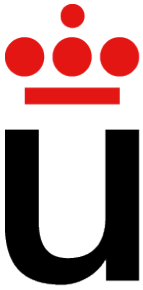

Universidad
Rey Juan Carlos

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

Curso Académico 2022/2023

Trabajo Fin de Grado

HERRAMIENTA DEDOS-EDITOR WEB

Autor : Alejandro García-Gasco Pérez

Tutor : David Roldán Álvarez

Trabajo Fin de Grado

Adaptación de la aplicación de escritorio DEDOS-Editor al entorno web.

Autor : Alejandro García-Gasco Pérez

Tutor : David Roldán Álvarez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 202X, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 202X

Dedicado a

mi madre, mi abuela

y en especial a mi Tía Beatríz.

II

Agradecimientos

Yo no debería estar entregando este trabajo ahora. Pertenezco a una promoción muy anterior a la actual, pues empecé la carrera hace más de diez años, allá por el 2008. Sin embargo, a falta de una asignatura y la redacción de la memoria de un TFG que ya tenía desarrollado, empecé a hacer prácticas en una pequeña startup como desarrollador y los derroteros de la vida me llevaron a abandonar mis obligaciones académicas en favor de mi carrera profesional.

Siempre fue un constante peso sobre mis hombros la sensación de haber dejado algo inacabado cuando ya estaba tocando el final con la yema de los dedos. Un vergonzoso secreto que ni mis más íntimos amigos ni mi madre sacaban a la luz, pues me incomodaba enormemente.

Y habría seguido así para siempre, hasta conseguir caer en el olvido, si mi tía no hubiera roto ese silencio cada vez que me veía, como un Pepito Grillo, preguntándome por esa maldita asignatura que me faltaba, manteniendo siempre esa mancha en mi mente. Me enfadaba con ella cuando en realidad estaba enfadado conmigo mismo.

No sin afrontar muchas dificultades y barreras mentales, he conseguido tras más de seis años alejado de la facultad, completar los créditos para graduarme. Sin embargo, mi tía ya no está aquí para darle la noticia. Falleció durante esta Navidad, de un cáncer tan inesperado como agresivo.

Ojalá pudiera llamarla y que me volviese a preguntar como hacia siempre. Menuda sorpresa se iba a llevar.

Este TFG se lo dedico a ella, a mi tía Beatriz.

Resumen

El proyecto DEDOS-Editor es una herramienta software de escritorio para el desarrollo de actividades educativas que busca gamificar la enseñanza y fomentar el uso de la tecnología en la educación infantil, primaria y estudiantes con necesidades especiales.

Forma parte de las denominadas herramientas de autor, software accesible que permiten la creación de contenido digital sin conocimientos técnicos previos por parte del usuario. Con una interfaz intuitiva, DEDOS-Editor permite el diseño de actividades y juegos orientados a la enseñanza. Los docentes pueden agregar texto e imágenes creando actividades interactivas.

El objetivo principal del proyecto es adaptar DEDOS-Editor a una plataforma web, aprovechando las ventajas de accesibilidad, colaboración, actualización y disponibilidad que ofrece. Esta adaptación permitirá a los docentes acceder a la herramienta desde cualquier navegador y dispositivo, facilitando la creación, edición y descarga de proyectos educativos.

En el presente documento se expondrán las tecnologías del *stack* tecnológico *MERN* (MongoDB, Express, React y Node.js) empleadas para el desarrollo de la aplicación web con el objetivo de construir una interfaz de usuario atractiva y segura, garantizando la usabilidad y el rendimiento óptimo de la herramienta.

Se abordarán los aspectos más destacables del desarrollo de la herramienta tanto en la parte frontal como en el lado del servidor, así como la descripción detallada de un caso de uso en el que se expondrán las principales funcionalidades de la solución DEDOS-Editor Web.

La adaptación de DEDOS-Editor a la plataforma web se presenta como un proyecto ejemplar para el estudio y análisis de tecnologías y técnicas web, proporcionando un amplio abanico de aspectos clave en su desarrollo.

Summary

The DEDOS-Editor project is a desktop software tool for developing educational activities that aims to gamify teaching and promote the use of technology in early childhood, primary education, and students with special needs.

It is part of the so-called authoring tools, accessible software that allows the creation of digital content without prior technical knowledge. With an intuitive interface, DEDOS-Editor allows the design of activities and games focused on teaching. Teachers can add text and images to create interactive activities.

The main objective of the project is to adapt DEDOS-Editor to a web platform, taking advantage of the accessibility, collaboration, update, and availability capabilities that it offers. This adaptation will allow teachers to access the tool from any browser or device, facilitating the creation, editing, and downloading of educational projects.

This document will present the technologies of the MERN technology stack (MongoDB, Express, React, and Node.js) used for the development of the web application, aiming to build an attractive and secure user interface, ensuring usability and optimal performance of the tool.

The most notable aspects of the tool's development will be addressed, both on the front-end and the server-side, as well as a detailed description of a use case that will expose the main functionalities of the DEDOS-Editor Web solution.

The adaptation of DEDOS-Editor to the web platform is presented as an exemplary project for the study and analysis of web technologies and techniques, providing a wide range of key aspects in its development.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este primer capítulo se expone la motivación principal que impulsó el desarrollo de este trabajo de fin de grado. Después de introducir las herramientas de autor y discutir los antecedentes generales del proyecto, se presentan las razones principales por las que se ha decidido llevar a cabo este desarrollo. Por último, se describe la estructura de este documento y se muestran los repositorios donde se encuentra el código del proyecto.

1.1. Antecedentes del proyecto

La herramienta DEDOS-Editor es una herramienta de autor que presenta un editor gráfico de escritorio que permite generar diferentes ejercicios orientados a la gamificación de la enseñanza para que puedan ser posteriormente ejecutados en dispositivos electrónicos como tabletas o pizarras electrónicas. Los alumnos podrán interactuar con estos dispositivos resolviendo las actividades programadas previamente por el educador a través del editor [4].

Estas actividades se basan en la creación de diferentes tarjetas o *tokens* a los que se asocian diferentes objetivos. Estos objetivos son acciones que el alumno puede realizar con ellas, como por ejemplo, seleccionar una tarjeta completando una tarea tipo test, relacionar dos tarjetas emparejadas u ordenarlas mediante contadores, etc. Estas acciones se pueden combinar para realizar actividades que contengan textos, imágenes y problemas matemáticos.

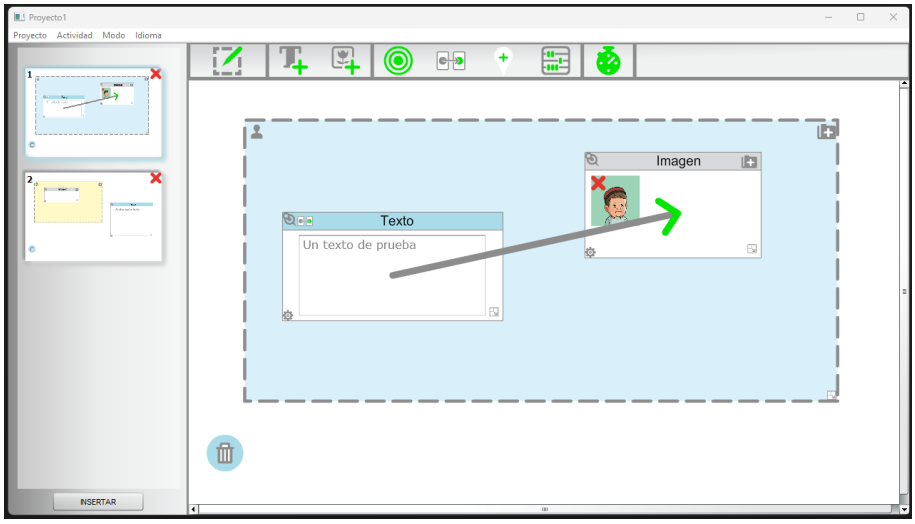

Figura 1.1: DEDOS-Editor en versión de escritorio.

Los objetivos principales de la herramienta fueron evaluar cómo ésta interfaz gráfica podría afectar a la curva de aprendizaje de los futuros docentes y estudiar los beneficios de incluir la tecnología en la educación infantil, primaria y de necesidades especiales.

Se destaca la importancia de reconvertir a los a los profesores en diseñadores de contenido digital y la adopción de la tecnología táctil para mejorar la experiencia de los estudiantes con habilidades motoras limitadas.

Además, se buscaba analizar el impacto del uso de las TIC en el aprendizaje y se realizaron experimentos en educación infantil, primaria y con estudiantes con discapacidades intelectuales y trastorno del espectro autista [5].

La motivación principal de este trabajo de fin de grado es la adaptación de la herramienta DEDOS-Editor a una plataforma web, actualizando así las tecnologías utilizadas y mejorando su usabilidad. De esta forma no será necesaria la instalación local de la herramienta y se suprimirán las limitaciones de disponibilidad y de compatibilidad con diferentes sistemas operativos o dispositivos. Los usuarios podrán disponer siempre de la ultima versión de la herramienta, así como de acceso instantáneo a sus proyectos en cualquier momento y cualquier lugar.

1.2. Herramientas de autor

Las herramientas de autor son aplicaciones informáticas diseñadas para facilitar la creación de contenido interactivo, como cursos en línea, presentaciones multimedia, simulaciones y apli-

caciones educativas. Estas herramientas permiten a los usuarios crear contenido sin necesidad de tener habilidades avanzadas de programación o diseño [3].

Una de las principales ventajas de las herramientas de autor es su facilidad de uso. Estas herramientas suelen ser intuitivas y amigables, lo que permite a cualquier persona crear contenido interactivo sin necesidad de conocimientos técnicos especializados. Además, ofrecen plantillas predefinidas y elementos reutilizables, lo que acelera el proceso de creación y evita tener que empezar desde cero.

Otra ventaja importante es la posibilidad de crear contenido interactivo. Las herramientas de autor permiten incorporar elementos interactivos, como cuestionarios, evaluaciones y actividades prácticas, lo que mejora la participación y el compromiso de los usuarios con el material. Esto ayuda a crear una experiencia de aprendizaje más dinámica y atractiva.

Además, las herramientas de autor simplifican el proceso de actualización y mantenimiento del contenido. Si se necesita realizar cambios en el material, estas herramientas suelen facilitar las actualizaciones y modificaciones sin tener que empezar de nuevo. Esto permite mantener el contenido actualizado y relevante con mayor facilidad [7].

Sin embargo, las herramientas de autor presentan limitaciones de personalización. Aunque ofrecen plantillas y elementos prediseñados, a veces pueden limitar la personalización y el diseño creativo avanzado. Esto puede resultar en una apariencia más estándar y menos diferenciada del contenido creado.

También hay que tener en cuenta que el uso de una herramienta de autor puede implicar cierta dependencia de la aplicación. Esto significa que el contenido creado puede estar limitado al modelo de datos o formato de archivo propietario de esa herramienta en particular. Esto puede dificultar la transferencia o migración a otras plataformas en el futuro por incompatibilidad de formatos.

Existen otros casos de éxito dentro del ámbito nacional de herramientas de autor similares, entre ellas destacan:

Ardora es un programa desarrollado por la iniciativa individual de José Manuel Bouzán. Se puede descargar y utilizar de manera gratuita, según se indica en su web, siempre y cuando sea usado de forma personal, sin carácter lucrativo y con fines estrictamente educativos.

Constructor es un programa desarrollado por la Consejería de Educación y Cultura de la Junta de Extremadura. Se distribuye bajo licencia *Creative Commons*.

JClic está desarrollado por la Xarxa Telemàtica Educativa de Cataluña. Se descarga de manera gratuita y es utilizado por educadores de todo el mundo. Tiene la ventaja además de que dispone de una amplia base de datos de actividades en la que se recoge un gran número de actividades de diferentes materias y niveles.

Cada uno de estos tres programas de autor nos ofrece un interesante repertorio de plantillas a partir de las cuales se puede elaborar diferentes tipos de actividades [6].

1.3. Tecnologías web

En la actualidad, las tecnologías web desempeñan un papel crucial en diversos ámbitos, incluyendo la educación, los negocios y la comunicación. La adopción y adaptación de herramientas de escritorio a plataformas web ha adquirido una gran relevancia debido a las ventajas que ofrece en términos de accesibilidad, colaboración, actualización y disponibilidad.

Una de las principales ventajas de adaptar una herramienta de escritorio a una plataforma web es la disponibilidad. Las aplicaciones web permiten acceder a las herramientas y recursos desde cualquier dispositivo con conexión a Internet, lo que brinda flexibilidad y elimina la necesidad de instalaciones locales. Además, las aplicaciones web son compatibles con diferentes sistemas operativos, lo que facilita el acceso para usuarios con diferentes preferencias y equipos. Esto resulta especialmente valioso en entornos educativos donde los estudiantes pueden acceder a contenido y realizar actividades de aprendizaje fuera del aula, y los docentes pueden generar los contenidos en cualquier momento y desde cualquier lugar y alojarlos en la nube con total disponibilidad [2].

La colaboración es otra ventaja significativa de las herramientas web. Al adaptar una herramienta de escritorio a una plataforma web, se facilita el trabajo colaborativo y en tiempo real. Los usuarios pueden compartir y editar proyectos y documentos simultáneamente, lo que mejora la comunicación y la productividad en entornos de trabajo y educativos. La capacidad de colaborar de manera efectiva en tiempo real fomenta el intercambio de ideas y el trabajo en equipo, lo que puede conducir a resultados más eficientes y de mayor calidad [1].

La adaptación de una herramienta de escritorio a una plataforma web también brinda ventajas en términos de actualización y mantenimiento. Las actualizaciones de software y correcciones de errores pueden implementarse fácilmente en una plataforma web centralizada, sin la

necesidad de actualizar individualmente las instalaciones en los dispositivos de los usuarios. Esto garantiza que todos los usuarios estén utilizando la versión más reciente de la herramienta y se beneficien de mejoras continuas sin interrupciones.

1.4. Estructura de la memoria

El presente documento se divide en seis secciones principales: la introducción, los objetivos del proyecto, tecnologías empleadas, el diseño e implementación, experimentos y validación y las conclusiones finales. Además, al final del documento se han añadido una serie de apéndices con información de interés.

A continuación se describen brevemente cada una de estas secciones:

- Capítulo 1 - Introducción: En este capítulo se explica la motivación principal por la que se ha decido realizar éste proyecto, así como una breve descripción del estado del arte y la estructura de la memoria.
- Capítulo 2 - Objetivos: En este capítulo se hace referencia a los objetivos generales del proyecto, concretando también algunos aspectos más específicos de estos y una planificación temporal de los desarrollos a seguir.
- Capítulo 3 - Tecnologías: Este apartado desglosa las herramientas más importantes que han sido utilizadas para el análisis y desarrollo de la aplicación que forma parte del proyecto.
- Capítulo 4 - Diseño e implementación: Este capítulo describe el diseño de los diferentes elementos y pantallas de la interfaz gráfica, la arquitectura *front-end* que se ha seguido para implementarla, así como la arquitectura del servidor *back-end* y el modelo de datos que han hecho posible el desarrollo conjunto del aplicativo.
- Capítulo 5 - Experimentos y validación: En esta sección se muestran evidencias del funcionamiento final de la aplicación, exponiendo un caso de uso en el que se seguirá con detalle el recorrido que recorre un usuario ficticio, desde que se registra por primera vez en la plataforma y hace uso normal de la aplicación editando proyectos y actividades, hasta que finalmente exporta el proyecto.

- Capítulo 6 - Conclusiones: En este capítulo se explica el grado de consecución de los objetivos, es decir, como se han ido desarrollando los diferentes apartados y alcanzando los diferentes hitos del proyecto. También se proponen algunos desarrollos para continuar mejorando y ampliando las funcionalidades de la solución. Finalmente, se hará mención a los aspectos de la formación del autor que han sido más relevantes para llevar a cabo este proyecto.

- Bibliografía: Finalmente, este apartado incluyen referencias a los estudios citados y a las diferentes *URLs* de la documentación técnica que ha sido utilizada para llevar acabo este proyecto.

- Apéndices: Se han incluido una serie de apéndices de especial interés.

  - Apéndice A - Esquema del modelo de datos: En este apéndice se exponen las diferentes colecciones de base de datos.
  - Apéndice B - Colección de métodos: Este apéndice muestra la relación de los diferentes servicios expuestos en el el servidor y describe brevemente su funcionamiento y finalidad.
  - Apéndice C - XML exportado: Es una muestra del documento *XML* que corresponde a la exportación del proyecto usado como ejemplo en el Capítulo 5, Experimentos y validación.
  - Apéndice D - Árbol completo de componentes: Muestra en varias páginas la arquitectura completa del árbol de componentes de la aplicación *front-end.*

1.5. Repositorios

El proyecto se encuentra íntegro en *GitHub* y es accesible a cualquier usuario que quiera consultarlo, clonarlo o contribuir. Se han utilizado dos repositorios para este fin, uno para la parte *front-end* del proyecto y otro distinto para la parte *back-end*.

- **Front-End:** https://github.com/Alexggp/dedos-editor-spa

- **Back-End:** https://github.com/Alexggp/dedos-editor-server

Capítulo 2

Objetivos

El el presente capítulo se hace referencia a los objetivos generales del proyecto, concretando también algunos aspectos más específicos de estos y una planificación temporal de los desarrollos a seguir.

2.1. Objetivo general

El objetivo principal de este proyecto es adaptar la herramienta DEDOS-Editor al entorno web. Para ello se ha querido, además, utilizar tecnologías de desarrollo web de última generación para aprovechar así al máximo los beneficios de rendimiento y funcionalidad que nos proporciona una plataforma web y asegurarnos de que la vida útil del producto sea óptima.

Para cumplir este último requisito se ha elegido el *framework React*, para construir la interfaz de usuario, y *Node.js* con *MongoDB*, para desarrollar la parte del servidor. También se han procurado seguir buenas prácticas de desarrollo, así como la utilización de las librerías de terceros que tengan una clara tendencia de uso en la actualidad. De esta manera, nos aseguramos de que se estén utilizando tecnologías que reciban soporte en el medio y largo plazo y no sea necesaria una refactorización del código hasta que cambie el paradigma del desarrollo web en un futuro lejano.

Este cambio tecnológico busca mejorar la experiencia de usuario para los docentes que generen contenido, sobre todo, durante las primeras fases de implantación de la herramienta en un centro. Al centralizar el acceso, dejará de ser necesaria la instalación del paquete de la herramienta y se podrá acceder a ella mediante cualquier navegador. Esto elimina el requisito

de utilizar un sistema operativo concreto y ofrece la ventaja de poder acceder a la herramienta desde cualquier equipo de forma remota.

En este TFG se busca crear una plataforma multi-usuario, gestionando la seguridad mediante un login con las más modernas garantías de seguridad web. De tal manera que el docente pueda acceder con sus credenciales de usuario a su cuenta personal en la plataforma, y siempre tenga acceso a sus proyectos desde cualquier lugar, podrá crearlos, editarlos y descargarlos siempre que quiera. Además podrá gestionar de forma paralela tantos proyectos como quiera desde una misma cuenta de usuario. Siempre con la seguridad de que ningún otro usuario de la plataforma pueda interferir de forma maliciosa en su desarrollo.

Por otro lado, un objetivo claro de esta refactorización es optimizar la apariencia y las mecánicas de usabilidad de la herramienta, mejorando fallos de usabilidad y *bugs* que se han podido encontrar durante el transcurso de los años. No obstante, se ha puesto especial hincapié en respetar aquellos aspectos de la herramienta que han sido ya validados por procesos de evaluación de usabilidad en su versión de escritorio. Debe ser fácil para usuarios antiguos de la herramienta DEDOS-Editor transicionar hacia el uso de la plataforma web y que las mecánicas y elementos con los que ellos estén más familiarizados no hayan cambiado de forma abrupta, que sea un cambio que se sienta natural.

Al desarrollar la aplicación con tecnologías de última generación en el sector del desarrollo web, uno de mis objetivos ha sido que el código de este proyecto pueda servir como modelo para sentar las bases de desarrollos futuros, así como ejemplo de arquitectura y buenas prácticas. Por lo tanto, este documento se pretende redactar también como un manual técnico que permita a próximos desarrolladores comprender el código que se ha escrito y poder realizar trabajos sobre él.

