UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Aplicación para la ayuda a niños con problemas de dislexia

María Monserrat Sastre
Tutor: Carlos Aguirre Maeso

Julio 2019

Aplicación para la ayuda a niños con problemas de dislexia

AUTOR: María Monserrat Sastre
TUTOR: Carlos Aguirre Maeso

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2019

Resumen

En la actualidad, son muchos los niños que presentan algún tipo de problema a la hora de realizar procesos relacionados con la lectoescritura.

A medida que pasan los años, es mayor la dimensión de datos que una persona ha de saber procesar, por lo que es imprescindible tener un gran dominio del lenguaje.

Para ello, además de practicar en el colegio y tener soporte de profesionales como psicólogos o logopedas, es necesario entrenar en casa de forma activa.

Las metodologías que los padres actualmente pueden aplicar son limitadas, siendo las más populares el uso de ejercicios en papel los cuales se pueden descargar en Internet para después imprimirlos, y la utilización de aplicaciones bajo un sistema Android.

No existen muchas soluciones que se puedan realizar directamente desde un ordenador, y las que existen son pobres en contenido o no proporcionan un estudio adecuado para analizar las respuestas del usuario y por tanto saber en qué hay que mejorar.

Este proyecto está dedicado a la creación de una aplicación que permita a niños cuya edad está comprendida entre los 5 y los 8 años realizar ejercicios que mejoren dichas capacidades de lectoescritura, haciendo hincapié en las tareas de reconocimiento de símbolos y la disección y entendimiento de palabras.

Para el desarrollo de este proyecto se ha utilizado el famoso motor de videojuegos Unity debido a su potencia y variedad de opciones. Además, también se ha usado el entorno de desarrollo Visual Studio, mediante el cual se han generado diversos scripts de utilidad programados en C#.

Palabras clave

Dislexia, aplicación, niños, Unity, C#, dispositivo, ordenador, seguimiento.

Abstract

Nowadays, a considerable number of children have some problems when it comes to execute tasks that involves reading or writing.

As years pass by, the dimensions of data that people must process is becoming greater and greater, so it is essential to have a good domain of their language.

In order to achieve this, besides of practicing in their school and having some support from professionals such as psychologists or speech therapists, it is necessary that children train actively at home.

The methodologies that parents can currently apply are limited, the most popular being the following ones: exercises printed in paper, which can be downloaded on the Internet, and also the use of applications that are ran under an Android system.

There are not many solutions that can be ran directly from a computer, and those that exist are poor in content or do not provide adequate statistics to analyse user responses and therefore understand what needs to be improved.

This project is dedicated to the creation of an application that endorse children who are between 5 and 8 years old to perform exercises that allows then to improve those skills, emphasizing the tasks of symbol recognition and proper dissection of words.

For the development of this project, I have used Unity, a very popular videogame engine due to its power and variety of options. In addition, I also used Visual Studio, through which I coded various utility scripts in C#.

Keywords

Dyslexia, application, children, Unity, C#, device, computer, tracking.

Agradecimientos

Para empezar, me gustaría agradecerle a Carlos Aguirre el haber sido tan comprensivo en todo momento y haberme sabido guiar y orientar hasta el último momento.

Gracias también a mi familia, en especial a mis padres y a mi hermano, por haber estado a mi lado cuando más lo he necesitado. Sin vosotros no habría llegado hasta aquí.

También quiero agradecer a todos los amigos que siempre han estado ahí para apoyarme en momentos de estrés, principalmente a Adrián, David, Nacho y Kike. Vosotros creísteis en mí incluso cuando yo no lo hacía y habéis sido un pilar muy importante en mi vida este año.

Y por último, pero no menos importante, quiero agradecerle a mi otra mitad el tener siempre tiempo para compartir conmigo, independientemente de la cantidad de trabajo que tuviese que hacer. Gracias por enseñarme a volar y por sujetarme en mis innumerables caídas. No puedo esperar para saber qué nos deparará el futuro, pero sea lo que sea, lo afrontaremos siempre juntos Alberto, siempre.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

El objetivo de este Trabajo de Fin de Grado es el desarrollo de una aplicación para ordenadores que permita que niños cuya edad esté comprendida entre los 5 y los 8 años practiquen sus dificultades durante procesos de lectoescritura.

1.1 Motivación

A medida que pasan los años se detectan más y más personas con algún grado de dislexia.

Esto supone un problema muy grave porque vivimos en unos años en los que las personas que más avanzan académica y profesionalmente son aquellas que son capaces de procesar y entender grandes cantidades de información de la forma más breve y concisa posible.

En el momento en el que aparece algún síntoma propio de este trastorno, las capacidades de estas personas a la hora de realizar estas tareas de lectoescritura se ven muy reducidas.

Los niños tienen una capacidad de aprendizaje muy superior a la de los adultos, por lo que es imprescindible entrenar estas capacidades lo antes posible.

Existen numerosos ejercicios para practicar y corregir lo máximo posible (porque no es posible corregirlo del todo) este trastorno.

El que más se utiliza tanto por padres como por psicólogos y logopedas son ejercicios escritos en papel, de forma que el niño practique también la escritura de símbolos y letras.

Además, se ha popularizado mucho el uso de aplicaciones móviles, pero suelen estar enfocadas al uso tutelado por un adulto. Sin embargo, no existen muchos proyectos para ordenador que sean simples y que permitan un almacenamiento de los progresos de sus usuarios.

Estudios realizados recientemente demuestran que el uso de la tecnología aporta a los niños ciertas cualidades y beneficios que les es más complicado obtener de otra forma.

Principalmente, desarrollan un sentido de la independencia y una sólida mecánica de identificación y resolución de problemas complejos, además de comprender cómo separar dichos problemas en tareas más sencillas.

Seguidamente, la tecnología les permite desarrollar sus propios proyectos autónomos y profundizar en sus curiosidades e ideas de forma casi inmediata.

Podemos deducir por tanto que es importante también aprender a manejar correctamente un teclado y un ratón, motivo por el cual he decidido desarrollar este proyecto para que sea ejecutado en ordenadores.

1.2 Objetivos

El principal objetivo de este proyecto es desarrollar una aplicación con ejercicios para niños con dislexia que sea competente y eficaz para ser ejecutada en un ordenador.

Este proyecto ha de ser realizado mediante el motor de videojuegos Unity y el software de desarrollo Visual Studio, y todo el código generado ha de estar programado bajo el lenguaje C#.

Las diferentes metas que se quieren conseguir en la aplicación son las siguientes:

1. Generación y mantenimiento de perfiles de usuario identificativos.
2. Realización de ejercicios que entrenen la identificación de letras y símbolos.
3. Realización de ejercicios que entrenen la disección de palabras en sus correspondientes sílabas y la consecuente identificación del número de sílabas presentes.
4. Asociación de palabras con sus imágenes representativas.
5. Construcción de un sistema robusto de estadísticas que le permitan al usuario identificar qué tipo de ejercicio se le da peor (contabilizando el tiempo medio de realización y el número medio de errores).

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del arte**: sección en la que se hablará sobre todo el estudio previo a este proyecto que servirá como punto de origen para el desarrollo de la aplicación.
- **Diseño**: sección en la que se definirá el proyecto, su alcance, la metodología a aplicar y sus requisitos (tanto funcionales como no funcionales). Además, se incluyen varios diagramas orientativos y la descripción de las diferentes tecnologías y herramientas a utilizar, así como una breve explicación de la toma de decisiones a nivel de diseño.
- **Desarrollo**: sección en la que se explica la forma en la que se ha llevado a cabo el proyecto, incluyendo imágenes con el código más importante como apoyo.
- **Integración, pruebas y desarrollo**: sección en la que, para finalizar el cuerpo del documento, se incluyen las diferentes pruebas que se han realizado para comprobar que el proyecto funciona como se espera.
- **Conclusiones y trabajo futuro**: sección en la que se resume lo mencionado anteriormente, así como la finalidad del proyecto. Además, se incluyen comentarios acerca de cómo se podría mejorar el proyecto.
- **Referencias**: sección en la que se incluyen todas las referencias que han sido utilizadas a lo largo del desarrollo del proyecto.
- **Glosario**: sección que contiene las palabras clave del proyecto junto con una breve descripción.

2 Estado del arte

En este apartado se describe todo el estudio previo que ha tenido que realizarse para el correcto desarrollo del proyecto, así como el análisis de las diferentes aproximaciones digitales que existen en el mercado actualmente.

2.1 Entendiendo la dislexia

A continuación, procedemos a explicar qué es la dislexia y los datos más importantes que hay que tener en cuenta.

2.1.1 Definición

La dislexia (del latín científico dyslexia y del griego "dis-" (dificultad, incapacidad) y -lexia (habla, dicción)) es una condición que padece alrededor de un 20% de la población y que dificulta o imposibilita la capacidad de lectura o escritura.

Por norma general, las personas con dislexia suelen tener dificultad en algunos de los siguientes ámbitos:

- Comprensión lectora.
- Ortografía.
- Identificación de símbolos y patrones.
- Cálculo y álgebra.
- Orientación espacial.

Estudios actuales han probado que la dislexia tiene un origen neurobiológico, es decir, que durante el desarrollo embrionario cierto número de neuronas no se establecieron correctamente formando así unos cúmulos llamados ectopias, responsables de las interferencias durante las actividades de lectura y escritura.

En torno a un 60% de estas anomalías fueron generadas debido a un factor genético, siendo el desencadenante una posible mutación en el cromosoma número 15 (responsable de la formación de las anteriormente mencionadas ectopias).

Dependiendo de la edad del sujeto con dislexia, este puede presentar unos síntomas u otros, siendo los más comunes la dificultad de aprender, reconocer y reproducir patrones, problemas de conducta causados por la frustración de este y la aparición de una excesiva fatiga y cansancio al realizar procesos de lectoescritura.

2.1.2 Tipos de dislexia

Existen muchos tipos de dislexia, pero los más comunes son los siguientes:

2.1.2.1 Dislexia fonológica o indirecta

Las personas con dislexia fonológica presentan dificultades separando las palabras en sílabas y durante la lectura procesan las palabras enteras.

Esta variante permite una lectura mucho más rápida, pero imprecisa, ya que provoca dificultades al interpretar palabras desconocidas y hay más posibilidades de realizar errores de derivación, es decir, de realizar inferencias incorrectas a partir de indicios.

2.1.2.2 Dislexia visual o superficial

Las personas con dislexia visual presentan dificultades al reconocer palabras y no son capaces de inferirlas a partir de un contexto.

Este tipo de dislexia provoca una lectura extremadamente lenta ya que se necesita un análisis sílaba a sílaba de la palabra que se está leyendo, aumentando así los errores visuales como las rotaciones de las letras.

Además, estas personas no son capaces de identificar con rapidez palabras homófonas, es decir, palabras que suenan igual pero que se escriben diferente.

2.1.2.3 Dislexia mixta

Las personas con dislexia mixta pueden presentar síntomas de los otros tipos mencionados anteriormente.

2.1.3 Cómo se diagnostica

Como todo trastorno, la mejor forma de diagnosticarlo es mediante la realización de una serie de pruebas y tests.

Estas pruebas han de ser realizadas por profesionales (como psicólogos o neuropsicólogos) los cuales han de asegurarse de que no haya ningún otro problema clínico que lo esté provocando.

Estos tests pondrán en práctica la identificación de patrones y de símbolos, su comprensión y la velocidad en la que se están desarrollando estos procesos.

2.1.4 Posibles tratamientos

La dislexia no es un trastorno que se pueda curar, pero sí se puede entrenar y frenar.

Dependiendo del tipo y del grado de dislexia y de la edad del sujeto, se deberán de aplicar una metodología u otra.

A grandes rasgos, el tratamiento más eficaz es la práctica, es decir, realizar ejercicios relacionados con el susodicho tipo de dislexia hasta que el cerebro se acostumbre a identificar ciertos patrones o a contextualizar más fácilmente.

En el caso de la dislexia fonológica se debe practicar el reconocimiento de palabras mediante imágenes y la contextualización de dichas palabras, y en el caso de la dislexia visual, el orden de las letras y su orientación.

La efectividad de estas medidas depende de muchos factores, siendo los más importantes la profundidad del trastorno, las ganas del individuo de enfrentarse a dicho trastorno y al apoyo de su entorno tanto familiar como escolar.

2.2 Aproximaciones actuales

Para poder saber cómo afrontar el problema que se discute en el actual documento es importante conocer cuáles son las aproximaciones actuales y desarrollar una forma de adaptarlas y mejorarlas para que sean más efectivas.

Se ha realizado un análisis exhaustivo de varias aplicaciones que existen en el mercado que tienen objetivos similares al desarrollado en este proyecto, pero debido a la longitud de este documento sólo he recogido por escrito los datos más relevantes de dos de ellas.

Se describe su apariencia estética, la metodología que se ha decidido aplicar, para qué tipos de dislexia sirven, y las limitaciones y posibles mejoras observadas.

Se han tomado diversas capturas que muestran gráficamente dicha apariencia, y se pueden observar consultando las figuras que se encuentran en los anexos.

2.2.1 Dytective

2.2.1.1 Análisis

Lo primero que se observa nada más ejecutar la aplicación es la presencia de un menú de inicio de sesión, el cual contiene dos campos para el email o nombre del tutor y la contraseña. También se observa un botón para iniciar dicha sesión y otro para crear una cuenta en caso de que el usuario sea nuevo.

En caso de que se decida crear una nueva cuenta, la aplicación te deja introducir tu nombre y apellidos, un correo electrónico y una doble verificación de contraseña. A continuación, se permite elegir el tipo de usuario (familia, terapeuta, colegio), y se deben aceptar una serie de requisitos como las condiciones de servicio y el envío de newsletters.

En caso de que se decida iniciar sesión aparecerá un panel con diversas opciones, siendo los más relevantes los siguientes: creación de un usuario, realización de tests y modo jugar.

La opción de test va enfocada a la detección de alguna variante de dislexia. En caso de que se decida realizar un test, aparece un mensaje explicando que el diagnóstico ha de ser realizado por una persona cualificada y que sólo determina el posible riesgo de parecer alguno de estos trastornos.

Una vez que comienza el test, irán apareciendo diversas letras y palabras y el usuario deberá identificarlas dependiendo del sonido que se reproduzca justo antes del ejercicio.

A partir de los resultados obtenidos se establece la posibilidad de que el sujeto tenga dislexia.

En caso de que el usuario decida jugar, se mostrará una pantalla en la que se avisa al usuario de que tendrá 7 días de prueba en la que todos los usuarios podrán acceder a esta funcionalidad, ya que es originalmente de pago.

Una vez se acepta ese modal aparecerá un menú principal en la que podemos configurar el volumen, comprar la versión completa o comenzar el juego.

El juego está enfocado en la realización de submisiones que se van desbloqueando con el progreso, mecánica muy común presente en otros juegos.

El objetivo de estas submisiones va variando su dificultad según el progreso del individuo, pero generalmente consiste en localizar la letra que sobra, de ordenar letras correctamente o de identificar palabras con su representación gráfica.

2.2.1.2 Ideas extraídas o adaptaciones

Esta aplicación resume muy bien las ideas fundamentales que va a tener este proyecto.

La estética es muy atrayente a niños y realmente da la impresión de que estás jugando a un juego, lo cual facilita que los usuarios no se cansen usándola y sigan practicando y corrigiendo sus errores.

También me ha gustado el hecho de que la propia aplicación permita que la usen varios usuarios y mantenga el estado de sus progresos, pudiendo consultarlos en cualquier momento.

Los ejercicios eran acordes a lo que se prometía y su dificultad iba aumentando a medida que el usuario iba progresando.

También es importante mencionar el hecho de que haya opciones de configuración de forma que la herramienta se adapte a los gustos de los usuarios.

En resumen, de esta aplicación he obtenido ideas de cómo realizar una aplicación que sea atrayente a niños y de cómo mantener varias sesiones en una misma cuenta.

2.2.1.3 Limitaciones

Uno de los detalles que se van a eliminar es el hecho de que se muestren tantos mensajes informativos. En mi opinión, alejan al usuario de esa visión de videojuego y complican las cosas en exceso.

Otro factor para modificar es el que la aplicación sea de pago. Mi visión de este proyecto es que sea una herramienta realmente accesible para todo el mundo, por lo que no veo necesario añadir funcionalidad de pago a la misma.

Además, la aplicación no tiene en cuenta el grado, tipo o profundidad de dislexia a la hora de jugar, por lo que un usuario con un nivel bajo de dislexia se va a ver forzado al principio de realizar ejercicios muy simples para él.

