UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación web para fomentar el aprendizaje emocional en personas con trastorno del espectro autista.

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Granell Sendra, Alejandro

Tutor: Muñoz García, Adolfo

Curso 2021-2022

Resumen

Los individuos con trastorno del espectro autista suelen tener dificultades en el reconocimiento y comprensión tanto de sus propias emociones como la de las personas que les rodean. Esto les supone una gran limitación en sus interacciones sociales ya que malinterpretan situaciones y pueden responder con conductas inadecuadas.

Ante esta enfermedad sin tratamiento curativo pretendemos realizar una aplicación web que sirva de pequeña ayuda para que estas personas mejoren sus habilidades sociales mediante un juego en el que se reta al usuario a identificar y copiar expresiones faciales fotográficas y simbólicas (como emoticonos). Para ello integraremos los servicios de inteligencia artificial de Azure para reconocer los gestos faciales del usuario, captados directamente por *webcam*; el motor de Unity para desarrollar una interfaz atractiva y un conjunto de bases de datos fotográficas de uso libre para la realización de los distintos niveles.

El método que seguiremos para trabajar las habilidades emocionales se basará en la identificación tanto de expresiones faciales como en la comprensión de la relación entre situación y emoción.

Durante todo este proceso nos centraremos en las seis emociones básicas que estableció Paul Ekman: asco, miedo, sorpresa, alegría, enfado y tristeza, las cuales son esenciales para construir el resto de emociones complejas que somos capaces de sentir.

Palabras clave: Unity, Azure Cognitive services, Azure Face, RESTful API, Serious game, Trastorno del espectro autista, Emociones

Abstract

Individuals with autism spectrum disorder often have difficulties in recognizing and understanding both their own emotions and those of the people around them. This is a major limitation in their social interactions since they misinterpret situations and can respond with inappropriate behaviors.

Faced with this disease without curative treatment, we intend to make a web application that will serve as a small help for these people to improve their social skills through a game in which the user is challenged to identify and copy photographic and symbolic facial expressions (such as emoticons). To do this, we will integrate Azure artificial intelligence services to recognize the user's facial gestures, captured directly by webcam; the Unity engine to develop an attractive interface and a set of free-use photographic databases to create the different levels.

The method that we will follow to work on emotional skills will be based on the identification of both facial expressions and the understanding of the relationship between situation and emotion.

Throughout this process we will focus on the six basic emotions that Paul Ekman established: disgust, fear, surprise, joy, anger and sadness, which are essential to build the rest of the complex emotions that we are capable of feeling.

Key words: Unity, Azure Cognitive services, Azure Face, RESTful API, Serious game, Autism spectrum disorder, Emotions

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

A mi tutor, Adolfo Muñoz García, por su trabajo en la supervisión de este proyecto.

A mis compañeros Pablo, Marcos y Sonia por darme ese pequeño empujón cuando ha sido necesario.

A mi prima Coral, psicopedagoga, por ayudarme a entender la problemática de los niños con trastorno del espectro autista.

A mi madre y a Tania por estar siempre ahí apoyándome.

CAPÍTULO 1
Introducción

Los **trastornos del espectro autista (TEA)** son un conjunto de alteraciones neurológicas que afectan al desarrollo infantil. Se caracterizan por déficits persistentes en la comunicación e interacción social y por la presencia de patrones repetitivos y restringidos de la conducta.

Estos trastornos que afectan a 1 de cada 160 niños en todo el mundo, suponen un impacto muy importante en la autonomía y en el desarrollo de las personas afectadas y su familia.

1.1 Motivación

Dadas las limitaciones de relación que tienen estos niños, se pretende ayudarles a ampliar sus habilidades de comunicación y el reconocimiento de emociones; y de esta forma permitir mejorar la calidad de vida de los menores y de sus cuidadores.

Nos centraremos en los dos principales problemas sociales que tienen los niños con TEA:[1]

- La falta de comprensión de las emociones de las personas de su alrededor, ya que las analizan de forma aleatoria.
- La dificultad para utilizar sus propias emociones en las interacciones sociales, ya que suelen malinterpretar las situaciones y responder de forma inadecuada.

1.2 Objetivos

El objetivo de este trabajo consistirá en diseñar e implementar una aplicación web, que se llamará **Faces to learn**, que intentará ayudar a mejorar la inteligencia emocional de los niños con TEA.

Para ello se realizará de la forma más amena posible, como un juego con las características propias que se requieren en estos casos: contenidos claros y visuales,

con la menor información verbal posible, utilizando oraciones sencillas, actividades de corta duración, aclarando conceptos para facilitar el seguimiento. Y todo esto al realizarse dentro del entorno natural del niño le creará un ambiente más estructurado y tranquilo para trabajar.

Durante todo este proceso nos centraremos en las seis emociones básicas que estableció Paul Ekman: asco, miedo, sorpresa, alegría, enfado y tristeza, las cuales son esenciales para construir el resto de emociones complejas que somos capaces de sentir.[2]

Dadas las características especiales de estos niños se orientará este proyecto hacia la sencillez, basándose, por ejemplo en:

- Un menú principal y una selección de niveles sencillos e intuitivos.
- Interfaz llamativa, colorida, con sonidos y música atractivos.
- Libertad en la elección de niveles, para evitar un estancamiento en el seguimiento del juego.
- Una puntuación de los niveles mediante un sistema de recompensa de estrellas, según el número de objetivos que se consigan en dicho nivel e incluso con la posibilidad de repetirlos para aumentar la puntuación anterior.

Faces to learn se desarrollará mediante el motor de videojuegos Unity con la integración de los servicios de Azure Cognitive que se implementarán haciendo uso de llamadas RESTful API.

No se pretende implementar todas las posibles funcionalidades de la aplicación, sino una demo para mostrar las posibilidades de las que se dispone. Más adelante, se podrían plantear otras funcionalidades como las que se describen en el apartado Trabajo futuro.

1.3 Gamificación

La Gamificación es una técnica de aprendizaje que intenta conseguir mejores resultados utilizando la mecánica de los juegos en el ámbito educativo o profesional. Esta metodología, debido a su carácter lúdico, facilita la adquisición de conocimientos de una forma más divertida para el usuario. [3]

Para poder conseguir que un entorno no lúdico consiga incentivar y motivar a los usuarios como lo hace un juego hay que utilizar las herramientas de éstos.

Se utilizan la mecánica de juego y la dinámica de juego.

La primera de ellas hace referencia a la forma de recompensar al usuario en función de los objetivos alcanzados. Las técnicas más utilizadas son:

Figura 1.1: Técnicas de recompensa en gamificación.

La dinámica de juego hace referencia a la motivación del propio usuario para jugar y continuar jugando y así conseguir sus objetivos. Son, por ejemplo:

Figura 1.2: Técnicas de motivación en gamificación.

En nuestro caso pretendemos que el jugador vaya superando retos con el fin de lograr una satisfacción personal.

1.4 Convenciones

El texto de la memoria incluirá las siguientes normativas de marcado:

- Las palabras extranjeras y anglicismos se remarcarán en cursiva, excluyendo nombres propios.
- Se entrecomillarán las citas textuales externas a la obra.
- Se remarcarán ciertas palabras importantes del proyecto en negrita.
- Se marcarán entre apóstrofos las palabras que se utilicen en sentido figurado.

Todas las palabras contenidas en la presente memoria que aparecen en género masculino deben leerse indistintamente en género masculino o femenino cuando hagan referencia a personas.

CAPÍTULO 2
Estado del arte

Desde el nacimiento de la informática, los usuarios han tratado de usar el ordenador como fuente de entretenimiento.

En sus inicios, los primeros juegos de ordenador eran programas muy simples que sólo pretendían divertir. Pero a medida que los equipos han aumentado su potencia y sus posibilidades gráficas se han desarrollado cada vez juegos más complejos, más adictivos y motivadores para el usuario.

Son innumerables los estudios científicos que han demostrado que los videojuegos potencian las funciones cerebrales y como pueden servir como herramientas terapéuticas para la detección, rehabilitación o prevención de muchas enfermedades.

Entre unos de estos estudios está el realizado por el grupo de investigación Cognitive Media Technologies, con investigadores del Hospital de Sant Joan de Déu y de Mutua Terrassa, llevado a cabo en un grupo de niños con TEA de 4 a 6 años de edad y publicado en Research in Autism Spectrum Disorders.[4]

Según este estudio, "las intervenciones basadas en juegos que implican el uso de la tecnología han demostrado facilitar la motivación y los procesos de aprendizaje en niños con Trastornos del Espectro Autista (TEA).

El videojuego creado por este grupo (Pico's Adventure) tiene por objetivo facilitar la interacción social de los niños con autismo a través de una serie de experiencias lúdicas y colaborativas."

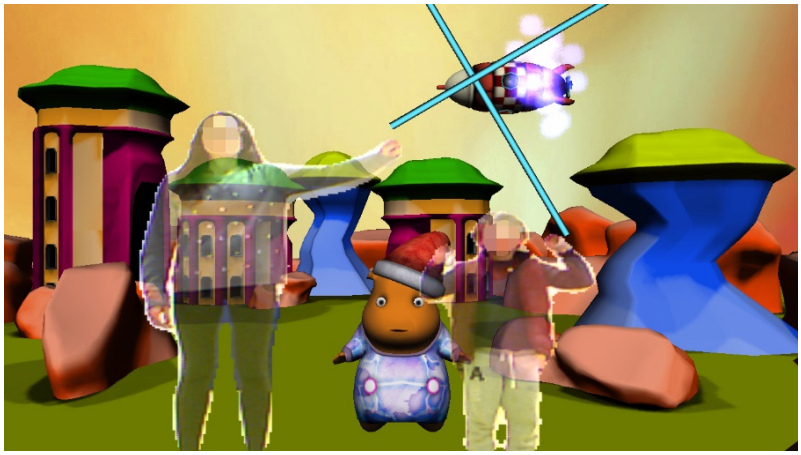

Figura 2.1: Captura de una imagen del videojuego Pico's Adventure, nivel 3.

Como manifiesta el investigador principal Narcís Parés, "los primeros estudios experimentales mostraron su eficacia como complemento de las terapias convencionales". "Desde entonces, Pico's Adventure, ha pasado a ser un importante referente en la búsqueda de herramientas basadas en Tecnologías de la Información y la Comunicación (TIC) para fomentar las conductas de iniciación social en niños con trastornos del espectro autista".

"Los resultados han mostrado que el videojuego favorece más las conductas de iniciación social que el juego libre, en niños con TEA cuando jugaban solos o por parejas, entendiendo como juego libre al juego con juguetes (cochecitos, muñecos, pelotas, etc.), sin guión ni reglas. Además, el videojuego ha mostrado ser igualmente eficaz en la reducción de comportamientos repetitivos y en el aumento de la expresión gestual de los niños."

"Los videojuegos podrían considerarse como herramienta adecuada para fomentar los comportamientos sociales y ser útil también como complemento de los tratamientos habituales, pero "se necesita seguir trabajando para poder apoyar esta hipótesis", indican los investigadores."

2.1 Crítica al estado del arte

En el contexto tecnológico actual se han encontrado diferentes aplicaciones que realizan funcionalidades parecidas a las que se propone desarrollar en este trabajo fin de grado (TFG).

Un ejemplo de este tipo de aplicaciones es Emocionatest, aplicación desarrollada por la Universidad Jaume I que permite evaluar y potenciar la competencia emocional y la capacidad para reconocer y gestionar las emociones de niños con Trastornos del Espectro Autista (TEA) pero también aplicable a la población infantil en general. [5]

Figura 2.2: Captura de pantalla de la aplicación Emocionatest.

Esta aplicación que ya no está disponible en la actualidad, era un entorno interactivo que permitía, entre otras opciones, relacionar sentimientos interiores con expresiones faciales, sentimientos antes y después de una situación concreta de la vida cotidiana... Pero, por el contrario, no contaba con niveles que incentivaran el trabajo de memoria, no utilizaba imágenes de personas reales ni el reconocimiento facial del usuario.

Otra aplicación de este tipo es Autimo – Descubra emociones que, según sus creadores, fue diseñada para ayudar a las personas autistas a aprender a reconocer las emociones y las expresiones faciales a través tres divertidos juegos (parejas, intrusos y adivinanzas). [6]

Figura 2.3: Captura de pantalla de la aplicación Autimo - Descubra emociones.

Como aspectos positivos destaca que se pueden añadir fotos, se puede grabar su propia voz para felicitaciones y que permite un análisis del progreso. Como parte negativa esta aplicación es parte de una suscripción y sólo se ofrece la versión completa de forma gratuita por 3 días. Según las reseñas en el apartado de opiniones acerca de la app, los usuarios comentan que para seguir avanzando hay que hacer el pago y que éste no está acorde a la relación calidad-precio.

Otro recurso también similar es la web Even better games con actividades diseñadas por la psicóloga Pilar Chanca Zardaín, totalmente gratuitas, que sirven para trabajar las habilidades socio-emocionales de niños con trastornos del espectro autista, entre ellos el síndrome de Asperger. Consta, de momento, de cuatro juegos educativos (Nuestras emociones, Cada oveja con su pareja, ¡Esfúmate! Y Ruleta de las emociones) que se pueden jugar en tres idiomas diferentes: Español, Inglés y Francés.[7]

Figura 2.4: Capturas de pantalla de la web Even Better Games.

Esta utilidad tiene, a destacar, como parte de su contenido, un apartado que ayuda al niño a reconocer las principales emociones explicándole los distintos cambios gestuales que se producen en la expresión facial.

Sin embargo, tiene algunos aspectos de desarrollo que se quedan un poco débiles. Por ejemplo, a veces, no queda muy claro cuando se ha cometido un error (que sólo se indica por un cambio bastante sutil de color del marco de la imagen); también se puede pasar de pantalla sin completarla y sin saber que no se ha completado.

Podemos encontrar, también, diferentes páginas web de recopilación de materiales con la misma finalidad de trabajar las emociones y mejorar la calidad de vida de las personas con TEA.

Son por ejemplo:

- Soluciones tecnológicas autismo - Fundación Orange (fundacionorange.es).

  La Fundación Orange promueve diferentes aplicaciones, programas de descarga gratuita, páginas web... en colaboración con universidades, empresas y asociaciones de usuarios.[8]

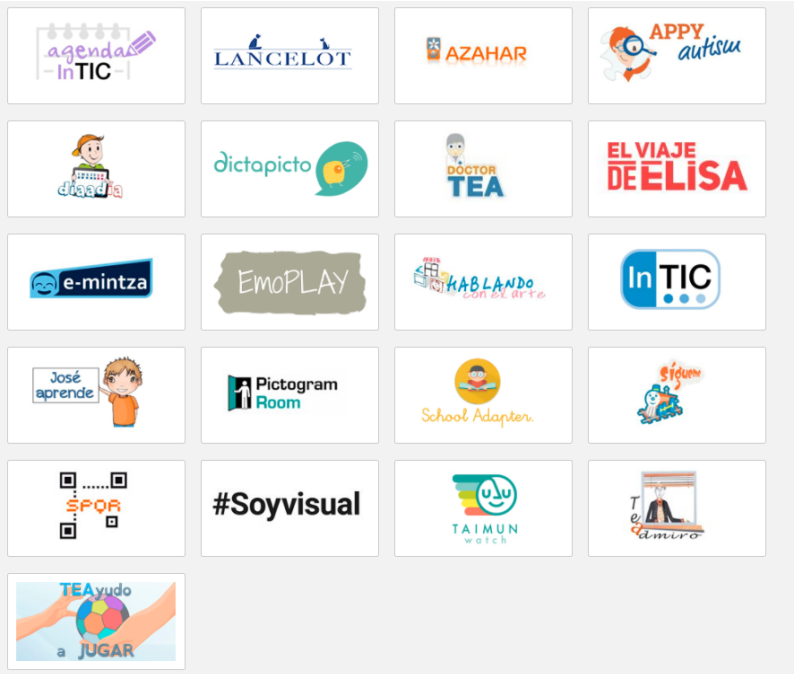

Figura 2.5: Captura de pantalla de las aplicaciones gratuitas de la Fundación Orange.

- Materiales para trabajar las Emociones en el TEA - Fundación Pictoaplicaciones.

  El contenido de esta web es una recopilación de diversas publicaciones y vídeos también para trabajar las emociones. Sin embargo, la mayoría ya no están disponibles (página no encontrada). Y del material que aún permanece, en ocasiones, es excesivamente sencillo, no adecuado para el rango de edad infantil o para personas con trastorno del espectro autista, son manuales nada interactivos, etc.[1]

2.2 Propuesta

Se han encontrado diferentes materiales en Internet que pueden ayudar a los niños con TEA a mejorar sus habilidades emocionales. Para el desarrollo de esta aplicación se utilizaran algunos de los recursos más positivos encontrados:

- Utilizar imágenes de personas reales.
- Uso de una interfaz adecuada al publico infantil y sobretodo sencilla ya que va dirigida concretamente a niños con TEA.
- Sistema de superación de retos asociado a otro de recompensa de estrellas que les motivará a continuar jugando para conseguir sus objetivos.

Pero lo más importante que se añadirá a la aplicación y que no se ha encontrado en otras es el reconocimiento facial por *webcam*, lo que supondrá un aliciente al usuario al reconocerse como protagonista del juego.

CAPÍTULO 3
Análisis del problema

3.1 Metodología

Una metodología de desarrollo de software consiste en hacer uso de diversas herramientas, técnicas y métodos para el desarrollo del mismo. Su objetivo es la formalización de las actividades relacionadas con la elaboración de software.

Existen dos corrientes de metodologías principales: las metodologías tradicionales y las metodologías ágiles.

Dado a que trabajaré yo solo en este proyecto y voy a tratar con tecnologías nuevas para mí, haré uso de metodologías ágiles que permitirán ir adaptando la forma de trabajo constantemente y, así, ir evaluando de forma continua los requisitos, los planes y los resultados con lo que se podrá responder con rapidez ante los cambios.

Esta forma de trabajo flexible es totalmente contraria a la denominada metodología tradicional que se desarrolla de forma lineal o 'en cascada', de forma que no se pasa a la siguiente fase hasta haber terminado completamente la anterior, lo que supone, generalmente, un incremento de tiempo para finalizar el proyecto al igual que éste podría no llegar a adaptarse a lo que se pretendía inicialmente y tener que volver a empezar de cero. Además de que no permitiría incluir nuevas funcionalidades al proyecto una vez comenzada su implementación.

Al utilizar metodologías ágiles, se pueden definir diversas tareas y priorizarlas, pudiendo adaptar así lo que entra en el *scope*.[1]

3.1.1. Matriz de prioridades de *features*[2]

Es posible hacer una estimación del esfuerzo e impacto de cada tarea. Gracias a esto, se puede establecer el orden de implementación de las mismas utilizando una matriz de prioridades.

[1] El *scope* o alcance del proyecto define los objetivos que se intentarán alcanzar durante el desarrollo del mismo.

[2] Feature: unidad funcional de un sistema de software que satisface un requisito, representa una decisión de diseño, y puede generar una opción de implementación.

Con esta, podremos determinar la prioridad de las distintas tareas, en función del impacto que van a conseguir al implementarlas con el esfuerzo que supone su realización.

