UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Generación automática relatos interactivos con Streamink para Twitch

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Hernandez Jacinto, Alejandro

Tutor/a: Rebollo Pedruelo, Miguel

CURSO ACADÉMICO: 2022/2023

Resumen

Las historias interactivas son una forma popular de entretenimiento que ofrece inmersión al permitir a los lectores tomar decisiones. En tiempos recientes, con la aparición de las narraciones interactivas digitales han surgido plataformas como StreamINK. Estas buscan acercar estas historias a un público más amplio mediante la retransmisión en directo, permitiendo una interacción con la audiencia.

En este Trabajo de Fin de Grado se describe el desarrollo de una aplicación web destinada a la creación, edición y gestión de historias interactivas diseñadas para ser jugadas en StreamINK, enfatizando en ofrecer simplicidad y usabilidad a los usuarios. El programa, desarrollado con Blazor WebAssembly, integra conceptos de la teoría de grafos para ofrecer una perspectiva distinta del relato.

Este documento detalla todas las fases por las que ha pasado el proyecto durante su desarrollo, desde el análisis hasta las pruebas, destacando las dificultades en el proceso y las soluciones adoptadas.

Palabras clave: aplicación web, teoría de grafos, historias interactivas, análisis de redes, Blazor WebAssembly

Abstract

Interactive stories are a popular form of entertainment that offers immersion by allowing readers to make choices. With the emergence of digital interactive storytelling in recent times, platforms such as StreamINK have emerged. These seek to bring these stories to a wider audience through live streaming, allowing interaction with the audience.

This Final Degree Project describes the development of a web application for the creation, editing and management of interactive stories designed to be played on StreamINK, emphasizing simplicity and usability for users. The program, developed with Blazor WebAssembly, integrates concepts from graph theory to offer a different perspective on storytelling.

This document details all the phases the project has gone through during its development, from analysis to testing, highlighting the difficulties in the process and the solutions adopted.

Keywords: web application, graph theory, interactive stories, network analysis, Blazor WebAssembly

Resum

Les històries interactives són una forma popular d'entreteniment que ofereix immersió en permetre als lectors prendre decisions. Amb l'aparició de les narracions interactives digitals en temps recents, han sorgit plataformes com ara StreamINK. Aquestes busquen apropar aquestes històries a un públic més ampli mitjançant la retransmissió en directe i permeten una interacció amb l'audiència.

En aquest Treball de Fi de Grau es descriu el desenvolupament d'una aplicació web destinada a la creació, edició i gestió d'històries interactives dissenyades per ser jugades a StreamINK, emfatitzant oferir simplicitat i usabilitat als usuaris. El programa,

desenvolupat amb Blazor WebAssembly, integra conceptes de la teoria de grafs per oferir una perspectiva diferent del relat.

Aquest document detalla totes les fases per les quals ha passat el projecte durant el desenvolupament, des de l'anàlisi fins a les proves, destacant les dificultats en el procés i les solucions adoptades.

Paraules clau: aplicació web, teoria de grafs, històries interactives, análisis de xarxes, Blazor WebAssembly

Tabla de contenidos

Índice de ilustraciones

1. Introducción

Desde la aparición de los libros de *Elige tu propia aventura* a finales de los años 70 hasta la actualidad, las historias interactivas han sido una forma popular de entretenimiento que permite a los lectores sumergirse en la narrativa y tomar decisiones que afectan a su desenlace. Con la evolución de la tecnología y la creciente popularidad de plataformas de *streaming* como Twitch, las historias interactivas han encontrado nuevas formas de llegar a un público más amplio.

Aplicaciones como StreamINK permiten jugar a este tipo de historias añadiendo la participación de la comunidad, aportando así un factor social. Sin embargo, la creación de estas narraciones puede ser un proceso complejo y laborioso, especialmente para aquellos que no están familiarizados con lenguajes de *scripting* cómo Ink.

Este trabajo de fin de grado tiene como objetivo desarrollar una herramienta fácil de usar e intuitiva que permita a los usuarios crear historias interactivas de forma sencilla, utilizando grafos para representar las distintas rutas posibles en la narrativa.

1.1 Motivación

Mi preferencia por este proyecto tiene sus orígenes en mi infancia, cuando comencé a leer libros que permitían al lector elegir el rumbo de la historia. Esta experiencia única de ser el protagonista y tomar decisiones que influenciaban el desenlace me cautivó. Con el tiempo, esta fascinación evolucionó hacia videojuegos que, aunque variados en su enfoque, compartían un núcleo común: la inmersión y la personalización de la experiencia narrativa.

En la actualidad, con la popularidad creciente de las plataformas de contenido como Twitch, tuve la oportunidad de conocer StreamINK. Este sistema moderniza el concepto de los libros interactivos, adaptándolos a la era digital. No obstante, identifiqué un obstáculo significativo: la complejidad involucrada en la creación de historias. Frente a este desafío, el propósito de este proyecto es desarrollar una herramienta que simplifique y haga más intuitivo el proceso de creación de historias interactivas.

1.2 Objetivos

El principal objetivo de este proyecto es diseñar y desarrollar un sistema que facilite a los usuarios la elaboración de historias interactivas para su posterior uso dentro del entorno StreamINK.

Como objetivos específicos se pueden enumerar:

O1 - Facilitar la creación, edición y gestión de historias y sus respectivos capítulos.

O2 - Presentar a los usuarios una representación gráfica de sus historias a través de una visualización en forma de grafo.

O3 - Asegurar una exportación compatible de las historias para su correcta ejecución en StreamINK.

1.3 Impacto esperado

Se espera generar un impacto en la comunidad de jugadores y lectores de estas historias de forma indirecta, con la aparición de un mayor número de historias disponibles gracias a la herramienta de creación.

Por otro lado, se identifica como usuario del programa al escritor de historias interactivas. En su caso podrá redactar historias con facilidad, centrándose en escribir y sin tener que preocuparse de perderse en sus múltiples caminos gracias a la estructura de grafo.

1.4 Metodología

En el desarrollo de este proyecto, se ha optado por la metodología en cascada como marco estructural para guiar cada fase del proceso. Esta metodología tradicional del desarrollo de software se distingue por su enfoque secuencial, en el que cada fase es una continuación de la anterior, funcionando de manera descendente.

La elección de esta metodología se basa en la estructuración clara y sistemática que ofrece, donde cada etapa tiene definiciones y objetivos específicos que permiten un avance coherente y metódico hacia la culminación del software. Estas serán las fases que se realizarán:

1. **Análisis de Requisitos:** En este paso inicial, el enfoque principal será la especificación detallada de requisitos, tanto funcionales como no funcionales. Además, se procederá a la identificación de los casos de uso que describen las interacciones de los usuarios con el sistema. Paralelamente, se explorarán posibles soluciones, culminando con una propuesta clara sobre cómo se abordarán los desafíos identificados.

2. **Diseño del Sistema:** Una vez establecidos y definidos los requisitos, el siguiente paso será abordar el diseño del sistema. En esta etapa, se planificará la arquitectura general del software, apoyada por diagramas de clase que definirán la estructura y las interacciones de las clases. Además, se definirá la estructura de los ficheros JSON que se utilizarán, y se comenzarán a diseñar prototipos para la interfaz de usuario, asegurando que cumplan con los requisitos y proporcionen una experiencia de usuario óptima.

3. **Codificación:** Una vez completado el diseño, la atención se centrará en la transformación de las especificaciones y prototipos en código ejecutable. Aquí, las decisiones tomadas en las fases anteriores se concretarán, y se llevará a cabo la implementación del software conforme a lo planeado.

4. **Pruebas:** Finalmente, antes de que el software esté listo para su entrega o despliegue, se realizarán pruebas unitarias. Estas pruebas tienen como objetivo validar que cada componente del software funcione como se espera, detectando posibles fallos o áreas de mejora.

Aunque la metodología en cascada presenta desafíos en términos de flexibilidad, su estructura claramente definida y su enfoque paso a paso hacen que sea una excelente opción para proyectos donde los requisitos están bien definidos desde el comienzo, como es el caso de este proyecto.

1.5 Estructura de la memoria

Este apartado tiene como objetivo ofrecer una visión general y estructurada de los contenidos que se tratan en cada uno de los capítulos de esta memoria.

1. **Introducción**: Este capítulo establece el contexto y la motivación del proyecto, introduciendo al lector en la temática principal y ofreciendo un vistazo general del trabajo realizado.

2. **Estado del arte**: Aquí se revisa la literatura y tecnologías existentes en el ámbito de las historias interactivas. Se realiza un análisis de los trabajos previos, herramientas y técnicas, estableciendo una base teórica para el proyecto.

3. **Análisis del problema**: Este capítulo se centra en el estudio detallado del problema a resolver, identificando sus requisitos, casos de uso y las necesidades específicas que debe cubrir la solución propuesta.

4. **Diseño de la solución**: Basándose en el análisis previo, se revisan las decisiones que se han tomado en el proceso de diseño de la solución. Se define la arquitectura del sistema, los componentes principales y cómo interactúan entre sí, así como las tecnologías utilizadas.

5. **Desarrollo**: En este capítulo se detalla el proceso de implementación del sistema. Se describen los retos enfrentados durante la codificación y cómo se superaron.

6. **Pruebas**: Se dedica a la evaluación del sistema desarrollado. Se expone la metodología de pruebas empleada, los resultados obtenidos y cómo estos reflejan la eficacia y eficiencia de la solución propuesta.

7. **Conclusiones**: Aquí se recapitulan los principales hallazgos y resultados del proyecto. Se reflexiona sobre las implicaciones del trabajo realizado, las lecciones aprendidas y qué han aportado las asignaturas del grado.

2. Estado del arte

Las narraciones interactivas, en sus diversas formas y adaptaciones, han jugado un papel fundamental en el entretenimiento humano, la educación y el arte durante el último siglo. Este tipo de narraciones, donde el usuario o lector puede influir en el desenlace de la historia mediante sus decisiones y acciones, ha experimentado un constante proceso de evolución, diversificándose y adaptándose a los avances tecnológicos y a las demandas cambiantes del público. Las narraciones interactivas han tocado numerosos dominios, desde literatura hasta videojuegos, enriqueciendo el paisaje de la creatividad y el diseño narrativo con multitud de innovaciones y propuestas.

El primer libro interactivo marcó el comienzo de esta trayectoria evolutiva, proporcionando a los lectores una forma completamente nueva de interactuar con la historia y sus personajes. Este concepto se desarrolló y refinó en los libros *Elije tu propia aventura*, que permitieron una mayor elección y personificación de la experiencia de lectura.

El advenimiento de los juegos de rol permitió aún más interactividad, dando a los jugadores la libertad de desempeñar roles y moldear el mundo a través de sus acciones y decisiones. Con el lanzamiento de *Dragones y mazmorras*, la narración interactiva alcanzó un nuevo nivel de profundidad y complejidad.

Mientras tanto, la revolución digital estaba en marcha, y los primeros juegos conversacionales llevaron la narración interactiva al mundo digital, proporcionando una experiencia de juego y narración completamente nuevas. Esto evolucionó hacia la creación de las primeras aventuras gráficas, como *Monkey Island*, que combinaban gráficos con diálogos y decisiones basadas en texto para proporcionar experiencias inmersivas y narrativas.

La industria del videojuego continuó innovando, concretamente los videojuegos de rol aparecieron para tratar de llevar al mundo digital los exitosos juegos de rol. La aparición de obras como *World of Warcraft* llevaron las historias interactivas a un nivel masivo, permitiendo a millones de jugadores de todo el mundo participar en historias épicas y aventuras.

Recientemente, las plataformas de *streaming* como Netflix han comenzado a experimentar con el enfoque interactivo, creando algunas películas o series con estas características. El exitoso episodio interactivo *Black Mirror: Bandersnatch* permitió a los espectadores tomar decisiones que influían en el curso de la historia.

El apartado 2.1, profundizará en los productos que se han mencionado, desglosando en detalle sus características y su impacto en el campo de las narraciones interactivas. Posteriormente, en la sección 2.2, nos centraremos en una enumeración de las diversas herramientas disponibles para la creación de estas formas de narrativa. En el apartado 2.3, cambiaremos el enfoque hacia las soluciones específicas existentes para StreamINK, poniendo en relieve las actuales prácticas en esta área. Finalmente, en el apartado 2.4, presentaremos nuestra propuesta única que busca aportar una perspectiva novedosa y mejorada al ámbito de las narraciones interactivas.

2.1 Productos

2.1.1 Librojuegos o libros de elige tu propia aventura

Los librojuegos o libros interactivos son una forma especializada de literatura que permite a los lectores participar activamente en la historia, decidiendo el curso de la trama y el destino de los personajes (1). En estos libros, la narrativa se bifurca en varios

puntos, lo que ofrece múltiples rutas de lectura (ver ilustración 1) y múltiples desenlaces. Los lectores deben tomar decisiones para los personajes y, dependiendo de sus elecciones, son dirigidos a diferentes secciones del libro para continuar la historia.

Esta forma de literatura interactiva comenzó a tomar forma en los primeros años del siglo XX. Un ejemplo pionero es la novela romántica *¡Considera las consecuencias!* de Doris Webster y Alden Hopkins, publicada en 1930, que posee más de una docena de posibles finales.

Sin embargo, la popularización de los librojuegos no se daría hasta los años 70 con la aparición de la primera serie titulada *Tracker Books*, publicada en Reino Unido a partir de 1972. Mientras tanto, en Estados Unidos surgía la serie *The Adventures of you* en 1976, que posteriormente se renombraría como *Elige tu propia aventura* (*Choose your own adventure*). Esta última serie alcanzó tal popularidad que los librojuegos a menudo se conocen como libros de Elige tu propia aventura.

Cabe destacar que, con el auge de los juegos de rol, algunos librojuegos comenzaron a implementar normas propias de estos, como lanzar dados para determinar el resultado de ciertas acciones. Un ejemplo de estos libros es *Buffalo Castle*, escrito por Rick Loomis en 1976.

A partir de la entrada al siglo XXI, los librojuegos han sufrido un declive en su popularidad, debido en parte a la transición de la mayoría del público hacia los videojuegos. Sin embargo, en los últimos años han surgido alternativas digitales, como StreamINK, que buscan revitalizar este formato literario en el ámbito digital.

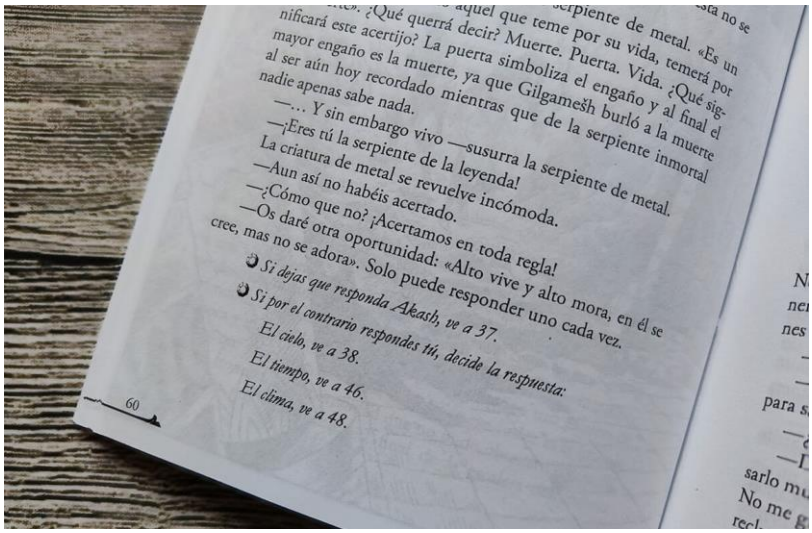

Ilustración 1. Librojuego

2.1.2 Juegos de rol

Los juegos de rol son una forma de narración colaborativa e improvisada regida por un conjunto de reglas (2). Usualmente los participantes asumen los roles de personajes e interactúan dentro de un mundo ficticio gobernado por un moderador, a menudo referido como el director de juego (*dungeon master*). Estos juegos se caracterizan por su énfasis en la colaboración, la narración y la inmersión en el personaje, ofreciendo a los jugadores una experiencia de juego profundamente personalizada y social. Los jugadores toman decisiones basadas en sus personajes, lo que influye en el desarrollo de la trama y en las interacciones con el mundo del juego y los demás personajes.

El origen de los juegos de rol modernos se puede rastrear hasta mediados del siglo XX, cuando los juegos de mesa de guerra (*wargames*) comenzaron a incorporar

elementos de caracterización y narrativa (3). Sin embargo, no fue hasta la publicación de *Dungeons & Dragons* (*D&D*), en España *Dragones y mazmorras*, en 1974 que el género de los juegos de rol se estableció y popularizó. Creado por Gary Gygax y Dave Arneson, *D&D* presentaba un mundo de fantasía medieval inspirado en la literatura de J.R.R. Tolkien, y un sistema de juego que hacía hincapié en la interpretación de roles y la resolución de conflictos a través de la narración y los dados.

Dragones y mazmorras tuvo un impacto enorme y duradero en el desarrollo de los juegos de rol. Las reglas y convenciones establecidas por *D&D*, desde el uso de dados poliédricos hasta la estructura de los personajes basada en rasgos y habilidades, se han convertido en estándares del género. *D&D* también popularizó el concepto de campañas, o series de sesiones de juego que se desarrollan en una narrativa continua, permitiendo a los personajes crecer y evolucionar con el tiempo.

A lo largo de los años, los juegos de rol han evolucionado para abarcar una amplia gama de estilos, géneros y sistemas de juego. Además de las tradicionales ambientaciones de fantasía, hay juegos que exploran la ciencia ficción, el terror (ver ilustración 2), la historia alternativa e incluso mundos basados en obras de literatura y cine.

Con la llegada del siglo XXI y el advenimiento de la tecnología digital, los juegos de rol también han visto su popularidad reducida a un nicho. Los videojuegos de rol que luego mencionaremos son sus herederos directos.

Ilustración 2. Juego de mesa Dragones & Mazmorras Castle Ravenloft

2.1.3 Juegos conversacionales y aventuras gráficas

Los juegos conversacionales y las aventuras gráficas son videojuegos centrados en la narrativa y la resolución de problemas en lugar del combate o la competencia directa. En su forma más simple, los juegos conversacionales, también conocidos como aventuras de texto, utilizan solo texto para describir el entorno y la interacción, mientras que las aventuras gráficas añaden representaciones visuales y, a menudo, interfaces de *Point and Click* (4).

Las aventuras de texto son uno de los géneros más antiguos de los videojuegos, apareciendo por primera vez a finales de los años 70. Su origen se remonta a *Colossal Cave Adventure* o simplemente *Adventure*, desarrollado por Will Crowther en 1976 y ampliado por Don Woods en 1977. Este juego, basado completamente en texto, permitía a los jugadores explorar un mundo ficticio resolviendo acertijos para progresar. Las

acciones y las interacciones del juego se realizaban a través de comandos de texto, estableciendo el patrón que muchos juegos conversacionales seguirían.

Las aventuras gráficas surgen como una evolución natural de las aventuras de texto, agregando elementos visuales para complementar y mejorar la experiencia del jugador. Los primeros ejemplos de este género fueron *Mystery House* de 1980, desarrollado por On-Line Systems (más tarde conocida como Sierra Entertainment), que añadió imágenes estáticas para ilustrar las diferentes escenas descritas en el texto.

