App para aprender inglés de forma interactiva, enfocada al segundo ciclo de primaria.

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Eduardo Méndez Dunjó

TUTORIZADO POR:

Francisco José Santana Pérez

Fecha 06/2021

Agradecimientos

A mi familia, por apoyarme y dejarme ser desde siempre. Sobre todo, a mi madre por tener tanta paciencia conmigo y estar siempre dispuesta a ayudarme.

A cada profesor que me ha hecho interesarme por una asignatura y a toda la gente con la que me he cruzado en esta carrera, por hacer siempre más llevadera esta etapa.

A mi tutor Francisco Santana Pérez, por guiarme con el desarrollo de este proyecto y ayudarme cuando me encontraba más perdido.

Por último, a la gente que está ahí siempre. A Jorge por ayudarme con las naves espaciales, a Manuel por estar dispuesto a echarme un cable y gracias a Sofía, por ser tan especial y proporcionarme tanto apoyo.

Resumen

ENGAMES, nombre del producto desarrollado en la ejecución del proyecto, es una aplicación de entretenimiento y enseñanza destinada a niños de entre 8 y 10 años.

El objetivo de ésta es, mediante una serie de juegos interactivos, enseñar inglés de una forma llamativa y divertida.

Es necesaria una interfaz amigable e intuitiva, que resulte sencillo para cualquier niño de esta edad.

Angular es el *framework* que se ha utilizado para el desarrollo de esta app.

ENGAMES pretende relacionar el aprendizaje de un idioma con algo totalmente positivo. ¡Disfruta aprendiendo!

Abstract

ENGAMES is the name of this proyect. Is an entertaining and learning app, which target audience are kids from 8 to 10 years old.

The idea is teaching English in a fun and striking way, through interactive games.

This app pretends to have an intuitive and friendly interface, so every kid is able to have fun with it.

I have used Angular framework to develop this project.

ENGAMES aims to link learning a language to something totally positive. Enjoy learning!

Índice general

Índice imágenes

1. Introducción

Los niños absorben como una esponja la información, sin embargo, además de ser una tarea difícil, es crucial captar su atención para conseguir que se interesen por lo que se pretende enseñar, de modo que cuenten con la predisposición necesaria para ello. Por esto nació ENGAMES, una *WebApp responsive*, cuyo principal objetivo es enseñar vocabulario en inglés a niños, intentando entretenerlos mientras aprenden, consiguiendo que relacionen el aprendizaje con algo positivo.

Esta app se adapta a distintos tipos de pantalla para poder llegar a un público diverso, en igualmente diversas situaciones, dando versatilidad al producto.

La aplicación consta de 3 juegos distintos (nos referiremos a ellos como niveles), en cada uno hay imágenes y palabras que hay que relacionar de distinta forma según el nivel en el que se encuentre el usuario. Cada acción que se realice dentro de los juegos mostrará un cambio de color en el elemento en el que se haga click (verde si acierta y rojo si falla), permitiendo a los niños saber rápidamente si han respondido correctamente.

Consta también de una pantalla principal que te lleva a cada nivel y de una barra de navegación lateral que permite ir a la pantalla principal en cualquier momento y que tiene una funcionalidad de borrado de datos, para empezar ENGAMES desde cero.

En el *LateralNavBar* hay espacio para hacer un *display* del personaje personalizado por los usuarios y una forma de visitar su propio perfil para poder hacer uso de la funcionalidad de personalización del personaje. Se propuso un componente llamado perfil en el TFT01, que resultó implementado en la barra de navegación lateral, ya que el objetivo era tener forma de poder visualizar siempre al personaje, una vez hubieras pulsado en el diálogo y este hubiera desaparecido. A pesar de que la funcionalidad de personalización del personaje no estuvo contemplada en los documentos entregados inicialmente, puesto que surgieron durante el desarrollo del proyecto como mejora del mismo, se dejó planteada una idea de las funcionalidades que se pretendía que tuviera este componente.

Se ha utilizado Angular como *framework* para el desarrollo de esta aplicación por usar lenguajes muy extendidos como HTML y CSS y un lenguaje menos conocido, como es

TypeScript, pero muy útil por las facilidades que ofrece su CLI (Command Line Input) y por ser una tecnología muy utilizada por los desarrolladores de este ámbito. Como entorno de desarrollo he utilizado Visual Studio Code por tener un sistema de control de versiones integrado y por ofrecer un sistema de CLI para no tener necesidad de salir de la aplicación para realizar acciones tan comunes como la creación de componentes, servicios y el citado control de versiones.

2. Estado del arte

En este apartado se mostrarán algunas aplicaciones para aprender inglés, cuyo público objetivo son los niños. Las marcas debajo listadas ofrecen gran variedad de herramientas de aprendizaje, habiendo algunas que cuentan solamente con aplicaciones y otras, que cuentan con libros de ejercicios y lectura.

También se hablará de ENGAMES teniendo en cuenta toda esta información y haciendo comparaciones. Se listarán las fortalezas y debilidades de este proyecto, teniendo en cuenta la variedad del mercado.

2.1. Aplicaciones ya existentes

2.1.1. Poptropica English

A través de audios e imágenes, esta aplicación trata diferentes tipos de vocabulario. Se presenta una imagen y al darle al *play*, iremos escuchando las respuestas necesarias para completar el ejercicio, debiendo rellenarlo después.

Los puntos positivos que trata esta aplicación son:

- Tener una parte auditiva, permitiendo desarrollar el nivel de inglés del *Speaking* de los usuarios.

- Los audios se pueden parar y reanudar a voluntad.
- Disponen de distintos niveles de enseñanza de inglés, pudiendo alcanzar un mayor público.

Algunos puntos negativos que resaltar son:

- Al tratarse de un audio, no se sabe qué personaje dice cada cosa, por lo tanto, se cambia de voz cada vez que se dice una frase, dando por hecho así que cambia el personaje que está hablando, avanzando siempre linealmente en el escenario.
- No parece muy interactivo, teniendo que escuchar el audio si quieres obtener información.

Tiene más de 1 millón de descargas en la Play Store de Android, pero la marca, además, posee distintas apps, libros de ejercicios para el aprendizaje del inglés para alumnos, maestros y padres. Además, Poptropica English oferta diferentes niveles de dificultad en sus libros de ejercicios.

Ilustración 1. Imagen de Poptropica English.
La imagen muestra la apariencia de esta aplicación.

Esta imagen corresponde a un "nivel" de la aplicación Poptropica Islands. Se puede manejar el reproductor de audio libremente, dentro de las opciones permitidas. En este ejercicio se trabajan los ámbitos auditivo y oral.

2.1.2. Lingokids

Esta aplicación se dedica a enseñar a través de canciones y temas concretos, es decir, trata por ejemplo el tema juguetes y van saliendo en la pantalla de forma individual varios juguetes. Una vez han salido todos, comienza una canción que va hablando de los objetos uno a uno, además de mostrarlos visualmente.

Las fortalezas de esta aplicación son:

- Enseñar a través de canciones resulta muy útil, el factor pegadizo de las canciones, que normalmente nos aprendemos sin mucho esfuerzo, permite aunar la facilidad de aprendizaje de los niños con la facilidad de aprendizaje que otorga la música.
- Enseña *Reading* y *Speaking* a la vez, poniendo la letra de las canciones mientras van sonando, a modo de karaoke.

Las debilidades son:

- No parece haber demasiada interacción con el usuario, pareciéndose más a un show televisivo para la enseñanza del inglés, que a un juego.
- La versión de pago son 15 euros al mes, permitiendo el acceso a todo el contenido de la app. Pagar mensualmente una app que no se use mucho, no resulta muy atractivo.
- Poptropica parece estar sobre todo enfocado en escuelas, teniendo libros de actividades. Esto no es un punto negativo en sí, lo negativo es que su aplicación parece más bien eso, un libro digital con la posibilidad de escuchar un audio.

Lingokids está disponible para la AppStore y para la PlayStore, teniendo más de 10 millones de descargas en la Play Store.

Ilustración 2. Imagen de Lingokids.
La imagen muestra la apariencia de esta aplicación.

Esta imagen muestra cómo es un "nivel" en esta aplicación. El video transcurre nombrando diferentes juguetes en una canción.

2.1.3. Little Smart Planet

Esta app no se centra en el aprendizaje del inglés, sino que lo complementa a partes iguales con el refuerzo de las matemáticas y del español. Todo esto, usando pequeños videos musicales.

Lo más resaltable positivamente de esta app podría ser:

- El intento por enseñar una base mucho más amplia que el inglés a los usuarios.
- Tiene un periodo de prueba gratuita de 15 días.

No obstante, si hablamos de sus puntos negativos:

- Tenemos la gran cantidad de información que ofrece a la vez. Es decir, si el usuario no se centra en aprender los números, si no en fijarse en toda la historia detrás de éstos, podría resultar mucha información.
- Está enfocada para niños muy pequeños, pero los escenarios no parecen adecuarse a enseñar, más bien, parecen soltar mucha información sobre palabras, que los usuarios deberán conectar mentalmente a través de imágenes.

Little Smart Planet está disponible para la AppStore y para la PlayStore. Entre sus 2 aplicaciones en la Play Store, suma más de 11 mil descargas.

Ilustración 3. Imagen de Little Smart Planet.
La imagen muestra la apariencia de esta aplicación.

Esta es la apariencia de un "nivel" de la aplicación. Durante esta cinemática en la que se van enseñando los números, una voz va narrando una historia, cambiando junto con los números, las palabras que van apareciendo en la parte superior central de la pantalla.

2.2. ENGAMES

Para poder tener una idea general del proyecto, procederemos a explicar de forma básica la aplicación, ya que se profundiza en este tema a lo largo de la memoria.

ENGAMES es una aplicación con distintos niveles, 3 en total, cuyos nombres son THE GRID, CHOOSE IT! y LATERAL BUTTONS. El objetivo del usuario es ir completando cada nivel o juego, siempre pulsando en la respuesta que considere correcta. Cada nivel es diferente, tratando de evitar la monotonía.

La pantalla principal de esta aplicación, Adventure (Your Adventure Map), mostrada a continuación, es la encargada de mostrar, tantos los juegos disponibles, como el diálogo con nuestro protagonista Space George.

Ilustración 5. Imagen de la pantalla principal de ENGAMES

Ilustración 4. Imagen de la pantalla principal de ENGAMES

Las naves espaciales son los niveles disponibles. Para desbloquear más, el usuario debe completar los disponibles.

El personaje y el diálogo son los encargados de guiar al usuario a través de esta aplicación.

2.2.1. Fortalezas y debilidades de ENGAMES

Los puntos positivos de ENGAMES, teniendo en cuenta todo lo anterior sobre las aplicaciones ya existentes son:

- La interactividad de la aplicación. El usuario no debe sólo sentarse y mirar, debe interactuar con el juego para ir avanzando, convirtiéndolo en una forma de aprendizaje más entretenida a sólo escuchar y mirar.
- La facilidad de los niveles, permitiendo entender la situación y los objetivos de forma muy rápida e intuitiva.
- Se ofrece la información en cantidades pequeñas, intentando no saturar a los niños con un exceso de información.
- Relacionar mentalmente imágenes con sus palabras resulta super efectivo para el aprendizaje.
- Es un software gratuito, permitiendo que cualquier persona pueda beneficiarse del uso de este, enriqueciendo su nivel de inglés.
- Al tratarse de una *WebApp*, logra alcanzar un público muy grande, no se necesita descargar nada, sólo tener Internet.

Sin embargo, ENGAMES tiene también sus puntos negativos:

- El poco contenido impartido en la aplicación. Al tener sólo 3 niveles, el contenido de la app es escaso, esto significa poco material disponible para aprender, que puede derivar en el rápido aburrimiento de los usuarios al no tener una amplia variedad de juegos, que cumpla con sus expectativas.
- El contenido no está dividido en temas. Dividir el contenido en temas permite una mayor comprensión y relación mental de lo aprendido. Es más fácil aprender el nombre de diferentes objetos si todos están relacionados entre sí.
- Al no contar con una persona encargada del desarrollo gráfico de la aplicación, la parte visual del proyecto no tiene un aspecto muy profesional. Esto puede provocar rechazo por la aplicación a primera vista en el usuario potencial de ENGAMES.

No obstante, todas estas deficiencias son fácilmente superables en posibles versiones posteriores a la que nos encontramos actualmente.

3. Objetivos

Llegar a niños con problemas con el aprendizaje del inglés fue el primer objetivo del proyecto, por lo que se partía de la idea de plantear una app fácil de entender y manejar por cualquier niño, utilizando los medios gratuitos disponibles para el desarrollo de esta aplicación.

El mayor problema que tiene una app destinada a este tipo de usuarios es conseguir captar la atención de su público, resultándole entretenida. Una aplicación que no consiga resultar atractiva, por muy completa que sea y por mucho material didáctico que posea, no logrará hacer tanto impacto en los niños, conllevando que la usen poco o se cansen rápido de ella. Por este motivo, se concluyó que para solventar este problema habría de conseguirse una interfaz y unos juegos atractivos, *friendly* e intuitivos, con colores, dibujos y funcionalidades llamativas para el usuario, además de un *feedback* inmediato sobre las respuestas que se dan (si aciertan o si fallan).

Por otra parte, para llegar a una mayor cantidad de niños con este proyecto, se pensó en las facilidades que ofrece la web frente a las apps ofertadas sólo para los dispositivos Apple y Android, a través de sus tiendas virtuales. Es mucho más rápido entrar en una web que descargarse una app y, en muchos casos, esta es la primera opción contemplada. Esto implicó hacer una *WebApp responsive*, para que fuera cual fuera el tamaño de pantalla de la que dispusiera el usuario, pudiera hacer uso de ENGAMES.

Finalmente, la mejora de mis aptitudes en los distintos lenguajes utilizados en este proyecto, así como el aprendizaje de un *framework* muy utilizado y su lenguaje de programación, ha sido uno de los objetivos académicos de este TFT, dada la importancia de renovar y ampliar el conocimiento para ser un buen profesional en el mundo del desarrollo del software, un entorno muy cambiante.

4. Recursos necesarios

A continuación, se listarán los recursos necesitados para la realización de este Trabajo de Fin de Título.

4.1. Recursos Software

Angular

Angular es un *framework* creado por Google cuyo objetivo es facilitar la creación de aplicaciones web de tipo SPA (Single Page Application). Es una herramienta muy potente y completa, con un CLI (Command Line Input) muy útil. Está basado en componentes y es modular y escalable.

Visual Studio Code

Visual Studio Code es un IDE (Entorno de Desarrollo Integrado) desarrollado por Microsoft para Windows, Linux y macOS. Incluye un sistema de consolas para ejecutar comandos, soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código y refactorización de código.

Word

Microsoft Word es un procesador de texto desarrollado por Microsoft. Su fecha de lanzamiento fue el 25 de octubre de 1983, bajo el nombre Multi-Tool Word.

Excel

Microsoft Excel es una aplicación para hojas de cálculo, desarrollada por Microsoft. Cuenta con cálculo, herramientas gráficas, tablas calculares y un lenguaje de programación macro llamado Visual Basic para aplicaciones.

4.2. Tecnologías

Las tecnologías que utiliza Angular son:

HTML

El lenguaje de marcado de hipertexto (HTML) define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, entre otros.

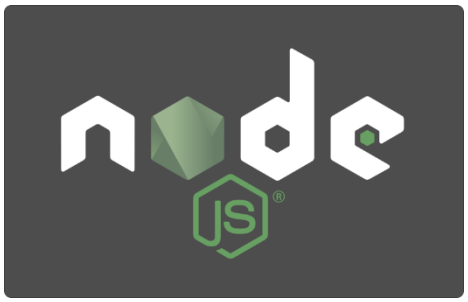
CSS

Hojas de Estilo en Cascada (CSS) es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML. CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel o en otros medios.

TypeScript

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. TypeScript es usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor, o extensiones para programas (Node.js y Deno).

NodeJS



Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.

Git

Es un software de control de versiones pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo de varias personas en un repositorio.

GitHub

Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en Ruby on Rails.

4.3. Recursos Hardware

Para la realización de este proyecto, he utilizado un ordenador de sobremesa y una pantalla, además de un ratón y teclado. Las características de este ordenador de sobremesa, montado por piezas, son:

- Un procesador Intel Core i7, con una frecuencia de 2,5 GHz.
- Una tarjeta gráfica NVIDIA GeForce GT 710 1 GB.
- Un disco duro interno de 1 TB de capacidad.
- Una memoria de 16 GB de RAM.

También he utilizado un ordenador portátil con un procesador Intel Core i5 de octava generación, 8 GB de memoria RAM y una tarjeta gráfica NVIDIA GEFORCE GTX, para los momentos en los que quería trabajar fuera de casa. El almacenamiento de este ordenador portátil está divido en 2, teniendo un disco sólido (SSD) de 128 GB de capacidad y un disco duro (HDD) de 1 TB de capacidad.

No es necesario un hardware demasiado potente para que Angular funcione, además, la app no es muy extensa, así que no se producen unos tiempos de carga desorbitados.

5. Análisis, Diseño y Desarrollo

Se ha utilizado el modelo de proceso evolutivo para el desarrollo de la aplicación ENGAMES.

"Este modelo de proceso para el desarrollo, explotación y mantenimiento de un producto software, está caracterizado por el cambio continuo. Es un modelo iterativo basado en el desarrollo de múltiples versiones, en el que cada versión incorpora las experiencias de las anteriores, con el objetivo de mejorarlas.

En cada iteración se ejecuta cada una de las fases del ciclo de vida del software. Una vez finalizada cada versión, se utiliza en paralelo con la anterior hasta que la última sea finalmente aceptada." (PDF 02. Casos de uso 2019 – IS1)

Se ha ejecutado cada fase del ciclo de vida en cada iteración y experimentado en cada una de las iteraciones con el contenido del proyecto, intentando así encontrar defectos y mejoras para el código.