También me gustaría que la aplicación tuviese ejercicios más variados y que el usuario sea el que elija qué tipo de ejercicio desea realizar, ya que al desbloquearse los niveles de forma secuencial estás forzado a realizar un tipo de ejercicio para poder desbloquear el siguiente.

Tampoco voy a implementar la funcionalidad de test ya que considero que la dislexia es un trastorno muy complejo y variable entre personas, por lo que veo necesario que sea un profesional el que lo diagnostique. Se añadirá un panel de información al comienzo de la aplicación que avise al usuario que, si sospecha que puede padecer de dislexia, que realice alguna prueba proporcionada por algún profesional para confirmarlo.

2.2.2 Sanapalabras

2.2.2.1 Análisis

Lo primero que se ve nada más iniciar la aplicación es un menú principal en el que se incluyen varias opciones seleccionables correspondientes a las diferentes modalidades de juego que contiene.

La primera modalidad, llamada “Silabeatum”, consiste en seleccionar las palabras que tienen el número de sílabas que se indica al comienzo. Se puede seleccionar el nivel de dificultad, que incluye un modo fácil y uno difícil.

Una vez comienza el minijuego, has de controlar a la mascota de la aplicación para que vuele y recoja las palabras que contienen dicho número de sílabas.

La forma que tiene el usuario de hacer a la mascota volar es haciendo clic en alguno de los círculos rojos que se incluyen.

Además, se incluye una barra de vida y otra de progreso, de forma que el usuario vea cuántas respuestas ha tenido bien y cuántas mal (de forma que si supera el límite de alguna de ellas se acaba el minijuego).

No hay límite de tiempo por lo que el individuo puede tardar el tiempo que necesite.

El siguiente modo de juego se llama “Consonitus”, y consiste en atrapar las palabras que tienen el sonido que se indica al comienzo. También se puede seleccionar el nivel de dificultad.

De igual forma que en el anterior minijuego, el usuario ha de hacer volar a la mascota presionando los botones rojos de forma que, si se acerca a la palabra, esta cuenta como recogida.

También se incluye una barra de vida y otra de progreso, y tampoco hay límite de tiempo.

La siguiente modalidad de juego se llama “Consonantis”, y consiste en completar la palabra que se muestra con una de las sílabas disponibles. También se puede elegir el grado de dificultad.

Si el usuario se equivoca dos veces seguidas de sílaba, la mascota llorará y perderá puntos de vida, si por contrario responde correctamente, derrotará a un animal que custodia la palabra y se sumarán puntos de victoria.

Tampoco hay tiempo límite para realizar este ejercicio.

La última modalidad se llama “Diglexiatis”, y consiste en intercambiar letras para formar una palabra correctamente. De igual forma que en las anteriores, se puede escoger la dificultad del minijuego.

El usuario ha de seleccionar una de las letras de la palabra original, e intercambiarla por otra de las que se incluyen más abajo.

Si se equivoca la mascota llorará y si acierta se sumarán puntos de victoria.

Tampoco hay límite de tiempo.

También se incluye una página en la que se pueden visualizar los diferentes puntajes obtenidos en los minijuegos mencionados anteriormente.

Además, se incluyen premios que se van obteniendo a medida que se baten récords en cada uno de los distintos ejercicios, lo cual ayuda a los usuarios a sentirse más gratificados intentando superarse a sí mismos.

2.2.2.2 Ideas extraídas o adaptaciones

Me ha encantado la simplicidad de la aplicación. Es muy fácil de usar y cualquier persona sin conocimientos de tecnología o informática sería capaz de usarla.

Es interesante que se haya incluido una mascota animada para acompañar al usuario durante sus ejercicios, ya que así se siente más cómodo.

La variedad de ejercicios es clave, ya que, a diferencia de la aplicación anterior, se puede escoger el ejercicio que se quiera practicar y su dificultad, dándole más libertad al individuo.

Además, dichos ejercicios sirven para los dos tipos de dislexia más comunes (dislexia fonológica y visual), por lo que es más amplia y completa.

2.2.2.3 Limitaciones

La estética de la aplicación es algo torpe y poco cuidada. Me gustaría adaptarla para que fuese más minimalista, clara y profesional.

Los nombres de los ejercicios, a pesar de que son originales, son muy difíciles de pronunciar y de entender.

No se te explica muy bien cómo interactuar con la interfaz de los minijuegos, y especialmente en los dos primeros (en los cuales hay que hacer a la mascota volar) me costó adivinar cómo controlar al personaje y cómo recoger las palabras.

Sólo se permite el uso de la aplicación para un sólo usuario, por lo que las estadísticas de los minijuegos son compartidas entre todos los individuos que utilicen la aplicación.

3 Diseño

En este apartado se definirá el alcance del proyecto, así como las metodologías y herramientas que han sido necesarias durante su definición.

Se detallarán además la estructura de la aplicación, los requisitos funcionales y los no funcionales.

3.1 Introducción y definición del proyecto

La aplicación ha de servir como punto de apoyo para niños cuya edad esté entre los 5 y los 8 años que necesiten realizar ejercicios de apoyo para entrenar su dislexia.

He creído conveniente que el proyecto resulte intuitivo a usuarios de tan baja edad, de forma que quieran usarla en sus casas y su experiencia sea entretenida, divertida y educativa.

Para ello, las vistas de la aplicación son sencillas y coloridas. Todas las acciones vienen explicadas para que sean altamente entendibles y el diseño visual contiene muchas imágenes ordenadas y colocadas de forma que resulten atractivas visualmente y relajen al usuario.

3.1.1 Alcance

La funcionalidad presente en la aplicación ha sido decidida mediante la cooperación de una profesional en el ámbito de la psicología, ajustándose a las necesidades presentes en la tecnología actual y definidas previamente en el apartado del estudio del arte.

El proyecto puede dividirse en dos bloques significativos, la organización de perfiles de usuarios y las actividades educativas que dichos usuarios pueden realizar.

Los usuarios han de poder tener un perfil de usuario identificable a través de un nombre y de una representación visual mediante un avatar.

Dicho perfil ha de almacenar el número de monedas obtenidas, así como todos los complementos para el avatar adquiridos.

Todas las estadísticas generadas al interactuar con las actividades educativas han de ser también almacenadas y tienen que poder consultarse en cualquier momento.

Los perfiles han de poder crearse y eliminarse siempre y cuando se respete el límite de perfiles disponibles en esta aplicación (3 para ser más concretos).

En relación con el segundo bloque, los usuarios han de poder jugar a minijuegos educativos, mediante los cuales entrenar sus dificultades presentes durante procesos de lectoescritura.

Dichos minijuegos han de reforzar tanto la distinción e identificación de símbolos y letras como la comprensión de cómo se separan las palabras en sílabas, además de cómo reorganizar dichas sílabas para formar palabras correctas.

Durante dichos minijuegos ha de contabilizarse el tiempo que está tardando el usuario en completar el ejercicio como el número de aciertos y errores que se estén cometiendo.

Cada vez que un usuario finalice un minijuego, ya sea satisfactoriamente como todo lo contrario, la aplicación debe guardar los datos mencionados con anterioridad para que posteriormente se pueda realizar un estudio estadístico.

En caso de que el usuario complete la actividad con el número total de aciertos necesarios, se añadirán 5 monedas a su inventario, las cuales podrá gastar en accesorios para su avatar.

3.1.2 Metodología empleada

La metodología que más se ajustaba a las necesidades de este proyecto es la basada en el desarrollo ágil e incremental.

Lo más eficiente en este caso ha sido dividir el proyecto en tareas más pequeñas, y sobre esas tareas aplicar un ciclo de vida propio, de forma que la aplicación fuese altamente modificable y ajustable a posibles cambios.

Los prototipos generados al final de cada iteración eran puestos a prueba y ajustados hasta que cumpliesen con los requerimientos definidos en el alcance de la aplicación.

3.1.3 Definición de requisitos

A partir de la funcionalidad especificada en el alcance de la aplicación, podemos extraer los siguientes requisitos.

3.1.3.1 Requisitos funcionales

RF 1. El usuario ha de poder crear un perfil identificativo mediante los siguientes datos:

- Nombre.
- Avatar:
    - Cuerpo con diferentes tonalidades de piel.
    - Cara con distintas expresiones faciales.
    - Peinado con colores y formas dispares.
    - Atuendos de diferentes estilos.

RF 2. El usuario ha de poder eliminar su perfil en cualquier momento.

RF 3. El usuario podrá seleccionar su perfil para acceder a las funcionalidades restantes en caso de que se haya creado uno.

RF 4. Se podrá modificar el aspecto visual del usuario mediante los diferentes complementos que vaya adquiriendo.

RF 5. El usuario podrá acceder a una estadística completa que represente su desempeño en las diferentes actividades de la aplicación. La estadística tendrá en cuenta los siguientes datos:

- Número de veces que se ha jugado a un minijuego.
- Tiempo promedio en terminar la actividad.
- Número de errores cometidos en dicha actividad.

RF 6. Un minijuego se considerará terminado en caso de que se cumplan alguno de los siguientes casos:

- Se han realizado 5 aciertos, en cuyo caso la actividad finalizará satisfactoriamente.
- Se han cometido 5 fallos, en cuyo caso la actividad finalizará erróneamente.

RF 7. En caso de que un minijuego finalice satisfactoriamente, se incrementarán el número de monedas del usuario en 5.

RF 8. Se creará una tienda en la que el usuario pueda gastar las monedas obtenidas en complementos para su avatar. Dichos complementos serán los siguientes:

- Peinados.
- Atuendos.

RF 9. Se podrá modificar el aspecto del avatar mediante una vista en la que seleccionar los peinados y atuendos obtenidos.

RF 10. Mediante el primer minijuego se practicará la identificación del número de sílabas que definen una palabra.

> **RF 10.1**. Se escogerá aleatoriamente el número de sílabas (desde palabras monosílabas hasta polisílabas).
>
> **RF 10.2**. Se generarán aleatoriamente palabras desde la zona inferior de la pantalla, y dichas palabras irán ascendiendo hasta llegar al límite superior y desaparecer.
>
> **RF 10.3**. Dichas palabras serán seleccionables de forma que se produzca alguno de los siguientes resultados:
>
> - Si la palabra tiene el número de sílabas especificado, se incrementará en uno el número de aciertos y la palabra se coloreará de verde.
> - Si la palabra no tiene el número de sílabas especificado, se incrementará en uno el número de fallos y la palabra se coloreará de rojo.
>
> **RF 10.4**. Un timer contabilizará el tiempo que está tardando el usuario en finalizar la actividad.
>
> **RF 10.5**. Si el usuario llega al máximo de aciertos permitidos, se mostrará una vista con un mensaje positivo y un botón que permita el retroceso al menú de juegos.
>
> **RF 10.6**. Si el usuario llega al máximo de errores permitidos, se mostrará una vista con un mensaje que le anime a realizar otro intento y un botón que permita el retroceso al menú de juegos.
>
> **RF 10.7**. La vista del minijuego deberá incluir un botón de retroceso que le permita retroceder al menú de juegos.
>
> **RF 10.8** Al comienzo, se deberá mostrar una explicación del minijuego, así como dos botones que le permitan al usuario retroceder al menú de juegos y avanzar al propio minijuego.

RF 11. Mediante el segundo minijuego se practicará la identificación de sílabas, así como de palabras que contengan dicha sílaba.

> **RF 11.1**. Se escogerá aleatoriamente la sílaba a buscar.
>
> **RF 11.2**. Se generarán aleatoriamente palabras desde la zona inferior de la pantalla, y dichas palabras irán ascendiendo hasta llegar al límite superior y desaparecer.

RF 11.3. Dichas palabras serán seleccionables de forma que se produzca alguno de los siguientes resultados:

- Si la palabra contiene la sílaba especificada, se incrementará en uno el número de aciertos y la palabra se coloreará de verde.
- Si la palabra no contiene la sílaba especificada, se incrementará en uno el número de fallos y la palabra se coloreará de rojo.

RF 11.4. Un timer contabilizará el tiempo que está tardando el usuario en finalizar la actividad.

RF 11.5. Si el usuario llega al máximo de aciertos permitidos, se mostrará una vista con un mensaje positivo y un botón que permita el retroceso al menú de juegos.

RF 11.6. Si el usuario llega al máximo de errores permitidos, se mostrará una vista con un mensaje que le anime a realizar otro intento y un botón que permita el retroceso al menú de juegos.

RF 11.7. La vista del minijuego deberá incluir un botón de retroceso que le permita retroceder al menú de juegos.

RF 11.8 Al comienzo, se deberá mostrar una explicación del minijuego, así como dos botones que le permitan al usuario retroceder al menú de juegos y avanzar al propio minijuego.

RF 12. Mediante el tercer minijuego se practicará la ordenación de sílabas para formar la palabra correspondiente a la imagen mostrada.

RF 12.1. Se escogerá aleatoriamente la palabra a formar.

RF 12.2. Se distribuirán las sílabas en un panel de 3x3 aleatoriamente, de forma que las sílabas que conforman la palabra estén mezcladas con otras similares.

RF 12.3. Dichas sílabas serán seleccionables de tal manera que se vayan concatenando, formando una palabra.

RF 12.4. Dicha concatenación será visible de forma inmediata a medida que se vayan seleccionando sílabas.

RF 12.5. Se mostrará el número de sílabas que contiene la palabra a formar como pista para el usuario.

RF 12.6. En caso de que el usuario haya seleccionado el mismo número de sílabas que las que conforman la palabra, se analizará la concatenación produciendo alguno de los siguientes resultados:

- En caso de que la concatenación sea correcta, se incrementará en uno el contador de palabras correctas, se coloreará la concatenación en verde y se mostrará un botón que permita generar la siguiente palabra.
- En caso de que la concatenación sea incorrecta, se incrementará en uno el contador de palabras erróneas, se coloreará la concatenación en rojo y se mostrará un botón que permita generar la siguiente palabra.

RF 12.7. Un timer contabilizará el tiempo que está tardando el usuario en finalizar la actividad.

RF 12.8. Si el usuario llega al máximo de aciertos permitidos, se mostrará una vista con un mensaje positivo y un botón que permita el retroceso al menú de juegos.

RF 12.9. Si el usuario llega al máximo de errores permitidos, se mostrará una vista con un mensaje que le anime a realizar otro intento y un botón que permita el retroceso al menú de juegos.

RF 12.10. La vista del minijuego deberá incluir un botón de retroceso que le permita retroceder al menú de juegos.

RF 12.11. Al comienzo, se deberá mostrar una explicación del minijuego, así como dos botones que le permitan al usuario retroceder al menú de juegos y avanzar al propio minijuego.

RF 13. Mediante el cuarto minijuego se practicará la selección de la sílaba correcta para completar la palabra mostrada.

RF 13.1. Se escogerá aleatoriamente la palabra a formar y se mostrará en un panel situado a la izquierda de la vista.

RF 13.2. Se distribuirán las sílabas en un panel de 1x4 aleatoriamente, de forma que la sílaba que completa la palabra esté mezclada con otras similares.

RF 13.3. Dichas palabras serán seleccionables de forma que se produzca alguno de los siguientes resultados:

- Si la palabra contiene la sílaba especificada, se incrementará en uno el número de aciertos, la sílaba se coloreará de verde y se mostrará un botón que permita generar otra palabra.
- Si la palabra no contiene la sílaba especificada, se incrementará en uno el número de errores, la sílaba se coloreará de rojo y se mostrará un botón que permita generar otra palabra.

RF 13.4. Un timer contabilizará el tiempo que está tardando el usuario en finalizar la actividad.

RF 13.5. Si el usuario llega al máximo de aciertos permitidos, se mostrará una vista con un mensaje positivo y un botón que permita el retroceso al menú de juegos.

RF 13.6. Si el usuario llega al máximo de errores permitidos, se mostrará una vista con un mensaje que le anime a realizar otro intento y un botón que permita el retroceso al menú de juegos.

RF 13.7. La vista del minijuego deberá incluir un botón de retroceso que le permita retroceder al menú de juegos.

RF 13.8. Al comienzo, se deberá mostrar una explicación del minijuego, así como dos botones que le permitan al usuario retroceder al menú de juegos y avanzar al propio minijuego.

3.1.3.2 Requisitos no funcionales

RNF 1. La aplicación ha de desarrollarse modularmente de forma que su ampliación sea sencilla.

RNF 2. Se ha de garantizar robustez de forma que se garantice un correcto funcionamiento a lo largo de todas las actividades de la aplicación.

RNF 3. El paso de una vista a otra no debe de durar más de 2 segundos.

RNF 4. Todos los elementos visuales han de ajustarse correctamente a los distintos tamaños y dimensiones de los dispositivos en los que va a ser ejecutada esta aplicación.