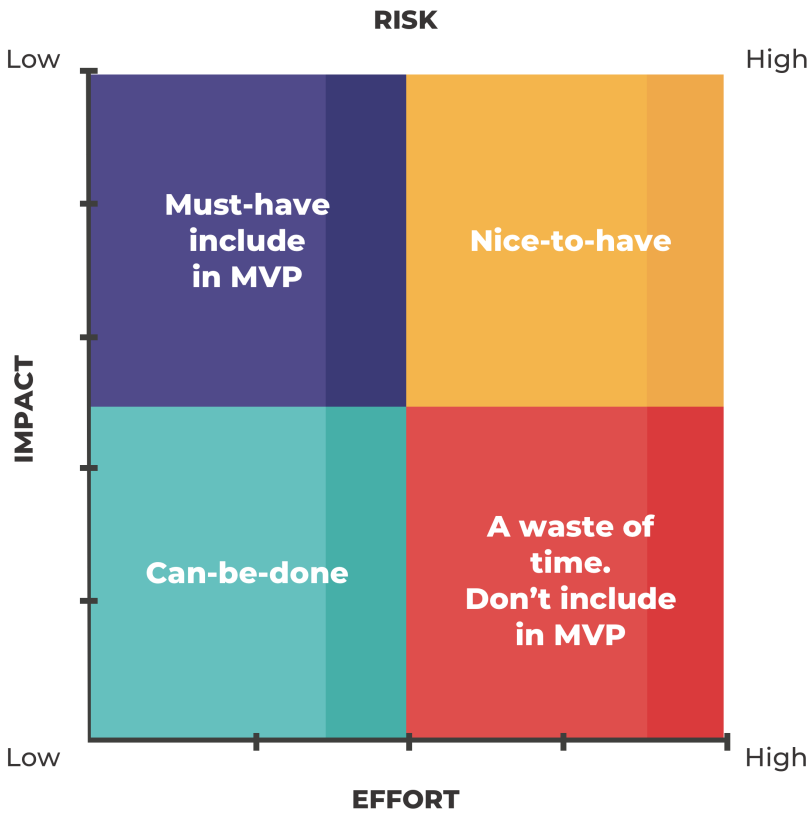

Figura 3.1: Gráfico de una matriz de prioridades.

Nuestra matriz de 2x2 nos va a mostrar cuatro cuadrantes de priorización:

- *Must have*
  1. Captura de imágenes mediante la *webcam*.
  2. Reconocimiento de emociones en imágenes.
- *Nice to have*
  1. Multiplataforma.
  2. Interfaz llamativa, colorida, con sonidos y música atractivos.
- *Can be done*
  1. Puntuación de niveles.
  2. Carga de niveles desde la base de datos.
  3. Puntuación de niveles.
  4. Sistema de usuarios.

- *Don't include*

    1. Implementación base de datos documental.
    2. Publicación en Stores.

De forma que las *features* asignadas a la categoría *must have* son las primeras que se deben realizar.

3.1.2. *Kanban*

Kanban es una palabra japonesa que significa 'tarjetas visuales', donde Kan es visual, y Ban corresponde a tarjeta.

La aplicación del método Kanban implica la generación de un tablero de tareas que permitirá mejorar el flujo de trabajo. Al ser un equipo de desarrollo de una sola persona, se minimiza el trabajo en la especificación de las distintas tareas y permite un mejor aprovechamiento del tiempo de trabajo.

Para este proyecto, se hará uso de una página web llamada Trello. Esta cuenta con un plan gratuito, ideal para proyectos pequeños y simples, donde se pueden crear hasta 10 tableros diferentes.

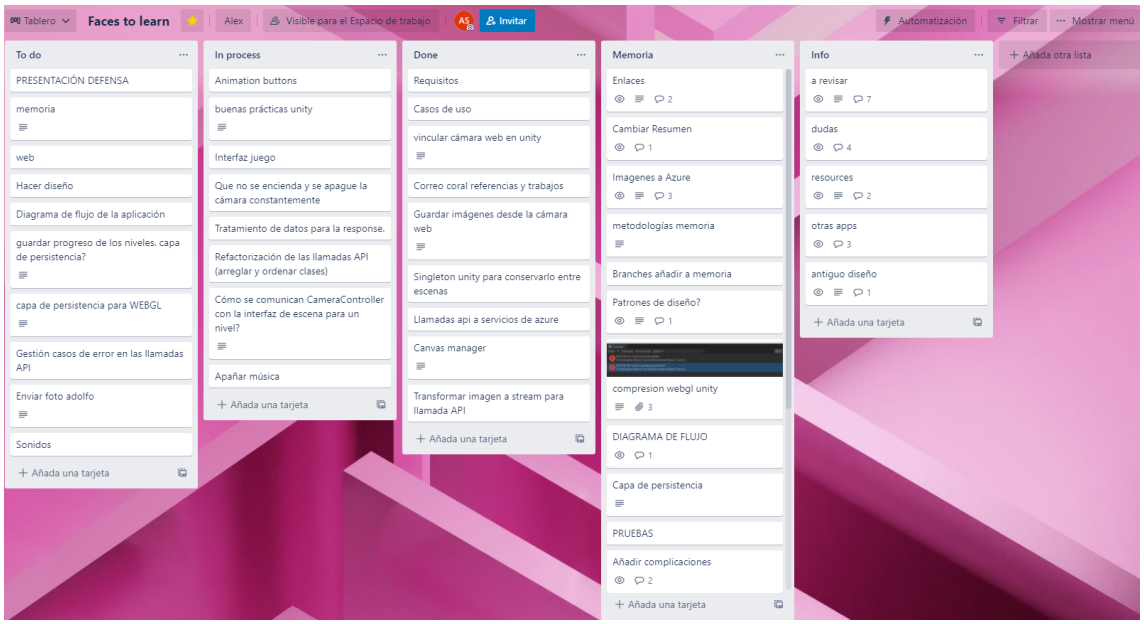

Figura 3.2: Captura de pantalla del tablero Trello utilizado en este proyecto.

3.2 Especificación de requisitos

3.2.1. Introducción

Propósito

El propósito de esta sección tiene como finalidad conocer lo que se demanda y espera obtener de la aplicación, facilitando y guiando el desarrollo de la misma.

Ámbito

Se ha decidido crear una aplicación web educativa que se llamará Faces to learn para ayudar a los niños con trastorno del espectro autista a desarrollar sus habilidades sociales mediante la replicación de emociones en imágenes faciales.

3.2.2. Descripción general

Perspectiva del producto

La aplicación que se desarrollará hará uso de otros sistemas para el análisis emocional de las imágenes proporcionadas por los usuarios y/o sistemas que proporcionen imágenes para la creación de los distintos niveles.

No hay intención en monetizar la aplicación.

Características de los usuarios

La primera versión de la aplicación solo distinguirá un tipo de usuario, que será el jugador.

Restricciones generales

Para hacer uso de la aplicación con todas sus funcionalidades y una presentación adecuada se recomienda Google Chrome, Mozilla Firefox, Safari o Microsoft Edge. La aplicación generará archivos JavaScript que el propio navegador se encargará de *renderizar* y ejecutar las operaciones pertinentes sin necesidad de ningún complemento adicional.

3.2.3. Requisitos específicos

Requisitos funcionales

A continuación, se especificarán los requisitos funcionales entendidos como un conjunto de entradas, comportamientos y salidas. Éstos pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que nuestro sistema debe cumplir.

Tabla 3.1: Requisitos funcionales.

Id	Nombre de requisito	Descripción	Prioridad (1-5)
[RF1]	Selección de dificultad	Desde el menú principal se podrá hacer clic en un botón para elegir la dificultad deseada y el sistema cambiará la pantalla a la selección de niveles de la misma.	3

[RF2]	Cargar progreso	Cuando se inicie la aplicación, el sistema cargará el progreso de los niveles ya completados por el usuario en su anterior sesión de juego.	4
[RF3]	Acceder a ajustes	Desde el menú principal, el usuario podrá hacer clic en un botón para abrir la ventana de ajustes.	5
[RF4]	Gestión de sonido	El usuario podrá activar/desactivar los efectos de sonido desde el menú de ajustes. Requisitos relacionados: [RF3].	5
[RF5]	Selección de nivel	Una vez seleccionada la dificultad el usuario podrá hacer clic en un nivel y el sistema deberá mostrar la pantalla del mismo. Requisito relacionado: [RF2].	3
[RF6]	Gestión *webcam*	Al acceder a un nivel, el sistema deberá ser capaz de detectar y mostrar por pantalla la *webcam* del usuario.	1
[RF7]	Panel de error en la *webcam*	Si el sistema no detecta ninguna *webcam* compatible para el uso, deberá mostrar por pantalla un mensaje de error avisando al usuario. Requisito relacionado: [RF6].	5
[RF8]	Carga de imágenes	Al acceder a un nivel, el sistema deberá mostrar en pantalla las imágenes con emociones faciales relacionadas al nivel que el usuario deberá interpretar, reconocer y replicar.	2
[RF9]	Volver al menú principal	El usuario podrá hacer clic en un botón y el sistema debe volver al menú principal.	4
[RF10]	Tomar foto	Desde la pantalla de un nivel, una vez mostrada la *webcam* del usuario, este podrá hacer clic en un botón y el sistema deberá congelar la imagen de la *webcam*.	1
[RF11]	Aceptar foto	Una vez tomada una foto, el usuario podrá hacer clic en un botón de confirmación y el sistema deberá guardar la imagen y enviarla al servicio de análisis de imágenes correspondiente. Al finalizar se esconderán los botones de confirmación. Requisito relacionado: [RF10].	1

[RF12]	Cancelar foto	Una vez tomada una foto, el usuario podrá hacer clic en un botón de cancelación y el sistema deberá eliminar la fotografía tomada y volver a mostrar la *webcam* del usuario. Al finalizar se esconderán los botones de confirmación. Requisito relacionado: [RF10].	4
[RF13]	Tratamiento de la respuesta	El sistema deberá recibir y tratar la respuesta del servicio de análisis de imágenes y compararlo con los parámetros del nivel.	1
[RF14]	Completar nivel	Si el usuario ha superado los parámetros establecidos del nivel, el sistema deberá mostrar una pantalla de resultado felicitando al usuario, desde donde este podrá volver al menú principal o elegir un nuevo nivel. Requisitos relacionados: [RF13], [RF9].	3
[RF15]	Fracasar nivel	Si el usuario no ha superado los parámetros establecidos del nivel, el sistema deberá mostrar una pantalla de resultado alentando al usuario a volver a intentarlo. Este podrá volver al menú principal o reiniciar el nivel. Requisitos relacionados: [RF13], [RF9].	4
[RF16]	Reintentar nivel	Si el usuario ha fracasado el nivel, este podrá hacer clic en un botón y el sistema deberá volver a mostrar y reiniciar la pantalla del nivel en el que se encontraba. Requisito relacionado: [RF15].	4
[RF17]	Guardar progreso	Cada vez que el usuario termine un nivel, el sistema deberá guardar un estado del progreso de los mismos. Requisito relacionado: [RF2].	4
[RF18]	Gestión de errores en la respuesta	Si el sistema de análisis de imágenes nos da como respuesta un error, el sistema debe tratarlo de una forma correcta y avisar por pantalla al usuario. Requisito relacionado: [RF13].	4
[RF19]	Salir del juego	Desde el menú principal, el usuario podrá hacer clic en un botón y el sistema deberá ser capaz de cerrarse de forma segura.	5

Requisitos no funcionales

A continuación se detallarán aquellos requisitos que pueden usarse para calificar operaciones del sistema. No describen funciones a realizar ni información a guardar sino características de funcionamiento.

1. Toda funcionalidad del sistema debe responder al usuario en menos de 2 segundos.

2. La aplicación se visualizará en la pantalla del usuario de forma sencilla e intuitiva.

3. La aplicación no violará la privacidad del usuario, y la información entre aplicación y servidor estará encriptada.

4. La aplicación no necesitará ninguna interfaz hardware específica y puede ser visitada desde cualquier ordenador que pueda conectarse a internet y navegar.

3.3 Casos de uso

Los diagramas de casos de uso describen el comportamiento de la aplicación y enfatizan lo que debe suceder en el sistema modelado. Podemos observar dos posibles situaciones: cuando el usuario se encuentra en el menú principal y cuando está dentro de un nivel.

3.3.1. Menú principal

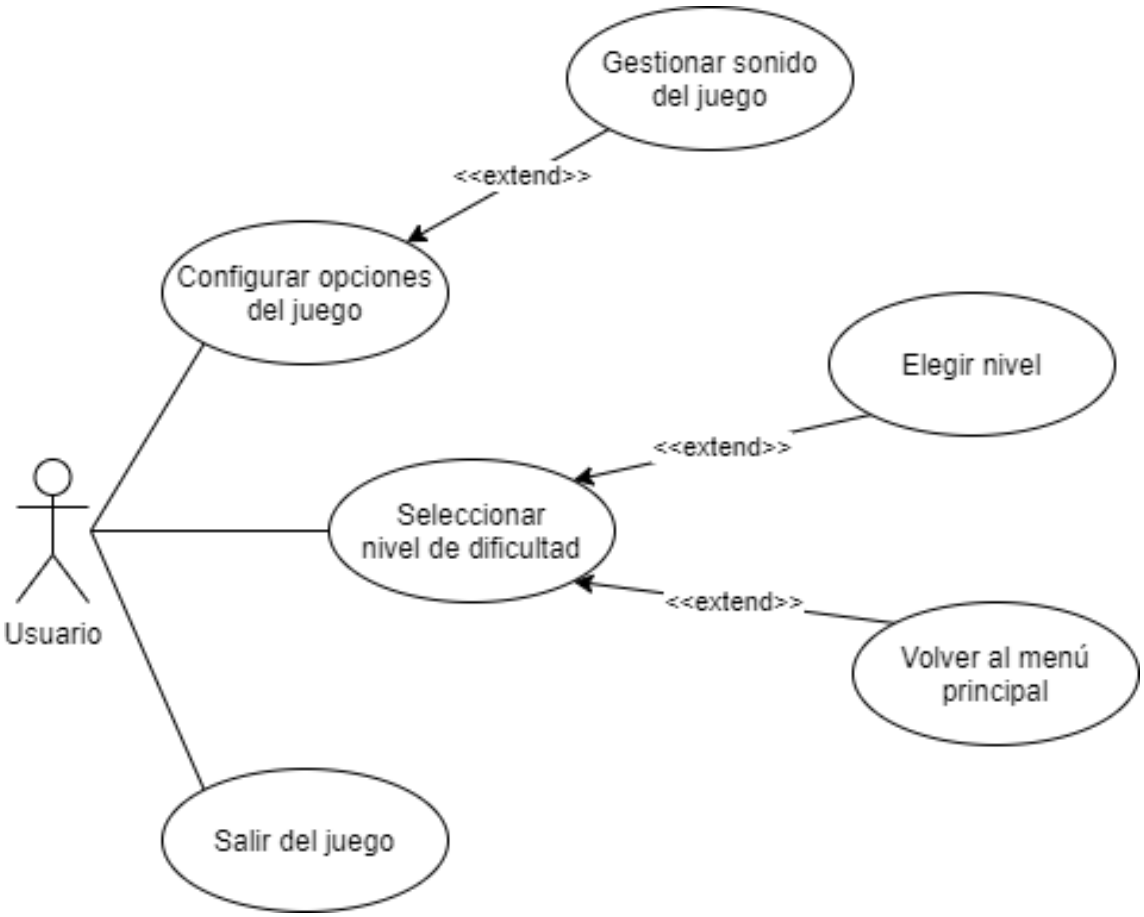

Figura 3.3: Casos de uso en el menú principal.

3.3.2. Nivel

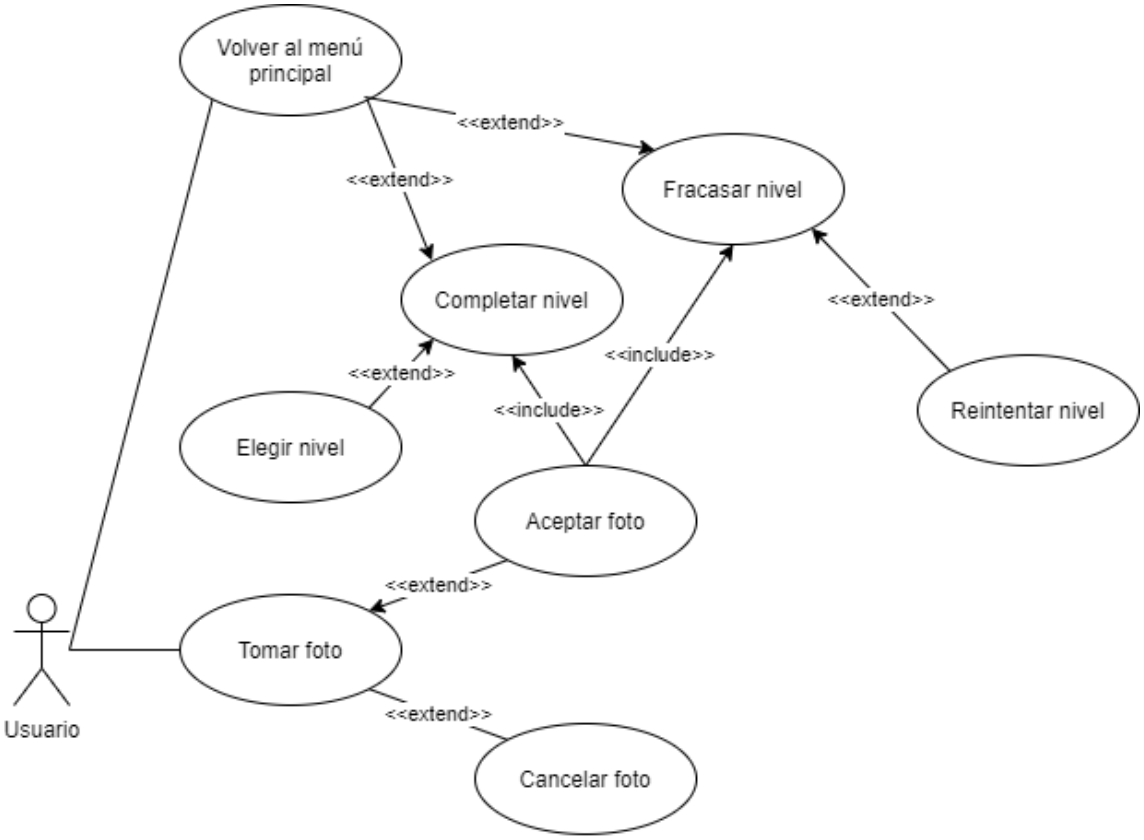

Figura 3.4: Casos de uso dentro de un nivel.

3.4 Análisis de las soluciones

3.4.1. Motores

Un motor de videojuegos es un conjunto de herramientas o *framework* que permiten, al programador, agilizar el proceso de desarrollo de un videojuego y así dedicar menos tiempo a aspectos básicos y enfocarse en desarrollar buenos juegos sin perder tiempo en otras tareas. Y, de esta manera, no es necesario reinventar la rueda cada vez que desarrollamos un videojuego.

En la última década ha habido un auge del número de motores gráficos disponibles para los desarrolladores. Amén de que muchos de estos cuentan con licencias gratuitas para facilitar el desarrollo de videojuegos en estudios pequeños.

Para retratar una imagen general de este ámbito del sector, se analizarán tres de los motores más relevantes a día de hoy.

Unity

Unity es un potente motor de videojuegos multiplataforma creado por Unity Technologies.

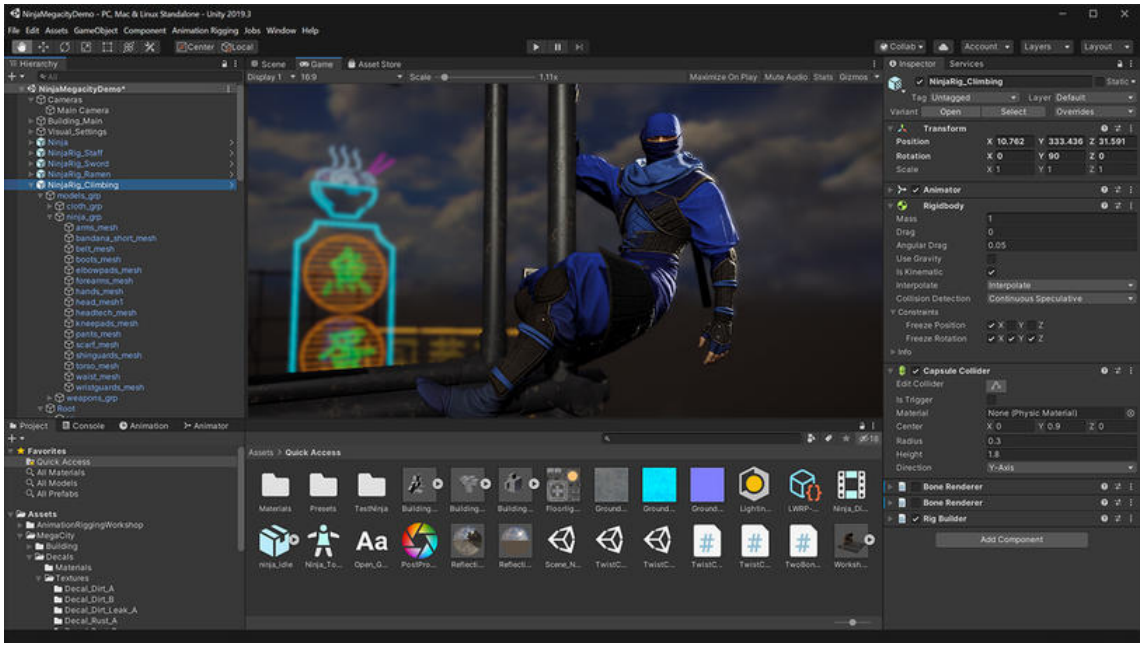

Figura 3.5: Captura de pantalla del editor de Unity.