Una vez identificados los requisitos del componente que se iba a llevar a cabo, se detallaba un diseño y se desarrollaba. A continuación, se comprobaba y testeaba el componente, si aparecía un fallo menor y fácilmente corregible, era resuelto.

Existen dos tipos de modelos de proceso evolutivo. Se ha seguido un enfoque utilizando prototipos ya que se ha trabajado con la versión disponible para ir mejorando el proyecto poco a poco. El objetivo de este enfoque es *"entender los requisitos del usuario y trabajar para mejorar la calidad de los requisitos. El prototipo ayuda a terminar de definir estos requisitos."* (PDF 02. Casos de uso 2019 – IS1)

5.1. Análisis

A pesar de seguir un modelo de proceso evolutivo en este proyecto, existió una fase de análisis, en la que se ideó el tipo de aplicación que se deseaba desarrollar, teniendo en cuenta lo ya existente en el mercado y las soluciones propuestas para intentar captar la atención del mayor número de usuarios posibles.

A través de la información sacada de distinto tipos de juegos, se decidió sobre el estilo que tendría ENGAMES. Es decir, qué funcionalidades debía tener y qué tipo de interfaz tendría.

Ayudar a niños a desarrollar su nivel de inglés desde edades tempranas, una etapa de vital importancia en su desarrollo mental, construyendo un entorno en el que se sintieran cómodos y se divirtieran, fue la idea que surgió tras este proceso de investigación. Además de la elección de *framework* para el desarrollo de esta aplicación.

5.1.1. Casos de uso

La primera vista que se ideó fue la de la pantalla principal (Adventure). A través de distintos juegos de rol de la Play Store, se tomaron ideas de la funcionalidad básica que debía tener Adventure.

Se eligió el diagrama de casos de uso para representar parte del contenido de este proyecto, debido a su componente visual. Es decir, es más fácil de entender a simple vista que otro tipo de diagramas.

En el siguiente diagrama de casos de uso, mostraremos las funcionalidades de la pantalla principal, parte esencial en el desarrollo del proyecto, debido a ser el nexo de los distintos niveles. Tras este diagrama se muestra la apariencia de: Adventure y de la barra de navegación lateral, respectivamente.

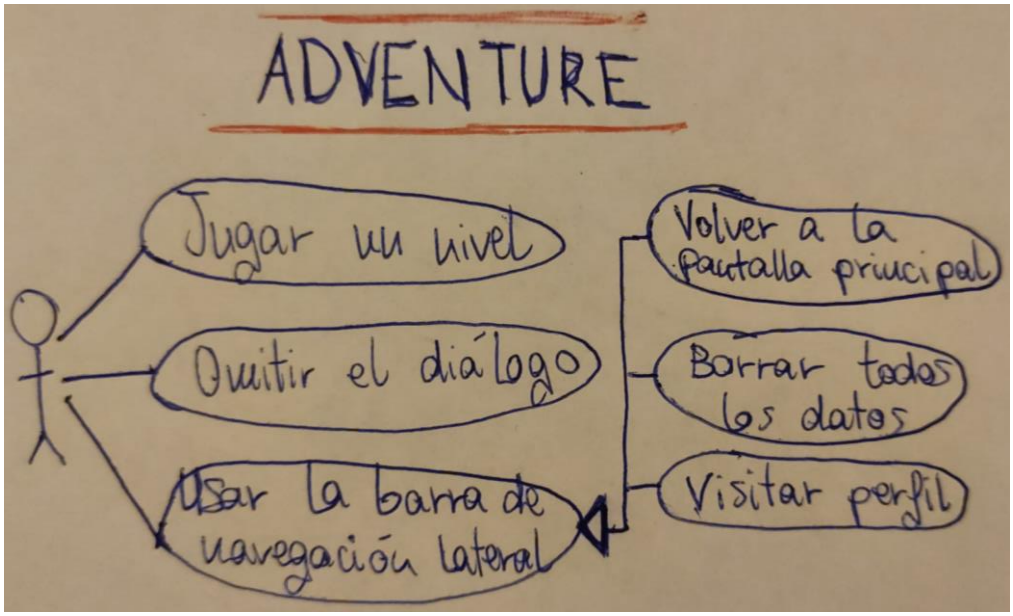


Ilustración 6. Casos de uso de Adventure

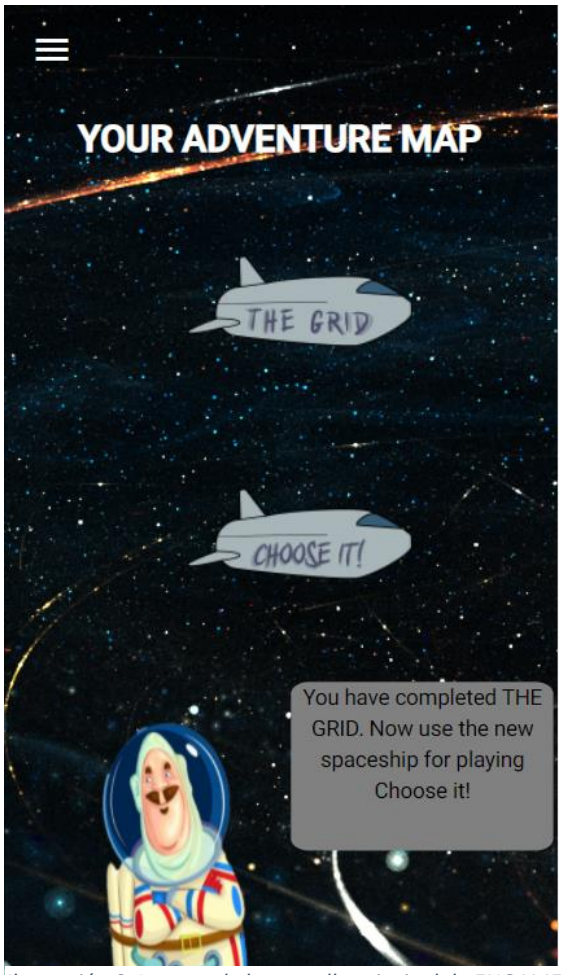


Ilustración 8. Imagen de la pantalla principal de ENGAMES

Ilustración 7. Imagen de la barra de navegación lateral de ENGAMES

Para explicar este diagrama de casos de uso, procederemos a hacer un *tour* por cada una de las funcionalidades de la pantalla principal.

Cada una de las naves espaciales de Adveture corresponde a un nivel. El usuario sólo debe pulsar en la nave deseada para transportarse al juego seleccionado. Una vez allí, pulsará un botón para comenzar el nivel. El juego transcurrirá normalmente hasta el momento en el que se acabe, en el que aparecerá una pantalla de fin de juego con un botón. Finalmente, si pulsas ese botón te encontrarás de nuevo en esta pantalla.

Si pulsamos en el icono de las tres rayas horizontales de la esquina superior izquierda, se desplegará una barra de navegación lateral. Este *LateralNavBar* incluye una forma de ir a la pantalla principal, un botón de borrado de progreso (para empezar de nuevo el juego) y un *display* del personaje, que te lleva a un lugar en el que poder modificar la apariencia del protagonista (componente en desarrollo).

Por último, Space George se comunicará con el usuario. Al pulsar en él o en el diálogo, ambos desaparecerán hasta que desbloqueemos el siguiente nivel.

Tras esto se hizo uso de StarUML (aplicación para realizar todo tipo de diagramas) para una mejor disposición y visualización de los casos de uso.

5.1.1.1. Adventure

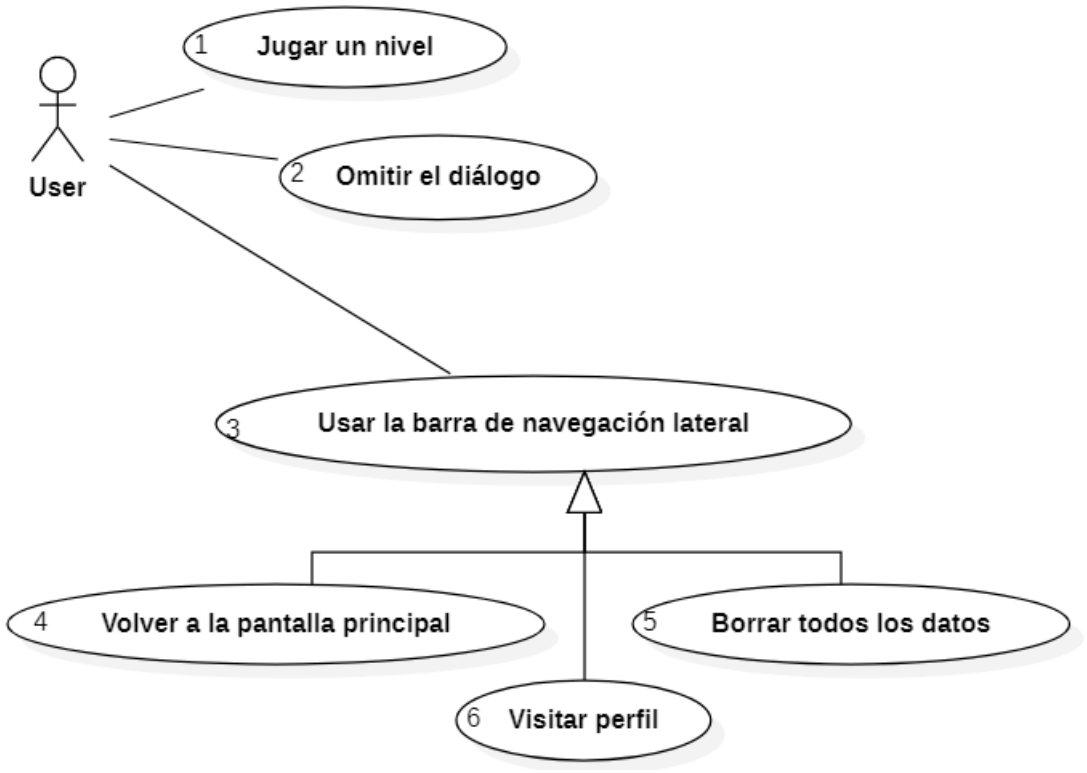


Ilustración 9. Casos de uso de Adventure

Este diagrama de casos de uso representa la funcionalidad que posee la pantalla principal (Componente Adventure).

Además, establecí el resto de los casos de uso. A continuación, mostramos como ejemplo los del segundo juego, CHOOSE IT!

5.1.1.2. Choose It!

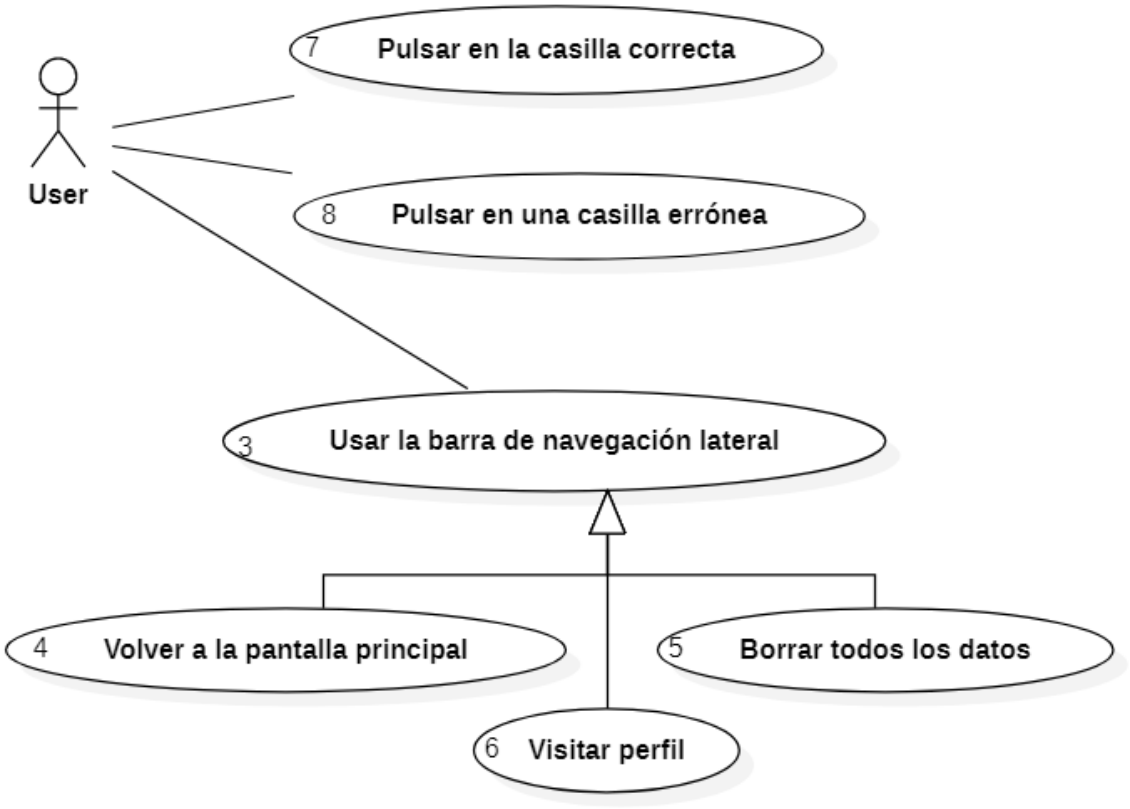


Ilustración 10. Casos de uso de ChooseIt

Como decíamos, en este diagrama se representan los casos de uso del segundo nivel. Existe una consecuencia distinta para los casos de uso 7 y 8 por parte de ENGAMES, cambiando a un color o a otro según el caso de uso (verde si se acierta y rojo si se falla).

5.1.2. Tablas de especificación de casos de uso

Realicé en Excel las tablas de especificación de los casos de uso correspondientes a cada diagrama. A continuación, mostraremos algunas tablas relevantes.

La siguiente tabla corresponde al caso de uso 01 “Jugar un nivel”, funcionalidad disponible en la pantalla principal. El desbloqueo de los niveles es lineal, es decir, para jugar al último nivel debes haber completado todos los niveles anteriores, mostrándose así como disponibles en la pantalla principal.

CU01	Jugar un nivel
Descripción	El usuario pulsa en la nave espacial deseada, que le permitirá jugar ese nivel.
Precondiciones	El usuario debe haber completado el nivel anterior, en el caso de tener, para que se muestre y sea seleccionable ese juego
Flujo normal	
1	El usuario pulsa en la nave espacial
2	El usuario pulsa en START para comenzar el juego
3	El usuario avanza a través del nivel pulsando las respuestas que considere correctas
4	El usuario pulsará un botón para continuar una vez finalizado el nivel. Este botón te transporta a la pantalla principal
Postcondiciones	Se desbloqueará un nivel nuevo, si es que lo hay. También, se mostrará un diálogo nuevo en la pantalla principal al completar el nivel

Ilustración 11. Tabla de especificación del caso de uso 01

La siguiente tabla de especificación de caso de uso es la correspondiente al CU04, disponible para todos los componentes de ENGAMES.

CU04	Volver a la pantalla principal
Descripción	Una forma de volver a la pantalla principal en cualquier momento
Precondiciones	La barra de navegación debe estar abierta
Flujo normal	
1	El usuario pulsa en la nave Home del *LateralNavBar* , estando en un juego
2	El usuario aparece en la pantalla principal
Variantes	
1a	El usuario pulsa en la nave espacial Home estando en el perfil
1b	El usuario pulsa en la nave espacial Home estando en la misma pantalla principal
Postcondiciones	La barra de navegación lateral se cerrará automáticamente

Ilustración 12. Tabla de especificación del caso de uso 04

5.2. Diseño

Para el diseño de ENGAMES, se buscó inspiración en distintos juegos que contuvieran una historia. Esta historia que cuentan te introduce en su universo, todos los detalles que poseen este tipo de aplicaciones, sumados a la apariencia cuidada y facilidad de la interfaz, te invitan a jugar.

De la observación de estos juegos se concluyó la importancia de contar, entre otras cosas, con una personalización de los niveles, una interfaz intuitiva y sencilla, así como la existencia de un personaje que te guíe a través de esta aventura. Todo ello se materializó en ENGAMES con naves espaciales que te llevan hasta tu nivel, en una simplicidad

(como sinónimo de intuitividad) a la hora de hablar de la interfaz, sin sobrecargarla de contenido, y el diálogo con tu guía.

Con respecto a la pantalla principal, los distintos niveles debían estar bien diferenciados, debía tener un tema y un fondo acordes y debía aparecer un personaje llamativo que introdujera al usuario en la aplicación.

El primer juego se decidió realizar con un sistema de comprobación de aciertos, que permitiría mostrar si el usuario ha dado con la respuesta correcta, debido a la distribución de imágenes de este nivel, los cambios de colores permanecen constantes hasta el final de este, para lograr un estado visual del avance del juego. Más tarde, se aplicaría este sistema de comprobación de aciertos en cada nivel.

Se muestra a continuación, y uno a uno, el resultado final de cada nivel, para poder entender con mayor claridad lo comentado.

Ilustración 13. Imagen del primer nivel: THE GRID

Podemos observar el sistema de comprobación de aciertos con las franjas de colores bajo cada imagen, además de un contador de aciertos y otro de fallos.

Iremos pulsando en la imagen que consideremos correcta, relacionando la palabra mostrada (Carrot en este ejemplo) con la imagen.

El segundo juego, mostrado a continuación, se diseñó para complementar el primer juego, basándose uno en las imágenes y el otro en el vocabulario.