Con el avance de la tecnología y el aumento de la capacidad gráfica, las aventuras gráficas experimentaron una evolución significativa. Los videojuegos de Lucasfilm Games (más tarde LucasArts) como *Maniac Mansion* (1987) y *The Secret of Monkey Island* (1990) (ver ilustración 3) son a menudo citados como el apogeo del género gracias a su narrativa fuerte y humorística, su sistema de comandos innovador y su enfoque en los rompecabezas y la interacción con el entorno en lugar de la acción.

Los juegos conversacionales y las aventuras gráficas han tenido un impacto duradero en la industria del videojuego inspirando una variedad de géneros y subgéneros, desde las novelas visuales hasta los juegos de exploración. A pesar de la disminución de su popularidad en los años 90 frente a los juegos de acción y los juegos en 3D, estos géneros han experimentado un renacimiento en el siglo XXI con la popularidad de los juegos independientes con títulos como *Disco Elysium*, que evoluciona el género fusionando con los videojuegos de rol.

Ilustración 3. The Secret of Monkey Island

2.1.4 Videojuegos de rol

Los videojuegos de rol (RPG) son uno de los géneros más populares de los videojuegos. Inspirados y evolucionados a partir de los juegos de rol de mesa, los RPG traducen conceptos como el desarrollo de personajes, la progresión por niveles y la narrativa en experiencias digitales inmersivas (5).

Los primeros videojuegos de rol comenzaron a aparecer a mediados de la década de 1970, siguiendo los conceptos de juegos de rol de mesa como *Dragones y Mazmorras* (6). Un ejemplo temprano es *Dungeon*, creado en 1975 para la plataforma PLATO. Este juego ya contenía características básicas de los RPG, como mazmorras para explorar, monstruos para combatir y tesoros para recolectar.

Una influencia crucial en el desarrollo de los RPG fue el juego *Wizardry: Proving Grounds of the Mad Overlord*, lanzado en 1981, que introdujo características esenciales como el uso de una perspectiva en primera persona para la exploración de mazmorras

y la creación y gestión de un grupo de personajes. Otro hito fue *Ultima* de Richard Garriott, que comenzó en 1981 y tuvo varias secuelas, notablemente evolucionando en términos de gráficos y narrativa.

En la década de 1990, con el advenimiento de las consolas de juegos y la popularidad creciente de los videojuegos, los RPG experimentaron una explosión de popularidad y evolución. Juegos como *Final Fantasy*, *Dragon Quest* y *The Elder Scrolls* definieron aún más el género, cada uno aportando sus propias innovaciones.

No obstante, la cumbre de los RPG, especialmente en el formato de los juegos de multijugador online masivos (*Massively Multiplayer Online Role-Playing Games* — MMORPG—), se produjo con *World of Warcraft* (*WoW*) de Blizzard Entertainment. Lanzado en 2004, *WoW* (ver ilustración 4) se ha convertido en uno de los videojuegos más exitosos y reconocibles de todos los tiempos, con millones de jugadores en todo el mundo. *WoW* ha tenido un impacto profundo en la cultura popular y ha establecido el estándar para futuros MMORPG con su mundo persistente, su sistema de progresión de personajes y sus actividades de grupo como mazmorras y asaltos.

Los videojuegos de rol han dejado una huella indeleble en la industria del videojuego, evolucionando desde sus modestos comienzos hasta convertirse en experiencias multiplataforma y multijugador masivas. Continúan siendo un género importante y en constante evolución, al combinar narrativas profundas, sistemas de juego complejos y una inmersión sin precedentes en mundos de fantasía y más allá.

Ilustración 4. World of Warcraft

2.1.5 Películas y series interactivas

Las películas y series interactivas pertenecen a una rama de la narración digital, permitiendo a los espectadores participar activamente en el argumento al tomar decisiones que afectan a la historia. Estos trabajos interactivos están diseñados para ofrecer experiencias de visualización personalizadas y ramificadas, a diferencia de la linealidad fija de las películas y series tradicionales.

Los primeros intentos de crear contenido audiovisual interactivo datan de los años 60, con la creación de instalaciones de arte que permitían a los espectadores intervenir en la narrativa (7). Pero fue en los años 90, con la aparición de los DVD, cuando este concepto empezó a adoptar una forma más reconocible. Los DVD interactivos, aunque rudimentarios, permitían a los espectadores tomar decisiones simples que influían en el desarrollo de la historia.

En los últimos años, con el auge de las plataformas de *streaming* (video a la carta), la narrativa ha evolucionado de manera considerable. Un punto de inflexión importante fue el lanzamiento de *Black Mirror: Bandersnatch* por Netflix en 2018 (ilustración 5). Esta película, ambientada en la popular serie de ciencia ficción *Black Mirror*, invitaba a los espectadores a tomar decisiones por el protagonista, con cada elección ramificando la narrativa en diferentes direcciones y desenlaces. Este modelo fue seguido por otros títulos de Netflix, como *You vs. Wild*, una serie interactiva de aventuras que permitía a los espectadores tomar decisiones en situaciones de supervivencia.

A pesar del crecimiento del contenido mencionado sigue siendo un medio de relativa baja popularidad en comparación a otros medios como los videojuegos.

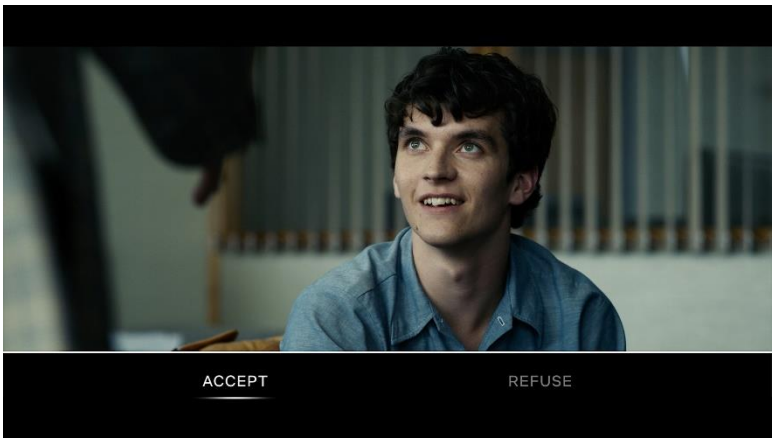

Ilustración 5. Black Mirror: Bandersnatch

2.2 Herramientas

2.2.1 Creación de aventuras gráficas

Las herramientas de creación de aventuras gráficas han evolucionado con el tiempo, permitiendo a los desarrolladores llevar a cabo sus visiones creativas con mayor eficiencia y sofisticación.

Adventure Game Studio (AGS) se ha establecido como un recurso potente en la creación de aventuras gráficas. Como herramienta de software libre, AGS favorece la realización de juegos en el estilo clásico de los años 90 (8), pero su flexibilidad le permite dar vida a juegos contemporáneos igualmente robustos, como la aventura de terror *The Cat Lady*, publicada en 2012. Con su interfaz intuitiva (ver ilustración 6), los desarrolladores pueden crear personajes, escenarios, objetos y diálogos con relativa facilidad.

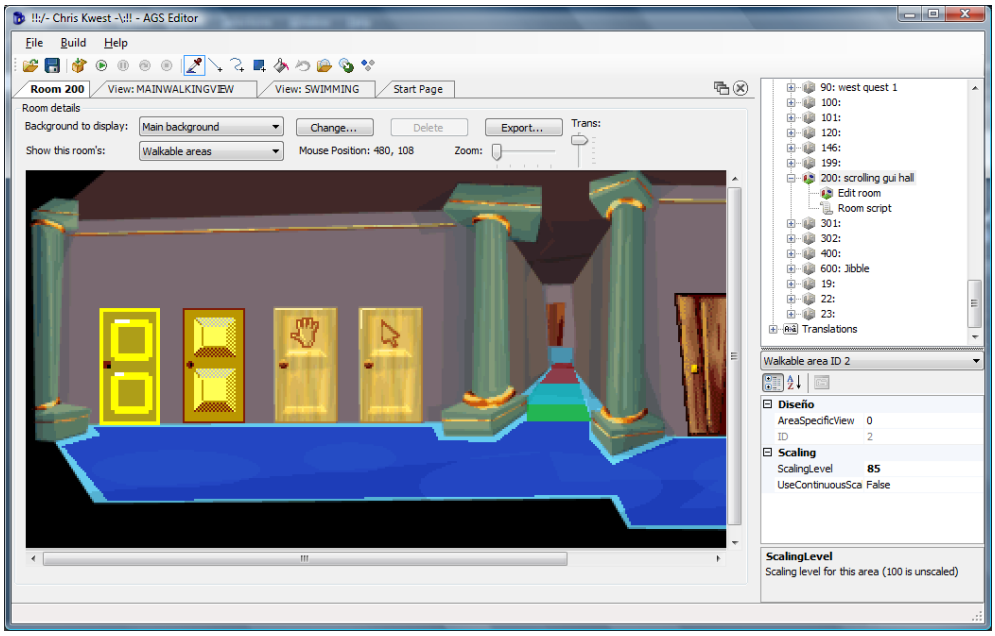

Ilustración 6. Interfaz de Adventure Game Studio

Visionaire Studio es otro recurso relevante en este campo, especialmente reconocido por su enfoque en las aventuras gráficas del tipo *Point and Click* (9). Este software multipropósito se destaca por su funcionalidad de animación, manejo de escenas y gestión de diálogos, habiendo respaldado la creación de numerosos juegos *indie* y comerciales que han recibido elogios de la crítica como por ejemplo *Deponia*.

El motor de videojuegos Unity, del que hablaremos luego en la creación de videojuegos de rol, a pesar de no centrarse en aventuras gráficas, posee características que permiten a los desarrolladores abordar cualquier tipo de proyecto de juego, incluidas las aventuras gráficas.

2.2.2 Creación de videojuegos de rol

El desarrollo de videojuegos de rol ha experimentado un avance significativo a lo largo de las décadas y la aparición de diversas herramientas de creación ha jugado un papel crucial en este proceso. Estos recursos de desarrollo de software han facilitado la producción de juegos de rol y brindado a los desarrolladores una mayor libertad creativa.

Unity (ilustración 7) es uno de los motores de juego más utilizados y reconocidos en la industria del desarrollo de videojuegos (10). Ofrece una amplia gama de funcionalidades que incluyen renderizado en 3D, soporte de físicas, animación, y un poderoso sistema de *scripting*. Unity es especialmente apreciado por su versatilidad, permitiendo el desarrollo tanto de pequeños proyectos *indie* como de grandes títulos AAA (de gran presupuesto). Juegos de rol tan populares como *Pillars of Eternity* han sido creados con Unity.

Ilustración 7. Interfaz de Unity

Por otro lado, Unreal Engine, desarrollado por Epic Games, es otro motor de juegos ampliamente utilizado en la industria (11). Al igual que Unity, Unreal ofrece una gran variedad de características, incluyendo un avanzado sistema de iluminación y renderizado, soporte para la creación de mundos abiertos, y un potente sistema de física de partículas. Juegos de rol de renombre como *Mass Effect* y *Final Fantasy VII Remake* han sido desarrollados utilizando Unreal Engine.

Otra herramienta es GameMaker (12), una plataforma muy conocida por su simplicidad y accesibilidad, ideal para desarrolladores principiantes o aquellos que buscan un desarrollo más rápido y eficiente. Aunque está más orientada a juegos en 2D, GameMaker ha sido utilizado para crear RPG exitosos como *Undertale*.

En conclusión, estas herramientas de creación han evolucionado para ser más intuitivas y poderosas, abriendo un abanico de posibilidades para los desarrolladores y permitiendo la creación de videojuegos de rol cada vez más inmersivos y complejos.

2.3 Las historias interactivas en el entorno digital

Con la llegada de la era digital, las narraciones interactivas de texto que recuerdan a los librojuegos han ganado nuevamente popularidad. Esto se debe a varias herramientas y productos que han hecho más fácil tanto su creación como su lectura, acercándolas a un público más amplio. En este apartado, examinaremos algunas de estas propuestas, destacando sus puntos fuertes y áreas de mejora.

2.3.1 Inky

Ink es un lenguaje de script desarrollado por Inkle que se creó con la idea de ayudar en la producción de guiones interactivos (13). Aunque puede ser utilizado para redactar historias del tipo Elige tu propia aventura, Ink está diseñado para manejar narraciones más complejas y se puede usar incluso en el desarrollo de videojuegos. El lenguaje utiliza un sistema de marcado que busca ser directo y fácil de comprender, simplificando así su aprendizaje. Además, ofrece una integración con el motor de videojuegos Unity, lo que permite una mayor versatilidad en su uso.

Inky es el principal editor (ilustración 8) para Ink. Esta herramienta proporciona una plataforma donde los usuarios pueden probar sus narraciones a medida que las escriben (14). Entre sus características se incluyen el resaltado de sintaxis, que ayuda a diferenciar el código del texto narrativo, y un sistema de identificación de errores. Asimismo, Inky permite la exportación a formato JSON, lo que es útil para su integración en otros sistemas.

Sin embargo, Inky tiene sus aspectos negativos. La necesidad de escribir código, aunque diseñado para ser simple, puede resultar menos intuitiva, especialmente para aquellos acostumbrados a interfaces gráficas más directas. Además, Inky carece de una representación gráfica del flujo narrativo, lo que podría complicar la visualización general de historias particularmente extensas o intrincadas.

Ilustración 8. Interfaz de Inky

2.3.2 Inklewriter

Inklewriter, al igual que Ink e Inky, es otra creación de Inkle. Mientras Ink e Inky se centran en ofrecer soluciones basadas en el código, Inklewriter toma una dirección diferente. Esta herramienta se distingue por brindar una experiencia centrada en la interfaz gráfica (ver ilustración 9), lo que facilita enormemente el proceso creativo para aquellos menos familiarizados con la programación. La plataforma web de Inklewriter no solo proporciona un entorno de escritura intuitivo, sino que también permite a los usuarios «jugar» la historia conforme la van desarrollando (15).

Un aspecto destacado de Inklewriter es su capacidad para mostrar un grafo de la narración, lo que ofrece una clara representación visual de la estructura y flujo de la historia. Al no incorporar directamente el lenguaje Ink, Inklewriter se ha diseñado con un enfoque en la simplicidad, lo que puede resultar en tramas más lineales. La herramienta permite la exportación en formato JSON para quienes desean integrar sus historias en diferentes plataformas o aplicaciones. No obstante, si se desea convertir una historia de Inklewriter al formato Ink, esta funcionalidad no viene incluida en la herramienta. Para ello, los usuarios deben recurrir a un conversor externo (16) que también desarrolla Inkle.

En cuanto a las limitaciones de Inklewriter, a pesar de su enfoque intuitivo, su simplicidad puede restringir las opciones de personalización en comparación con soluciones más robustas como Ink. La falta de una opción integrada para exportar directamente a Ink puede ser vista como una desventaja. Además, la necesidad de

registrarse y crear una cuenta para guardar las historias podría ser un inconveniente para algunos usuarios.

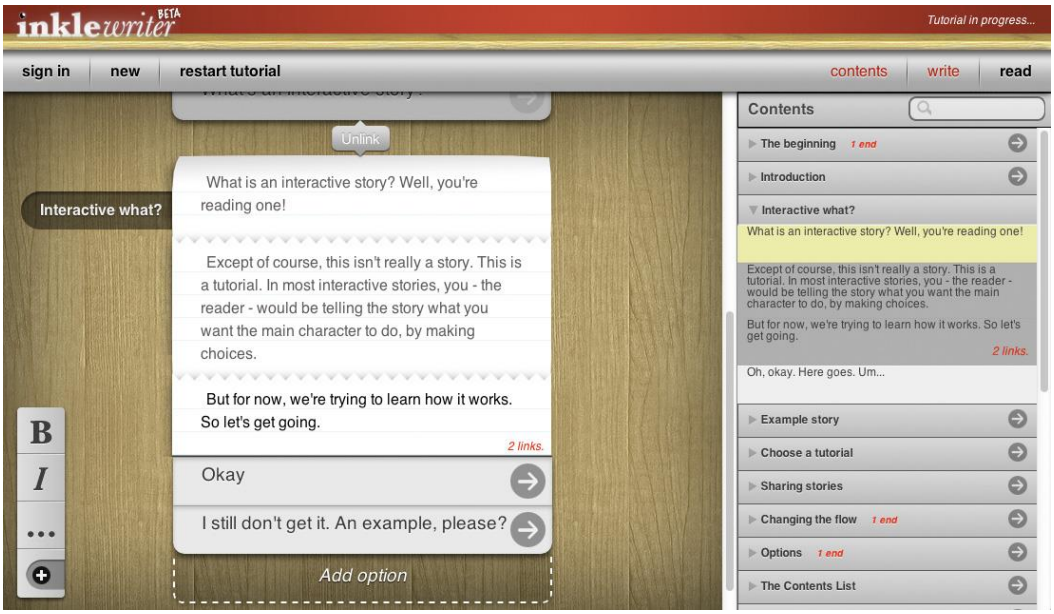

Ilustración 9. Interfaz de Inklewriter

2.3.3 Twine

Twine es una herramienta de código abierto (17) que ha ganado popularidad entre escritores y diseñadores que buscan crear historias interactivas. Su interfaz, basada en la web, permite a los usuarios construir historias no lineales mediante una visualización de nodos (ver ilustración 10), lo que facilita la comprensión y la estructuración de las múltiples rutas y decisiones que pueden tomar los lectores.

Una de las principales características de Twine es su accesibilidad. No se requieren habilidades de programación para empezar, pero aquellos con conocimientos de HTML, CSS y JavaScript pueden aprovechar estos lenguajes para añadir funcionalidades más avanzadas o personalizar la estética de sus narraciones (18).

Más allá de la creación de historias, Twine incorpora funciones que eran ajenas a los librojuegos tradicionales. Por ejemplo, permite registrar decisiones y acciones realizadas por el jugador en etapas anteriores de la narración (19), como comprobar si el jugador recogió una llave en un momento específico. Esta capacidad da a los autores la oportunidad de diseñar rompecabezas, enriqueciendo la experiencia del lector. Un ejemplo del alcance y versatilidad de Twine es su utilización en la elaboración del guion para el episodio de *Black Mirror: Bandersnatch* (20), que mencionamos en anteriores apartados.

No obstante, Twine presenta algunas limitaciones. Una de las más destacadas es la falta de una opción para exportar a Ink. Además, los archivos HTML generados por Twine tienden a ser de un tamaño considerable, lo que puede suponer un desafío cuando se buscan distribuir en plataformas con restricciones de almacenamiento.

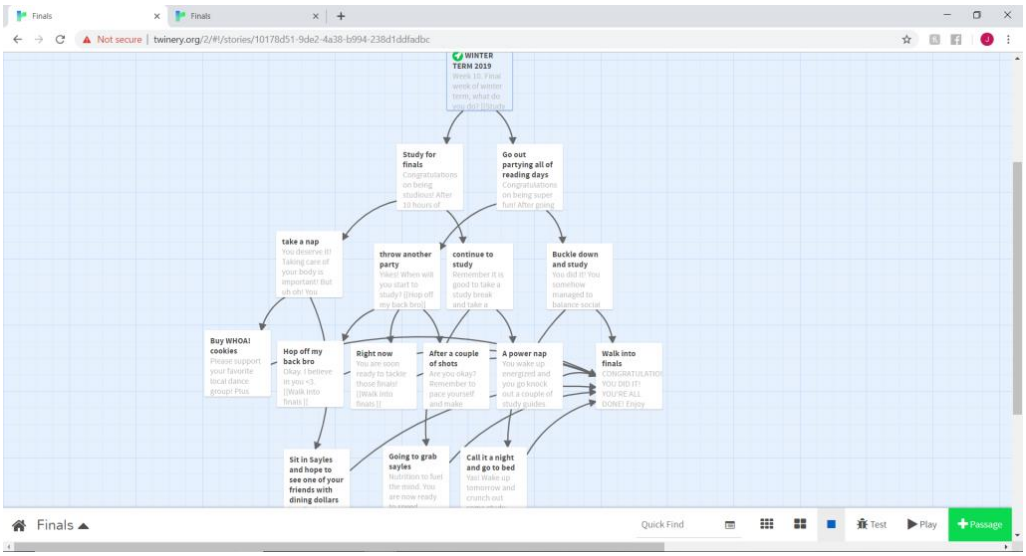

Ilustración 10. Interfaz de Twine

2.3.4 Comparativa de las herramientas

La tabla 1 contiene un resumen comparando las herramientas que hemos analizado.

