TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN SISTEMAS DE LA INFORMACIÓN

Desarrollo de una plataforma educativa en favor del aprendizaje de la ortografía española para niños en edad temprana a través del uso de la tecnología y ludificación

Estudiante: Mateo Osende González

Dirección: Óscar Pedreira Fernández

A Coruña, abril de 2024

A mi familia, por motivarme a continuar. A mí, por hacerlo.

Agradecimientos

A todos los amigos y compañeros de trabajo encontrados en el camino, a la empresa Singular Factory por la confianza y el apoyo, y a Óscar Pedreira por ser mi tutor y acompañante en esta última etapa.

Resumen

Este proyecto tiene como objetivo central el desarrollo de una herramienta educativa que potencie las habilidades ortográficas en niños en edad temprana, haciendo uso de la tecnología y estrategias de ludificación. Se ha creado una plataforma web interactiva que emplea inteligencia artificial para generar dictados adaptados al nivel del usuario. Además, se ha diseñado un algoritmo de corrección propio que ofrece retroalimentación personalizada, identificando y priorizando los errores ortográficos para una mejora efectiva. Para optimizar la retención a largo plazo, se ha integrado el algoritmo de Leitner, facilitando así el proceso de aprendizaje. La inclusión de elementos lúdicos como puntos, niveles y desafíos tiene como propósito motivar activamente a los usuarios y mantener su interés en el aprendizaje. Esta propuesta representa un avance en la enseñanza de la ortografía, ofreciendo una experiencia educativa envolvente y efectiva que promueve el desarrollo de habilidades lingüísticas de manera atractiva para los niños.

Abstract

This project aims to develop an educational tool that enhances spelling skills in young children through the utilization of technology and gamification strategies. An interactive web platform has been created, leveraging artificial intelligence to generate dictations tailored to the user's level. Additionally, a proprietary correction algorithm has been designed to provide personalized feedback, identifying and prioritizing spelling errors for effective improvement. To optimize long-term retention, the Leitner algorithm has been integrated, streamlining the learning process. The incorporation of gamified elements such as points, levels, and challenges aims to actively motivate users and sustain their interest in learning. This proposal represents a step forward in spelling education, offering an engaging and effective educational experience that promotes the development of language skills in an appealing manner for children.

Palabras clave:

- JavaScript
- HTML
- CSS
- Ludificación
- Gamificación
- Aprendizaje
- Leitner

Keywords:

- JavaScript
- HTML
- CSS
- Ludification
- Gamification
- Learning
- Leitner

Índice General

Índice de Figuras

Índice de tablas

1 Introducción

En la actualidad, la integración de la tecnología en el ámbito educativo ha revolucionado la forma en que los niños adquieren conocimientos. El acceso a dispositivos digitales ha transformado el proceso de aprendizaje, proporcionando oportunidades sin precedentes para la exploración, la interactividad y la personalización del aprendizaje. Con la creciente importancia de las habilidades digitales en el panorama global, se ha vuelto imperativo desarrollar herramientas educativas innovadoras que no solo faciliten el aprendizaje, sino que también lo hagan más atractivo y efectivo.

En este contexto, surge la necesidad de abordar áreas específicas del aprendizaje, como la ortografía, que son fundamentales para el desarrollo del lenguaje escrito. La ortografía española, en particular, presenta desafíos significativos para los niños en edad temprana, ya que requiere la memorización de reglas complejas y la práctica constante para su dominio. Es en este punto donde el presente trabajo cobra relevancia al centrarse en la creación de una experiencia educativa única destinada a mejorar la ortografía española en niños en edad temprana.

Al aprovechar las posibilidades que ofrece la tecnología, este trabajo busca superar las limitaciones de los métodos tradicionales de enseñanza de la ortografía. La aplicación de enfoques innovadores, como la inteligencia artificial y la ludificación, permite diseñar una herramienta educativa dinámica y envolvente, que no solo motive a los niños a participar activamente en su aprendizaje, sino que también les proporcione las herramientas y recursos necesarios para mejorar sus habilidades ortográficas de manera efectiva.

Este trabajo representa un paso adelante en la búsqueda de soluciones creativas y efectivas para mejorar el proceso de aprendizaje de la ortografía española en niños en edad temprana. Al proporcionar una experiencia educativa innovadora y atractiva, se aspira a no solo fortalecer las habilidades lingüísticas de los niños, sino también a fomentar su amor por el aprendizaje y su confianza en sí mismos como estudiantes

1.1 Contexto

El dominio de la ortografía es fundamental en el proceso de adquisición del lenguaje escrito. Sin embargo, enseñar ortografía de manera tradicional puede resultar desafiante y poco motivador para los estudiantes. Con el fin de abordar este desafío se han tomado por base los dictados tradicionales y se han transformado por completo a través del uso de la tecnología y la ludificación.

Esta aplicación no solo ofrece una forma dinámica y atractiva de practicar ortografía, sino que también utiliza algoritmos avanzados para corregir los errores de manera eficiente. Además, incorpora un enfoque pedagógico basado en el algoritmo de Leitner, que optimiza el proceso de aprendizaje al identificar y priorizar las palabras que requieren mayor atención por parte del usuario.

1.2 Ludificación

Resulta especialmente complicado realizar una buena labor en el campo educativo en niños de edades tempranas. Los profesionales en el sector conocen muy bien y hacen uso en su día a día de distintas metodologías educativas. Sin embargo, una de las mayores dificultades radica en mantener un nivel de atención constante por parte de los alumnos durante el proceso de aprendizaje. Es aquí donde la ludificación se presenta como una herramienta invaluable.

Lo primero es dejar claro en qué consiste la ludificación (Borrás-Gené, 2022) o también llamada en su defecto, gamificación; término acuñado por Nick Pelling en el año 2002 (Marczewski, 2013). Es el uso de elementos propios del juego en contextos diferentes, con el fin de motivar la participación y el aprendizaje de los involucrados. Su objetivo es hacer uso de la motivación intrínseca que tienen las personas por naturaleza hacia el juego para impulsar otros ámbitos, como el que es protagonista en este proyecto: el aprendizaje.

En este momento, hay que tomar consciencia de la diferencia entre jugar y el juego; siendo este último, la formalización del primero (Borrás-Gené, 2022). De aquí nace la motivación del proyecto: mediante el uso de la tecnología, la ludificación y diversos métodos y técnicas, conseguir una experiencia de aprendizaje eficaz y atractiva de la ortografía española.

La ludificación no solo agrega elementos lúdicos a las actividades educativas, sino que también promueve un ambiente de competencia amigable, recompensas, desafíos y retroalimentación inmediata, lo que resulta especialmente eficaz en el contexto de la enseñanza. Al incorporar elementos de juego en la página web se busca captar la atención de los niños, mantener su interés y motivar su participación en el proceso de aprendizaje. De esta manera, la ludificación se convierte en una herramienta clave para superar las barreras tradicionales en la enseñanza y garantizar una experiencia educativa efectiva y enriquecedora para los estudiantes.

1.3 Objetivos

El objetivo principal del proyecto es desarrollar una herramienta educativa innovadora que mejore significativamente el aprendizaje de la ortografía española en niños en edad temprana, mediante el uso de la tecnología y la implementación de la ludificación. Para lograr este propósito, se plantean los siguientes objetivos:

1. Diseñar y desarrollar una página web interactiva que utilice la inteligencia artificial proporcionada por OpenAI (API Reference OpenAI, 2024) para la generación de dictados.

2. Implementar un algoritmo de corrección propio que identifique y priorice los errores ortográficos cometidos por los usuarios, con el fin de ofrecer una retroalimentación precisa y personalizada que fomente la mejora continua.

3. Integrar el algoritmo de Leitner como parte del proceso de enseñanza, para optimizar la retención y el aprendizaje a largo plazo de las palabras.

4. Incorporar elementos de ludificación en la página web como puntos, niveles y desafíos con el objetivo de estimular la participación de los usuarios, mantener su atención y motivar su compromiso con el aprendizaje de la ortografía española.

2 Metodología

A lo largo de todo el proyecto se ha utilizado una metodología iterativa e incremental en la que en cada entrega se añaden funcionalidades completamente nuevas. Esta comienza por un análisis de los requisitos del sistema y posteriormente, se lleva a cabo el respectivo desarrollo.

Al ser un desarrollo incremental, en cada iteración se va a obtener una "pieza" más del sistema y esta va a ser la base de la siguiente iteración. Al ser un desarrollo iterativo, con cada iteración se obtiene el feedback del cliente, lo que va a permitir un desarrollo más preciso.

Ventajas:

- El feedback constante del cliente permite una mayor precisión.
- El cliente es consciente en tiempo real del avance del proyecto.
- Se obtienen resultados prácticos antes de la finalización total del proyecto.

Desventajas:

- Participación e involucración continua del cliente.
- Se requiere la realización para cada iteración de pruebas continuas para no destrozar funcionalidades ya hechas.

2.1 Contraposición de la metodología

Inicialmente, se barajó la metodología Scrum (McCarthy, 2020) para la gestión de este proyecto. Sin embargo, tras un exhaustivo análisis, se llegó a la conclusión de que no se adecuaba a las necesidades del presente proyecto, principalmente debido a las particularidades de los roles que implica.

Scrum, al ser una metodología diseñada para proyectos gestionados por equipos multidisciplinarios, establece roles claramente definidos (Watts, 2013), como el Scrum Master, el Product Owner y el equipo de desarrollo. Estos roles tienen sus propias responsabilidades y funciones específicas en el proceso de desarrollo del proyecto.

En el contexto del trabajo actual, desarrollado por una única persona, la estructura de roles de Scrum resulta poco práctica y, en ciertos aspectos, innecesaria. La figura del Scrum Master, por ejemplo, se encarga de facilitar el proceso y eliminar los obstáculos que puedan surgir durante el desarrollo. Sin embargo, en un proyecto individual, esta función recae naturalmente en el propio desarrollador, lo que hace redundante la asignación de un rol específico para esta tarea.

De manera similar, el Product Owner en Scrum es responsable de representar los intereses del cliente y priorizar el backlog del producto. Sin embargo, en un proyecto individual el desarrollador es el único responsable de definir los requisitos y prioridades del proyecto, lo que elimina la necesidad de un rol dedicado al Product Owner.

Por otro lado, el equipo de desarrollo en Scrum está formado por profesionales con diversas habilidades y roles complementarios, que trabajan juntos para llevar a cabo las tareas asignadas en cada sprint. En un proyecto individual, el desarrollador asume todas las responsabilidades del equipo de desarrollo, lo que hace que la estructura de roles de Scrum resulte redundante y poco práctica.

Por lo tanto, se optó por seguir una metodología iterativa e incremental (Gadam, 2023) que prescinde de la estructura de roles definida por Scrum (¿Qué es la metodología Scrum? Y Gestión de proyectos Scrum, 2024) y se centra en las necesidades y capacidades del desarrollador individual. Esto permite una mayor flexibilidad y adaptabilidad en la gestión del proyecto, asegurando que cada fase se lleve a cabo de manera eficiente y sin las limitaciones impuestas por roles específicos.

3 Análisis de requisitos global

El análisis de requisitos (Análisis de requisitos de software, 2023) es un paso fundamental en el proceso de desarrollo de software, ya que establece las funcionalidades y características que el sistema debe cumplir para satisfacer las necesidades de los usuarios. En este contexto, se ha realizado un análisis de los requisitos del sistema desde el punto de vista del usuario; abordando aspectos clave como los actores involucrados y los casos de uso identificados.

3.1 Actores

Los actores en un sistema representan a las entidades que interactúan con el mismo. En este caso, se han identificado dos actores principales: el usuario no autenticado y el usuario autenticado. Cada uno de estos actores tiene un conjunto específico de acciones que puede realizar dentro del sistema, dependiendo de su estado de autenticación.

Actores:

- Usuario no autenticado: sólo puede iniciar sesión o registrarse en el sistema
- Usuario autenticado: tiene total acceso a todas las funcionalidades.

3.2 Casos de uso

Los casos de uso describen las interacciones entre los actores y el sistema, detallando las diferentes funcionalidades que el sistema ofrece. En esta sección, se han enumerado y descrito los distintos casos de uso identificados en el sistema, desde las acciones básicas como registrarse e iniciar sesión, hasta funcionalidades más avanzadas como modificar la cuenta o acceder a diferentes modos de práctica ortográfica.

- CU-01 Registrarse: el usuario puede crear una cuenta para identificarse. Para ello debe introducir su nombre, correo electrónico, nombre de usuario y contraseña.
- CU-02 Iniciar sesión: el usuario especifica su correo electrónico y contraseña. Una vez realizada la autenticación, el usuario tendrá acceso a las funcionalidades del sistema.
- CU-03: Cerrar sesión: el usuario autenticado puede cerrar sesión, pasando a ser un usuario no autenticado.
- CU-04: Cambiar usuario: el usuario autenticado puede cerrar sesión, pasando a ser un usuario no autenticado, pero abriéndose de forma automática el “CU-02” Iniciar sesión.
- CU-05: Borrar cuenta: el usuario autenticado puede borrar la cuenta si lo desea. Para ello tiene que realizar una doble confirmación.

- CU-06: Modificar cuenta: el usuario autenticado puede cambiar su nombre, correo electrónico y nombre de usuario, actualizando los datos existentes.

- CU-07: Cambiar contraseña: el usuario autenticado puede cambiar la contraseña. Para ello tiene que especificar la contraseña anterior, una nueva contraseña y repetirla a modo de confirmación.

- CU-08: Añadir amigos: el usuario autenticado puede añadir amigos escribiendo el nombre de usuario de estos.

- CU-09: Eliminar amigos: el usuario autenticado puede eliminar amigos, para ello tiene que pulsar el botón "eliminar" que hay al lado de cada nombre de usuario de cada amigo.

- CU-10: Cambiar nivel: el usuario puede elegir un nivel (entre 1 y 3). La aplicación de este cambio repercutirá en la longitud de los posteriores dictados.

- CU-11: Dictado aleatorio: el usuario podrá acceder al modo de "Dictado aleatorio". En el cual se generará un dictado aleatorio en el que se llevará un seguimiento de sus fallos.

- CU-12: Reglas ortográficas: el usuario podrá acceder a la pantalla de reglas ortográficas, en la cual se visualizará las distintas reglas ortográficas y otros elementos del castellano.

- CU-13: Dictado reglas ortográficas: el usuario podrá acceder al modo de "Dictado reglas ortográficas". En el cual podrá realizar un dictado en relación con la regla ortográfica o al otro elemento del castellano seleccionado.