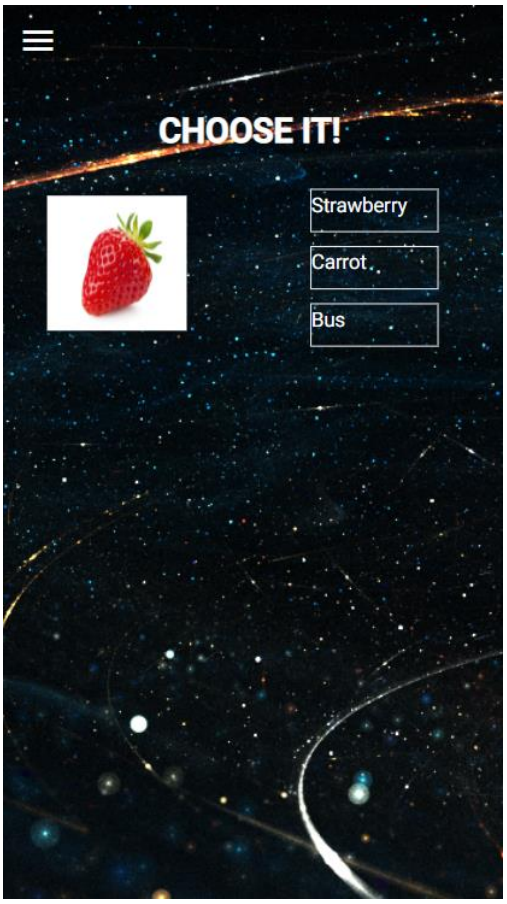


Ilustración 14. Imagen del segundo nivel: CHOOSE IT!

El usuario deberá pulsar en la casilla deseada, avanzando así a través de CHOOSE IT!

El sistema de comprobación de aciertos no se muestra en esta imagen debido al cambio de pregunta efectuado al pulsar en una de las 3 casillas disponibles.

El tercer juego comenzó siendo un *this or that*, una especie de juego en el que se debe elegir la opción preferida entre dos posibles, sin embargo, como no tenía tanto sentido esto, evolucionó al juego tal y como está actualmente, que trata de hacer una relación entre la palabra o imagen mostrada y la opción elegida. Se mostrará este nivel para mayor entendimiento.

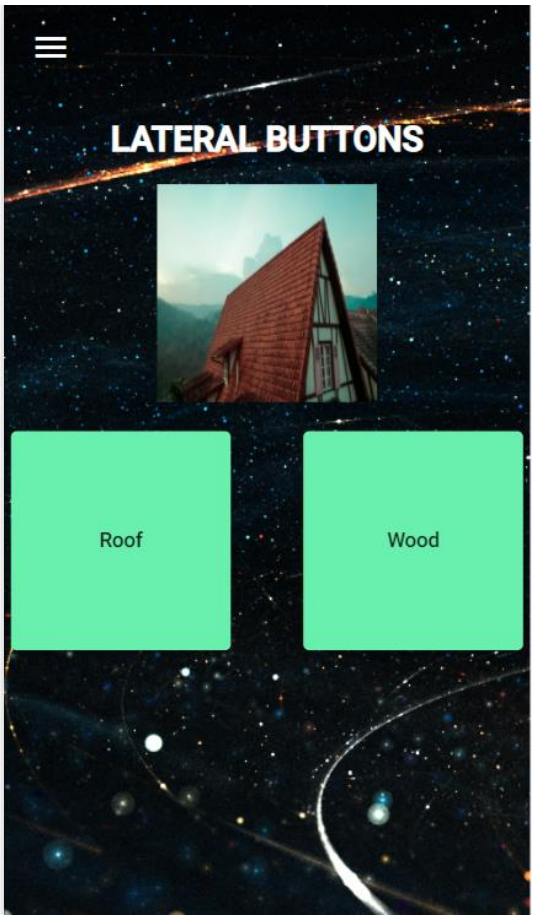


Ilustración 15. Imagen del tercer nivel: LATERAL BUTTONS

Para avanzar a través de este juego, el usuario debe ir pulsando en el botón que considere correcto.

El botón pulsado cambiará de color temporalmente a verde si el usuario acierta y a rojo si falla (sistema de comprobación de aciertos).

5.3. Desarrollo

El primer paso fue la instalación de todo lo necesario para el desarrollo de este proyecto. Una vez se tenían estas herramientas listas, se desarrolló y testeó el primer juego, el componente en el que se invirtió más tiempo, debido a la falta de conocimientos. Estos primeros pasos conforman la iteración 1.

Después de este comienzo, se desarrolla el segundo juego. La parte más difícil fue el sistema de comprobación de aciertos. Cuando el usuario pulsaba en una opción, el color cambiaba correctamente, pero o bien no se iba este cambio de color, o se iba pero ya no volvía a cambiar de color esta casilla pulsada.

Más tarde se desarrolló la pantalla principal, además de testear todo hasta el momento. Ya asegurado su funcionamiento básico, se tenían los cimientos de la aplicación. Al observarse que el contenido de ENGAMES era escaso (sólo estaban pensados 2 juegos al inicio del proyecto), se decidió añadir un tercer juego, este tercer nivel ocupa lugar en la siguiente iteración.

Tras todos estos componentes desarrollados, la pantalla principal comenzaba a no ser útil, ya que cada vez que se revisitaba un componente (cada vez que se cambiaba de un componente a otro), la información de las variables se perdía. Para lograr solucionar este problema y tras investigar soluciones, se hizo uso del *Local Storage*, un método que ofrece la web para almacenar cierta información. Tras comprobarse todos los escenarios posibles de la pantalla principal, se dio por concluida la parte de funcionalidad que correspondía a los niveles 1 y 2, agrupando los eventos mencionados en la iteración 2.

Seguimos con la programación y prueba del nivel 3, además de testear lo ya existente. También, se desarrolló una barra de navegación lateral (*LateralNavBar*) como forma de salir de los juegos y volver a la pantalla principal siempre que el usuario quisiera, además de probar, tanto cada componente con el ahora implementado *LateralNavBar*, como el nivel 3. El desarrollo de este componente ocupa la iteración número 3.

Lo siguiente en la lista fue profundizar en el aspecto visual y en la historia que se pretende contar, mediante el cambio de la apariencia de los accesos a los niveles y la elección de un *background* y un diálogo de un personaje que guiará al usuario a través de ENGAMES. Otra tarea a realizar fue seleccionar definitivamente el vocabulario a utilizar y las imágenes que aparecerían.

Siguiendo un tutorial de Angular, se sentaron las bases para obtener información de una base de datos. Sin embargo, actualmente se está utilizando una especie de base de datos *In-Memory* y el *Local Storage* para guardar toda la información necesaria para el desarrollo de la partida de cada usuario. A medida que el juego vaya creciendo en tamaño y complejidad, podría ser necesario el uso de una base de datos. El tema visual y la parte del desarrollo de un servicio *In-Memory* corresponden a la iteración 4.

Finalmente, se terminó de convertir la app en *responsive*, probando diferentes tamaños de pantalla de dispositivo en la pantalla principal y en cada nivel y cambiando de tamaño los elementos para que no ocuparan demasiado espacio en pantalla. Además de terminar de afinar todos los aspectos de la app, tanto estéticos como técnicos, concluyendo así con la última iteración, la número 5.

Node.js ha sido el motor gracias al que se ha podido ejecutar este proyecto. Esta tecnología actúa como servidor en el que se despliega la aplicación. Es un entorno de tiempo de ejecución en tiempo real. La ventaja más notable de Node.js para este proyecto,

es el consumo de menos memoria RAM que el resto de entornos de ejecución, pudiendo ejecutarse en dispositivos con unas especificaciones hardware limitadas.

5.3.1. Clase Adventure (Pantalla Principal)

A continuación, mostraremos y explicaremos el código en TypeScript del componente Adventure.

src > app > game1 > pages > adventure > adventure.component.ts > A

```
1   import { Component, OnInit } from '@angular/core';
2   import { Game1logic } from '../../../game1logic';
3   import { db } from '../../../db';
4   import { DbService } from "../../../db.service";
5
6
7   @Component({
8     selector: 'app-game1',
9     templateUrl: './adventure.component.html',
10    styleUrls: ['./adventure.component.scss'],
11    providers: [Game1logic, DbService]
12  })
13
14  export class AdventureComponent implements OnInit {
15    db: db[];
16
17    game1Finished: any;//local storage vars
18    game2Finished: any;
19
```

Ilustración 16. Imagen de la clase Adventure

La primera línea de código corresponde a unos *imports* de librerías *core* para Angular.

Los siguientes 3 *imports* se encargan de hacer una conexión con componentes o servicios existentes en este proyecto, pudiendo usar así las variables y los métodos de estos.

@Component tiene un selector que permite inyectar este componente en otra parte del código, también sirve para dar información sobre qué ficheros se encargarán de la parte visual e indica los elementos que inyectaremos en el constructor a través del provider.

Las líneas 15, 17 y 18 crean variables que utilizaremos más adelante. En Db guardaremos el contenido del servicio DbService. En las otras 2 guardaremos información sobre el nivel en el que se encuentra el usuario.

Ilustración 17. Imagen de la clase Adventure

En el constructor inyectamos el componente Game1Logic y el servicio DbService para poder usar sus variables y contenido.

A partir de la línea 22 comienza el método ngOnInit(), un método especial de Angular que se ejecuta cada vez que se visita o se hace uso de un componente.

Comenzamos este método con las dos variables creadas anteriormente, actualizando su valor con lo almacenado en el *Local Storage*, si es que hubiera algo guardado.

Con respecto al primer if(), como el avance en el contenido de ENGAMES es lineal, en el caso de que game2Finished tenga como valor “finished”, significaría que el usuario ha completado los 2 niveles anteriores, ya que al finalizar el segundo nivel, se guardó en el *Local Storage* esta información. Por lo tanto, deberemos cambiar a *true* el valor de la variable correspondiente al diálogo que queremos mostrar. Las variables game.text1,

game.text2 y game.text3 son las encargadas, en el archivo .html, de mostrar u ocultar los diálogos.

Continuamos con el else if(), preguntando si el primer juego ha sido completado y si es así, cambiamos el valor de la variable encargada del diálogo que queramos mostrar en este caso.

Por último, el else. Si ninguno de los juegos está completo, mostraremos el primer diálogo de todos.

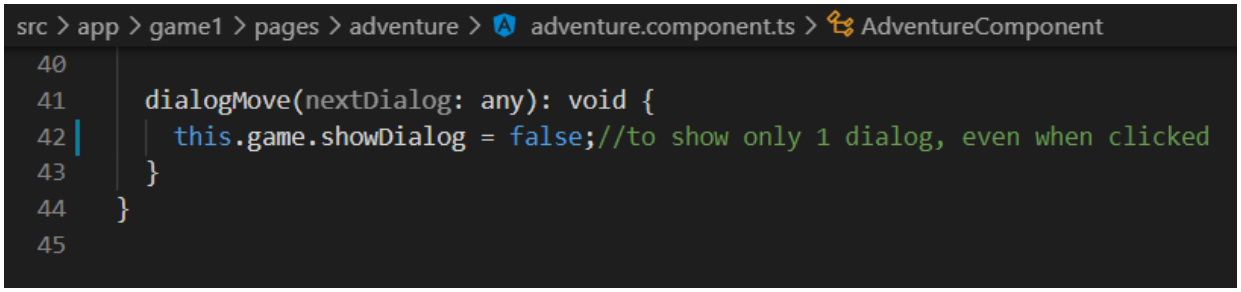


Ilustración 18. Imagen de la clase Adventure

El método dialogMove() desempeña la función de cambiar a *false* el valor de game.showDialog. La función dialogMove() es llamada cada vez que se pulsa en el diálogo, consiguiendo así ocultar el diálogo tras hacer click en él.

6. Temporalización

El objetivo de este apartado es esclarecer cómo se divide y en qué se ha gastado el tiempo disponible para este proyecto.

Para ello, se mostrará primero la tabla de temporalización aceptada por el tribunal, incluida en el documento TFT01 y después se comparará con la tabla resultante tras todo el desarrollo del proyecto.

Fases	***Duración Estimada*** ***(horas)***	***Tareas*** ***(nombre y descripción, obligatorio al menos una por fase)***
Estudio previo / Análisis	70	Tarea 1.1: Elección de tipo de *framework* de desarrollo y juegos que se implementarán
		Tarea 1.2: Familiarización con el entorno Angular (cursos y videos)
Diseño / Desarrollo / Implementación	120	Tarea 2.1: Implementación del primer juego
		Tarea 2.2: Implementación del segundo juego
		Tarea 2.3: Implementación de la pantalla de inicio y del perfil
Evaluación / Validación / Prueba	50	Tarea 3.1: Comprobar la implementación de la pantalla de inicio
		Tarea 3.2: Comprobar la implementación de los juegos
		Tarea 3.3: Comprobar la implementación del perfil
Documentación / Presentación	60	Tarea 4.1: Redacción de la memoria final del trabajo y realización de la presentación

Para la explicación de la segunda tabla, mostrada a continuación, dividiremos la primera fase “estudio previo y análisis”, en el visionado, estudio y puesta en práctica de:

- Videos y tutoriales de TypeScript en YouTube.
- Un curso de Coursera de familiarización con el *framework* de desarrollo Angular.
- La lectura de un libro sobre la enseñanza del inglés a niños y otro sobre TypeScript.

En esta segunda tabla, agruparemos estos eventos en la Tarea 1.1: Aprendizaje sobre el *framework*, los lenguajes a utilizar y el tipo de público al que va dirigida la aplicación.

Fases	***Duración Estimada (horas)***	***Tareas (nombre y descripción, obligatorio al menos una por fase)***
Estudio previo / Análisis	50	Tarea 1.1: Aprendizaje sobre el *framework*, los lenguajes a utilizar y el tipo de público al que va dirigida la aplicación
		Tarea 1.2: Elección sobre el tema y contenidos de la pantalla principal y los 2 primeros niveles
Iteración 1	30	Tarea 2.1: Instalación y creación de lo necesario para el proyecto
		Tarea 2.2: Implementación y testeo del primer juego
Iteración 2	55	Tarea 3.1: Implementación y testeo del segundo juego
		Tarea 3.2: Implementación y testeo de la pantalla principal
		Tarea 3.3: Implementación y testeo del *Local Storage en* la pantalla principal
Iteración 3	35	Tarea 4.1: Desarrollo y prueba del nivel 3
		Tarea 4.2: Desarrollo del *LateralNavBar*
Iteración 4	40	Tarea 5.1: Mejora de la parte visual del proyecto
		Tarea 5.2: Desarrollo del servicio *In-Memory*
Iteración 5	30	Tarea 6.1: Convertir la app en responsive
		Tarea 6.2: Ultimar detalles de ENGAMES
Documentación / Presentación	60	Tarea 7.1: Documentación de la memoria del TFT
		Tarea 7.2: Realización de una presentación para el proyecto

Como se puede observar, se consumió menos tiempo del esperado en la fase de estudio previo / Análisis.

Sumando las horas resultantes de todas las iteraciones, obtenemos un total de 190 horas, un tiempo mayor al estimado en el TFT01 para las fases de Diseño y Evaluación. Esto se debe a la falta de experiencia en la parte gráfica y visual. También, se consumió mas tiempo del esperado en la parte del contenido educativo incluido en ENGAMES.

Con respecto a la primera iteración, en la que se consiguió sólo el desarrollo del primer juego debido a ser la primera vez usando estas tecnologías, mencionar la lentitud de desarrollo.

En la segunda iteración podría decirse también que no se avanzó muy rápido, sin embargo, a partir de ahí, en las iteraciones posteriores empezó a notarse un cambio en la velocidad de programación y testeo de fallos.

El desarrollo del *LateralNavBar* dio algunos problemas, pero sobre todo los causó la implementación del servicio *In-Memory*. Como Angular avanza, lo hace TypeScript también, contribuyendo a que guías o tutoriales de Internet para realizar ciertas modificaciones se queden obsoletas, convirtiendo a veces la búsqueda de información en una ardua tarea.

Por último, mencionar sobre la temporalización que, la parte *responsive* de la aplicación, en la iteración 5, no tomó mucho tiempo, ya que se intentó trabajar en la parte visual siempre con tecnología *FLEX*, centrando así los elementos y la disposición de la pantalla. En esta iteración hubo que modificar la posición de ciertos elementos que no se había logrado modificar con anterioridad.

7. Prueba y Mantenimiento

Los distintos componentes y servicios del proyecto han sido probados durante y a posteriori de su implementación, comprobando todas las posibilidades del flujo de cada uno de ellos.

Durante la fase de implementación se le dio prioridad a arreglar los fallos más importantes, por ejemplo, el sistema de comprobación de aciertos (verde si se acierta y rojo si se falla), que fue una de las funcionalidades que más problemas produjo, junto con la implementación del *LateralNavBar* y el sistema *In-Memory*, además del uso del *Local Storage* en la pantalla inicial.

Una vez se finalizaron las funcionalidades básicas, se le dio un papel principal a intentar hacer que fallara la app, cambiando el contenido o saliendo del juego antes de acabar el nivel.

Con el tiempo y el uso de la aplicación, se pueden descubrir fallos o mejoras posibles de ENGAMES. Un buen mantenimiento de la aplicación sería hacer una corrección de los fallos a medida que vayan surgiendo, además de conseguir una mejora de las capacidades y utilidades que ofrece.

7.1. Caso real práctico

Se llegó a probar la aplicación con niños, el público objetivo de ENGAMES. Gracias a esto, se detectaron fallos o mejoras que a lo mejor nunca se hubiesen llegado a descubrir.

Contacté con un conocido que en ese momento estaba realizando las prácticas de magisterio. Tras muchas conversaciones y demostraciones, accedió a hablar con su profesor al cargo sobre el tema y, a este, pareció agradarle la idea, ya que accedió a que se hiciera una prueba con algunos de los alumnos.

También se probó la aplicación con un niño de 8 años y una niña de 10. Estos dos participaron en más de una ocasión, ofreciendo mucha información muy útil para el desarrollo del proyecto.