2.2. Plan de trabajo

Para alcanzar los objetivos generales, se han planteado una serie de hitos, que de manera ordenada, se deben ir cumpliendo para desarrollar esta aplicación de forma ordenada.

1. **Toma de requisitos**: Antes de iniciar los desarrollos, es necesario un estudio detallado del funcionamiento de la herramienta DEDOS-Editor, así como un análisis de su impacto

en el ámbito educativo, para ayudar al diseño de las nuevas funcionalidades, corrección de errores y la adaptación óptima al entorno web.

2. **Creación de la plataforma web:** Consiste en la creación de la plataforma web en la que el usuario podrá navegar por diferentes páginas, crear y editar proyectos y actividades.

3. **Control de usuarios:** La plataforma debe estar securizada con un sistema de registro basado en usuario y contraseña. Además debe mostrar un contenido diferente asociado a cada usuario.

4. **Editor de actividades:** Este hito está centrado en las funcionalidades principales del área de edición de actividades (véase subsección 4.1.3), la creación de items, áreas y tokens, y sus interacciones.

5. **Creación de objetivos:** Se definirán e implementarán las diferentes funcionalidades que pueden desempeñar las herramientas de edición de actividades, conocidas como objetivos, que modelan la finalidad de cada actividad y como el usuario puede interactuar con ésta.

6. **Exportación de proyectos:** El objetivo de este punto es desarrollar las funcionalidades necesarias para que el usuario pueda descargar en su navegador el archivo *ZIP* que contiene toda la información exportada del proyecto seleccionado que previamente creó en el editor.

7. **Pruebas y resolución de errores:** Se reservará un periodo de tiempo para realizar pruebas de uso en la aplicación y detectar posibles fallos, que deberán ser solucionados.

8. **Documentación**: Se trata del esfuerzo destinado a redactar la presente memoria del TFG.

Cada uno de estos hitos se trabajará desde cuatro frentes distintos. Por un lado, el diseño de la interfaz gráfica, por otro, su implementación en la aplicación *front-end*, su integración con los desarrollos de servicios *back-end* y por último, el modelo de datos gestionado desde base de datos.

2.3. Planificación temporal

A continuación se mostrará la planificación temporal que se ha estimado para realizar las tareas descritas en el apartado anterior, mediante un diagrama de Gantt.

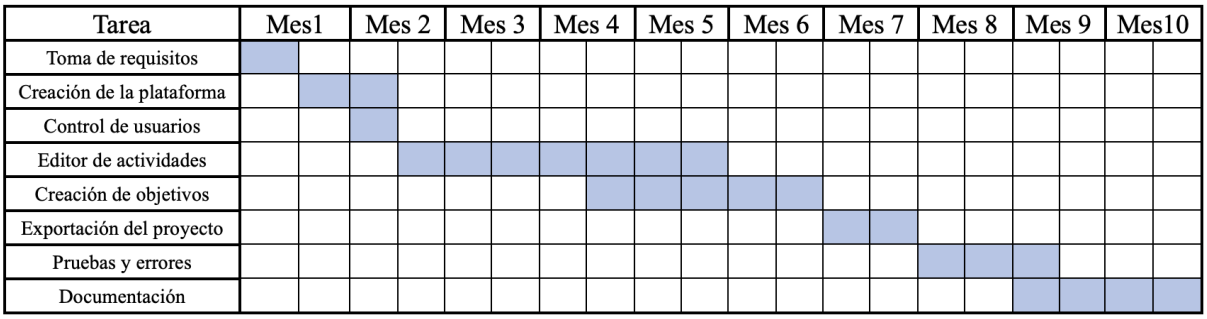

Figura 2.1: Planificación temporal.

Cabe destacar que para realizar dicha planificación se ha tenido que tomar en consideración la reducida disponibilidad durante la semana lectiva que se le ha podido dedicar, dado que ha sido necesario compatibilizar el desarrollo de la aplicación y la redacción de la presente memoria del proyecto con mi vida laboral, puesto que soy empleado a jornada completa.

Capítulo 3

Tecnologías

Para alcanzar uno de los principales objetivos de este proyecto, utilziar tecnologías web de ultima generación, se ha optado por el *stack* tecnológico *MERN*: *MongoDB, Express, React y Node*.

A continuación se explicarán dichas tecnologías junto con el resto de librerías y herramientas implicadas en el desarrollo del proyecto. Veremos las herramientas empleadas para el desarrollo del *front-end*, el *back-end* y otras herramientas y clientes de escritorio, para facilitar el desarrollo local.

3.1. Front-End

En esta sección se presentan el framework de desarrollo y las diferentes librerías de terceros utilizadas para el desarrollo del componente front-end del proyecto.

El front-end, también conocido como lado del cliente, es la parte de una aplicación web que el usuario final ve y con la que interactúa directamente. Incluye la estructura visual, la funcionalidad y la interactividad de la aplicación, así como el diseño de la interfaz de usuario y la disposición de los elementos en la página.

3.1.1. React

React es un *framework*, basado en el lenguaje de programación *JavaScript*, de código abierto desarrollado por *Facebook* para construir aplicaciones web interactivas y de una sola página

(*SPA*). Se basa en la programación declarativa y utiliza el concepto de componentes para construir una interfaz de usuario modular y reutilizable.

Una de las principales funcionalidades de *React* es su capacidad para manejar el estado de la aplicación de manera eficiente. *React* utiliza una técnica llamada *Virtual DOM* para actualizar la interfaz de usuario cuando cambia el estado de la aplicación. Esto significa que solo se actualizan los componentes que han cambiado y, por lo tanto, reduce la cantidad de operaciones costosas de actualización del *DOM*, lo que resulta en una mejor eficiencia y rendimiento de la aplicación.

Además, *React* es altamente modular y se puede integrar fácilmente con otras bibliotecas y *frameworks*. También es muy popular y tiene una gran comunidad de desarrolladores que contribuyen constantemente a la mejora del *framework*.

Entre los beneficios de utilizar *React* se encuentra su enfoque en la reutilización de código y la modularidad, lo que permite un desarrollo más rápido y eficiente. También tiene una curva de aprendizaje relativamente suave, lo que lo hace accesible para desarrolladores de diferentes niveles de experiencia.

3.1.2. Redux

Redux es una biblioteca de gestión de estado de aplicaciones en *JavaScript* que se utiliza comúnmente con *React*. Su principal funcionalidad es la gestión centralizada del estado de la aplicación, lo que simplifica el flujo de datos y permite una mejor depuración del estado de la aplicación.

En una aplicación *React* típica, los datos se pasan entre componentes mediante *props* (propiedades) y *callbacks* (funciones que se pasan como argumento y se ejecuta más tarde cuando se cumple una condición), lo que puede volverse complicado en aplicaciones grandes y complejas. Con *Redux*, el estado de la aplicación se almacena centralmente en un árbol de estado global llamado *Store*, lo que permite un acceso inmediato desde cualquier componente y mejora el seguimiento de los cambios de estado, eliminando así la necesidad de comunicar los cambios entre componentes.

3.1.3. Redux Persist

Redux Persist es una biblioteca de persistencia de estado para *Redux* que permite guardar y cargar el estado de la aplicación en almacenamiento local o remoto. Esta biblioteca proporciona una solución fácil para mantener el estado de la aplicación incluso después de que se cierre el navegador o se actualice la aplicación. Con *Redux-Persist,* los datos pueden ser guardados en diferentes formatos como *JSON*, *MongoDB*, *Firebase*, entre otros, y se pueden configurar diferentes opciones como el tiempo de expiración de los datos o la persistencia selectiva de algunos elementos del estado.

3.1.4. Axios

Axios es una biblioteca de *JavaScript* utilizada para realizar solicitudes *HTTP* desde el navegador o desde *Node.js*. Es un cliente *HTTP* basado en promesas que ofrece una interfaz fácil de usar y una gran cantidad de funciones avanzadas para la gestión de solicitudes, como la gestión de encabezados, la cancelación de solicitudes y la transformación de datos. *Axios* se utiliza comúnmente en aplicaciones web para interactuar con *API* y enviar y recibir datos del servidor.

3.1.5. Mui

Mui (Material-UI) es una biblioteca de componentes de interfaz de usuario para *React* que implementa el lenguaje de diseño de materiales de *Google*. Ofrece una amplia variedad de componentes predefinidos como botones, tarjetas, formularios, tablas, etc, para crear interfaces de usuario atractivas y coherentes. Además, *Mui* también proporciona una *API* para la personalización de temas y estilos, lo que permite adaptar los componentes al estilo visual de la aplicación.

3.1.6. i18next

i18next es una biblioteca de internacionalización para aplicaciones web que facilita la traducción y localización de contenido. Permite separar el contenido de texto de la aplicación de su código, lo que facilita la traducción de la aplicación en diferentes idiomas.

3.1.7. React RnD

React Rnd (Resizable and Draggable) es una biblioteca de componentes para *React* que permite la creación de componentes redimensionables y arrastrables en una aplicación web.

3.1.8. React DnD

React DnD (Drag and Drop) es una biblioteca de componentes para *React* que permite la implementación de la funcionalidad de arrastrar y soltar elementos en una aplicación web.

3.1.9. React Xarrows

React-Xarrows es una biblioteca de componentes para *React* que permite dibujar flechas y líneas curvas entre elementos *HTML*. Esta biblioteca facilita la creación de gráficos y diagramas en aplicaciones web mediante la creación de conexiones visuales entre diferentes elementos de la interfaz.

3.2. Back-End

En esta sección se presentan el *framework* de desarrollo, la base de datos y las diferentes librerías de terceros utilizadas para el desarrollo del componente *back-end* del proyecto.

El *back-end*, también conocido como servidor, se refiere al componente del sistema que se encarga de procesar la información y proveer servicios y funcionalidades a través de una interfaz de programación de aplicaciones (*API*). En él se desarrolla la lógica de negocio de la aplicación, accediendo a los recursos del servidor como almacenamiento, bases de datos y servicios de *APIs* de terceros.

3.2.1. Node

Node.js es un entorno en tiempo de ejecución de *JavaScript* construido sobre el motor de *JavaScript* V8 de *Google Chrome*. Se utiliza principalmente para crear aplicaciones de red escalables y de alta velocidad.

Se basa en un modelo de programación asíncrono, lo que le permite manejar múltiples solicitudes simultáneamente sin bloquear el hilo de ejecución. Esto lo hace ideal para aplicaciones que deben manejar un gran volumen de solicitudes por minuto.

Es muy flexible y se puede utilizar en una amplia variedad de aplicaciones, desde servidores web hasta aplicaciones de escritorio y móviles. Además, cuenta con una gran comunidad de desarrolladores y una amplia gama de bibliotecas y módulos disponibles para su uso.

3.2.2. Express

Express es un *framework* de aplicación web de *Node.js* que permite crear aplicaciones web y *APIs* de manera rápida y fácil. *Express* proporciona una variedad de características y herramientas útiles para la creación de aplicaciones web, incluyendo enrutamiento, *middleware*, manejo de errores y manejo de solicitudes *HTTP*.

Es altamente personalizable, se puede adaptar fácilmente para satisfacer las necesidades concretas de cada aplicación. Permite agregar *middleware* personalizado para realizar tareas específicas, como la autenticación o la autorización, y también pueden usar una amplia variedad de módulos de terceros para extender la funcionalidad de la aplicación.

3.2.3. MongoDB

MongoDB es una base de datos *NoSQL* de código abierto utilizada para almacenar y administrar grandes cantidades de datos de manera eficiente. A diferencia de las bases de datos relacionales, *MongoDB* utiliza un modelo de datos basado en documentos, cada registro se almacena como un documento *JSON*, no requiere un esquema fijo, lo que hace que la lectura y escritura de datos sea más fácil, flexible y rápida. Está diseñado para ser escalable y poder manejar grandes cantidades de datos de manera eficiente.

Es fácilmente accesible a través de los controladores y módulos de *Node.js*. Lo que la convierte en una opción de base de datos lógica para éste proyecto, ya que es sencillo interactuar con la base de datos y realizar operaciones *CRUD* (Crear, Leer, Actualizar y Eliminar).

3.2.4. Mongoose

Mongoose es una librería de modelado de objetos de *MongoDB* para *Node.js*. Se utiliza para proporcionar una capa de abstracción sobre la base de datos *MongoDB*, lo que facilita la manipulación de los datos y la creación de esquemas para los mismos. Proporciona una gran variedad de características útiles, como validación de esquemas, transformación de datos, consultas y agregaciones, así como la creación de *middleware* personalizado para la manipulación de datos y eventos.

3.2.5. Helmet

Helmet es una librería de seguridad para aplicaciones web en *Node.js*. Proporciona una serie de *middleware* que ayuda a proteger una aplicación de una variedad de ataques web comunes, como inyecciones de código, ataques *CSRF* y *XSS*.

3.2.6. Multer

Multer es una librería *middleware* para *Node.js* que se utiliza para manejar la carga de archivos en aplicaciones web. Su función principal es analizar y almacenar los datos del archivo en un servidor, lo que permite a los usuarios cargar archivos en la aplicación.

3.2.7. Passport

Passport es una librería de autenticación para aplicaciones web en *Node.js*. Se utiliza para autenticar usuarios y proteger recursos restringidos en la aplicación. Proporciona un marco flexible y modular que permite utilizar diferentes estrategias de autenticación para adaptarse a las necesidades de su aplicación. Entre las estrategias de autenticación se encuentran la autenticación con nombre de usuario y contraseña, la autenticación con tokens y la autenticación con proveedores de terceros, como *Google* o *Facebook*

3.2.8. Bcrypt

Bcrypt es una librería de encriptación para aplicaciones web en *Node.js*. Se utiliza para cifrar las contraseñas de los usuarios y almacenarlas de forma segura en la base de datos de la

aplicación.

3.2.9. JsonWebToken

JsonWebToken es una librería de autenticación y autorización para aplicaciones web en *Node.js*. Se utiliza para generar y verificar tokens de acceso *JWT* que se utilizan para autenticar y autorizar usuarios en la aplicación.

3.2.10. Archiver

Archiver es una librería de *Node.js* que se utiliza para crear y manipular archivos comprimidos en múltiples formatos como *ZIP*, *TAR* y *GZIP*.

3.2.11. Node-html-to-image

Node-html-to-image es una librería de *Node.js* que se utiliza para convertir un archivo *HTML* en una imagen.

Esta librería es útil para crear imágenes de vista previa de sitios web, generar imágenes de contenido generado por el usuario, o crear imágenes de gráficos personalizados. En este caso, se ha utilizado para generar, desde el servidor, las miniaturas de los proyectos en el momento de la descarga del archivo.

3.3. Herramientas de desarrollo

En esta sección se presentan el resto de herramientas de desarrollo que han sido utilizadas en la construcción del aplicativo.

3.3.1. Visual Studio Code

Visual Studio Code es un editor de código abierto gratuito desarrollado por *Microsoft*. Es compatible con múltiples lenguajes de programación y sistemas operativos.

VS Code cuenta con una interfaz de usuario intuitiva y personalizable que ofrece diversas funciones y herramientas para facilitar la escritura de código, como la sugerencia de autocom-

pletado, la resaltación de sintaxis, terminal y depurador integrado y la integración con herramientas de control de versiones como *Git*.

3.3.2. Redux DevTools

Redux DevTools es una extensión para navegadores web que permite visualizar y manipular el estado de la aplicación en tiempo real, examinar el historial de acciones de *Redux*, revertir y reproducir las acciones, y ver cómo afectan al estado de la aplicación.

3.3.3. ReacTree

ReacTree es una extensión para el editor de código *Visual Studio Code* que permite visualizar y explorar el árbol de componentes de una aplicación *React*. La extensión muestra un árbol jerárquico de todos los componentes de la aplicación y sus relaciones entre sí.

3.3.4. MongoDB Compass

MongoDB Compass es una interfaz gráfica de usuario para *MongoDB* que permite visualizar, analizar y manipular datos almacenados en bases de datos *MongoDB* de manera más intuitiva y eficiente

3.3.5. Postman

Postman es un cliente *API* que proporciona una interfaz gráfica de usuario para crear y enviar solicitudes *HTTP/HTTPS*, y recibir y visualizar las respuestas de manera clara y organizada. También permite la automatización de pruebas de regresión y la generación de documentación.

3.3.6. Git

Git es un sistema de control de versiones distribuido. Se utiliza para el seguimiento de cambios en el código fuente durante el proceso de desarrollo de software, así como repositorio de seguridad del código.

Permite a los desarrolladores trabajar en el mismo código fuente de manera colaborativa, registrando los cambios realizados sobre el código en un repositorio central. De esta manera, se

pueden mantener diferentes versiones del código y coordinar el trabajo de diferentes desarrolladores que trabajen en un mismo proyecto.

Capítulo 4

Diseño e implementación

El propósito principal de la aplicación es el de actualizar la herramienta DEDOS-Editor al entorno web, utilizando tecnologías de uso emergente en el ámbito del desarrollo de aplicaciones.

Al pasar de aplicación de escritorio a plataforma web, ha sido necesario añadir nuevas funcionalidades como la gestión de usuarios y proyectos, de manera que cualquier usuario pueda acceder a sus proyectos desde cualquier terminal.

La aplicación debe permitir a los docentes crear diferentes actividades educativas, así como editar las previamente existentes. Este objetivo se consigue a través del uso de tarjetas de texto o de imagen con las que los alumnos podrán interactuar para alcanzar los objetivos propuestos.

4.1. Interfaz gráfica

En esta sección se expondrá el diseño y experiencia de usuario (*UX*) de la aplicación así como todas las funcionalidades a las que el usuario tiene acceso.

Se ha realizado un rediseño de la interfaz original de la aplicación DEDOS-Editor para adaptarla al entorno web tratando de mantener la misma estructura y usabilidad.

Por otro lado, ha sido necesario añadir nuevos elementos y pantallas para poder implementar las funcionalidades exclusivas de esta versión, como la gestión de idioma, usuarios y proyectos.

El diseño *UX* es crucial en una aplicación web ya que influye directamente en la satisfacción del usuario y su capacidad para completar tareas de forma eficiente. Una buena *UX* se centra en diseñar la interfaz de usuario de manera intuitiva y fácil de usar, reduciendo posibles

confusiones.

Su diseño debe ser responsivo, es decir, que busca adaptar el contenido y la apariencia de la aplicación al tamaño de la pantalla del dispositivo utilizado para visualizarla.

En las siguientes subsecciones se describirá el diseño y funcionalidad de cada una de las tres secciones principales de la aplicación.

4.1.1. Pantalla de ingreso

Es la pantalla de inicio de la aplicación desde la cuál el usuario podrá acceder al sistema o registrarse en él.

Figura 4.1: Pantalla inicial de ingreso.

El modo por defecto es el modo acceso, el usuario podrá rellenar el formulario con su email y contraseña y acceder a la aplicación. También tendrá la opción de cambiar al modo registro si no ha creado previamente una cuenta de usuario.

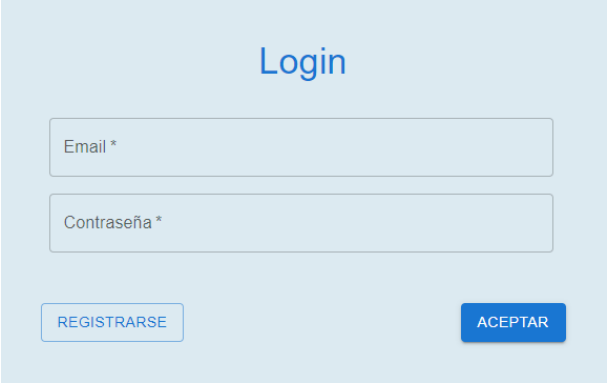

Figura 4.2: Formulario de acceso.

Si se intenta acceder con un usuario o contraseña incorrectos, se le comunicará al usuario mediante una pastilla de error.

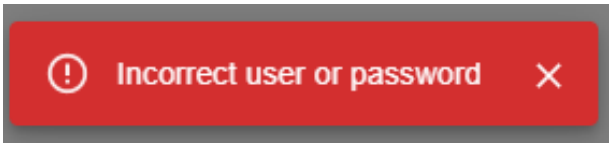

Figura 4.3: Error de acceso.

Esta pastilla llamada componente ErrorBanner, será utilizada para representar mensajes de error y alertar al usuario de posibles fallos de la aplicación. Cada uno de los servicios que conectan con el back-end, tendrán acceso a este componente para manejar su error de manera centralizada.