- CU-14: Dictado competición: el usuario podrá acceder al modo de "Dictado competición". En el cual podrá realizar un dictado que se contabilizará para los respectivos rankings de la competición.

- CU-15: Escuchar audio: el usuario podrá escuchar el audio generado.

- CU-16: Visualizar fallos: el usuario podrá visualizar en el modo "Dictado aleatorio" las palabras programadas para su repaso, así como el nivel en que el que se encuentran y la fecha prevista para su repaso.

- CU-17: Visualizar algoritmo de Leitner: el usuario podrá visualizar en el modo "Dictado aleatorio" una explicación del algoritmo de Leitner.

- CU-18: Seleccionar elemento: El usuario podrá seleccionar la regla ortográfica u otro elemento del castellano en el modo "Reglas ortográficas".

- CU-19: Visualizar teoría: EL usuario podrá visualizar la teoría correspondiente a la regla ortográfica u otro elemento del castellano seleccionado en el modo "Reglas ortográficas".

- CU-20: Visualizar ranking global: el usuario podrá visualizar en el modo “Competición” el ranking global de la competición.

- CU-21: Visualizar ranking amigos: el usuario podrá visualizar en el modo “Competición” el ranking exclusivo de amigos de la competición.

4 Planificación

A lo largo de este capítulo se explicará cómo se ha llevado a cabo la planificación del proyecto siguiendo la metodología iterativa e incremental según sus diversas iteraciones.

4.1 Iteración 0

Se ha realizado el análisis global de los requisitos del sistema, estableciendo así los casos de uso a implementar y diseñando sus respectivos mockups.

4.2 Iteración 1

Creación de la página de inicio, del footer y del esqueleto base. Así como la definición del modelo Entidad-Relación de la base de datos, del Esquema Relacional básico y de su correcta implementación, permitiendo el correcto desarrollo de los siguientes casos de uso:

- CU-01 Registrarse
- CU-02 Iniciar sesión

4.3 Iteración 2

Creación de la página del menú, del manejo de niveles y de la implementación de los casos de usos relacionados con la manejabilidad de la cuenta y las características de los atributos del usuario:

- CU-03: Cerrar sesión
- CU-04: Cambiar usuario
- CU-05: Borrar cuenta
- CU-06: Modificar cuenta
- CU-07: Cambiar contraseña
- CU-10: Cambiar nivel

4.4 Iteración 3

Creación de la página correspondiente (y común para los tres modos diferentes) para la realización del dictado. Así como la generación de dictados y su posterior paso a audio. Realizando el siguiente caso de uso:

- CU-15: Escuchar audio

4.5 Iteración 4

Creación del algoritmo de corrección y la visualización del flujo de corrección.

4.6 Iteración 5

Implementación total del algoritmo de Leitner. Correcto manejo de las palabras de la base de datos y su visualización en la pantalla. Así como la actualización de la generación de dictados en "Dictado aleatorio", generándose estos según la palabra que sea objeto de estudio. Realizando así los siguientes casos de uso:

- CU-16: Visualizar fallos
- CU-17: Visualizar algoritmo de Leitner
- CU-11: Dictado aleatorio

4.7 Iteración 6

Creación de la página de las reglas ortográficas. Definición de las matrices, elaboración de la teoría correspondiente y definición de ejemplos de otros elementos del castellano. Permitiendo generar dictados correspondientes a la regla ortográfica u al otro elemento del castellano seleccionado. Realizando los siguientes casos de uso:

- CU-12: Reglas ortográficas
- CU-18: Seleccionar elemento
- CU-19: Visualizar teoría
- CU-13: Dictado reglas ortográficas

4.8 Iteración 7

Correcta implementación y manejo de los amigos del usuario. Creación del sistema de puntuación y de ambos rankings. Realizando los siguientes casos de uso:

- CU-08: Añadir amigos
- CU-09: Eliminar amigos
- CU-14: Dictado competición
- CU-20: Visualizar ranking global
- CU-21: Visualizar ranking amigos

4.9 Iteración 8

Esta iteración está dedicada exclusivamente al desarrollo de la memoria final a partir del resultado obtenido y la corrección de pequeños errores.

4.10 Resultado final

A lo largo de cada iteración se ha estimado el tiempo en horas necesario para completarlas. Así como la fecha de inicio real, la fecha de fin y las horas reales necesarias. Debido a que la dedicación al proyecto ha sido casi exclusiva, el tiempo dedicado diariamente fue regular a lo largo de todo el desarrollo, siendo este una aproximación de 4 horas diarias.

Iteración	Fecha inicio	Fecha fin	Estimación	Tiempo real
Iteración 0	01/02/2024	6/02/2024	20	22
Iteración 1	07/02/2024	14/02/2024	30	33
Iteración 2	15/02/2024	23/02/2024	35	39
Iteración 3	24/02/2024	4/03/2024	40	44
Iteración 4	5/03/2024	18/03/2024	50	55
Iteración 5	19/03/2024	29/03/2024	30	33
Iteración 6	30/03/2024	10/04/2024	40	44
Iteración 7	11/04/2024	21/04/2024	35	39
Iteración 8	22/04/2024	10/05/2024	80	81s

TOTAL	01/02/2024	10/05/2025	360	390

Tabla 1: Planificación del proyecto

Se puede observar que las iteraciones 4 y 8 son las que más tiempo han consumido. Era predecible en última iteración dado a que la redacción de la memoria es un trabajo considerable. Sin embargo, la que más sorprende es la iteración número 4; que es la correspondiente a la creación del algoritmo propio de corrección. Lo que requirió pruebas constantes y una gran cantidad de tiempo para lograr un resultado óptimo.

5 Costes

Para realizar el cálculo del coste del proyecto se tienen en cuenta tanto el coste de las horas trabajadas como el de los demás recursos necesarios. Partiendo de un sueldo de 30 euros/hora:

Coste de las horas trabajas = 390 horas * 30 euros/hora = 11.700 euros

Aparte de las horas de trabajo, hay que calcular los costes asociados a servicios y recursos. Para esto se tiene en cuenta el tiempo que habría llevado dedicándose de manera completa al proyecto:

Tiempo a jornada completa = 390 horas / 160 horas/mes = 2.43 meses.

Los recursos utilizados a lo largo del proyecto fueron:

- Equipamiento informático: ordenador portátil con un coste de 800 euros. Considerando que el tiempo de amortización es de 3 años, su coste es de 800 * (2.43 / 36) = 54 euros.
- Electricidad: considerando que cada mes se gastaron 55 euros en electricidad, el coste imputable al proyecto sería de 55 * 2.43 = 134 euros.
- Internet: considerando un gasto de 30 euros/mes, el coste imputable sería de 30*2.43 = 80 euros

Concepto	Coste (euros)
Horas trabajadas	11.700
Equipo informático	54
Electricidad	134
Internet	80
Coste total	11968

Tabla 2: Costes

6 Diseño y fundamentos tecnológicos

6.1 Modelo de datos

El modelo entidad relación (Figura 6.1) consta de una única entidad fuerte: “usuario” y de 2 entidades débiles: “amigos” y “palabra”. La totalidad de los atributos de las entidades son monovaluados simples, y las claves primarias están organizadas de la siguiente manera:

- Usuario:
  - Clave primaria: id_usuario
- Amigos
  - Clave primaria compuesta: id_usuario y id_amigo
- Palabra
  - Clave primaria compuerta: id_usuario y id_palabra

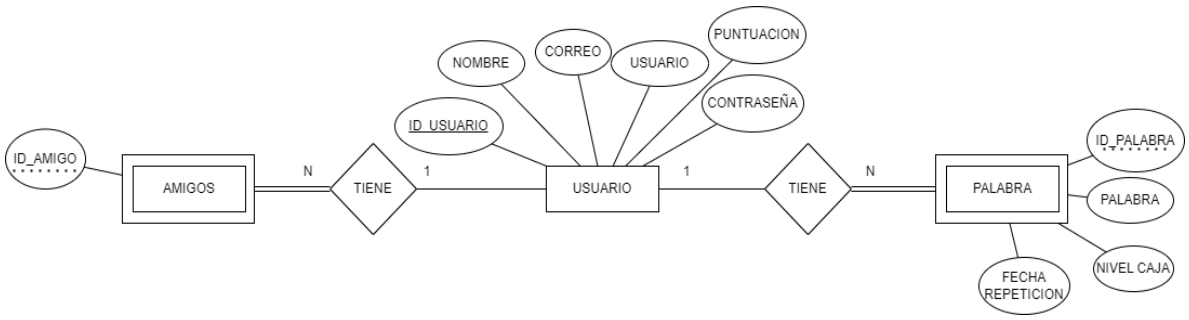


Figura 6.1: Modelo Entidad Relación

Se dispone de la necesidad de almacenar (Figura 6.2) para cada usuario su nombre, su nombre de usuario, su correo electrónico y su contraseña; atributos necesarios para el manejo de los usuarios. También se necesita el atributo “puntuación”, dado que cada usuario va a tener una puntuación asociada que se mostrará posteriormente en los rankings del modo “Dictado competición”.

Un usuario necesita también poder almacenar las palabras en las que ha cometido algún tipo de fallo en el modo “Dictado aleatorio” para que estas sean así objeto de estudio en posteriores dictados. Un usuario puede tener múltiples palabras asociadas, pero pueden existir usuarios que no tengan ninguna palabra asociada (participación parcial). Sin embargo, toda palabra de la base de datos va a estar asociada a un único usuario, y para que esta exista, es necesario que haga referencia a un usuario (participación total). De esta entidad se guardará como atributo la correspondiente palabra. Del mismo modo (pero careciendo de otros atributos que no sean el propio id del usuario ni el id de amigos) se implementará la entidad de amigos.

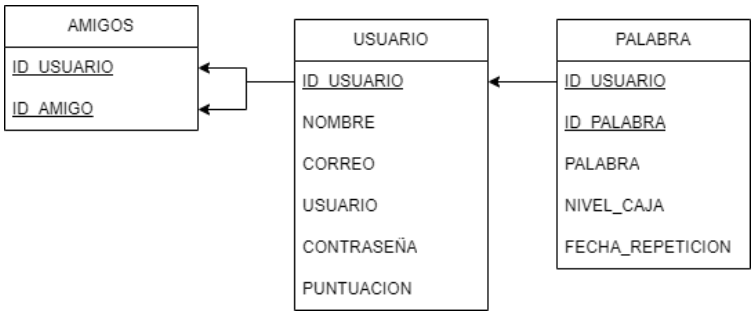


Figura 6.2: Esquema Relacional

La tabla “usuario” se relaciona (Figura 6.1) con el resto de las tablas mediante relaciones binarias “1:N”, en las que se tiene una clave foránea en la tabla en “la parte N” que referencia a la de “la parte 1”. Además, esta tiene una participación parcial en la “parte 1” y total en la “parte N” con ambas tablas. De tal manera que un usuario puede tener múltiples palabras y múltiples amigos.

6.1 Arquitectura del sistema

El diagrama (Figura 6.3: MVC) muestra la arquitectura básica que separa claramente las responsabilidades del front-end y back-end de la aplicación, con una capa adicional para el acceso a datos. Este esquema es esencialmente una representación del patrón de diseño Model-View-Controller (MVC), aunque en este caso, el énfasis está en la separación entre la interfaz de usuario (front-end), la lógica de negocio (back-end), y la gestión de datos (acceso a datos).

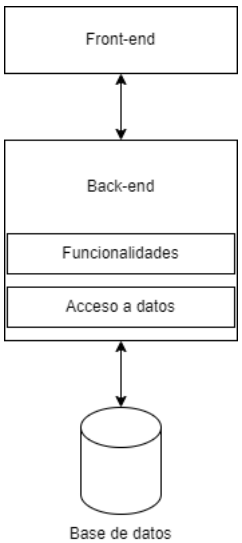


Figura 6.3: MVC

El front-end es la capa de presentación de la página web. Es la parte que interactúa directamente con el usuario. El back-end es la lógica de negocio, es dónde se procesan las solicitudes del front-end y se gestionan las respuestas. La base de datos es el almacenamiento persistente de la aplicación, donde se guardan todos los datos.

La interacción entre los diferentes componentes se realiza de la siguiente manera: el usuario interactúa con la interfaz de usuario en el front-end, estas interacciones generan eventos que son enviados al back-end. El back-end las recibe y procesa las solicitudes aplicando la lógica de negocio en las clases de funcionalidades. Si es necesario acceder o modificar datos, el back-end delega la tarea a la capa de acceso a datos. La capa de acceso a datos ejecuta las consultas necesarias en la base de datos y los resultados de estas, son devueltos a la capa de funcionalidades. Si es necesario obtener o modificar datos en la base de datos, el back-end envía una respuesta al front-end. Finalmente, el front-end recibe la respuesta y actualiza la interfaz de usuario en consecuencia, proporcionando al usuario la información o resultado de la acción realizada.

6.2 Tecnologías utilizadas

6.3.1 Visual Studio Code

Entorno de desarrollo integrado (Página web de Visual Studio Code, 2024) creado por Microsoft. Tiene soportes para múltiples lenguajes de programación: como JavaScript, Python, HTML y CSS. Además de permitir la agregación de extensiones para admitir más lenguajes y funcionalidades. Cuenta también con herramientas de edición y depuración de código, resaltando sintaxis, finalización automática de código, refactorización (Johnson, 2019)…

6.3.2 HTML

Acrónimo de Hyper Text Markup Language, (Página web de HTML, 2024) o también llamado Lenguaje de Marcado de Hipertexto en español. Es el lenguaje utilizado para la creación y diseño de páginas web. En él se define la estructura y el contenido de una página web mediante el uso de atributos y etiquetas. Estas páginas son interpretadas por los navegadores web, procesando el código HTML y representándolo visualmente para los usuarios (Hart-Davis, 2023).

6.3.3 CSS

Siglas de Cascading Style Sheets (Página web de CSS, 2024), o también llamado Hojas de Estilo en Cascada en español. Es un lenguaje utilizado para definición del estilo y presentación de

los elementos HTML. A diferencia de este último, CSS se encarga de definir cómo se presentan los elementos en términos de diseño y estilo (Hart-Davis, 2023).

