UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

INGENIERÍA INFORMÁTICA

# Desarrollo de herramienta software para ayudar en el aprendizaje de la lectura

# Computer tool for learning to read in early childhood education


Realizado por
**FRANCISCO MACHUCA MUÑOZ**

Tutorizado por
BEATRIZ BARROS BLANCO

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA



MÁLAGA, septiembre de 2025

# Resumen

El presente Trabajo de Fin de Grado aborda el diseño y desarrollo de una herramienta software para educación infantil (Las Letras Que Andan), en formato de aplicación web. Es una aplicación para ayudar en el aprendizaje de la lectoescritura. El proyecto se fundamenta en el método fonético-sintético de Letrilandia, una metodología ampliamente utilizada en las aulas españolas. El software se ha desarrollado como una aplicación de página única del lado del cliente (SPA), sin el uso de frameworks, utilizando tecnologías web estándar: HTML5, CSS3 y JavaScript.

Un objetivo central de este proyecto ha sido la creación de un sistema de animación que combine la eficiencia de las animaciones CSS con la flexibilidad y el control de la lógica de JavaScript. Se optó por una estrategia híbrida, donde las animaciones de la interfaz (transiciones) son gestionadas por el navegador para un rendimiento óptimo, mientras que los movimientos de personajes se controlan de forma programática con mecanismos tipo *setInterval*. Esto permite una sincronización precisa entre el desplazamiento de los personajes y el ciclo de sus imágenes, logrando una animación fluida y coherente, un aspecto vital para mantener la atención del público infantil.

Además de las animaciones, la memoria detalla la arquitectura del software, la gestión de la carga de recursos mediante estrategias de precarga y la implementación de un diseño responsivo para garantizar la accesibilidad en múltiples dispositivos. El resultado es una aplicación que, a través de una interfaz interactiva y lúdica, se integra con una metodología educativa establecida para apoyar de manera efectiva el aprendizaje de la lectura.

**Palabras clave: Lectoescritura, Letrilandia, Educación infantil, Aplicación web, Página única del lado del cliente**

# Abstract

The present Final Degree Project addresses the design and development of an educational software tool (Las Letras Que Andan), in the format of a web application for early childhood education. This application is intended to aid in the learning of literacy. The project is based on the phonetic-synthetic method of Letrilandia, a methodology widely used in Spanish classrooms. The software has been developed as a client-side single-page application (SPA) without the use of frameworks, utilizing standard web technologies: HTML5, CSS3, and JavaScript.

A central objective of this project has been the creation of an animation system that combines the efficiency of CSS animations with the flexibility and control of JavaScript logic. A hybrid strategy was chosen, where interface animations (transitions) are managed by the browser for optimal performance, while character movements are programmatically controlled with *setInterval*-like mechanisms. This allows for precise synchronization between character displacement and their image cycles, achieving a fluid and coherent animation, a vital aspect for maintaining the attention of a young audience.

In addition to the animations, the report details the software architecture, the management of resource loading through preloading strategies, and the implementation of a responsive design to ensure accessibility on multiple devices. The result is an application that, through an interactive and playful interface, integrates with an established educational methodology to effectively support the learning of reading.

**Keywords: Literacy, Letrilandia, Early childhood education, Web application, Client-side single-page application**

# Índice

# 1 Introducción

## 1.1. Motivación

El aprendizaje de la lectura juega un papel primordial en el desarrollo cognitivo de un niño, teniendo especial importancia en las primeras etapas de crecimiento ya que es en estas en las que existe una mayor plasticidad cerebral.

Para la enseñanza de la lectura, generalmente se hace uso de dos métodos, los sintéticos y los analíticos. Los sintéticos parten de los elementos más simples para llegar a las unidades más complejas, mientras que los analíticos parten de palabras, frases o enunciados llegando a las palabras, las sílabas y las letras. Dentro de los métodos sintéticos, nos encontramos con dos grupos, los silábicos y los no silábicos. Entre los no silábicos se encuentra uno de los métodos más utilizados en España, llamado Letrilandia [Usero Alijarde(2004)].

Letrilandia es un método que se utiliza para aprender tanto la lectura como la escritura. Este parte de las letras, presentándolas a través de los personajes de un país de fantasía, contando primero un cuento sobre cada letra para después realizar actividades con esa letra.

En el proyecto PATIO (2008) se hizo una animación del método utilizando Flash, pero dado que este ya no cuenta con soporte, se encuentra obsoleto. Dada esta situación, se plantea la posibilidad de abordar un proceso de reingeniería de esa versión animada junto con nuevas funcionalidades usando JavaScript, que actualmente cuenta con un amplio soporte.

## 1.2. Contexto del TFG

El presente Trabajo de Fin de Grado se enmarca en la intersección entre la tecnología y la educación, específicamente en el desarrollo de herramientas digitales para la enseñanza de la lectoescritura. La lectura y la escritura son habilidades fundamentales que se adquieren a lo largo de un proceso continuo, influenciado no solo por la instrucción escolar, sino también por

el entorno familiar y social. Comprender cómo los niños construyen su conocimiento sobre el lenguaje escrito y los métodos pedagógicos que facilitan este aprendizaje es esencial para el diseño de una herramienta efectiva y significativa.

### 1.2.1. La enseñanza de la lectura a niños pequeños

Enseñar a leer a los niños pequeños es un proceso complejo que va más allá de la simple memorización de letras. Implica un enfoque multifacético que tiene en cuenta tanto el desarrollo cerebral como el entorno social del niño. La alfabetización es «un proceso permanente y social que se inicia en la infancia y continúa a lo largo de toda la vida», siendo el entorno y la interacción con textos y con otras personas elementos clave para su desarrollo [Junta de Andalucía(2017)].

#### La decodificación y el cerebro

«A diferencia del lenguaje hablado, la lectura no es una habilidad que el cerebro tenga programada para desarrollar. Aprender a leer requiere el trabajo coordinado de varias regiones del cerebro que corresponden a diversas habilidades cognitivas». El proceso fundamental para que un niño aprenda a leer es la fonética, que consiste en «asociar las letras impresas con los sonidos específicos que les corresponden para así decodificar las palabras». Con la práctica, esta asociación se automatiza, lo que «libera la capacidad de la memoria funcional para la comprensión lectora» [Miller(sf)].

Para los niños con dificultades, como los que tienen dislexia, esta automatización no se desarrolla con normalidad. En lugar de ello, «dependen en exceso de las áreas frontales de sus cerebros, pronunciando cada palabra una y otra vez», lo que les obliga a un esfuerzo constante para pronunciar cada palabra [Miller(sf)].

#### Enfoques pedagógicos

A lo largo de los años, han existido diferentes enfoques para la enseñanza de la lectura, algunos más efectivos que otros:

- **Enfoque del 'lenguaje integral'**: Este método, «popular en el pasado», anima a los niños a «adivinar las palabras utilizando pistas visuales o el contexto de las imágenes».

«Según los expertos, este enfoque no es eficaz, ya que desvía la atención de los niños de las letras y los sonidos, y puede perpetuar estrategias de lectura deficientes» [Miller(sf)].

- **Enfoque funcional-comunicativo**: Esta metodología propone «abordar la enseñanza del código (la fonética) en paralelo con la producción escrita en contextos que sean significativos para el niño». «Se define la lectura como un proceso activo de interpretación y comprensión, donde el lector formula hipótesis y construye el significado a partir de sus conocimientos previos» [Junta de Andalucía(2017)].

- **La enseñanza sistemática de la fonética**: «Considerado el método más eficaz, especialmente para niños con dificultades» , este enfoque enseña «de forma estructurada una progresión de habilidades fonéticas, desde los patrones más comunes hasta los más complejos». Además, se ha demostrado que la conciencia fonológica (la capacidad de reconocer los sonidos en el lenguaje, como la rima) es un buen predictor de futuros problemas de lectura, lo que permite la identificación temprana de niños en riesgo [Miller(sf)].

#### La escritura como proceso

La escritura «no es solo la caligrafía o el dominio del código; es un proceso que implica la planificación (generar ideas), la textualización (darle forma al texto) y la revisión (mejorar lo escrito)». «Los niños, a medida que aprenden, pasan por diferentes etapas de escritura, desde la presilábica hasta la ortográfica, que no están ligadas a la edad, sino a la oportunidad de interactuar y reflexionar sobre la escritura» [Junta de Andalucía(2017)].

### 1.2.2. Métodos de lectoescritura del español

La enseñanza de la lectoescritura en español se ha abordado tradicionalmente con una variedad de métodos que se pueden clasificar en dos grandes categorías: métodos sintéticos (que parten de las unidades más pequeñas del lenguaje) y métodos analíticos (que parten de unidades más grandes) [Almanza(2013)].

### Métodos sintéticos

Estos métodos se centran en enseñar las partes del lenguaje para luego unirlas. El proceso va de las unidades más pequeñas a las más grandes (letras, sílabas, palabras) [López(2017)].

- **Método fonético o fónico**: Se enfoca en el sonido de las letras (fonemas). Se enseña primero el sonido de cada vocal y luego las consonantes, para después unirlas y formar sílabas y palabras. Es un método muy directo que ayuda a la pronunciación correcta desde el principio.

- **Método silábico**: Este método se centra en la sílaba como la unidad básica. Los alumnos aprenden primero las vocales y luego las consonantes para formar las sílabas, que luego se combinan en palabras. Se apoya en ejercicios repetitivos.

### Métodos analíticos

Estos métodos se basan en un enfoque más holístico. El proceso va de las unidades más grandes a las más pequeñas (palabras, frases, oraciones) [Almanza(2013)].

- **Método de la palabra generadora**: Parte de una palabra completa y significativa para el niño ('papá', 'mamá', 'oso', etc.). La palabra se memoriza, se descompone en sílabas y luego en letras. Es un método muy popular en Latinoamérica y España.

- **Método global o de la marcha analítica**: Se basa en la idea de que los niños perciben las palabras y frases como un todo. Se utilizan carteles, fichas y juegos para que los alumnos reconozcan visualmente palabras y oraciones. La descomposición en unidades más pequeñas (sílabas y letras) se hace después.

### Métodos de lectoescritura específicos

Dentro de estas categorías, existen metodologías específicas muy populares en el ámbito hispano.

- **Método MICHO**: Este método, creado en España por Francisca García y otras autoras, se centra en un enfoque silábico-fonético [García et al.(1981)García, Sahuquillo, and Martínez].


Figura 1: Portadas de los libros del método Micho

Utiliza un libro de texto y ejercicios específicos que guían al alumno a través del aprendizaje de las letras y las sílabas (figura 1). Se caracteriza por su estructura gradual y repetitiva, que garantiza que el alumno domine cada paso antes de avanzar.

- **Letrilandia**: Es un método fonético-sintético que se presenta a través de un cuento. Fue creado por la profesora Aurora Usero. Cada letra es un personaje con una forma y un sonido que los niños aprenden a través de una historia en el 'País de las Letras'. La narrativa y la fantasía hacen que el aprendizaje sea más ameno y memorable. Se considera un método muy popular por su enfoque lúdico y creativo [Elera Seclen(2023)]. En la figura 2, se muestran las portadas de los libros con las dos versiones del método.

- **Método Palau**: El método Palau es un método fotosilábico para la enseñanza de la lectura. Fue creado por el profesor Antonio Palau. Es de enfoque sintético-silábico y utiliza un sistema de pictogramas para asociar el sonido de las letras (figura 3). A través de dibujos y esquemas, los niños conectan los fonemas con sus grafías de una forma visual. Su diseño está pensado para facilitar la memorización de las vocales y las consonantes de forma individual antes de empezar a combinarlas [Palau(1987)].


Figura 2: Portadas de los libros de Letrilandia


Figura 3: Portadas de los libros de Palau

### 1.2.3. Justificación de la elección de Letrilandia

La elección del método de lectoescritura Letrilandia para nuestra aplicación se justifica por su robusta base pedagógica y, sobretodo, por su naturaleza inherentemente adaptable a la animación digital.

A diferencia de los métodos puramente silábicos, como MICHO o Palau, que pueden llevar a los niños a 'silabear' en lugar de leer de manera seguida, Letrilandia se asemeja más a un método global al ser un proceso cercano a los cuentos. Su enfoque lúdico, basado en la personificación de las letras, fomenta la imaginación y la comprensión del lenguaje de manera más fluida. La naturaleza de los personajes facilita además la superación de algunas de las mayores dificultades del español, como la distinción fonética en la escritura, por ejemplo, en la historia de la 'h' o de la 'g' y 'j' con la vocal 'e' [Pascual(2013)].

Además de su componente narrativo y motivador (impulsado por la música y las canciones que lo acompañan), la principal ventaja de Letrilandia para este proyecto técnico es que su estructura se basa en personajes. Esta característica lo hace perfecto para ser el sujeto de una implementación animada, lo que se alinea directamente con el objetivo central del TFG de desarrollar una aplicación interactiva y visualmente atractiva utilizando tecnologías web modernas.

### 1.2.4. Trabajos previos

En este apartado se van a resumir brevemente algunos trabajos previos desarrollados en el grupo de investigación IAIA (Inteligencia Artificial Ingeniera y Apliaciones (TIC 135)), de los que parte este TFG.

El proyecto PATIO (Técnicas de aPrendizAje colaboraTivo y modelado de usuarIo aplicadas a la integraciÓn multicultural, proyecto de excelencia de la Junta de Andalucia, convocatoria 2008) se desarrolló a lo largo de tres años y dio como resultado un conjunto de aplicaciones para enseñanza individual y de grupo en educación infantil.

Uno de los resultados de PATIO fue una aplicación web 'Las Letras Que Andan (LLQA)' (figura 4) que estaba implementada en Flash [Adobe Inc.(2017)] y tenía dos tareas, 'Escribe tu nombre' y 'Construye Palabras'. Esta aplicación se utilizó para tareas del proyecto (véase por ejemplo el video de youtube con Aurora Usero 'https://www.youtube.com/watch?v=BGuNiYlDkzY')

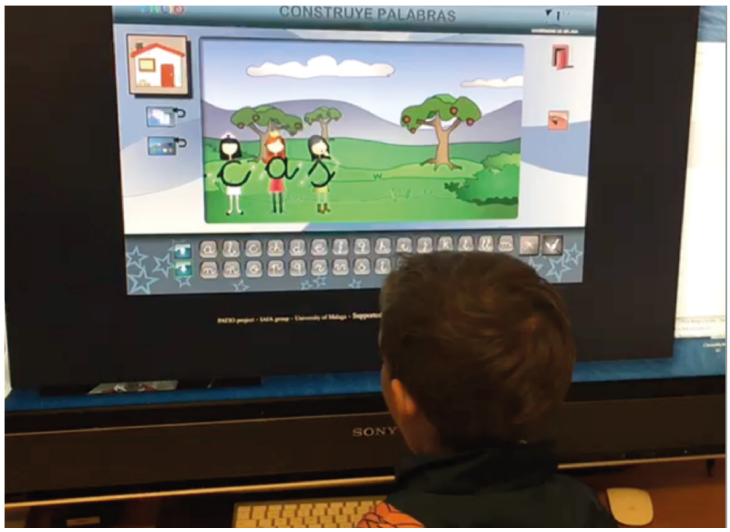


Figura 4: Interfaz de la aplicación 'Las Letras Que Andan' desarrollada en el proyecto PATIO (año 2009) en Flash

y como parte de otras tareas que precisaban de destrezas de lectoescritura que fueron parte de un Trabajo Fin de Carrera desarrollado por Noelia Jiménez Núñez [Jiménez Núñez(2009)]. Las tareas fueron las siguientes:

- Nube de Letras: El niño debe identificar una letra específica en una 'nube' de letras. Al acertar, se muestra y se reproduce una palabra que la contiene, y puede practicar la escritura de la palabra.
- Escribir: El objetivo es que el niño aprenda a trazar una letra viendo un video del fonema asociado. Puede practicar su caligrafía sobre una plantilla punteada y luego ver la imagen y escuchar el sonido de una palabra asociada a la letra.
- Comprensión de Palabras: Guiado por el profesor, el niño forma palabras, las lee y las asocia con una imagen. La actividad se centra en el análisis y síntesis de palabras, así como en la memoria visual y auditiva.
- Encadenar Palabras: El niño debe completar una palabra de dos sílabas arrastrando la sílaba correcta de una colección, ejercitando la asociación entre significado y significante.
- Emparejar Palabras: El alumno une palabras de una columna con las imágenes correspondientes en otra, reforzando la asociación significado-significante.

- Escuchar: El niño debe escuchar y leer una palabra para luego identificar la imagen que le corresponde, trabajando su comprensión auditiva y lectora.

- Ordenar Frase: El objetivo es ordenar palabras desordenadas para formar una frase coherente, apoyándose en la mayúscula y el punto para guiar la comprensión de la oración.

## 1.3. Objetivos

El software original del proyecto PATIO quedó obsoleto debido a su incompatibilidad con las tecnologías web modernas. Concretamente, el programa 'Las Letras Que Andan' se construyó con Adobe Flash, una plataforma que dejó de tener soporte y compatibilidad oficial a partir del 31 de diciembre de 2020 [Adobe Inc.(2017)]. En este contexto, el auge de JavaScript y otras tecnologías web de código abierto ofrecieron la oportunidad de reconstruir la aplicación desde cero.

El objetivo del TFG es **desarrollar una herramienta que ayude a facilitar el aprendizaje de la lectura basada en el método Letrilandia con letras animadas**. Este objetivo se puede dividir en los siguientes subobjetivos:

- **Desarrollar un sistema que resulte atractivo para los niños y que los anime a aprender a leer**. Esto implica no solo replicar la funcionalidad del proyecto original, sino también mejorar la experiencia de usuario (UX) y la interfaz de usuario (UI) para crear un entorno de aprendizaje que capte y mantenga la atención de los niños. El diseño debe ser intuitivo, visualmente atractivo y utilizar elementos interactivos que fomenten la curiosidad y la participación activa. La meta es transformar el aprendizaje de la lectoescritura en una actividad divertida y no en una tarea.

- **Implementar en JavaScript una versión animada de elementos del método de Letrilandia**. Este subobjetivo técnico se centra en la reimplementación del software en un entorno web moderno. Se busca construir una aplicación SPA (Aplicación de página única) que utilice JavaScript puro, HTML5 y CSS3 para asegurar un alto rendimiento, portabilidad y accesibilidad. La implementación debe ser lo suficientemente robusta para replicar las animaciones, interacciones y el contenido educativo del proyecto original, garantizando que la aplicación pueda funcionar de manera óptima en diferentes dispo-

sitivos y navegadores, y que no dependa de complementos o software de terceros para su ejecución.

## 1.4. Tecnologías utilizadas

El desarrollo de cualquier proyecto web moderno se asienta sobre un pilar fundamental: la trinidad de HTML, CSS y JavaScript (figura 5). Cada una de estas tecnologías cumple un rol indispensable, y su combinación armónica es lo que da vida y funcionalidad a las páginas que utilizamos a diario. Se podría pensar en HTML como el esqueleto, CSS como la piel y la ropa que le da estilo, y JavaScript como el sistema nervioso que le permite interactuar con el mundo. Juntas, estas tres tecnologías forman el estándar de facto para la creación de experiencias en línea, desde simples sitios estáticos hasta complejas aplicaciones web interactivas.


Figura 5: Tecnologías utilizadas: 'HTML, CSS, JS'

El primero de estos pilares, HTML (HyperText Markup Language), es el lenguaje de marcado que define la estructura y el contenido de una página web. A través de etiquetas y elementos, como <h1>para encabezados o <p>para párrafos, se organizan los textos, imágenes, videos y enlaces que componen la información que se muestra al usuario. Es un lenguaje sencillo pero

poderoso, ya que establece la base semántica y la jerarquía de la información, permitiendo a los navegadores entender qué es cada parte del contenido. Sin HTML, una página web sería un simple bloque de texto sin forma ni significado, incapaz de ser interpretado correctamente por los navegadores.

Por último, CSS (Cascading Style Sheets) y JavaScript entran en juego para llevar esa estructura básica al siguiente nivel. CSS se encarga de la estética: el diseño, los colores, las tipografías, los márgenes y la disposición de los elementos en la pantalla. Permite separar el contenido de su presentación, haciendo que los sitios web sean más fáciles de mantener y más adaptables a diferentes dispositivos. Por su parte, JavaScript es el cerebro detrás de la interactividad. Es un lenguaje de programación que permite manipular el contenido, cambiar los estilos de una página en tiempo real, validar formularios, crear animaciones y responder a las acciones del usuario. Con JavaScript, un sitio web deja de ser estático y se convierte en una experiencia dinámica y viva.

## 1.5. Estructura del documento

El presente Trabajo de Fin de Grado ha sido estructurado para ofrecer una lectura clara y secuencial, presentando el proyecto desde su concepción inicial hasta las conclusiones finales. La organización en seis capítulos tiene como objetivo guiar al lector a través de cada fase del desarrollo.

1. **Introducción**: Este capítulo establece el contexto del proyecto, presentando su motivación y los objetivos que se pretenden alcanzar. Aborda la problemática de la enseñanza de la lectoescritura en la etapa de educación infantil y justifica la elección del método Letrilandia como base pedagógica. Además, introduce las tecnologías utilizadas y delimita el alcance del trabajo.

2. **Teoría y diseño de animaciones en la web**: Este capítulo aborda el marco teórico que sustenta la creación de la aplicación. Se explora la diferencia entre la animación basada en CSS y la basada en JavaScript, se justifican los principios de diseño de aplicaciones para niños y se describen los mecanismos de animación específicos implementados en el proyecto.

3. **Análisis de requisitos y arquitectura**: Este capítulo detalla los requisitos funcionales y no funcionales que guían el desarrollo de la aplicación. Se describe la arquitectura de software elegida, justificando la implementación como una aplicación de página única (SPA) y la decisión de no emplear frameworks, para mantener el control completo sobre la estructura del código.

4. **Interfaz e interacción con el usuario**: Este capítulo describe el diseño de la interfaz de usuario de la aplicación, haciendo especial énfasis en la experiencia de usuario. Se abordan las decisiones de diseño tomadas para el público infantil, como la usabilidad y la retroalimentación, y se detallan las interacciones clave de los elementos de la interfaz.

5. **Ciclos de desarrollo**: Este capítulo describe la metodología de desarrollo del proyecto. Se detallan las fases iterativas de diseño, prototipado, implementación y pruebas. Se explica la justificación de cada ciclo y se presentan los resultados y conclusiones extraídos de las pruebas con usuarios finales.

6. **Conclusiones**: Este capítulo final resume los hallazgos clave del proyecto, evaluando el cumplimiento de los objetivos planteados inicialmente. Además, se reflexiona sobre el proceso de desarrollo y se proponen posibles mejoras para futuras iteraciones de la herramienta.

# 2

# Teoría y diseño de animaciones en la web

La evolución del desarrollo web ha transformado la forma en que interactuamos con las aplicaciones. Atrás quedaron las páginas estáticas y las animaciones restrictivas de tecnologías propietarias como Flash. Hoy en día, el desarrollo web moderno se apoya en los estándares de HTML5, CSS3 y JavaScript, que no solo permiten una mayor interactividad, sino que también garantizan la portabilidad, el rendimiento y la accesibilidad. Este capítulo establece los fundamentos teóricos que sustentan la creación de la aplicación, explorando los fundamentos de la animación web, los principios de diseño específicos para el público infantil y los mecanismos técnicos que se han empleado para dar vida a los personajes de Letrilandia.