Todo el *feedback* generado con estas pruebas resultó muy útil para mejorar la aplicación. Las conclusiones a las que se llegó, sobre los cambios o funciones que se debía desarrollar, con todas estas pruebas realizadas son:

- La importancia del tamaño de los botones para las pantallas del tipo *Tablet* (debían tener un tamaño mayor).
- El sistema de comprobación de aciertos (cambio de color cuando se acierta y se falla), debido a que parecía que los niños no sabían si lo habían hecho bien o no.
- El desarrollo del *LateralNavBar*, ya que una vez habían completado ENGAMES parecían querer volver a la pantalla principal una vez estaban en un nivel.
- Resultó esencial el uso del *Local Storage*.
- Finalmente, se sacó en claro la duración mínima que debían tener los niveles.

8. Análisis de Costos

Procederemos a nombrar y especificar el coste del desarrollo del proyecto ENGAMES, teniendo en cuenta distintas categorías.

Para calcular el coste que ha supuesto el proyecto, distinguiremos entre distintos tipos de costes.

8.1. Costes de hardware

El valor del hardware utilizado en el desarrollo del proyecto será dividido en:

Hardware	Cantidad	Precio
Ordenador (sobremesa)	1	950€
Monitor	1	140€
Teclado	1	12€
Ratón	1	8€
Ordenador (portátil)	1	650€
Total		**1760€**

8.2. Costes software

Se ha utilizado el paquete de Office 365, proporcionado por la ULPGC a los estudiantes. Todo el software utilizado hasta el momento es gratuito, sin embargo, en el caso de necesitar una base de datos para seguir ampliando la app, se ha elegido MongoDB por su buen rendimiento y fácil escalabilidad. En la siguiente tabla se representa el contenido de un pack de pago de MongoDB, en el que se ofrece, por un precio mensual:

Característica (MongoDB)	Cantidad ofertada
Almacenamiento	40 GB
RAM	8 GB
vCPUs	2 vCPUs
Precio total	**362,88 euros al mes**

Estimando un tiempo de desarrollo de 3 meses, el precio total del alquiler de una base de datos de MondoDB ascendería a **1088,64€**.

8.3. Coste de humano

Se ha analizado cuánto costaría contratar un desarrollador *Junior* para la realización del proyecto. Se estimará un tiempo de desarrollo de 3 meses para esto.

Dividiremos el trabajo en 2 partes, por un lado, el desempeño como programador y por otro, la función del desarrollador gráfico, que tiene un perfil totalmente distinto al del programador. El salario de un desarrollador *Junior* medio es de 1200 euros al mes. Teniendo en cuenta esto, y proporcionando a las dos funciones a desempeñar un sueldo equivalente, la siguiente tabla mostrará el tiempo desempeñado en cada función del desarrollo.

Función	Meses de trabajo desempeñados
Desarrollador gráfico	1 mes
Programador	2 meses
Total	**3600€**

La función de desarrollador gráfico incluye las tareas visuales del proyecto y también las de encargado de contenido, historia y tema de ENGAMES.

La parte de programador es la encargada de toda la parte de las funcionalidades que ofrece esta aplicación.

8.4. Extras

Entendiendo que debería existir un beneficio inicial en el desarrollo del proyecto, se ha estimado la cantidad de 1500€ en concepto de la realización en solitario de la app, además de la idea, los objetivos y el estudio que ha supuesto la realización del proyecto.

8.5. Desglose de costes

La siguiente tabla aúna todos los costes detallados en los anteriores apartados, especificando el precio total del desarrollo completo de la aplicación.

Tipo de coste	Costes
Costes hardware	1760€
Costes software	1088,64€
Salario	3600€
Extras	1500€
Precio total	**7948,64€**

En conclusión, estimando un tiempo de desarrollo de 3 meses, el precio total del proyecto ascendería a **7948,64€**.

9. Modelo de negocio

Con respecto a ENGAMES, existen dos posibilidades:

- Vender ENGAMES es la primera opción. En el caso de que alguna entidad o persona compre la app, deberán ocuparse de la mejora y mantenimiento de esta, a no ser que se llegue a algún tipo de acuerdo para continuar con mi presencia en este proyecto.
- La otra posibilidad sería mantener la aplicación y seguir investigando y mejorando mis conocimientos para lograr aumentar en gran medida el valor añadido de esta. Ocupándome también del mantenimiento de ENGAMES.

Se barajaron las siguientes posibilidades para rentabilizar el desarrollo de esta aplicación:

- Poner anuncios y una versión premium sin anuncios, recibiendo ingresos por ambas partes.

- Crear un sistema de apariencia para el personaje y cobrar por los distintos atuendos que podría tener el protagonista de la historia.
- Venderlo a algún colegio o profesores particulares de inglés por cierta cantidad mensual.
- Poner anuncios voluntarios como método para conseguir atuendos u otro tipo de recompensa.

El último, es el método de uso de anuncios mejor visto por los usuarios y que más beneficios genera en aplicaciones, siendo además el menos intrusivo.

Tras finalizar un nivel, el usuario recibiría una pieza para personalizar al personaje, existiendo distintas posibilidades de uso para el sistema de recompensa por visionado de anuncio:

- La posibilidad de cambiar la recompensa conseguida.
- Ganar una tirada en una ruleta de recompensas para cambiar el icono del jugador.
- Recibir una pieza aleatoria de aspecto del personaje extra.

El objetivo de estos anuncios con recompensas es que el usuario se sienta recompensado por ver un anuncio y no piense que está perdiendo el tiempo. El usuario tiene total poder sobre los anuncios y si no desea verlos, no lo hará.

10. Conclusiones

ENGAMES ha sido un proyecto muy interesante. Sobre todo, resaltar el valor de lo aprendido sobre Angular y los lenguajes utilizados, más específicamente TypeScript. Este proyecto consiguió inculcar la constancia de seguir buscando y probando, aun cuando no se encuentra la respuesta. Por otra parte, ha resultado vital para entender la importancia que tiene el aprendizaje de un idioma tan extendido como el inglés, a una edad temprana, así como saber el tipo de aplicaciones que oferta este mercado.

Al principio del desarrollo, al no tener un tema o ambiente elegidos y por mi escasez de conocimientos en HTML y CSS, era común la desconfianza sobre lograr una interfaz amigable y atractiva. Sin embargo, podemos considerarlo un buen resultado final, sobre

todo si hablamos de la funcionalidad de cada componente, dado que TypeScript era un lenguaje de programación en el que no tenía experiencia.

Para el desarrollo de este proyecto se han utilizado muchos de los contenidos aprendidos en el transcurso del grado. Se ha dado uso a técnicas y habilidades adquiridas en esta etapa. El paso por esta carrera ha determinado también la importancia de la búsqueda de información de forma autodidacta y el uso de *clean code* con el fin de facilitar al máximo la comprensión del proyecto.

Hay distintos perfiles necesarios en el desarrollo de una aplicación. Se ha tenido que asumir muchos de ellos sin experiencia previa, como podría ser el papel de diseñador gráfico o el de "profesor", que de alguna manera se han desempeñado. Esto se traduce en un mayor tiempo invertido en esas tareas del previsto inicialmente.

Esta aplicación ha permitido analizar aspectos que no había tenido en cuenta anteriormente. Por ejemplo, el análisis de costos, idear una forma de monetizar la aplicación… Este tipo de cosas pasan por alto cuando alguien te contrata sólo para realizar una parte del proyecto.

Poner en prueba real este proyecto ha servido para obtener un *feedback* sobre los cambios necesarios. Uno puede hacerse una idea general sobre las necesidades de la aplicación, pero como el público objetivo son los niños, es necesario que ellos mismos valoren la aplicación y propongan cambios posibles para mejorar ENGAMES

También, en el caso de mantener el proyecto funcionando, habría que estudiar un plan de mantenimiento para poder arreglar los fallos e ir actualizando poco a poco la aplicación para que el usuario no sufra las consecuencias de un software obsoleto.

A pesar de esto, se debe considerar que hay varios aspectos a mejorar que hubiese estado bien poder abarcar, pero no se pudieron llevar a cabo debido a la falta de tiempo.

11. Trabajos futuros

Al principio del proyecto se habían barajado menos elementos de los incluidos actualmente, por tanto, se ha aumentado la funcionalidad y el número de niveles con respecto a lo estimado. Sin embargo, a medida que avanzaba el proyecto, aumentaba el número de ideas para mejorar ENGAMES.

El resultado es mejor al pensado al principio, pero existen muchas mejoras que aplicar para hacer más profesional esta aplicación. Dividiremos las mejoras en tres tipos: gráficas/visuales, de software y del contenido impartido.

11.1. Mejoras gráficas o visuales

A continuación, se expondrá una lista de las mejoras gráficas o visuales pensadas para este proyecto:

- Mejorar o cambiar gráficamente todas las imágenes, utilizando un mismo estilo de dibujo para todos los elementos del proyecto.
- También es importante mejorar visualmente todo lo existente, es decir, decorar y embellecer los juegos y los elementos de cada componente para conseguir un aspecto mucho más profesional.

11.2. Mejoras de software

Se hará una lista de las mejoras a nivel de software que podría presentar ENGAMES en un futuro:

- Para empezar, la mejora más importante a realizar es un sistema de recompensas por niveles, premiando al usuario cada vez que completa un nivel, otorgándole una prenda de vestir o un accesorio que colocar al personaje de esta aventura.
- Para que la funcionalidad anterior tenga sentido, se debe implementar una vista perfil, en la que poder intercambiar las distintas apariencias del protagonista de ENGAMES a voluntad.

- Acompañando a las anteriores mejoras, se debe ampliar el surtido de niveles, aumentando el número de niveles disponibles y la diversidad de los juegos.
- Hacer versiones para incorporar la aplicación a la PlayStore y la AppleStore para alcanzar aún más público.
- En el momento en el que la aplicación crezca de esa manera, hará falta hacer conexiones con bases de datos para tratar toda la información, necesitando de su creación, conexión y mantenimiento. MongoDB es una base de datos muy utilizada y con una buena fama. Es por esto que, en el caso de necesitar este tipo de servicios, se utilizaría esta base de datos.
- Implementar una forma de avanzar a través del juego según el desempeño del usuario. Esto se traduce en, según el conocimiento demostrado por cada usuario, permitir avanzar más rápido a través de la aplicación para evitar el aburrimiento y la repetitividad en el caso de tratarse de niños con un nivel de inglés mayor a la media.
- Para concluir, me gustaría añadir un apoyo auditivo del contenido de la aplicación, para que los usuarios puedan aprender la pronunciación de cada palabra o diálogo que aparezca en ENGAMES y además practicar su capacidad de *Listening*.

11.3. Mejoras del contenido impartido

Se mostrará una lista de posibles mejoras para el contenido impartido, aumentando así el conocimiento adquirido por los usuarios:

- Aumentar el contenido de cada nivel para alcanzar un mayor conocimiento de vocabulario gracias a la aplicación, consiguiendo que aprendan y se diviertan más.
- Existe también la posibilidad de añadir otros idiomas para que los usuarios aprendan vocabulario básico de más lenguajes.
- Podría incluirse otro tipo de juegos, para desarrollar aspectos como la memoria, la abstracción o el espacio en los niños.
- Otra posible mejora sería tematizar los niveles, aprendiendo vocabulario específico sobre ciertos temas, ayudándoles a hacer una mejor asociación de palabras y consiguiendo un mayor avance en el aprendizaje del usuario.

12. Referencias bibliográficas

A continuación, y dividida en dos categorías, se muestra la bibliografía utilizada para el desarrollo de este proyecto. Tras estas dos categorías, encontraremos las competencias adquiridas.

12.1. Bibliografía de libros

[1] Scott, W. A., & Ytreberg, L. H. (1995). *Lklt: Teaching Eng Children Scott*. Pearson Education ESL. http://www.cje.ids.czest.pl/biblioteka/6940128-Teaching-English-To-Children.pdf

[2] Serrano, C. (2020). TypeScript*: Curso Práctico*. Editorial Ra-Ma.

12.2. Webgrafía

[1] *Angular Crash Course (2019)*. (2019, 23 enero). [Vídeo]. YouTube. https://www.youtube.com/watch?v=Fdf5aTYRW0E&ab_channel=TraversyMedia

[2] *Angular Tutorial for Beginners: Learn Angular &* TypeScript. (2017, 5 septiembre). [Vídeo]. YouTube. https://www.youtube.com/watch?v=k5E2AVpwsko&ab_channel=ProgrammingwithMosh

[3] *LearnHow: Make a CYOA Game using HTML & CSS*. (2016, 3 octubre). [Vídeo]. YouTube. https://www.youtube.com/watch?v=8iNzUoUyuis&ab_channel=LearnHow

[4] Muppala, J. K. (s. f.). *Front-End JavaScript Frameworks: Angular*. Coursera. Recuperado 2020, de https://www.coursera.org/learn/angular

[5] Jaquez, R. (2020, 27 agosto). *Simple Sliding Side Bar for your Angular Web Apps - ITNEXT*. Medium. https://itnext.io/simple-sliding-side-bar-for-your-angular-web-apps-d54fef7c1654

[6] *Angular - get data from a server*. (s. f.). Angular. Recuperado abril de 2021, de https://angular.io/tutorial/toh-pt6

[7] *Aumenta la productividad en el trabajo*. (s. f.). Google Actívate. Recuperado 2021, de https://learndigital.withgoogle.com/activate/course/increase-productivity

[8] *Attention Required! | Cloudflare*. (s. f.). Pexels. Recuperado 2021, de https://www.pexels.com/es-es/

[9] *Material design*. (s. f.). Material design. Recuperado 2021, de https://material.io/

[10] *TOYS for KIDS 🧸 VOCABULARY, SONGS and GAMES | Lingokids*. (2021, 31 mayo). [Vídeo]. YouTube. https://www.youtube.com/watch?v=TqYFkVtNlnQ&ab_channel=LingokidsSongsandPlaylearning

[11] *ACTIVITY 1 - primero básico*. (2020, 29 marzo). [Vídeo]. YouTube. https://www.youtube.com/watch?v=7OWRn0Anohg&ab_channel=MissDayanaGallardo-LearningEnglish

[12] *Little Smart Planet | The Numbers | Cartoons & Baby Songs | The Short Story*. (2020, 27 julio). [Vídeo]. YouTube. https://www.youtube.com/watch?v=cS72jZL23Z4&ab_channel=Lolipapi

12.3. Competencias

Las siguientes competencias están extraídas de la web de la Escuela de Ingeniería Informática según referencias del Grado en Ingeniería Informática (Plan 2011), de la Universidad de Las Palmas de Gran Canaria.

12.3.1. Competencias Comunes a la Ingeniería Informática (CII)

- CII07: "Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema."

- CII08: "Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados."

12.3.1. Competencias de la rama Ingeniería del Software (IS)

- IS01: "Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software."

- IS03: “Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.”

- IS04: “Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.”

13. Anexo I: Manual de usuario de la aplicación

Bienvenido al manual de usuario.

Aquí se explicará cómo utilizar la aplicación para guiarte a través de la aventura didáctica que es ENGAMES.

Se hará uso de la versión móvil para el manual de usuario debido a que posibilita una mejor visión de la interfaz por su ancho reducido.

Ilustración 19. Imagen de la pantalla principal de ENGAMES

Nos encontramos en el espacio, donde nos recibe Space George, el personaje que guiará a los niños a través de los distintos niveles de este proyecto. Este se presentará y nos animará a coger la primera nave espacial para transportarnos a THE GRID, el primer juego de esta aventura.

Ilustración 20. Imagen de la pantalla principal de ENGAMES

La pantalla principal posee tres naves espaciales personalizadas. Cada una de ellas te lleva a un nivel distinto y todas las naves tienen una animación de movimiento al pasar el cursor por encima.

Los niveles se irán desbloqueando a medida que el usuario vaya completando cada uno de los distintos juegos disponibles.

Al pulsar en la primera nave, Space George te transportará al juego THE GRID, en el que debes ir seleccionando casillas según la palabra que debas relacionar. El objetivo es pulsar en la imagen que coincida con la palabra mostrada arriba.

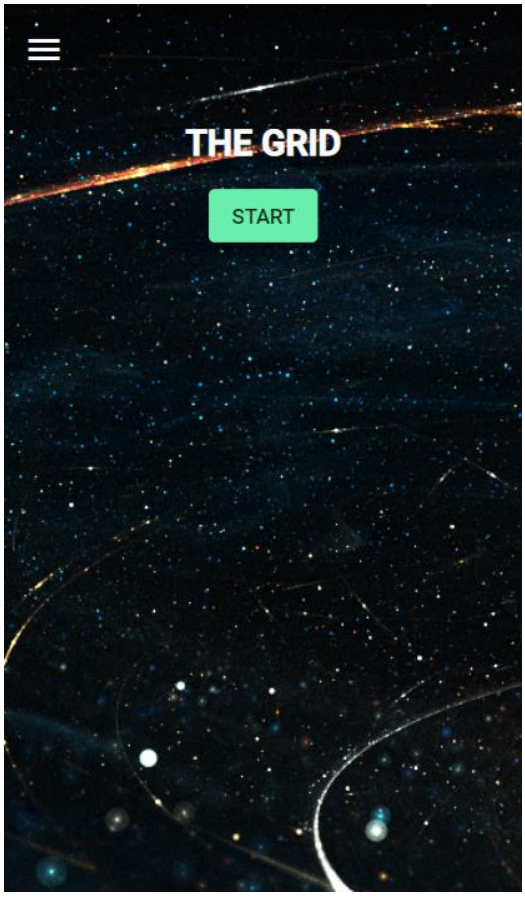


Ilustración 21. Imagen de la pantalla de bienvenida del juego THE GRID, perteneciente a ENGAMES

Cada nivel tiene un botón de START al entrar en él. Al pulsarlo, da comienzo el juego en el que se encuentre el usuario actualmente. Esta pantalla da la bienvenida al juego al usuario y le permite volver a la pantalla principal a través de la barra de navegación lateral, cuyo icono de apertura está ubicado en la esquina superior izquierda de la pantalla.