Tabla 1. Comparativa de las herramientas

	Inky	Inklewriter	Twine
Grafo de estructura	X	✓	✓
No requiere código	X	✓	✓
Exportación a Json	✓	✓	X
Historias complejas	✓	X	✓
Formato Ink	✓	Requiere una herramienta externa para conversión	X
Plataforma web	X	✓	✓
Jugar mientras escribes	✓	✓	✓

2.3.5 StreamINK

En la última década, plataformas de *streaming* como Twitch (21) y YouTube han experimentado un crecimiento exponencial. Estos medios digitales han trascendido más allá del simple entretenimiento, convirtiéndose en auténticas comunidades donde los creadores y su público interactúan de manera directa y constante. Esta relación bidireccional no solo ha transformado la forma en que consumimos contenido, sino que ha abierto la puerta a nuevas posibilidades en cuanto a la experiencia del usuario y la participación activa de la audiencia.

Surge entonces StreamINK (22), una propuesta innovadora de Rothio Tome que busca unificar el potencial de las narraciones interactivas con la inmediatez de las plataformas de *streaming*. La idea central de StreamINK es sencilla pero potente: llevar las historias en formato Ink a los directos, permitiendo así que la comunidad, en tiempo real, tenga un papel determinante en el desarrollo de la trama. Este enfoque no solo fomenta la lectura y el consumo de historias interactivas, sino que potencia la interacción y el compromiso de la audiencia, convirtiendo la experiencia narrativa en un acto colectivo y participativo.

Desde un punto de vista técnico, StreamINK es una herramienta construida sobre el motor Unity. Se integra de manera sencilla con Twitch, permitiendo a los creadores conectar sus cuentas y cargar archivos Ink de manera directa. Una vez que la historia se pone en marcha en la emisión en directo (ilustración 11), cuando surgen opciones narrativas la audiencia tiene la oportunidad de decidir el curso de la trama a través de sus comentarios en el chat. StreamINK interpreta estas interacciones como votaciones, y la opción más popular dicta el siguiente paso en la narración. Esta dinámica refuerza la idea de una experiencia compartida, en la que tanto el creador como la audiencia se embarcan juntos en un viaje narrativo, determinando colectivamente su destino.

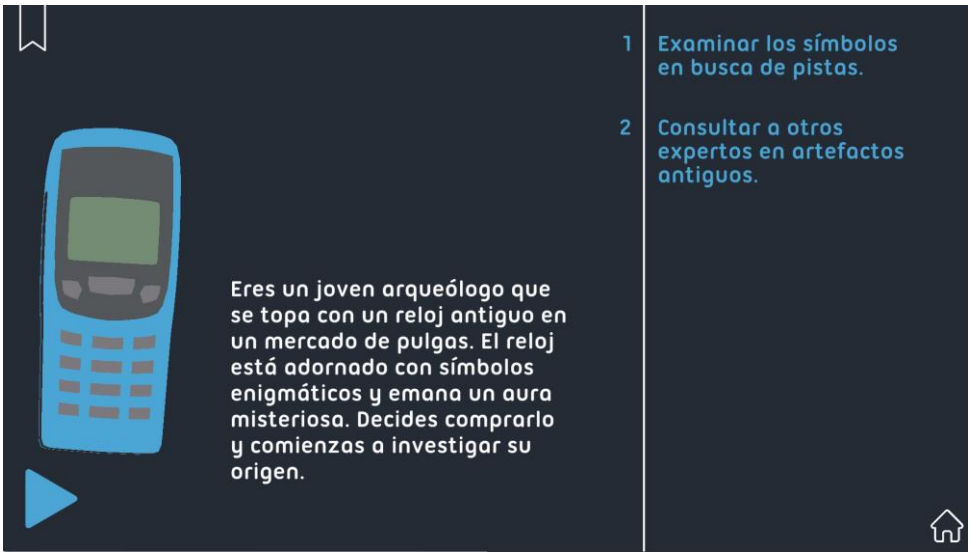

Ilustración 11. Interfaz de StreamINK

2.4 Propuesta

Teniendo en cuenta el análisis realizado sobre las alternativas existentes en el mercado, se propone desarrollar una herramienta que aborde las limitaciones de los programas mencionados sin añadir complejidad adicional que dificulte su utilización. Con este fin, se plantea sistema que no requiera de una cuenta, priorizando la simplicidad y accesibilidad para el usuario.

La creación de la historia se llevará a cabo mediante una interfaz a base de formularios, evitando así la necesidad de programar en el lenguaje Ink y, al mismo tiempo, previniendo posibles errores en la sintaxis. Este enfoque facilita la comprensión de la estructura narrativa y mejora la experiencia de usuario en comparación con las soluciones existentes.

Además, la herramienta propuesta permitirá importar y abrir archivos para su edición y posterior guardado, ofreciendo mayor versatilidad y adaptabilidad a las necesidades de los usuarios. También contará con la inclusión de un grafo para

visualizar la estructura de la historia, así como unas estadísticas básicas con las que entender mejor la narración.

En resumen, la propuesta consiste en el desarrollo de una solución que mejore las funcionalidades de las herramientas actuales sin incrementar su complejidad, proporcionando una experiencia de usuario más intuitiva y eficiente en la creación de historias interactivas.

3. Análisis del problema

El análisis del problema es una etapa crítica en cualquier proyecto, donde se busca comprender a profundidad las necesidades y restricciones del sistema a desarrollar. Durante esta fase, se identifican los requerimientos específicos, se detallan los casos de uso y se estudian diferentes enfoques o soluciones posibles. Una vez recopilada y analizada toda la información relevante, se selecciona la solución propuesta que mejor se adapte al contexto y objetivos del proyecto. En los siguientes apartados, abordaremos cada uno de estos elementos de manera detallada.

3.1 Análisis de Requisitos

El análisis de requisitos es esencial para definir las expectativas del sistema. Distinguimos entre requisitos funcionales, agrupados según funcionalidades concretas, que especifican lo que el sistema debe hacer; y los no funcionales, que definen características de calidad como usabilidad o rendimiento. A continuación, presentaremos la lista de los requisitos identificados.

3.1.1 Requisitos funcionales

RF de gestión de historia:

RF1 - Creación de historias: El sistema debe permitir al usuario crear una nueva historia desde cero.

RF2 - Guardado de historias: El usuario puede guardar su historia en curso en formato JSON en su dispositivo.

RF3 - Acceso a historias existentes: El usuario puede abrir una historia que haya guardado previamente en formato JSON desde su dispositivo.

RF4 - Exportación de historias: El sistema debe permitir al usuario exportar la historia a un archivo local en formato Ink, compatible con StreamINK.

RF5 - Modificación del título de la historia: El usuario debe poder cambiar el título de su historia en cualquier momento.

RF de gestión de capítulos:

RF6 - Creación de capítulos: El sistema debe permitir al usuario añadir un nuevo capítulo proporcionando un título.

RF7 - Acceso a capítulos: El usuario debe poder visualizar una lista de capítulos y seleccionar uno para abrirlo y editar o leer su contenido.

RF8 - Renombrado de capítulos: El usuario debe poder modificar el título de un capítulo existente.

RF9 - Eliminación de capítulos: El sistema debe proporcionar una opción para que el usuario pueda eliminar un capítulo específico.

RF10 - Búsqueda de capítulos: El sistema debe permitir realizar una búsqueda de los capítulos por nombre o contenido.

RF de gestión de contenido:

RF11 - Edición de texto de capítulos: El sistema debe proporcionar un editor para que el usuario pueda modificar el texto principal de un capítulo.

RF12 - Adición de opciones en capítulos: El usuario debe poder añadir opciones dentro de un capítulo que dirigirán la historia hacia diferentes rutas, ya sea a otros capítulos o al mismo capítulo.

RF13 - Modificación de opciones en capítulos: El sistema debe proporcionar una interfaz donde el usuario pueda editar las opciones interactivas ya existentes en un capítulo.

RF14 - Eliminación de opciones en capítulos: El usuario debe poder eliminar una opción específica de un capítulo.

RF de análisis de la historia:

RF15 - Representación gráfica de la historia: Basándose en los capítulos y opciones, el sistema debe generar un grafo que visualice la estructura y flujo de la historia.

RF16 - Estadísticas de la historia: El sistema debe ofrecer un resumen con estadísticas sobre la historia, como los caminos más cortos o el PageRank.

3.1.2 Requisitos no funcionales

RNF1 - Usabilidad: El sistema debe presentar una interfaz de usuario intuitiva y amigable, diseñada de tal manera que permita a personas, incluso sin conocimientos técnicos previos, crear historias sin dificultades.

RNF2 - Rendimiento: La aplicación debe garantizar interacciones ágiles y rápidas, asegurando tiempos de carga que sean aceptables para el usuario.

RNF3 - Interoperabilidad: El sistema debe ser capaz de exportar historias en formato Ink de manera que sean compatibles y fácilmente integrables con StreamINK.

RNF4 - Disponibilidad: La plataforma debe estar accesible para los usuarios el 99.99% del tiempo, minimizando interrupciones no planeadas.

RNF5 - Fiabilidad: La aplicación debe operar de manera estable y sin fallos que comprometan la experiencia del usuario o la integridad de los datos.

3.2 Casos de uso

Los casos de uso representan las interacciones entre el usuario y el sistema, detallando cómo se espera que funcione la solución ante diversas situaciones o requerimientos. A través de ellos, es posible entender de manera clara y estructurada las funciones principales que el software debe cumplir y cómo los usuarios se beneficiarán de ellas. A continuación, se presentarán los casos de uso específicos que han sido identificados y desarrollados para este proyecto.

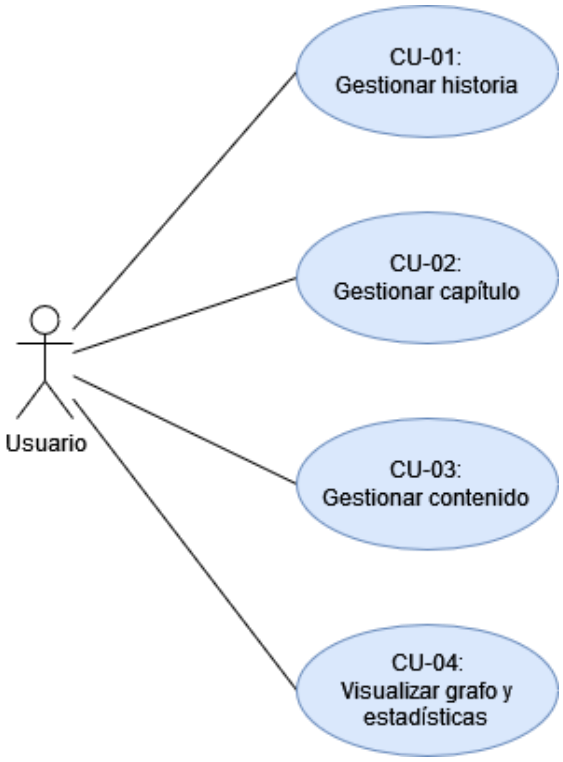

Ilustración 12. Diagrama de casos de uso

CU-01	Gestionar historia
Descripción	El usuario debe ser capaz de crear una nueva historia, modificar el título, guardar y cargar la historia mediante archivos JSON y exportar la historia a formato Ink.
Actores	Usuario
Precondición	Ninguna
Resultado	El usuario tiene una historia generada lista para ejecutar.
Requisitos	RF1, RF2, RF3, RF4, RF5

CU-02	Gestionar capítulo
Descripción	El usuario debe ser capaz de administrar los capítulos dentro de una historia, permitiendo crear nuevos capítulos, buscarlos, abrirlos, modificar su título, y eliminar capítulos.
Actores	Usuario
Precondición	Tener una historia.
Resultado	El usuario tiene un capítulo.
Requisitos	RF6, RF7, RF8, RF9, RF10

CU-03	Gestionar contenido
Descripción	El usuario debe ser capaz de modificar el texto de un capítulo, así como añadir, modificar y eliminar opciones interactivas en el capítulo.
Actores	Usuario
Precondición	Tener un capítulo abierto.
Resultado	El usuario tiene el contenido de un capítulo y sus redirecciones.
Requisitos	RF11, RF12, RF13, RF14

CU-04	Visualizar grafo y estadísticas
Descripción	El usuario debe ser capaz de visualizar un grafo representativo de la historia, así como a una sección de estadísticas con información de la historia.
Actores	Usuario
Precondición	Tener una historia.
Resultado	El usuario puede acceder al grafo y a las estadísticas.
Requisitos	RF15, RF16

3.3 Identificación y análisis de soluciones posibles

En el proceso de búsqueda de la solución más adecuada para el desarrollo de la herramienta, se han considerado diversas opciones. A continuación, se analizan sus ventajas y desventajas con el objetivo de tomar una decisión apropiada para nuestros requisitos.

3.3.1 Aplicación móvil

Una aplicación móvil es accesible desde cualquier lugar y en cualquier momento. Este tipo de solución puede llegar a un amplio número de usuarios y proporcionar una experiencia optimizada para pantallas táctiles.

Sin embargo, esta solución se descartó debido a las complejidades asociadas con el desarrollo multiplataforma. El proceso de desarrollo de una aplicación móvil que funcione tanto en Android como en iOS es costoso y consume mucho tiempo. Además, la naturaleza del proyecto, centrada en la edición y creación de historias interactivas, se presta mejor a una pantalla más grande donde puede entrar más contenido.

3.3.2 Aplicación de escritorio

Las aplicaciones de escritorio utilizan al máximo los recursos de un ordenador. Muchos usuarios las prefieren debido a la familiaridad que ofrecen, ofreciendo una experiencia consistente.

A pesar de sus ventajas, esta solución también se descartó. El proceso de instalación puede ser una barrera para algunos usuarios, ya que implica descargar e instalar la aplicación antes de poder usarla. Además, el auge de las soluciones basadas en la nube ha llevado a una disminución en la demanda de aplicaciones de escritorio tradicionales.

3.4 Solución propuesta

La solución propuesta para el proyecto es una aplicación web. Esto nos ofrece ventajas como la accesibilidad desde cualquier dispositivo equipado con un navegador y conexión a internet, eliminando la necesidad de instalaciones. Además, evitan las complejidades asociadas con el desarrollo multiplataforma.

Aunque normalmente este tipo de aplicaciones tiene la desventaja de requerir una conexión constante, la utilización del *framework* Blazor WebAssembly, en el que profundizaremos posteriormente en el apartado 4.3, nos va a evitar esta limitación.

En términos de características, la aplicación web nos ofrecerá:

- **Compatibilidad multiplataforma:** Funcionamiento en diversos navegadores y dispositivos.
- **Interfaz amigable:** La plataforma nos permite mostrar la información de forma clara y una navegación intuitiva.
- **Modularidad:** La aplicación deberá estar construida de manera modular y escalable, permitiendo la adición de nuevas funcionalidades.

Para el desarrollo del proyecto se ha elegido el modelo de desarrollo en cascada, una metodología lineal y secuencial que se ajusta a las características y objetivos del proyecto. A continuación, se describen las distintas fases por las que pasará el desarrollo:

1. Análisis de Requisitos: Durante esta etapa, se recopilan todas las necesidades y especificaciones del proyecto, tanto funcionales como no funcionales.
2. Diseño del Sistema: Una vez establecidos los requisitos, se procede al diseño global y detallado de la aplicación. Esto incluye tanto el diseño de la arquitectura del software como la interfaz de usuario.
3. Desarrollo: Durante esta fase, se lleva a cabo la implementación siguiendo las especificaciones definidas en las fases anteriores.
4. Pruebas: Una vez el código esté listo, se realizan unas pruebas funcionales para verificar si el software realiza las funciones especificadas en los requisitos.

4. Diseño

El diseño de un sistema informático es esencial para garantizar que el software resultante sea robusto, eficiente y cumpla con los requisitos previamente establecidos. En esta sección, abordaremos la estructura global del software a través de la Arquitectura del Sistema, seguido de un acercamiento más profundo con el Diseño Detallado, y finalizaremos con la elección y justificación de las herramientas en el apartado de Tecnología Utilizada.

4.1 Arquitectura del Sistema

Dentro del desarrollo de software, la elección de una arquitectura adecuada es crucial para obtener un sistema que se adapte a tus necesidades. La arquitectura del sistema establece la organización del proyecto y guía la implementación de las funcionalidades y características requeridas. Para este proyecto, nos hemos inspirado en el patrón Cliente-Servidor, un modelo bien establecido y ampliamente utilizado. Sin embargo, hemos introducido algunas adaptaciones beneficiosas (ver ilustración 12) gracias al uso del *framework* Blazor WebAssembly, con las que adaptarnos más a nuestras necesidades. A continuación, se describen los principales bloques de la arquitectura:

- **El servidor:** Su principal tarea es facilitar la primera carga de la aplicación. A diferencia de la tradicional arquitectura Cliente-Servidor, donde el servidor tiene un papel más activo en el procesamiento, en esta adaptación la intervención del servidor es mínima. Esencialmente, actúa como un distribuidor inicial del software al cliente.
- **El cliente:** Tras la carga inicial, el cliente se encarga de realizar todas las operaciones y procesamientos. Gracias al uso de WebAssembly, todo el trabajo de la aplicación se ejecuta directamente en el navegador del usuario. Esto permite un rendimiento rápido y optimizado, ofreciendo una experiencia más fluida.
- **Ficheros JSON:** Estos ficheros juegan un papel crucial en el sistema de guardado y carga de las historias. Ofrecen una forma estructurada y estándar de almacenar y recuperar la información, garantizando la integridad y coherencia de las historias creadas.
- **Ficheros Ink:** Una vez que las historias se han creado y estructurado, pueden exportarse en formato Ink. Esta exportación permite el uso de las historias en la plataforma StreamINK para su posterior ejecución por parte de los jugadores.