RNF 5. La aplicación ha de ser compatible con dispositivos que no tengan acceso a internet.

RNF 6. La tipografía y elementos visuales escogidos han de ser adecuados a niños cuya edad esté comprendida entre los 5 y los 8 años.

RNF 7. El proceso de creación de avatares ha de ser lo más genero neutro posible, de forma que los niños puedan expresarse libremente. Además, se deberá tener una correcta representación de las diferentes tonalidades de piel que puedan tener dichos usuarios.

3.1.4 Diagramas de flujo

En el siguiente apartado se muestran los diferentes diagramas de uso que serán útiles para definir el comportamiento de diferentes elementos en la aplicación.

3.1.4.1 Diagrama de flujo correspondiente a la vida de la aplicación

Este diagrama de flujo debe mostrar las diferentes acciones que se pueden tomar en la aplicación, así como las vistas que se muestran al ejecutar dichas acciones.

Se encuentra en el Anexo B junto con los demás diagramas (Figura 3-1).

3.1.4.2 Diagrama de flujo correspondiente a la primera actividad

Este diagrama de flujo debe mostrar el comportamiento que ha de tener el primer minijuego.

Se encuentra en el Anexo B junto con los demás diagramas (Figura 3-2).

3.1.4.3 Diagrama de flujo correspondiente a la segunda actividad

Este diagrama de flujo debe mostrar el comportamiento que ha de tener el segundo minijuego.

Se encuentra en el Anexo B junto con los demás diagramas (Figura 3-3).

3.1.4.4 Diagrama de flujo correspondiente a la tercera actividad

Este diagrama de flujo debe mostrar el comportamiento que ha de tener el tercer minijuego.

Se encuentra en el Anexo B junto con los demás diagramas (Figura 3-4).

3.1.4.5 Diagrama de flujo correspondiente a la cuarta actividad

Este diagrama de flujo debe mostrar el comportamiento que ha de tener el cuarto minijuego.

Se encuentra en el Anexo B junto con los demás diagramas (Figura 3-5).

3.1.5 Herramientas y tecnologías

En este apartado se van a explicar las diferentes tecnologías y herramientas que han sido necesarias para el correcto desarrollo de este proyecto.

3.1.5.1 Herramientas

3.1.5.1.1 Zotero

Zotero es un gestor de referencias de código libre y gratuito que permite recopilar, organizar, filtrar y citar información recogida de Internet.

Tiene soporte para diferentes formatos (páginas web, vídeos, PDF, artículos periodísticos o de investigación, etc.) y se puede sincronizar entre distintos dispositivos.

3.1.5.1.2 Unity 2018.2.10

Unity es un motor de videojuegos multiplataforma muy popular entre desarrolladores debido a su potencia y su interfaz intuitiva.

El principal motivo por el que elegí este motor es la experiencia, ya que en ocasiones anteriores he desarrollado proyectos usando Unity y tenía un alto conocimiento previo.

Además, al ser multiplataforma, la publicación del proyecto en varios dispositivos se convierte en una realidad.

3.1.5.1.3 Visual Studio 2017

Microsoft Visual Studio es un entorno de desarrollo que permite el desarrollo de código en diferentes lenguajes, entre ellos C#.

Todos los scripts necesarios para este proyecto han sido codificados mediante este entorno.

He elegido este programa debido a que, al igual que con Unity, tenía mucha práctica debido a proyectos anteriores.

3.1.5.1.4 GIMP 2.10.8

GIMP es un programa de edición de imágenes gratuito y de código libre.

He hecho uso de este programa para modificar los diferentes sprites de la aplicación para que se ajustasen correctamente a las necesidades del mismo.

He preferido GIMP sobre cualquier otro programa de edición fotográfica debido a su simplicidad y a su interfaz intuitiva, lo cual me ha permitido ganar rapidez durante los procesos de edición de sprites.

3.1.5.1.5 Cacoo

Cacoo es un servicio web que permite la creación de diferentes tipos de diagramas y esquemáticos.

No toda la funcionalidad disponible es gratuita, pero he decidido usar este servicio debido a la posible sincronización con Google Drive.

3.1.5.1.6 Google Drive

Google Drive es un servicio web de almacenamiento de archivos más utilizado de todos los existentes.

He necesitado utilizar este servicio a la hora de guardar versiones del proyecto y de sus diferentes archivos multimedia.

3.1.5.2 Tecnologías

Todo el código presente en la aplicación ha sido programado mediante el lenguaje C#. Se ha escogido este lenguaje ya que es el utilizado por Unity (esto es así porque es el lenguaje orientado a objetos más fácil de aprender para desarrolladores de videojuegos y es potente y flexible).

La estructuración de los datos de los usuarios y los catálogos se ha realizado mediante el formato JSON, debido a que C# contiene funcionalidades muy completas que dan soporte a dicho formato. Además, esta generación de colecciones de datos permite una lectura y escritura muy rápida y eficiente.

3.2 Diseño de la aplicación

En este apartado se profundizará en las diferentes tomas de decisiones en referencia al diseño del proyecto, en concreto su interfaz gráfica y las diferentes clases existentes.

En todo momento se ha tenido en cuenta que esta aplicación estaba destinada a niños pequeños, por lo que se ha priorizado la sencillez y la facilidad de uso, así como la inclusión de imágenes llamativas y mecánicas que consigan que el usuario quiera seguir haciendo uso de dicha aplicación.

3.2.1 Interfaz gráfica

A la hora de tomar decisiones en cuanto a la interfaz gráfica del proyecto, he tenido muy en cuenta dos artículos publicados respectivamente por la Universitat Oberta de Catalunya, y por Yussef Hassan Montero (las referencias concretas a ambos artículos se encuentran recogidas en el apartado Referencias de este mismo documento).

La información más importante que he podido extraer es la siguiente:

- **Instrucciones**: Los niños no suelen leer instrucciones ni manuales de uso, por lo que toda acción ha de ser lo más intuitiva posible. En caso de que se necesite el uso de alguna instrucción, esta ha de ser lo más breve posible.

- **Distracciones**: Los niños suelen distraerse bajo situaciones desconocidas o que requieren cierto nivel de concentración, por lo que la aplicación ha de ser llamativa y que constantemente despierte su atención.
- **Motivación**: Es altamente necesario que la aplicación les proporcione un alto grado de motivación para que no se cansen o se desalienten.
- **Colores**: Los diseños han de ser ricos en colores llamativos.
- **Implicación emocional**: Hay que intentar animar a los niños cuando no realizan una tarea correctamente de forma que no se desmotiven, consiguiendo por el contrario que quieran seguir probando y aprendiendo. También hay que reforzar positivamente cuando el ejercicio ha sido finalizado satisfactoriamente.
- **Animaciones**: Los niños disfrutan descubriendo diferentes animaciones en pantalla, pero hay que tener cuidado para no desorientarles o distraerles.
- **Lenguaje**: El lenguaje debe estar adecuando a la edad de sus usuarios, de forma que no sea muy complejo.

3.2.2 Arquitectura y diagrama de clases

En este apartado se describirán la arquitectura de la aplicación, así como la representación de las diferentes clases mediante un diagrama.

3.2.2.1 Arquitectura lógica

La arquitectura lógica del proyecto está basada en la programación por capas, la cual se basa en el desacoplamiento de las distintas partes que forman dicho proyecto.

Las capas que componen la arquitectura lógica de la aplicación son las siguientes:

1. **Capa de usuario**: responsable de mostrar la información disponible al usuario y recopilar las respuestas del usuario, las cuales serán enviadas a la capa de negocio.
2. **Capa de negocio**: es donde se reúnen las diferentes reglas y normas que se deben cumplir. Esta capa solicita información a la capa de datos en forma de respuesta a las solicitudes recibidas de la capa de usuario.
3. **Capa de datos**: responsable del almacenamiento de todos los datos y de su consecuente acceso.

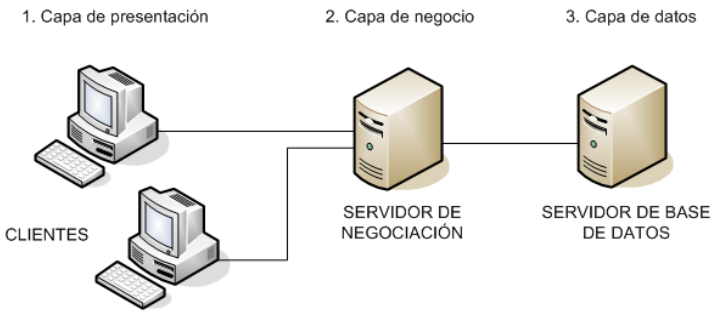

Figura 3 - 1: Arquitectura lógica por capas. De https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas#/media/File:Tres_capas.PNG

En este caso, todas estas capas residen en un mismo dispositivo, es decir, el ordenador en el que el usuario esté ejecutando la aplicación.

3.2.2.2 Diagrama de clases

En la Figura 3-7 almacenada en el Anexo B se muestra el diagrama de clases que representa el modelo presente en esta aplicación:

A continuación, se explicará la funcionalidad de cada clase:

- **Game**: Clase abstracta que representa la instancia de un minijuego o actividad producida cuando un usuario ha terminado dicho minijuego. He decidido que sea abstracta y que el resto hereden de ella para facilitar la posterior creación de cada uno de ellos haciendo que el código sea mucho más reducido. Aunque en la actualidad los cuatro minijuegos presentes en la aplicación tienen una estructura idéntica, es preferible que cada uno tenga su propia estructura para futuras modificaciones o alteraciones.
- **Game1**: Clase que representa el primer minijuego. Como campos, contiene el número de errores que se han cometido y el tiempo en el que el usuario ha finalizado la actividad.
- **Game2**: Clase que representa el segundo minijuego. Como campos, contiene el número de errores que se han cometido y el tiempo en el que el usuario ha finalizado la actividad.
- **Game3**: Clase que representa el tercer minijuego. Como campos, contiene el número de errores que se han cometido y el tiempo en el que el usuario ha finalizado la actividad.
- **Game4**: Clase que representa el cuarto minijuego. Como campos, contiene el número de errores que se han cometido y el tiempo en el que el usuario ha finalizado la actividad.
- **Word**: Clase que representa las diferentes instancias de palabras que se generan aleatoriamente en el primer y segundo minijuego. Como campos, contiene un flag que permite identificar si la palabra ha sido seleccionada por el usuario o no.
- **TotalGames**: Clase que representa el desempeño del usuario en los diferentes minijuegos de la aplicación. Esto se consigue almacenando todas las instancias de dichos juegos para su posterior análisis. Como campos, contiene cuatro listas con las distintas clases que representan dichas actividades.
- **Avatar**: Clase que representa la representación visual del usuario. La clase Avatar contiene los siguientes campos:
    - **Body**: String que almacena el nombre del sprite que representa el cuerpo del avatar.
    - **Face**: String que almacena el nombre del sprite que representa la cara del avatar.
    - **Hair**: String que almacena el nombre del sprite que representa el peinado del avatar.
    - **Kit**: String que almacena el nombre del sprite que representa el atuendo del avatar.

- **Element**: Clase que representa cada uno de los elementos disponibles para comprar en la tienda, así como los elementos ya comprados y disponibles en el armario del usuario. Como campos, contiene el nombre del sprite y el precio correspondiente.
- **Catalogue**: Clase que representa la lista de productos de la tienda y los obtenidos por el usuario. Como campos, contiene una lista de peinados y de atuendos (representados mediante la clase Element).
- **Profile**: Clase que representa cada una de las instancias relacionadas con los usuarios de la aplicación. Como campos, contiene el nombre y avatar del usuario, todas las instancias de las actividades a las que ha jugado, el número de monedas que tiene en su inventario, y el catálogo de productos que tiene disponible en su armario.

4 Desarrollo

En este apartado se entrará en detalle acerca de los aspectos fundamentales del proyecto, por lo que es el apartado más importante y costoso de producir.

Una vez que se ha hecho el análisis y diseño previo, hay que transformar toda esa información en código funcional y en escenas de Unity.

No me ha resultado muy complicado esta implementación debido a la enorme práctica que ya tenía con todas las herramientas y tecnologías necesarias y ya mencionadas anteriormente, pero ha sido un reto realizarla de forma que sea útil y práctica para niños pequeños.

A continuación, se explicará la estructura de la aplicación y las diferentes vistas o escenas que la componen.

4.1 Estructura de la aplicación

Los diferentes ficheros que son necesarios para el correcto almacenamiento de la aplicación se almacenan de forma estructurada en carpetas.

La siguiente imagen muestra las diferentes carpetas que han sido necesarias y su funcionalidad:

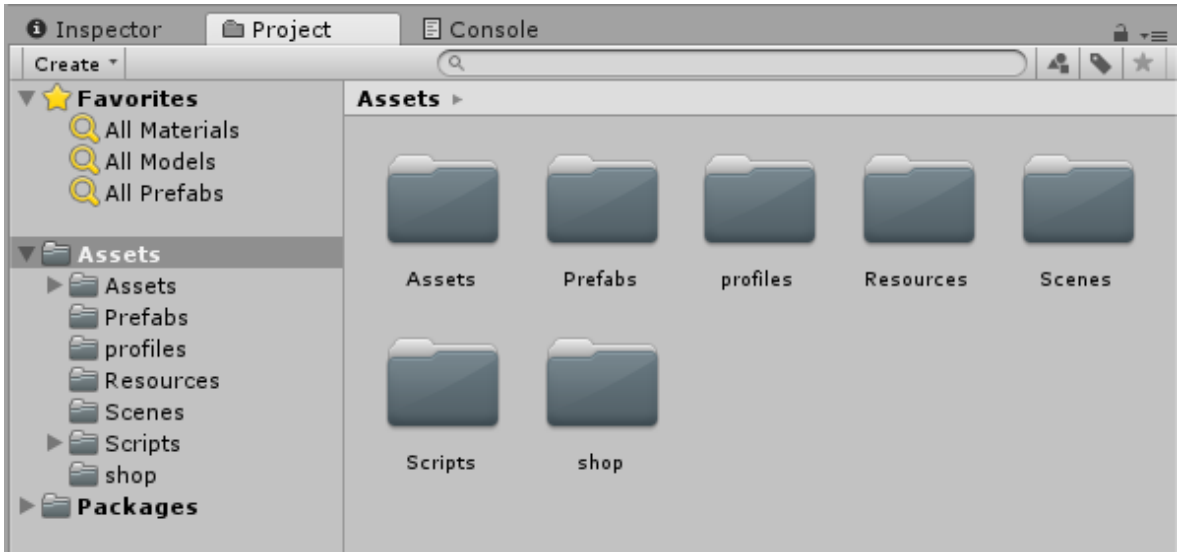

Figura 4 - 1: Distribución carpetas del proyecto

Las carpetas que podemos observar son las siguientes:

- **Assets**: En esta carpeta se guardan los diferentes sprites y fuentes de texto que se han necesitado. Todo ha sido extraído de páginas web que ofrecen recursos gratuitos y libres de licencia.
- **Prefabs**: En esta carpeta se guardan los prefabs que he tenido que crear, es decir, elementos que van a ser comunes en varias escenas y por tanto se pueden guardar para su posterior reproducción.

- **Profiles**: En esta carpeta se guardarán los ficheros relacionados con los perfiles de los usuarios. Estos ficheros están escritos en formato JSON, y serán descritos más adelante.
- **Resources**: En esta carpeta se guardan los sprites que han de poder ser accedidos por Unity mediante la clase ya existente Resources. Esto se hace para agilizar el proceso de carga o sustitución de dichos sprites.
- **Scenes**: En esta carpeta se guardan todas las escenas que componen la aplicación.
- **Scripts**: En esta carpeta se guardan todos los ficheros C# que contienen la funcionalidad de la aplicación, así como otros ficheros TXT de utilidad. Los scripts y su contenido serán explicados más adelante.
- **Shop**: En esta carpeta se almacena el fichero JSON que contiene la lista de elementos que pueden ser comprados en la tienda.

4.2 Estructura de las escenas

En este apartado se profundizará en las distintas escenas que componen la aplicación, sus componentes y los diferentes scripts que hacen que dicha escena funcione correcta.

4.2.1 Escena 1 – Menú principal

4.2.1.1 Descripción

Esta escena presenta la primera vista que se muestra al usuario nada más iniciar la aplicación.

Desde aquí, el usuario puede crear un perfil nuevo, seleccionar uno existente o salir de la aplicación.

La siguiente imagen muestra la representación visual de esta escena.