## 2.1. Fundamentos de la animación web

La animación en la web se puede lograr a través de dos enfoques principales: las transiciones y animaciones de CSS, y la manipulación del DOM con JavaScript. La elección entre ambos depende de la complejidad de la animación, el rendimiento deseado y la naturaleza del proyecto [Aguado(2017)].

### 2.1.1. Animaciones basadas en CSS

Las animaciones y transiciones de CSS son la opción preferida para la mayoría de los movimientos de interfaz de usuario. Su principal ventaja reside en su rendimiento optimizado, ya que el navegador puede delegar su ejecución directamente a la GPU (Unidad de Procesamiento Gráfico). Esto las hace fluidas y eficientes, sin sobrecargar el hilo principal del navegador.

- **Transiciones**: Permiten la interpolación de propiedades CSS de un estado a otro de forma suave. Se utilizan para cambios simples, como el efecto de hover de un botón o la

aparición de un elemento en pantalla.

- **@keyframes y animaciones**: Ofrecen un control más granular, permitiendo definir secuencias de animación complejas con múltiples 'fotogramas' intermedios. Se utilizan para movimientos más elaborados, como el ciclo de un personaje saltando o caminando, o una transición completa de la pantalla de bienvenida.

### 2.1.2. Animaciones basadas en JavaScript

Aunque las animaciones CSS son ideales para muchas tareas, JavaScript ofrece la flexibilidad y el control necesarios para animaciones más complejas y con lógica. En el contexto de esta aplicación, JavaScript es indispensable para:

- **Coordinación de múltiples animaciones**: Cuando un evento del usuario debe desencadenar una secuencia de animaciones en diferentes elementos.
- **Lógica interactiva**: Para que las animaciones respondan a datos variables, como la puntuación del usuario o el progreso en el juego.
- **Animaciones basadas en setInterval**: Permite un control preciso sobre la secuencia de fotogramas, lo cual es útil para animaciones que dependen de cálculos complejos en tiempo real. Esta es la técnica utilizada en el proyecto para las animaciones de las letras.

En el contexto de esta aplicación, se optó por una estrategia híbrida, donde JavaScript actúa como el 'director de orquesta'. Para las animaciones de la interfaz (como el resaltado de botones), manipula clases CSS para que el navegador ejecute las animaciones predefinidas, garantizando el mejor rendimiento posible al aprovechar la aceleración por hardware del CSS. Para las animaciones de personajes y sus movimientos continuos, JavaScript se encarga de cambiar la posición y la imagen de los elementos de forma periódica con setInterval, manteniendo en este caso la lógica del juego y la animación en un solo lugar.

## 2.2. Principios de diseño para aplicaciones infantiles

Una aplicación educativa para niños debe ser mucho más que una simple herramienta, debe ser una experiencia atractiva y segura [Morales(2011)].

- **Diseño lúdico e interactivo**: El diseño debe ser intuitivo y estar orientado al juego. Esto incluye el uso de colores vibrantes, personajes amigables y una tipografía legible que se adapte al público infantil. La interfaz debe ser simple, con botones y elementos visuales claros que eviten la frustración y fomenten la exploración.

- **Feedback visual y sonoro**: Los niños aprenden a través de la causa y el efecto. Las animaciones y los sonidos deben proporcionar una retroalimentación inmediata a cada acción del usuario. Un botón que se ilumina o un sonido de acierto al completar una tarea refuerza el aprendizaje y la motivación.

- **Accesibilidad visual**: Para garantizar la inclusión, es crucial utilizar un lenguaje visual que sea comprensible para todos los niños. El uso de los pictogramas ARASAAC [Gobierno de Aragón(2024)] en los botones de la aplicación proporciona una comunicación universal, permitiendo que incluso los usuarios con barreras de lectoescritura o necesidades especiales puedan navegar por la interfaz de forma independiente.

- **Narrativa y personajes**: Un diseño basado en personajes y una narrativa atractiva convierte la experiencia de aprendizaje en una aventura. En este caso, la personificación de las letras de Letrilandia en personajes animados hace que el proceso de aprendizaje sea más memorable y emocionante.

## 2.3. Mecanismos de animación y optimización en el proyecto

La arquitectura de la aplicación está diseñada para ser eficiente y modular, con una clara separación entre estructura (index.html), presentación (styles.css) y lógica (app.js).

- **Animación por manipulación de clases CSS**: Este es el mecanismo principal para las animaciones de la interfaz. La lógica de la aplicación en app.js detecta un evento (como un clic de ratón) y añade o elimina una clase CSS de un elemento HTML. El styles.css contiene las definiciones de animación preestablecidas para esas clases, lo que permite que el navegador ejecute la animación de forma optimizada.

- **Animación por secuencia de imágenes (setInterval)**: Para movimientos complejos, como el ciclo de caminata de las letras, se utilizan secuencias de imágenes. En este caso,

JavaScript cambia la fuente de la imagen (src) de un elemento a una velocidad constante, creando una ilusión de movimiento controlada por un temporizador setInterval.

- **Desplazamiento de personajes (setInterval con left)**: El desplazamiento de las letras a través de la pantalla se gestiona de forma programática. Se utiliza un setInterval que actualiza la propiedad left del elemento de la letra. En cada intervalo de tiempo, se modifica la posición en un número fijo de píxeles, lo que genera un desplazamiento constante y controlado.

- **Posicionamiento dinámico y redimensionamiento**: Para asegurar que la aplicación sea responsiva y funcione bien en diferentes dispositivos, se utiliza un sistema de posicionamiento dinámico. Por un lado, los elementos estáticos de la interfaz como los botones, cuentan con propiedades en CSS que determinan tanto sus dimensiones como su posición, de manera relativa a la pantalla. En el caso de las letras animadas, en app.js se calculan las coordenadas y el tamaño de las letras en relación a la pantalla y a la longitud de la palabra que se va a escribir, para que se adapten correctamente a cualquier posible redimensionamiento de la pantalla, lo que garantiza que la disposición se mantenga en cualquier dispositivo.

## 2.4. Ejemplos prácticos de animación

Para ilustrar con mayor detalle la implementación de la animación y la gestión de la interfaz de usuario, se presentan a continuación ejemplos prácticos específicos del proyecto.

### 2.4.1. Animación de botones y elementos de la interfaz

La aplicación utiliza animaciones sutiles para guiar al usuario y proporcionar retroalimentación visual, especialmente durante la explicación de las pantallas de juego.

- **Resaltado durante la explicación**: En las pantallas de 'Componer Palabra' y 'Componer Nombre', se usan animaciones para dirigir la atención del usuario a los botones clave mientras la guía vocal explica su función. La lógica en app.js gestiona una secuencia de eventos. Primero, se reproduce una explicación sonora. Mientras tanto, JavaScript añade y retira una clase CSS, por ejemplo, .highlight, al botón correspondiente. Esta clase está

definida en styles.css con una animación que modifica la opacidad, la escala, o incluso el posicionamiento, creando un efecto que llama la atención sin ser intrusivo.

- **Aparición y resaltado al pasar el ratón**: La visibilidad de los botones se controla con lógica de JavaScript y transiciones CSS. Un botón puede estar inicialmente oculto con opacity: 0 y pointer-events: none. JavaScript le asigna la clase :visible que cambia su opacidad a 1 y activa una transición, haciéndolo aparecer de forma suave. Además, para la interacción con el usuario, se utiliza el pseudo-selector :hover en CSS, que aplica una ligera transformación de escala o color cuando el cursor se sitúa sobre el botón. Este efecto intuitivo refuerza la sensación de interactividad.

- **Estado deshabilitado y pulsado**: Los botones tienen diferentes estados visuales controlados con clases CSS. Cuando un botón se deshabilita, JavaScript añade una clase :disabled que reduce su opacidad y modifica el cursor a not-allowed, comunicando visualmente que no se puede interactuar con él. Al ser pulsado, se le añade una clase :active que puede aplicar un cambio de color o de escala para dar una retroalimentación instantánea de que la acción ha sido reconocida.

- **Aparición de las barras superior e inferior**: La lógica de la aplicación en app.js se encarga de controlar la aparición de la barra superior e inferior. Cuando se carga una nueva pantalla, se activa una transición que hace que la barra superior descienda suavemente desde la parte superior de la pantalla, mientras que la barra inferior asciende desde abajo. Este efecto se consigue con transiciones CSS predefinidas en styles.css que modifican las propiedades top y bottom de los elementos. El uso de transiciones CSS para este tipo de animaciones de una sola vez garantiza un alto rendimiento, ya que la animación es gestionada por el navegador de forma optimizada.

### 2.4.2. Ejemplo de animación de una letra 'andando'

Para ilustrar el funcionamiento de una animación continua, como la de un personaje de letra que camina, se utiliza una combinación de setInterval y la manipulación de la imagen y la posición.

- **Activación**: La animación se inicia al llamar a una función de JavaScript, por ejemplo

animarNuevaImagen(). Esta función configura dos temporizadores setInterval: uno para el desplazamiento y otro para el ciclo de la imagen.

- **Desplazamiento**: Un setInterval se encarga de actualizar la posición del elemento de la letra. Cada 17 milisegundos (FREC_MOVIMIENTO_LETRAS), la función añade 4 píxeles (PIXELES_POR_FREC_MOVIMIENTO_LETRAS) a la propiedad left del estilo del elemento, hasta que llega a la posición destino. Esto crea un desplazamiento constante, simulando un desplazamiento por la pantalla. En la figura 6 podemos ver un ejemplo.

```
// Mover la imagen hacia la derecha PIXELES_POR_FREC_MOVIMIENTO_LETRAS
// pixeles cada FREC_MOVIMIENTO_LETRAS milisegundos
const moverInterval = setInterval(() => {
  // Pixeles que se mueven las letras por cada intervalo de frecuencia
  posicionHorizontal += PIXELES_POR_FREC_MOVIMIENTO_LETRAS;
  imagen.style.left = posicionHorizontal + "px";

  // Comprobar si la imagen ha salido de la pantalla
  if (posicionHorizontal > window.innerWidth) {
    clearInterval(moverInterval);
    clearInterval(cambioInterval);
    imagen.remove(); // Eliminar la imagen del DOM
    resolve(); // La animación ha terminado
  }
}, FREC_MOVIMIENTO_LETRAS); // Frecuencia de movimiento de la letra
```

Figura 6: Ejemplo de setInterval para desplazamiento en animarNegacionYSalida()

- **Ciclo de la imagen**: Simultáneamente, un segundo setInterval cambia la imagen del personaje a una velocidad de 125 milisegundos (FREC_CAMBIO_IMAGEN_LETRAS). Por ejemplo, letra.src = 'Letras/M1.png', luego letra.src = 'Letras/M2.png', y así sucesivamente, creando la ilusión de que el personaje está en movimiento mientras se desplaza. El temporizador reinicia el ciclo cuando llega al final de la secuencia de imágenes. En la figura 7 podemos ver un ejemplo.

```
// Obtener el array de imágenes correspondiente a la letra
const arrayImagenes = getImagenesPorLetra(letra);
let imagenIndice = 0;

// Cambiar la imagen cada cierto tiempo para dar la ilusión de movimiento
const cambioInterval = setInterval(() => {
  if (arrayImagenes && arrayImagenes.length > 0) {
    imagenIndice = (imagenIndice + 1) % arrayImagenes.length;
    imagen.src = arrayImagenes[imagenIndice];
  }
}, FREC_CAMBIO_IMAGEN_LETRAS); // Frecuencia de cambio de imagen de letras
```

Figura 7: Ejemplo de setInterval para ciclo de imagen en animarNegacionYSalida()

- **Combinación de desplazamiento y ciclo de imagen**: La combinación de estos dos setInterval, consigue que en la pantalla se cree la ilusión de que la letra está viva. Para que esta animación sea creíble, la velocidad del desplazamiento y la frecuencia de los ciclos de imagen deben ser coherentes para evitar que la letra 'patine' o se vea como una estatua deslizándose. De ahí que los valores de FREC_MOVIMIENTO_LETRAS, PIXELES_POR_FREC_MOVIMIENTO_LETRAS y FREC_CAMBIO_IMAGEN_LETRAS, no se hayan escogido al azar. La variable FREC_MOVIMIENTO_LETRAS se ha escogido para que el refresco se realice aproximadamente 60 veces por segundo, que es la frecuencia de refresco más extendida en las pantallas digitales. La variable PIXELES_POR_FREC_MOVIMIENTO_LETRAS se define junto con FREC_MOVIMIENTO_LETRAS para que el desplazamiento no sea ni demasiado rápido ni demasiado lento. Finalmente, la variable FREC_CAMBIO_IMAGEN_LETRAS, se ha seleccionado para que, junto al desplazamiento de la letra, el ciclo de imagen se perciba de forma natural. Gracias a esta configuración, hemos logrado que las letras se sientan vivas y dinámicas.

## 2.5. Consideraciones adicionales

- **Optimización de recursos**: Un aspecto crítico del proyecto fue la gestión de recursos. Para evitar retrasos en las animaciones y garantizar un rendimiento fluido, se implementaron dos estrategias principales:

    1. **Reducción del tamaño de imágenes**: Se llevó a cabo un proceso de compresión de las imágenes para reducir su peso en aproximadamente un 90 % sin sacrificar la calidad visual, lo que acelera significativamente el tiempo de descarga.
    2. **Carga dinámica de recursos**: La aplicación implementa una estrategia de carga inteligente para optimizar el rendimiento y la gestión de recursos. Al seleccionar una letra, solo se precargan las imágenes de las palabras que comienzan con ella. De manera similar, al elegir una palabra, se precargan únicamente las imágenes de las letras que la componen. Este enfoque de carga, que se activa en cada etapa del juego, minimiza el tiempo de carga inicial y limita la descarga de recursos a solo las imágenes de las letras incorrectamente seleccionadas para la palabra en curso, lo que se traduce en un consumo de datos reducido y una experiencia de usuario

fluida.

- **Diseño responsivo**: Para garantizar que la aplicación funcione en cualquier dispositivo, se implementó un diseño adaptable. Esto se logró principalmente en styles.css con el uso de media queries, que ajustan el tamaño, el espaciado y el posicionamiento de los elementos de la interfaz en función del tamaño de la pantalla. Esto asegura que tanto los botones como las letras se vean de forma correcta en dispositivos móviles y de escritorio.
- **El papel del audio**: El audio juega un papel esencial. No solo se usa para la voz de la guía vocal, sino también para dar retroalimentación sobre cada acción del usuario. Por ejemplo, la reproducción del fonema de las letras al alcanzar su posición final en la pantalla no solo valida la acción del usuario, sino que también ofrece un refuerzo positivo directo que contribuye de forma activa al proceso de aprendizaje.

# 3

# Análisis de requisitos y arquitectura

## 3.1. Análisis de requisitos

Un análisis de requisitos es esencial en el ciclo de desarrollo de software. Mediante este, se define qué necesita el sistema para satisfacer al usuario. Se comprenden cuales son los objetivos del proyecto y las necesidades del usuario final, para describir de forma precisa lo que el sistema debe hacer y cómo debe hacerlo. Esto establece una base sólida para el diseño e implementación, además de ayudar a prevenir errores costosos en el futuro. De cara a definir qué debe hacer nuestra aplicación y cómo debe hacerlo, presentamos el análisis de requisitos [Sommerville(2015)].

### 3.1.1. Requisitos funcionales

Los requisitos funcionales especifican lo que la aplicación debe poder hacer. Describen las funciones y el comportamiento que la aplicación debe realizar, y que el usuario podrá ver e interactuar directamente. A continuación, hablaremos de los requisitos funcionales del sistema.

1. **Requisitos de gestión de la aplicación y navegación**

   - **Requisito de configuración**: La aplicación debe ser configurable mediante constantes en cuanto a la longitud máxima de las palabras/nombres para que las letras sigan siendo fácilmente visibles, al tamaño de las letras, la velocidad de movimiento de las letras, y el volumen de los sonidos de la música, los fonemas y los efectos generales.
   - **Requisito de localización**: La aplicación debe presentarse en español.

- **Requisito de salida a pantalla (Debugging)**: La aplicación debe ser capaz de generar mensajes de registro para depuración y seguimiento del estado.
- **Reproducción de sonidos múltiples**: La aplicación debe ser capaz de reproducir varios sonidos a la vez.
- **Detención de sonido**: La aplicación debe ser capaz de detener todos los sonidos en reproducción pertinentes, tanto si ponemos el modo de sonido Fonemas o Silencio, como si cambiamos de pestaña en el navegador, minimizamos el navegador o bloqueamos el dispositivo.
- **Gestión de pantallas**: La aplicación debe controlar la visibilidad de diferentes pantallas o estados (bienvenida, menú inicial, modos de juego) de forma dinámica, mostrando u ocultando los elementos HTML correspondientes.
- **Transición de pantallas**: Debe realizar una transición visual suave entre las pantallas mediante un overlay de color y animaciones de desvanecimiento.
- **Navegación**: La aplicación debe permitir al usuario navegar entre las diferentes pantallas, incluyendo un botón de 'volver' que lo retorne a la pantalla anterior.
- **Bienvenida**: La aplicación debe mostrar al inicio una presentación de esta, mostrando su nombre y el proyecto al que pertenece.

2. **Requisitos de interacción de usuario y controles**

- **Interacción con botones**: La aplicación debe detectar los clics en los botones para iniciar acciones específicas (cambiar de modo de juego, escuchar un sonido, etc.).
- **Gestión de la barra superior**: Debe haber una barra superior, que contenga botones de acción y que será visible en las pantallas de juego, pero oculta en la pantalla de bienvenida.
- **Gestión de sonido**: El usuario debe poder alternar entre tres estados de sonido principales: Sonido (música de fondo y efectos), Fonemas (solo sonidos de fonemas y efectos, pero con un aumento de volumen en los fonemas) y Silencio (ningún sonido).

- **Explicación en menú inicio**: La aplicación debe mostrar en el menú inicio un botón que explique la aplicación en la que nos encontramos y los modos de juego que tiene.

3. **Requisitos del modo 'Construye palabras'**

   - **Selección de letras iniciales**: El usuario debe poder elegir una letra inicial de un teclado de letras para comenzar el juego.
   - **Selección de palabras**: Tras elegir la letra, la aplicación debe mostrar un conjunto de palabras que comiencen con esa letra para que el usuario elija una.
   - **Presentación de palabra**: Tras elegir la imagen de la palabra, la aplicación debe mostrarla centrada en la pantalla y reproducir el sonido de la misma.
   - **Construcción de palabras**: La aplicación debe permitir al usuario colocar letras para formar la palabra seleccionada, y estás se irán sumando una a una.
   - **Reproducción de sonidos de fonemas**: Al colocar cada letra, debe reproducir el fonema correspondiente.
   - **Corrección de palabras**: La aplicación debe permitir al usuario corregir una palabra compuesta desde que esté colocada la primera letra. En caso de haber errores, las letras pertinentes negarán con la cabeza y se marcharán, dejando el hueco libre para otra letra.
   - **Eliminación última letra**: La aplicación debe permitir eliminar la última letra de todas las introducidas.
   - **Reproducción de palabra**: El usuario debe poder escuchar la pronunciación completa de la palabra objetivo.
   - **Ocultar fondo**: El usuario debe poder alternar entre ver un fondo de pantalla aleatorio o un fondo en blanco.
   - **Reproducción de la explicación**: El usuario debe poder ver la explicación de lo que hay que hacer, así como indicaciones para saber donde están los botones de eliminar la última letra y el de corregir.
   - **Animación de letras**: Las letras de las palabras deben mostrarse con una animación de 'andar' o movimiento secuencial.

- **Celebración**: En caso de acertar todas las letras, se mostrará una celebración y se escuchará la palabra.
- **Gestión de la puntuación o gamificación**: Por cada palabra acertada, aumentará en 1 la puntuación total.
- **Bucle**: Tras acertar una palabra, la aplicación volverá a la selección de palabras que comienzan por la letra inicial seleccionada anteriormente.

4. **Requisitos del modo 'Escribe tu nombre'**

   - **Entrada de texto**: El usuario debe poder escribir su nombre usando un teclado virtual o el teclado del dispositivo, y las letras introducidas se mostrarán ocultas para que no las vea el usuario.
   - **Validación de entrada**: La aplicación debe validar la entrada para asegurarse de que el nombre no exceda la longitud máxima permitida, además de que no tenga espacios y que solo contenga letras del abecedario español. En caso de haber algo incorrecto, saltará un mensaje informativo para el usuario.
   - **Construcción de nombre**: La aplicación debe permitir al usuario colocar letras para formar el nombre introducido, y estás se irán sumando una a una.
   - **Reproducción de sonidos de fonemas**: Al colocar cada letra, debe reproducir el fonema correspondiente.
   - **Corrección de nombre**: La aplicación debe permitir al usuario corregir el nombre desde que esté colocada la primera letra. En caso de haber errores, las letras pertinentes negarán con la cabeza y se marcharán, dejando el hueco libre para otra letra.
   - **Eliminación última letra**: La aplicación debe permitir eliminar la última letra de todas las introducidas.
   - **Reproducción de nombre**: El usuario debe poder escuchar la pronunciación completa de los fonemas de las letras que componen el nombre introducido.
   - **Ocultar fondo**: El usuario debe poder alternar entre ver un fondo de pantalla aleatorio o un fondo en blanco.

- **Reproducción de la explicación**: El usuario debe poder ver la explicación de lo que hay que hacer, así como indicaciones para saber dónde se encuentran los botones de eliminar la última letra y el de corregir.
- **Animación de letras**: Las letras del nombre deben mostrarse con una animación de 'andar' o movimiento secuencial.
- **Celebración**: En caso de acertar todas las letras, se mostrará una celebración.
- **Gestión de la puntuación o gamificación**: Por cada nombre acertado, aumentará en 1 la puntuación total.
- **Bucle**: Tras acertar el nombre, la aplicación volverá a la pantalla de entrada de texto para que se introduzca otro nombre.

### 3.1.2. Requisitos no funcionales

Los requisitos no funcionales describen cómo debe ser el sistema. Estos se centran en las cualidades de rendimiento de la aplicación como la usabilidad, la fiabilidad o la compatibilidad. En los siguientes apartados se enumerarán los requisitos no funcionales del sistema.

1. **Rendimiento y usabilidad**

   - **Fluidez**: Las animaciones y transiciones deben ejecutarse de manera suave, sin retrasos notables mediante el uso de temporizaciones específicas.
   - **Carga rápida**: La aplicación debe cargar rápidamente, ya que todos los archivos se encuentran del lado del cliente y no se requiere ninguna llamada a un servidor externo.
   - **Retroalimentación**: La aplicación debe proporcionar feedback visual (animaciones, cambios de opacidad) y auditivo (sonidos, música) para informar al usuario sobre el éxito o fracaso de sus acciones.
   - **Eficiencia de la carga de recursos**: La aplicación debe cargar todos los recursos de imagen y sonido de manera eficiente para evitar esperas. Los sonidos se cargarán al inicio de la aplicación. En cuanto a las imágenes de las palabras, se precargarán las correspondientes a una letra tras ser seleccionada. En el caso de

los recursos de imagen de las letras en movimiento, se cargarán solo los de las necesarias para la palabra seleccionada o el nombre introducido, y en caso de seleccionarse una letra errónea, se precargarán los recursos de esta antes de que salga la letra en pantalla para evitar tirones o parones durante la animación.