Ilustración 22. Imagen del primer juego de ENGAMES

Nos encontramos con el primer juego, en el que hay que pulsar en la imagen que indica el texto que está encima del contador de respuestas correctas. Cuando pulsemos en una imagen, el texto cambiará, indicando la siguiente imagen que debemos pulsar. Gracias al sistema de comprobación de aciertos desarrollado, si acertamos al pulsar, debajo de la imagen se rellenará de color verde una franja, indicando que la respuesta ha sido correcta. En el caso contrario, la franja se rellenará de rojo.

Los cambios de color son permanentes en este nivel, debido a la distribución del mismo, permitiendo saber rápidamente el número de aciertos o fallos, además están las dos variables de contado de aciertos o fallos para esto.

Una vez completes el primer nivel, aparecerá una ventana de felicitación y un botón para continuar. Además, desbloquearás la segunda nave espacial, que te lleva al juego CHOOSE IT!.

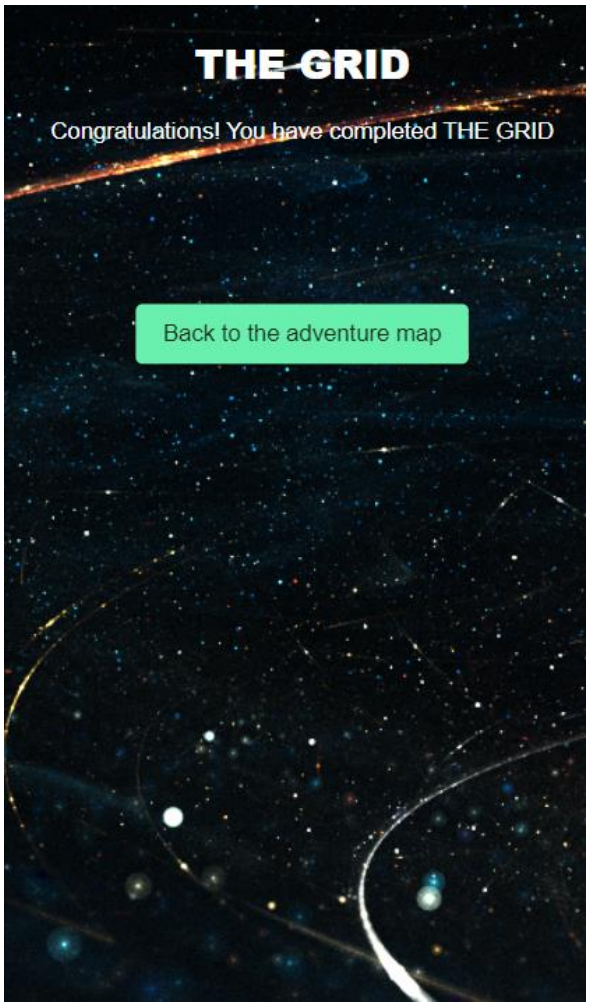


Ilustración 23. Imagen de la pantalla de continuado del juego THE GRID, perteneciente a ENGAMES

Al terminar cada nivel y según la duración de cada uno, aparecerá antes o después una ventana para permitir al usuario continuar. Este botón le llevará de vuelta a la pantalla principal. Donde al usuario le espera un nuevo nivel para jugar.

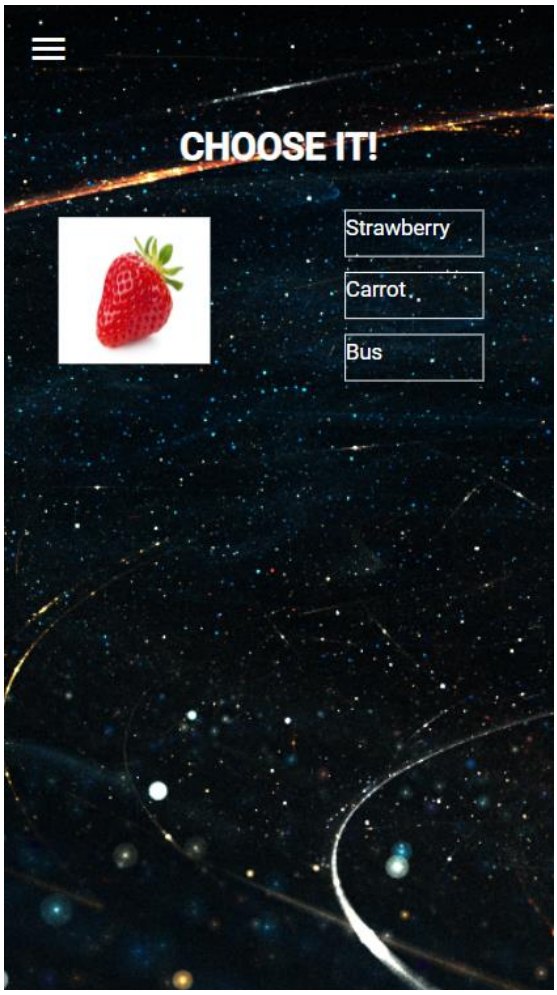


Ilustración 24. Imagen del segundo juego de ENGAMES

Choose It! es el segundo nivel del juego. Sólo si completamos el primer nivel, podremos avanzar hacia el siguiente juego.

El objetivo es seleccionar el rectángulo que corresponde a la imagen. Si acertamos en nuestra respuesta, significará que la palabra escrita y la imagen coinciden, ocurriendo un cambio de color en la casilla seleccionada. El sistema de comprobación de aciertos es igual para todos los niveles, es decir, si se acierta, se coloreará el elemento de verde y si se falla se coloreará de rojo.

Este cambio de color es temporal, no como el del primer nivel.

Una vez hayas logrado completar los dos primeros juegos, desbloquearás el tercer y último nivel, llamado LATERAL BUTTONS, que tiene una interfaz muy intuitiva.

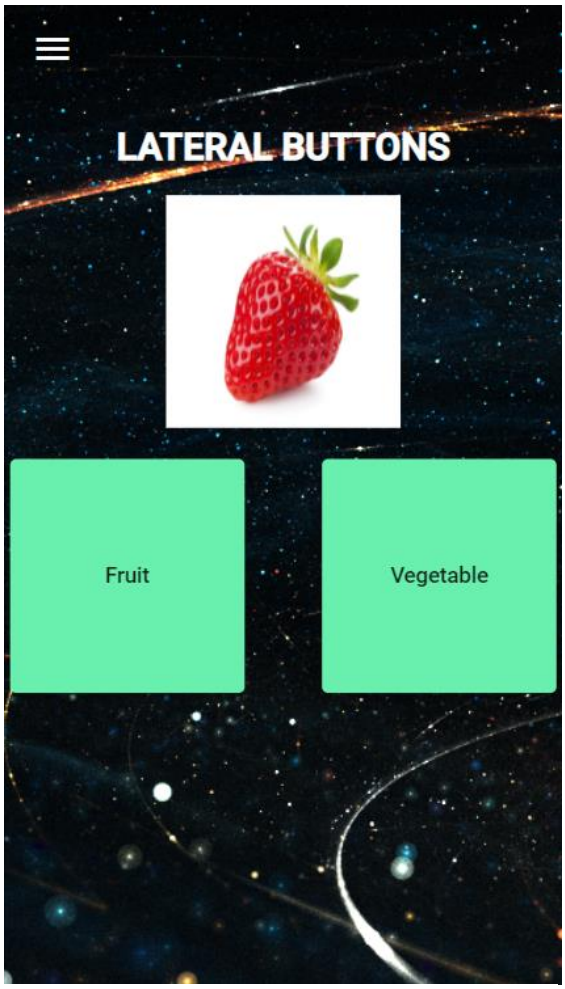


Ilustración 25. Imagen del tercer juego de ENGAMES

El tercer y último juego disponible hasta el momento, trata de dos botones. Se mostrará una imagen o un texto y, en los dos botones, dos categorías o nombres distintos. El objetivo es relacionar correctamente la imagen o palabra actualmente mostrada con el botón que debe ser.

En el caso de que el usuario acierte, el botón tendrá un cambio de color temporal a verde, de igual modo, en el caso de fallar la respuesta, el botón tendrá un cambio de color a rojo, volviendo a su color original tras 700 milisegundos.

Una vez explicados los juegos, vamos a ver la apariencia y funcionalidades de la barra de navegación lateral.

 Icono de apertura del menú lateral - Este icono ubicado en la esquina superior izquierda de cualquier vista, activa la barra de navegación lateral, abriéndola.

 Icono de cerrado del menú lateral - Este otro icono, ubicado en el mismo sitio que el anterior, se muestra sólo cuando el *SideNav* está activo, apareciendo este icono en vez del de las tres rayas horizontales. Su función es cerrar la barra de navegación lateral.

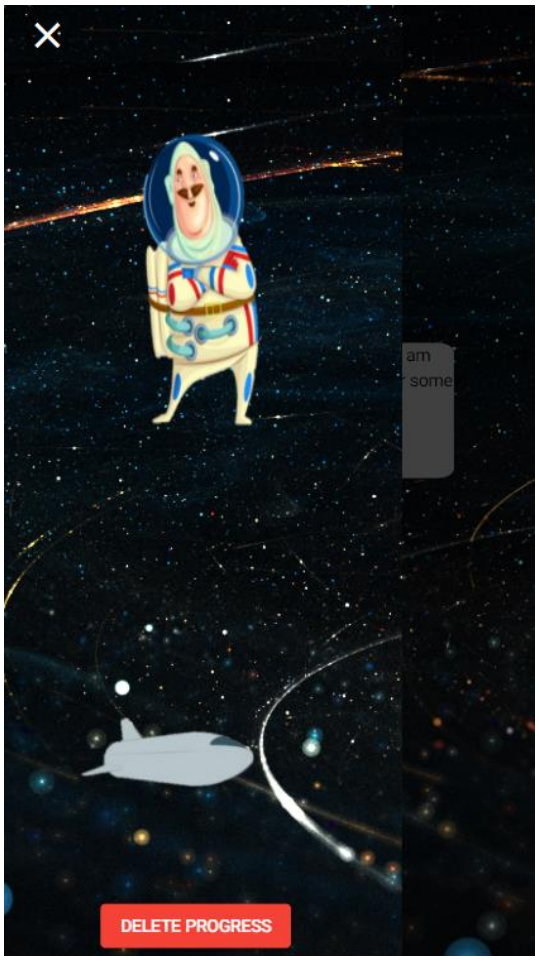


Ilustración 26. Imagen de la barra de navegación lateral de ENGAMES

Cuando el *LateralNavBar está activo*, se superpone a la pantalla actual, quedándose en una especie de segundo plano y oscureciéndose esa pantalla en la que estuviéramos. En el caso de pulsar fuera del *SideNav* o hacer click en el icono de cerrado del menú lateral, el menú se cerrará, dando paso otra vez a la vista que tuviéramos antes de abrirlo.

Observamos a Space George en la parte superior. Si pulsamos en él, nos llevará al perfil, donde podremos, en una versión futura, cambiarle el atuendo a nuestro protagonista.

En la parte inferior del menú lateral, están la nave que nos lleva a la pantalla principal y un botón rojo que no debes pulsar. Si pulsas ese botón borrarás todo tu progreso en ENGAMES. No obstante, si lo haces no debes preocuparte, puedes volver a jugar a todos los juegos cuantas veces quieras. El objetivo de este botón es permitirte empezar desde 0, pudiendo volver a ver todos los niveles y diálogos.

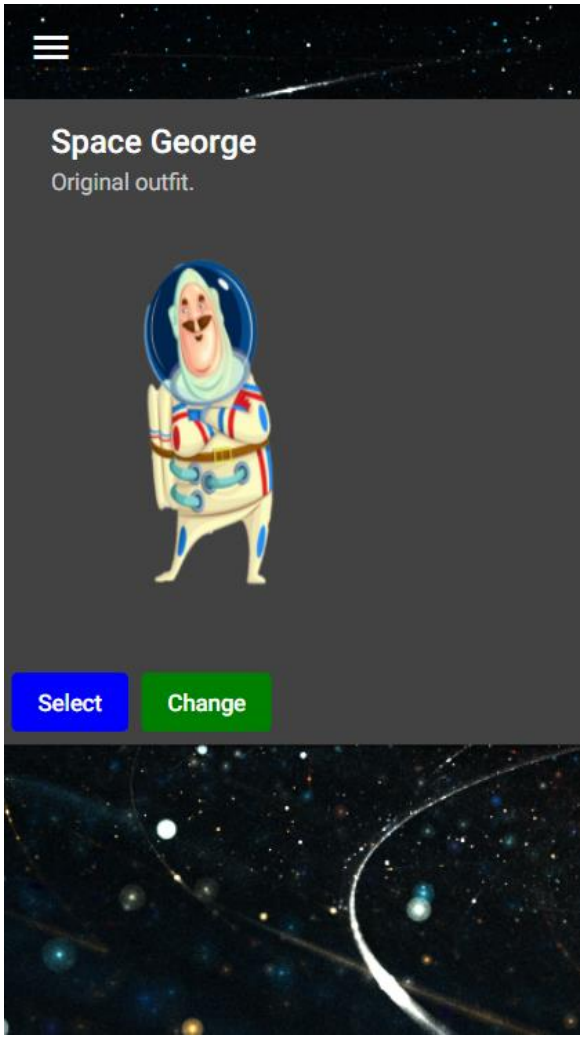


Ilustración 27. Imagen del Perfil de ENGAMES

Bienvenido a tu perfil.

Este componente está todavía en desarrollo, sin embargo, su objetivo es permitirte personalizar a Space George. Podrás ponerle sombreros, gafas de todo tipo, zapatos o cualquier cosa que imagines.

Se mostrarán todos los accesorios o atuendos disponibles para seleccionar una combinación a gusto del usuario.

14. Anexo II: Clases y variables utilizadas

Explicaremos, una a una, todas las clases y variables utilizadas en el funcionamiento de ENGAMES.

14.1. Variables utilizadas

14.1.1. Game1Logic

Estas variables son accesibles globalmente utilizando this.game.VAR. Esto se debe a que Game1Logic estará inyectado en cada componente, así se podrá hacer uso de las funciones y variables aquí incluidas.

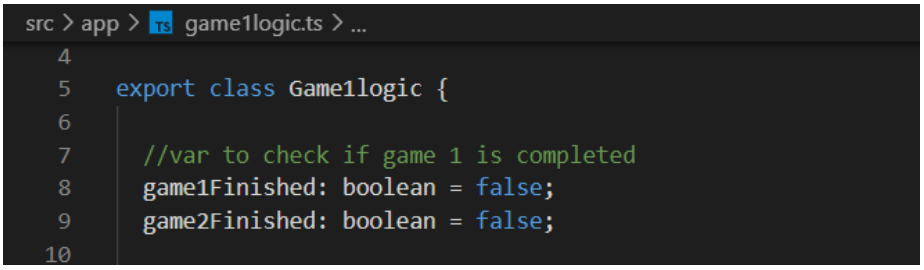


Ilustración 28. Imagen de las variables de la clase Game1Logic

Estas dos variables son utilizadas para comprobar si el primer o el segundo juego han sido completados y así ir mostrando los siguientes niveles a medida que vaya avanzando el usuario a través de ENGAMES.

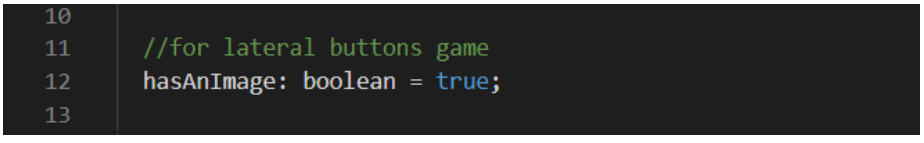


Ilustración 29. Imagen de las variables de la clase Game1Logic

hasAnImage sirve para el tercer nivel, LATERAL BUTTONS, indicando si el próximo elemento a mostrar es una imagen o un texto.

```
13
14    //vars to advance in adventureText.json
15    showDialog: boolean = true;
16    text1: boolean = false;
17    text2: boolean = false;
18    text3: boolean = false;
19
```

Ilustración 30. Imagen de las variables de la clase Game1Logic

Estas variables son un *checkpoint* para los diálogos, al igual que con game1Finished y game2Finished, sirven para mostrar el diálogo correspondiente según vaya avanzando el usuario a través de los niveles.

```
21
22    gameStatus: Status;
```

Ilustración 31. Imagen de las variables de la clase Game1Logic

Status es un *enum* con 2 posibilidades: STOP o START. Esto servirá como variables de control en el HTML y, en la parte de programación, servirá como forma de iniciar cada nivel.

```
24
25    correctAnswers: number = 0;
26    wrongAnswers: number = 0;
27
```

Ilustración 32. Imagen de las variables de la clase Game1Logic

correctAnswers y wrongAnswers sirven como contador de aciertos para el primer juego, una llevará la cuenta del número de respuestas acertadas y la otra, del número de respuestas falladas, respectivamente.

```
27
28    nextDificulty: boolean = false;
29    flag: number = 1; //to show a button for next difficulty
30
```

Ilustración 33. Imagen de las variables de la clase Game1Logic

nextDificulty es una variable de control para el HTML de cada nivel. Mostrando unas opciones para continuar una vez has finalizado un juego.

flag es un contador que va avanzando hasta el límite impuesto en cada nivel. Una vez llegue a su máximo, servirá como forma de cambiar el valor de nextDificulty a true, indicando así la finalización del juego.

```
30
31    random: number = 0;
32    randomCopy: number = 0;
33
```

Ilustración 34. Imagen de las variables de la clase Game1Logic

Estas dos variables son utilizadas para generar un número aleatorio y, tras algunas operaciones, transformar los resultados en números del 0 al 8. Son utilizadas para ver por qué imagen se empieza en el nivel 1.