6.3.4 JavaScript

Lenguaje de programación (Página web de JavaScript, 2024) utilizado para la agregación de interactividad y funcionalidad en páginas HTML. Este lenguaje (Haverbeke, 2018) permite a los desarrolladores crear experiencias interactivas y dinámicas que responden a las acciones del usuario, modificando así el contenido de la página en tiempo real (Azaustre, 2021).

6.3.5 Python

Lenguaje de programación interpretado (Página web de Python, 2024), multiparadigma y de alto nivel. Es ampliamente utilizado en diversas áreas del desarrollo software debido a su facilidad de comprensión del código. Destaca también por su versatilidad y su amplio abanico de frameworks y bibliotecas. Siendo utilizado en el contexto del desarrollo web tanto en el lado del servidor como en el del cliente (JIMÉNEZ, 2021).

6.3.6 Xampp

Paquete de software libre y de código abierto desarrollado por Apache Friends (Página web de Xampp, 2024). Está diseñado para facilitar la instalación y configuración de un entorno de desarrollo web local. Es utilizado para crear y probar sitios web de forma local antes de desplegarlos en un servidor en línea. También es compatible con múltiples sistemas operativos y consta de una interfaz intuitiva y un proceso de instalación sencillo.

6.3.7 Git

Sistema de control de versiones (Página web de GitHub, 2024) distribuido de código abierto diseñado para rastrear cambios en archivos y coordinar el trabajo entre múltiples personas en proyectos de desarrollo de software. Creado por Linus Torvalds, se ha convertido en la herramienta de control de versiones más utilizada. Resulta fundamental para gestionar el código fuente de proyectos de todo tipo de tamaños, permitiendo la colaboración conjunta y su respectiva gestión y seguimiento de los cambios realizados (Tsitoara, Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer, 2019).

7 Desarrollo

7.1 Iteración 1

7.1.1 Análisis

En esta primera iteración, el objetivo principal es establecer la estructura básica del proyecto y definir los elementos fundamentales necesarios para el correcto funcionamiento de las funcionalidades futuras. Se procederá a crear la página de inicio del sitio web, así como el footer que estará presente en todas las páginas. Además, se establecerá el esqueleto base del proyecto, definiendo la estructura de archivos y directorios.

> Debido a la edad temprana del público objetivo, es de especial importancia que la página de inicio sea muy simple. Añadiendo únicamente un eslogan y un único botón (además del footer). De esta manera se obtiene una página sencilla con una curva de aprendizaje reducida.

Para respaldar la funcionalidad del sistema, se realizará un análisis preliminar para definir el modelo Entidad-Relación (ER) de la base de datos. Esto implicará identificar las entidades principales y sus relaciones, así como los atributos clave de cada entidad. A partir de este modelo ER, se procederá a diseñar el Esquema Relacional básico, que definirá la estructura de las tablas en la base de datos. Debido al conocimiento de los requisitos con anterioridad, se procederá también a su implementación.

Al final de esta iteración, se tienen que haber realizado con éxito los siguientes casos de uso:

- CU-01 Registrarse
- CU-02 Iniciar sesión

7.1.2 Diseño e implementación

Página de inicio

La creación de la página (Figura 7.1) sirve como punto de entrada para los usuarios.

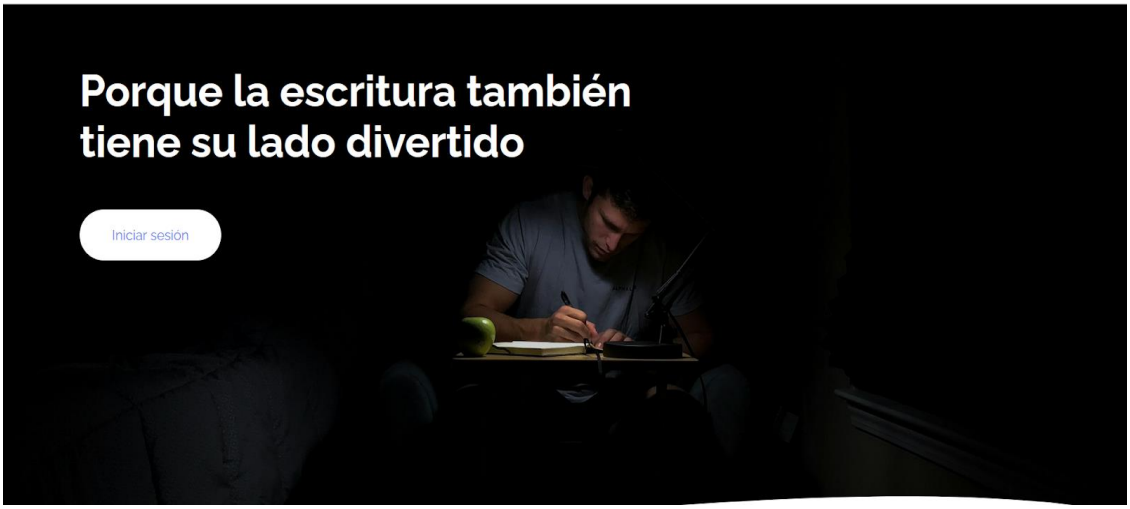


Figura 7.1: Página de inicio

Esta dispone del eslogan “Porque la escritura también tiene su lado divertido” y un único botón, que lleva a los respectivos casos de uso creados: CU-01 Registrarse y CU-02 Iniciar sesión (compartiendo el mismo modal).

Footer

La creación del footer (Figura 7.2) estará presente en todas las páginas del sitio web. Este contendrá enlaces útiles como el origen del trabajo, información de contacto, una guía para padres y un archivo con las preguntas frecuentes. Además, contendrá enlaces a las respectivas redes sociales. Este ha sido diseñado de manera que sea coherente con el estilo y la entidad visual de la totalidad de las páginas.

Figura 7.2: Footer

Base de datos

Para su aplicación se ha utilizado Xampp (Figura 7.3) Es un servidor web local multiplataforma que utiliza Python como lenguaje de programación. Básicamente es una distribución de Apache gratuita que permite la gestión de bases de datos MySQL.

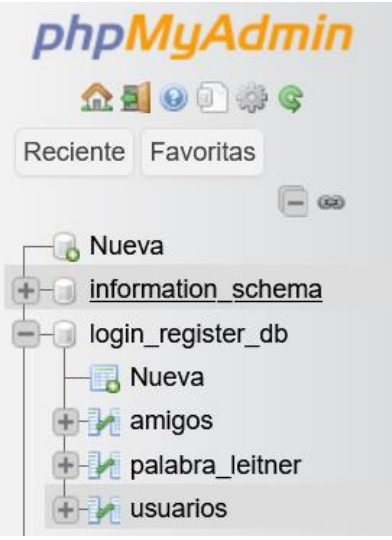


Figura 7.3: phpMyAdmin

CU-01 Registrarse

El primer paso fue la creación de su estilo y diseño mediante HTML Y CSS. Teniendo como resultado el formulario (Figura 7.4) a nivel estético, pero no funcional.

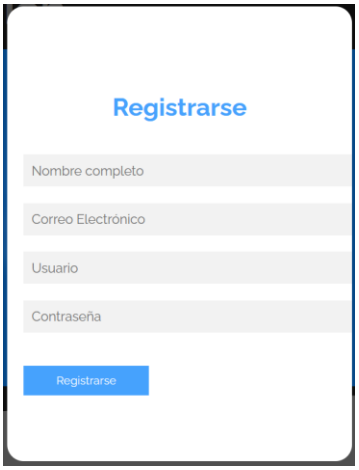


Figura 7.4: Formulario registro

Con el diseño terminado (Figura 7.4) y la base de datos correctamente implementada en Xampp, se procedió a la creación del archivo .php que tiene objetivo registrar un usuario. Para ello se ha utilizado una query de inserción que utiliza los valores introducidos en el formulario para actualizar los atributos (junto con una serie de querys de control para que estos tengan un formato

específico).

Pseudocódigo: registro usuario

```
INCLUIR 'conexion_be.php' // para el acceso a la base de datos

nombre = OBTENER_VALOR_DE_POST('nombre') // variable que apunta
al nombre completo del formulario registro

correo = OBTENER_VALOR_DE_POST('correo') // variable que apunta al
correo del formulario registro

usuario = OBTENER_VALOR_DE_POST('usuario') // variable que apunta
al usuario del formulario registro

contrasena = OBTENER_VALOR_DE_POST('contrasena') // variable que
apunta a la contraseña del formulario registro

contrasena_encriptada = ENCRYPTAR_CON_SHA512(contrasena) // se
encripta la contraseña

query = "INSERT INTO usuarios(nombre, correo, usuario, contrasena)
// query para insertar datos en la tabla "usuarios"

VALUES(nombre, correo, usuario, contrasena_encriptada )"

/*QUERYS DE CONTROL */
```

En este archivo se crea la conexión (y se cierra al final) con la base de datos. En él se realizan distintas comprobaciones para asegurarse de que el nombre del usuario y el correo son únicos y no existen en la base de datos actual. Además de encriptar la contraseña y de establecer una longitud mínima y formato común para el nombre, el nombre de usuario y el correo electrónico.

CU-02 Iniciar sesión

El primer paso fue la creación de su estilo y diseño mediante HTML Y CSS. Teniendo como resultado el formulario a nivel estético (Figura 7.5), pero no funcional.

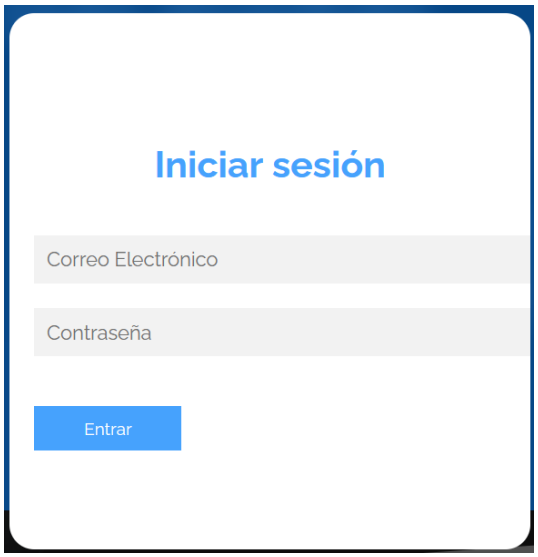


Figura 7.5: Formulario inicio sesión

Posteriormente se procedió a la creación del archivo .php que tiene como objetivo autenticar al usuario

Pseudocódigo: login usuario

```
INICIAR SESIÓN // para que se inicialicen las sesiones al principio (lo de $_SESSION)

INCLUIR 'conexion_be.php' // para el acceso a la base de datos

correo = OBTENER_VALOR_DE_POST('correo') // variable que apunta al correo del formulario login

contrasena = OBTENER_VALOR_DE_POST('contrasena') // variable que apunta a la contraseña del formulario login

contrasena_desencriptada = ENCRYPTAR_CON_SHA512(contrasena) // se desencripta la contraseña

validar_login = EJECUTAR_CONSULTA_SQL("SELECT * FROM usuarios WHERE correo=correo AND contrasena=contrasena_desencriptada") // si el correo y la contraseña coinciden...

SI FILAS_ENCONTRADAS(validar_login) > 0 ENTONCES // si el correo y la contraseña coinciden...

   usuario = OBTENER_DATOS_DE_USUARIO(validar_login) // obtener datos del usuario

   GUARDAR_EN_SESION('id_usuario', usuario['id']) // se almacena el ID del usuario en la sesión (cookies del navegador)

   GUARDAR_EN_SESION('usuario', correo) // se almacena el correo en la sesión (cookies del navegador)

   REDIRIGIR_A("/../AprendeDictando/menu.php") // redirige a otra página

   CERRAR_CONEXIÓN(conexion) // cerramos la conexión

   SALIR() // sale del script y no sigue ejecutándose.

SINO

   IMPRIMIR('El usuario o la contraseña son incorrectos. Por favor, inténtelo de nuevo.')

   REDIRIGIR_A("../index.php?formulario_erroneo=true")

   CERRAR_CONEXIÓN(conexion) //cerramos la conexión

   SALIR() //sale del script y no sigue ejecutándose

FIN SI
```

En este archivo se crea la conexión (y se cierra al final) con la base de datos. En él se realizan las distintas comprobaciones para dilucidar si los datos introducidos son válidos. De ser así, el usuario se autentifica y se completa el CU-02 Iniciar sesión.

7.2 Iteración 2

7.2.1 Análisis

En esta iteración el enfoque se centra en crear el menú de la página web. Este tiene que ser sencillo e incluir lo estrictamente necesario para permitir la correcta ejecución de los casos de uso. Se tienen que incluir visualmente (aunque no estén implementados) la totalidad de los elementos que va a tener la página web en el momento de la entrega. Refiriéndose así, a los distintos modos de los dictados y a los elementos necesarios para abordar los casos de uso relacionados con la gestión de la cuenta del usuario, los amigos y el nivel de los dictados.

El objetivo principal es proporcionar al usuario un acceso intuitivo a las funcionalidades principales del sistema, así como permitirle gestionar su cuenta de manera eficiente y segura. Para lograr esto, el desarrollo se basa en el resultado obtenido en la iteración anterior, asegurándose de implementar de manera correcta las siguientes funcionalidades:

- CU-03: Cerrar sesión
- CU-04: Cambiar usuario
- CU-05: Borrar cuenta
- CU-06: Modificar cuenta
- CU-07: Cambiar contraseña
- CU-10: Cambiar nivel

7.2.2 Diseño e implementación

Menú

Se procede a la creación del diseño y estilo del menú (Figura 7.6) que contendrá los botones y enlaces para las respectivas secciones y funcionalidades del sistema. Evitando caer en complicaciones y en diseños meticulosos, consiguiendo un resultado simple centrado en la navegabilidad y usabilidad de la aplicación.

Figura 7.6: Menú

Como se puede observar (Figura 7.6), los tres modos de dictados están visibles en la parte izquierda del menú. Y las distintas funcionalidades relativas a la cuenta, a los amigos y al nivel, se encuentran arriba a la derecha. No hay nada más a parte del footer.

Si se hace clic en el nombre de usuario, se abrirá un modal (Figura 7.7) que dará paso a la totalidad de funcionalidades relativas a los casos relativos a la manejabilidad de la cuenta.