- **Accesibilidad visual en los botones**: La aplicación mostrará al usuario el estado de los elementos en pantalla cuando intente interactuar con ellos. Los botones no disponibles cambiarán a un color más oscuro y mostrarán una señal de prohibido al pasar el ratón por encima. Los botones disponibles mostrarán un color más claro y una mano al pasar por encima con el ratón.
- **Portabilidad**: En caso de disponer de los ficheros de la aplicación, esta podrá ejecutarse de manera offline simplemente abriendo el archivo index.html en un navegador moderno, sin ninguna instalación ni entorno de servidor.

2. **Fiabilidad**

- **Manejo del estado**: El código debe mantener el estado de la aplicación (sonidos en reproducción, modo de juego seleccionado y puntuación, entre otros) para asegurar que el comportamiento de los modos de juego sea predecible y consistente.
- **Resiliencia a errores**: La aplicación debe manejar la ausencia de imágenes o sonidos sin bloquearse.

3. **Compatibilidad y diseño**

- **Diseño responsivo**: La interfaz debe ser completamente adaptable a diferentes tamaños y orientaciones de pantalla (móvil, tablet, escritorio) gracias al uso de media queries y unidades relativas (vw, vh).
- **Diseño intuitivo**: El diseño de la interfaz debe ser claro y simple, guiando al usuario con elementos como iconos (por ejemplo, el icono de ayuda ?) y etiquetas descriptivas.
- **Coherencia visual**: La aplicación debe mantener una estética visual y un esquema de color consistentes en todas sus pantallas.

4. **Mantenimiento y organización**

    - **Fácil mantenimiento y lectura**: El código tanto de app.js como de styles.css e index.html, hará uso de comentarios explicativos y se dividirá en regiones por funcionalidad, para que el código sea fácil de mantener, leer y entender para otros desarrolladores.

## 3.2. Casos de uso en UML

Tras definir los requisitos funcionales y no funcionales, el siguiente paso a seguir en la fase de análisis es visualizar cómo el usuario interactúa con el sistema. Esto lo vamos a hacer mediante diagramas de casos de uso en UML (Lenguaje Unificado de Modelado). Con estos, podremos ver a alto nivel, las funciones del sistema y como el usuario puede interactuar con ellas, proporcionándonos una visión general del comportamiento esperado del sistema [Fowler(2004)].

Para mayor claridad, hemos dividido el análisis en tres diagramas:

1. **Diagrama de casos de uso general**: Muestra las opciones principales que tiene el usuario al iniciar la aplicación.

2. **Detalle del modo de juego 'Construye palabras'**: Este diagrama detalla el flujo completo una vez que el usuario ha decidido iniciar el modo de juego 'Construye palabras'.

3. **Detalle del modo de juego 'Escribe tu nombre'**: Este diagrama detalla el flujo completo una vez que el usuario ha decidido iniciar el modo de juego 'Escribe tu nombre'.

### 3.2.1. Diagrama de casos de uso general

Los elementos del diagrama son los siguientes (Figura 8) :

- **Actor**: Usuario, que podría ser el niño/a o el niño/a con alguna otra persona adulta que le ayude.

- **Casos de uso**:

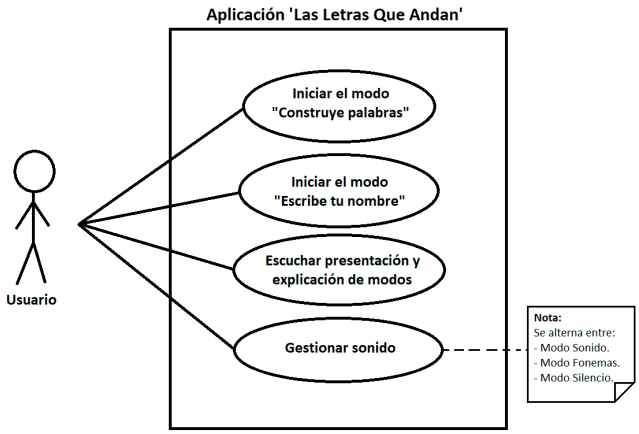


Figura 8: Aplicación 'Las Letras Que Andan'

- **Iniciar el modo 'Construye palabras'**. Este caso de uso corresponde a que el usuario pulse el botón 'Construye palabras' en el menú de inicio, seleccionando de esta manera el modo de juego pertinente.
- **Iniciar el modo 'Escribe tu nombre'**. Este caso de uso corresponde a que el usuario pulse el botón 'Escribe tu nombre' en el menú de inicio, seleccionando de esta manera el modo de juego pertinente.
- **Escuchar presentación y explicación de modos**. Este caso de uso corresponde a que el usuario pulse el botón '?' en el menú de inicio, comenzando de esta manera una guía auditiva en la que se presentará la aplicación y los botones que se deben pulsar según el modo de juego que se quiera seleccionar.
- **Gestionar sonido**. Este caso de uso corresponde a que el usuario pulse el botón de sonido posicionado en la barra superior, concretamente en el extremo derecho de la pantalla. Mediante la pulsación de este botón se alternará entre los modos siguientes:
  - **Sonido**. Se escuchará tanto la música de fondo como los efectos de sonido y las explicaciones.
  - **Fonemas**. Se silenciará la música de fondo y se escucharán los efectos de sonido y las explicaciones, pero con una amplificación de volumen para los fonemas.

- **Silencio**. Se silenciarán todos los sonidos de la aplicación.

### 3.2.2. Detalle del modo de juego 'Construye palabras'

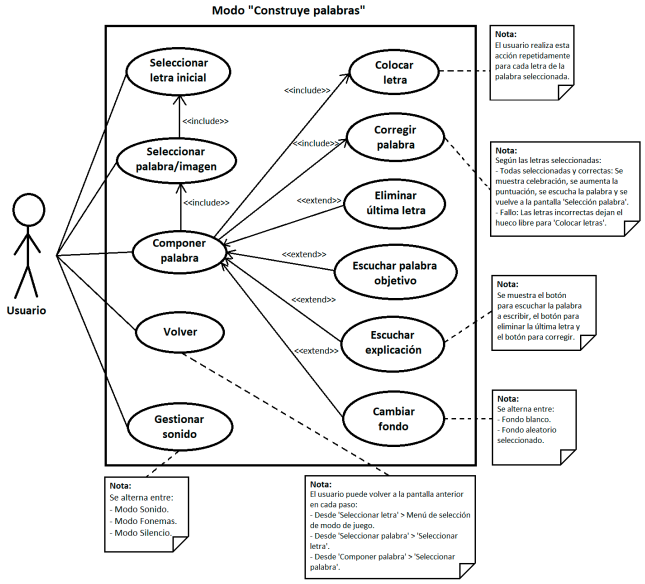


Figura 9: Detalle del modo 'Construye palabras'

El diagrama de la figura 9 se refiere al modo 'Construye Palabras' de la aplicación y se detalla a continuación:

- **Actor**: Usuario, que podría ser el niño/a o el niño/a con alguna otra persona adulta que le ayude.
- **Casos de uso**:
  - **Seleccionar letra inicial**. Lo primero que hace el usuario en el modo 'Construye palabras' es seleccionar la letra inicial de la palabra que construirá. Este caso de uso corresponde a que el usuario pulse el botón de una de las letras iniciales.
  - **Seleccionar palabra/imagen**. Tras seleccionar la letra inicial, se presentan imágenes de palabras que empiezan por esa letra, por lo que es obligatorio que antes de seleccionar la imagen, se seleccione la letra inicial. Este caso de uso corresponde a que el usuario pulse el botón de una de las imágenes que representan la palabra a construir.

- **Componer palabra**. Tras seleccionar la imagen, lo próximo será escribir la palabra correspondiente, por lo que es obligatorio que antes de componer la palabra, se seleccione la imagen. Este caso de uso corresponde al núcleo de este modo de juego y se compone de dos casos de uso obligatorios:
    - **Colocar letra**. Este caso de uso corresponde a que el usuario pulse los botones de las letras en el teclado final de la barra inferior, y lo hará tantas veces como sea la longitud de la palabra.
    - **Corregir palabra**. Este caso de uso corresponde a que el usuario pulse el botón Corregir de la barra superior, que estará disponible desde que hay una letra seleccionada, para verificar si ha elegido las letras correctas. Si hay alguna letra incorrecta, esta dejará el hueco libre para poder ocuparla con otra. Si se han elegido todas las letras correctas, se mostrará una celebración, aumentará la puntuación, se escuchará la palabra y se volverá a la pantalla anterior para seleccionar una imagen.

  Durante la composición de la palabra, el usuario puede realizar una serie de acciones opcionales:
    - **Eliminar última letra**. Este caso de uso corresponde a que el usuario pulse el botón Eliminar de la barra superior, que estará disponible desde que hay una letra seleccionada, el cual provocará que la última letra de la palabra que esté seleccionada se marche para dejar el hueco a otra.
    - **Escuchar palabra objetivo**. Este caso de uso corresponde a que el usuario pulse el botón de la imagen de la palabra seleccionada de la barra inferior, el cual reproducirá la palabra.
    - **Escuchar explicación**. Este caso de uso corresponde a que el usuario pulse el botón '?' de la barra inferior, el cual muestra el botón de 'Escuchar palabra objetivo', el botón de 'Eliminar última letra' y el botón de 'Corregir palabra', mientras se explica durante una reproducción de audio.
    - **Cambiar fondo**. Este caso de uso corresponde a que el usuario pulse el botón Fondo de la barra superior, con el cual se alternará entre un fondo en blanco, y un fondo aleatorio que se selecciona cada vez que se entra en la pantalla de

’Componer palabra’.

- **Volver**. Este caso de uso corresponde a que el usuario pulse el botón Volver de la barra superior, el cual estará disponible en todas las pantallas de este modo de juego y servirá para ir a la pantalla anterior. Desde ’Seleccionar letra’ se irá al menú inicial, desde ’Seleccionar palabra/imagen’ se irá a ’Seleccionar letra’, y desde ’Componer palabra’ se irá a ’Seleccionar palabra/imagen’.
- **Gestionar sonido**. Este caso de uso se corresponde con el explicado en ’Diagrama de casos de uso general’, y estará disponible en todas las pantallas de este modo de juego.

### 3.2.3. Detalle del modo de juego ’Escribe tu nombre’

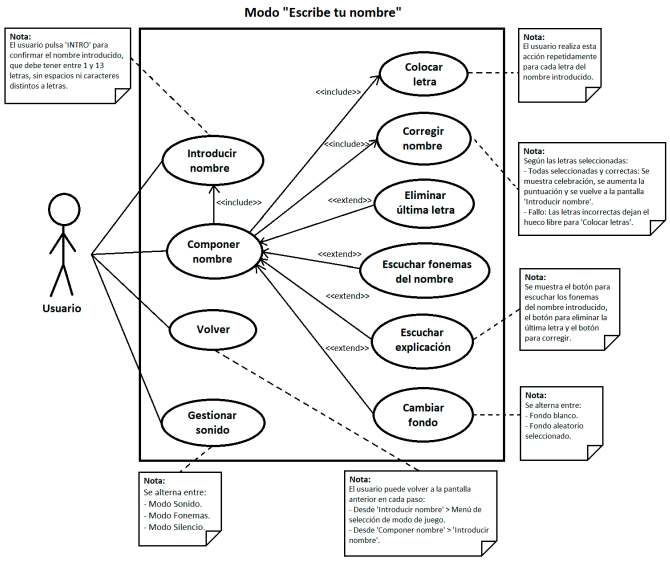


Figura 10: Detalle del modo ’Escribe tu nombre’

A continuación, explicamos el diagrama (figura 10) :

- **Actor**: Usuario, que podría ser el niño/a o el niño/a con alguna otra persona adulta que le ayude.
- **Casos de uso**:
  - **Introducir nombre**. Lo primero que hace el usuario en el modo ’Escribe tu nombre’ es introducir su nombre en un campo de texto y pulsa Enter para confirmarlo.

Este caso de uso corresponde a escribir un nombre sin espacios ni caracteres diferentes a las letras del abecedario español, y con una longitud entre 1 y 13 caracteres.

- **Componer nombre**. Tras introducir el nombre, lo próximo será escribirlo, por lo que es obligatorio que antes de componer el nombre, se introduzca un nombre válido. Este caso de uso corresponde al núcleo de este modo de juego y se compone de dos casos de uso obligatorios:
    - **Colocar letra**. Este caso de uso se corresponde con el explicado en 'Detalle del modo de juego 'Construye palabras'.
    - **Corregir nombre**. Este caso de uso corresponde a que el usuario pulse el botón Corregir de la barra superior, que estará disponible desde que hay una letra seleccionada, para verificar si ha elegido las letras correctas. Si hay alguna letra incorrecta, esta dejará el hueco libre para poder ocuparla con otra. Si se han elegido todas las letras correctas, se mostrará una celebración, aumentará la puntuación, y se volverá a la pantalla anterior para introducir un nombre.

  Durante la composición del nombre, el usuario puede realizar una serie de acciones opcionales:
    - **Eliminar última letra**. Este caso de uso se corresponde con el explicado en 'Detalle del modo de juego 'Construye palabras'.
    - **Escuchar fonemas del nombre**. Este caso de uso corresponde a que el usuario pulse el botón de la imagen del nombre introducido de la barra inferior, el cual reproducirá los fonemas de cada una de las letras que componen el nombre.
    - **Escuchar explicación**. Este caso de uso corresponde a que el usuario pulse el botón '?' de la barra inferior, el cual muestra el botón de 'Escuchar fonemas del nombre', el botón de 'Eliminar última letra' y el botón de 'Corregir nombre', mientras se explica durante una reproducción de audio.
    - **Cambiar fondo**. Este caso de uso se corresponde con el explicado en 'Detalle del modo de juego 'Construye palabras'.
- **Volver.** Este caso de uso corresponde a que el usuario pulse el botón Volver de

la barra superior, el cual estará disponible en todas las pantallas de este modo de juego y servirá para ir a la pantalla anterior. Desde 'Introducir nombre' se irá al menú inicial, y desde 'Componer nombre' se irá a 'Introducir nombre'.

- **Gestionar sonido**. Este caso de uso se corresponde con el explicado en 'Diagrama de casos de uso general', y estará disponible en todas las pantallas de este modo de juego.

## 3.3. Análisis de la viabilidad técnica y justificación de tecnologías

Al comenzar el desarrollo de la aplicación, el análisis de la viabilidad técnica y la justificación de las tecnologías seleccionadas se convirtieron en pasos fundamentales. Este apartado detalla por qué se optó por una arquitectura de página única del lado del cliente (SPA) y el uso de JavaScript puro junto a sus tecnologías de apoyo (HTML y CSS), en lugar de frameworks o librerías más complejas. La decisión se basó en una evaluación de los requisitos del proyecto, su alcance y las ventajas técnicas que este enfoque ofrecía.

### 3.3.1. Justificación de la arquitectura de página única (SPA)

La elección de una arquitectura SPA fue la que consideramos más viable y eficiente para el propósito de este proyecto. A diferencia de una arquitectura cliente-servidor tradicional, donde cada interacción del usuario podría requerir una nueva carga de página desde el servidor, la SPA carga todos los recursos principales de la aplicación (HTML, CSS, JS) en una única solicitud inicial, para después acceder a los recursos auxiliares tales como imágenes y audios, de manera inteligente en caso necesario.

Esta arquitectura nos permite cumplir con los siguientes requisitos no funcionales:

- **Rendimiento y usabilidad**: Esta arquitectura elimina los tiempos de espera de las recargas de página, lo que resulta en una experiencia de usuario rápida y fluida. Las transiciones entre las pantallas de la aplicación son instantáneas, lo cual es crucial para mantener la atención de los usuarios jóvenes. La sensación de ser una aplicación de escritorio, y no una página web, mejora significativamente la usabilidad y la satisfacción del usuario.

- **Viabilidad y portabilidad**: Al no depender de un servidor externo para la lógica de negocio, la aplicación es altamente portátil y autónoma. No requiere una infraestructura de servidor o base de datos, lo que minimiza los costos de desarrollo y mantenimiento. Puede ser distribuida y ejecutada en cualquier navegador con solo los archivos locales, cumpliendo con el requisito no funcional de que la aplicación pueda funcionar sin conexión a Internet.

### 3.3.2. Justificación de la elección de tecnologías

La decisión de no utilizar frameworks de JavaScript como React, Angular o Vue.js, y de basar el proyecto en tecnologías web puras, fue tomada con la idea de estar alineados con los objetivos del TFG.

- **HTML (Estructura)**: Se eligió HTML por su simplicidad y su rol fundamental como el lenguaje de marcado estándar para la web. Sirvió como una base sólida para crear una estructura accesible. El index.html fue diseñado como un único contenedor para todas las vistas de la aplicación, permitiendo que JavaScript las manipule eficientemente sin necesidad de múltiples archivos de página.

- **CSS (Presentación y comportamiento visual)**: El uso de CSS se justificó por su eficiencia para manejar el estilo estático y las animaciones. El diseño fue construido para ser totalmente adaptable, utilizando unidades de medida relativas (vw, vh) y @media queries para asegurar que la interfaz se vea y funcione correctamente en cualquier dispositivo, cumpliendo con el requisito de diseño responsivo. Además, las transiciones CSS fueron la herramienta ideal para las animaciones de fundido, ya que son muy eficientes a nivel de rendimiento.

- **JavaScript (Lógica y orquestación)**: El uso de JavaScript puro, sin la carga de un framework, fue la elección que tomamos dado que la complejidad del proyecto no justificaba la curva de aprendizaje ni el tamaño adicional de un framework. Este enfoque permitió centrar toda la atención en la lógica de negocio del juego. La aplicación utiliza APIs nativas del navegador, como la API del DOM, para acceder y manipular la estructura (index.html) y el estilo (styles.css) de la página, la API de Eventos, para reaccionar a las acciones del usuario, la API de Audio Web, para gestionar y reproducir todos los

sonidos de la aplicación, la API de animación a través de CSS para gestionar la fluidez de las animaciones de transición y el resto de propiedades definidas, la API de Web Storage, para almacenar datos del lado del cliente como el estado de la aplicación mientras dura la sesión de navegación (sessionStorage), y la API de Media Queries para interpretar las reglas de styles.css y que el diseño sea adaptable. Esta dependencia en las APIs del navegador nos asegura una alta compatibilidad y un rendimiento óptimo.

### 3.3.3. Alternativas descartadas

Para validar la elección de las tecnologías, se consideraron y descartaron otras opciones:

- **Frameworks de JavaScript**: Se descartaron porque habrían introducido una complejidad innecesaria y no nos podríamos haber centrado tanto en la lógica de negocio del juego. Un framework sería más adecuado para aplicaciones con un estado de datos muy complejo o que requieren una gestión avanzada de componentes.
- **Desarrollo en servidor (Backend)**: Se consideró innecesario, ya que la aplicación no maneja datos de usuario persistentes ni requiere procesamientos complejos en un servidor. Todos los recursos y la lógica necesaria para el funcionamiento del juego se encuentran del lado del cliente, por lo que una infraestructura de backend habría sido una complicación adicional sin un beneficio claro.

En conclusión, la arquitectura de la aplicación y la selección de tecnologías se basaron en un análisis técnico que priorizó la simplicidad, el rendimiento y la viabilidad, resultando en un sistema robusto, autónomo y eficiente que cumple de manera óptima con todos los requisitos del proyecto.

## 3.4. Arquitectura general de la aplicación

Después de haber definido los requisitos funcionales y no funcionales, y de haber visualizado los casos de uso, es necesario establecer la arquitectura general de la aplicación. La arquitectura del sistema es el esqueleto o la estructura principal de la aplicación. No es simplemente la elección de tecnologías, sino la organización de los componentes de la aplicación, como se relacionan entre sí y las reglas que siguen para su funcionamiento. Una arquitectura bien definida es fundamental, ya que influye directamente en la calidad, la escalabilidad,

la mantenibilidad y el rendimiento del sistema a largo plazo. En este apartado se detallará el diseño general del sistema, así como la división en componentes y la interacción entre ellos, asegurándonos de esta manera que tenemos las ideas claras para la fase de implementación [Pressman(2010)].

### 3.4.1. La Arquitectura: Aplicación de página única del lado del cliente

La arquitectura de la aplicación es una aplicación de página única (SPA) del lado del cliente (figura 11). Este modelo es ideal para aplicaciones que no requieren una base de datos persistente ni un servidor para la lógica de negocio. Toda la aplicación se carga en el navegador del usuario en una única página HTML y las interacciones posteriores son gestionadas dinámicamente por JavaScript, sin necesidad de recargar la página, ofreciendo de esta manera una experiencia fluida y similar a la de una aplicación de escritorio.

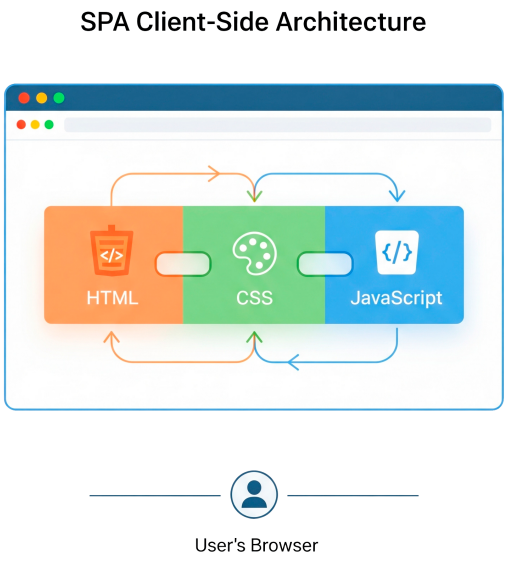


Figura 11: Arquitectura SPA del lado del cliente

### 3.4.2. Las capas de la aplicación

La aplicación se divide en tres archivos principales, cada uno con una responsabilidad bien definida, lo que facilita el desarrollo y el mantenimiento. A continuación, presentamos cada

uno de los archivos:

1. **Capa de contenido y estructura (index.html)**. Este archivo actúa como la estructura principal de la aplicación. Su única responsabilidad es contener el esqueleto de la interfaz de usuario. En lugar de tener múltiples archivos HTML, este único archivo contiene todas las pantallas o vistas que el usuario puede ver. Aunque algunas de estas pantallas están ocultas inicialmente, están siempre presentes en el DOM (Modelo de Objetos del Documento), listas para ser manipuladas por la lógica de JavaScript.

2. **Capa de presentación y comportamiento visual (styles.css)**. Este archivo se encarga de la apariencia visual, y las animaciones de los botones y de las transiciones entre pantallas. Su responsabilidad va más allá del estilo estático. Define cómo se ve la aplicación en cualquier momento. Controla los colores, el tamaño y la disposición de los elementos. Además, es el responsable del diseño responsive a través de media queries para garantizar que la interfaz se adapte a diferentes dispositivos y tamaños de pantalla. También, mediante el uso de transiciones CSS, gestiona las animaciones visuales, como la aparición o desaparición de elementos de manera suave, que hacen que la experiencia sea fluida.

3. **Capa de lógica y orquestación (app.js)**. Este es el cerebro y el director de orquesta de la aplicación. Su rol es triple:

    - Gestión del estado: Mantiene el estado de la aplicación en tiempo real mediante variables (por ejemplo, el estado del sonido o la pantalla actual), lo que le permite saber qué lógica ejecutar en cada momento.
    - Lógica de la aplicación: Contiene todas las reglas de juego y el comportamiento de la aplicación, como la selección de palabras, la gestión de la puntuación y las animaciones de las letras.
    - Interacción con las APIs del navegador: Utiliza APIs nativas, como la API del DOM y la API de Audio Web, para manipular el HTML y el CSS. Por ejemplo, al detectar un clic en un botón, utiliza la API de Eventos para ejecutar una función que cambia la opacidad de un elemento a través del DOM, que a su vez activa una transición definida en el CSS.