En el modo de registro, además de tener la opción de volver al modo acceso, podrá rellenar un pequeño formulario con su nombre, email y contraseña. De esta manera se creará un nuevo usuario y se le dará acceso automático a la aplicación.

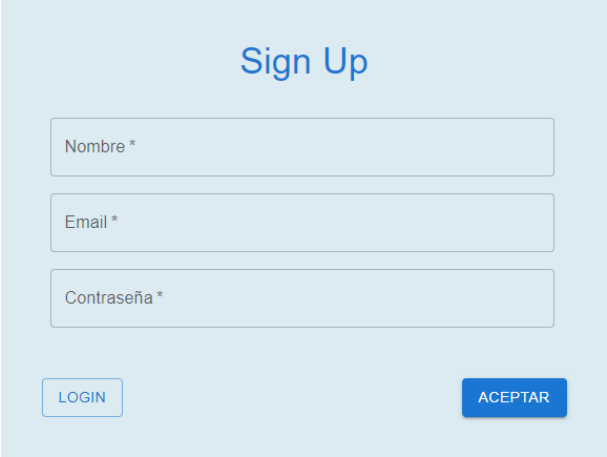

Figura 4.4: Formulario de registro.

Ambos formularios cuentan con validación, de manera que si no se introducen los campos obligatorios, que en ambos casos serían todos los campos, se notificará al usuario con un error indicándole el fallo correspondiente.

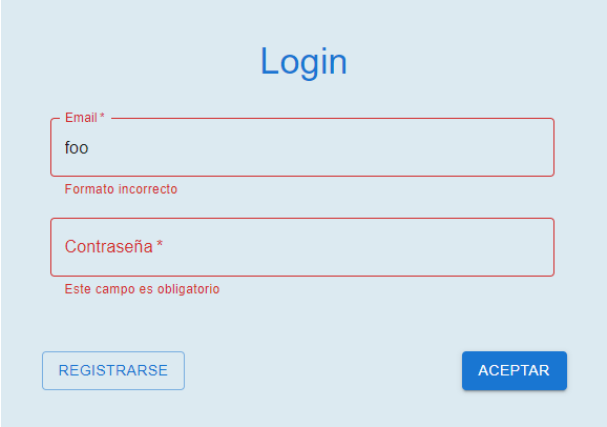

Figura 4.5: Validación en el formulario de acceso.

De la misma manera sucederá si la validación detecta que los formatos introducidos para los campos del formulario son incorrectos.

4.1.2. Menú de proyectos

Tras acceder a la aplicación, el usuario será redirigido al menú de proyectos. Aquí podrá ver todos los proyectos que ha creado previamente y acceder a ellos, así como crear proyectos nuevos.

Figura 4.6: Menú de proyectos.

Los proyectos se presentan con un título y una breve descripción. En la zona inferior de la caja del proyecto, se da la opción de descargar, editar y eliminar el proyecto. La descarga se realizará en formato *ZIP* compatible con la antigua versión de DEDOS-Editor.

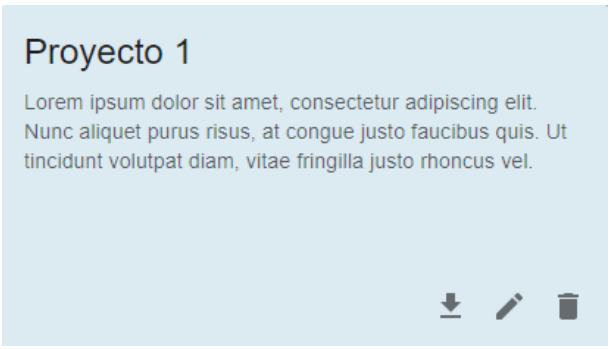

Figura 4.7: Presentación de un proyecto en el menú.

El usuario podrá crear tantos proyectos como necesite si hace click en el elemento *Crear nuevo* que aparece integrado como la última tarjeta del menú de proyectos.

Tanto si está creando un nuevo proyecto o editando uno existente, se mostrará el formulario de edición y creación de proyectos.

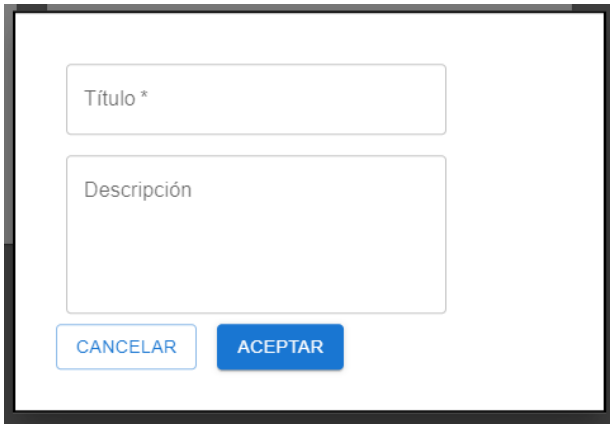

Figura 4.8: Formulario de edición y creación de proyectos.

En la esquina superior derecha del menú de proyectos se encuentra el selector de idioma, desde el que el usuario podrá cambiar el idioma de toda la aplicación. Actualmente disponible en español e inglés.

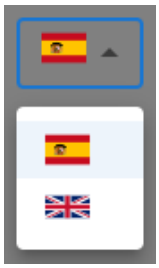

Figura 4.9: Selector de idioma.

Junto a este selector se encuentra el menú de usuario, un desplegable desde el cuál el usuario podrá ver sus datos personales y cerrar sesión en la aplicación.

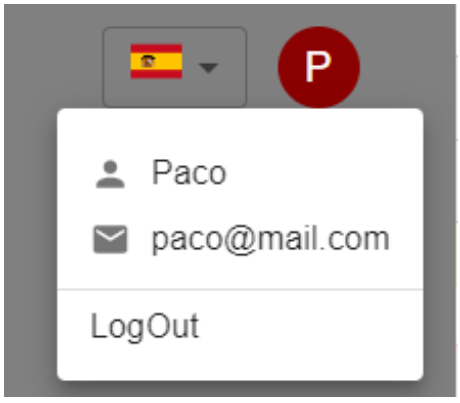

Figura 4.10: Menú de usuario.

Al navegar entre diferentes pantallas, a veces, se mostrará un componente *Spinner* que simboliza la espera hasta la carga completa de los elementos. Su tiempo de aparición en pantalla dependerá de lo que tarde la aplicación en realizar las transacciones con el servidor. Si la velocidad de red es buena y la carga de datos baja, posiblemente no llegue a mostrarse nunca.

Figura 4.11: Componente LoadingPage.

Si introducimos manualmente un *path* erróneo en el navegador, se nos mostrará el componente página *NotFound*.

Figura 4.12: Componente NotFound.

4.1.3. Editor de proyectos

Si el usuario selecciona uno de los proyectos del menú entrará en la herramienta de edición.

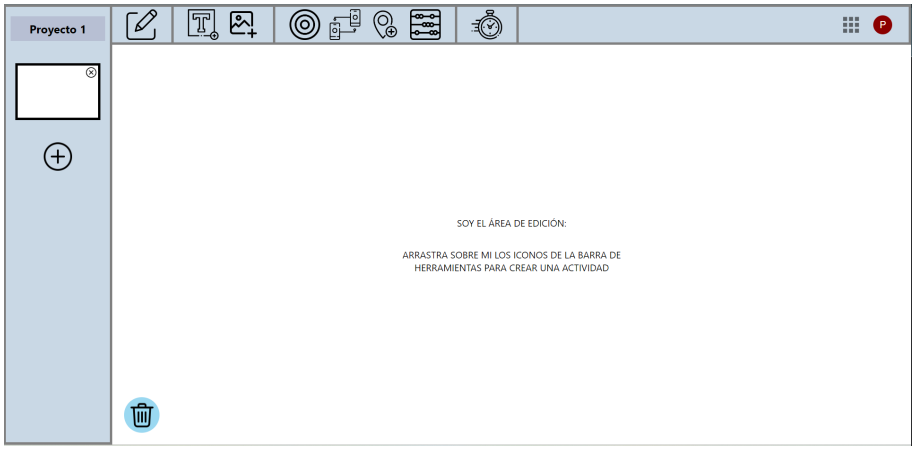

Figura 4.13: Editor de proyectos.

Podemos apreciar tres zonas bien diferenciadas dentro del editor.

Área de edición

El área de edición es la zona principal del editor, donde crearemos y previsualizaremos los diferentes componentes de nuestra actividad.

Al iniciar un nuevo proyecto ésta aparecerá vacía. Si arrastramos los distintos iconos de la barra de herramientas podremos crear diferentes combinaciones de elementos que le darán sentido a nuestra actividad.

En la esquina inferior izquierda del área de edición, se encuentra la papelera. Podremos arrastrar hasta ella cualquier elemento del área de edición para eliminarlo.

Figura 4.14: Papelera.

Barra de herramientas

La barra de herramientas nos proporciona una serie de elementos que podremos arrastrar dentro del área de edición.

Figura 4.15: Barra de herramientas.

En la siguiente subsección se explicarán en detalle las diferentes herramientas que podremos utilizar, sus efectos y como combinarlas entre sí.

Cabe destacar que, en el margen derecho, volvemos a encontrar el menú de usuario y a su izquierda el icono de vuelta al menú de proyectos.

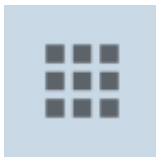

Figura 4.16: Volver al menú de proyectos.

Menú de actividades

Un proyecto se puede componer de más de una actividad, y para poder visualizarlas, en el margen izquierdo del editor se encuentra la barra lateral con el menú de actividades. Además, en la parte superior se muestra el título del proyecto.

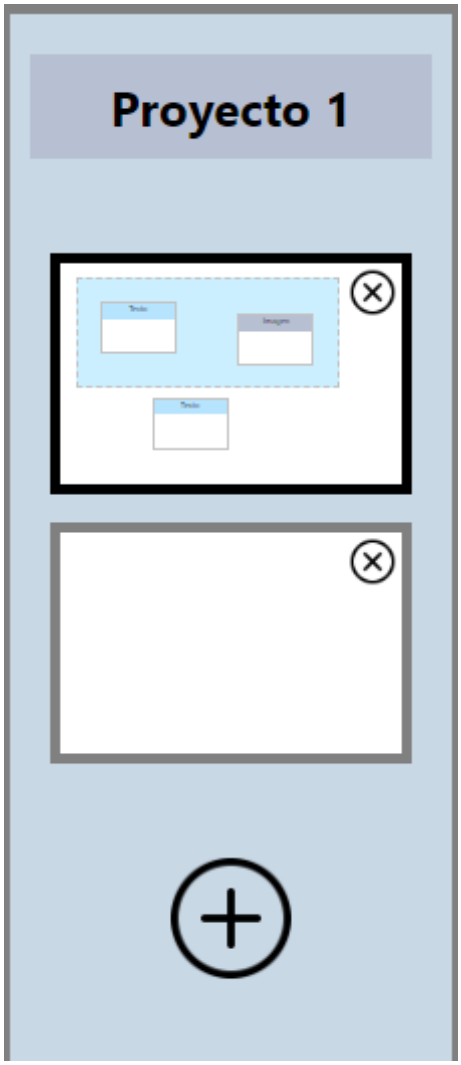

Figura 4.17: Menú de actividades.

Las diferentes actividades se presentarán como miniaturas de cada una de ellas, mostrándose en ellos una versión reducida de los elementos que la integran y actualizando su aspecto en tiempo real si se modifica la actividad en el editor. Marcándose con borde negro la actividad que está siendo editada en cada momento.

Pueden realizarse diferentes acciones desde el menú, como eliminar una actividad haciendo click en el aspa de su esquina superior derecha o crear nuevas actividades desde el icono central del más.

4.1.4. Herramientas de edición

En este apartado presentaremos las diferentes herramientas de edición, sus efectos y como interactúan entre sí.

Área

Figura 4.18: Icono de área.

Las áreas son tableros de juego que contienen las tarjetas de imagen o texto y permiten a los usuarios interactuar con ellas.

Lo añadiremos arrastrando su icono desde la barra de herramientas y y soltándolo en cualquier punto del área de edición, donde se creará el elemento.

Figura 4.19: Área de jugador.

Si mantenemos pulsado el ratón sobre el elemento, podremos arrastrarlo por toda el área de edición.

El icono de la esquina inferior derecha nos permite redimensionar el área y el icono superior, con forma de carpeta, nos abre el explorador de archivos para añadir una imagen de fondo.

Si hacemos click en el icono de la esquina superior izquierda, transformamos el área de tipo jugador en área tipo juego.

Figura 4.20: Área de juego.

Al hacer de nuevo click, volveremos a modo jugador.

Tarjeta de texto

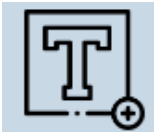

Figura 4.21: Icono de tarjeta de texto.

Las tarjetas de texto permiten al usuario introducir texto para formar parte de la actividad.

Lo añadiremos arrastrando su icono desde la barra de herramientas y y soltándolo en cualquier punto del área de edición, donde se creará el elemento.

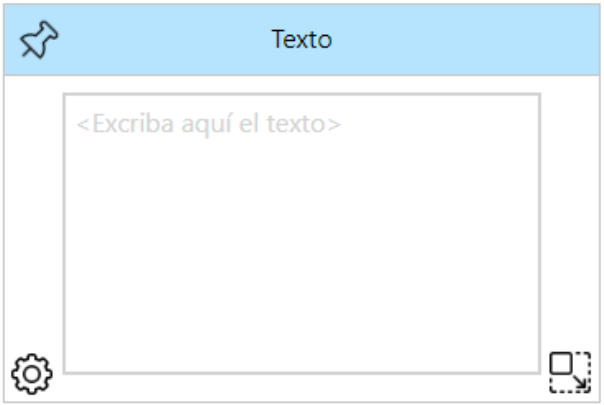

Figura 4.22: Tarjeta de texto.

Además de la entrada de texto, éste elemento presenta las funcionalidades de tarjeta que se comparten con la tarjeta de imagen que veremos a continuación.

Si mantenemos pulsado el ratón sobre la cabecera del elemento, podremos arrastrarlo por toda el área de edición. Para evitar este comportamiento, podemos activar el icono de la chincheta a la izquierda de la misma cabecera, que anclará la tarjeta en el área de edición.

El icono de la esquina inferior derecha nos permite redimensionar la tarjeta.

En la esquina inferior derecha, podemos encontrar el icono del engranaje, que nos da paso al menú de opciones de configuración de la tarjeta.

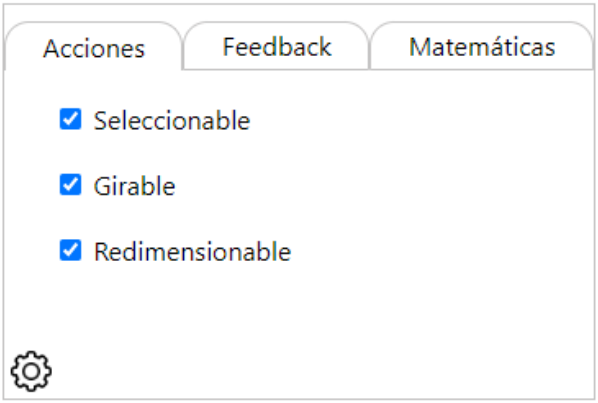

Figura 4.23: Configuración de tarjeta.

Tarjeta de imagen

Figura 4.24: Icono de tarjeta de imagen.

Las tarjetas de imagen permiten al usuario introducir imágenes para formar parte de la actividad.

Lo añadiremos arrastrando su icono desde la barra de herramientas y y soltándolo en cualquier punto del área de edición, donde se creará el elemento.

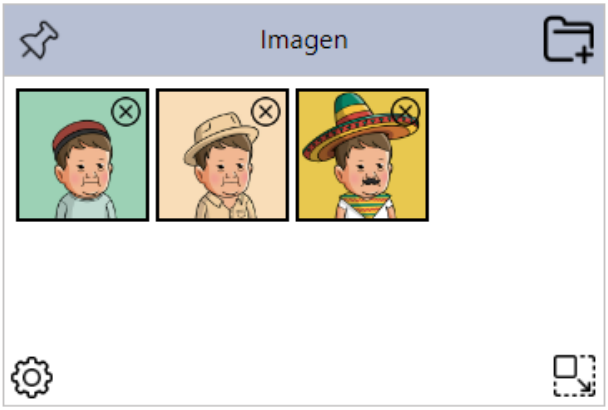

Figura 4.25: Tarjeta de imagen.

Las imágenes se añaden gracias al icono de la carpeta en la esquina superior derecha, que abrirá el navegador de archivos. Se podrán añadir tantas imágenes como el usuario desee.

Para eliminar imágenes de la tarjeta, podemos presionar el icono del aspa en la esquina superior derecha de la misma.

Éste elemento presenta también todas las funcionalidades de tarjeta, que se comparten con la tarjeta de texto, que hemos visto en el aparto anterior.

Objetivo de selección

Figura 4.26: Icono de objetivo de selección.

El objetivo de selección permite convertir tarjetas en elementos seleccionables por el usuario que realice la actividad.

Lo añadiremos arrastrando su icono desde la barra de herramientas y hasta la tarjeta objetivo.

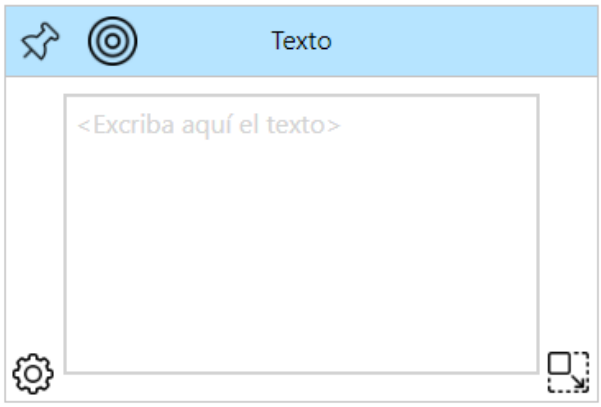

Figura 4.27: Tarjeta seleccionable.

Se notificará que un objetivo de selección ha sido añadido a una tarjeta mostrando el icono del objetivo dentro de la cabecera en su margen izquierdo.

Si queremos eliminar dicho objetivo de la tarjeta, bastará con arrastrar este icono hasta la papelera del área de edición.

Objetivo de emparejamiento

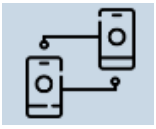

Figura 4.28: Icono de objetivo de emparejamiento.

El objetivo de emparejamiento permite relacionar tanto áreas y tarjetas entre sí para crear actividades de emparejamiento.

Lo añadiremos arrastrando su icono desde la barra de herramientas y hasta la tarjeta o área objetivo.

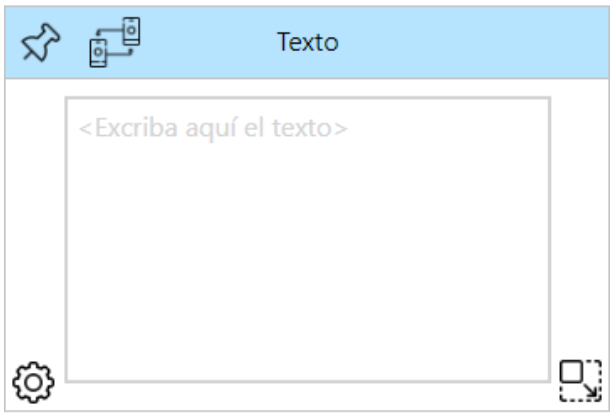

Figura 4.29: Tarjeta emparejable.

Se notificará que un objetivo de selección ha sido añadido a una tarjeta mostrando el icono del objetivo dentro de la cabecera en su margen izquierdo o en el área en su esquina superior izquierda.

Inmediatamente aparecerá en pantalla una flecha que el usuario debe unir con el área o tarjeta objetivo para crear el emparejamiento

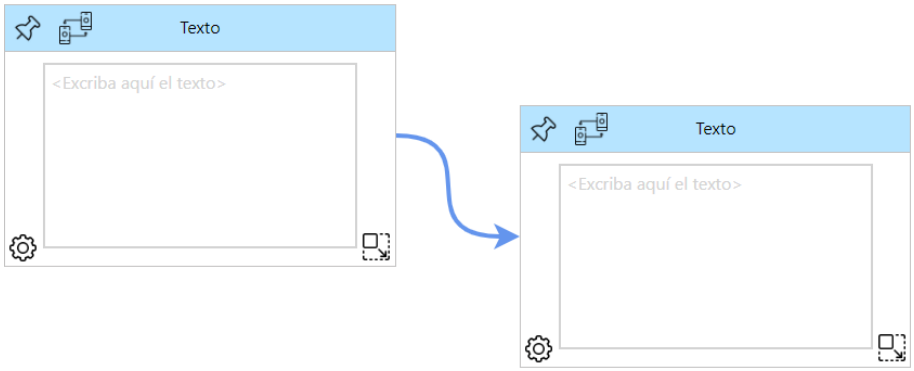

Figura 4.30: Tarjetas emparejadas.