14.1.2. Adventure

Estas variables y algunas de la clase 14.2.2. Game1Logic sirven para el funcionamiento de la pantalla principal.

```
src > app > game1 > pages > adventure > adventure.component.ts > AdventureComponent
15
16  export class AdventureComponent implements OnInit {
17    db: db[];
18
```

Ilustración 35. Imagen de las variables de la clase Adventure

db almacenará toda la información sobre los diálogos.

```
18
19    game1Finished: any;//local storage vars
20    game2Finished: any;
21
```

Ilustración 36. Imagen de las variables de la clase Adventure

Estas dos variables almacenarán contenido sobre los niveles actuales a mostrar, es decir, mostrarán más niveles disponibles o menos, según el avance de cada usuario por ENGAMES, utilizando el recurso web L*ocal Storage*.

14.1.3. Game1

Estas variables y algunas de la clase 14.2.2. Game1Logic sirven para el funcionamiento del primer nivel.

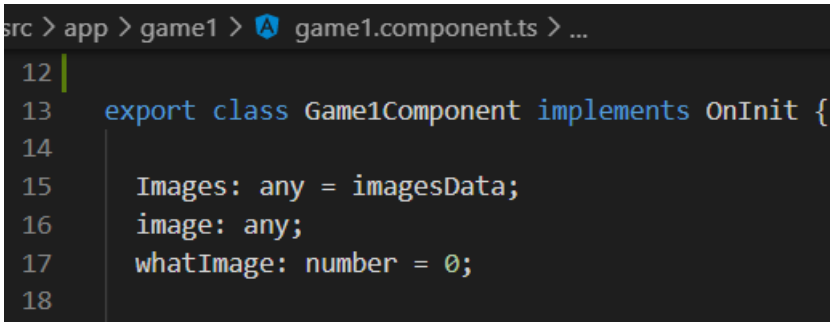


Ilustración 37. Imagen de las variables de la clase Game1

whatImage es un contador para ir mostrando la imagen correspondiente a través del id que posee cada imagen.

image recupera la información del contenido de images.json.

Images es usado para la parte HTML como variable en la que se va almacenando el contenido de las imágenes.

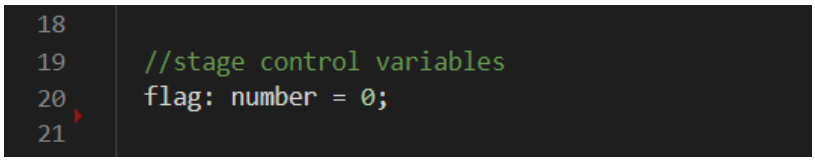


Ilustración 38. Imagen de las variables de la clase Game1

Esta variable es un contador que controla el final del juego.

14.1.4. ChooseIt

Estas variables y algunas de la clase 14.2.2. Game1Logic sirven para el funcionamiento del segundo nivel.

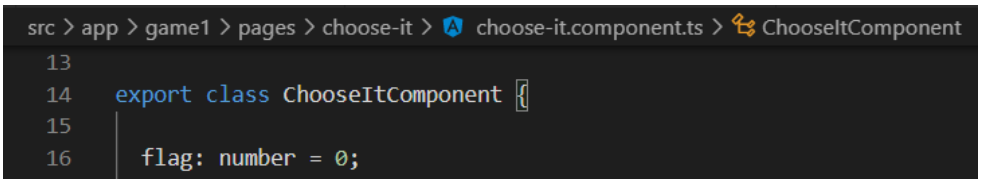


Ilustración 39. Imagen de las variables de la clase ChooseIt

flag es una variable contador, utilizada para marcar el final del juego.

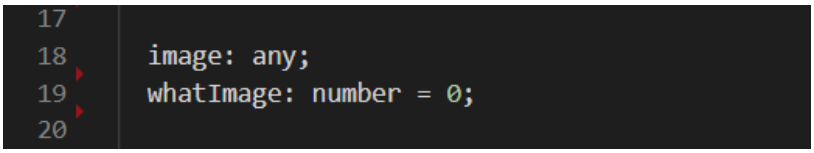


Ilustración 40. Imagen de las variables de la clase ChooseIt

whatImage es una variable que controla el índice de la imagen a mostrar, controlado por la variable image.

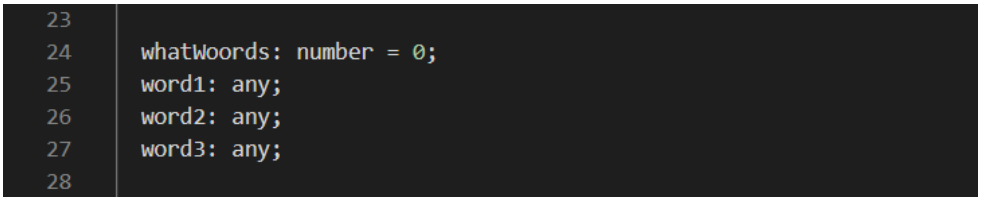


Ilustración 41. Imagen de las variables de la clase ChooseIt

whatWords sirve como índice para mostrar la palabra que toca, usando word1, word2 y word3 para ello.

Ilustración 42. Imagen de las variables de la clase ChooseIt

Esta variable servirá para averiguar si la palabra pulsada es la correcta o no.

14.1.5. LateralButtons

Estas variables y algunas de la clase 14.2.2. Game1Logic sirven para el funcionamiento del tercer y último nivel disponible.

Ilustración 43. Imagen de las variables de la clase LateralButtons

Esta variable de tipo contador, a medida que avanza, cuando llegue a su límite cambiará el valor de una variable para dar por finalizado el nivel.

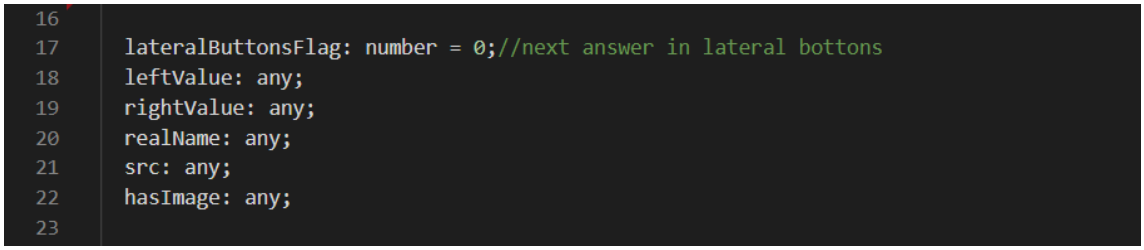


Ilustración 44. Imagen de las variables de la clase LateralButtons

Estas variables son cada uno de los tipos de dato que posee un elemento en el tercer nivel.

lateralButtonsFlag es una variable de tipo entero, que marcará el elemento a mostrar, actuando como índice de un array, usando el resto de variables para mostrar una cosa u otra.

leftValue y rightValue mostrarán el contenido del botón de la izquierda y de la derecha respectivamente.

realName es el valor verdadero de la pregunta, usado para confirmar la veracidad de la respuesta del usuario.

hasImage indicará si la pregunta debe mostrar o no una imagen, en el caso de no mostrar una, el campo src no será necesario.

src es la ruta de la imagen.

14.1.6. SideNav

Estas variables son utilizadas para el funcionamiento de la barra de navegación lateral, disponible en todas las pantallas.

Utilizaremos los términos: *LateralNavBar*, barra de navegación lateral y *SideNav* como equivalentes.

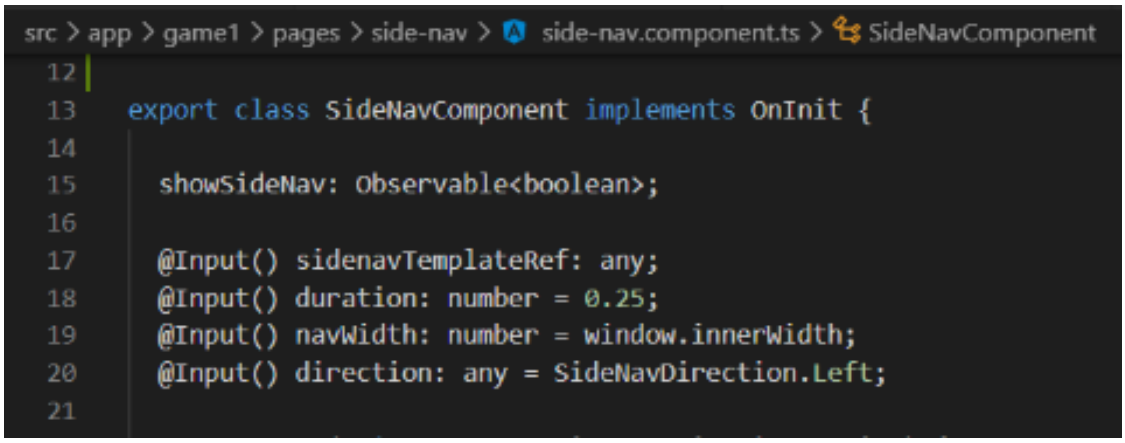


Ilustración 45. Imagen de las variables de la clase SideNav

Estas 4 variables son concretadas en app.component.html y proporcionan información sobre las características del *LateralNavBar*.

sidenavTemplateRef es una referencia para el S*ideNav*.

duration indicará la duración de la animación de apertura y cerrado de la barra de navegación lateral.

navWidth sirve para proporcionarle un ancho al S*ideNav*.

direction indica desde dónde se realizará la animación de apertura del *LateralNavBar*.

14.1.7. Profile

Hacemos mención sobre este componente que, por razones temporales, no pudo llegar a implementarse. Lo mostrado en anterioridad sobre el Profile es sólo apariencia y no posee ninguna funcionalidad.

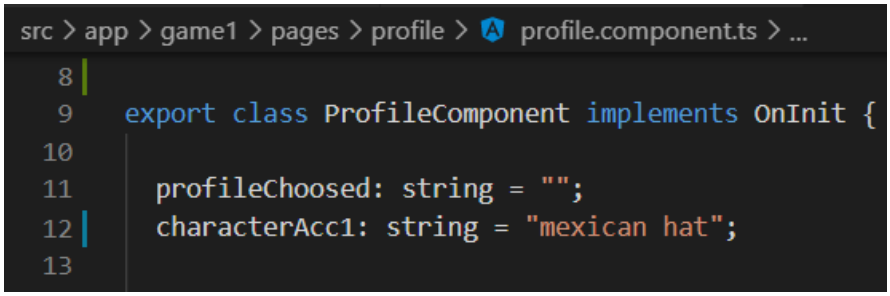


Ilustración 46. Imagen de las variables de la clase Profile

14.1.8. Ficheros JSON

Un fichero JSON contiene información usada para el intercambio de datos, pudiendo almacenar cualquier tipo de información que pueda ser escrito en un formato de texto sencillo.

14.1.8.1. images.json

Este archivo JSON almacena los datos utilizados para poder mostrar imágenes.

Existen 3 tipos de datos:

id representa un identificador único para cada imagen.

name es el nombre de la imagen.

src es la ruta de la imagen

src > app > game1 > data > {..} images.json > ...

```
[{
  "id": 1,
  "name": "Strawberry",
  "src": "assets/images/strawberry.jpg"
},
{
  "id": 2,
  "name": "Dog",
  "src": "assets/images/dog.jpg"
},
{
  "id": 3,
  "name": "Buffet",
  "src": "assets/images/buffet.png"
},
{
  "id": 4,
  "name": "Carrot",
  "src": "assets/images/carrot.jpg"
},
{
  "id": 5,
  "name": "Agriculture",
  "src": "assets/images/agriculture.jpg"
},
{
  "id": 6,
  "name": "Building",
  "src": "assets/images/building.jpg"
},
```

Ilustración 47. Imagen del fichero images.json

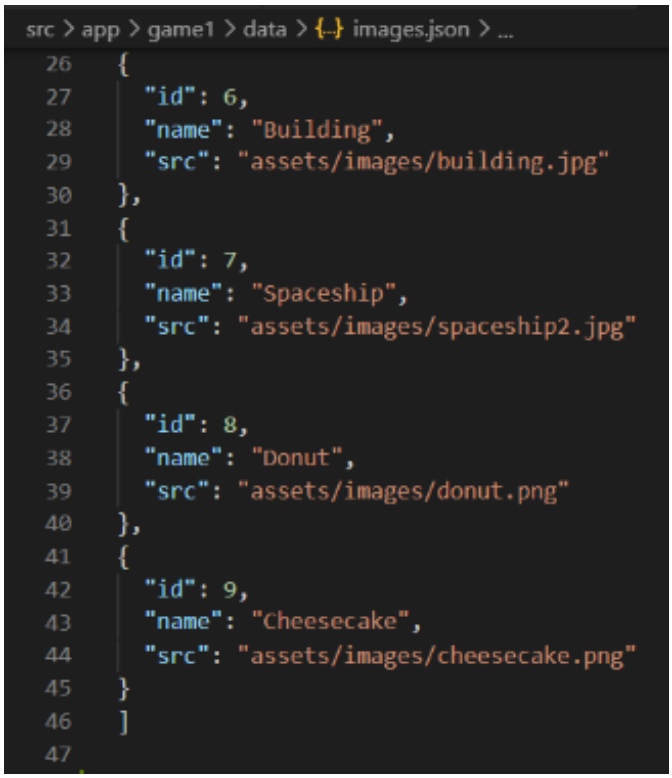


Ilustración 48. Imagen del fichero images.json

14.1.8.2. words.json

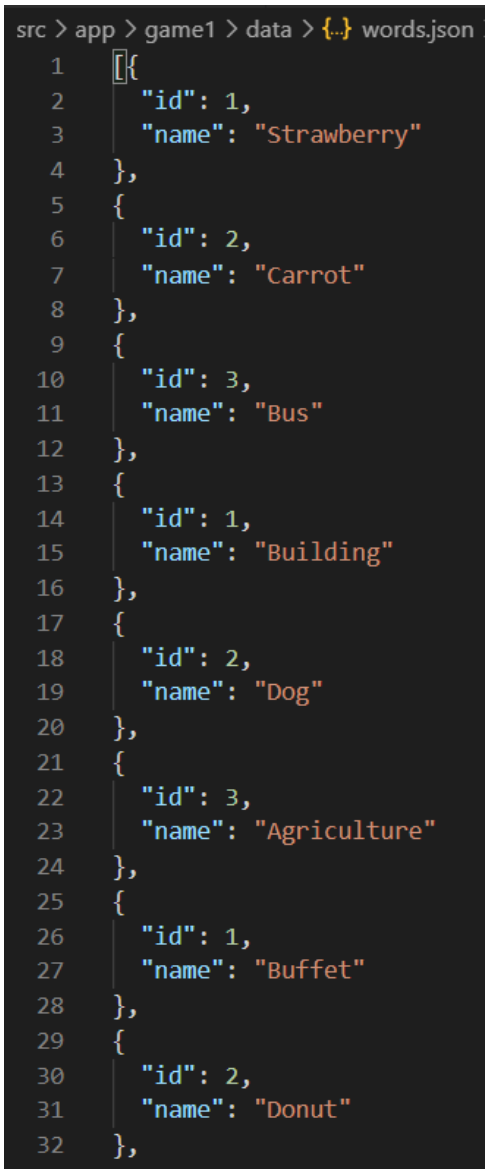
Este archivo JSON almacena los datos de las palabras utilizadas en ENGAMES.

Cada elemento tiene 2 tipos de datos:

id indicará en qué posición del segundo nivel aparecerá.

name es la palabra.

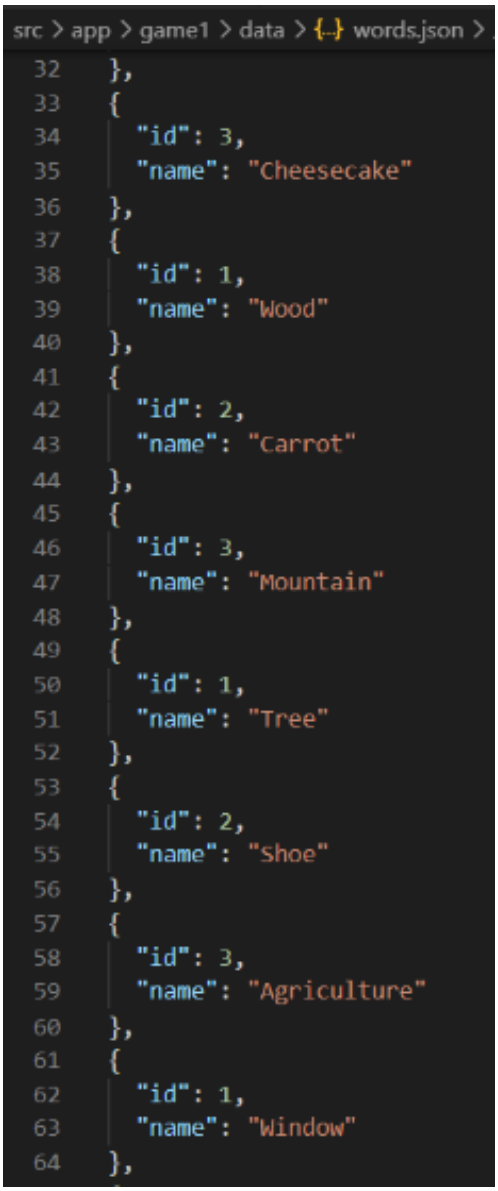
src > app > game1 > data > {...} words.json



```
[{
  "id": 1,
  "name": "Strawberry"
},
{
  "id": 2,
  "name": "Carrot"
},
{
  "id": 3,
  "name": "Bus"
},
{
  "id": 1,
  "name": "Building"
},
{
  "id": 2,
  "name": "Dog"
},
{
  "id": 3,
  "name": "Agriculture"
},
{
  "id": 1,
  "name": "Buffet"
},
{
  "id": 2,
  "name": "Donut"
},
```

Ilustración 49. Imagen del fichero words.json

src > app > game1 > data > {...} words.json >



```
},
{
  "id": 3,
  "name": "Cheesecake"
},
{
  "id": 1,
  "name": "Wood"
},
{
  "id": 2,
  "name": "Carrot"
},
{
  "id": 3,
  "name": "Mountain"
},
{
  "id": 1,
  "name": "Tree"
},
{
  "id": 2,
  "name": "Shoe"
},
{
  "id": 3,
  "name": "Agriculture"
},
{
  "id": 1,
  "name": "Window"
},
```

Ilustración 50. Imagen del fichero words.json

Ilustración 51. Imagen del fichero words.json

14.1.8.3. lateralButtons.json

Este archivo JSON almacena los datos utilizados en el tercer nivel.