### 3.4.3. Flujo de interacción

La interacción entre las tres capas es fluida y completamente del lado del cliente. La aplicación se inicia cuando el navegador carga el index.html. Este archivo, a su vez, carga el styles.css para aplicar el diseño y el app.js para iniciar la lógica. Desde ese momento, el app.js toma el control, respondiendo a las acciones del usuario y manipulando los elementos de la interfaz en tiempo real, sin requerir ninguna comunicación adicional con el servidor, más allá de descargar los elementos multimedia que se requieran y no se hayan descargado aún.

# 4

# Interfaz e interacción con el usuario

## 4.1. La interfaz con el usuario (front end)

Tras haber definido la arquitectura interna de la aplicación, el siguiente paso a realizar es conceptualizar y diseñar la interfaz con el usuario (front end). Esta es la parte de la aplicación con la que el usuario final interactúa directamente. No se trata solo de la estética, sino de la usabilidad, la accesibilidad y la experiencia del usuario (UX/UI).

Una interfaz bien diseñada es intuitiva, fácil de navegar y responde de manera eficiente, lo que influye directamente en la satisfacción y adopción del sistema. En este apartado, se detallarán los principios de diseño, y los componentes que darán forma a la cara visible de la aplicación, asegurando que la interacción sea fluida y satisfactoria para el usuario [Nielsen and Norman(2013)].

En la figura 12 se muestra un esquema general de las interfaces que conforman la aplicación desarrollada que se irán describiendo a lo largo de este capítulo.


Figura 12: Organización general de la interfaz 'Las Letras Que Andan'

### 4.1.1. Pantalla de bienvenida

La pantalla de bienvenida (figura 13) tiene además los siguientes apartados (figura 14):

- **Pantalla de bienvenida - Textos**


Figura 13: Pantalla de bienvenida 'Las Letras Que Andan'

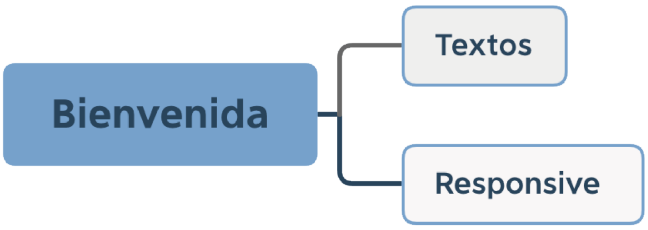


Figura 14: Apartados de la interfaz de Bienvenida

Nada más iniciar la aplicación, lo primero que se muestra es lo que en este apartado llamaremos 'Pantalla de bienvenida' (figura 13). En esta se podrá ver el nombre de la aplicación, 'Las Letras Que Andan', y el proyecto al que pertenece, 'PROYECTO PATIO', además de un texto que relaciona ambas partes, 'una aplicación de'. Los textos se presentan centrados en columna, en formato negrita con color blanco y con sombreado para facilitar su visualización. Los textos de la aplicación, en general, no tienen un estilo de fuente particular para evitar depender de fuentes de las que algún navegador no disponga, por lo que se usará la fuente por defecto de cada uno.

En esta pantalla se muestra por primera vez el que será el fondo de pantalla principal de nuestra aplicación en todas las pantallas, a excepción de en las pantallas de 'Componer palabra' y 'Componer nombre', que presentarán un fondo de pantalla aleatorio como ya explicaremos en el apartado correspondiente. Este fondo de pantalla principal presenta un conjunto de tonalidades azules, y es un homenaje a la aplicación original, puesto que era el que presentaba.

Volviendo a los textos, estos aparecen con un tamaño relativo al ancho de la pantalla. El nombre de la aplicación tiene un tamaño de letra del 6 % del ancho de la pantalla, el texto 'una aplicación de', tiene un tamaño de letra del 4 % del ancho de pantalla y el texto del proyecto tiene un tamaño de letra del 5 % del ancho de pantalla. Al asignarle al texto, un tamaño relativo al tamaño de la pantalla, conseguimos que los textos sean siempre visibles y consistentes independientemente de la pantalla en la que se muestren, o de que cambiemos la ventana de tamaño durante la ejecución.

- **Pantalla de bienvenida - Responsive**

  Para cumplir el requisito no funcional de que el diseño sea responsivo, además de usar porcentajes relativos al tamaño de la pantalla, en el caso de que la aplicación se ejecute en una pantalla vertical en lugar de una horizontal, los tamaños de las letras del nombre de la aplicación serán del 8 % del ancho de pantalla, las del texto 'una aplicación de' serán del 6 % y las del proyecto del 7 %, en lugar del 6 %, 4 % y 5 %, respectivamente. De esta manera, los textos seguirán viéndose correctamente, pero sin llegar a salirse de la pantalla.

### 4.1.2. Menú inicio

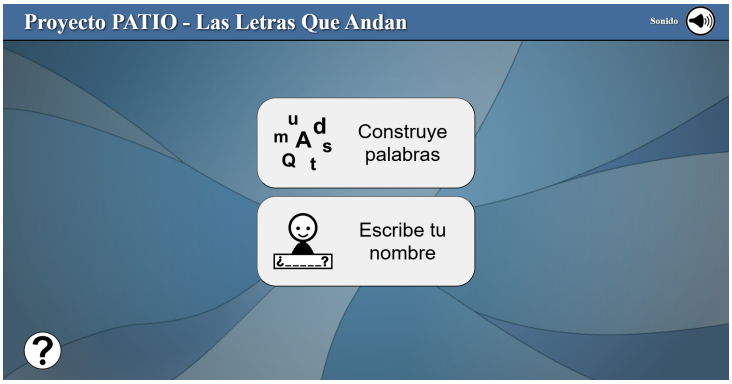


Figura 15: Pantalla de menú inicio

Tras la pantalla de bienvenida, se muestra el menú de inicio (figura 15), en el cual podremos elegir los modos de juego principales de la aplicación. En este aparece la barra superior que ya permanecerá durante el resto del transcurso de la ejecución, y que contendrá varios elementos

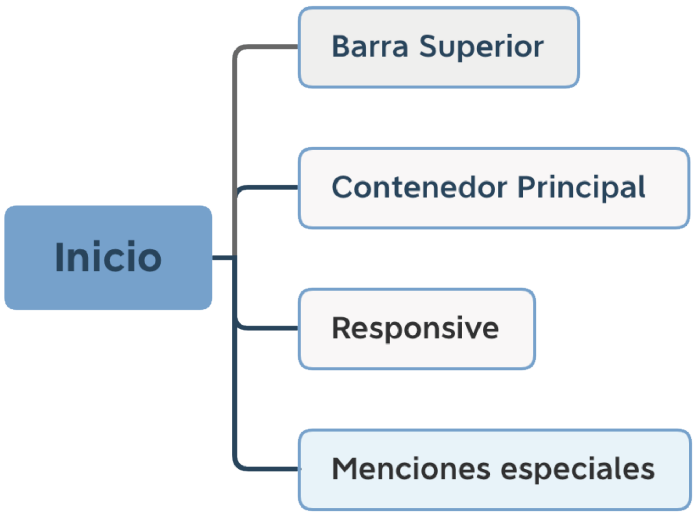


Figura 16: Apartados de la interfaz del menú inicio

que irán variando según la pantalla en la que nos encontremos. La interfaz del menú inicio cuenta a su vez con los apartados que se muestran en la figura 16.

- **Menú inicio - Barra superior**

  La barra superior tiene un color azul fijo en todas las pantallas [RGB (67, 108, 155)], siendo este un poco más oscuro que el azul del fondo de la pantalla principal y con un borde negro en el inferior para poder diferenciarse de este. Ocupa todo el ancho de la pantalla y el 10 % superior de la altura de la pantalla.

  En el menú de inicio, la barra superior presenta dos elementos que son un título que incluye el proyecto y el nombre de la aplicación a la izquierda, y un botón para gestionar el sonido a la derecha. Este botón estará presente en todas las pantallas junto con la barra superior, permitiendo gestionar el sonido en todo momento.

  El título se presenta centrado verticalmente respecto a la barra superior, y con un margen a la izquierda de un 1 % del ancho de la pantalla, en formato negrita con color blanco y con sombreado para facilitar su visualización. El tamaño de la letra es de un 3 % del ancho de la pantalla y el título está configurado para que en caso de que ocupe más del 75 % del ancho de la pantalla, se muestre el resto con puntos suspensivos sin que se divida el texto en varias líneas, adaptándose de esta manera a todos los tipos de pantallas.

El botón para gestionar el sonido presenta una imagen identificativa, con un fondo blanco y un borde negro para facilitar su visualización, teniendo una proporción 1/1 y un tamaño del 8 % del alto de la pantalla para que nunca se salga de la barra superior, además de un texto descriptivo fuera del botón a su izquierda a una distancia del 1 % del ancho de la pantalla con un tamaño de letra 1.3 % el ancho de la pantalla, en formato negrita con color blanco y con sombreado para facilitar su visualización. Este tamaño de letra, aunque pueda parecer pequeño, nos asegura que cuando haya más botones en la barra superior en otras pantallas, no se solapen los textos de un botón con las imágenes de otro.

En cuanto a las imágenes y el texto que presentan el botón de gestión de sonido según el modo de sonido seleccionado, son 'Sonido', 'Fonemas' y 'Silencio', junto con las imágenes siguientes para que sea intuitivo:


Figura 17: Modo Sonido


Figura 18: Modo Fonemas


Figura 19: Modo Silencio

- **Menú inicio - Contenedor principal**

  Fuera de la barra superior, en el centro de la pantalla, se muestran los botones para iniciar los modos de juego principales de la aplicación, 'Construye palabras' y 'Escribe tu nombre', alineados en columna uno encima de otro. Estos tienen el fondo blanco y un fino borde negro para facilitar su visualización. Con un tamaño horizontal del 30 % del

ancho de la pantalla y vertical del 24 % del alto de la pantalla, para adaptarse a cualquier tamaño de ventana. En su interior presentan una imagen identificativa del modo de juego pertinente, que ocupa un 70 % de la altura del botón y un texto descriptivo en color negro con un tamaño de letra del 2.7 % de la anchura de la pantalla.

Por último, en la esquina inferior izquierda, dejando por debajo un margen del 3 % de la altura de la pantalla y a la izquierda un 3 % de la anchura de la pantalla, se encuentra el botón de presentación que se encargará de presentar la aplicación y de explicar el botón que se debe pulsar, según el modo de juego que se quiera seleccionar. Este tiene el fondo blanco y un fino borde negro para facilitar su visualización. Presenta una imagen del carácter '?', y su tamaño es del 10 % de la altura de la pantalla tanto en altura como en anchura.

- **Menú inicio - Responsive**

  Para cumplir el requisito no funcional de que el diseño sea responsivo, además de usar porcentajes relativos al tamaño de la pantalla, en el caso de que la aplicación se ejecute en una pantalla vertical en lugar de una horizontal, se realizarán los siguientes cambios:

  - El título tendrá un tamaño de letra de un 4 % del ancho de la pantalla en lugar del 3 %, además de que solo podrá ocupar el 60 % del ancho de la pantalla en lugar del 75 %. En caso de pasarse del 60 %, el texto se partirá en varias líneas.
  - El botón para gestionar el sonido tendrá un tamaño del 7 % del alto de la pantalla en lugar del 8 %, para que no haya solapamientos entre los botones de la barra superior en las pantallas que presentan más botones.
  - El texto descriptivo del botón para gestionar el sonido no se mostrará para evitar solapamientos.
  - Los botones para iniciar los modos de juego principales, 'Construye palabras' y 'Escribe tu nombre', presentarán un ancho del 80 % del ancho de la pantalla y una altura del 18 % de la altura de la pantalla, con un tamaño de letra del 5 % del ancho de la pantalla, en lugar de los porcentajes 30, 24 y 2.7, respectivamente.

- **Menú inicio - Menciones especiales**

Como mención especial, tras pulsar el botón de presentación, los botones 'Construye palabras' y 'Escribe tu nombre' aumentarán de escala en 1.5 para pantallas horizontales y en 1.1 para pantallas verticales, consecutivamente para después volver a su tamaño original.

### 4.1.3. Modo 'Construye palabras'

Tras pulsar el botón del menú inicial 'Construye palabras', se iniciará este modo, el cual es uno de los dos modos principales de la aplicación. Dado que en el transcurso de este modo hay varias pantallas, las dividiremos en subapartados que describiremos a continuación (figura 20).

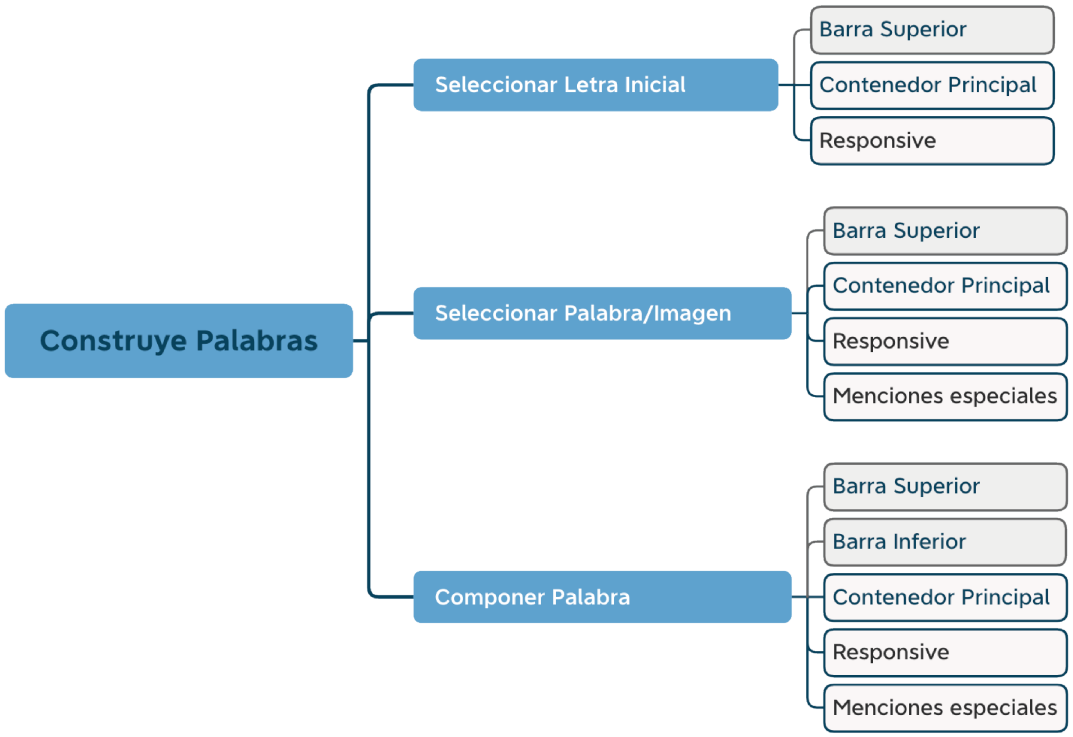


Figura 20: Apartados de las interfaces del modo 'Construye palabras'

- **Modo 'Construye palabras': Seleccionar letra inicial**

  La primera pantalla del modo 'Construye palabras', es la de 'Seleccionar letra inicial' (figura 21), y se puede acceder a esta pulsando el botón del menú inicial 'Construye palabras', o el botón 'Volver' en la pantalla 'Seleccionar palabra/imagen'.

- **Modo 'Construye palabras': Seleccionar letra inicial - Barra superior**

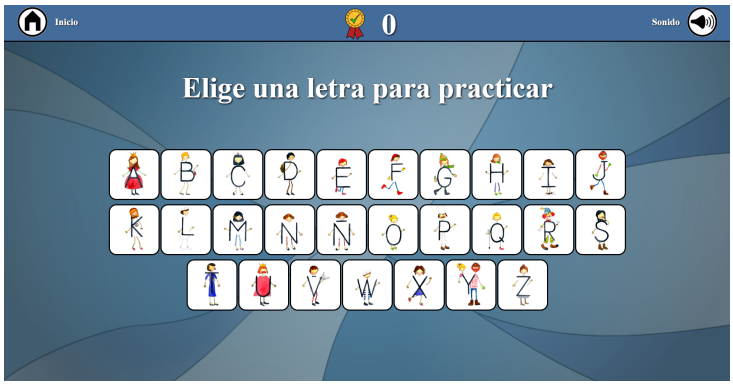


Figura 21: Pantalla 'Seleccionar letra inicial'

En esta pantalla se oculta el título de la barra superior que se muestra en el menú inicial, y aparece, además del botón de gestionar el sonido, el botón 'Volver' en la esquina superior izquierda, con el mismo formato que explicamos para el botón de gestionar el sonido en el apartado del menú inicial, pero en lugar de tener el texto descriptivo a la izquierda, lo tiene a la derecha. El texto dice 'Inicio', para indicar que el botón te llevará al menú inicio, además de que presenta una imagen que lo describe para que sea intuitivo para el usuario.

La barra superior está dividida en dos, donde los botones que aparezcan en la parte derecha tendrán el texto descriptivo a la izquierda, mientras que los botones que aparezcan a la izquierda tendrán el texto descriptivo a la derecha. En el centro de esta división se encuentra la puntuación.

La puntuación está representada mediante una imagen de una medalla y un número que lleva la cuenta de palabras/nombres acertados. La imagen tiene un tamaño del 9 % de la altura de la pantalla para que nunca se salga de la barra superior, mientras que el número tiene un tamaño del 8 % de la altura de la pantalla y se presenta en formato negrita con color en blanco y con sombreado para facilitar su visualización.

- **Modo 'Construye palabras': Seleccionar Letra Inicial - Contenedor principal**

  Fuera de la barra superior, se presentan un texto descriptivo con lo que hay que hacer en la pantalla 'Elige una letra para practicar', y un teclado de letras. Ambos se encuentran en un con display: flex, para poder posicionarlos de manera relativa apilados en columna.

El texto descriptivo se muestra en la parte superior, con formato negrita con color blanco y con sombreado para facilitar su visualización. Las letras tienen un tamaño del 4 % del ancho de la pantalla.

El teclado de letras se presenta en 3 filas de botones centrados con un tamaño del 7 % del ancho de la pantalla. Estos botones están representados con imágenes de cada una de las letras sobre un fondo blanco, y con un fino borde para facilitar su visualización y hacerlos atractivos al usuario.

- **Modo 'Construye palabras': Seleccionar letra inicial - Responsive**

  Para cumplir el requisito no funcional de que el diseño sea responsivo, además de usar porcentajes relativos al tamaño de la pantalla, en el caso de que la aplicación se ejecute en una pantalla vertical en lugar de una horizontal, se realizarán los siguientes cambios:

  - La imagen de la puntuación tendrá un tamaño del 8 % de la altura de la pantalla, en lugar del 9 %.
  - El texto de la puntuación tendrá un tamaño del 8 % del ancho de la pantalla en lugar de respecto a la altura, y además para evitar solapamientos con el resto de botones de la barra superior, el texto se mostrará encima de la imagen de la puntuación.
  - Los botones 'Volver' y 'Sonido', presentarán el mismo formato que el descrito en el botón 'Sonido' del menú inicio.
  - El texto descriptivo 'Elige una letra para practicar', tendrá un tamaño del 7 % respecto al ancho de la pantalla, en lugar del 4 %, para que siga siendo visible.
  - Los botones del teclado de letras, tendrán un tamaño del 19 % del ancho de la pantalla, en lugar del 7 %, para que se sigan viendo con facilidad.

- **Modo 'Construye palabras': Seleccionar palabra/imagen**

  La segunda pantalla del modo 'Construye palabras', es la de 'Seleccionar palabra/imagen' (figura 22), y se puede acceder a esta pulsando el botón de una letra en la pantalla 'Seleccionar letra inicial', pulsando el botón 'Volver' en la pantalla 'Componer palabra', o acertando la palabra en la pantalla 'Componer palabra'.

- **Modo 'Construye palabras': Seleccionar palabra/imagen - Barra superior**


Figura 22: Pantalla 'Seleccionar palabra/imagen'

En esta pantalla se muestran los mismos elementos de la barra superior que en la pantalla 'Seleccionar letra inicial', pero en el caso del botón 'Volver', el texto descriptivo que se muestra es 'Volver', y presenta una imagen que lo describe para que sea intuitivo para el usuario.

- **Modo 'Construye palabras': Seleccionar palabra/imagen - Contenedor principal**

  Fuera de la barra superior, se presentan un texto descriptivo con lo que hay que hacer en la pantalla 'Elige una imagen', y un teclado de imágenes/palabras que comienzan por la letra que se seleccionó en la pantalla 'Seleccionar letra inicial'. Ambos se encuentran en un con display: flex, para poder posicionarlos de manera relativa apilados en columna.

  El texto descriptivo se muestra en el mismo formato que el descrito para 'Elige una letra para practicar' en la pantalla 'Seleccionar letra inicial'.

  El teclado de imágenes/palabras se presenta en 2 filas de botones centrados con un tamaño horizontal del 10 % del ancho de la pantalla, y un tamaño vertical del 25 % de la altura de la pantalla. Estos botones están representados con imágenes de cada una de las palabras y con un fino borde para facilitar su visualización y hacerlos atractivos al usuario.

- **Modo 'Construye palabras': Seleccionar palabra/imagen - Responsive**

  Para cumplir el requisito no funcional de que el diseño sea responsivo, además de usar porcentajes relativos al tamaño de la pantalla, en el caso de que la aplicación se ejecute

en una pantalla vertical en lugar de una horizontal, se realizarán los siguientes cambios:

- Los elementos de la barra superior presentarán el mismo formato que el descrito en la pantalla 'Seleccionar letra inicial'.
- El texto descriptivo 'Elige una imagen', presentará el mismo formato que el texto descriptivo de la pantalla 'Seleccionar letra inicial'.
- Los botones del teclado de imágenes/palabra, tendrán un tamaño horizontal del 32 % del ancho de la pantalla, y un tamaño vertical del 16 % de la altura de la pantalla, en lugar del 10 % y 25 %, respectivamente.

- **Modo 'Construye palabras': Seleccionar palabra/imagen - Menciones especiales**

  Como mención especial, tras seleccionar la palabra, la pantalla se oscurece y la imagen seleccionada se ve resaltada en el centro de la pantalla con un tamaño del 50 % de la dimensión más corta de la pantalla, ya sea la vertical o la horizontal, para que sea más atractivo para el usuario.

- **Modo 'Construye palabras': Componer palabra**

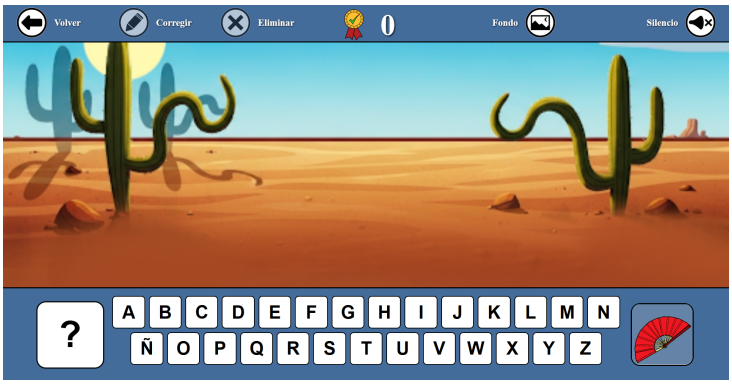


Figura 23: Pantalla 'Componer palabra'

La tercera pantalla del modo 'Construye palabras', es la de 'Componer palabra' (figura 23), y se puede acceder a esta pulsando el botón de una imagen/palabra en la pantalla 'Seleccionar palabra/imagen'.