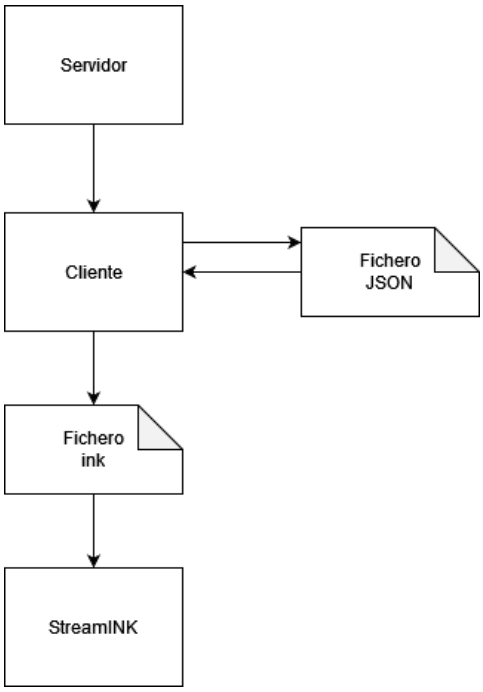

Ilustración 13. Arquitectura del sistema

4.2 Diseño Detallado

Tras establecer los cimientos de nuestra arquitectura, vamos a entrar en detalle en el diseño de la solución, proporcionando una visión clara y profunda de cómo se organizará y estructurará cada componente del proyecto. En este apartado, vamos a describir la organización del sistema a través de la estructura de directorios. Posteriormente presentaremos los diagramas de clases que detallarán las relaciones entre las entidades de nuestro sistema. Por último, se mostrarán los *mockups* de la interfaz, ofreciendo una previsualización del aspecto y usabilidad que los usuarios tendrán al interactuar con la aplicación.

4.2.1 Estructura de la aplicación

A continuación, detallaremos la organización interna de nuestro proyecto. Si bien el uso del *framework* Blazor WebAssembly nos proporciona una estructura de archivos predefinida, como se puede ver en la ilustración 14, es esencial entender la disposición y propósito de cada directorio:

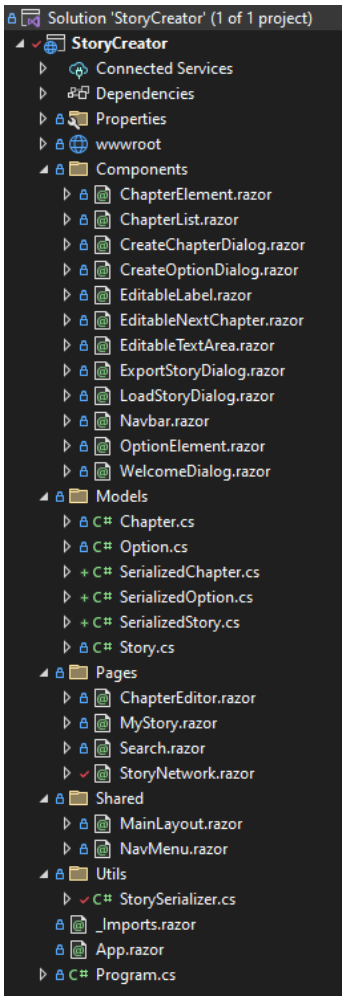

Ilustración 14. Estructura de la aplicación

● ***Connected Services*, *Dependencies* y *Properties*:** Estas carpetas contienen configuraciones del proyecto y dependencias de librerías externas.

● ***wwwroot*:** Aquí se aloja el contenido estático típico de una web, incluyendo los ficheros de estilo CSS, scripts JavaScript para funcionalidades que no se logren mediante C# y la carpeta de recursos donde se almacenan imágenes.

● ***Components*:** Esta carpeta alberga los componentes reutilizables que se integran en distintas páginas, facilitando así una experiencia uniforme para el usuario.

● ***Models*:** En este directorio se definen las clases que representan la historia y facilitan el funcionamiento de la aplicación. Asimismo, se encuentran clases auxiliares para la serialización y el almacenamiento en formato JSON.

● ***Pages*:** Contiene las diversas páginas con las que el usuario interactúa, definiendo así la experiencia principal de navegación.

● ***Shared*:** Aquí se localizan elementos de diseño que forman parte de la estructura base de la interfaz, como una barra lateral persistente o cabeceras comunes.

● ***Utils*:** Alberga herramientas adicionales, como el serializador que posibilita el sistema de carga y almacenado de ficheros en formato JSON.

● ***_Imports.razor, App.razor* y *Program.cs*:** Estos archivos son cruciales para el arranque y la operativa general de la aplicación, además de definir importaciones comunes de librerías y módulos.

4.2.2 Diagrama de clases

Para comprender mejor la estructura y relaciones de las entidades que conforman una historia en nuestro sistema, se ha realizado un diagrama de clases UML (ver ilustración 15). Este diagrama nos proporciona una representación visual de las clases principales, sus atributos, métodos y cómo se relacionan entre sí, ofreciendo una perspectiva clara de la organización de las entidades en el sistema.

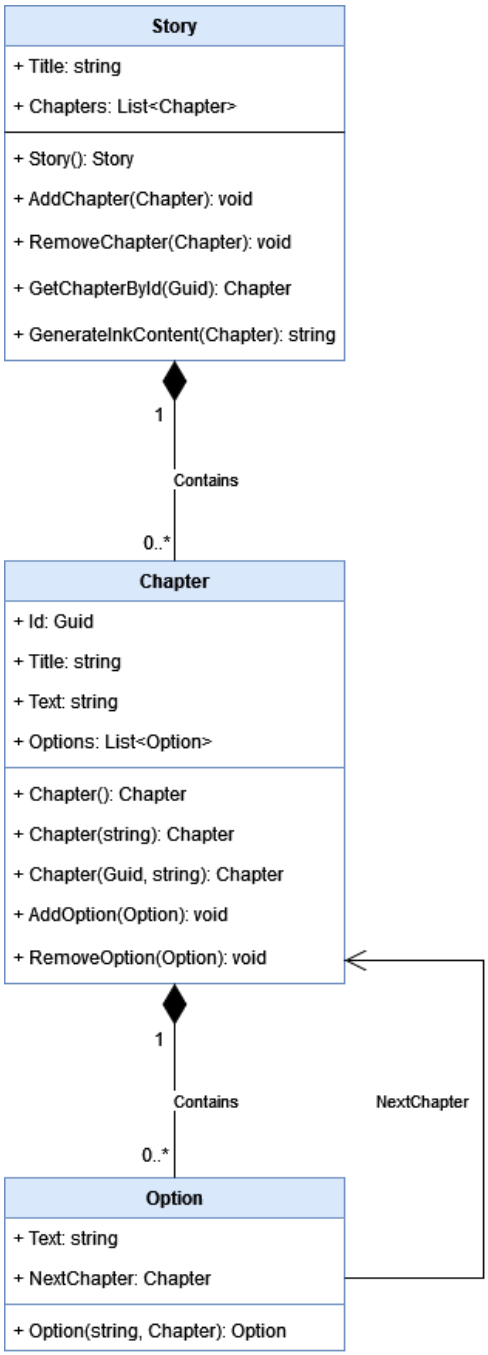

Ilustración 15. Diagrama de clases

- `Story`: Esta clase alberga la información principal de la historia, comprendiendo el título y una lista de capítulos que la componen. Proporciona métodos para añadir, eliminar y encontrar capítulos específicos dentro de la historia. Adicionalmente, posee el método que facilita la conversión de la historia al formato Ink.
- `Chapter`: Representa un capítulo individual dentro de una historia. Contiene datos como el título y el contenido del capítulo, así como una lista de opciones interactivas asociadas a él. Entre sus métodos destacan aquellos que permiten añadir o eliminar opciones a este capítulo.

- `Option`: Se encarga de representar las opciones interactivas dentro de un capítulo. Incluye detalles como el texto de la opción y una referencia al capítulo al que el usuario será dirigido si elige esta opción.

Aunque se profundizará más adelante en el capítulo 5, es pertinente mencionar que las clases originales pueden presentar complicaciones al intentar serializarse, dadas las múltiples referencias cruzadas entre ellas. Es por esta razón que, con el objetivo de implementar un sistema de guardado y carga de la historia a través de ficheros JSON, se han diseñado clases auxiliares que simplifican el proceso de serialización y deserialización a este formato. La ilustración 16 muestra el diagrama de clases de estas entidades auxiliares.

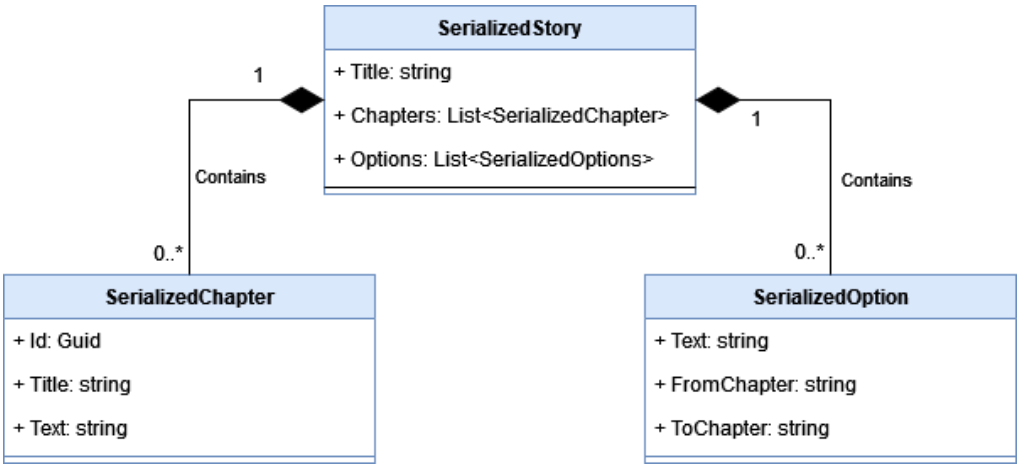

Ilustración 16. Diagrama de clases auxiliares

Como se observa, la principal modificación radica en que tanto los capítulos como las opciones ahora se integran directamente dentro de la historia. Además, la clase `SerializedOption` ya no hace referencia directa a los objetos `SerializedChapter`, sino a sus identificadores. Dado que las opciones están ahora alojadas en la historia y no en el capítulo, se ha añadido un atributo que indica el capítulo de origen de la opción. Estas adaptaciones no solo facilitan una serialización más directa, sino que, además, suelen ser una representación típica de grafos direccionales mediante clases.

4.2.3 Estructura de los ficheros JSON

Los ficheros JSON utilizados en nuestra solución tienen como finalidad almacenar la información de una historia, permitiendo que los usuarios puedan guardar, cargar y compartir sus historias en un formato estructurado y fácilmente legible. A continuación se define su estructura, que no es más que la serialización de las clases mencionadas en el último apartado:

- `Title`: Representa el título general de la historia.
- `Chapters`: Es una lista de capítulos que componen la narrativa principal de la historia. Cada capítulo contiene:
  - `Id`: Es un identificador único que permite distinguir cada capítulo de los demás.
  - `Title`: El título específico de ese capítulo.
  - `Text`: La narrativa del capítulo, donde se desarrolla una parte específica de la historia.
- `Options`: Representa una lista de opciones o decisiones que el jugador puede tomar en el transcurso de la historia. Cada opción contiene:

- `Text`: Describe la acción o decisión que puede tomar el jugador.
- `FromChapter`: Identificador del capítulo donde esta opción es presentada.
- `ToChapter`: Identificador del capítulo al que lleva esta opción al ser seleccionada.

A continuación, la ilustración 17 muestra un ejemplo de una historia corta de 3 capítulos ya en formato JSON.


```json
{
  "Title": "El amuleto perdido",
  "Chapters": [
    {
      "Id": "4eb58a01-efbf-415c-97c6-938fafd04917",
      "Title": "1 - El encuentro",
      "Text": "En un mercado ex\u00F3tico, descubres un viejo amuleto...
    },
    {
      "Id": "22fc8762-2ab9-427b-8421-8afbe3a73234",
      "Title": "2 - Los Secretos Revelados",
      "Text": "Despu\u00E9s de estudiar las inscripciones, descifras...
    },
    {
      "Id": "0c9ef405-ac52-4b19-a466-78852f9a6124",
      "Title": "3 - El Tesoro Antiguo",
      "Text": "Siguiendo las pistas del amuleto, llegas a una cueva...
    }
  ],
  "Options": [
    {
      "Text": "Examinar las inscripciones en busca de pistas. ",
      "FromChapter": "4eb58a01-efbf-415c-97c6-938fafd04917",
      "ToChapter": "22fc8762-2ab9-427b-8421-8afbe3a73234"
    },
    {
      "Text": "Consultar a un historiador sobre el amuleto. ",
      "FromChapter": "4eb58a01-efbf-415c-97c6-938fafd04917",
      "ToChapter": "0c9ef405-ac52-4b19-a466-78852f9a6124"
    },
    {
      "Text": "Seguir las pistas del amuleto.",
      "FromChapter": "22fc8762-2ab9-427b-8421-8afbe3a73234",
      "ToChapter": "0c9ef405-ac52-4b19-a466-78852f9a6124"
    },
    {
      "Text": "Investigar m\u00E1s sobre la historia de la c\u00E1mara ...
      "FromChapter": "22fc8762-2ab9-427b-8421-8afbe3a73234",
      "ToChapter": "0c9ef405-ac52-4b19-a466-78852f9a6124"
    }
  ]
}
```


Ilustración 17. Ejemplo de fichero JSON

El uso de esta estructura en formato JSON permite:

1. **Flexibilidad:** Las historias pueden tener tantos capítulos y opciones como el creador desee.
2. **Intuitividad:** Gracias a la clara segmentación entre capítulos y opciones, es fácil entender el flujo de la historia.

3. **Interconexión:** Los identificadores únicos permiten conectar fácilmente las opciones con los capítulos correspondientes, creando un entramado interactivo para el jugador.
4. **Interoperabilidad:** Al estar en un formato estándar como JSON, las historias pueden ser compartidas, editadas o incluso interpretadas por otros sistemas o aplicaciones.

4.2.4 Diseño de la interfaz

El diseño de la interfaz de nuestro aplicativo web se ha planteado para ser intuitivo, garantizando una experiencia de usuario fluida y agradable. El prototipado se ha realizado mediante *mockups*.

Además, es relevante señalar que la aplicación ha sido diseñada en inglés con el objetivo de acceder a un público más amplio y global. Así, las capturas de los prototipos que se mostrarán a continuación estarán en dicho idioma.

Principios del diseño

Antes de detallar los prototipos, es importante mencionar los principios que han guiado el diseño de la interfaz:

1. **Usabilidad:** La interfaz debe ser sencilla y fácil de usar, evitando complicaciones innecesarias.
2. **Claridad:** Todos los elementos y textos deben ser legibles y estar claramente etiquetados.
3. **Consistencia:** Los elementos similares deben comportarse de la misma manera en toda la aplicación.

Prototipos

Los *mockups* son bocetos que representan la apariencia y funcionalidad básica de la aplicación. Estos son los *mockups* diseñados para nuestra solución:

- **Página de inicio:** Diálogo de inicio de la aplicación, con la opción de crear o cargar una historia. Vinculado a los requisitos funcionales RF1, RF3 y a los casos de uso CU-01 (ilustración 18).

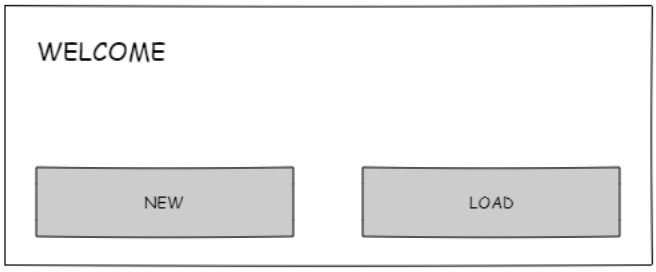

Ilustración 18. Prototipo página de inicio

- **Cargar historia:** Diálogo para cargar ficheros locales de historias. Vinculado al requisito funcional RF3 y al caso de uso CU-01 (ilustración 19).

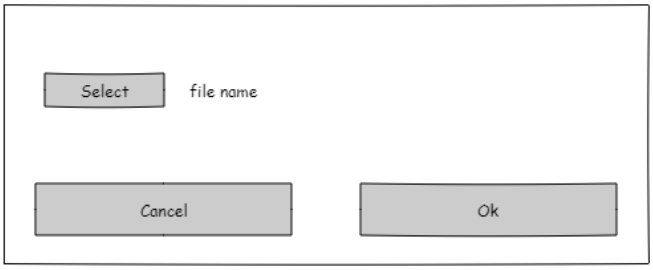

Ilustración 19. Prototipo diálogo para cargar historias

- **Opciones de la historia:** Página con opciones de la historia como el título, guardar en JSON o exportar la historia a Ink. Vinculado a los requisitos funcionales RF2, RF4, RF5 y al caso de uso CU-01 (ilustración 20).

Ilustración 20. Prototipo página de opciones de la historia

- **Crear capítulo:** Diálogo para añadir nuevos capítulos. Vinculado al requisito funcional RF6 y al caso de uso CU-02 (ilustración 21).

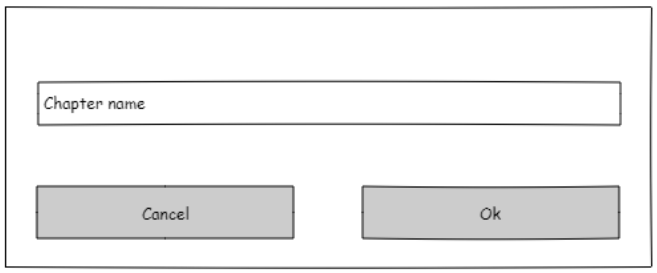

Ilustración 21. Prototipo diálogo de creación de capítulos

- **Ver capítulo:** Página de visualización y edición del capítulo y su contenido. Vinculado a los requisitos funcionales RF7, RF8, RF9, RF11, RF12, RF13, RF14 y a los casos de uso CU-02 y CU-03 (ilustración 22).

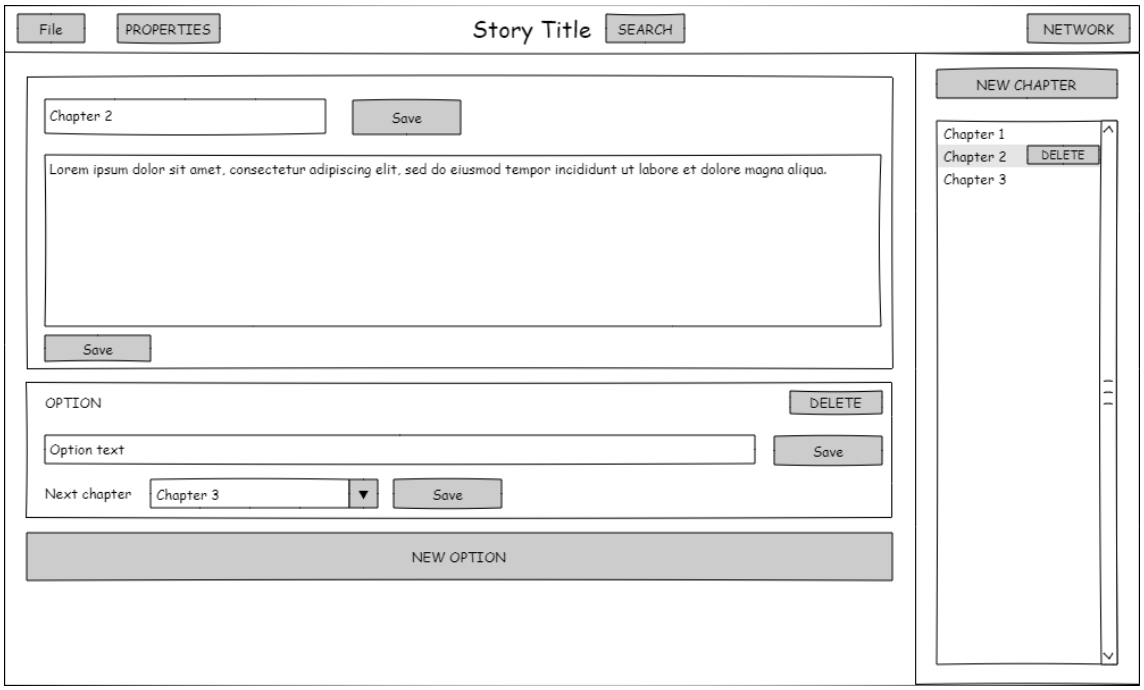

Ilustración 22. Prototipo página de capítulo

- **Crear opción:** Diálogo para la creación de una opción de un capítulo. Vinculado al requisito funcional RF12 y al caso de uso CU-03 (ilustración 23).