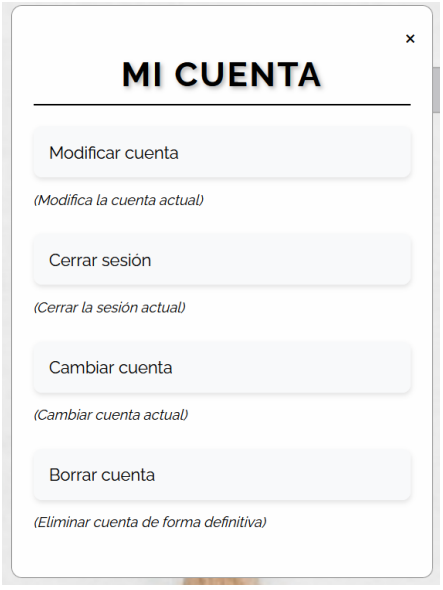


Figura 7.7: Mi cuenta

CU-03: Cerrar sesión

El usuario autenticado puede cerrar sesión (Figura 7.8), lo que le llevará a ser un usuario no autenticado y, por tanto, será redirigido a la página de inicio.

Figura 7.8: Cerrar sesión

Su aplicación es muy sencilla:

Pseudocódigo: cerrar sesión

```
INICIAR SESIÓN // inicializamos la sesión

DESTRUIR SESIÓN // destruye la sesión

REDIRIGIR_A("../index.php") // redirige a la página principal
```

CU-04: Cambiar usuario

El usuario autenticado puede cambiar de usuario (Figura 7.9), lo que le llevará a ser un usuario no autenticado. Será redirigido a la página de inicio, en la que se abrirá de manera automática el modal de inicio de sesión.

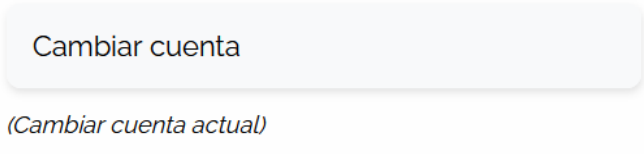


Figura 7.9: Cambiar cuenta

CU-05: Borrar cuenta

El usuario autenticado puede borrar su cuenta (Figura 7.10) mediante una doble confirmación (Figura 7.11). Garantizando así la seguridad y la prevención de eliminaciones accidentales.

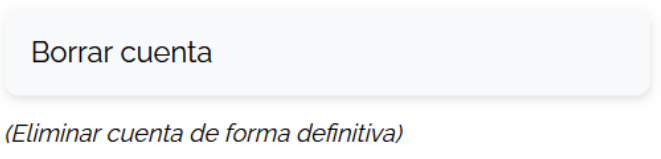


Figura 7.10: Borrar cuenta

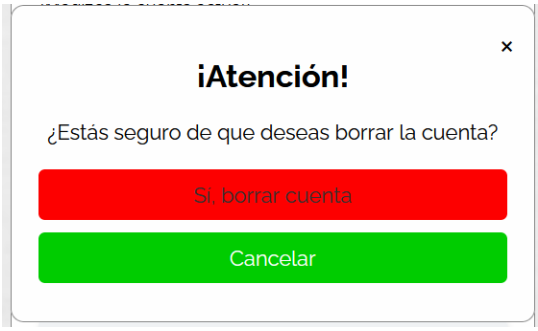


Figura 7.11: Doble confirmación

CU-06: Modificar cuenta

El usuario autenticado puede modificar su nombre, correo electrónico y nombre de usuario, actualizando así los datos existentes (Figura 7.12).

Figura 7.12: Modificar cuenta

Se realiza de la siguiente forma:

Pseudocódigo: actualizar usuario

```
INICIAR SESIÓN

INCLUIR 'conexion_be.php'

OBTENER_VARIABLES_DE_FORMULARIO (nombre_nuevo,
correo_nuevo, usuario_nuevo)

id_usuario = OBTENER_VARIABLE_DE_SESION('id_usuario')

SI (TODOS_LOS_CAMPOS_ESTÁN_LLENOS_Y_VÁLIDOS(nombre_nuevo,
correo_nuevo, usuario_nuevo)) ENTONCES

  SI (ACTUALIZAR_DATOS_DE_USUARIO(nombre_nuevo, correo_nuevo,
usuario_nuevo, id_usuario)) ENTONCES

    MOSTRAR_ALERTA("Datos actualizados exitosamente")

  SINO

    MOSTRAR_ALERTA("Error al actualizar los datos. Inténtalo de
nuevo.")

  FIN SI

SINO

  MOSTRAR_ALERTA("Por favor, completa todos los campos.")

FIN SI

CERRAR_CONEXIÓN()
```

En este archivo también realizan las mismas comprobaciones que en el momento de registrar un usuario. De esta manera se evita que se incumpla el formato establecido en un primer momento. Si todo sale satisfactoriamente, se lanza la consulta y se actualizan los datos. En caso contrario, se lanza una advertencia indicando el motivo del fallo.

CU-07: Cambiar contraseña

El usuario autenticado puede cambiar su contraseña proporcionando los campos necesarios (Figura 7.13): la contraseña anterior, la nueva contraseña y su confirmación.

Figura 7.13: Modificar contraseña

Su implementación se lleva a cabo de la siguiente manera:

Pseudocódigo: actualizar contraseña

```
INICIAR SESIÓN

INCLUIR 'conexion_be.php'

OBTENER_VARIABLES_DE_FORMULARIO(contrasena_actual,
nueva_contrasena, confirmar_contrasena)

id_usuario = OBTENER_VARIABLE_DE_SESION('id_usuario')

SI
(TODOS_LOS_CAMPOS_ESTÁN_LLENOS_Y_VÁLIDOS(contrasena_actual,
nueva_contrasena, confirmar_contrasena)) ENTONCES

   contrasena_encriptada_actual =
OBTENER_CONTRASEÑA_DE_BASE_DE_DATOS(id_usuario)

   SI (CONTRASEÑA_ACTUAL_ES_CORRECTA(contrasena_actual,
contrasena_encriptada_actual)) ENTONCES

      SI
(NUEVA_CONTRASEÑA_Y_CONFIRMAR_CONTRASEÑA_COINCIDEN(nue
va_contrasena, confirmar_contrasena)) ENTONCES

         nueva_contrasena_encriptada =
ENCRIPTAR_CON_SHA512(nueva_contrasena)

         SI
(ACTUALIZAR_CONTRASEÑA_EN_BASE_DE_DATOS(nueva_contrasena_e
ncriptada, id_usuario)) ENTONCES

            MOSTRAR_ALERTA("Contraseña actualizada exitosamente")

         SINO

            MOSTRAR_ALERTA("Error al actualizar la contraseña. Inténtalo
de nuevo.")

         FIN SI

      SINO
```

En el archivo se recogen los datos del formulario, se realizan las respectivas comprobaciones y si todo sale bien, se encripta la nueva contraseña y se actualiza.

CU-10: Cambiar nivel

El usuario puede elegir un nivel entre 1 y 3 (Figura 7.14). Esto afectará a la longitud de los dictados futuros (con la excepción del modo competición).

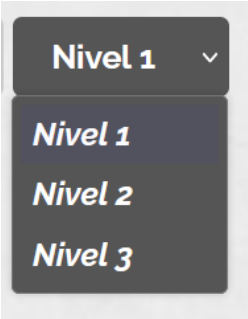


Figura 7.14: Cambiar nivel

7.3 Iteración 3

7.3.1 Análisis

El objetivo es el diseño final (y común) de la página para los distintos modos de dictados (aunque posteriormente, se realicen modificaciones en cada uno de ellos). Tiene que ser una página intuitiva y fácil de usar. Entre sus elementos, tienen que constar los siguientes:

- Nivel en el que se encuentra
- Audio por reproducir
- Espacio apropiado para que el usuario escriba el dictado
- Botón para volver al menú

Estos cuatro elementos tienen que ser completamente funcionales, pero para ello hay que realizar el diseño. Tiene que estar estéticamente adecuado para la edad del público (colores claros, elementos fáciles de diferenciar visualmente…). Una vez terminado el diseño, se tiene que proceder a la generación de dictados (a través de inteligencia artificial) y su posterior paso a audio. Terminando por implementar la capacidad que el usuario pueda escuchar el audio generado:

- CU-15: Escuchar audio

7.3.2 Diseño e implementación

Página principal de dictados

El resultado final del diseño de la interfaz es el siguiente:

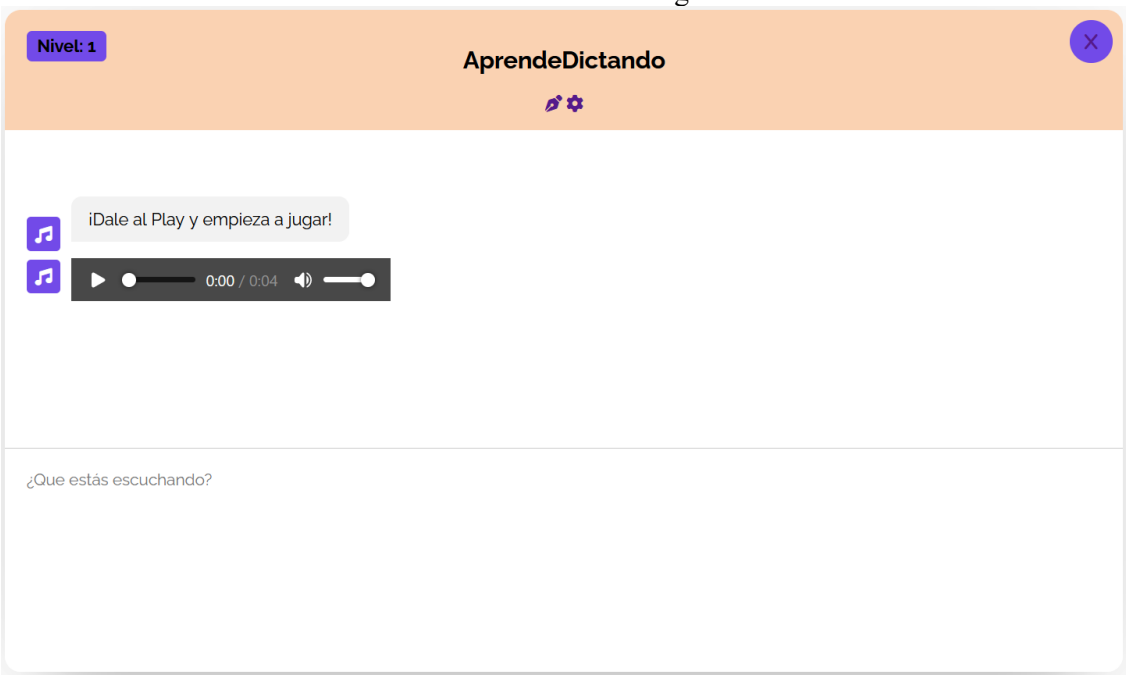


Figura 7.15: Página dictado

Es una interfaz limpia que contiene los elementos requeridos fácilmente distinguibles: nivel del usuario, espacio para escribir el dictado, botón para volver al menú y control para reproducir el audio.

Generación de dictados

La generación de los dictados se lleva a cabo a través de la inteligencia artificial proporcionada por OpenAI. Esta simplemente consiste en una petición a su API pidiendo que genere un dictado aleatorio para niños cuya edad esté en torno a los 6 años:

Pseudocódigo: Generar dictado aleatorio

```
función generarDictadoAleatorio() {

   const API_URL = "https://api.openai.com/v1/chat/completions";

   const requestOptions = {

      método: "POST",

      cabeceras: {"Content-Type": "application/json", "Authorization":
`Bearer ${API_KEY}`},

      cuerpo: JSON.stringify({modelo: "gpt-4", mensajes: [{rol: "usuario",
contenido: contenido}], frequency_penalty: 2, temperatura: 1})

   }

   retornar fetch(API_URL, requestOptions) .entonces(res => res.json())
.entonces(data => { respuestaDictado = data.choices[0].message.content;
respuestaDictado = respuestaDictado.slice(1, -1);

         si (!respuestaDictado) retornar;

    lista__respuesta__dictado.appendChild(createChatLi(respuestaDictado,
"respuesta__dictado"));

         dejar respuestaDictadoConSignos =
agregarSignosDePuntuacion(respuestaDictado);

         retornar respuestaDictadoConSignos;

      })

      .capturar((error) => { console.error("Hubo un error al generar el
dictado aleatorio:", error);lanzar error; });

}
```

Paso a audio

Una vez obtenido el dictado en formato de texto, se realiza su paso a audio a través de una petición a OpenAI:

Pseudocódigo: Pasar dictado a audio

```
función dictadoAAudio() {

   const API_URL = "https://api.openai.com/v1/audio/speech";

   const textoGenerado = esperar generarDictadoAleatorio();

   const datos = {modelo: 'tts-1-hd', entrada: textoGenerado, voz: 'onyx',
idioma: español, velocidad: 0.65 };

   const opciones = {método: "POST",

     cabeceras: { "Authorization": `Bearer ${API_KEY}`, "Content-Type":
"application/json; charset=utf-8" },

    cuerpo: JSON.stringify(datos)

   };

   intentar {

      const respuesta = esperar fetch(API_URL, opciones);

      const contenidoArchivoAudio = esperar respuesta.blob();

      const urlAudio = URL.createObjectURL(contenidoArchivoAudio);

      const fuenteAudio = document.getElementById('audioSource');

      fuenteAudio.src = urlAudio;

     const elementoAudio = document.getElementById('audioAleatorio');

      elementoAudio.load();

   } atrapar (error) { console.error("Hubo un error:", error); }

}
```

El modelo elegido es “tts-1-hd” porque, aunque tarda más que su rival “tts-1”, es de mejor calidad. La voz a utilizar es “onyx”, porque a base de prueba y error con el resto de las voces, se comprobó que daba mejores resultados narrando un texto en español. La velocidad se ha establecido en 0.65 para que narre de manera lenta y se entienda todo lo que se diga.

El paso a audio es el principal problema de este proyecto. La manera actual sólo da buenos resultados cada cierto tiempo. Es común que el narrador cuando está hablando en castellano se salte letras, aumente su velocidad de manera inesperada o pronuncie mal palabras al azar; al igual que lo es que cambie de idioma y haga una especie de mezcla de narración en inglés con el texto en castellano (aunque el idioma español esté especificado en el cuerpo de la petición).

La manera de solventar este problema y conseguir mejores resultados sería realizando los dictados manualmente. Disponiendo así de una base de datos con una cantidad de dictados suficientes y variables, y que, para cada uno de estos, la narración esté realizada por profesionales. Pero esto requeriría un tiempo de trabajo e inversión económica mucho mayor que la actual. Y aunque los resultados no sean los esperados, son suficientes para tomar como ejemplo de toda la metodología que tiene detrás.