Ventajas:

- Facilidad de uso.
- Curva de aprendizaje amigable.
- Permite crear videojuegos tanto en 2D como 3D, incluso sin conocer la mayoría de las posibilidades de la herramienta.
- Soporta tres tipos de lenguaje para los *scripts*: C# (el más utilizado), UnityScript y Boo.
- Recursos externos en la Asset Store, la tienda más completa de complementos para juegos (escenarios, armas, sonidos, módulos de control...). Permite ahorrar mucho tiempo de trabajo.
- Herramientas de animación y cinemáticas.
- Multiplataforma. Despliegue a más de 19 plataformas de manera sencilla.
- Posibilidad de licencia gratuita. Se debe pagar a la compañía en función de las ganancias.
- Gran cantidad de documentación disponible en cuanto a manuales y tutoriales.

Inconvenientes:

- El proyecto ocupa gran tamaño (aunque no el ejecutable final).
- No consigue el mejor rendimiento ya que no aprovecha al máximo los núcleos extra de la mayoría de dispositivos.
- Los desarrolladores no disponen del código fuente del motor.

CryEngine

CryEngine, desarrollado por Crytek, es uno de los motores más potentes que existen. Sin embargo, no es tan conocido como Unity o Unreal.

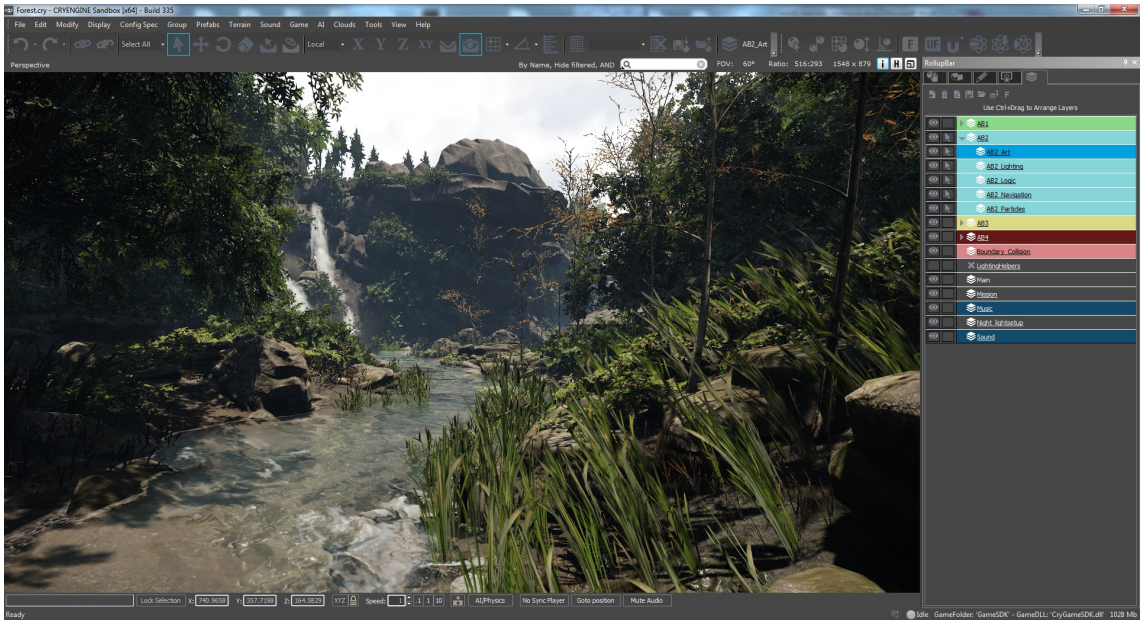

Figura 3.6: Captura de pantalla del editor de CryEngine.

Ventajas:

- Gran capacidad gráfica.
- Permite crear escenarios de gran calidad.

Inconvenientes:

- Es uno de los motores más complicados, por lo que no es la mejor opción para aprender a crear videojuegos.
- Necesario apuntarse a un sistema de suscripción para poder usarlo libremente.
- No es el mejor motor para hacer juegos en 2D.

Unreal Engine

Unreal Engine, creado por la compañía Epic Games, es un motor de videojuegos para PC y plataformas de consola especializado en 3D. Nacido con el juego Unreal, es sin duda alguna el más extendido en el mundo profesional del videojuego.

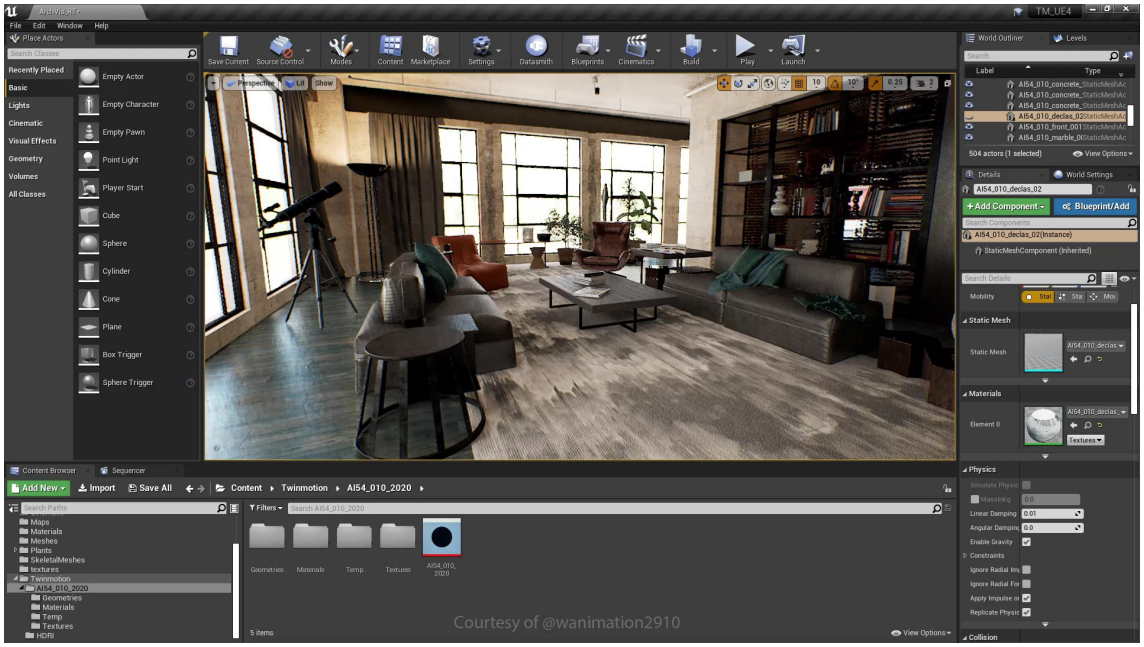

Figura 3.7: Captura de pantalla del editor de Unreal.

Ventajas:

- Está escrito en C++, aunque existe la posibilidad de utilizar *Blueprints*.
- Posibilidad de uso gratuito (si se utiliza con un propósito no lucrativo). Se debe pagar a la compañía en función de las ganancias.
- Una de las mejores herramientas para crear juegos 3D y de realidad virtual.
- Gran rendimiento.
- El mejor motor en cuanto a gráficos realistas.
- Gran variabilidad de usos y aplicaciones (ingeniería, medicina, arte, arquitectura...).
- El código fuente del motor está disponible para los desarrolladores.

Inconvenientes:

- Dificultad de uso para estudios pequeños.
- Comunidad de usuarios más reducida, por lo que encontrar documentación es más difícil.
- No es la mejor herramienta para crear juegos 2D, aunque es posible hacerlo.
- Gran tamaño de los proyectos y consume bastante memoria lo que hace que pueda no ser soportada por cualquier ordenador.

Código vs *Blueprints*

Blueprints es un lenguaje de programación que permite crear complejas funciones y elementos interactivos sin necesidad de conocimientos de programación. Son assets dentro del editor de Unreal Engine y consisten en un sistema de secuencias de comandos visuales, a modo de 'cajas' o nodos, que son funciones y que uniéndolas entre sí consiguen la funcionalidad deseada.

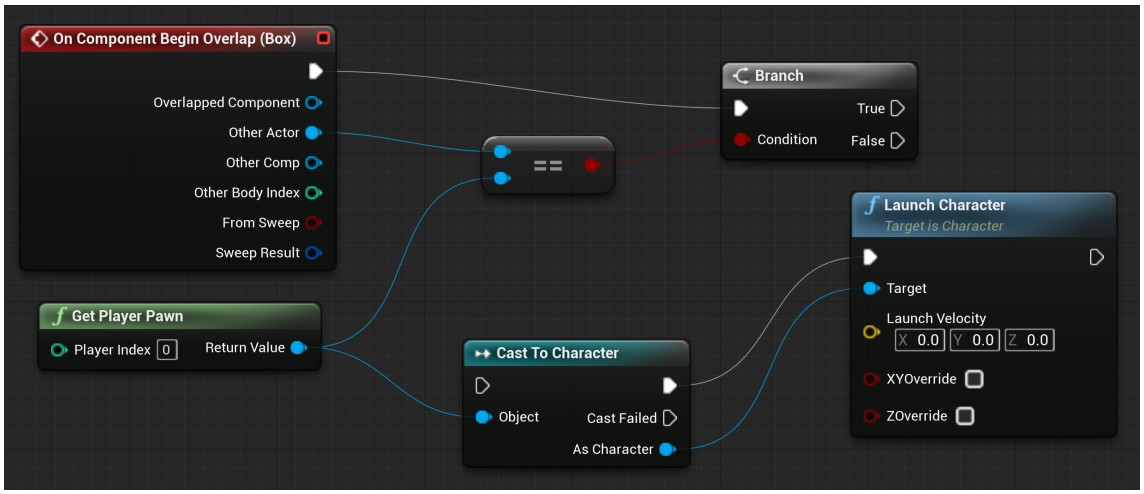

Figura 3.8: Captura de pantalla de la ventana de grafo del editor Unreal.

Su objetivo es otorgar una herramienta intuitiva para no programadores, de forma que puedan crear mecánicas de juego sin tener que escribir lineas de código. Se caracteriza por su accesibilidad, sencillez y rapidez.

Se pueden usar para cualquier cosa, desde crear la lógica de un objeto, el material del mismo o la animación de un personaje... Sin embargo, conforme se van añadiendo más nodos para obtener nuevas funcionalidades, se puede convertir en una locura. Por eso, se puede preferir hacer uso código y tener así las clases más organizadas.

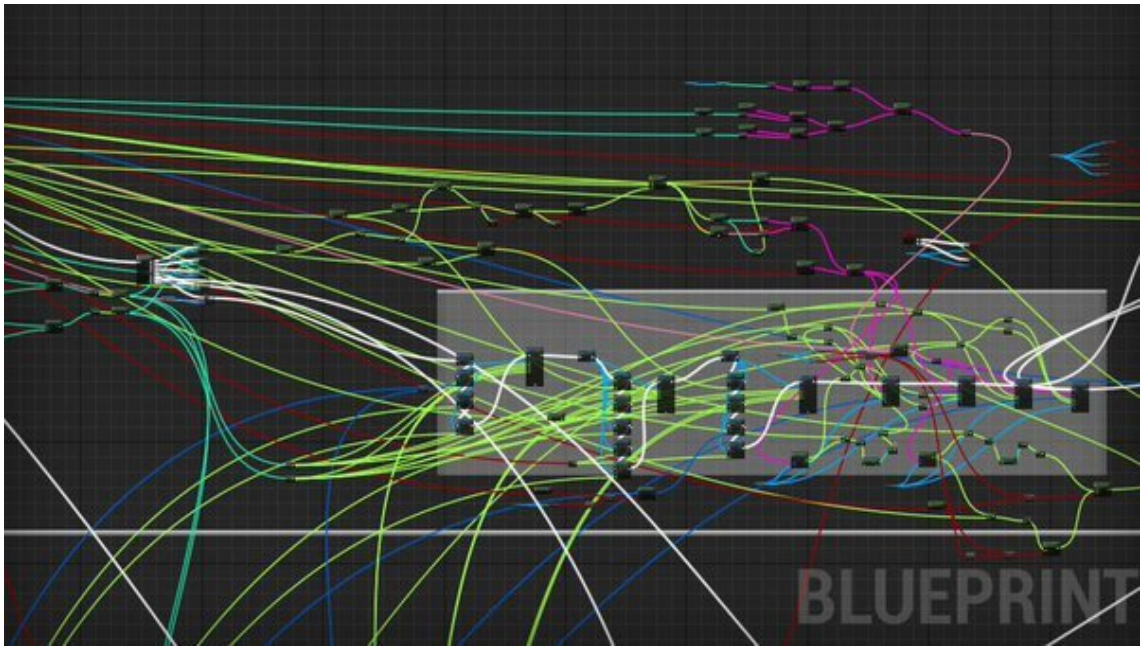

Figura 3.9: Ejemplo de programación en *blueprints* ilegible.

Además, los nodos que se pueden usar están limitados a lo que oferta Unreal Engine, y, por tanto, si se necesitan nuevas funcionalidades o se quiere usar unas propias para simular cualquier comportamiento, se tiene que hacer por código.

En Fortnite[3], un juego de Epic Games, gran parte de la optimización hecha en el juego fue migrar la lógica en *blueprints* a componentes en c++, dado que estos eran más eficientes.[4]

3.4.2. Reconocimiento de emociones

Para realizar el análisis emocional de las imágenes proporcionadas por el usuario se ha decidido hacer uso de diferentes servicios, ya que tener que diseñar, desarrollar y entrenar una inteligencia artificial (IA) sería demasiado costoso para este proyecto.

Azure Face

Azure Cognitive Services pone la inteligencia artificial al alcance de todos los desarrolladores sin necesidad de experiencia en *machine learning*[5]. Basta con gestionar llamadas API (Interfaz de Programación de Aplicaciones) para hacer uso de sus servicios en nuestras aplicaciones.

El catálogo de servicios de Azure Cognitive Services se puede dividir en cinco pilares principales: Vision, Voz, Lenguaje, Web Search y Decision.

[3]https://www.epicgames.com/fortnite/en-US/

[4]https://www.youtube.com/watch?v=KHWquMYtji0

[5]El machine learning o aprendizaje automático es la rama de la inteligencia artificial que dota a las máquinas de la habilidad de "aprender" a partir del análisis de datos con el fin de identificar patrones y apoyar en la toma decisiones con la mínima intervención humana.

Para nuestra aplicación, nos interesaremos por el apartado Vision. Más concretamente Azure Face. Este es un servicio de reconocimiento facial cuyas características incluyen:

- Detección de caras, rostros y atributos en una imagen.
- Identificación de la persona.
- Reconocimiento de emociones percibidas.
- Reconocimiento y agrupación de caras similares en imágenes.

Al ser parte de Microsoft, cuenta con una extensa documentación[6] y paquetes ya desarrollados en .net.

Amén de que se podrían agregar otros servicios como CosmosDB, una base de datos documental.

Existen diversos *tiers* de pago en función de lo que se vaya a utilizar y se cobra en función del número de transacciones.

Product	Features	Price
Computer Vision S1 up to 10 requests per second	Tag Face GetThumbnail Color Image Type GetAreaOfInterest	0-1M transactions — €0.894 per 1,000 transactions 1M-10M transactions — €0.581 per 1,000 transactions 10M-100M transactions — €0.536 per 1,000 transactions 100M+ transactions — €0.358 per 1,000 transactions
	OCR Adult Celebrity Landmark Detect, Objects Brand	0-1M transactions — €0.894 per 1,000 transactions 1M-10M transactions — €0.581 per 1,000 transactions 10M-100M transactions — €0.536 per 1,000 transactions 100M+ transactions — €0.358 per 1,000 transactions
	Describe* Read	0-1M transactions — €1.340 per 1,000 transactions 1M+ transactions — €0.536 per 1,000 transactions
Content Moderator S0 up to 10 requests per second	Moderate, Review	0-1M transactions - €0.894 per 1,000 transactions 1M-5M transactions - €0.670 per 1,000 transactions 5M-10M transactions - €0.536 per 1,000 transactions 10M+ transactions - €0.358 per 1,000 transactions
Face API Standard up to 10 requests per second	Face Detection Face Verification Face Identification Face Grouping Similar Face Search	0-1M transactions - €0.894 per 1,000 transactions 1M-5M transactions - €0.715 per 1,000 transactions 5M-100M transactions - €0.536 per 1,000 transactions 100M+ transactions - €0.358 per 1,000 transactions
	Face Storage	€0.009 per 1,000 faces per month

Figura 3.10: Captura de pantalla del apartado *Pricing* de Azure Cognitive Services.

Es posible registrar una cuenta gratuita de Azure[7] que oferta:

- Servicios populares de forma gratuita durante 12 meses.

[6] https://docs.microsoft.com/en-us/azure/cognitive-services/face/face-api-how-to-topics/howtodetectfacesinimage

[7] https://azure.microsoft.com/en-us/free/

- Más de veinticinco servicios siempre gratuitos.

- 200 dólares de crédito que se podrán usar en sus servicios durante los primeros 30 días.

Luxand Face recognition API

Luxand ofrece una amplia gama de herramientas relacionadas con la inteligencia artificial y tecnologías de identificación biométrica. Uno de ellos es Luxand Face recognition API, que cuenta con diversas funcionalidades:

- Detección de edad y género.

- Reconocimiento facial.

- Verificación facial.

- Detección de emociones.

- Detección de *landmarks*.

- Detección de celebridades.

Tiene diferentes clientes como: Universal Pictures, Samsung, P&G, Boeing, Danone y Ford entre otros.

Ofrece la posibilidad de realizar una demo de cada una de estas funcionalidades desde su página web[8] de forma sencilla, subiendo una imagen desde nuestro almacenamiento. La aplicación mostrará por pantalla una representación del análisis de la imagen sobre la misma (Figura 3.11) junto al JSON[9] que se recibiría a través de la API.

[8] https://luxand.cloud/emotion-recognition-api/

[9] JSON o JavaScript Object Notation es un formato ligero de intercambio de datos. Es de fácil lectura y escritura para los usuarios y se utiliza principalmente para transferir datos entre un servidor y un cliente.