Si queremos eliminar el objetivo de emparejamiento, bastará con arrastrar el icono de una de las tarjetas o áreas hasta la papelera del área de edición. Desaparecerá del editor tanto la flecha como el objetivo en ambos elementos.

Objetivo de caminos

Figura 4.31: Icono de objetivo de caminos.

El objetivo de caminos no ha sido implementado en esta versión de la aplicado dado que quedó fuera del alcance del proyecto.

Aún así ha quedado preparado el icono para ser implementado en un futuro.

Contador de tarjetas

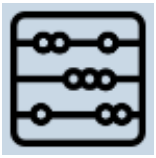

Figura 4.32: Icono de contador de tarjetas.

Permite añadir un contador a las tarjetas y a las áreas.

Lo añadiremos arrastrando su icono desde la barra de herramientas y hasta la tarjeta o área objetivo.

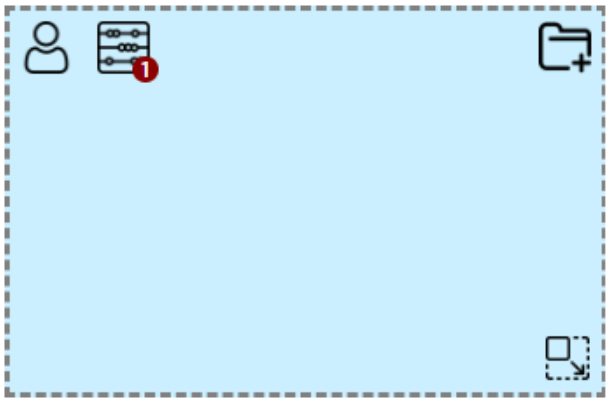

Figura 4.33: Área con contador

Se notificará que un contador ha sido añadido a una tarjeta mostrando el icono dentro de la cabecera en su margen izquierdo o en el área en su esquina superior izquierda.

Una pequeña burbuja granate mostrará el número asignado en el contador. Si hacemos click en ella, se nos permitirá modificar este valor.

Figura 4.34: Modal del contador

Si queremos eliminar el contador, bastará con arrastrar su icono desde el elemento hasta la papelera del área de edición.

Límite de tiempo

Figura 4.35: Icono de límite de tiempo.

Permite añadir un límite de tiempo para realizar la actividad.

Lo añadiremos arrastrando su icono desde la barra de herramientas y hasta el área de edición.

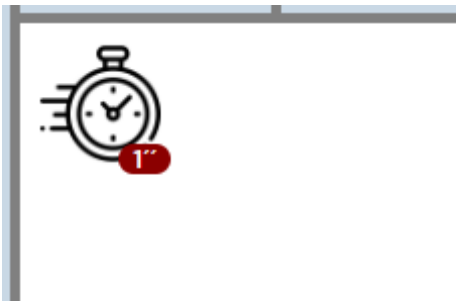

Figura 4.36: Actividad con límite de tiempo

Se notificara que el límite de tiempo ha sido añadido mostrando el icono dentro del área de edición en su esquina superior izquierda.

Una pequeña burbuja granate mostrará, en segundos, el tiempo del contador. Si hacemos click en ella, se nos permitirá modificar este valor.

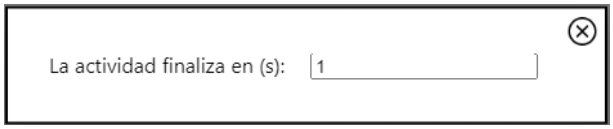

Figura 4.37: Modal del temporizador

Si queremos eliminar el límite de tiempo, bastará con arrastrar su icono hasta la papelera del área de edición.

4.2. Arquitectura Front-end

Las arquitecturas f*ront-end* de aplicaciones construidas con *React* se basan en componentes. Estos son elementos modulables e independientes que interactúan entre sí y que pueden repetirse y colocarse en diferentes partes del código y unirse para crear otros componentes.

Un componente debe contar con todos los elementos necesarios, lógica y estilos *CSS* necesarios para funcionar de forma autónoma al recibir unos parámetros de entrada.

Estos forman una árbol de jerarquías que conforma la estructura del proyecto. Al final del documento, en el apéndice D se muestra el árbol de componentes completo utilizado en este proyecto.

Como existe una jerarquía clara de componentes en el proyecto, se ha estructurado el proyecto en consecuencia; los archivos de los componentes hijos han sido almacenados dentro de la carpeta de los componentes padre. Sin embargo, si un componente puede ser reutilizado en varios puntos del código, se ha colocado en el nivel superior de la carpeta de componentes.

Podemos observar también que cada componente está formado por su archivo *JS*, que describe su lógica y su archivo *CSS* que modifica sus estilos. De esta forma, cada componente es independiente del resto del proyecto.

4.2.1. Tipos de componentes

Componentes I/O

Los componentes de entrada y salida son los elementos más básicos y numerosos de la aplicación. Se limitan a mostrar y recoger datos y en ellos no se incluye lógica de negocio que procese estos datos. Un ejemplo podría ser un botón o un input de texto.

Contenedores

Los contenedores son componentes que renderizan dentro de sí uno o más componentes I/O. Pueden realizar lógica de negocio y procesamiento de la información, así como coordinar el funcionamiento de los distintos componentes que lo integran. Por ejemplo, un formulario que carga diferentes entradas de texto y el botón de enviar los datos, y gestiona las diferentes acciones asociadas a ellos.

Páginas

Una página es un contenedor de más alto nivel que representa una vista completa de nuestra aplicación. Es un contenedor de contenedores asociado a cierta ruta de nuestra aplicación.

HOC

Un *Higher-Order Component (HOC)* es una técnica en *React* para reutilizar la lógica común de componentes. Un *HOC* es una función que toma un componente y devuelve un nuevo componente con una funcionalidad adicional.

4.2.2. Store de Redux

Como se explica en la subsección 3.1.2, *Redux* es una biblioteca que realiza una gestión centralizada del estado de la aplicación, lo que permite simplificar el flujo de datos entre componentes.

En la práctica, funcionaría como un repositorio de variables globales, una especie de base de datos no relacional cargada en la memoria del navegador, que permite a cada componente acceder a ella directamente y extraer, añadir o modificar los valores de las variables.

El *store* de *Redux* se compone de diferentes colecciones de objetos, llamadas *reducers*, en ellos se almacenarán los datos de nuestra aplicación. Los componentes de nuestra aplicación estarán observando cualquier posible cambio de estos datos en tiempo real, y cuando esto suceda, se dispararán en ellos diferentes acciones, como mostrar los datos, pedirlos, ocultar o mostrar componentes o lanzar alertas al usuario.

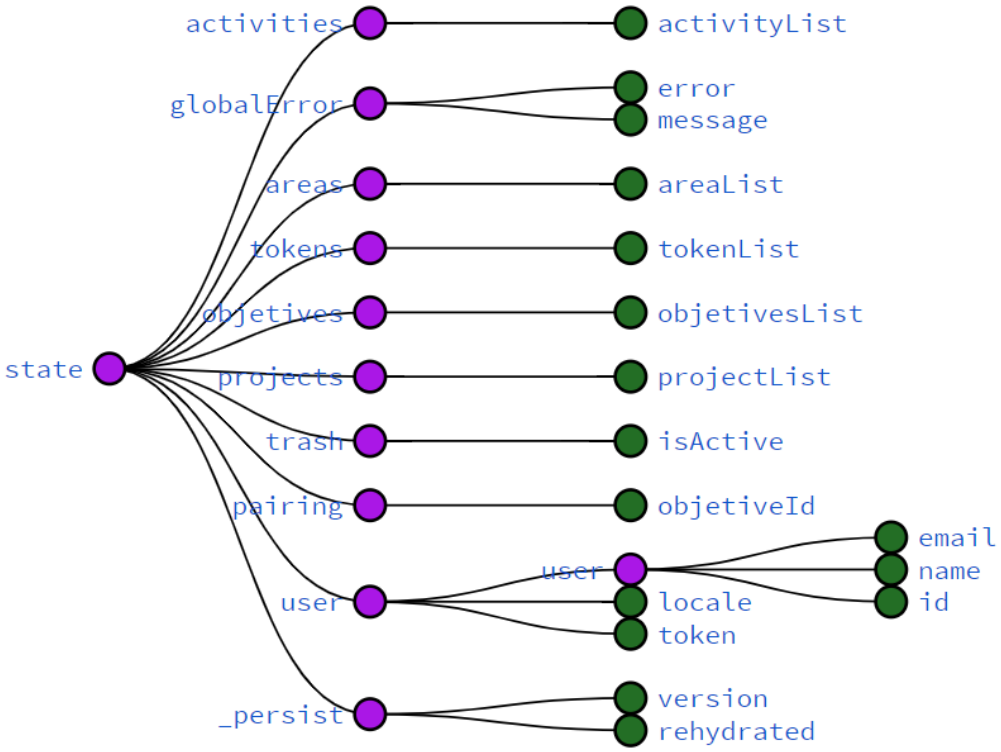

Figura 4.38: Árbol de *reducers* del *store* de Redux del proyecto.

Los *reducers activities, areas, tokens, objetives, projects y user* almacenarán las colecciones de datos de nuestra aplicación de forma segmentada una vez han sido recogidos de base de datos.

GlobalError, Trash y Pairing sirven para controlar el estado de los mensajes de error, la papelera y la funcionalidad de emparejamiento y ayudar al funcionamiento de los componentes asociados.

El *reducer persist* es una colección virtual, generada por la librería *ReduxPersist* (3.1.3) que nos permite gestionar la persistencia de los datos del *store* en la memoria del navegador y forzar así que cierta información de la aplicación se mantenga al recargar la página o al abrir

una pestaña nueva, sin la necesidad de volver a llamar a base de datos.

Se ha implementado una lista blanca, para persistir los datos de una o varias colecciones específicas. En este caso sólo afecta al *reducer user*. Esto ayuda a la gestión de la sesión y permite recargar la página sin la necesidad de requerir de nuevo el *login* al usuario.

4.2.3. Internacionalización

La internacionalización consiste en la capacidad, por parte de la aplicación, de soportar más de un idioma de tal manera que la usabilidad no se vea afectada de cara al usuario final.

En esta aplicación, se realiza a través de la librería *i18next* (3.1.6), con la cual se han configurado dos diccionarios de idioma, para español e inglés.

Estos diccionarios son colecciones de clave y valor que contienen todos los literales, o textos, de la web almacenados como archivos *JSON* dentro del proyecto. Posteriormente todos los extractos de código que contengan esos textos deben ser etiquetados con las claves para ser inyectados con los valores en forma de texto a la hora de ser renderizados.

El idioma estará determinado desde el *store* de *Redux* por el valor de la variable *locale* del *reducer user* . Se ha programado para que, por defecto, se tome el valor del idioma del navegador. Sin embargo, el usuario podrá cambiar siempre que quiera el idioma utilizando el selector de idiomas presente en el menú de proyectos (figura 4.9).

Siempre que se requiera, se podrán añadir nuevos idiomas de forma sencilla con la única acción de añadir más diccionarios.

4.2.4. Gestión de capas

Uno de los mayores retos en el desarrollo de la solución *front-end* ha sido simular una interfaz gráfica de escritorio con ventanas desplazables en el área de edición. Para lograr la interacción natural entre los elementos, se diseñó un complejo sistema de capas que afecta a todos los elementos del área de edición y requiere actualizaciones constantes. Esto se logró utilizando la propiedad *z-index* de *CSS3*, que permite controlar el nivel de posicionamiento de los elementos dentro de un *div* específico.

El comportamiento por defecto en la maquetación web es que los elementos añadidos posteriormente se coloquen en capas superiores. Sin embargo, esto plantea un problema cuando se

trata de tarjetas y áreas, ya que las tarjetas deben estar siempre por delante de las áreas, independientemente del orden de creación. Además, las áreas deben poder desplazarse por encima de otras áreas sin afectar las tarjetas que contienen.

Para abordar esta complejidad, se implementó la siguiente solución: cada actividad tiene un contador llamado *zIndexTop*, que comienza en 0 y se utiliza como referencia para los elementos creados. Cada elemento del área de edición tiene una propiedad *zIndex* que modifica el valor de *z-index* en los estilos *CSS*, colocándolo en el nivel de capa correspondiente.

Cuando se crea un nuevo elemento en el área de edición, se le asigna el valor actual de *zIndexTop* de la actividad. Luego, se incrementa el valor de *zIndexTop* en uno. Si el usuario interactúa con un elemento existente, su valor *zIndex* se actualiza con el valor de *zIndexTop* y se incrementa el contador de la actividad.

Cada vez que se actualiza alguno de estos valores, se refleja en el *store* de *Redux* y en la base de datos, asegurando que el elemento con el que el usuario interactúa siempre esté en la capa superior de forma persistente.

Además de gestionar las capas, también se debe gestionar la interacción entre tarjetas y áreas. Cada área tiene un listado de tarjetas asociadas, pero también pueden existir tarjetas fuera de un área. Cuando una tarjeta se desplaza y se detecta que intersecta con un área, pasa a ser contenida por ese área. Esto implica que su elemento contenedor cambia y su posición relativa se basa en el espacio dentro del área.

Al mover un área, se actualizan los valores *zIndex* tanto del área como de las tarjetas que contiene. Esto se realiza en forma de cascada para mantener la jerarquía de capas dentro del área. El valor *zIndexTop* de la actividad también se actualiza en paralelo.

De esta manera, al interactuar con un área, esta se coloca en primer plano, mostrando las tarjetas en capas superiores y actualizando el contador de capa de la actividad en consecuencia. Por ejemplo, si un área contiene cinco tarjetas, el valor de *zIndexTop* aumentará en seis unidades.

4.2.5. Puesta en marcha

Modo local

Para poder lanzar el proyecto de forma local, es necesario tener instalado *node.js* en el equipo y que el proceso del servidor back-end esté lanzado de forma paralela, ya sea en local o

en un recurso remoto.

El primer paso es instalar las dependencias del proyecto con el comando:

npm install

Posteriormente habrá que modificar el archivo de configuración */src/config/index.js* para cambiar el valor *server.url* y apuntarlo al dominio y puerto del servidor.

Una vez hecho esto, podemos ejecutar en la consola el comando:

npm start

Y nuestra aplicación se lanzará y será accesible desde el navegador, desde la ruta que indica la consola:

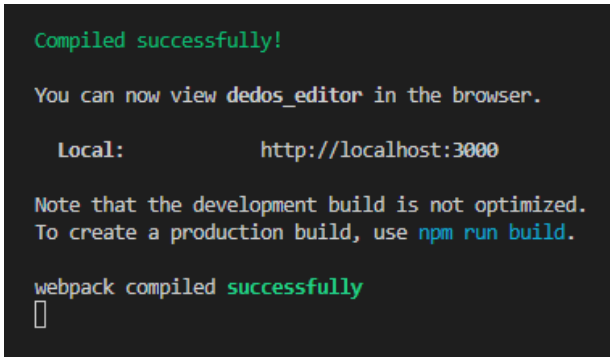

Figura 4.39: Aplicación ejecutada en local.

Si lanzamos el proyecto de esta manera, será ejecutado mediante un *daemon* de *React* que recompilará el código y actualizará el servidor de forma automática cada vez que guardemos un cambio en algún archivo del proyecto. El navegador se refrescará sólo y podremos ver los cambios realizados prácticamente en tiempo real.

Modo productivo

Si queremos lanzar el proyecto en un entorno de producción, la formula anterior no sería adecuada, ya que no está optimizada y es únicamente recomendable para fases de desarrollo local.

Tendremos que crear un artefacto de producción, transpilando el proyecto en *React* en archivos fuente minimizados de *JavaScript*. Esto se consigue con el comando:

npm run build

Una vez ejecutado satisfactoriamente podremos observar como se ha generado el archivo */build/* que contendrá todos los archivos estáticos minimizados que componen el proyecto.

Éstos archivos estáticos son los únicos que deben ser expuestos para dar servicio a la aplicación. Se podrá hacer tanto desde un servidor dedicado, un servidor tipo *Nginx* o *Apache*, un proveedor de *hosting* o un producto *PaaS* de cualquier plataforma *cloud* como *AWS*, *GCP* o *Azure*.

Servidor dedicado

El proyecto *front* cuenta con su propio servidor *node* dedicado. Se trata de un sencillo *script* que sirve los ficheros estáticos de la carpeta */build/* desde la ruta raíz /".

Se encuentra en el fichero */index.js* y para lanzarlo basta con ejecutar en el terminal:

node ./index.js

Figura 4.40: Servidor *front* dedicado.

Si accedemos a *http://localhost:3000* (nótese que 3000 es el puerto que nos indica en este caso la consola), podremos acceder, desde el navegador, a la aplicación servida por el servidor dedicado.

4.3. Arquitectura Back-end

El servidor *back-end* está diseñado como una *API REST*. Expone unos servicios sobre los que el *front-end* puede realizar una petición en cualquier momento y tras consultar en base de datos y procesar la respuesta, devuelve un paquete de datos con la información requerida en formato *JSON*.

No guarda estados, esto incluye a las sesiones de usuario, por lo que toda la información necesaria deberá estar incluida en la petición del frontal.

Las arquitecturas basadas en *Express.js* (3.2.2) se conciben como una sucesión continua de diferentes *middleware*, funciones que procesan la información, y rutas, que van guiando el flujo de ejecución por diferentes ramas de *middlewares*.

Recordemos que al trabajar con *Node.js* (3.2.1), tenemos un único hilo de ejecución para procesar todas las tareas. Esto nos obliga a utilizar técnicas de asincronía para llevar a cabo procesos que de otro modo bloquearían el hilo de ejecución, como las consultas a base de datos.

4.3.1. Autenticación y autorización

La autenticación y la autorización son esenciales para garantizar que los usuarios puedan acceder únicamente a sus recursos de forma segura. Sin ellas, cualquier persona podría acceder a información confidencial o realizar acciones maliciosas en el sistema. Al requerir que los usuarios se autentiquen antes de acceder a ciertos recursos, se pueden garantizar que solo los usuarios autorizados puedan ver o manipular sus datos personales.

Autenticación

En la presente aplicación, la autenticación se ha desarrollado con la librería *Passport* (3.2.7) que nos proporciona funcionalidades para llevar a cabo una estrategia local bajo un esquema *OAuth2*. En este tipo de estrategias, es la propia aplicación la que da acceso validando las credenciales de usuario contra la propia base de datos del sistema.

A parte de esta estrategia se pueden implementar estrategias para validar la autenticación del usuario frente a un servicio de terceros, como podría ser un control acceso gestionado con una cuenta de *Google* o *Facebook*.

Para llevar a cabo nuestra estrategia local, se ha expuesto un *endpoint* para el registro del usuario, que añade a la base de datos los datos personales requeridos del usuario y su contraseña previamente encriptada.

Por otro lado, se ha expuesto otro *endpoint* para el acceso del usuario, que acepta su identificador de usuario y su contraseña y le concede un *token JWT* que le dará la autorización necesaria para acceder al sistema.

Un *token JWT (JSON Web Token)* es una cadena de caracteres cifrada que se utiliza para transmitir de manera segura información entre dos partes, como el identificador del usuario y otros datos relevantes. Está firmado con una clave secreta que comparten ambas partes, de tal manera que se verifica la integridad del token y garantiza que no haya sido alterado durante la transmisión.

Autorización

Una vez conseguido el *token JWT* que nos da autorización, la aplicación *front-end* lo almacena en el *store* de *Redux* en el *reducer user*, para poder acceder a el siempre que lo necesite.

Como es lógico, a la pantalla de ingreso se puede acceder sin tener posesión del *token*, pero únicamente si este token existe y es válido, el usuario podrá acceder a las rutas de las páginas del menú de proyectos y el editor de proyectos. Si estando en una de estas vistas restringidas el *token* caducase, la aplicación redirigirá automáticamente al usuario a la pantalla de ingreso.