Figura 4 - 2: Menú principal Dixy

4.2.1.2 Scripts utilizados en esta escena

La única funcionalidad que se ha necesitado implementar para esta escena es la que consigue navegar entre escenas, de forma que cuando se pulsa en cualquiera de los botones se muestre la escena correcta.

Para ello, se ha generado el script Botones.cs, el cual contiene toda la funcionalidad relacionada con dichos botones.

La función que permite navegar entre escenas se representa en la Figura 4-3 almacenada en el Anexo C, junto con las demás figuras relacionadas con código.

Cuando se hace clic en alguno de los botones se ejecuta la función que está encapsulada en su procedimiento onClick, siendo esta la mostrada en la anterior imagen. Se recibe como parámetro el nombre de la escena a cargar y se procede con el intercambio.

Los dos primeros botones muestran las escenas correspondientes con su nombre, y el tercero y último cierra la aplicación.

4.2.2 Escena 2 – Creación de un perfil nuevo

4.2.2.1 Descripción

Esta escena permite al usuario crear un nuevo perfil para ser utilizado posteriormente.

Dicho usuario ha de escribir su nombre y seleccionar los sprites de personalización que más le gusten (actualizando en tiempo real el avatar que se muestra a la izquierda), y para finalizar la creación deberá pulsar el botón de guardar, mostrando al usuario la escena correspondiente a la lista de minijuegos.

En caso de que cambie de opinión y no quiera crear un nuevo perfil, siempre puede seleccionar el botón de retroceso que se muestra arriba a la izquierda.

La siguiente imagen muestra la representación visual de esta escena.

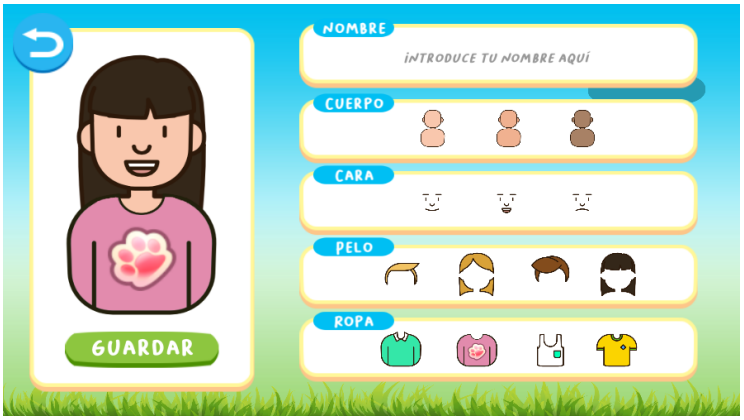

Figura 4 - 3: Creación perfil Dixy

4.2.2.2 Scripts utilizados en esta escena

Para esta escena ha sido necesario la implementación de un script que permita el almacenamiento de los datos correspondientes a la creación de un perfil de usuario, así como la actualización en tiempo real del avatar de dicho usuario.

Para ello, se han utilizado los siguientes scripts:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso o cuando el usuario haya guardado el perfil.
- **Perfiles.cs**: En este script se almacenan todas las funciones correspondientes a la gestión de perfiles de usuario. En este caso, se han utilizado las siguientes funciones:
    - **SaveProfile**: Función que se encarga de guardar un perfil de usuario nuevo a partir de los datos proporcionados por dicho usuario. Para ello, hace un chequeo previo para comprobar si el usuario ha rellenado todos los campos pertinentes (que en este caso es sólo el nombre puesto que siempre hay seleccionado un sprite por defecto), y en caso de que todo esté correcto se procede con la creación. Este proceso consiste en generar una estructura de tipo Perfil con los datos proporcionados, parsear esa estructura a formato JSON y guardarla en un fichero correspondiente a ese perfil.
    - **JSONToFile**: Función que realiza el parseo de una estructura de tipo Profile a un fichero JSON. Esto se consigue mediante la clase ya proporcionada JsonUtility, la cual se encargará de dicha transformación. En la siguiente imagen se puede observar todo el procedimiento que se ha realizado.
- **CambiarAvatar.cs**: Este script se encarga de actualizar el avatar en tiempo real. Esto se consigue añadiendo una función que se ejecute cada vez que se seleccione alguno de los sprites disponibles (los cuales son botones) la cual cambie el sprite del avatar al que está asociado al botón, teniendo en cuenta si dicho botón representa el cuerpo, la cara, el peinado o el atuendo.

4.2.3 Escena 3 – Selección de un perfil

4.2.3.1 Descripción

Esta escena le permite al usuario seleccionar un perfil creado anteriormente, o crear uno en alguno de los slots que estén vacíos.

Desde aquí, el usuario puede crear un perfil nuevo, seleccionar uno existente o retroceder a la escena anterior (menú principal) mediante el botón de retroceso.

La siguiente imagen muestra la representación visual de esta escena.

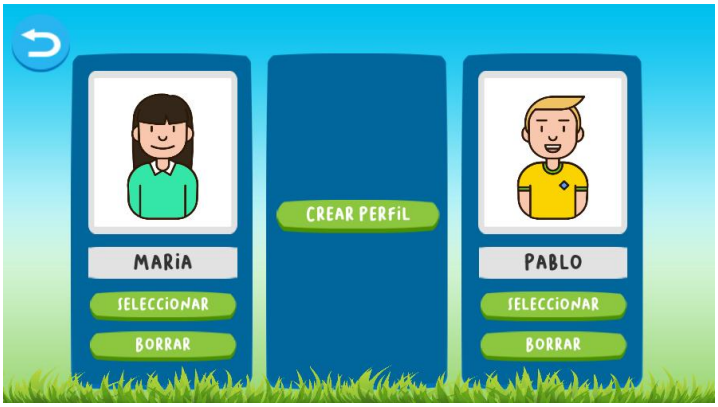

Figura 4 - 4: Selección perfil Dixy

4.2.3.2 Scripts utilizados en esta escena

Los scripts que han sido necesarios para esta escena son los siguientes:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso.
- **Perfiles.cs**: Al inicio de esta escena se analiza si hay algún perfil creado, en caso afirmativo se actualiza el slot correspondiente con los datos de dicho perfil y en caso negativo, se añade un botón de creación de perfil. Esta lectura de datos es posible mediante la función ParseJSONFile, la cual lee un fichero JSON y lo parsea a su consecuente clase Profile. Además, se han implementado cuatro funciones nuevas en este script:
    - **CreateProfile**: Función que le permite al usuario navegar a la escena de creación de usuario teniendo en cuenta el slot en el que se va a guardar este nuevo perfil.
    - **SelectProfile**: Función que le permite al usuario navegar a la escena de menú de minijuegos correspondiente al perfil que se ha seleccionado.
    - **DeleteProfile**: Función que le permite al usuario borrar un perfil ya creado. Para ello sólo hace falta borrar el fichero JSON correspondiente a dicho perfil.

4.2.4 Escena 4 – Menú de minijuegos

4.2.4.1 Descripción

Esta escena contiene la lista de minijuegos disponibles para el jugador, así como el acceso a la tienda, el armario y las estadísticas generadas.

También puede volver al menú principal haciendo uso del botón de retroceso.

La siguiente imagen muestra la representación visual de esta escena.

Figura 4 - 5: Menú minijuegos Dixy

4.2.4.2 Scripts utilizados en esta escena

Para esta escena se han requerido de los siguientes scripts:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso o para acceder a las estadísticas, la tienda y el armario. También se ha implementado la función ChangeMinigame, la cual tiene en cuenta el minijuego que se ha seleccionado.
- **Perfiles.cs**: Se ha implementado una nueva función (llamada ChangeMiniAvatar) la cual permita modificar el nombre y avatar del usuario. Para ello, se sustituye el sprite actual por el que está almacenado en el perfil del usuario (y este perfil ha sido creado a partir de la función ParseJSONFile ya explicada anteriormente).

4.2.5 Escena 5 – Minijuego 1 – Cuentasílabas

4.2.5.1 Descripción

En esta escena se incluye toda la funcionalidad correspondiente al primer minijuego, llamado Cuentasílabas.

Este minijuego consiste en pulsar las palabras (las cuales se generan con posición y texto aleatorios) que contengan el mismo número de sílabas que las que se especifican en el recuadro amarillo.

Si la palabra es correcta, se incrementa en uno el contador de palabras correctas y dicha palabra se colorea de verde, en caso contrario, se incrementa en uno el contador de palabras incorrectas y la palabra se colorea de rojo.

Cuando se haya alcanzado un total de cinco palabras tanto correctas como incorrectas, el minijuego se detendrá, mostrando un resultado diferente dependiendo del final obtenido.

Las siguientes imágenes muestran la representación visual de esta escena.

Figura 4 - 6: Modal inicial minijuego

Figura 4 - 7: Escena primer minijuego

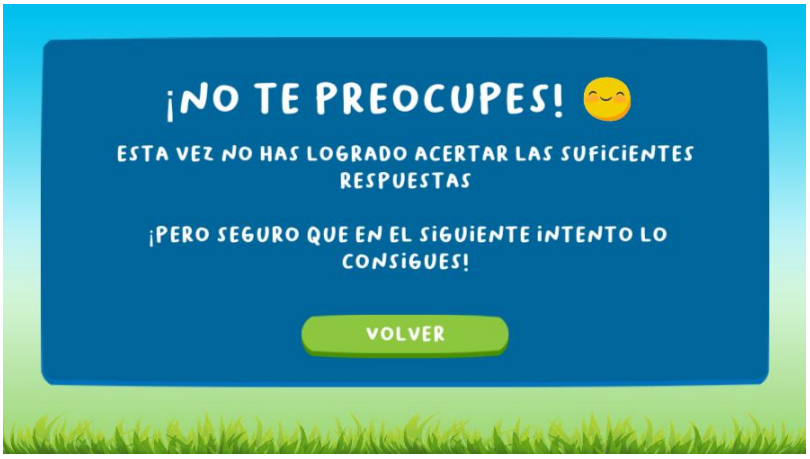

Figura 4 - 8: Modal minijuego completado incorrectamente

Figura 4 - 9: Modal minijuego completado correctamente

4.2.5.2 Scripts utilizados en esta escena

Para esta escena se han necesitado los siguientes scripts:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso o cuando se ha finalizado el minijuego.

- **Timer.cs**: En este script se almacenan todas las funciones que gestionan el contador de tiempo presente en todos los minijuegos. Para esta escena ha sido necesario conseguir que el contador avanzase a medida que pasaban los segundos, y eso se ha conseguido mediante la creación de un campo serializable el cual gestionará la velocidad del reloj.
- **GenerarPalabras.cs**: En este script se almacenan las funciones que son necesarias para el propio funcionamiento del minijuego. Las funciones más importantes que se han generado son las siguientes:
    - **Start**: Función propia de Unity la cual se ejecuta nada más arrancar la escena. En esta función se ha añadido la funcionalidad que permite la lectura de los ficheros que almacenan las palabras con sus diferentes números de sílabas, su consecuente almacenamiento en listas y la elección aleatoria del número de sílabas a buscar.
    - **Update**: Función propia de Unity la cual se ejecuta al inicio de cada frame. En esta función se ha añadido la funcionalidad que permite generar las palabras en tiempo real y el código necesario para procesar si la palabra seleccionada es correcta o no y su posterior procesamiento. Para ello, se ejecutan los siguientes pasos:
        1. Se genera una entidad de tipo Palabra, se le otorga una posición, una fuente de texto y se le asocia la función de tipo onClick CheckWord, la cual procesará si la palabra es correcta o no.
        2. Se añade la entidad a la escena.
        3. Se espera el número de segundos especificado para generar la siguiente palabra.
    - **CheckWord**: Función que comprueba si la palabra contiene el mismo número de sílabas que las que se le especifica al usuario. Para ello, nada más seleccionar una palabra, se ejecutan los siguientes pasos:
        1. Se marca la palabra como seleccionada mediante su atributo HasBeenSelected.
        2. Se comprueba si la palabra seleccionada está contenida en la lista correspondiente a las palabras con el número de sílabas especificado.
        3. En caso correcto, se colorea la palabra de verde y se incrementa en uno el contador de palabras correctas.
        4. En caso incorrecto, se colorea la palabra de rojo y se incrementa en uno el contador de palabras incorrectas.
        5. Si se ha llegado al número máximo permitido de palabras correctas o incorrectas, se muestra el modal final correspondiente y se ocultan el resto de los elementos presentes en la escena. Además, se genera una instancia del juego mediante la clase Game1 y se añade a la lista de juegos del perfil del usuario mediante la función GenerateGameJSON.

6. En caso de que la actividad haya finalizado satisfactoriamente, se añadirán cinco monedas al inventario del jugador para que pueda gastarlas en la tienda.

- **GenerateGameJSON**: Función que genera una instancia de Game1 y la guarda en la lista del perfil del usuario activo. Para ello, almacena el número de errores y el tiempo que ha tardado en completar el ejercicio y modifica el archive JSON asociado a dicho usuario.

4.2.6 Escena 6 – Minijuego 2 – Atrapalabras

4.2.6.1 Descripción

En esta escena se incluye toda la funcionalidad correspondiente al segundo minijuego, llamado Atrapalabras.

Este minijuego consiste en pulsar las palabras (las cuales se generan con posición y texto aleatorios) que contengan la misma sílaba que la que se especifica en el recuadro amarillo.

Si la palabra es correcta, se incrementa en uno el contador de palabras correctas y dicha palabra se colorea de verde, en caso contrario, se incrementa en uno el contador de palabras incorrectas y la palabra se colorea de rojo.

Cuando se haya alcanzado un total de cinco palabras tanto correctas como incorrectas, el minijuego se detendrá, mostrando un resultado diferente dependiendo del final obtenido.

La siguiente imagen muestra la representación visual de esta escena.

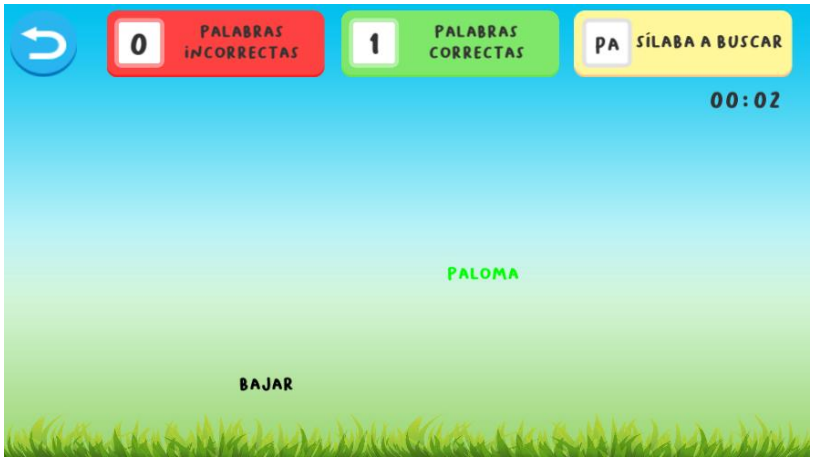

Figura 4 - 10: Escena segundo minijuego

4.2.6.2 Scripts utilizados en esta escena

Para esta escena se han necesitado los siguientes scripts:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso o cuando se ha finalizado el minijuego.
- **Timer.cs**: En este script se almacenan todas las funciones que gestionan el contador de tiempo presente en todos los minijuegos. Para esta escena ha sido necesario reutilizar la misma funcionalidad mencionada en el primer minijuego.

- **GenerarSilabas.cs**: En este script se almacenan las funciones que son necesarias para el propio funcionamiento del minijuego. La mayoría de funcionalidad necesaria es muy similar a la mencionada en el primer minijuego, así que no voy a entrar en demasiado detalle. En este caso, se necesita la lectura de ficheros que almacenan palabras dependiendo de la sílaba que contienen.

4.2.7 Escena 7 – Minijuego 3 – Palabras revueltas

4.2.7.1 Descripción

En esta escena se incluye toda la funcionalidad correspondiente al tercer minijuego, llamado Palabras Revueltas.

Este minijuego consiste en ordenar las sílabas que se muestran para formar la palabra que representa la imagen de la izquierda, y que tiene el mismo número de sílabas que las que se muestran en el recuadro amarillo.

A medida que se van pulsando las sílabas, estas se van concatenando hasta formar una palabra final. Esta palabra se analiza para comprobar si contiene las sílabas que ha de contener y se contabiliza como acierto o como error antes de pasar a la siguiente palabra.

Si la palabra es correcta, se incrementa en uno el contador de palabras correctas y dicha palabra se colorea de verde, en caso contrario, se incrementa en uno el contador de palabras incorrectas y la palabra se colorea de rojo.

Cuando se haya alcanzado un total de cinco palabras tanto correctas como incorrectas, el minijuego se detendrá, mostrando un resultado diferente dependiendo del final obtenido.