- **Modo 'Construye palabras': Componer palabra - Barra superior**

  En esta pantalla se muestran los mismos elementos de la barra superior que en la pantalla 'Seleccionar palabra/imagen' pero, además, se muestran tres botones más con el

mismo formato que los anteriores. Los botones adicionales son 'Corregir' y 'Eliminar' en la parte izquierda de la barra superior, mientras que, en la parte derecha de la barra superior, se añade el botón 'Fondo'. Todos estos presentan tanto el texto descriptivo como una imagen representativa, con el mismo formato que los demás botones de la barra superior. Estos están distribuidos de manera que el botón 'Volver' y 'Sonido', quedan en los extremos izquierdo y derecho respectivamente, el botón 'Corregir' tiene un margen izquierdo con el extremo de la pantalla de un 14 % del ancho de la pantalla, el botón 'Eliminar', tiene un margen izquierdo del 28 % del ancho y el botón 'Fondo' tiene un margen derecho del 22 % del ancho.

- **Modo 'Construye palabras': Componer palabra - Barra inferior**

  Aquí también se muestra la barra inferior, que solo está presente en las pantallas 'Componer palabra' y 'Componer nombre'. Ocupa todo el ancho de la pantalla y el 25 % inferior de la altura de la pantalla. Presenta el mismo color que la barra superior [RGB (67, 108, 155)] y un borde superior, para distinguirla con facilidad del resto de la pantalla.

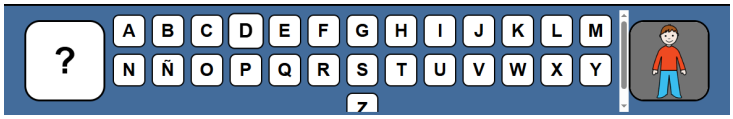


Figura 24: Barra inferior con desplazamiento

La barra inferior (figura 24) contiene, en esta pantalla - de izquierda a derecha-, el botón de explicación, los botones del teclado de letras finales y el botón de escuchar palabra.

El botón de explicación tiene una proporción 1/1 y un tamaño del 18 % del alto de la pantalla para que nunca se salga de la barra inferior. Tiene un fondo blanco, un fino borde y un texto '?', en formato negrita, color negro y un tamaño del 5 % del ancho de la pantalla.

El teclado de letras finales se presenta en una fila justificada al centro y que puede ocupar un 75 % del ancho total de la pantalla y un 100 % de la altura total de la barra inferior. Cuando los botones superan el 75 % del ancho de la barra inferior, la fila se divide hasta que quepan todos. Cuando los botones superan el 100 % de la altura de la barra inferior, aparece una barra de scroll para de esta manera poder acceder a todas las letras, sin tener que reducir su tamaño y que sigan siendo visibles con facilidad.

En cuanto a los botones del teclado, presentan un texto de la letra que representan en mayúscula, con una proporción 1/1, un tamaño del botón del 4.5 % del ancho de la pantalla, fondo blanco, un fino borde, formato negrita, color negro y un tamaño de letra de 2.5 % del ancho de la pantalla.

El botón de escuchar palabra tiene una proporción 1/1 y un tamaño del 18 % del alto de la pantalla para que nunca se salga de la barra inferior. Presenta la imagen seleccionada en la pantalla 'Seleccionar palabra/imagen', un fino borde y un fondo blanco por si hubiese alguna imagen con transparencias.

- **Modo 'Construye palabras': Componer palabra - Contenedor principal**

  En el contenedor central, fuera de las barras superior e inferior, se mostrará un fondo aleatorio o bien un fondo en blanco, según si hemos pulsado el botón de fondo o no. Además, este será el espacio en el que aparecerán las letras animadas. Las letras podrán ocupar el 90 % del ancho de la pantalla y el 55 % de la altura de la pantalla, ajustándose a esos máximos cada vez que se redimensione la ventana o se posicione una nueva letra. Actualmente todas tienen una proporción 195 / 330, y la base de estas se coloca en el 70 % de la pantalla mirando desde la parte superior. Estas, además, presentan un pequeño solapamiento entre ellas de un 5 %, para dar la sensación de que se dan la mano tras la celebración de acierto.

- **Modo 'Construye palabras': Componer palabra - Responsive**

  Para cumplir el requisito no funcional de que el diseño sea responsivo, además de usar porcentajes relativos al tamaño de la pantalla, en el caso de que la aplicación se ejecute en una pantalla vertical en lugar de una horizontal, se realizarán los siguientes cambios:

  - Los elementos de la barra superior presentarán el mismo formato que el descrito en la pantalla 'Seleccionar letra inicial'.
  - El botón de explicación y el botón de escuchar palabra presentarán un tamaño del 6 % del alto de la pantalla, en lugar del 18 %, para que no ocupen mucho sitio y se pueda ver bien el teclado de letras finales.
  - Los botones del teclado de letras finales, tendrán un tamaño del 7.5 % del ancho de la pantalla, y un tamaño de letra del 5 % del ancho de la pantalla, en lugar del 4.5 %

y 2.5 %, respectivamente.

- **Modo 'Construye palabras': Componer palabra - Menciones especiales**

  Como menciones especiales:

  - Tras pulsar el botón de explicación, los botones de escuchar palabra, eliminar y corregir, se desplazarán aproximadamente al centro de la pantalla y aumentarán de escala en 2.5, 1.8 y 1.8, respectivamente, para después volver a su sitio, uno tras otro.
  - Tras acertar todas las letras y darle al botón corregir, se mostrará una celebración aleatoria antes de que las letras se vayan de la mano.

### 4.1.4. Modo 'Escribe tu nombre'

Tras pulsar el botón del menú inicial 'Escribe tu nombre', se iniciará este modo, el cual es uno de los dos modos principales de la aplicación. Dado que en el transcurso de este modo hay varias pantallas, las dividiremos en subapartados que describiremos a continuación (figura 25).

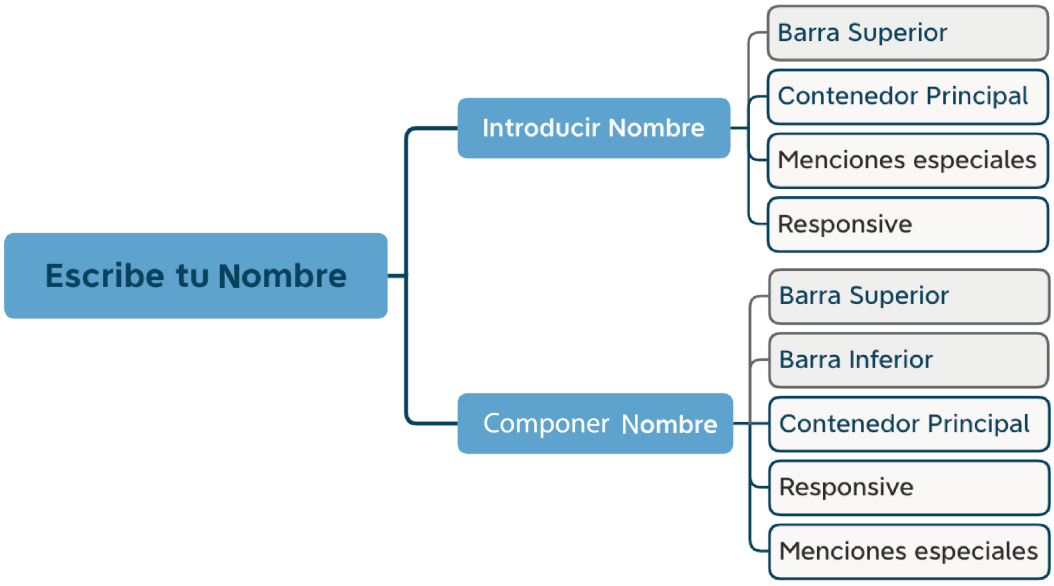


Figura 25: Apartados de las interfaces del modo 'Escribe tu nombre'

- **Modo 'Escribe tu nombre': Introducir nombre**


Figura 26: Pantalla 'Introducir nombre'

La primera pantalla del modo 'Escribe tu nombre', es la de 'Introducir nombre' (figura 26), y se puede acceder a esta pulsando el botón del menú inicial 'Escribe tu nombre', pulsando el botón 'Volver' en la pantalla 'Componer nombre' o acertando el nombre introducido.

- **Modo 'Escribe tu nombre': Introducir nombre - Barra superior**

  En esta pantalla se muestran los mismos elementos de la barra superior que en la pantalla 'Seleccionar letra inicial' del modo 'Construye palabras'.

- **Modo 'Escribe tu nombre': Introducir nombre - Contenedor principal**

  Fuera de la barra superior, se presentan un texto descriptivo con lo que hay que hacer en la pantalla 'Escribe un nombre y pulsa INTRO', y una caja de texto para poder introducir el nombre.

  El texto descriptivo presenta el mismo formato que el de la pantalla 'Seleccionar letra inicial' del modo 'Construye palabras'.

  La caja de texto tiene un ancho del 50 % del ancho de la pantalla, y los caracteres escritos se muestran como asteriscos o puntos porque está definido como tipo password, para que, por ejemplo, el padre del usuario meta el nombre, sin que el propio usuario vea como se escribe.

- **Modo 'Escribe tu nombre': Introducir nombre - Responsive**

  Para cumplir el requisito no funcional de que el diseño sea responsivo, además de usar porcentajes relativos al tamaño de la pantalla, en el caso de que la aplicación se ejecute

en una pantalla vertical en lugar de una horizontal, se realizarán los siguientes cambios:

- Tanto los elementos de la barra superior como el texto descriptivo se comportan de la misma forma que en la pantalla 'Seleccionar letra inicial' del modo 'Construye palabras'.

- **Modo 'Escribe tu nombre': Introducir nombre - Menciones especiales**

  Como mención especial, tras introducir el nombre y pulsar INTRO, si el texto introducido no pasa la validación, aparecerá en pantalla un mensaje, indicando las características que debe tener el nombre.

- **Modo 'Escribe tu nombre': Componer nombre**


Figura 27: Pantalla 'Componer nombre'

La segunda pantalla del modo 'Escribe tu nombre', es la de 'Componer nombre' (figura 27), y se puede acceder a esta introduciendo un nombre válido en la pantalla 'Introducir nombre' y pulsando INTRO.

Esta pantalla se comporta exactamente igual que la de 'Componer palabra' del modo 'Construye palabras' (página 57). Tiene los mismos elementos con los mismos formatos, tanto con la pantalla horizontal como vertical, pero tiene una sola diferencia.

En el botón de escuchar palabra, como en este modo no se ha seleccionado una imagen de una palabra, la imagen que se presenta es la misma imagen representativa del botón 'Escribe tu nombre' de menú inicio.

## 4.2. Interacción de los elementos de la aplicación

Una vez que se han definido los elementos estáticos de la interfaz de usuario, es fundamental describir cómo estos elementos se comportan y comunican entre sí. El apartado de 'Interacción de los Elementos de la Aplicación' se centra en el comportamiento dinámico de la interfaz, que es lo que realmente da vida a la experiencia del usuario.

Este apartado profundiza en cómo la aplicación responde a las acciones del usuario, detallando la lógica y las animaciones que ocurren al interactuar con los componentes. Esto incluye la orquestación de transiciones entre pantallas, la respuesta de los botones y otros elementos interactivos al ser pulsados, y la aparición y desaparición de componentes como fondos o barras de botones.

La interacción de los elementos de la aplicación se lleva a cabo a través del código JavaScript (app.js), que manipula la estructura de la página (index.html) y sus estilos (styles.css) en respuesta a las acciones del usuario. Este proceso crea una experiencia fluida y dinámica, simulando la navegación entre diferentes pantallas sin necesidad de recargar la página.

### 4.2.1. Flujo de interacción en la pantalla de bienvenida

Al iniciar la aplicación, el usuario lo primero que ve es la pantalla de bienvenida (div id='splash-screen' en index.html). Las transiciones que se realizan para mostrar esta pantalla forman parte del método inicializarAplicacion() en app.js que se ejecuta una vez que el DOM se ha cargado completamente.

El contenedor donde se encuentran los textos de la bienvenida está visible por defecto, pero los textos tienen en styles.css la propiedad de opacidad igual a 0 y también tienen definida una transición de opacidad de 1.5 segundos. De ahí que tras que app.js le asigne la opacidad igual a 1 en el método mostrarSplashScreen(), a cada una de las líneas de manera consecutiva y con pausas definidas con setTimeout(), de la API de temporización de JavaScript, estos se muestran escalonados y con una transición suave.

Tras una pausa para que dé tiempo a leer los textos, app.js le asigna opacidad igual a 0 al contenedor de los textos de bienvenida y una vez que son invisibles, le asigna la propiedad display igual a none. Este proceso genera el efecto de fundido de color de los textos, un requisito de usabilidad para una transición suave.

### 4.2.2. Flujo de interacción en la pantalla del menú inicio

La pantalla del menú inicio es donde comienza la interacción directa del usuario. Este es el centro de control de la aplicación, desde donde el usuario toma la decisión de qué modo de juego quiere realizar. Las transiciones que se realizan para mostrar esta pantalla tras la pantalla de bienvenida forman parte del método inicializarAplicacion() en app.js.

1. **Aparición de botones del menú inicio**

   El contenedor de los botones del menú inicio 'Construye palabras' y 'Escribe tu nombre', está oculto por defecto en sus propiedades de styles.css, al igual que el botón de presentación. Tanto el contenedor del menú inicio (div id='menu-inicio' en index.html) como el botón de presentación (<button id='boton-presentacion'>en index.html) tienen definidas propiedades de transición, por lo que cuando desde app.js usamos la función auxiliar fadeIn() para hacer visible el contenedor del menú inicio, los botones 'Construye palabras' y 'Escribe tu nombre' se muestran con una transición fluida.

2. **Aparición de barra superior y botón presentación**

   Tras hacer visible el contenedor del menú inicio, lo siguiente que se muestra es la barra superior ( en index.html). Esta se muestra de una manera diferente. Por defecto es visible, pero está desplazada por encima de la pantalla, por lo que cuando desde app.js la desplazamos hacia abajo, gracias a la propiedad de transición que tiene definida en styles.css, esta aparece con un movimiento fluido hacia abajo hasta ocupar su lugar. Para cuando se muestra la barra, previamente se hicieron visibles el título (<p id='titulo-inicio'>en index.html) y el botón de sonido ( en index.html), por lo que se desplazan hacia abajo junto con la barra.

   A la vez que se desplaza hacia su posición la barra superior, el botón de presentación se hace visible mediante el uso de la función auxiliar fadeIn() en app.js.

3. **Interacción general con botones**

   Para cuando aparecen los botones, la lógica de app.js ya ha hecho que los botones estén activos y listos para recibir clics.

Los distintos botones cuando están habilitados tienen una propiedad (cursor: pointer) en styles.css que modifica el cursor por una mano al pasar el ratón por encima, dando la información al usuario de que es un elemento con el que se puede interactuar, y además tienen definidas propiedades tipo hover, por lo que, al pasar el ratón por encima, estos también reaccionan aumentando su escala y cambiando su sombreado, dando de esta manera feedback al usuario para una interacción más satisfactoria. Por otro lado, tienen definidas propiedades disabled para mostrarse con una mayor opacidad cuando están deshabilitados, además de que el cursor se modifica por una señal de prohibido (cursor: not-allowed).

Si pulsamos el botón de sonido, se iniciará la función de app.js toggleSonidoGlobal(). Este alternará de manera consecutiva entre los distintos modos de sonido cambiando su imagen y texto descriptivo. Además, hay definida una espera de medio segundo para evitar clics múltiples, y durante esta pausa, podremos ver la transición del botón habilitado a deshabilitado, y viceversa. Este botón funcionará de la misma manera en todas las pantallas, permitiendo escuchar todos los sonidos en el modo 'Sonido', escuchando todos los sonidos menos la música de fondo y con un aumento de volumen en los fonemas en el modo 'Fonemas', y sin escuchar ningún sonido en el modo 'Silencio'. Este último modo, hará que la aplicación viaje más rápido de una pantalla a otra porque se omitirán las pausas para escuchar las explicaciones de cada pantalla.

Si pulsamos el botón de presentación, se iniciará la función de app.js reproducirPresentacion(). Todos los botones se deshabilitan por parte de app.js durante el transcurso de la función para evitar interrupciones. Durante esta se reproducen unos audios explicativos, que hablan de la aplicación en la que nos encontramos y sobre las acciones de los botones 'Construye palabras' y 'Escribe tu nombre'. Gracias a propiedades highlight que tienen definidos estos botones en styles.css, mientras se habla de cada uno ellos, estos se resaltan aumentando su escala y sombreado, haciendo que la explicación sea mucho más efectiva y visual. Una vez que acaba la explicación, los botones vuelven a estar habilitados.

Si pulsamos sobre los botones 'Construye palabras' o 'Escribe tu nombre', se iniciará la función de app.js iniciarModoConstruye() o iniciarModoNombre() respectivamente.

Todos los botones se deshabilitan por parte de app.js para evitar clics múltiples y se desvanecen en una transición suave tanto los botones del menú inicio, como el botón de presentación y el título, gracias al uso de la función auxiliar fadeOut(), para posteriormente mostrarse la pantalla pertinente.

Adicionalmente, si volvemos a esta pantalla por el uso del botón volver en otra pantalla, tanto el título, como los botones del menú inicio y el de presentación, aparecen con una transición suave de transparente a opaco con el uso de fadeIn(), pero los botones estarán deshabilitados mientras se escuche un audio en el que se le pregunta al usuario que quiere hacer a continuación, para después habilitar los botones de nuevo, dando feedback al usuario en todo momento.

### 4.2.3. Flujo de interacción en la pantalla 'Seleccionar letra inicial' del modo 'Construye palabras'

La pantalla 'Seleccionar letra inicial' del modo 'Construye palabras', es la primera fase de este modo de juego. Las transiciones que se realizan para mostrar esta pantalla tras la pantalla del menú inicio forman parte del método iniciarModoConstruye() en app.js, pero en caso de ir a esta pantalla pulsando el botón volver ( en index.html) desde 'Seleccionar palabra/imagen', la función de app.js encargada de manejar las transiciones será volverALetras().

1. **Desaparición de elementos de la pantalla previa**

   Tanto iniciarModoConstruye() como volverALetras(), la primera acción que realizan es deshabilitar los botones de la pantalla en la que se encuentran para evitar clics múltiples, para después hacer desaparecer los botones y elementos pertinentes con la función auxiliar fadeOut(), brindando al usuario una experiencia fluida y suave gracias a las propiedades de transición definidas para cada elemento en styles.css.

   En el caso de iniciarModoConstruye(), los elementos que se ocultan son el título, los botones del menú inicio y el de presentación con una transición suave mediante el uso de fadeOut() en app.js. En el caso de volverALetras(), se ocultarán el texto descriptivo de la pantalla 'Elige una imagen' y el teclado de botones de palabras/imágenes con una transición suave mediante el uso de fadeOut() en app.js, además de que se cambiará la

imagen y el texto descriptivo del botón volver de 'Volver' a 'Inicio', y se detendrá la música de fondo de la pantalla 'Seleccionar palabra/imagen'.

2. **Aparición de elementos de la barra superior**

   Lo primero que se muestra en esta pantalla es el texto y la imagen de la puntuación (y <img src='Botones/puntos.png' alt='Puntos'>, respectivamente en index.html), junto con el botón volver, con una transición suave gracias a las propiedades definidas en styles.css. El texto descriptivo del botón volver en esta pantalla es 'Inicio' y presenta una imagen representativa. Además, se reproducirá la música de fondo correspondiente a esta pantalla.

   Hay que añadir que esta puntuación que se irá sumando, se mantendrá durante toda la ejecución de la aplicación independientemente de que se cambie entre los distintos modos principales de juego. Aunque la puntuación desaparezca al volver al menú inicio, cuando aparezca de nuevo al elegir modo de juego, esta será la misma que se obtuvo anteriormente.

3. **Aparición de elementos del contenedor principal**

   Lo segundo que se muestra son el texto descriptivo de la pantalla 'Elige una letra para practicar' (<p id='prompt-letras'>en index.html) y el teclado de letras iniciales ( en index.html) de manera escalonada y con una transición de transparente a opaco gracias a las propiedades que tienen en styles.css, haciendo uso de la función fadeIn() en app.js, tal y como ya hemos explicado para otros elementos.

4. **Interacción general con botones**

   Para cuando aparecen los botones, la lógica de app.js ya ha hecho que los botones estén activos y listos para recibir clics, pero por unos instantes se encontrarán deshabilitados mientras se escucha un audio explicativo de la acción a realizar en la pantalla, para de esta manera mantener siempre informado al usuario. Una vez que acabe el audio explicativo, los botones estarán habilitados.

   Al igual que con los botones de la pantalla del menú inicio, los botones de esta pantalla cuentan en styles.css con las propiedades, (cursor: pointer) y (cursor: not-allowed) para

cambiar el icono del ratón al pasar por encima según si están habilitados o deshabilitados, informando a los usuarios de que son elementos con los que se puede interactuar y dándoles información de su estado. Por otro lado, también cuentan con propiedades tipo hover para que cambien de escala al pasar por encima, y en el caso del botón volver y el botón sonido, también cuentan con una propiedad disabled para mostrarse con una mayor opacidad cuando están deshabilitados. Para los botones del teclado de letras iniciales, no se ha incluido la propiedad disabled para mostrarse más opaco mientras está deshabilitado para evitar parpadeos molestos durante la experiencia del usuario.

Si pulsamos el botón volver, se iniciará la función de app.js handleVolverClick(), que identificará que estamos en esta pantalla y ejecutará las funciones ocultarModoLetrasIniciales() y reiniciarAlInicio(). Se deshabilitarán todos los botones para evitar múltiples clics, se parará la música de fondo, se ocultarán con una transición suave los elementos de la puntuación, el botón volver, y los elementos del contenedor principal. Tras esto se mostrará la pantalla del menú inicio tal y como hemos explicado anteriormente, manteniendo deshabilitados los botones mientras se escucha un audio en el que se le pregunta al usuario que quiere hacer a continuación.

Si pulsamos sobre algún botón del teclado de letras iniciales, se deshabilitarán los botones para evitar clics múltiples, se parará la música de fondo, se precargarán las imágenes de las palabras que empiezan por la letra seleccionada para que a la hora de mostrar el teclado de imágenes/palabras en la siguiente pantalla, no haya tirones, y se ejecutará la función mostrarPalabras() de app.js donde se iniciarán las transiciones para mostrar los elementos de la pantalla 'Seleccionar palabra/imagen'.

### 4.2.4. Flujo de interacción en la pantalla 'Seleccionar palabra/imagen' del modo 'Construye palabras'

La pantalla 'Seleccionar palabra/imagen' del modo 'Construye palabras', es la segunda fase de este modo de juego. Las transiciones que se realizan para mostrar esta pantalla tras la pantalla 'Seleccionar letra inicial', forman parte del método mostrarPalabras() en app.js, pero en caso de ir a esta pantalla pulsando el botón volver o acertando una palabra desde 'Componer palabra', la función de app.js encargada de manejar las transiciones será volverAPalabras().