Por su parte, el *back-end* realiza también una comprobación de autorización en cada una de sus rutas salvo en las expuestas para realizar el registro y el acceso a la aplicación. Es decir, cada petición que realice el *front-end* para recuperar o modificar la información del usuario en base de datos, deberá ir acompañada en sus cabeceras por el *token JWT*.

Este *token* será validado en el *middleware* de autenticación de las rutas y se extraerá de él el identificador de usuario que será utilizado en la consulta a la base de datos. En caso de no ser un *token* válido, se devolverá como respuesta un error *401 Unauthorized*.

Cuando el *front-end* reciba esta respuesta de error, notificará el error mediante una pastilla de alerta (figura: 4.3) y devolverá al usuario a la pantalla de *login* para que introduzca de nuevo las credenciales del sistema, repitiéndose de nuevo todo el proceso que acabamos de describir.

4.3.2. Carga de imágenes

Como hemos visto, existen dos componentes en la aplicación que manejan carga de imágenes. Estos son el componente *Area* (4.1.4), que permite la carga de un *background*, y el componente *Image* (4.1.4), cuya funcionalidad principal es la carga de una colección de imágenes.

Para la gestión de la carga, se ha expuesto un servicio en el *back-end* que, gracias a la librería *Multer* (3.2.6), procesa el body de la petición *HTTP* del frontal y almacena el fichero dentro del propio proyecto en el archivo */public*, en la raíz del proyecto.

La imagen será renombrada por el servicio con un *uuid* único y será entregada como respuesta la ruta absoluta de la imagen, por la que se puede acceder al documento desde la aplicación frontal.

Para ello, se ha expuesto en el servidor la ruta */public* que permite acceder a todos los estáticos almacenados en la carpeta */public* en la raíz del proyecto.

4.3.3. Exportación del proyecto

Uno de los requisitos principales de la aplicación es dar la posibilidad de descargar el proyecto en el que el usuario esté trabajado, pulsando desde el frontal en el botón de descarga de cada uno de los proyectos presentes en el menú de proyectos.

Esta exportación debe ser idéntica a la realizada por el antiguo Editor Dedos de aplicación de escritorio, dado que deben ser totalmente compatibles.

La primera acción que se realiza dentro del controlador del servicio de descarga es la recuperación de toda la información del proyecto y actividades que lo conforman, incluyendo todos los elementos como tokens, áreas, objetivos, etc. Estos datos serán procesados y unificados dentro de una variable de texto con el formato *XML* requerido, utilizando para ello la plantilla una plantilla.

Se hará uso de la carpeta temporal */tmp*, localizada en la raíz del proyecto, para crear los documentos a exportar. En ella se irán alojando los diferentes ficheros, se creará en ella el *XML* con los datos del proyecto y se copiarán las imágenes utilizadas en las diferentes actividades.

Se utilizará la librería *Node-html-to-image* (3.2.11) para maquetar, a partir de los datos extraídos anteriormente, una miniatura de cada actividad del proyecto, que serán alojadas también en la carpeta */tmp*.

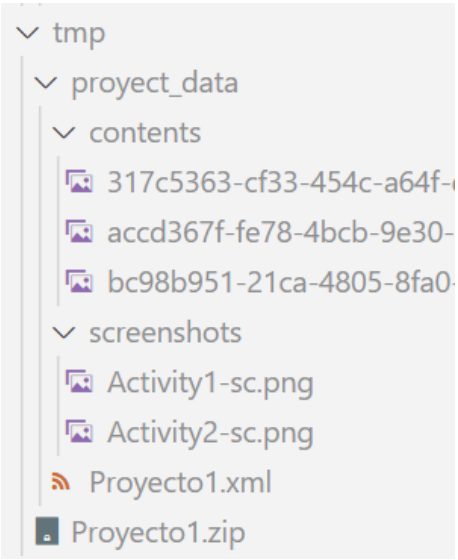

Figura 4.41: Ejemplo carpeta tmp.

Una vez generados todos los archivos que componen la exportación del proyecto, se com-

prime gracias a la librería *Archiver* (3.2.10). Y el *ZIP* nombrado como el proyecto, se descarga en el navegador del usuario.

Para simplificar el proceso, se ha optado que cada vez que se lanza este servicio, la carpeta */tmp* sea sobrescrita. De este modo no es necesario un proceso extra que elimine estos archivos temporales que se generen en su interior. Dado que si no se hiciera una gestión de ellos, esta carpeta crecería sin control pudiendo dejar nuestro equipo sin espacio de almacenamiento.

4.3.4. Puesta en marcha

Modo local

Para poder lanzar el proyecto de forma local, es necesario tener instalado tanto *Node.js* como *MongoDB* en el equipo. Además, debemos asegurarnos de que la instancia local de *MongoDB* esté ejecutándose.

El primer paso es instalar las dependencias del proyecto con el comando:

npm install

Podemos modificar el archivo de configuración */index.js* si queremos cambiar el puerto en el que se va a lanzar el servidor. Por defecto es el puerto 5000.

Una vez hecho esto, podemos ejecutar en la consola el comando:

npm start

Y nuestra servidor estará listo para escuchar peticiones *HTTP*:

```
> dedos_editor_server@1.0.0 start
> node index.js

Server starterd at port 5000
database is connected!
```

Figura 4.42: Servidor ejecutado en local.

Si lanzamos el proyecto de esta manera, será necesario reiniciar el proceso cada vez que se realice un cambio en el código para que este se vea reflejado en tiempo de ejecución.

Modo productivo

El servidor está en este punto perfectamente preparado para ser ejecutado en modo productivo, no sería necesario compilar o transiplar el código.

Podemos lanzarlo *on-premise*s desde cualquier máquina física o virtual utilizando una herramienta como *Forever* o *PM2*, dado que si se lanza como se ha expuesto anteriormente para modo local, el proceso no podría ser escalado o reiniciado al producirse un error.

También podría ser lanzado, sin necesidad de modificar el código, desde un contenedor *Docker* o desde un producto *PaaS* de cualquier plataforma *cloud* como *AWS*, *GCP* o *Azure*.

4.4. Modelo de datos

En esta sección se expondrá el modelo de datos utilizando para la gestión de la base de datos.

Recordemos que utilizamos *MongoDB* (3.2.3) que es una base de datos no relacional y por lo tanto no es necesario utilizar esquemas definidos para almacenar los datos, éstos se guardan en colecciones de objetos que podrán tener la configuración que resulte oportuna en cada momento.

Sin embargo, al trabajar sobre *MongoDB* con *Mongoose* (3.2.4), esta herramienta nos permite generar esquemas para las diferentes colecciones y poder llevar un control sistemático de la configuración de los objetos que vamos a almacenar.

Así pues, se enumeran a continuación las diferentes colecciones utilizadas en nuestra base de datos.

- **Users Schema**: Contiene toda la información personal del usuario.
- **Proyects Schema**: Contiene los metadatos de los diferentes proyectos.
- **Activities Schema**: A parte de referencias para relacionar proyectos y elementos, su único dato relevante es *zIndexTop* comentado en el apartado 4.2.4 - Gestión de capas.
- **Areas Schema**: Contiene todos los datos relativos a áreas.
- **Tokens Schema**: Contiene todos los datos relativos a las tarjetas.
- **Objetives Schema**: Contiene todos los datos relativos a los objetivos de áreas y tarjetas.

- **Files Schema**: Metadatos de las imágenes cargadas en la aplicación por parte del usuario.

En el Apéndice A se muestra el esquema del modelo de datos de forma más exhaustiva, con todas los campos de cada colección.

Capítulo 5

Experimentos y validación

En esta sección se expondrá un caso de uso en el que se podrá observar el proceso completo de un usuario nuevo se que se registra en el sistema, crea varios proyectos con diferentes actividades, las modela añadiendo componentes, objetivos, modifica sus datos, etc; hasta que finalmente, descarga la exportación de uno de los proyectos.

Daremos por hecho que se han cumplido los requisitos y pasos necesarios para tener la aplicación levantada, ya sea en modo local o productivo, como se ha descrito en los apartados 4.2.5, para el *front-end*, y 4.3.4, para el *back-end*.

5.1. Registro en la aplicación

Al acceder por primera vez a la *url* de la aplicación desde su navegador, el usuario encontrará en primera instancia el formulario de acceso.

Al ser la primer vez que accede a la aplicación, el usuario no tiene una cuenta con la que poder acceder al sistema, así que debe pulsar sobre el botón *REGISTRARSE*. Entonces, se le mostrará el formulario de registro que el usuario deberá rellenar.

Figura 5.1: Formulario de registro.

Una vez enviado este formulario con éxito, el usuario ingresará en la aplicación y se le mostrará en menú de proyectos vacío.

Si desplegamos el menú del usuario en la esquina superior derecha de la pantalla, podremos observar que efectivamente, los datos del usuario coinciden.

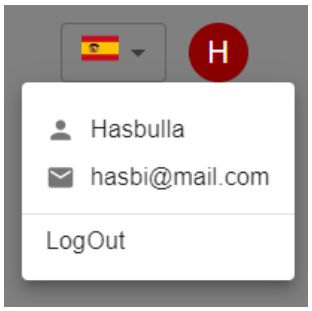

Figura 5.2: Datos del usuario.

Si el usuario hace click en *LogOut*, volverá de nuevo al formulario de acceso a la aplicación.

5.2. Creación y edición de proyectos

Cuando el usuario se encuentra en el menú de proyectos vacío, y hace click sobre el botón con el símbolo más: *Crear nuevo...*, se le desplegará el formulario de creación de proyecto.

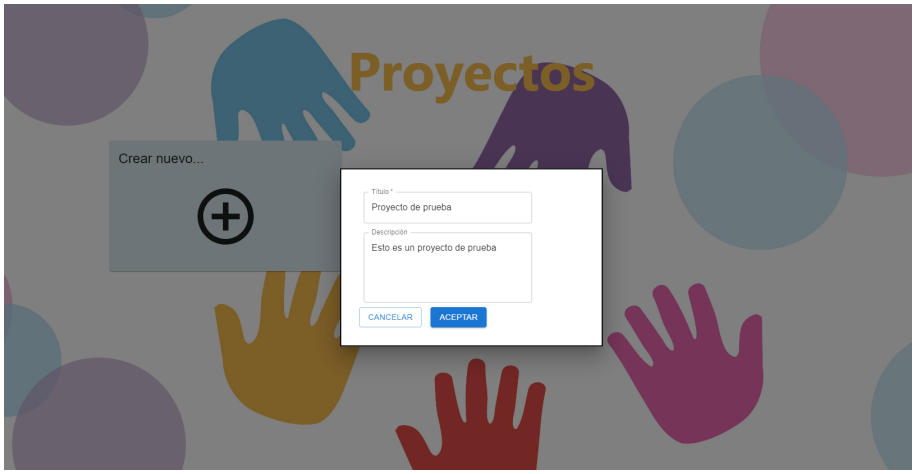

Figura 5.3: Formulario de creación de proyectos.

El usuario introduce unos textos para el título y la descripción y confirma la creación del proyecto presionando el botón *Aceptar*.

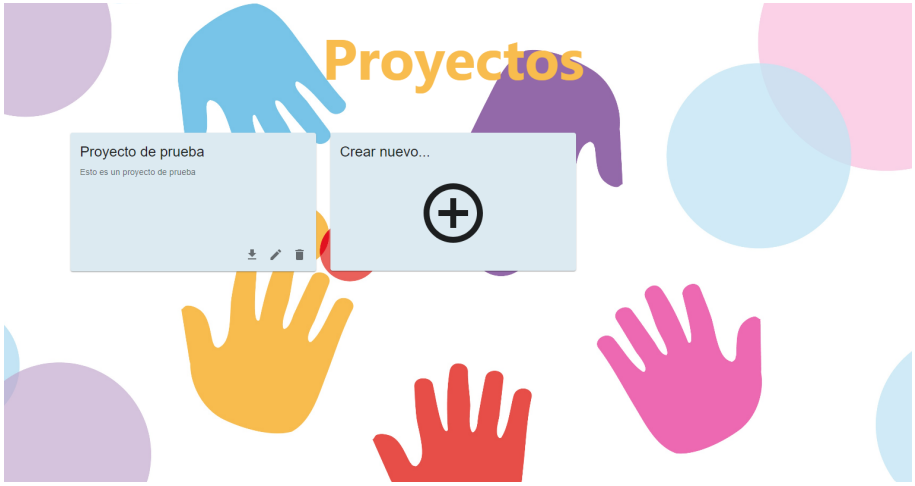

Figura 5.4: Nuevo proyecto en el menú.

En el menú de proyectos, aparecerá el proyecto que acabamos de crear, al que el usuario podrá acceder pulsando sobre el con el ratón. Esto le redirigirá la pantalla de edición de proyectos.

5.3. Cambio de idioma

Desde esta pantalla, el usuario puede cambiar de idioma en el botón incluido para tal finalidad en la esquina superior derecha (figura 4.9).

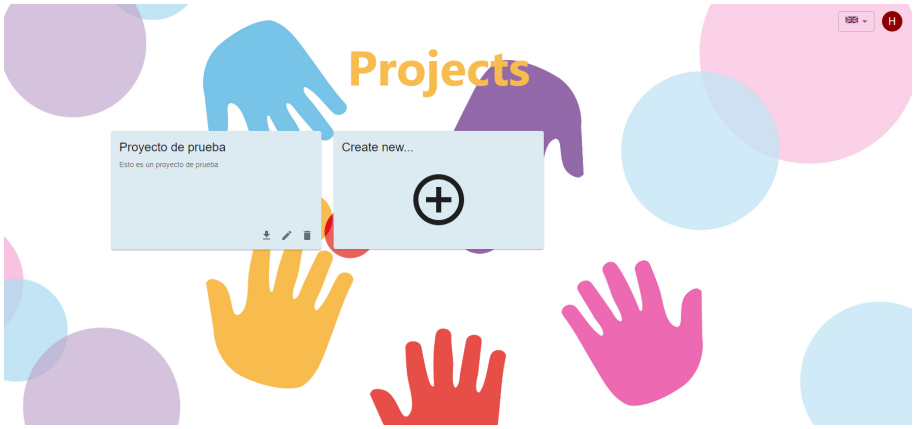

Figura 5.5: Aplicación en inglés.

Vemos que cambian los textos propios de la aplicación, sin embargo los textos introducidos del usuario jamás cambiarán.

5.4. Edición de actividades

El usuario pulsa con el ratón en el el proyecto que acaba de crear en el menú de proyectos y es redirigido a la pantalla de edición, la cual mostrará una actividad vacía.

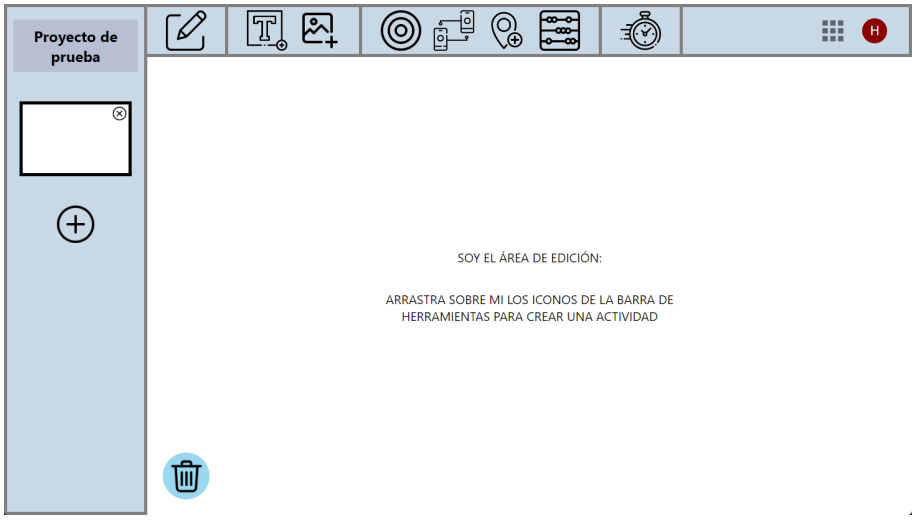

Figura 5.6: Pantalla de edición vacía.

El usuario procede a dar forma a su primera actividad empezando por la creación de un área. Para ello, arrastra el icono de área de la barra de herramientas sobre el área de edición.

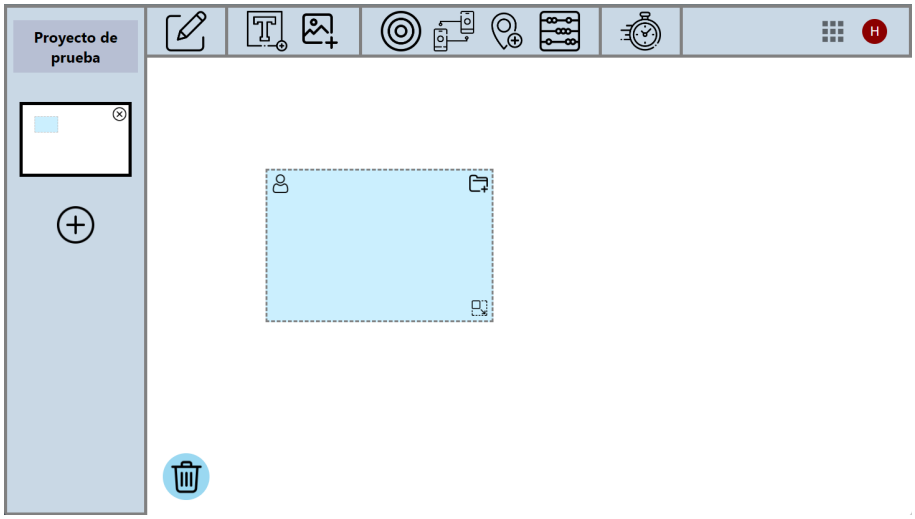

Figura 5.7: Creación de un área.

Ha aparecido un área nueva en el área de edición. Ahora el usuario la desplaza por la pantalla hasta el lugar que le parece más oportuno y lo hace más grande para que cubra mayor espacio.

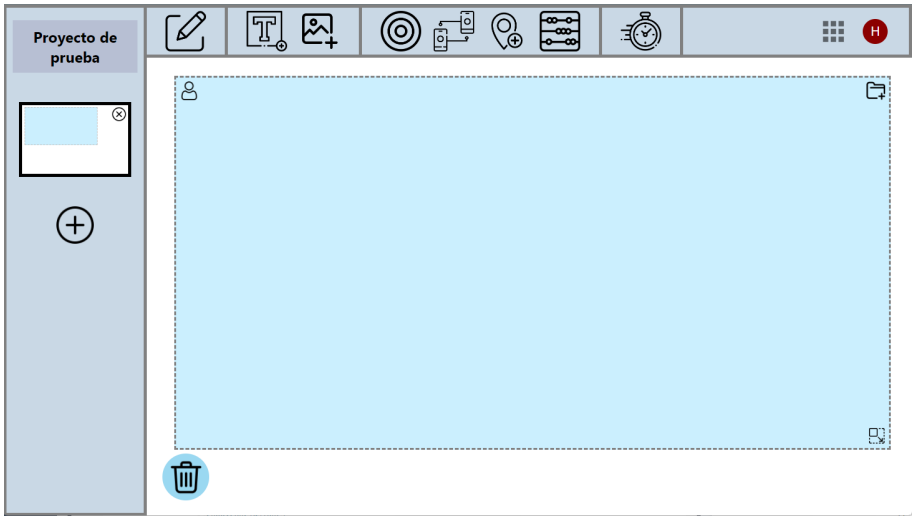

Figura 5.8: Arrastrando y redimensionando un área.

Ahora el usuario procede a crear *tokens*. Creará uno de tipo texto y otro de tipo imagen. Se repetirán los mismos procesos vistos anteriormente con el área, para desplazar y redimensionar los elementos en la pantalla, con consecuencias similares en el servidor y en la base de datos.