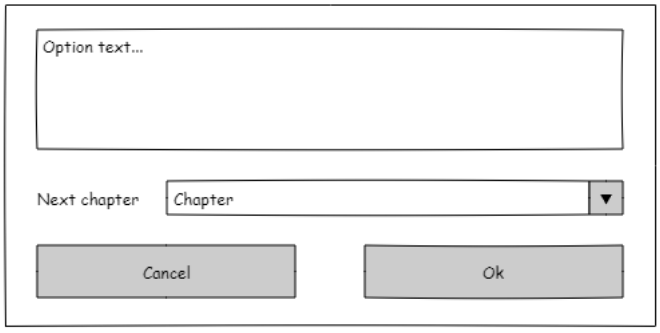

Ilustración 23. Prototipo diálogo de creación de opciones

- **Buscar:** Página que permite realizar búsquedas en la historia. Vinculado al requisito funcional RF10 y al caso de uso CU-02 (ilustración 24).

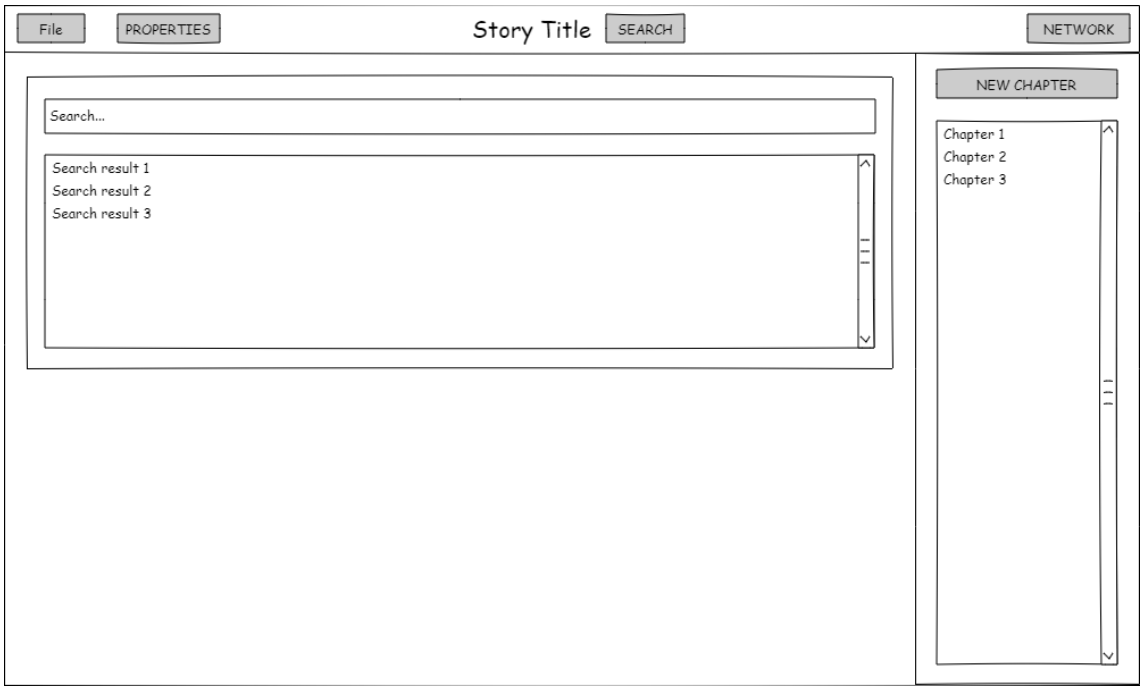

Ilustración 24. Prototipo página de búsqueda

- **Grafo y análisis de la historia:** Página para visualizar el grafo representativo de la historia y obtener un análisis de la misma. Vinculado a los requisitos funcionales RF15, RF16 y al caso de uso CU-04 (ilustración 25).

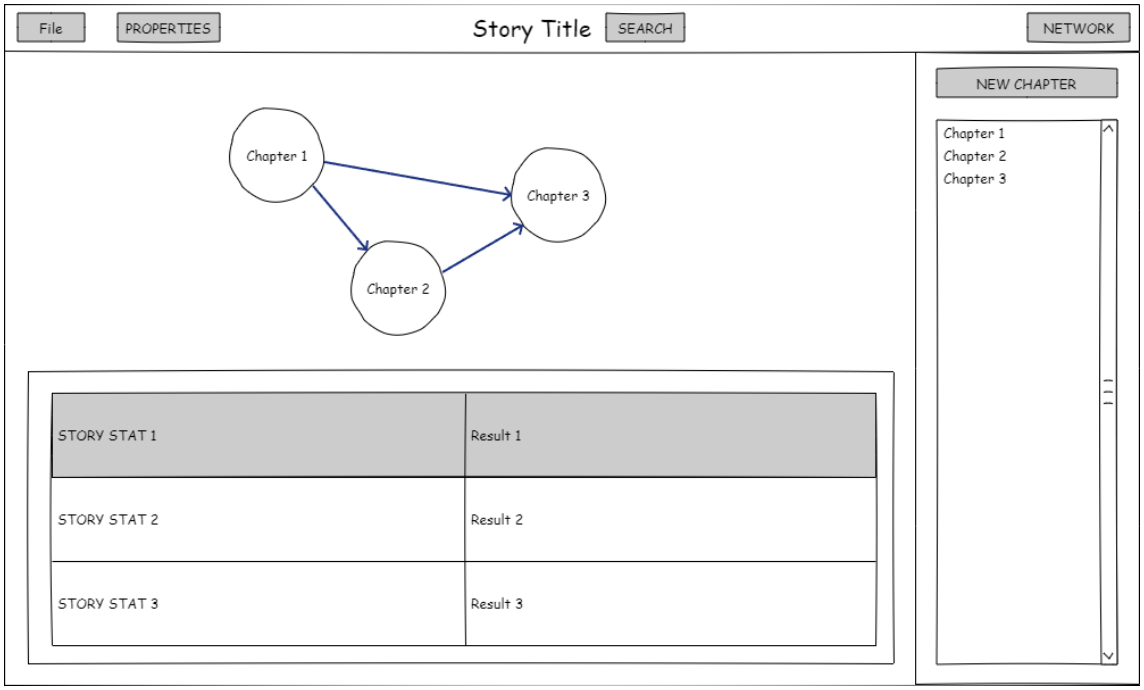

Ilustración 25. Prototipo página de grafo y análisis

- **Exportar a Ink:** Diálogo que permite exportar la historia al formato Ink. Vinculado al requisito funcional RF4 y al caso de uso CU-01 (ilustración 26).

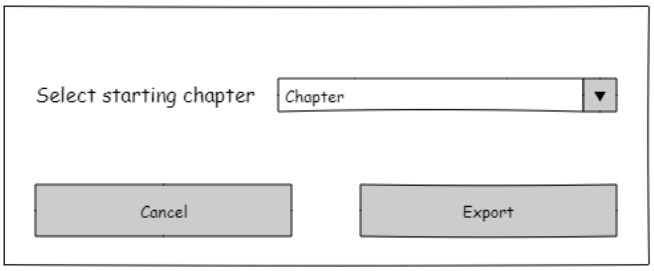

Ilustración 26. Prototipo diálogo de exportación

4.3 Tecnología Utilizada

El proceso de desarrollo de un producto de software funcional se basa en gran medida en las tecnologías empleadas. Las decisiones en torno a los lenguajes de programación, librerías y herramientas no sólo determinan la viabilidad del proyecto, sino también su eficiencia y robustez. En este apartado detallaremos el entorno tecnológico que rodea al proyecto, desglosando cada elemento y su rol dentro de la solución planteada.

4.3.1 C#

C#, desarrollado por Microsoft, es un lenguaje de programación orientado a objetos reconocido por su robustez y versatilidad en diversos campos del desarrollo. Destaca por su sintaxis intuitiva, el tipado fuerte y su capacidad multiplataforma (23), características que lo han consolidado como el cuarto lenguaje de programación más popular en la actualidad (24).

La elección de C# para este proyecto no solo se basa en estos atributos, sino también en el respaldo de una comunidad activa y la vasta gama de bibliotecas disponibles. Además, la sinergia entre C# y el *framework* Blazor WebAssembly fortalece aún más la decisión, ofreciendo un entorno potente y adecuado para el desarrollo web.

4.3.2 Blazor WebAssembly

Blazor WebAssembly es un *framework* de Microsoft que permite a los desarrolladores construir aplicaciones web interactivas usando C# en lugar de JavaScript. Esta tecnología se asienta sobre WebAssembly, una especificación web que facilita la ejecución de código a velocidades cercanas a las nativas directamente en el navegador (25). Una de las grandes ventajas de Blazor WebAssembly es su capacidad de combinar el rendimiento eficiente con una fácil interoperabilidad con JavaScript. Esto permite combinar las librerías propias de C# con el uso de librerías específicas para JavaScript.

Elegir Blazor WebAssembly para este proyecto nos elimina una de las desventajas clásicas de las aplicaciones web. Una vez que se realiza la carga inicial desde el servidor, toda la aplicación se ejecuta en el cliente. Esto significa que, después de la carga inicial, no es necesario mantener una conexión constante a Internet, ofreciendo una experiencia más fluida y menos dependiente de la conectividad.

Además, el empleo tanto de Blazor WebAssembly como de C# se ve reforzado por una familiaridad previa con el ecosistema .NET al que ambos pertenecen, lo que facilita trabajar en un entorno coherente y bien comprendido, optimizando así el proceso de desarrollo.

4.3.3 HTML

HTML, que proviene de las siglas en inglés *HyperText Markup Language*, es el lenguaje estándar de marcado utilizado para crear y diseñar páginas web (26). Aunque Blazor WebAssembly permite desarrollar aplicaciones web principalmente con C#, aún se hace uso de HTML para definir la estructura y contenido de la interfaz. En este proyecto, HTML desempeña un papel fundamental en la creación de componentes visuales y plantillas, que posteriormente se enriquecen y dinamizan mediante la lógica de negocio implementada con Blazor y C#.

4.3.4 CSS

CSS, siglas de *Cascading Style Sheets*, es la herramienta principal para definir la presentación visual de las páginas web (27). Funciona como el marco estético que complementa la estructura establecida por el HTML. A través de CSS, se determinan aspectos como colores, tipografías, espaciados y la disposición general de los elementos en una página. En el contexto de este proyecto, CSS ha sido esencial para lograr una interfaz de usuario atractiva y coherente, garantizando así una experiencia de usuario de alta calidad.

4.3.5 JavaScript

JavaScript es un lenguaje de programación ampliamente utilizado para agregar interactividad y dinamismo a las páginas web. A pesar de su omnipresencia en el desarrollo web, en este proyecto su uso ha sido limitado. Se ha recurrido a JavaScript únicamente para la implementación de ciertas características específicas que no son directamente soportadas por Blazor, asegurando así que la aplicación mantenga una funcionalidad completa.

4.3.6 Radzen

Radzen Blazor Components es una biblioteca compuesta por un conjunto de componentes UI para Blazor (28), la cual facilita la creación de interfaces de usuario modernas y *responsive*. Estos componentes están diseñados para ser fácilmente integrables, ofreciendo una amplia variedad de elementos de interfaz, desde controles básicos, como botones y desplegables, hasta componentes más avanzados, como gráficos y tablas. Su integración en el proyecto ha permitido agilizar el desarrollo de la interfaz, proporcionando una experiencia de usuario cohesiva y profesional sin necesidad de construir cada componente desde cero.

4.3.7 VisNetwork.Blazor

VisNetwork.Blazor (29) es una adaptación para Blazor de la reconocida librería JavaScript, Vis-network (30), especializada en la visualización de grafos. Este *port*, desarrollado por la comunidad, permite aprovechar gran parte de las funcionalidades de Vis-network directamente en aplicaciones Blazor, eliminando la necesidad de utilizar JavaScript. En el contexto del proyecto, VisNetwork.Blazor ha sido esencial para la representación gráfica del grafo de la historia, permitiendo una visualización clara y dinámica de su estructura y relaciones.

4.3.8 Dijkstra.NET

Dijkstra.NET (31) es una biblioteca desarrollada por la comunidad que implementa eficientemente dos algoritmos esenciales para el tratamiento de grafos: el algoritmo Dijkstra, destinado a encontrar el camino más corto entre nodos, y el algoritmo PageRank, utilizado generalmente para determinar la importancia relativa de los nodos dentro de un grafo. Dentro de este proyecto, Dijkstra.NET ha sido crucial para calcular y analizar algunas de las estadísticas presentadas, ofreciendo al usuario información valiosa durante el proceso de creación de historias.

4.3.9 Visual Studio

Visual Studio 2022, creado por Microsoft, es un entorno de desarrollo integrado (IDE) ampliamente reconocido por su eficacia y gama de herramientas destinadas a facilitar la programación y diseño de aplicaciones. Su interfaz intuitiva, conjuntamente con capacidades avanzadas de diagnóstico, simplifica la tarea de depuración y permite identificar problemas de manera más eficiente. La elección de este IDE para el proyecto se basa en varias razones: no solo es el entorno recomendado para trabajar con Blazor (32), sino que también proporciona funcionalidades destacadas, como el *hot reload*. Esta característica agiliza notablemente el proceso de desarrollo al permitir visualizar cambios en tiempo real, evitando la necesidad de reiniciar la aplicación y mejorando la experiencia de desarrollo.

4.3.10 Github

GitHub ha sido la plataforma seleccionada para el control de versiones de este proyecto. Su capacidad para gestionar las diferentes versiones del código y asegurar un desarrollo ordenado ha sido esencial. Con GitHub, no solo se ha mantenido un registro de las modificaciones, sino que también ha ofrecido la flexibilidad necesaria para trabajar desde distintos entornos sin inconvenientes.

4.3.11 Github Copilot

GitHub Copilot es una herramienta de inteligencia artificial desarrollada por GitHub y OpenAI que sugiere líneas o bloques de código y se adapta a tu estilo de desarrollo (33). Actúa como un asistente, ofreciendo soluciones en tiempo real mientras el desarrollador escribe.

Su capacidad para ofrecer soluciones rápidas ha facilitado y agilizado el proceso de desarrollo. Sus sugerencias a menudo han servido como base para adaptar soluciones específicas al proyecto, resultando en un código más eficiente.

5. Desarrollo de la solución propuesta

En este capítulo, nos adentraremos en los detalles del desarrollo de la solución, desvelando los desafíos, obstáculos y decisiones tomadas que han dado forma al resultado final. Finalmente, concluiremos con un ejemplo del aplicativo web resultante.

5.1 Teoría de grafos

Dada la naturaleza de nuestras historias interactivas, donde los capítulos están interconectados mediante opciones, es apropiado representar una historia como un grafo. Un grafo consiste en un conjunto de nodos unidos por arcos (34). En el contexto de nuestras historias, los capítulos actúan como nodos y las opciones interactivas como arcos. Dado que las opciones representan una forma de avanzar con una dirección definida entre capítulos, estamos en presencia de un grafo dirigido. Esta representación no solo es útil para visualizar gráficamente la historia, sino que también introduce la aplicación de la teoría de grafos.

La teoría de grafos se enfoca en el estudio de las propiedades de los grafos. Gracias a esto, es posible obtener medidas sobre la estructura y dinámica de la historia que pueden ser útiles para el autor al decidir cómo estructurar, expandir o modificar su narrativa. Este enfoque fue considerado durante el análisis, reflejado en el requisito funcional RF16, denominando a esta información como estadísticas de la historia.

Es esencial recordar que el usuario no debería requerir conocimientos técnicos para utilizar y comprender el aplicativo. Dado esto, y asumiendo que podrían no estar familiarizados con la teoría de grafos y sus medidas, hemos adaptado la terminología a un lenguaje más adecuado para el contexto narrativo. Además, se han proporcionado explicaciones detalladas para cada estadística, facilitando así su entendimiento. Cabe mencionar que, dado que la aplicación ha sido desarrollada en inglés, todos estos términos y descripciones están presentados en dicho idioma.

Partiendo del ejemplo sencillo de grafo de la ilustración 27, detallaremos las métricas de la teoría de grafos que se han incorporado:

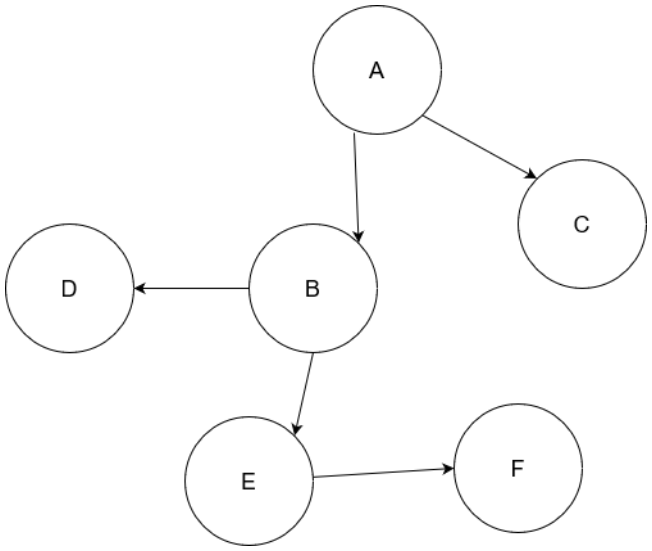

Ilustración 27. Ejemplo de grafo dirigido

- **Grado de entrada y salida de cada nodo:** En la teoría de grafos, el grado de un nodo hace referencia al número de aristas conectadas a él. En grafos dirigidos, se distingue entre el grado de entrada, que cuenta las aristas que llegan al nodo, y el grado de salida, que cuenta las aristas que parten de él.

  Tomando como referencia la ilustración 27, en la que se muestra un nodo A con dos conexiones salientes y un nodo B con una entrada desde A y dos salidas hacia otros nodos: el grado de salida de A es 2, mientras que el grado de entrada y salida de B son 1 y 2, respectivamente.

  En el contexto narrativo de nuestra aplicación, hemos renombrado el grado de entrada como caminos previos (*previous paths*) y lo definimos como el número de opciones de otros capítulos que conducen directamente a un capítulo. Por otro lado, el grado de salida lo hemos denominado caminos posibles (*possible paths*), refiriéndonos al número de opciones disponibles desde un capítulo para avanzar a otros capítulos.

- **Camino más largo:** El camino más largo de un grafo se refiere al mayor número de aristas consecutivas entre dos nodos sin repetir ningún nodo; a esto se le conoce como un camino simple. En el contexto de nuestra aplicación, únicamente consideramos los caminos que comienzan desde el capítulo inicial, que corresponde al primer capítulo creado. Siguiendo con el ejemplo mostrado anteriormente (ilustración 27), y tomando como partida el nodo A, el camino más largo sería A, B, E, F, con una longitud de 3 decisiones o aristas.

  Para el usuario de nuestra plataforma, hemos bautizado esta métrica como ruta más larga (*longest route*) y la hemos definido como la secuencia más extensa de decisiones que un lector puede tomar desde el comienzo hasta un final determinado de la narrativa, sin pasar dos veces por el mismo capítulo.

