect(self.get_success_url())
        else:
```

```
            return super(Login, self).dispatch(request, *args, *kwargs)

    def form_valid(self, form):
        username = form.cleaned_data['username']
        password = form.cleaned_data['password']
        user = authenticate(username=username, password=password)
        token, _ = Token.objects.get_or_create(user=user)
        if token:
            login(self.request, form.get_user())
            return super(Login, self).form_valid(form)
```

A su vez, la plantilla ‘no-logged/login.html’ contiene el formulario para iniciar sesión, el cual se muestra a continuación.

```
<form class="form-body" autocomplete="off" method="post">
      {% csrf_token %}

      <h1 class="form-title">Iniciar sesión</h1>

      <div class="form-element">
        <label for="username">Nombre de usuario</label>
        <input id="username" name="username" type="text" class="form-control
input-text" style=""
          placeholder="Nombre de usuario" required>
      </div>

      <div class="form-element">
        <label for="password" class="mb-1">Contraseña</label>
        <div class="form-control password-input">
          <input id="password" name="password" type="password"
placeholder="Contraseña" required>
          <span id="toggle_password_visibility" title="Mostrar contraseña"><i
class="fa fa-eye"></i></span>
        </div>
      </div>

      <div class="form-element">
        <button type="submit" class="form-submit-btn">Entrar</button>
      </div>

      <div class="form-element">
        <p class="text-footer">¿Eres nuevo? <a href="{% url 'register'
%}">Regístrate</a></p>
      </div>
```

```
    </form>
```

4.3 Creación de historias y etiquetas

La lógica de la aplicación se basa en historias, para esto es necesario generar las tablas correspondientes en la base de datos. Para la autenticación de usuarios no se realizó este procedimiento porque Django ya incluye tablas por defecto para guardar los datos de sesión.

Para agregar otras tablas basta con crear un modelo e indicarle las columnas con sus tipos de dato necesarios, en el siguiente fragmento de código se ilustra el código de la clase “Tag” y la clase “Story”, para guardar los datos de etiquetas e historias.

```
# Tag
class Tag(models.Model):
    class Meta:
        db_table = prefix + 'tag'

    # keys
    id = models.AutoField(primary_key=True)
    # fields
    name1 = models.CharField(max_length=100)
    name2 = models.CharField(max_length=100)

    def __str__(self) -> str:
        return self.name1

# Story
class Story(models.Model):
    class Meta:
        db_table = prefix + 'story'

    # primary key
    id = models.AutoField(primary_key=True)
    # fields
    title1 = models.CharField(max_length=255)
    title2 = models.CharField(max_length=255)
```

```
    cover = models.ImageField(upload_to=settings.IMAGES_PATH,
default=default_cover_img_path, verbose_name='cover')
    description1 = models.CharField(max_length=255, null=False, blank=True,
default='')
    description2 = models.CharField(max_length=255, null=False, blank=True,
default='')

    xp_required = models.IntegerField(default=0)

    tags = models.ManyToManyField(Tag, blank=True)

    likes_number = models.IntegerField(default=0)
    route = models.SlugField(max_length=255, unique=True, null=False,
default='')
    created_at = models.DateTimeField(auto_now_add=True, null=True)
    updated_at = models.DateTimeField(auto_now=True, null=True)
```

En el código anterior se muestran todos los datos necesarios para la creación de la entidad “Story” que almacenará los datos de las historias, así como la entidad “Tag” encargada de almacenar las etiquetas.

Cabe resaltar algunos aspectos aquí, como las relaciones. En Django hacer relaciones es relativamente fácil pues al igual que una columna, sólo se debe escribir el tipo de dato que es, en este caso la entidad “Story” cuenta con un campo llamado “tags” y se le indica que tiene una relación muchos a muchos con la entidad “Tag”, de esta manera es posible relacionar fácilmente a las entidades.

En las figuras 4.3.1 y 4.3.2 podemos observar el modo administrador que la aplicación tiene, disponible para poder crear y editar contenido, como las historias y etiquetas.

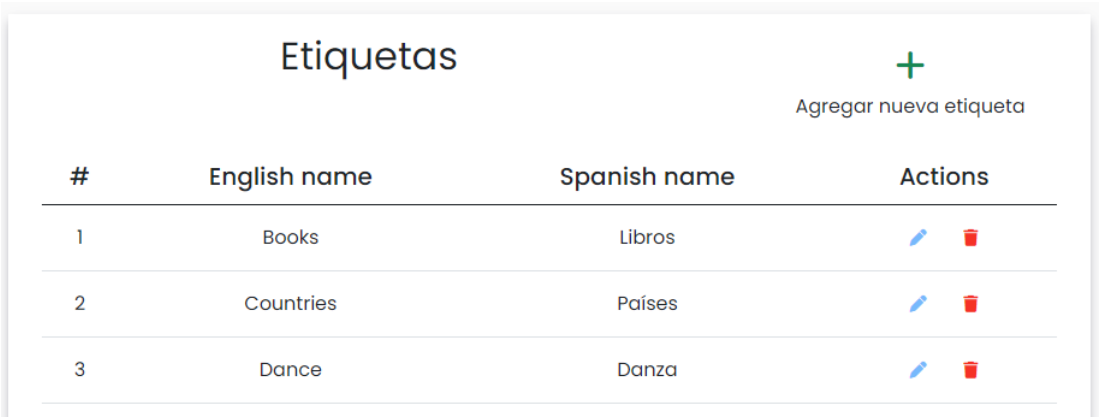

Figura 4.3.1. Pantalla de administración de etiquetas. Muestra una tabla con botones para editar o eliminar etiquetas, además tiene un botón que permite la creación de nuevas, sólo usuarios con permisos de staff pueden acceder aquí.

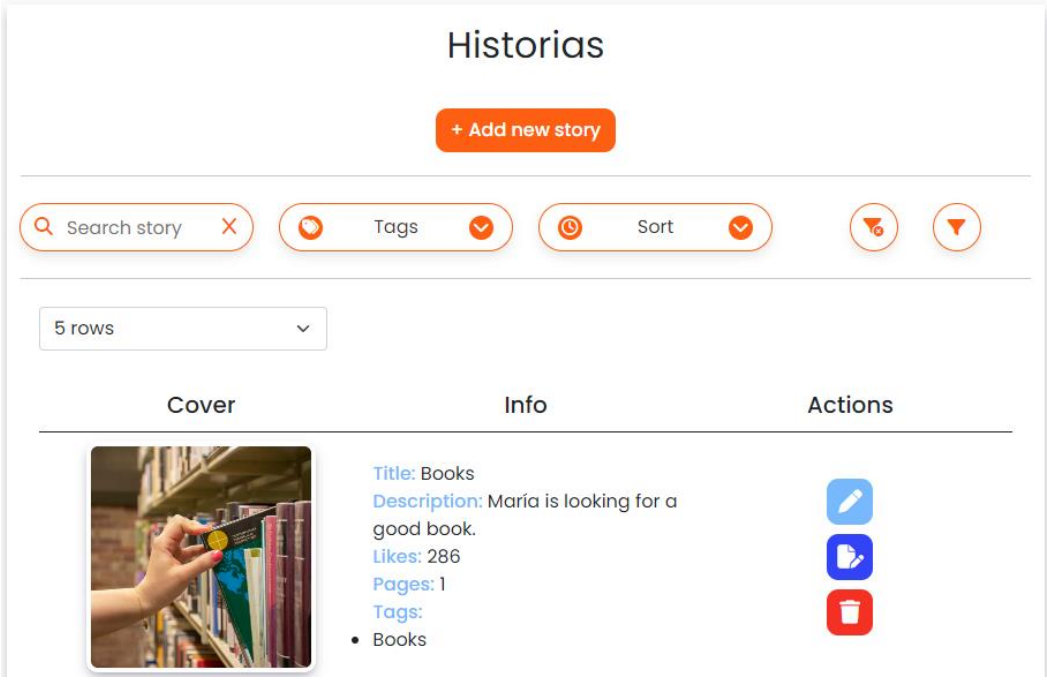

Figura 4.3.2. Pantalla de administración de historias. Muestra botones para editar o eliminar una historia, otro botón para editar las páginas de la historia, y en lo alto un botó para agregar una nueva historia.

Figura 4.3.3. Formulario para la creación de una nueva etiqueta.

Figura 4.3.4. Creación de una nueva historia (1).

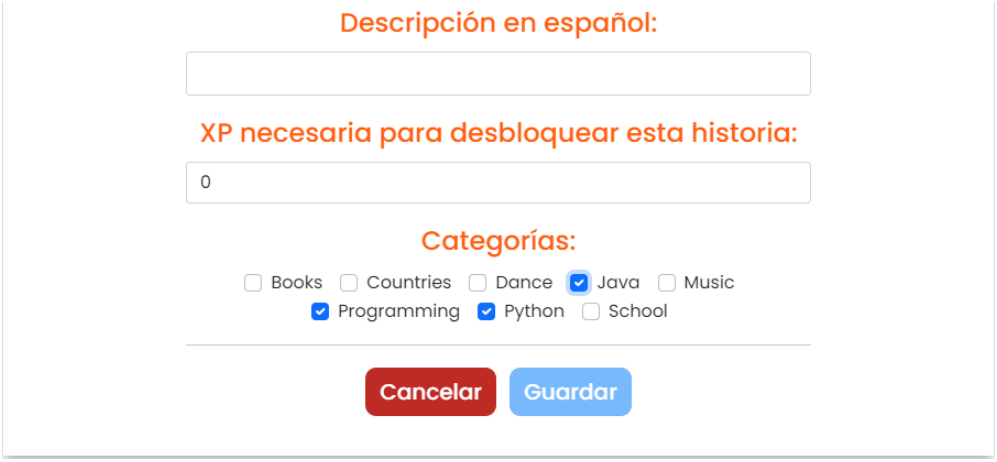

Figura 4.3.5. Creación de una nueva historia (2).

4.3.1 Eliminación de archivos con django-cleanup

Gracias a las relaciones de Django, la relación de los registros de etiquetas con historias es automática, de tal manera que si una etiqueta es actualizada o eliminada simplemente se actualizará o eliminará de todas las historias en las que esté relacionada, sin embargo, este comportamiento no sucedía con las imágenes que eran subidas para la portada de la historia.

Una solución era eliminar directamente los archivos cuando su entidad principal, en este caso la de la historia, eliminara, cambiara la imagen o fuera eliminada de la base de datos. Aplicar este comportamiento es una solución válida, no obstante, continué investigando y resultado de mis investigaciones fue hallar una librería llamada “django-cleanup”.

Esta librería maneja automáticamente la tarea de eliminar archivos que son reemplazados por otros o eliminarlos si su registro principal ha sido eliminado, se instala mediante el comando pip, siendo este “pip install django-cleanup”. Adicionalmente se debe agregar esta aplicación en el archivo “settings.py” como se muestra en el siguiente fragmento de código.

```
# Application definition
INSTALLED_APPS = [
    …
    'django_cleanup.apps.CleanupConfig',
    …
]
```

4.4 Pantallas de historia e información de la historia

Una vez que historias y etiquetas fueron creadas, se necesitaba una manera de mostrarla al usuario, es por eso que se optó por una página que funciona como una galería de historias y otra para mostrar la información de esta respecto al usuario en cuestión.

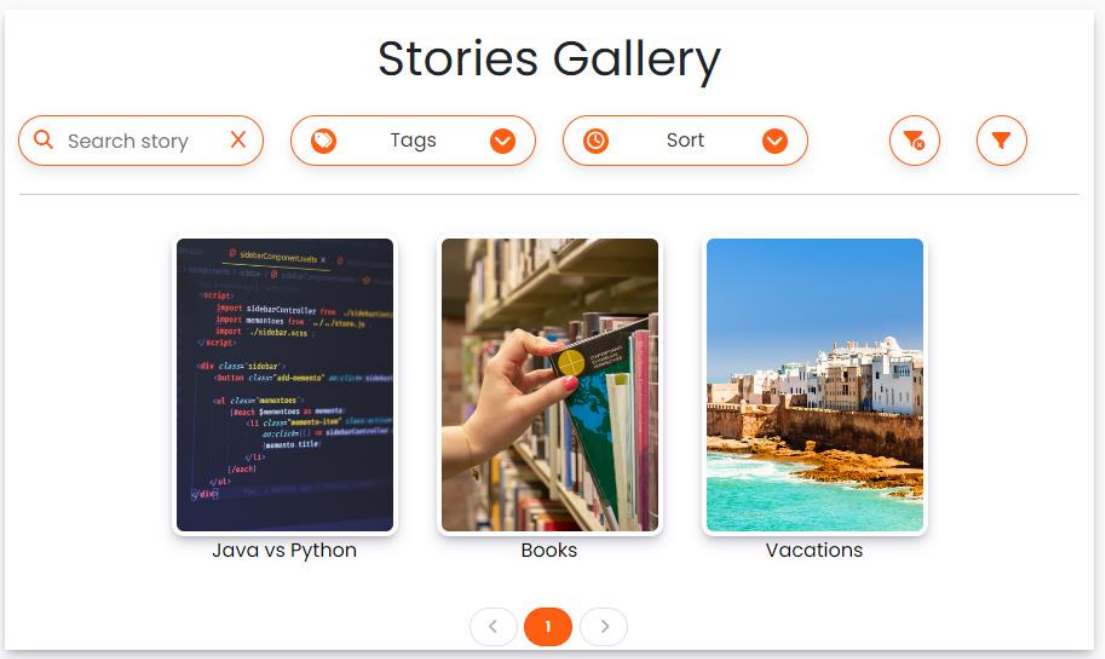

Figura 4.4.1. Galería de historias.

Figura 4.4.2. Información de la historia (1). Cabe aclarar que el engranaje de configuración en la parte superior derecha es un atajo para editar la historia y sólo aparece para los usuarios con los permisos para edición.

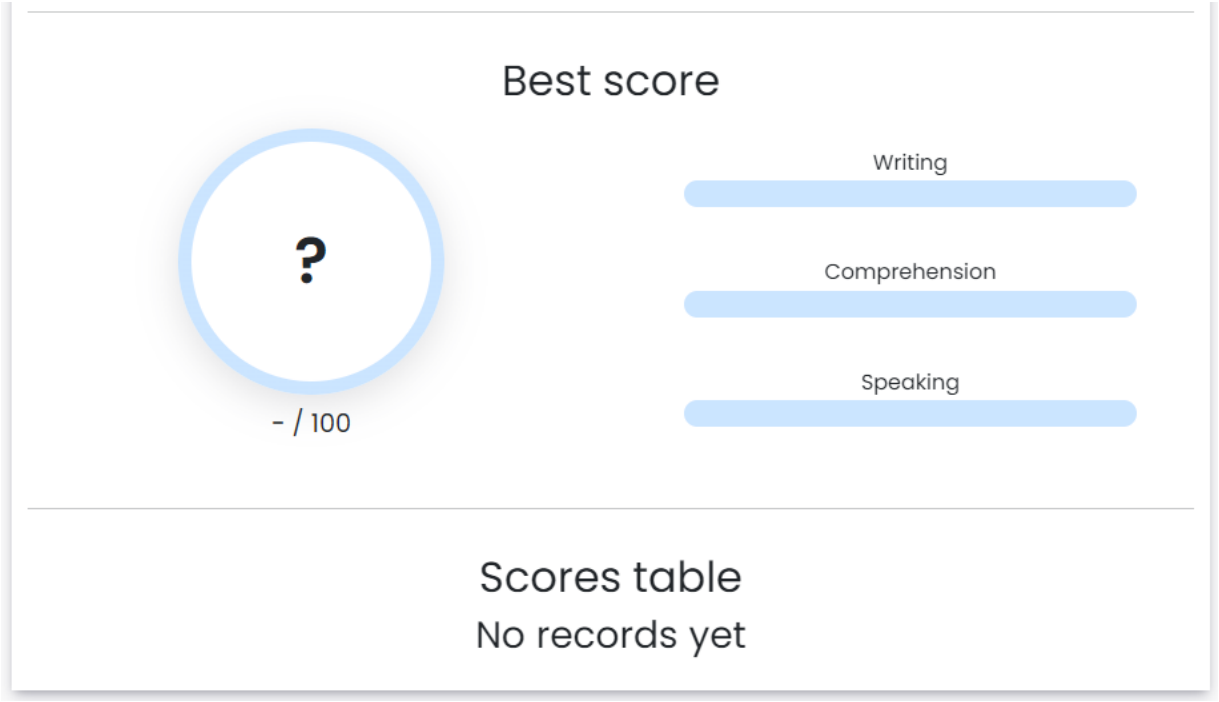

Figura 4.4.3. Información de la historia (2).

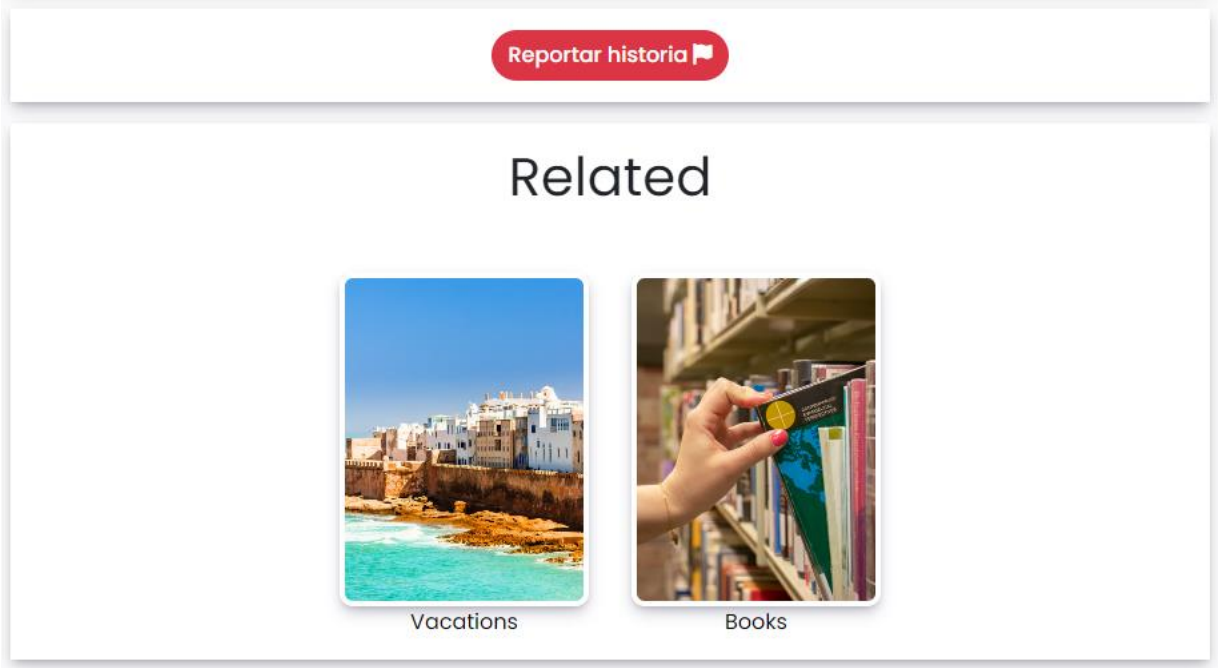

Figura 4.4.4. Información de la historia (3).

4.5 Creación de páginas de una historia

La aplicación cuenta con un sitio para la administración de páginas de una historia, en donde se debe seleccionar una plantilla para agregar una nueva página a la historia o bien, botones para eliminar o editar estas.

Figura 4.5.1. Administrador de páginas.

Al momento de elegir una plantilla, se muestra el contenido que es posible agregar conforme se ha especificado en las tareas, por ejemplo, en las figuras 4.5.2, 4.5.3 y 4.5.4 podemos observar las plantillas 1, 2 y 3 respectivamente, las cuales tienen una estructura diferente y por lo tanto permiten agregar distintos tipos de contenido o ejercicios.

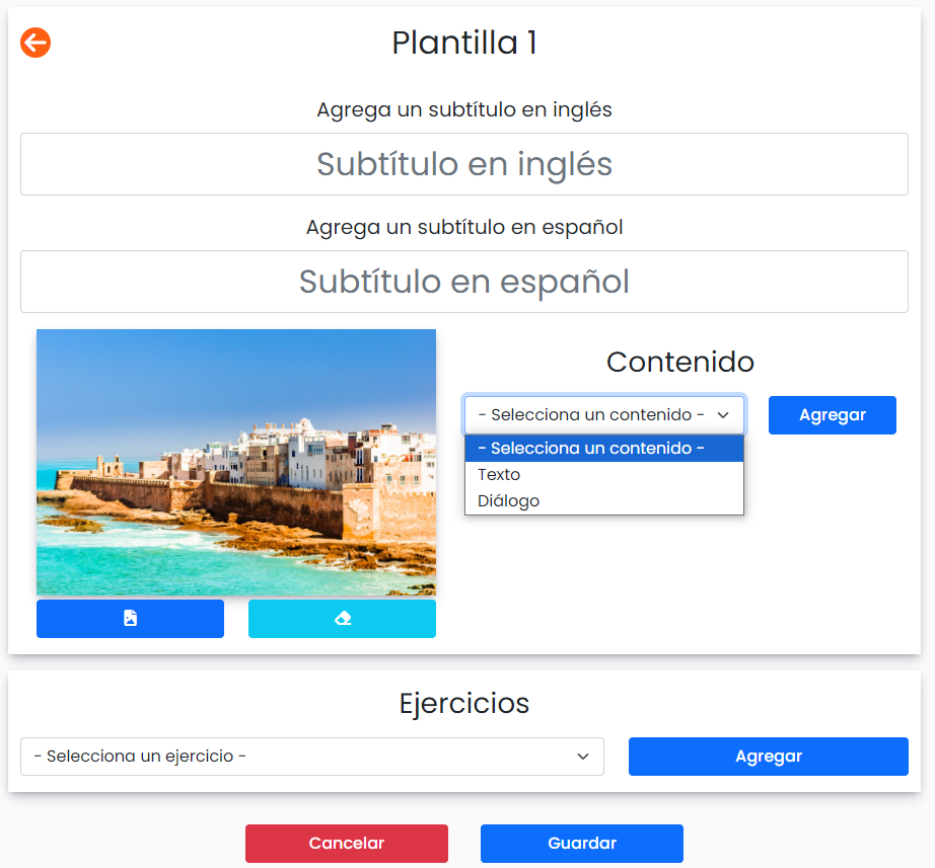

Figura 4.5.2. Plantilla de página tipo 1.

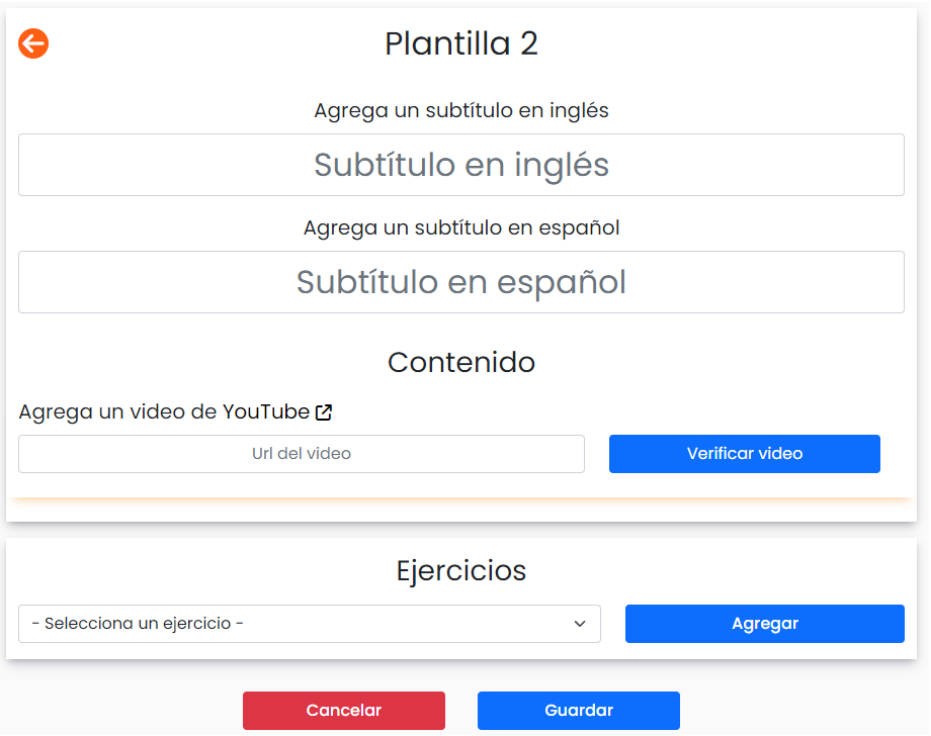

Figura 4.5.3. Plantilla de página tipo 2.

Figura 4.5.4. Plantilla de página tipo 3.

4.6 Contenido y ejercicios dentro de una página

En este documento, se define como “contenido” a las entidades que aporten información audiovisual tales como audios, diálogos, imágenes, textos o videos. A su vez, se denomina “ejercicio” a las entidades que solicitan la intervención del usuario y pueden ser contestados, un ejercicio siempre debe devolver retroalimentación cuando es contestado y al evaluar una historia se utiliza para sumar puntos de experiencia al usuario y sumar porcentaje en una de tres categorías: escritura, comprensión y pronunciación. Adicionalmente, se utiliza la palabra “elemento” para referirse tanto a contenido como a ejercicio.

La aplicación se desarrolló de manera modular para que fuera fácil agregar y eliminar elementos, cada elemento cuenta con un “número de elemento” en la base de datos, el cuál sirve para mantener una organización entre cada uno de ellos.

4.6.1 Modelos

Para la creación de modelos se sigue la siguiente lógica.

Un elemento tiene una relación muchos a uno con una página y una página también tiene esta relación con una historia, es decir, un elemento pertenece a una página y una página puede tener muchos elementos, a su vez que una página pertenece a una historia y una historia puede tener muchas páginas.

Esto se logra agregando una columna de tipo “models.ForeignKey” que toma como atributo la clase del modelo al que pertenece. El siguiente fragmento de código muestra la creación de los modelos de página y elementos definidos en las tareas.