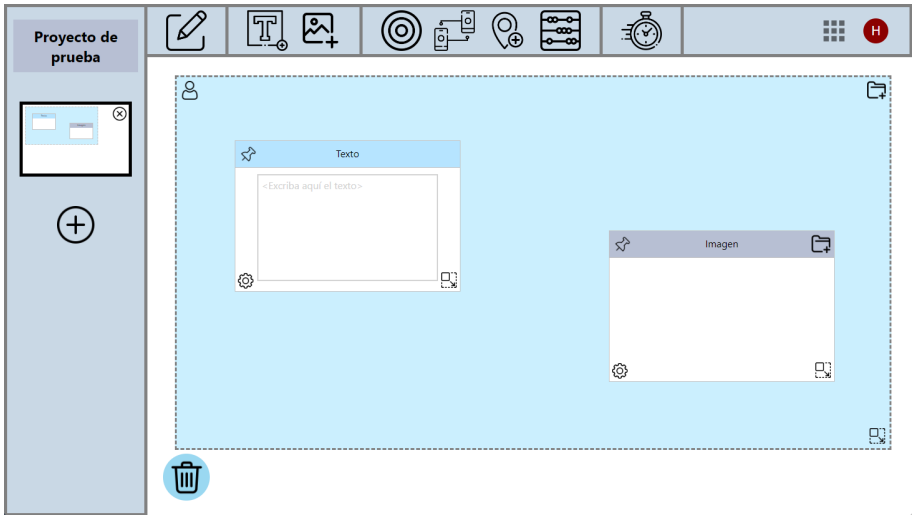

Figura 5.9: Creación de *tokens*.

El usuario quiere editar la tarjeta de imagen, añadiendo diferentes imágenes. Para ello pincha en el botón con la imagen de la carpeta, en la barra superior de la tarjeta. Se le abrirá el asistente para la carga de imágenes del navegador.

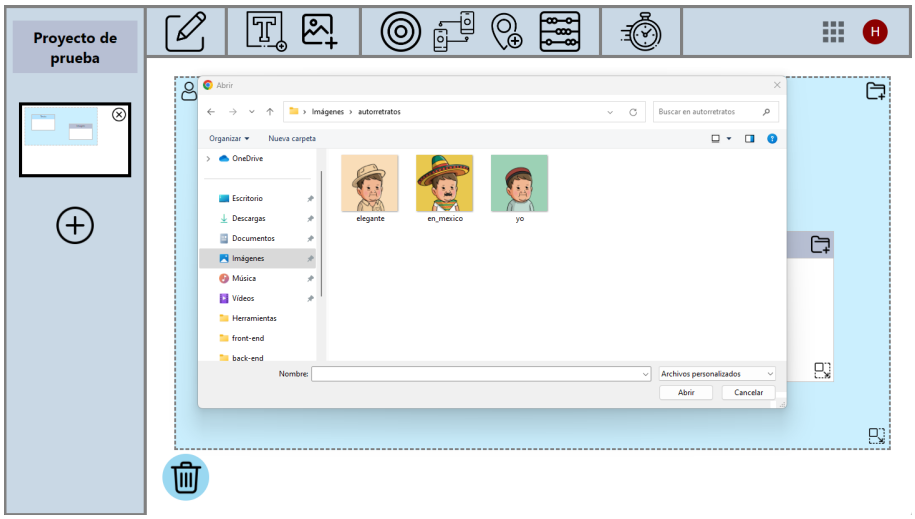

Figura 5.10: Carga de imágenes.

También modificará la tarjeta de texto añadiendo un pequeño texto a su elección.

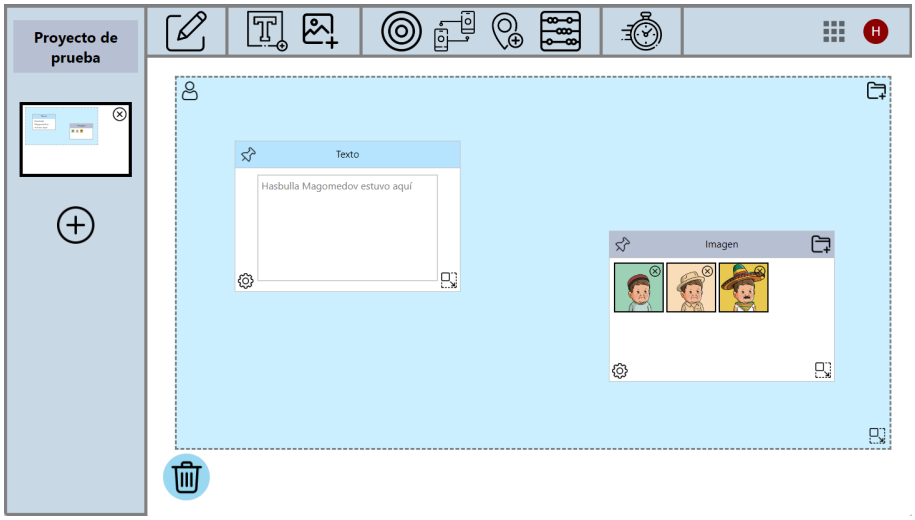

Figura 5.11: Edición de *tokens*.

Ahora, decide modificar el área cargando una imagen de fondo, el procedimiento será similar al seguido para cargar imágenes, salvo que se accionará al tocar en el icono de la carpeta del mismo área que se quiere modificar.

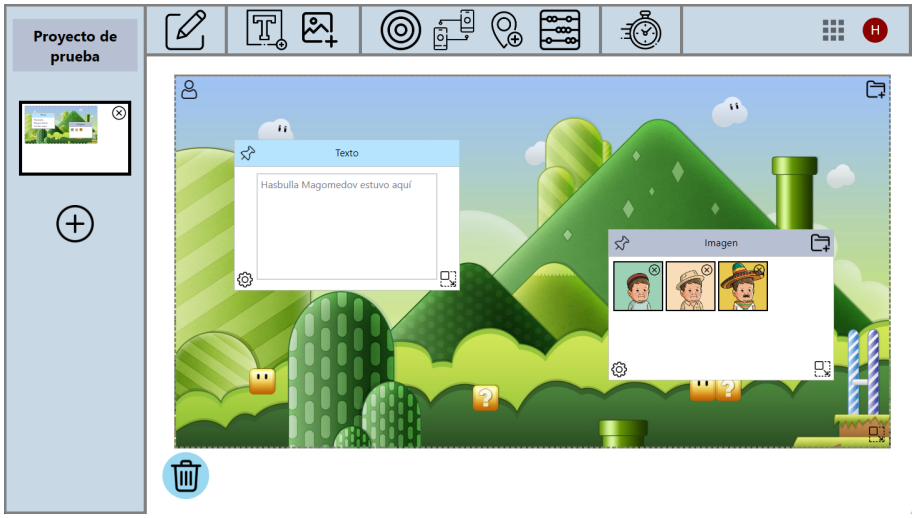

Figura 5.12: Imagen de fondo de un área.

El usuario, se dispone a crear una nueva actividad pulsando en el símbolo más del menú de actividades, a la izquierda de la pantalla. En ella creará también un área y un par de tokens, sin embargo está vez el área será de tipo ”Área de juego”. Para cambiar el tipo del área, hace click en el icono de su esquina superior izquierda.

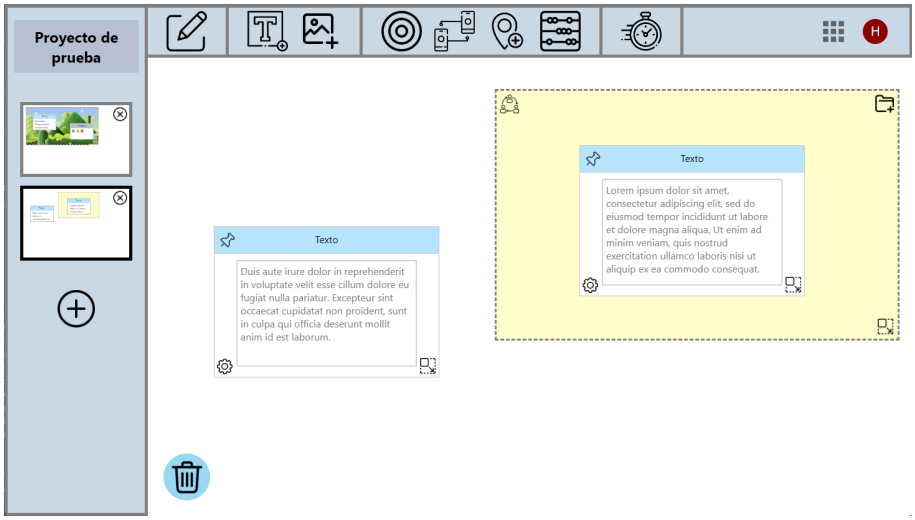

Figura 5.13: Varias actividades.

Se puede observar en el menú de actividades las dos miniaturas, que nos orientan de la configuración de cada una de las actividades. La miniatura de la actividad que se esté editando en este momento queda marcada con un borde negro mucho más pronunciado.

5.5. Añadiendo objetivos

El usuario va a añadir objetivos en las diferentes actividades. Empezará por objetivos de selección en la actividad 2, arrastrando el icono de objetivo de selección (figura 4.27) sobre cada una de las tarjetas de texto.

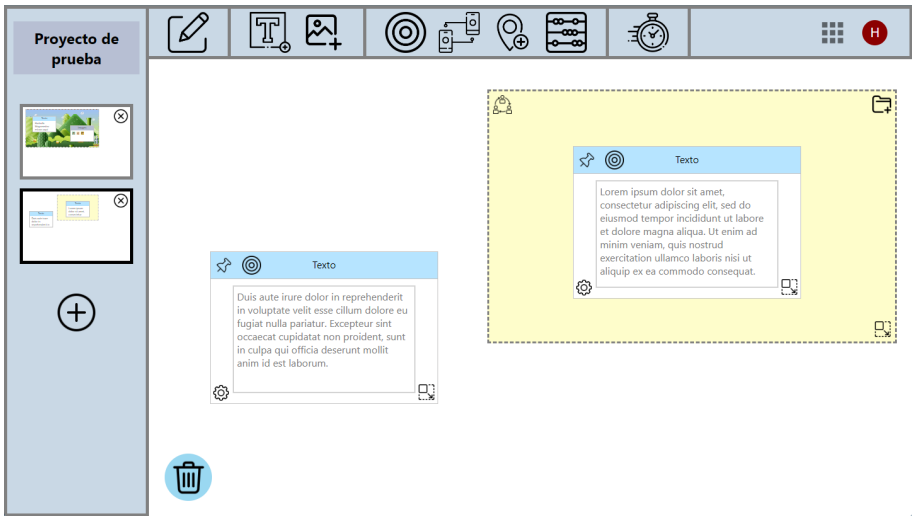

Figura 5.14: Añadiendo objetivos de selección.

Como podemos ver, el icono de objetivo de selección ha quedado asociado a ambas tarjetas en su cabecera.

Ahora, el usuario vuelve a la actividad uno y se dispone a crear un objetivo de emparejamiento entre las dos tarjetas del área. Para ello arrastra el icono de objetivo de emparejamiento (figura 4.28) sobre la primera tarjeta.

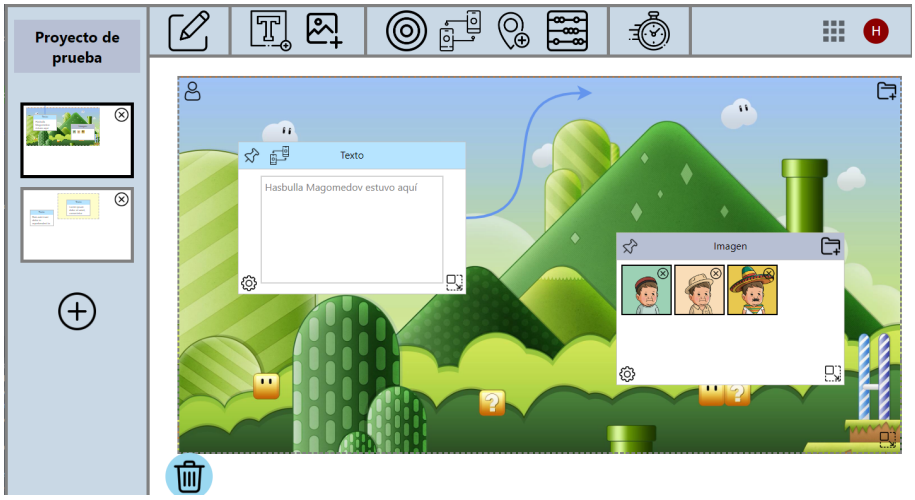

Figura 5.15: Añadiendo objetivos de emparejamiento: fase inicial.

Una vez marcada la tarjeta origen saldrá de ella una flecha que seguirá el cursor del ratón, esperando que seleccionemos la tarjeta objetivo. El usuario selecciona entonces la tarjeta de imagen y ambas quedan emparejadas.

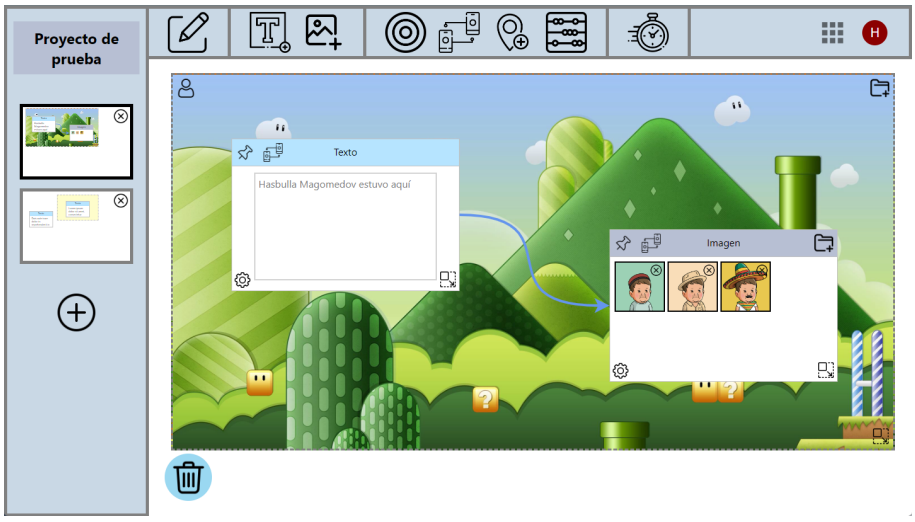

Figura 5.16: Añadiendo objetivos de emparejamiento: fase final.

El usuario mueve las diferentes tarjetas dentro del área, y se comprueba que la flecha sigue el movimiento dejando las tarjetas siempre emparejadas.

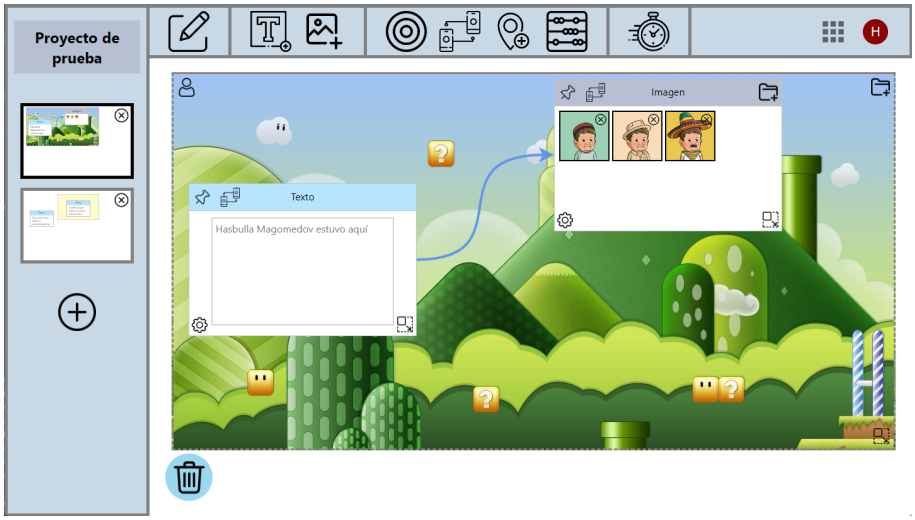

Figura 5.17: Añadiendo objetivos de emparejamiento: movimiento.

Volviendo a la actividad dos, prueba a añadir un contador de tarjetas dentro del área y un límite de tiempo, arrastrando el icono del cronómetro dentro del área de edición.

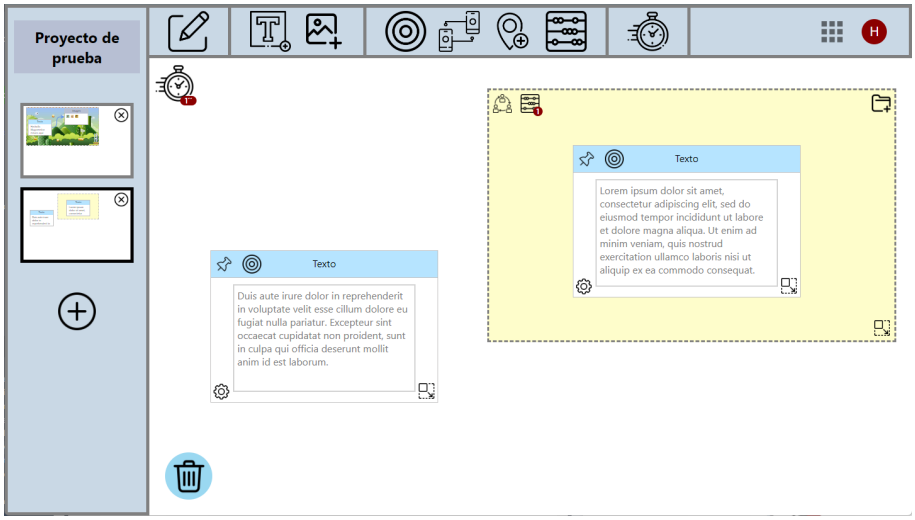

Figura 5.18: Añadiendo contadores.

Haciendo click en la burbuja roja del icono del contador de tarjetas añadido al área, el usuario modifica el valor del contador gracias al modal que aparece en pantalla.

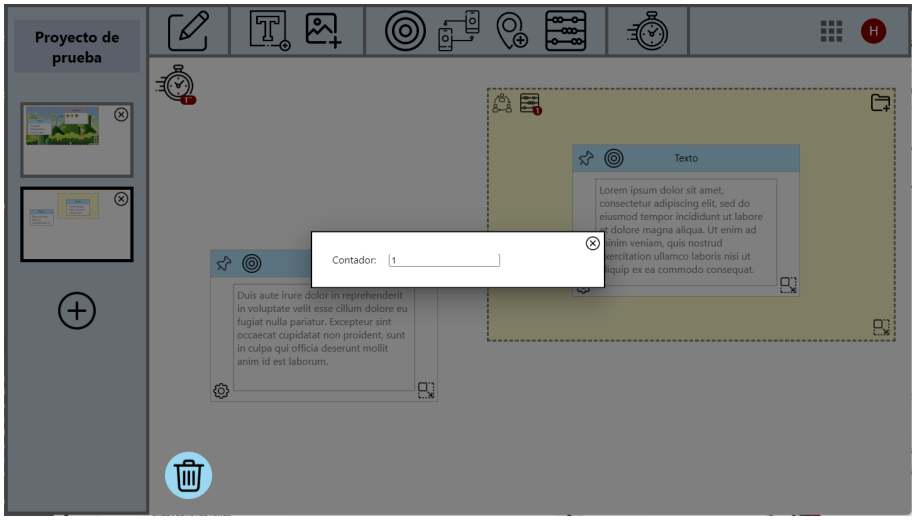

Figura 5.19: Modificando el valor de los contadores.

Realiza la misma acción sobre el icono de límite de tiempo y cambia los segundos en la modal.

Figura 5.20: Contadores modificados.

Podemos observar como los valores tanto del contador de tarjetas como del límite de tiempo han sido modificados satisfactoriamente por el usuario.

5.6. Eliminación de elementos

El usuario no se encuentra satisfecho con el resultado de la actividad dos, así que decide eliminar algunos elementos, empezando por el objetivo de selección de la tarjeta de texto de la

izquierda.

Para ello arrastra el icono del objetivo desde la cabecera de la tarjeta hasta la papelera del área de edición.

Figura 5.21: Eliminación de un objetivo.

Ahora el usuario decide borrar el área de tipo *Área de juego*, para ello lo arrastra de igual forma sobre la papelera.

Figura 5.22: Eliminación de un área.

Podemos observar, que al eliminar el área, se ha eliminado la tarjeta que contenía así como su objetivo asociado, el contador de tarjetas.