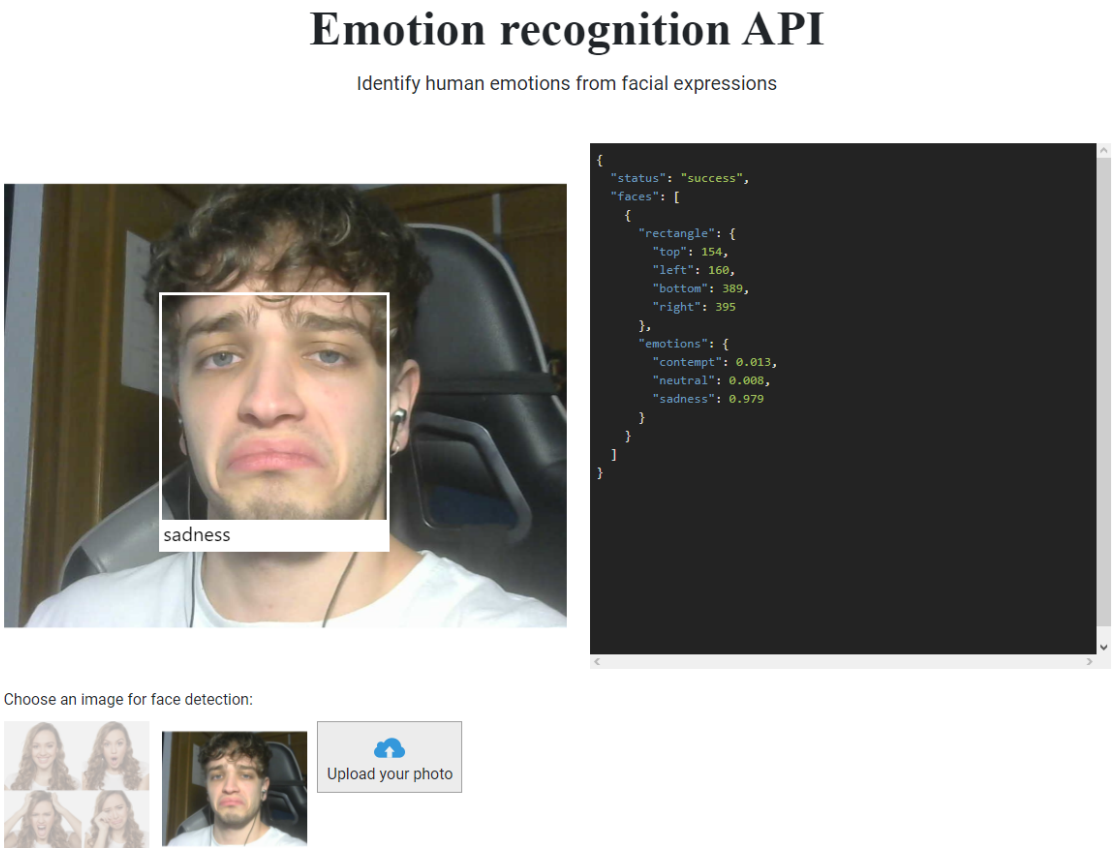

Figura 3.11: Captura de pantalla del resultado de la demo de Luxand Face recognition API.

Cuenta con una página de documentación para la API[10], sencilla y breve para todas sus funciones.

Luxand Face recognition API cuenta con 4 licencias disponibles:

- Basic.
- Pro.
- Ultimate.
- Enterprise.

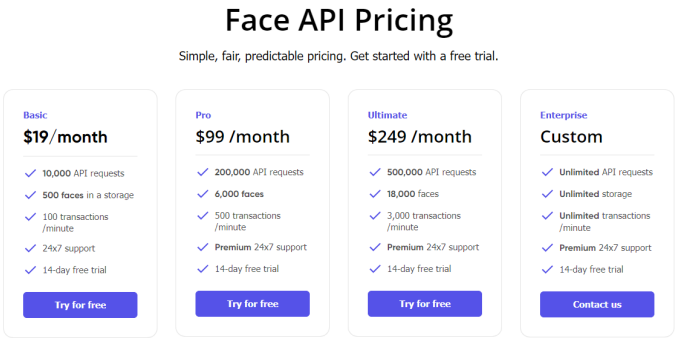

Figura 3.12: Licencias disponibles en Luxand Face recognition API.

Sightcorp FACE API

Desarrollada por Sightcorp, una empresa derivada de la inteligencia artificial de la Universidad de Ámsterdam que se especializa en software de análisis facial, permite analizar a través de una imagen distintas características como:

[10]https://docs.luxand.cloud

- Expresiones faciales.
- Estimación de edad.
- Estimación de género.
- Etnicidad.
- Estimación del estado de ánimo.
- Estimación de la postura de la cabeza.
- Estilo de ropa.

Es posible realizar una demo de forma sencilla desde su página web[11], donde podremos analizar una imagen sacada de nuestra *webcam*, subirla desde nuestro almacenamiento o introducir un enlace a la misma. La aplicación muestra por pantalla una representación del análisis de la imagen sobre la misma (Figura 3.13) junto al JSON que se recibiría a través de la API.

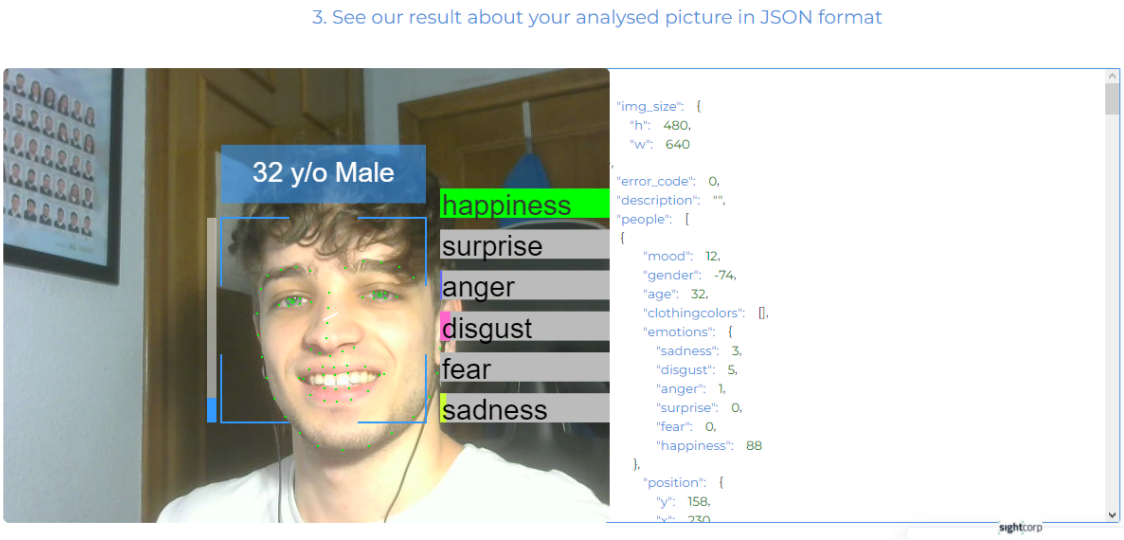

Figura 3.13: Captura de pantalla del resultado de la demo de Sightcorp FACE API.

Sightcorp cuenta con 3 licencias disponibles:

- Beginner.
- Professional.
- Enterprise.

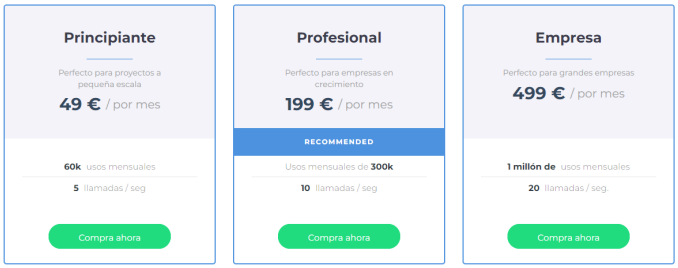

Figura 3.14: Licencias disponibles en Sightcorp FACE API.

[11] https://face-api.sightcorp.com/demo-basic/

3.4.3. Imágenes para los niveles

Para el desarrollo de esta aplicación se van a necesitar diferentes imágenes de expresiones faciales. En este momento del proyecto, se quiere añadir un número relativamente pequeño de ellas pero suficiente para que la aplicación pueda mostrar las funcionalidades especificadas.

Una de las alternativas sería hacer uso de bases de datos de imágenes libres de derechos aunque presentan un *stock* limitado ya que la mayoría son de pago.

Por otra parte, se pueden utilizar también imágenes generadas con la página web Generated photos[12] que permite obtener imágenes de caras (de personas no reales) a nuestro gusto, eligiendo parámetros como sexo, edad, humor, coloración de la piel...

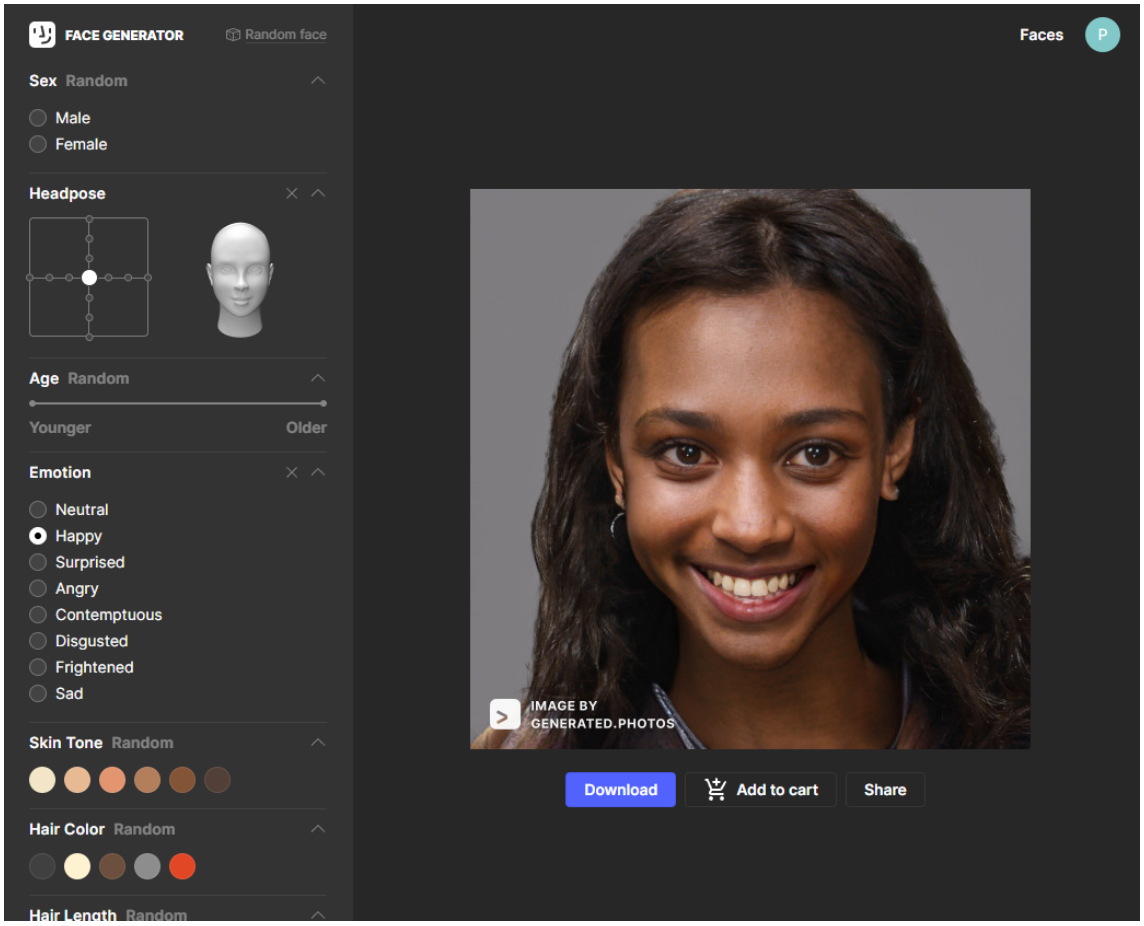

Figura 3.15: Funcionalidad Face Generator de Generated Photos.

También cuenta con una API que podría utilizarse para la generación automática de imágenes de caras.

Este programa tiene una modalidad gratuita que ofrece:

- Más de 2.500.000 de fotos de baja calidad.
- Solo para uso no comercial.

[12]https://generated.photos

- Requiere atribución.
- Sin fondo transparente.
- Con una resolución 512x512 JPG.

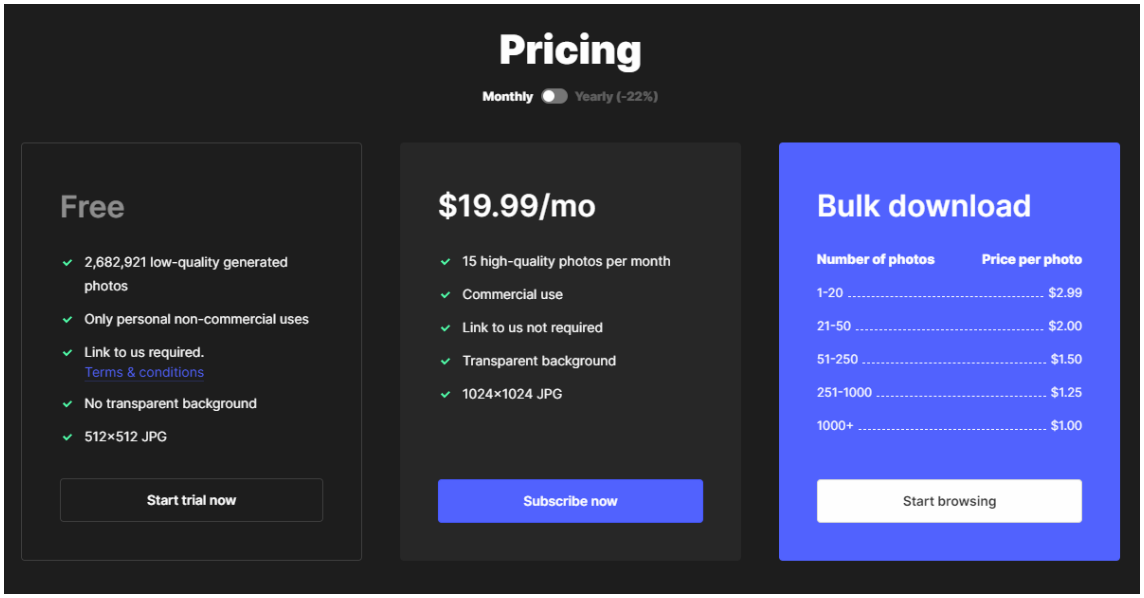

Figura 3.16: Captura de pantalla de las diferentes licencias de Generated Photos.

3.5 Solución propuesta

La solución que se ha elegido en este proyecto teniendo en cuenta las herramientas mostradas en el anterior apartado gira en torno al motor **Unity** debido a la facilidad que tiene para desarrollar juegos en 2D, su fácil exportación a WebGL (que nos permitirá un **deploy** más sencillo) y a que ya tengo nociones previas por haberlo utilizado en la asignatura **EDV** (Entornos de desarrollo de videojuegos).

El entorno de desarrollo utilizado será **Microsoft Visual Studio**, que puede seleccionarse al instalar Unity y permitirá una rápida y cómoda edición de los scripts.

La *build* del juego ya compilada se almacenará en **Github** que cuenta con **Github Pages**, que nos permite alojar proyectos web de forma gratuita y sencilla.

Github es una plataforma de desarrollo de software, con finalidad colaborativa, para alojar el código de las aplicaciones de cualquier desarrollador. El almacenamiento del código es de forma pública, aunque con una cuenta de pago, se puede hacer de forma privada.

La web utiliza el sistema de control de versiones Git que permite comparar el código de un archivo para ver las diferencias entre las versiones, restaurar versiones antiguas si se ha producido un error y fusionar los cambios de distintas versiones.

Acerca de Github Pages, se trata de un servicio estático de *hosting* que la plataforma GitHub ofrece a los desarrolladores para publicar contenido web de una

manera fácil y sencilla sin necesidad de tener conocimientos en servidores, diseñado para alojar nuestros proyectos directamente en un repositorio.

Los sitios alojados en un repositorio se asocian con el dominio .github.io y utilizan el protocolo HTTPS (Hypertext Transfer Protocol Secure).

No se implementará base de datos, se utilizará **IndexedDB** para persistir información en el navegador del usuario.

IndexedDB es una base de datos que permite almacenar información estructurada, de forma persistente, dentro del navegador de un usuario, independientemente de la disponibilidad de la red, lo que le permite funcionar tanto en línea como fuera de línea.

Está soportada por todos los navegadores modernos.

A diferencia de otras, es perfecta para almacenar grandes cantidades de datos.

También proporciona una API de búsqueda basada en índices para recuperar los datos que necesita, aunque solo se recuperan a través de valores clave. Existen librerías que pueden ser de ayuda, ya que su implementación es compleja.

Las modificaciones de las imágenes se realizarán en **GIMP**.

GIMP es una herramienta gratuita de edición y retoque de imágenes.

Este software libre ofrece una interfaz personalizable y está disponible para cualquier sistema operativo.

Posee su propio formato de almacenamiento, el xcf y soporta la mayoría de ficheros gráficos, como jpg, gif, png, tiff, etc.

El *backend* seleccionado será **Azure Face** que en cuanto a complejidad de implementación, coste y funcionalidades es lo que mejor se adecua al proyecto. También permite una mayor escalabilidad pudiendo implementar más de sus numerosos servicios.

Las imágenes utilizadas para los distintos niveles se recopilarán mediante **bases de datos de imágenes gratuitas** como:

- Freepik.
- Pixabay.
- Pexels.

y el servicio **Generated Photos**.

Como mejora futura, se podría implementar una generación automática de imágenes mediante la API de Generated Photos.[13]

3.5.1. Control de versiones

Para el manejo de control de versiones en un repositorio se ha utilizado **Github** y **GitKraken**.

La metodología de uso de Git ha sido **GitFlow**.

[13] https://generated.photos/api

GitKraken es una herramienta gráfica multiplataforma que ayuda a manejar el sistema de control de versiones, de forma sencilla y permitiendo lograr una mayor eficiencia y productividad.

La usan multitud de desarrolladores conocidos e incluso empresas como Netflix, Tesla y Apple.

GitFlow es una estrategia creada para mejorar la organización de Branchs (ramificaciones) dentro del repositorio y, de esta forma, dar más fluidez al proceso de nuevas *Features* y *Releases*[14]. Las principales ramas definidas en GitFlow son:

- Las ramas principales: rama Master y la rama Develop.
- Las ramas auxiliares: rama Feature, rama Release y rama Hotfix.

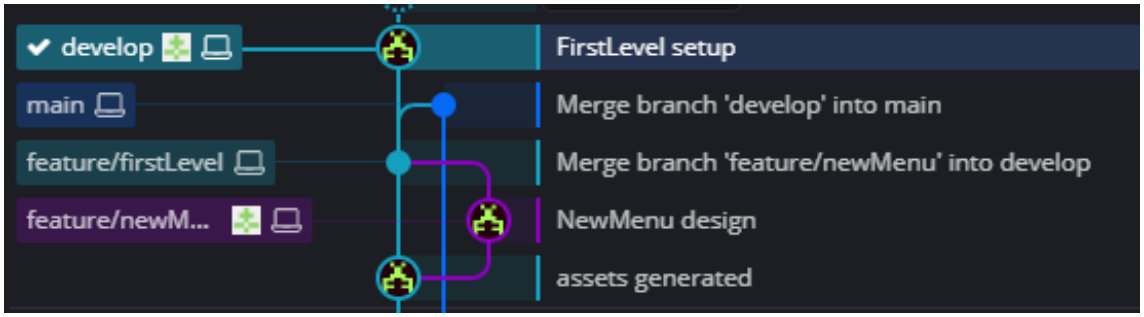

Figura 3.17: Extracto del proyecto visualizado desde GitKraken.

[14] *Releases*: es una build particular de la aplicación que se hace pública o accesible a ciertos usuarios.