- **Caminos más cortos:** Cuando hablamos de un grafo, el camino más corto entre dos nodos es aquella ruta que posee la menor cantidad de aristas o decisiones. Para la implementación de nuestra aplicación, estos caminos se calculan desde el capítulo inicial, es decir, el primer capítulo creado, hasta cada uno de los finales. Se consideran finales aquellos capítulos que no poseen opciones de salida, es decir, tienen un grado de salida de 0.

  Siguiendo el grafo de la ilustración 27 que hemos presentado, identificamos 3 finales: C, D y F. De ellos, emergen 3 caminos más cortos con longitudes de 1, 2 y 3 respectivamente.

  Esta métrica ha sido etiquetada como rutas directas (*direct routes*) en nuestro sistema. La definimos como la mínima cantidad de decisiones que un lector puede tomar desde el inicio hasta un final determinado de la historia. Adicionalmente, se proporciona información sobre la ruta más larga, la más corta y la longitud media.

- **PageRank:** Originado por Larry Page y Sergey Brin, fundadores de Google, en 1996, el algoritmo PageRank evalúa la importancia de las páginas web basándose en cómo están interconectadas (35). La lógica es que las páginas importantes son aquellas enlazadas por otras páginas igualmente relevantes. Es un algoritmo iterativo que actualiza los valores de importancia hasta que estos convergen a un valor estable. Sin embargo, para adaptarlo a nuestro contexto de grafos no muy complejos, hemos decidido utilizar una versión simplificada que realiza solamente dos iteraciones.

En relación con nuestras narraciones interactivas, hemos renombrado este concepto como clasificación de capítulos (*chapter rank*). Esta métrica se refiere a cuán influyente o relevante es un capítulo dentro de la historia, sirviendo como una medida de la influencia o relevancia de un capítulo en el contexto general de la narrativa.

5.2 Problemas e implementaciones relevantes

A lo largo del desarrollo de cualquier proyecto de software, es inevitable encontrar obstáculos, retos técnicos y decisiones cruciales que influirán en el producto final. En este apartado, se abordarán los problemas más significativos que surgieron durante la creación de nuestro aplicativo, así como las implementaciones relevantes que destacaron por su complejidad o impacto en la funcionalidad general.

5.2.1 Algoritmos y medidas de grafos

La elección de C# como lenguaje de programación, a pesar de su popularidad creciente, presentó desafíos específicos en el ámbito del procesamiento de grafos. No es comúnmente utilizado para este propósito, y a diferencia de otros lenguajes como Python que cuenta con librerías robustas como NetworkX, C# carece de herramientas consolidadas para la manipulación y estudio de grafos. Esta falta nos llevó a utilizar múltiples librerías menores con funcionalidades parciales, y en ocasiones, a implementar manualmente ciertas funcionalidades.

Uno de los primeros desafíos fue la visualización de los grafos. Sin una herramienta nativa para C#, recurrimos a un *port* de la librería Vis.js, originalmente diseñada para JavaScript. La creación del grafo para esta librería consistió en recorrer los capítulos de nuestra instancia de la historia para crear los nodos y por cada capítulo recorrer sus opciones para crear los arcos. La ilustración 28 muestra este proceso:

```
var edges = new List<VisNetwork.Blazor.Models.Edge>();
var nodes = new List<Node>();

foreach (var chapter in TheStory.Chapters)
{
    nodes.Add(new Node(chapter.Id.ToString(), chapter.Title, 1, "circle"));

    foreach (var option in chapter.Options)
    {
        var edge = new VisNetwork.Blazor.Models.Edge(chapter.Id.ToString(),
        option.NextChapter.Id.ToString(),
        option.Text)
        {
            Arrows = new Arrows { To = new ArrowsOptions { Enabled = true } }
        };
        edges.Add(edge);
    }
}

_networkData = new NetworkData
{
    Edges = edges,
    Nodes = nodes
};
```

Ilustración 28. Código que convierte la historia en un grafo de la librería de Vis.js

El siguiente problema de este ámbito fue a la hora intentar calcular el camino más largo del grafo. A pesar de probar con diversas bibliotecas, todas resultaron insatisfactorias. Al final se optó por implementar un algoritmo DFS (*Depth-First Search*), para ello se utilizó una versión del algoritmo implementada en Python, y se llevó a cabo su traducción a C#. La ilustración 29 muestra el resultado de esta implementación:

```
private int DFS(string nodeId, HashSet<string> visited)
{
    if (visited.Contains(nodeId)) return 0;

    visited.Add(nodeId);

    int maxDepth = 0;
    foreach (var edge in _networkData.Edges.Where(e => e.From == nodeId))
    {
        maxDepth = Math.Max(maxDepth, 1 + DFS(edge.To, visited));
    }

    visited.Remove(nodeId);
    return maxDepth;
}
```

Ilustración 29. Algoritmo Depth-First Search

Finalmente, para calcular los caminos más cortos y el PageRank, utilizamos la librería Dijkstra.NET. Esta librería requirió la creación de un objeto grafo distinto, por lo que optamos por derivarlo del grafo de Vis.js `_networkData`. Además, para mantener una correspondencia coherente entre las dos representaciones y facilitar la manipulación de datos entre ellas, introducimos un diccionario. Esto nos permitió relacionar de manera efectiva ambos grafos. El siguiente fragmento (ilustración 30) muestra cómo se logró esta integración y la estructura del diccionario utilizado:

```
_graph = new Graph<int, string>();
_guidToIntegerMapping = new Dictionary<string, int>();
int uniqueId = 1;

foreach (var node in _networkData.Nodes)
{
    _guidToIntegerMapping[node.Id] = uniqueId;
    _graph.AddNode(uniqueId);
    uniqueId++;
}

foreach (var edge in _networkData.Edges)
{
    int fromNode = _guidToIntegerMapping[edge.From];
    int toNode = _guidToIntegerMapping[edge.To];

    _graph.Connect((uint)fromNode, (uint)toNode, 1, edge.Title);
}
```

Ilustración 30. Código que convierte el grafo de la librería Vis.js a un grafo de la librería Dijkstra.NET

5.2.2 Referencias circulares en la serialización

Una de las tareas más críticas en el desarrollo de nuestro aplicativo fue la necesidad de serializar las historias en formato JSON. Este proceso consiste en transformar nuestros objetos en memoria a una representación en cadena de caracteres en formato JSON, que es ampliamente usado para almacenar y transmitir datos en aplicaciones web.

Sin embargo, nos encontramos con un desafío técnico de particular interés: la estructura intrínseca de nuestra clase `Story`. Para entenderlo en detalle, consideremos la composición de esta clase. Una `Story` está compuesta por una lista de capítulos de tipo `Chapter`, y cada `Chapter` a su vez tiene una lista de opciones de tipo `Option`.

Cada `Option`, por su naturaleza interactiva, contiene una referencia a otro `Chapter`, indicando la dirección narrativa que toma la historia tras seleccionar esa opción.

Aquí es donde emerge el problema. Imaginemos que estamos serializando un `Chapter` en particular y llegamos a una de sus opciones. Al intentar serializar esta `Option`, nos damos cuenta de que dentro tiene una referencia a un capítulo. Si seguimos esta referencia y serializamos ese capítulo, volvemos a encontrar más opciones que podrían llevarnos a otros capítulos o, en algunos casos, incluso retroceder a capítulos anteriores. Este proceso puede continuar indefinidamente, creando un bucle de serialización.

Este fenómeno se conoce como referencia circular, y es un problema común al serializar estructuras de datos complejas. Las referencias circulares son un verdadero desafío porque pueden llevar a una serialización infinita, consumiendo recursos y eventualmente colapsando la aplicación.

Para abordar este problema, tuvimos que implementar clases auxiliares diseñadas específicamente para la serialización. Estas clases, `SerializedStory`, `SerializedChapter` y `SerializedOption`, reflejan la estructura de nuestras clases originales (`Story`, `Chapter` y `Option`), pero con modificaciones fundamentales para evitar la recursividad indeseada.

En lugar de incluir una referencia directa al siguiente capítulo en cada `Option`, hemos reestructurado la relación. La clase `SerializedOption` ahora contiene dos campos identificativos: `FromChapter` y `ToChapter`. Estos campos almacenan identificadores únicos de los capítulos de origen y destino, respectivamente, eliminando la necesidad de representar todo el capítulo y sus opciones recursivamente.

La clase `SerializedChapter` se centra únicamente en los detalles del capítulo, sin incluir las opciones directamente en su estructura. Y es en `SerializedStory` donde agrupamos las listas de capítulos y opciones por separado, dándonos una estructura clara y lineal.

Estas adaptaciones nos permiten tener una representación plana y manejable de la historia para la serialización. Al deserializar, utilizamos los identificadores de `FromChapter` y `ToChapter` para reconstruir las relaciones entre capítulos y opciones en memoria. Las siguientes ilustraciones 31 y 32 ilustran el proceso de serialización y deserialización de la historia.

```
public static string Serialize(Story story)
{
    var representation = new SerializedStory
    {
        Title = story.Title,
        Chapters = story.Chapters.Select(chapter => new SerializedChapter
        {
            Id = chapter.Id,
            Title = chapter.Title,
            Text = chapter.Text
        }).ToList(),
        Options = story.Chapters
            .SelectMany(chapter => chapter.Options
                .Select(option => new SerializedOption
                {
                    Text = option.Text,
                    FromChapter = chapter.Id.ToString(),
                    ToChapter = option.NextChapter.Id.ToString()
                }))
            .ToList()
    };
    var _jsonOptions = new JsonSerializerOptions
    {
        WriteIndented = true
    };

    return JsonSerializer.Serialize(representation, _jsonOptions);
}
```

Ilustración 31. Método de serialización de la historia

```
public static Story Deserialize(string json)
{
    var representation = JsonSerializer.Deserialize<SerializedStory>(json);

    var idToChapter = representation
        .Chapters.ToDictionary(chapter => chapter.Id, chapter => new Chapter(chapter.Id, chapter.Title)
    {
        Text = chapter.Text
    });

    foreach (var option in representation.Options)
    {
        Guid fromChapterId = Guid.Parse(option.FromChapter);
        Guid toChapterId = Guid.Parse(option.ToChapter);

        if (idToChapter.TryGetValue(fromChapterId, out var fromChapter)
        && idToChapter.TryGetValue(toChapterId, out var toChapter))
        {
            var newOption = new Option(option.Text, toChapter);
            fromChapter.AddOption(newOption);
        }
    }

    var story = new Story
    {
        Title = representation.Title,
        Chapters = idToChapter.Values.ToList()
    };

    return story;
}
```

Ilustración 32. Método de deserialización de la historia

5.2.3 Exportación a Ink

Dada la relevancia de exportar a Ink para la ejecución de historias en StreamINK, es esencial destacar cómo se implementó esta funcionalidad. El proceso se basa en transformar la estructura interna de nuestras historias en el formato de script de Ink, permitiendo así que nuestros relatos sean interpretados y ejecutados por StreamINK.

A pesar de que en el análisis de grafos se considera como capítulo inicial aquel que fue creado en primer lugar, durante la fase de exportación, creímos conveniente permitir al usuario seleccionar un punto de inicio diferente. Esta decisión añade flexibilidad en la presentación y exploración de la historia, permitiendo distintas visiones de la misma narrativa.

El lenguaje de *scripting* Ink, diseñado para narraciones interactivas, presenta una estructura particular (ver ilustración 33) que facilita la creación y navegación de historias. A continuación, se describe brevemente dicha estructura:

1. **Nudos**: Los nudos representan capítulos o secciones de la narración. Cada nudo se declara mediante tres signos de igualdad seguidos del identificador del capítulo y otros tres signos de igualdad. Posterior al encabezado, se escribe el contenido del capítulo.
2. **Desvíos**: Son redirecciones a otro capítulo. Permiten encadenar diferentes partes de la narración. Se declaran con una flecha seguida del identificador del capítulo al que se desea redirigir.
3. **Opciones**: Las elecciones de un capítulo que alteran el rumbo de la historia. Se declaran con un signo de más seguido del texto de la opción entre corchetes y la redirección al capítulo correspondiente.
4. **Finalización**: Para concluir la historia y señalar que no hay más contenido a seguir, se utiliza la redirección al final.

```
-> id_capitulo

=== id_capitulo ===

Texto del capítulo...

+ [Opción 1] -> id_siguiente_capitulo_1
+ [Opción 2] -> id_siguiente_capitulo_2

-> END
```

Ilustración 33. Ejemplo de formato Ink

El proceso de exportación en sí se basa en la función `GenerateInkContent()`. Esta función utiliza un `StringBuilder` para construir iterativamente el *script* de Ink, empezando con el capítulo seleccionado como inicio. Optamos por usar el identificador del capítulo en formato `Guid` en lugar de su título por dos razones principales: en primer lugar, evita posibles ambigüedades si el usuario crea varios capítulos con el mismo nombre. En segundo lugar, Ink tiene ciertas restricciones, como no permitir espacios para identificar los nudos. Al utilizar el método `ToString("N")` con el `Guid`, obtenemos una cadena única en formato hexadecimal de 128 bits, con 32 dígitos y sin separaciones. El código que sigue a continuación (ver ilustración 34) muestra la implementación del proceso de conversión.

```
public string GenerateInkContent(Chapter firstChapter)
{
    StringBuilder _stringBuilder = new StringBuilder();

    _stringBuilder.AppendLine($"-> {firstChapter.Id.ToString("N")}");
    _stringBuilder.AppendLine();
    foreach (Chapter chapter in Chapters)
    {
        _stringBuilder.AppendLine($"=== {chapter.Id.ToString("N")} ===");
        _stringBuilder.AppendLine(chapter.Text);
        _stringBuilder.AppendLine();
        if (chapter.Options.Count > 0)
        {
            foreach (Option option in chapter.Options)
            {
                _stringBuilder.AppendLine($"+ [{option.Text}] -> {option.NextChapter.Id.ToString("N")}");
            }
            _stringBuilder.AppendLine();
        }
        else
        {
            _stringBuilder.AppendLine("-> END");
        }
    }
    return _stringBuilder.ToString();
}
```

Ilustración 34. Método que genera un texto en formato Ink a partir de la historia

5.2.4 Patrón observador

El patrón de diseño observador es uno de los patrones más utilizados en el desarrollo de software. Este patrón define una relación de uno a muchos entre objetos, donde un objeto (el sujeto) mantiene una lista de objetos dependientes (observadores) que son notificados automáticamente sobre cualquier cambio de estado del objeto sujeto (36). En términos más simples, permite que varios objetos sean notificados cuando un objeto en particular cambia de estado.

En el contexto de este proyecto, hemos implementado el patrón observador en las clases `Story`, `Chapter` y `Option`. Esta implementación garantiza la coherencia y la actualización en tiempo real en toda la aplicación cuando ocurren modificaciones en alguno de estos componentes. Un ejemplo claro de esta aplicación es que cuando se modifica el título de un capítulo, se actualiza automáticamente y en tiempo real en la barra lateral de los capítulos.

Para lograr esta funcionalidad, hemos empleado la interfaz `INotifyPropertyChanged` en las clases mencionadas anteriormente. Esta interfaz, parte la plataforma .NET, está expresamente diseñada para realizar implementaciones del patrón observador.

Tomando como ejemplo la clase `Story` (ilustración 35), al implementar esta interfaz, se añade un evento del tipo `PropertyChangedEventHandler` denominado `PropertyChanged`. Junto a este evento, se implementa el método `OnPropertyChanged()`. Este método, al ser invocado, dispara el evento mencionado anteriormente. Así, cada vez que sucede un cambio en la historia, como el añadido de un nuevo capítulo o una modificación en el título, el método `OnPropertyChanged()` se invoca, notificando a todos los observadores suscritos.

```
public class Story : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        private string? _title;

        public string? Title
        {
            get {  return _title; }
            set
            {
                if (_title != value)
                {
                    _title = value;
                    OnPropertyChanged(nameof(Title));
                }
            }
        }

        private void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
```

Ilustración 35. Implementación del patrón observador en la clase Story

Podemos ver un ejemplo práctico en el componente de `Navbar` (ilustración 36), que conforma la barra de navegación superior de la aplicación. Al inicializar el componente, este se suscribe a los cambios en la historia, especificando que al ocurrir un cambio, se ejecute el método de Blazor `StateHasChanged()`. Este método notifica al componente de un cambio en su estado, lo que provoca que el componente se vuelva a renderizar. Así, si cambia el título de la historia, el nuevo título aparece automáticamente en la barra superior. Es crucial, para la gestión de recursos, desuscribirse de este evento cuando ya no es necesario. En nuestro caso, realizamos esta acción en el método `Dispose()`.

```
protected override void OnInitialized()
{
    TheStory.PropertyChanged += (sender, args) => StateHasChanged();
    base.OnInitialized();
}

public void Dispose()
{
    TheStory.PropertyChanged -= (sender, args) => StateHasChanged();
}
```

Ilustración 36. Código de la suscripción y desuscripción al patrón observador desde el componente Navbar

La adopción del patrón observador a través de la interfaz `INotifyPropertyChanged`, ha otorgado una robustez y una respuesta en tiempo real a la aplicación, garantizando una experiencia fluida y coherente para el usuario final.

5.3 Ejemplo y aplicativo final

En este apartado, presentaremos un relato de muestra que se alinea con el grafo de la ilustración 27, abordado previamente en el apartado 5.1. Este ejemplo servirá para demostrar las vistas resultantes del aplicativo web.

A - El comienzo enigmático

Despiertas en una habitación desconocida. A tu alrededor hay herramientas y artefactos extraños. En la puerta hay un símbolo grabado: una serpiente que se muerde la cola. A través de una ventana ves dos caminos: uno que se adentra en un oscuro bosque y otro que conduce a un castillo brillante.

- **Opción 1**: Te aventuras en el bosque. [Ir a B]
- **Opción 2**: Te diriges hacia el castillo brillante. [Ir a C]

B - Bosque de sombras

El bosque es denso y oscuro, pero puedes oír el canto de los pájaros y el crujir de las hojas bajo tus pies. A medida que avanzas, encuentras dos objetos: una llave oxidada y un cristal brillante.

- **Opción 1**: Tomas la llave y la usas en una puerta escondida entre los árboles. [Ir a D]
- **Opción 2**: Tomas el cristal brillante y sigues un camino iluminado por él. [Ir a E]

C - El castillo del destino

Al entrar al castillo, una voz etérea resuena a tu alrededor: "Has elegido el camino del destino. Aquí termina tu viaje". Las paredes del castillo comienzan a brillar, mostrándote imágenes de tu vida pasada y futura. Finalmente, la voz declara: "Has encontrado tu destino. Descansa ahora y encuentra la paz".
[Fin]

D - El secreto olvidado

La puerta chirría al abrirse, revelando una habitación repleta de objetos antiguos y libros polvorientos. En el centro hay un pedestal con un libro abierto. Al leerlo, descubres que contiene los secretos de la vida y la muerte. Sin embargo, al intentar salir, la puerta se cierra detrás de ti. Estás atrapado en esta sala con el conocimiento más grande del universo, pero sin la libertad de usarlo.
[Fin]

E - El refugio de cristal

Siguiendo el camino iluminado por el cristal, llegas a una cueva cuyas paredes están formadas por cristales brillantes. En el centro hay un estanque claro. Al sumergir el cristal en el agua, un portal se abre.