7.4 Iteración 4

7.4.1 Análisis

Se llevará a cabo el proceso de creación del algoritmo de corrección ortográfica
para evaluar los dictados realizados por el usuario. Además de la creación del flujo de corrección para que el usuario pueda ver y comprender sus errores.

El objetivo principal del algoritmo es que sea capaz de realizar una correcta corrección. Tiene que estar pensado desde el punto de vista del usuario y que controle casos de uso indebido (que el usuario haga uso de un mal lenguaje, que envíe otro texto que no sea el dictado…). Pero haciendo siempre énfasis en el núcleo: la corrección. Con el flujo de corrección se mejora la experiencia del usuario al proporcionar retroalimentación inmediata sobre los errores cometidos en los dictados.

7.4.2 Diseño e implementación

Algoritmo de corrección

En un primer momento se pensó en la utilización de algún algoritmo corrector individual de palabras. Pero su aplicación carecía de sentido al faltar el componente del contexto. Esto hacía que aunque el usuario escribiera una frase que no tuviera ninguna relación con el dictado en sí, se daría por válida si esta estuviera bien escrita; aplicable incluso al caso, de que no fuese ni siquiera una frase, sino un conjunto de palabras al azar correctamente escritas.

Ante la inexistencia de algoritmos que proporcionasen la funcionalidad deseada, la solución se basó en la creación de un algoritmo propio, que tuviera en cuenta su contexto y que permitiera adaptabilidad a requisitos futuros (Figura 7.16).

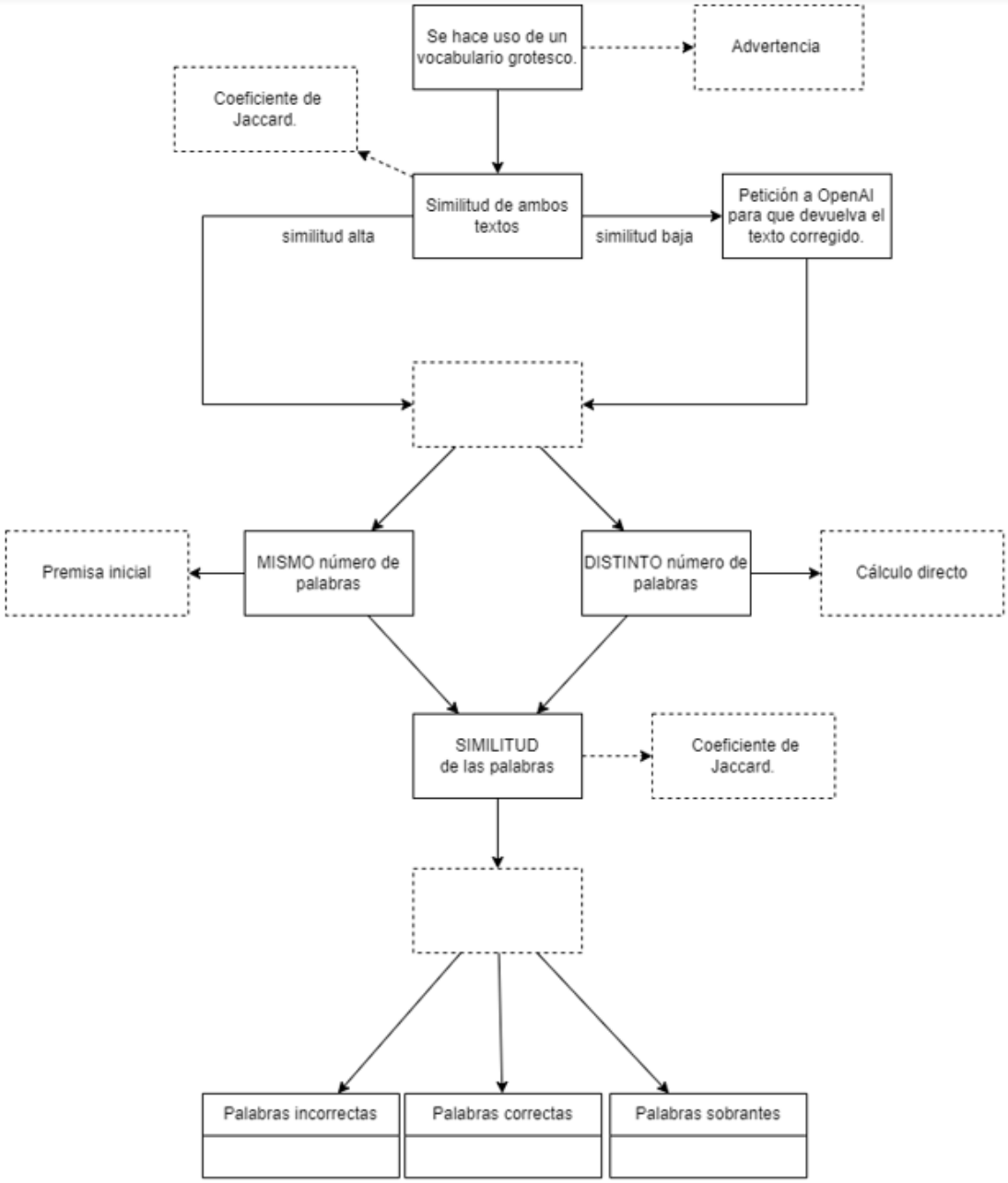


Figura 7.16: Algoritmo de corrección

Lenguaje grotesco

El objetivo principal se basa en que el usuario intente realizar el dictado, aunque este esté lleno de errores (de hecho, a mayor cantidad, mayor será el aprendizaje). Pero nada asegura que el usuario haga un buen uso de la aplicación. Y debido a la edad del público, la limitación del uso del lenguaje grotesco debe ser clara.

El primer paso es analizar el mensaje que ha enviado el usuario. Si se detecta el uso de lenguaje grotesco, se para el algoritmo y salta una advertencia explicando el por qué está mal. Este análisis se podría haber realizado contrastando el mensaje con una base de datos que contuviera este tipo de lenguaje, pero a causa de su rigidez y limitaciones, se ha descartado. La solución final es el uso de la inteligencia artificial (y como al igual que en el resto del proyecto, viene de la mano de la proporcionada por OpenAI):

Pseudocódigo: Lenguaje grotesco

```
función filtradoPalabras(resultado) {

   palabras_correctas = resultado.palabrasCorrectas

   palabras_incorrectas = resultado.palabrasIncorrectas

   para cada palabra_correcta en palabras_correctas {

      fallo_por_mayuscula = falloPorMayuscula(palabra_correcta,
palabra_incorrecta) // ¿la 1º letra de cada palabra se escribe diferente
(una mayúscula y otra minúscula)?

      intentar {

         mayuscula_natural = await mayusculaNatural(palabra_correcta) //
¿la palabra correcta se escribe con mayúscula inicial de forma natural?

         si (fallo_por_mayuscula == 1) { // si entre los fallos está que no
escribió con mayúscula una palabra que debería

            si (mayuscula_natural == 0) { // y que esa palabra NO se escribe
de manera natural en mayúscula…

               diferencias = numeroDiferencias(palabra_correcta,
palabra_incorrecta) // número de caracteres distintos de ambas palabras

               si (diferencias <= 1) { // si no tiene más fallos a parte de la
mayúscula, se eliminan

                  // se eliminan ambas palabras (la correcta y la incorrecta)

                  palabras_correctas.splice(indice_de(palabra_correcta), 1)

                  palabras_incorrectas.splice(indice_de(palabra_incorrecta), 1)

                  disminuir_indice() // se disminuye el índice para compensar
el splice

               }

            }

         }

      } capturar(error) {imprimir "Error: " + error } }

}
```

Lo más destacable en la petición es el valor bajo de la temperatura, buscando así un comportamiento más determinista.

Similitud en los textos

Una vez descartada la posibilidad del uso de este lenguaje, salen dos nuevas posibilidades. Que el usuario haya intentado hacer el dictado, o que haya escrito algo totalmente independiente. En ambos casos la corrección se tiene que llevar a cabo; a pesar de que el usuario no se ha ceñido al objetivo principal, ha realizado un proceso de escritura que puede ser igual de beneficioso.

Para saber si el usuario ha intentado realizar el dictado, se va a hacer uso del coeficiente de Jaccard. Este coeficiente permite el cálculo de la similitud entre dos conjuntos, definiéndose como el tamaño de la intersección dividido por el tamaño de la unión de los conjuntos. En este caso, tenemos dos textos: el dictado original y el mensaje del usuario:

Pseudocódigo: Coeficiente de Jaccard en textos

```
función jaccard_textos(texto1, texto2) {

  // se calcula los textos a conjuntos de palabras únicas

  conjunto1 = crearConjunto(texto1.dividir(' '));

  conjunto2 = crearConjunto(texto2.dividir(' '));



  interseccion = nuevaInterseccion(conjunto1, conjunto2); //
intersección de los conjuntos

  union = nuevaUnion(conjunto1, conjunto2); // unión de los conjuntos

  coeficienteJaccard = tamaño(interseccion) / tamaño(union); // cálculo
del coeficiente
```

Como resultado de la aplicación de la función, se obtiene la similitud de ambos textos. Esta se compara con un umbral preestablecido (que tras la realización de diversas pruebas se ha estimado en “0.2”). Si la similitud obtenida es mayor que el umbral, se puede deducir que el usuario ha intentado realizar el dictado. Si la similitud es menor, se deduce lo contrario. Cabe destacar que aunque este método proporcione buenos resultados, su tasa de éxito no es del 100%, por lo que se podría dar el caso de que el usuario aún intentado hacer el dictado, lo escriba de tal manera que se obtenga una similitud por debajo del umbral (que es más común en dictados de menor longitud).

En el caso de encontrarse ante una similitud baja, se estaría en una situación en la que se carece de texto de referencia (en la situación opuesta, se tiene el dictado original). Esto se resuelve nuevamente con el uso de la inteligencia artificial, pidiendo que devuelva el mensaje corregido:

Pseudocódigo: Generar respuesta

```
función generarRespuestaDictado() {

   const API_URL = "https://api.openai.com/v1/chat/completions";

   const requestOptions = {  método: "POST", cabeceras: {"Content-Type":
"application/json", // tipo de contenido"Authorization": `Bearer
${API_KEY}` // clave},

      cuerpo: JSON.stringify({ modelo: "gpt-4", // el modelo a utilizar
mensajes: [{rol: "usuario", contenido: `Quiero que me corrijas el siguiente
texto: "${userMessage}". Devuélvemelo directamente corregido,sin
ningún tipo de introducción.`}], temperatura: 0.1 }) }

   retornar fetch(API_URL, requestOptions)

      .entonces(res => res.json())

      .entonces(data => {

         respuestaDictado  = data.choices[0].message.content; // se
actualiza el valor con el nuevo "texto de referencia"

         respuestaDictado = respuestaDictado.replace(/"/g, ''); // se
eliminan las comillas

         lista__respuesta__dictado.innerHTML = ''; // se elimina el contenido
existente


lista__respuesta__dictado.appendChild(createChatLi(respuestaDictado,
"respuesta__dictado")); // se añade el nuevo contenido

         retornar respuestaDictado;

      })

      .capturar((error) => {console.error("Hubo un error al generar el
dictado aleatorio:", error);lanzar error; // propaga el error para que
pueda ser capturado en la llamada a la función});

}      .capturar((error) => {console.error("Hubo un error al generar el
dictado aleatorio:", error); lanzar error; // propaga el error para que
pueda ser capturado en la llamada a la función });
```



```
}
```

Obteniendo la referencia necesaria para continuar con su corrección.

Número de palabras

En este punto se tienen para ambas situaciones el mensaje original del usuario y el texto para contrastar. Se va a llevar a cabo una corrección por comparación. Cada palabra del mensaje del usuario se va a comparar con aquella palabra del texto de referencia que tenga una mayor similitud.

Si ambos textos son coincidentes en el número de palabras, para una mayor eficiencia se empieza con la premisa de que las palabras a comparar comparten las mismas posiciones. Se calcula para estas, usando nuevamente el coeficiente de Jaccard, su similitud:

Pseudocódigo: Coeficiente de Jaccard en palabras

```
función jaccard_palabras(palabra1, palabra2) {

   conjunto1 = nuevoConjunto(palabra1);

   conjunto2 = nuevoConjunto(palabra2);

   interseccion = nuevaInterseccion(conjunto1, conjunto2); // tamaño de
la intersección

   union = nuevaUnion(conjunto1, conjunto2); // tamaño de la unión

   coeficienteJaccard = tamaño(interseccion) / tamaño(union); // cálculo
del coeficiente

   retornar similitud;

}
```

Si la similitud supera un umbral preestablecido (que se estimó en “0.5”, nuevamente a base de prueba y error), se deduce que ambas palabras deben ser objeto de comparación (Figura 7.17). En caso contrario, se toma la palabra del usuario y se calcula su similitud en relación con la totalidad de las palabras del texto de referencia (Figura 7.18). Dando como resultado el cálculo de la similitud entre todas sus posibilidades. Quedándose (siempre que se supere el umbral), con aquella palabra en la que la similitud sea mayor. En caso de no encontrar ninguna palabra (dado que no se supera el umbral), esta se descarta y se añade al array de “palabras sobrantes”.

Primer ejemplo práctico (coincidentes en número de palabras)

Inicio del proceso:
(1) El usuario proporciona un texto: "me gusta el color amarillo".
(2) Se tiene el texto de referencia: "Me gusta el color amarillo".
Comparación de palabras:
(3) Se toma por premisa que las palabras a comparar, comparten posición.
(4) Se calcula la similitud de la primera palabra del texto del usuario ("me") con la primera palabra del texto de referencia ("Me").
Similitud superada:
(5) Dado que la similitud entre "me" y "Me" supera el umbral, se toman como objetivo de comparación, se eliminan de ambos textos y se procede con la siguiente palabra.
Continuación de la comparación:
(6) Se toma por premisa que las palabras a comparar, comparten posición.
(7) Se compara la segunda palabra del texto del usuario ("gusta") con la siguiente palabra del texto de referencia ("gusta")...
Iteración:
(8) El proceso continúa comparando palabra por palabra entre el texto del usuario y el texto de referencia, calculando la similitud y evaluando si supera el umbral.