Existen 7 tipos de datos:

id representa un identificador único para cada elemento.

name es el nombre del elemento.

left indica el valor que aparecerá en el botón de la izquierda.

right indica el valor del botón de la derecha.

trueOne es la respuesta correcta, usada más tarde para comparar la respuesta del usuario con la respuesta verdadera.

src es la ruta de la imagen.

image sólo tiene 2 valores, true o false, indicando si la pregunta tiene imagen o no. En el caso de ser false, el tipo de dato src no existe en el elemento.

src > app > game1 > data > {...} lateralButtons.json > ...

```
[{
  "id": 1,
  "name": "Strawberry",
  "left": "Fruit",
  "right": "Vegetable",
  "trueOne": "Fruit",
  "src": "assets/images/strawberry.jpg",
  "image": "true"
},
{
  "id": 2,
  "name": "Wood",
  "left": "Animal",
  "right": "Not an animal",
  "trueOne": "Not an animal",
  "image": "false"
},
{
  "id": 3,
  "name": "Roof",
  "left": "Roof",
  "right": "Wood",
  "trueOne": "Roof",
  "src": "assets/images/roof.jpg",
  "image": "true"
},
{
  "id": 4,
  "name": "Wood",
  "left": "Wood",
  "right": "Good",
  "trueOne": "Wood",
  "src": "assets/images/wood.jpg",
  "image": "true"
},
{
```

Ilustración 53. Imagen del fichero lateralButtons.json

src > app > game1 > data > {...} lateralButtons.json > ...

```
},
{
  "id": 5,
  "name": "Deer",
  "left": "Dear",
  "right": "Deer",
  "trueOne": "Deer",
  "src": "assets/images/deer.jpg",
  "image": "true"
},
{
  "id": 6,
  "name": "Boulder",
  "left": "Bolder",
  "right": "Boulder",
  "trueOne": "Boulder",
  "src": "assets/images/boulder.jpg",
  "image": "true"
},
{
  "id": 7,
  "name": "Boulder",
  "left": "Bolder",
  "right": "Boulder",
  "trueOne": "Boulder",
  "src": "assets/images/boulder.jpg",
  "image": "true"
}
]
```

Ilustración 52. Imagen del fichero lateralButtons.json

14.2. Clases utilizadas

14.2.1. Game1Status

Esta clase es de tipo enum. Existen dos posibilidades de estado de Status:

Ilustración 54. Imagen de la clase Game1Status

14.2.2. Game1Logic

Game1Logic es una clase que servirá de apoyo para todas las demás clases.

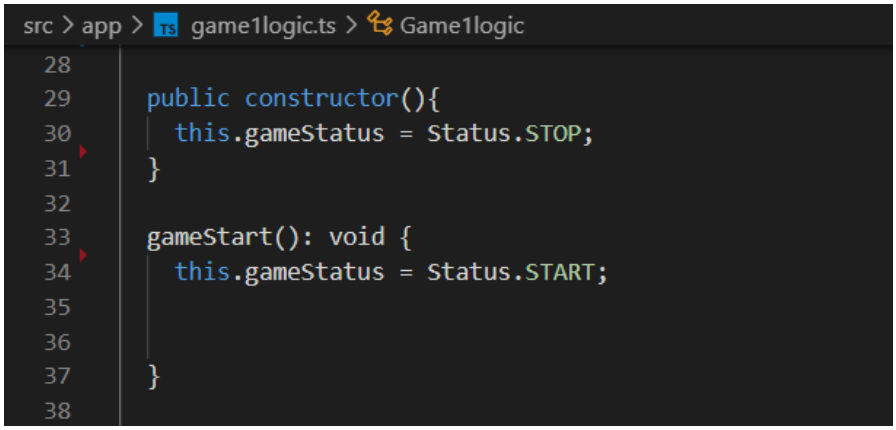


Ilustración 55. Imagen de la clase Game1Logic

En esta sección de la clase tenemos el constructor(), donde se le da el valor 0 o STOP a gameStatus, y la función gameStart(), donde se le da el valor 1 o START a la variable.

```
38
39    //random number generator from 0 to 8
40    randomImage(): number {
41      this.random = Math.random();
42      this.randomCopy = this.random;
43      this.randomCopy = this.randomCopy * 10 % 1;
44      return ((this.random * 10) - this.randomCopy) - 1;
45    }
46
47  }
```

Ilustración 56. Imagen de la clase Game1Logic

Este método genera un número aleatorio que después es tratado con diferentes operaciones para mantener el número entre 0 y 8.

14.2.3. Adventure

Esta clase le da funcionalidad a la pantalla principal.

```
src > app > game1 > pages > adventure > adventure.component.ts > AdventureComponent
19
20    constructor(public game: Game1logic, private DbService: DbService) { }
21
22    ngOnInit(): void {
23      this.game1Finished = localStorage.getItem('game1');
24      this.game2Finished = localStorage.getItem('game2');
25
```

Ilustración 57. Imagen de la clase Adventure

En el constructor, pasamos como parámetro un elemento de tipo Game1logic y otro de tipo DbService, que utilizaremos más adelante.

Inicializamos game1Finished y game2Finished con el contenido del L*ocal Storage*. Sólo habrá guardado 1 elemento en el *Local Storage* sobre el número de juegos que se debe mostrar, el otro elemento no tendrá nada guardado.

```
25
26      if (this.game2Finished === "finished"){
27        this.game.text1 = false;
28        this.game.text2 = false;
29        this.game.text3 = true;
30      } else if(this.game1Finished === "finished"){
31        this.game.text1 = false;
32        this.game.text2 = true;
33      } else {
34        this.game.text1 = true;
35      }
36
```

Ilustración 58. Imagen de la clase Adventure

En esta sección comprobamos qué elemento es el que tiene la información guardada y, según cuál sea, marcamos un diálogo a mostrar u otro.

Si game2Finished tiene como valor "finished", significa que el usuario se ha pasado todos los niveles hasta el que se ocupa de actualizar esta variable, así que deberemos mostrar el diálogo correspondiente. Esto se aplica para cada nivel pasado, es decir, el primer y el segundo (ya que el tercero es el último, no se puede avanzar más a partir de ahí)

Si el usuario no ha completado ningún juego, mostraremos el diálogo inicial.

```
36
37      this.DbService.getDB().
38      subscribe(result => this.db = result);
39    }
40
```

Ilustración 59. Imagen de la clase Adventure

Las líneas 37 y 38 son la forma de poder ver y utilizar el contenido de DbService.

Así, acaba el método ngOnInit().

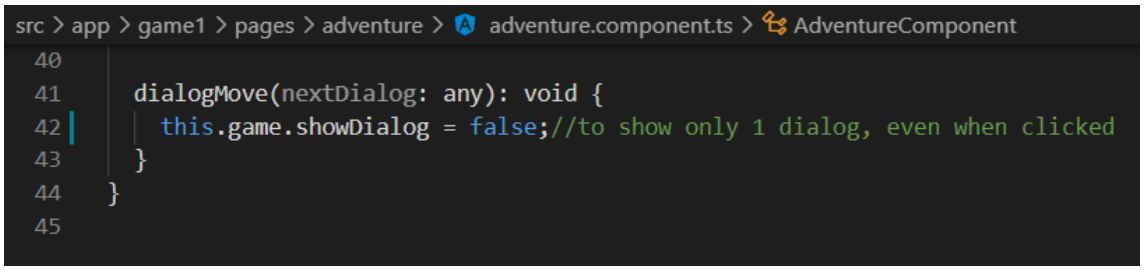


Ilustración 60. Imagen de la clase Adventure

En el método dialogMove(), se cambia el valor de la variable showDialog para que se oculte el diálogo tras pulsar en ellos. Este método se ejecuta cada vez que se hace click en el diálogo.

14.2.3. Game1

Sigue la clase Game1, que le da funcionalidad al primer nivel.

Ilustración 61. Imagen de la clase Game1

En el constructor, pasamos por parámetros un elemento de tipo Game1logic que usaremos más adelante.

En startGame() se le da el valor START a Status y se llama al método gameStart() de la clase 14.2.2. Game1Logic.

También se le da un valor inicial aleatorio, usando el método randomImage() de Game1Logic, a la variable que indica qué imagen se mostrará.

Por último, se carga en image la imagen elegida.

```
35
36  clickSubfield (subfield: any): void {
37    if(this.game.gameStatus === 1) {
38      const pictureName = subfield.currentTarget.getAttribute('id');
39
40      if(this.image.name === pictureName){//si coincide el nombre con el del campo clickado
41        subfield.currentTarget.classList.add('right-answer');
42        this.game.correctAnswers++;
43      } else {
44        subfield.currentTarget.classList.add('wrong-answer');
45        this.game.wrongAnswers++;
46      }
47
48
```

Ilustración 62. Imagen de la clase Game1

El método clickSubfield() es el método llamado cada vez que se pulsa en un campo en el primer nivel.

Comienza con un if de comprobación, si Status no tiene de valor 1 o START, no ocurre nada en la app.

Obtenemos el nombre del elemento pulsado en el HTML a través del atributo id.

Comparamos el nombre del elemento pedido para seleccionar y si coinciden, cambiamos el color del elemento a verde y sumamos 1 a la variable correctAnswers de la clase 14.2.2. Game1Logic, si no coinciden cambiamos el color a rojo y sumamos 1 a wrongAnswers.

```
47
48
49      if(this.flag === 8){//for showing next diffculty
50        this.game.nextDificulty = true;
51        localStorage.setItem('game1', 'finished');
52        this.game.showDialog = true;
53        this.game.text1 = false;
54      }
55
```

Ilustración 63. Imagen de la clase Game1

Comprobamos el valor de flag, si es igual a 8 significa que el juego ha terminado, por lo que cambiamos el valor de nextDificulty a true para mostrar la pantalla de continuar.

Guardamos en el *Local Storage* que el usuario ha completado el nivel 1 y actualizamos los valores respectivos al diálogo (de las variables showDialog y text1).

```
55
56          this.flag++;//for next difficulty
57          this.whatImage++;//+1 to the index
58
59          if(this.whatImage > 8){//if its over the possible index
60            this.whatImage -= 9;//-9 to start over
61          }
62
63          this.image = imagesData[this.whatImage];
64       }
65
66    }
67
68  }
69
```

Ilustración 64. Imagen de la clase Game1

Actualizamos las variables flag y whatImage para la siguiente ronda.

Comprobamos si whatImage está por encima del índice más alto (debido al método random) y si es así, restamos 9 para volver a empezar.

Para finalizar, actualizamos la imagen por la que se preguntará.

14.2.4. ChooseIt

Esta clase es la responsable de la funcionalidad del segundo nivel.

```
src > app > game1 > pages > choose-it > choose-it.component.ts > ChooseItC
28    constructor(public game: Game1logic) { }
29
30
31    startGame(): void{
32      this.game.gameStatus = Status.START;
33      this.game.gameStart();
34      this.image = imagesData[this.whatImage];
35
36      this.whatWoords = 0;
37      this.word1 = wordsToChoose[this.whatWoords];
38      this.word2 = wordsToChoose[this.whatWoords+1];
39      this.word3 = wordsToChoose[this.whatWoords+2];
40
41
42    }
43
```

Ilustración 65. Imagen de la clase ChooseIt

En el constructor, pasamos como parámetro un elemento de tipo Game1Logic que utilizaremos más tarde para acceder a sus variables.

Le damos el valor 1 a Status y llamamos al método gameStart() de la clase 14.2.2. Game1Logic.

Cargamos la primera imagen en image.

Para finalizar el método startGame() cargamos las 3 primeras palabras que se usarán para el segundo nivel.

```
43
44    clickSubfield (subfield: any): void {
45      const pressedButton = subfield.currentTarget;
46      if(this.game.gameStatus === 1) {
47        this.buttonID = pressedButton.getAttribute('id');
48
49        //checking the right answer
50        if(this.whichAnswer(pressedButton)){
51          pressedButton.classList.add('right-answer');
52          setTimeout(()=>{
53          pressedButton.classList.remove('right-answer');
54        },700)
55        } else {
56          pressedButton.classList.add('wrong-answer');
57          setTimeout(()=>{
58            pressedButton.classList.remove('wrong-answer');
59          },700)
60        }
61
```

Ilustración 66. Imagen de la clase ChooseIt

El método clickSubfield() del segundo juego, al igual que el del primero, es el encargado de la funcionalidad del nivel, cada vez que el usuario hace una acción en el juego Choose It!, se llama a esta función.

Comenzamos guardando información sobre el elemento pulsado en pressedButton y comprobamos el Status del juego.

En el HTML se le otorgó un id a cada botón, así que obtenemos el del pulsado.

Comprobamos, usando otro método (wichAnswer()), si el elemento pulsado es el correcto y si es así, cambiamos a verde el color del elemento y esperamos 700 milisegundos para eliminar el color verde, lo mismo para una respuesta fallida.

La eliminación de los colores fue necesaria tras muchas pruebas, al ser la única forma de conseguir que no se trabaran los colores.

```
62
63        if(this.flag === 5){//for showing next diffculty
64          this.game.nextDificulty = true;
65          localStorage.setItem('game2', 'finished');
66          this.game.showDialog = true;
67        }
68
69        //moving to next image and words
70        this.whatImage++;
71        this.whatWoords = this.whatWoords + 3;
72        this.flag++;
73
74        //for showing next image and words
75        this.image = imagesData[this.whatImage];
76        this.word1 = wordsToChoose[this.whatWoords];
77        this.word2 = wordsToChoose[this.whatWoords+1];
78        this.word3 = wordsToChoose[this.whatWoords+2];
79
80      }
81
82    }
83
```

Ilustración 67. Imagen de la clase ChooseIt

Si flag llega a la cifra indicada, el juego habrá concluido, por lo tanto, cambiamos el valor de nextDificulty para mostrar la pantalla para continuar, guardamos información de que el juego número 2 ha concluido y mostramos el diálogo pertinente cambiando la variable showDialog a true.

Actualizamos las variables que sirven como índice para saber qué elemento mostrar.

Por último, apuntamos hacia las siguientes palabras e imagen, con los índices una vez actualizados.

```
83
84    whichAnswer(pressedButton: any): boolean{//method for checking the right answer
85      if(this.word1.id && this.buttonID === '1'){
86        if(this.word1.name === this.image.name)
87          return true;
88      } else if (this.word2.id && this.buttonID === '2'){
89        if(this.word2.name === this.image.name)
90          return true;
91      } else if (this.word3.id && this.buttonID === '3'){
92        if(this.word3.name === this.image.name)
93          return true;
94      } else {
95        console.log("LA RESPUESTA NO ESTA ENTRE LAS CORRECTAS");
96        return false;
97      }
98      return false;
99    }
100 }
101
```

Ilustración 68. Imagen de la clase ChooseIt

Este método se encarga de comprobar si el resultado pulsado es el correcto.

Comprobamos si el id de la palabra (que representa en qué botón debe ir colocado) y del botón son el mismo. Una vez comprobado, se confirma que el valor del botón y el nombre de la imagen coinciden. Haremos estas operaciones para cada botón.

Si ocurre un fallo y la respuesta correcta no estaba entre las 3 opciones, se muestra un mensaje de error.

14.2.5. LateralButtons

La clase LateralButtons se ocupa del funcionamiento del tercer nivel.

Ilustración 69. Imagen de la clase LateralButtons

Pasamos Game1Logic por parámetros al constructor e inicializamos cada tipo de dato del tercer juego, para cargar el primer elemento a mostrar, utilizando como índice la variable lateralButtonsFlag.

En el método startGame(), como en los dos anteriores niveles, ponemos el valor de Status a START y hacemos una llamada al método gameStart() de la clase 14.2.2. Game1Logic.

Ilustración 70. Imagen de la clase LateralButtons

El método LateralButtons() se ocupará de manejar el funcionamiento, ejecutándose cada vez que se pulsa un botón en el juego.

Comenzamos el método guardando en pressedButton información sobre el botón que ha sido pulsado.

Si el botón pulsado es el izquierdo y además el valor de ese botón es el correcto, coloreamos el botón de verde temporalmente. Si el botón pulsado por el usuario es el derecho y el valor de ese botón es el correcto de los dos, también cambiamos el color del botón a verde temporalmente. Sin embargo, si el botón pulsado no es el correcto, cambiamos el color temporalmente del botón que se pulso.

```
61
62      //moving to next question
63      this.lateralButtonsFlag++;
64
65      //saving it
66      this.leftValue = lateralGame[this.lateralButtonsFlag].left;
67      this.rightValue = lateralGame[this.lateralButtonsFlag].right;
68      this.realName = lateralGame[this.lateralButtonsFlag].name;
69      this.src = lateralGame[this.lateralButtonsFlag].src;
70      this.hasImage = lateralGame[this.lateralButtonsFlag].image;
71
```

Ilustración 71. Imagen de la clase LateralButtons

Aumentamos en 1 el valor de la variable que estamos usando como índice para apuntar al siguiente elemento.

Actualizamos cada uno de los valores del siguiente elemento.

```
71
72      if(this.hasImage === "true"){
73        this.game.hasAnImage = true;
74      } else {
75        this.game.hasAnImage = false;
76      }
77      this.flag++;
78      if(this.flag === 6){//for showing next diffculty
79        this.game.nextDificulty = true;
80        localStorage.setItem('game3', 'finished');
81        this.game.text3 = false;
82      }
83    }
84  }
85
```

Ilustración 72. Imagen de la clase LateralButtons

Si el siguiente elemento a mostrar es una imagen, actualizamos el valor de hasAnImage de la clase 14.2.2. Game1Logic a true, si no, lo cambiaremos a false. Usaremos esta variable en el HTML para mostrar una imagen o una palabra.