- **Opción 1**: Decides entrar al portal. [Ir a F]

F - La tierra desconocida

Apareces en un mundo completamente diferente. Las nubes son de colores brillantes, los árboles flotan y las criaturas se comunican a través de la música. Una entidad amigable se acerca y te dice: "Has traído el cristal. Eres el elegido para unir nuestros mundos". Al aceptar tu papel, te conviertes en el guardián de dos mundos, asegurando que siempre haya un puente entre ellos.
[Fin]

5.3.1 Interfaz página de inicio

La primera captura que presentamos corresponde a la interfaz de la página de inicio de nuestro aplicativo web. Como se aprecia en la ilustración 37, en esta página el usuario tiene a su disposición las distintas opciones relacionadas con la historia que se esté gestionando en ese momento. Es aquí donde se puede editar el título del relato, guardar la historia en formato JSON para su posterior uso, o incluso exportarla en formato Ink, adaptado para StreamINK.

En cuanto a elementos de diseño recurrentes en toda la plataforma, destaca la barra de navegación superior. Esta barra integra varias herramientas esenciales para el usuario: la posibilidad de iniciar una nueva historia, la opción de cargar una previamente guardada, accesos rápidos para guardar y exportar, así como botones que redirigen al buscador y a la sección de análisis de grafos y estadísticas.

Complementando la experiencia de navegación, contamos con la barra lateral dedicada exclusivamente a los capítulos. Esta barra, omnipresente en el aplicativo, enumera todos los capítulos creados por el usuario, permitiendo un acceso directo a cada uno de ellos. Además, facilita la creación de nuevos capítulos a través de un botón dedicado para tal propósito.

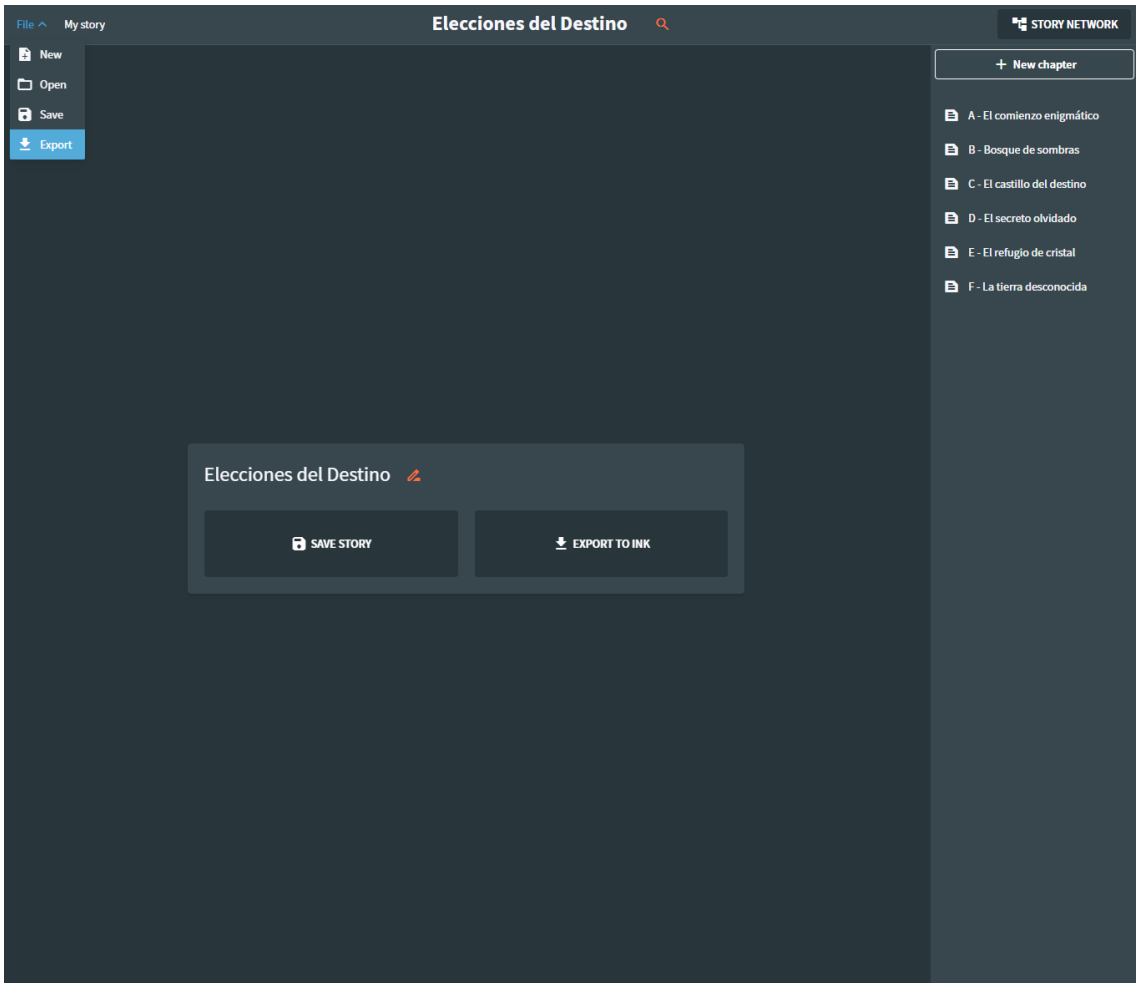

Ilustración 37. Página de inicio

5.3.2 Interfaz página de capítulo

La ilustración 38 muestra la interfaz de la página dedicada a un capítulo individual. En esta vista, se despliega todo el contenido asociado al capítulo en cuestión: desde su texto narrativo hasta las opciones interactivas. Los usuarios disponen de herramientas intuitivas para la edición, permitiéndoles modificar el título y el contenido narrativo con facilidad.

Las opciones se sitúan directamente debajo del texto principal en una disposición vertical. Al final de estas opciones, hay un botón que permite añadir nuevas opciones al capítulo. Cada opción cuenta con su propio conjunto de herramientas: un botón para eliminar la opción, un campo para editar su texto y otro para cambiar la redirección a otro capítulo.

Al seleccionar un capítulo, la barra lateral ofrece un botón adicional que permite eliminar ese capítulo específico, otorgando al usuario una gestión eficiente de sus capítulos.

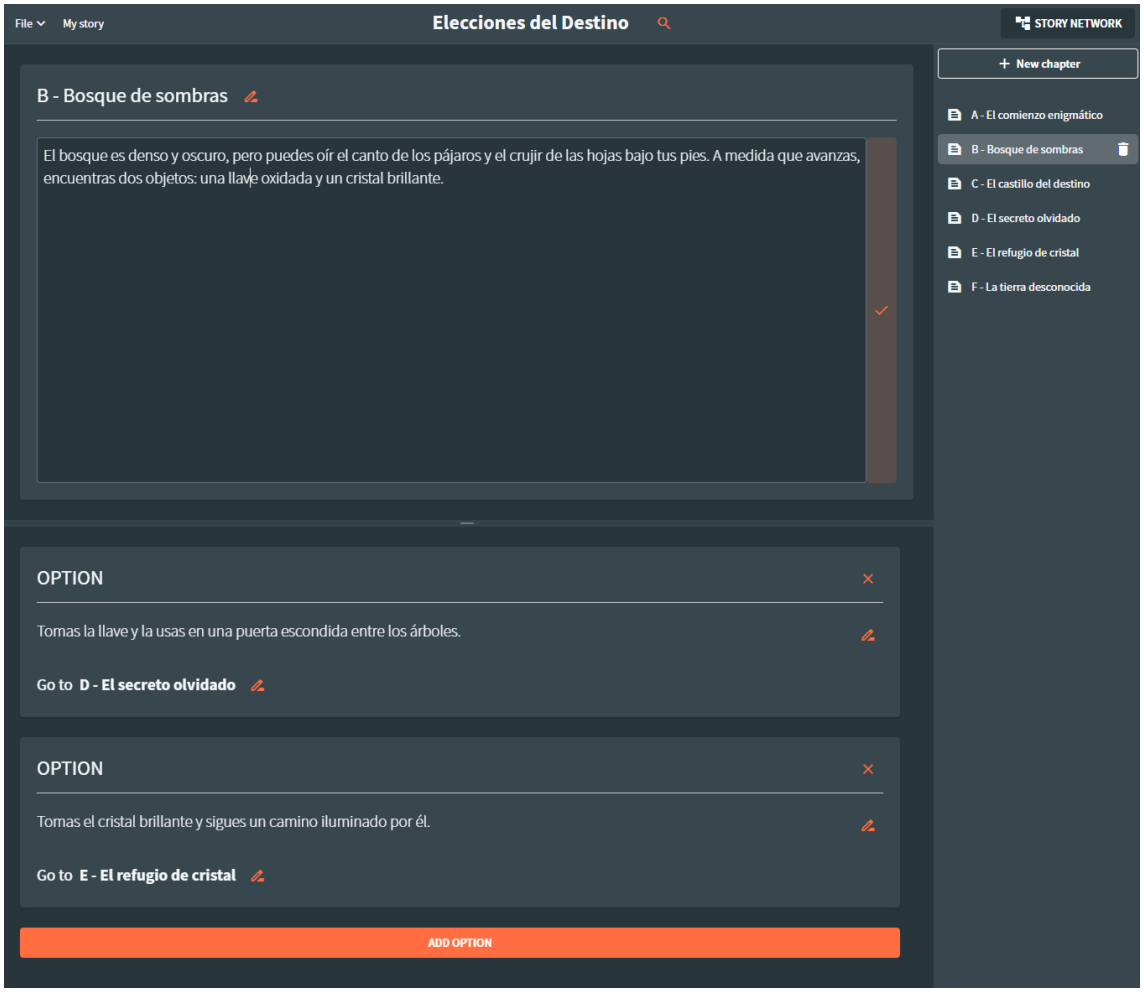

Ilustración 38. Página de capítulo

5.3.3 Interfaz página de grafo y estadísticas

Las ilustraciones 39 y 40 muestran la interfaz de la página dedicada a la visualización del grafo y las estadísticas de la historia. En la parte superior, se destaca el grafo representativo de la narrativa. Este grafo, dotado de físicas, no solo es interactivo, sino que también brinda la posibilidad de clicar en un nodo específico para ser redirigido a la página correspondiente al capítulo que representa.

En la sección inferior, se encuentran las estadísticas del relato, tal como se describieron en el apartado 5.1. Estas se presentan organizadas en tablas y, para proporcionar claridad al usuario, cuentan con cuadros de información ubicados debajo de cada estadística, que explican de forma concisa su significado. Debido a la extensiva información proporcionada, hemos dividido esta sección en dos capturas para su mejor visualización.

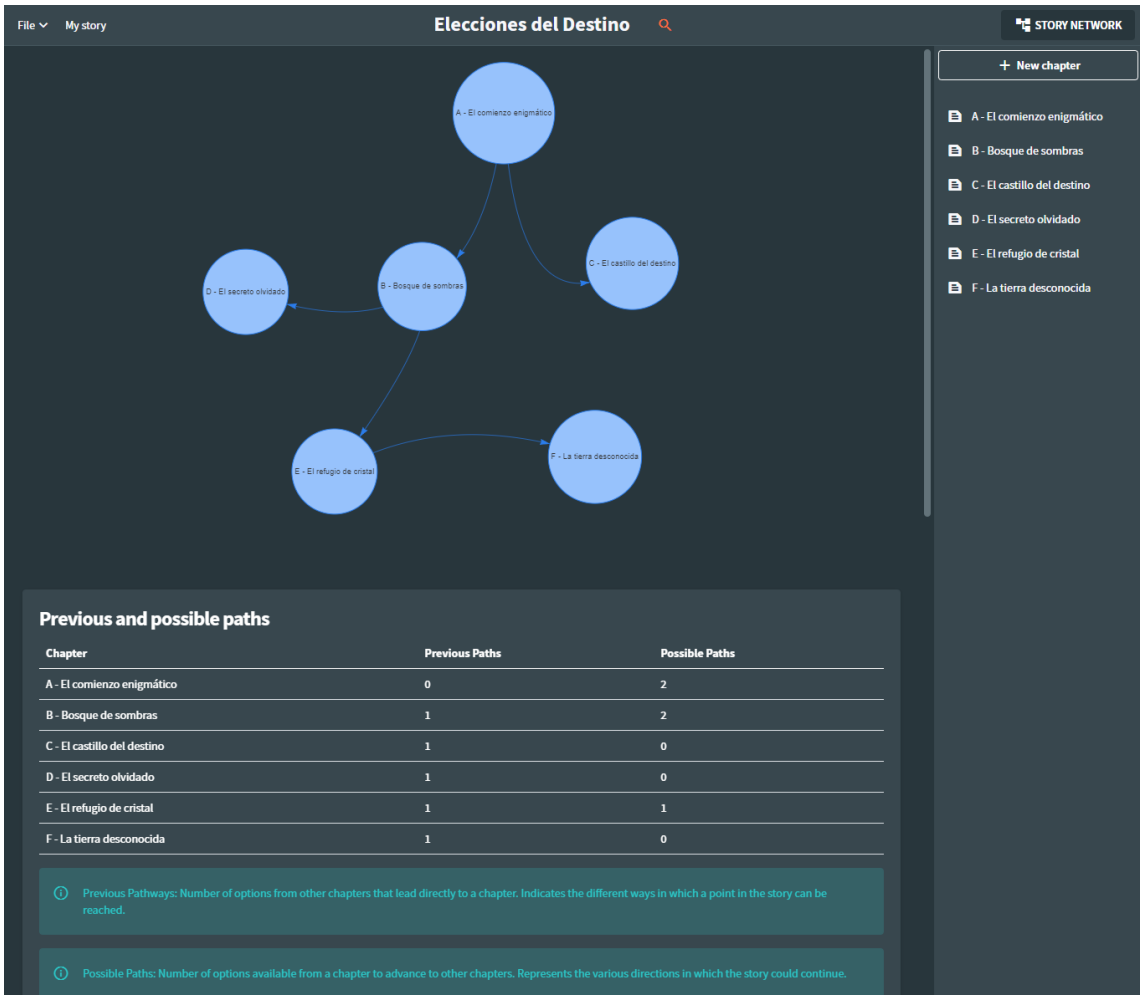

Ilustración 39. Página de grafos y estadísticas 1

Tal como se puede apreciar en las ilustraciones, el grafo de la historia de ejemplo coincide plenamente con el grafo discutido en el apartado 5.1 (ver ilustración 27). Esta coherencia se manifiesta en diversos aspectos como, por ejemplo, en los grados de entrada y salida: el nodo B muestra un grado de entrada de 1 y un grado de salida de 2. En cuanto al camino más largo identificado por el aplicativo, coincide con nuestra exposición al señalar una longitud de 3. En lo referente a los caminos más cortos, se evidencia que el trayecto más breve posee una longitud de 1, mientras que el más extenso es de 3, confirmando nuestra explicación anterior. Finalmente, es interesante destacar que, de acuerdo con el algoritmo de PageRank, el nodo de mayor relevancia es el nodo F.

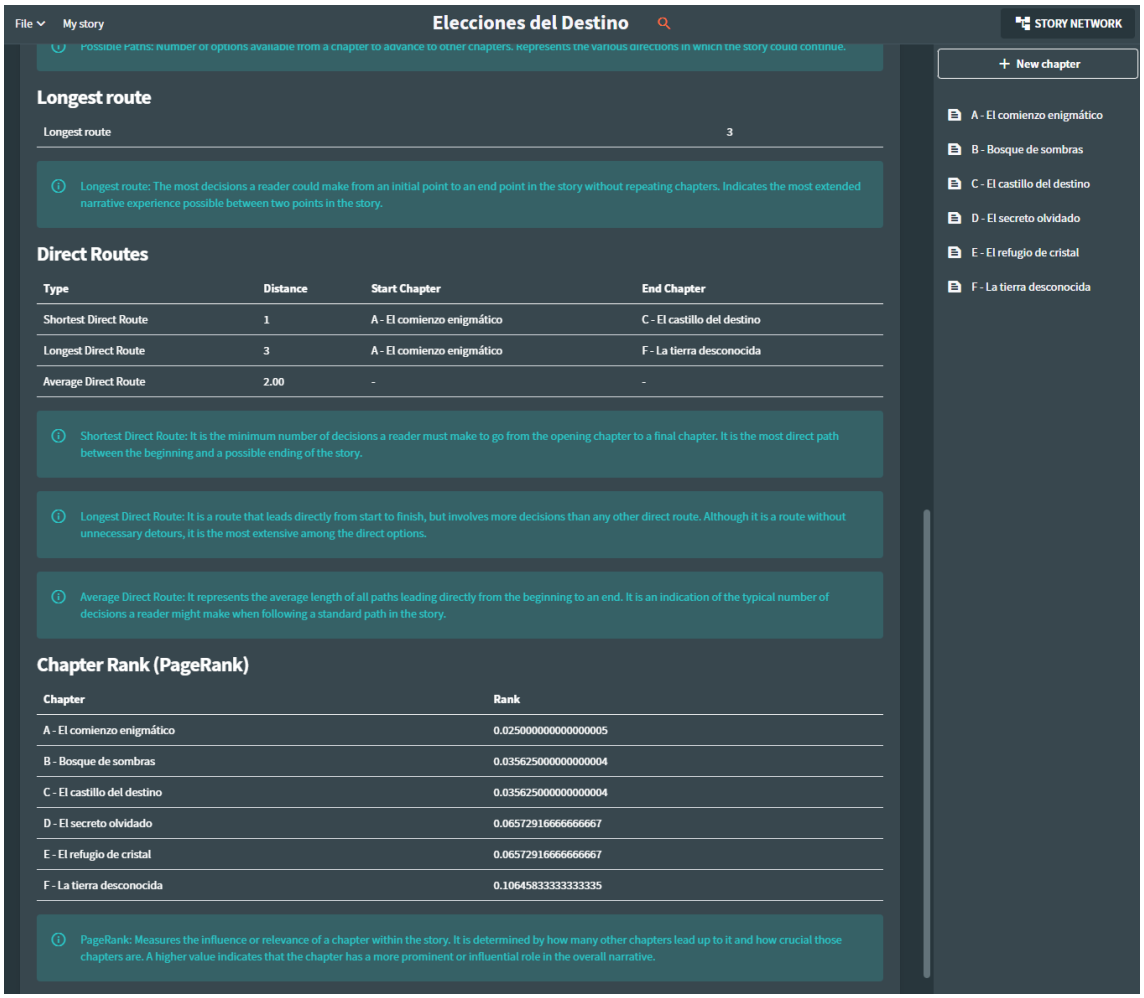

Ilustración 40. Página de grafos y estadísticas 2

6. Pruebas

Durante el desarrollo de nuestro aplicativo no hemos optado por un enfoque dirigido por pruebas. Sin embargo, para garantizar la calidad y fiabilidad del software, implementamos pruebas unitarias utilizando la herramienta MSUnit de Microsoft. Estas pruebas se centraron principalmente en las clases `Story` y `Chapter`, ya que contienen los métodos encargados de la manipulación de historias. Además, llevamos a cabo pruebas en las funciones estáticas encargadas de la serialización y deserialización. El objetivo principal de estas pruebas era asegurar que no se presentaran fallos al manipular las historias, tanto durante el proceso de guardado y carga en formato JSON, como durante la exportación al formato Ink, que posteriormente debe ser interpretado por StreamINK.

6.1 Pruebas de manipulación de la historia

En este apartado, nos centraremos en las pruebas efectuadas en las clases `Story` y `Chapter`, vitales para la manipulación de historias.