1. **Desaparición de elementos de la pantalla previa**

   Tanto mostrarPalabras() como volverAPalabras(), la primera acción que realizan es deshabilitar los botones de la pantalla en la que se encuentran para evitar clics múltiples, para después hacer desaparecer los botones y elementos pertinentes con la función auxiliar fadeOut(), brindando al usuario una experiencia fluida y suave gracias a las propiedades de transición definidas para cada elemento en styles.css.

   En el caso de mostrarPalabras(), los elementos que se ocultan son el texto descriptivo de la pantalla 'Elige una letra para practicar' y los botones del teclado de letras iniciales con una transición suave mediante el uso de fadeOut() en app.js. En el caso de volverAPalabras(), se ocultarán los botones de la barra superior de corregir, eliminar y fondo con una transición suave mediante el uso de fadeOut() en app.js, desaparecerá el fondo de pantalla aleatorio desplazándose hacia arriba, y la barra inferior junto con sus botones desaparecerá desplazándose hacia abajo.

2. **Aparición de elementos de la barra superior**

   En el caso de mostrarPalabras(), se dará un cambio en el botón volver en su texto descriptivo de 'Inicio' a 'Volver' además de que presentará una imagen representativa. En el caso de volverAPalabras(), no se producirá ningún cambio en la barra superior más allá de la desaparición de los elementos que hemos comentado ya. Además, se reproducirá la música de fondo correspondiente a esta pantalla.

3. **Aparición de elementos del contenedor principal**

   Lo siguiente que se muestra son el texto descriptivo de la pantalla 'Elige una imagen' (<p id='prompt-palabras'>en index.html) y el teclado de palabras/imágenes ( en index.html) de manera escalonada y con una transición de transparente a opaco gracias a las propiedades que tienen en styles.css, haciendo uso de la función fadeIn() en app.js, tal y como ya hemos explicado para otros elementos.

   Además, en el caso específico del teclado de palabras/imágenes, sus botones no se mostrarán de manera instantánea, sino que hay definida una pausa de varios milisegundos en la constante RETRASO_APARICION_BOTON_MS de app.js, y mediante setTimeout() podemos ver cómo van apareciendo los botones uno a uno, junto con un sonido de 'pop'

cada vez que aparece uno nuevo, haciendo la animación de aparición más dinámica y divertida para el usuario.

4. **Interacción general con botones**

   Para cuando aparecen los botones, la lógica de app.js ya ha hecho que los botones estén activos y listos para recibir clics, pero por unos instantes se encontrarán deshabilitados mientras se escucha un audio explicativo de la acción a realizar en la pantalla, para de esta manera mantener siempre informado al usuario. Una vez que acabe el audio explicativo, los botones estarán habilitados.

   Al igual que con los botones de la pantalla 'Seleccionar letra inicial', los botones de esta pantalla cuentan en styles.css con las propiedades, (cursor: pointer) y (cursor: not-allowed) para cambiar el icono del ratón al pasar por encima según si están habilitados o deshabilitados, informando a los usuarios de que son elementos con los que se puede interactuar y dándoles información de su estado. Por otro lado, también cuentan con propiedades tipo hover para que cambien de escala al pasar por encima, y en el caso del botón volver y el botón sonido, también cuentan con una propiedad disabled para mostrarse con una mayor opacidad cuando están deshabilitados. Para los botones del teclado de imágenes/palabras, al igual que con los del teclado de letras iniciales, no se ha incluido la propiedad disabled para mostrarse más opaco mientras está deshabilitado para evitar parpadeos molestos durante la experiencia del usuario.

   Si pulsamos el botón volver, se iniciará la función de app.js handleVolverClick(), que identificará que estamos en esta pantalla y ejecutará la función volverALetras(). Se deshabilitarán todos los botones para evitar múltiples clics, se parará la música de fondo, y se ocultarán los elementos del contenedor principal. Tras esto se mostrará la pantalla 'Seleccionar letra inicial' tal y como hemos explicado anteriormente, manteniendo deshabilitados los botones mientras se escucha un audio explicativo de la pantalla, para mantener informado al usuario.

   Si pulsamos sobre algún botón del teclado de imágenes/palabras, se deshabilitarán los botones para evitar clics múltiples, y se ejecutará la función mostrarPreviewPalabra() de app.js. Esta función se encargará de crear un elemento 'div' que llamamos 'overlay' y un elemento 'img' que llamamos 'previewImg', que usarán las propiedades definidas

en #preview-overlay y #preview-imagen en styles.css respectivamente para realizar su función. El elemento 'overlay' se encarga de oscurecer toda la pantalla junto con sus elementos para dejarlos en segundo plano y que de esta manera nos centremos en 'previewImg', la cual mostrará en el centro de la pantalla y a gran tamaño la imagen/palabra seleccionada. A continuación, se escuchará la palabra seleccionada, se precargarán los recursos de las letras correspondientes a la palabra para evitar tirones en la siguiente pantalla, desaparecerá el overlay, desaparecerán los elementos del contenedor principal, se detendrá la música y se ejecutará la función crearTeclado() de app.js donde se iniciarán las transiciones para mostrar los elementos de la pantalla 'Componer palabra'.

#### 4.2.5. Flujo de interacción en la pantalla 'Componer palabra' del modo 'Construye palabras'

La pantalla 'Componer palabra' del modo 'Construye palabras', es la tercera y última fase de este modo de juego, y en la que realmente se comienza a practicar la escritura. Las transiciones que se realizan para mostrar esta pantalla tras la pantalla 'Seleccionar palabra/imagen', forman parte del método crearTeclado() en app.js.

1. **Desaparición de elementos de la pantalla previa**

   En el transcurso de ejecución de mostrarPreviewPalabra() en app.js, como ya hemos comentado previamente, se muestra la imagen seleccionada y posteriormente se reproduce el sonido de la palabra, se precargan los recursos de imagen de las letras que contiene la palabra, desaparece el overlay con la imagen de la palabra seleccionada y posteriormente desaparecen el texto explicativo 'Elige una imagen' y el teclado de imágenes/palabras. También se detiene la música de fondo de esta pantalla.

2. **Aparición de elementos de la barra superior**

   El último paso de mostrarPreviewPalabra() es la ejecución de crearTeclado(), la cual se encarga de mostrar la pantalla 'Componer palabra'. En cuanto a la barra superior, aparecerán los botones corregir ( en index.html), eliminar ( en index.html) y fondo ( en index.html), además de los elementos ya existentes en la pantalla 'Seleccionar palabra/imagen'. Estos,

al igual que pasó con el botón volver en la pantalla 'Seleccionar letra inicial', aparecerán con una transición suave de transparente a opaco gracias a las propiedades que tienen asignadas en styles.css y al uso de la función auxiliar fadeIn().

Los botones corregir y eliminar, estarán deshabilitados al comienzo puesto que solo se habilitan en el momento en el que hay al menos una letra colocada. En el caso del botón fondo, sí que estará habilitado desde un principio.

3. **Aparición de elementos del contenedor principal**

   Lo primero que se muestra en el contenedor principal será un fondo de pantalla aleatorio ( en index.html). Este tiene unas propiedades en styles.css por las que mientras app.js no le añada la clase 'visible', se encontrará oculto en la parte superior de la pantalla, por lo que cuando sea visible, este descenderá de manera suave, dando una sensación bastante vistosa y dinámica. Además, se complementa muy bien con la aparición de la barra inferior que se muestra una vez que esté colocado el fondo aleatorio, con un desplazamiento hacia arriba como explicaremos más adelante.

   El contenedor principal también será el escenario donde irán apareciendo y desapareciendo las letras a medida que se vayan seleccionando y eliminando o corrigiendo.

4. **Aparición de elementos de la barra inferior**

   Como ya hemos comentado por encima anteriormente, la barra inferior ( en index.html), se muestra tras ocupar su posición final el fondo aleatorio. Esta tiene unas propiedades en styles.css por las que mientras app.js no le añada la clase 'visible', se encontrará oculta en la parte inferior de la pantalla, por lo que cuando sea visible, esta ascenderá de manera suave hasta su posición, complementando de esta manera la entrada del fondo aleatorio.

   Dentro de la barra inferior, se encontrarán los botones de explicación (<button id='boton-explicacion'>en index.html), de escuchar palabra (<button id='boton-palabra-escuchar'>en index.html) y el teclado de letras finales ( en index.html). En el caso del botón de escuchar palabra, este presentará la imagen de la palabra seleccionada. Además, se reproducirá la música de fondo correspondiente a esta pantalla.

5. **Interacción general con botones**

Para cuando aparecen los botones, la lógica de app.js ya ha hecho que los botones estén activos y listos para recibir clics, pero la primera vez que se acceda a esta pantalla en la ejecución actual de la aplicación, se reproducirá automáticamente la explicación de la pantalla, llevada a cabo por la función reproducirExplicacionTeclado() de app.js. Durante la ejecución de la explicación, se deshabilitarán los botones en pantalla para evitar conflictos, y se explicará el botón escuchar palabra, eliminar y corregir. Gracias a propiedades highlight que tienen estos botones en styles.css, cada vez que se hable de uno de estos botones, se resaltará desplazándolo suavemente hasta el centro de la pantalla y aumentando su escala para posteriormente volver a su tamaño y posición original, dando lugar a una explicación más efectiva y atractiva para el usuario. Una vez acabada la explicación, volverán a estar habilitados los botones en pantalla a excepción del botón corregir y eliminar, que no lo estarán hasta que se coloque al menos una letra.

Al igual que con los botones de la pantalla 'Seleccionar palabra/imagen', los botones de esta pantalla cuentan en styles.css con las propiedades, (cursor: pointer) y (cursor: not-allowed) para cambiar el icono del ratón al pasar por encima según si están habilitados o deshabilitados, informando a los usuarios de que son elementos con los que se puede interactuar y dándoles información de su estado. Por otro lado, también cuentan con propiedades tipo hover para que cambien de escala al pasar por encima, y en el caso de los botones de la barra superior, el botón de explicación y el de escuchar palabra, también cuentan con una propiedad disabled para mostrarse con una mayor opacidad cuando están deshabilitados. Para los botones del teclado de letras finales, al igual que con los del teclado de letras iniciales y el de imágenes/palabras, no se ha incluido la propiedad disabled para mostrarse más opaco mientras está deshabilitado para evitar parpadeos molestos durante la experiencia del usuario.

Si pulsamos el botón volver, se iniciará la función de app.js handleVolverClick(), que identificará que estamos en esta pantalla y ejecutará la función volverAPalabras(). Se deshabilitarán todos los botones para evitar múltiples clics, se parará la música de fondo, las letras colocadas desaparecerán, así como los botones corregir, eliminar y fondo que desaparecerán con una transición suave de opaco a transparente. La barra infe-

rior desaparecerá junto con sus botones desplazándose hacia abajo, y el fondo aleatorio desaparecerá desplazándose hacia arriba. Tras esto se mostrará la pantalla 'Seleccionar palabra/imagen' tal y como hemos explicado anteriormente, manteniendo deshabilitados los botones mientras se escucha un audio explicativo de la pantalla, para mantener informado al usuario.

Si pulsamos el botón fondo, se iniciará la función de app.js cambiarFondo(), que desvanecerá el fondo aleatorio hasta quedarse el fondo blanco en una transición suave gracias a las propiedades que tiene en styles.css. Esto nos permitirá ver las imágenes de las letras con mayor claridad. Si pulsamos de nuevo, volverá a mostrarse el fondo aleatorio que había previamente con la misma transición a la inversa. Como dato adicional, si volvemos a la pantalla 'Seleccionar palabra/imagen' mientras el fondo está en blanco, lo primero que hace el fondo es restaurarse al fondo aleatorio antes de desplazarse hacia arriba para desaparecer.

Si pulsamos el botón eliminar, se iniciará la función de app.js borrarUltimaLetra(). Se deshabilitarán todos los botones para evitar múltiples clics a excepción de los botones de fondo y de sonido, se comprobará cual es la última letra en cuanto a posición, y esta se marchará mediante una animación, realizada por la función animarNegacionYSalida(), en la que se alternarán las imágenes a la vez que se desplaza hacia la derecha de la pantalla para dar la sensación de que está andando hasta salir de pantalla y dejar el hueco libre a otra letra. Tras esto se volverán a habilitar los botones pertinentes.

Si pulsamos el botón corregir, se iniciará la función de app.js corregirPalabra(). Se deshabilitarán todos los botones para evitar múltiples clics a excepción de los botones de fondo y de sonido, se comprobarán las letras y en el caso de haber alguna incorrecta, se reproducirá un sonido de negación y se ejecutará la función animarNegacionYSalida(). Esta hará la animación de negar con la cabeza alternando entre distintas imágenes, para posteriormente irse andando de la misma manera que como hemos explicado para el botón eliminar. Esta comprobación se realiza con previa normalización de la palabra, mediante la función normalizarTexto() de app.js, por lo que no se tendrán en cuenta las tildes, pero si la Ñ. Si no hay letras incorrectas, pero no están todas las letras seleccionadas, no hará nada, pero en caso de estar seleccionadas todas las letras correctas, se

procederá a la ejecución de la función animacionPalabraCorrecta() de app.js. Esta función mostrará una imagen tipo gif de celebración aleatoria, reproducirá unos aplausos aleatorios, reproducirá una felicitación, las letras subirán y bajarán las manos alternadas para dar la sensación de baile, se reproducirá el sonido de la palabra, aumentará la puntuación, las letras se darán las manos y se irán de la pantalla andando con las manos dadas. Tras esto se ejecutará la función de app.js volverAPalabras(), que realizará las acciones descritas anteriormente para el botón volver.

Si pulsamos el botón escuchar palabra, se iniciará la función de app.js handlePalabraEscucharClick(), que identificará el modo de juego y llamará a la función reproducirSonidoPalabra(), que reproducirá el sonido de la palabra/imagen. Mientras se escucha la palabra se deshabilitarán todos los botones menos el de fondo y sonido, para evitar conflictos por dobles clics.

Si pulsamos el botón explicación, se iniciará la función de app.js reproducirExplicacionTeclado(). Esta es la misma función que se ejecuta automáticamente la primera vez que accedemos a esta pantalla en la ejecución actual, por lo que las acciones son las mismas que las anteriormente descritas.

Si pulsamos sobre algún botón del teclado de letras finales, se deshabilitarán todos los botones para evitar clics múltiples, a excepción de los botones de fondo y de sonido, se comprobará si los recursos de imagen de la letra seleccionada ya están precargados y los precarga en caso contrario para evitar tirones durante la animación, y se ejecutará la función gestionarImagenesAnimadas() de app.js. Esta función comprueba cual es la primera posición libre de la palabra seleccionada. Si no hay hueco, no hace nada, pero si lo hay, calcula cual debe ser el tamaño de la letra, así como su posición final basándose en las dimensiones de la pantalla para posteriormente realizar la animación llamando a la función de app.js animarNuevaImagen(). Esta se encargará de definir la frecuencia a la que se alternan las imágenes de la letra para dar la sensación de movimiento, así como cuanto avanza y cada cuanto, hasta llegar a la posición destino. Una vez en el destino, se recalculará la posición y tamaño de todas las letras colocadas por si hubiese habido cambios del tamaño de la pantalla durante la animación, para asegurarnos de que las letras siempre se verán correctamente. Se reproducirá el sonido del fonema correspondiente a

la letra y se reemplazará la imagen por la de la letra cuando está parada. Finalmente se habilitarán los botones pertinentes.

#### 4.2.6. Flujo de interacción en la pantalla 'Introducir nombre' del modo 'Escribe tu nombre'

La pantalla 'Introducir nombre' del modo 'Escribe tu nombre', es la primera fase de este modo de juego. Las transiciones que se realizan para mostrar esta pantalla tras la pantalla del menú inicio forman parte del método iniciarModoNombre() en app.js, pero en caso de ir a esta pantalla pulsando el botón volver desde la pantalla 'Componer nombre' o tras acertar todas las letras correctas en dicha pantalla, la función de app.js encargada de manejar las transiciones será volverAIntroducirNombre().

1. **Desaparición de elementos de la pantalla previa**

   Tanto iniciarModoNombre() como volverAIntroducirNombre(), la primera acción que realizan es deshabilitar los botones de la pantalla en la que se encuentran para evitar clics múltiples, para después hacer desaparecer los botones y elementos pertinentes con la función auxiliar fadeOut(), brindando al usuario una experiencia fluida y suave gracias a las propiedades de transición definidas para cada elemento en styles.css.

   En el caso de iniciarModoNombre(), los elementos que se ocultan son el título, los botones del menú inicio y el de presentación con una transición suave mediante el uso de fadeOut() en app.js. En el caso de volverAIntroducirNombre(), se ocultarán los botones de la barra superior de corregir, eliminar y fondo con una transición suave mediante el uso de fadeOut() en app.js, desaparecerá el fondo de pantalla aleatorio desplazándose hacia arriba, y la barra inferior junto con sus botones desaparecerá desplazándose hacia abajo, además de que se detendrá la música de fondo de la pantalla 'Componer nombre'.

2. **Aparición de elementos de la barra superior**

   Lo primero que se muestra en esta pantalla es la imagen y el texto de la puntuación, junto con el botón volver, con una transición suave gracias a las propiedades definidas en styles.css. El texto descriptivo del botón volver en esta pantalla es 'Inicio' y presenta una imagen representativa. Además, se reproducirá la música de fondo correspondiente a esta pantalla.

Hay que añadir que, como ya explicamos anteriormente, esta puntuación que se irá sumando, se mantendrá durante toda la ejecución de la aplicación independientemente de que se cambie entre los distintos modos principales de juego.

3. **Aparición de elementos del contenedor principal**

   Lo segundo que se muestra son el texto descriptivo de la pantalla 'Escribe un nombre y pulsa INTRO' (<p id='prompt-nombre'>en index.html) y la caja de texto para introducir el nombre (<input type='password' id='input-nombre' maxlength='13'>en index.html) de manera escalonada y con una transición de transparente a opaco gracias a las propiedades que tienen en styles.css, haciendo uso de la función fadeIn() en app.js, tal y como ya hemos explicado para otros elementos.

4. **Interacción general con botones**

   Para cuando aparecen los botones, la lógica de app.js ya ha hecho que los botones estén activos y listos para recibir clics. En esta pantalla, al contrario de las de 'Seleccionar letra inicial' o 'Seleccionar palabra/imagen', no habrá audio explicativo porque está pensada para que sea el adulto que acompaña al usuario quien introduzca el nombre sin que este vea como se escribe. De ahí que, al introducirlo, no se muestren las letras, gracias a type='password' en la definición de la caja en index.html.

   Al igual que con los botones de la pantalla del menú inicio, los botones de esta pantalla cuentan en styles.css con las propiedades, (cursor: pointer) y (cursor: not-allowed) para cambiar el icono del ratón al pasar por encima según si están habilitados o deshabilitados, informando a los usuarios de que son elementos con los que se puede interactuar y dándoles información de su estado. Por otro lado, también cuentan con propiedades tipo hover para que cambien de escala al pasar por encima, y en el caso del botón volver y el botón sonido, también cuentan con una propiedad disabled para mostrarse con una mayor opacidad cuando están deshabilitados.

   Si pulsamos el botón volver, se iniciará la función de app.js handleVolverClick(), que identificará que estamos en esta pantalla y ejecutará las funciones ocultarModoNombre() y reiniciarAlInicio(). Se deshabilitarán todos los botones para evitar múltiples clics, se parará la música de fondo, se ocultarán con una transición suave los elementos de

la puntuación, el botón volver, y los elementos del contenedor principal. Tras esto se mostrará la pantalla del menú inicio tal y como hemos explicado anteriormente, manteniendo deshabilitados los botones mientras se escucha un audio en el que se le pregunta al usuario que quiere hacer a continuación.

Si introducimos un nombre y pulsamos intro, se iniciará la función de app.js handleNombreInput(), que comprobará que el nombre introducido no contenga espacios ni caracteres que no pertenezcan al abecedario español, además de que la longitud máxima sea la definida en MAX_WORD_LENGTH. Esta longitud máxima actualmente está definida en 13 caracteres y responde a la necesidad de que las letras sean fácilmente legibles, pero sin limitarnos demasiado. Si el texto introducido no pasa la validación, se le muestra un mensaje al usuario explicando las limitaciones que debe tener el nombre. Si el nombre es correcto, se precargan los recursos de imagen de las letras que componen el nombre, se ocultan los elementos del contenedor principal con una transición suave gracias a las propiedades de styles.css y al uso de la función auxiliar fadeOut(), se cambia el texto descriptivo del botón volver a 'Volver' junto con una imagen representativa, se detiene la música de fondo de esta pantalla y se ejecuta la función crearTeclado() de app.js, para preparar la pantalla 'Componer nombre'.

### 4.2.7. Flujo de interacción en la pantalla 'Componer nombre' del modo 'Escribe tu nombre'

La pantalla 'Componer nombre' del modo 'Escribe tu nombre', es la segunda y última fase de este modo de juego, y en la que realmente se comienza a practicar la escritura. Las transiciones que se realizan para mostrar esta pantalla tras la pantalla 'Introducir nombre', forman parte del método crearTeclado() en app.js.

Tanto la desaparición de elementos de la pantalla previa como la aparición de botones en la barra superior, la aparición del fondo aleatorio, la aparición de la barra inferior junto con sus botones, así como el funcionamiento de cada uno de estos, es prácticamente idéntico a la explicación dada para la pantalla 'Componer palabra' del modo 'Construye palabras', por lo que en este subapartado nos limitaremos a describir las diferencias solamente.

1. **Diferencias con la pantalla 'Componer palabra' del modo 'Construye palabras'**

La explicación realizada por la función reproducirExplicacionTeclado(), tanto al pulsar el botón explicación como durante su primera reproducción automática, funciona de la misma manera, pero al resaltar el botón de escuchar palabra, se escucha una explicación personalizada para esta pantalla.

Si pulsamos el botón escuchar palabra, se iniciará la función de app.js handlePalabraEscucharClick(), que tras identificar el modo de juego llamará a la función reproducirFonemasDePalabra(), y lo que se escucharán serán los fonemas que componen el nombre introducido uno tras otro, teniendo en cuenta la posible formación de los digrafos 'CH', 'LL' y 'RR'. Mientras se escucha la palabra se deshabilitarán todos los botones menos el de fondo y sonido, para evitar conflictos por dobles clics. Además, la imagen presentada en este botón es la misma que presenta el botón 'Escribe tu nombre' del menú inicio, para que el usuario pueda identificar fácilmente donde escuchar los fonemas del nombre.

Durante la celebración tras acertar todas las letras, la única diferencia es que no se escucha el nombre tal y como se escucha la palabra en la pantalla 'Componer palabra'.

Si pulsamos el botón volver, se iniciará la función de app.js handleVolverClick(), que identificará que estamos en esta pantalla y ejecutará la función volverAIntroducirNombre() que deshabilitará y ocultará los elementos de la pantalla 'Componer nombre' de la misma manera que lo hace volverAPalabras() con la pantalla 'Componer palabras', para posteriormente ejecutar la función de app.js iniciarModoNombre(), que preparará la pantalla 'Introducir nombre' del modo 'Escribe tu nombre' tal y como ya explicamos en su apartado correspondiente.

### 4.2.8. Menciones especiales de interacción

En este subapartado mencionaremos algunas interacciones importantes que no se pueden encasillar en ninguna pantalla específica.