La siguiente imagen muestra la representación visual de esta escena.

Figura 4 - 11: Escena tercer minijuego

4.2.7.2 Scripts utilizados en esta escena

Para esta escena se han necesitado los siguientes scripts:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso o cuando se ha finalizado el minijuego.

- **Timer.cs**: En este script se almacenan todas las funciones que gestionan el contador de tiempo presente en todos los minijuegos.
- **GenerarOrdenSilabas.cs**: En este script se almacenan las funciones que son necesarias para el propio funcionamiento del minijuego. Las funciones más importantes que se han generado son las siguientes:
    - **Start**: En esta función se ha añadido la funcionalidad que permite la lectura del fichero que almacena las palabras con su número de sílabas y las propias sílabas ya separadas, y la selección aleatoria de la palabra a recomponer.
    - **PrepareRound**: Función que se llama al inicio de cada ronda y que prepara tanto la imagen como la distribución de las sílabas a lo largo del panel de 3x3.
    - **ClicOnSyllabe**: Función que se llama al seleccionar cualquiera de las sílabas del panel, y que realiza las siguientes tareas:
        1. Concatena la sílaba a la palabra formada.
        2. Comprueba si el número de sílabas de la palabra formada coincide con el de la palabra total.
        3. En caso correcto se colorea la palabra de verde, se incrementa en uno el contador de palabras correctas y se muestra un botón que permite avanzar a la siguiente ronda.
        4. En caso erróneo se colorea la palabra de rojo, se incrementa en uno el contador de palabras incorrectas y se muestra un botón que permite avanzar a la siguiente ronda.
        5. Si se ha llegado al número máximo permitido de palabras correctas o incorrectas, se muestra el modal final correspondiente y se ocultan el resto de los elementos presentes en la escena. Además, se genera una instancia del juego mediante la clase Game1 y se añade a la lista de juegos del perfil del usuario mediante la función GenerateGameJSON.
        6. En caso de que la actividad haya finalizado satisfactoriamente, se añadirán cinco monedas al inventario del jugador para que pueda gastarlas en la tienda.
    - **GenerateGameJSON**: Esta función es la misma que la mencionada en los minijuegos anteriores, por lo que no se va a proceder a su explicación.

4.2.8 Escena 8 – Minijuego 4 – Sílaba escondida

4.2.8.1 Descripción

En esta escena se incluye toda la funcionalidad correspondiente al cuarto y último minijuego, llamado Sílaba Escondida.

Este minijuego consiste en seleccionar la sílaba que le falta a la palabra que se muestra en el panel de la izquierda. El panel de la derecha muestra cuatro sílabas diferentes, y el usuario ha de seleccionar la que le parezca correcta.

Si la sílaba es correcta, se incrementa en uno el contador de respuestas correctas y dicha sílaba se colorea de verde, en caso contrario, se incrementa en uno el contador de respuestas incorrectas y la sílaba se colorea de rojo.

Cuando se haya alcanzado un total de cinco rondas tanto correctas como incorrectas, el minijuego se detendrá, mostrando un resultado diferente dependiendo del final obtenido.

La siguiente imagen muestra la representación visual de esta escena.

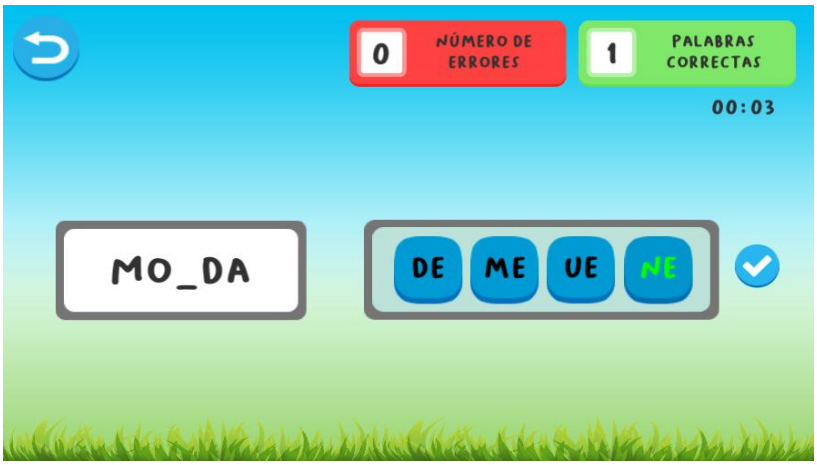

Figura 4 - 12: Escena cuarto minijuego

4.2.8.2 Scripts utilizados en esta escena

Para esta escena se han necesitado los siguientes scripts:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso o cuando se ha finalizado el minijuego.
- T**imer.cs**: En este script se almacenan todas las funciones que gestionan el contador de tiempo presente en todos los minijuegos.
- **GenerarCompletarPalabra.cs**: En este script se almacenan las funciones que son necesarias para el propio funcionamiento del minijuego. Las funciones más importantes que se han generado son las siguientes:
    - **Start**: En esta función se ha añadido la funcionalidad que permite la lectura del fichero que almacena las palabras con su número de sílabas y las propias sílabas ya separadas, y la selección aleatoria de la palabra a recomponer.
    - **PrepareRound**: Función que se llama al inicio de cada ronda y que prepara tanto la imagen como la distribución de las sílabas a lo largo del panel de 1x4.
    - **ClicOnSyllabe**: Función que se llama al seleccionar cualquiera de las sílabas del panel, y que realiza las siguientes tareas:
        1. Comprueba si la sílaba seleccionada coincide con la sílaba que le falta a la palabra.

2. En caso correcto se colorea la palabra de verde, se incrementa en uno el contador de palabras correctas y se muestra un botón que permite avanzar a la siguiente ronda.
3. En caso erróneo se colorea la palabra de rojo, se incrementa en uno el contador de palabras incorrectas y se muestra un botón que permite avanzar a la siguiente ronda.
4. Si se ha llegado al número máximo permitido de palabras correctas o incorrectas, se muestra el modal final correspondiente y se ocultan el resto de los elementos presentes en la escena. Además, se genera una instancia del juego mediante la clase Game1 y se añade a la lista de juegos del perfil del usuario mediante la función GenerateGameJSON.
5. En caso de que la actividad haya finalizado satisfactoriamente, se añadirán cinco monedas al inventario del jugador para que pueda gastarlas en la tienda.

- **GenerateGameJSON**: Esta función es la misma que la mencionada en los minijuegos anteriores, por lo que no se va a proceder a su explicación.

4.2.9 Escena 9 – Estadísticas del jugador

4.2.9.1 Descripción

En esta escena se incluye un listado de los diferentes resultados obtenidos en los minijuegos.

Los datos que se muestran son los siguientes:

- Número de veces que el usuario ha jugado a cada minijuego.
- Tiempo medio en resolver el ejercicio.
- Media de fallos contabilizando todas las veces que el usuario ha jugado a ese minijuego.

La siguiente imagen muestra la representación visual de esta escena.

Figura 4 - 13: Estadísticas del usuario

4.2.9.2 Scripts utilizados en esta escena

Para esta escena se han necesitado los siguientes scripts:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso.
- **Estadisticas.cs**: En este script se incluyen funciones que permiten el análisis de las diferentes instancias de los minijuegos que el jugador ha ido generando. Para ello, se han necesitado las siguientes funciones:
    - **Start**: En esta función se añade el código necesario para la lectura del fichero JSON correspondiente al perfil del usuario actual (mediante la función ParseJSONFile mencionada anteriormente), el cálculo estadístico de los datos extraídos y la representación de los resultados por pantalla.
    - **CalculateAverageTime**: Función que calcula el tiempo promedio que ha necesitado el usuario para completar las diferentes instancias de los minijuegos. Para ello, se hace uso de la clase TimeSpan teniendo en cuenta que esta clase muestra también los milisegundos, por lo que hay que eliminarlos antes de retornar el resultado final.
    - **CalculateAverageErrors**: Función que calcula el número promedio de errores que ha cometido el usuario en las diferentes instancias de los minijuegos. Para ello, se hace una simple media algebraica de todos los errores almacenados.

4.2.10 Escena 10 – Tienda de complementos

4.2.10.1 Descripción

En esta escena se incluye una tienda de complementos para el avatar del usuario, de forma que pueda obtener diferentes recompensas a cambio de su esfuerzo y su trabajo duro practicando.

Los elementos que se pueden comprar son diferentes peinados y atuendos, los cuales una vez comprados, aparecerán en el armario del jugador para poder modificar su avatar actual.

Cada vez que el usuario compre un elemento, se descontarán las monedas correspondientes al precio de su inventario. En caso de que se intente comprar un elemento cuyo precio sea superior al presupuesto disponible, se mostrarán tanto el total de monedas obtenidas como el precio del artículo en rojo.

La siguiente imagen muestra la representación visual de esta escena.

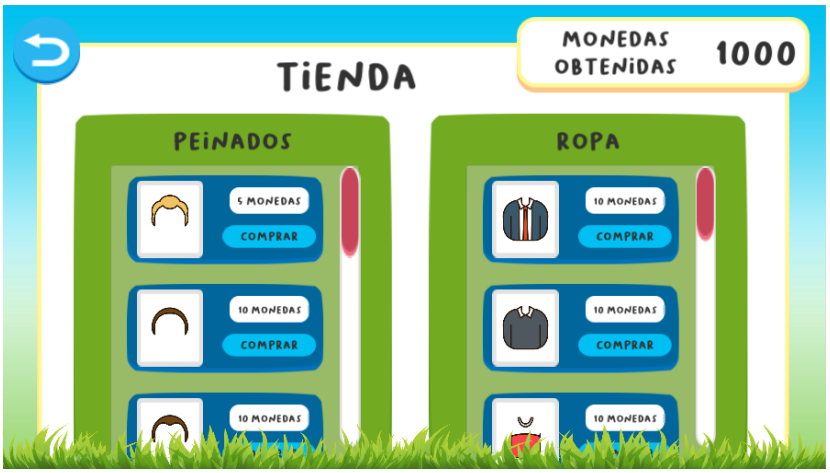

Figura 4 - 14: Tienda de complementos

4.2.10.2 Scripts utilizados en esta escena

Para esta escena se han necesitado los siguientes scripts:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso.
- **Tienda.cs**: Este script contiene las funciones necesarias para el funcionamiento de la tienda. Estas funciones son las siguientes:
    - **Start**: En esta función se lee el fichero JSON del usuario actual (mediante la función ParseJSONFile mencionada anteriormente), se extrae el número de monedas obtenidas por dicho usuario, y se muestra por pantalla los diferentes elementos que se encuentran almacenados en el catálogo de la tienda. Este catálogo se ve representado por otro fichero JSON pero que esta vez hace referencia a la clase Catalogue.
    - **OnBuyHair**: Esta función se ejecuta cuando el usuario selecciona un peinado de la lista para intentar comprarlo. Primero, comprueba que el precio no exceda el número de monedas disponibles, y en caso de que la compra sea posible, descuenta la cantidad correspondiente y añade el elemento a la lista de peinados obtenidos por el usuario, modificando el fichero JSON de dicho usuario mediante la función ya mencionada anteriormente JSONToFile. Una vez esta transacción ha finalizado, se procede a eliminar este elemento de la lista de objetos comprables por el usuario.
    - **OnBuyKit**: Función que se ejecuta cuando el usuario intenta comprar un atuendo de todos los disponibles. Su funcionamiento es idéntico a la función anterior excepto que esta vez se modifica la lista de atuendos en vez de la de peinados.

4.2.11 Escena 11 – Armario del usuario

4.2.11.1 Descripción

En esta escena se incluyen todos los elementos que o se proporcionan por defecto para que el usuario cree su propio avatar, o han sido comprados en la tienda.

Se muestra una representación de dicho avatar a la izquierda y una serie de botones a la derecha con los sprites correspondientes a los elementos obtenidos.

El avatar se puede guardar con el botón de guardado, el cual retornará a la escena de los menús de juegos, o puede volver hacia atrás mediante el botón de retroceso.

La siguiente imagen muestra la representación visual de esta escena.

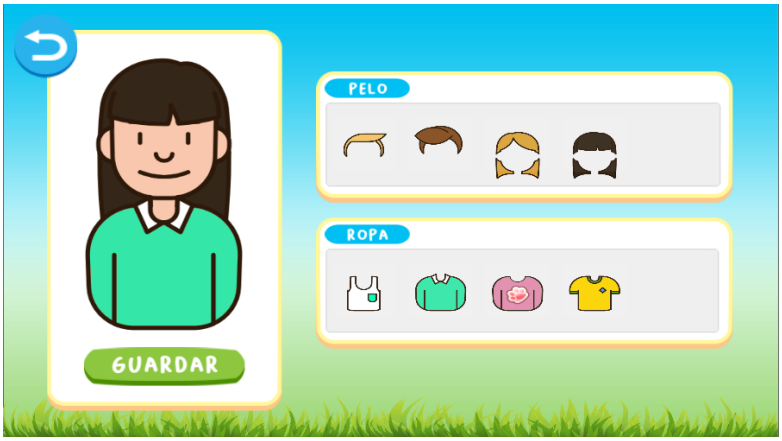

Figura 4 - 15: Armario del usuario

4.2.11.2 Scripts utilizados en esta escena

Para esta escena se han necesitado los siguientes scripts:

- **Botones.cs**: Se ha necesitado la función ChangeScene mencionada anteriormente para cambiar de escena cuando el usuario quiera volver hacia atrás usando el botón de retroceso o al guardar el avatar.
- **AvatarScript.cs**: Este script contiene todas las funciones necesarias para el funcionamiento de esta escena. Estas funciones son las siguientes:
    - **Start**: En esta función se lee el fichero JSON del usuario actual (mediante la función ParseJSONFile mencionada anteriormente), y se extrae tanto el avatar actual como los distintos elementos que el usuario ha obtenido.
    - **OnClicHair**: Esta función se ejecuta cuando el usuario selecciona un peinado, intercambiando así el sprite del modelo de la izquierda por el seleccionado.
    - **OnClicKit**: Función que se ejecuta cuando el usuario selecciona un atuendo de todos los disponibles. Su funcionamiento es idéntico a la función anterior.
    - **SaveProfile**: Función que se ejecuta al presionar el botón de guardado. Primero se modifica la clase Profile generada anteriormente durante la función Start con los nuevos sprites seleccionados, y luego se modifica el fichero JSON del perfil del usuario mediante la función JSONToFile ya explicada anteriormente.

5 Integración, pruebas y resultados

En este capítulo se explicarán las diferentes pruebas que se han realizado para comprobar el funcionamiento del producto final.

5.1 Comprobación requisitos funcionales

Como primera tarea a realizar, se debe comprobar si se han cumplido todos los requisitos funcionales que se habían propuesto anteriormente.

La siguiente tabla contiene el código del requisito funcional, su descripción abreviada y si se ha podido realizar durante la ejecución de la aplicación o no.

REQUISITO	DESCRIPCIÓN	RESULTADO
RF 1	Creación perfil mediante nombre y avatar	CORRECTO
RF 2	Eliminación de perfil de usuario	CORRECTO
RF 3	Acceso a perfil de usuario creado	CORRECTO
RF 4	Creación de avatar de usuario	CORRECTO
RF 5	Estadísticas con número de veces, tiempo promedio y número de errores	CORRECTO
RF 6	Finalización minijuego al obtener 5 aciertos o 5 fallos.	CORRECTO
RF 7	Incremento de las monedas del usuario en 5 unidades en caso de terminar un minijuego satisfactoriamente	CORRECTO
RF 8	Tienda de complementos con peinados y atuendos	CORRECTO
RF 9	Modificación avatar usuario	CORRECTO
RF 10	Minijuego 1 – Seleccionar palabras que contengan un cierto número de sílabas	CORRECTO
RF 11	Minijuego 2 – Seleccionar palabras que tengan cierta sílaba	CORRECTO
RF 12	Minijuego 3 – Ordenar sílabas hasta formar la palabra especificada	CORRECTO
RF 13	Minijuego 4 – Seleccionar la sílaba que le falta a una palabra dada	CORRECTO

Tabla 5 - 1: Tabla cumplimiento requisitos

Como se puede apreciar, se han cumplido con todos los requisitos que se habían establecido en un principio.