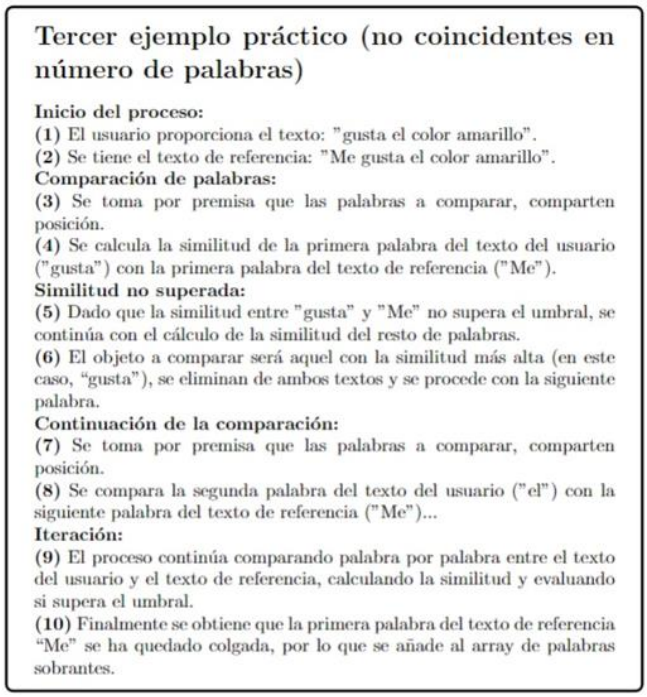
Figura 7.17: 1º Ejemplo práctico

Segundo ejemplo práctico (coincidentes en número de palabras)

Inicio del proceso:
(1) El usuario proporciona el texto: "gusta me el color amarillo".
(2) Se tiene el texto de referencia: "Me gusta el color amarillo".
Comparación de palabras:
(3) Se toma por premisa que las palabras a comparar, comparten posición.
(4) Se calcula la similitud de la primera palabra del texto del usuario ("gusta") con la primera palabra del texto de referencia ("Me").
Similitud no superada:
(5) Dado que la similitud entre "gusta" y "Me" no supera el umbral, se continúa con el cálculo de la similitud del resto de palabras.
(6) El objeto a comparar será aquel con la similitud más alta (en este caso, "gusta"), se eliminan de ambos textos y se procede con la siguiente palabra.
Continuación de la comparación:
(7) Se toma por premisa que las palabras a comparar, comparten posición.
(8) Se compara la segunda palabra del texto del usuario ("me") con la siguiente palabra del texto de referencia ("Me")...
Iteración:
(9) El proceso continúa comparando palabra por palabra entre el texto del usuario y el texto de referencia, calculando la similitud y evaluando si supera el umbral.

Figura 7.18: 2º Ejemplo práctico

En caso de que el número de palabras sea distinto, se calcula directamente la similitud para la totalidad de palabras, dejando de lado la premisa inicial (Figura 7.19).



Tercer ejemplo práctico (no coincidentes en número de palabras)

Inicio del proceso:
(1) El usuario proporciona el texto: "gusta el color amarillo".
(2) Se tiene el texto de referencia: "Me gusta el color amarillo".
Comparación de palabras:
(3) Se toma por premisa que las palabras a comparar, comparten posición.
(4) Se calcula la similitud de la primera palabra del texto del usuario ("gusta") con la primera palabra del texto de referencia ("Me").
Similitud no superada:
(5) Dado que la similitud entre "gusta" y "Me" no supera el umbral, se continúa con el cálculo de la similitud del resto de palabras.
(6) El objeto a comparar será aquel con la similitud más alta (en este caso, "gusta"), se eliminan de ambos textos y se procede con la siguiente palabra.
Continuación de la comparación:
(7) Se toma por premisa que las palabras a comparar, comparten posición.
(8) Se compara la segunda palabra del texto del usuario ("el") con la siguiente palabra del texto de referencia ("Me")...
Iteración:
(9) El proceso continúa comparando palabra por palabra entre el texto del usuario y el texto de referencia, calculando la similitud y evaluando si supera el umbral.
(10) Finalmente se obtiene que la primera palabra del texto de referencia "Me" se ha quedado colgada, por lo que se añade al array de palabras sobrantes.

Figura 7.19: 3º Ejemplo práctico

Fallos

Las palabras que van a ser objeto de comparación se analizan para ver si el usuario ha cometido algún fallo. En el caso de que el número de caracteres sea distinto, se sabe a ciencia cierta que está mal y, por ende, se añadirán (respetando siempre su posición) al array de palabras incorrectas y correctas respectivamente. Si coinciden en el número de caracteres hay que revisar manualmente si tienen alguna diferencia y de ser así, se añadirán de la misma forma.

Pseudocódigo: diferencia entre palabras

```
función numeroDiferencias(palabra1, palabra2) {

  si (longitud(palabra1) !== longitud(palabra2)) {  // se verifica si las palabras tienen la misma longitud

    lanzar nuevo Error("Las palabras deben tener la misma longitud para comparar las diferencias.");

  }

  dejar diferencias = 0;

  para (dejar i = 0; i < longitud(palabra1); i++) { // se compara cada carácter de las palabras

    si (palabra1[i] !== palabra2[i]) { diferencias++; }
  }

  retornar diferencias;

}
```

De tal manera que, si queremos obtener para cada palabra incorrecta, su corrección, basta con recuperar la posición de esta y trasladarla al otro array.

Flujo de corrección

En este momento se tiene como resultado las palabras incorrectas, correctas y sobrantes. Lo que va a permitir la creación de un flujo de corrección común para todos los dictados:

1. Mostrar el mensaje original del usuario
2. Mostrar únicamente las palabras incorrectas
3. Mostrar únicamente las palabras correctas
4. Mostrar el dictado de referencia

Este flujo permite al usuario visualizar de manera clara y concisa sus fallos, así como las correcciones sugeridas. La presentación de la información se realiza de forma simplificada para una mejor comprensión y focalización en los aspectos que requieren atención. Es importante destacar que la implementación específica de este flujo puede variar según el diseño de la interfaz de usuario y las preferencias de presentación. La clave reside en ofrecer una experiencia intuitiva y fácil de entender para el usuario, permitiéndole identificar y comprender sus errores de manera eficiente.

7.5 Iteración 5

7.5.1 Análisis

A lo largo de esta iteración, se llevará a cabo la implementación del algoritmo de Leitner para mejorar el proceso de aprendizaje y memorización de palabras por parte del usuario. Además, del manejo de las palabras almacenadas en la base de datos y de su visualización en la pantalla. Posteriormente, se realizará la generación de dictados para que se basen en las palabras que están siendo objeto de estudio.

El objetivo principal es proporcionar al usuario una herramienta efectiva para aprender palabras en forma de juego, además de mejorar la generación de dictados para que estén alineados con el proceso de aprendizaje. Se implementarán de esta manera, los siguientes casos de uso:

- CU-16: Visualizar fallos
- CU-17: Visualizar algoritmo de Leitner
- CU-11: Dictado aleatorio

7.5.2 Diseño e implementación

CU-11: Dictado aleatorio

Este modo tiene como objetivo el seguimiento de los errores del usuario y su respectivo repaso. Para ello habrá que trabajar con el resultado obtenido: palabras incorrectas, correctas y sobrantes. Pero antes hay que decidir cuáles van a ir a la base de datos.

Guardar las palabras sobrantes carecería de sentido ya que estas no indican que fuesen un error como tal; como el propio nombre indica, son palabras que al final han sobrado (ya sea del mensaje del usuario o del texto de referencia). Guardar las palabras incorrectas tampoco resultaría útil, ya que la manera de repasar las propias palabras es añadiéndolas en un nuevo dictado. Y no puede darse el caso de realizar dictados que contengan palabras mal escritas. Por lo que la mejor opción es almacenar las palabras correctas correspondientes.

Añadir en la base de datos la totalidad de estas palabras sería defendible, pero no eficiente. El objetivo principal es el aprendizaje de estas y como he dicho anteriormente, para conseguirlo se van a añadir en posteriores dictados. Por lo que la manera de aprender sería repitiendo (y así repasando), las palabras falladas.

Los dictados siempre van a seguir el mismo formato: empiezan con mayúscula, terminan con punto final y utilizan signos de puntuación (en su mayoría comas y puntos). Por lo que añadir un fallo relativo a estas condiciones no sería eficiente, dado que en los posteriores dictados se repetirían (y se volverían a repasar).

Para una mayor eficiencia, las palabras deberían pasar por un posterior filtrado:

Pseudocódigo: Filtrar palabras

```
Función filtradoPalabras(resultado):

   palabras_correctas = resultado.palabrasCorrectas

   palabras_incorrectas = resultado.palabrasIncorrectas

   Para cada i desde 0 hasta la longitud de palabras_correctas - 1:

      fallo_por_mayuscula = falloPorMayuscula(palabras_correctas[i],
palabras_incorrectas[i]) // ¿la 1º letra de cada palabra se escribe
diferente (una mayúscula y otra minúscula)?

      Intentar:

         mayuscula_natural = await
mayusculaNatural(palabras_correctas[i]) // ¿la palabra correcta se
escribe con mayúscula inicial de forma natural?

         Si fallo_por_mayuscula es igual a 1: // si entre los fallos está que no
escribió con mayúscula una palabra que debería

            Si mayuscula_natural es igual a 0: // y que esa palabra NO se
escribe de manera natural en mayúscula...

               diferencias = numeroDiferencias(palabras_correctas[i],
palabras_incorrectas[i]) // número de caracteres distintos de ambas
palabras

               Si diferencias es menor o igual a 1: // si no tiene más fallos a
parte de la mayúscula, se eliminan

                  // se eliminan ambas palabras (la correcta y la incorrecta)

                  palabras_correctas.splice(i, 1)

                  palabras_incorrectas.splice(i, 1)

                  i-- // se disminuye el índice para compensar el splice

      Capturar error:

         Imprimir "Error", error
```

En el que sólo se dejan las palabras si el fallo es por mayúscula natural. Es decir, si el fallo de la palabra fue únicamente la mayúscula inicial y esta se escribe de forma natural en mayúscula (nombres propios, de países, de ríos…) sí que se almacenaría. Pero en el caso de que no se escribiese de forma natural en mayúscula ni tuviese algún otro fallo, se descarta.

Una vez sabiendo las palabras que hay repasar, hay que decidir en qué momento se va a llevar a cabo y cuando se van a dar por aprendidas. Para ello, se va a hacer uso del método Leitner.

Método Leitner

El algoritmo de Leitner es un método popular utilizado en el aprendizaje espaciado para ayudar a recordar información de manera efectiva. Funciona mediante la división del material de estudio en diferentes cajas o niveles, y al realizar correctamente una respuesta, la palabra se mueve a una caja de nivel superior, mientras que una respuesta incorrecta resulta en que la palabra retroceda a una caja de nivel inferior.

Cajas o niveles

El material de estudio se organiza en varias cajas (Figura 7.20), que representan el grado de dominio de una palabra. Cada caja representa el momento en el que la palabra se vuelve a estudiar. Las palabras que están en cajas con un nivel inferior tienen una fecha de estudio más próxima y repetitiva. Sin embargo, aquellas palabras que están en cajas con niveles más altos tienen una fecha de estudio más alejada.

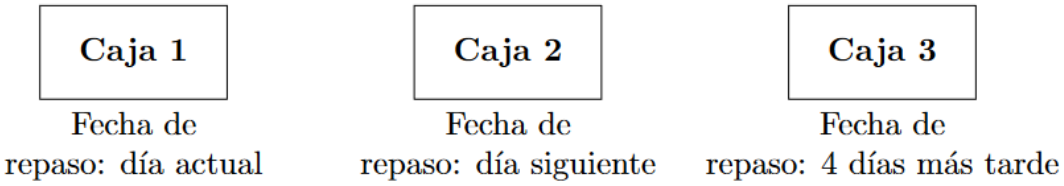


Figura 7.20: Cajas método Leitner

Palabras de estudio

Cada palabra de estudio se coloca inicialmente en la caja de nivel más bajo. Las palabras se mueven entre las cajas dependiendo de si la respuesta (en el juego que sale al final de cada dictado) fue correcta o incorrecta.

Estudio inicial y curva del olvido

Durante el estudio, se seleccionan las palabras cuya fecha de estudio coincide con el día actual (o anteriores). De esta manera, se consigue un estudio acorde a la curva del olvido. Los repasos serán más numerosos al principio y a medida que estos vayan dando un resultado

satisfactorio, se prolongarán en el tiempo, dejando espacio al resto de palabras que se necesiten aprender.

Prueba de aprendizaje

Se selecciona una palabra de estudio de la caja actual y se intenta responder correctamente a la pregunta del juego.

Respuestas correctas o incorrectas

Si la respuesta es correcta, la palabra se mueve a la siguiente caja de nivel superior. Si la respuesta es incorrecta, la palabra se mueve a la primera caja. El objetivo es mover las palabras hacia las cajas de nivel más altas, hasta llegar a un punto en el que la palabra se da por aprendida.

Aplicación

Este método se aplica a través del juego una vez terminado el dictado. Se sacan aleatoriamente las palabras que el usuario ha escrito mal junto con la pregunta "¿Las siguientes palabras están bien escritas?" y dos botones: "Sí" y "No" (Figura 7.21). Al sacarse las palabras de manera aleatoria, pueden ser pertenecientes al array de palabras incorrectas o a su respectivo array de palabras correctas. Por lo que el usuario tiene que estar atento para poder acertar. Esto tiene como excepción el caso de que el usuario realice el dictado de manera perfecta (sin ningún error), por lo que se daría la palabra como acertada sin necesidad de realizar el juego.

Como se ha explicado anteriormente, en caso de que acierte en el juego se pasa a la siguiente y en caso contrario, se vuelve a la principal. Además, los dictados son generados según la palabra que sea objeto de estudio. Es decir, se escoge una palabra perteneciente a la base de datos de las palabras a repasar que tenga una fecha de estudio actual o anterior. Y se genera un dictado en el que esta palabra esté presente.

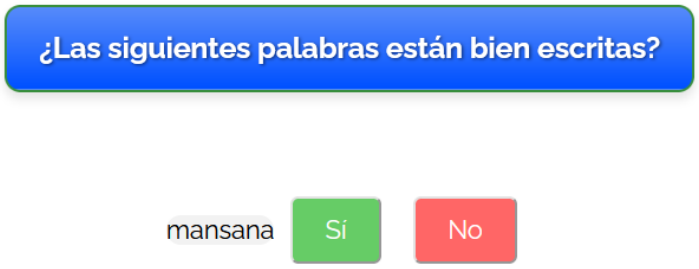


Figura 7.21: Juego Leitner

CU-16: Visualizar fallos y CU-17: Visualizar algoritmo de Leitner

El usuario podrá acceder a los casos de uso 15 y 16 haciendo clic en el engranaje (Figura 7.22) que se encuentra debajo del título.