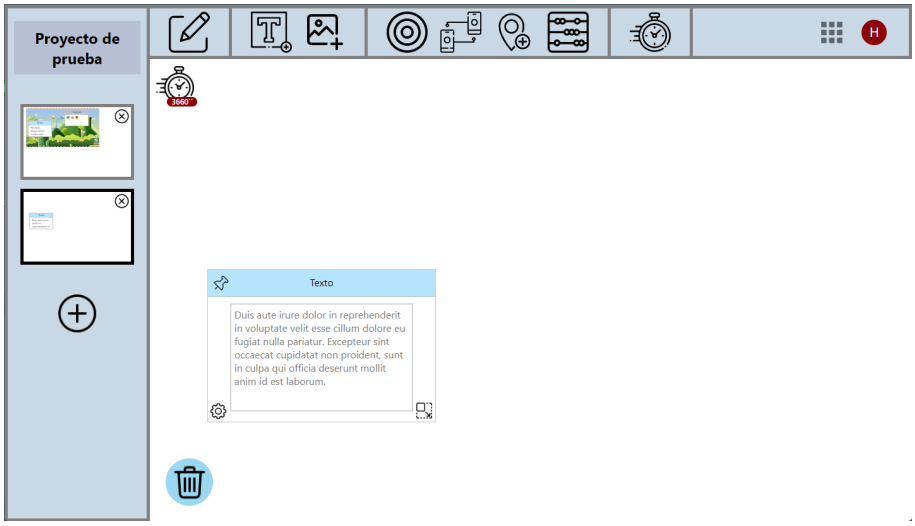

Figura 5.23: Área eliminada.

Como se puede observar, el área y todos sus elementos asociados han desaparecido tanto del área de edición como de la miniatura de la actividad.

El usuario sigue insatisfecho con como le está quedando la actividad dos y, al final, decide borrarla por completo, accionando el aspa de su miniatura.

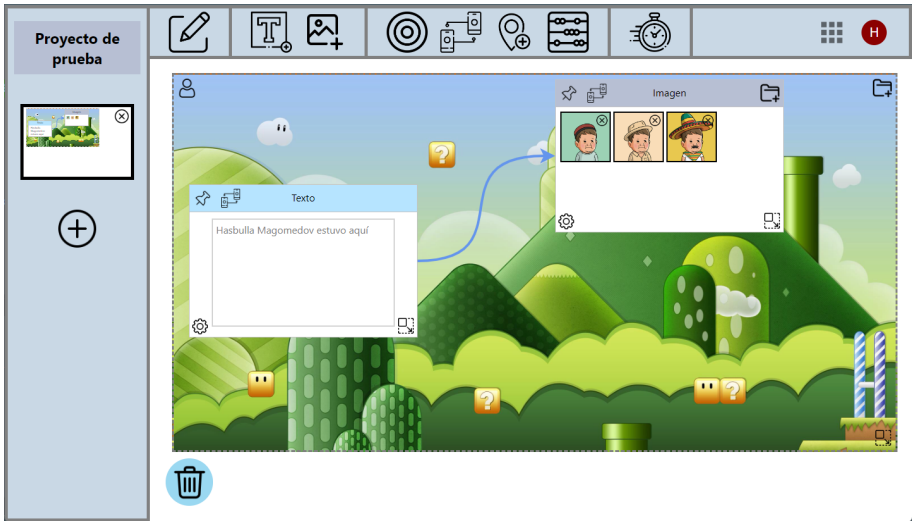

Figura 5.24: Actividad eliminada.

La actividad dos ha desaparecido y la aplicación nos posiciona automáticamente en la actividad uno.

5.7. Exportación del proyecto

Haciendo gala de su habitual indecisión, el usuario decide ahora rehacer la actividad dos tal cual estaba antes de ponerse a eliminar cosas y dar el proyecto por finalizado. En este punto, decide exportarlo.

Para ello vuelve al menú de proyectos y selecciona el icono de exportación en la caja del proyecto.

Figura 5.25: Icono de descarga.

Esta acción desencadena que el navegador muestre su asistente de selección de ubicación para la descarga. El proceso de descarga se producirá automáticamente una vez seleccionada la ubicación y nombre del archivo *ZIP*. Por defecto, el nombre será el nombre del proyecto que se está exportando.

Si inspeccionamos el archivo descargado podemos comprobar que en su interior se encuentran todos los archivos de la exportación del proyecto con la misma configuración que se exportan en la aplicación DEDOS-Editor de escritorio.

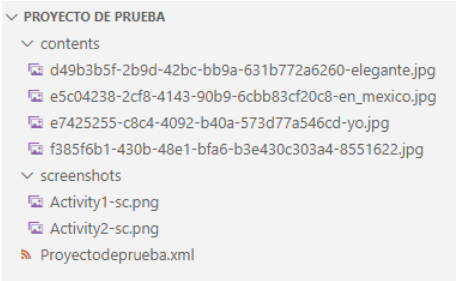

Figura 5.26: Archivos exportados.

El archivo *Proyectodeprueba.xml* con toda la información del proyecto, las imágenes cargadas por el usuario en la carpeta *contents* y las miniaturas autogeneradas por el *back-end*

correspondientes a las dos actividades en la carpeta *screenshots*.

El archivo *Proyectodeprueba.xml* se muestra íntegro al final de esta memoria, en el apéndice C.

Como prueba final definitiva, cargaremos el proyecto exportado en la antigua versión de escritorio DEDOS-Editor.

Figura 5.27: Proyecto importado en DEDOS-Editor versión de escritorio.

Como podemos ver, el resultado de la importación es satisfactorio y queda así demostrada la retrocompatibilidad de la aplicación.

Capítulo 6

Conclusiones

Una vez finalizado el desarrollo de la aplicación, podemos dar cuenta del grado de consecución de los objetivos, es decir, como se han ido desarrollando los diferentes apartados y alcanzado los hitos propuestos en el capítulo 2.

Como todo desarrollo web, éste también es susceptible de mejora y a lo largo de su desarrollo y su posterior testeo se han anotado diferentes funcionalidades para continuar con su evolución.

6.1. Conclusiones finales

A continuación se expondrá de forma breve como se ha ido desarrollando la aplicación para cumplir con los objetivos propuestos.

1. **Toma de requisitos:** Se estudiaron minuciosamente las funcionalidades que ofrece la aplicación de escritorio DEDOS-Editor y se analizó el contenido de los proyectos exportados, para reproducir su comportamiento completo en la nueva aplicación y asegurarnos de la compatibilidad de ambos sistemas. También se revisó toda la documentación aportada acerca de la herramienta DEDOS-Editor, y estudios relacionados.

2. **Creación de la plataforma web:** Se inicializó el proyecto creando una plataforma web sencilla con paginación. Permitiendo al usuario navegar entre las pantallas del menú de proyectos (véase sección 4.1.2) y el editor de proyectos (4.1.3). Permitiendo a un usuario genérico crear proyectos y actividades y que estas quedasen registradas en base de datos.

3. **Control de usuarios:** Posteriormente se desarrollo el control de acceso mediante la librería *Passport* en el servidor. Se incluyó la pantalla de login en la aplicación frontal y se modificó la base de datos y los métodos del servidor relacionados, para poder asignar registros de base de datos a usuarios concretos.

4. **Editor de actividades:** Esta parte del desarrollo está centrada en el editor de proyectos y actividades. Creando los diferentes componentes del editor, como la barra de herramientas y los elementos que se pueden añadir al área de edición como *Áreas* y *Tokens* y desarrollando los diferentes métodos para que el usuario pudiese interactuar con ellos.

   Sin duda éste fue el hito más complejo de todos, debido que para su desarrollo hubo que salvar dos importantes retos. El primero fue dotar a los diferentes elementos de la capacidad de ser arrastrados por la pantalla y de ser escalados en tamaño. Para ello hubo que hacer una importante labor de investigación para encontrar las librerías de terceros adecuadas que ofreciesen la funcionalidad necesaria. Al final se optó por utilizar dos librerías diferentes, dado que la forma de arrastrar los Tokens y los iconos de la barra de herramientas son acciones muy distintas. Además hubo que adaptar el modelo de datos para poder reflejar los cambios de posición de cada elemento respecto a la pantalla y de elementos dentro de otros elementos.

   El otro reto, derivado del primero, fue gestionar los diferentes niveles de capas de los elementos que se pueden desplazar por la pantalla y determinar un sistema para mantener la jerarquía visual. Para conocer más detalles sobre la solución, se recomienda la lectura de la sección 4.2.4. Esto implicó cierta remodelación del modelo de datos y la construcción de los servicios necesarios para la implementación de la funcionalidad.

5. **Creación de objetivos:** La asignación de los objetivos basada en *drag and drop* no estuvo exenta de complejidad, sobre todo, la gestión de las flechas del objetivo de emparejamiento (Objetivo de emparejamiento: 4.1.4), que deben seguir el desplazamiento de las tarjetas y respetar el sistema de capas.

6. **Exportación de proyectos:** Se desarrollaron las funcionalidades necesarias para que el usuario pudiese descargar en su navegador el archivo *ZIP* que contiene toda la información exportada de un proyecto objetivo, poniendo especial cuidado en la retro-

compatibilidad del *XML* con el antiguo sistema DEDOS-Editor de escritorio.

7. **Pruebas y resolución de errores:** Además de probar las funcionalidades en tiempo de desarrollo, se hicieron multitud de pruebas de usabilidad al finalizar el proyecto, con diferentes casos de uso y combinaciones de edición para comprobar que el funcionamiento de la aplicación era el deseado. La mayoría de errores que se encontraron fueron relativos al sistema de capas y la superposición de elementos desplazables del área de edición.

Tras el termino satisfactorio del desarrollo de la aplicación DEDOS-Editor Web, podemos valorarlo como un proyecto ideal para el estudio de las tecnologías web. Se utilizan tecnologías de referencia en el sector como *React*, *Node.js* y *MongoDB*; y se abordan multitud de aspectos clave, como la gestión de usuario, seguridad mediante token *JWT*, *drag and drop*, técnicas avanzadas de maquetación *CSS*, utilización de librerías de componentes, internacionalización, carga y descarga de archivos, edición de documentos y gestión del de archivos desde el servidor y renderización web en el *back-end*.

Por lo tanto, parece un proyecto idóneo para poner como ejemplo o analizarlo para profundizar en conocimientos de estas tecnologías y técnicas.

6.2. Trabajos futuros

Existen multitud de mejoras y nuevas funcionalidades que se pueden realizar a partir de este proyecto, en este apartado se sugieren algunas propuestas:

6.2.1. Evaluación de usabilidad de la herramienta

Para garantizar que los usuarios tengan una experiencia óptima es recomendable realizar una evaluación de usabilidad.

Esta evaluación implica observar y recopilar datos sobre cómo los usuarios interactúan con la aplicación, identificando posibles obstáculos, confusiones o problemas de diseño que puedan surgir. Al realizar esta evaluación, se pueden descubrir áreas de mejora y corregir cualquier problema identificado, lo que conduce a una experiencia de usuario más fluida y satisfactoria.

6.2.2. Mejorar el menú en la pantalla de edición

Una de las mejoras más inmediatas que se pueden realizar es añadir el menú de selección de idioma (figura: 4.9) a la pantalla de edición.

También se puede incluir en esta pantalla el botón para descargar el proyecto. Esto facilitaría el flujo de trabajo de manera que el usuario no tuviese que volver al menú de proyectos para descargarlo.

No obstante, hay que tener en cuenta que la barra de herramientas tendría entonces demasiados elementos y en pantallas de pequeñas dimensiones, como desde un portátil, los iconos entrarían en conflicto solapándose.

Por esto se propone una gestión responsiva del menú, de tal manera que estos elementos se emplacen en un menú expansible, similar al de usuario (figura 4.10), pero con un icono de estilo hamburguesa, en pantallas medianas y pequeñas.

6.2.3. Nuevas estrategias de login

Como se comento en la sección 4.3.1, que trataba sobre la autenticación en la aplicación, se pueden implementar de forma sencilla gracias a la librería *Passport* (3.2.7) diferentes estrategias de terceros para garantizar el acceso seguro a nuestro aplicativo. Estas pueden ser *Google* o *Facebook*.

De hecho, en el modelo de datos del usuario, se han dejado ya preparados los campos necesarios para la integración con el *login* de *Facebook*.

6.2.4. Gestión de recursos

En el apartado destinado a la carga de imágenes en el servidor (4.3.2) se explicó que todas las imágenes añadidas por los usuarios se almacenan en la carpeta *public* alojada en la carpeta raíz del proyecto.

Esto puede resultar problemático si el servidor está corriendo en una máquina con una capacidad de almacenamiento limitada, ya que puede exceder su capacidad. También puede darse un conflicto en el caso de que se lancen varias instancias del servidor en una arquitectura basada en contenedores, ya que no compartirán archivos y por tanto los recursos serán inaccesibles para otras instancias.

Por ello se recomienda extraer el alojamiento de los recursos estáticos en un repositorio común, como un *bucket S3* de *Amazon Web Services*, un *bucket* de *Google Cloud Platform* o un servicio que cumpla las mismas funciones, como un *CMS* o un servidor dedicado.

6.2.5. Concurrencia

Como se expuso en el apartado dedicado a la exportación del proyecto (4.3.3), para simplificar la solución y dado que no se espera un número elevado de usuarios conectados de forma simultanea, se optó por realizar todos los procesos relacionados con el preparativo de los archivos a exportar dentro de la carpeta */temp* alojada en la raíz del proyecto.

Esta estrategia tiene la ventaja de que cada vez que se lanza el proceso de exportar un proyecto, la carpeta */temp* se sobrescribe y no es necesario gestionar el borrado de sus datos.

Sin embargo, tiene un gran inconveniente y es que pueden existir conflictos de concurrencia si dentro de un limitado espacio de tiempo se realizan dos peticiones casi simultáneas de exportación de proyectos.

La segunda petición hará que se sobrescriba el fichero temporal antes de que la primera petición haya finalizado, ocasionando un grave error en el servicio. Esto es especialmente grave si esperamos un elevado número de peticiones por segundo.

Para evitar esto, se propone generar carpetas independientes, dentro de */temp*, para cada proyecto que se esté exportando, nombradas con un *timestamp* o identificador único. De ésta manera será imposible que las carpetas se sobrescriban y entren en conflicto.

Si se opta por llevar a cabo esta solución, hay que tener en cuenta que el fichero */temp* crecerá entonces sin control al exportar diferentes proyectos.

Para ello se propone la creación de una tarea *cron*, que cada cierto periodo de tiempo definido revise la carpeta temporal eliminando aquellos ficheros cuya vida sea superior a una fecha dada.

6.3. Aplicación de lo aprendido

La experiencia que adquirí durante los años de carrera en las asignaturas de *Sistemas Telemáticos*, *Ingeniería de Sistemas Telemáticos* y *Servicios y Aplicaciones Telemática*s resultó fundamental para desarrollar esta aplicación utilizando *React*, *Node.js* y *MongoDB*.

Durante estas asignaturas, aprendí a trabajar con tecnologías clave para el desarrollo web, como *JavaScript*, *HTML5* y *CSS*. Gracias a mis conocimientos en *JavaScript*, pude implementar la lógica y funcionalidad necesaria en la aplicación. Además, *HTML5* y *CSS* me permitieron diseñar y estilizar la interfaz de usuario, creando una experiencia visual atractiva y fácil de usar.

En cuanto a la comunicación entre el *front-end* y el servidor, las asignaturas me proporcionaron conocimientos sobre protocolos *HTTP*, así como sobre la implementación de comunicaciones seguras. Esto fue fundamental para establecer una comunicación eficiente y segura entre el *front-end* desarrollado en *React* y el *backiend* basado en *Node.js*.

Además, el estudio de bases de datos en estas asignaturas me familiarizó con conceptos fundamentales para el almacenamiento y gestión de datos. Gracias a esto, pude utilizar *MongoDB* como base de datos para mi aplicación, permitiendo un manejo eficiente y escalable de la información.

Quiero poner de manifiesto que estas asignaturas no sólo han tenido un papel importante en el desarrollo del trabajo de fin de grado, si no que han sido determinantes en mi desarrollo profesional.

Bibliografía

[1] Luis Alberto Condor Sicha. «PAGINAS WEB EDUCATIVAS Introducción, La Web en los ambientes educativos, educación y Web, ventajas y desventajas, diseño, tipos de información, herramientas para construir una página Web, aplicación». En: (2018).

[2] Carlos Fernando Meléndez Tamayo. «Plataformas virtuales como recurso para la enseñanza en la universidad: análisis, evaluación y propuesta de integración de Moodle con herramientas de la web 2.0». En: (2012).

[3] José Luis Rodrıéguez. «Herramientas de autor para el desarrollo de software educativo». En: *Comunicación, Lenguaje y Educación* 4.13 (1992), págs. 111-124.

[4] David Roldán-Álvarez et al. «DEDOS: An authoring toolkit to create educational multimedia activities for multiple devices». En: *IEEE Transactions on Learning Technologies* 11.4 (2018), págs. 493-505.

[5] Sarah Shepherd et al. «Evaluation of the Usability of DEDOS-Editor». En: (2014).

[6] Raúl Tárraga Mıénguez y Carla Colomer Diago. «Revisión de herramientas de autor para el diseño de actividades educativas.» En: *Revista Didáctica, Innovación y Multimedia, 2013, vol. 9, num. 25, p. 1-11* (2013).

[7] Marıéa Lucıéa Violini y Cecilia Verónica Sanz. «Herramientas de Autor para la creación de Objetos de Aprendizaje». En: *XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016).* 2016.

Otras *URL* de interes

- https://tv.urjc.es/series/579f2b69d68b140a378b475f
- https://es.reactjs.org/

- https://reactrouter.com/en/main
- https://es.redux.js.org/
- https://www.npmjs.com/package/redux-persist
- https://axios-http.com/es/docs/intro
- https://mui.com/material-ui/getting-started/overview/
- https://www.i18next.com/
- https://www.npmjs.com/package/react-rnd
- https://react-dnd.github.io/react-dnd/about
- https://www.npmjs.com/package/react-xarrows
- https://nodejs.org/es/docs
- https://expressjs.com/
- https://www.mongodb.com/docs/
- https://mongoosejs.com/docs/documents.html
- https://helmetjs.github.io/
- https://github.com/expressjs/multer
- https://www.passportjs.org/docs/
- https://www.npmjs.com/package/bcrypt
- https://www.npmjs.com/package/jsonwebtoken
- https://www.archiverjs.com/docs/quickstart
- https://www.npmjs.com/package/node-html-to-image
- https://code.visualstudio.com/docs
- https://www.postman.com/

- https://www.mongodb.com/products/compass
- https://git-scm.com/
- https://reactree.dev/
- https://github.com/reduxjs/redux-devtools
- https://jwt.io/

Apéndice A

Esquema del modelo de datos

A continuación se describen las diferentes colecciones y su modelo de datos que componen la base de datos *MongoDB* del proyecto:

Users Schema

- id: ObjectId,
- facebookId: String,
- name: String,
- photo: String,
- email: String,
- password: String

Se han dejado creados en el esquema algunos campos extra no utilizados en la presente versión, para desarrollar en un futuro una estrategia de autenticación basada en el sistema de autenticación de *FaceBook*.

Proyects Schema

- id: ObjectId,
- userId: ObjectId, ref: 'users',
- title: String,
- description: String,
- locale: String
- screenResolution: String,

Activities Schema

- id: ObjectId,
- projectId: ObjectId, ref: 'Projects',
- zIndexTop: Number

Areas Schema

- id: ObjectId,
- projectId: ObjectId, ref: 'Projects',
- activityId: ObjectId, ref: 'Activities',
- type: String,
- offset: Object,
- size: Object,
- background: String,
- zIndex: Number

Tokens Schema

- id: ObjectId,
- projectId: ObjectId, ref: 'Projects',
- activityId: ObjectId, ref: 'Activities',
- areaId: ObjectId, ref: 'Areas',
- type: String,
- offset: Object,
- size: Object,
- screenOffset: Object,
- clickable: Boolean,
- rotatable: Boolean,
- resizable: Boolean,
- movable: Boolean,
- mathematics: Number,
- content: Object,
- zIndex: Number

Objetives Schema

- id: ObjectId,
- projectId: ObjectId, ref: 'Projects',
- activityId: ObjectId, ref: 'Activities',
- type: String,
- origin: ObjectId, refPath: 'modelType',
- target: ObjectId, refPath: 'modelType',
- modelType: ['Token', 'Areas'],

- value: Number,

Tanto el campo *origin* como el campo *target* describen los identificadores de los componentes de la actividad que son afectados por los objetivos, pueden ser tanto *Areas* como *Tokens*. Para reflejar esta doble referencia de *ObjectId* a diferentes esquemas, *Mongoose* utiliza el campo *modelType*, que declara los esquemas que pueden ser asignados como el origen de esta referencia.