5.2 Pruebas unitarias

Para comprobar que el código era correcto y no tenía fallos de funcionalidad, se han realizado las siguientes pruebas unitarias:

- El usuario no puede crear un perfil de usuario hasta que no proporcione un nombre.
- No se puede seleccionar varias veces una misma palabra en el primer minijuego.
- No se puede seleccionar varias veces una misma palabra en el segundo minijuego.
- En el tercer minijuego, una vez se ha llegado al tope de sílabas para seleccionar no se pueden seleccionar más hasta que no se pase la ronda.
- En el cuarto minijuego, no se puede seleccionar una sílaba si otra ya ha sido seleccionada previamente hasta que no se pase la ronda

5.3 Comprobación utilidad

Otro factor que es muy importante revisar es si la aplicación es efectivamente útil para la realización de ejercicios que mejoren la condición de dislexia en niños cuya edad esté comprendida entre los 5 y los 8 años.

Para ello, he tomado la decisión de entregarle esta aplicación a una experta en esta materia para que la utilizase durante sus sesiones de terapia.

Junto al proyecto, también le entregué una tabla con diferentes entradas para que evaluase el rendimiento de la aplicación.

Estos han sido los comentarios que se han recibido:

CUESTIONES	COMENTARIOS
La aplicación ha funcionado sin problemas y de forma fluida	Completamente de acuerdo
El sistema de perfiles es útil y fácil de utilizar	Completamente de acuerdo
Los niños buscan de forma activa utilizar la aplicación porque les resulta divertida	De acuerdo
Los minijuegos son fáciles de entender	De acuerdo
Las estadísticas son útiles para comprender qué se debe mejorar	Completamente de acuerdo
El conjunto de minijuegos cubre todos los tipos de dislexia lingüística que existen	De acuerdo
La obtención de monedas y la inclusión de la tienda motiva a los usuarios	Completamente de acuerdo

La aplicación es estética y atractiva visualmente	Completamente de acuerdo

Tabla 5 - 2: Tabla comentarios utilidad

Además, se han aportado los siguientes comentarios extra que no se veían reflejados en la tabla anterior.

- Para futuras actualizaciones, sería útil añadir un sistema de audio que permita escuchar las palabras para reforzar mediante la audición.
- Se podrían añadir más minijuegos que profundicen en la identificación de símbolos.

5.4 Resultados

Como producto final de este Trabajo de Fin de Grado, se ha obtenido una aplicación que le permite a niños cuya edad esté comprendida entre los 5 y los 8 años con dislexia realizar ejercicios que les permita practicar y mejorar directamente desde su casa.

Los resultados obtenidos han sido muy satisfactorios y acordes a lo que se había planteado en un inicio con la profesional.

6 Conclusiones y trabajo futuro

En este apartado se incluyen las conclusiones y el posible trabajo futuro que habría que aplicar sobre este proyecto.

6.1 Conclusiones

El proyecto propuesto consistía en la creación de una aplicación para ordenadores que les permitiese a niños pequeños, cuya edad estuviese entre los 5 y los 8 años, realizar ejercicios orientados a la práctica de tareas basadas en procesos de lectoescritura.

Para ello, se debían plantear ciertos ejercicios que entrenasen las siguientes características:

- Identificación y reconocimiento de símbolos y letras.
- Correcta separación de palabras en sílabas junto con la identificación del número total de sílabas de dichas palabras.
- Asociación de palabras con su representación visual.

Esta aplicación tenía que ser programada en C# utilizando el motor de videojuegos Unity y el software de desarrollo Visual Studio, haciendo también uso de otros elementos como el formato JSON.

La aplicación debía ser tutelada por una profesional en el campo de la psicología y ha sido ella quien debía sugerir qué ejercicios plantear.

Era fundamental organizar a los distintos usuarios mediante la implementación de un sistema de perfiles, el cual permita diferenciarlos y personalizarlos mediante un nombre y un avatar.

Además, también se tenía que añadir algún sistema que consiguiese motivar a los usuarios a seguir utilizando la aplicación y por tanto seguir aprendiendo.

Esto se ha conseguido mediante las siguientes características:

- Creación de un sistema de recompensa por monedas, de forma que cuando el usuario finalizase satisfactoriamente un ejercicio se le recompensase otorgándole 5 monedas. Estas monedas pueden ser gastadas en una tienda en la que se ofertan elementos con los que alterar el avatar de dicho usuario, y todos los elementos ya comprados han de ser almacenados en su armario.
- Presentación de un mensaje positivo y alentador en caso de que el usuario no haya sido capaz de superar el ejercicio correctamente para que no se sienta desanimado.

Todas las funcionalidades que se habían planteado se han podido programar, obteniendo como resultado el proyecto presentado en este Trabajo de Fin de Grado.

6.2 Trabajo futuro

En este apartado se expondrán las diferentes funcionalidades a añadir en un posible trabajo futuro:

- Inclusión de un sistema de reproducción de sonidos, de forma que las distintas palabras y sílabas puedan ser escuchadas por el usuario.

- Inclusión de un sistema de escucha, el cual le permita al usuario hablar por el micrófono y repetir las palabras o sílabas que se le pidan.
- Añadir más sprites a la lista de elementos de personalización del avatar.
- Incluir Android e iOS como sistemas operativos compatibles con la aplicación.
- Añadir más de 3 perfiles de usuarios.

Referencias

[1] «Understanding Dyslexia». [Online]. Disponible en: https://www.dyslexiacenterofutah.org/Statistics. [Accedido: 10-abr-2019]

[2] «Tipos de dislexia | La Dislexia». [Online]. Disponible en: http://www.ladislexia.net/tipos-clasificacion/, http://www.ladislexia.net/tipos-clasificacion/. [Accedido: 10-abr-2019]

[3] «SÍNTOMAS DE LA DISLEXIA POR EDADES – Blog de Change Dyslexia». [Online]. Disponible en: https://blog.changedyslexia.org/sintomas-de-la-dislexia-por-edades/. [Accedido: 10-abr-2019]

[4] «Metodología de un proyecto | Sinnaps», *Gestor de proyectos online*. [Online]. Disponible en: https://www.sinnaps.com/blog-gestion-proyectos/metodologia-de-un-proyecto. [Accedido: 30-may-2019]

[5] «La dislexia para padres y profesionales». [Online]. Disponible en: http://www.ladislexia.net/, http://www.ladislexia.net/. [Accedido: 10-abr-2019]

[6] «Horizontal scrollview with images constant height in Unity3D UI - How?», *Stack Overflow*. [Online]. Disponible en: https://stackoverflow.com/questions/39099265/horizontal-scrollview-with-images-constant-height-in-unity3d-ui-how. [Accedido: 28-may-2019]

[7] «Ejemplos de palabras monosílabas, bisílabas, trisílabas, tetrasílabas, pentasílabas, hexasílabas, heptasílabas, octosílabas, eneasílabas y decasílabas. Palabras polisílabas. Juegos de palabras.» [Online]. Disponible en: http://www.juegosdepalabras.com/silabas/silaba2.htm. [Accedido: 15-abr-2019]

[8] «DISFAM – Dislexia una Dificultad Específica de Aprendizaje». [Online]. Disponible en: https://www.disfam.org/dislexia/. [Accedido: 10-abr-2019]

[9] «Diseño de interfaces para niños». [Online]. Disponible en: http://www.torresburriel.com/weblog/2016/08/18/diseno-de-interfaces-para-ninos/. [Accedido: 01-jun-2019]

[10] «Diseño de interfaces multimedia». [Online]. Disponible en: http://cv.uoc.edu/annotation/77847c78a26395a6bb77f8e08b504b8a/485065/PID_00159828/modul_1.html#w26aab5b9b3. [Accedido: 01-jun-2019]

[11] «Crear un diagrama de flujo básico». [Online]. Disponible en: https://support.office.com/es-es/article/crear-un-diagrama-de-flujo-b%c3%a1sico-e207d975-4a51-4bfa-a356-eeec314bd276. [Accedido: 30-may-2019]

[12] «Causas de la dislexia | La Dislexia». [Online]. Disponible en: http://www.ladislexia.net/causas-etiologia/. [Accedido: 10-abr-2019]

[13] «c# - Random Word Generator #2», *Stack Overflow*. [Online]. Disponible en: https://stackoverflow.com/questions/18110243/random-word-generator-2. [Accedido: 15-abr-2019]

[14] «7 Positive Benefits of Technology for Children | Pros & Cons in Early Youth Development», *iD Tech*. [Online]. Disponible en: https://www.idtech.com/blog/benefits-of-technology-for-children. [Accedido: 01-jun-2019]

[15] «¿Qué factores intervienen en la evolución de la dislexia? | La Dislexia». [Online]. Disponible en: http://www.ladislexia.net/factores-evolucion/, http://www.ladislexia.net/factores-evolucion/. [Accedido: 10-abr-2019]

[16] «Zotero», *Wikipedia, la enciclopedia libre*. 11-may-2019 [Online]. Disponible en: https://es.wikipedia.org/w/index.php?title=Zotero&oldid=115883403. [Accedido: 01-jun-2019]

[17] «Tipos de metodología de un proyecto», *Capitalismo Consciente*. 26-mar-2019 [Online]. Disponible en: https://capitalismoconsciente.es/blog/tipos-de-metodologia-de-un-proyecto/. [Accedido: 30-may-2019]

[18] «Dislexia», *Wikipedia, la enciclopedia libre*. 14-mar-2019 [Online]. Disponible en: https://es.wikipedia.org/w/index.php?title=Dislexia&oldid=114589578. [Accedido: 10-abr-2019]

[19] «Programación por capas», *Wikipedia, la enciclopedia libre*. 03-dic-2018 [Online]. Disponible en: https://es.wikipedia.org/w/index.php?title=Programaci%C3%B3n_por_capas&oldid=112423566. [Accedido: 01-jun-2019]

[20] «Dislexia: qué es, síntomas, tipos y ejercicios para disléxicos», *Blog NeuronUP*. 08-nov-2018 [Online]. Disponible en: https://blog.neuronup.com/dislexia-que-es-sintomas-tipos-ejercicios-dislexicos/. [Accedido: 10-abr-2019]

[21] «Dislexia Direccional», *Dislexia feliz*, 12-feb-2018. [Online]. Disponible en: https://www.dislexiafeliz.online/tipos/direccional/. [Accedido: 10-abr-2019]

[22] «What Is Agile Methodology», *Luís Gonçalves*. 24-feb-2017 [Online]. Disponible en: https://luis-goncalves.com/es/que-es-la-metodologia-agil/. [Accedido: 30-may-2019]

[23] «Subtipos de dislexia: modelo de doble ruta», *Mentelex*. 20-abr-2015 [Online]. Disponible en: https://blog.mentelex.com/subtipos-de-dislexia/. [Accedido: 10-abr-2019]

[24] E. equipo de Understood, «Tratamientos para niños con dislexia». [Online]. Disponible en: https://www.understood.org/es-mx/learning-attention-issues/treatments-approaches/treatment-options/treatment-for-kids-with-dyslexia. [Accedido: 10-abr-2019]

[25] E. equipo de Understood, «Hoja informativa sobre la dislexia». [Online]. Disponible en: https://www.understood.org/es-mx/learning-attention-issues/child-learning-disabilities/dyslexia/dyslexia-fact-sheet. [Accedido: 10-abr-2019]

[26] E. equipo de Understood, «Entender la dislexia en niños | Tratamientos y señales de la dislexia». [Online]. Disponible en: https://www.understood.org/es-mx/learning-attention-

issues/child-learning-disabilities/dyslexia/understanding-dyslexia. [Accedido: 10-abr-2019]

[27] E. equipo de Understood, «Entender la discalculia | Discapacidades de aprendizaje en matemáticas». [Online]. Disponible en: https://www.understood.org/es-mx/learning-attention-issues/child-learning-disabilities/dyscalculia/understanding-dyscalculia. [Accedido: 10-abr-2019]

[28] E. equipo de Understood, «Dislexia: Qué es y qué no es». [Online]. Disponible en: https://www.understood.org/es-mx/learning-attention-issues/child-learning-disabilities/dyslexia/dyslexia-what-it-is-and-isnt. [Accedido: 10-abr-2019]

[29] J. L. Quijado, «La tecnología JSON – Recursos para programadores». [Online]. Disponible en: https://eldesvandejose.com/2016/10/06/la-tecnologia-json/. [Accedido: 01-jun-2019]

[30] A. Mkhitaryan, «Why Is C# Among The Most Popular Programming Languages in The World?», *Medium*. 13-oct-2017 [Online]. Disponible en: https://medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb. [Accedido: 01-jun-2019]

[31] Y. Hassan Montero, «Diseño web orientado a niños», *No Solo Usabilidad*, n.º 3, sep. 2004 [Online]. Disponible en: http://www.nosolousabilidad.com/articulos/diseno_orientado_ninos.htm. [Accedido: 01-jun-2019]

[32] R.- ASALE, «"Diccionario de la lengua española" - Edición del Tricentenario», *«Diccionario de la lengua española» - Edición del Tricentenario*. [Online]. Disponible en: http://dle.rae.es/. [Accedido: 10-abr-2019]

Glosario

Unity	Motor de videojuegos que permite realizar proyectos de diferentes tipos
Sprite	Mapa de bits que representa algún objeto visual
JSON	Formato de texto utilizado para el intercambio de datos
C#	Lenguaje de programación orientado a objetos desarrollado por Microsoft

Anexos

A Manual de instalación

Debido a que la aplicación es un ejecutable generado por Unity, sólo haría falta hacer doble clic para arrancar la aplicación.

Dicho ejecutable genera siguiendo los siguientes pasos:

1. Seleccionar la opción Build and Run contenida dentro de la sección File de la barra de herramientas.
2. Escoger una ruta destino en la que se almacenará el ejecutable junto con los ficheros necesarios.

Una vez se haya generado el ejecutable, al seleccionarlo debería mostrarse la siguiente ventana:

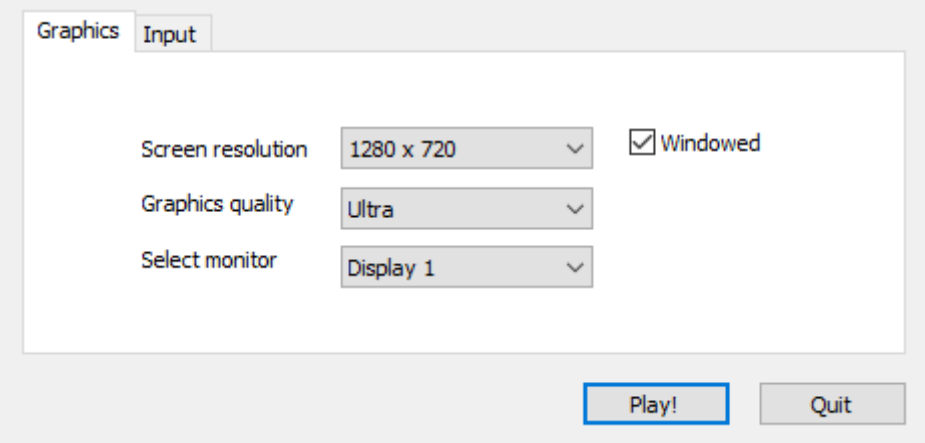

Figura A - 1: Ventana ejecución de la aplicación

El usuario podrá modificar los parámetros a su gusto y finalmente ejecutar la aplicación.