Aumentamos en 1 flag y si su valor es 6, significará que el nivel ha acabado y por lo tanto, cambiamos el valor de nextDificulty a true, para que muestre la pantalla de continuar, además de ocuparnos de hacer uso del *Local Storage* para guardar el progreso actual y de actualizar el diálogo.

14.2.6. NavigationService

Esta clase es un servicio cuyo objetivo es manejar la información de la barra de navegación lateral.

Angular distingue en la creación de un servicio y un componente, cuando usas su CLI (Command Line Input), creando solamente un archivo .service.ts y otro .service.spec.ts (un archivo especial para pruebas), al dar la orden de crear un servicio. Sin embargo, cuando creas un componente mediante el uso del CLI, Angular crea un elemento de tipo .component.ts, otro .component.spec.ts y un archivo de tipo .component.html y otro de tipo .component.scss.

En conclusión, cuando generas un service a través del CLI sólo crea ficheros relacionados con la parte de comportamiento o funcionamiento, sin embargo, al crear un componente, se crean 2 ficheros de funcionamiento y 2 ficheros para la apariencia.

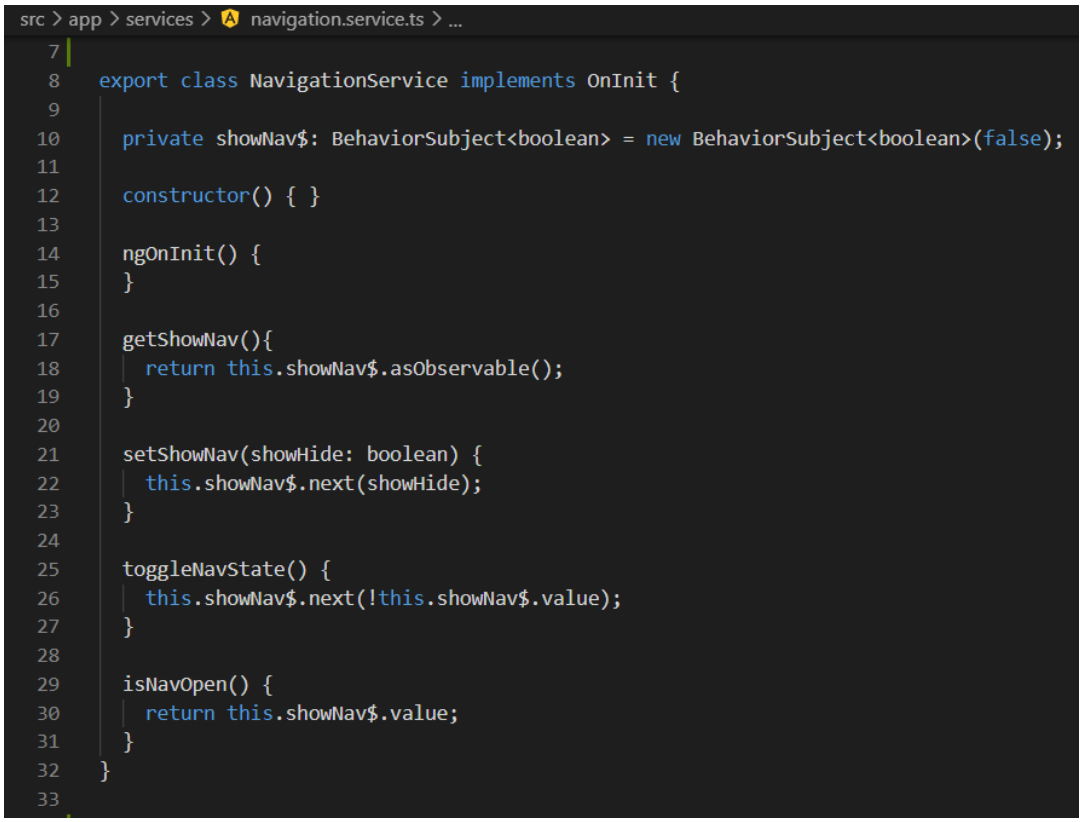


Ilustración 73. Imagen de la clase Navigation

Un observable nos permite escuchar los valores o eventos emitidos.

La variable showNav$ guardará la información del navigation service.

El método getShowNav() crea el observer sobre la variable showNav$, con la función asObservable().

setShowNav() le dirá al *LateralNavBar* si abrirse o cerrarse, dependiendo del valor del boolean pasado por parámetro.

toggleNavState() indica si la siguiente posición en la que debe estar la barra de navegación lateral.

Para finalizar esta clase, tenemos el método isNavOpen(), que nos informará de si el *LateralNavBar* está abierto o cerrado.

14.2.7. SideNav

Esta clase se ocupa de la apariencia, forma de interactuar y del cerrado de la barra de navegación lateral.

Entre esta clase y la clase 14.2.8. SideNavContent, se define el funcionamiento del *LateralNavBar*.

```
21
22    constructor(private navService: NavigationService) {
23
24    }
25
26    ngOnInit(): void {
27      this.showSideNav = this.navService.getShowNav();
28    }
29
30    onSidebarClose() {
31      this.navService.setShowNav(false);
32    }
33
34    getSideNavBarStyle(showNav: boolean) {
35      let navBarStyle: any = {};
36
37      navBarStyle.transition = this.direction + ' ' + this.duration + 's, visibility ' + this.duration + 's';
38      navBarStyle.width = this.navWidth + 'px';
39      navBarStyle[this.direction] = (showNav ? 0 : (this.navWidth * -1)) + 'px';
40
41      return navBarStyle;
42    }
43  }
44
```

Ilustración 74. Imagen de la clase SideNav

Lamamos a la función getShowNav() del servicio 14.2.6. NavigationService y guardamos el resultado en nuestra variable showSideNav.

El método onSidebarClose() se ocupa de cerrar la barra de navegación lateral cuando es llamado.

Finalmente, en getSideNavBarStyle() nos ocuparemos de indicar cómo debe ser la animación de uso del *LateralNavBar* y qué apariencia tendrá.

14.2.8. SideNavContent

Esta clase se complementa con el servicio NavigationService y el componente SideNav, anteriormente explicados.

Ilustración 75. Imagen de la clase SideNavContent

toggleSideNav() se encarga de cerrar la barra de navegación lateral cuando se hace uso de éste método.

14.2.9. InMemoryData

Este servicio, junto con el servicio 14.2.10. Db, conforman la base de datos In-Memory, que combinamos con el *Local Storage* para el guardado de información de la aplicación.

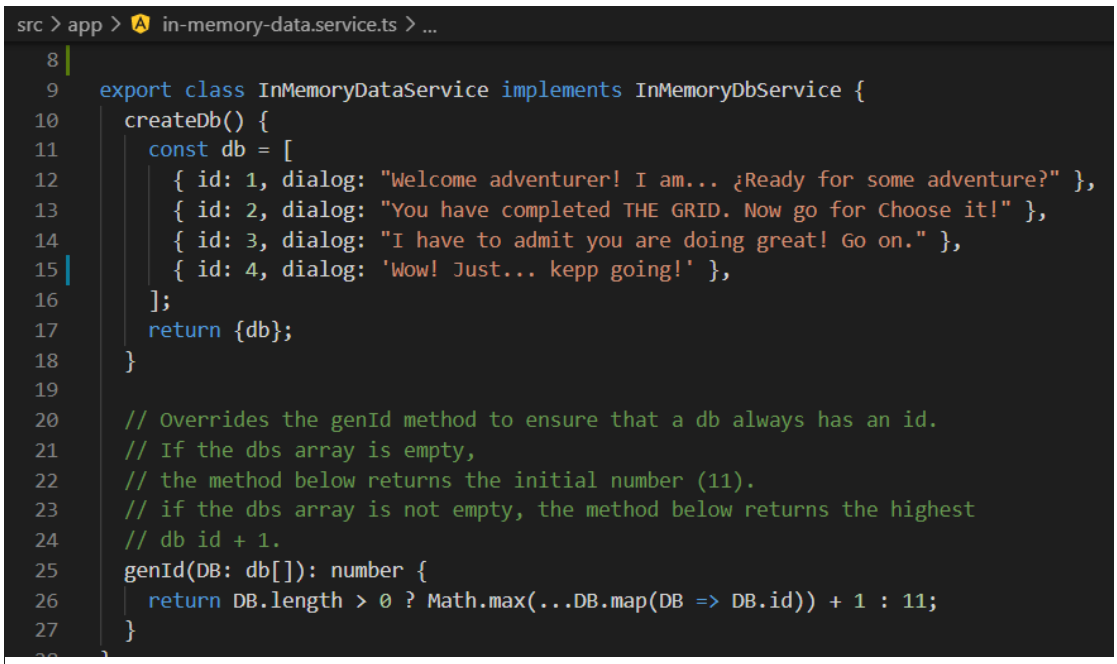


Ilustración 76. Imagen de la clase InMemory

En el método createDb(), creamos un array en el que guardaremos toda la información de los diálogos que se irán mostrando a medida que vayamos avanzando en ENGAMES.

En el siguiente método, genId(), devuelve el valor del id más alto de nuestro array, sumándole 1. Esto en el caso de no estar vacío, si estuviera vacío devolvería 11.

14.2.10. Db

Db es el servicio que se encarga de obtener los datos que necesitamos.

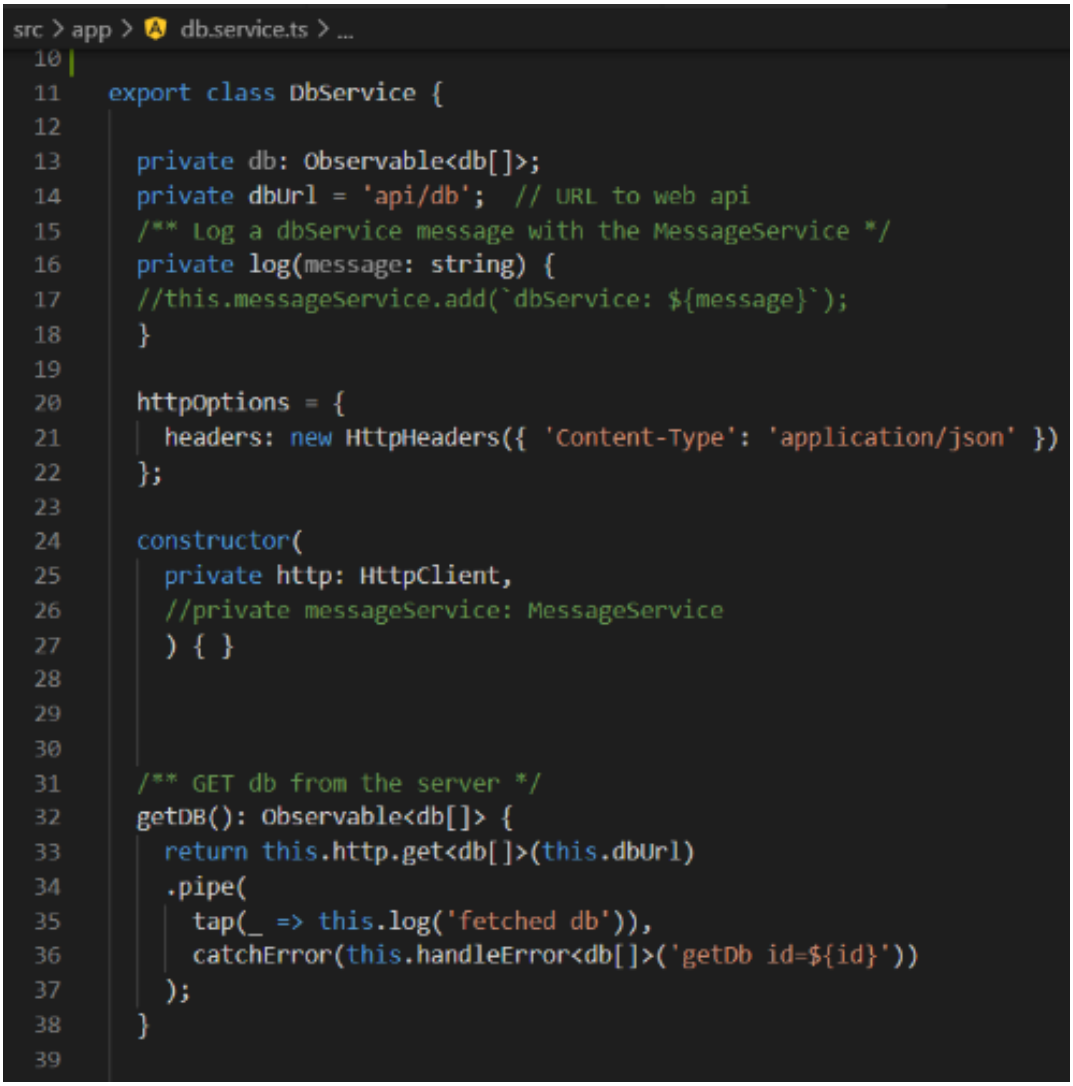


Ilustración 77. Imagen de la clase Db

Creamos variables que usaremos más adelante, como db, para guardar la información y dbUrl para indicarle dónde buscar.

En el constructor creamos un objeto de tipo HttpCliente ya que hemos importado esa librería. Utilizaremos métodos de esta clase en getDB().

El método importante es getDB(), que nos devuelve el contenido de lo que guardamos en la clase 14.2.9. InMemoryData, a través del http.get.

Para encontrar errores, se hace uso de .pipe. Hemos importado con anterioridad la librería catchError.

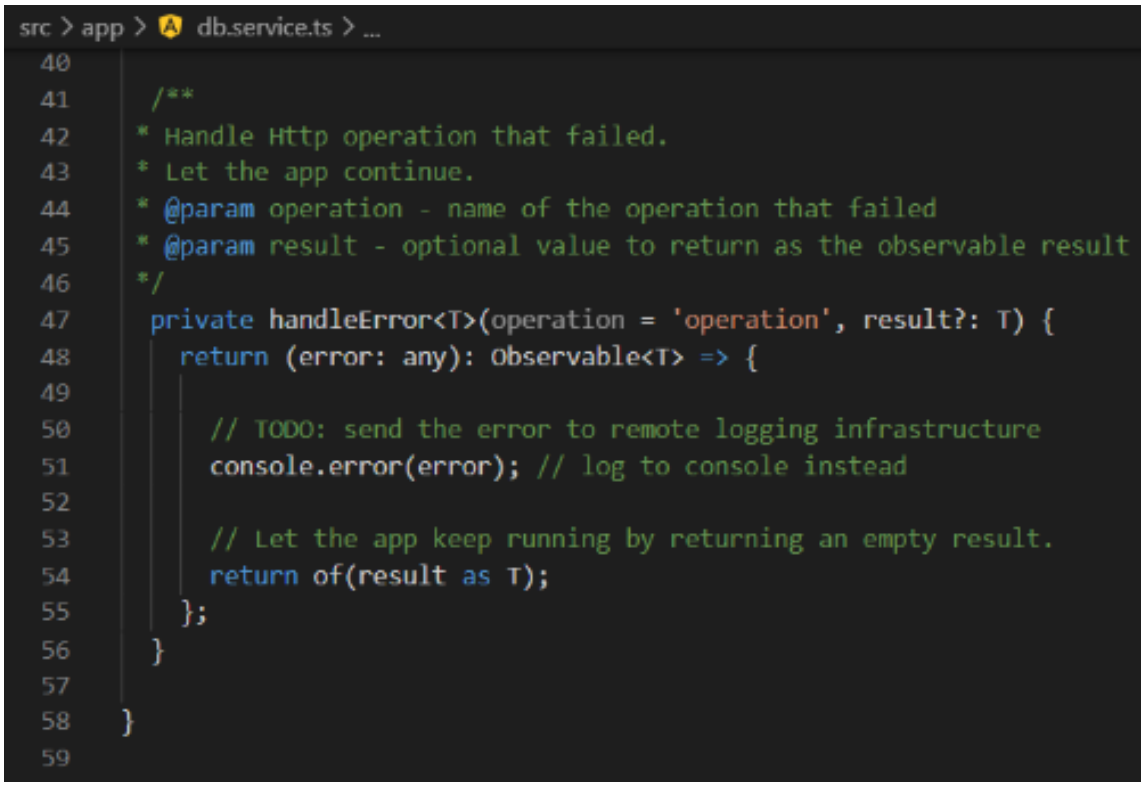


Ilustración 78. Imagen de la clase Db

Este método se ocupa de manejar las operaciones http que fallen.

14.2.11. db

Esta clase es de tipo enum, conteniendo el tipo de elementos que tendrá cada diálogo.

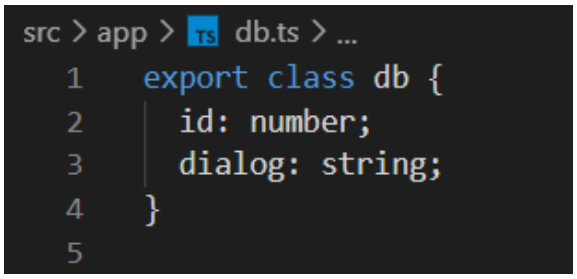


Ilustración 79. Imagen del enum db

Existe un número identificador único para cada diálogo (id) y un contenido del diálogo(dialog), es decir, qué dice el personaje en cada momento.

14.2.12. Profile

Este componente, desgraciadamente y por falta de tiempo, no está implementado, sólo tiene un boceto de la parte visual, en el que se muestra a Space George, junto con dos botones. El objetivo de esta clase es manejar toda la funcionalidad que hay detrás del personalizado del personaje.

Ilustración 80. Imagen de la clase Profile