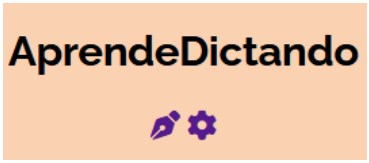


Figura 7.22: Engranaje

Donde se encontrará con las palabras (Figura 7.23) que ha fallado anteriormente. Para cada una de estas se especifica su nivel de caja y la próxima fecha de estudio programada. Estas palabras se actualizarán en tiempo real a medida que el usuario responda a las preguntas del juego final.

Figura 7.23: Palabras Algoritmo Leitner

Además, si el usuario hace clic en el icono del lápiz, saldrá una explicación elaborada (Figura 7.24) en LaTeX sobre el funcionamiento del algoritmo de Leitner.

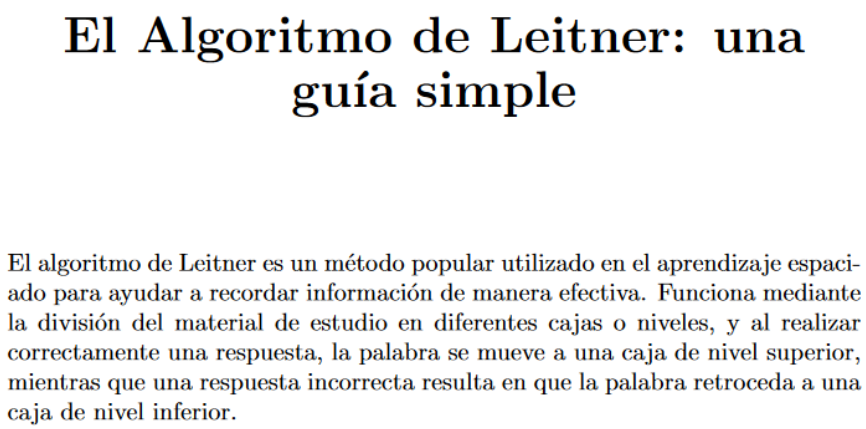


Figura 7.24: explicación algoritmo Leitner

7.6 Iteración 6

7.6.1 Análisis

A lo largo de esta iteración se realizará la creación de la página dedicada a mostrar las diferentes reglas ortográficas. Se tendrán que definir dos matrices para diferenciar las reglas ortográficas de los otros elementos del castellano. Se elaborará también la teoría correspondiente a cada regla y se definirán los ejemplos de otros elementos del castellano. Cuando se seleccione una determinada regla y se elija la opción de realizar un dictado, este tiene que estar elaborado en función a la regla seleccionada.

El objetivo principal es proporcionar al usuario una herramienta educativa completa que le permita aprender y practicar las reglas ortográficas del español de manera efectiva. Implementando así los siguientes casos de uso:

- CU-12: Reglas ortográficas
- CU-18: Seleccionar elemento
- CU-19: Visualizar teoría
- CU-13: Dictado reglas ortográficas

7.6.2 Diseño e implementación

CU-12: Reglas ortográficas

La página referente a las reglas ortográficas (Figura 7.25) tiene como principal característica su simpleza. Únicamente dispone de dos matrices, un buscador y un botón para volver al menú. Ambas matrices se encuentran en el centro diferenciadas claramente entre sí. En cada matriz se pueden ver sus distintos elementos, y aunque son visualmente distinguibles, la barra de búsqueda (Figura 7.26) facilita esta tarea.

Figura 7.25: Página reglas ortográficas

Figura 7.26: Barra de búsqueda

Cabe destacar que la generación de dictados en segunda matriz se realiza de una manera diferente de la del resto de dictados. Esto es debido a que en el momento de practicar con el refranero español (o demás elementos) los dictados obtenidos resultaban pobres y no hacían honor a su riqueza lingüística en el castellano. Por lo que la totalidad de "otros elementos" que se encuentran en la página web, son resultado de una búsqueda y filtrado a mano; a través en su mayoría, de libros y artículos. Obteniendo así, una cantidad suficiente, de longitud variable, de fuentes fiables y adaptadas a la comprensión que se espera de una edad tan temprana. Aunque es cierto que la dificultad tiene una mayor presencia debido a que se ha priorizado la inclusión de algunos ejemplos por su uso cotidiano o importancia en la cultura.

CU-17: Seleccionar elemento y CU-19: Visualizar teoría

Haciendo clic en el elemento deseado se abrirá un modal (Figura 7.27) en el que se podrá elegir dos opciones: visualizar la teoría correspondiente al elemento seleccionado o realizar un dictado en base a este:

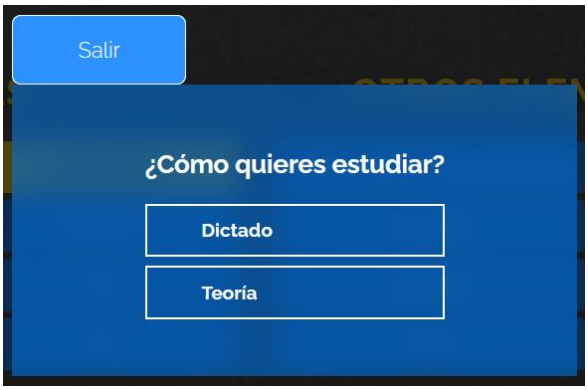
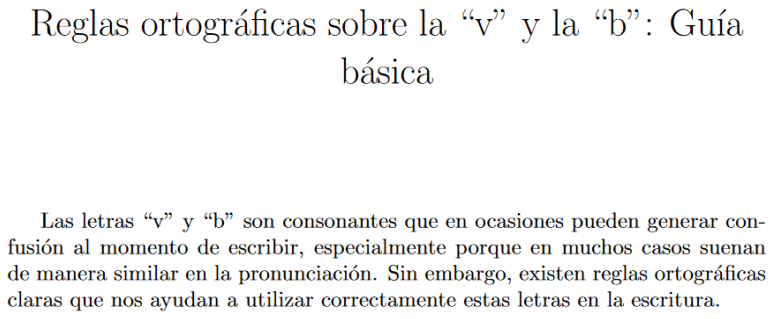


Figura 7.27: Seleccionar elemento

La teoría (Figura 7.28) ha sido redactada con LaTeX para darle una mayor claridad. Tiene una redacción sencilla para facilitar su comprensión y que se adecúe al público objetivo.



Reglas ortográficas sobre la "v" y la "b": Guía básica

Las letras "v" y "b" son consonantes que en ocasiones pueden generar confusión al momento de escribir, especialmente porque en muchos casos suenan de manera similar en la pronunciación. Sin embargo, existen reglas ortográficas claras que nos ayudan a utilizar correctamente estas letras en la escritura.

Figura 7.28: Teoría reglas ortográficas

CU-13: Dictado reglas ortográficas

La realización de los dictados según el elemento seleccionado se hace de manera diferente según el elemento se encuentre en la primera o segunda matriz. De encontrarse en la primera, los dictados serán generados a través de la inteligencia artificial, simplemente se modifica el contenido de la petición para que el dictado se adecúe a las reglas. De pertenecer a la segunda matriz, el dictado se obtendrá de los ejemplos elaborados a mano.

7.7 Iteración 7

7.7.1 Análisis

A lo largo de esta iteración se llevará a cabo tanto el sistema de gestión de amigos del usuario como la creación del sistema de puntuación y ambos rankings: global y exclusivo de amigos. Además, se aplicarán estos cambios al modo de "Dictado competición", en el que los usuarios podrán competir (mediante la realización de dictados) para acumular puntos y mejorar su posición en los rankings correspondientes.

El objetivo principal es mejorar la interacción social entre los usuarios al permitirles agregar y eliminar amigos, así como competir entre ellos en dictados. De esta forma se implementan los siguientes casos de uso:

- CU-08: Añadir amigos
- CU-09: Eliminar amigos
- CU-14: Dictado competición
- CU-20: Visualizar ranking global
- CU-21: Visualizar ranking amigos

7.7.2 Diseño e implementación

CU-08: Añadir amigos y CU-09: Eliminar amigos

Se podrá acceder a los respectivos amigos haciendo clic en el botón correspondiente que se encuentra en el menú (Figura 7.29).

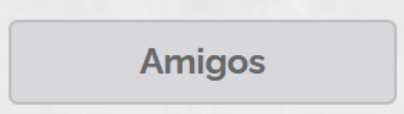


Figura 7.29: Botón amigos

Se abrirá un modal (Figura 7.30) en el se mostrarán los respectivos amigos que tiene un usuario con un botón de "eliminar". Al hacer clic en este botón se eliminará el respectivo usuario como amigo. En caso de querer añadir nuevos amigos, bastará con escribir su nombre de usuario y pulsar en el botón de "Añadir".

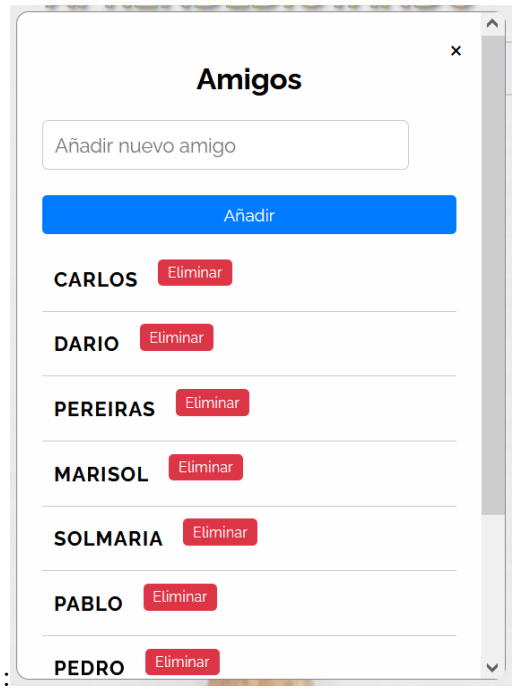


Figura 7.30: Modal amigos

CU-20: Visualizar ranking global y CU-21: Visualizar ranking amigos

Se podrá acceder a los respectivos ránquines haciendo clic en el icono que se encuentra en el modo de competición (Figura 7.31).

Figura 7.31: Icono ranking

Al hacerlo se abrirá un modal en el que se encuentran ambos ránquines (Figura 7.32). En el que los usuarios están ordenados por orden ascendente de puntuación y los tres primeros usuarios con más puntuación están resaltados en dorado.

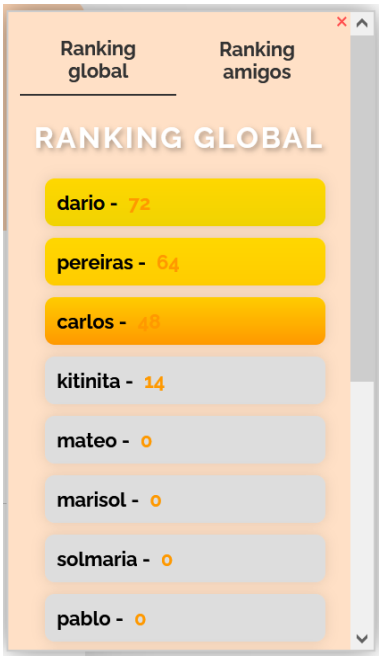


Figura 7.32: Ranking

CU-14: Dictado competición

En la competición se va a buscar el espíritu competitivo de los usuarios y por consecuencia, la generación de dictados va a estar limitada en el nivel 3 (buscando la mayor complejidad).

Constará de un total de dos ránquines: uno global y otro de amigos (Figura 7.32). En el primero se verá la totalidad de personas que se encuentran en la base de datos y en el segundo, solamente los amigos que el usuario tenga añadidos.

Los puntos van de 0 a 100. Al principio se asigna una puntuación a cada usuario de 0 puntos y esta irá creciendo a medida que el usuario realice dictados con pocos fallos. La manera de puntuar es únicamente en base del número de fallos.

Está diseñada para no frustrar al usuario con una puntuación baja, pero para tampoco desmotivarlo poniendo muy fácil el hecho de sacar una puntuación alta. Es por eso que, a menor puntuación, habrá una mayor facilidad para subir en el ranking. Pero a mayor puntuación, los fallos penalizarán más y se ganarán menos puntos. De esta manera se espera que la media de usuarios se mantenga en un equilibrio entre una puntuación ni demasiado baja ni demasiado alta, y que los usuarios que más trabajen y estudien, obtengan las puntuaciones más altas (Figura 7.33).