B Figuras representativas de la sección Estado del Arte

Figura C - 1: Inicio sesión aplicación Dytective

Figura C - 2: Inicio sesión aplicación Dytective

Figura C - 3: Inicio sesión aplicación Dytective

Figura C - 4: Inicio sesión aplicación Dytective

Figura C - 5: Menú principal Dytective

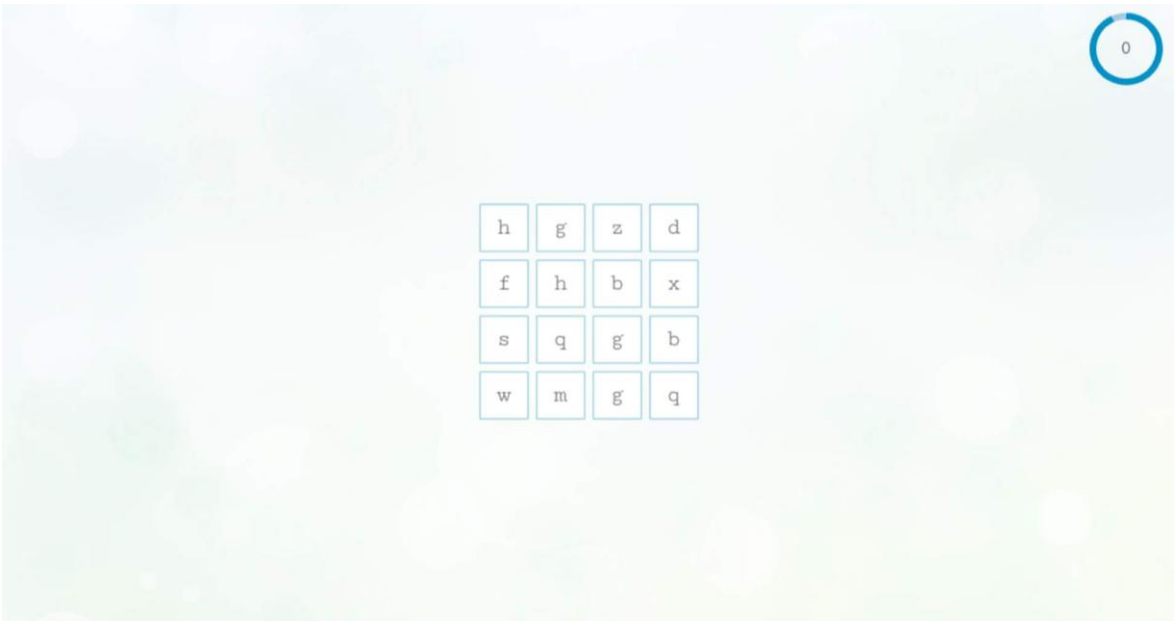

Figura C - 6: Ejemplo test Dytective

Figura C - 7: Menú juegos Dytective

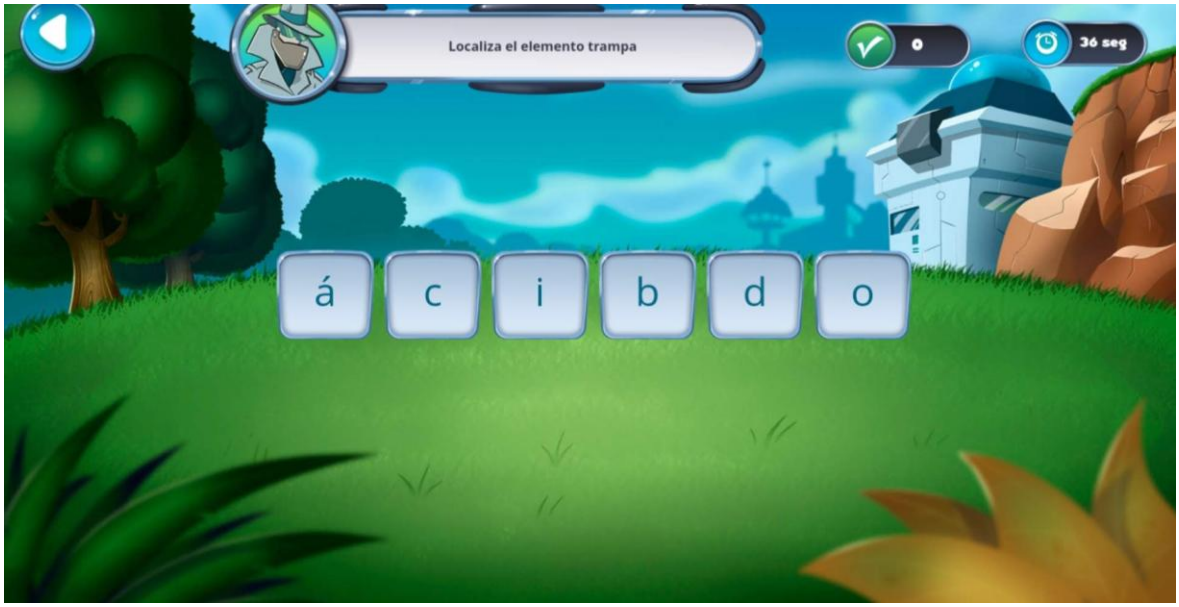

Figura C - 8: Ejemplo minijuego Dytective

Figura C - 9: Menú principal Sanapalabras

Figura C - 10: Ejemplo primer minijuego Sanapalabras

Figura C - 11: Ejemplo instrucciones segundo minijuego Sanapalabras

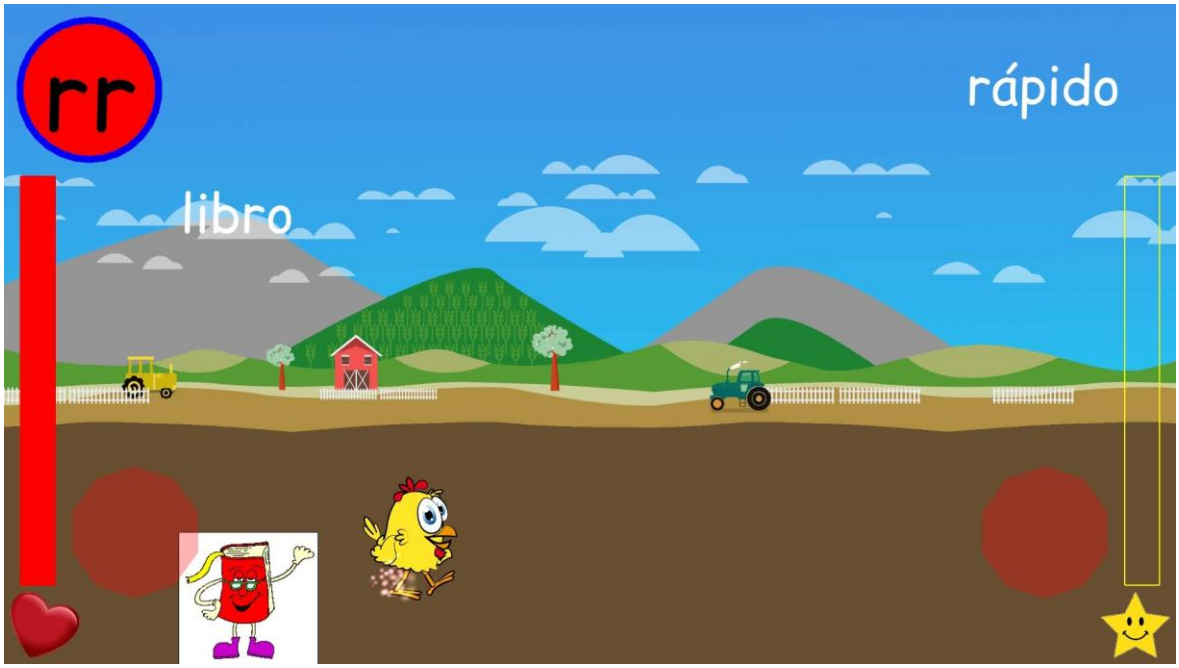

Figura C - 12: Ejemplo segundo minijuego Sanapalabras

Figura C - 13: Ejemplo instrucciones tercer minijuego Sanapalabras

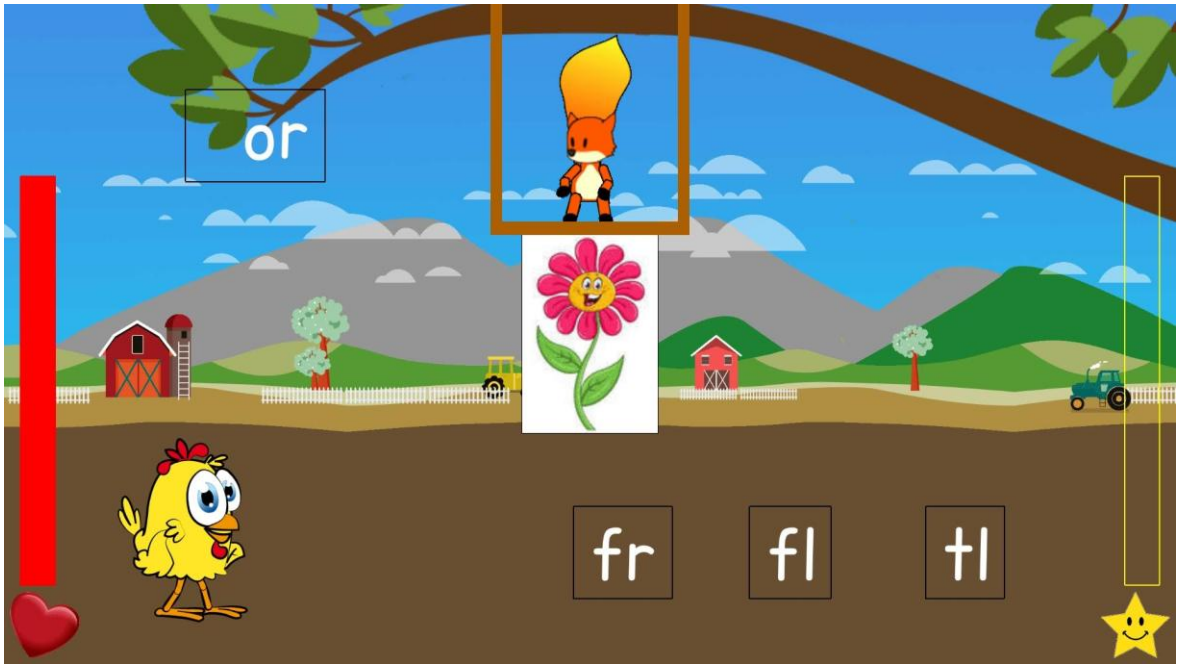

Figura C - 14: Ejemplo tercer minijuego Sanapalabras

Figura C - 15: Ejemplo instrucciones cuarto minijuego Sanapalabras

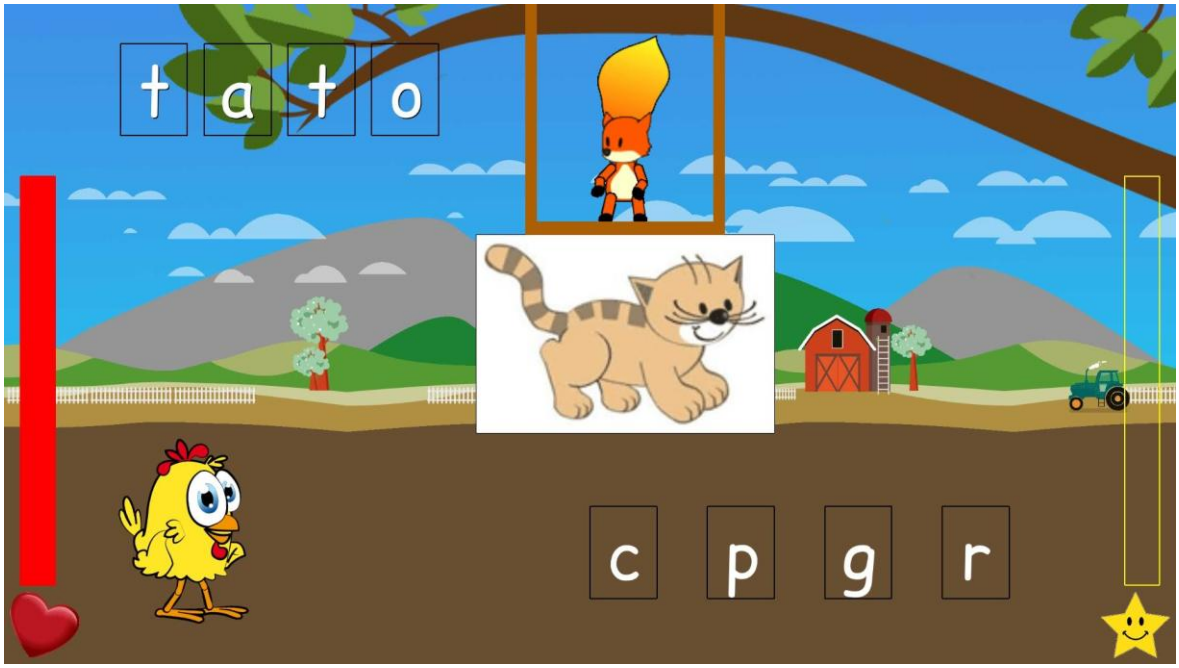

Figura C - 16: Ejemplo cuarto minijuego Sanapalabras

Figura C - 17: Estadísticas Sanapalabras

C Figuras de diagramas de clase y de flujo

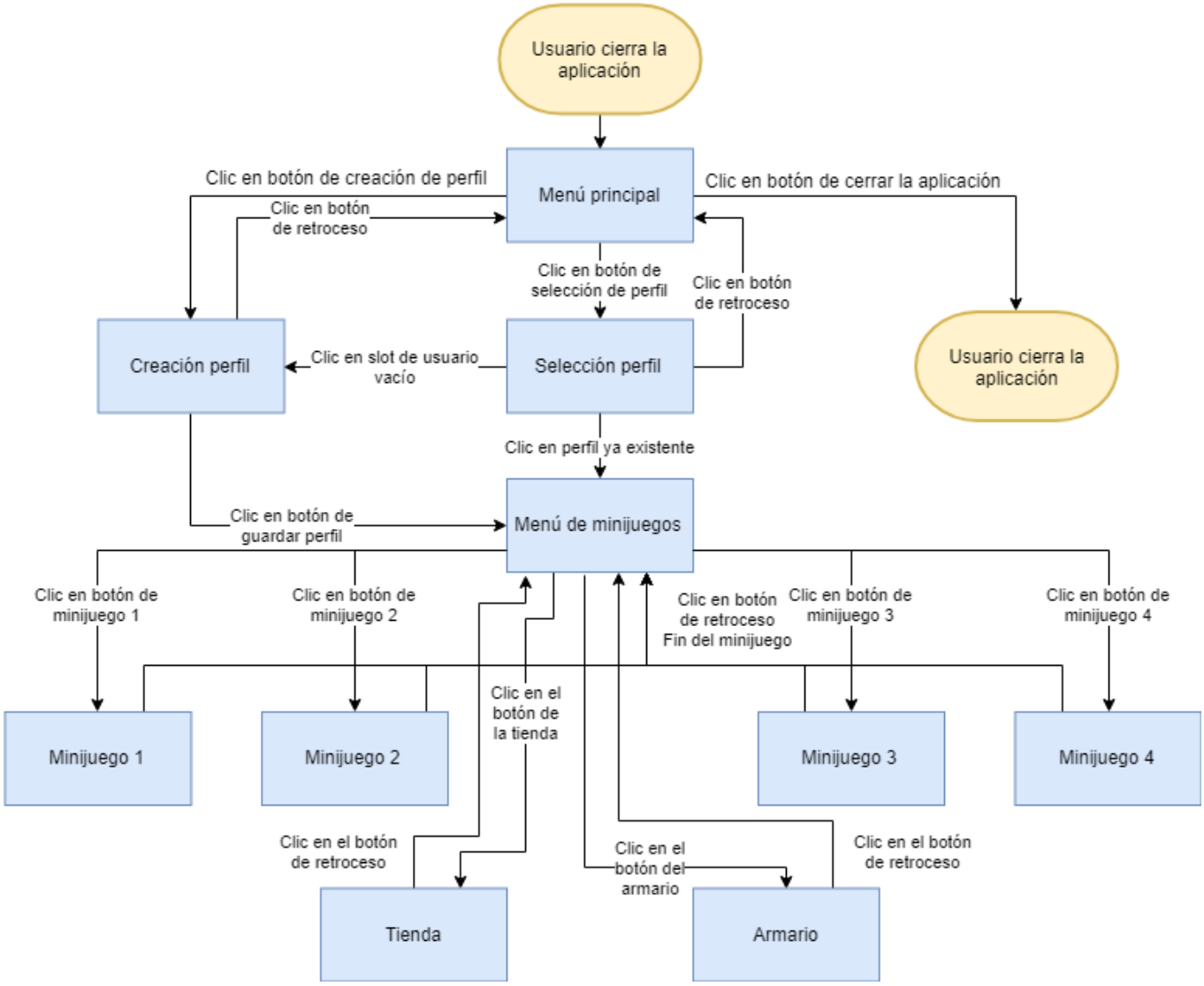

Figura 3 - 2: Diagrama de flujo vida de la aplicación

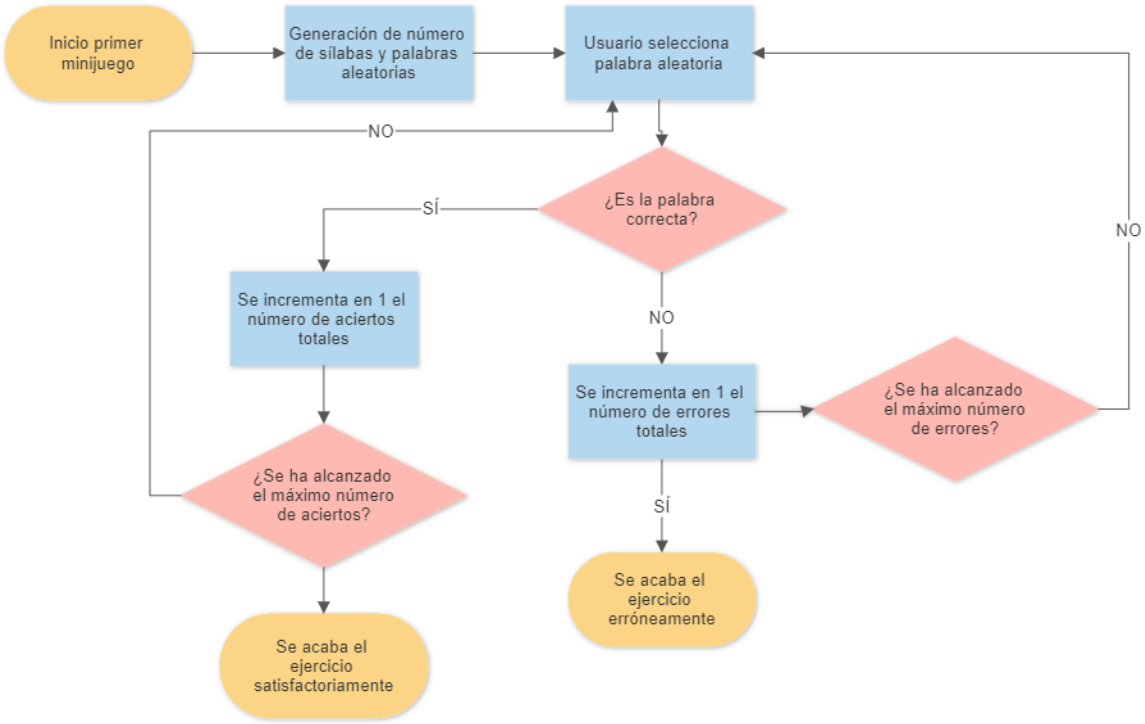

Figura 3 - 3: Diagrama de flujo primer minijuego

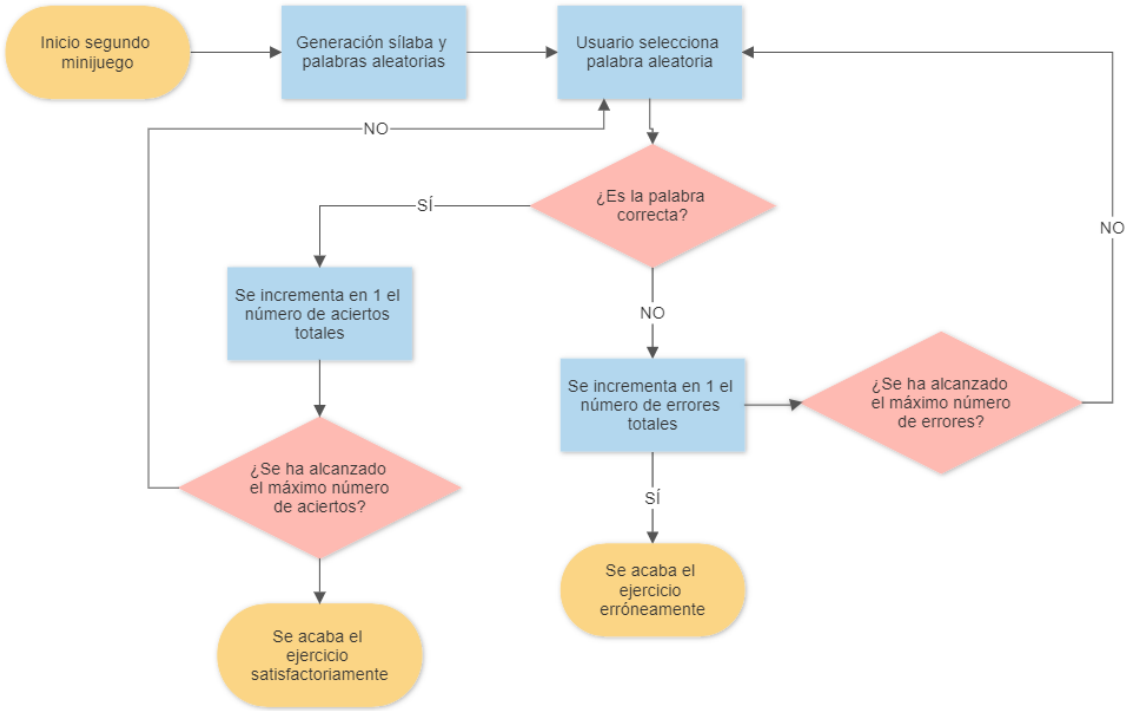

Figura 3 - 4: Diagrama de flujo segundo minijuego

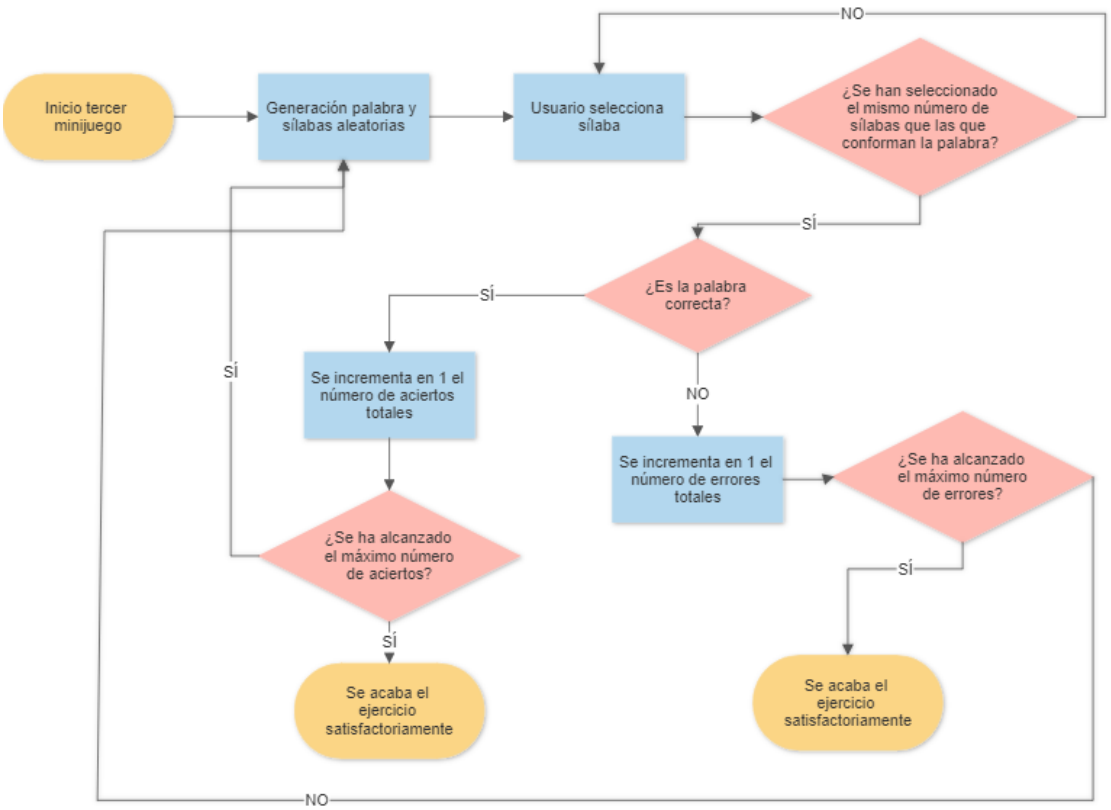

Figura 3 - 5: Diagrama de flujo tercer minijuego

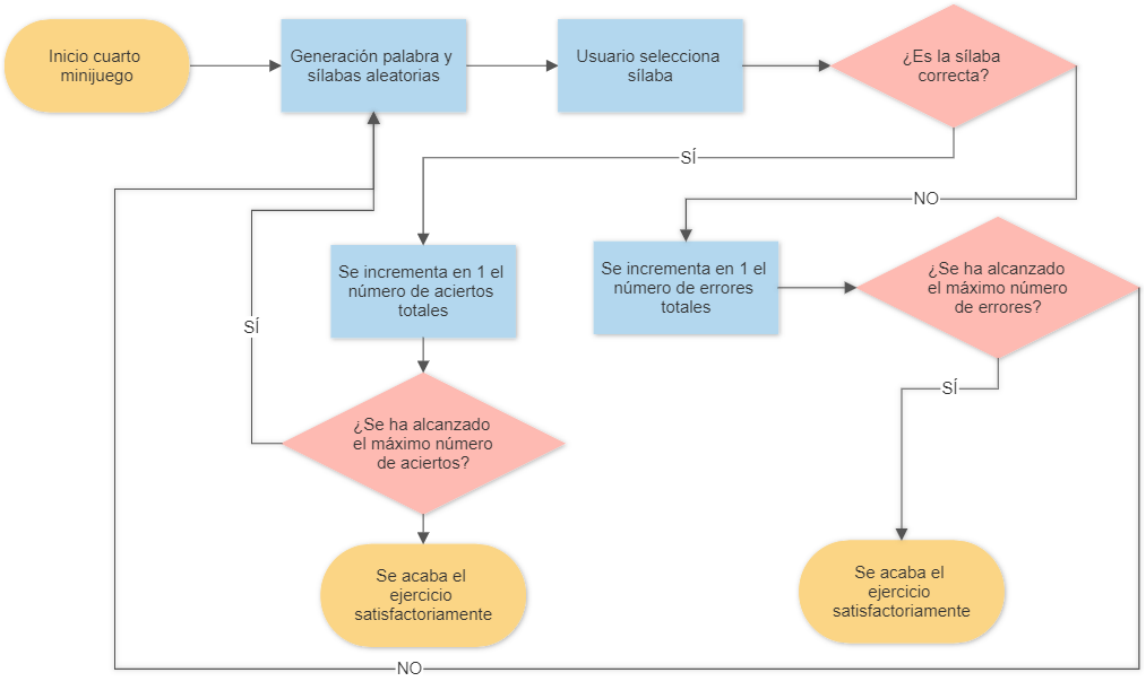

Figura 3 - 6: Diagrama de flujo cuarto minijuego

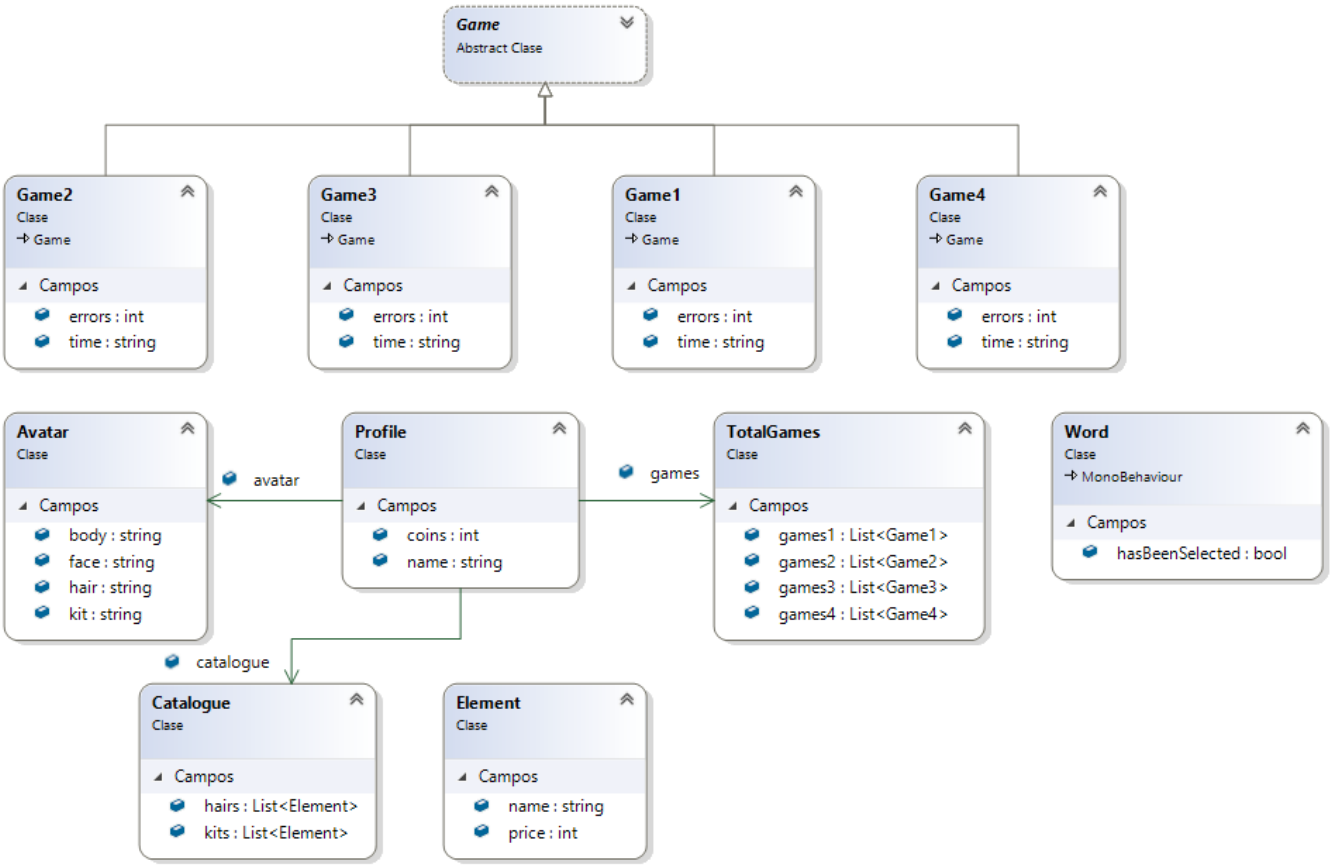

Figura 3 - 7: Diagrama de clases de la aplicación

D Figuras de código relevante

```
// Change the current scene
public void ChangeScene(string sceneName)
{
    SceneManager.LoadScene(sceneName);
}
```

Figura 4 - 16: Código función ChangeScene

```
private void JSONToFile(Profile profileAux, string directoryPath, string filePath)
{
    string json = JsonUtility.ToJson(profileAux);

    if (!Directory.Exists(directoryPath))
        Directory.CreateDirectory(directoryPath);

    if (File.Exists(filePath))
        File.Delete(filePath);