Clase `Story`:

1. **Método `AddChapter()`:** Se realizaron pruebas con el propósito de validar que un capítulo se añade adecuadamente a la historia. A través de este test, confirmamos que al invocar el método, el capítulo se incorpora sin conflictos y que se refleja correctamente en la estructura interna de la historia.

2. **Método `RemoveChapter()`:** Este método permite eliminar un capítulo específico de la historia. Las pruebas no solo se aseguraron de que el capítulo se eliminase, sino que también verificaron un comportamiento adicional. Si existen capítulos con opciones que redireccionan al capítulo que se está eliminando, dichas opciones también deben ser suprimidas para evitar referencias rotas.

3. **Método `GetChapterById()`:** El propósito de este método es recuperar un capítulo específico mediante su identificador único. Las pruebas (ver ilustración 41) se encargaron de confirmar que, al proporcionar un ID existente, el método retorna el capítulo correspondiente. Asimismo, si el ID no coincide con ningún capítulo existente, el método debe devolver `null`. Las pruebas confirmaron este comportamiento adecuadamente.

Clase `Chapter`:

1. **Método `AddOption()`:** Este método tiene la tarea de agregar opciones a un capítulo específico. Las pruebas unitarias se encargaron de corroborar que las opciones se añadiesen correctamente.

2. **Método `RemoveOption()`:** Similar al proceso de eliminación en la clase `Story`, este método elimina una opción específica de un capítulo. Las pruebas confirmaron que la opción se suprime adecuadamente.

```
[TestMethod]
0 references
public void GetChapterById_ReturnsCorrectChapter()
{
    var story = new Story();
    var id = Guid.NewGuid();
    var chapter = new Chapter(id, "Title");
    story.Chapters.Add(chapter);

    var result = story.GetChapterById(id);

    Assert.AreEqual(chapter, result);
}
```

Ilustración 41. Prueba unitaria del método GetChatpterById()

6.2 Pruebas de exportación y serialización

Es importante en aplicativos que manipulan datos garantizar que los procesos de serialización y exportación se ejecuten sin problemas. En este apartado detallaremos las pruebas efectuadas sobre estos procesos.

1. **Método `GenerateInkContent()`:** Es el encargado de transformar la historia al formato Ink. Las pruebas llevadas a cabo para este método tenían el objetivo de asegurar que la cadena de texto generada fuese la esperada basándose en la estructura de la historia creada. Con ello, garantizamos que cualquier historia puede ser transformada adecuadamente al formato Ink, permitiendo su posterior uso en StreamINK.

2. **Método `Serialize()`:** Este método es responsable de la transformación de la historia a un formato JSON, permitiendo su almacenamiento y transmisión. Las pruebas realizadas corroboraron que la historia se serializa correctamente, manteniendo la integridad de todos sus elementos y estructura. Además, fue esencial comprobar el comportamiento del método frente a escenarios límite, como verificar que al intentar serializar una historia vacía, el método no presente fallos.

3. **Método `Deserialize()`:** A la inversa del proceso anterior, el método de deserialización es encargado de transformar un JSON en una instancia de historia. Las pruebas llevadas a cabo confirmaron que la historia resultante es idéntica a la que se serializó originalmente. No obstante, es probable que el aplicativo en algún momento se encuentre con ficheros JSON que no sigan el formato esperado o estén vacíos. Por ello, era crucial asegurar que este método lanzase las excepciones pertinentes en tales escenarios, permitiendo a la aplicación capturarlas y notificar al usuario acerca del error.

6.3 Pruebas de ejecución en StreamINK

Además de las pruebas internas realizadas dentro del aplicativo, ha sido esencial asegurarse de que las exportaciones al formato Ink funcionan correctamente en su entorno objetivo. Por ello, hemos llevado a cabo pruebas ejecutando las historias exportadas directamente en el programa StreamINK (ver ilustración 42). Este proceso ha sido crucial para confirmar que las historias no sólo se exportan adecuadamente, sino que también se ejecutan y fluyen en StreamINK tal y como se esperaba. Estas pruebas han permitido corroborar la compatibilidad y coherencia de nuestra herramienta y StreamINK.

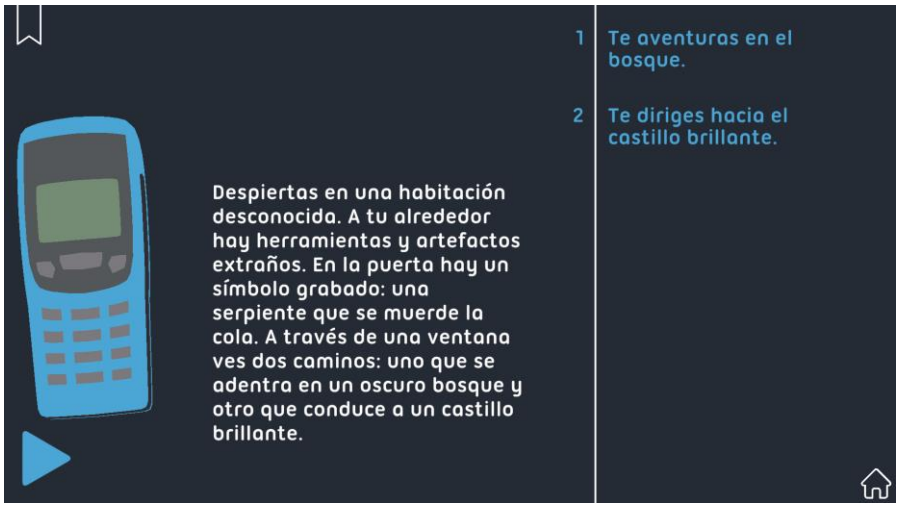

Ilustración 42. Prueba de ejecución en StreamINK

7. Conclusiones

El camino que ha marcado este proyecto estuvo determinado desde el principio por un objetivo principal: simplificar y facilitar el proceso de creación de historias interactivas. Acorde con este propósito, se establecieron tres objetivos específicos que definirían el éxito del proyecto:

O1 - Facilitar la creación, edición y gestión de historias y sus respectivos capítulos.

O2 - Presentar a los usuarios una representación gráfica de sus historias a través de una visualización en forma de grafo.

O3 - Asegurar una exportación compatible de las historias para su correcta ejecución en StreamINK.

Respecto al primer objetivo (O1), la plataforma ha demostrado ser una herramienta eficiente que facilita significativamente la creación, edición y gestión de historias y sus respectivos capítulos, ofreciendo a los usuarios una experiencia intuitiva y fluida al construir sus narrativas. En cuanto al segundo objetivo (O2), se ha logrado presentar a los usuarios una representación gráfica de sus historias, donde la visualización en forma de grafo no sólo les otorga una perspectiva clara de la estructura de su relato, sino que también enriquece su proceso creativo al identificar de manera visual las conexiones entre capítulos. Finalmente, relativo al tercer objetivo (O3), la funcionalidad de exportación ha sido probada y validada satisfactoriamente, asegurando que las historias son compatibles y pueden ser ejecutadas sin inconvenientes en StreamINK. En conjunto, estos logros reafirman que los objetivos específicos trazados al inicio del proyecto han sido alcanzados con éxito.

El resultado es un aplicativo web robusto y actualizado, desarrollado con tecnologías vanguardistas y siguiendo una metodología de trabajo estructurada. Esta metodología incluyó fases como el análisis exhaustivo de requisitos, la elaboración de diagramas y la construcción de prototipos de interfaces que orientaron el desarrollo posterior.

El proceso de desarrollo ha implicado una curva de aprendizaje significativa. Personalmente, tuve que familiarizarme con tecnologías como Blazor y sumergirme dentro del desarrollo web. Aunque mi formación en Ingeniería del Software me aportó conocimientos muy útiles dentro de la programación y diseño del software, no había tenido una exposición profunda en el desarrollo web, por lo que los conocimientos que he ido adquiriendo durante este trabajo me han aportado a mi crecimiento profesional.

Es inevitable en cualquier proyecto de estas características enfrentar obstáculos y desafíos. En este caso, tuve que superar inconvenientes tales como los problemas en la serialización y complejidades alrededor del trabajo con grafos. Sin embargo, con determinación y análisis, fui capaz de plantear soluciones viables y efectivas para cada desafío.

Finalmente, una reflexión vital que este proyecto ha reforzado es la importancia de una planificación meticulosa en las primeras etapas de desarrollo de software. Analizar el problema y diseñar soluciones es crucial para la correcta construcción del producto. Es en estas etapas donde se sientan las bases y se definen especificaciones que guiarán todo el proceso posterior. Con ello se evitarán contratiempos y sobre todo se garantizará que el producto final responda a las necesidades y objetivos planteados desde el comienzo.

7.1 Relación del trabajo desarrollado con los estudios cursados

El desarrollo de este proyecto es el resultado acumulativo de los conocimientos y habilidades que he adquirido durante mi tiempo en el grado. Elegí especializarme en la rama de Ingeniería del Software, que pone un particular énfasis en las metodologías y prácticas asociadas con el proceso de desarrollo de software.

Una de las materias fundamentales en este proyecto fue **Diseño de Software**. Esta asignatura no solo me permitió diseñar la arquitectura del sistema de manera coherente y eficiente, sino que también me introdujo a conceptos vitales como los patrones de diseño y las técnicas para realizar pruebas unitarias. Estos aprendizajes resultaron ser de gran valor durante la fase de implementación.

Por otro lado, las asignaturas de **Matemática Discreta** y **Estructuras de Datos y Algoritmos** jugaron un papel central en las características basadas en la teoría de grafos del aplicativo web. También fueron necesarias para la implementación de algoritmos de búsqueda, tales como el DFS.

El diseño de interfaces centradas en el usuario fue posible en gran medida gracias a la asignatura **Interfaces Persona Computador**. Los principios de esta materia me guiaron tanto en la fase de prototipado como en la creación de la versión final de la interfaz.

En cuanto a la fase de especificación de requisitos, **Análisis y Especificación de Requisitos** fue de suma importancia. A través de ella, pude abordar esta etapa con un enfoque metódico y detallado.

Finalmente, tanto la asignatura de **Ingeniería del Software** como la de **Proyecto de Ingeniería de Software** fueron mis primeras experiencias en la realización de proyectos completos de software. Los desafíos y oportunidades que surgieron durante los proyectos de estas asignaturas me aportaron las bases para afrontar con confianza y profesionalismo este trabajo de fin de grado.

En conclusión, completar el grado en Ingeniería Informática me ha dotado de las herramientas y el entendimiento crucial para emprender proyectos de software de manera eficiente.

7.2 Trabajos futuros

El proyecto actual, aunque completo en su propósito inicial, ofrece un amplio margen para la incorporación de mejoras y nuevas funcionalidades en futuras actualizaciones.

Un aspecto a considerar es la integración de un sistema de sugerencias en la redacción. Utilizando modelos de lenguaje como GPT-4, los usuarios podrían recibir recomendaciones mientras escriben, facilitando así la construcción de historias más ricas y ayudando a superar posibles bloqueos en la escritura.

En adición, la ampliación del soporte lingüístico de la herramienta para incluir otros idiomas además del inglés mejoraría la accesibilidad y comprensión de la interfaz para un público más amplio y diverso, especialmente para aquellos que no son hablantes nativos de inglés.

Otra mejora potencial se relaciona con la personalización de la interfaz. Proporcionar herramientas que permitan a los usuarios adaptar el diseño y funcionalidades del software a sus propias preferencias podría mejorar la experiencia general de uso, haciéndola más amigable y adaptada a las necesidades individuales.

En definitiva, el proyecto tiene potencial para crecer y adaptarse a las necesidades cambiantes de los usuarios, y estas sugerencias podrían ser un buen punto de partida para futuras actualizaciones.

8. Referencias

1. **Martínez, Elena.** Los librojuegos: una magnífica forma de involucrar al lector. *Lecturalia.* [En línea] 11 de noviembre de 2020. https://www.lecturalia.com/blog/2020/11/11/los-librojuegos-una-magnifica-forma-de-involucrar-al-lector/.

2. **Brennan, Marie.** *Dice Tales: Essays on Roleplaying Games and Storytelling.* s.l. : Book View Cafe, 2018.

3. **Peterson, Jon.** *Playing at the World: A History of Simulating Wars, People and Fantastic Adventures, from Chess to Role-Playing Games.* s.l. : Unreason Press , 2012. 978-0615642048.

4. **Tones, John.** En el principio fue la aventura conversacional. *Xataka.* [En línea] 30 de junio de 2015. https://www.xataka.com/literatura-comics-y-juegos/en-el-principio-fue-la-aventura-conversacional.

5. **Rollings, Andrew y Adams, Ernest.** *Andrew Rollings and Ernest Adams on Game Design.* s.l. : New Riders Pub, 2003. 1592730019.

6. **Cobbett, Richard.** The history of RPGs . *PC Gamer.* [En línea] 19 de mayo de 2017. https://www.pcgamer.com/the-complete-history-of-rpgs/.

7. **Wikipedia, Colaboradores.** Interactive film. *Wikipedia.* [En línea] https://en.wikipedia.org/wiki/Interactive_film.

8. **Durán, Ciro.** Creando juegos de aventura con Adventure Game Studio. *El Chigüire Literario.* [En línea] 10 de abril de 2007. https://www.elchiguireliterario.com/2007/04/10/creando-juegos-de-aventura-con-adventure-game-studio-i/#more-171.

9. **Visionaire Studio.** An Introduction to Visionaire Studio: Adventure Game Engine. *Visionaire Studio Wiki.* [En línea] https://wiki.visionaire-tracker.net/wiki/An_Introduction_to_Visionaire_Studio:_Adventure_Game_Engine.

10. **Brodkin, Jon.** How Unity3D Became a Game-Development Beast. *Dice.* [En línea] 3 de junio de 2013. https://www.dice.com/career-advice/how-unity3d-become-a-game-development-beast.

11. **Wikipedia, Colaboradores.** Unreal Engine. *Wikipedia.* [En línea] https://en.wikipedia.org/wiki/Unreal_Engine.

12. **GameMaker.** Introduction To GameMaker. *GameMaker Manual.* [En línea] https://manual.yoyogames.com/#t=Introduction%2FIntroduction_To_GameMaker_Studio_2.htm.

13. **Inkle.** Writing with ink. *GitHub.* [En línea] https://github.com/inkle/ink/blob/master/Documentation/WritingWithInk.md.

14. —. Inky. *GitHub.* [En línea] https://github.com/inkle/inky.

15. —. Inklewriter. *Inklestudios.* [En línea] https://www.inklestudios.com/inklewriter/.

16. —. Inklewriter JSON to ink converter. *Inklestudios.* [En línea] https://www.inklestudios.com/inklewriter/to-ink/.

17. **Twine.** Twine. *Twinery.* [En línea] http://twinery.org.

18. —. Editing Story JavaScript and CSS. *Twinery.* [En línea] http://twinery.org/reference/en/editing-stories/js-and-css.html.

19. **Petit, Carolyn.** Power to the People: The Text Adventures of Twine. *GameSpot.* [En línea] 21 de enero de 2013. https://www.gamespot.com/articles/power-to-the-people-the-text-adventures-of-twine/1100-6402665/.

20. **Reynolds, Matt.** The inside story of Bandersnatch, the weirdest Black Mirror tale yet. *Wired.* [En línea] 28 de diciembre de 2018. https://www.wired.co.uk/article/bandersnatch-black-mirror-episode-explained.

21. **Andreu, Abraham.** El liderazgo de Twitch acerca a las marcas a la plataforma: Ibai, AuronPlay y otros casos de éxito para conectar con las nuevas audiencias. *Business Insider.* [En línea] 14 de octubre de 2022. https://www.businessinsider.es/liderazgo-twitch-atractivo-streamers-marcas-1138697.

22. **Tome, Rothio.** StreamINK. *Itch.io.* [En línea] https://rothiotome.itch.io/streamink.

23. **Skeet, Jon.** *C# in Depth.* s.l. : Manning Publications, 2008. 1617294535.

24. **PYPL.** PYPL PopularitY of Programming Language. *PYPL.* [En línea] https://pypl.github.io/PYPL.html.

25. **Wright, Toi B.** *Blazor WebAssembly by Example: A project-based guide to building web apps with .NET, Blazor WebAssembly, and C#.* s.l. : Packt Publishing, 2021. 978-1800567511.

26. **Mozilla.** HTML basics. *MDN Web Docs.* [En línea] https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics.

27. —. CSS basics. *MDN Web Docs.* [En línea] https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics.

28. **Ramel, David.** Radzen Open Sources 60+ Blazor Components. *Visual Studio Magazine.* [En línea] 21 de enero de 2021. https://visualstudiomagazine.com/articles/2021/01/21/radzen-open-source.aspx.

29. **Kroknes, Stian.** VisNetwork.Blazor. *GitHub.* [En línea] https://github.com/stiankroknes/VisNetwork.Blazor.

30. **Vis.js.** vis-network. *GitHub.* [En línea] https://github.com/visjs/vis-network.

31. **Mazurek, Mateusz.** Dijkstra.NET. *GitHub.* [En línea] https://github.com/matiii/Dijkstra.NET.

32. **Microsoft.** Blazor. *.NET.* [En línea] https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor.

33. **GitHub.** GitHub Copilot. *GitHub.* [En línea] https://github.com/features/copilot.

34. **Trudeau, Richard J.** *Introduction to Graph Theory.* s.l. : Dover Publications, 1993. 0486678709.

35. **Sullivan, Danny.** What Is Google PageRank? A Guide For Searchers & Webmasters. *Search Engine Land.* [En línea] 26 de abril de 2007.

https://searchengineland.com/what-is-google-pagerank-a-guide-for-searchers-webmasters-11068.

36. **Gamma, Erich, y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1994. 0-201-63361-2.

9. Anexo: ODS

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Tabla 2. Objetivos de Desarrollo Sostenibles

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**				**X**
ODS 4. **Educación de calidad.**		**X**		
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**				**X**
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**				**X**
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Las historias interactivas, como las que este proyecto permite crear, tienen un potencial educativo significativo. A través de la interactividad, se pueden diseñar experiencias de aprendizaje más envolventes y personalizadas. Los estudiantes ya no son meros receptores de información, sino que se convierten en participantes activos en su propio proceso de aprendizaje, tomando decisiones, explorando escenarios y recibiendo retroalimentación inmediata basada en sus elecciones.

Además, este tipo de herramientas digitales son especialmente útiles para adaptarse a diferentes estilos de aprendizaje. Mientras que algunos estudiantes pueden beneficiarse de la lectura o la escucha, otros pueden encontrar un mayor entendimiento y retención a través de la interacción y la toma de decisiones. En este sentido, las historias interactivas pueden ayudar a nivelar el campo de juego, ofreciendo múltiples caminos de aprendizaje y permitiendo a los estudiantes encontrar el método que mejor se adapte a sus necesidades individuales.

Por otra parte, al tratarse de un aplicativo web, el proyecto elimina barreras geográficas, permitiendo que cualquier persona, independientemente de su ubicación, pueda acceder y beneficiarse de los contenidos interactivos creados. Este aspecto es especialmente relevante en áreas rurales o en países en desarrollo donde el acceso a materiales educativos de calidad puede ser limitado.

Por último, al fomentar la creación de historias interactivas, estamos potenciando habilidades cruciales del siglo XXI, como el pensamiento crítico, la resolución de problemas y la creatividad. Las personas que utilizan la herramienta no sólo consumen contenido, sino que también se convierten en creadores, diseñadores y narradores, roles que son esenciales en una economía basada en el conocimiento.