1. **Gestión del audio y la guía vocal**

   En primer lugar, destacaremos el papel de la guía vocal a lo largo de todo el flujo de la aplicación. Hemos mencionado varias veces a lo largo de todo este apartado, que se reproducían explicaciones, ya fuera al pulsar un botón o al mostrarse una nueva pantalla, o que se felicitaba al usuario al acertar una palabra o nombre. Cada una de estas

explicaciones y felicitaciones se llevan a cabo por la guía vocal que da voz a la aplicación. Queríamos remarcarlo porque pensamos que es un gran valor añadido para la aplicación que el usuario se sienta acompañado en todo momento por las explicaciones y felicitaciones de la guía vocal, otorgándole refuerzo positivo para su aprendizaje.

Además, también queremos mencionar la importancia del resto de sonidos del sistema que acompañan al usuario en cada interacción brindando una experiencia de usuario más placentera y completa, tales como los sonidos de los fonemas al seleccionar la letra en 'Componer palabra' o "Componer nombre", o al pulsar el botón de escuchar palabra en 'Componer nombre' que son un recurso de gran importancia para el aprendizaje del usuario, los sonidos de celebración [Epidemic Sound(sf)] junto con los GIFs [Pixabay(sf)] para reforzar positivamente, los sonidos de las palabras tras seleccionar una en 'Seleccionar palabra/imagen' para confirmar la elección, así como los de negación al corregir una palabra, entre otros.

Dada la importancia de este recurso, con el objetivo de ofrecer la mejor calidad posible en este sentido, se usaron herramientas de edición de sonido (Audacity) para normalizar todos los sonidos del sistema vía LUFS para que no hubiese picos de sonido muy altos ni muy bajos, y todo fuera consistente.

2. **Accesibilidad visual y la usabilidad de los pictogramas**

   Para asegurar que la aplicación sea intuitiva y accesible para los niños pequeños, especialmente para aquellos con necesidades educativas especiales, la mayoría de los botones y elementos gráficos se basan en los pictogramas de ARASAAC [Gobierno de Aragón(2024)]. ARASAAC (Aragonés de Comunicación Aumentativa y Alternativa) es un proyecto que ofrece una vasta biblioteca de pictogramas de libre distribución, diseñados para facilitar la comunicación a personas con diversas barreras lingüísticas, cognitivas o funcionales. El uso de estas imágenes estandarizadas y ampliamente reconocidas es beneficioso para la experiencia de usuario porque:

   - **Promueve la accesibilidad**: Los pictogramas proporcionan un sistema de comunicación visual universal que reduce la dependencia de la lectura y la comprensión del lenguaje escrito.

- **Mejora la comprensión**: Las imágenes, claras y sencillas, ayudan a los niños a entender la función de cada botón de manera instantánea, sin necesidad de leer etiquetas. Esto hace que la navegación sea más fluida y natural.
- **Refuerza el aprendizaje**: Al asociar una acción (como escuchar un sonido o ver una animación) con un pictograma, se refuerzan las conexiones cognitivas, lo que contribuye al objetivo educativo del proyecto.

3. **Inicialización asíncrona de los recursos sonoros**

   Dado que los navegadores modernos bloquean la reproducción de sonido sin una interacción previa del usuario, se hace necesario implementar la función handlePrimeraInteraccion() en app.js. Tras hacer click en la aplicación, independientemente de donde se haga, se ejecuta handlePrimeraInteraccion() que llama a la función inicializarAudioConInteraccion(). Esta comprueba si el sonido está activado y si lo está no hace nada, pero si no lo está lo marca como activado y precarga los sonidos principales para que estén listos para reproducirse.

   Además, la función handlePrimeraInteraccion() también está preparada para que, tras pulsar en pantalla por primera vez, se muestre la explicación de reproducirPresentacion(), pero actualmente esta opción está deshabilitada para hacer la aplicación más dinámica. En caso de que el usuario quiera ver la presentación, puede pulsar el botón presentación del menú inicio.

4. **Gestión de audio en segundo plano**

   Para evitar que la aplicación siga sonando cuando cambiamos de pestaña, minimizamos el navegador, bloqueamos el dispositivo móvil o dejamos el navegador en segundo plano, se ha implementado la función handleVisibilityChange() de app.js. Esta escucha a un evento del tipo 'visibilitychange', por lo que en el caso de que la aplicación deje de estar siendo visualizada de manera directa, se encargará de detener todos los sonidos que haya en curso. De igual manera, cuando el usuario se vuelva a centrar en la aplicación, si estamos en alguna pantalla que cuente con música de fondo y nos encontramos en el modo de sonido completo, esta se reanudará.

De esta manera, la aplicación funciona de manera más profesional y ahorrará batería del dispositivo del usuario.

5. **Diseño adaptable y posicionamiento dinámico de elementos**

   Como ya hemos explicado en los distintos apartados de interfaces, los elementos de la aplicación como los botones, los textos, y las barras superior e inferior, tienen definidas sus posiciones y tamaños con proporciones relativas a las dimensiones de la ventana. Además, a parte de las propiedades generales para pantallas horizontales, también están definidas en styles.css varias propiedades encapsuladas en '@media (orientation: portrait)' en el caso de que la pantalla sea vertical. De esta manera, el diseño de la aplicación es responsivo y se adapta automáticamente a los cambios de dimensión u orientación de la pantalla.

   Pero en el caso de las letras que se muestran en el contenedor principal de la pantalla 'Componer nombre' y 'Componer palabra', esto es diferente. Las posiciones y el tamaño que deben tener cada una, se calculan en el momento de su colocación basándose en el tamaño de la ventana y el número de letras de la palabra o nombre para que estas se vean lo mejor posible, cumpliendo los parámetros establecidos por la aplicación en las constantes mencionadas en el apartado de interfaces.

   Esto hace necesaria la implementación de un evento de tipo 'resize', que cada vez que detecte el redimensionamiento o cambio de orientación de la ventana, llame a la función recalcularPosicionesLetras() de app.js. Esta recalcula y ajusta la posición de todas las letras colocadas en pantalla, así como su tamaño.

   Hay que añadir que este evento, no llama a la función recalcularPosicionesLetras() hasta que han pasado más de 100ms sin que se redimensione la pantalla. Esta es una técnica de optimización llamada debounce que nos permite tener un mayor rendimiento al evitar que el evento llame a la función constantemente durante un redimensionamiento de la ventana.

6. **Precarga dinámica de recursos para optimizar la experiencia**

   Para evitar que durante las animaciones de las letras en movimiento dé la sensación de que son estatuas desplazándose, debido al tiempo de descarga de las imágenes necesarias

independientemente de la conexión a Internet del usuario, se hace necesario implementar unas funciones de precarga inteligente de recursos. Dado que durante la creación y presentación del teclado de imágenes/palabras, también se hace uso dinámicamente de varias imágenes para mostrar cada botón, en este caso también es necesario precargar los recursos para evitar tirones y que todo sea fluido.

Dada esta casuística se implementaron 3 funciones de precarga de recursos:

- **precargarRecursosDeLetraInicial()**: Tras pulsar un botón del teclado de letras iniciales en "Seleccionar letra inicial", se encarga de precargar las imágenes de las palabras que comienzan por la letra seleccionada. De esta manera a la hora de mostrar el teclado de imágenes/letras, se hará de manera fluida.
- **precargarRecursosDePalabra()**: Tras seleccionar una palabra o introducir un nombre, se encarga de precargar todos los recursos de imagen de las letras que componen la palabra, como la negación, andar con mano y sin mano dada, y estar paradas, puesto que serán las letras que con seguridad tendrá que usar el usuario. Además, se llevará un registro de las letras que se precargaron completamente para que no se vuelvan a precargar, realizando de esta manera una precarga inteligente.

  Si la letra a precargar ya se precargó completamente, no se hace nada, pero si la letra se precargó parcialmente en precargarRecursosLetraIncorrecta(), se precargarán los recursos que faltan de la letra para que esté precargada completamente, y se registra.
- **precargarRecursosLetraIncorrecta()**: Tras seleccionar una letra incorrecta en el teclado de letras finales para la palabra o nombre que se va a escribir, se encarga de precargar solamente los recursos de la letra que va a utilizar en este caso, que son los de negar, andar sin mano dada, y estar parada. Además, se llevará un registro de las letras que se precargaron parcialmente para que no se vuelvan a precargar de esta manera.

  Si la letra a precargar ya se precargó parcial o completamente, no se hace nada.

Para el caso de las funciones precargarRecursosDePalabra() y precargarRecursosLetraIncorrecta(), la aplicación espera hasta que se acaban de ejecutar. De esta manera, nos aseguramos de que la aplicación funciona de manera dinámica y fluida en el movimiento

de las letras que es algo crucial para la experiencia de usuario, al mismo tiempo que precargamos solamente los recursos necesarios de manera inteligente, ahorrando datos móviles del usuario.

Adicionalmente, en el proceso de optimización de la aplicación, para facilitar la precarga de recursos de imagen de las letras, se usaron herramientas de optimización de imagen (RIOT) para eliminar los metadatos y reducir la resolución de estas sin sacrificar la calidad visual. Gracias a esto, se pudo reducir el peso de las imágenes en aproximadamente un 90 %, pasando de ocupar todos estos recursos 185MB a ocupar 21MB.

## 4.3. Orquestación final

Después de haber analizado el flujo de interacción de los elementos de la aplicación, es fundamental entender cómo todas las partes trabajan en conjunto para crear una experiencia dinámica y fluida. La orquestación final no es más que el trabajo en equipo entre los tres archivos principales (index.html, styles.css y app.js) y las APIs del navegador (figura 28), que actúan como la columna vertebral de todo el sistema.

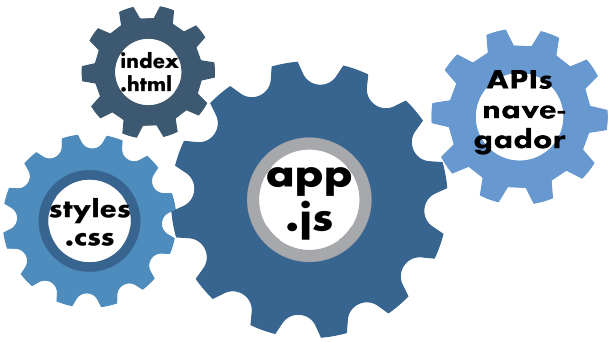


Figura 28: Orquestación final: index.html, styles.css, app.js y las APIs del navegador (imagen adapada de freepik.com)

El modelo de la aplicación, que como ya explicamos anteriormente es una aplicación de página única del lado del cliente (SPA), es lo que permite esta orquestación. No hay un servidor central que vaya ofreciendo cada página, sino que, en su lugar, el navegador del usuario se convierte en un escenario donde tres actores principales interpretan sus roles de manera coordinada.

### 4.3.1. El escenario y los actores (index.html)

El archivo HTML cumple el papel del escenario. Desde el principio, contiene todos los elementos de la interfaz de usuario, incluso los que están ocultos. Los contenedores de las pantallas de bienvenida, el menú de inicio y los modos de juego están todos ahí, como actores esperando detrás del escenario. El HTML no hace nada por sí mismo, sino que simplemente proporciona la estructura y los elementos que la orquesta necesita para funcionar. Es una base estática que contiene la totalidad de la interfaz de usuario de la aplicación.

### 4.3.2. El director de escenografía (styles.css)

El CSS es el responsable de cómo se ve y se mueve el escenario. Define las reglas visuales para todos los elementos, desde sus colores y tipo de letra hasta sus animaciones y su comportamiento en diferentes tamaños de pantalla y orientaciones. El styles.css contiene las directrices para la presentación dinámica. Por ejemplo, las propiedades transition para los fundidos de pantalla están predefinidas aquí, al igual que las propiedades definidas en cabeceras de tipo highlight para resaltar botones, hover para reaccionar al ratón, y disabled para mostrar visualmente que un botón está deshabilitado, entre otras. Espera las órdenes del director para activarse.

### 4.3.3. El director de orquesta (app.js)

El JavaScript es el cerebro y el director de la orquesta y su rol comprende varias funciones:

1. **Escuchar**: Está constantemente a la espera de las acciones del usuario, como clics en los botones, la entrada de texto, redimensionamientos de pantalla o que la aplicación pase a segundo plano.

2. **Procesar**: Ejecuta la lógica del juego. Esto incluye cambiar el estado del sonido, verificar si una letra es correcta o alternar entre modos de juego, entre otros.

3. **Dirigir**: Utiliza las APIs del navegador para manipular los otros dos archivos. Le ordena al index.html qué elementos mostrar y qué ocultar, y le dice al styles.css qué animaciones ejecutar al añadir o quitar clases CSS.

### 4.3.4. La sinfonía final

La orquestación de la aplicación se da en un ciclo constante de evento-proceso-respuesta. El usuario realiza una acción (evento), el app.js la procesa (proceso), y luego el app.js dirige al HTML y al CSS para que cambien la interfaz de usuario (respuesta). Toda esta interacción ocurre íntegramente en el navegador del usuario, lo que crea la ilusión de una aplicación de escritorio fluida y sin interrupciones.

En definitiva, la arquitectura de la aplicación no es solo la suma de sus partes, sino la colaboración compenetrada entre ellas, y al separar las responsabilidades de cada archivo, el sistema es más claro, robusto y fácil de mantener.

# 5

# Ciclos de desarrollo

Tras haber definido las especificaciones y el diseño de la aplicación en el capítulo anterior, el siguiente paso es detallar el proceso de desarrollo que se siguió para convertir esos planos en un producto funcional. Este capítulo se centra en el cómo de la construcción de la aplicación, describiendo la metodología de trabajo, las fases clave y el enfoque adaptativo que se adoptó.

A diferencia de los modelos de desarrollo rígidos, el proyecto se abordó mediante un ciclo de vida iterativo, donde la implementación, las pruebas y la optimización se llevaron a cabo de manera continua. Este enfoque permitió la rápida incorporación de descubrimientos técnicos y la validación constante de funcionalidades.

El presente capítulo desglosa este proceso en las siguientes etapas principales: la fase de investigación y prototipado, la implementación del núcleo de la aplicación, el posterior refinamiento de la interfaz de usuario, las pruebas de calidad y optimización, y los ajustes finales (figura 29).

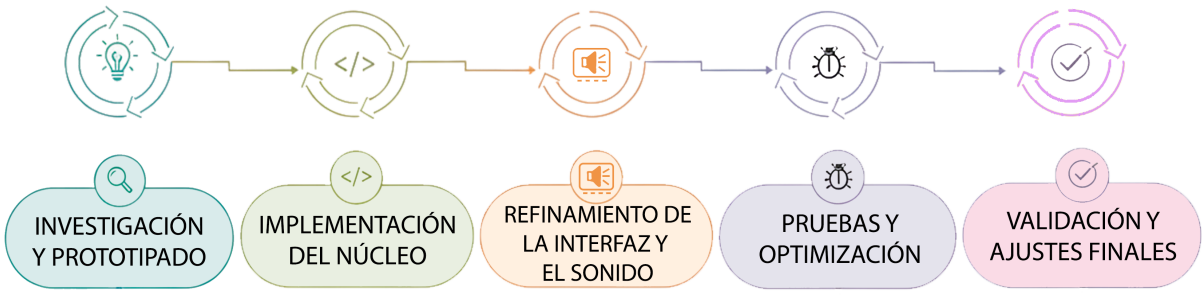


Figura 29: Fases de desarrollo

## 5.1. Metodología de desarrollo: Un enfoque ágil adaptativo

Para el desarrollo de la aplicación, se optó por una metodología ágil y adaptativa. Como se describe en la obra de Sommerville [Sommerville(2015)], este enfoque permite una mayor flexibilidad y capacidad de respuesta al cambio, lo que se ajustó de manera óptima a las características del proyecto. Al ser un Trabajo de Fin de Grado con un único desarrollador, se pudo incorporar nuevos descubrimientos y refinar funcionalidades de forma continua.

La implementación no siguió una secuencia rígida, sino que se realizó a través de iteraciones o pequeños ciclos de trabajo que se centraron en la investigación, la implementación de prototipos y la mejora constante. Este proceso ágil permitió validar la viabilidad técnica de las animaciones y las interacciones en etapas tempranas, evitando la acumulación de problemas y permitiendo una toma de decisiones informada.

A diferencia de un modelo en cascada, donde cada fase debe completarse antes de pasar a la siguiente, este enfoque iterativo facilitó una experimentación controlada. Los prototipos y las pruebas de concepto iniciales permitieron tomar decisiones cruciales, como la elección de JavaScript puro sobre los frameworks, o la optimización de los recursos visuales, lo que aseguró un producto final más robusto, eficiente y alineado con los objetivos del proyecto.

## 5.2. Fases de desarrollo

El proceso de desarrollo de la aplicación se ejecutó en una serie de fases lógicas e iterativas, cada una con objetivos específicos que contribuyeron a la consecución del proyecto final. A lo largo de estas etapas, se implementaron funcionalidades, se validó su comportamiento y se optimizó el rendimiento, demostrando una metodología de trabajo que fue más allá de la simple codificación. A continuación, se desglosa el proceso de desarrollo en las fases clave.

### 5.2.1. Fase de investigación y prototipado

La fase inicial del proyecto fue crucial para sentar las bases del desarrollo y validar las decisiones técnicas antes de la implementación a gran escala. Su principal objetivo fue determinar la viabilidad de la reingeniería de la aplicación original de Flash y elegir las herramientas tecnológicas más adecuadas para su desarrollo.

1. **Análisis de recursos originales**: El primer paso fue una investigación exhaustiva para acceder a los recursos visuales y de audio del proyecto original. Esto incluyó la extracción de las imágenes en movimiento sin manos dadas de las letras animadas y el fondo de pantalla principal, que fueron la base para el rediseño, así como los sonidos e imágenes de la mayoría de las palabras utilizadas, y los sonidos de los fonemas. Para poder acceder a los recursos se utilizó 'Flash Decompiler Trillix'.

   Se realizaron capturas de pantalla fotograma a fotograma de las animaciones, que posteriormente se editaron en un programa de diseño para eliminar los fondos y obtener las letras con fondos transparentes necesarios para las animaciones. Para la edición de imágenes se utilizó 'Adobe Photoshop 2021'.

2. **Pruebas de concepto y elección tecnológica**: Se llevó a cabo un proceso de prototipado para evaluar las distintas opciones de animación. Se realizaron pruebas con la técnica de animación por sprite, utilizando JavaScript y librerías como jQuery para manipular la posición de los fondos. Este proceso de experimentación fue clave para entender las limitaciones y la complejidad de las animaciones, lo que llevó a la decisión estratégica de utilizar JavaScript puro para tener un mayor control de las animaciones, ya que sería algo que facilitaría las cosas en la adición de nuevas animaciones en el futuro. Esta elección permitió un mayor control sobre el comportamiento de la aplicación y un rendimiento optimizado, sin la carga de un framework innecesario para el alcance del proyecto.

### 5.2.2. Fase de implementación del núcleo

Tras validar las decisiones técnicas en la fase de prototipado, el siguiente paso fue la implementación de la funcionalidad central de la aplicación. Esta etapa se centró en construir el modo de juego 'Construye palabras' de manera modular, sentando las bases para el resto de la aplicación.

1. **Desarrollo de la funcionalidad básica**: Inicialmente, se implementó el modo de juego sin las barras superiores e inferiores, con una interfaz minimalista de botones de texto. La lógica principal se desarrolló en este punto:

- **Selección de letra y palabra**: Se implementó la transición entre las pantallas de selección de letra y de palabra, asegurando que la aplicación reaccionara correctamente a la entrada del usuario.
- **Manejo de la animación**: El desafío técnico más importante fue animar las letras. Se implementó una lógica para que las letras se movieran caminando por la pantalla, se detuvieran en su posición correcta y se corrigieran en caso de error, marchándose, andando.

  Para el movimiento de las letras, se hace uso de la función setInterval() de la API de temporización de JavaScript. Se usa un setInterval() para que la imagen de la letra se alterne de manera constante entre las distintas capturas de movimiento, y otro setInterval() para que la imagen, se desplace horizontalmente hasta su destino, dando la sensación de que la letra está andando. Se puede configurar tanto la velocidad de alternancia entre imágenes como el desplazamiento horizontal en unidad de tiempo, gracias a constantes definidas en la región de constantes de app.js.

2. **Preparación y normalización de recursos**: A medida que la funcionalidad del juego tomaba forma, se hizo evidente la necesidad de optimizar los recursos visuales. Se realizó una edición exhaustiva de las imágenes de las letras animadas para asegurar que todas tuvieran el mismo tamaño y la misma proporción. Esta normalización fue crucial, ya que resolvió el problema de escala que hacía que las letras pequeñas se vieran borrosas y las grandes se vieran encogidas, garantizando así una experiencia visual consistente y de alta calidad.

3. **Implementación de las animaciones de interacción**: Finalmente, se añadieron las animaciones que enriquecían la interacción. Esto incluyó la creación y edición de nuevas imágenes para las animaciones de negación, darse la mano y andar con las manos dadas, así como la implementación en la lógica de app.js de estas nuevas animaciones de las letras.

   Para el caso de la animación de darse la mano, las nuevas imágenes se diseñaron para que todas las letras acabaran colocando la mano en el mismo punto imaginario en los extremos derecho e izquierdo de la imagen, por lo que al tener todas las imágenes el mismo tamaño, los dedos de las manos de las distintas letras se estarían tocando. Es-

to junto a que, al colocar las letras, se ponen una seguida de la otra con un pequeño solapamiento del 5 %, da la sensación de que se están dando la mano.

Estos pequeños detalles, aunque complejos de implementar, fueron vitales para cumplir el objetivo del proyecto de recrear el encanto de la aplicación de Flash.

### 5.2.3. Fase de refinamiento de la interfaz y sonido

Una vez que el núcleo del juego estuvo funcional, la siguiente fase se centró en la integración de la lógica en una interfaz de usuario completa y pulida. Esta etapa transformó los prototipos básicos en la aplicación visualmente atractiva y funcional que se presenta.

1. **Integración de barras de navegación**: Se añadieron las barras superior e inferior, elementos clave de la interfaz que proporcionaron una navegación consistente en todas las pantallas. Se implementaron botones dinámicos en la barra superior (como los de 'Inicio', 'Volver' y 'Sonido') y en la barra inferior, que servían como el teclado de letras finales.

2. **Adición de modos de juego y pantallas complementarias**: Basándose en la lógica ya desarrollada para el modo 'Construye palabras', se implementó el modo 'Escribe tu nombre' para reutilizar funcionalidades como el teclado de letras finales y la animación. Se crearon también las pantallas de bienvenida y el menú principal, que actuaron como puntos de entrada y navegación para el usuario.

3. **Implementación y normalización del sistema de audio**: Se llevó a cabo un proceso de normalización de todos los recursos de audio para garantizar una experiencia sonora uniforme, usando la herramienta de edición de audio 'Audacity'. Se implementaron los sonidos del sistema, las explicaciones y los fonemas en cada modo de juego. Además, se añadió la funcionalidad de gestión de sonido, permitiendo al usuario controlar si escucha todos los sonidos incluyendo la música de fondo (modo sonido), todos los sonidos menos la música de fondo con un aumento de volumen en los fonemas (modo fonemas), o si silencia completamente la aplicación (modo silencio). Este nivel de control contribuyó a una experiencia más accesible y personalizable.