    StreamWriter writer = new StreamWriter(filePath, true);
    writer.Write(json);
    writer.Close();
}
```

Figura 4 - 17: Código función JSONToFile

```
public Profile ParseJSONFile(string filePath)
{
    if (!File.Exists(filePath))
        return null;

    string json = "";
    using (StreamReader sr = new StreamReader(filePath, Encoding.GetEncoding("iso-8859-1")))
    {
        string line;
        while ((line = sr.ReadLine()) != null)
        {
            json = line;
        }
    }

     return JsonUtility.FromJson<Profile>(json);
}
```

Figura 4 - 18: Código función ParseJSONFile

```
private void ChangeMiniAvatar(Transform profile, Profile profileAux)
{
    hairAux.GetComponent<Image>().sprite = Resources.Load<Sprite>(profileAux.avatar.hair);
    faceAux.GetComponent<Image>().sprite = Resources.Load<Sprite>(profileAux.avatar.face);
    bodyAux.GetComponent<Image>().sprite = Resources.Load<Sprite>(profileAux.avatar.body);
    kitAux.GetComponent<Image>().sprite = Resources.Load<Sprite>(profileAux.avatar.kit);
}
```

Figura 4 - 19: Código función ChangeMiniAvatar

```
[SerializeField]
private readonly float timerSpeed = 2f;

void Update () {
    elapsed += Time.deltaTime;
    if (elapsed >= timerSpeed)
    {
        elapsed = 0f;
```

Figura 4 - 20: Código función Update de la clase Timer

```
private void GenerateGameJSON()
{
    Estadisticas estadisticas = new Estadisticas();
    Game game = null;
    game = estadisticas.CreateGame1(minutes.GetComponentInChildren<Text>().text + ":" +
         seconds.GetComponentInChildren<Text>().text, int.Parse(incorrect.GetComponent<Text>().text));
    Perfiles profile = new Perfiles();
    profile.AddGameToProfile(game);
}
```

Figura 4 - 21: Código función GenerateGameJSON

```
private string CalculateAverageTime(List<string> times)
{
    var averageTime = times.Select(TimeSpan.Parse).Average(x => x.TotalMilliseconds);
    var averageTimeAux = TimeSpan.FromMilliseconds(averageTime);

    return averageTimeAux.ToString().Substring(0, averageTimeAux.ToString().Length - 3);
}

private int CalculateAverageErrors(List<int> errors)
{
    return (int) Mathf.Ceil(errors.Sum() / errors.Count);
}
```

Figura 4 - 22: Código funciones CalculateAverageTime y CalculateAverageErrors