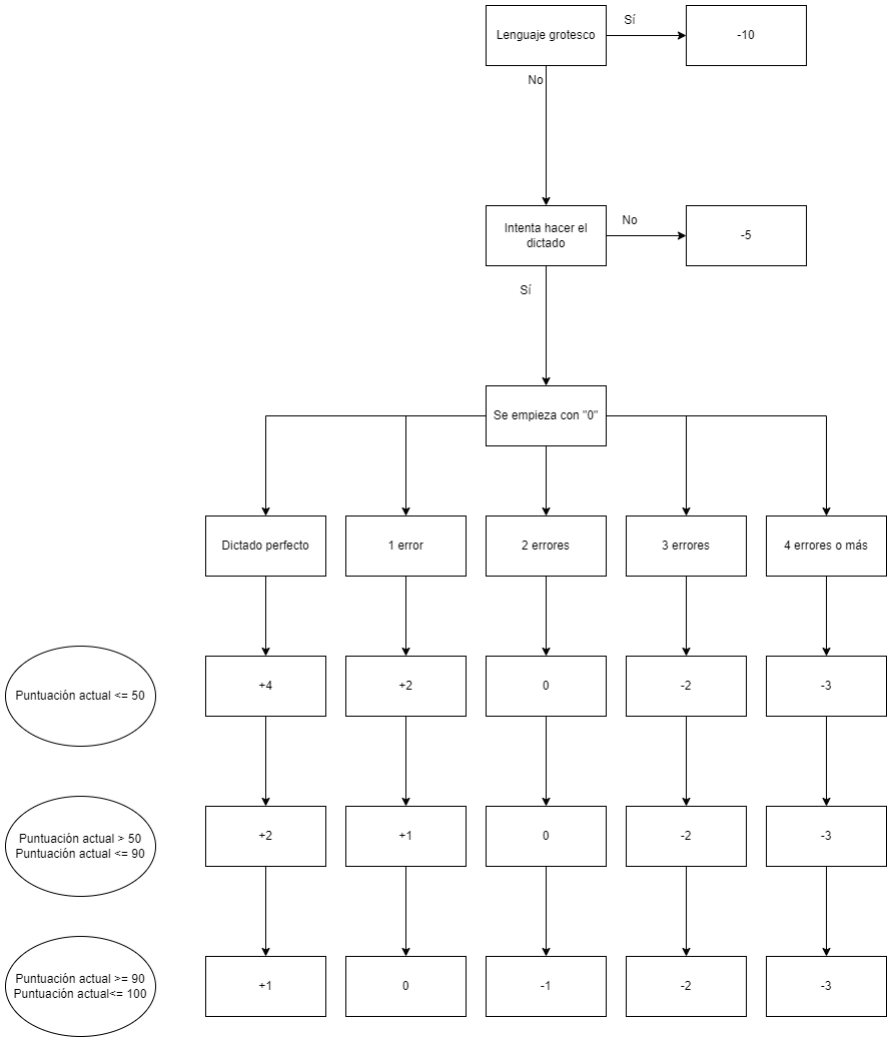


Figura 7.33: Puntuación

7.8 Iteración 8

A lo largo de esta iteración se redactó la memoria y realizó la corrección de algunos fallos menores y actualizaciones de tamaño pequeño:

- Actualización del diseño de la página de “Dictado competición”.
- Corrección en el ranquin de amigos: el usuario actual también se muestra.
- Eliminación de código sobrante.
- Las palabras en el modal de la base de datos del dictado ahora se ordenan por antigüedad.

- Corregido algoritmo de Leitner (la fecha se actualiza conforme el día actual y no el día que tenía establecido).
- Cambio de diseño del botón de "Mi cuenta" del menú. Su nombre ya no es “Mi cuenta”, sino el nombre del usuario actual.

7.8.1 Pruebas finales

A pesar de realizarse para cada iteración de forma individual pruebas de manera continua para no destrozar las funcionalidades ya hechas, en esta última iteración se realizó un plan de pruebas final para asegurar la eficacia y precisión de la página web. Este incluye los siguientes aspectos:

1. Dictados de distintos niveles:
    a. Nivel 1: Dictados sencillos con frases cortas.
    b. Nivel 2: Dictados sencillos con frases un poco más largas.
    c. Nivel 3: Dictados sencillos con frases considerablemente más largas
2. Pruebas manuales: Dada la imposibilidad de automatizar las pruebas debido entre otros factores a la variabilidad de las respuestas, se generaron dictados y se introdujeron respuestas intencionalmente incorrectas para verificar su corrección.
3. Pruebas de validación con distintos usuarios: Un grupo diverso con distintos perfiles probaron la herramienta en sus distintos niveles, analizando así la corrección de sus correspondientes dictados.

A continuación, se presentan algunos ejemplos de dictados utilizados en las pruebas, junto con los resultados obtenidos:

- Nivel 1
    - “El sol brilla en el cielo”
        - Respuesta del usuario: “El sol brila en el cielo”.
        - Corrección: Error en “brila”. Corrección: “brilla”.
    - La gata duerme profundamente
        - Respuesta del usuario: “La gata durme profundamente”.
        - Correción: Error en “durme”. Corrección: “duerme”.

- Nivel 2
    - “La pelota del niño es roja y grande”
        - Respuesta del usuario: “La pelota es roga y gande”.
        - Corrección: Error en “roga” y “gande”. Corrección: “roja” y “grande”.
    - El perro corre rápido en el parque
        - Respuesta del usuario: “El perro corre rapido en el parqe”
        - Corrección: Error en “rapido” y “parqe”. Corrección: “rápido” y “parque”
- Nivel 3
    - "El gato blanco juega con una bola de lana roja en el jardín."
        - Respuesta del usuario: "El gato blanco juega con una bola de lana roga en el ardin."
        - Corrección**:** Error en “roga” y “ardin"”. Corrección: “roja” y “jardín”.
    - "La niña corre por el parque mientras su hermano monta en bicicleta."
        - Respuesta del usuario: "La nina corre por el parque mientras su ermano monta en bicicleta."
        - Corrección**:** Error en “niña” y “ermano”. Corrección: “niña” y “hermano”.

El plan de pruebas y las validaciones con los usuarios demostraron que la herramienta es efectiva y cumple con los objetivos planteados.

8 Conclusiones y trabajo futuro

8.1 Conclusiones

El desarrollo de la página web educativa enfocada en mejorar la ortografía española en niños de edad temprana ha sido un proceso enriquecedor y gratificante. A través de la integración de tecnología, inteligencia artificial y ludificación, se ha creado una herramienta innovadora que busca hacer del aprendizaje una experiencia más efectiva y atractiva.

Durante el proceso de desarrollo, se han alcanzado varios hitos importantes, como el diseño e implementación de algoritmo de corrección ortográfica y la integración del algoritmo de Leitner para optimizar el proceso de aprendizaje a largo plazo.

Además, la incorporación de elementos de ludificación ha contribuido significativamente a mantener el interés y la motivación de los usuarios, convirtiendo el aprendizaje de la ortografía en una experiencia más atractiva y divertida.

Sin embargo, aún existen áreas de mejora y desarrollo futuro que pueden explorarse para enriquecer aún más la aplicación y maximizar su impacto educativo.

8.2 Trabajo futuro

El siguiente paso por realizar en el trabajo actual está claro: abarcar la generación de los audios desde otro punto de vista. De querer mantener la generación de dictados actual (en su mayoría a través de inteligencia artificial), el paso a audio tiene que realizarse de la misma manera dado a la imposibilidad de realizar una grabación profesional (no se puede grabar aquello que no se tiene).

En esta opción habría que explorar la posibilidad de mejorar la calidad de los audios generados por la inteligencia artificial. Esto implica investigar y desarrollar técnicas avanzadas de procesamiento de voz y síntesis de audio que puedan producir resultados más naturales y realistas. Además, se podría considerar el uso de herramientas de posprocesamiento de audio para mejorar la calidad y la claridad de los audios generados.

Otra opción sería tener una base de datos con suficientes dictados. Esto implicaría realizar un trabajo exhaustivo de recopilación y selección de dictados. Esta recopilación tiene que ser variada y extensa para evitar que se repitan los dictados con cierta frecuencia y posteriormente, realizar la grabación de estos. Sería aconsejable que la grabación la realizaran profesionales para así garantizar una experiencia de audio óptima.

Independientemente de la opción elegida, es crucial asegurar que los audios generados sean claros, precisos y de alta calidad para proporcionar una experiencia de aprendizaje efectiva y agradable para los usuarios.

Sería interesante establecer un sistema que evite que la puntuación dependa únicamente de los dictados realizados. Actualmente podría darse el caso de que existan usuarios con una puntuación alta que dejen de realizar dictados con el objetivo de mantener esta puntuación (dado que, si no los realizan, ni ganan ni pierden puntos). Este sistema podría estar basado en que cada número determinado de dictados perfectos que realice un usuario repercuta en una bajada de puntos al resto de usuarios. También se podría implementar que cada día se pierda un número determinado de puntos. Sea cual sea el método escogido, es importante evitar estancamientos y priorizar el dinamismo en los ránquines.

Además, la página cuenta con numerosas áreas de mejora y de desarrollo futuro que se pueden explorar para una mayor riqueza de la aplicación:

- Ampliación del contenido: Se puede ampliar el contenido de la aplicación para incluir más reglas ortográficas, ejercicios y actividades que aborden diferentes aspectos de la ortografía.
- Personalización del aprendizaje: Incorporar funcionalidades que permitan adaptar el contenido y la dificultad de los dictados según las necesidades y habilidades individuales de cada usuario.
- Integración de herramientas de retroalimentación: Implementar herramientas que proporcionen retroalimentación más detallada y específica sobre los errores cometidos por los usuarios, con sugerencias para mejorar.
- Expansión a otras plataformas: Expandir la aplicación a otras plataformas (como aplicaciones móviles), para llegar a un público más amplio.
- Colaboración con instituciones educativas: Establecer colaboraciones con instituciones educativas para integrar la aplicación en programas de enseñanza y utilizarla como una herramienta complementaria en el aula.
- Implementar dictados que tengan en cuenta la diversidad lingüística y regional para abarcar los diferentes acentos y variantes del idioma español.

Lista de acrónimos

URL: Uniform Resource Locator. Secuencia de caracteres que especifica la ubicación de un recurso en Internet, como una página web, un archivo o un servicio.

BD: Base de Datos. Conjunto organizado de datos relacionados que se almacenan y se acceden electrónicamente desde un sistema informático. Las bases de datos se utilizan para gestionar y manipular grandes volúmenes de información de manera eficiente.

HTML: HyperText Markup Language. Lenguaje de marcado estándar utilizado para crear y diseñar páginas web. Utiliza etiquetas para definir la estructura y el formato del contenido de una página web.

CSS: Cascading Style Sheets. Lenguaje utilizado para definir la presentación y el estilo visual de documentos HTML. CSS permite separar el contenido de la presentación, lo que facilita la creación de diseños y estilos coherentes en las páginas web.

API: Application Programming Interface. Conjunto de reglas y protocolos que permiten a diferentes aplicaciones o sistemas comunicarse entre sí. Las API facilitan la integración de diferentes componentes de software y la creación de aplicaciones complejas.

SQL: Structured Query Language. Lenguaje de programación utilizado para gestionar y manipular datos en bases de datos relacionales. SQL permite realizar consultas, inserciones, actualizaciones y eliminaciones de datos, así como definir y gestionar la estructura de las bases de datos.

Glosario

Feedback: Retroalimentación que se recibe sobre un trabajo o proceso. Puede ser tanto positivo como negativo y sirve para mejorar y ajustar futuras acciones o resultados.

Footer: Sección al final de una página web que contiene información adicional, como enlaces de navegación, información de contacto, derechos de autor, entre otros.

OpenAI: Empresa de investigación en inteligencia artificial que desarrolla y ofrece una variedad de modelos de lenguaje avanzados utilizados en una amplia gama de aplicaciones.

LaTeX: Sistema de composición de textos, utilizado principalmente para la creación de documentos científicos y técnicos. Se caracteriza por su alta calidad tipográfica y su capacidad para manejar fórmulas matemáticas complejas.

Login: Proceso mediante el cual un usuario accede a un sistema o aplicación introduciendo sus credenciales de autenticación, como un nombre de usuario y una contraseña.

.php: Extensión de archivo utilizada en archivos de código fuente PHP, un lenguaje de programación ampliamente utilizado en el desarrollo web para la creación de sitios dinámicos e interactivos.

Query: Solicitud de información o instrucción dirigida a una base de datos para recuperar, manipular o actualizar datos.

phpMyAdmin: Herramienta de administración de bases de datos MySQL basada en web. Permite a los usuarios administrar bases de datos MySQL mediante una interfaz gráfica de usuario.

ID: Acrónimo de "Identificación". En el contexto de la informática, se refiere a un identificador único asignado a un objeto, usuario o entidad en un sistema.

Framework: Conjunto de herramientas, bibliotecas y convenciones que proporcionan una estructura para el desarrollo de aplicaciones de software. Ayuda a los desarrolladores a crear aplicaciones de manera más rápida y eficiente.

Scrum Master: Rol dentro del marco de trabajo ágil Scrum. El Scrum Master es responsable de facilitar el equipo y ayudarlo a adoptar y mantener prácticas ágiles.

Product Owner: Rol dentro de Scrum. El Product Owner es responsable de maximizar el valor del producto y del trabajo del equipo de desarrollo, representando los intereses del cliente y gestionando el Product Backlog.

Product Backlog: Lista priorizada de todas las características, funciones, mejoras y correcciones de errores pendientes para un producto. Es propiedad del Product Owner y se utiliza para planificar y gestionar el trabajo del equipo de desarrollo.

Bibliografía

¿Qué es la metodología Scrum? Y Gestión de proyectos Scrum. (2024). Obtenido de https://www.nimblework.com/es/agile/que-es-scrum/

Análisis de requisitos de software. (2023). Obtenido de https://appmaster.io/es/blog/analisis-de-requisitos-de-software

API Reference OpenAI. (2024). Obtenido de https://platform.openai.com/docs/api-reference

Azaustre, C. (2021). *Aprendiendo JavaScript: Desde cero hasta ECMAScript 6+.* Independently published.

Bahit, E. (2020). *Python para Principiantes.* Independently published.

Borrás-Gené, O. (2022). *Introducción a la gamificación o ludificación (en educación).*

Gadam, S. (2023). *What is iterative and incremental development?* Obtenido de https://blog.logrocket.com/product-management/what-is-iterative-incremental-development-process-examples/

Hart-Davis, G. (2023). *Teach Yourself VISUALLY HTML and CSS.* Wiley.

Haverbeke, M. (2018). *Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming.* No Starch Press.

JIMÉNEZ, Ó. R. (2021). *Python a fondo.* Marcombo.

Johnson, B. (2019). *Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers.* Wiley.

Marczewski, A. (2013). *Gamification: A Simple Introduction.*

McCarthy, R. (2020). *Agile y Scrum: Descubra el poder de la gestión de proyectos Agile, Lean Thinking, el proceso Kanban y Scrum.* Independently published.

Página web de CSS. (01 de 04 de 2024). Obtenido de https://developer.mozilla.org/es/docs/Web/CSS

Página web de GitHub. (07 de 04 de 2024). Obtenido de https://github.com/

Página web de HTML. (27 de 03 de 2024). Obtenido de https://developer.mozilla.org/es/docs/Web/HTML

Página web de JavaScript. (02 de 04 de 2024). Obtenido de https://developer.mozilla.org/es/docs/Web/JavaScript

Página web de Python. (03 de 04 de 2024). Obtenido de https://www.python.org/

Página web de Visual Studio Code. (27 de 03 de 2024). Obtenido de https://code.visualstudio.com/

Página web de Xampp. (06 de 04 de 2024). Obtenido de https://www.apachefriends.org/es/index.html

Tsitoara, M. (2019). *Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer.* Apress.

VV.AA. (2012). *Las mejores frases y citas célebres.* Plutón Ediciones.

Watts, G. (2013). *Scrum Mastery: From Good To Great Servant-Leadership.* Inspect & Adapt Ltd.