Files Schema

- id: ObjectId,
- projectId: ObjectId, ref: 'Projects',
- activityId: ObjectId, ref: 'Activities',
- containerId: ObjectId, refPath: 'modelType,
- modelType: ['Token', 'Areas'],
- fileName: String

En este caso, el campo *containerId*, que hace referencia al elemento de la actividad donde se encuentra la imagen, puede ser el identificador tanto de un *Area* como de un *Token* y, por lo tanto, se utiliza la misma estrategia *modelType*, descrita en el esquema anterior.

Apéndice B

Colección de métodos

A continuación se desglosan todos los métodos expuestos en la *API* del *back-end*.

Nótese que *http://localhost:5000* es el dominio del servidor local, si se quieren utilizar éstos métodos contra la misma *API* alojada en otro dominio, habrá que cambiar este valor.

GET Projects

http://localhost:5000/projects/

Cabeceras:

- Authorization: token

Recupera de base de datos todos los proyectos creados por el usuario.

GET Projects By Id

http://localhost:5000/projects/:proyectId

Cabeceras:

- Authorization: token

Recupera toda la información de un proyecto en concreto y su información relacionada: actividades, áreas, tokens, objetivos, etc.

POST Projects

http://localhost:5000/projects

Cabeceras:

- Authorization: token

Parámetros del body:

- title
- description
- screenResolution

Permite la creación de un nuevo proyecto vacío.

PUT Projects

http://localhost:5000/projects/:proyectId

Cabeceras:

- Authorization: token

Parámetros del body:

- title
- description
- screenResolution

Modifica la información de un proyecto concreto.

DELETE Projects

http://localhost:5000/projects/:proyectId Cabeceras:

- Authorization: token

Elimina el proyecto de base de datos y todos sus datos asociados en cascada: actividades, áreas, tokens, objetivos, etc.

POST Activities

http://localhost:5000/activities

Cabeceras:

- Authorization: token

Parámetros del body:

- projectId
- zIndexTop

Permite la creación de una actividad vacía dentro de un proyecto.

PUT Activities

http://localhost:5000/activities/:activityId

Cabeceras:

- Authorization: token

Parámetros del body:

- zIndexTop

Modifica la información de la actividad, únicamente se usa para modificar el zIndexTop.

DELETE Activities

http://localhost:5000/activities/:activityId Cabeceras:

- Authorization: token

Elimina la actividad de base de datos y todos sus datos asociados en cascada: áreas, tokens, objetivos, etc.

POST Areas

http://localhost:5000/areas

Cabeceras:

- Authorization: token

Parámetros del body:

- projectId
- activityId
- type

- offset
- size
- background
- zIndex

Permite la creación de un nuevo área dada una actividad y proyecto.

PUT Areas

http://localhost:5000/areas/:areaId

Cabeceras:

- Authorization: token

Parámetros del body:

- projectId
- activityId
- type
- offset
- size
- background
- zIndex

Modifica la información del área.

DELETE Areas

http://localhost:5000/areas/:areaId Cabeceras:

- Authorization: token

Elimina el área de base de datos y todos los tokens que contiene así como los objetivos por los que se ve afectado.

POST Tokens

http://localhost:5000/tokens

Cabeceras:

- Authorization: token

Parámetros del body:

- projectId
- activityId
- areaId
- type
- offset
- screenOffset
- size
- clickable
- rotatable
- resizable
- movable
- mathematics
- feedback
- content (text¡string¿o img¡[url de imagen]¿)
- zIndex

Permite la creación de un nuevo token, de tipo imagen o texto, dada una actividad y proyecto.

PUT Tokens

http://localhost:5000/tokens/:tokenId

Cabeceras:

- Authorization: token

Parámetros del body:

- projectId
- activityId
- areaId
- type
- offset
- screenOffset
- size
- clickable
- rotatable
- resizable
- movable
- mathematics
- feedback
- content (text¡string¿o img¡[url de imagen]¿)
- zIndex

Modifica la información del token.

DELETE Tokens

http://localhost:5000/tokens/:tokenId Cabeceras:

- Authorization: token

Elimina el token de base de datos y todos los objetivos por los que se ve afectado.

POST Objetives

http://localhost:5000/objetives

Cabeceras:

- Authorization: token

Parámetros del body:

- projectId
- activityId
- type
- origin
- target
- value

Permite la creación de un nuevo objetivo de cualquier tipo. Siempre se asociará a un elemento *target* y, dependiendo del tipo de objeto, a un elemento *origin*.

PUT Objetives

http://localhost:5000/objetives/:objetiveId

Cabeceras:

- Authorization: token

Parámetros del body:

- projectId
- activityId
- type
- origin
- target
- value

Modifica la información del objetivo.

DELETE Objetives

http://localhost:5000/objetives/:objetiveId Cabeceras:

- Authorization: token

Elimina el objetivo de base de datos.

POST Signup

http://localhost:5000/user/signup

Parámetros del body:

- email
- name
- password

Permite la creación de un nuevo usuario en base de datos. Devolverá el token *JWT* de acceso. Esta llamada no necesita ser acompañada de cabeceras de autorización.

POST Signin

http://localhost:5000/user/signin

Parámetros del body:

- email
- password

Concede tokens *JWT* de acceso a las peticiones que envían las credenciales correctas. Esta llamada no necesita ser acompañada de cabeceras de autorización.

POST Files

http://localhost:5000/files

Parámetros del body:

- projectId
- activityId
- containerId
- fileName

Permite cargar documentos de tipo imagen en el servidor y crea su relación en base datos.

DELETE Files

http://localhost:5000/files/:fileName

Parámetros del body:

- projectId
- activityId
- containerId
- fileName

Elimina un archivo dado de la base de datos y el archivo físico almacenado en el servidor.

GET Download

http://localhost:5000/download/:proyectId

Cabeceras:

- Authorization: token

Permite la exportación del proyecto en el formato *.zip* requerido por el sistema DEDOS-Editor.

Se realizará la descarga de los datos en el equipo del navegador que realizó la petición.

Apéndice C

XML exportado

A continuación se muestra un ejemplo de archivo *XML* exportado, que contiene toda la configuración e información necesaria de un proyecto. Corresponde al archivo *Proyectodeprueba.xml* de la exportación del proyecto creado en el caso de uso del capítulo 5: Experimentos y validación.

```
<Project version="3">
 <resolution x="1690" y="947"/>
 <language code="es"/>
 <title>Proyecto de prueba</title>
 <description>
  Esto es un proyecto de prueba
 </description>
 <Activity>
  <Objetives>
   <obj type="pair"
   origen="645fde47eaf81fe7923ccd10" tokenMeter="false" >
  <Targets>
     <target name="645fde49eaf81fe7923ccd24"/>
   </Targets>
   </obj>
  </Objetives>
  <TokenList>
```

```
    </TokenList>
    <AreaList>
      <Area id="645fd332eaf81fe7923ccd09" type="Player"
         numValue="undefined">
  <pos x="35" y="22" />
  <size height="593" width="1143"/>
  <rotation value="0"/>
  <posfondo x="0" y="0"/>
  <bg url="f385f6b1-430b-48e1-bfa6-b3e430c303a4-8551622.jpg"/>
  <TokenList>
    <Token id="645fde47eaf81fe7923ccd10" type="txt"
       numValue="undefined">
  <pos x="63" y="199" />
  <size height="240" width="360"/>
  <rotation value="0"/>
  <clickable>true</clickable>
  <rotatable>true</rotatable>
  <resizable>true</resizable>
  <movable>true</movable>
  <content>
    <text>Hasbulla Magomedov estuvo aqui</text>
    <feedback/>
  </content>
</Token><Token id="645fde49eaf81fe7923ccd24" type="img"
    numValue="undefined">
  <pos x="643" y="33" />
  <size height="240" width="360"/>
  <rotation value="0"/>
  <clickable>true</clickable>
  <rotatable>true</rotatable>
  <resizable>true</resizable>
  <movable>true</movable>
  <content>
```

```
    <urlList>
       <url>e7425255-c8c4-4092-b40a-573d77a546cd-yo.jpg</url>
       <url>d49b3b5f-2b9d-42bc-bb9a-631b772a6260-elegante.jpg</url>
       <url>e5c04238-2cf8-4143-90b9-6cbb83cf20c8-en_mexico.jpg</url>
     </urlList>
    <feedback/>
  </content>
</Token>
  </TokenList>
</Area>
    </AreaList>
    <Arrows>
      <arrow origin="645fde47eaf81fe7923ccd10"
         dest="645fde49eaf81fe7923ccd24" />
    </Arrows>
  </Activity><Activity>
    <Objetives>
      <obj type="sel"
      obj="645fe617eaf81fe7923ccd8d" />
    <obj type="sel"
      obj="645fe64ceaf81fe7923ccdb7" />
    <obj type="tokenMeter"
      id="645fe60eeaf81fe7923ccd7f" numValue="3" >
    <OriginTokens/>
      <OriginZones/>
      </obj>
    <obj type="time"
      time="3660" />
    </Objetives>
    <TokenList>
      <Token id="645fe64ceaf81fe7923ccdb7" type="txt"
         numValue="undefined">
  <pos x="97" y="305" />
```

```
    <size height="240" width="360"/>
    <rotation value="0"/>
    <clickable>true</clickable>
    <rotatable>true</rotatable>
    <resizable>true</resizable>
    <movable>true</movable>
    <content>
      <text>
        Duis aute irure dolor in reprehenderit in voluptate velit
           esse cillum dolore eu fugiat nulla pariatur.
        Excepteur sint occaecat cupidatat non proident, sunt in
           culpa qui officia deserunt mollit
        anim id est laborum.
      </text>
      <feedback/>
    </content>
  </Token>
    </TokenList>
    <AreaList>
      <Area id="645fe60eeaf81fe7923ccd7f" type="Game"
         numValue="undefined">
    <pos x="537" y="48" />
    <size height="397" width="644"/>
    <rotation value="0"/>
    <posfondo x="0" y="0"/>
    <bg url=""/>
    <TokenList>
      <Token id="645fe617eaf81fe7923ccd8d" type="txt"
        numValue="undefined">
    <pos x="694" y="141" />
    <size height="240" width="360"/>
    <rotation value="0"/>
    <clickable>true</clickable>
```

```
  <rotatable>true</rotatable>
  <resizable>true</resizable>
  <movable>true</movable>
  <content>
    <text>
     Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
        do eiusmod tempor incididunt ut
     labore et dolore magna aliqua. Ut enim ad minim veniam, quis
        nostrud exercitation ullamco
     laboris nisi ut aliquip ex ea commodo consequat.
    </text>
    <feedback/>
  </content>
</Token>
  </TokenList>
</Area>
    </AreaList>
    <Arrows>
    </Arrows>
  </Activity>
</Project>
```

Apéndice D

Árbol completo de componentes

A continuación se muestra el árbol de componentes completo de la arquitectura *front-end* para que se pueda apreciar su estructura.

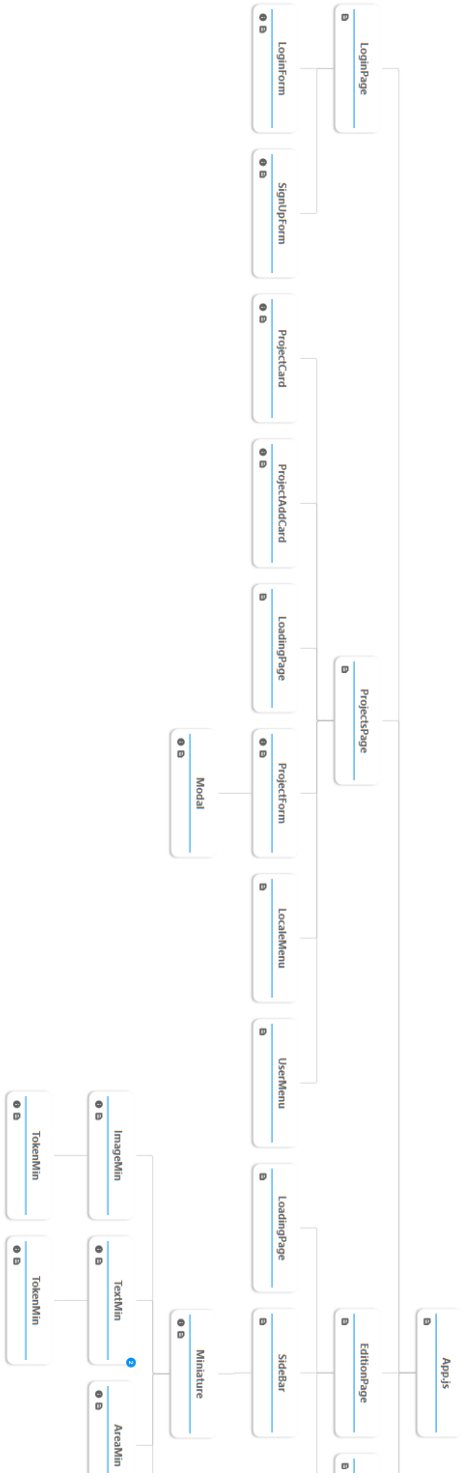

Figura D.1: Árbol de componentes 1.

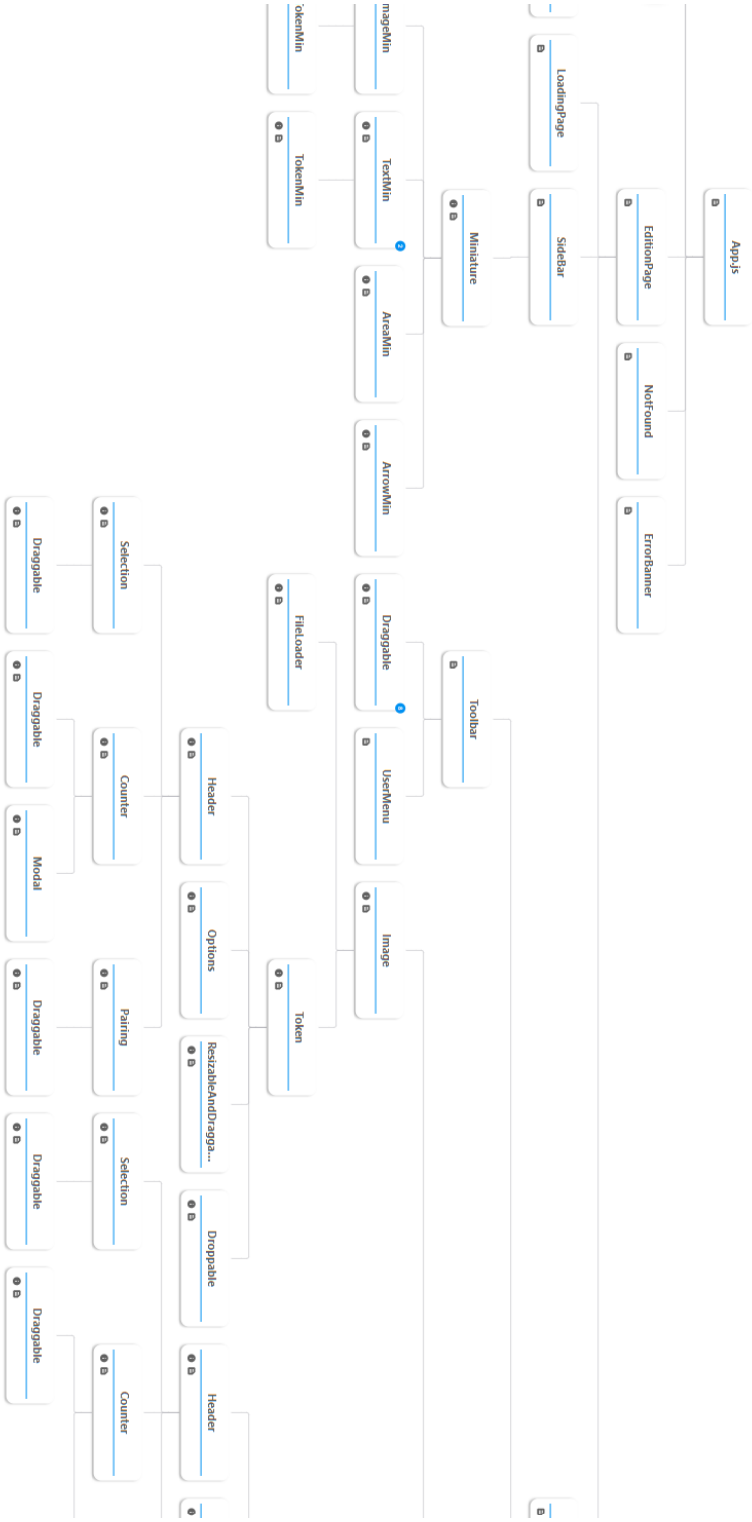

Figura D.2: Árbol de componentes 2.

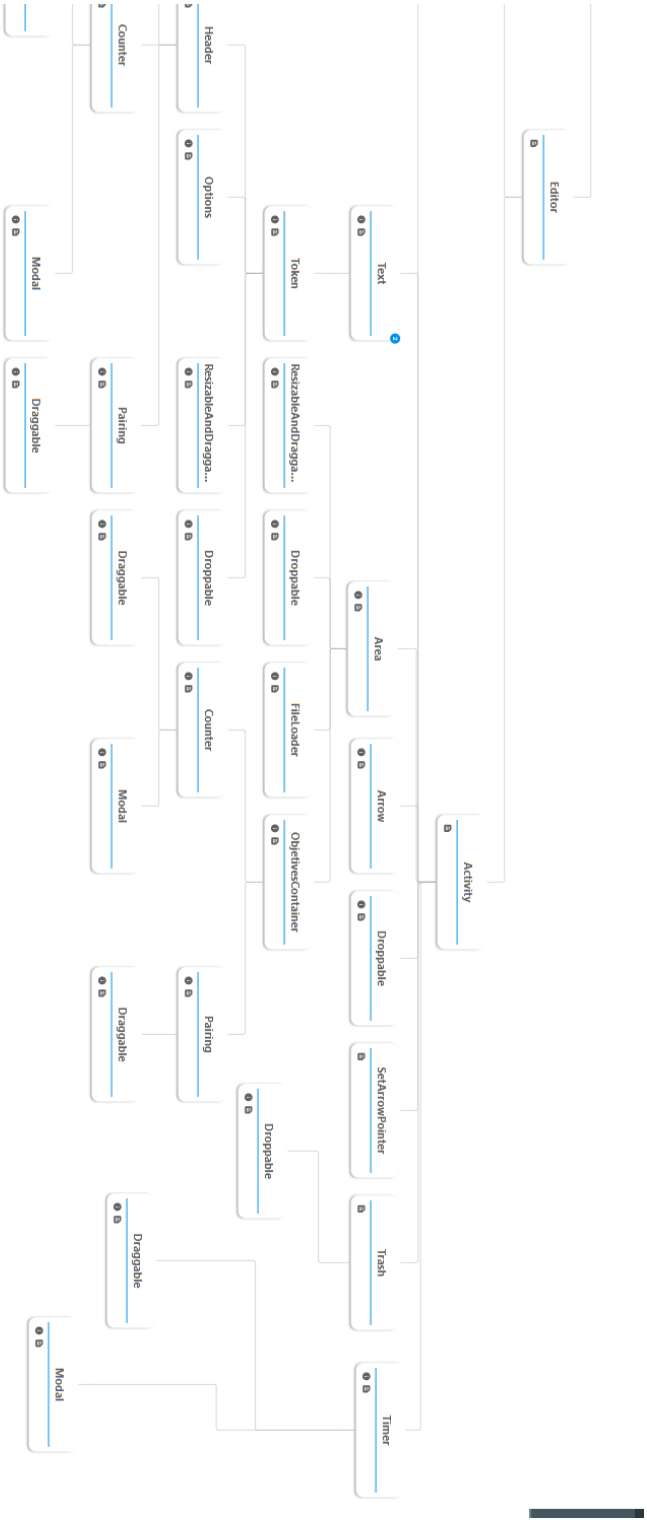

Figura D.3: Árbol de componentes 3.