CAPÍTULO 4
Diseño de la solución

4.1 Análisis de las herramientas

4.1.1. Unity

Interfaz

Unity presenta una interfaz muy intuitiva, donde la ventana del editor principal está compuesta por ventanas con pestañas que se pueden reorganizar según la preferencia personal y el tipo de trabajo a realizar. Por lo que, el aspecto del editor puede ser diferente de un proyecto a otro. Inicialmente, la disposición predeterminada de las ventanas permite el acceso a las ventanas más comunes.[9]

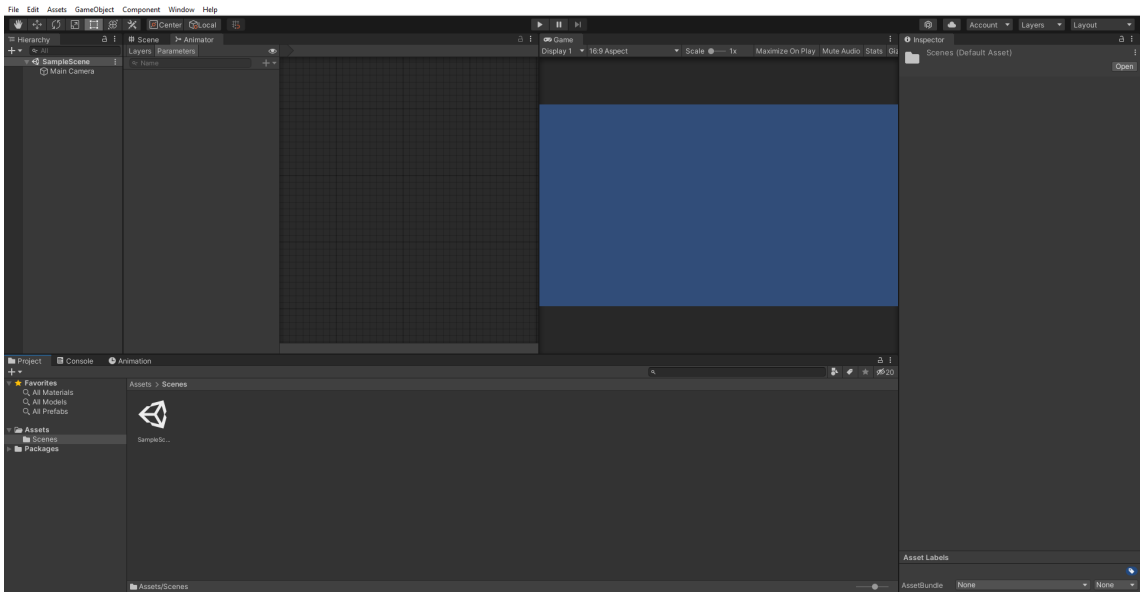

Figura 4.1: Editor de unity, ventanas principales.

La **ventana del proyecto** consta de un explorador de archivos donde se pueden encontrar todos los elementos que se pueden usar en el juego o proyecto y que se denominan *assets*.

Se muestra en el panel izquierdo del navegador la estructura de carpetas del proyecto como una lista jerárquica y en el panel de la derecha el contenido de la carpeta seleccionada y los *assets* individuales se muestran como iconos que indican su tipo.

Encima de la lista de estructura del proyecto hay una sección de favoritos donde se pueden guardar los elementos de uso frecuente para facilitar el acceso.

El directorio raíz, que se identifica como *Assets*, se sincroniza a tiempo real con la carpeta del sistema. Se pueden añadir elementos a esta carpeta para su uso en un futuro, ya que a la hora de compilar, Unity solo tendrá en cuenta los *assets* que se usen en las escenas del proyecto.

La **ventana de escena** permite una visión interactiva del mundo que está creando. Se puede utilizar para seleccionar y colocar escenarios, personajes, cámaras, luces y todos los demás tipos de *Gameobjects*.

Esta ventana contiene un conjunto de controles de navegación para ayudar a moverse de manera rápida y eficiente:

- *Scene Gizmo*, que muestra la orientación de la cámara de vista de escena y permite modificar con rapidez el ángulo de visión y el modo de proyección.
- Las herramientas *Move*, *Orbit* y *Zoom* que se pueden realizar de distintas formas:
  - Movimiento de flecha.
  - La herramienta de mano.
  - Modo *flythrough*.
  - Velocidad de la cámara.
  - Atajos de movimiento.
- La herramienta *Center* para centrar la vista de escena en un GameObject.

La **ventana de juego** es una previsualización final de lo que el jugador observará en su dispositivo, una representación del juego ya finalizado. Los cambios que se realicen durante la ejecución del juego son temporales y se restablecen al terminar la reproducción.

El **Inspector** muestra información detallada del elemento seleccionado (*Gameobjects*, *Assets*, Materiales...) y permite editar y modificar su funcionalidad en la escena.

Además de las ventanas principales, la interfaz de Unity cuenta con otras **ventanas adicionales**:

Tabla 4.1: Ventanas adicionales del editor de Unity.

Ventana	Función	Área o tema
Consola	Muestra registros de mensajes, advertencias y errores.	Scripting
Profiler	Se utiliza para investigar y encontrar los cuellos de botella en el rendimiento del juego.	Análisis
Iluminación	Se utiliza para administrar la iluminación en la escena.	*Lighting*
Occlusion Culling	Se utiliza para para mejorar el rendimiento.	Cámara

La **ventana de animación**, donde se podrán realizar clips de animación desde cero y modificar frame a frame o mediante *keypoints* las propiedades deseadas de componentes a lo largo de la línea temporal.

La **ventana de animator**, donde se puede crear máquinas de estado y asignarles animaciones a cada uno de los mismos, pudiendo transicionar de una animación a otra.

Assets

Un *asset* es cualquier elemento que se puede utilizar en el proyecto.

Se almacenan en un mismo directorio y pueden provenir de un archivo creado fuera de Unity, como un modelo 3D, un archivo de audio, una imagen, etc.

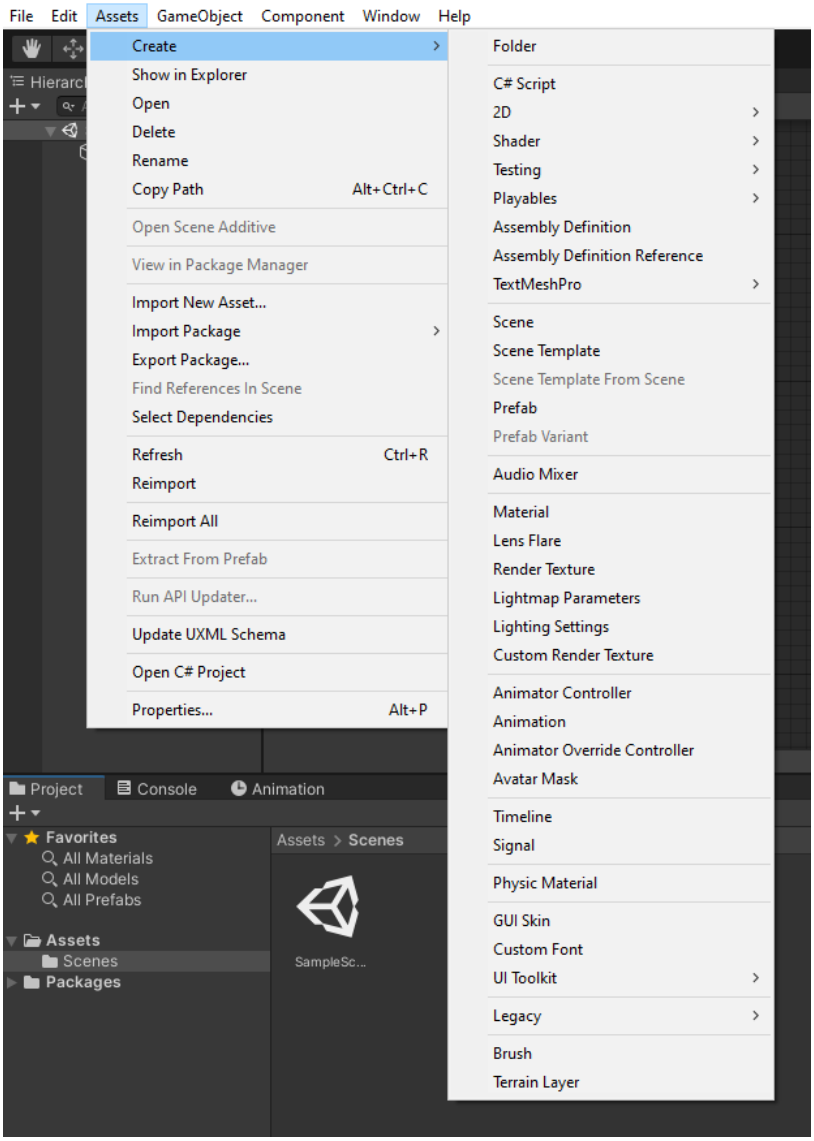

Figura 4.2: Pestaña de *Assets* desplegada.

Scenes

Las escenas son las diferentes pantallas de nuestro juego, independientes entre si, que contienen los entornos y menús del mismo.

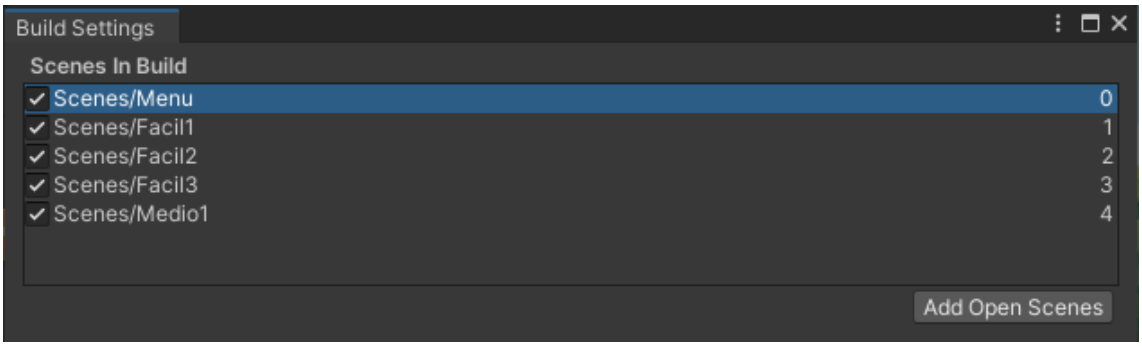

Figura 4.3: Pestaña *build*, escenas agregadas en la compilación del proyecto.

GameObjects

Es el concepto más importante en el Editor de Unity.

Corresponde a cada elemento u objeto del juego (personajes, luces, cámaras...). Para que adquiera la funcionalidad que se precisa se le deben añadir diferentes componentes.

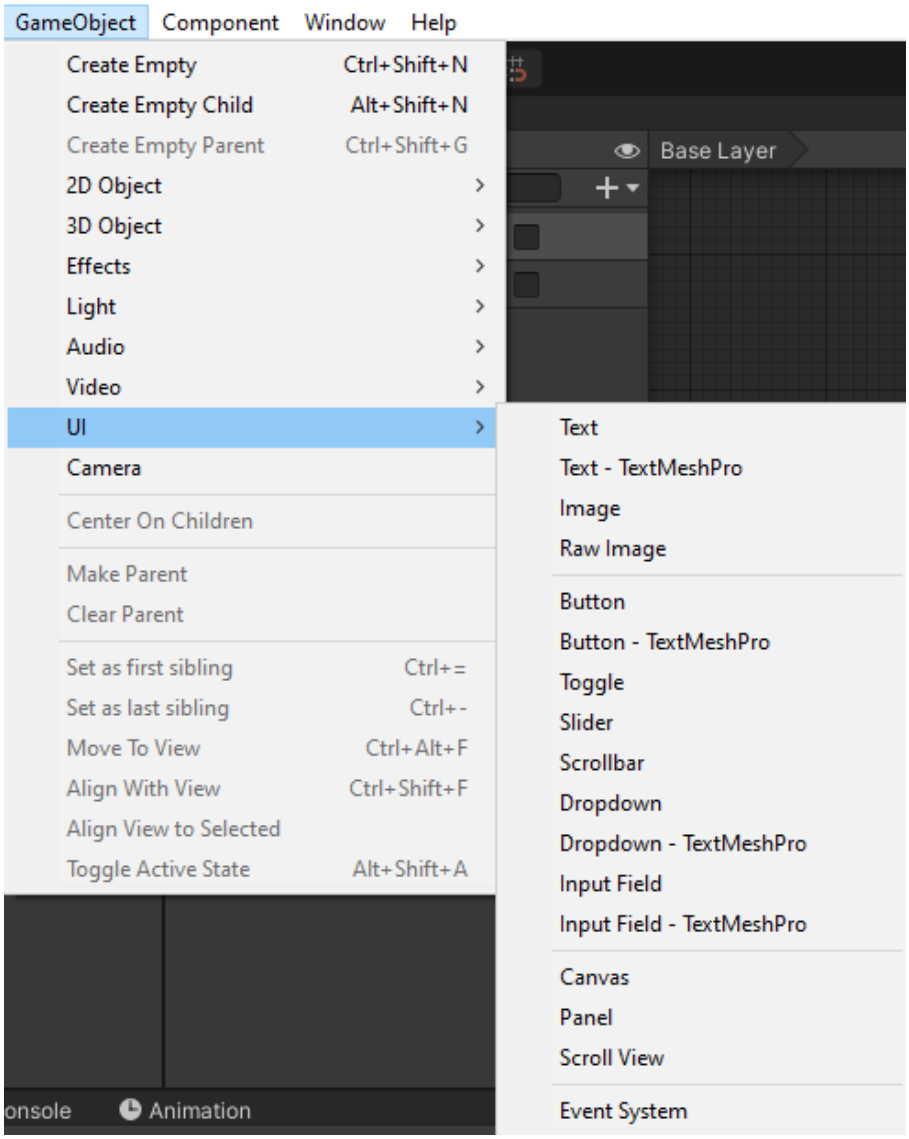

Figura 4.4: Pestaña de *GameObject* desplegada.

En nuestro juego utilizaremos sobretodo *GameObjects* de categoría *UI*.

Prefabs

Son objetos reutilizables creados desde los recursos del proyecto y que serán instanciados en el videojuego cada vez que se necesite. Es decir, permite crear 'copias' fácilmente con una serie de ventajas:

- Se puede diferenciar los objetos que están conectados a un *prefab* desde la jerarquía de objetos.
- Se puede modificar los valores de un elemento del *prefab* instanciado, sin afectar al resto de instancias implementadas, y sin romper la conexión con el *prefab* creado en los recursos del proyecto.
- Al *prefab* creado se le pueden modificar las propiedades y si se instancia en diferentes zonas de la escena, no es necesario modificar de manera individual cada instancia ya que los cambios realizados afectarán a todas las instancias implementadas.

Components

Son los que implementan la verdadera funcionalidad de los *GameObjects* y definen su comportamiento.

Algunos de los más importantes en este proyecto son:

- *Transform*: es el componente que siempre está presente en un *GameObject* y que define la posición, orientación y escala del objeto en el mundo del juego y la vista de escena. Y no puede ser eliminado.
- *Camera*: es el que permite visualizar la escena de una forma u otra, tanto desde la ventana de juego como en la propia aplicación.
- *AudioSource*: Se encarga de reproducir sonidos. A estos se les llaman *AudioClip*.
- *Animator*: Podremos asignar a los *GameObjects* los controladores de animaciones creados en la ventana del animator (explicada anteriormente).

Scripts

Los *scripts* son el componente que dota al juego de la mayoría de su funcionalidad al modificar propiedades y valores de cualquier objeto.

Están escritos en lenguaje C# y la mayor parte del tiempo haremos que estos cumplan una determinada función dentro de Unity, utilizando los objetos y componentes de la jerarquía y realizando operaciones lógicas y matemáticas necesarias.

Por defecto, este extenderá de la clase *MonoBehaviour* y traerá definidos dos métodos: *Start* y *update*.

A diferencia de la mayoría de los otros *assets*, los *scripts* se crean normalmente directamente en Unity.

4.1.2. Azure

Azure Portal es una página web de gestión de los diferentes recursos y servicios que Azure proporciona. Esta puede compilar, gestionar y supervisar todo,

desde aplicaciones web sencillas hasta complejas implementaciones en la nube. Se pueden crear paneles personalizados para tener una vista organizada de los diferentes recursos.

Está diseñado para proporcionar resistencia y disponibilidad continua. Está continuamente actualizándose y no tiene tiempos de inactividad para las actividades de mantenimiento.

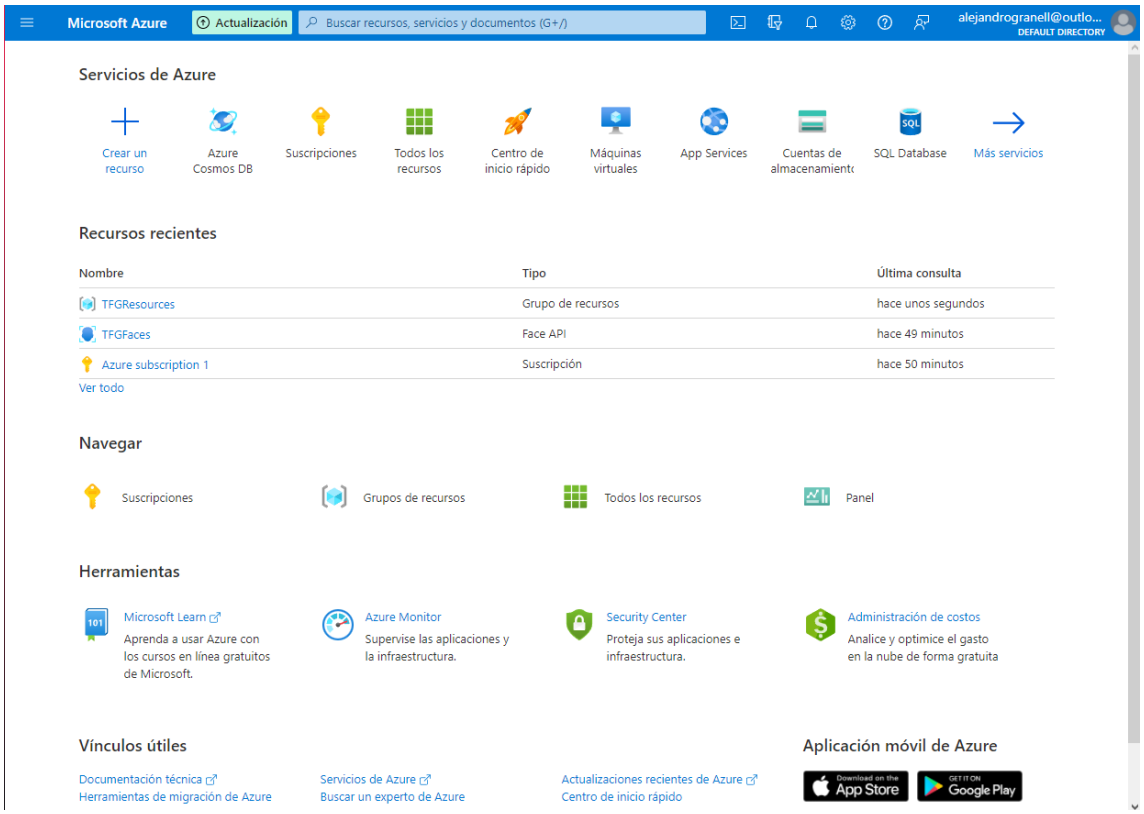

Figura 4.5: Captura de pantalla de Azure Portal.

Lo primero que haremos será crear un grupo de recursos. En este, podremos crear y vincular el resto de recursos que usaremos en nuestro proyecto. También tendremos que asociarle una subscripción.

Figura 4.6: Creación de un grupo de recursos en Azure Portal.

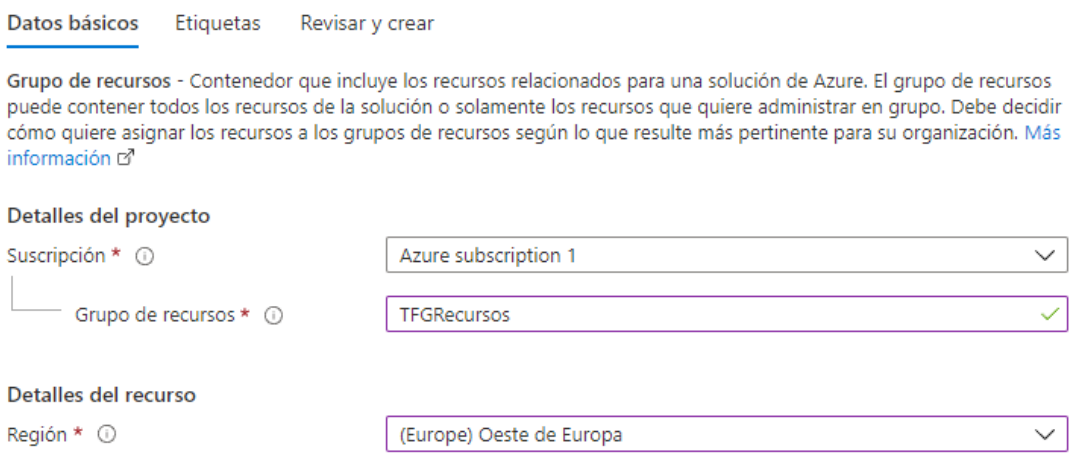

Figura 4.7: Proceso de creación de un grupo de recursos en Azure Portal.

Una vez creado, añadiremos un recurso desde la página de gestión del mismo. Para este proyecto, utilizaremos Face.

Figura 4.8: Creación del recurso Face dentro de un grupo de recursos.

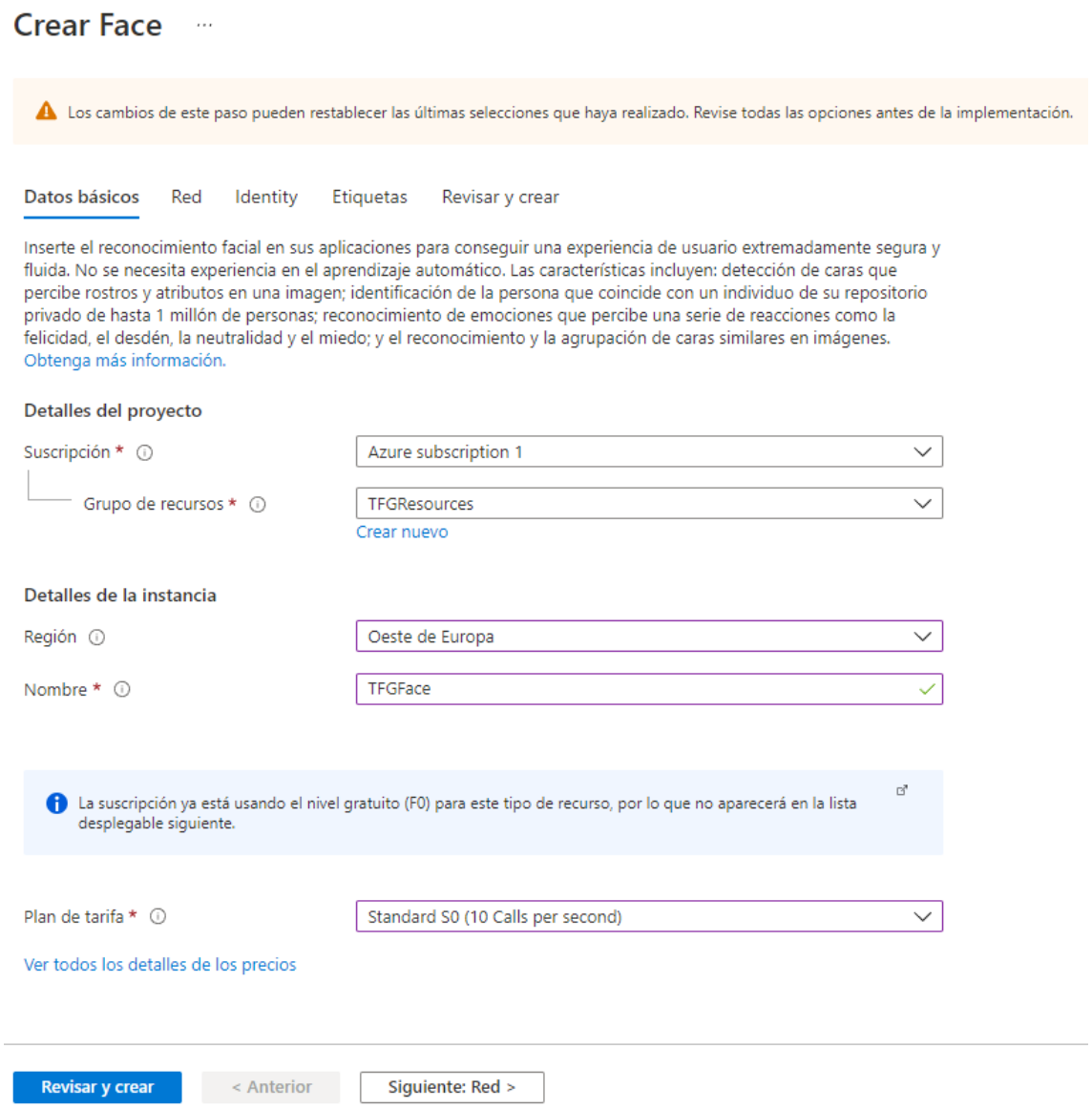

Figura 4.9: Proceso de creación del recurso Face.

Marcaremos oeste de Europa para la región y seleccionaremos el plan de tarifa gratuito.

Desde la página de gestión del recurso, se pueden encontrar numerosas funcionalidades de administración de recursos, información general, supervisión, automatización y soporte. Entre ellas se encuentran:

Inicio rápido

Ilustra una guía rápida y sencilla para la implementación del servicio en nuestras aplicaciones.

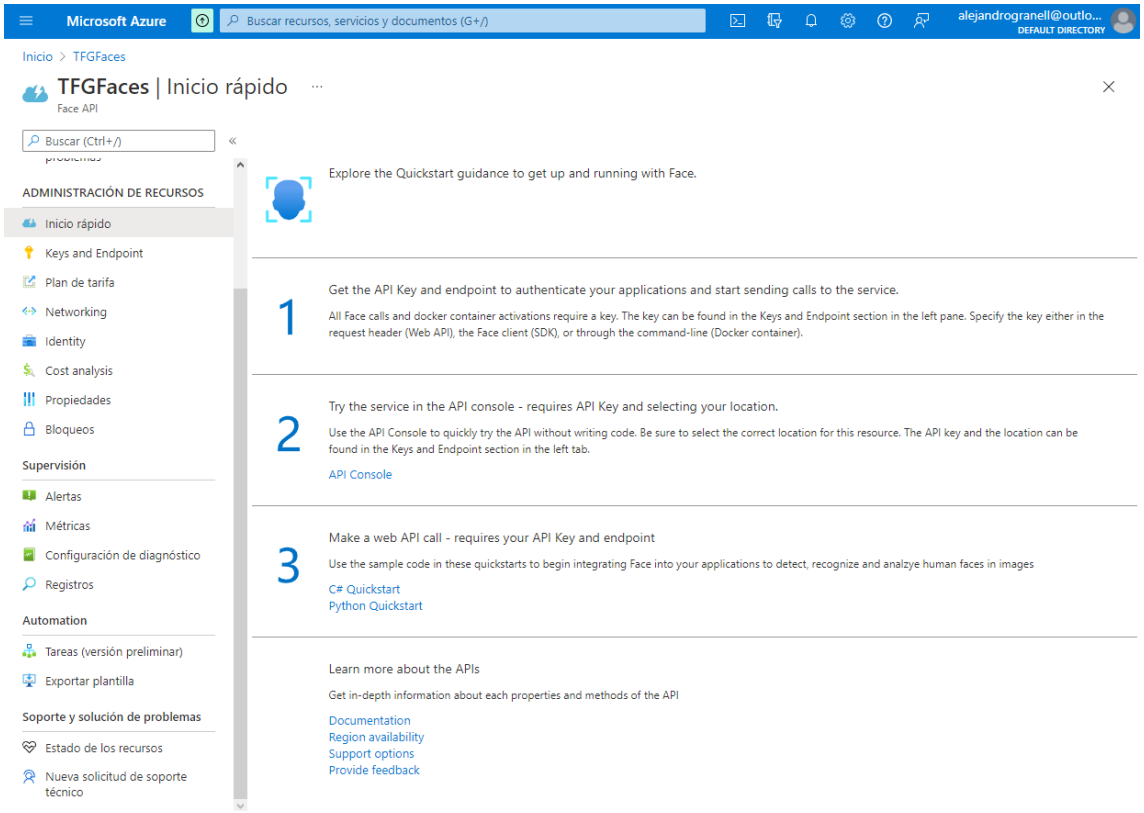

Figura 4.10: Captura de pantalla del inicio rápido de Azure Face.

Aquí podremos encontrar el acceso a diversas páginas como la documentación[1] y la referencia API[2] entre otras. Estas nos serán de gran utilidad a la hora de implementar el servicio en nuestro proyecto.

[1]https://docs.microsoft.com/es-es/azure/cognitive-services/face/overview
[2]https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/

Keys and Endpoint

Almacena y gestiona las *keys* para realizar las llamadas al servicio.

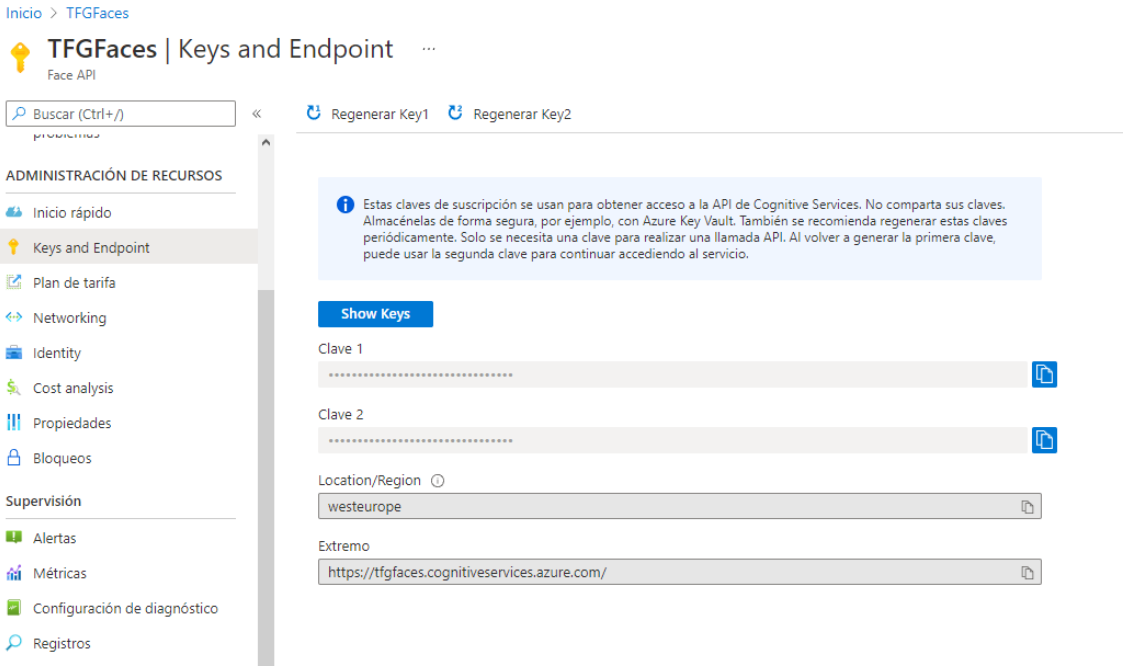

Figura 4.11: Captura de pantalla de la sección *Keys and Endpoint* de Azure Face.

4.2 Arquitectura software

A continuación se explica la arquitectura de la aplicación mediante un diagrama de arquitectura (Figura:4.12), mostrando los diferentes servicios e interacciones.

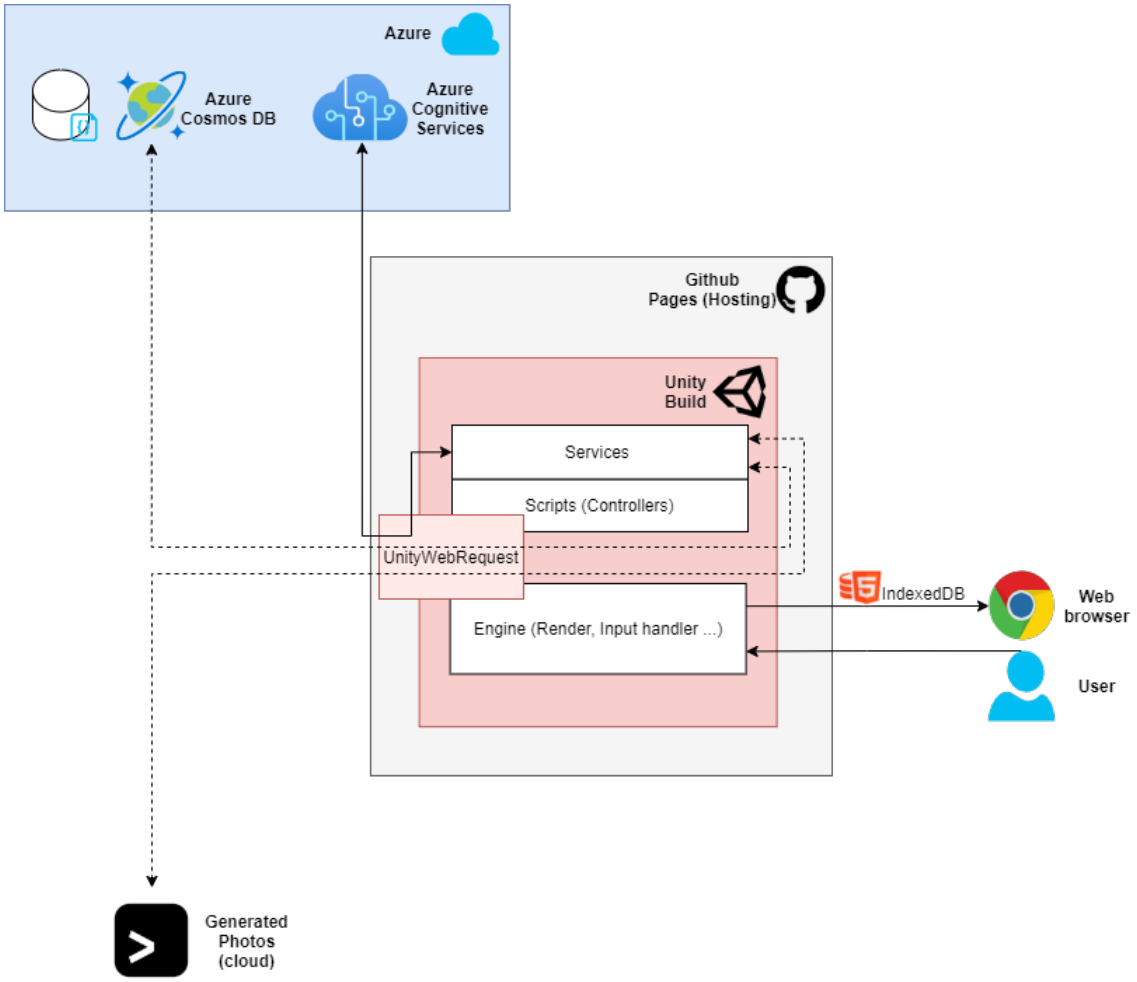

Figura 4.12: Diagrama de arquitectura del videojuego.

Se ha decidido que el servicio de Generated Photos no se utilizará dado que la generación de niveles de forma automática queda fuera del *scope* del proyecto.

Cosmos DB tampoco se va a utilizar debido a que la capa de persistencia es simple. Si se quisiera hacer un sistema de autenticación o de carga de niveles mediante la DB (base de datos) se podría implementar.

4.2.1. Lógica de la aplicación

A continuación se explica el comportamiento de la aplicación gracias al siguiente diagrama de flujo (Figura:4.13), mostrando las acciones que se pueden realizar desde las distintas pantallas y su navegación entre ellas. La leyenda de las figuras que se utilizan son las siguientes:

- Triángulo: representa el inicio o fin de la ejecución de la aplicación.
- Cuadrado: representa las diferentes vistas de la aplicación.
- Círculo: representa las acciones que se pueden realizar.
- Rombo: representa situaciones donde el sistema evalúa una condición y determina el camino a seguir.

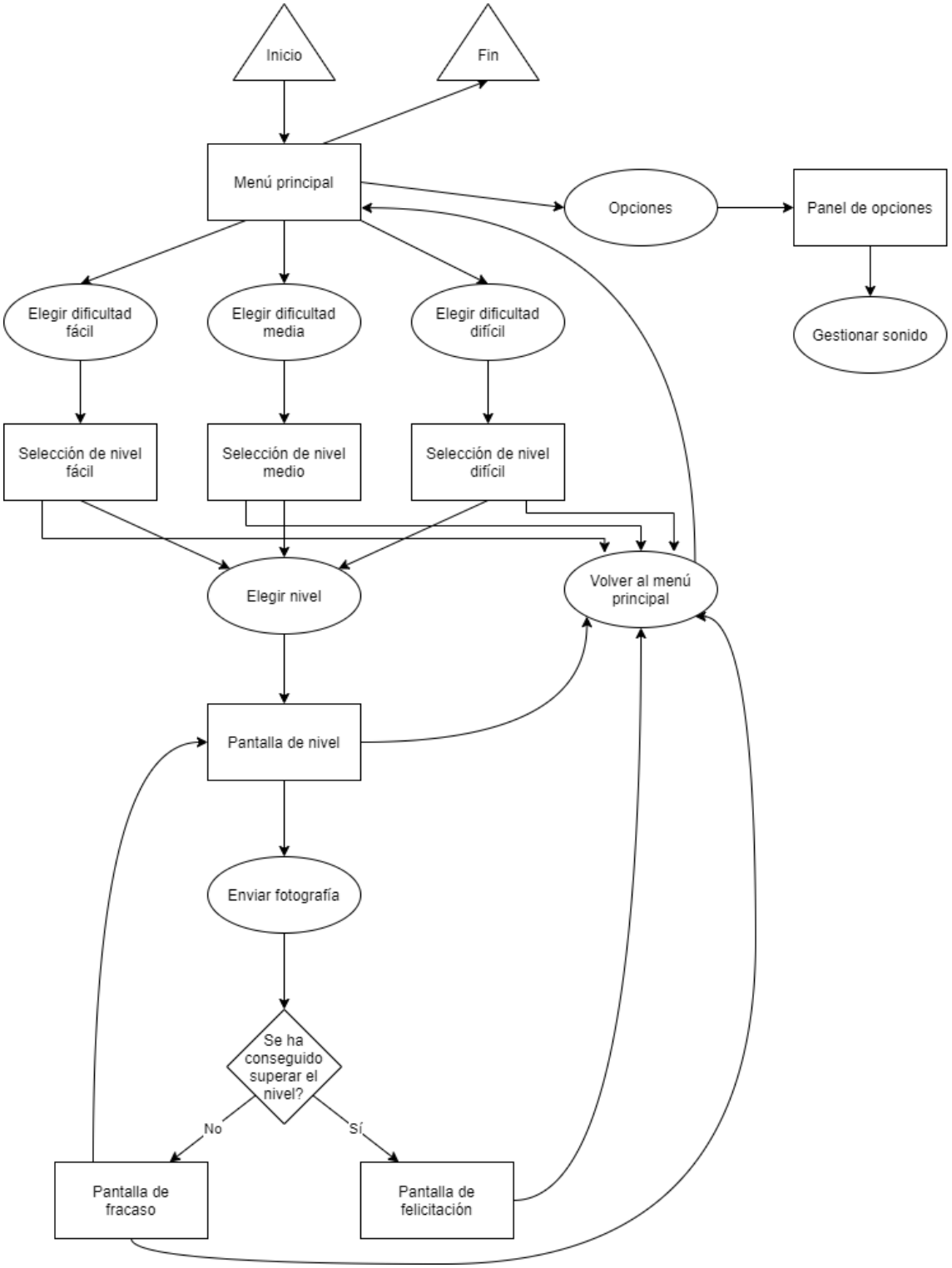

Figura 4.13: Diagrama de flujo del videojuego.

4.2.2. Capa de persistencia

No se implementará base de datos. Se utilizará IndexedDB para persistir información en el navegador del usuario. *Serializaremos* una clase específica que contendrá toda la información que queramos almacenar (El progreso de niveles). Se podrá ampliar ya que la capacidad de esta es elevada.

4.2.3. Interfaz de usuario

A medida que se fueron definiendo los requisitos de la aplicación se elaboraron los siguientes bocetos de interfaz para la misma:

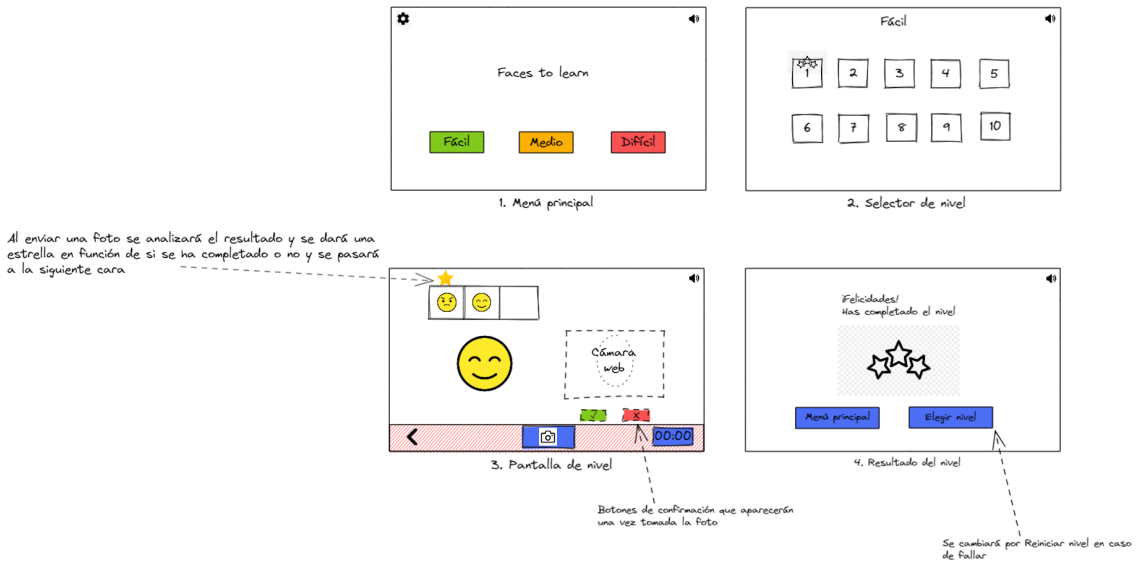

Figura 4.14: Boceto de interfaz del videojuego realizado en *Excalidraw*.

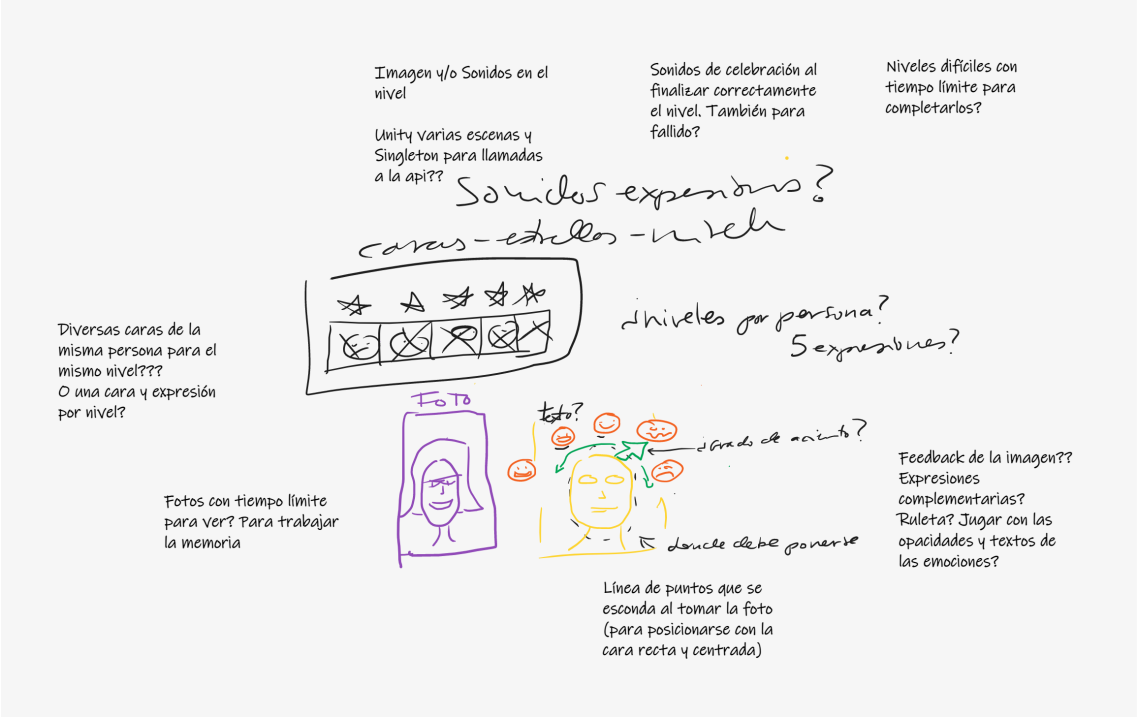

Figura 4.15: Boceto de nivel realizado en *Whiteboard* de la herramienta Teams.

4.2.4. Seguridad

No se almacenan las fotos de los usuarios ni en la aplicación ni en Azure.[3]
Todo el tráfico de datos se realiza mediante llamadas cifradas HTTPS.
La aplicación no persiste información personal ni *cookies*.

[3]https://docs.microsoft.com/es-es/azure/search/search-security-overview

CAPÍTULO 5
Implementación

A continuación, se explicará el proceso de implementación que se ha seguido para desarrollar Faces to learn.

5.1 *Webcam*

Como primera instancia, se ha desarrollado la capacidad de gestionar y visualizar las diferentes *webcam* del dispositivo y la posibilidad de realizar fotos y codificarlas a PNG (Portable Network Graphics).

Para ello, se hará uso de diversos *GameObjects* de tipo UI. Estos nos permiten crear una interfaz para nuestro juego 2D con: botones para realizar las diferentes acciones, una *RawImage* para representar la imagen de la *webcam* y de la foto tomada por pantalla y un *GameObject* vacío llamado CameraController que se encargue de realizar toda la lógica.

Este último tendrá un *script* que llamaremos CameraScript que gestionará los diferentes métodos que se ejecutarán al pulsar los botones correspondientes. Estarán asignados mediante el inspector de Unity. (En los siguientes apartados se comentará una mejor implementación de esto).

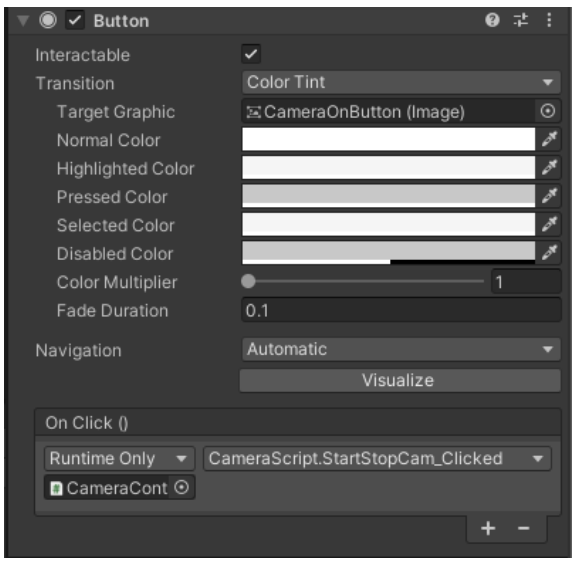

Figura 5.1: Componente Button del *GameObject* CameraOnButton.

Se utilizará una clase del motor de Unity llamada WebCamTexture que nos permitirá acceder a la lista de dispositivos disponibles y reproducir los mismos. Crearemos un objeto WebCamTexture con el parámetro del nombre del dispositivo y luego se la asignaremos al RawImage para mostrarla por pantalla.

```
//encender apagar cámara botón
1 referencia
public void StartStopCam_Clicked() {
try
    {
    if (tex != null) // Apagar cámara
    {
        StopWebcam();
    }
    else // encender cámara
    {
        WebCamDevice device = WebCamTexture.devices[currentCamIndex];
        tex = new WebCamTexture(device.name);
        display.texture = tex;
        tex.Play();
        PhotoButton.SetActive(true);
        startStopText.text = "Apagar cámara";
    }
    }
    catch(IndexOutOfRangeException) //Por si no detecta cámaras
    {
        ErrorPanel.SetActive(true);
        Text ErrorText = ErrorPanel.GetComponentsInChildren<Text>()[0];
        ErrorText.text = "Error! No se ha podido detectar la cámara";
    }
}
```

Figura 5.2: Método StartStopCam_Clicked de la clase CameraScript.

```
// cambiar cámara botón
0 referencias
public void SwapCam_Clicked() {
    ErrorPanel.SetActive(false);
    if (WebCamTexture.devices.Length > 0)
    {
        currentCamIndex += 1;
        currentCamIndex %= WebCamTexture.devices.Length;


        if (tex != null) //por si hay alguna cámara activa
        {

            StopWebcam();
            StartStopCam_Clicked();
        }
    }
}
```

Figura 5.3: Método SwapCam_Clicked de la clase CameraScript.

Para hacer la captura de la imágen, crearemos un objeto Texture2D y lo 'pintaremos' con los píxeles de la *webcam*, se codificará a PNG y posteriormente se guardará en el almacenamiento (esto se cambiará en implementaciones futuras para hacer la llamada a la API).

```
public void TakePhoto_Clicked()
{
    bytes = null;
    Texture2D texture = new Texture2D(display.texture.width, display.texture.height, TextureFormat.ARGB32, false);
    texture.SetPixels(tex.GetPixels());
    texture.Apply();
    bytes = texture.EncodeToPNG();
    File.WriteAllBytes(Application.dataPath + "/images/testimg.png", bytes);
    //congelar la imagen y botones de confirmación
    ConfirmationButtons.SetActive(true);
    display.texture = texture;
}
```

Figura 5.4: Método TakePhoto_Clicked de la clase CameraScript.

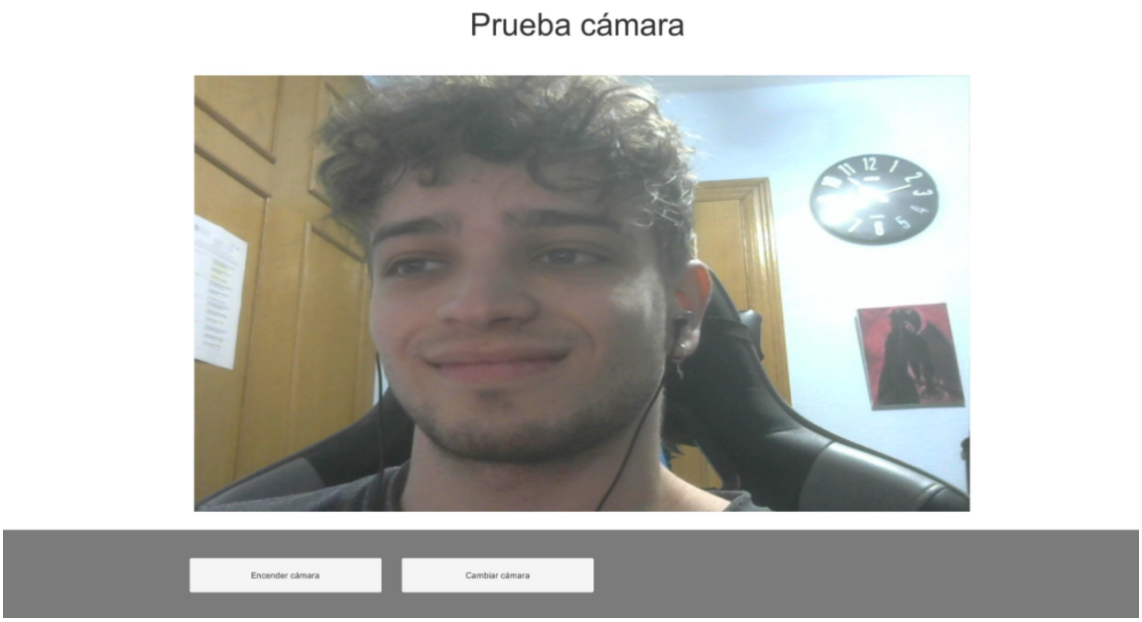

Figura 5.5: Captura de pantalla de la primera versión de la aplicación.

5.2 RESTful API

A continuación, se ha desarrollado la conexión entre la aplicación y Azure Face mediante llamadas RESTful.

REST es un estilo de arquitectura de software que se utiliza para describir cualquier interfaz entre diferentes sistemas que utilice HTTPS para comunicarse. Este término significa *REpresentational State Transfer* (transferencia de estado representacional), lo que quiere decir que entre dos llamadas cualquiera, el servicio no guarda los datos. Estas se caracterizan por:

- El cliente y el servidor están débilmente acoplados, es decir, el cliente no necesita conocer los detalles de implementación del servidor y el servidor no se preocupa de cómo utiliza el cliente los datos.
- No hay estado, es decir, cada petición que recibe el servidor es independiente.
- Se utilizan los verbos HTTP GET, POST, PUT y DELETE para el acceso, creación, actualización y borrado de recursos.
- Las llamadas son cacheables para así evitar pedir varias veces un mismo recurso.
- La interfaz es uniforme, es decir, cada recurso del servicio REST debe tener una única dirección URI.

Podemos ver en la referencia a la API de Face, que para llamar el recurso Detect (el que nosotros utilizaremos para detectar emociones) deberemos realizar una llamada POST con diferentes parámetros. Los que nosotros necesitaremos son:

- Request URL: la URL del recurso Detect https://endpoint/face/v1.0/detect donde debemos modificar el *endpoint* por donde esté alojado el recurso. En nuestro caso introduciremos el *endpoint* de oeste de Europa. También añadiremos como parámetro returnFaceAttributes=smile,emotion para que nos devuelva en la respuesta tanto el valor de sonrisa como las diferentes emociones.

- Request headers:

  - Content-Type: el tipo del body que enviaremos a la API. En nuestro caso será application/octet-stream ya que le enviaremos el stream de la imagen.

  - Ocp-Apim-Subscription-Key: el valor de la *Key* que consultamos previamente en la página *Keys and Endpoints*.

- Body: enviaremos el stream de la imagen codificada en PNG. Azure Face cuenta con algunas restricciones a la hora de subir la imagen (las podremos comprobar en la documentación o en la referencia a la API).

Para llevar a cabo estas llamadas, crearemos diversos scripts:

- Singleton.

- RestWebClient.

- APIController.

- Modelos como:

  - Response.

  - RequestHeader.

  - AzureFacesResponse.

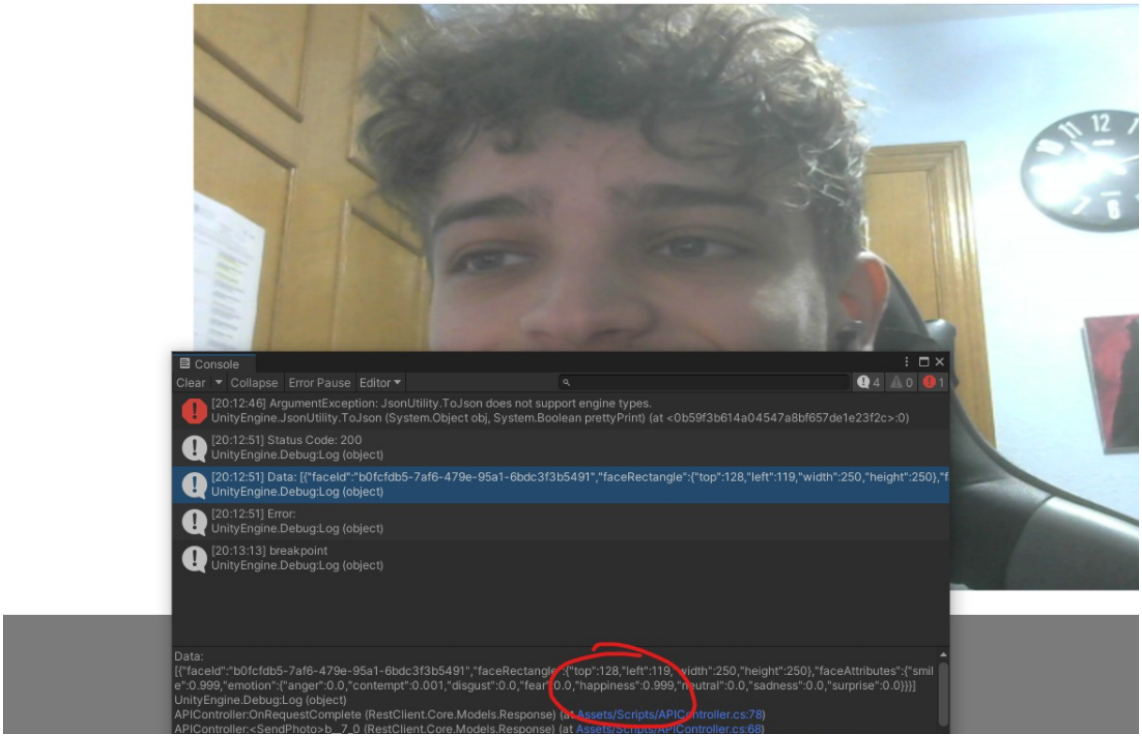

Figura 5.6: Mensaje de Debug con la respuesta de la llamada API.

5.2.1. Singleton

El patrón de diseño Singleton, o instancia única, está diseñado para restringir la creación de objetos de una clase a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

```
using UnityEngine;

namespace RestClient.Core.Singletons
{
    Script de Unity | 4 referencias
    public class Singleton<T> : MonoBehaviour
        where T : Component
    {
        private static T _instance;
        11 referencias
        public static T Instance
        {
            get
            {
                if (_instance == null)
                {
                    var objs = FindObjectsOfType(typeof(T)) as T[];
                    if (objs.Length > 0)
                        _instance = objs[0];
                    if (objs.Length > 1)
                    {
                        Debug.LogError("There is more than one " + typeof(T).Name + " in the scene.");
                    }
                    if (_instance == null)
                    {
                        GameObject obj = new GameObject();
                        obj.name = string.Format("_{0}", typeof(T).Name);
                        _instance = obj.AddComponent<T>();
                    }
                }
                return _instance;
            }
        }
```

Figura 5.7: Extracto de la implementación del patrón de diseño Singleton.

Cuando se llame a una clase que herede de ella, podremos usar obj.Instance para acceder de forma global a la única instancia de esa clase y si no existe, crearla.

5.2.2. Modelos

Se implementarán modelos para gestionar la respuesta del servicio, *deserializar* el JSON recibido y facilitar la programación de otras clases.

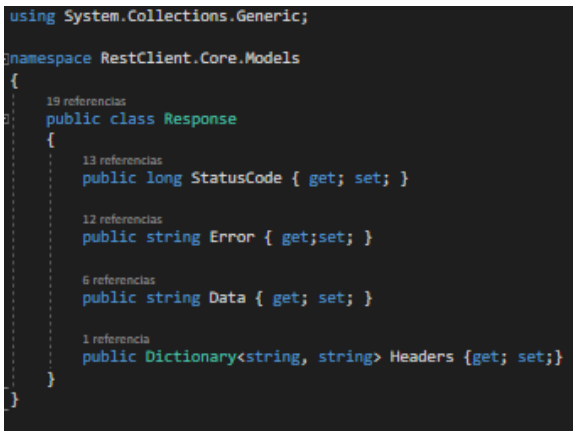

Figura 5.8: Implementación de la clase Response.

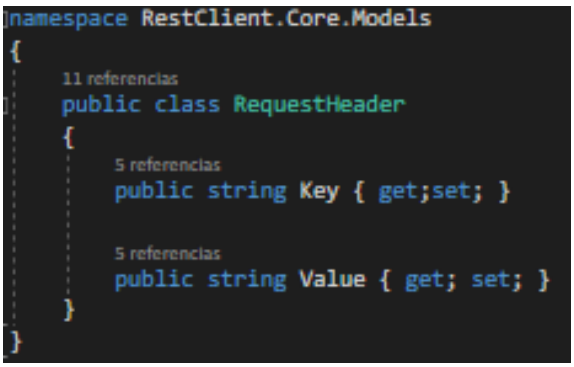

Figura 5.9: Implementación de la clase RequestHeader.

La estructura de AzureFacesResponse debe ser exacta a la estructura del JSON recibido. En apartados futuros se explicará el porqué del atributo *result*.

```
[Serializable]
3 referencias
public class AzureFacesResponse
{
    public AzureFacesResponseArray[] result;
}

[Serializable]
1 referencia
public class AzureFacesResponseArray
{
    public string faceId;

    public AzureFacesRectangle faceRectangle;

    public AzureFacesAttributes faceAttributes;

}
[Serializable]
1 referencia
public class AzureFacesRectangle
{
    public int top;
    public int left;
    public int width;
    public int height;
}

[Serializable]
1 referencia
public class AzureFacesAttributes
{
    public double smile;
    public AzureEmotion emotion;
}

[Serializable]
1 referencia
public class AzureEmotion
{
    public double anger;
    public double contempt;
    public double disgust;
    public double fear;
    public double happiness;
    public double neutral;
    public double sadness;
    public double surprise;
}
```

Figura 5.10: Implementación de la clase AzureFacesResponse.

```
/*
[
    {
        "faceId": "76de8fe0-adb5-4239-9236-03d28f389b77",
        "faceRectangle": {
            "top": 302,
            "left": 448,
            "width": 336,
            "height": 336
        },
        "faceAttributes": {
            "smile": 0,
            "emotion": {
                "anger": 0,
                "contempt": 0,
                "disgust": 0,
                "fear": 0,
                "happiness": 0,
                "neutral": 0.382,
                "sadness": 0.618,
                "surprise": 0
            }
        }
    }
]
*/
```

Figura 5.11: JSON recibido en una llamada al recurso Detect con los parámetros returnFaceAttributes=smile,emotion.

5.2.3. RestWebClient

Esta clase se encargará de montar y realizar las llamadas API haciendo uso de UnityWebRequest. Heredará de la clase Singleton para garantizar que solo existe una instancia de la misma y pueda ser accedida globalmente. UnityWebRequest hace uso de dos atributos: uploadHandler y downloadHandler que serán los que utilizaremos para realizar el intercambio de información con el servidor.

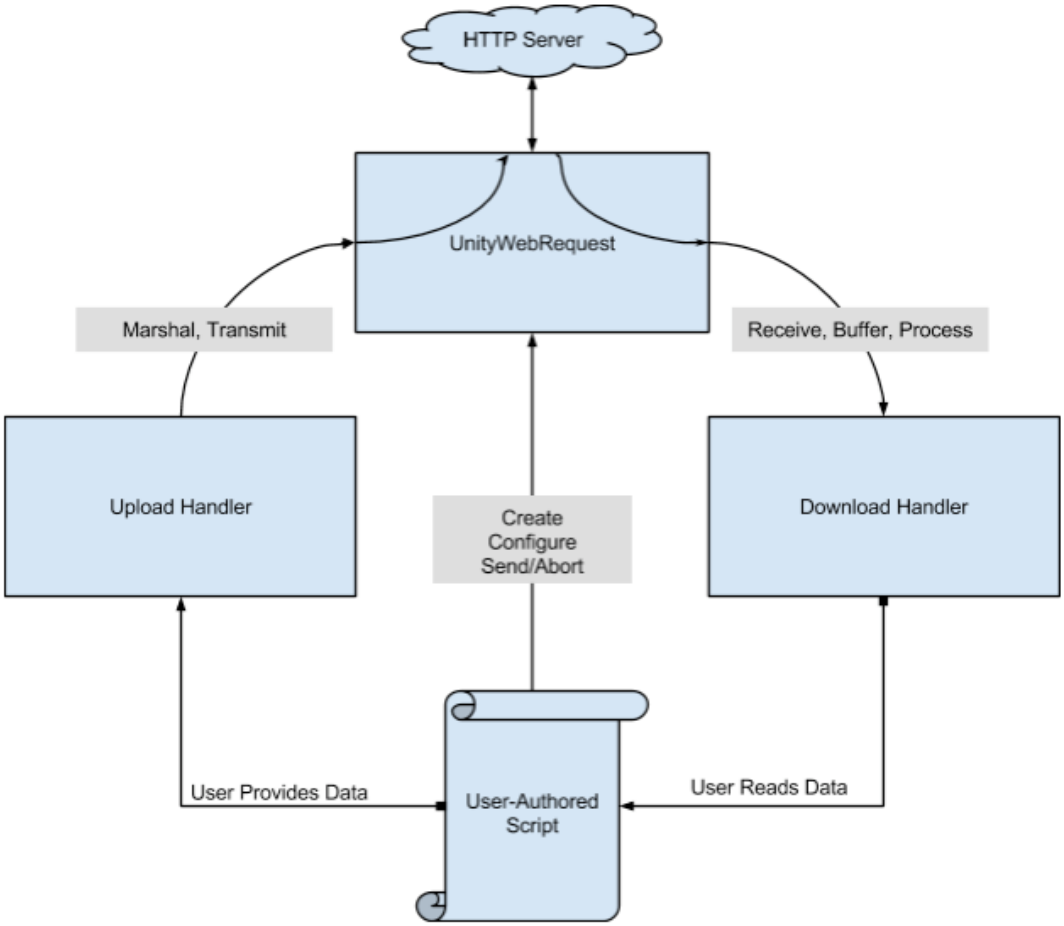

Figura 5.12: Diagrama de arquitectura de UnityWebRequest.

A continuación, se muestra la implementación del método HttpPostStream (Figura:5.13).

```
public IEnumerator HttpPostStream(string url, byte[] body, System.Action<Response> callback, IEnumerable<RequestHeader> headers = null)
{
    using (UnityWebRequest webRequest = UnityWebRequest.Post(url, ""))
    {
        if (headers != null)
        {
            foreach (RequestHeader header in headers)
            {
                webRequest.SetRequestHeader(header.Key, header.Value);
            }
        }
        webRequest.uploadHandler.contentType = "application/octet-stream";
        webRequest.uploadHandler = new UploadHandlerRaw(body);

        yield return webRequest.SendWebRequest();

        if (webRequest.isNetworkError)
        {
            callback(new Response
            {
                StatusCode = webRequest.responseCode,
                Error = webRequest.error
            });
        }

        if (webRequest.isDone)
        {
            string data = System.Text.Encoding.UTF8.GetString(webRequest.downloadHandler.data);
            callback(new Response
            {
                StatusCode = webRequest.responseCode,
                Error = webRequest.error,
                Data = data
            });
        }
    }
}
```

Figura 5.13: Implementación del método HttpPostStream de la clase RestWebClient.

Al recibir respuesta del servidor, se crea un objeto Response con la información de la misma.

5.2.4. APIController

Esta clase se encargará de tener la información de las llamadas y será accedida por CameraScript para hacer la llamada API y hará el tratamiento de la respuesta.

```
1 referencia
public void SendPhoto(byte[] photo)
{
    StartCoroutine(RestWebClient.Instance.HttpPostStream(baseUrl,photo , (r) => OnRequestComplete(r), new List<RequestHeader>
    {
        clientSecurityHeader,
        contentTypeHeader
    }));
}
```

Figura 5.14: Implementación del método SendPhoto de la clase APIController.

5.3 Persistencia

Para lograr la persistencia se ha hecho uso de IndexedDB.

Se ha creado un *script* llamado GameData que almacenará la información que se quiere persistir. En una primera instancia solo contendrá un diccionario para guardar el progreso de niveles, siendo la clave tipo string (identificador del nivel) y un valor tipo int (puntuación del mismo). Si no existe entrada para un nombre, significará que este no se ha completado.

Se usará esta misma clase si en un futuro se quieren persistir más datos.

Otro *script* llamado DataAccess será el encargado de guardar y cargar los datos.

```
public class DataAccess
{
    [DllImport("__Internal")]
    1 referencia
    private static extern void SyncFiles();

    [DllImport("__Internal")]
    1 referencia
    private static extern void WindowAlert(string message);

    1 referencia
    public static void Save(GameData gameData)
    {
        string dataPath = string.Format("{0}/GameData.dat", Application.persistentDataPath);
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        FileStream fileStream;

        try
        {
            if (File.Exists(dataPath))
            {
                File.WriteAllText(dataPath, string.Empty);
                fileStream = File.Open(dataPath, FileMode.Open);
            }
            else
            {
                fileStream = File.Create(dataPath);
            }

            binaryFormatter.Serialize(fileStream, gameData);
            fileStream.Close();

            if (Application.platform == RuntimePlatform.WebGLPlayer)
            {
                SyncFiles();
            }
        }
        catch (Exception e)
        {
            PlatformSafeMessage("Failed to Save: " + e.Message);
        }
    }
```

Figura 5.15: Extracto de la implementación de la clase DataAccess.

Para que esta pueda comunicarse con el navegador, se tendrá que añadir en la carpeta de Assets, un fichero Assets/Plugins/WebGL/HandleIO.jslib

```
var HandleIO = {
    WindowAlert : function(message)
    {
        window.alert(Pointer_stringify(message));
    },
    SyncFiles : function()
    {
        FS.syncfs(false,function (err) {
            // handle callback
        });
    }
};

mergeInto(LibraryManager.library, HandleIO);
```

Figura 5.16: Implementación del fichero HandleIO.jslib.

Se añadirá al controlador del nivel (que también heredará de Singleton) las llamadas pertinentes DataAccess cuando se quiera guardar el progreso.

5.4 Buenas prácticas

Cuando se trata de referencias, se puede definir un atributo como público y luego desde el inspector, arrastrar a él lo que se quiera referenciar. Es una forma

muy cómoda pero en proyectos grandes, si se pierden estas referencias, puede ser muy tedioso volver a crearlas.

CanvasController

Se define una enumeración llamada CanvasType y se crea una clase llamada CanvasController con solo este parámetro. Al añadirlo a un *GameObject* de tipo canvas, se podrá definir por el inspector si el canvas es el menú principal, la selección de niveles fáciles, el panel de opciones, etc.

CanvasManager

Será el encargado de buscar todos los *GameObject* que tengan un componente de tipo CanvasController y guardará en una lista las referencias a ellos. Implementará un método que active el canvas seleccionado y desactive el anterior.

```
Script de Unity (1 referencia de recurso) | 3 referencias
public class CanvasManager : Singleton<CanvasManager>
{
    List<CanvasController> canvasControllerList;
    CanvasController lastActiveCanvas;

    Mensaje de Unity | 1 referencia
    protected override void Awake()
    {
        canvasControllerList = GetComponentsInChildren<CanvasController>().ToList();
        canvasControllerList.ForEach(x => x.gameObject.SetActive(false));
        SwitchCanvas(CanvasType.MainMenu);
    }

    2 referencias
    public void SwitchCanvas(CanvasType _type)
    {
        if (lastActiveCanvas != null)
        {
            lastActiveCanvas.gameObject.SetActive(false);
        }

        CanvasController desiredCanvas = canvasControllerList.Find(x => x.canvasType == _type);
        if (desiredCanvas != null)
        {

            desiredCanvas.gameObject.SetActive(true);
            lastActiveCanvas = desiredCanvas;
        }
        else { Debug.LogWarning("El canvas seleccionado no ha podido encontrarse"); }
    }
}
```

Figura 5.17: Extracto de la implementación de CanvasManager.

ButtonController

Se asignará a un botón con la enumeración CanvasType siendo el canvas que queremos activar. Se inicializa en el Start un *listener* que, cuando se produzca el evento onClick, se llamará al método OnButtonClicked.

```
Mensaje de Unity | 0 referencias
private void Start()
{
    button = GetComponent<Button>();
    animator = GetComponent<Animator>();

    //cuando el evento se produce, se llaman a todas las funciones registradas como listeners
    button.onClick.AddListener(OnButtonClicked);
    canvasManager = CanvasManager.Instance;
}

1 referencia
void OnButtonClicked()
{

    canvasManager.SwitchCanvas(desiredCanvasType);
    animator.SetBool("pressed", false);
}
```

Figura 5.18: Métodos Start y OnButtonClicked de la clase ButtonController.

5.5 Release

Se añadirán todas las escenas en las opciones de la *build* y se cambiará a WebGL. Habrá que desactivar una opción de compresión o dará fallo a la hora de subirlo a GitHub Pages.

Figura 5.19: Captura de pantalla de Faces to Learn en GitHub Pages.

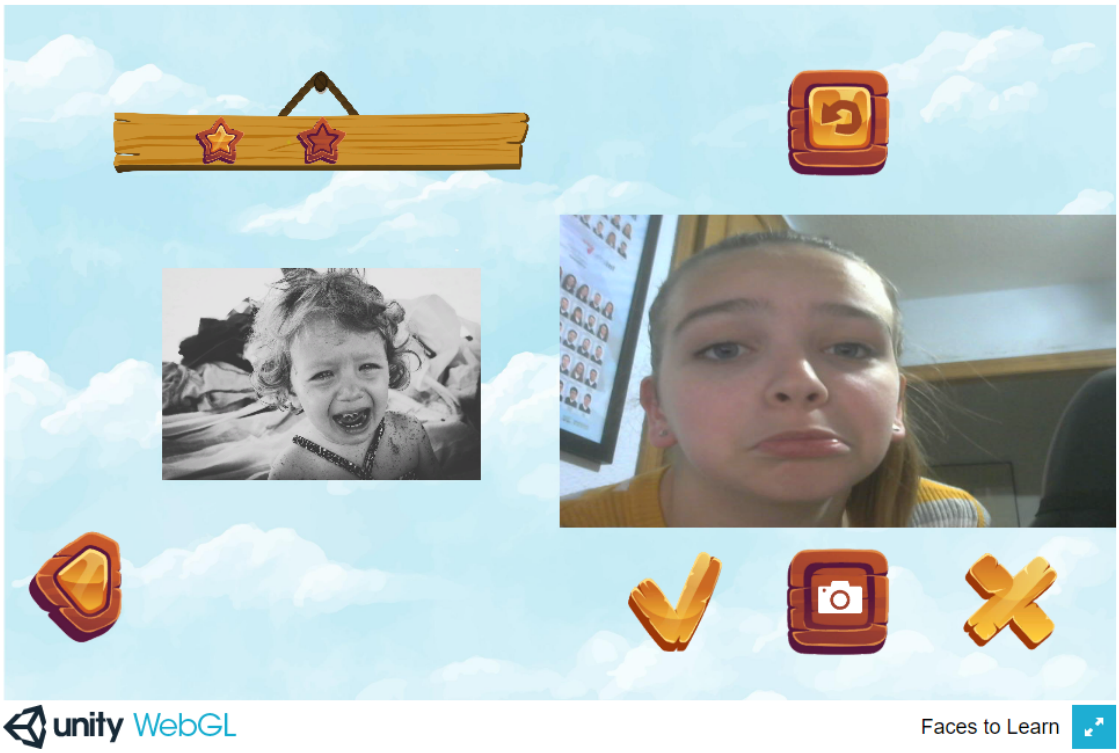

Figura 5.20: Captura de pantalla de un nivel de Faces to Learn en GitHub Pages.

CAPÍTULO 6
Conclusiones

El resultado final ha sido bastante satisfactorio. Se han completado los principales objetivos que se habían propuesto.

Aunque Unity ha simplificado el trabajo y acortado el tiempo de desarrollo del proyecto, la falta de tiempo no ha permitido implementar diferentes recursos (que se comentarán en el apartado trabajos futuros) que hubieran mejorado mucho la aplicación. A pesar de ello, he podido conectar diversos módulos y tener unos resultados muy interesantes.

Por otra parte, aunque la realización del proyecto se ha visto ayudada por los conocimientos adquiridos en diversas asignaturas de la carrera, gran parte de la formación necesaria, en este caso, se ha obtenido de manera autodidacta. Ahora bien, estos conocimientos me permitirán afrontar otros proyectos en el futuro.

CAPÍTULO 7
Trabajo futuro

Como se ha comentado anteriormente, por limitaciones de tiempo, no se han podido implementar todas las funcionalidades deseadas, quedando pendiente diferentes mejoras del proyecto para futuras versiones.

Una de las ampliaciones que se le podría hacer a la aplicación, es crear, dentro de cada nivel de dificultad, diferentes tipos de juego. Por ejemplo, mostrar una emoción mediante una imagen o sonido y que el usuario deba recomponer un rostro vacío con las piezas necesarias para igualar el sentimiento expresado en la original.

Otro de los juegos que se podrían añadir consistiría en relacionar objetos (nombres de emociones, expresiones faciales y sonidos) entre diferentes columnas.

Igualmente se podría incluir una actividad tipo '*memory* o juego de parejas', que consista en colocar una serie de fichas o cartas boca abajo, e ir destapándolas de dos en dos hasta encontrar todas las parejas de imágenes que coinciden entre sí, pero con el mínimo de movimientos posibles.

Por otra parte, se podría mejorar la aplicación aumentando el nivel de dificultad a varios niveles, siempre y cuando se mantuviera la posibilidad de seguir jugando con la dificultad menor dadas las limitaciones técnicas que pueden tener alguno de estos niños por las características de su enfermedad. A este respecto se podría:

- limitar el paso al siguiente nivel y que fuera necesario desbloquearlo tras conseguir superar los tres niveles (fácil, medio, difícil). Dado que podría darse el caso de que el usuario necesitara ayuda, se podría crear un pin parental que permitiera el desbloqueo del nivel que no consigue pasar el niño.
- Establecer limites de tiempo para conseguir el objetivo y en función del tiempo utilizado crear un sistema de recompensas que le permitieran obtener otros méritos.
- Añadir múltiples idiomas.

A niveles de cuestiones más técnicas, de arquitectura, se podría implementar el servicio de CosmosDB (pudiendo implementar un sistema de autenticación) para que el progreso del usuario sea indistinto del ordenador en el que se encuentre.

Esto permitiría implementar gran cantidad de *features* como un *ranking* con las mejores puntuaciones, una carga de los niveles directamente desde la base de datos (permitiendo la modificación de estos sin tener que volver a compilar el juego)

También se podría implementar el servicio de Generated Photos para la posibilidad de una generación automática de niveles.

CAPÍTULO 8
Atribuciones

La mayoría de los recursos utilizados son gratis y no es necesario reconocimiento.

Reconocimientos a:

- Cielo: Freepik.com". El fondo de pantalla ha sido diseñado usando imágenes de Freepik.com
- Botones de madera y oro: Freepik.com". Los botones han sido diseñados usando imágenes de Freepik.com
- Marco y tablón de madera: Freepik.com". La decoración en pantalla ha sido diseñada usando imágenes de Freepik.com
- Botón power: Freepik.com". El botón de salir ha sido diseñado usando imágenes de Freepik.com
- Paisaje naturaleza: Freepik.com". El fondo de pantalla ha sido diseñado usando imágenes de Freepik.com

Bibliografía

[1] Materiales para trabajar las Emociones en el TEA. Consultado en https://www.pictoaplicaciones.com/2019/11/15/materiales-para-trabajar-las-emociones-en-el-tea/.

[2] Teoría de las emociones de Paul Ekman. Consultado en https://www.psicologia-online.com/teoria-de-las-emociones-de-paul-ekman-5391.html.

[3] Gamificación: el aprendizaje divertido. Consultado en https://www.educativa.com/blog-articulos/gamificacion-el-aprendizaje-divertido/.

[4] Los videojuegos de interacción de cuerpo entero podrían promover las habilidades de iniciación social en niños con trastornos de espectro autista. Consultado en https://www.upf.edu/es/inicio/-/asset_publisher/1fBlrmbP2HNv/content/id/228825209/maximized#.YaEpObqCGM8.

[5] Emocionatest, una app que evalúa y potencia las emociones de niños autistas. Consultado en https://www.nobbot.com/pantallas/emocionatest-autismo/.

[6] Enlace a la aplicación Autimo - Descubra emociones. https://play.google.com/store/apps/details?id=com.auticiel.autimo&hl=es&gl=US.

[7] Enlace a la aplicación Even Better Games. http://www.czpsicologos.es/evenbettergames/.

[8] Enlace a la página web Fundación Orange. https://www.fundacionorange.es/junto-al-autismo/soluciones-tecnologicas/.

[9] Enlace a la documentación del editor Unity. https://docs.unity3d.com/Manual/UsingTheEditor.html.