```
# Page
class Page(models.Model):
    class Meta:
        db_table = prefix + 'page'
```

```
    # keys
    id = models.AutoField(primary_key=True)
    story = models.ForeignKey(Story, on_delete=models.CASCADE,
related_name='pages')
    # fields
    subtitle1 = models.CharField(max_length=100, default="")
    subtitle2 = models.CharField(max_length=100, default="")
    page_type = models.IntegerField(default=0)
    date_created = models.DateTimeField(auto_now_add=True, null=True)
    time_created = models.TimeField(auto_now_add=True, null=True)

# Audio
class Audio(models.Model):
    class Meta:
        db_table = prefix + 'audio'

    page = models.ForeignKey(Page, on_delete=models.CASCADE,
related_name='audios')
    # fields
    element_number = models.IntegerField()
    audio_file = models.FileField(upload_to=settings.AUDIOS_PATH, null=True,
blank=True)
    label_name = models.CharField(max_length=255, blank=True)
    show_description = models.BooleanField(default=False)
    description = models.CharField(max_length=600, blank=True)
    t_description = models.CharField(max_length=600, blank=True)


# Video
class Video(models.Model):
    class Meta:
        db_table = prefix + 'video'
    # keys
    id = models.AutoField(primary_key=True)
    page = models.ForeignKey(Page, on_delete=models.CASCADE,
related_name='videos')
    # fields
    element_number = models.IntegerField()
    url = models.CharField(max_length=2048)


# Image
class Image(models.Model):
    class Meta:
        db_table = prefix + 'image'
```

```
    # keys
    id = models.AutoField(primary_key=True)
    page = models.ForeignKey(Page, on_delete=models.CASCADE,
related_name='images')
    # fields
    element_number = models.IntegerField()
    image = models.ImageField(upload_to=settings.IMAGES_PATH, null=True,
blank=True)

# Text
class Text(models.Model):
    class Meta:
        db_table = prefix + 'text'
    # keys
    id = models.AutoField(primary_key=True)
    page = models.ForeignKey(Page, on_delete=models.CASCADE,
related_name='texts')

    language1 = models.TextField(null=False, blank=False)
    language2 = models.TextField(null=False, blank=False)
    element_number = models.IntegerField()


# Dialogue
class Dialogue(models.Model):
    class Meta:
        db_table = prefix + 'dialogue'
    # keys
    id = models.AutoField(primary_key=True)
    page = models.ForeignKey(Page, on_delete=models.CASCADE,
related_name='dialogues')
    # fields
    name = models.CharField(max_length=30)
    content1 = models.CharField(max_length=255)
    content2 = models.CharField(max_length=255)
    color = models.CharField(max_length=7, default='#000000')
    element_number = models.IntegerField()


# ---------- Exercises ---------- #
# Repeat phrase
class RepeatPhrase(models.Model):
    class Meta:
        db_table = prefix + 'repeat_phrase'
    # keys
```

```
    id = models.AutoField(primary_key=True)
    page = models.ForeignKey(Page, on_delete=models.CASCADE,
related_name='repeat_phrases')
    # fields
    content1 = models.CharField(max_length=255)
    content2 = models.CharField(max_length=255)
    show_text = models.BooleanField(default=True)
    element_number = models.IntegerField()


# Spellcheck
class Spellcheck(models.Model):
    class Meta:
        db_table = prefix + 'spellcheck'

    # keys
    id = models.AutoField(primary_key=True)
    page = models.ForeignKey(Page, on_delete=models.CASCADE,
related_name='spellchecks')
    # fields
    element_number = models.IntegerField()
    wrong_text = models.CharField(max_length=600)
    right_text = models.CharField(max_length=600)
    translated_right_text = models.CharField(max_length=600)


# Question
class MultipleChoiceQuestion(models.Model):
    class Meta:
        db_table = prefix + 'multiple_choice_question'
    # keys
    id = models.AutoField(primary_key=True)
    page = models.ForeignKey(Page, on_delete=models.CASCADE,
related_name='questions')
    # fields
    element_number = models.IntegerField()
    text = models.CharField(max_length=255)
    t_text = models.CharField(max_length=255)
    randomize_choices = models.BooleanField(default=True)


# Question Option
class QuestionChoice(models.Model):
    class Meta:
        db_table = prefix + 'question_choice'
```

```
    # keys
    id = models.AutoField(primary_key=True)
    question = models.ForeignKey(MultipleChoiceQuestion,
on_delete=models.CASCADE, related_name='choices')
    # fields
    choice_number = models.IntegerField(default=0)
    text = models.CharField(max_length=255)
    t_text = models.CharField(max_length=255)
    correct = models.BooleanField(default=False)
```

4.6.2 Creación de elementos

Para la creación de elementos, se creó una función en JavaScript que crea pequeñas plantillas del elemento correspondiente. Para la edición de estos, se llama a estas funciones y se van rellenando con los datos que manda la vista.

El procedimiento es el siguiente, en el documento HTML existe una etiqueta especial, cuando se desea introducir un nuevo elemento, esta etiqueta se duplica para obtener dos iguales, la primera es reemplazada con el contenido del elemento y de esta manera se logra una estructura uniforme.

El siguiente fragmento de código muestra el funcionamiento del menú para agregar elementos, disponible en la plantilla de diseño libre, podemos observar que la etiqueta especial se duplica.

```
<div class="col-12 col-md-8">
  <select class="form-select" id="element_selected">
    <option selected>- Selecciona un elemento -</option>
    <option value="1">Audio</option>
    <option value="2">Diálogo</option>
    <option value="3">Imagen</option>
    <option value="4">Texto</option>
    <option value="5">Video</option>
    <option>-------------------------</option>

    <option value="6">Pregunta de opción múltiple</option>
    <option value="7">Repetir oración</option>
    <option value="8">Revisar ortografía</option>
  </select>
</div>
```

```
<div class="col-12 col-md-4 mb-3">
  <button class="btn btn-primary shadow-none fw-bold w-100" type="button"
onclick="addElement()">Agregar</button>
</div>

<script>
  function addElement() {
    let selected = document.getElementById('element_selected').value;
    document.querySelector('.new_element').outerHTML =
      `
      <div class="new_element"></div>
      <div class="new_element"></div>
      `;

    switch (parseInt(selected)) {
      case 1:
        addAudio();
        break;
      case 2:
        addDialogue();
        break;
      case 3:
        addImage();
        break;
      case 4:
        addText();
        break;
      case 5:
        addVideo();
        break;
      case 6:
        addMultipleChoiceQuestion();
        break;
      case 7:
        repeatPhrase();
        break;
      case 8:
        addSpellcheck();
        break;
    }
  }
</script>
```

Cada función del switch reemplaza la primera etiqueta especial, haciendo posible un ciclo para agregar un elemento tras otro. En el siguiente fragmento de código se muestra un ejemplo sobre el elemento de imagen.

```
// Image
function addImage() {
  let string_id = generateStringId();
  document.querySelector(".new_element").outerHTML =
    `
    <div class="row exercise-sep" id="element_${max_elem}" name="images">
      <div class="row">
        <div class="col-12 d-flex">
          <div class="col-10 col-md-11">
            <h3 class="fs-4 text-center my-3">Imagen</h3>
          </div>
          ${html_modal_delete(max_elem, '¿Desea eliminar esta imagen?')}
        </div>
      </div>

      <div class="col-12">
        <div style="display:flex; justify-content:center; width:100%;
margin:1rem 0;">
          <img class="box-shadow d-none" alt="No se pudo cargar la imagen"
name="preview" id="preview_${string_id}"
          style="max-height:450px; max-width:100%;">
        </div>
      </div>
      <div class="col-12">
        <input type="file" id="file_${string_id}" name="image_file"
class="form-control" accept="image/*">
        <input type="text" name="default_image" readonly hidden>
      </div>
      <input value="${max_elem}" name="element_number" hidden>
      <input name="id" hidden>
    </div>
  `;

  document.getElementById('file_' + string_id).addEventListener('change', (e)
=> {
    let preview_id = e.target.id.replace('file_', 'preview_');
    let preview_image = document.getElementById(preview_id);

    if (e.target.value == "") {
```

```
      let default_image = e.target.parentElement.querySelector('[name =
default_image]').value;

      preview_image.setAttribute('src', default_image);

      if (default_image == '')
        preview_image.classList.add('d-none');
      else
        preview_image.classList.remove('d-none');
    }

    if (e.target.files && e.target.files[0]) {
      let reader = new FileReader();
      reader.onload = function (r) {
        preview_image.setAttribute('src', r.target.result);
        preview_image.classList.remove('d-none');
      };
      reader.readAsDataURL(e.target.files[0]);
    }
  });

  max_elem++;
}
```

En el siguiente código se observa la forma de recopilar la información y enviarla al servidor, los datos son recogidos buscando en la estructura de su HTML, se crea un objeto JSON y posteriormente se envía con ayuda de JQuery.

```
function savePage() {
    let formData = new FormData();
    // Contents
    // Images
    let images = document.getElementsByName("images");
    let jsonImages = [];
    let image_counter = 0;
    for (let i of images) {
        let jsonI = {
            "id": i.querySelector('[name = id]').value,
            "element_number": i.querySelector('[name =
element_number]').value,
            "file_key": ''
        }
```

```
        let image_file = i.querySelector('[name = image_file]').files[0];
        if (image_file != undefined) {
            let file_key = 'image_file_' + image_counter.toString();

            jsonI.file_key = file_key;
            formData.append(file_key, image_file);
            image_counter++;
        }

        jsonImages.push(jsonI);
    }

    // Videos
    let videos = document.getElementsByName('videos');
    let jsonVideos = [];
    for (let video of videos) {
        let jsonVideo = {
            "id": video.querySelector('[name = id]').value,
            "element_number": video.querySelector('[name =
element_number]').value,
            "url": video.querySelector('[name = url]').value
        }
        jsonVideos.push(jsonVideo);
    }

    // Audios
    let audios = document.getElementsByName("audios");
    let jsonAudios = [];
    let audios_counter = 0;
    for (let a of audios) {
        let jsonA = {
            "id": a.querySelector('[name = id]').value,
            "element_number": a.querySelector('[name =
element_number]').value,

            "label_name": a.querySelector('[name = label_name]').value,
            "show_description": a.querySelector('[name =
show_description]').checked,
            "description": a.querySelector('[name = description]').value,
            "t_description": a.querySelector('[name = t_description]').value,
            "file_key": ''
        }

        let audio_file = a.querySelector('[name = audio_file]').files[0];
```

```
        if (audio_file != undefined) {
            let file_key = 'audio_file_' + image_counter.toString();

            jsonA.file_key = file_key;
            formData.append(file_key, audio_file);
            audios_counter++;
        }
        jsonAudios.push(jsonA);
    }

    // Texts
    let texts = document.getElementsByName('texts');
    let jsonTexts = [];
    for (let text of texts) {
        let jsonText = {
            "id": text.querySelector('[name = id]').value,
            "element_number": text.querySelector('[name =
element_number]').value,
            "language1": text.querySelector('[name = language1]').value,
            "language2": text.querySelector('[name = language2]').value
        }
        jsonTexts.push(jsonText);
    }

    // Dialogues
    let dialogues = document.getElementsByName('dialogues');
    let jsonDialogues = [];
    for (let d of dialogues) {
        let jsonD = {
            "id": d.querySelector('[name = id]').value,
            "element_number": d.querySelector('[name =
element_number]').value,
            "name": d.querySelector('[name = name]').value,
            "color": d.querySelector('[name = color]').value,
            "language1": d.querySelector('[name = language1]').value,
            "language2": d.querySelector('[name = language2]').value
        }
        jsonDialogues.push(jsonD);
    }


    // Exercises
    // Repeat phrases
    let repeatPhrases = document.getElementsByName('repeatPhrases');
    let jsonRepeatPhrases = [];
```

```
    for (let r of repeatPhrases) {
        let jsonR = {
            "id": r.querySelector('[name = id]').value,
            "element_number": r.querySelector('[name =
element_number]').value,
            "language1": r.querySelector('[name = language1]').value,
            "language2": r.querySelector('[name = language2]').value,
            "show_text": r.querySelector('[name = show_text]').checked
        }
        jsonRepeatPhrases.push(jsonR);
    }

    // Spellchecks
    let spellchecks = document.getElementsByName('spellchecks');
    let jsonSpellchecks = [];
    for (let s of spellchecks) {
        let jsonS = {
            "id": s.querySelector('[name = id]').value,
            "element_number": s.querySelector('[name =
element_number]').value,
            "wrong_text": s.querySelector('[name = wrong_text]').value,
            "right_text": s.querySelector('[name = right_text]').value,
            "translated_right_text": s.querySelector('[name =
translated_right_text]').value
        }
        jsonSpellchecks.push(jsonS);
    }

    // Multiple choice questions
    let mc_questions = document.getElementsByName('mc_questions');
    let jsonMc_questions = [];
    for (let q of mc_questions) {
        // Object for questions
        let qObj = {
            "id": q.querySelector('[name = id]').value,
            "element_number": q.querySelector('[name =
element_number]').value,
            "text": q.querySelector('[name = text]').value,
            "t_text": q.querySelector('[name = t_text]').value,
            "randomize_choices": q.querySelector('[name =
randomize_choices]').checked,
            "choices": []
        }
        // Object for choices
        let choices = q.querySelectorAll('.choice');
```

```
        for (let c of choices) {
            let cObj = {
                "id": c.querySelector('[name = id]').value,
                "choice_number": c.querySelector('[name =
choice_number]').value,
                "text": c.querySelector('[name = text]').value,
                "t_text": c.querySelector('[name = t_text]').value,
                "correct": c.querySelector('[name =
correct]').classList.contains('checked')
            }
            // Add choices to the question
            qObj.choices.push(cObj);
        }
        jsonMc_questions.push(qObj);
    }

    // Build Json to send
    let jsonData = {
        "page_id": page_id,
        "sub1": sub1 = document.getElementById('sub1').value,
        "sub2": sub2 = document.getElementById('sub2').value,
        "images": jsonImages,
        "videos": jsonVideos,
        "audios": jsonAudios,
        "texts": jsonTexts,
        "dialogues": jsonDialogues,
        "repeatPhrases": jsonRepeatPhrases,
        "spellchecks": jsonSpellchecks,
        "mc_questions": jsonMc_questions,
        "deleted": objects_deleted
    }

    formData.append("data", JSON.stringify(jsonData));

    const csrftoken = getCookie('csrftoken');
    $.ajax({
        type: "POST",
        url: URL_SAVE_PAGE,
        data: formData,
        processData: false,  // Tell jQuery not to process the data
        contentType: false,  // Tell jQuery not to set the content type
        headers: { "X-CSRFToken": csrftoken },
        success: function (response) {
            switch (response.message) {
                case "success":
```

```
                    window.location.href = URL_VIEW_PAGES;
                    break;
            }
        },
        error: function (response) {
            //console.error(response);
        }
    });
}
```

El último paso es que el servidor reciba estos datos y los guarde o los rechace si encontró algo mal, en el siguiente fragmento de código sólo se muestra la forma con la que el objeto de imagen es guardado, pues de lo contrario podría llegar a ser muy extenso.

```
# collecting data
        data = request.POST["data"]
        data = json.loads(data)

        # creating page
        pgObj = pgForm.save(commit=False)
        if data["page_id"] != 0:
            pgObj = Page.objects.get(id=int(data["page_id"]))

        pgObj.story = story
        pgObj.subtitle1 = data["sub1"]
        pgObj.subtitle2 = data["sub2"]
        pgObj.page_type = page_type


        # Images
        images_to_submit = []
        images = data["images"]

        for img in images:

            imgObj = Image()
            imgObj.page = pgObj

            if img["id"] != "":
                imgObj = Image.objects.get(id=int(img["id"]))

            imgObj.element_number = img["element_number"]
```

```
            file_key = img['file_key']
            if file_key == '': # if user didn't send an image file
                if not imgObj.image: # previous imgObj don't have an image
                    if pgObj.page_type == 1: # doesn't matter if imgObj don't
have image because it can
                        continue
                    else: # Any other template with image input empty will be
returned as an error
                        return JsonResponse({'message': 'Must select an image
file'})

            else: # user has sent an image file
                img_file = request.FILES.get(file_key)

                extension = img_file.name.split('.')[-1]
                random_name = f'{uuid.uuid4()}.{extension}'

                imgObj.image.save(random_name, File(img_file), save=False)
            images_to_submit.append(imgObj)


# Images
        for image in images_to_submit:
            image.save()
```

4.6.3 Eliminación de elementos

El proceso de eliminar elementos es sencillo, cuando el editor decide eliminar un elemento, el id de este se almacena en un objeto JSON y al momento de guardar la página, el servidor lee los id almacenados y los elimina.

```
let objects_deleted = {
    "images": [],
    "videos": [],
    "audios": [],
    "texts": [],
    "dialogues": [],
    "repeatPhrases": [],
    "spellchecks": [],
    "mc_questions": [],
    "question_choices": []
```

```
};

function deleteElement(id) {
    let element_to_delete = document.getElementById("element_" + id);
    let id_database = element_to_delete.querySelector('[name = id]').value;
    if (id_database) {
        let name = element_to_delete.getAttribute("name"); // IMPORTANT: NAME
MUST BE EQUAL TO INSIDE OF objects_deleted
        objects_deleted[name].push(id_database);
    }
    element_to_delete.remove();

    let html = `
    <div class="alert alert-success alert-dismissible fade show mb-2 mt-2"
role="alert">
        <div style="text-align: center;">
            <strong>Elemento eliminado correctamente</strong>
        </div>
        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>
    </div>`;
    document.getElementById("alert_removed_success").innerHTML = html;
}
```

```
# Delete objects that were deleted
        deleted = data["deleted"]
        for d in deleted['images']:
            safe_delete(Image, d)
```

```
def safe_delete(model, id):
    try:
        model.objects.get(id=int(id)).delete()
    except:
        pass
```

4.6.4 Resultados

Los resultados son muy satisfactorios pues fueron el resultado de implementar las tareas de manera clara y precisa.

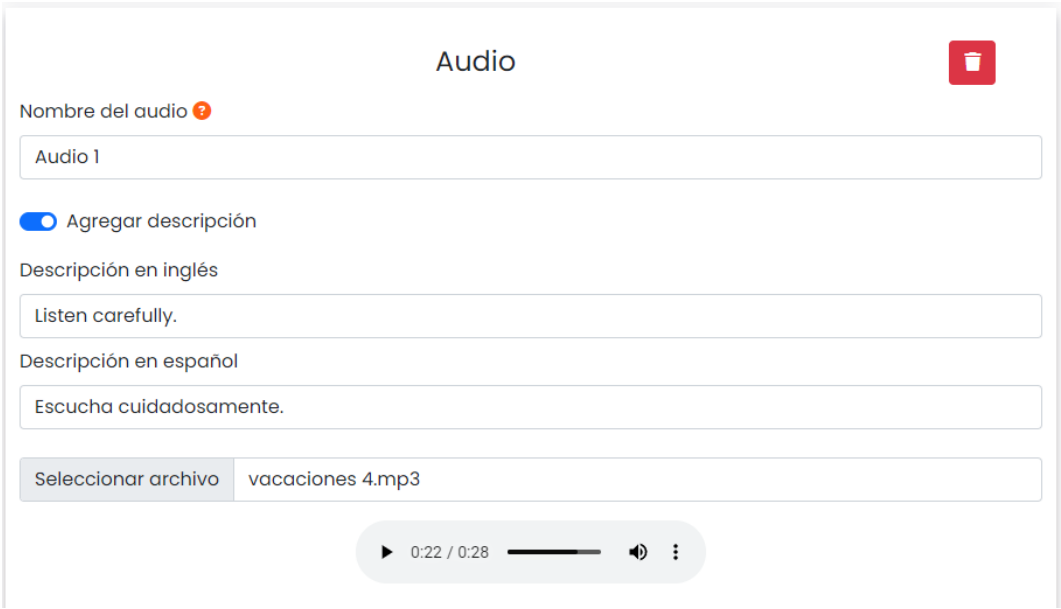

Figura 4.6.4.1. Elemento de audio.

Figura 4.6.4.2. Elemento de diálogo.

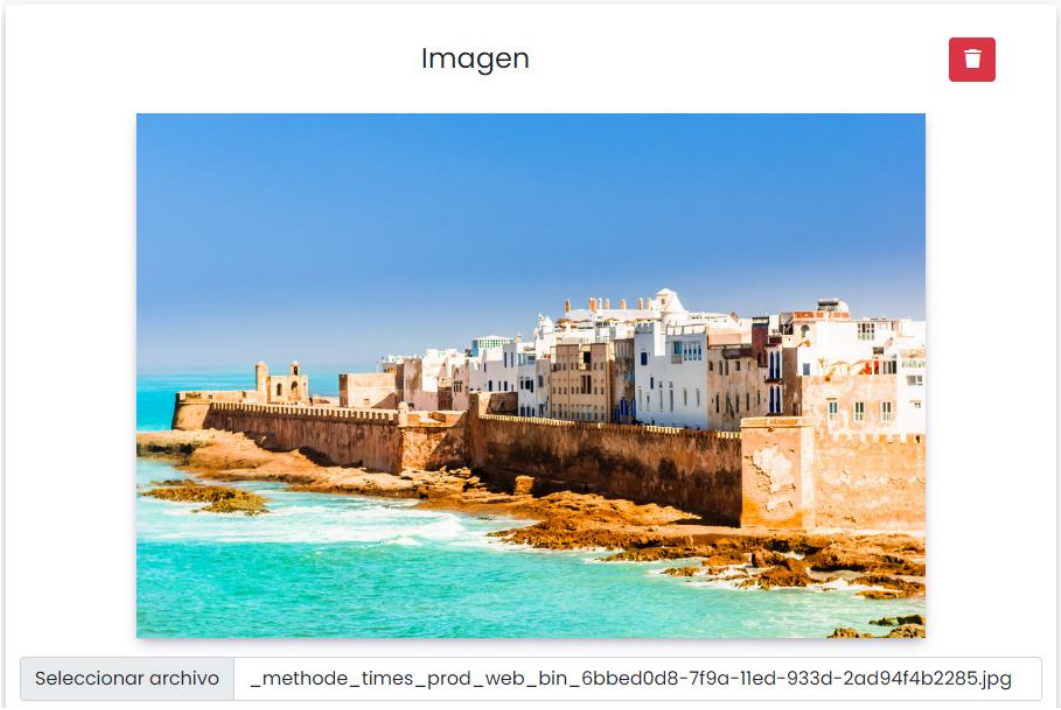

Figura 4.6.4.3. Elemento de imagen.

Figura 4.6.4.4. Elemento de texto.

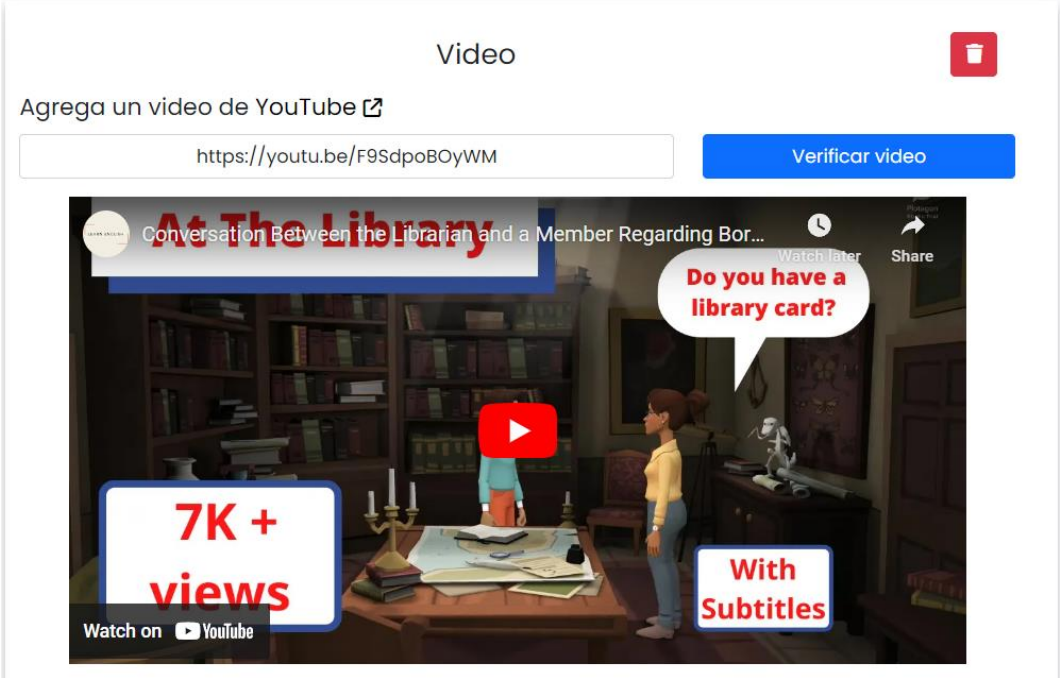

Figura 4.6.4.5. Elemento de video.

Figura 4.6.4.6. Elemento de pregunta de opción múltiple.

Figura 4.6.4.7. Elemento de repetir una oración.

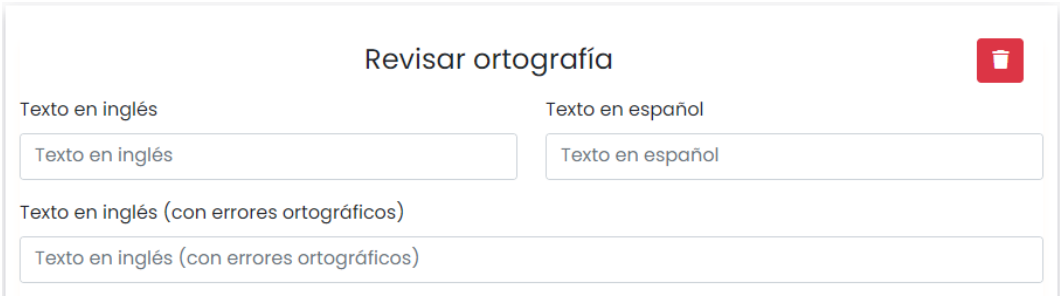

Figura 4.6.4.8. Elemento de revisión de ortografía.

4.7 Mostrar las páginas de una historia al usuario

Todo inicia con la vista para ver la información de una historia, aquí dentro, el usuario puede pulsar el botón “Start” para empezar a contestar la historia.

A continuación, se muestran fragmentos de código mostrando el procedimiento del lado del servidor, los datos de la página se serializan con ayuda de un serializador, posteriormente se elimina contenido no deseado como las respuestas de los ejercicios, de esta forma nos aseguramos de que el usuario no pueda saber la respuesta correcta, porque la respuesta en ningún momento es enviada al cliente, acto seguido se transforma en un objeto de tipo JSON listo para ser enviado al cliente.

```
@login_required(login_url='/login/')
def pageDisplayer(request, route, page_number):
```

```
    story = Story.objects.get(route=route)
    user_profile = request.user.profile

    page_index = page_number - 1
    pages = story.pages.all().order_by('date_created', 'time_created')
    total_pages = len(pages)

    prev_page = None
    next_page = None
    current_page = pages[page_index]

    if page_index >= 1:
        prev_page = page_number - 1
    if page_index < len(pages) - 1:
        next_page = page_number + 1

    s_page = PageSerializer(current_page)
    # Quit answers
    # Questions
    for q in s_page.data['questions']:
        for c in q['choices']:
            c['correct'] = True
    # Spellchecks
    for s in s_page.data['spellchecks']:
        s['right_text'] = 'Hola, señor tramposo'
    # Repeat phrases
    for rp in s_page.data['repeat_phrases']:
        if not rp['show_text']:
            rp['content1'] = 'Hola, señor tramposo'
            rp['content2'] = 'Hola, señor tramposo'

    json_page = JSONRenderer().render(s_page.data).decode('utf-8')

    …
    context = {

            …

            'json_page': json_page,
            …
            }
        return render(request, 'page_display/page_display.html', context)
```

La lógica para mostrar el contenido del lado del cliente se basa en un diseño modular, primero se revisa el tipo de página que se solicita mostrar, a continuación, los elementos se ordenan y son colocados en pantalla.

El siguiente es un fragmento de HTML donde se usan las “template tags” de Django para decidir la estructura que va a tener la página.

```
{% if current_page.page_type == 1 %}
{% include 'page_display/page_display_template_1.html' %}
{% elif current_page.page_type == 2 %}
{% include 'page_display/page_display_template_2.html' %}
{% elif current_page.page_type == 3 %}
{% include 'page_display/page_display_template_3.html' %}
{% endif %}
```

Adicionalmente, el contenido de la página se carga mediante funciones de JavaScript.

```
/* Load page */

// Set subtitle
let page_subtitle = document.getElementById("subtitle");
if (page_subtitle != null)
  page_subtitle.flipText();

switch (parseInt(page_type)) {
  case 1:
    loadTemplate1(total_translations);
    break;
  case 2:
    loadTemplate2(total_translations);
    break;
  case 3:
    loadTemplate3(total_translations);
    break;
}

…

function loadTemplate3(total_translations) {
  sortCollections();
```

```
  for (let e of collections_sorted) {
    document.querySelector('.new_element').outerHTML = `
    <div class="new_element"></div>
    <div class="new_element"></div>`;
    loadElement(e, total_translations);
  }
  setFunctionality(total_translations);
}

…
// Switch for load element
function loadElement(element, total_translations) {
  switch (element.type) {
    case "image":
      renderImage(element);
      break;
    case "video":
      renderVideo(element);
      break;
    case "audio":
      renderAudio(element);
      break;
    case "text":
      renderText(element);
      break;
    case "dialogue":
      renderDialogue(element, total_translations);
      break;
    case "repeat_phrase":
      renderRepeatPhrase(element, total_translations);
      break;
    case "spellcheck":
      renderSpellcheck(element, total_translations);
      break;
    case "mc_question":
      renderMCQuestion(element);
      break;
  }
}
// End of switch for load element
```

Puesto que son bastantes elementos, se procede a mostrar únicamente el código para renderizar audio.

```
function renderAudio(audio) {
  let desc = '';
  if (audio.show_description) {
    desc = `
    <div class="text-center" style="margin-top:1rem;"
id="audio_flip_description_${audio.id}">
      <div class="front px-1 styles-front">${audio.description}</div>
      <div class="back px-1 styles-back">${audio.t_description}</div>
    </div>
    `;
  }

  document.querySelector(".new_element").outerHTML =
    `
    <div class="row exercise-sep">
      <div class="col-12">
        ${desc}
        <div style="display:flex; justify-content:center; width:100%;
margin:1rem 0;">
          <audio controls id="audio_player_${audio.id}"
controlsList="nodownload">
            <source src="" type="audio/mpeg">
          </audio>
        </div>
      </div>
    </div>
  `;
  let audio_player = document.getElementById('audio_player_' + audio.id);

  if (desc != '')
    document.getElementById('audio_flip_description_' + audio.id).flipText();

  // get the audio as blob
  fetch(audio.audio_file)
    .then(response => response.blob())
    .then(blob => {
      audio_player.volume = 0.5;
      audio_player.src = URL.createObjectURL(blob);
    })
    .catch(error => {
      console.error('Error fetching the audio file:', error);
    });
```

```
}
```

4.7.1 Función para traducir texto

Un aspecto fundamental de la aplicación son las traducciones, se encontraron bastantes formas de traducción en las investigaciones realizadas, sin embargo, la mayoría eran poco atractivas en el contexto.

Hay dos métodos de traducción pero que hacen exactamente lo mismo tanto en diseño como en lógica, hacer esta funcionalidad fue un gran desafío, pero sin duda vale la pena cada minuto dedicado debido a que es una función que está presente en toda la aplicación.

La función descrita aquí recibe el nombre de flipText y su uso es sencillo.

```
/*
  Flip text: Example of use
  <div id="flip_text">
    <p class="front">This text goes to front</p>
    <p class="back">This text goes to back</p>
  </div>
  document.getElementById('flip_text').flipText();
*/
```

Se trata de una función prototipo que puede ser llamada mediante “HTML.flipText()”, el elemento al que se le aplique esta función debe tener una etiqueta con la clase “front” y otra etiqueta con la clase “back”, la función tomará el texto contenido dentro de cada etiqueta y meterá cada palabra dentro de un elemento div, a través de diversos efectos con CSS y JavaScript generará el efecto de un texto resaltado cuando el mouse se encuentre por encima del párrafo y un efecto de media vuelta por cada palabra cuando se haga un click sobre este.

Código en JavaScript para la función flipText().

```
HTMLElement.prototype.flipText = function (render) {
  if (this.getAttribute('fliptext') != null)
    return;

  front = this.querySelector('.front');
```

```
  back = this.querySelector('.back');

  if (front === null || back === null)
    return;

  // Classes
  front_classes = front.classList.value + "";
  back_classes = back.classList.value + "";

  front_classes = front_classes.replace('front ', '');
  back_classes = back_classes.replace('back ', '');

  // Text
  text1 = front.innerText;
  text2 = back.innerText;

  let arr1 = text1.split("|");
  let arr2 = text2.split("|");

  // Html
  let html = ``;
  let flip_ids = [];
  for (let i = 0; i < arr1.length; i++) {
    // Id
    let paper_id = generateStringId();

    let words1 = arr1[i].replace(/ {2,}/g, ' ').trim(); // removes extra
spaces
    let words2 = arr2[i].replace(/ {2,}/g, ' ').trim(); // removes extra
spaces

    words1 = words1.split(" ");
    words2 = words2.split(" ");

    let max = words1.length > words2.length ? words1.length : words2.length;
    for (let j = 0; j < max; j++) {
      let has_content = true;
      try {
        if (words1[j] == '' || words1[j] == '')
          has_content = false;
      } catch (e) { }

      if (!has_content)
        continue;
```

```
      html += `<div class="paper paper_${paper_id}">`;

      if ((j < words1.length) && (j < words2.length)) {
        if (j < words1.length)
          html += `<div class="front ${front_classes}"> ${words1[j]} </div>`;

        if (j < words2.length)
          html += `<div class="back p-abs ${back_classes}"> ${words2[j]}
</div>`;
      }
      else {
        if (j < words1.length)
          html += `<div class="front extra ${front_classes}"> ${words1[j]}
</div>`;
        else if (j >= words1.length)
          html += `<div class="front extra"
style="border:none;color:rgba(0,0,0,0);background:rgba(0,0,0,0);width=0;"></di
v>`;

        if (j < words2.length)
          html += `<div class="back extra hide-extras p-abs ${back_classes}">
${words2[j]} </div>`;
        else if (j >= words2.length)
          html += `<div class="back extra hide-extras p-abs px-1"
style="border:none;color:rgba(0,0,0,0);background:rgba(0,0,0,0);"></div>`;
      }
      html += `</div>`;
      let flag = false;
      try {
        if (words1[j].includes("<span></span>"))
          flag = true;
      } catch (e) { }

      try {
        if (words2[j].includes("<span></span>"))
          flag = true;
      } catch (e) { }

      if (flag)
        html += `<br>`;
    }
    flip_ids.push(paper_id);
  }

  if (render == 'outter')
```

```
    this.outerHTML = html;
  else
    this.innerHTML = html;

  this.setAttribute('fliptext', true);
  for (let id of flip_ids)
    flipPapers(id);
}

function flipPapers(id) {
  let papers = document.getElementsByClassName("paper_" + id);

  for (let paper of papers) {
    paper.addEventListener("click", () => {
      let fronts = document.querySelectorAll(".paper_" + id + " .front");
      let backs = document.querySelectorAll(".paper_" + id + " .back");
      let ex = document.querySelectorAll(".paper_" + id + " .extra");

      for (let e of ex)
        e.classList.toggle("hide-extras");

      for (let f of fronts)
        f.classList.toggle("p-abs");

      for (let b of backs)
        b.classList.toggle("p-abs");

      for (let paper2 of papers)
        paper2.classList.toggle("flip");
    });

    paper.addEventListener("mouseenter", () => {
      let fronts = document.querySelectorAll(".paper_" + id + " .front");
      let backs = document.querySelectorAll(".paper_" + id + " .back");
      for (let f of fronts)
        f.classList.add("color-hover");
      for (let b of backs)
        b.classList.add("color-hover");
    });

    paper.addEventListener("mouseleave", () => {
      let fronts = document.querySelectorAll(".paper_" + id + " .front");
      let backs = document.querySelectorAll(".paper_" + id + " .back");
      for (let f of fronts)
        f.classList.remove("color-hover");
```

```
      for (let b of backs)
        b.classList.remove("color-hover");
    });
  }
}
```

Clases de CSS para la funcionalidad de flipText().

```
.paper {
  display: inline-block;
  position: inherit;
  perspective: 1000px;
  transform-origin: center;
  width: auto;
  background-color: rgba(0, 0, 0, 0);
}

.paper:hover {
  cursor: pointer;
}

.color-hover {
  background-color: rgb(253, 253, 164);
}

.front,
.back {
  width: auto;
  backface-visibility: hidden;
  overflow: hidden;
  white-space: nowrap;
  transition: transform 0.5s ease-in-out;
}

.front {
  transform: rotateY(0deg);
}

.back {
  top: 0;
  transform: rotateY(180deg);
}

.flip .front {
```

```
  transform: rotateY(180deg);
}

.flip .back {
  transform: rotateY(360deg);
}

.p-abs {
  position: absolute;
}

.hide-extras {
  background: rgba(0, 0, 0, 0) !important;
  border: none !important;
  color: rgba(0, 0, 0, 0) !important;
}
```

4.7.2 Resultados

Los resultados fueron increíblemente bien, en especial la función de traducción la cual se emplea a través de toda la aplicación.

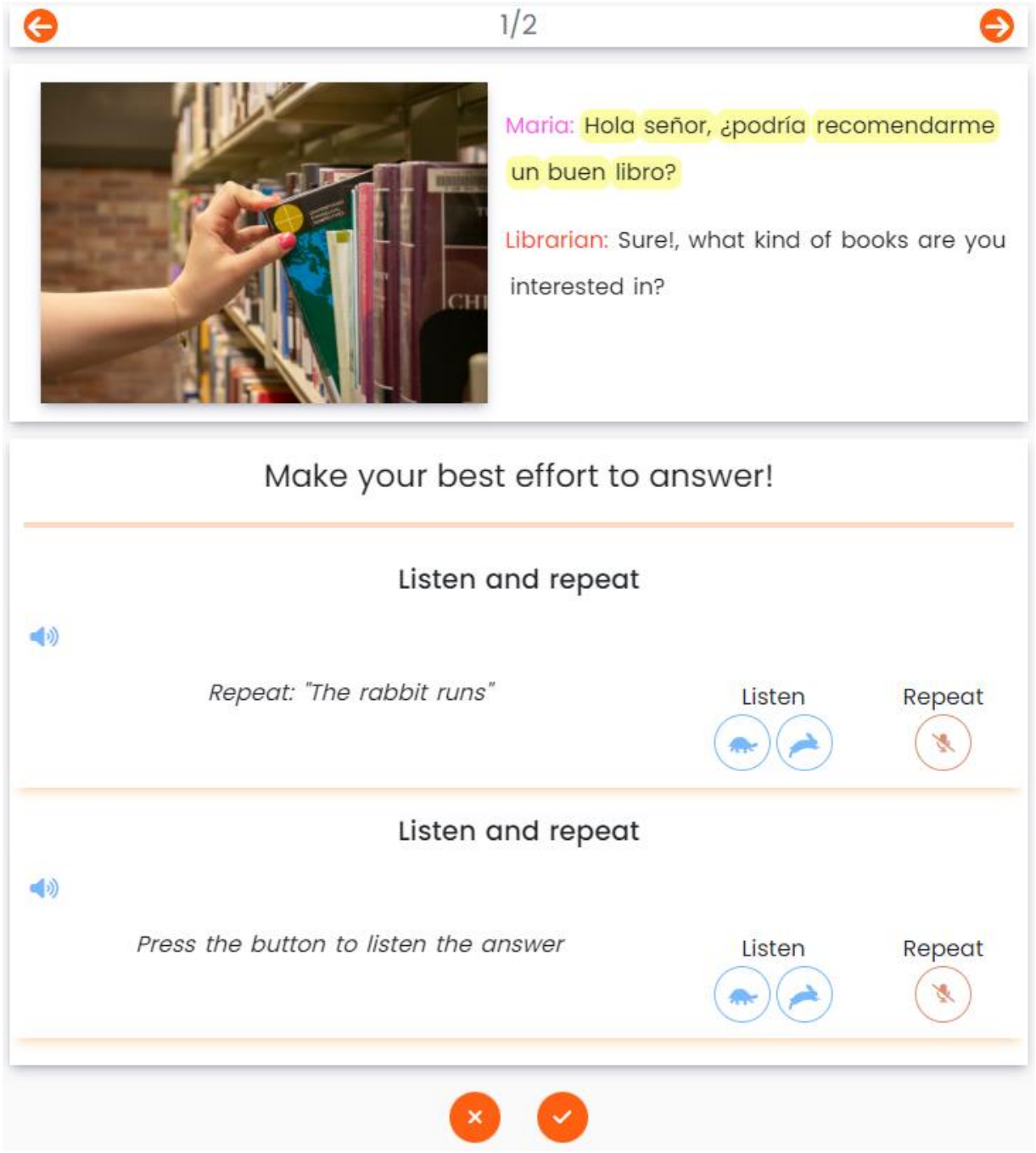

Figura 4.7.2.1. Plantilla de tipo 1 renderizada. En la figura se muestran dos ejercicios renderizados con la plantilla de tipo 1, además de mostrar dos diálogos. El primer diálogo ha sido traducido desde el inglés y se encuentra resaltado en color amarillo debido a la función flipText().

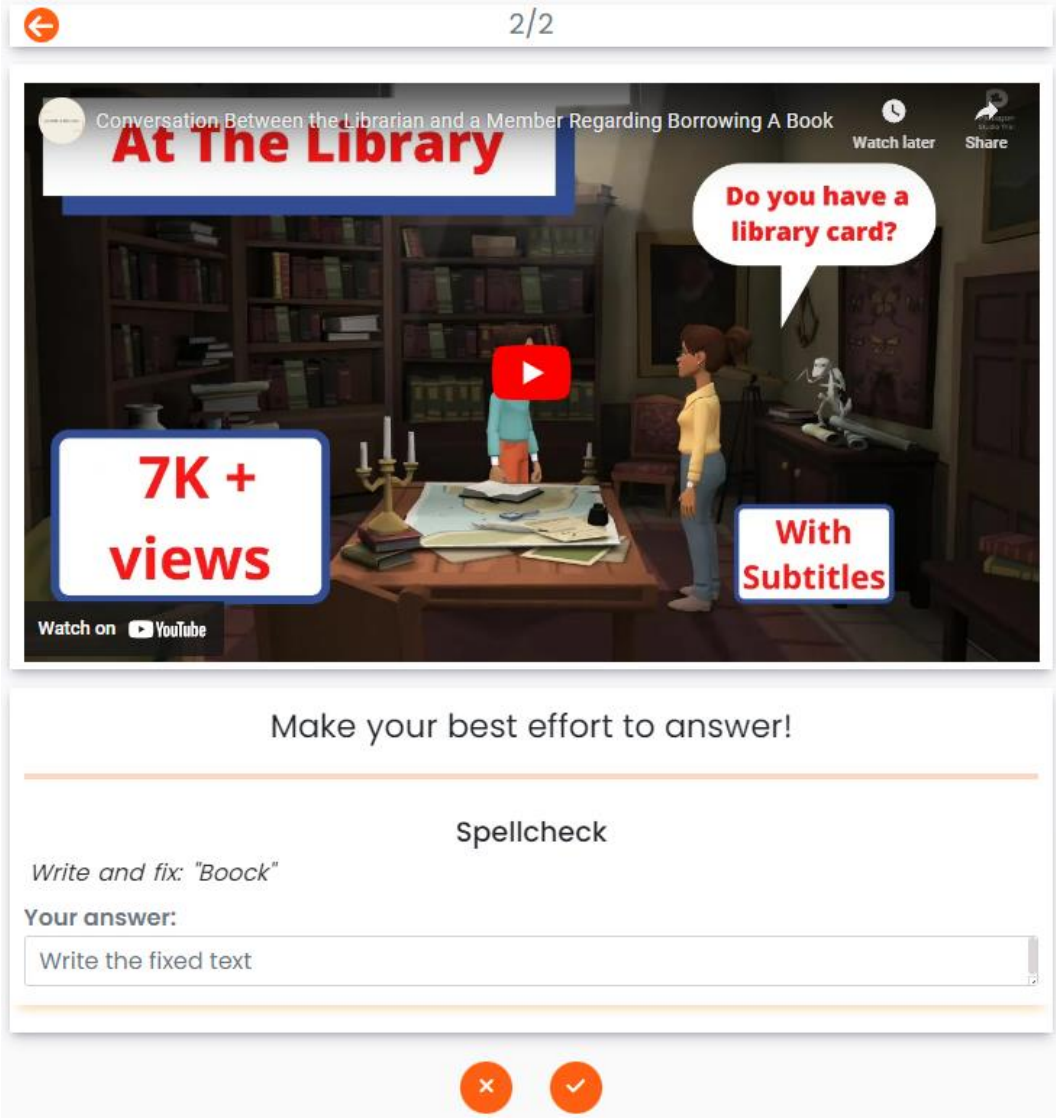

Figura 4.7.2.2. Plantilla de tipo 2 renderizada. En la figura se muestra el ejercicio para revisar ortografía, junto con el contenido de video propio de este tipo de plantilla.

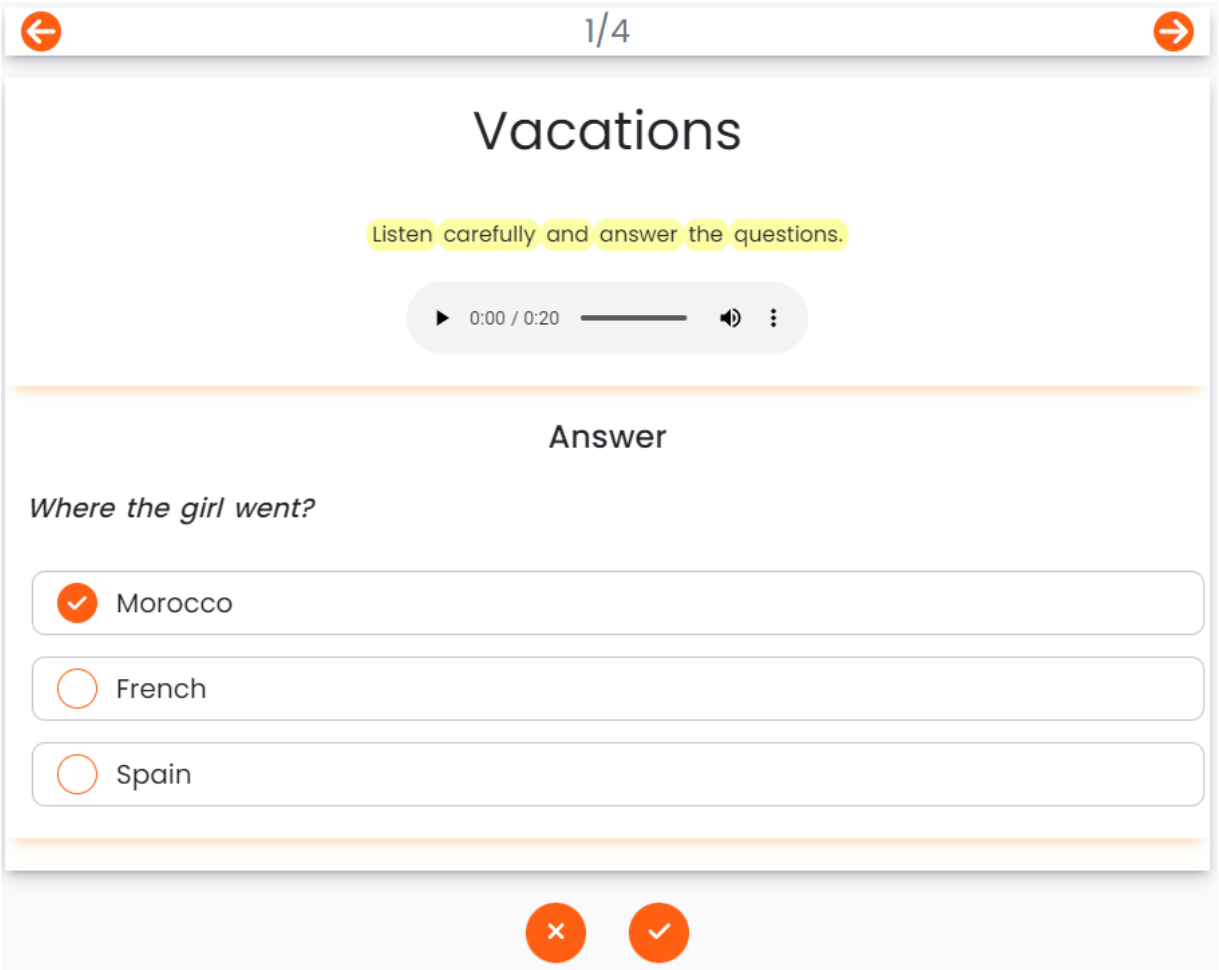

Figura 4.7.2.3. Plantilla de tipo 3 renderizada. En la figura se aprecia el contenido de audio junto con un ejercicio de pregunta de opción múltiple. El texto descriptivo del audio se encuentra resaltado en color amarillo debido a la función flipText().

4.8 Proceso para calificar historias

Las historias son calificadas desde la vista que manda los datos para renderizar las páginas de una historia. Django permite la ejecución de diferente código dependiendo de la solicitud, la solicitud enviada para ver el contenido de una página es GET, mientras que se utiliza POST para la lógica de calificar las páginas y guardar las respuestas.

La vista para renderizar páginas de una historia sigue el siguiente flujo, verifica si la llamada es GET o POST, si es GET manda la información necesaria

para renderizar la página, si es POST verifica que la página no haya sido calificada previamente, luego revisa si se desea una evaluación, si es el caso revisa todas las respuestas guardadas, las evalúa, crea un registro en la tabla de puntuaciones y adjunta esos datos en la respuesta a la solicitud, para finalizar, devuelve la retroalimentación apropiada.

```
if request.method == 'POST':
        update = False
        if db_answers:
            if db_answers[0].submited:
                return JsonResponse({'message': 'This page is already
submited'})
            else: # update answers
                update = True

        # Get data from client
        evaluate = request.POST['evaluate']
        submit = request.POST['submit']
        answers = request.POST['answers']

        evaluate = json.loads(evaluate)
        answers = json.loads(answers)
        submit = json.loads(submit)

        # Filter exercises
        rp_answered = db_answers.filter(exercise_type=rp_content_type)
        spc_answered = db_answers.filter(exercise_type=spc_content_type)
        mcq_answered = db_answers.filter(exercise_type=mcq_content_type)

        # Save answers
        for exercise in answers:
            match exercise['type']:
                case 'repeat_phrase':
                    rp = RepeatPhrase.objects.get(pk=int(exercise['id']))

                    if update:
                        answer_update_obj =
rp_answered.filter(exercise_id=rp.pk)
                        if exercise['answer'] == '':
                            answer_update_obj.update(submited=submit)
                        else:
                            answer_update_obj.update(
                                answer=exercise['answer'],
```

```
                                submited=submit,
                            )
                    else:
                        answer_obj = UserAnswer.objects.create(
                            user_profile=user_profile,
                            story=story,
                            page=current_page,
                            exercise_type=ContentType.objects.get_for_model(rp
),
                            exercise_id=rp.pk,
                            answer=exercise['answer'],
                            submited=submit,
                        )
                case 'spellcheck':
                    spc = Spellcheck.objects.get(pk=int(exercise['id']))

                    if update:
                        answer_update_obj =
spc_answered.filter(exercise_id=spc.pk)
                        if exercise['answer'] == '':
                            answer_update_obj.update(submited=submit)
                        else:
                            answer_update_obj.update(
                                answer=exercise['answer'],
                                submited=submit,
                            )
                    else:
                        answer_obj = UserAnswer.objects.create(
                            user_profile=user_profile,
                            story=story,
                            page=current_page,
                            exercise_type=ContentType.objects.get_for_model(sp
c),
                            exercise_id=spc.pk,
                            answer=exercise['answer'],
                            submited=submit,
                        )
                case 'mc_question':
                    q =
MultipleChoiceQuestion.objects.get(pk=int(exercise['id']))

                    if update:
                        answer_update_obj =
mcq_answered.filter(exercise_id=q.pk)
                        if exercise['answer'] == '':
```

```
                        answer_update_obj.update(submited=submit)
                    else:
                        answer_update_obj.update(
                            answer=exercise['answer'],
                            submited=submit,
                        )
                else:
                    answer_obj = UserAnswer.objects.create(
                        user_profile=user_profile,
                        story=story,
                        page=current_page,
                        exercise_type=ContentType.objects.get_for_model(q)
,
                        exercise_id=q.pk,
                        answer=exercise['answer'],
                        submited=submit,
                    )


        # Get updated objects
        story_answers = UserAnswer.objects.filter(user_profile=user_profile,
story=story)
        db_answers = story_answers.filter(page=current_page)
        feedback_answers = map_answers(db_answers)

        response = {}
        response['answers'] = feedback_answers

        results = ""
        if evaluate:
            # Give form to answers
            answers = []
            if story_answers:
                evaluated = story_answers[0].evaluated
                if evaluated:
                    return JsonResponse({'message': 'This story is already
answered'})

                # Map all the story answers
                answers = map_answers(story_answers)

            results = evaluateAnswers(story, answers)

            #score_form = ScoreForm(request.POST or None)
            #if not score_form.is_valid():
```

```
            #    pass

            #score_form_obj = score_form.save(commit=False)

            score_form_obj = Score()

            score_form_obj.user_profile = user_profile
            score_form_obj.story = story

            score_form_obj.score = results["score"]
            score_form_obj.score_limit = results["score_limit"]
            score_form_obj.writing_percentage = results["writing_percentage"]
            score_form_obj.comprehension_percentage =
results["comprehension_percentage"]
            score_form_obj.speaking_percentage =
results["speaking_percentage"]

            score_form_obj.score_percentage = results['score_percentage']

            letter_grade = rateSkills(results['score_percentage'])
            results['letter_grade'] = letter_grade

            score_form_obj.save()

            user_profile.xp += results.get('score', 0)
            # update level
            lvl_obj = LevelManager()
            level_statistics = lvl_obj.get_level_statistics(user_profile.xp)
            user_profile.level = level_statistics['level']
            user_profile.save()

            # Update story answers to evaluated
            story_answers.update(evaluated=True)
            story_answers.update(submited=True)
            response['results'] = results

        return JsonResponse(response)
```

4.8.1 Algoritmo de similitud

Los ejercicios se califican diferente dependiendo de qué tipo de ejercicio sea, en este sentido, se creó un algoritmo de similitud entre dos oraciones que califica de una manera agradable al usuario.

Este algoritmo es utilizado por el ejercicio de repetir una frase y consiste en comparar dos textos de entrada, uno de ellos es la respuesta correcta y el otro es la respuesta proporcionada por el usuario.

Comienza creando una lista con cada texto proporcionado, por lo que se trabaja con dos listas de palabras en lugar de dos textos.

La lista con las respuestas del usuario UL y la lista con las respuestas correctas CL son iteradas de una manera peculiar.

Se comienza a iterar UL y dentro de esta iteración se itera CL, UL empieza a iterar desde el inicio y CL desde un punto de guardado SP (en un principio 0), por lo que su iteración debe ser con ayuda una variable auxiliar j que recorra la lista.

En cada iteración que ocurre dentro de CL, se comparan los valores de la iteración actual de UL u junto con el valor de la iteración actual de CL CL[j].

Si u y CL[j] coinciden, se aumenta un punto a una variable que almacena las palabras correctas cw, se establece el valor de SP en j + 1 y se rompe el ciclo de iteración interna, de esta forma se avanza con la iteración de UL y comienza la de CL desde el punto de guardado SP.

Una vez obtenida la cantidad de cw, se hacen operaciones matemáticas obtener una calificación agradable para el usuario, de tal manera que no se le brinde una calificación muy baja. Este ejercicio aumenta las estadísticas en “speaking”.

```
def evaluateRepeatPhrase(user_answer, correct_answer):
    if user_answer == "":
        results = {
            "score": 0,
```

```
            "speaking_percentage": 0
        }
        return results

    UL = user_answer.split(' ')
    CL = correct_answer.split(' ')

    total_user_words = len(UL)
    total_correct_words = len(CL)

    cw = 0
    penalty_words = total_user_words - \
        total_correct_words if total_user_words > total_correct_words else 0

    SP = 0

    for u in UL:
        for j in range(SP, len(CL)):
            if u == CL[j]:
                cw = cw + 1
                SP = j + 1
                break

    score = 0

    speaking_percentage = 0
    quit_points = 0
    if tolerance_error != 0:
        quit_points = penalty_words / (total_correct_words * tolerance_error)

    speaking_percentage = (
        cw / total_correct_words) - abs(quit_points)
    speaking_percentage = 0 if speaking_percentage < 0 else
speaking_percentage
    speaking_percentage = round(speaking_percentage, 2)

    score = points_per_correct_answer - quit_points *
points_per_correct_answer
    score = 0 if (score < 0) else score

    results = {
        "score": round(score, 2),
        "speaking_percentage": speaking_percentage
    }
    return results
```

En la figura 4.8.1.1 se visualizan diversos resultados de aplicar este algoritmo, suponiendo un nivel de tolerancia de 4 y los puntos por respuesta correcta como 100, nos podemos dar cuenta de que la puntuación del usuario es buena en general, sin embargo, el porcentaje no lo es, aquí es donde entran en juego las estadísticas pues, aunque el usuario se sienta satisfecho por obtener puntajes altos, nunca va a obtener un 100% en habilidad a menos que tenga una coincidencia total.

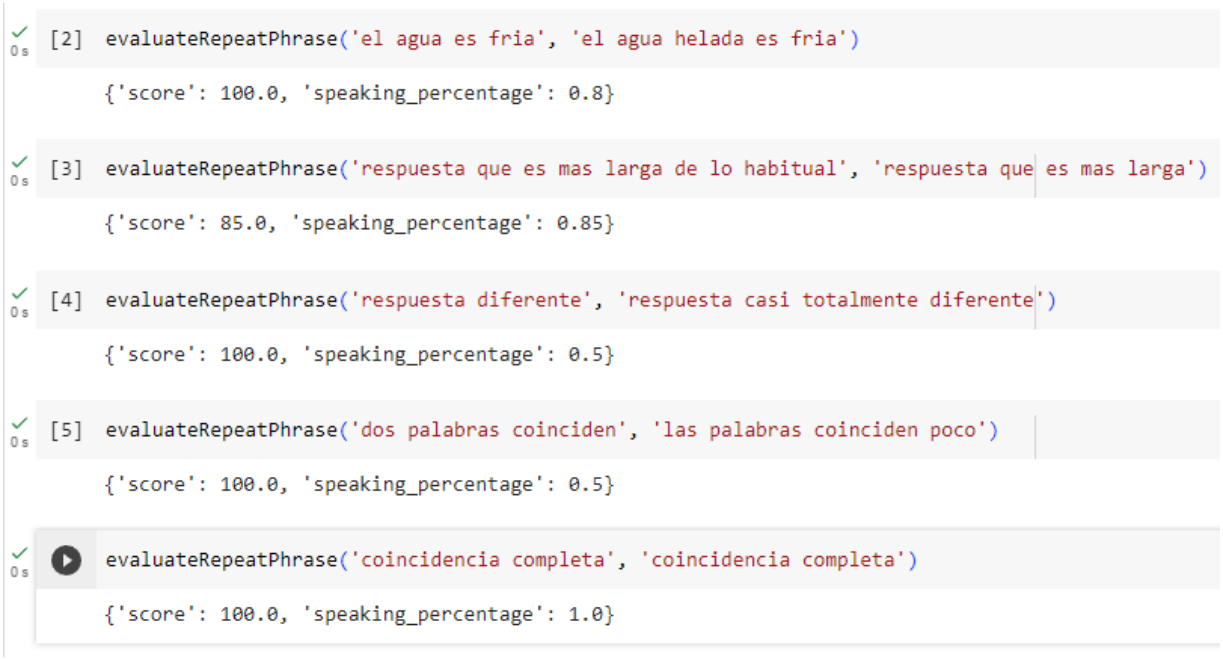

Figura 4.8.1.1. Resultado de evaluar con el algoritmo de similitud.

4.8.2 Algoritmo para subir de nivel

El algoritmo para calcular el nivel se basa en una función matemática, esta sección muestra como el comportamiento de dicha función puede resultar de gran ayuda en un algoritmo de programación.

Primero se define un valor constante c.

$$c = \frac{1}{\sqrt{10}}$$

Para calcular el nivel, se multiplica la constante por la raíz cuadrada de la experiencia del usuario y se le suma un 1.

$$l = c\left(\sqrt{x}\right) + 1$$

Naturalmente, es posible despejar esta ecuación para obtener la experiencia en lugar del nivel.

$$l = c\left(\sqrt{x}\right) + 1$$

$$l - 1 = c\left(\sqrt{x}\right)$$

$$\frac{l-1}{c} = \sqrt{x}$$

$$\left(\frac{l-1}{c}\right)^2 = x$$

$$\frac{l^2 - 2l + 1}{c^2} = x; \quad c = \frac{1}{\sqrt{10}}$$

$$\frac{l^2 - 2l + 1}{\frac{1}{10}} = x$$

$$x = 10(l^2 - 2l + 1)$$

Recapitulando, el algoritmo para obtener el nivel del usuario queda de la siguiente manera.

```
constant = 1/math.sqrt(10)
```

```
def get_level(xp):
  return constant*math.sqrt(xp) + 1
```

Con el siguiente algoritmo se obtiene la experiencia.

```
def get_xp(level):
  return (level**2 - 2*level + 1)/(constant**2)
```

También es posible optimizar la velocidad del código sustituyendo el valor de la constante, de esta manera el código tiene una operación menos que ejecutar.

```
def get_xp(level):
  return 10*(level**2 - 2*level + 1)
```

La razón detrás de elegir obtener una raíz cuadrada para hacer operaciones es simple, la función de raíz cuadrada tiene la propiedad de crecer más rápidamente en valores de x pequeños a comparación de valores de x más grandes. Para graficar se crea un par de puntos ordenados (x, y) siendo el primer punto x el valor de entrada y el segundo punto y el resultado de evaluar dicha entrada en la función.

El propósito de la constante c es ajustar el nivel, en las figuras 4.8.2.1 (F1) y 4.8.2.2 (F2) observamos la ejecución de un código con pequeñas diferencias. En ambos se crean dos listas de puntos xl y yl que posteriormente son graficados, sin embargo, F2 realiza la ejecución con el valor de la constante mientras que F1 no lo hace, como resultado F1 alcanza niveles de 100 con una experiencia base cercana a 10 mil, mientras que F2 apenas llega a sobrepasar el nivel 30 con la misma experiencia.

```python
import math
import matplotlib.pyplot as plt
fig, ax = plt.subplots()

xl = [i*100 for i in range(100)]
yl = [(math.sqrt(i*100)+1) for i in range(100)]

ax.plot(xl, yl)
plt.show()
```

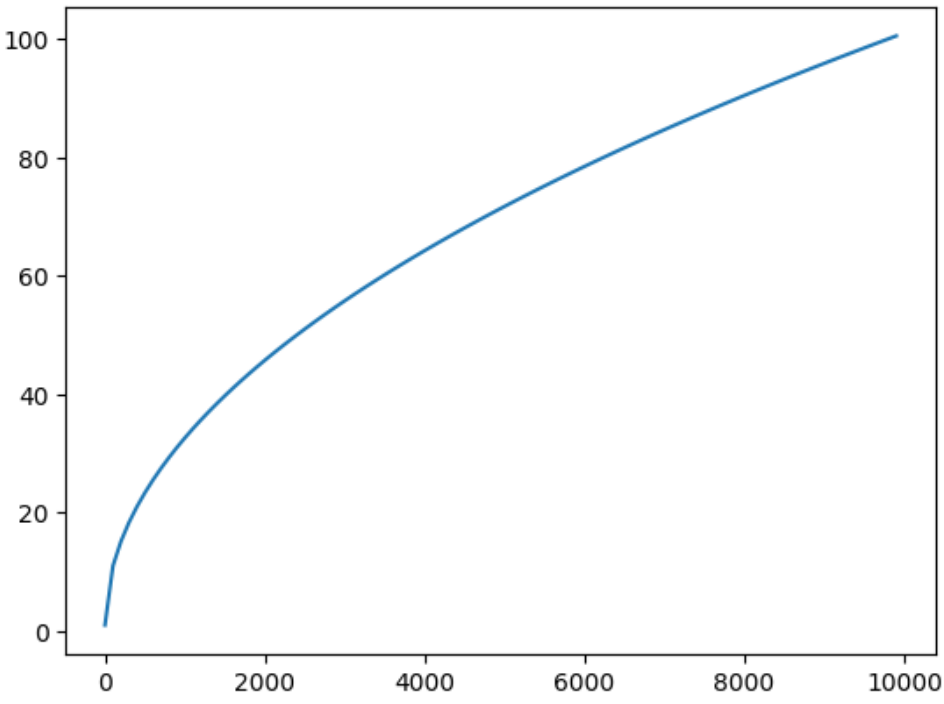

Figura 4.8.2.1. Gráfica del algoritmo implementado sin utilizar el valor de la constante.

```
import math
import matplotlib.pyplot as plt
fig, ax = plt.subplots()

c = 1/math.sqrt(10)
xl = [i*100 for i in range(100)]
yl = [(c*math.sqrt(i*100)+1) for i in range(100)]

ax.plot(xl, yl)
plt.show()
```

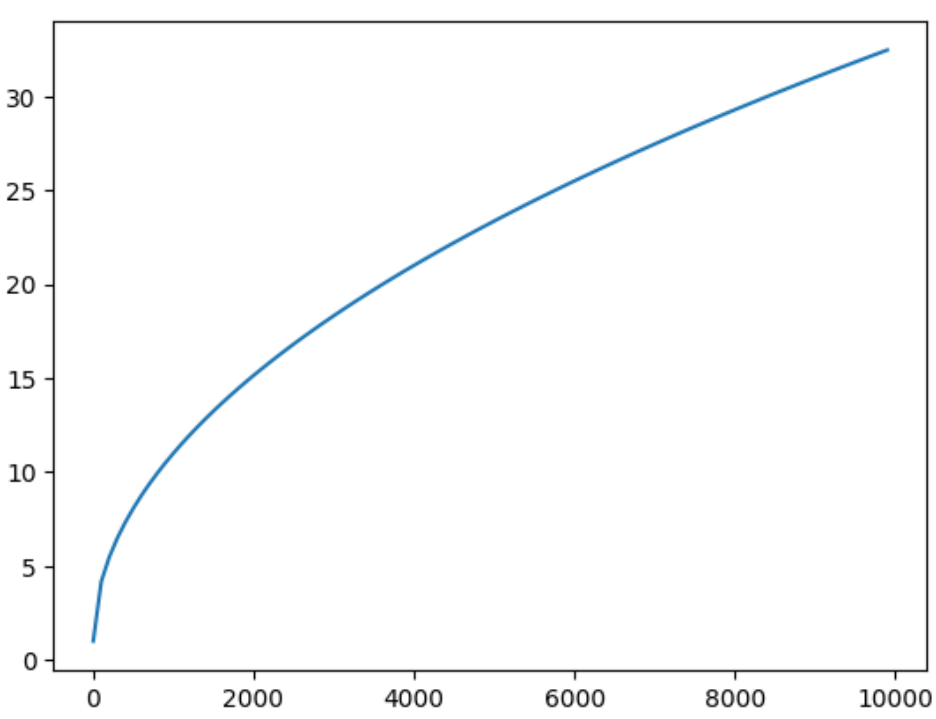

Figura 4.8.2.2. Gráfica del algoritmo implementado utilizando el valor de la constante.

Multiplicar por una constante se hace con el fin de evitar niveles muy altos, de esta forma el usuario no andará con miles de niveles en la aplicación y será visualmente más fácil para él recordar su nivel.

4.9 Sistema de recomendación basado en contenido

El algoritmo de recomendación basado en contenido sigue los siguientes pasos:

1. Recoge todas las palabras de la descripción, las etiquetas y el título de las historias, para obtener las palabras más relevantes.
2. Crea una matriz de similitud entre historias, es decir, historias similares entre ellas mismas y guarda esta matriz en memoria caché.
3. Ordena individualmente cada fila de la matriz de similitud de mayor a menor.
4. Recomienda contenido buscando la historia en la matriz de similitudes y devuelve las historias con mayor similitud.

El siguiente código en Python es el resultado de implementar el algoritmo de recomendación basado en contenido.

```
import pandas as pd

from django.core.cache import cache
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity


from main.models import Story, CBRSettings


class ContentBasedRecommender:
    def __init__(self) -> None:
        self.cache_key = 'cb_recommender_similarities'
        self.timeout = self.get_db_timeout(None)


    @staticmethod
    def get_db_timeout(default):
        settings = CBRSettings.objects.first()
        if not settings:
            return default

        timeout = settings.timeout
        if timeout < 30:
            timeout = None
```

```
        return timeout


    def update_timeout(self):
        similarities = cache.get(self.cache_key)
        if similarities:
            cache.set(self.cache_key, similarities, timeout=self.timeout)


    def train(self, max_data=5000):
        stories = Story.objects.all()
        if not stories:
            return False
        data = []
        for story in stories:
            tag_names = [tag.name1 for tag in story.tags.all()]
            tag_names_str = ' '.join(tag_names)
            data.append({
                'id': story.id,
                'title': story.title1,
                'tags': tag_names_str,
                'description': story.description1
            })

        stories = pd.DataFrame(data).fillna("")
        #print(f'Training...
dataframe\n{stories.head()}\n...{stories.tail()}')
        if len(stories) > max_data:
            stories = stories.sample(n=max_data)

        stories['data'] = stories['title'] \
                        + ' ' + stories['tags'] \
                        + ' ' + stories['description'].str.replace(r'\n', ' ',
regex=True)


        tfidf = TfidfVectorizer(analyzer='word', stop_words='english')
        stories_matrix = tfidf.fit_transform(stories['data'])
        cosine_similarities = cosine_similarity(stories_matrix)
        similarities = {}
        for i in range(len(cosine_similarities)):
            similar_indices = cosine_similarities[i].argsort()[:-50:-1]
            similarities[stories['id'].iloc[i]] = [(stories['id'][x],
cosine_similarities[i][x]) for x in similar_indices][1:]
```

```
        # Set the similarities into cache
        cache.set(self.cache_key, similarities, timeout=self.timeout)
        return True

    def recommend(self, story_id, max_recommendations=10):
        # Get the similarities from cache
        matrix_similarities = cache.get(self.cache_key)
        if not matrix_similarities:
            #print(f'\n\nThe recommender is not trained yet, training...\n\n')
            trainning = self.train()
            if trainning:
                return self.recommend(story_id=story_id,
max_recommendations=max_recommendations)
            return []
        #print(f'\nMatrix similarities\n{matrix_similarities}')

        # Recommendations is an array with the story id and similarity
predicted
        recommendations = matrix_similarities.get(story_id)
        #print(f'\n\n{recommendations}\n\n')

        recommendations_story = []
        if (recommendations):
            recommendations = recommendations[:max_recommendations]
            for r in recommendations:
                try:
                    recommendations_story.append(Story.objects.get(id=r[0]))
                except:
                    pass
        return recommendations_story
```

Python provee numerosas librerías que son muy útiles en ciencia de datos e inteligencia artificial, en este caso la función “TfidfVectorizer” de la librería sklearn hace uso del procesamiento del lenguaje natural, un campo de la inteligencia artificial, para obtener las palabras más importantes del conjunto de información brindado (título, descripción y etiquetas).

Este filtro se utiliza en una sección de “Relacionado”, disponible en la página que permite ver la información de una historia. El entrenamiento del mismo puede

realizarse cuando se actualiza una historia, es decir, cada que una historia es creada, modificada o eliminada.

4.10 Sistema de recomendación basado en un filtro colaborativo usuario a usuario

El algoritmo de filtro colaborativo usuario a usuario utilizado en esta aplicación funciona de la siguiente manera:

1. Recopila los id y títulos de todas las historias, también recopila todas las historias que los usuarios han marcado como favoritas y los junta en una sola tabla.
2. Calcula el total de “me gusta” recibidos por cada historia, los ordena en orden descendiente y los guarda en memoria caché.
3. Crea una matriz que contiene los puntajes que un usuario le da a una historia y guarda esta matriz en memoria caché.
4. Crea una matriz de similitud entre los usuarios, es decir, usuarios similares entre ellos mismos y la guarda en memoria caché.
5. Para recomendar a un usuario que no tiene “me gusta” registrados, se le envía como recomendación se le envían algunas de las historias más populares y algunas otras aleatoriamente.
6. Para recomendar historias a un usuario con interacciones, utiliza la matriz de similitudes para elegir a los diez usuarios con mayor compatibilidad. Posteriormente se basa en los gustos de estos usuarios similares para calcular un puntaje para cada historia, por último, hace un proceso similar a cuando un usuario no ha interactuado con historias, filtrando como recomendación algunas de las historias con los puntajes más altos y algunas otras aleatoriamente.

El objetivo de enviar historias aleatorias parte de la premisa de que el usuario puede tener algunos gustos que no ha explorado aún, además evita que el filtro le recomiende las mismas historias una y otra vez, previniendo una situación en la que el usuario tenga la sensación de quedarse siempre con lo mismo.

El siguiente código en Python es el resultado de implementar el algoritmo de recomendación basado en un filtro colaborativo de usuario a usuario.

```
import pandas as pd

from django.core.cache import cache
from sklearn.metrics.pairwise import cosine_similarity

from main.models import Story, LikedStory, UBRSettings


class UserBasedRecommender:
    def __init__(self) -> None:
        self.cache_most_rated_stories_key = 'ubr_most_rated_stories'
        self.cache_stories_key = 'ubr_stories'
        self.cache_cosines_key = 'ubr_cosines'
        self.timeout = self.get_db_timeout(1800)


    @staticmethod
    def get_db_timeout(default):
        settings = UBRSettings.objects.first()
        if not settings:
            return default

        timeout = settings.timeout
        if timeout < 30:
            timeout = None
        return timeout


    def update_timeout(self):
        most_rated_stories = cache.get(self.cache_most_rated_stories_key)
        stories_matrix = cache.get(self.cache_stories_key)
        cosines = cache.get(self.cache_cosines_key)
```

```
        if most_rated_stories is None or stories_matrix is None or cosines is
None:
            return

        cache.set(self.cache_most_rated_stories_key, most_rated_stories,
timeout=self.timeout)
        cache.set(self.cache_stories_key, stories_matrix,
timeout=self.timeout)
        cache.set(self.cache_cosines_key, cosines, timeout=self.timeout)


    def train(self):
        # Get stories
        stories = Story.objects.all().values('id', 'title1')
        stories = pd.DataFrame(list(stories))

        stories.rename(columns={'id': 'story_id'}, inplace=True)
        #print(f'Head stories:\n{stories.head()}')


        # Get ratings
        ratings = LikedStory.objects.all().values('user_profile', 'story')
        if ratings.count() == 0:
            return False # There aren't ratings in the database
        ratings = pd.DataFrame(list(ratings))
        ratings.rename(columns={'user_profile': 'user_profile_id','story':
'story_id'}, inplace=True)
        ratings['rating'] = 1
        #print(f'Head ratings:\n{ratings.head()}')


        # Merge tables
        df = pd.merge(ratings, stories, on='story_id', how='inner')
        #print(f'Head table:\n{df.head()}')


        # Count likes
        agg_ratings = df.groupby('story_id').agg(number_of_ratings =
('rating', 'count')).reset_index()
        agg_ratings_sorted = agg_ratings.sort_values(by='number_of_ratings',
ascending=False)
        cache.set(self.cache_most_rated_stories_key, agg_ratings_sorted,
timeout=self.timeout)
        #print(f'agg ratings\n{agg_ratings_sorted}')
```

```
        # Create matrix
        matrix = df.pivot_table(index='user_profile_id', columns='story_id',
values='rating').fillna(0)
        cache.set(self.cache_stories_key, matrix, timeout=self.timeout)
        #print(f'Matrix\n{matrix.head()}')


        # Get cosine similarities
        cosine_similarities = cosine_similarity(matrix)
        #print(f'Cosine similarities\n{cosine_similarities}')

        # Cosine similarities as dataframe
        cosine_similarities_df = pd.DataFrame(cosine_similarities,
index=matrix.index, columns=matrix.index)
        cache.set(self.cache_cosines_key, cosine_similarities_df,
timeout=self.timeout)

        return True

    # Return stories
    def recommend(self, user_id, max_recommendations=10):
        most_rated_stories = cache.get(self.cache_most_rated_stories_key)
        stories_matrix = cache.get(self.cache_stories_key)
        cosines = cache.get(self.cache_cosines_key)

        # Check if recommender is trained
        if most_rated_stories is None or stories_matrix is None or cosines is
None:
            #print(f'\n\nThe recommender is not trained yet, training...\n\n')
            trainning = self.train()
            if trainning:
                return self.recommend(user_id=user_id,
max_recommendations=max_recommendations)
            else:
                return Story.objects.none()

        # Integer to know how many similar user stories recommend
        strong_recommended = round(max_recommendations / 2)


        # Stories matrix only with user's row
        user_row = stories_matrix[stories_matrix.index == user_id]
        #print(f'picked row\n{user_row}')

        # If user has no likes return popular stories
```

```
        if user_row.empty:
            # Limit of stories to process
            most_rated_stories = most_rated_stories[:1000]
            #print(f'most_rated_stories \n{most_rated_stories}')

            # Get ids of most rated stories
            mr_ids = most_rated_stories['story_id'].tolist()
            # Crop most rated ids
            mr_ids = mr_ids[:strong_recommended]


            # Get random stories to fill the maximum
            random_recommended = max_recommendations - len(mr_ids)
            stories_random_ids =
Story.objects.all().values('id').exclude(pk__in=mr_ids).order_by('?')
            stories_random_ids = [x['id'] for x in stories_random_ids]
            # Crop random ids
            stories_random_ids = stories_random_ids[:random_recommended]


            # Prepare ids to search
            stories_to_search = stories_random_ids + mr_ids

            # Get querysets
            stories =
Story.objects.filter(pk__in=stories_to_search).order_by('?')
            return stories


        # Remove current user from matrix
        cosines.drop(index=user_id, inplace=True)
        similar_users =
cosines[cosines[user_id]>0][user_id].sort_values(ascending=False)[:10]


        # Stories matrix only without user's row
        similar_user_stories =
stories_matrix[stories_matrix.index.isin(similar_users.index)].dropna(axis=1,
how='all')
        #print(f'similar user stories\n{similar_user_stories}')


        item_score = {}
        excluded_ids = []
        # Iterate by columns
        for i, j in zip(similar_user_stories.columns, user_row.columns):
```

```
            # if user selected has rated story jumps to another column
            if user_row[j][user_id] > 0:
                excluded_ids.append(j)
                continue

            story_rating = similar_user_stories[i]

            total = 0
            # Similar users id

            for u in similar_users.index:
                score = similar_users[u] * story_rating[u]
                total += score

            item_score[i] = total

        # Convert dictionary to pandas dataframe
        item_score = pd.DataFrame(item_score.items(), columns=['story',
'score'])
        # Sort by score
        sorted_stories_ids = item_score.sort_values(by='score',
ascending=False)


        # Cut only required stories
        sorted_stories_ids = sorted_stories_ids[:strong_recommended]
        sorted_stories_ids = sorted_stories_ids['story'].tolist()

        # Recommend random stories
        random_recommended = max_recommendations - len(sorted_stories_ids)

        # search ids
        excluded_ids = excluded_ids + sorted_stories_ids
        stories_random_ids =
Story.objects.all().values('id').exclude(pk__in=excluded_ids).order_by('?')
        stories_random_ids = [x['id'] for x in stories_random_ids]

        # Crop stories
        stories_random_ids = stories_random_ids[:random_recommended]

        # Get stories queryset
        stories_to_search = sorted_stories_ids + stories_random_ids
        stories_recommended =
Story.objects.filter(pk__in=stories_to_search).order_by('?')
```

```
        return stories_recommended
```

Este tipo de filtrado es significativamente más complejo que el basado en contenido y su principal ventaja es que ofrece una recomendación personalizada para cada usuario, por otro lado, los algoritmos colaborativos tienden a tener problemas cuando hay pocos datos. Este escenario se tomó en cuenta y por esta razón si el usuario no ha interactuado con alguna historia se le muestran las historias que más han sido calificadas positivamente por los usuarios.

4.11 Implementar un sistema de reportes en tiempo real

Bastantes aplicaciones implementan funciones de tiempo real, en este caso se creó un sistema de reportes con dichas características, de esta forma un usuario puede reportar algún error en una historia y el reporte llegará directamente a un administrador sin necesidad de que este recargue la página.

4.11.1 Instalación

Django inicialmente utiliza un protocolo HTTP para manejar solicitudes entre cliente y servidor. La librería “Channels” nos permite utilizar otros protocolos en una aplicación de Django, entre ellos los WebSockets que sirven para establecer una comunicación bidireccional entre cliente y servidor, permitiendo el envío de mensajes desde cualquiera de las dos partes.

Channels necesita de dos aspectos fundamentales para funcionar, un servidor asíncrono y una “channel layer” para compartir información a través de la aplicación. Para el primer caso se utilizó el servidor Daphne y para el segundo el servidor Redis, para la instalación de ambos se realiza la ejecución de los comandos “pip install daphne” y “pip install channel_redis” respectivamente.

Una vez instaladas las librerías mencionadas, es necesario instalar y ejecutar el servidor Redis, en el ambiente Linux (Ubuntu 22.04.3 LTS x86_64) utilizado en la aplicación esto se hizo con los comandos “sudo apt-get update” y

“sudo apt-get install redis”. Redis corre en el puerto 6379 por defecto, pero esta información también es visible al ejecutar comando “systemctl status redis”.

4.11.2 Configuración

El en al archivo settings.py se realizan algunas configuraciones.

Se incluye ‘daphne’ en la parte superior de las aplicaciones instaladas y también se agrega ‘channels’.

```
INSTALLED_APPS = [
    'daphne',
    'django.contrib.admin',
    'django.contrib.auth',
    …
    'channels',
]
```

Se le indica la información de ASGI_APPLICATION y también se agrega la configuración de Redis, que recibe los valores del host y del puerto.

```
ASGI_APPLICATION = 'app.asgi.application'
CHANNEL_LAYERS = {
    "default": {
        "BACKEND": "channels_redis.core.RedisChannelLayer",
        "CONFIG": {
            "hosts": [("127.0.0.1", 6379)],
        },
    },
}
```

El archivo asgi.py se configura como se muestra.

```
import os
from channels.auth import AuthMiddlewareStack
from channels.routing import ProtocolTypeRouter, URLRouter
from channels.security.websocket import AllowedHostsOriginValidator
from django.core.asgi import get_asgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'app.settings')
# Initialize Django ASGI application early to ensure the AppRegistry
# is populated before importing code that may import ORM models.
django_asgi_app = get_asgi_application()

import main.routing

application = ProtocolTypeRouter({
  "http": django_asgi_app,
  "websocket": AllowedHostsOriginValidator(
    AuthMiddlewareStack(
            URLRouter(main.routing.websocket_urlpatterns)
        ),
  ),
})
```

4.11.3 Funcionamiento en el servidor

Las rutas de WebSockets toman como argumentos una dirección y una clase que herede de la clase WebSocketConsumer.

```
from django.urls import path

from . import consumers

websocket_urlpatterns = [
    path("ws/report-notifications/",
consumers.ReportNotificationsConsumer.as_asgi()),
]
```

El cliente en ningún momento envía información mediante un objeto de tipo WebSocket u algún otro medio, en su lugar, envía la información mediante un formulario a través de una solicitud HTTP para guardar el reporte en la base de datos y es Django quien desencadena este proceso al guardar el reporte.

En el siguiente fragmento de código se observa el archivo signals.py junto con la señal responsable de desencadenar el evento ‘report_created’ del grupo ‘report-notifications’, enviando como valor el id del reporte creado.

```
from django.dispatch import receiver
```

```
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from asgiref.sync import async_to_sync
from channels.layers import get_channel_layer
from main.models import UserProfile, StoryReport

…

@receiver(post_save, sender=StoryReport)
def send_notification_on_report_create(sender, instance, created, **kwargs):
  if created:
    channel_layer = get_channel_layer()
    group_name = 'report-notifications'
    event = {
      'type': 'report_created',
      'content': instance.id
    }
    async_to_sync(channel_layer.group_send)(group_name, event)
```

Para hacer funcionar las señales se necesita inicializarlas desde el archivo apps.py

```
from django.apps import AppConfig
from django.db.models.signals import post_migrate


class MainConfig(AppConfig):
    …
    def ready(self) -> None:
        …
        from . import signals
        return super().ready()
        …
```

La clase mostrada a continuación hereda de WebSocketConsumer e incluye los métodos “connect” y “disconnect” utilizados para conexión y desconexión respectivamente. Esta clase se conecta al grupo ‘report-notifications’ que es el grupo designado para compartir los mensajes de los reportes y crea otro método llamado “report_created” que busca el reporte mediante el id proporcionado y devuelve la plantilla ‘parts/reports_table.html’ renderizada con dicha información.

```
from channels.generic.websocket import WebsocketConsumer
from asgiref.sync import async_to_sync
from django.template import loader

from main.models import StoryReport


class ReportNotificationsConsumer(WebsocketConsumer):
  def connect(self):
    self.notifications_group_name = 'report-notifications'
    # Join room group
    async_to_sync(self.channel_layer.group_add)(
      self.notifications_group_name, self.channel_name
    )
    self.accept()


  def disconnect(self, close_code):
    async_to_sync(self.channel_layer.group_discard)(
      self.notifications_group_name, self.channel_name
    )

  def report_created(self, event):
    username = self.scope["user"]

    if username.is_staff or username.is_superuser:
      template = loader.get_template('parts/reports_table.html').render(
        context={
          'reports': StoryReport.objects.filter(pk=int(event['content'])),
            'new_report': True
        }
      )
      self.send(text_data=template)
```

4.11.4 Funcionamiento en el cliente

El cliente establece una conexión de WebSocket con el servidor, esta conexión puede llevarse a cabo de diversas maneras, en la aplicación se utiliza una forma de conexión para notificaciones y otra forma para renderizar el contenido de los reportes.

La conexión para las notificaciones se hace mediante el objeto “WebSocket”, este objeto utiliza la función “onmessage’ para ejecutar una función

cada que recibe una respuesta por parte del servidor. La función 'onmessage' implementada para el algoritmo de notificaciones aumenta una variable con el objetivo de contar los reportes no leídos y renderiza el número en forma de notificación.

```
/* Handle notifications */
reportNotificationsSocket = new
WebSocket(`ws://${window.location.host}/ws/report-notifications/`);

reportNotificationsSocket.onmessage = function (e) {
  let reports_number_html =
document.querySelector('#sidebar_report_icon_container .notification-pill-
container p');

  let reports_number = reports_number_html.innerText || 0;
  if (!reports_number.includes('+')) {
    if (reports_number == 0) {
      reports_number_html.parentElement.classList.remove('d-none');
    }
    if (reports_number == 99)
      reports_number_html.innerHTML = reports_number + '+';
    else
      reports_number_html.innerHTML = parseInt(reports_number) + 1;
  }
}
```

Otra forma de conectar WebSockets es utilizar librerías, la librería HTMX permite establecer una conexión con WebSockets de manera sencilla, para esto se toma un elemento HTML con el atributo "hx-ext" en un valor de "ws" y el atributo "ws-connect" con la dirección para conectar el WebSocket, dentro de la etiqueta con estos atributos debe haber un contenedor con un "id" que se utilizará para brindar una respuesta, en este caso el "id" toma el valor "response".

```
<div id="ws_report_notifications_div" class="m-auto main-content" hx-ext="ws"
ws-connect="/ws/report-notifications/">
  <div id="response">
    <!--Contenedor que espera una respuesta-->
  </div>
</div>
```

El retorna una plantilla con el mismo id y un atributo “hx-swap-oob” para indicar la posición donde el mensaje debe ser colocado, por ejemplo, el valor “afterbegin” renderiza el template al principio como si se tratara de una pila de platos, mientras que el valor “beforeend” renderiza después del último elemento renderizado.

```
<div id="response" hx-swap-oob="afterbegin">
  Este mensaje aparece siempre que el servidor envía datos
</div>
```

El código que se utiliza es bastante similar al mostrado previamente, se trata de una estructura de tabla que espera un id “reports_table_body”, la plantilla ‘reports_table.html’ no es únicamente para renderizar los mensajes del servidor, sino que también renderiza todos los reportes de la base de datos, sin embargo, la variable “new_report”, se utiliza para saber si ha sido renderizada desde la función del WebSocket para agregarle el atributo “hx-swapp-oob” y mostrarlo como un nuevo reporte al principio de la tabla.

```
{% load filters %}
<tbody id="reports_table_body" {% if new_report %} hx-swap-oob="afterbegin" {%
endif %}>
  {% for r in reports %}
  <tr class="report-{{ r.status }} report-row" id="report_{{ r.id }}"
report_id="{{ r.id }}"
    hx-get="{% url 'get_report' r.id %}" hx-target="#report_content" hx-
swap="innerHTML" hx-trigger="wait-for-function"
    onclick="openReport('{{ r.id }}')">
    <td onclick="selectReport(event, '{{ r.id }}')">
      <span class="check-icon" title="Seleccionar">
        <i class="fa-solid fa-check"></i>
      </span>
    </td>
    <td class="report-story-title-lg">{{ r.story.title1 }}</td>
    <td>
      <span class="report-story-title-sm report-{{ r.status }}">{{
r.story.title1 }}</span>
      {{ r.description|cut_str:20 }}
    </td>
    <td>{{ r.created_at|date:"d M" }}</td>
  </tr>
```

```
  {% endfor %}
</tbody>
```

Los objetos de HTMX también permiten la manipulación de eventos mediante JavaScript, por ejemplo, el siguiente fragmento de código es utilizado para evitar recibir reportes automáticos, el código revisa si la casilla de actualización automática está apagada o si hay algún otro motivo para no renderizar los reportes y detiene el renderizado automático.

```
// htmx
// handle notifications from messages
let ws_report_notifications_div =
document.getElementById('ws_report_notifications_div');
ws_report_notifications_div.addEventListener('htmx:wsBeforeMessage', (e) => {

  // First case: Auto update is turned off
  if (!document.getElementById('auto_update').checked) {
    e.preventDefault();
    return;
  }

  // Second case: check if is inside of unread or all messages to see changes.
  else {
    let tabSelected = document.querySelector('.tab-item.active');
    let tabSelectedShowing = tabSelected.getAttribute('show');
    if (tabSelectedShowing != 'all' && tabSelectedShowing != 'unread')
      e.preventDefault();
  }
});
```

4.11.4 Resultados

Se consiguió un sistema de administración bastante robusto y con diversas funcionalidades, además de que se logró la ejecución del sistema en tiempo real.

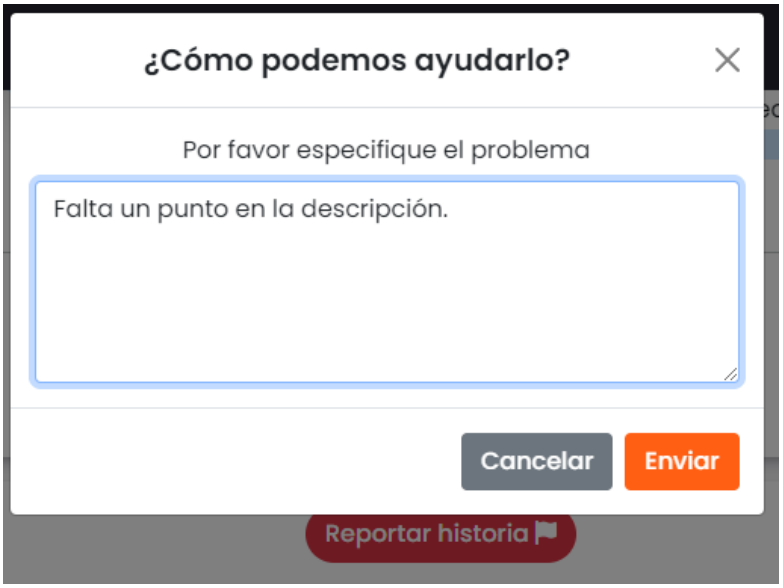

Figura 4.11.4.1. Formulario de reporte.

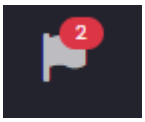

Figura 4.11.4.2. Notificación de reportes no leídos.

Figura 4.11.4.3. Sistema de administración de reportes.

CAPÍTULO V

EVALUACIÓN

Para evaluar el funcionamiento correcto de la aplicación se probó en diferentes ambientes una vez que estaba en producción, gracias a esto se pudo verificar el correcto funcionamiento de la misma.

De la misma forma, las pruebas evaluadas evidencian que la aplicación cumple con las funciones establecidas en un principio. Esto ha sido posible gracias a la ejecución de pruebas definidas aquí como funcionales y visuales.

Funcionales: son pruebas que comprueban el correcto flujo de la aplicación.

Visuales: estas pruebas evalúan que la aplicación se visualice correctamente.

5.1 Página de inicio

Esta sección prueba la página de inicio, que incluye autenticación y registro.

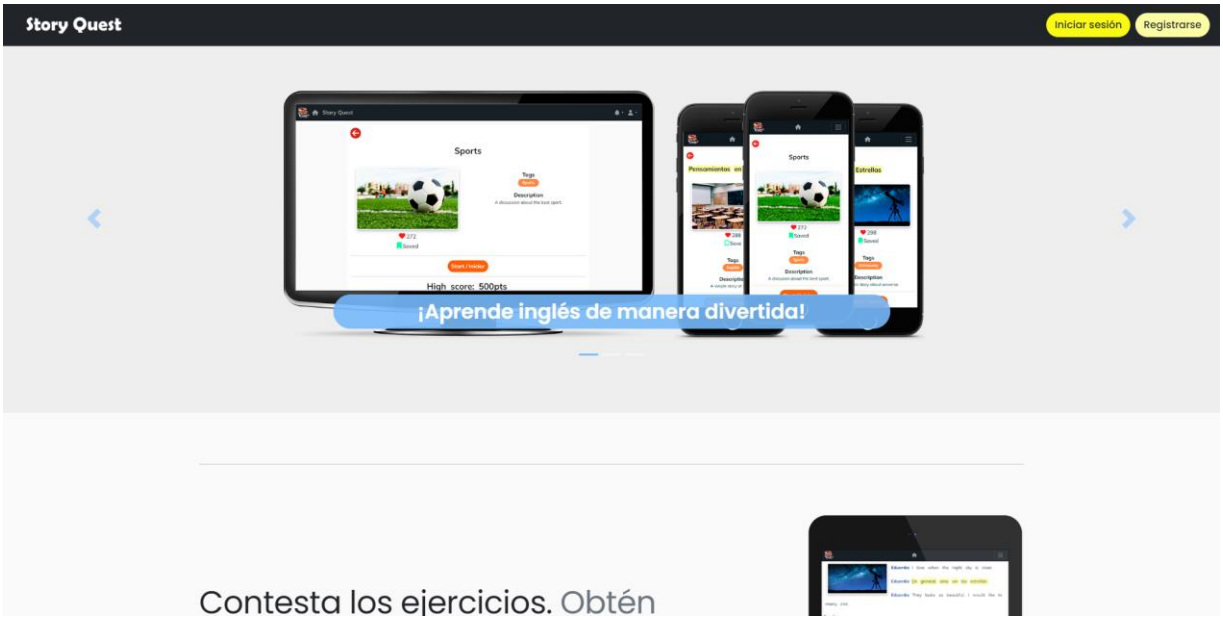

Figura 5.1.1. Prueba de pantalla de presentación (1).

Figura 5.1.2. Prueba de pantalla de presentación (2).

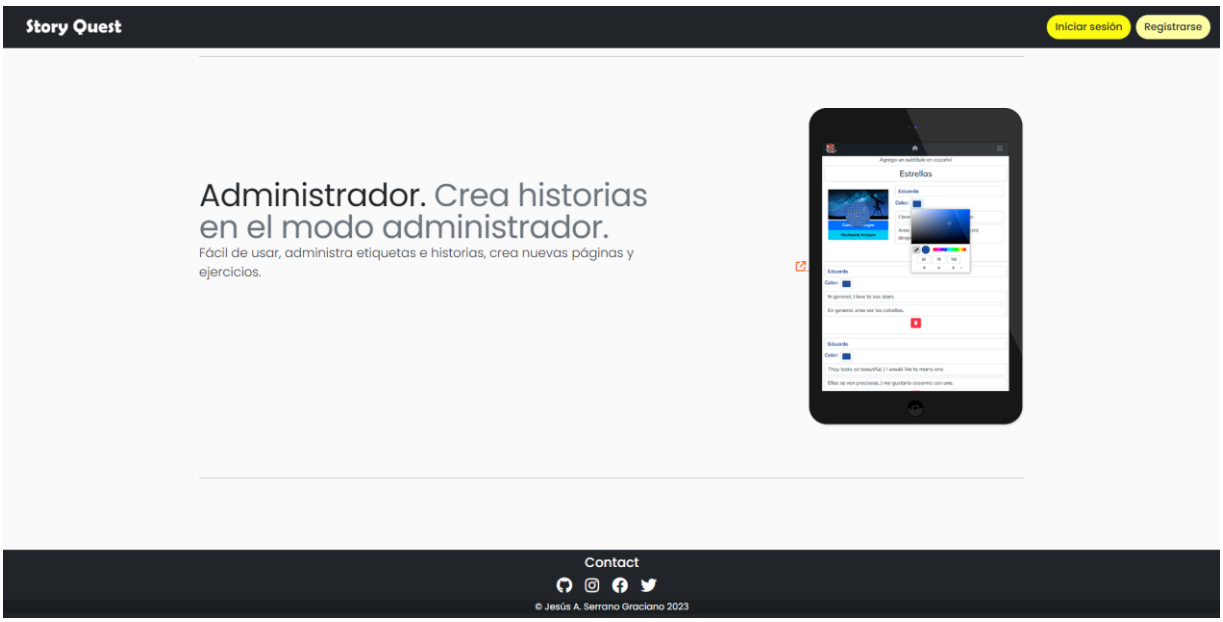

Figura 5.1.3. Prueba de pantalla de presentación (3).

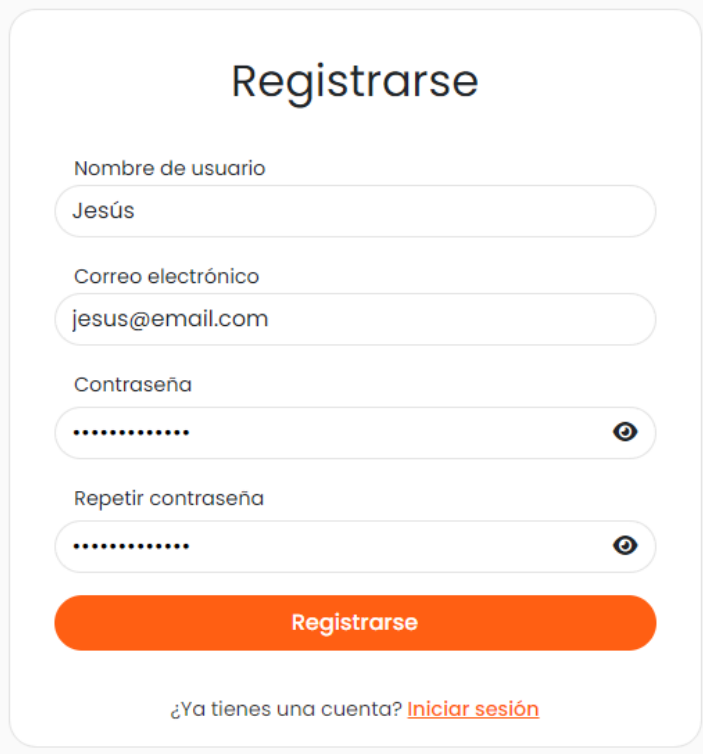

Figura 5.1.4. Prueba de registro.

Figura 5.1.5. Prueba de inicio de sesión.

En esta sección se pudo comprobar el funcionamiento correcto de la estructura de la aplicación, así como las acciones que debían realizar, en la figura 5.1.4 se observa como el botón para mostrar contraseña no ha sido activado por lo que la contraseña no se muestra, en cambio, en la figura 5.1.5 si se aprecia este cambio.

5.2 Roles

A continuación, se muestra la pantalla principal cuando se accede a la aplicación.

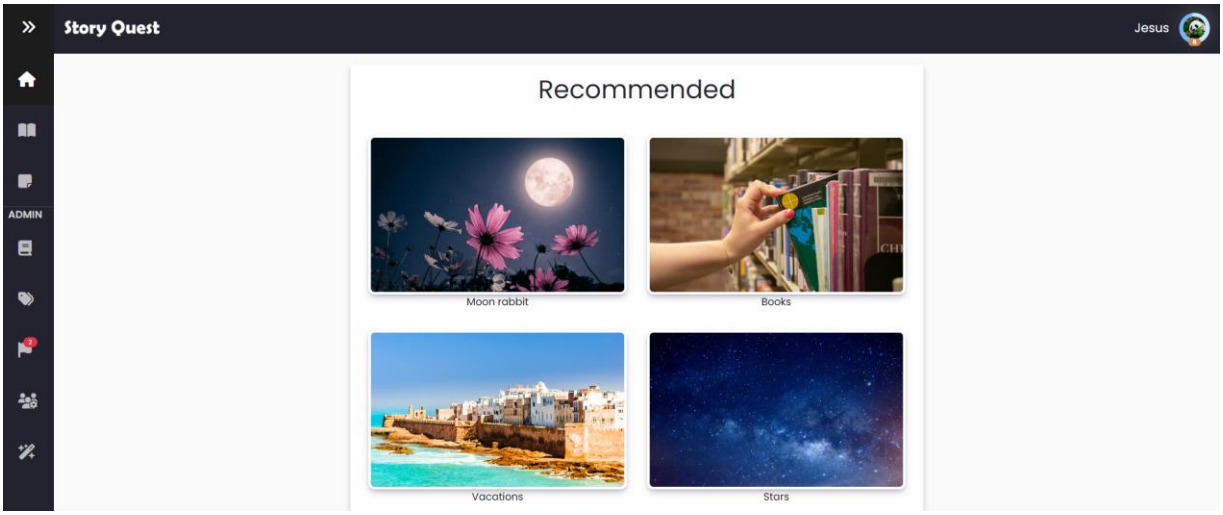

Figura 5.2.1. Prueba de entrar a la aplicación (1).

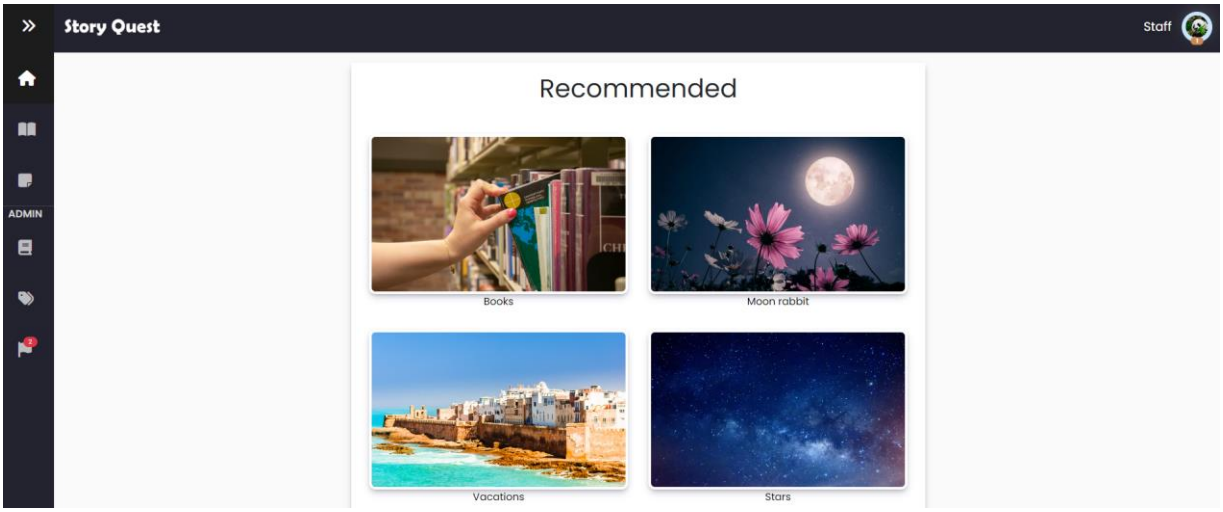

Figura 5.2.2. Prueba de entrar a la aplicación (2).

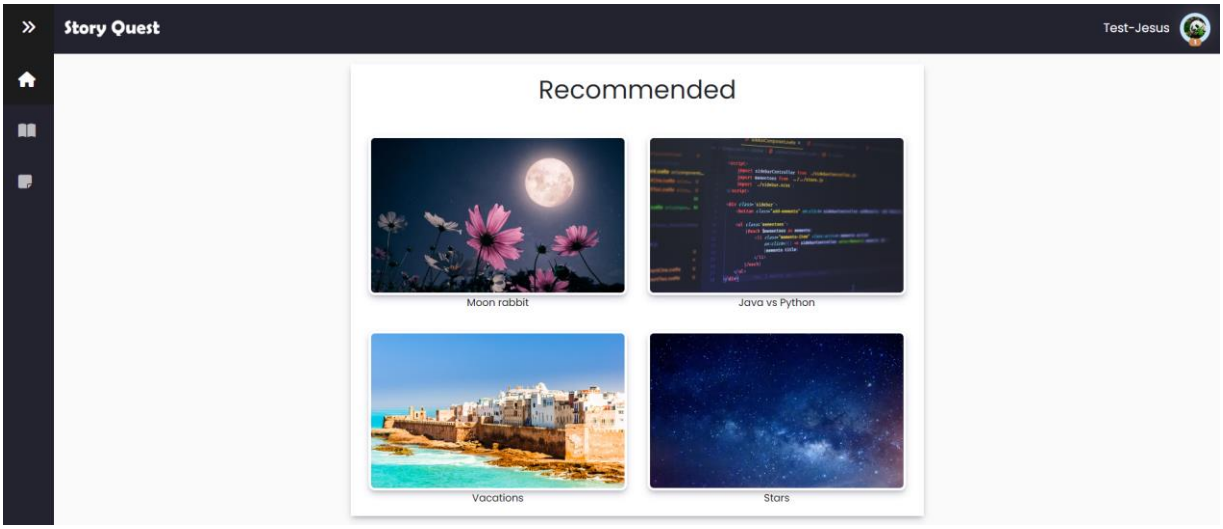

Figura 5.2.3. Prueba de entrar a la aplicación (3).

Podemos dar por buena esta prueba, en la figura 5.2.1 se puede apreciar el panel de un usuario administrador y se observa que se le muestra todo el apartado administrativo, posteriormente en la figura 5.2.2 se observa un usuario con permisos de staff, para terminar, se observa un usuario sin permisos administrativos en la figura 5.2.3 el cual no contiene ninguna función de administración disponible en el panel izquierdo.

5.3 Agregar historias y etiquetas

Para esta prueba se agregaron etiquetas e historias para verificar que el flujo de la aplicación. También se revisó el correcto funcionamiento de las imágenes, que estas se mantuvieran correctamente durante una actualización o que se eliminaran si se daba el caso.

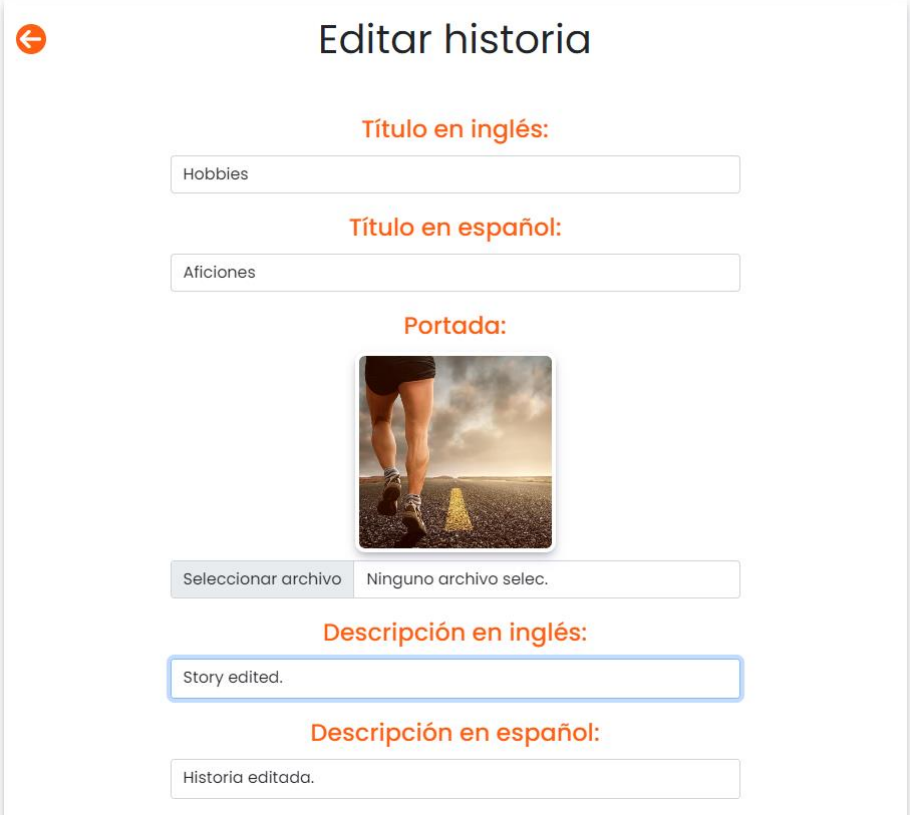

Figura 5.3.1. Prueba para crear o editar una historia.

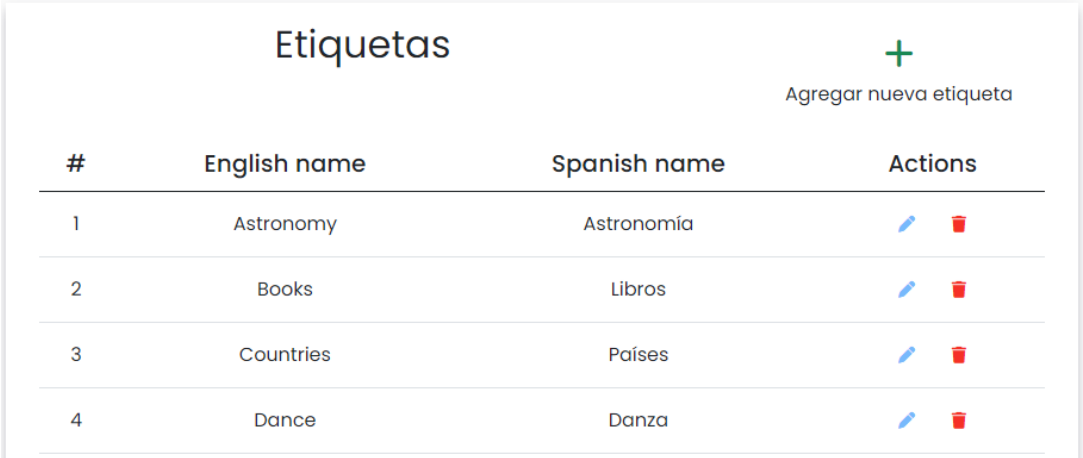

Figura 5.3.2. Prueba de etiquetas.

5.4 Agregar páginas de historia

Se comprobó el correcto funcionamiento al agregar, editar y eliminar páginas de una historia, así como las mismas funciones para los elementos dentro de una página.

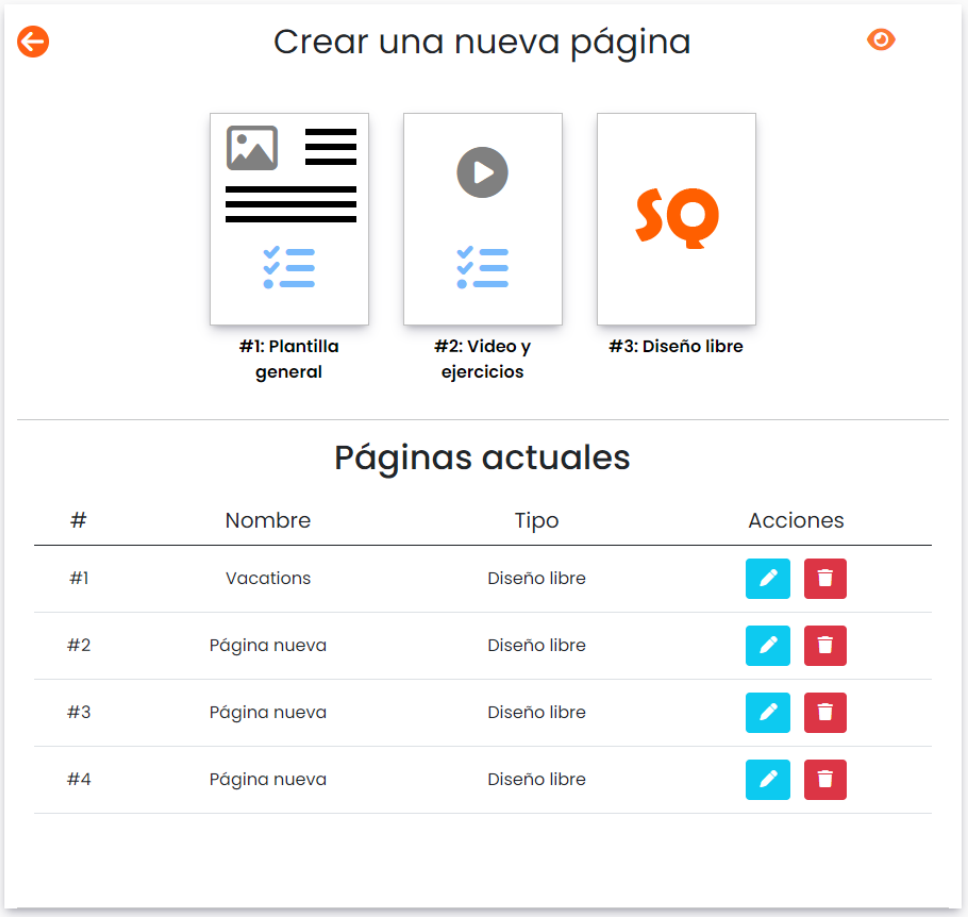

Figura 5.4.1. Prueba de páginas de una historia.

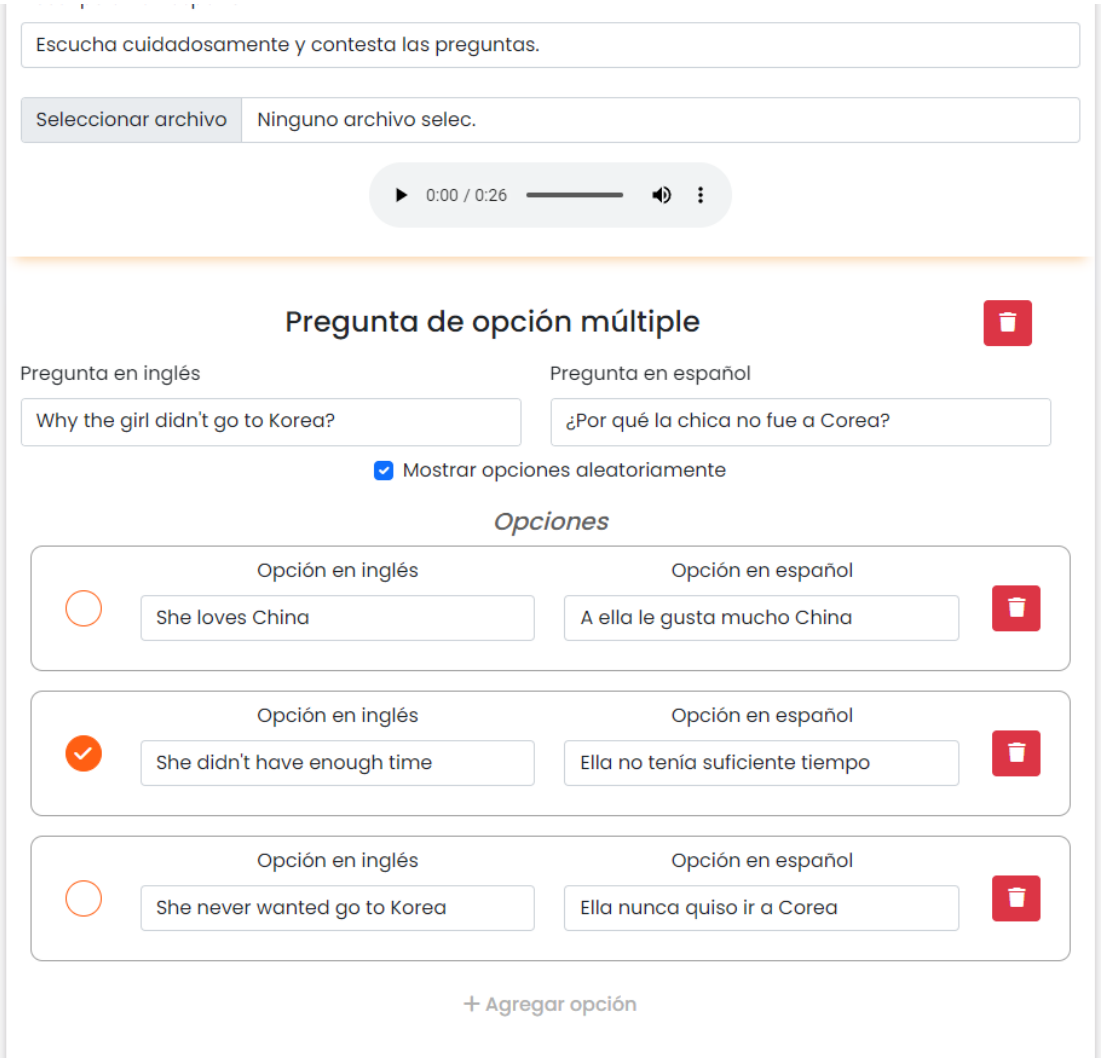

Figura 5.4.2. Prueba de elementos de la página de una historia.

5.5 Contestar historia y sus implicaciones

Para las pruebas realizadas en esta sección, se contestaron los tres tipos de ejercicios, el resultado esperado es obtener puntuaciones más altas o más bajas dependiendo de lo contestado.

Contestar una historia implica obtener retroalimentación del ejercicio contestado, cuando una historia es evaluada se obtiene una gráfica con los porcentajes de las habilidades, la puntuación obtenida se guarda y se muestra en una tabla de puntuaciones de la historia junto con la gráfica de la puntuación más alta. Además en la página del perfil se muestra la historia con su mayor puntuación

y la experiencia del usuario aumenta, si aumenta lo suficiente el usuario sube de nivel.

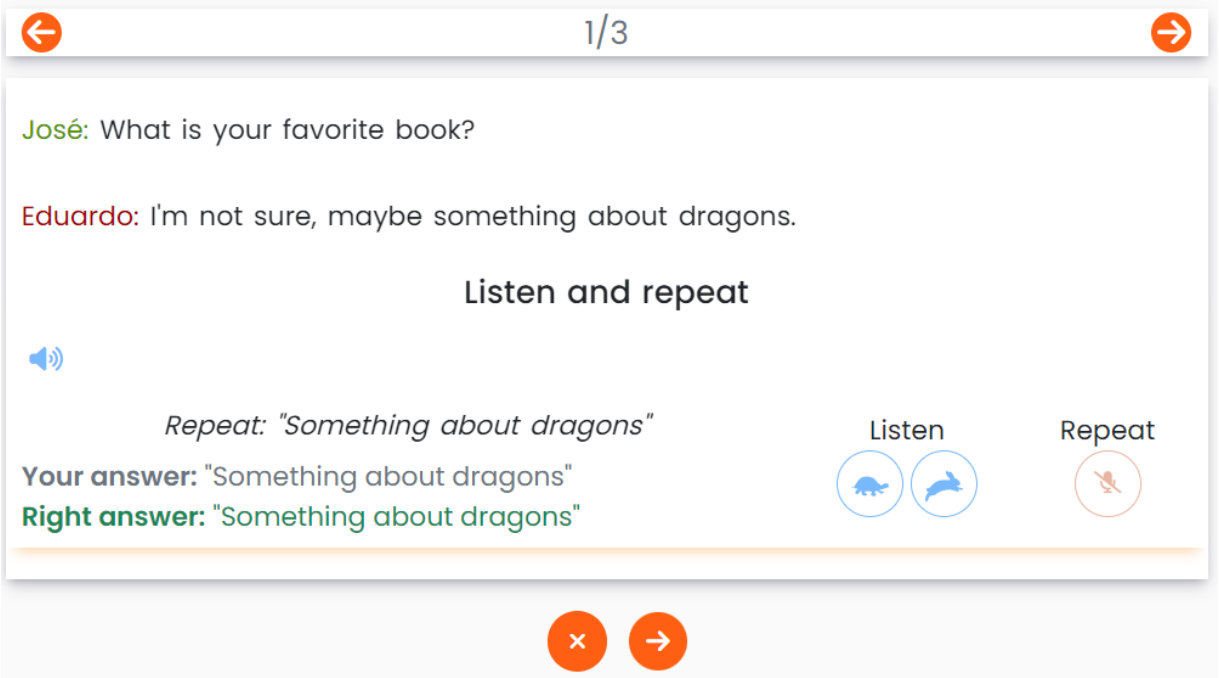

Figura 5.5.1. Prueba resolviendo correctamente ejercicios (1).

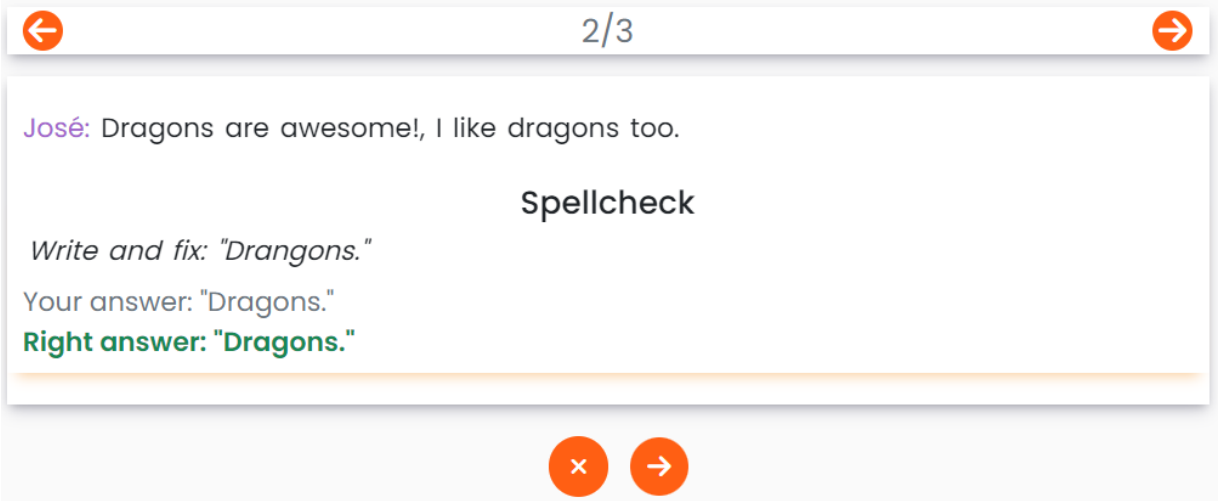

Figura 5.5.2. Prueba resolviendo correctamente ejercicios (2).

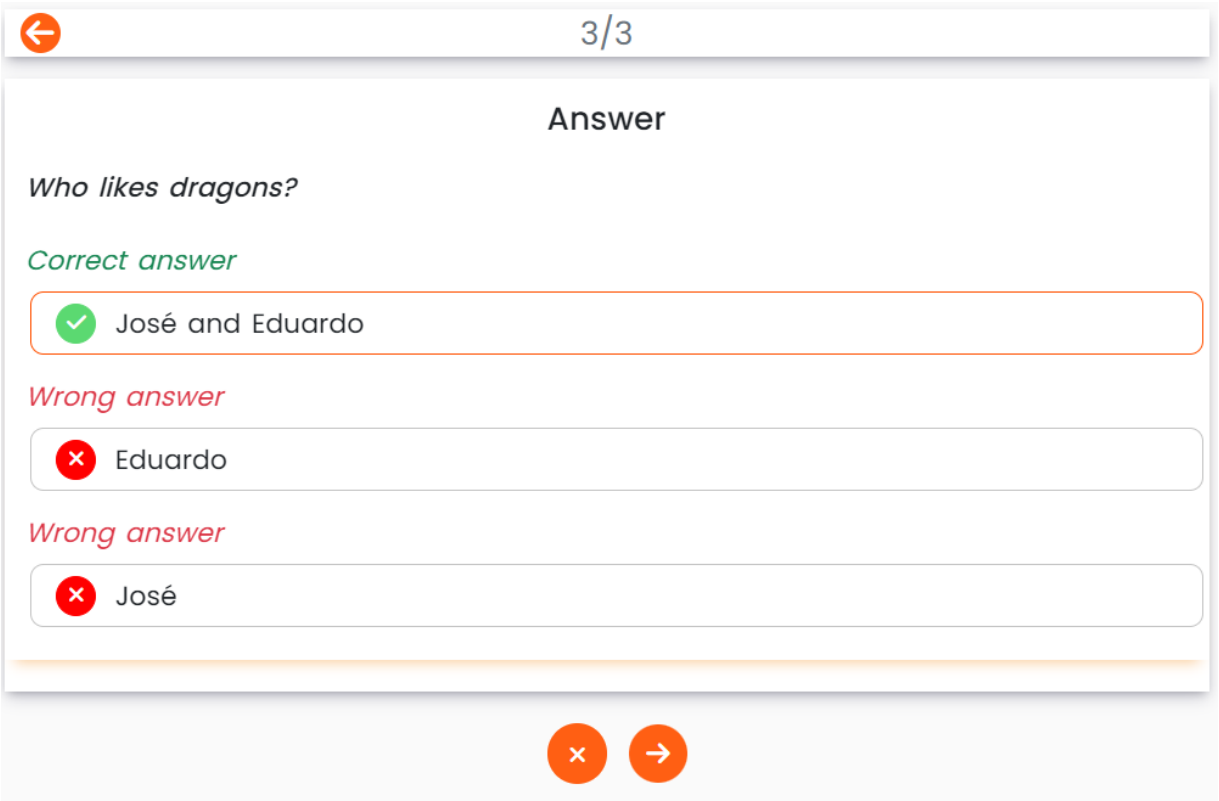

Figura 5.5.3. Prueba resolviendo correctamente ejercicios (3).

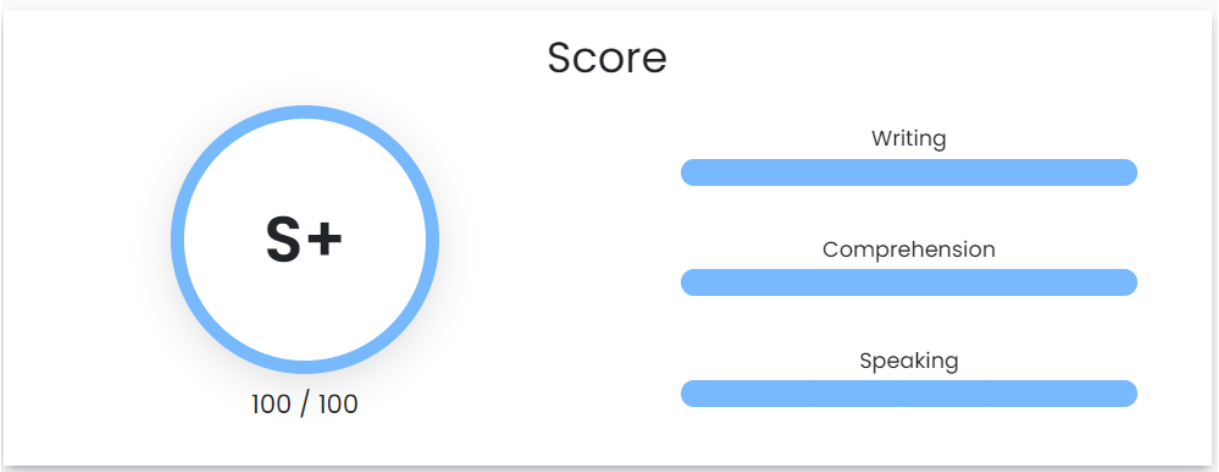

Figura 5.5.4. Gráfica al resolver todo correctamente.

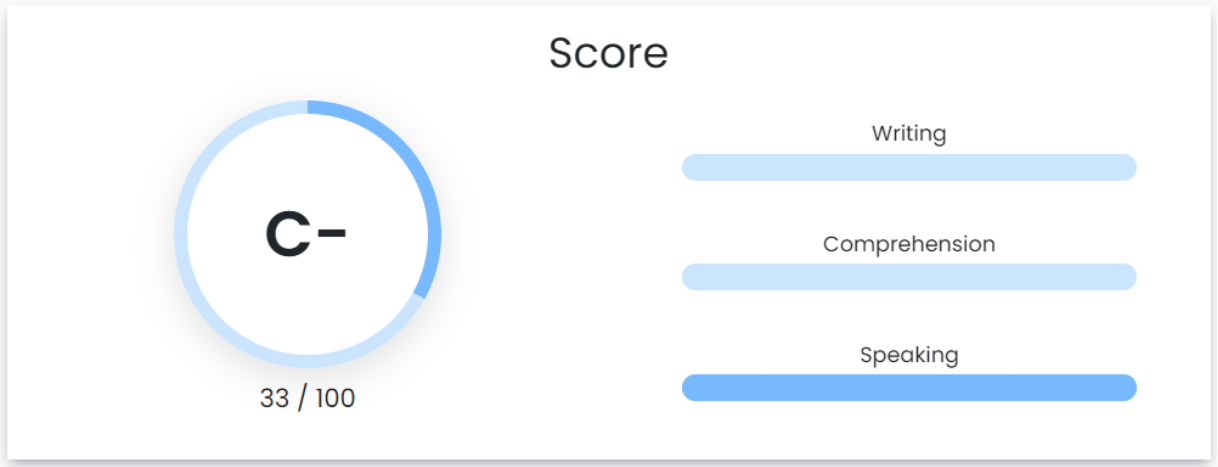

Figura 5.5.5. Gráfica al resolver únicamente el ejercicio de repetir oración de manera correcta.

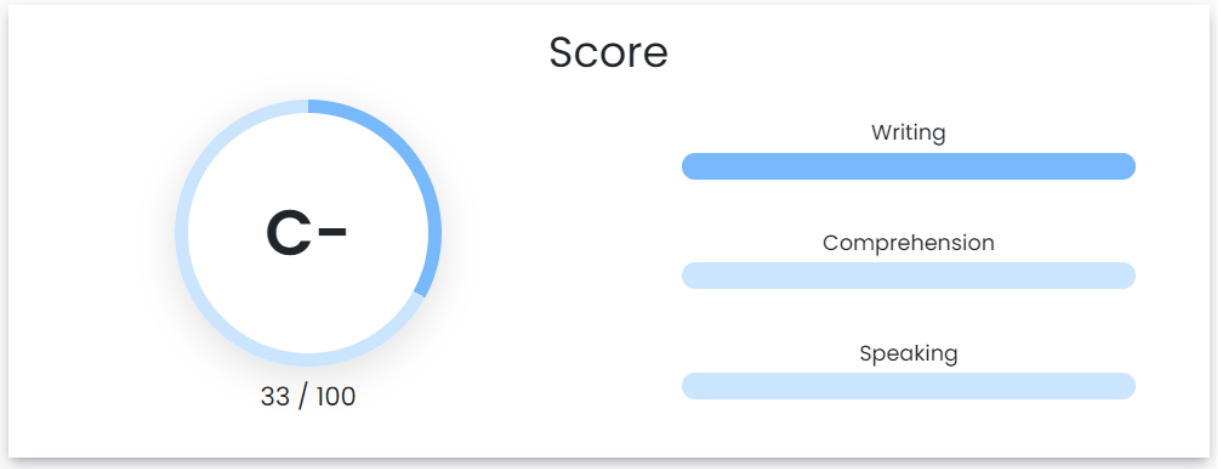

Figura 5.5.6. Gráfica al resolver únicamente el ejercicio de arreglar texto de manera correcta.

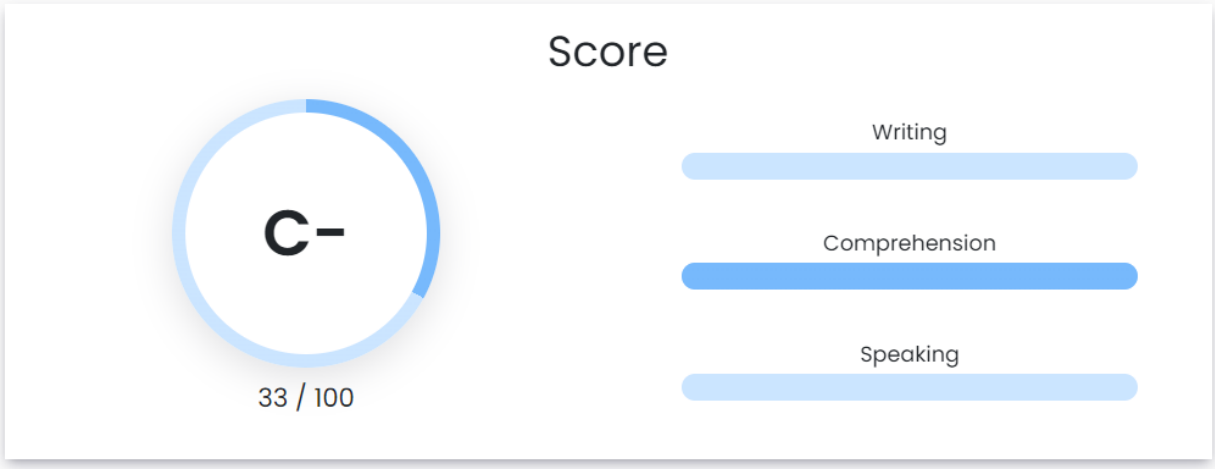

Figura 5.5.7. Gráfica al resolver únicamente el ejercicio de pregunta de opción múltiple de manera correcta.

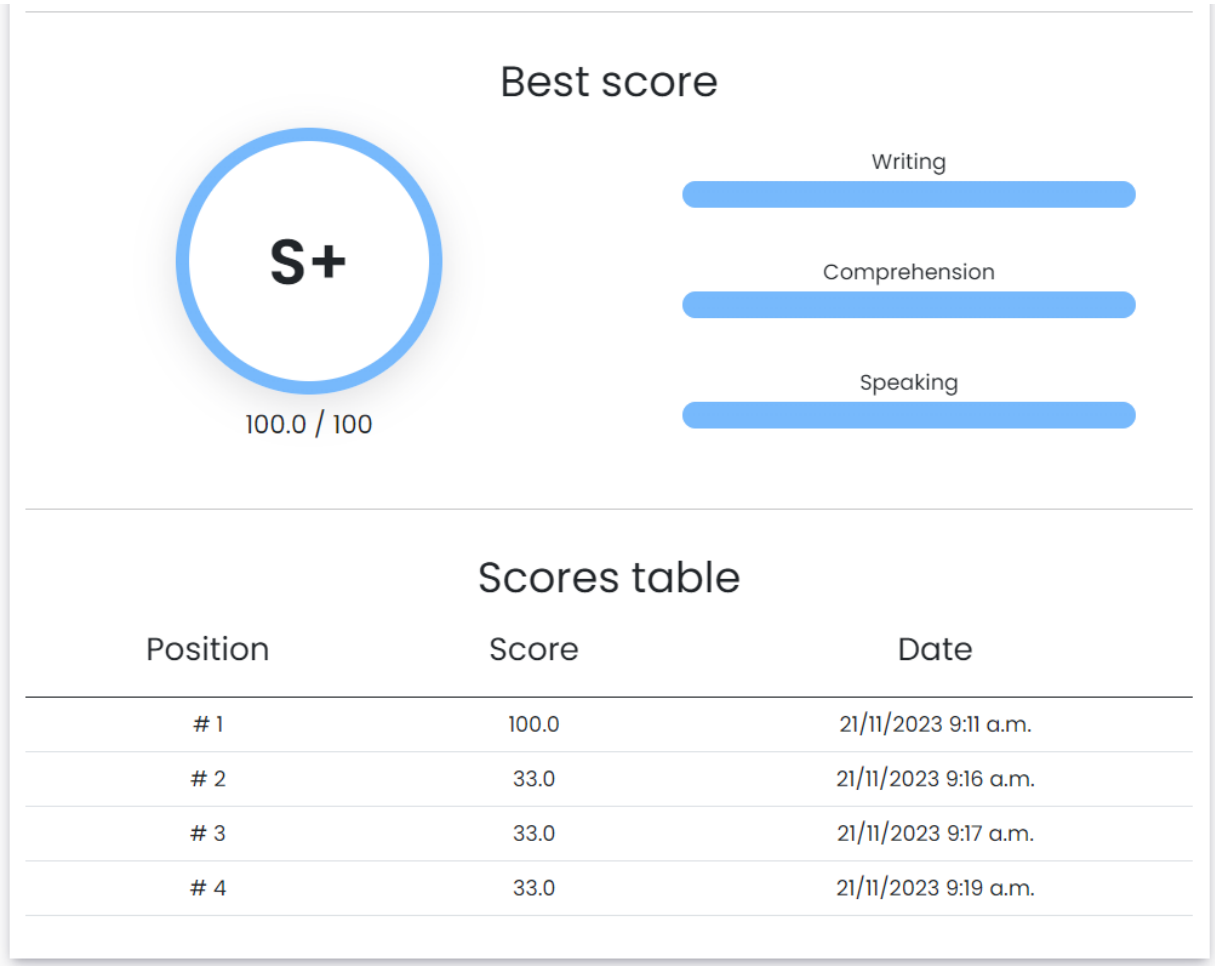

Figura 5.5.8. Prueba de la gráfica de mejor puntuación y tabla de puntuaciones.

Los resultados fueron los esperados y no se encontraron errores.

5.6 Perfil de usuario

La página de perfil de usuario muestra estadísticas sobre la interacción del usuario con las historias, además contiene una imagen personalizable que se muestra dentro de una barra de progreso circular que a su vez contiene el nivel y la experiencia actual del usuario.

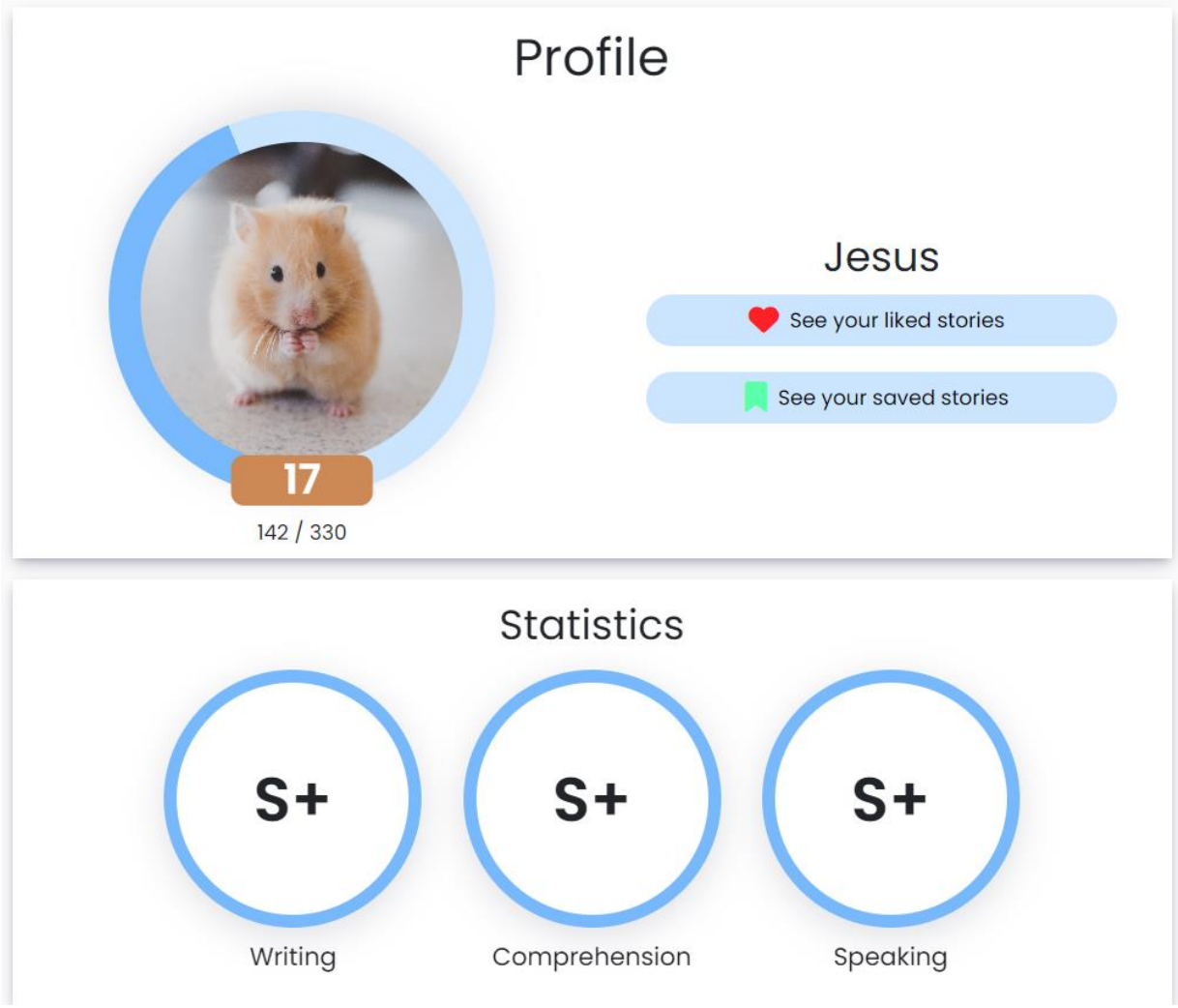

Figura 5.6.1. Página de perfil de usuario (1).

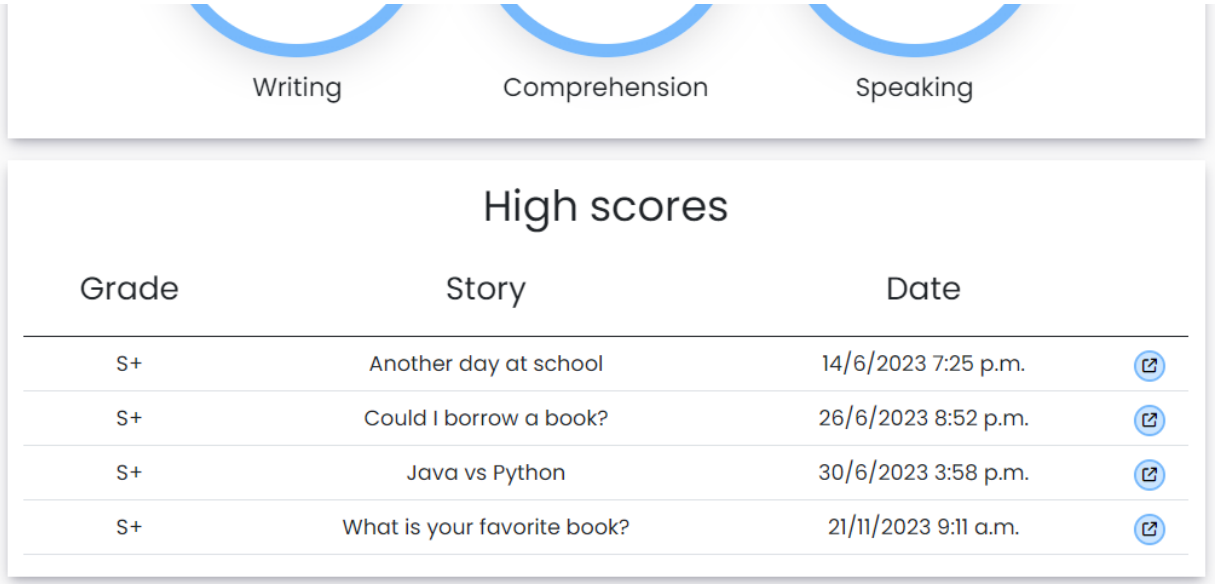

Figura 5.6.2. Página de perfil de usuario (2).

Los resultados fueron los esperados, tomando como base la figura 5.5.8 que se muestra en la sección anterior, podemos observar que la hora y fecha de la mejor puntuación coinciden con la mejor puntuación de la figura 5.6.2.

5.7 Flashcards

Los flashcards o tarjetas de estudio permiten al usuario realizar pequeñas notas de una manera organizada y eficiente. Para esto un usuario debe crear una colección de flashcards para almacenar su tarjeta de estudio dentro. Se probaron los métodos de crear, eliminar y actualizar tanto flashcards como colecciones de estas con el objetivo de identificar posibles errores.

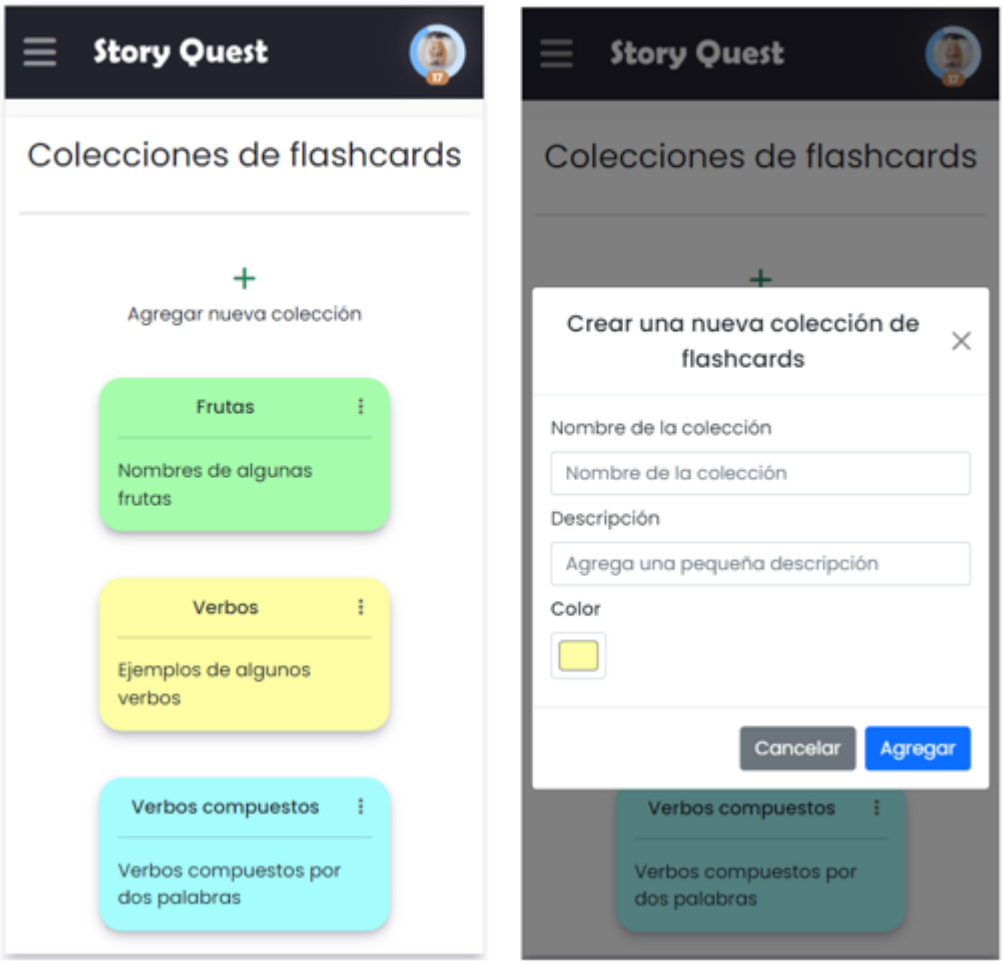

Figura 5.7.1. Vista móvil de las colecciones de flashcards y diálogo para agregar una nueva.

Figura 5.7.2. Flashcards de frutas (frente).

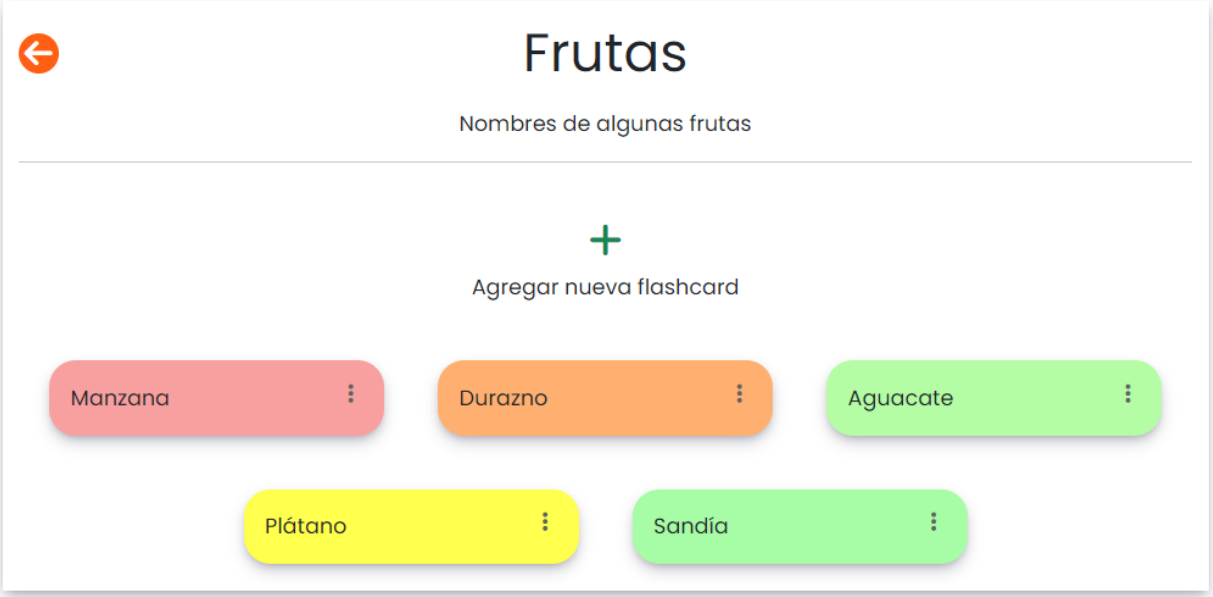

Figura 5.7.3. Flashcards de frutas (reverso).

Los resultados fueron los esperados.

5.8 Filtros de recomendación

En esta parte se probaron los dos filtros de recomendación, el basado en contenido y el basado en usuario.

5.8.1 Filtro basado en contenido

Se agregaron bastantes historias con diferentes etiquetas y se probó su funcionamiento, en la figura 5.8.1.2 se observa el resultado del filtro basado en contenido cuando se accede a una historia con la etiqueta “Sports”. En la figura 5.8.1.2 se aprecia otro ejemplo de una historia con las etiquetas “Astronomy”, “Fantasy” y “Movies”, recomendando historias con contenido similar.

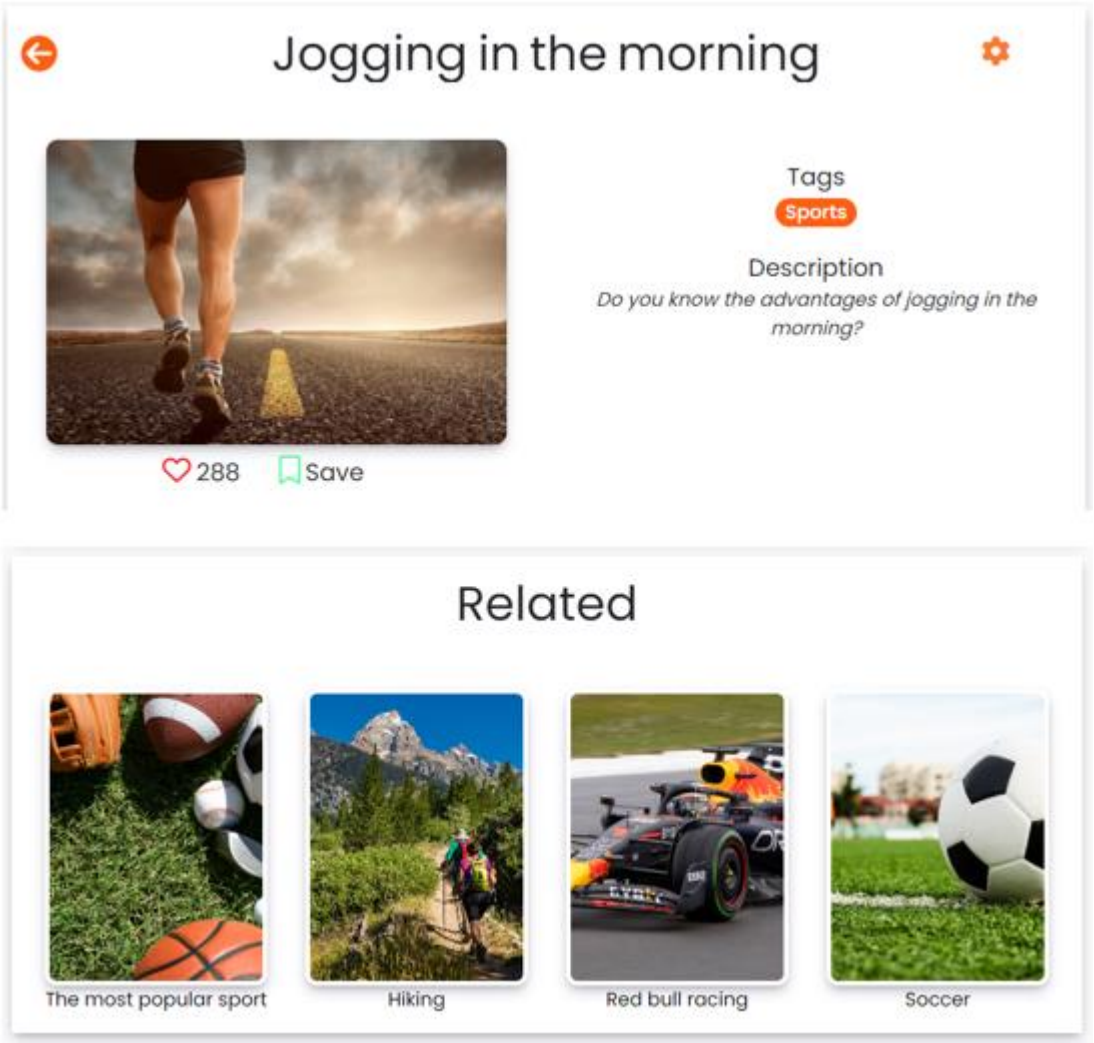

Figura 5.8.1.1. Prueba de filtro basado en usuario (1).

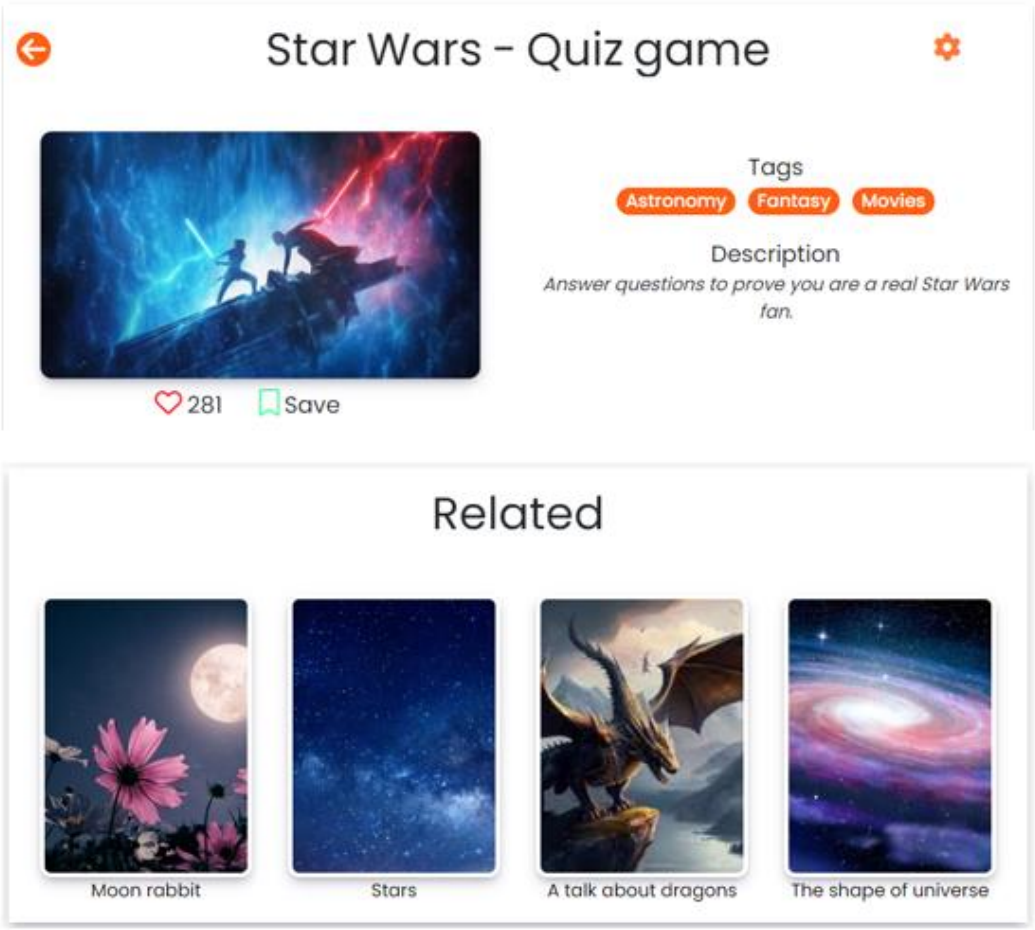

Figura 5.8.1.2. Prueba de filtro basado en usuario (2).

5.8.2 Filtro colaborativo basado en usuario

Para realizar estas pruebas se crearon dos usuarios, Rodrigo y Daniel.

Se pretende que Daniel y Rodrigo se recomienden contenido entre sí, para esto se hizo que en un principio ambos tuvieran los mismos gustos, marcando las mismas historias como favoritas y acto seguido ir añadiendo más historias favoritas a su colección.

El resultado esperado es que Rodrigo obtenga como recomendación historias que Daniel ha marcado como favoritas y viceversa.

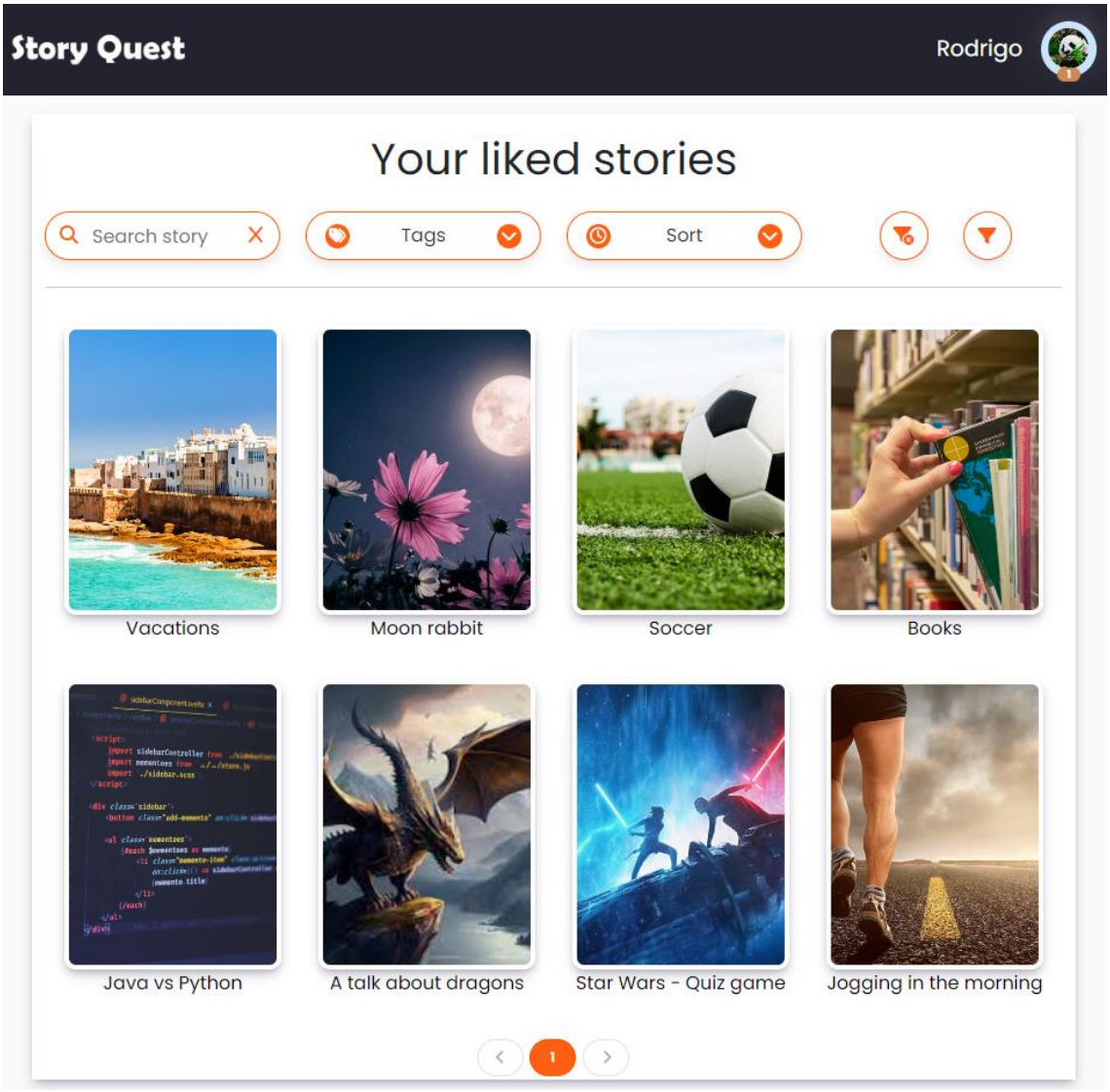

Figura 5.8.2.1. Historias favoritas de Rodrigo.

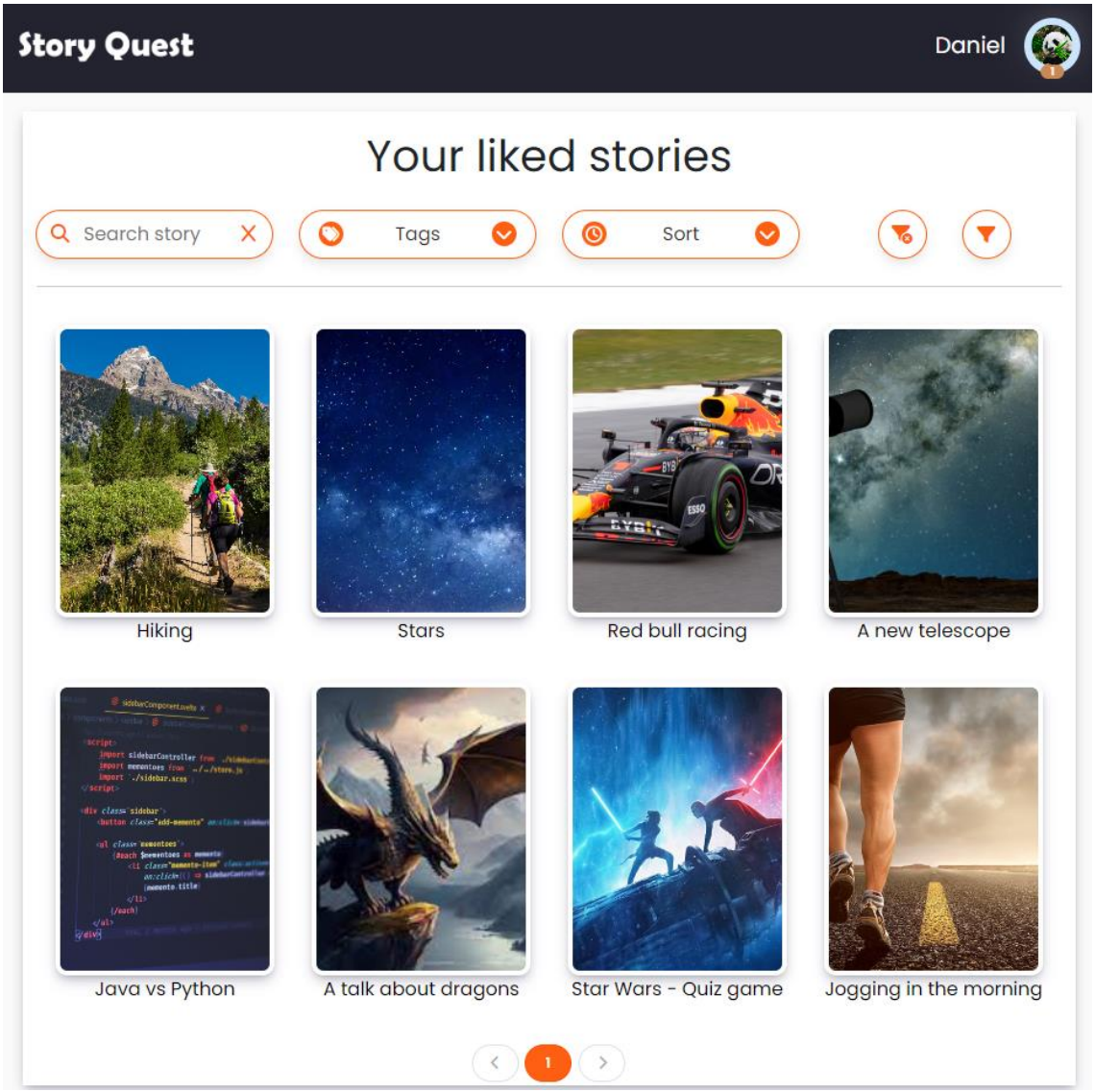

Figura 5.8.2.2. Historias favoritas de Daniel.

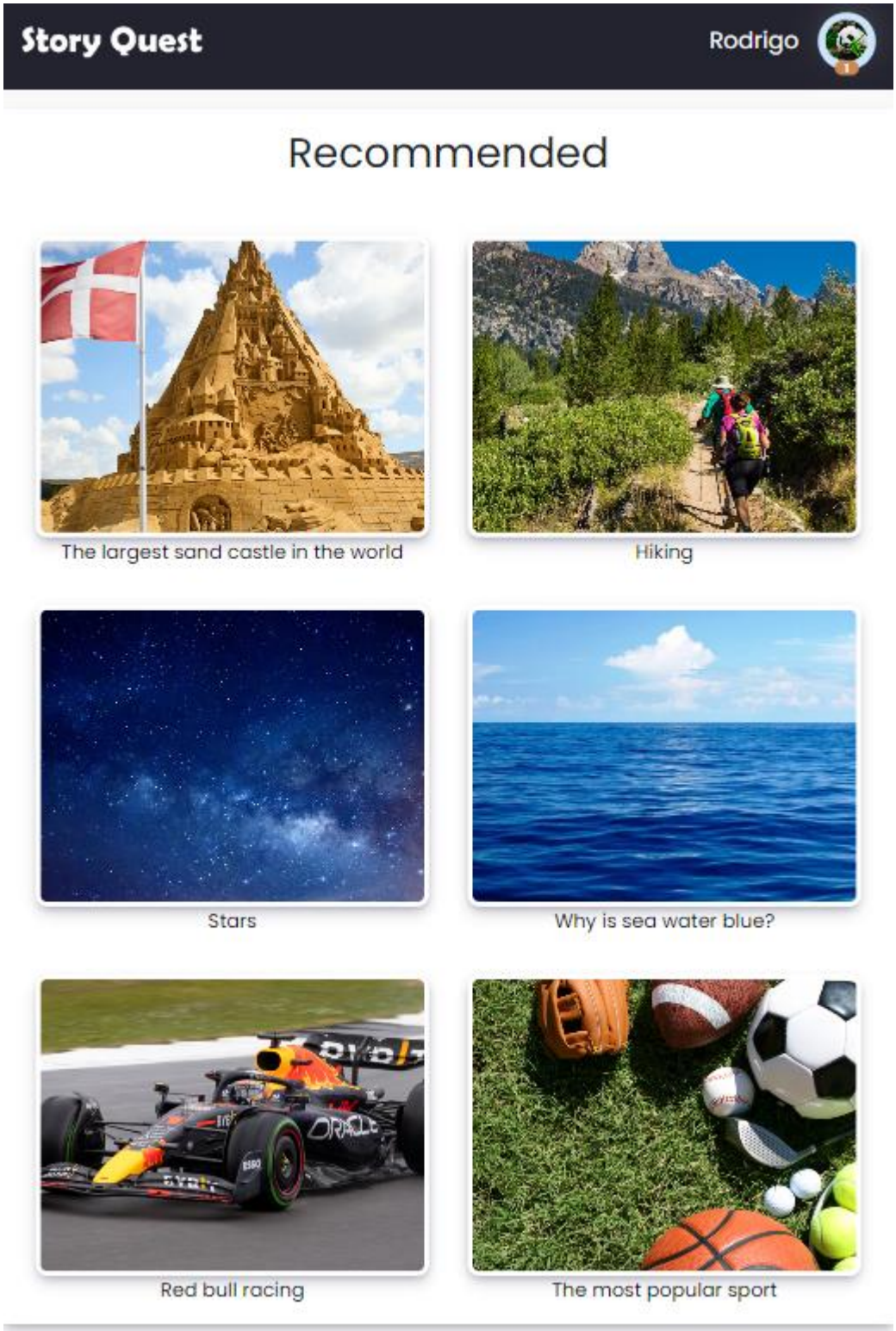

Figura 5.8.2.3. Recomendaciones para Rodrigo.

Figura 5.8.2.4. Recomendaciones para Daniel.

Se obtuvieron resultados satisfactorios, en las figuras 5.8.2.1 y 5.8.2.2 se observa que Rodrigo y Daniel tienen los mismos gustos para las últimas 4 historias, sin embargo, cada uno tiene 4 historias diferentes marcadas como favoritas.

Rodrigo tiene en su colección los títulos “Vacations”, “Moon rabbit”, “Soccer” y “Books”, mismos que Daniel no tiene.

Daniel tiene en su colección los títulos “Hiking”, “Stars” y “Red bull racing”, mismos que Rodrigo no tiene.

En las recomendaciones de Rodrigo podemos observar títulos como “Hiking”, “Stars”, “Red bull racing” y “A new telescope”.

En las recomendaciones de Daniel podemos observar títulos como “Vacations”, “Moon rabbit”, “Soccer” y “Books”.

Estos resultados demuestran que el filtro funciona correctamente, hay que recordar que el algoritmo sólo muestra la mitad de recomendaciones gracias a un usuario y la otra mitad aleatoriamente, esto con el fin de evitar recomendar siempre lo mismo al usuario.

5.9 Reportes

Para probar el funcionamiento en tiempo real se abrieron dos ventanas y se verificó que las notificaciones de reportes no leídos subieran, además de que se probó abrir la página de administración de reportes para verificar que estos se renderizaran automáticamente cuando un usuario los creaba.

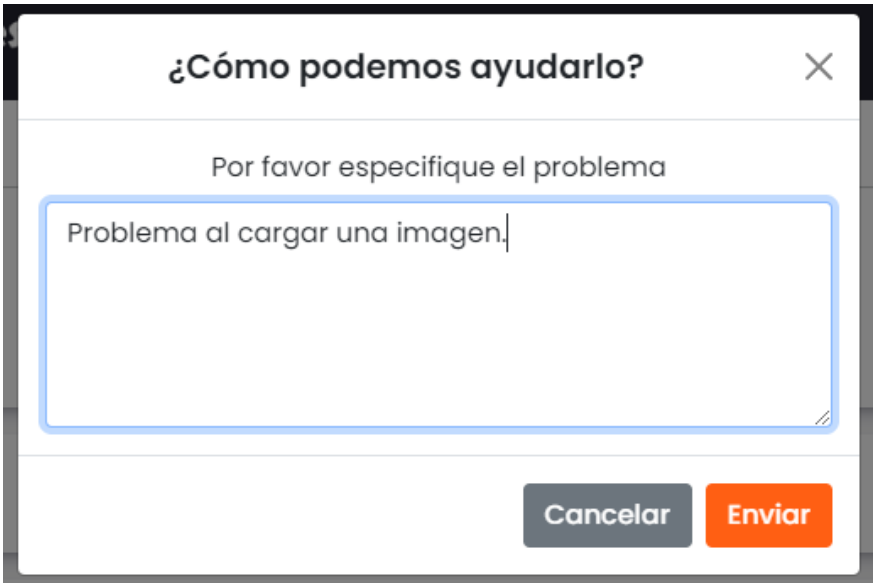

Figura 5.9.1. Prueba del formulario para crear un reporte.

Figura 5.9.2. Prueba de notificación de reportes subiendo.

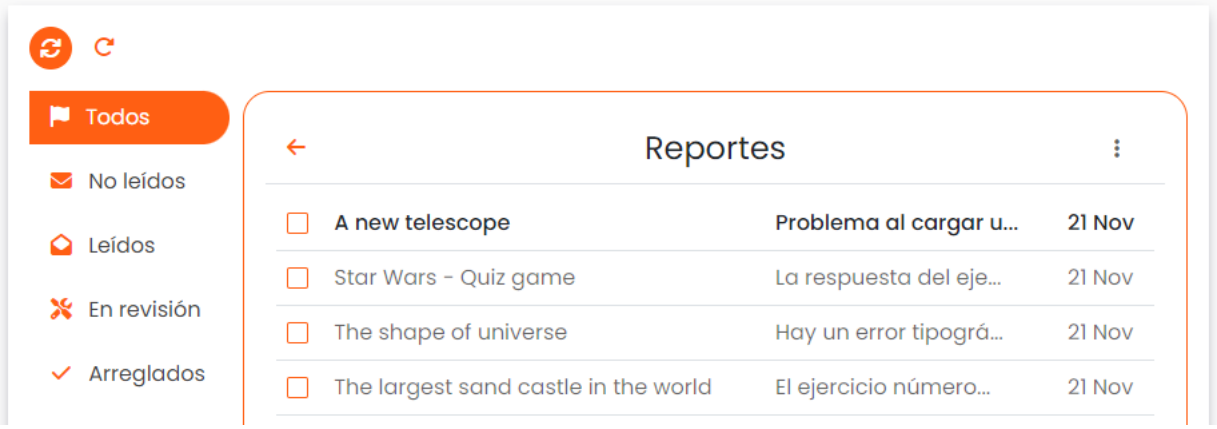

Figura 5.9.3. Nuevo reporte renderizado en la parte superior.

Figura 5.9.4. Reporte abierto.

Figura 5.9.5. Marcando reporte abierto para revisión.

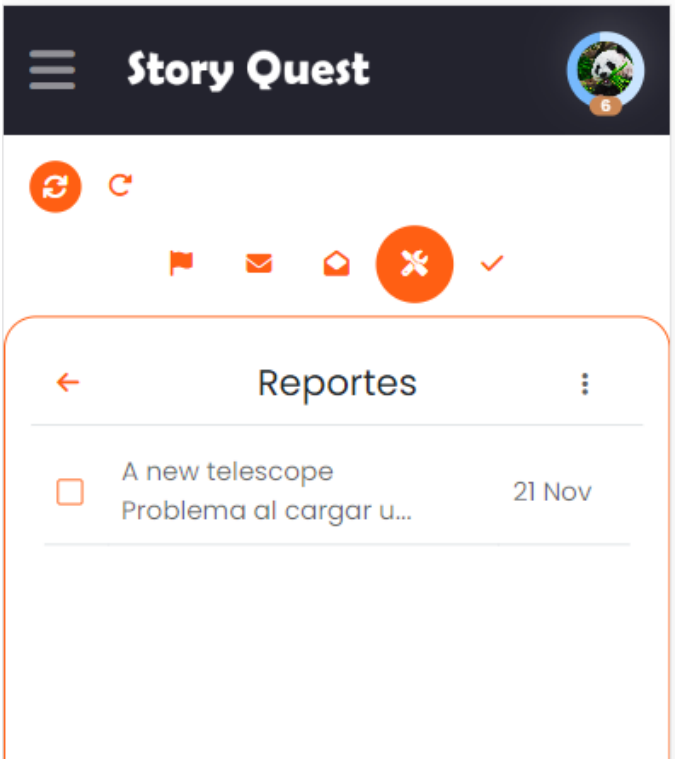

Figura 5.9.6. Reportes en revisión (móvil).

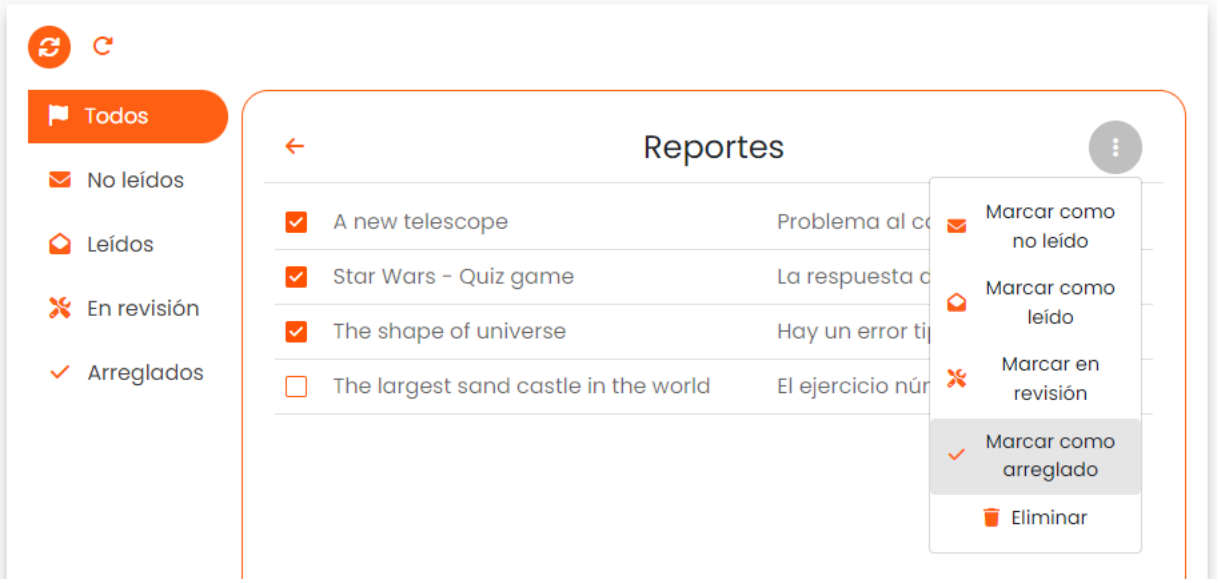

Figura 5.9.7. Marcar múltiples reportes como arreglados.

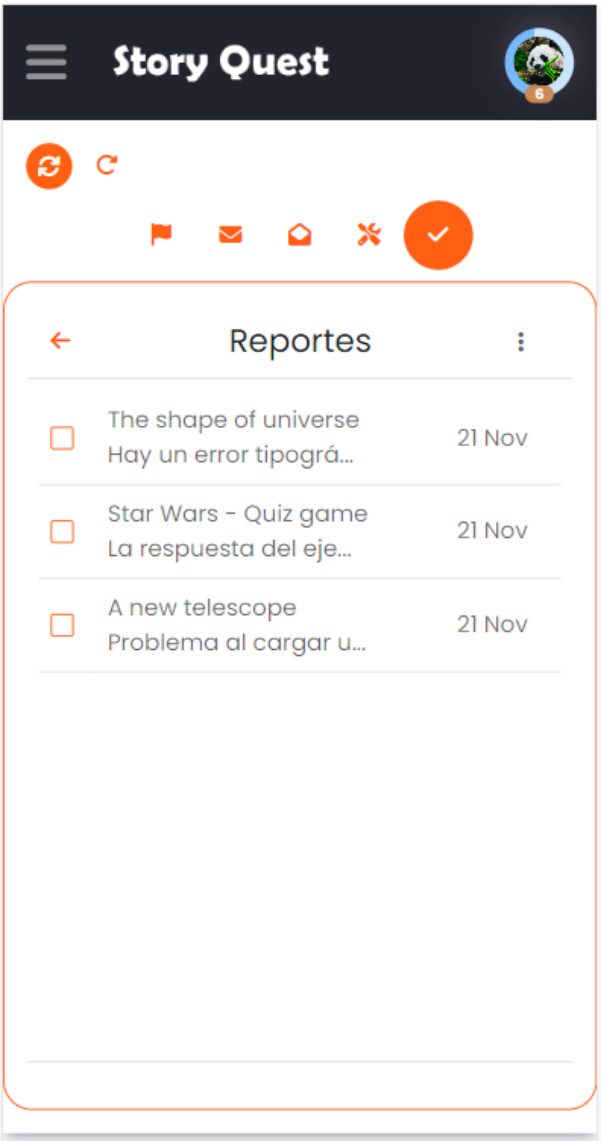

Figura 5.9.8. Reportes arreglados (móvil).

5.10 Pruebas de seguridad

Existen algunos lugares a los que el usuario no debería poder acceder sin cierta autoridad, este también es un punto de seguridad que debe ser evaluado.

5.10.1 Redireccionamiento

En el siguiente fragmento de código se observa la vista “index” encargada de renderizar la plantilla de las etiquetas, arriba de esta función encontramos dos decoradores “@login_required” y “@user_passes_test”, la primera redirecciona a

un usuario si no ha iniciado sesión y la segunda lo hace si el usuario no tiene el rol "staff", pues es la condición que se necesita para modificar etiquetas.

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

def is_staff(user):
    return user.is_staff


@login_required(login_url='/login/')
@user_passes_test(is_staff, login_url='/login/')
def index(request):
    if request.method == 'GET':
        tags = Tag.objects.all().order_by('name1')
        context = {
            'tags': tags
        }
        return render(request, 'admin/tags.html', context)
```

La función dentro de @user_passes_test puede cambiar, el siguiente fragmento de código pertenece a la vista encargada de mostrar los sistemas de recomendación y en lugar de "is_staff" se le pasa la función "is_superuser".

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

def is_superuser(user):
    return user.is_superuser


@login_required(login_url='/login/')
@user_passes_test(is_superuser, login_url='/login/')
def index(request):
    if request.method == "GET":
        ubr_settings = UBRSettings.objects.first()
        cbr_settings = CBRSettings.objects.first()

        context = {
            'ubr_settings': ubr_settings,
            'cbr_settings': cbr_settings
        }

        return render(request, 'admin/recommenders.html', context=context)
```

5.10.2 Almacenamiento de información

Información sensible como es el caso de la contraseña es encriptada y guardada en la base de datos, en la figura 5.10.2.1 se aprecia un query realizado para obtener información del usuario.

```
mysql> select id, username, password from auth_user;
+----+------------+--------------------------------------------------------------------------------------------------------------------------------+
| id | username   | password                                                                                                                       |
+----+------------+--------------------------------------------------------------------------------------------------------------------------------+
|  1 | Jesus      | pbkdf2_sha256$600000$JiHIjeiMzEO5jG4tBSdJiR$6jZbaM7gVg6ihX7y668T6hTONt6pGFNdYSLaYSz0IwI= |
|  2 | cuenta1    | pbkdf2_sha256$600000$Qaro6zoCYjpQseG1798F5u$YTLWEDyLqht4s2Ch3Jf6u9aMfDlEXfuXk4AN9hy+A7s= |
|  3 | DraMeliza  | pbkdf2_sha256$600000$HPOkawX4TY7HkyZ7oGLdep$arICJcBTddw8QtmiUj1FESF0t+ttX1rxRp/656rIVQM= |
|  4 | Test1      | pbkdf2_sha256$600000$H3JAfFi0OTShLhxFsM1khV$W7bTl+vVoEWbORrJ9uBwM7VkP0Iw5QFowUA2b+56+vQ= |
|  5 | Test2      | pbkdf2_sha256$600000$NUD0cS7XhrcmEANDUXgfPi$CeoswrSBv4lsB7jL8uVk/qnOUGhOOWCRqWmY/rw9ynk= |
|  6 | test-jesus | pbkdf2_sha256$600000$U1IpKP9o9BBYLRj0UAj4Pc$m2lmCgQGEeGtnE4s97RuZ9FiwccBYhmMDP+AJPHWgcM= |
```

Figura 5.10.2. Reportes arreglados (móvil).

5.10.3 Protección ante saturación de solicitudes

El siguiente fragmento de código realiza un conteo de las veces que un usuario da “me gusta” a una historia. Si el usuario llega a un límite su solicitud es bloqueada por 60 segundos, de esta manera se evita la posibilidad de que un usuario sature el servidor con cientos o miles de solicitudes.

```python
@login_required(login_url='/login/')
def likeStory(request, story_id):
    if request.method == 'POST':
        key = f'user_likes_{request.user.id}'
        rate_limit = 150
        rate_limit_time = 60

        num_requests = cache.get(key, 0)
        if num_requests > rate_limit:
            return JsonResponse({'error': 'Rate limit exceeded'}, status=429)
        cache.set(key, num_requests + 1, rate_limit_time)

        user_profile = request.user.profile
        story = Story.objects.get(id=story_id)

        if story in user_profile.liked_stories.all():
            user_profile.liked_stories.remove(story)
            story.likes_number -= 1
```

```
        else:
            user_profile.liked_stories.add(story)
            story.likes_number += 1

        user_profile.save()
        story.save()
        return JsonResponse({'success': True})
```

CONCLUSIONES Y TRABAJO FUTURO

Sin duda alguna el desarrollo web es un campo de estudio que implementa una gran variedad de funcionalidades que permiten realizar una aplicación altamente compleja y su principal característica es que no está ligada a un sistema operativo como Windows, Linux o Android, problema con el que cuentan bastantes aplicaciones dedicadas a la enseñanza de idiomas

A pesar de que en la actualidad existen muchas aplicaciones dedicadas a la enseñanza de inglés, ninguna de ellas toma tanto en cuenta la satisfacción del usuario estudiante y el usuario administrador a tal punto de implementar sistemas de notas, sistemas de recomendación o sistemas de tiempo real en una misma aplicación, por no mencionar la seguridad contra solicitudes múltiples o algunas otras características propias de la aplicación.

La aplicación web desarrollada durante esta investigación demostró la posibilidad de implementar diversas áreas de estudio como programación web, ciencia de datos o inclusive matemáticas con el fin de facilitar a los estudiantes del idioma una manera interactiva de seguir aprendiendo, brindándoles una experiencia educativa más completa y enriquecedora diseñada pensando en la comodidad de usuario aprendiz y usuario creador de contenido. A continuación, se presentan las contribuciones del presente estudio, atado a las limitaciones expuestas que dejan la oportunidad abierta para implementar trabajo futuro que complemente el mismo.

Contribuciones

Las principales contribuciones de este estudio se indican a continuación:

Algoritmo para traducir texto: Se diseñó la función de traducir texto como una alternativa a los métodos comúnmente utilizados. En este sentido el algoritmo para traducir fue fundamental para mantener una visualización atractiva para el usuario.

Algoritmo de similitud: El algoritmo de similitud proporciona una manera segura y baja en costo computacional para calificar al estudiante de una manera justa que no califica como errónea la respuesta del usuario si este falló en algo, en su lugar, ofrece una alternativa de calificación en donde el usuario es evaluado principalmente por sus logros y sus errores le bajan puntos de una manera gentil.

Diseño de páginas altamente flexibles: La forma modular en la que los elementos de contenido o ejercicios son creados dentro de una página permite un sinfín de patrones o plantillas posibles para que cada persona que desee crear sus historias lo haga de la manera que más le guste. El contenido audiovisual ayuda en gran medida a obtener una educación más enriquecedora que si sólo se contara con contenido de texto.

Libertad: El usuario es quien decide qué historia contestar y no necesita seguir una rigurosa lección sobre algún tema en específico, sino que simplemente puede encontrar un tema de su agrado y resolver los ejercicios que contiene.

Sistema de puntuaciones y retroalimentación de ejercicios: El sistema de puntuaciones abre la puerta para que el estudiante se esfuerce por obtener una calificación alta, la retroalimentación que se brinda es de suma importancia para desarrollar un progreso significativo.

Sistema de notas: Se resuelve la problemática común en aplicaciones de enseñanza de no contar con un sistema de apuntes, permitiendo mantener todo organizado dentro de la misma aplicación con un sistema de notas mediante tarjetas de estudio altamente intuitivas e interactivas.

Sistemas de recomendación: Los filtros de recomendación ayudan a que un usuario se sienta cómodo dentro de la aplicación, pues por un lado recomienda contenido de historias similares y por otro recomienda los gustos de personas que compartan sus intereses, además de agregar un factor aleatorio para que no siempre se le recomiende lo mismo y este pueda explorar nuevos horizontes.

Sistema de reportes en tiempo real: Los reportes en tiempo real garantizan que un miembro con permisos de staff reciba las últimas actualizaciones sobre las historias

que podrían contener fallas. Esta es una solución que evita el constante envío de solicitudes para lograr un efecto similar.

Limitaciones

La principal limitación de poner en marcha la aplicación a gran escala es contar con una máquina que actúe como un servidor web capaz de servir la aplicación, pues la aplicación requiere instalar una gran cantidad de servidores como MySQL, Nginx, Gunicorn, Daphne, Memcached y Redis.

Trabajo a futuro

Algunas propuestas de trabajo futuro que contribuirían al trabajo realizado en la presente tesis son:

Más elementos para las páginas: La creación o modificación de elementos dentro de una página, estos nuevos elementos pueden incluir ejercicios con límite de tiempo, ejercicios que permitan pistas o definiciones que el servidor proporcione a manera de ayuda.

Mejorar los sistemas de recomendación: Recopilar datos como la cantidad de preguntas promedio de una historia que le ha gustado al usuario, el tiempo que pasa contestando una página, el número de veces que visita la información de una historia, entre muchos otros datos para mejorar la experiencia de los filtros de recomendación.

Agregar una sección de comentarios en tiempo real: Implementar un sistema de comentarios en tiempo real dentro de la información de una historia y también dentro de una página de una historia para fomentar la comunicación entre los miembros de la aplicación.

Creación de grupos de publicaciones privadas: Actualmente todas las historias son creadas de manera general y están disponibles de manera pública para todo aquel que tenga una cuenta, una forma de mejorar esto sería la creación de grupos privados que generen una liga de acceso, nadie podrá ver el contenido de un grupo a menos que se una a este grupo mediante la liga de acceso. Esto puede resultar bastante útil

para profesores o instituciones que deseen crear un curso con temas específicos dentro de la aplicación.

BIBLIOGRAFÍA

1. Brandes, U. & Erlebach, T. (2005). Network Analysis: Methodological Foundations. Germany: Springer.
2. Coronel, C., & Morris, S. (s. f.). Database Systems: Design, Implementation, &
3. DiMarzio, J. (2008). Android: A Programmer's Guide. U.S.: Mc Graw Hill.
4. Sommerville, I. (2011). Ingeniería de Software. México: Pearson.
5. Šmite, D. (2010). Agility Across Time and Space: Implementing Agile Methods in Global Software Projects. Germany: Springer.
6. Management. Cengage Learning.
7. Silberschatz, A; Korth, H. & Sudarshan, S. (2002). Fundamentos de Bases de Datos. España: Mc Graw Hill.
8. Dooley, J. F. (2017). Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring. Apress.
9. Maeda, J. (2006). The laws of simplicity. The MIT Press.
10. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2002). Database Systems: The Complete Book. Upper Saddle River, NJ: Pearson Education, Inc.
11. Laplante, P. A., & Ovaska, S. J. (2007). Real-Time Systems Design and Analysis: Tools for the Practitioner. Piscataway, NJ: IEEE Press.
12. Hodent, C. (2021) The Psychology of Video Games. Milton Park, Oxon: Routledge.
13. Norman, D. A. (2002). The design of everyday things. Basic Books.

Webliografía

14. Manuel. (2022, November 22). Cuántas personas hablan inglés en México en la actualidad. AmazingTalker®. https://www.amazingtalker.com.mx/blog/mx-es/ingles/69888/
15. 10 Estadísticas del inglés en México: ¿Cuál es su nivel de inglés? | Papora. (n.d.). https://www.papora.com/es/estudios/estadisticas-ingles-mexico/

16. Redacción. (2021, February 3). Y a todo esto. . . ¿Cuál es el nivel de inglés de los mexicanos? El Financiero. https://www.elfinanciero.com.mx/tech/y-a-todo-esto-cual-es-el-nivel-de-ingles-de-los-mexicanos/
17. Englishman. (2021, December 15). Cake conversación en inglés. Aplicaciones Para Aprender Ingles. https://aplicacionesparaaprenderingles.com/cake/
18. Cake https://play.google.com/store/apps/details?id=me.mycake&hl=es_BO&pli=1
19. ¿Qué es Duolingo? (n.d.). Centro De Ayuda De Duolingo. https://support.duolingo.com/hc/es/articles/204829090--Qu%C3%A9-es-Duolingo-
20. Duolingo https://play.google.com/store/apps/details?id=com.duolingo&hl=es&gl=US
21. LearnEnglish sounds right. (2010, September 29). LearnEnglish. https://learnenglish.britishcouncil.org/apps/learnenglish-sounds-right
22. LearnEnglish Sounds Right https://play.google.com/store/apps/details?id=com.britishcouncil.phonemicchart&hl=es_MX&gl=US
23. Universidades, S. (2023, July 31). Metodologías de desarrollo software | Blog Becas Santander. Becas Santander. https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html
24. Rehkopf, D. M. (n.d.). Todo lo que necesitas saber sobre los sprints de scrum. Atlassian. https://www.atlassian.com/es/agile/scrum/sprints
25. Atlassian. (n.d.). ¿Qué es scrum? [+ Cómo empezar] | Atlassian. https://www.atlassian.com/es/agile/scrum
26. Rehkopf, D. M. (n.d.). Historias de usuario | Ejemplos y plantilla | Atlassian. Atlassian. https://www.atlassian.com/es/agile/project-management/user-stories
27. ¿Qué es una base de datos relacional? | IBM. (n.d.). https://www.ibm.com/mx-es/topics/relational-databases
28. ¿Qué es una base de datos? - Explicación de las bases de datos en la nube - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/database/

29. Helenclu. (n.d.). Definir relaciones entre tablas en una base de datos de Access - Microsoft 365 Apps. Microsoft Learn. https://learn.microsoft.com/es-es/office/troubleshoot/access/define-table-relationships
30. Palabras reservadas de SQL - Soporte técnico de Microsoft. (n.d.). https://support.microsoft.com/es-es/office/palabras-reservadas-de-sql-b899948b-0e1c-4b56-9622-a03f8f07cfc8
31. Martinekuan. (n.d.). Datos no relacionales y NoSQL - Azure Architecture Center. Microsoft Learn. https://learn.microsoft.com/es-es/azure/architecture/data-guide/big-data/non-relational-data
32. Tablado, F. (2020, September 10). Base de datos no relacional. ¿Qué es? Características y ejemplos. Ayuda Ley Protección Datos. https://ayudaleyprotecciondatos.es/bases-de-datos/no-relacional/
33. ¿Qué es una base de datos de clave-valor?. Amazon Web Services, Inc. https://aws.amazon.com/es/nosql/key-value/
34. ¿Qué es una base de datos de documentos?. Amazon Web Services, Inc. https://aws.amazon.com/es/nosql/document/
35. MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN. (2023, November 13). MDN Web Docs. https://developer.mozilla.org/es/docs/Glossary/MVC
36. Django overview. (n.d.). Django Project. https://www.djangoproject.com/start/overview/
37. Django. (n.d.). Django Project. https://docs.djangoproject.com/en/4.2/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names
38. Generalidades del protocolo HTTP - HTTP | MDN. (2023, September 22). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/HTTP/Overview
39. Collis, C. (2023, June 2). What are WebSockets and how do they work? - DiffusionData. DiffusionData. https://www.diffusiondata.com/what-are-web-sockets-and-how-do-they-work/

40. RFC 6455: the WebSocket Protocol. (2011, December 11). IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc6455
41. IBM documentation. (2023, January 26). https://www.ibm.com/docs/es/was/9.0.5?topic=applications-websocket
42. Equipo editorial de IONOS. (2020, July 24). El diagrama de casos de uso en UML. IONOS Digital Guide. https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/
43. GraphEverywhere, E. (2019, December 2). Sistemas de recomendación | Qué son, tipos y ejemplos. GraphEverywhere. https://www.grapheverywhere.com/sistemas-de-recomendacion-que-son-tipos-y-ejemplos/
44. Dia-Mfaia. (2023, July 17). Sistemas de recomendación: personalizando la experiencia del usuario en la era digital. MFAIA. https://mfaia.ws.fi.upm.es/sistemas-de-recomendacion-personalizando-la-experiencia-del-usuario-en-la-era-digital/
45. What are Flashcards? - Answered - Twinkl Teaching Wiki. (n.d.). Twinkl. https://www.twinkl.com.mx/teaching-wiki/flashcards
46. Using flash cards with young learners. (n.d.). TeachingEnglish. https://www.teachingenglish.org.uk/professional-development/teachers/managing-resources/articles/using-flash-cards-young-learners
47. Bootstrap 5 https://getbootstrap.com/docs/5.3/getting-started/introduction/
48. Htmx https://htmx.org/
49. Redis https://redis.io/
50. Django Channels — Channels 4.0.0 documentation. (n.d.). https://channels.readthedocs.io/en/latest/
51. Escribiendo aplicaciones con WebSockets - Referencia de la API Web | MDN. (2023, July 24). MDN Web Docs.

https://developer.mozilla.org/es/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

52. Serrano.Academy (2018). How does Netflix recommend movies? Matrix Factorization. Recuperado de https://www.youtube.com/watch?v=ZspR5PZemcs
53. Grab N Go Info (2022). User-Based Collaborative Filtering In Python | Machine Learning. Recuperado de https://youtu.be/cxcFi3RDrEw?si=-gk-SOAh3VxBu5Y5
54. Eugenia Inzaugarat (2020). Music recommender system. Recuperado de https://github.com/ugis22/music_recommender