### 5.2.4. Fase de pruebas y optimización

La fase de pruebas y optimización fue crítica para garantizar que la aplicación no solo funcionara según lo previsto, sino que también ofreciera una experiencia de usuario fluida y de alto rendimiento en todos los dispositivos. Esta etapa se centró en identificar y solucionar los cuellos de botella y los problemas de compatibilidad que surgieron durante el desarrollo.

1. **Pruebas funcionales y detección de errores**: Se llevaron a cabo pruebas exhaustivas para verificar el correcto funcionamiento de todas las funcionalidades. Este proceso permitió identificar y corregir problemas, como la lógica de habilitar y deshabilitar botones para evitar condiciones de carrera, y asegurar que las transiciones entre pantallas fueran fluidas y uniformes. Estas pruebas fueron esenciales para validar la lógica del juego y la interacción de la interfaz.

2. **Pruebas en dispositivos reales y optimización de rendimiento**: Hasta este momento, se había hecho uso de un repositorio privado de GitHub como control de versiones, pero como todas las pruebas se habían realizado en local y existía la necesidad de probar la aplicación desde un servidor, esta se subió a un repositorio público de GitHub que nos permitía hacer uso de GitHub Pages, y probar la aplicación lanzada desde un servidor. Gracias a esto se pudieron realizar pruebas en una variedad de dispositivos (PC, móvil y tableta). Estas pruebas revelaron tres problemas principales:

    - **Tirones en las animaciones**: Los 'tirones' eran causados por la carga de las imágenes de animación, que eran visualmente pesadas. Para solucionarlo, se implementó una precarga inteligente de recursos y se optimizaron las imágenes, reduciendo su peso en un 90 % sin sacrificar la calidad visual, mejorando de esta manera la fluidez de la aplicación al mismo tiempo que reducimos el gasto de datos móviles del usuario. Para la optimización de los recursos de imagen se usó 'RIOT (Radical Image Optimization Tool)'.
    - **Problemas de sonido**: Se descubrió que el sonido continuaba reproduciéndose cuando la aplicación pasaba a segundo plano. Esto se resolvió con la implementación del evento 'visibilitychange', que permite detener la reproducción del audio

cuando el usuario no está en la aplicación, mediante la función handleVisibilityChange() de app.js, mejorando la lógica de la aplicación y ahorrando batería del dispositivo del usuario.

- **Inconsistencias visuales**: Tras probar las distintas orientaciones de pantalla, en dispositivos de distinto tamaño se identificaron problemas en la presentación de algunos textos y botones en pantalla, que se corrigieron mediante el ajuste de propiedades en styles.css. Esto por ejemplo permitió que apareciera automáticamente una barra de desplazamiento en el teclado de letras finales en 'Componer palabra' y 'Componer nombre', cuando los botones no caben en la barra inferior, permitiendo que los botones no se salgan de la barra, al mismo tiempo que se permite mantener un tamaño de los botones lo suficientemente grande en pantallas pequeñas como para poder tener una buena experiencia de usuario.

  Además, se identificó que, al cambiar el tamaño de la pantalla, habiendo letras ya colocadas, estas no se adaptaban como el resto de los elementos, dando lugar a letras colocadas en distintas posiciones y con distintos tamaños. Esto se resolvió con la implementación del evento 'resize', que llama a la función recalcularPosicionesLetras(), asegurándonos de esta manera de que todos los elementos en pantalla, siempre se muestren correctamente. Hay que añadir, que el evento 'resize', hace uso de una técnica de optimización de rendimiento, llamada 'debounce', por la que en lugar de llamar constantemente a recalcularPosicionesLetras() durante el redimensionamiento de la pantalla, este no llama a la función hasta que han pasado 100ms sin que se esté redimensionando, mejorando de esta manera la experiencia de usuario, sin un alto coste para el rendimiento.

3. **Pruebas con usuarios finales**: Además de que todo funcionara según fue diseñado, también era necesario comprobar que la aplicación era atractiva para el público objetivo. Para estas pruebas comprobamos con un usuario de 5 años, como reaccionaba a las distintas interacciones con la aplicación. De esta manera, pudimos comprobar, que por ejemplo gracias al sonido de los fonemas, el usuario pudo llegar a escribir una palabra que no sabía escribir previamente. También comprobamos que las explicaciones resaltando los elementos en pantalla eran efectivas y útiles para el usuario, y que este

reaccionaba de manera muy positiva a las distintas animaciones de las letras, así como a las celebraciones tras acertar, queriendo probar a escribir palabras que comenzaran con todas las letras.

4. **Reorganización y comentado del código**: Hasta este punto se había hecho uso de una estructura por regiones y comentarios explicativos, para facilitar su lectura y entendimiento, así como el mantenimiento por parte de otros desarrolladores, pero dado que, en este punto, se había llegado prácticamente a la versión final de la aplicación, se realizó una batida completa al código, para reestructurar tanto app.js por regiones como styles.css con comentarios a modo de regiones, por funcionalidad. También se complementó el comentado explicativo del código. De esta manera, a la hora de leer el código, se hace más fácil entender el flujo de la aplicación y donde se implementa cada cosa, para su efectivo mantenimiento.

En resumen, esta fase no solo sirvió para corregir errores, sino también para mejorar significativamente el rendimiento y la usabilidad de la aplicación, demostrando una sólida capacidad de depuración y optimización.

### 5.2.5. Fase de validación y ajustes finales

La última fase del desarrollo fue crucial para validar la calidad del producto y asegurar su alineación con los requisitos iniciales. Este proceso se centró en la validación externa y la incorporación de la retroalimentación, lo que llevó a las mejoras finales y a la versión definitiva de la aplicación.

1. **Incorporación de la retroalimentación**: La presentación del proyecto a la tutora permitió recibir una retroalimentación constructiva que mejoró la usabilidad. Se sugirieron dos cambios clave:

    - **Botón de eliminación de letra**: Se añadió un nuevo botón en la barra superior para permitir a los usuarios eliminar la última letra colocada. Esta funcionalidad, que no existía en el diseño original, mejoró la experiencia del usuario al proporcionar un mayor control sobre el proceso de construcción de la palabra o nombre.

- **Activación del botón de corrección**: Se modificó la lógica para que el botón de corrección estuviera disponible desde la primera letra colocada en la pantalla, en lugar de esperar a que la palabra estuviera completa. Esto permitió una validación en tiempo real y guía al usuario de manera más eficiente.

2. **Validación del producto final**: Estos ajustes finales, junto con las optimizaciones realizadas en las etapas anteriores, consolidaron la versión final de la aplicación. Se llevaron a cabo pruebas finales para confirmar que todas las funcionalidades y mejoras se habían implementado correctamente, resultando en un sistema robusto, eficiente y listo para su uso. Esta fase demostró la capacidad del proyecto para adaptarse a la retroalimentación y evolucionar hacia una solución más completa.

# 6
# Conclusiones

## 6.1. Conclusiones

Este Trabajo de Fin de Grado ha logrado con éxito el objetivo de reingeniería de una aplicación educativa obsoleta, migrándola de una plataforma propietaria y descontinuada como Flash a un entorno web moderno y universal. El proyecto no solo ha replicado la funcionalidad de la versión original, sino que le ha añadido una serie de mejoras, demostrando la viabilidad de utilizar tecnologías web nativas para crear herramientas educativas robustas y de alto rendimiento.

El objetivo general de desarrollar una herramienta que facilite el aprendizaje de la lectura basado en el método Letrilandia con letras animadas se ha cubierto de manera satisfactoria. La aplicación 'Las Letras Que Andan' es un software educativo funcional que cumple con las expectativas iniciales.

El primer subobjetivo, que consistía en desarrollar un sistema que resultara atractivo para los niños y que los anime a aprender a leer, se ha logrado a través de varias decisiones de diseño e implementación. La interfaz de usuario es intuitiva y visualmente atractiva, utilizando elementos interactivos que fomentan la participación activa de los pequeños. El diseño se ha centrado en mejorar la experiencia de usuario (UX) mediante animaciones fluidas, un sistema de guía vocal que acompaña al niño en todo momento y el uso de pictogramas ARASAAC para asegurar la accesibilidad visual y la comprensión intuitiva de los botones. Este enfoque lúdico y cuidadosamente diseñado transforma el aprendizaje de la lectoescritura en una actividad divertida, un aspecto clave para captar y mantener la atención del público infantil.

El segundo subobjetivo, la implementación en JavaScript de una versión animada de los elementos del método Letrilandia, ha sido resuelto con éxito mediante la elección de tecnologías web nativas. La aplicación fue diseñada como una SPA (Aplicación de página única), lo que la mantiene ligera, rápida y portátil. El uso de JavaScript puro, HTML5 y CSS3 ha de-

mostrado ser una elección técnica acertada, ya que permitió una optimización meticulosa de recursos, como la compresión de imágenes y una precarga inteligente, lo que garantiza un rendimiento óptimo en diversos dispositivos, incluso sin conexión a internet. La implementación fue lo suficientemente robusta como para replicar y mejorar las animaciones e interacciones del proyecto original de Flash, asegurando la alta calidad del producto final sin la dependencia de software propietario.

## 6.2. Líneas futuras

Para continuar con la evolución del proyecto y expandir su utilidad, se proponen las siguientes líneas de desarrollo futuro:

1. **Nuevos modos de juego y funcionalidades**: Se podría expandir la oferta educativa con la inclusión de nuevos modos de juego, como un 'Modo Aventura' que guíe al usuario a través de un recorrido de aprendizaje secuencial, un modo en el que se trabaje a nivel de sílabas en lugar de a nivel de letras, o la posibilidad de que los padres y educadores puedan crear sus propias listas de palabras para personalizar la experiencia.

2. **Almacenamiento del progreso**: Actualmente, la aplicación no guarda el progreso del usuario, tras acabar la sesión de juego. Una buena mejora sería la implementación de una base de datos local o la integración con una base de datos en la nube para permitir que el progreso y los logros del usuario se almacenen de forma persistente a lo largo de las sesiones. Esto podría incluir la gestión de múltiples usuarios, lo que permitiría a la aplicación ser utilizada en un entorno de aula.

3. **Expansión a nuevas plataformas**: Aunque la aplicación funciona en cualquier navegador, una siguiente etapa podría ser empaquetarla como una PWA (Aplicación web progresiva), lo que permitiría instalarla en dispositivos móviles y de escritorio sin necesidad de una tienda de aplicaciones. Además, se podría considerar una versión nativa para iOS y Android utilizando frameworks de desarrollo híbrido como React Native o Flutter.

En conclusión, este proyecto TFG no solo ha demostrado ser un ejercicio de reingeniería de software exitoso, sino que también ha validado la viabilidad y el potencial del desarrollo

web moderno para crear soluciones educativas innovadoras. Al transformar un programa obsoleto en una aplicación SPA robusta y adaptable, se ha sentado una base sólida para futuras evoluciones y mejoras.

# Referencias


[Usero Alijarde(2004)] A. Usero Alijarde, Letrilandia, Libro de lectura, volume 1-6, Edelvives, Madrid, 2004.

[Junta de Andalucía(2017)] Junta de Andalucía, La alfabetización inicial. Aprender a leer y escribir, 2017. URL: https://www.juntadeandalucia.es/educacion/portals/delegate/content/1d42a078-8dc4-4098-906d-7ce55b4e9c4b/La%20alfabetizaci%C3%B3n%20inicial.%20Aprender%20a%20leer%20y%20escribir.

[Miller(sf)] C. Miller, Cómo aprenden los niños a leer, s.f. URL: https://childmind.org/es/articulo/como-aprenden-los-ninos-a-leer/, versión en PDF del artículo.

[Almanza(2013)] M. T. Almanza, Métodos de lectura en español, Master's thesis, University of Texas-Pan American, 2013. URL: https://scholarworks.utrgv.edu/leg_etd/885.

[López(2017)] L. M. López, La didáctica de la lectoescritura, Editorial Síntesis, Madrid, 2017.

[García et al.(1981)García, Sahuquillo, and Martínez] F. García, M. I. Sahuquillo, P. Martínez, Micho. Método de lectura, Editorial Bruño, Madrid, 1981.

[Elera Seclen(2023)] A. C. Elera Seclen, Letrilandia como método para la lectoescritura en niños de cinco años de una institución educativa de Chiclayo, Master's thesis, Universidad Católica Santo Toribio de Mogrovejo, 2023. URL: https://alicia.concytec.gob.pe/vufind/Record/UCVV_4feb866c8f38c8d5f213d55c4e8b911c/Cite.

[Palau(1987)] M. d. C. Palau, Palau. Método de lectoescritura, Editorial La Pizarra, Madrid, 1987.

[Pascual(2013)] M. A. Pascual, La didáctica de la lectoescritura en Educación Infantil y Primaria, Editorial Universitas, Madrid, 2013.

[Adobe Inc.(2017)] Adobe Inc., Adobe anuncia el fin de Flash, 2017. URL: https://blog.adobe.com/en/publish/2017/07/25/adobe-flash-update.

[Jiménez Núñez(2009)] N. Jiménez Núñez, Lectoescritura en grupo para educación infantil con un método fonético, Trabajo de fin de grado, Universidad de Málaga, 2009.

[Aguado(2017)] V. Aguado, Desarrollo de Aplicaciones Web: Tecnologías y Frameworks, Editorial RA-MA, Madrid, 2017.

[Morales(2011)] P. Morales, Diseño de interfaces de usuario para el aprendizaje: principios y modelos, McGraw-Hill Interamericana, Madrid, 2011.

[Gobierno de Aragón(2024)] Gobierno de Aragón, ARASAAC: Aragonese Portal of Augmentative and Alternative Communication, 2024. URL: https://www.arasaac.org.

[Sommerville(2015)] I. Sommerville, Ingeniería del software, 10 ed., Pearson, Madrid, 2015.

[Fowler(2004)] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3 ed., Addison-Wesley, Boston, 2004.

[Pressman(2010)] R. S. Pressman, Ingeniería del software: un enfoque práctico, 7 ed., McGraw-Hill Interamericana, México, 2010.

[Nielsen and Norman(2013)] J. Nielsen, D. A. Norman, The Design of Everyday Things, revised and expanded edition ed., Basic Books, New York, 2013.

[Epidemic Sound(sf)] Epidemic Sound, Applause Sound Effect, https://www.epidemicsound.com/es/sound-effects/categories/crowds/applause, s.f. Efecto de sonido de aplausos.

[Pixabay(sf)] Pixabay, Gifs, s.f. URL: https://pixabay.com/es/gifs/.

# Apéndice A
# Manual de usuario: Guía rápida de modos de juego

Este manual describe las dos experiencias de juego principales de la aplicación 'Las Letras Que Andan'. La interfaz ha sido diseñada para ser intuitiva para niños en edad preescolar, guiándoles a través de una serie de actividades lúdicas.

Lo primero será la selección del modo de juego en el menú inicio (figura 30), tras lo cual veremos una transición hacia el modo de juego seleccionado.

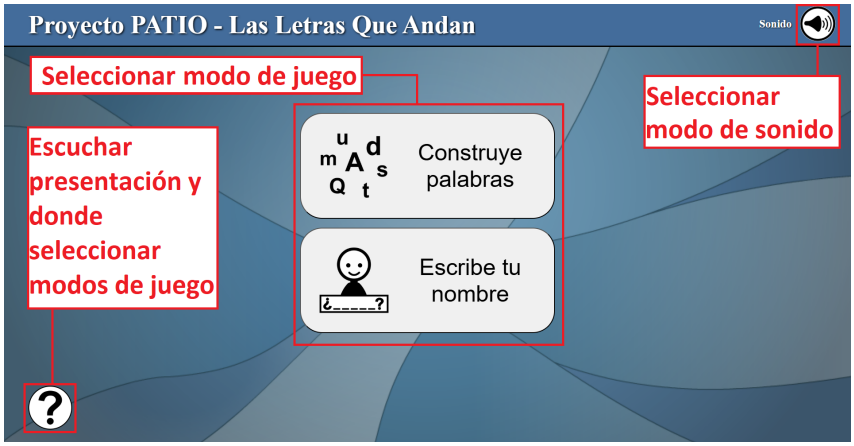


Figura 30: Explicación de la pantalla del menú inicio

## A.1. Modo: 'Construye palabras'

Este modo está diseñado para que el usuario aprenda a formar palabras a partir de una letra inicial dada. El modo se divide en tres pantallas secuenciales.

### A.1.1. Pantalla 'Seleccionar letra inicial'

- **Objetivo**: Elegir la letra con la que el usuario desea comenzar a trabajar (figura 31).

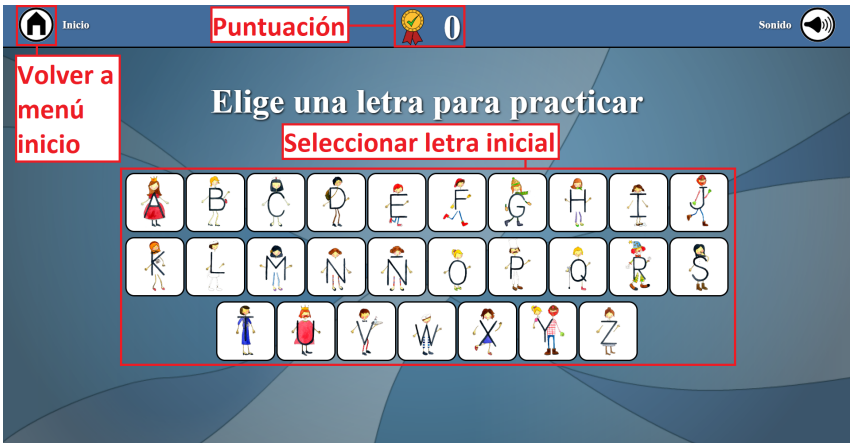


Figura 31: Explicación de la pantalla de selección de letra inicial

- **Interacción**: Se presentan todos los personajes de las letras de Letrilandia en pantalla. La guía vocal indica que se debe seleccionar una letra para practicar y el usuario puede tocar o hacer clic en el personaje que desee. Una vez seleccionado se inicia la transición a la siguiente pantalla.

### A.1.2. Pantalla 'Seleccionar palabra/imagen'

- **Objetivo**: Elegir una palabra que comience con la letra seleccionada en la pantalla anterior (figura 32).


Figura 32: Explicación de la pantalla de selección de palabra

- **Interacción**: En la pantalla se muestran varias imágenes de objetos o animales cuyo nombre comienza con la letra elegida. La guía vocal pide al usuario que elija una palabra.

Al seleccionar una imagen, se reproduce su nombre y se inicia una animación que guía al usuario a la siguiente pantalla.

### A.1.3. Pantalla 'Componer palabra'

- **Objetivo**: Formar la palabra seleccionada, pulsando en el teclado de la barra inferior las letras que la componen (figura 33).

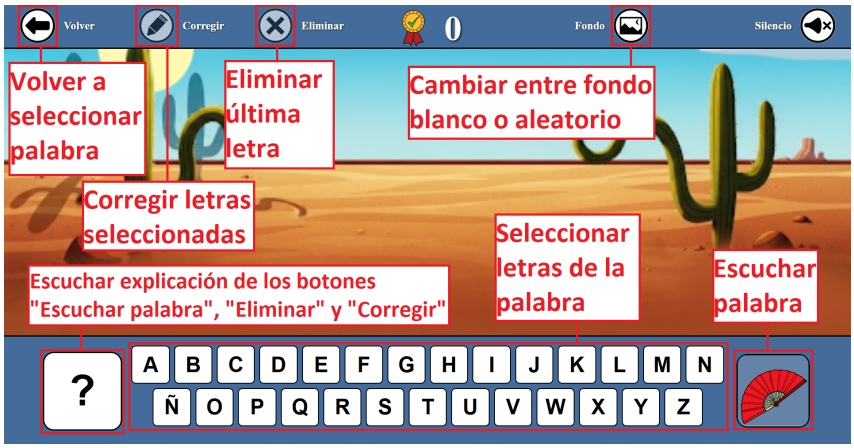


Figura 33: Explicación de la pantalla de composición de palabra

- **Interacción**: La imagen seleccionada se muestra en la parte inferior derecha, pudiendo escuchar la palabra pulsando en la imagen. En el teclado de la parte inferior, se podrán seleccionar las letras que forman la palabra. El usuario debe seleccionar las letras correctas en el orden correspondiente para construir la palabra. La aplicación ofrece retroalimentación visual y auditiva al seleccionar cada una de las letras. Al seleccionar todas las letras correctas y pulsar el botón Corregir de la barra superior, se activa una animación de celebración y se aumenta la puntuación del usuario en un punto.

## A.2. Modo: 'Escribe tu nombre'

Este modo permite al usuario aprender las letras que forman su propio nombre de una manera personalizada y divertida. El modo se divide en dos pantallas secuenciales.

### A.2.1. Pantalla 'Introducir nombre'

- **Objetivo**: Escribir el nombre del usuario para que la aplicación lo reconozca y adapte el juego (figura 34).


Figura 34: Explicación de la pantalla de introducir nombre

- **Interacción**: Un adulto utiliza un teclado virtual en pantalla o un teclado físico para introducir el nombre. La aplicación solo acepta nombres con una longitud máxima de 13 letras, sin espacios. Si el nombre introducido no cumple las limitaciones, saltará un mensaje que informará al adulto para que lo intente de nuevo. Una vez que se ha escrito el nombre y pulsado INTRO, se inicia la transición al juego.

### A.2.2. Pantalla 'Componer nombre'

- **Objetivo**: Identificar y seleccionar las letras del propio nombre para componerlo (figura 35).

- **Interacción**: Esta pantalla es similar a la de 'Componer Palabra', pero adaptada al nombre introducido. La imagen del modo de juego 'Escribe tu nombre' se muestra en la parte inferior derecha, pudiendo escuchar los fonemas de las letras que componen el nombre pulsando en la imagen. En el teclado de la parte inferior, se podrán seleccionar las letras que forman el nombre. El usuario debe seleccionar las letras correctas en el orden correspondiente para construir el nombre. La aplicación ofrece retroalimentación visual y auditiva al seleccionar cada una de las letras. Al seleccionar todas las letras correctas y


Figura 35: Explicación de la pantalla de composición de nombre

pulsar el botón Corregir de la barra superior, se activa una animación de celebración y se aumenta la puntuación del usuario en un punto.

# Apéndice B
# Manual de instalación

La aplicación web ha sido desarrollada con tecnologías estándares y puede ser instalada y ejecutada localmente o en un servidor web.

## B.1. Requisitos del sistema

- **Navegador web**: Un navegador moderno compatible con HTML5, CSS3 y JavaScript (como Google Chrome, Mozilla Firefox o Microsoft Edge).
- **Servidor web (opcional)**: Un servidor web local o remoto, como Apache o Nginx, si se desea acceder a la aplicación a través de una URL.

## B.2. Pasos de instalación

1. **Descargar los archivos**: Obtén una copia completa del código fuente del proyecto. El proyecto se compone de los siguientes archivos principales:
    - index.html: Estructura de la página principal.
    - styles.css: Hojas de estilo y animaciones.
    - app.js: Lógica de la aplicación y animaciones.
    - Resto de carpetas: Carpetas que contienen todas las imágenes, audios y otros recursos.
2. **Ejecución local (sin servidor web)**: Simplemente abre el archivo index.html en tu navegador web. La aplicación se cargará y ejecutará correctamente sin necesidad de un servidor.

3. **Ejecución en un servidor web**: Sube todos los archivos y carpetas del proyecto a la carpeta raíz (public_html o similar) de tu servidor web. Una vez subidos, la aplicación estará disponible en la URL de tu servidor.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA