UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Ciencias de la Computación

TRABAJO FIN DE GRADO

MOVI-HIIT: Sistema de representación 3D web para realizar intervenciones guiadas de actividad física en el aula

Samuel González Linde

julio, 2021

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

Ciencias de la Computación

TRABAJO FIN DE GRADO

MOVI-HIIT: Sistema de representación 3D web para realizar intervenciones guiadas de actividad física en el aula

Autor: Samuel González Linde

Tutor: Carlos González Morcillo

Co-Tutor: Cristian Gómez Portes

julio, 2021

MOVI-HIIT

Este texto ha sido preparado con la plantilla LaTeX de TFG para la UCLM publicada por Jesús Salido en GitHub[1] y Overleaf [2] como parte del curso *«LaTeX esencial para preparación de TFG, Tesis y otros documentos académicos»* impartido en la Escuela Superior de Informática de la Universidad de Castilla-La Mancha.

[1] https://github.com/JesusSalido/TFG_ESI_UCLM
[2] https://www.overleaf.com/latex/templates/plantilla-de-tfg-escuela-superior-de-informatica-uclm/phjgscmfqtsw

Tribunal:

Presidente: _____

Vocal: _____

Secretario: _____

Fecha de defensa: _____

Calificación: _____

Presidente	Vocal	Secretario
Fdo.:	Fdo.:	Fdo.:

Nunca es tarde si la dicha es buena

Resumen

La obesidad infantil y el sedentarismo se consideran como un problema de salud pública que provoca consecuencias negativas en los niños, por ejemplo, en relación con su aceptación social, pero sobre todo en su estado físico y mental. Además, este problema incluso continua en algunos casos hasta la etapa adulta. En líneas generales, las personas que padecen obesidad en su niñez triplican la probabilidad de seguir obesos en su etapa adulta. Sin embargo, hoy en día existen programas para revertir este problema. Entre ellos se encuentra MOVI, llevado a cabo por miembros del centro de estudios sociosanitarios (CESS) de la Universidad de Castilla-La Mancha, cuyo objetivo es abordar desde la raíz el problema de la obesidad infantil realizando intervenciones de actividad física integrada directamente en las aulas.

En este contexto, el presente proyecto consiste en desarrollar un software relativo al programa de prevención de obesidad y mejora de rendimiento académico en niños de educación infantil: *MOVI-HIIT*.

MOVI-HIIT es un nuevo programa de intervención realizado por el grupo investigador MOVI y proyecto de investigación llevado a cabo en la UCLM, surgido de la colaboración entre la facultad de Magisterio y la Escuela Superior de Informática de Ciudad Real.

El objetivo del proyecto es realizar un estudio sobre el impacto que supone la incorporación de actividad física sobre las capacidades cognitivas, la adiposidad y la forma física en niños de educación infantil. Para integrar esta actividad física se requiere de una plataforma web que proporcione acceso de forma rápida y sencilla a los docentes para poder realizar los llamados descansos activos en las aulas. Estos descansos activos están planificados para poder llevarse a cabo en el interior de las aulas, sin necesidad apenas de utilizar material.

La solución propuesta consiste en un entorno interactivo 3D en el que, con la ayuda de avatares virtuales, guiar la actividad física propuesta para el descanso activo. La solución además posibilita la monitorización del progreso de cada uno de los colegios pertenecientes al grupo de intervención del programa MOVI-HIIT, así como la reproducción de los descansos activos.

Abstract

Childhood obesity and sedentary lifestryle is considered a public health problem that causes negative consequences in children, for example, in relation to their social acceptance, but especially in their physical and mental state. Furthermore, this problem can even continue in their adulthood. Generally, people who are obese in childhood are three times more likely to remain obese as adults. However, there are nowedays programs in order to reverse this problem. Among them the MOVI program is highlighted, which is carried out by members of the center for socio-health studies (CESS) from the University of Castilla-La Mancha, whose objective is to tackle down the problem of childhood obesity from the root by performing integrated physical activity interventions directly in the classroom.

In this context, this project consists in the development of a software related to the obesity prevention and academic performance improvement program in pre-school children: *MOVI-HIIT*.

MOVI-HIIT is a new intervention program carried out by the MOVI research group and a research project carried out at the UCLM, arising from the colaboration between the Faculty of Education and the School of Computer Science in Ciudad Real.

The project aims to carry out a study on the impact of the incorporation of physical activity on cognitive abilities, adiposity and physical fitness in children in early childhood education. To integrate this physical activity, a web platform is required in order to provide quick and easy access to teachers to be able to take the so-called active breaks in the classrooms. These are planned to be carried out inside the classrooms, without the need of using any extra material.

The proposed solution consists of an interactive 3D environment that will guide, with the help of virtual avatars, the physical activity proposed for the active break. The solution also makes it possible to monitor the progress of each of the schools belonging to the MOVI-HIIT intervention group, as well as the reproduction of the active breaks.

Agradecimientos

Agradecer a todos aquellos que han formado parte del desarrollo del proyecto y que, sin su ayuda y colaboración, este Trabajo de Fin de Grado no habría llegado a buen puerto.

A Furious Koalas, por brindarme cobijo durante una situación complicada para todos.

A Carlos González Morcillo, por su apoyo y confianza al cargar este proyecto sobre mis espaldas.

A Cristian Gómez, por su ayuda y paciencia hasta el último momento.

A aquellos alumnos que acabaron superando al maestro y que tan buenos consejos me ofrecieron.

No podrían faltar tampoco todos aquellos compañeros de viaje que me brindó la carrera.

Por supuesto agradecer a mi familia por su confianza y ánimos durante todo este trayecto.

Y por último, a aquellas escapadas a Madrid para verte.

Samuel González Linde
Ciudad Real, 2021

Índice general

Índice de figuras

Índice de tablas

Índice de listados

Índice de algoritmos

CAPÍTULO 1

Introducción

La malnutrición según la Organización Mundial de la Salud (OMS) [10] consiste en las carencias, excesos y desequilibrios en la ingesta calórica y de nutrientes de una persona y abarca la desnutrición que incluye emaciación (peso insuficiente respecto a la talla), retraso del crecimiento (talla insuficiente para la edad) e insuficiencia ponderal (peso insuficiente para la edad); carencia o exceso de vitaminas o minerales y el sobrepeso, la obesidad y aquellas enfermedades no transmisibles relacionadas con la alimentación (como cardiopatía, diabetes, etc.).

A nivel mundial la mayoría de la población vive en países donde el sobrepeso y la obesidad están vinculados con un mayor número de muertes de personas que la insuficiencia ponderal, es decir, un mayor número de personas con obesidad que con un peso inferior al normal. La obesidad es una enfermedad crónica de etiología multifactorial y carácter epidemiológico, originada a partir de la interacción entre factores genéticos y ambientales. De acuerdo con la OMS, el sobrepeso y la obesidad se definen como una acumulación anormal o excesiva de grasa que puede ser perjudicial para la salud.

La causa de esta acumulación excesiva de grasa se debe a un desequilibrio energético entre las calorías consumidas y gastadas debido a un aumento de consumo de alimentos de alto contenido calórico ricos en grasa y un gran descenso en la actividad física debido al sedentarismo. Las consecuencias del sobrepeso y la obesidad son las enfermedades cardiovasculares, la diabetes, posibles trastornos del aparato locomotor y algunos tipos de cáncer.

Según datos de la OMS [9], la cantidad de personas que padecen obesidad se ha triplicado en todo el mundo desde 1975. Desde 2016 más de 1900 millones de adultos de 18 años o más tienen sobrepeso. Esto supone un 39 % de la población mundial de los cuales más de 650 millones de personas adultas habrían desarrollado obesidad, constituyendo esto un 13 % de la población total. En 2016 también se contaba ya con más de 340 millones de niños y adolescentes comprendidos entre las edades de 5 a 19 años con sobrepeso u obesidad. En el caso de niños menores de cinco años, la cifra alcanzaba hasta los 41 millones.

El sobrepeso y la obesidad pueden prevenirse en su mayoría optando por entornos más saludables en términos de alimentación y actividad física periódica. La OMS establece que para que una actividad física sea periódica deben realizarse un mínimo de 60 minutos diarios en personas jóvenes y 150 minutos en personas adultas. Para una alimentación saludable, la OMS establece limitar la ingesta energética procedente de grasas y azúcares, y aumentar el consumo de frutas verduras, así como legumbres, cereales integrales y frutos secos [9].

Sobre estos últimos datos se centran los esfuerzos del presente proyecto. Se pretende realizar una intervención física integrada en el aula mediante un sistema *software*. El objetivo es estudiar el impacto que supondría esta actividad física sobre la función cognitiva, la adiposidad y la forma física en niños de educación infantil.

1.1. DEFINICIÓN DEL PROBLEMA

La obesidad infantil es un problema serio que pone en riesgo la salud de niños y adolescentes. Esto está asociado con una mayor probabilidad de obesidad, muerte prematura y discapacidad en la edad adulta.

Aquellos niños que padecen a edades tempranas obesidad triplican la probabilidad de tener obesidad al ser adultos. El riesgo de obesidad en edad adulta es mayor cuanto antes se produzca el rebote adiposo. El rebote adiposo es un aumento significativo de la grasa corporal que se produce durante el primer año de vida y que posteriormente cae entre los 4 y 6 años de edad para mas tarde, durante la adolescencia, provocar un nuevo incremento. Existe evidencia de que cuanto antes se produzca este rebote adiposo, mayor es el riesgo de padecer obesidad en edades posteriores.

La prevalencia del sobrepeso y obesidad en niños y adolescentes de entre 5 a 19 años ha ido en aumento desde 1975 que rondaba un 4 % llegando hasta un 18 % en 2016. En la actualidad estos datos alcanzan alrededor de un 20 % para niños de entre 4 a 6 años y más de un 25 % en niños de 8 a 11 años.

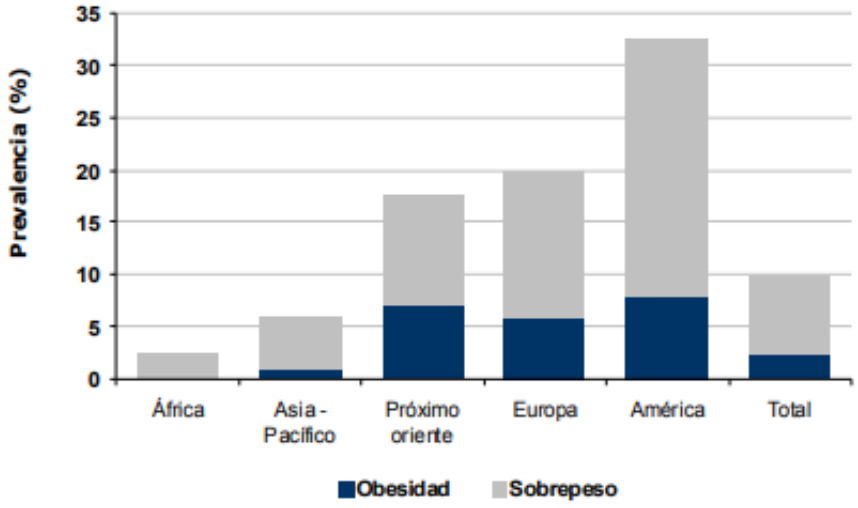

Figura 1.1: Prevalencia de obesidad y sobrepeso de niños entre 5 y 17 años en distintas zonas del mundo. Tabla extraída del Observatorio de salud de la infancia y adolescencia [12]

Como ya hemos indicado anteriormente, las principales causas de la obesidad son una ingesta elevada de grasas y el descenso de la actividad física. El tiempo total que los niños emplean en las conductas sedentarias incrementa el riesgo cardiometabólico independientemente de otros factores como la actividad física o la obesidad. Aquellos niños que realizaban interrupciones de los periodos de sedentarismo mediante 3 minutos de caminatas a intensidad moderada-alta tenían mejores valores de insulina y glucosa que los niños que no las realizaban [1].

En el aula se producen periodos de sedentarismo prolongados lo cual provoca efectos perjudiciales en la salud. Para reducir el impacto producido por estos periodos de sedentarismo, el grupo de investigación a cargo del proyecto Movi-Hiit ya ha realizado anteriormente diversas intervenciones para promover la actividad física en niños de educación infantil y educación primaria. En la siguiente sección vamos a revisar la trayectoria de los programas MOVI para ponernos en situación con el presente proyecto.

1.2. ANTECEDENTES DEL PROYECTO MOVI

El grupo investigador del proyecto MOVI [1] se define a si mismo como un grupo multidisciplinar, integrado por miembros del Centro de Estudios Sociosanitarios (CESS) de la Universidad de Castilla-La Mancha, que realizan intervenciones de actividad física en colegios de infantil y primaria. Bajo el nombre de MOVI crean programas de prevención de obesidad infantil, llegando a realizar varias intervenciones promoviendo actividad física en niños de una manera lúdica-recreativa y no competitiva, abierta y apta para todos los niños.

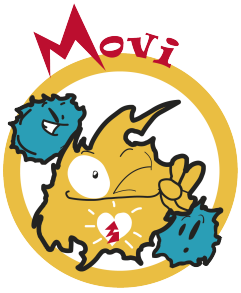

Figura 1.2: Logo del proyecto Movi

Para adentrar más al lector sobre el contexto del actual proyecto que se ha llevado a cabo en colaboración con la facultad de Magisterio y la Escuela Superior de Informática de Ciudad Real, los siguientes apartados tratan de resumir los diferentes programas que se han llevado a cabo con anterioridad por parte de este grupo investigador.

1.2.1. MOVI y MOVI-2

Primera y segunda edición de las intervenciones, realizadas en Cuenca, España. Consistieron en la evaluación de la efectividad de la intervención de actividad física en tiempo libre en escolares de 4º y 5º curso de Educación Primaria. Se buscaba un aumento del tiempo de actividad física semanal para evaluar la condición física y rendimiento académico (entre otros) a raíz de dichas intervenciones, así como el gasto energético en cada juego a partir del consumo de oxígeno utilizando medidores portátiles.

Estas dos ediciones se realizaron durante dos cursos académicos (de 2004 a 2006) y un curso académico (de 2010 a 2011), respectivamente. Ambos programas trataron con 20 colegios de Educación Primaria de diferentes localidades en la provincia de Cuenca. De estos colegios se asignaron 10 como grupo de intervención, y otros 10 colegios como grupo de control que mantuvieron la actividad física que ya realizaban hasta el momento, sin ninguna intervención.

El programa MOVI incorporaba tres sesiones semanales de actividad física de una duración de 90 minutos durante los días lectivos en horario extraescolar. En el caso de MOVI-2 se incluyeron una sesión durante el fin de semana de 150 minutos de duración y otras 2 sesiones de 90 minutos en días lectivos durante el horario extraescolar.

1.2.2. MOVI-KIDS

En esta ocasión el programa era muy similar a sus predecesores MOVI y MOVI-2, pero esta vez iba dirigido a niños de preescolar para evaluar la efectividad de una intervención de actividad física basada en juegos para la prevención de la obesidad durante el periodo del rebote adiposo, ya explicado anteriormente. Así mismo este programa tenía como objetivos principales medir la mejora del rendimiento académico, la mejora de la forma física y la disminución del grosor de la capa

íntima-media de la carótida, cuyo incremento de grosor puede suponer el riesgo de desarrollar una enfermedad cardiovascular.

Se llevó a cabo a lo largo de dos años, durante los cursos de 2013 a 2015, y en él participaron 24 colegios (22 públicos y 2 concertados) de las provincias de Cuenca y Ciudad Real. Participaron escolares de 3º de Educación Infantil y 1º de Educación Primaria pertenecientes a los 24 colegios mencionados anteriormente.

En este estudio no solo se centró en la prevención de la obesidad, si no que también se llevó a cabo un estudio sobre la efectividad de la actividad física sobre la mejoría del rendimiento académico en niños con y sin riesgo TDAH.

1.2.3. MOVI-daFit y MOVI-da10

Ambos programas tuvieron lugar desde octubre de 2017 a mayo de 2018. El primero consistía en un programa de actividad física basada en juegos para el desarrollo de las cualidades físicas básicas como la fuerza, la resistencia o la flexibilidad; habilidades motrices básicas e iniciación a deportes, todo ello adaptado para el nivel de desarrollo de escolares de 9 a 11 años. Se diseñó siguiendo recomendaciones de la OMS sobre la actividad física diaria y se realizaron 4 sesiones semanales de 60 minutos cada una durante el horario extraescolar.

El segundo programa consistió en realizar 2 descansos activos al día de 10 minutos de actividad física durante los 5 días de la semana dentro del aula de 3er curso de Educación Infantil. Estos descansos activos se centraban en desarrollar contenidos curriculares dirigidos a escolares de 5 a 6 años, como por ejemplo, matemáticas o letras, haciendo uso de la actividad física. Estos descansos activos se estructuraron de manera que tuvieran 2 minutos de calentamiento, 6 minutos de actividad física de intensidad moderada-vigorosa y 2 minutos de vuelva a la calma (ejercicios de respiración-relajación). Los docentes podían elegir qué descansos activos deseaban realizar y en qué momento deseaban realizarlo.

1.3. PROYECTO MOVI-HIIT

Como ya hemos visto las intervenciones de actividad física son una oportunidad para mejorar la salud y favorecer el aprendizaje desde una edad muy temprana. La solución que plantea el presente Trabajo de Final de Grado (TFG) surge a raíz de un nuevo programa por parte del proyecto MOVI bajo el nombre de MOVI-HIIT y que pretende esta vez integrar la actividad física en el aula haciendo uso de las nuevas tecnologías. Para ello, el proyecto cuenta con la colaboración de la ESI de Ciudad Real y la empresa **Furious Koalas S.L**, la cual es una spin-off de la propia UCLM dedicada al desarrollo de soluciones interactivas aplicadas a la gamificación, los juegos serios, simulaciones y visión por computador.

Se pretende seguir una estrategia muy similar a la de MOVI-da10 (Sección 1.2.3), por la cual los niños en el aula realizarán unos periodos cortos de ejercicio físico, en torno a los 5 minutos, para dar así al cerebro un descanso de la actividad académica y romper con el sedentarismo. Estos **Descansos activos** o **Active breaks** van a estar basados en la metodología HIIT, por sus siglas en inglés High-Intensity Interval Training, que consiste en realizar ejercicios de alta intensidad intercalados con periodos de intensidad moderada. Para estos HIITs se va a utilizar el llamado método **Tabata**, que consiste en un entrenamiento que combina intervalos de alta intensidad de una mayor duración junto con intervalos de menor intensidad de una duración más reducida. Las duraciones para las fases de alta intensidad del ejercicio consistirán en 20 segundos, mientras que las fases de intensidad moderada o fases de descanso tendrán una duración de 10 segundos.

Por lo tanto, cada Active Break consistirá en un HIIT a realizar compuesto a su vez por 4 ejercicios, cuya aceptación e integración ya han sido probadas previamente en niños de educación infantil, siguiendo el método anteriormente descrito. Una vez realizados los 4 ejercicios esto se repetirán una segunda vez y entonces dará por concluido el Descanso activo. El programa plantea un máximo de 2 Descansos activos diarios a realizar.

Toda actividad física será guiada mediante dos avatares que se encargarán de explicar los ejercicios durante una fase inicial de preparación. Posteriormente se guiará el ritmo de cada uno de los ejercicios y, finalmente, se realizará una fase de vuelta a la calma donde los niños tendrán que imitar unos ejercicios de respiración para dar por finalizado el descanso activo y regresar de nuevo a la actividad académica.

Figura 1.3: Logo propuesto por Furious Koalas S.L. para el proyecto Movi-Hiit

A diferencia del caso del programa anterior, donde se les proporcionaba a los docentes una pequeña formación sobre los ejercicios y documentación de estos para que pudieran elegir cual hacer en cada caso, con este programa se pretende liberar de esa carga por completo a los docentes y centrar sus preocupaciones exclusivamente en acceder a una plataforma y que esta se encargue de indicarles que deben hacer en todo momento. Por lo tanto no requieren de conocimiento sobre los ejercicios ni de formación previa sobre los HIITs que se van a realizar, ya que serán los propios investigadores del proyecto MOVI los que se encargarán de decidir que ejercicios van a ser los que compongan cada uno de estos HIITs e incorporarlos a la plataforma y a la propia planificación del programa.

1.4. ESTRUCTURA DEL DOCUMENTO

El presente documento se ha estructurado siguiendo las indicaciones de la normativa de Trabajos de Fin de Grado de la Escuela Superior de Informática de la Universidad de Castilla-La Mancha:

- **Capítulo 2: Objetivos**
  En este capítulo se describen los objetivos generales y específico establecido para realizar el desarrollo del proyecto, definiendo así en profundidad cada una de las características del sistema que se quieren implementar.

- **Capítulo 3: Estado del Arte**
  Este capítulo realiza una revisión en torno a los temas que tratan en el trabajo, además de incluir un análisis sobre algunas alternativas ya existentes o plataformas similares a la propuesta del presente proyecto.

- **Capítulo 4: Metodología**
  En este capítulo se detalla la metodología de trabajo que se ha seguido para afrontar el desarrollo del proyecto, así como los diferentes medios y herramientas que se han empleado.

- **Capítulo 5: Arquitectura del sistema**
  En este capitulo entramos en detalle sobre el diseño e implementación de la solución propuesta para el proyecto.

- **Capítulo 6: Resultados**
  Este capítulo recopila los resultados que se han obtenido para lograr cada uno de los objetivos propuestos para el proyecto. Se describen también los distintos cambios en la arquitectura e implementación que ha sufrido el software a lo largo del proceso de desarrollo.

- **Capítulo 7: Conclusiones**
  En este último capítulo se exponen las conclusiones que han sido sacadas a partir de los resultados obtenidos a lo largo del desarrollo del proyecto.

CAPÍTULO 2

Objetivos

Dado el enfoque del problema y la solución planteada en el capítulo de introducción, este capítulo se centra en detallar cuál es el camino a seguir para el desarrollo de este trabajo desde su etapa inicial hasta su resultado final, definiendo tanto el objetivo principal como los objetivos específicos derivados de este.

2.1. OBJETIVO GENERAL

El objetivo principal de este TFG consiste en desarrollar un sistema software cuya finalidad sea representar actividad física de manera amena mediante un enfoque interactivo en 3D con avatares virtuales.

Esto se logrará haciendo uso de una plataforma web que permitirá la reproducción en el interior de las aulas de Educación Infantil de los colegios de la provincia de Ciudad Real que formen parte del grupo de intervención del proyecto MOVI-HIIT.

2.2. OBJETIVOS ESPECÍFICOS

A continuación se presentan los diversos objetivos específicos que se han planteado a partir del objetivo general del proyecto.

- *Integración de avatares virtuales para incentivar la actividad física.* Se pretende que estos modelos virtuales entren dentro de un perfil amigable y acogedor para atraer la atención de niños de Educación Infantil y que se sientan identificados.

- *Desarrollo de una solución multiplataforma web.* El desarrollo del sistema debe ser una plataforma basada en estándares web que permita su uso desde diferentes plataformas.

- *Facilidad de uso de la plataforma.* La plataforma debe estar orientada al profesorado que participa en el programa de prevención tenga o no conocimientos técnicos en informática. Por lo tanto, el sistema debe ser fácil de usar e intuitivo.

- *Representación 3D con capacidad de integración con tecnologías web.* El sistema debe poder integrarse en la plataforma web y representar tanto a los avatares 3D como las animaciones de los ejercicios de una forma clara y concisa siguiendo las pautas de los descansos activos establecidos para el programa MOVI-HIIT (Preparación, Ejecución de los ejercicios y Descanso).

- *Representación de los ejercicios por medio de animaciones.* El programa debe precisar que los ejercicios propuestos para el mismo sean generados con forma de animación para su posterior integración y adaptación con los avatares virtuales.

- *Desarrollo de un módulo de gestión de la plataforma y control de usuarios.* La plataforma debe de permitir el seguimiento de los usuarios pertenecientes al programa y la gestión de los descansos activos. En otras palabras, la posibilidad de ver cuántos descansos activos se han realizado hasta la fecha y poder modificar ejercicios existentes o añadir nuevos descansos activos.

- *Integración de técnicas de gamificación y juegos serios.* El sistema debe incentivar y animar a los usuarios a utilizar la plataforma haciendo uso de recompensas, puntuaciones y un sistema de rankings.

- *Reproducción de sonidos para delimitar las fases del descanso activo.* El sistema debe reproducir musica, sonidos y voces para motivar a los usuarios durante la ejecución de los ejercicios, diferenciar claramente las partes del descanso activo y dar las explicaciones necesarias sobre los ejercicios.

CAPÍTULO 3

Estado del Arte

En este capitulo se recogen los principales elementos sobre los que está asentado el trabajo y que constituyen su base tecnológica y de conocimiento. Para cada una de las tecnologías utilizadas, se hace una recopilación del estado actual y sus antecedentes. En primera instancia, se hace un repaso de la tecnología web y las fases por las que ha pasado a lo largo de su historia, así como una breve descripción de los lenguajes estandarizados para el desarrollo web. Para continuar se repasa el concepto de motor de videojuegos y se analiza su funcionamiento, para posteriormente realizar una comparación de diversos motores y sus características. Se realiza también un análisis de los principales software de gráficos 3D en la industria cinematográfica y de videojuegos.

Finalmente, se da una descripción algunos ejemplos de otros sistemas que hacen uso de una representación tridimensional y técnicas de gamificación para incentivar la actividad física en niños.

3.1. TECNOLOGÍAS WEB

Comúnmente se tiende a confundir los términos **Web** e **Internet** a pesar de que existe una clara diferenciación entre ambas. El término *Internet* se refiere al conjunto mundial de redes descentralizadas que están interconectadas entre si mediante el uso de protocolos de Internet (TCP/IP) para garantizar que las redes físicas heterogéneas que la componen constituyen una red lógica única de alcance mundial [16].

Por otro lado la *Web* o *World Wide Web* (WWW), es un conjunto de protocolos que permiten consultar archivos de hipertexto de forma remota, disponibles en Internet a través de la tecnología digital. Entendemos el hipertexto como una mezcla en un mismo documento de datos de texto, gráficos, sonido o vídeo y referencias cruzadas o hipervínculos a otros documentos.

Según la recopilación de información sobre Internet realizada por Marino Latorre [7] es posible clasificar la evolución de la web en cuatro fases diferentes:

- ***Web 1.0***: la primera que apareció (1990) y la más primitiva, los usuarios solo podían consumir o publicar información pero de forma unidireccional, sin posibilidad de que se genere ninguna interacción con el contenido de la página. Servía para utilizar el correo electrónico, los navegadores y motores de búsqueda entre otros.
- ***Web 2.0***: acuñado por O'Reilly en 2004, posibilita la conexión entre personas de una manera dinámica e interactiva. Al contrario que con la Web 1.0, la Web 2.0 es bidireccional y proporciona contenido a través de webs interactivas y visuales. En la actualidad es la que utilizamos la mayor parte de los consumidores y supuso el inicio de los foros, blogs y más tarde las redes sociales.
- ***Web 3.0***: operativa desde el 2010, la *Web 3.0* son aplicaciones web conectadas a aplicaciones web. También conocida como la «web semántica», está gestionada en la nube y constituye un

nuevo tipo de web que añade contenido semántico a los documentos que la forman, permitiendo así ser rastreados por los sistemas de procesamiento y descubrir información relevante según los diferentes perfiles de red.

- ***Web 4.0***: comenzó en 2016 y está centrada en ofrecer comportamientos más inteligentes haciendo uso de sistemas desarrollados con técnicas de Deep Learning y Machine Learning. Este tipo de web incluye el uso del Procesamiento de Lenguaje Natural (NLP), la información del contexto (como posición GPS o datos registrados por wereables) y la comunicación máquina a máquina. Cualquier dispositivo con conexión a internet sirve como suministrador de datos a ordenadores almacenados en la nube y permite el aprovechamiento de estos para realizar peticiones y procesamiento de datos.

Sistemas como Siri, Google Now o las recomendaciones de anuncios que realizan los smartphones (pertenecientes a la *Web 4.0*), están a la orden del día en nuestro uso cotidiano. A pesar de esto, las redes sociales y webs interactivas (*Web 2.0*) siguen siendo el principal foco de atención para los usuarios comunes.

La creación de webs interactivas es una práctica cada vez más extendida. En sus inicios su desarrollo consistía en la mezcla de lenguajes de etiquetas y lenguajes de programación. Actualmente existen numerosas facilidades que otros desarrolladores ponen en juego. Desde el uso de blogs como Wordpress, que facilitan la creación de páginas web sin uso de código, hasta el uso de frameworks que proporcionan herramientas para solucionar la mayoría de problemas comunes en el desarrollo de aplicaciones web. Hablamos de problemas como autenticación de usuarios, manejo de peticiones y respuestas, formularios, seguridad y protección, etc.

A continuación se repasan los lenguajes básicos para el desarrollo de páginas web, así como algunos de los frameworks más destacados en la industria.

3.1.1. Lenguajes para el desarrollo web

En el mundo del desarrollo web, *HTML (HyperText Markup Language)* es el componente más básico de una página y generalmente es utilizado junto con otras tecnologías para **describir la apariencia de una página Web**, como *CSS* (Cascading StyleSheets), o **definir la funcionalidad de la misma** (*JavaScript*). También existen otras tecnologías que sirven para programar scripts del lado del servidor y que pueden comunicarse con el resto de tecnologías, permitiendo así la **creación de webs más dinámicas**. Un ejemplo de ello sería el popular lenguaje de programación *PHP* (Hypertext Preprocessor). En las siguientes secciones trataremos de forma resumida sobre el estado actual de cada uno de estos lenguajes.

Figura 3.1: Logos de los principales lenguajes utilizados en el desarrollo web

HTML

Es un lenguaje de etiquetado estándar compuesto por elementos que definen el significado y la estructura del contenido web. Estos elementos o etiquetas le indican al navegador cómo tiene que ser

mostrado el contenido de una página web. Estas marcas se utilizan para etiquetar texto, imágenes y otros contenidos.

Este lenguaje está definido por el Consorcio Internacional de la WWW (W3C)[1], encargado de generar recomendaciones y estándares para asegurar el crecimiento de la Web. Ha especificado diversas versiones de forma oficial para llevar a cabo un uso estandarizado, guiando así su evolución con el paso de las décadas:

- ***HTML***: es un lenguaje de marcas basado en el estándar SGML (Standard Generalized Markup Language), definido en 1990 bajo el proyecto creado por Tim Berners-Lee, *World Wide Web*.
- ***HTML 2.0***: a finales de 1993 se comenzaron a diseñar sucesores para HTML. Esto dio lugar a HTML+ (finalmente nombrado HTML 2.0) que fue la primera versión oficial de HTML y fue publicada en 1995.
- ***HTML 3.2***: en 1995 se propuso el borrador para el estándar HTML 3.0 e introdujo nuevas capacidades como facilidades para la creación de tablas o mostrar elementos matemáticos complejos. Finalmente HTML 3.0 quedó atrás para dar paso a HTML 3.2 que abandonaba la mayoría de las nuevas características que introducía su antecesor, publicándose en 1997.
- ***HTML 4.0***: especificación publicada a finales de 1999. Después de esta publicación el W3C se centró en desarrollar el estándar XHTML que, finalmente debido a la presión de las empresas que formaban la asociación WHATWG, acabó siendo integrada en la versión HTML 5.0.
- ***HTML 5***: es la quinta revisión del lenguaje HTML. Liberado como estándar oficial en 2014, introduce nuevas etiquetas y la posibilidad de incluir audio y video de forma directa en una página web sin hacer uso de plugins ni complementos en los navegadores.

Actualmente la última versión oficial publicada es de finales de 2017 bajo el nombre de HTML 5.2, en la que han introducido nuevas características para facilitar el desarrollo de páginas web. Es posible consultar la documentación en la página web oficial de la World Wide Web Consortium [15].

CSS

Es un lenguaje que describe el estilo de los elementos dentro de un documento HTML. Fue propuesto por primera vez a finales de 1994 por Håkon Wium Lie y no fue hasta finales de 1996 que se lanzó por primera vez. El concepto es bastante simple, pero permite ahorrar tiempo y trabajo, ya que permite realizar un estilo en un archivo y ser aplicado sobre uno o varios documentos HTML. Se denomina Hojas de estilo en cascada porque se aplican de arriba a abajo.

CSS está diseñado principalmente para separar el apartado de presentación de la información contenida en los documentos HTML. Esta separación supone la principal ventaja de evitar modificaciones en los archivos HTML y tener todas las características del apartado de presentación en un único lugar permitiendo así una mayor flexibilidad y control. También nos permite evitar la duplicación de estilos y el versionado de estos, ya que pueden definirse de forma modular y aplicarse solo los que sean necesarios.

Este lenguaje tiene una sintaxis sencilla que consiste en una serie de reglas o conjunto de reglas definidas por uno o varios selectores. Un selector es un identificador que declara a qué elementos o etiquetas dentro del Document Object Model (DOM) del documento HTML se le aplicarán los estilos definidos dentro de un bloque de declaración. Pueden ser aplicados a todos los elementos de un tipo o a elementos seguidos de atributos particulares (como sería un identificador o la clase de una etiqueta).

La especificación CSS es mantenida también por el W3C y a su vez proporciona una herramienta de validación de CSS gratuita para este tipo de documentos.

[1] https://www.w3c.es/estandares/htmlcss

JavaScript

JavaScript es un lenguaje de programación orientado a objetos. También es conocido como el lenguaje de scripting para páginas web y permite la implementación de funciones complejas en páginas web para darles dinamismo. Introduce características tales como la actualización dinámica de contenido, mapas interactivos, gráficos 3D y 2D o control de multimedia.

Fue desarrollado por Brendan Eich pasando por los nombres de *Mocha* y *LiveScript* antes de adoptar el nombre de JavaScript en 1995. Finalmente a mediados de 1997 fue adoptado como un estándar ECMA (ECMAScript) para poco tiempo despues adoptarse como un estándar ISO (International Organization for Standardization).

Este lenguaje es utilizado principalmente del lado del cliente y emplea una sintaxis similar a C, pero adopta nombres y convenciones del lenguaje de programación Java. Comenzó siendo uno de los lenguajes de programación más populares en internet y, con la incorporación de AJAX (Asynchronous JavaScript and XML) que permite la creación de aplicaciones web asíncronas, su uso y popularidad se ha catapultado aún más. A raíz de ello atrajo la atención de muchos programadores y dio lugar a una gran cantidad de frameworks y bibliotecas.

3.1.2. Frameworks para el desarrollo web

El término framework o entorno de trabajo se refiere a una aplicación software que incorpora un conjunto de módulos que permiten agilizar el proceso de desarrollo, promoviendo la reutilización de código y las buenas prácticas como el uso de patrones. Permiten también ser configurados añadiendo diferentes módulos para adaptarse a las necesidades del desarrollo de una aplicación en concreto.

En regla general, trabajan utilizando el patrón **Modelo-Vista-Controlador**, que permite la separación de las aplicaciones en tres capas: la capa de *Modelo*, que representa los datos de la aplicación, la capa de *Vista*, que representa la entrada y salida de información de forma visual, y la capa de *Controlador*, que procesa las peticiones de los usuarios y el control del flujo de ejecución del sistema [4].

A continuación se analiza brevemente algunos de los frameworks Web más populares según una encuesta realizada por la web StackOverflow [11] en Febrero de 2020 en la que participaron alrededor de 65 mil desarrolladores:

- ***jQuery***: encabezando la lista se encuentra jQuery, una biblioteca multiplataforma de JavaScript que simplifica la forma de interactuar que éste lenguaje tiene con los elementos del árbol DOM de documentos HTML. Incorpora las técnicas AJAX, manejo de eventos y animaciones.
- ***React***: en segundo puesto encontramos a React, una biblioteca de JavaScript de codigo abierto que facilita la creación de interfaces de usuario interactivas encargándose de renderizar de forma eficiente y correcta los componentes. Es mantenido por Facebook y la comunidad de software libre.
- ***Angular***: seguimos con Angular, framework de frontend de código abierto desarrollado por Google. Se utiliza para la creación de aplicaciones web de escritorio y móvil de una sola página.
- ***ASP.NET***: entorno desarrollado y comercializado por Microsoft que hace uso del lenguaje de programación C# para la creación de webs interactivas.

Como se puede ver, la gran mayoría de estos frameworks son utilizados en JavaScript. En esta encuesta también destacan otros frameworks escritos en lenguajes diferentes como son Flask, Spring o Django, que están escritos en Python o Ruby on Rails, cuyo framwork es de codigo abierto escrito en el lenguaje de programación Ruby.

3.2. MOTORES DE VIDEOJUEGOS

El término «motor de videojuego» fue acuñado a mediados de los años 90 haciendo referencia al popular videojuego de tiros en primera persona (FPS) *Doom* desarrollado por la compañía ID Software en 1993. Su arquitectura definía muy bien una separación entre sus principales componentes software y los assets artísticos, los niveles del mundo y las reglas que comprenden la experiencia de juego del jugador. Esto provocó que los desarrolladores comenzaran a retocar los apartados artísticos y técnicos para crear nuevos productos, surgiendo así la «comunidad mod»[2].

A raíz del surgimiento de estas prácticas, los desarrolladores de videojuegos diseñaban sus productos teniendo en cuenta su reutilización y las modificaciones de la comunidad. Empezaron a surgir motores que podían ser customizados haciendo uso de lenguajes de scripting y se comenzó a licenciar algunos de esos motores para que otros desarrolladores pudieran utilizar partes de sus componentes software para crear videojuegos.

Actualmente, un motor de juego es un Entorno de Desarrollo Integrado (IDE) principalmente utilizado para la creación de diferentes tipos de videojuegos utilizando una gran variedad de lenguajes de programación que proveen al programador de características y configuraciones que hacen más sencillo el desarrollo de estos. Entre las características principales que incluye un motor de videojuegos podemos encontrar: un motor para la renderización de gráficos en 2D o 3D, un motor de físicas, que simula las leyes de la física o permite detectar colisiones entre los diferentes elementos de la escena, animación, inteligencia artificial, sonido y scripting, entre otros.

Figura 3.2: Logos de diferentes motores de videojuegos

Existen una gran cantidad de motores de videojuegos y cada uno proporciona herramientas diferentes. Según las necesidades y objetivos del programador existen infinidad de motores, por ejemplo, con un enfoque mayor en el desarrollo de juegos en 2D, como podrían ser GameMaker Studio[3], Construct[4] o Godot[5]; otros están enfocados mayormente en desarrollo de gráficos 3D como podrían ser Frostbite[6] o Cryengine[7], y existen otros cuyo uso está extendido tanto para desarrollo en 2D como en 3D. Ejemplos de ello son Unity[8] o Unreal Engine[9].

[2] La «comunidad mod» o *modders*, es una comunidad de personas que se dedican a realizan modificaciones en productos o servicios. En el ámbito de los videojuegos suelen ser jugadores que realizan cambios en los distintos apartados de un juego (por ejemplo, apartado artístico o reglas del juego).

[3] https://www.yoyogames.com/es/gamemaker

[4] https://www.construct.net/en

[5] https://godotengine.org/

[6] https://www.ea.com/frostbite

[7] https://www.cryengine.com/

[8] https://unity.com/es

[9] https://www.unrealengine.com/en-US/

En la Tabla 4.1, cuya información ha sido extraída y actualizada de una publicación de la Universidad de Ontario [2], pueden encontrarse algunos de los motores con las principales herramientas y características a las que dan soporte para su uso por parte de los desarrolladores.

Tabla 3.1: Algunos motores de juegos y sus características

	Unity	Unreal	HTML5	Game Maker	Godot	Source	Construct
Editor visual	●	●		●	●	●	●
Scripting	●	●	●	●	●		●
Networking	●	●	●	●	●	●	●
Graficos 3D	●	●			●	●	●
Shaders	●	●		●	●	●	●
Sombras dinámicas	●	●			●	●	●
Físicas	●	●		●	●	●	●
Inteligencia artificial	●	●		●	●	●	
Uso no comercial gratuito	●	●	●		●	●	●
Uso comercial gratuito			●		●		
Dispositivos móviles	●	●	●	●	●		●
Reproducción Web	●	●	●	●	●		●

Los motores de videojuegos Unity y Unreal Engine han sido los que mayor popularidad han alcanzado durante la última década. Esto se puede ver reflejado en las Tablas 3.2 y 3.3, análisis realizado por Marcus Toftedahl y Henrik Engström [14], con información extraída de dos de las mayores plataformas de venta de videojuegos: itch.io[10] y Steam[11]. Ambos motores de videojuegos aparecen en el top 10 de motores más utilizados para el desarrollo de videojuegos en las dos plataformas, posicionándose Unity en primer lugar en la primera plataforma con casi un 50 % de uso.

Tabla 3.2: Motores de juegos usados en juegos publicados en Itch.io (datos de 2018). Tabla extraída de la Universidad de Skövde [14]

Game Engine	**Number of projects**	**% of total games**
Unity	24200	47.3 %
Construct	6275	12.3 %
GameMaker	5643	11.0 %
Twine	3184	6.2 %
RPG Maker	1982	3.9 %
Bitsy	1683	3.3 %
PICO-8	1479	2.9 %
Unreal	1458	2.8 %
Godot	1274	2.5 %
Ren'Py	1008	2.0 %
Other engines	2993	5.9 %
Total games	**51179**	

Conociendo estos datos, las siguientes secciones se centran en los dos motores más conocidos, Unity y Unreal Engine, los cuales son los más utilizados en el mundo de la representación 3D, tanto para videojuegos como para proyectos 3D de otra índole. No es de extrañar que al contar con tanta

[10] https://itch.io/
[11] https://store.steampowered.com

Tabla 3.3: Motores de juegos usados en juegos publicados en Steam (datos de 2018). Tabla extraída de la Universidad de Skövde [14]

Game Engine	Number of projects	% of total games
Unreal	1726	25.6 %
Unity	889	13.2 %
Source	270	4.0 %
Cryengine	238	3.5 %
Gamebryo	215	3.2 %
IW	192	2.9 %
Anvil	166	2.5 %
id Tech	113	1.7 %
Essence	73	1.1 %
Clausewitz	68	1.0 %
Other engines	3266	48.4 %
Total games	**6743**	
Unknown games	**42538**	
Total games in Steam database	**49281**	

popularidad ambos motores disponen de las más extensas documentaciones y comunidades dentro del desarrollo de videojuegos. Además, las siguientes secciones proporcionan información sobre las características de los motores.

3.2.1. Unreal Engine

Unreal Engine es un motor de juego creado por la compañía de Epic Games presentado en 1998 con el lanzamiento del juego Unreal. En 2006 se lanzó Unreal Engine 3 y junto con ella surgieron sus primer usos para dos títulos cuyo desarrollo suponía una gran cantidad de presupuesto (llamados titulos AAA o titulos Triple A): Gears of War y Mass Effect. Su desarrolló inicial se basó principalmente para la creación de juegos de PC del estilo shooter en primera persona. Sin embargo, esto se extendido al resto de géneros de videojuegos y ha sido también adoptado por muchas industrias para desarrollar soluciones en los ámbitos de la arquitectura, cinematografía y entretenimiento, eventos en vivo, simulaciones y automovilismo.

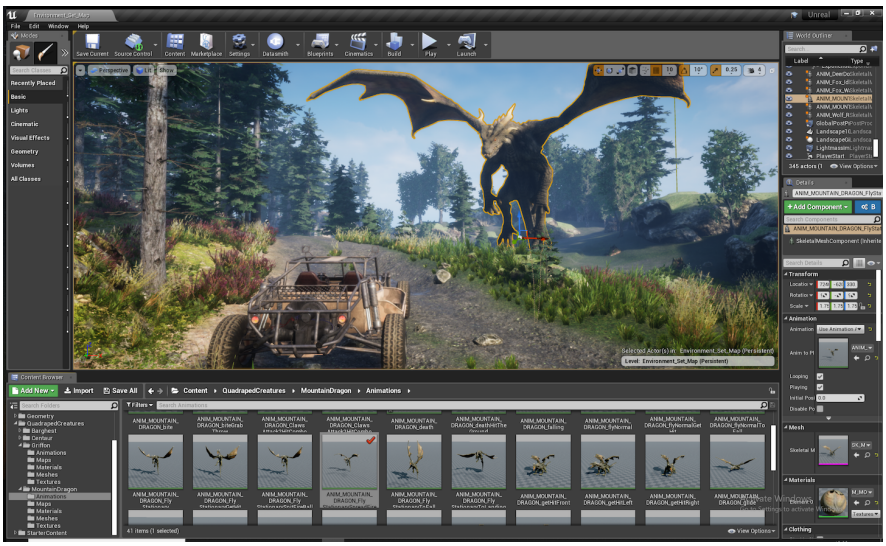

Figura 3.3: Imagen de la interfaz de usuario de Unreal Engine

Desde 2014 y con su versión Unreal Engine 4, la cual supuso una gran mejora en el apartado gráfico y de scripting de la herramienta, este software puede utilizarse de forma gratuita permitiendo su uso comercial con un modelo basado en *royalties*. Estas royalties consisten en recibir un porcentaje de los ingresos a los desarrolladores cuando alcanzan cierto umbral de ingresos con sus ventas.

Una de las características a destacar de esta plataforma es el sistema de scripting visual basado en *blueprints*. En la figura 3.4 podemos ver un ejemplo de uso de este sistema. Se basa en el concepto de planos interconectados por lineas y que se utilizan para definir, dentro del paradigma de la programación orientada a objetos, clases y objetos dentro del motor. Este tipo de sistema combina bien con el lenguaje de programación que utiliza (C++), y permite a los programadores diseñar implementaciones que sirven como bases para el sistema. Además, este enfoque permite que diseñadores puedan extender partes del videojuego generadas por programadores al ser un lenguaje muy visual y fácil de entender.

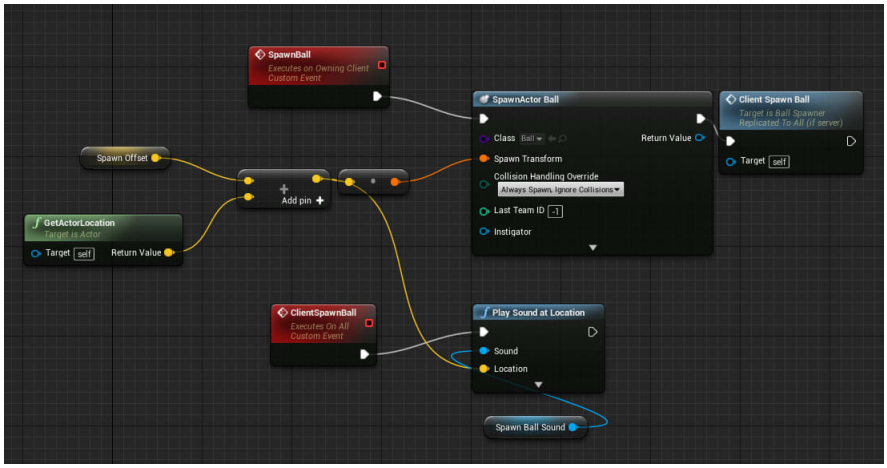

Figura 3.4: Desarrollo con blueprints en Unreal Engine

3.2.2. Unity

Unity es otro motor de juego creado por la compañía Unity Technologies. La primera vez que hizo aparición fue en la Conferencia Mundial de Desarrolladores de Apple en 2005, desarrollado para hacer funcionar y generar proyectos en los equipos de la plataforma Mac. Acabó alcanzando suficiente éxito como para continuar con el desarrollo del motor y el resto de herramientas.

Las versiones posteriores se centraron en desarrollar herramientas que pudieran captar la atención de desarrolladores más grandes, así como permitir su uso a desarrolladores independientes. En 2015 lanzan la versión 5 de Unity que permite soporte para docenas de plataformas y supone grandes mejoras para el apartado gráfico, así como soporte para vídeo en HD y Realidad Virtual.

El lenguaje de programación que se utilizado para la creación de *scripts* es C#. Estos permiten proveer de comportamientos a los diversos objetos de la escena. Actualmente la plataforma permite integración con herramientas de diseño como Blender, Maya, ZBrush o Adobe Photoshop. Cualquier tipo de cambio que se desarrolle en alguno de los objetos creados con estas herramientas, se actualiza de forma automática dentro del proyecto en Unity.

Otra de las característica que ofrece Unity es el soporte para diversas plataformas, permitiendo el desarrollo para Android e iOS, WebGL, PS4, Windows, MacOS y Linux, entre otros.

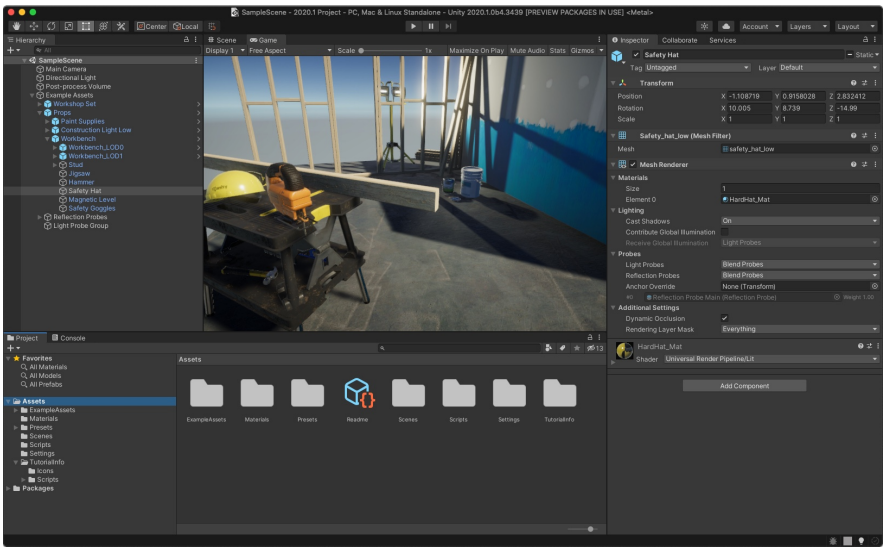

Figura 3.5: Imagen de la herramienta Unity

3.3. SISTEMAS DE CAPTURA DE MOVIMIENTO

Los sistemas de captura de movimiento o Mocap son un conjunto de técnicas que se utilizan para la grabación de movimiento de un sujeto real (persona o animal) y representar dicho movimiento en un modelo digital. Se utilizan principalmente en la industria del cine y de los videojuegos aunque también tiene aplicaciones militares, deportivas y médicas.

Principalmente existen dos tipos de tecnologías utilizadas en los sistemas para la captura de movimiento: sistemas ópticos y no-ópticos [8].

3.3.1. Sistemas ópticos

Dentro de los sistemas de captura que usan la primera tecnología, los **sistemas ópticos**, podemos distinguir varios tipos de captura de movimiento: captura óptica infrarroja y captura basada en vídeo. Las principales diferencias entre estas técnicas, a parte del apartado tecnológico, son el coste y complejidad de las mismas lo que influye en la accesibilidad a estos sistemas. Dentro de la captura óptica infrarroja podemos distinguir otras dos técnicas: captura óptico-pasiva y óptico-activa.

- **Técnica óptico-pasiva**: esta técnica hace uso de sensores que reflejan fuentes de radiación para indicar la posición de cada una de ellas a un detector. Este detector o cámara infrarroja iluminan los sensores y almacenan las coordenadas de cada uno de ellos para luego procesar la información y calcular los parámetros del movimiento.
- **Técnica óptico-activa**: en esta técnica se utilizan sensores que emiten rayos hacia los detectores en lugar de reflejarlos como en la técnica anterior. Estos sensores suelen ser diodos de emisión (LED) conectados al traje de captura de movimiento y un software es encargado de triangular sus posiciones y obtener así el movimiento del actor.
- **Método por vídeo**: no hace uso de sensores ni trajes. Se emplean cámaras para seguir el movimiento del sujeto y mediante el uso de algoritmos avanzados se realizan cálculos en tiempo real para el rastreo del movimiento. Es el método más barato y accesible actualmente.

Existen diversas empresas que comercializan con sistemas de captura de movimiento y que emplean estas técnicas. A continuación vamos a mencionar algunas de ellas y los productos que tienen a su disposición junto con la técnica que utilizan:

- *Biosense Medical Simi*[12]:
  - *Aktisys*[13]: emplea marcadores LED activos para emitir color que luego es procesado con su sistema software. Se puede utilizar tanto con grabaciones como con streaming de vídeo.
  - *Simi Motion 2D/3D*[14]: sistema que utiliza 8 cámaras junto con sensores pasivos y tecnología infrarroja. También incorpora técnicas basadas en imagen.
  - *Simi Shape 3D* [15]: es un sistema de captura 3D que no hace uso de sensores. Utiliza 8 cámaras para extraer la silueta de la persona que está grabando y así generar un modelo 3D con las animaciones.
- *Qualisys*[16]: provee de diferentes tipos de cámaras y sensores para su uso combinado con su software. Ofrecen tanto sistemas para *tracking* con sensores activos como sin sensores (método por vídeo).
- *OptiTrack*[17]: al igual que la compañía anterior, provee del hardware necesario para su uso junto con su sistema de software. Disponen de componentes para el uso de las tres tipos de técnicas.

3.3.2. Sistemas no-ópticos

Por otra parte, existen sistemas que están basados en la segunda tecnología, denominados **sistemas no-ópticos** o también conocidos como sistemas de captura inerciales. En este tipo de sistemas los trajes incorporan sensores que hacen uso de acelerómetros y giroscopios para mantener un constante registro de las velocidades de rotación y traslación. Mediante una comunicación *wireless* con el receptor, los emisores proporcionan la posición y orientación de cada una de las partes del actor para obtener la animación del movimiento.

- *Xsens*[18]: oferta su sistema en 3 packs diferentes. Todos incorporan 17 sensores *wireless* y trajes para acoplarlos. Los rangos de comunicación varían segun el pack.
  - *MVN Awinda Starter.*
  - *MVN Awinda.*
  - *MVN Link.*
- *STT Systems*[19]: esta compañía ofrece el sistema *iSens*[20] que incluye el hardware y software necesario para la captura de movimiento inercial.

Figura 3.6: Sistemas de captura de movimiento Xsens

[12] https://biosensemedical.com/
[13] https://biosensemedical.com/active-marker-motion-capture/
[14] https://biosensemedical.com/passive-marker-motion-capture/
[15] https://biosensemedical.com/markerless-motion-capture/
[16] https://www.qualisys.com/
[17] https://optitrack.com/
[18] https://www.xsens.com/
[19] https://www.stt-systems.com/
[20] https://www.stt-systems.com/motion-analysis/inertial-motion-capture/isen/

3.4. SOFTWARE DE GRÁFICOS 3D

Son sistemas utilizadas en el ámbito de los gráficos por computador que hacen uso de la representación tridimensional de datos geométricos para producir y manipular Computer-Generated Imagery (CGI). Este tipo de software es ampliamente utilizado en las industrias cinematográfica y de videojuegos para generar imágenes 3D que representen desde modelos anatómicos a modelos industriales, animaciones o Efectos Visuales (VFX). Estas herramientas también son utilizadas para otros propósitos como son simulaciones interactivas lo cual permite su uso extendido en otros campos como la arquitectura, la automovilística o la medicina.

A continuación se presenta un listado con algunas de las aplicaciones de modelado y animación 3D con mayor popularidad:

- ***Blender***: es un software multiplataforma gratuito y de código libre para la creación de gráficos tridimensionales lanzado inicialmente en 1994. Da soporte a una gran cantidad de características del mundo 3D como son el modelado y la escultura, animación, simulación y rigging. También dan soporte a edición de video, captura de movimiento y animación 2D. Integra las tecnologías de Eevee y Cycles para el renderizado de escenas en 3D.
- ***3D Studio Max***: software para la creación y animación de gráficos 3D desarrollado por Autodesk. Salió a la venta por primera vez en 1990 y ha ido evolucionando a lo largo de los años hasta su última versión de marzo del 2020. Ofrece herramientas para la creación de diseños 3D de alta calidad, animación y efectos visuales. Hace uso del renderizado Arnold para la manipulación de personajes, escenas y efectos de gran complejidad.
- ***ZBrush***: es una herramienta de modelado 3D, escultura y pintura digital. Se basa en el uso de pinceles para dar forma, textura o pintar arcilla en tiempo real y crear así modelos e ilustraciones con gran rapidez. Su lanzamiento inicial fue en 1999 y constituye un estándar en la industria del arte digital.
- ***Maya***: también desarrollado por Autodesk, se lanzó oficialmente a la venta a principios de 1998. Es un software 3D de animación por ordenador con herramientas avanzadas para el desarrollo profesional de gráficos por computador, efectos especiales, animación y dibujo. Posee también herramientas para la simulación de ropa, cabellos y simulaciones de fluidos.
- ***Cinema 4D***: es un software profesional dedicado a la creación de gráficos 3D, animación, simulación y renderizado. Posee una alta velocidad de renderización y una gran cantidad de herramientas para hacer más accesibles y eficientes los flujos de trabajo en 3D. Una de las características más destacadas de Cinema 4D es su capacidad de añadir módulos especializados en función de las necesidades del proyecto.

3.5. SISTEMAS DE REPRESENTACIÓN DE ACTIVIDAD FÍSICA

En la actualidad, existen diversos programas que buscan realizar intervenciones de actividad física en niños. La mayor parte de ellos hacen uso de vídeos para la representación de ejercicios. Hasta el momento, no se ha encontrado ningún sistema basado en una plataforma web que se asemeje a la solución aquí propuesta.

3.5.1. GoNoodle

La plataforma de GoNoodle está creada por expertos en el desarrollo de los niños y engancha a millones de niños cada mes para que se levanten y realicen actividad física por medio de juegos tanto en casa como en los colegios de forma gratuita.

Ofrecen dos aplicaciones móvil disponibles tanto en Android como en iOS para que los niños realicen actividad física en cualquier lugar: GoNoodle Games y GoNoodle Kids Videos. Su objetivo es hacer que los niños sean mas activos mientras juegan.

Figura 3.7: Logo de la plataforma GooNoodle

En 2013 comenzaron a colaborar con profesores para llenar las aulas con vídeos que fueran atractivos para los niños y que los mantuvieran en movimiento. Finalmente en 2018, lanzaron su primera aplicación **GoNoodle Kids Videos** en diferentes plataformas, donde recopilan más de 300 vídeos con canciones para niños, bailes, yoga y ejercicios.

La segunda aplicación, GoNoodle Games, fue lanzada en 2020. En los últimos años el tiempo que los niños pasaban frente a la pantalla ha sufrido un incremento significativo. Con esta aplicación buscan intervenir durante este tiempo. Ha sido creada en colaboración con la empresa Dubit, lider en desarrollo de juegos para niños en plataformas móviles. Esta aplicación utiliza la librería de código abierto OpenCV que incorpora visión por computador y técnicas de machine learning para la detección del movimiento de los niños.

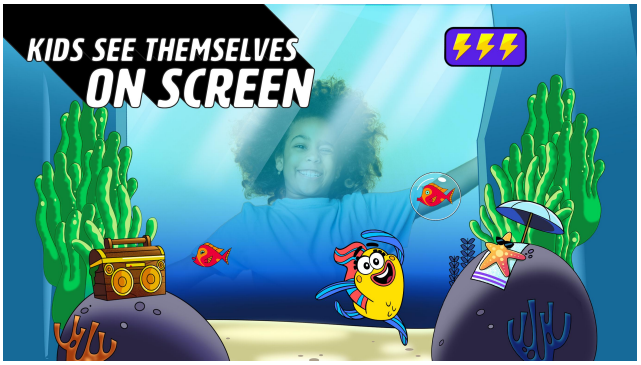

Figura 3.8: Imagen promocional de la aplicación GoNoodle Games

3.5.2. Little Sports

Cabe mencionar que existe una gran variedad de vídeos que pueden ser encontrados en YouTube que sirven como guía para mucha gente a la hora de realizar actividad física. Aunque no sean sistemas software como tal, hay algunos que utilizan software de creación de gráficos en 3D y animaciones. Un ejemplo de ello son los vídeos creados y subidos en el canal de YouTube de *Little Sports*.

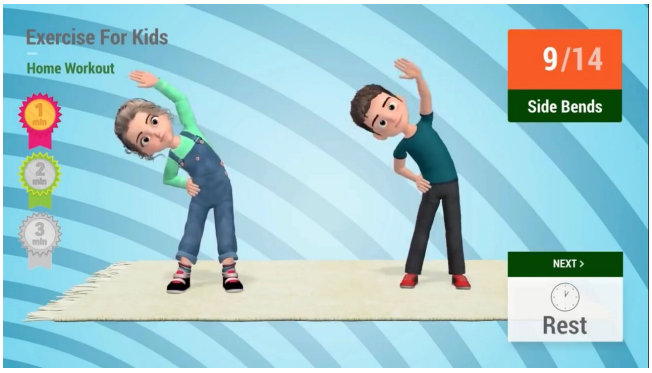

Figura 3.9: Imagen extraída de los vídeos de YouTube del canal de Little Sports

CAPÍTULO 4

Método de trabajo

A lo largo de este capítulo se especifica el método de trabajo que se ha llevado a cabo, el contexto sobre el que se ubica el proyecto, y los medios que han sido empleados en el desarrollo de este. En otras palabras, este apartado se centra en la metodología seguida, las herramientas software y lenguajes que han sido utilizados, así como los medios hardware de los que se han dispuesto para el mismo.

4.1. METODOLOGÍA

Anteriormente, en el apartado de introducción, se había mencionado que este proyecto está desarrollado bajo supervisión de la spin-off Furious Koalas S.L. y, por lo tanto, llevado a cabo por un equipo de desarrollo de la misma empresa. Dicho equipo se compone de 4 miembros pertenecientes a diversas ramas los cuales se han encargado de distintos aspectos del proyecto:

- *Animador*: se ha encargado de darle vida a los avatares utilizando las animaciones capturadas con el sistema de captura de movimiento adquirido.
- *Diseñadora Gráfica*: ha proporcionado todos los diseños para el apartado visual del sistema. Esto consiste en el diseño de la propia UI del juego y el diseño de las diferentes páginas web.
- *Programador Frontend*: se ha encargado de desarrollar toda la funcionalidad de la parte visual del sistema: la traducción de requisitos del sistema en funcionalidades y darle vida a los diseños. El autor del presente documento se ha centrado en las tareas relacionadas con este apartado.
- *Programador Backend*: se ha encargado del apartado de almacenamiento persistente del sistema, desarrollando una base de datos que se pueda utilizar para almacenar y consultar datos por parte de la plataforma web.

4.1.1. Desarrollo iterativo e incremental

Dada la naturaleza del proyecto, y la necesidad de mantener un feedback sobre las características del proyecto, se ha necesitado una metodología que permitiera realizar de forma paralela la implementación de nuevas funcionalidades junto con la revisión de las ya existentes. Para el desarrollo del proyecto, se ha seguido una metodología basada en un **desarrollo iterativo e incremental**[6]. Con esta metodología ágil, se pretende establecer hitos durante ciertos periodos de tiempo para ir adaptando nuestros planes según vayamos aprendiendo.

La Figura 4.1 muestra de forma visual y generalizada el proceso de desarrollo de una iteración. Cada una de las iteraciones definidas se establecen como desarrollos cíclicos pasando por las fases de *Análisis de requisitos*, *Diseño*, *Implementación* y *Pruebas*.

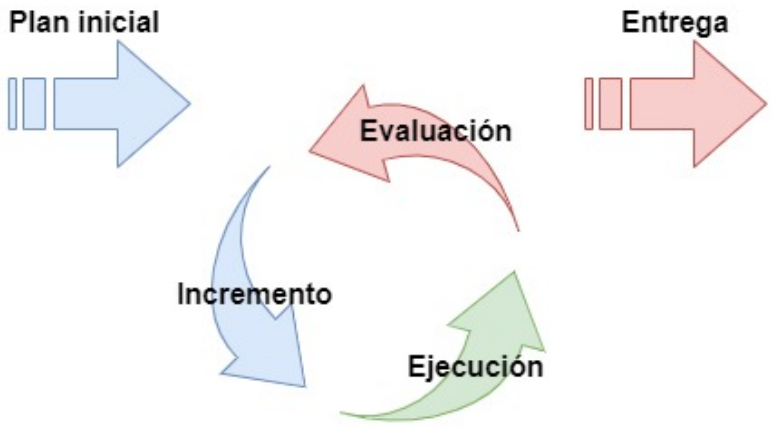

Figura 4.1: Modelo de desarrollo iterativo e incremental

Planificación

En esta sección se pretende dar a conocer la planificación que se ha llevado a cabo para la realización de cada uno de los objetivos propuestos en el Capítulo 2. A continuación, se trata de manera muy resumida cada uno de los hitos realizados a lo largo de los meses de desarrollo:

- **Evaluación y selección del avatar**: es una de las primera tareas que se realizó. Desde el Consistió en la búsqueda y selección de posibles avatares para integrar en el sistema. La duración de esta tarea se alargó desde el 02/12/2020 hasta el 15/12/2020 debido a que se realizaron pruebas de aceptación con niños.
- **Estudio de patrones de diseño**: debido a los escasos conocimientos sobre el desarrollo de videojuegos, se tuvo que realizar un estudio sobre los diferentes patrones de diseño utilizados en este campo. También se estudiaron diferentes arquitecturas de juego y tuvo una duración desde el 22/12/2020 hasta 19/01/2021.
- **Implementación en Unity**: comprende el desarrollo de las funcionalidades del módulo de representación 3D. Es la tarea que más se alargó en el tiempo debido a su posterior incorporación del sistema de eventos, animaciones y diseños, así como revisión de errores y cambios debido al feedback. Esta tarea se comenzó 20/01/2021 y finalizó el 24/06/2021.
- **Sistemas de eventos**: esta tarea ha sido el eje principal en el que se ha basado el módulo de representación 3D. Integra un sistema de eventos que permite un total desacoplamiento entre los diferentes componentes del proyecto en Unity. Se realizó una primera fase de estudio previa para su posterior integración al sistema con una duración desde 03/02/2021 hasta 17/02/2021.
- **Estudio del sistema de captura de movimiento**: dada la incorporación de un sistema completamente nuevo se realizó un estudio del funcionamiento del mismo, evaluaciones previas sobre la generación de animaciones y la integración de las mismas. Comenzó 24/02/2021 y duró hasta 16/03/2021. Abarca 2 sesiones de grabaciones de prueba y el tratamiento de las animaciones (junto con documentación del procedimiento).
- **Grabaciones de las animaciones**: una vez preparado el listado de ejercicios a incorporar, se realizó una sesión de grabación. Posteriormente, algunas de ellas fueron adaptadas a los avatares y se dejó documentado el proceso a seguir para posponer el desarrollo y que otro miembro del proyecto la retomase más adelante. Se realizó una sesión de grabación el días 26/03/2021.
- **Implementación de la plataforma web**: tarea destinada a la implementación de la plataforma web. Su desarrollo comenzó el 08/04/2021 con la incorporación del segundo programador y se fue desarrollando paralelamente la parte del *frontend* y *backend*. Debido al bajo conocimiento

de programación web, esta tarea se alargó hasta el 15/07/2021 mejorando en el proceso el diseño y arquitectura de la aplicación.

- **Implementación de diseños**: tarea dedicada a incorporar tanto a la plataforma web como al módulo de representación 3D sus diseños. Se comenzó a desarrollar este hito al recibir los diseños de la web el 06/05/2021 y unas nuevas modificaciones en el diseño que incluían esta vez el apartado visual del módulo de representación 3D en 07/06/2021. Se terminó con esta tarea el 15/07/2021.

Tabla 4.1: Tiempo empleado en cada tarea para el desarrollo del sistema

	Dic	Ene	Feb	Mar	Abr	May	Jun	Jul
1. Evaluación y selección del avatar								
2. Estudio de patrones de diseño								
3. Implementación en Unity								
4. Sistema de eventos								
5. Estudio del sistema de captura								
6. Grabaciones de las animaciones								
7. Implementación plataforma web								
8. Implementación de diseños								

4.1.2. Control y coordinación

Para llevar a cabo un control y una coordinación de las tareas y características que se han desarrollado por parte de los programadores, se consensuó el uso de una herramienta basada en la metodología *Kanban*. Esto permite visualizar el flujo del trabajo de una manera sencilla en cada una de las fases del proceso de desarrollo. La herramienta que está al cargo de esta supervisión es **Trello**, la cual nos proporciona tableros que podemos dividir en varias columnas. En cada una de las columnas se crean las tarjetas que se necesitan para el desarrollo de cada una de las tarea y los encargados de llevar a cabo esa tarea pueden unirse a dicha tarjeta. Las siguientes columnas son las que se han establecido para llevar a cabo el desarrollo del sistema:

- ***Backlog***: esta columna contiene las tareas que han ido surgiendo en base a peticiones por parte de los clientes.
- ***To-Do***: aquí podemos encontrar tareas que están listas para ser realizadas. Son aquellas tareas que surgen de forma urgente o que ya están listas para ser llevadas a cabo.
- ***In Progress***: tareas que se están llevando a cabo en un preciso momento a lo largo del proyecto.
- ***Blocked***: aquí encontramos tareas que, por algún tipo de problema o necesidad, han tenido que ser pausadas ya sea por mayor prioridad o urgencia.
- ***Testing***: aquellas tareas que están siendo puestas bajo pruebas. Se están comprobando que funcionen correctamente antes de marcarlas como finalizadas.
- ***Finished***: una vez una tarea ha sido desarrollada sin problemas son movidas a la columna de tareas acabadas.

Hay que tener también en cuenta, que para llevar a cabo una organización más descriptiva, las tarjetas van a hacer uso de un sistema de etiquetado que también proporciona la plataforma *Trello*. Indican qué aspecto del sistema corresponde a cada tarea. Las etiquetas que se han elaborado para este fin son las siguientes:

- *Backend*: tareas correspondientes con el desarrollo de la base de datos y las consultas (backend). Se utiliza el color verde para distinguirlo.

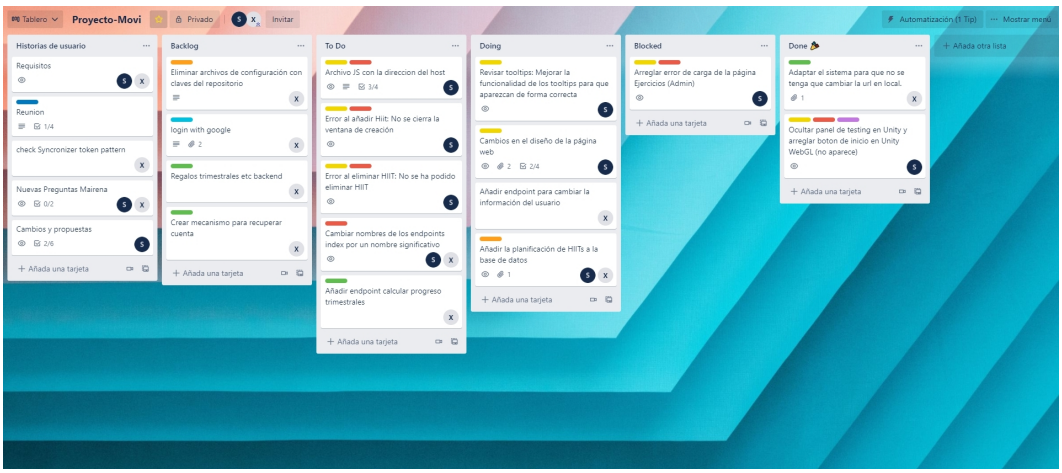

Figura 4.2: Captura del tablero creado en Trello para el proyecto MOVI-HIIT

- *Frontend*: usa el color amarillo y es asignado a aquellas tareas para el desarrollo de la parte web (frontend).
- *Management*: para indicar tareas que traten algo relacionado con la gestión de archivos del proyecto o con sistema de control de versiones. Utiliza el color naranja.
- *Bug*: con esta etiqueta, que utiliza el color rojo, se indica que la tarea se encarga de solucionar algún error o problema que aparezca en el sistema.
- *Unity*: En color morado están todas aquellas etiquetas que indican que una tarea se corresponde con una característica de Unity.
- *Preguntas*: a lo largo del proyecto surgen dudas que tendrán que ser consultadas y resueltas con los clientes. Para ello se ha establecido esta etiqueta con el color azul oscuro.
- *EuMove*: hay también unas tareas específicas que se corresponden con una variante del proyecto a nivel europeo. Se dejarán marcadas con una etiqueta en color azul claro.

4.2. MEDIOS SOFTWARE

En esta sección se describen los medios software utilizados a lo largo del desarrollo del proyecto. Para ello, se han definido tres apartados separados dentro de esta sección que se corresponden con: herramientas software, lenguajes de programación. bibliotecas y extensiones.

Dentro de las herramientas se encuentra todo el software que ha sido utilizado durante el desarrollo. En lenguajes de programación se incorpora una breve descripción del lenguaje utilizado junto con la justificación del uso del mismo. Y por último las bibliotecas y extensiones utilizadas para facilitar tareas dentro del desarrollo del proyecto. Estas se han dividido en dos secciones: bibliotecas utilizadas en el Package Manager de Unity y extensiones instaladas en Visual Studio Code.

La coordinación de archivos y documentos con el resto de integrantes del equipo se realizará a través de un repositorio compartido de *Drive*. En este repositorio podemos se encuentran los avatares que se han utilizado, todas las animaciones que se han generado durante las sesiones de grabación con el sistema de captura de movimiento, un primer prototipo del proyecto, los diagramas correspondientes al sistema y documentación relacionada con el proyecto. También se tienen almacenadas diversas presentaciones que se han realizado en diversas reuniones.

Para definir el diseño del sistema y su propio comportamiento, se ha hecho uso de la aplicación *diagrams.net*, que nos permite integrar diagramas UML en el propio Drive.

4.2.1. Herramientas software

- **Windows 10 Home 64-bit**: versión del Sistema Operativo que ha sido empleado para desarrollar el proyecto. Se ha utilizado por la compatibilidad que tiene con el resto de herramientas.
- **Visual Studio Code v1.58.2**[1]: editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Incluye resaltado de sintaxis, recomendación inteligente de código y refactorización, entre otras características. Es altamente personalizable y permite el uso de extensiones creadas por los propios usuarios para adaptarse a las necesidades del programador.
- **Unity** [2]: como ya hemos podido observar en el Capítulo 3, Unity incorpora facilidades para su integración en plataformas web a través de Unity WebGL. Para hacer uso de este motor se ha recurrido a dos software pertenecientes a Unity Technologies:
  - *Unity Hub*: herramienta que facilita la instalación de las diferentes versiones del Editor de Unity y permite la gestión de proyectos indicando la versión y la plataforma que se quieren utilizar para cada uno de ellos.
  - *Unity 2020.3.13f1 (LTS)*: es la última versión Long Term Support de 2020 del Editor, instalada a través de Unity Hub para llevar a cabo el desarrollo del proyecto. Es un IDE que permite llevar un control de todos los componentes dentro del workspace, de la escena y configurarlos según necesidades específicas.
- **Blender 2.9** [3]: software multiplataforma gratuito y de código libre para la creación de gráficos tridimensionales. Proporciona soporte para modelado y la escultura, animación, simulación y rigging.
- **Git v2.27** [4]: software de control de versiones utilizado para coordinar los cambios realizados en los archivos del código del proyecto. Esta coordinación se ha llevado a cabo empleado Git junto con el servicio de alojamiento *Bitbucket*.
- **Amazon AWS**[5]: Amazon Web Services es una plataforma que proporciona servicios de cómputo en la nube para el procesamiento y almacenamiento de datos. En el caso del presente proyecto ha sido utilizado para la creación y puesta en vivo de una base de datos *PostgreSQL*. Las características del servicio utilizado son las siguientes:
  - Base de datos: PostgreSQL versión 12.6
  - Ubicación del servidor: Frankfurt (eu-central-1)
  - Características principales: 1 vCPU y 1 GB de RAM
  - Capacidad de almacenamiento: 20 GB
- **PostgreSQL v12.6**[6]: es un sistema de gestión de bases de datos relacionales de código abierto. Se ha empleado para la creación y lanzamiento de la base de datos que va a dar soporte al backend del proyecto.
- **DbSchema v8.4**[7]: aplicación para el diseño y gestión de esquemas de bases de datos. Se ha utilizado esta aplicación por parte del programador backend para realizar el diseño de la base de datos. También ha permitido que el autor consulte el diseño y contenido de la base de datos.
- **Postman v8.9.1**[8]: es una plataforma colaborativa para el diseño y ejecución de peticiones REST, SOAP y GraphQL. Al igual que con la aplicación anterior, se hizo uso por parte del miembro encargado del backend para crear las peticiones y así permitir al autor consultar el funcionamiento de las mismas.
- **Servidor FTP**: es un programa que se ejecuta en el lado del servidor para la transferencia

[1] https://code.visualstudio.com/
[2] https://unity.com/es
[3] https://cloud.blender.org/welcome/
[4] https://git-scm.com/
[5] https://aws.amazon.com/es/
[6] https://www.postgresql.org/
[7] https://dbschema.com/
[8] https://www.postman.com/

de archivos mediante el protocolo de red FTP. OVH[9] ha servido de host para permitir el lanzamiento en vivo de la plataforma web.
- **Cyberduck v7.8.5**[10]: es una aplicación cliente de código abierto con soporte para los protocolos FTP y SFTP, entre otros. Se ha hecho uso de esta herramienta software para subir los archivos necesarios y lanzar la plataforma web
- **MVN Animate** [11]: es el software de captura de movimiento de Xsens. Permite la calibración del traje Xsens Awinda, visualización en tiempo real y grabación de los datos de captura de movimiento.

Cabe destacar que también se han empleado otro tipos de herramientas software para realizar conferencias y reuniones con el fin de discutir diferentes aspectos a lo largo del ciclo de vida del proyecto:

- **Whereby**[12]: plataforma que ofrece salas gratuitas para realizar videoconferencias y compartir pantalla entre los miembros de la misma sala. Se ha empleado para realizar reuniones entre los miembros pertenecientes a Furious Koalas.
- **Microsoft Teams**[13]: esta plataforma sirve para la comunicación y la colaboración entre usuarios y ha servido de sala de reuniones para los integrantes del proyecto MOVI-HIIT pertenecientes a la facultad de Magisterio y los miembros de Furious Koalas que todavía forman parte directa de la UCLM.

.

4.2.2. Lenguajes de programación

Todos los archivos de código pertenecientes a los distintos lenguajes de programación que han sido utilizados en el proyecto, han sido codificados utilizando el editor de código fuente Visual Studio Code junto con una serie de extensiones que facilitan la redacción y legibilidad del propio código. A continuación vamos a hacer un repaso de aquellos lenguajes que se han empleado y la justificación de su uso:

- **C#**: Es el lenguaje de programación que se ha utilizado para realizar scripts de comportamiento en el editor de Unity.
- **HTML**: es el lenguaje de marcado que se ha utilizado para el desarrollo y despliegue de la plataforma web.
- **Cascade Style Sheet (CSS)**: lenguaje de diseño gráfico que se ha empleado para convertir los diseños realizados por la diseñadora gráfica en parte de la plataforma web.
- **JavaScript**: toda la funcionalidad de la plataforma web ha sido construida utilizando JavaScript. También se ha utilizado la técnica de desarrollo AJAX para realizar peticiones a la base de datos y crear así una plataforma dinámica.
- **PHP**: el ultimo lenguaje a mencionar tiene un papel muy importante en el proyecto. Forma parte del código que sirve de intermediario entre las peticiones realizadas por JavaScript y la base de datos. También ha servido para lanzar la parte backend como un servidor en local y realizar pruebas junto con la extensión Live Server.

[9] https://www.ovh.es/
[10] https://cyberduck.io/
[11] https://www.xsens.com/products/mvn-animate
[12] https://whereby.com/
[13] https://www.microsoft.com/es-es/microsoft-teams/group-chat-software

4.2.3. Bibliotecas y Extensiones

- **Package Manager de Unity**: dentro del propio editor de Unity encontramos un gestor de bibliotecas que nos permiten configurar el proyecto para adaptarlo a las necesidades del proyecto. Las bibliotecas que han sido instaladas a través de este gestor son las siguientes:
  - *Unity UI*: biblioteca para el desarrollo de UI para juegos y aplicaciones. Está basado en el uso de GameObjects y Componentes para definir el diseño y la posición de los diferentes elementos del UI.
  - *TextMeshPro*: sirve de reemplazo para el tipo de dato Text y amplía las funcionalidades de los textos en las UI de Unity.
  - *Visual Studio Code Editor*: ofrece soporte para la integración de Visual Studio Code como editor de código en Unity.
  - *SerializableDictionary*[14]: desarrollado por BeetleCircus, consiste en una biblioteca que proporciona diccionarios genéricos serializables en la propia interfaz del editor.
- **Extensiones de Visual Studio Code**:
  - *Beautify*: permite estructurar el código de un archivo JS, HTML, CSS y JSON de una manera que sea más legible. Esta extensión sirve de gran ayuda para organizar las etiquetas en los documentos HTML.
  - *Live Server*: esta extensión ha sido crucial a la hora de realizar pruebas sobre los cambios realizados en la plataforma web. Permite lanzar de forma local un servidor para cargar páginas web estáticas y dinámicas. Para poder hacer uso de las peticiones en AJAX se ha combinado con el uso de PHP para la creación de servidores en local.

4.2.4. Medios Hardware

Para finalizar este capítulo, se van a listar los medios hardware que han sido utilizados para la realización de este proyecto:

- **Computadora**: Ordenador sobremesa propiedad del autor, utilizado para el desarrollo del proyecto. Las características de las que dispone el computador son las siguientes:
  - Sistema operativo Windows 10 Home de 64-bit.
  - Procesador con 4 núcleos a 3.50GHz.
  - Gráfica GTX 1060 con 6GB de VRAM.
  - 16 GB de memoria RAM.
- **Sistema de captura de movimiento Xsens Awinda**: sistema facilitado por el Instituto de Tecnologías y Sistemas de Información (ITSI), concretamente por el laboratorio *WeCareLab* para su uso en la grabación y creación de las animaciones del proyecto.

[14] https://wiki.unity3d.com/index.php/SerializableDictionary

CAPÍTULO 5

Arquitectura del sistema

Este capítulo está dedicado al análisis de los diferentes componentes que conforman la arquitectura del proyecto que ha sido desarrollado a lo largo de estos meses. En primer lugar, se ofrece una visión panorámica de cómo está formado el sistema y los principales componentes, así como su funcionamiento. Después, se profundiza en cada uno de estos componentes principales, describiendo de forma más detallada su funcionamiento junto con los elementos que forman parte de cada uno de ellos. Además, se se explica el papel que desempeña cada elemento dentro del sistema. Aparte, este capítulo también se centra en el uso de patrones de diseño, los cuales han permitido elaborar una arquitectura robusta y escalable.

Pero, antes de entrar en detalle es importante establecer una primera visión general del sistema. A continuación se detalla, a rasgos generales, los objetivos que desempeñan los componentes principales del sistema:

- **Servidor**: este elemento es el encargado de recibir y gestionar las peticiones que realiza el cliente. Entre las tareas más importantes, el servidor almacena información de los usuarios que utilizan la plataforma, de la actividad física que se va a llevar a cabo durante el programa y de las recompensas. También permite la gestión de la misma por parte de los encargados del programa.
- **Cliente**: este elemento es el encargado de solicitar y recibir la información almacenada por el servidor, procesarla y mostrarla a los usuarios según sus peticiones. Integra también el módulo de representación 3D que va a mostrar la actividad física, pilar fundamental de este proyecto.

Tanto el servidor como el cliente están formados a su vez por más elementos (ver Figura 5.1). Estos elementos dotan a cada uno de los componentes principales de funcionalidades para realizar tareas específicas y conseguir los objetivos definidos en el apartado 2.2. En la figura 5.1 podemos observar una vista genérica de los elementos que forman parte del cliente y del servidor. A pesar de que en los apartados 5.2 y 5.1 se profundiza sobre el funcionamiento del resto de los elementos, a continuación se describe qué trata de conseguir cada uno de ellos:

- **Sistema de acceso a la plataforma**: está formado por una página de login y una página de registro. Permite el acceso a usuarios y administradores para el uso de la plataforma. Actualmente el registro libre está deshabilitado para la ejecución del programa MOVI-HIIT, ya que se va a hacer uso de enlaces para registrar a los participantes.
- **Plataforma para el Usuario**: se encarga de mostrar a los usuarios aquellas páginas con información relativa a su desempeño a lo largo del programa, así como la actividad física programada para cada día y la ejecución de la actividad que les corresponda.
- **Plataforma para el Administrador**: es la encargada de mostrar a los usuarios con rol de administración aquellas páginas para la gestión de los datos sobre la actividad física y los participantes.
- **Sistema de almacenamiento en la nube**: se encarga de almacenar en la base de datos toda

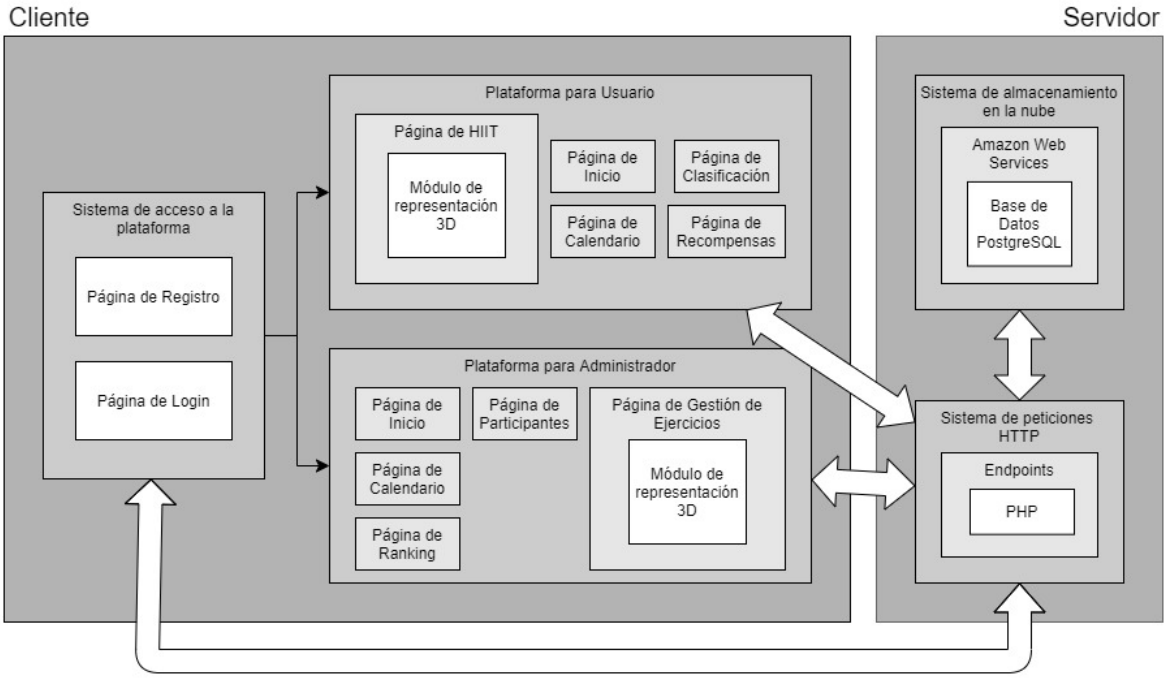

Figura 5.1: Vista general de la arquitectura del sistema.

la información que necesita la plataforma web.

- **Sistema de peticiones**: establece un enlace de comunicación entre la base de datos y la plataforma web. Envía la información solicitada por la plataforma a través de peticiones HTTP (HyperText Transfer Protocol).

Como podemos observar en la Figura 5.1, los diversos sistemas pertenecientes al Cliente, tanto el sistema de acceso como ambas plataformas, se comunican con el Servidor a través del Sistema de peticiones HTTP que a su vez tiene acceso a la Base de Datos en el sistema de almacenamiento en la nube. El patrón de diseño **Modelo-Vista-Controlador** (**MVC**) está presente en este tipos de sistemas, ya que se hace una separación entre la presentación de la aplicación, la lógica y los datos.

Así pues, las páginas HTML (junto con sus diseños en CSS) pertenecientes a los diferentes sistemas del Cliente forman parte de la **Vista**, ya que presentan la información obtenida en un formato adecuado para interactuar con él. Los archivos en JavaScript (también pertenecientes al Cliente) forman el componente del **Controlador**, ya que son los encargados de responder a las interacciones del usuario mediante eventos y realizar peticiones para solicitar información. Finalmente, el componente del **Modelo** estará representado por el Sistema de peticiones HTTP, que se encarga de operar con la información realizando consultas, gestionando los accesos a la misma y actualizándola.

En el Anexo A, el lector podrá encontrar información sobre la estructura del proyecto, así como el enlace al repositorio con el código fuente del mismo.

5.1. SERVIDOR

En este primer apartado describiremos los subcomponentes que forman parte del servidor del sistema, componente de vital importancia para el funcionamiento del proyecto. Hay que tener en cuenta que este componente no ha sido el objetivo de desarrollo por parte del autor, aunque haya colaborado en la definición de los requisitos del mismo. Por tanto, se describirá de una forma más breve que el apartado del Cliente (ver Apartado 5.2).

El Servidor está formado por un sistema de almacenamiento en la nube, y un sistema de peticiones

HTTP. Ambos sistemas juntos forman parte del backend de la plataforma y proveen a la misma de acceso a la información necesaria para realizar las distintas funcionalidades que ofrece: gestión de los datos, control de los usuarios, reproducción de actividad físicas, etc.

Vamos a comenzar describiendo el funcionamiento del sistema de almacenamiento en la nube y el diseño de la base de datos, y más adelante trataremos el funcionamiento del sistema de peticiones HTTP.

5.1.1. Sistema de almacenamiento en la nube

Se ha empleado los servicios de computación en la nube para mantener un servidor en constante funcionamiento. Este servidor ofrece acceso a una base de datos PostgreSQL que es encargada de almacenar la información necesaria para la plataforma. Las principales razones de utilizar PostgreSQL son la gran estabilidad y tolerancia a fallos que ofrece siendo un proyecto de código abierto. Es considerado el motor de base de datos más avanzado en la actualidad [3].

Para realizar el diseño de la base de datos se ha basado en los requisitos establecidos por el proyecto. El sistema requería almacenar información de los usuarios que la van a utilizar, diferenciar entre usuarios que utilizaran la plataforma para realizar la actividad física y otros usuarios que pudieran gestionar la información contenida en el sistema para realizar adaptaciones o cambios durante la puesta en marcha del programa; almacenar las recompensas que van desbloqueando los usuarios y almacenar los ejercicios y sus combinaciones para crear HIITs. Dicho esto, podemos diferenciar entre varios sistemas que han sido destacados en diferentes colores en la Figura 5.2.

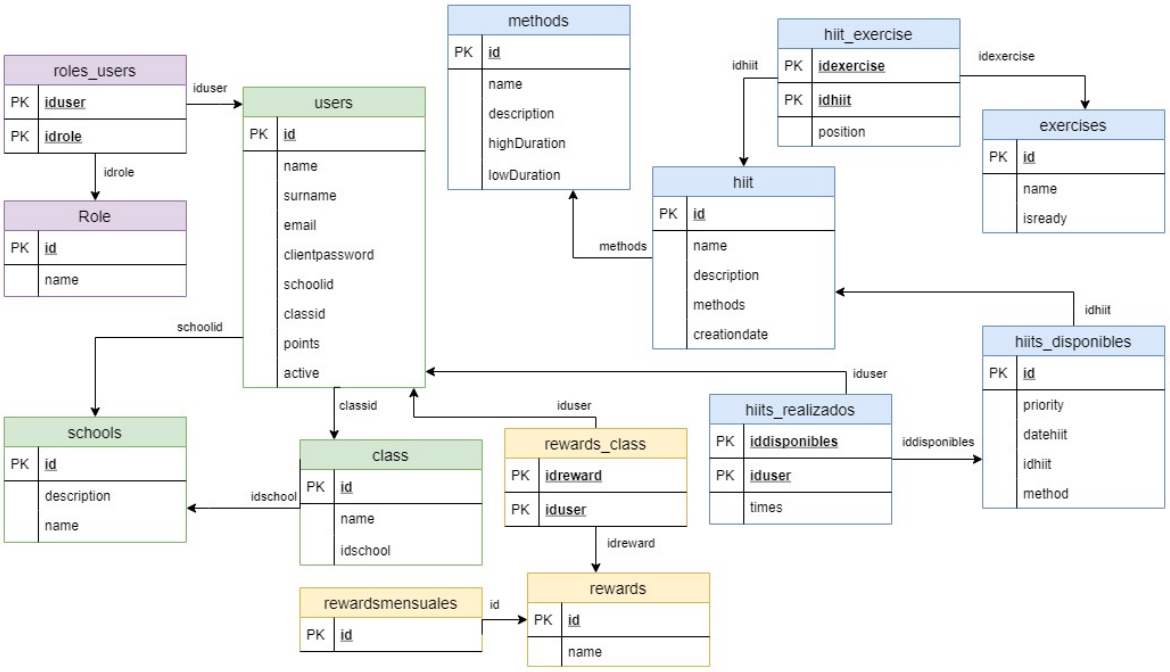

Figura 5.2: Diseño de la base de datos extraído del esquema en dbSchema.

En *verde* encontramos el **sistema de usuarios**. Con estas tablas almacenamos toda la información de los usuarios, así como la escuela y clase a la que pertenecen.

Las tablas que forman el **sistema de recompensas** aparecen en *amarillo*. Para cada una de las clases dentro de cada colegio, existen recompensas que van a quedar almacenadas aquí.

En cuanto a las tablas en *azul*, estas se encargan de almacenar información sobre los ejercicios y los HIITs, las cuales forman parte del **sistema de organización de la actividad física**. A cada

HIIT le corresponden 4 ejercicios y un método. En un principio el método iba a ser siempre el mismo, pero para permitir al sistema adaptarse a posibles cambios se decidió establecer una referencia y almacenar diferentes métodos si fuera necesario. Por otro lado, se lleva un control de los HIITs que se han realizado y los que quedan por hacer para cada uno de los usuarios.

Y por último, en *morado* encontramos el **sistema de roles** donde, actualmente, se pueden diferenciar los roles de usuario y administrador. Este diseño permite en cualquier momento incorporar cualquier otro tipo de rol según avance el proyecto y surjan las necesidades.

5.1.2. Sistema de peticiones HTTP

Este sistema es esencial para la obtención de la información almacenada por parte de la base de datos en el servidor. Se basa en el uso de *endpoints* que son, dentro de una red, puntos finales de comunicación que responden a diversas peticiones por parte de un cliente. A continuación se hace un repaso sobre los diferentes endpoints que dan funcionalidad al sistema de peticiones organizados según su finalidad:

- **HIITs**:
  - *[hiit/createHiit]* (createHiit): endpoint para la gestión de HIITS. Permite la creación de un nuevo HIIT en el sistema por parte de los administradores.
  - *[hiit/createHiit]* (updateHiit): permite actualizar algún HIIT ya existente en la base de datos. Entra dentro de la gestión de HIITS y solo puede ser usado por administradores.
  - *[hiit/index]* (getHiit): devuelve la información de un HIIT específico mediante una petición a este endpoint y el envío de un id de HIIT.
  - *[hiit/lastHiits]* (getLastHiitsCreated): devuelve los últimos 3 ejercicios que han sido añadidos al sistema.
  - *[hiits/addHiitRealizado]* (addHiitRealizado): permite que cuando se realice una sesión de actividad física, se marque como registrada para una clase en concreto.
  - *[hiits/getHiitsInterval]* (getHiitsInterval): obtiene los HIITs programados para los días indicados dentro de un intervalo enviado en la petición.
  - *addHiitDiario*:
  - *[hiit/getMyHiitsDailyLeft]* (getHiitsDiariosRestantes): devuelve el número de veces disponibles que se puede realizar un HIIT en cierto día (hasta 2 veces por día un mismo HIIT).
  - *[hiit/deleteHiit]* (deleteHiit): enpoint para la gestión de HIITs. Se encarga de borrar un HIIT específico del sistema.
- **Recompensas**:
  - *[rewards/getUserRewards]* (getUserRewards): devuelve las recompensas que han sido desbloqueadas por un Usuario en particular. Forma parte del control de usuarios para su seguimiento.
  - *[rewards/getRewards]* (getRewards): endpoint del que se obtiene todas las recompensas existentes en el sistema.
- **Ejercicios**:
  - *[exercises/index]* (getExercises): obtiene todos los ejercicios que han sido registrados en la base de datos.
  - *[exercises/getExercisesReady]* (getNameExercisesReady): este endpoint se utiliza actualmente para mostrar aquellos ejercicios que ya han sido transformados en animación para su visualización en el módulo de representación 3D por parte de los administradores para llevar tareas de corrección.
- **Escuelas**:
  - *[schools/getSchools]* (getSchools): endpoint para la obtención de las aulas de cada una de las escuelas que participan en el programa.

- **Participantes**:
  - *[participants/getParticipants]* (getParticipants): permite la obtención de las escuelas que participan en el programa.
- **Ranking**:
  - *[ranking/getRanking]* (getRanking): forma parte de las técnicas de gamificación para incentivar el uso de la plataforma. Devuelve un ranking global con todas las clases ordenadas según los puntos obtenidos al realizar los HIITs.
  - *[ranking/getRankingSchool]* (getRankingSchool): al igual que el endpoint anterior, devuelve un ranking ordenado pero esta vez solicitando única y exclusivamente las clases pertenecientes a una escuela según el id enviado en la petición.
- **Registro**:
  - *[register/register]* (register): permite el registro de un usuario haciendo uso únicamente de un email o usuario y una contraseña. El objetivo de este endpoint es el de crear enlaces de registro para invitar exclusivamente a aquellas aulas de los colegios participantes en el programa.
  - *[register/registerRole]* (registerRol): registra un rol para un usuario en concreto. Los roles existentes en el sistema son: usuario, administrador y desarrollador.
- **Usuarios**:
  - *[login/index]* (login): realiza una petición al servidor con los credenciales introducidos en la pantalla de Log In para comprobar que son correctos. En caso de ser correcto devuelve un token generado por parte de la base de datos para verificar a ese usuario.
  - *[checkRol/index]* (checkRol): comprueba el rol del usuario que está accediendo a la plataforma a través de un token de seguridad. Esto evita que se acceda a páginas u otros endpoints no destinados a ciertos usuarios.

El uso de cualquiera de estos endpoints está sujeto a una comprobación de un JSON Web Token (JWT)[1] del usuario que hace acceso a la plataforma. Un JWT sirve para definir una manera segura de transmisión de información a través de objetos JSON. Cuando el usuario realizar el login, tras ser verificados los credenciales, el Servidor devuelve un token específico que el Cliente guarda en el almacenamiento local. Cada vez que se quiera solicitar información al Servidor, se tendrá que enviar el token para su comprobación.

Esto permite realizar peticiones de forma segura, asegurándonos que el rol que tiene asignado dicho usuario no está realizando una petición de una consulta a la que no debería tener acceso.

5.2. CLIENTE

En este punto se define la estructura del Cliente y sus funciones principales. Como hemos podido observar en la Figura 5.1, el Cliente está formado por 3 elementos principales: el sistema de acceso a la plataforma, la plataforma para el usuario y la plataforma para los administradores. Cada uno de estos elementos está formado por las diversas páginas webs en cuyo conjunto conforman el diseño de la plataforma.

Antes de comenzar a indagar en el funcionamiento de cada uno de estos componentes del Cliente, hay que mencionar que la plataforma web está en funcionamiento y puede ser accedida, siempre y cuando se tengan credenciales de acceso a ella, a través del siguiente enlace:

https://movi-hiit.furiouskoalas.com/

Como ya se ha indicado durante el Capítulo 4, la plataforma web hace uso de un servicio de hosting FTP bajo la dirección que acabamos de mencionar. Esto permite que cualquier usuario pueda acceder y obtener aquellos archivos que hacen posible la ejecución de la web (HTML, CSS y JS).

[1] https://jwt.io/

Con respecto al diseño de la plataforma, aprovechando la capacidad de CSS de generar archivos y poder ser reutilizados, se han diseñado estilos genéricos para poder ser utilizados por todas las páginas. Para ello se han detectado los componentes comunes dentro de las distintas páginas y se han generado los archivos CSS:

- **sidebar.css**: se corresponde con el diseño del menú lateral donde encontramos la información sobre el usuario que ha accedido, las distintas páginas a las que puede acceder, y un botón de cierre de sesión. Este diseño se ha llevado a cabo siguiendo un tamaño de pantalla de 1920x1080. En caso de consumirse mediante un dispositivo móvil, este mismo archivo es el encargado de rediseñar la página y adaptarla. A partir de los 960 píxeles de anchura el diseño es adaptado a un diseño móvil. Se puede visualizar en las Figuras B.3 y B.4 del Anexo B.
- **content-side.css**: corresponde con todos los componentes que forman parte del apartado visual de la plataforma. Desde los paneles que forman las distintas secciones en la pantalla como los títulos y textos (ver Figuras B.5 y B.14). Al igual que en el estilo anterior, está diseñado para una visualización óptima en una resolución de 1920x1080. En ambos casos se han planteado unos diseños *responsive*.
- **scrollbar.css**: para consensuar un diseño de scrollbar personalizado y común entre las páginas web se ha creado este archivo CSS. Se elimina el uso de scrollbars en el eje x y solo se permite su existencia en el eje y para así ser adaptado a distintas resoluciones de forma *responsive* (ver Figuras B.16 y B.13).
- **month-calendar.css**: diseño para representar un calendario mensual. Se hace uso de la propiedad *grid-template-area* para definir cómo van a ser mostrados los distintos elementos. La plantilla define dos filas. En la primera encontramos los siguientes elementos de izquierda a derecha: botón izquierdo, nombre del mes y botón derecho. En la segunda fila encontramos los días de la semana junto con cada uno de los días que entran dentro de dicho mes. Se complementa con el archivo JavaScript *month-calendar.js* para crear un calendario mensual completamente funcional y sencillo de utilizar (ver Figuras B.11 y B.15).
- **weekly-calendar.css**: diseño para representar un calendario semanal. En este caso se hace uso de la propiedad *grid-template-columns* y se definen 5 columnas correspondientes a cada uno de los días de la semana (ver Figuras B.10 y B.14). Al igual que en el estilo anterior, tiene su versión en JavaScript para completar la funcionalidad.
- **tooltip.css**: muestra un pequeño *tooltip* con los ejercicios que corresponden a un determinado HIIT al pasar por encima de él con el ratón. Este estilo es utilizado principalmente en los calendarios mensuales y semanales.

Esto mismo ocurre con algunas funcionalidades de la plataforma que han sido recogidas en archivos independientes JavaScript para ser consumidos de forma genérica. A continuación vamos a listar los scripts que se han creado siguiendo estas pautas:

- **const-host.js**: este archivo de JavaScript sirve para almacenar una constante con la dirección del host al que tiene que hacerse las peticiones HTTP.
- **const-addr.js**: parecido al archivo anterior. Contiene constantes con las direcciones de cada uno de los endpoints. Las constantes tienen nombres representativos de lo que realiza el endpoint al que señalan.
- **checkAuth.js**: es el script encargado de comprobar en cada una de las páginas que se ha accedido de forma correcta a la plataforma haciendo uso del Log In y no directamente desde la URL (ver Algoritmo 5.1).
- **sidebar.js**: este archivo es el más importante para navegar por la plataforma. Se encarga realizar una inyección de código HTML en el cuerpo de la página web con las etiquetas correspondientes al menú lateral. Genera un menú lateral diferente según si la cuenta es del tipo usuario o administrador, lo cual es determinado obteniendo dicha información de la url de la página a la que se ha accedido.
- **date-manager.js**: sirve de soporte para aquellas páginas que necesitan realizar peticiones en

Algoritmo 5.1: Comprobación de la autenticación

Resultado: Comprobación del token de autentificación

```
1 token ← token del localStorage;
2 settings ← datos de la peticion HTTP;
3 Realizar petición;
4 if not success then
5 |   Eliminar token del almacenamiento local;
6 |   Devolver a la página de Log In;
7 end
```

las que hay involucrado algún tipo de fecha. Proporciona 3 funciones para el tratamiento de fechas:
  - *formatDate*: recibe la fecha actual y le da formato para ser utilizado tanto por la plataforma como por las peticiones HTTP.
  - *obtainWeek*: devuelve el resto de los días de la semana según el día que recibe, es decir, se obtiene la semana (de lunes a viernes) del día introducido.
  - *obtainWeekAndMonth*: según una fecha dada obtiene los días de la semana y el mes al que pertenece cada día. Esta función es muy útil combinada con los calendarios mensuales.
- **month-calendar.js**: dota de funcionalidad a los calendarios mensuales de la plataforma. Proporciona los siguientes metodos:
  - *showMonthlyPlan*: carga el plan mensual y mantiene un registro del mes actual que se está mostrando.
  - *previousMonth*: calcula el mes anterior al que se esta mostrando actualmente y lo muestra. Se utiliza en el botón para ir hacia atrás en el calendario.
  - *nextMonth*: calcula el mes siguiente al que se esta mostrando actualmente y lo muestra. Se utiliza en el botón de avanzar en el calendario.
  - *obtainMonthlyPlan*: obtiene el plan mensual para el mes que se está mostrando actualmente. Recorre cada uno de los días que tiene dicho mes y para lanzar una llamada a *getHiitPerDay* y obtener si hay o no un HIIT para cada uno.
  - *getHiitPerDay*: lanza la petición HTTP con el día designado y devuelve el HIIT correspondiente (si lo hay). También lo añade al calendario y le incorpora un tooltip con los ejercicios haciendo uso de la funcion *addExerciseTooltip*.
- **tooltip.js**: tiene definida la función *addExerciseTooltip(id, html)*, que recibe un id de un HIIT y la etiqueta HTML a la que añadir el tooltip. Realiza la petición para obtener los ejercicios del HIIT e inyecta en el HTML el tooltip con los ejercicios recibidos.

En el mismo directorio que residen estos scripts genéricos podemos encontrar también los archivos correspondientes a jQuery, el cual juega un papel fundamental en el resto de ellos. Toda la inyección de HTML se realiza haciendo uso de los selectores de etiquetas (elementos, clases e ids) proporcionados por jQuery así como el uso de AJAX para las peticiones.

Hay que tener en cuenta que todas las páginas que se han definido,están formadas por un mínimo de 3 elementos que, combinados entre sí, dotan de funcionalidad y visualización a la página. Estos elementos consisten en un HTML, un CSS propio y un JS propio. Además, pueden estar combinados con el resto de archivos de estilo y JavaScripts genéricos que se han diseñado.

Para realizar cualquier petición que se tengan que realizar al servidor, se hace uso de la técnica AJAX. Este tipo de peticiones asíncronas HTTP nos permiten utilizar un conjunto de pares clave/valor para configurar nuestra petición.

En el Listado 5.1 podemos ver como se definiría este objeto de configuración, indicándole la url a la que realizar la petición, el método (GET, POST, PUT, DELETE, PATCH, etc.) y las cabeceras (de

acceso, de autentificación, etc.). En el caso de nuestra aplicación se han definido únicamente los métodos POST y GET para realizar las peticiones, así como cabeceras de *Authorization* que requiere de un *Bearer Token* para poder confirmar la petición que se está realizando.

Finalmente, la ejecución de la petición es bastante sencilla. Haciendo uso de jQuery llamamos a Ajax junto con el objeto de configuración. El uso de la llamada *done* nos asegura de mantenernos a la espera hasta que se recibe una respuesta. Esta respuesta es un objeto en formato JSON que nos devolverá, en caso de que se haya realizado correctamente, la información que se ha solicitado o, en caso contrario, una comunicación del error que ha ocurrido.

```
1  var settings = {
2          "url": host + endpoint, //Definido con const-host.js +
               cons-addr.js
3          "method": "GET", //GET o POST
4          "timeout": 0,
5          "headers": {
6              "Authorization": "Bearer " + ↩
                   ↪ localStorage.getItem('token'),
7              "access": "application/json",
8          },
9  };
10 $.ajax(settings).done(function (response) {
11     // Bloque de código a ejecutar tras recibir la respuesta
12 });
```

Listado 5.1: Código empleado para realizar las peticiones asínronas HTTP al servidor

Ahora que conocemos los diseños, funcionalidades genéricas y la estructura del Cliente, a continuación nos centraremos en el resto de elementos que lo componen.

5.2.1. Sistema de acceso a la plataforma

El sistema de acceso a la plataforma consiste en dos páginas web, la página de log in y la de registro (actualmente deshabilitada para evitar el registro de usuarios sin permiso). Ambas consisten en formularios sencillos con un botón que sirve para enviar la información introducida y realizar las peticiones necesarias.

A continuación, vamos a revisar por separado el funcionamiento de cada una de estas páginas.

Login

Es la primera pantalla que va a mostrarse cuando hacemos uso de la plataforma. Consta de un panel con un formulario que solicita dos campos de información: el nombre de usuario y la contraseña. Al hacer uso del botón de Log In los datos dentro de los campos son obtenidos a través de jQuery y son incorporados a un JSON que va a ser utilizado como parámetro para la petición HTTP. Una vez la petición se ha procesado correctamente, un token JWT será devuelto y guardado en el almacenamiento local del navegador (ver Listado 5.2). Este token nos va permitir mantener el inicio de sesión durante un tiempo y realizar las peticiones que sean oportunas.

Registro

Esta pantalla contiene un formulario de registro con tres campos. Los campos necesarios para crear una cuenta son el usuario y la contraseña (más el campo de repetir contraseña como medida de

```
$.ajax(settings).done(function (response) {
    if (response.hasOwnProperty("success"))
    if (response.success == 0) {
        console.error("[status " + response.status + "] " + ↩
            ↪ response.message);
        $("#fail-alert").html(response.message);
        $("#fail-alert").css("display", "block");
      } else {
        $("#fail-alert").css("display", "none");

        localStorage.setItem('token', response.token);
        localStorage.setItem('user', ↩
            ↪ JSON.stringify(response.userInfo));

        if (response.roles.name == "user") window.location.href ↩
            ↪ = "../home/user/home.html";
        else if (response.roles.name == "administrator") ↩
            ↪ window.location.href = "../home/admin/home.html";
        else {
          $("#fail-alert").html("Error");
          $("#fail-alert").css("display", "block");
        }
    }
 });
```

Listado 5.2: Extracto del código fuente de la clase login.js: petición de comprobación del login

seguridad). El sistema se encargará de asegurar que los campos han sido rellenados y no están vacíos, y comprobará también que las contraseñas que se han introducido coinciden (ver Listado 5.3). Una vez estas evaluaciones han sido superadas, se enviará la información mediante una petición al servidor, el cual se encargará de realizar el registro de forma satisfactoria.

```
function checkRegister() {
    if ($('#user').val() != '' && $('#password').val() != '' && ↩
        ↪ $('#repeat_password').val() != '') {
        if ($('#password').val() == ↩
            ↪ $('#repeat_password').val()) {
            Register($('#user').val(), $('#password').val())
        } else {
            $('#fail-alert').html("Las contraseñas deben de ↩
                ↪ coincidir.");
            $('#fail-alert').css("display", "block");
        }
    } else {
        $('#fail-alert').html("Los campos no pueden estar ↩
            ↪ vacíos.");
        $('#fail-alert').css("display", "block");
    }
}
```

Listado 5.3: Extracto del código fuente de register.js: comprobación de los campos

5.2.2. Plataforma para el Usuario

Contiene todas aquellas funcionalidades que van a estar al servicio del usuario. Cuenta con una página principal donde podrá observar información relevante con respecto a su estado en el programa MOVI-HIIT, una página de Calendario para poder consultar la programación de forma mensual, una

Clasificación para ver su situación, una Recompensa y, por último, pero la más importante, la página del HIIT Diario.

Con los siguientes puntos nos vamos a centrar en definir como están construidas las distintas páginas web y las funcionalidades que incorporan cada una de ellas.

HIIT del Día

Esta página web contiene el módulo de representación 3D que se corresponde con la escena de los HIITS. Para integrar UnityWebGL al navegador se ha generado una compilación WebGL desde el editor de Unity. Existen varias maneras de realizar las compilaciones según el método de compresión que le indiquemos. La opción utilizada es gzip, pero también es posible utilizar brotli o ninguna. Otra opción que se ha utilizado a la hora de generar la Build es la descompresión por retroceso o decompression fallback. Esto nos ha permitido asegurarnos que el navegador descomprima el contenido, ya que integra un descompresor de JavaScript en la compilación. Es el motivo por la que nuestros archivos contienen la extension final *.unityweb*. Esta opción también es utilizada cuando no es posible configurar el servidor para realizar la descompresión.

Las extensiones de los archivos que necesitamos, y que han sido generados por esta compilación, son los siguientes:

- **.loader.js**: es el archivo más importante. Contiene todo el código JavaScrit que se encarga de cargar el contenido de Unity en la web.
- **.data.unityweb**: contiene toda la información sobre los assets y las escenas.
- **.framework.js.unityweb**: máquina virtual encargada de interpretar y ejecutar el código JavaScript en el navegador. También contiene los plugins necesarios.
- **.wasm.unityweb**: contiene el código binario WebAssembly. WebAssembly es también conocido como Wasm y está designado para tener un tamaño pequeño, poco tiempo de carga, así como uso de memoria más eficiente.

Estos archivos han sido utilizados en un script para cargar el contenido en un canvas. Como podemos observar en el Listado 5.4, se genera un JSON que se utiliza posteriormente para crear una instancia de Unity en el canvas seleccionado. En nuestro caso cargamos la escena en la etiqueta con el id: unityContainer.

Una vez se ha generado la instancia de Unity podremos utilizarla para interactuar y comunicarnos con la aplicación de Unity WebGL. Más adelante, durante la explicación sobre el módulo de representación 3D, daremos más detalles sobre como funciona la comunicación entre el navegador y Unity.

Para indicarle al módulo qué actividad física hay que representar se hace uso de la función *ObtainDailyHiit*. Si observamos el flujo de eventos de la Figura 5.3 se puede comprobar que Unity avisa a la página web de que está listo y esta se encarga de realizar la petición de la información del HIIT diario.

Una vez esté finalizado el HIIT, se hará una petición para actualizar la cantidad de puntos que tiene el aula correspondiente. Cada uno de los HIITs que se realicen añadirán un punto al sistema.

Home

En esta página de inicio encontramos 3 elementos que hacen uso del sistema de peticiones para actualizar su contenido de manera dinámica (ver Figura B.10). Hace uso de las funciones que ofrecen *date-manager.js* y *tooltip.js*, junto con las peticiones HTTP de su propio JavaScript.

```
<head>
    ...
    <!-- Unity Scripts -->
    <script src="Build/2021-7-19.loader.js"></script>
    ...
</head>

<body>
    ...
    <canvas id="unityContainer"></canvas>
    <script>
        canvas = document.querySelector("#unityContainer");
        config = {
            dataUrl: "Build/2021-7-19.data.unityweb",
            frameworkUrl: "Build/2021-7-19.framework.js.unityweb",
            codeUrl: "Build/2021-7-19.wasm.unityweb",
            streamingAssetsUrl: "StreamingAssets",
            companyName: "Furious Koalas S.L.",
            productName: "MOVI-HIIT",
            productVersion: "1",
        }

        onload = () => {
            createUnityInstance(canvas, config, (progress) => {}).then((unityInstance) => {
             parent.GlobalUnityInstance = unityInstance;
             }).catch((message) => {alert(message);});
        };

        function Loaded() { ObtainDailyHiit(); }
        function ToggleFullScreen(number){ GlobalUnityInstance.SetFullscreen(number); }
        function IncreasePoints(){ addPointReward(); }
    </script>
    ...
</body>
```

Listado 5.4: Código correspondiente con la etiqueta HTML de tipo canvas y el script para cargar UnityWebGL

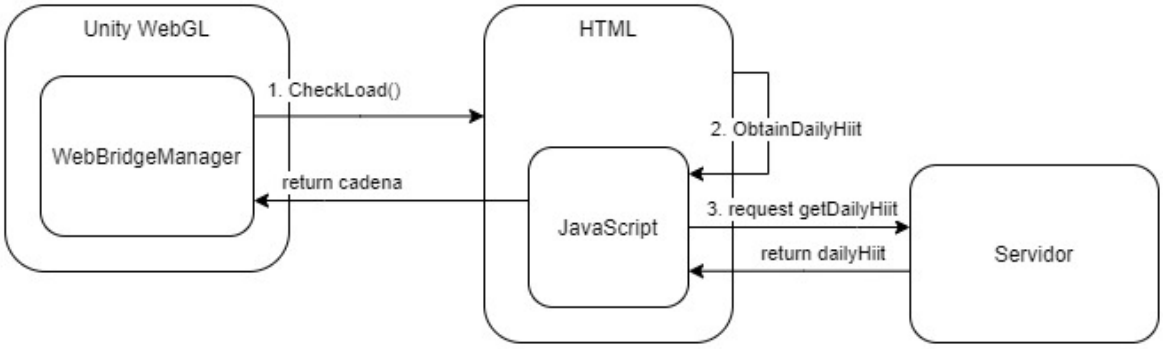

Figura 5.3: Flujo de comunicación entre UnityWebGL, la página Web y el servidor para obtener el HIIT y comenzar su ejecución

- *Información HIIT del día*: se realiza una petición para obtener la información del HIIT del día. Haciendo inyección de código HTML con jQuery se añade el nombre del HIIT al panel junto con los nombres de los ejercicios.
- *Planificación semanal*: haciendo uso de *date-manager.js* obtenemos los días de la semana correspondientes a la semana actual y son enviados en una petición al endpoint *hiitInterval*. Esto nos devolverá cada uno de los HIITs de cada día de la semana sobre los que se utilizará la la función de *tooltip.js* para añadir el tooltip con los ejercicios.
- *Ranking top 3*: haciendo uso del endpoint *getRankingMySchool* obtenemos las 3 aulas con más puntos del colegio al que pertenece el usuario e inyectamos las etiquetas necesarias para mostrar los paneles de la clasificación.

Cada uno de estos paneles que contiene información resumida del sistema. Ellos disponen de un botón para acceder a las páginas correspondientes (Hiit del Día, Calendario y Ranking).

Calendario

Para mostrar la planificación mensual de los HIITs se ha realizado un diseño similar al de un calendario, todo ello desde cero (siguiendo siempre el diseño aportado por la diseñadora gráfica). La idea era implementar un calendario que muestre los días de Lunes a Viernes y que permita desplazarse entre los meses para poder revisar la planificación completa. Para ello se ha hecho uso combinado de los scripts *month-calendar.js*, *data-manager.js* y la hoja de estilo *month-calendar.css*.

Si observamos el código extraído de *month-calendar.js* para la obtención del plan mensual, podemos comprobar que utiliza la función de *data-manager.js* para dar formato a las fechas y poder ser utilizadas por el calendario. Una vez tenemos las fechas preparadas, se itera cada uno de las etiquetas con el id calendar-body. Cada una de estas etiquetas «hijas» se corresponde con una fila, es decir, una semana en el calendario. Cada fila contiene 5 etiquetas correspondientes con cada uno de los días de la semana (L, M, X, J y V). Iterando de nuevo sobre estas etiquetas hijas se obtiene el HIIT para cada uno de los días, lanzando las petición a través de otro método definido en *month-calendar.js*.

Ranking

En esta página se muestra un Ranking de todas las clases que pertenecen a una misma escuela. Esto significa que un participante de un colegio no podrá ver las aulas de otros colegios, es decir, solo las del suyo propio. Esta restricción está implementada en la consulta a la base de datos con el endpoint *getRankingMySchool*.

A la hora de realizar la petición se realiza un registro de cuales son los 3 primeros elementos

recibidos en la respuesta a la misma, para así incorporarle a su etiqueta la clase *top-rank*, definida en la hoja de estilos genérica *content-side.css*. El panel donde está contenida el aula y la puntuación tomará color amarillo debido a que se encuentra en el top 3 del Ranking. En caso contrario, la clase que se le incorporará será *rank*, que define un color azul para el fondo del panel.

La función tomará de la respuesta de la petición el aula al que pertenece y la cantidad de puntos que tienen (incorporándolos a la etiqueta que va a ser inyectada). Se mantendrá también un índice sobre la posición en la que se encuentra esa aula, que irá incrementando al iterar sobre la información recibida (las clases vienen ordenadas de mayor a menor puntuación).

Recompensas

En esta página web se muestra un panel con todas las recompensas que hay en la base de datos. Están formadas por una imagen representativa de la recompensa y el nombre de la propia recompensa. Si una recompensa ha sido desbloqueada aparecerá a color y en caso contrario aparecerá en gris.

Para visualizar las recompensa se hace una petición a la base de datos para obtener las recompensas existentes, y por cada una de ellas se genera un panel que contiene la imagen de la recompensa (obtenida del directorio de recursos mediante el nombre de la recompensa) junto con su nombre.

Dentro de estas etiquetas también se almacena de forma invisible el id de la recompensa para poder ser utilizado a la hora de mostrar su información. Esto consiste en que, al seleccionar una de las recompensas, ese id será utilizado para realizar una petición HTTP y obtener la descripción correspondiente.

Estas recompensas son mensuales, y van a ser desbloqueadas alcanzando una puntuación correspondiente al 80 % de la puntuación máxima obtenible al realizar todos los HIITs programados para dicho mes.

Finalmente también se lleva un registro de los puntos totales que se han obtenido.

5.2.3. Plataforma para el Administrador

Contiene todas aquellas funcionalidades que van a estar al servicio del usuario. Al igual que en la plataforma para el Usuario, cuenta con una página principal, una página de Calendario y una de Clasificación. También se incorpora una página de Ejercicios para la gestión de los HIITs y visualización de las animaciones de los ejercicios y por último, una página de Participantes para el control de los usuarios.

Centrémonos en definir como están construidas las distintas páginas web y las funcionalidades que incorporan cada una de ellas.

Home

Incorpora funcionalidades similares a la página Home de la plataforma de los Usuarios. Incluye una planificación de los HIITs programados para la semana y un Ranking. A diferencia de la plataforma de Usuarios, en el Ranking aparecerán las 3 primeras aulas con más puntos del total de todos los colegios que participan. Se incluye el nombre del colegio, el aula, los puntos y el rango que tienen. Y finalmente un registro de los 3 ultimos HIITs que han sido añadidos al sistema (mostrando sus nombres y la fecha de creación).

Calendario

La funcionalidad de esta página es exactamente la misma que la definida en la plataforma para el Usuario. Consiste en un calendario mensual que permite visualizar la programación mensual. Si se desea volver a ver el funcionamiento de la misma, se recomienda ir al apartado Calendario de la sección 5.2.2

Ejercicios

Esta página facilita las funcionalidades para la gestión de los HIITs y la visualización de los ejercicios haciendo uso del módulo de representación 3D.

En primer lugar, el sistema de gestión de HIITs está representado de forma visual por un listado con paneles que contienen el nombre del HIIT y los ejercicios que le corresponden. Este panel cuenta también con el id oculto del HIIT y botones para la edición y borrado de los mismos.

Incorpora también un botón para la creación de un nuevo HIIT. En el caso de que queramos añadir uno nuevo o editar uno ya existente, se abrirá un formulario para rellenar la información correspondiente con el HIIT (ver Figuras B.17 y B.18). Esto es, al menos, introducir un nombre y una descripción, seleccionar un método de un selector (recibido por la base de datos) y 4 ejercicios (también mostrados en selectores y recibidos desde la base de datos). En el Listado 5.5 podemos ver que la información es obtenida del formulario, parseada y utilizada en la petición al endpoint *createHIIT* .

```
function saveNewHiit() {
  var [id, name, description, method, exercises] = getFormData();
  var settings = {
    "url": host + createHiit,
    "method": "POST",
    "timeout": 0,
    "headers": {
      "Authorization": "Bearer " + localStorage.getItem('token'),
      "access": "application/json"
    },
    "data": JSON.stringify({
      "hiitname": name,
      "hiitdescription": description,
      "methodid": method,
      "hiitExercises": exercises
    }),
  };
  $.ajax(settings).done(function (response) { ... });
}
```

Listado 5.5: Función para guardar un nuevo HIIT

De forma similar, al seleccionar la opción de editar un HIIT, se nos abrirá el mismo formulario que antes con los campos rellenos con la información del HIIT. Al enviar los cambios realizados se usa la función *saveEditredHIIT* que hace lo mismo que la función del Listado 5.5. Pero en este caso al existir el id en la base de datos, se procederá a sustituir el resto de campos por los nuevos introducidos.

La última funcionalidad de la que se dispone es la de eliminar HIITs. Para ello se ha incorporado un botón en forma de X en cada uno de los paneles de los HIITs del listado (ver Figura B.16). Por medidas de seguridad se ha creado un formulario de confirmación para evitar descuidos.

En el caso del módulo 3D aquí utilizado consiste en una compilación UnityWebGL de la escena

correspondiente con la visualización de ejercicios. Esta escena tiene una complejidad mucho menor que su contraparte en la plataforma de los Usuarios. Por lo tanto, este componente será explicado brevemente, dejando el apartado 5.2.4 completamente para la explicación de la arquitectura utilizada para el proyecto Unity y el funcionamiento de la reproducción del HIIT diario.

Esta escena contiene dos Avatares en la misma posición, ocultando a uno de ellos. Una interfaz con dos botones nos permite seleccionar cual de los dos Avatares queremos ver para reproducir la animación e incluye la posibilidad de rotar alrededor del avatar para ver la animación desde distintos ángulos. Esta funcionalidad se ha incorporado para poder revisar las animaciones de los ejercicios de forma precisa para solicitar, si es necesario, algún cambio o ajuste.

Para enviar el ejercicio que tienen que reproducir se hace uso de una llamada a la Instancia generada de Unity, de la misma manera que se ha mostrado durante la sección de HIIT del Día en la plataforma del Usuario. Al seleccionar un ejercicio, su nombre es enviado a través de un mensaje al objeto de la escena llamado AVATAR para que así los Avatares consulten la animación a reproducir.

Ranking

Funciona de manera muy similar a la página Ranking definida para la plataforma del usuario con la peculiaridad de que esta vez se recibe un Ranking General con la clasificación de cada una de las aulas de todos los colegios participantes (ver Figura B.21). Esto se consigue usando el endpoint *getRanking* en lugar de *getRankingMySchool*.

También contiene un panel que permite la selección de un colegio para mostrar únicamente el Ranking de dicho colegio. Se realiza una petición al endpoint *getSchools* para obtener todos los colegios que participan y poder así incorporarlos al selector. Cada escuela es añadida como una etiqueta *option* a la etiqueta del selector y se le incorpora como valor el id del colegio y como contenido HTML su nombre.

Una vez hagamos selección de uno de los colegios listados en el selector, se toma el valor incluido en su etiqueta y es enviado en una petición al endpoint *getSchoolRank* con el id del colegio añadido al final del url. La respuesta de esta petición es una clasificación de las aulas que es traducida a la estructura de etiquetas que se emplea para representar los rankings e inyectada en el panel correspondiente para su visualización.

Participantes

Finalmente, la página de los participantes permite consultar las clases que participan cada uno de los colegios. Para ello vamos a hacer uso de una petición al endpoint *getSchools* para añadir los colegios a un selector, al igual que se ha explicado anteriormente en el Ranking.

Una vez se ha seleccionado uno de los colegios se mostrarán las aulas pertenecientes a dicho colegio (ver Figura B.22). Esto se va a conseguir haciendo uso del endpoint *getRankingSchool* junto con el id del colegio. Se hace uso de este endpoint ya que contiene todas las aulas que le pertenecen (aunque tengan 0 puntos).

Al seleccionar uno de los colegios que nos aparece, se mostrará en el panel de al lado las recompensas. En este caso se utiliza una petición al endpoint *getUserRewards* para obtener las recompensas desbloqueadas por el aula que se ha seleccionado.

5.2.4. Módulo de representación 3D

En este apartado vamos a describir en detalle la arquitectura del módulo de representación 3D. Este elemento ha sido por completo desarrollado utilizando Unity 2020 y ha sido la base principal del proyecto.

Este módulo se presenta en dos sectores de la plataforma web: en la página web de Ejercicios (administrador) y en el HIIT del día (usuario). El comportamiento del módulo en cada una de ellas el módulo es diferente, es decir, existen dos escenas diferentes en Unity y cada una de ellas se utiliza en una u otra página.

Cabe recordar que en el apartado de Ejercicios de la sección 5.2.3 se hace una pequeña introducción a la escena para la visualización previa de los ejercicios. A continuación profundizaremos sobre la escena principal de la plataforma, la escena para la representación de los HIITs.

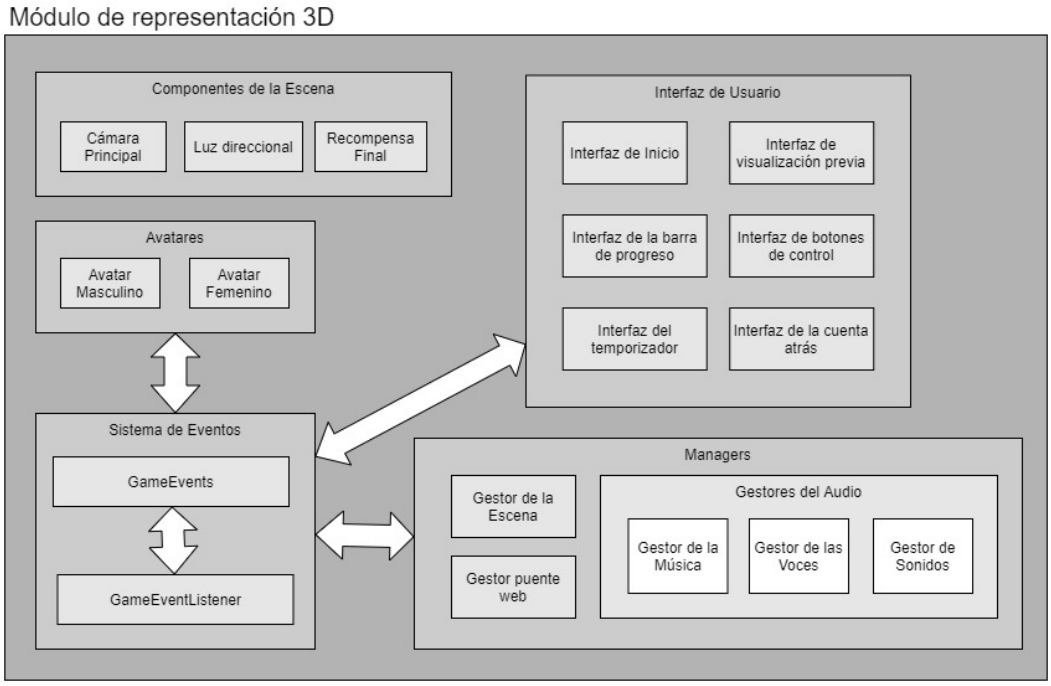

Figura 5.4: Componentes del módulo de representación 3D.

Para comenzar daremos una vista general de los sistemas y componentes que forman el módulo. Posteriormente, se describe con mayor detalle cada uno de ellos. En la Figura 5.4 podemos distinguir una separación entre los componentes de la escena según su finalidad:

- **Avatares**: formados por el avatar femenino y masculino. Se encargan de reproducir las animaciones de los ejercicios en cada una de las fases establecidas en el HIIT.
- **Componentes de la Escena**: elementos que tienen una representación tridimensional en la escena (a excepción de los Avatares). Está formado por la cámara principal, las luces y la recompensa final.
- **Interfaz de Usuario**: formado por cada uno de los elementos o subsistemas que muestran información al usuario en la pantalla. Guían al usuario durante la ejecución de la actividad física y delimitan de manera visual las diferentes fases.
- **Sistema de Eventos**: componente principal de la escena. Combina el uso de ScriptableObjects, el cual se explicará más adelante, con MonoBehaviours[2] para crear un sistema de eventos que

[2]La clase MonoBehaviour es la clase de la que derivan todos los scripts de Unity por defecto. Permite ser añadido como un Componente a los GameObjects en el editor. Esta clase provee de funciones como *Update*, *Start* u *OnEnable*, y permite el uso de rutinas asíncronas para realizar esperas de tiempo.

permita la comunicación entre el resto de componentes.
- **Managers o Gestores**: encargados de gestionar las fases del HIIT, la comunicación con la plataforma web y la reproducción de música, voces y sonidos.

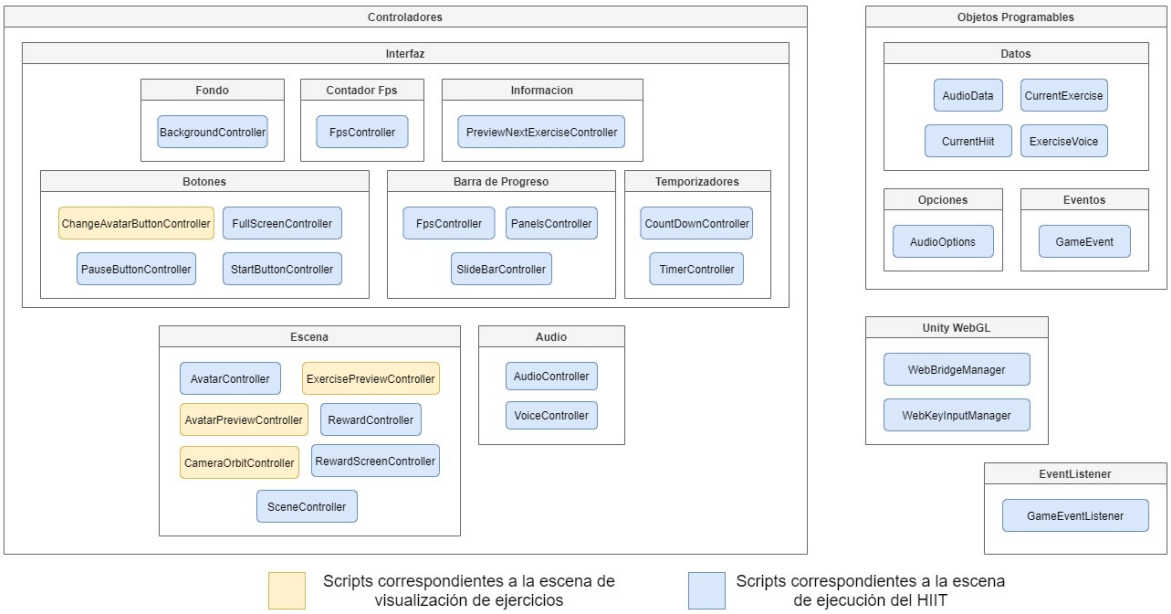

Figura 5.5: Clasificación y organización de los scripts dentro del proyecto.

Comunicación entre Unity WebGL y JavaScript

La comunicación entre Unity y JavaScript es algo indispensable para este proyecto, ya que es necesaria la información almacenada en la base de datos para realizar de forma rigurosa la planificación de la actividad física. Por ello es necesario utilizar los mecanismos de comunicación que nos ofrece Unity WebGL para interactuar con los scripts del navegador. En el caso del presente proyecto es necesario que la plataforma web notifique el HIIT con los ejercicios correspondientes al módulo de representación 3D para reproducirlos y, al acabar la actividad, Unity WebGL notifique a la plataforma web que esta ha sido finalizada.

Para ello se ha aplicado la recomendación que realiza Unity en su Manual de Usuario [13] de utilizar archivos con código JavaScript bajo la extensión *.jslib* en un directorio dentro del directorio de Assets llamado *Plugins*. Los archivos con esta extensión contienen las funciones que son utilizadas directamente en nuestros scripts de Unity para realizar esta comunicación.

Esta comunicación es bidireccional, por tanto podemos llamar a funciones de JavaScript desde Unity (como ya hemos visto), y también podemos llamar a funciones de Unity desde los scripts del navegador. Para conseguir esto hacemos uso de la llamada:

```
unityInstance.SendMessage(objectName, methodName, value);
```

ScriptableObjects

Durante la conferencia de Unity Austin de 2017, Ryan Hipple describe 3 pilares fundamentales sobre los que basarse a la hora de crear arquitecturas para juegos [5]. Cualquier sistema debería estar fundamentado en estos pilares de la ingeniería: Modularidad, Editabilidad y Debuggeabilidad. Es posible conseguir alcanzar una arquitectura de sistema basada en estos pilares haciendo uso de una herramienta que proporciona Unity, los conocidos como *ScriptableObjects*.

```
mergeInto(LibraryManager.library, {
  CheckLoad: function(){
    Loaded();
  },

  SetFullScreen: function(number){
    ToggleFullScreen(number);
  },

  finishedActivity: function()
  {
    IncreasePoints();
  }
});
```

Listado 5.6: Ejemplo de código fuente perteneciente a javascript.jslib

ScriptableObject es una clase serializable de Unity que permite el almacenamiento de datos compartidos de forma independiente a las instancias de los scripts. Estas clases sirven como plantillas para crear objetos en forma de assets dentro del proyecto destinados a contener datos. Estos scripts no necesitan estar adjuntos a GameObjects de la escena.

Este tipo de herramientas supone diversas ventajas a la hora de diseñar la arquitectura de nuestro proyecto. Basándonos en estos pilares fundamentales que hemos comentado anteriormente, podemos destacar las principales ventajas que supone el uso de los ScriptableObjects en un proyecto Unity:

- **Modularidad más completa**: las relaciones de dependencias que podemos establecer entre los scripts pasan a ser inyecciones de dependencias desde el editor. Esto significa que vamos a utilizar objetos persistentes creados dentro del propio proyecto para establecer referencias. Además, suponen una reducción de los problemas relacionados con las referencias, ya que no necesitan derivar de la clase MonoBehaviour y, por lo tanto, no necesitan ser añadidas a un GameObject. Esto elimina las posibilidades de error por referencias nulas debido a objetos que se hayan eliminado u ocultado.
- **Edición de valores en tiempo de ejecución**: en Unity, cuando estamos ejecutando el juego desde el propio editor, podemos seleccionar cualquiera de los objetos de la escena haciendo uso del panel de jerarquía. El inspector nos va a mostrar toda aquella información contenida dentro del propio GameObject. Si cualquier valor dentro del objeto es modificado durante el tiempo de ejecución, éste volverá a su valor original antes de haber inicializado la ejecución.
- **Mayor facilidad en la trazabilidad de errores**: al mantener una separación más completa entre los diversos sistemas que conforman el proyecto, hacemos que sea más sencilla la trazabilidad de errores. Esto supone una gran ventaja cuanto mayor sea la cantidad de funcionalidades que tenga nuestro proyecto.

Una vez explicado el uso de ScriptableObjects dentro del proyecto, vamos a hacer un repaso de cómo se ha aplicado esta herramienta para la creación de un sistema de Eventos.

Sistema de Eventos

En uno de los prototipos, del que hablaremos en el apartado 6 con mayor profundidad, se diseñó un sistema basado en eventos haciendo uso de una clase singleton. Esta clase se encargaba de subscribir los objetos a eventos y de avisarlos cuando un evento se lanzaba. Este tipo de arquitectura de eventos hace uso del Patrón de Diseño «Publicador-Subscriptor».

Si lo extrapolamos al uso de ScriptableObjects eliminamos por completo el uso de esta clase singleton, ahorrando así los posibles problemas de dependencia que pudiese ocasionar. La idea de

realizar un diseño de un sistema de eventos basado en ScriptableObjects reside en la modularización del código y desacoplamiento, permitiendo el envío de mensajes entre los distintos sistemas que conforman el proyecto. Ninguno de estos sistemas tiene por que saber de la existencia del resto, simplemente tienen que realizar una acción como respuesta a un mensaje recibido.

Este sistema de eventos está compuesto por dos partes: la clase *GameEventListener.cs* y la clase *GameEvent.cs*. Podemos ver el funcionamiento de ambas respectivamente en los Listados 5.7 y 5.8. Para poder comprender mejor el funcionamiento de ambos, a continuación entramos en mayor detalle sobre los dos scripts que componen este sistema.

Game Event Listener

Este script consiste en un clase que hereda de MonoBehaviour. Esta clase permite establecer diferentes comportamientos, permitiéndonos poder acoplarla como Componente a cualquier GameObject de la escena en Unity. Como podemos ver en el Listado 5.7 está formado por los siguientes elementos:

- Campos:
  - **Event**: es un objeto del tipo GameEvent y consiste en el Evento sobre el cual el script estará a la escucha para recibir alguna notificación. Estos Eventos habrán sido creados previamente como assets del proyecto. En el Editor podremos realizar inyección de dependencia en una casilla al lado de este campo. Esto nos permitirá arrastrar el Evento correspondiente hacia esa casilla y realizar la referencia a ese objeto.
  - **Response**: son eventos de Unity que se ejecutan en respuesta al alzamiento del Evento que se está escuchando. Se pueden añadir tantos Eventos de Unity como sean necesarios desde el Inspector.
- Métodos:
  - **OnEnable**: cuando el objeto pasa a estar activo o habilitado, se ejecuta este comportamiento. El objeto se registra como oyente en el Evento referenciado, a la espera de recibir cualquier cambio.
  - **OnDisable**: funciona de forma similar al método anterior pero en este caso se ejecuta el comportamiento cuando se deshabilita el objeto o pasa a estar desactivado. El objeto deja ya de estar a la escucha del Evento al que referencia.
  - **OnEventRaised**: este método entra en juego cuando se produce un levantamiento de un Evento. Como se puede ver en la linea 12 y 13 del Listado 5.8, todos los objetos que estén a la escucha son notificados a través de este método, se invocan como respuesta uno o varios Eventos de Unity según los tengamos definidos en el Inspector del GameObject, en el Editor de Unity.

Este script hace uso uso de la directiva #pragma para indicarle al compilador que desactive la alerta 0649. Esta alerta nos indica que un campo no ha sido inicializado a ningún valor y va a utilizar siempre su valor por defecto. El compilador detecta por lo tanto que no tiene asignado ningún valor. Este valor, como ya hemos comentado con anterioridad, es asignado a través de la inyección de dependencias en el Inspector, por lo tanto se ha decidido suprimir.

Para finalizar con el sistema de eventos, se detalla, a continuación, el funcionamiento de la segunda parte que lo compone: *Game Events*.

Game Event

Aprovechando el potencial de los ScriptableObjects, se crea la clase GameEvent. Son clases que heredan de ScriptableObjects y que contienen un listado de oyentes a los cuales notifican cada vez que un Evento específico es lanzado. Si observamos el Listado 5.8, podemos distinguir los siguientes elementos en la estructura de la clase:

- Cabecera:

```
1
2  using UnityEngine;
3  using UnityEngine.Events;
4
5  public class GameEventListener : MonoBehaviour
6  {
7      #pragma warning disable 0649
8      [SerializeField] private GameEvent Event;
9      [SerializeField] private UnityEvent Response;
10     #pragma warning restore 0649
11
12     private void OnEnable() => Event.RegisterListener(this);
13     private void OnDisable() => Event.UnregisterListener(this);
14
15     public void OnEventRaised() => Response.Invoke();
16 }
```

Listado 5.7: Código fuente de la clase derivada de MonoBehaviour: GameEventListener.cs

    - **CreateAssetMenu**: este atributo permite la creación de instancias de la clase como objetos dentro del directorio de assets. Al otorgarle un menuName y un fileName aparecerán de manera separada en el listado del menú de creación.
- Campos:
    - **listeners**: consiste en una lista de objetos del tipo GameEventListener que van a estar a la escucha de un evento.
- Métodos:
    - **Raise**: se corresponde con el alzamiento/notificación del Evento. La lista de oyentes es recorrida notificando a cada uno de estos de que el Evento ha sido lanzado. Se realiza el recorrido hacia atrás para poder iterar a través de la lista evitando, en el caso de que algún oyente se suprima de la lista (a consecuencia del evento), que no ocurra ninguna excepción *OutOfRange*.
    - **RegisterListener**: se registra un GameEventListener como oyente de un Evento.
    - **UnregisterListener**: se suprime un GameEventListener de la lista de oyentes de un Evento.

Estas clases son utilizadas como plantilla para crear todos los Eventos del propio sistema de Eventos bajo el directorio Assets >Scriptable Objects >Game Events. Como podemos ver en la Figura 5.6, se encuentran organizados según la finalidad de los eventos. Veamos una breve descripción de los eventos creados:

- **Web Events**: eventos previos a la ejecución del HIIT.
    - *WebBridgeReady*: evento que determina cuando se ha finalizado la comunicación con la web, y se ha recibido la información del HIIT a realizar.
    - *StartGame*: cuando el botón de inicio es presionado salta este evento indicando al resto de sistemas que va a comenzar la actividad.
- **Preparation Events**: eventos de la fase de preparación.
    - *PreparationStarted*: este evento es lanzado al comenzar esta fase.
    - *ExerciseExplanation*: evento lanzado al comenzar con las explicaciones de los ejercicios.
    - *StopExerciseExplanation*: indica que las explicaciones de los ejercicios han finalizado.
    - *PreparationFinished*: cuando la fase de preparación finaliza este evento avisa al resto de sistemas.
- **Exercise Events**: eventos para la fase de la actividad física.
    - *ExercisesStarted*: cuando da comienzo esta fase este evento es lanzado.
    - *HighIntensityInterval*: evento que indica la entrada al intervalo de alta intensidad.

```
1
2  using UnityEngine;
3  using System.Collections.Generic;
4
5  [CreateAssetMenu(menuName = "Scriptable Object/Game Event", ↩
       ↪ fileName = "New Game Event")]
6  public class GameEvent : ScriptableObject
7  {
8      private List<GameEventListener> listeners = new ↩
           ↪ List<GameEventListener>();
9
10     public void Raise()
11     {
12         for(int i = listeners.Count - 1; i >= 0; i--)
13             listeners[i].OnEventRaised();
14     }
15
16     public void RegisterListener(GameEventListener listener) => ↩
           ↪ listeners.Add(listener);
17     public void UnregisterListener(GameEventListener listener) ↩
           ↪ => listeners.Remove(listener);
18 }
```

Listado 5.8: Código fuente de la clase derivada de Scriptable Object: GameEvent.cs

  - *LowIntensityInterval*: evento que indica la entrada al intervalo de baja intensidad.
  - *ExercisesFinished*: este evento es lanzado al finalizar la fase de los ejercicios.
- **Calm Events**: eventos para la fase de vuelta a la calma.
  - *CalmStarted*: al comenzar esta fase se lanza este evento.
  - *CalmFinished*: al finalizar la fase se lanza este evento.
- **Final Events**: eventos para la fase de la recompensa.
  - *OpenChest*: evento para la apertura del cofre de la recompensa.
  - *ShowReward*: evento para mostrar la recompensa final.
- **Interface Events**: eventos de la interfaz de usuario.
  - *CountdownFinished*: cuando la cuenta atrás que da inicio a la fase de los ejercicios termina, se lanza este evento.
  - *TimerFinished*: cuanto el contador de segundos llega a 0 este evento es lanzado.

Estos GameEvents se utilizan junto con GameEventListener como Componentes en los diferentes GameObjets para establecer una comunicación entre ellos y guiar la ejecución. En los siguientes apartados indicaremos cómo se lanzan los eventos por los distintos componentes de la escena.

Avatares

Ambos avatares, tanto el masculino como el femenino, disponen de los mismos Componentes asociados a ellos. Por lo tanto, la explicación de uno de ellos servirá para definir cómo están formados ambos.

El primer componente que utilizan es un *AnimatorController* que permite la creación de una máquina de estados con diferentes animaciones (ver Figura 5.7). Cada uno de los estados intermedios corresponde con una animación que ha sido extraída de la línea de tiempo del propio avatar, el cual contiene los distintos fotogramas con las animaciones que fueron grabadas utilizando el sistema de captura Xsens (ver Figura 5.8).

El componente que va a regir el comportamiento principal de los avatares es *AvatarController.cs* (véase Listado 5.9). Se encarga de hacer uso del AnimatorController para ir haciendo transiciones entre

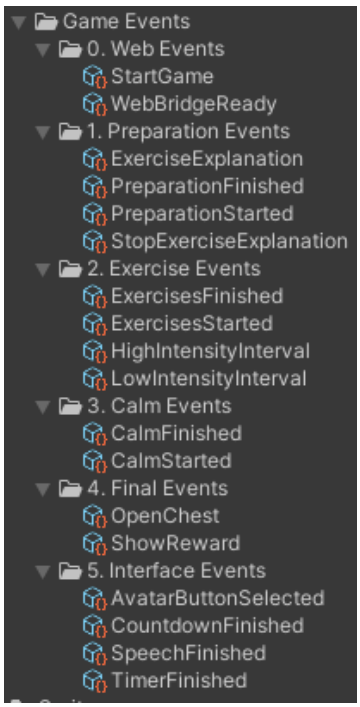

Figura 5.6: Todos los eventos creados con la clase GameEvent

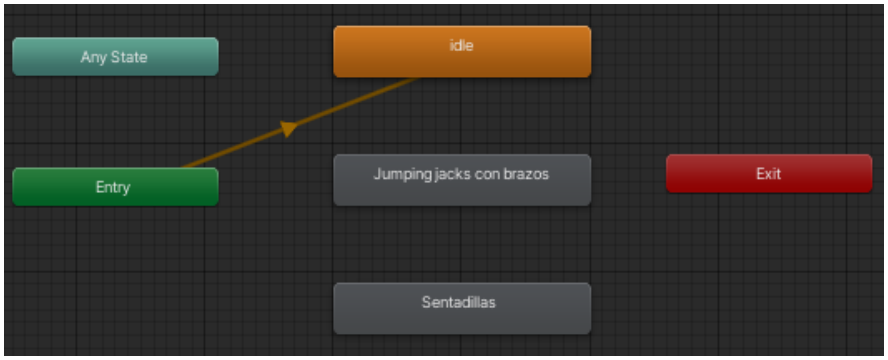

Figura 5.7: Máquina de estados de las animaciones de los avatares

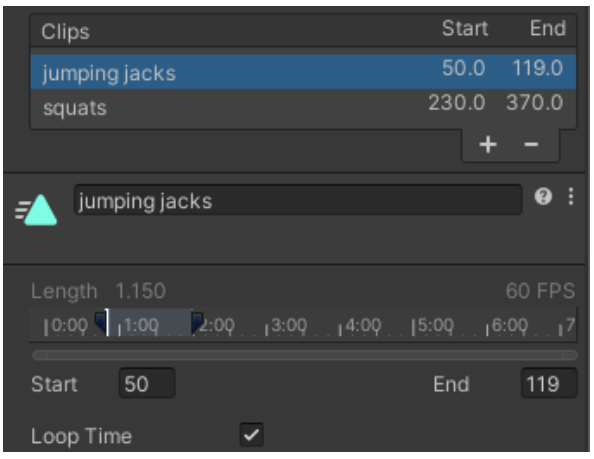

Figura 5.8: Línea temporal que contiene los fotogramas con las distintas animaciones

las distintas animaciones de los ejercicios que toquen realizar en un HIIT. Hace uso de referencias a los ScriptableObjects: *CurrentHiit* y *CurrentExercise*. Por un lado, CurrentHiit mantiene una lista de strings con los ejercicios que hay que realizar, aparte de los valores numéricos para la duración de la

alta y la baja intensidad. Por otro lado, CurrentExercise mantiene el nombre del ejercicio que se está ejecutando.

```
1  public class AvatarController : MonoBehaviour
2  {
3      [SerializeField] private CurrentHiit currentHiit;
4      [SerializeField] private CurrentExercise currentExercise;
5
6      public void PlayAnimation() => ↩
           ↪ StartCoroutine(PlayAnimationIEnum());
7      public void PlayExerciseExplanation() => ↩
           ↪ StartCoroutine(PlayExerciseExplanationIEnum());
8      public void TransitionToIdle() => animator.Play("idle");
9
10     public IEnumerator PlayAnimationIEnum()
11     {
12         yield return new WaitForSeconds(delay);
13         animator.speed = 1f;
14         animator.Play(currentExercise.Name);
15     }
16
17     public IEnumerator PlayExerciseExplanationIEnum()
18     {
19         yield return new WaitForSeconds(delay);
20         animator.speed = .75f;
21         animator.Play(currentHiit.Exercises[i++]);
22     }
23 }
```

Listado 5.9: Código de la clase AvatarController.cs

Los últimos Componentes que forman parte de los avatares son del tipo GameEventListener que servirá para cada uno de los Eventos a los que tiene que responder en el caso de que ocurran (ver Figura 5.9). A continuación vamos a listar los diferentes eventos a los que se encuentra en escucha y las respuestas que proporciona cuando se alcanza el Evento:

- **Evento 1**: al comenzar un intervalo de alta intensidad, se ejecuta la animación del ejercicio actual.
  - Evento: HighIntensityInterval
  - Respuesta: AvatarController.PlayAnimation()
- **Evento 2**: al comenzar un intervalo de baja intensidad, se vuelve a la animación de reposo (idle).
  - Evento: LowIntensityInterval
  - Respuesta: AvatarController.PlayAnimation()
- **Evento 3**: al comenzar las explicaciones de los ejercicios, se reproducen las animaciones correspondientes.
  - Evento: ExerciseExplanation
  - Respuesta: AvatarController.PlayExerciseExplanation()
- **Evento 4**: al finalizar las explicaciones de los ejercicios, se vuelve a la animación de idle.
  - Evento: StopExerciseExplanation
  - Respuesta: AvatarController.TransitionToIdle()

Componentes de la Escena

Los objetos que pertenecen a los componentes de la escena son: la *cámara principal*, *la luz direccional* y *la recompensa final*. La recompensa final hace uso de un GameEventListener a la escucha del

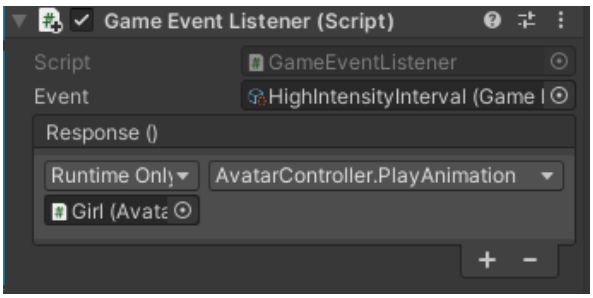

Figura 5.9: Ejemplo de combinación de un GameEvent con GameEventListener como Componente en el GameObject

GameObject *ShowReward* para activar un cofre que realiza una animación de apertura junto con un sistema de partículas que imita una explosión de confetti. Finaliza mostrando el corazón que simboliza el punto obtenido por la realización del HIIT. La cámara utiliza una proyección ortográfica para renderizar los elementos de la escena y la luz direccional está configurada para conseguir sombras en tiempo real.

Managers

- **WebBridgeManager**: este objeto es el encargado de la comunicación directa con la plataforma web. Al comenzar la ejecución envía un mensaje a la web solicitando la información del HIIT a realizar. Cuando recibe la información correspondiente se encarga de serializar los datos para poder ser utilizados y lanza el evento *WebBridgeReady*. También genera los datos que va a almacenar el ScriptableObject *currentHiit* (ver Listado 5.10).

```
1         public void GenerateHiit(string response)
2         {
3             string[] substrings = response.Split('/');
4
5             List<string> exercises = new ↩
                  ↪ List<string>(substrings[0].Split(','));
6             int high = int.Parse(substrings[1].Split(',')[0]);
7             int low = int.Parse(substrings[1].Split(',')[1]);
8
9             current.Exercises = exercises;
10            current.HighDuration = high;
11            current.LowDuration = low;
12
13            hiitReady.Raise();
14        }
```

Listado 5.10: Código encargado de serializar los datos recibidos de la petición HTTP para la ejecución del HIIT

- **SceneManager**: es el director de orquesta de las fases del HIIT. Las tres fases que están definidas son las siguientes:
  - *Preparación*: comprende las explicaciones previas sobre los ejercicios a realizar. Comienza cuando recibe la notificación del evento *StartGame* y lanza los eventos *PreparationStarted* al comenzar y *PreparationFinished* al finalizar. También se encarga de llamar al gestor de las voces para comenzar con las explicaciones de los ejercicios.
  - *Ejercicios*: esta fase comienza cuando se recibe una notificación del evento *Countdown-Finished*, que corresponde con la Interfaz de cuenta atrás para comenzar los ejercicios. Una vez en esta fase lanzará el evento *ExercisesStarted* y se encargará de ir iterando a

```
1            void StartPreparation() => ↩
                 ↪ StartCoroutine(PlayPreparation());
2            IEnumerator PlayPreparation()
3            {
4                preparationStarted.Raise();
5                yield return new WaitForSeconds(1f);
6
7                yield return ↩
                     ↪ voiceManager.GetComponent<VoiceController>().
8                    PlayPreparation();
9
10               preparationFinished.Raise();
11           }
```

Listado 5.11: Código correspondiente con la ejecución de la fase de preparation

través de cada uno de los ejercicios. Establecerá los momentos de alta y baja intensidad, notificándolo con sus respectivos Eventos (*HighIntensityInterval* y *LowIntensityInterval*). La duración de cada uno de estos intervalos es establecida por la información recibida de la web. Finalmente al acabar de iterar sobre los ejercicios, se lanzará el evento *ExerciseFinished*.

```
1            public void StartExercises() => ↩
                 ↪ StartCoroutine(PlayExercises());
2            IEnumerator PlayExercises()
3            {
4                exercisesStarted.Raise();
5                for (int i = 0; i < 2; ++i)
6                    foreach (var exercise in currentHiit.Exercises)
7                    {
8                        currentExercise.Name = exercise;
9                        highIntensityInterval.Raise();
10                       yield return new WaitForSeconds
11                           (currentHiit.HighDuration + 1);
12
13                       currentExercise.Name = "idle";
14                       lowIntensityInterval.Raise();
15                       yield return new WaitForSeconds
16                           (currentHiit.LowDuration + 1);
17                   }
18               exercisesFinished.Raise();
19           }
```

Listado 5.12: Código correspondiente con la ejecución de la fase de ejercicios

- *Vuelta a la Calma*: comprende la última fase de la ejecución de un ejercicio de respiración para volver a la calma. Actúa de forma similar a la fase de preparación, es decir, se lanza un primer evento de entrada *CalmStarted* y otro de salida *CalmFinished*. El tiempo que dure esta fase estará controlado por la ejecución de las explicaciones por parte del gestor de voces y el ejercicio a realizar.

- **AudioManagers**:
  - *MusicManager*: se encarga de ejecutar las pistas de música para cada fase. Hace uso de los ScriptableObjects del tipo AudioData y AudioOptions. AudioData mantiene un listado de canciones que van a reproducirse en cada fase y AudioOptions contiene los valores numéricos correspondientes a un volumen más alto cuando se le notifica la fase de alta intensidad y un volumen más bajo para la baja intensidad. Al recibir los eventos *HighIntensityInterval* o *LowIntensityInterval*, utilizará estos valores para actualizar el

```
1            public void StartCalm() => StartCoroutine(PlayCalm());
2            IEnumerator PlayCalm()
3            {
4                calmStarted.Raise();
5                yield return new WaitForSeconds(1f);
6
7                yield return ↩
                     ↪ voiceManager.GetComponent<VoiceController>()
8                     .PlayCalm();
9
10               calmFinished.Raise();
11           }
```

Listado 5.13: Código correspondiente con la ejecución de la fase de calma

volumen de la música. Cuando recibe las notificaciones *PreparationStarted*, *ExerciseStarted* y *CalmStarted* realizará un cambio de canción con el metodo *PlaySound(int number)*

```
1            [SerializeField] private AudioData audioClips;
2            [SerializeField] private AudioOptions audioOption;
3
4            ...
5
6            public void PlaySound(int number)
7            {
8                source.clip = audioClips.Clips[number];
9                source.Play();
10           }
11
12           public void ChangeVolume(bool intensity)
13           {
14               if(intensity) source.volume = ↩
                     ↪ audioOption.HighVolume;
15               else source.volume = audioOption.LowVolume;
16           }
```

Listado 5.14: Fragmentos de código extraidos de AudioController.cs

- *SoundsManager*: funciona de la misma manera que el gestor de la música, pero en lugar de utilizar el ScriptableObject *Music*, utiliza *SoundEffects*. Los efectos de sonido consisten en un pitido que se va a ejecutar cada vez que recibe los eventos *CountDownFinished*, *HighIntensityInterval*, *LowIntensityInterval* y *ExercisesFinished*. Cuando se recibe la recompensa, el evento *OpenChest* hace que se reproduzca el sonido de una pequeña explosión.
- *VoiceManager*: se encarga de reproducir los audios con las explicaciones de los ejercicios. Se han creado assets de AudioData que contienen un listado de clips de sonido. Se ha creado uno tanto para la fase de preparación como para la de calma. Y para explicar los ejercicios se ha generado un AudioData. Estos ejercicios están recogidos en otro ScriptableObject (ExerciseVoice) que contiene un diccionario cuyas claves son strings de los nombres de los ejercicios y los valores los AudioData correspondientes. Este diccionario se ha conseguido serializar en el editor gracias a la biblioteca *SerializableDictionary*.

Interfaz de Usuario

El Canvas o Lienzo es un GameObject con un Componente Canvas asociado que nos permite controlar un grupo de elementos para generar interfaces de usuario de una forma sencilla y configurarlas

```
1             IEnumerator PreparationVoices()
2             {
3                 source.clip = preparation.Clips[0];
4                 source.Play();
5                 yield return new ↩
                      ↪ WaitForSeconds(preparation.Clips[0].length);
6
7                 foreach(var exercise in currentHiit.Exercises)
8                     yield return StartCoroutine
9                         (PlayExerciseVoices(exercise));
10
11
12                source.clip = preparation.Clips[1];
13                source.Play();
14                yield return new ↩
                      ↪ WaitForSeconds(preparation.Clips[1].length);
15            }
16
17            IEnumerator PlayExerciseVoices(string key)
18            {
19                for(var i = 0; i < ↩
                      ↪ exercises.Dictionary[key].Clips.Count; i++)
20                {
21                        source.clip = ↩
                              ↪ exercises.Dictionary[key].Clips[i];
22                        source.Play();
23                        yield return new WaitForSeconds(exercises.
24                            Dictionary[key].Clips[i].length);
25
26                        if(i == 0)
27                        {
28                            explainExercise.Raise();
29                            yield return new WaitForSeconds(4f);
30                        }
31                        else
32                         stopExerciseExplanation.Raise();
33                }
34            }
35
36            IEnumerator CalmVoices()
37            {
38                source.clip = calm.Clips[0];
39                source.Play();
40                yield return new ↩
                      ↪ WaitForSeconds(calm.Clips[0].length);
41
42                source.clip = calm.Clips[1];
43                source.Play();
44                yield return new ↩
                      ↪ WaitForSeconds(calm.Clips[1].length);
45            }
```

Listado 5.15: Métodos para la ejecución de las voces en las fases

para que se muestren correctamente. La mayor parte de los elementos que forman la interfaz se encuentran dentro de este objeto y muestran información al usuario sobre la ejecución del HIIT:

- **Start**: boton de inicio del HIIT. Es mostrado al recibir una notificación del evento *WebBridgeReady* cuando ya está preparado todo. Al ser pulsado, lanza el evento *StartGame* para poner en marcha al resto de sistemas (y ocultarse a sí mismo).
- **Botones de control**: está formado por dos botones: un botón de pausa y otro de pantalla

completa. Al emplear el botón de pausa, se modifica el valor de *Time.timeScale* para parar o reanudar la ejecución de Unity. También itera a través de una lista con los gestores de audio (por inyección de dependencia) para pausar o reanudar los sonidos. Por último, el botón para pasar a pantalla completa realiza una llamada a una función de *.jslib* para utilizar la función en JavaScript de *SetFullScreen* sobre la *Instancia de Unity* que se ha creado en el documento HTML.

- **Barra de progreso**: se encarga de generar un panel por cada ejercicio (y repetir el proceso una segunda vez). Cada vez que recibe un evento *HighIntensityInterval*, *PanelController.cs* se encarga de avisar al siguiente panel (dentro de sus hijos) que le toque comenzar a rellenar la barra.

```
1         public void GenerateProgressBar()
2         {
3             int index = 0;
4             for(int i=0; i<2;++i) //Se realiza dos veces
5                 foreach(var exercise in current.Exercises) //Para
                      cada ejercicio activamos un panel y le incluimos su
                      nombre
6                 {
7                     var child = target.transform.GetChild(index)
8                         .gameObject;
9                     child.SetActive(true);
10                    child.GetComponentInChildren<TextMeshProUGUI>()
11                        .text = exercise;
12                    index++;
13                }
14        }
```

Listado 5.16: Fragmento de código extraidos de InfoPanelController.cs

- **Temporizador**: aparece y se oculta con los eventos de *ExerciseStarted* y *ExerciseFinished*, respectivamente. Tiene una referencia a *CurrentHIIT* para utilizar los valores de alta y baja intensidad definidos en la petición inicial al servidor. Cuando recibe una notificación de los eventos *HighIntensityInterval* o *LowIntensityInterval* actualiza su contador con los valores mencionados anteriormente. A través de la función *Update*, se decrementa su valor cada segundo y al llegar a cero, lanza el evento *TimerFinished*.
- **Visualización previa**: se muestra durante la fase de baja intensidad indicando el siguiente ejercicio. Aparece en la pantalla al recibir la notificación de *LowIntensityInterval* y desaparece al comenzar un intervalo de alta intensidad (con el evento *HighIntensityInterval*).

CAPÍTULO 6

Resultados

Una vez ya han sido expuestos la problemática, los objetivos planteados, la metodología de trabajo y las principales decisiones de diseño que se han tomado a la hora de desarrollar la solución, se exponen los resultados conseguidos en términos del aspecto final del sistema. También se aporta información sobre las estadísticas del proyecto y el código desarrollado, así como unos resultados tras someter a la plataforma web a ciertas pruebas de rendimiento. Para finalizar con el capitulo, se expone una estimación de los costes que supone el proyecto.

6.1. ASPECTO Y FUNCIONALIDAD DEL SISTEMA

En esta sección se tratan los resultados obtenidos para cada una de las iteraciones propuestas en la planificación del Capítulo 4. Cabe mencionar, que el aspecto final del sistema surge como resultado del desarrollo realizado entre los diferentes miembros del equipo que ha participado en el proyecto.

6.1.1. Iteración 1: Evaluación y selección del avatar

La primera iteración que se realizó, consistió en un estudio de mercado de los diferentes avatares existentes dentro de la tienda de Unity. Después de decidir cuales eran los posibles avatares más adecuados para el proyecto, se realizaron pruebas de aceptación para comprobar aquellos que más llamaban la atención en los niños.

Se realizó una primera adquisición del avatar *Robot Boy*[1], que fue el modelo con mayor aceptación. Pero finalmente, debido a problemas con las dimensiones del modelo, se terminó por adquirir el paquete *2 Toon Kids*[2].

Figura 6.1: Avatar *Robot Boy*

Figura 6.2: Avatares *2 Toon Kids*

[1] https://assetstore.unity.com/packages/3d/characters/robots/robot-boy-78484
[2] https://assetstore.unity.com/packages/3d/characters/humanoids/2-toon-kids-104298

6.1.2. Iteración 2: Estudio de patrones de diseño

En esta iteración se realizó el estudio de los diferentes patrones existentes para la programación de videojuegos y de cómo aplicarlos en Unity [3].

Varios de los patrones que se estuvieron analizando encajaban con el diseño de algunas piezas del sistema. A continuación se exponen y definen brevemente aquellos patrones que llamaron más la atención para su aplicación en el proyecto:

- **Máquina de Estados Finita (FSM)**: es un conjunto de unos estados del sistema, en los que puede estar la ejecución, y las transiciones existentes entre los diferentes estados definidos. Estas transiciones actúan como condicionantes que permiten avanzar a otro estado, y por ende, realizar otro tipo de acción.
- **Singleton**: consiste en el tratamiento de una clase como una instancia única y global a la que pueden acceder todos los objetos de la escena. Es útil para la creación de managers globales que contienen variables y funciones que son utilizadas por muchas otras clases. Durante el Capítulo 5 se muestran los ScriptableObjects, una excelente alternativa a este tipo de patrones.
- **Observador**: en este patrón los Observadores son conscientes de la existencia de los Sujetos, y los propios Sujetos mantienen un listado de aquellos que están observando. Cuando los Sujetos lanzan un evento, notifican al resto de Observadores en su lista.
- **Publicador-Subscriptor**: es un patrón que se basa en el concepto anterior pero desacoplando por completo a Publicadores y Subscriptores. Se emplea un Canal de Eventos en el cual los Subscriptores están a la escucha y los Publicadores lanzan mensajes o eventos.
- **Inyección de dependencias**: es un patrón de diseño que se basa en la idea de encargar la creación de instancias de objetos a un componente, para que inyecte dichos objetos en otra clase que los necesite. En el caso de este patrón, Unity nos facilita su uso a través de campos Serializables en el Editor.

Finalmente, para interiorizar mejor los conceptos de los patrones que se querían aplicar, se realizaron diversas implementaciones dentro de la herramienta de Unity.

6.1.3. Iteración 3: Implementación en Unity

Esta iteración consistió en el uso de la herramienta Unity para el desarrollo del módulo de representación 3D. A raíz de los conocimientos obtenidos durante la iteración anterior, se implementó un primer prototipo haciendo uso varios de los patrones de diseño.

En primer lugar, se puso en práctica la implementación de una *FSM* para definir las diferentes fases que forman el HIIT: Preparación, Ejercicios y Vuelva a la Calma.

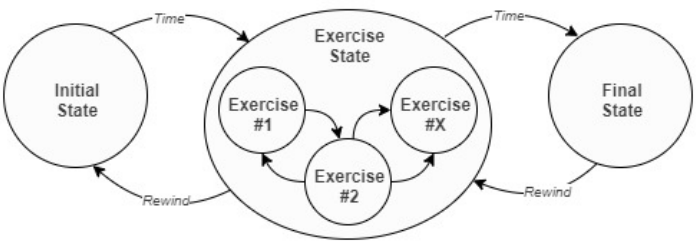

Figura 6.3: Diagrama de Máquina de Estados

Esta máquina de estados estaba definida en torno a una clase abstracta *BaseState*. En el Listado 6.1 se puede observar la definición de dos métodos, cuyas finalidades eran ser ejecutado al entrar en

[3] https://github.com/Habrador/Unity-Programming-Patterns

el estado (*EnterState*) y ser ejecutado en cada frame (*Update*). Para cada una de las fases del HIIT, se diseñó una nueva clase implementando a la clase abstracta BaseState: *InitialState* (Preparación), *ExerciseState* (Ejercicios) y *FinalState* (Vuelta a la Calma).

```
1 public abstract class BaseState
2 {
3     public abstract void Update(HiitController controller);
4     public abstract void EnterState(HiitController controller);
5 }
```

Listado 6.1: Código fuente de la clase BaseState.cs

Las nuevas clases definían los comportamientos de cada uno de los estados al entrar en él y durante su ejecución. Dentro del estado de los Ejercicios se realizaba una simulación, ejecutando un ejercicio detrás de otro tras pequeñas esperas de tiempo.

Para mantener el estado actual y cambiar entre ellos, se utilizaba una clase *HiitController* que derivaba de MonoBehaviour. Esta clase era un *Singleton* con una declaración del estado abstracto BaseState (utilizado como estado actual), y otras declaraciones para cada uno de los estados por los que podía transicionar.

A partir de este punto, se comenzó a indagar en un sistema de eventos que permitiera la comunicación entre los diferentes componentes de la UI y los gameObjects de la escena.

6.1.4. Iteración 4: Sistema de eventos

Al comienzo de esta iteración, el sistema de eventos planteado estaba basado en un patrón del *Observador*. Se hizo una implementación de un Broker de Eventos, encargado de notificar al resto de clases cuando un Evento había sido lanzado (ver Figura 6.4). Este tipo de diseño acabó durando poco debido al uso más en profundidad de los ScriptableObjects. Esto dio lugar a una implementación de una arquitectura modular y orientada al diseño, el Sistema de Eventos (ver Capitulo 5.2.4).

Este Sistema está basado en un patrón *Publicador-Subscriptor* en el que las clases Publicadoras hacen uso de una referencia a un asset del tipo GameEvent para lanzar el evento y el resto de clases Subscriptoras permanecen a la escucha hasta recibir ese mensaje. En este caso se hace uso de los ScriptableObjects como Canales de Eventos.

6.1.5. Iteración 5: Estudio del sistema de captura

Antes de poder realizar algún tipo de grabación, era necesaria la familiarización del autor con el sistema de captura de movimiento. Para ello se llevó a cabo una primera toma de contacto con Xsens Awinda durante esta iteración, en la que se realizaron algunas grabaciones para comprobar el funcionamiento del sistema. Una vez conocida la manera de trabajo con la herramienta, se volvió a realizar una segunda sesión con la finalidad de generar algunas animaciones e integrarlas en los Avatares desde la herramienta Blender.

Finalmente, tras realizar pruebas para comprobar que se hacía la integración de la animación de forma exitosa con los Avatares (ver Figura 6.5), se dio paso a la siguiente iteración: la grabación de las animaciones de los ejercicios.

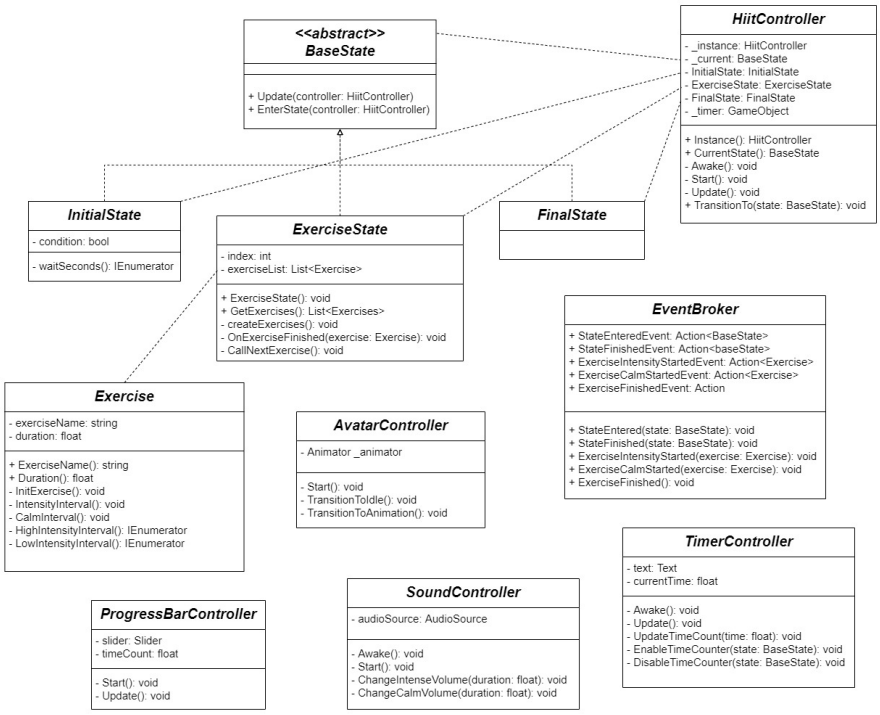

Figura 6.4: Diagrama de clases del prototipo inicial

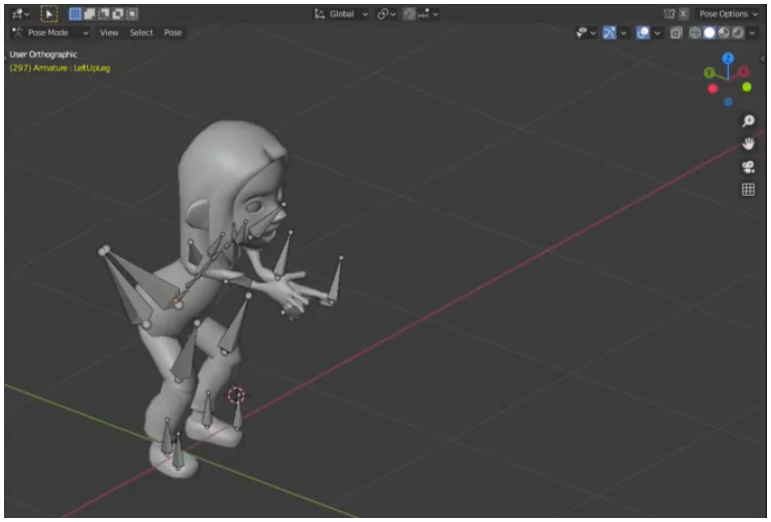

Figura 6.5: Captura de la herramienta Blender: Avatar con esqueleto integrado

6.1.6. Iteración 6: Grabaciones de las animaciones

Esta iteración tuvo un ciclo de vida de un único día. Consistió en una sesión de grabación que se realizó el día 26 de Marzo de 2021. Se utilizó el traje de Xsens Awinda junto con su software MVN Animate para grabar un total de 42 ejercicios. Las pautas y ejecución de los ejercicios fueron realizados por los miembros del proyecto de la facultad de Magisterio (ver Figura 6.6).

6.1.7. Iteración 7: Implementación de la plataforma web

Al comienzo de esta iteración se implementó la integración del módulo de representación 3D con la plataforma web, para poder obtener un feedback directo del resto de participantes del proyecto. La puesta en vivo de la plataforma fue posible gracias al uso del servidor FTP. Se continuó con el diseño

Figura 6.6: Sesión de captura de movimiento en Magisterio

de la base de datos según los requisitos establecidos para el sistema y una vez estuvo operativo el servicio de Amazon Web, se procedió a configurar la base de datos y definir los endpoints para las solicitudes.

Durante este tiempo se desarrolló de forma paralela, por parte de ambos programadores, las funcionalidades de la plataforma web y el sistema de peticiones HTTP. Aquí fue donde cobró una mayor importancia el uso de herramientas como *Trello* que nos permitió conocer qué tareas estaba realizando el otro miembro para así trabajar de forma simultánea.

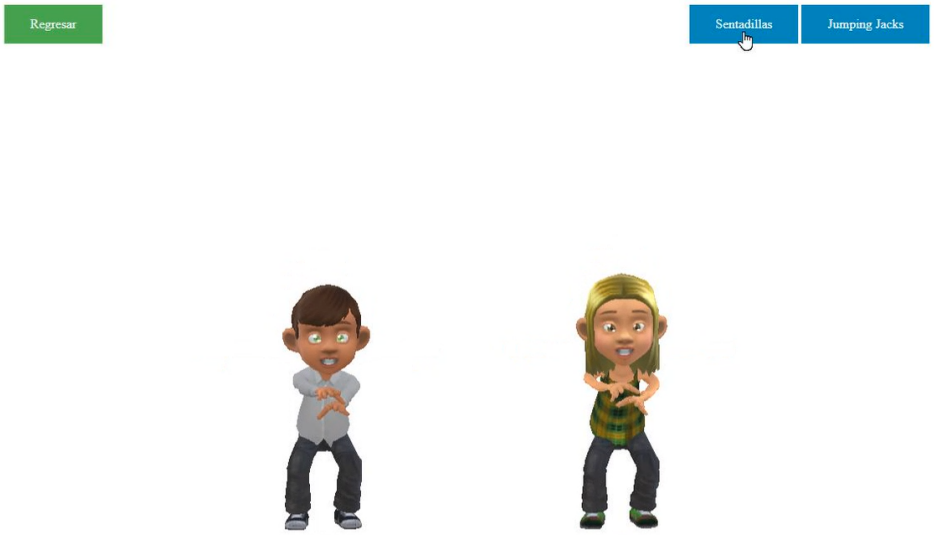

Figura 6.7: Módulo de representación 3D integrado en la plataforma web

En combinación con el desarrollo de la Iteración 8, se completó el apartado de funcionalidades con el apartado visual, dando como resultado la solución final del proyecto. Si se desea conocer en mayor detalle la funcionalidad de la plataforma, se recomienda al lector revisar el Capítulo 5.2.

6.1.8. Iteración 8: Implementación de los diseños

Durante esta iteración, se pasó de tener unos diseños básicos en la plataforma web (implementados en los inicios de la iteración anterior) a unos más orientados a la finalidad del propio proyecto: *utilizarse en aulas de Educación Infantil.*

En este punto se exponen los diferentes diseños artísticos que se han utilizado para definir el estilo de la plataforma. Hay que tener en cuenta que tanto en el diseño y organización de las hojas de estilo, como en las funcionalidades del sistema, se han realizado diversas decisiones tomando como referencia los diseños proporcionados.

En las Figuras 6.8 y 6.9 encontramos la paleta de colores que se ha utilizado para todo el diseño de las diferentes páginas web y la interfaz del módulo de representación 3D; y el tipo de letra utilizado en toda la plataforma. Los colores empleados están extraídos de los propios colores de la mascota del programa MOVI en el logotipo inicial (ver Figura 1.2).

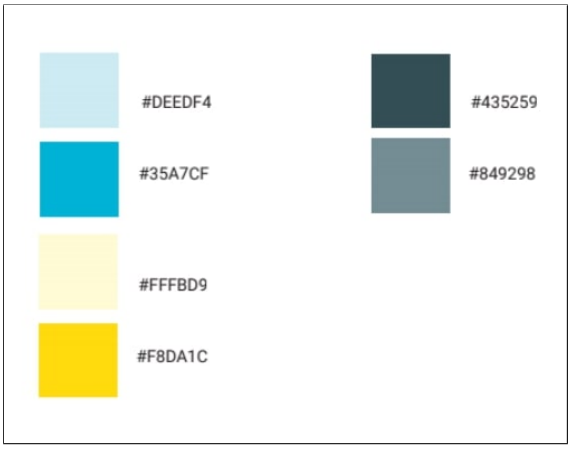

Figura 6.8: Paleta de colores utilizada para el diseño de las páginas

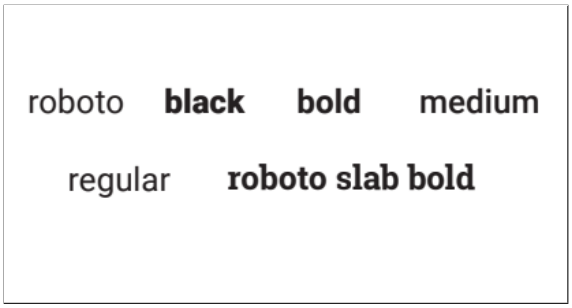

Figura 6.9: Fuente de letra en la plataforma

A continuación se exponen todos los diseños en los que se ha basado el desarrollo de la plataforma. En primer lugar, se muestran los diseños para el sistema de acceso a la plataforma en la Figura 6.10 compuestos por la página de Login y Registro. En segundo lugar, en la Figura 6.11 se muestran los diseños para las distintas páginas pertenecientes a la plataforma web para los Usuarios. Y por último, en la Figura 6.12, se muestran todos los diseños que han sido utilizados como referencia para la plataforma de Administradores.

Figura 6.10: Diseños para la página de Login y Registro

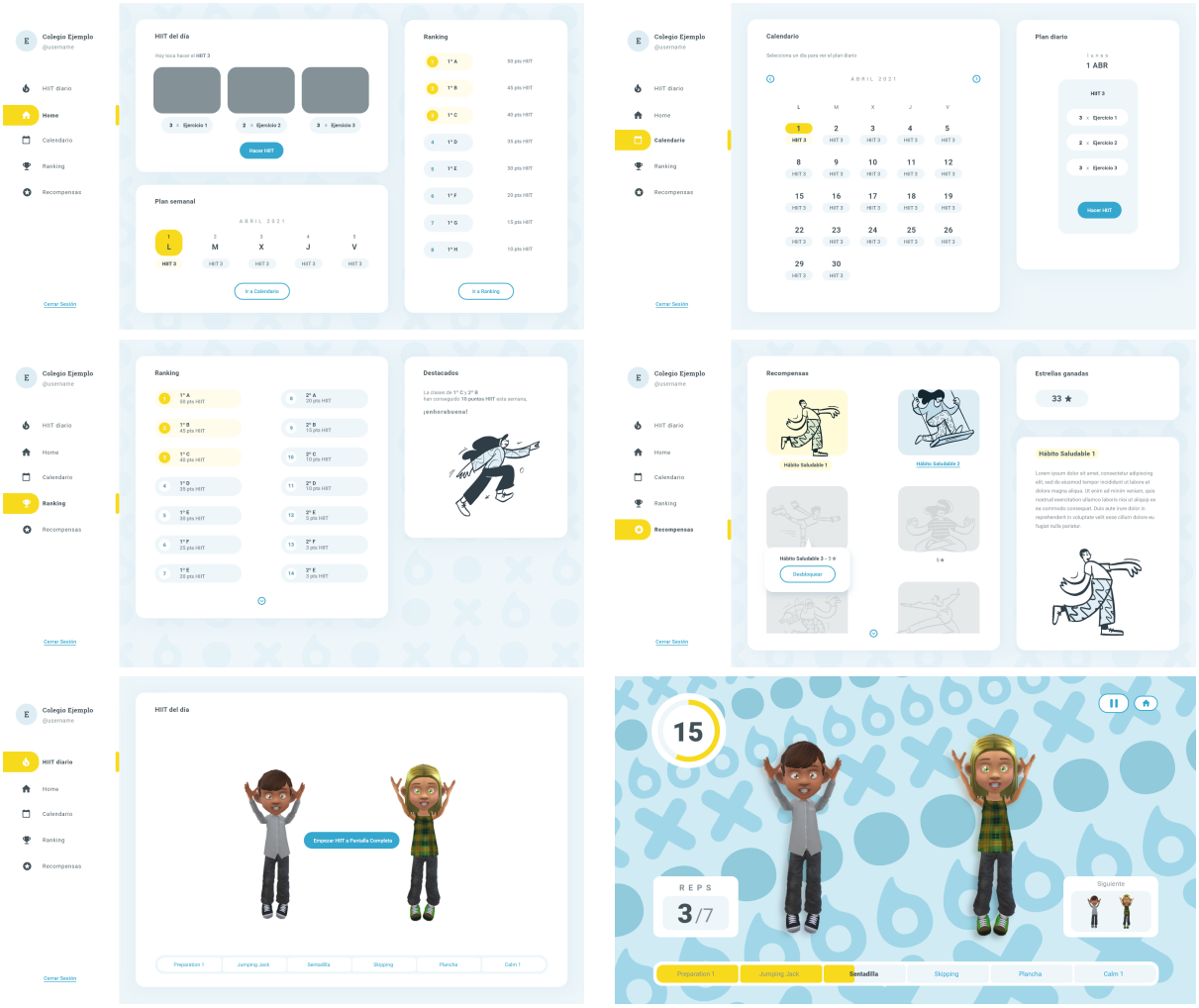

Figura 6.11: Diseños de la plataforma web para los Usuarios

Se incita al lector a revisar el Anexo B donde se define el Manual de Usuario para el uso de la plataforma. Dentro del mismo se podrá encontrar explicaciones de las Plataformas para el Usuario y para el Administrador de manera separada así como imágenes de cada una de las páginas implementadas.

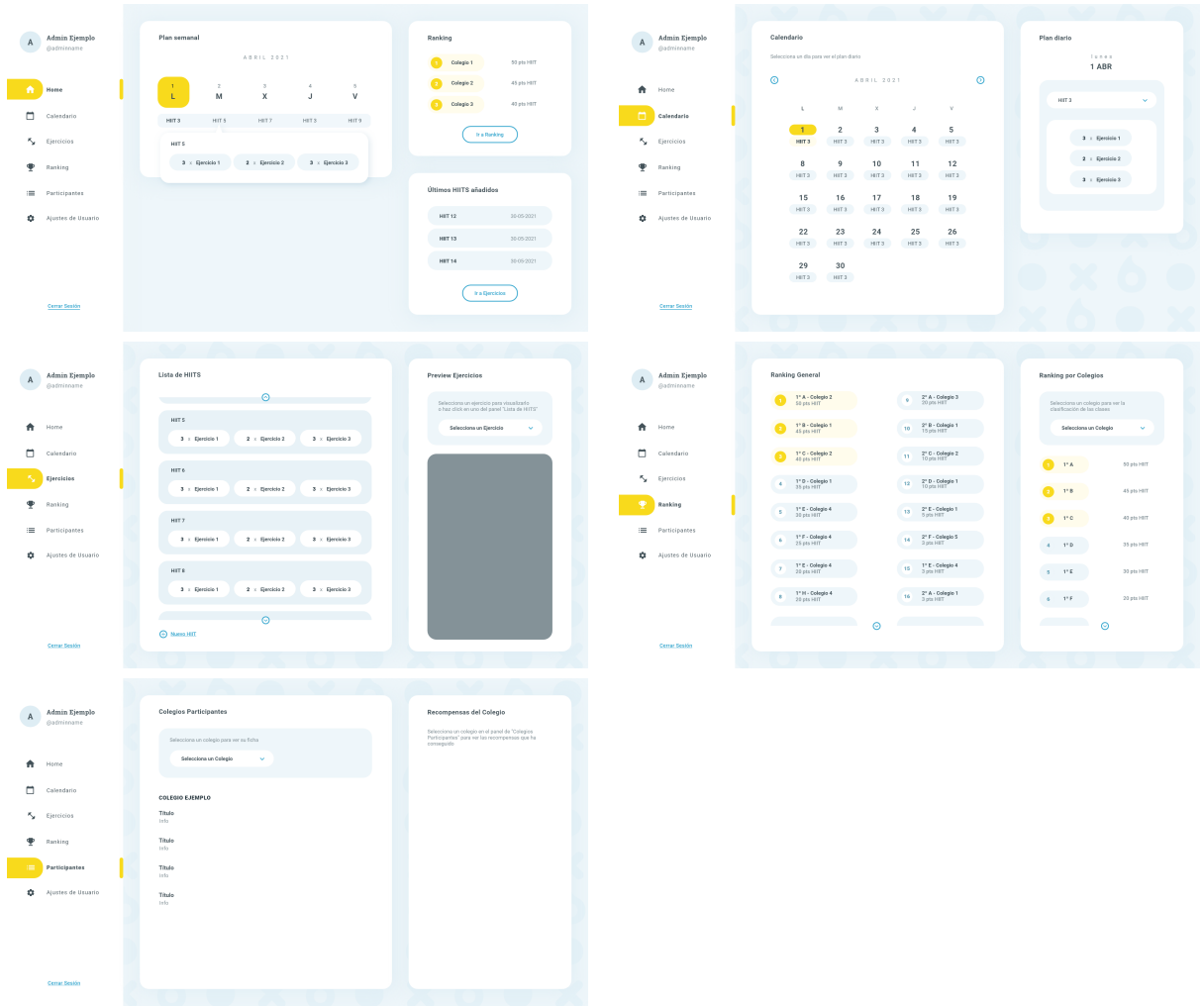

Figura 6.12: Diseños de la plataforma web para los Administradores

6.2. ESTADÍSTICA DEL PROYECTO

En esta sección se exponen las estadísticas del proyecto que han sido extraídas de los repositorios utilizados para el desarrollo del proyecto. En primer lugar, se muestran resultados obtenidos tras la ejecución de la herramienta *Count Lines Of Code*[4] sobre los repositorios utilizados para el desarrollo en Unity y para la plataforma web. Se han excluido todos aquellos directorios que no aportasen información esencial a estas estadísticas. Así mismo aquellos archivos que contienen lenguajes generados automáticamente también han sido excluidos. En segundo lugar, se ha hecho uso de un plug-in dentro de BitBucket, llamado *AwesomeGraphs*[5]. Este plug-in permite monitorizar los repositorios a través de representaciones visuales de las estadísticas del mismo.

En las primeras estadísticas de la Tabla 6.1 se muestran los datos obtenidos del módulo de representación 3D desarrollado en Unity. Se muestra el lenguaje utilizado, la cantidad de archivos, espacios en blanco, comentarios y líneas de código.

Tabla 6.1: Estadísticas para el módulo de representación 3D

Lenguaje	Archivos	Espacios en blanco	Comentarios	Líneas de código
C	32	165	41	714
Total	**32**	**165**	**41**	**714**

En estas segundas estadísticas, reflejadas en la Tabla 6.2, se muestra el análisis realizado sobre el repositorio de la plataforma web. Como hemos mencionado anteriormente, se han excluido todos aquellos archivos que no han sido desarrollados por el autor.

Tabla 6.2: Estadísticas para la plataforma web

Lenguaje	Archivos	Espacios en blanco	Comentarios	Líneas de código
HTML	12	206	273	1116
CSS	17	309	78	1264
JavaScript	19	232	152	1053
Total	**48**	**745**	**503**	**3425**

Si combinamos la información de ambas tablas, obtendremos las estadísticas resultantes para el componente del Cliente dentro de la totalidad del proyecto. Esto haría un total de:

- **Archivos**: 80
- **Espacios en blanco**: 910
- **Comentarios**: 544
- **Líneas de código**: 4139

Para proporcionar una información más rigurosa y detallada sobre el proyecto que se ha desarrollado, se ha incluido también, aunque no han sido desarrolladas por el autor, las líneas de código que componen el Servidor. Estos datos pueden se ven relflejados en la Tabla 6.3.

Tabla 6.3: Estadísticas para el Servidor

Lenguaje	Archivos	Espacios en blanco	Comentarios	Líneas de código
PHP	33	225	5	1975
Total	**33**	**225**	**5**	**1975**

Ahora si, podríamos obtener las estadísticas totales del proyecto combinando la información obtenida para la parte del Cliente y del Servidor. Esto resultaría en **113 archivos**, con **1135 espacios en blanco**, **549 comentarios** y un total de **6114 líneas de código**.

[4] https://github.com/AlDanial/cloc

[5] https://marketplace.atlassian.com/apps/1210934/awesome-graphs-for-bitbucket

En la Figura 6.13 se muestra una gráfica con la distribución del número de commits. El eje Y representa la cantidad de commits realizados y el eje X representa los meses de duración del desarrollo del proyecto. También podemos encontrar un registro del número total de commits que se han realizado y la cantidad de líneas insertadas y eliminadas en total.

Este gráfico recoge los commits realizados tanto para la plataforma web como para el módulo de representación 3D. Además, se puede ver reflejado el seguimiento de la planificación que se ha planteado en el Capitulo 4.

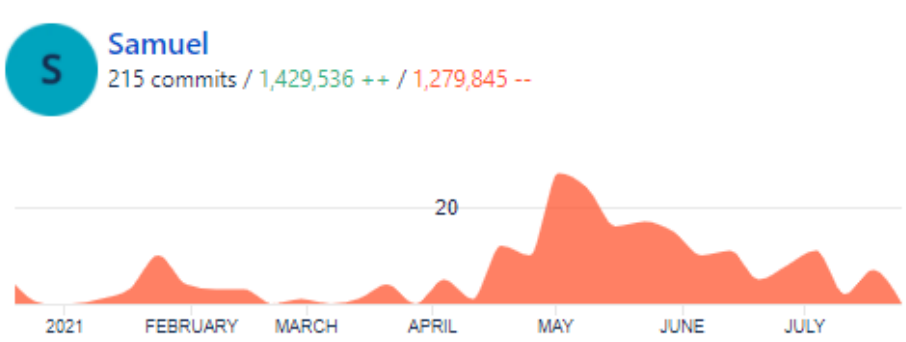

Figura 6.13: Contribución total individual al proyecto

6.3. COSTES

El desarrollo de este proyecto abarca desde el 02/12/2020 hasta 15/07/2021. Durante toda la duración del desarrollo del proyecto se trabajó a media jornada durante 5 días a la semana.

En el transcurso del año 2021, se realizaron las incorporaciones del resto de integrantes del proyecto en diferentes momentos del mismo. El segundo programador se incorporó a finales del mes de Abril trabajando a jornada completa durante los 5 días de la semana. La diseñadora gráfica se involucró en el diseño por una duración de 2 semanas y media a jornada completa. Finalmente el animador hizo su incorporación el mes de Junio a media jornada. Por motivos de confidencialidad del contrato, no se añaden los sueldo de los integrantes del proyecto. En su lugar, se realizan unas estimaciones de los sueldos haciendo uso de datos de sueldos obtenidos de profesionales *Junior* de cada uno de los sectores.

Según la web Tusalario.es [6] los sueldos medios mensuales para cada una de las profesiones a jornada completa son los siguientes[7]:

- **Programador aplicaciones informáticas**: 1373 €/mes, aproximadamente 8 €/hora.
- **Diseñador gráfico**: 972 €/mes, aproximadamente 6 €/hora.
- **Diseñador de animación**: 972 €/mes, aproximadamente 6 €/hora.

Por lo tanto, si calculamos los tiempos para cada uno de los profesionales obtendríamos:

- **Programador frontend**: 600 h
- **Programador backend**: 480 h
- **Diseñador gráfico**: 100 h
- **Animador**: 120h

Los tiempos empleados y los costes estimados se ven reflejados en la Tabla 6.4.

[6] https://tusalario.es/salario/comparatusalario#/

[7] Se han tomado los valores de sueldo mínimos para adecuarse con el salario de profesionales de categoría Junior

Tabla 6.4: Desglose de los costes y tiempos estimados para MOVI-HIIT

Recurso	Cantidad	Tiempo estimado	Coste
Sueldo programador (Frontend)	1	600 h	4800 €
Sueldo programador (Backend)	1	480 h	3840 €
Sueldo diseñador gráfico	1	100h	600 €
Sueldo animador 3D	1	120h	720 €
Avatares 3D	2		21 €
Xsens Awinda	1		4390 €
Total			**14371 €**

CAPÍTULO 7

Conclusiones

Para concluir con el presente documento, este capitulo incluye una pequeña reflexión acerca del resultado obtenido y de cómo han sido alcanzados cada uno de los objetivos propuesto. Además, se hace reflexión acerca de aquellas líneas de trabajo futuro que podrían continuarse para la mejora del sistema actual. Finalmente, se incluye una reflexión por parte del autor con sus conclusiones personales acerca del desarrollo del proyecto.

7.1. CONCLUSIONES

Como se estableció en los objetivos del Capítulo 2, el objetivo general del proyecto consiste en el desarrollo de una solución web multiplataforma, cuya finalidad principal es la representación 3D de actividad física desde un enfoque interactivo. Este objetivo ha sido logrado y puesto en marcha (véase https://movi-hiit.furiouskoalas.com/). La plataforma final es completamente funcional en distintos computadores y navegadores web.

Dentro de los objetivos específicos planteados se encuentra el requisito de **facilidad de uso de la plataforma**, ya que debe ser utilizado por profesores que pueden tener escasos conocimientos informáticos. Este objetivo específico se ha logrado con la ayuda de los diseños planteados tanto para la plataforma web como para el módulo de representación 3D, y que pueden ser revisados junto con el aspecto final del proyecto (véase Capitulo 6).

Otros objetivos, como son **representación 3D con integración web** y **reproducción de sonidos**, también se han logrado satisfactoriamente. Esto ha sido en parte gracias al uso de la herramienta Unity, la cual permite la exportación a múltiples plataformas y el uso de sistemas de audio programables. Por otro lado, la **integración de avatares virtuales** se ha logrado tras realizar pruebas de aceptación con niños. Esto ha permitido elegir los mejores avatares en base a sus criterios.

Para poder lograr el objetivo del **desarrollo de una solución web multiplataforma** se ha llevado a cabo una combinación de lenguajes, es decir, HTML, CSS, JS y PHP, aparte de utilizar AWS como solución cloud en lo que respecta al despliegue de la aplicación (véase Capitulo 5). En esta misma línea, también se ha logrado el objetivo del **desarrollo de un módulo para la gestión de la plataforma y control de usuarios** con páginas para poder crear, modificar y eliminar HIITs, así consultar la información de los participantes.

Por último, para lograr el objetivo específico de la **representación de ejercicios por medio de animaciones**, se ha hecho uso del sistema de captura de movimiento Xsene Awinda para generar archivos *.fbx* y su posterior integración con el avatar 3D a través de Blender.

7.2. LÍNEAS DE TRABAJO FUTURAS

Con respecto a las líneas de trabajo futuras, cabe destacar que el proyecto sigue en desarrollo y por lo tanto está sujeto a constantes cambios y evoluciones. Pese a ello, durante el desarrollo del mismo, han surgido ideas de mejoras o implementaciones que podrían ser interesantes de incorporar.

Una de ellas consistiría en la evolución del actual sistema de eventos propuesto para este proyecto, hacia un **sistema de eventos genérico**. La premisa detrás de esto reside en el uso de los Componentes *GameEventListeners* como sistema de eventos que permiten realizar respuestas basadas en *UnityEvents*. Actualmente estas respuestas consisten en llamadas a diferentes funciones que permiten argumentos de primitivas tales como enteros, booleanos y cadenas. La idea es poder definir el contenido de la respuesta basada en este tipo de objetos para almacenar cualquier tipo de dato, lo cual permita una mayor modularización del sistema.

Así mismo, el sistema MOVI-HIIT servirá como base para el desarrollo de un nuevo proyecto con características muy similares. Consiste en un proyecto a nivel europeo, denominado **EUMOVE**. Esta nueva plataforma se basará en el uso y mejora tanto del sistema de consultas del Servidor como del módulo de representación 3D. A continuación vamos a comentar algunas de las evoluciones y modificaciones que se pretenden realizar a partir de la base del presente proyecto.

En primer lugar, una de las funcionalidades que se pretende mejorar es el **sistema de acceso a la plataforma** mediante la incorporación de un **inicio de sesión a través de Google + y Facebook**. Esto no llegó a desarrollarse debido a la necesidad por parte del proyecto MOVI-HIIT de mantener un registro cerrado para el programa. Esta nueva integración supondría un beneficio a la hora de acceder a la plataforma por parte de los usuarios, ya que agilizaría el acceso a la misma y evitaría la generación de nuevas contraseñas.

Una de las incorporaciones más importantes que se espera llevar a cabo es la **implementación de un módulo de síntesis de voz**. Su objetivo es sintetizar texto de diversos lenguajes en audio para ser utilizado dentro del propio módulo de representación 3D descrito en este documento. Se pretende conseguir que las personas que usen la plataforma puedan crear sus propios HIITs junto con indicaciones o explicaciones de texto que se traducirán en audio.

Otra de las lineas de trabajo que se pretende expandir es la **implementación de un modulo de traducción**, dando soporte a la traducción del texto de la plataforma entre los lenguajes disponibles mediante el uso de inteligencia artificial.

7.3. JUSTIFICACIÓN DE COMPETENCIAS ADQUIRIDAS

En este apartado se describen y justifican las competencias correspondientes a la intensificación de *Computación*:

- **[CM1]**: *Capacidad para tener un conocimiento profundo de los principios fundamentales y modelos de la computación y saberlos aplicar para interpretar, seleccionar, valorar, modelar, y crear nuevos conceptos, teorías, usos y desarrollos tecnológicos relacionados con la informática.*
  Esta competencia se cumple en la medida que se ha interpretado el concepto de descanso activo. El desarrollo tecnológico ha consistido en una solución 3D interactiva que ayude a docentes a integrar descansos activos en el aula y prevenir la obesidad en niños, así como mejorar su rendimiento académico.
- **[CM4]**: *Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.*

Esta competencia está presente en lo que respecta a la utilización del entorno de Unity. Este motor basa su funcionamiento en objetos que son tratados como agentes. En este sentido, los scripts han sido diseñados para coordinarse dentro de la arquitectura de agentes en Unity con comportamientos específicos. Esta competencia también se cumple debido a que se ha utilizado un sistema de captura de movimiento inercial con el que se han creado animaciones para avatares consumidos en la aplicación web.

- **[CM6]**: *Capacidad para desarrollar y evaluar sistemas interactivos y de presentación de información compleja y su aplicación a la resolución de problemas de diseño de interacción persona computadora.*
  Este proyecto consiste en una plataforma web que recibe y transmite información por medio de la interacción directa de los usuario con el sistema. El sistema también representa información que puede ser consumida de una forma sencilla por los propios usuarios.

7.4. CONCLUSIÓN PERSONAL

Considero que el proyecto ha resultado una experiencia enriquecedora al haber sido desarrollado dentro del marco de un proyecto real. Han sido unos meses cargados de obstáculos, los cuales con constancia y paciencia han podido ser sacados adelante. Gracias al desarrollo de este trabajo he podido comprender en mucha mayor medida la magnitud de este tipo de proyectos, el significado del trabajo en equipo y la importancia en la organización durante el desarrollo.

Durante estos meses desarrollando MOVI-HIIT me he dado cuenta cuán complejo es el desarrollo de aplicaciones interactivas y he podido conocer más en profundidad herramientas como Unity o Blender que hasta ahora desconocía. Así mismo, teniendo un conocimiento más escaso sobre el tema, me he adentrado de lleno en la programación web.

He abierto las puertas al mundo de la programación de aplicaciones interactivas, la cual considero bastante interesante y donde la aplicación de la ingeniería juega un papel realmente. A pesar de haber ahondado mucho más en temas como patrones de diseño o diseño de arquitecturas, aún queda un largo camino por recorrer y estoy convencido de que los conocimientos obtenidos durante la realización de este Trabajo de Fin de Grado me servirán para realizar futuros proyectos de una manera más profesional.

Espero que este proyecto cumpla su función en su puesta en marcha en las diferentes aulas de los colegios durante la ejecución del programa MOVI-HIIT y logre los resultados deseados tanto por parte del autor como del resto de participantes del proyecto.

Bibliografía

[1] CESS. (2014). «Proyecto MOVI», Centro de Estudios Socio-Sanitarios, dirección: https://www.movidavida.org/.

[2] B. Cowan y B. Kapralos, *A Survey of Frameworks and Game Engines for Serious Game Development*. IEEE Computer Society, 2014.

[3] C. A. Dorantes. (2015). «PostgreSQL: qué es, cómo funciona y cuáles son sus ventajas», dirección: https://platzi.com/blog/que-es-postgresql/.

[4] J. Gutiérrez, *¿Qué es un framwork Web?* 2017.

[5] R. Hipple. (2017). «Game Architecture with Scriptable Objects», dirección: https://www.youtube.com/watch?v=raQ3iHhE_Kk.

[6] C. Larman y V. R. Basili, *Iterative and incremental developments. a brief history*. 2003.

[7] M. Latorre, *Historia de las Web, 1.0, 2.0, 3.0 y 4.0*. Universidad Marcelino Champagnat, 2018.

[8] MetaMotion. (). «Motion Capture - What is it?», dirección: https://metamotion.com/motion-capture/motion-capture.htm.

[9] OMS. (jul. de 2021). «Obesidad y Sobrepeso», Organización Mundial de la Salud, dirección: https://www.who.int/es/news-room/fact-sheets/detail/obesity-and-overweight.

[10] OMS'. (jun. de 2021). «Malnutrición», Organización Mundial de la Salud, dirección: https://www.who.int/es/news-room/fact-sheets/detail/malnutrition.

[11] StackOverflow. (2020). «2020 Developer Survey», dirección: https://insights.stackoverflow.com/survey/2020#technology-web-frameworks.

[12] A. Tarbal, *La Obesidad Infantil: una epidemia mundial*. 2010.

[13] U. Technologies. (2020). «WebGL: Interacting with browse scripting», dirección: https://docs.unity3d.com/es/2018.4/Manual/webgl-interactingwithbrowserscripting.html.

[14] M. Toftedahl y H. Engström, *A Taxonomy of Game Engines and the Tools that Drive the Industry*. University of Skövde, 2019.

[15] W3C. (2017). «HTML 5.2: W3C Recommendation», World Wide Web Consortium, dirección: https://www.w3.org/TR/html52/.

[16] Wikipedia. (2020). «Internet», Wikipedia, dirección: https://es.wikipedia.org/wiki/Internet.

ANEXOS

ANEXO A

Código Fuente

En este Anexo se detallan los principales directorios que estructuran el proyecto. Estos directorios organizan el código y los recursos del sistema de manera que sea mucho más fácil localizar

- **backend**: en este directorio se incluye el sistema de peticiones HTTP definido en 5.1. Contiene todos los endpoints accesibles del sistema.
- **resources**: directorio que contiene todas las fuentes, imágenes e iconos del sistema. También podemos localizar aquí los directorios CSS y JS que contienen todos aquellos archivos con funcionalidades o diseños de uso común para la plataforma.
- **webpages**: contiene todos los archivos HTML de cada una de las páginas web. Dentro de este directorio se han creado otros con un nombre representativo para cada web que, a su vez, estarán divididos en los directorios correspondientes a *users* y *admin*.
- **.htaccess**: sirve como archivo de redireccionamiento en caso de que accedamos a alguna página que no existe. También permite el uso del header «autorización» en las peticiones a la base de datos.
- **index.html**: es el encargado de hacer la redirección a la web de Log In. Actúa también como página inicial por parte del servidor FTP.

Este conjunto de directorios y archivos son los que están subidos al servidor FTP, encargado de enviar los archivos HTML del directorio *webpages* según la página web que estemos visitando, junto con los correspondientes recursos CSS y JS para su buena ejecución y visualización. En el caso del directorio *backend*, no es enviado por el servidor y es necesario conocer los endpoints para acceder de forma remota para poder realizar las peticiones a la base de datos en el servicio AWS.

Todo el código fuente del sistema se encuentra almacenado en un repositorio Bitbucket:

https://bitbucket.org/Samuglz6/movi-hiit

ANEXO B

Manual de Usuario

En este manual de usuario se recoge una breve guía de los pasos a seguir para el correcto uso de la plataforma MOVI-HIIT.

B.1. ACCESO A LA PLATAFORMA

Lo primero que necesitamos para poder acceder a la plataforma es ir al enlace de la página web:

https://movi-hiit.furiouskoalas.com/

Dentro de la página web encontraremos un formulario de Log In para poder verificarnos y así comenzar a utilizar la plataforma. Este formulario es muy sencillo y contiene los siguientes elementos:

- **Usuario**: aquí se debe introducir el nombre de usuario que se haya utilizado a la hora de hacer el registro en la plataforma.
- **Contraseña**: aquí se debe introducir la contraseña que establecimos cuando realizamos el registro en la plataforma.
- **Recuperar contraseña**: en el caso de que hayamos olvidado la contraseña de nuestro usuario podremos solicitar que nos la vuelvan a mandar a través del correo electrónico que teníamos registrado en la cuenta.
- **Botón de acceso**: una vez los credenciales han sido introducidos en los campos del usuario y la contraseña, utilizaremos este botón para verificar que lo que hemos introducido es correcto y acceder a la plataforma.

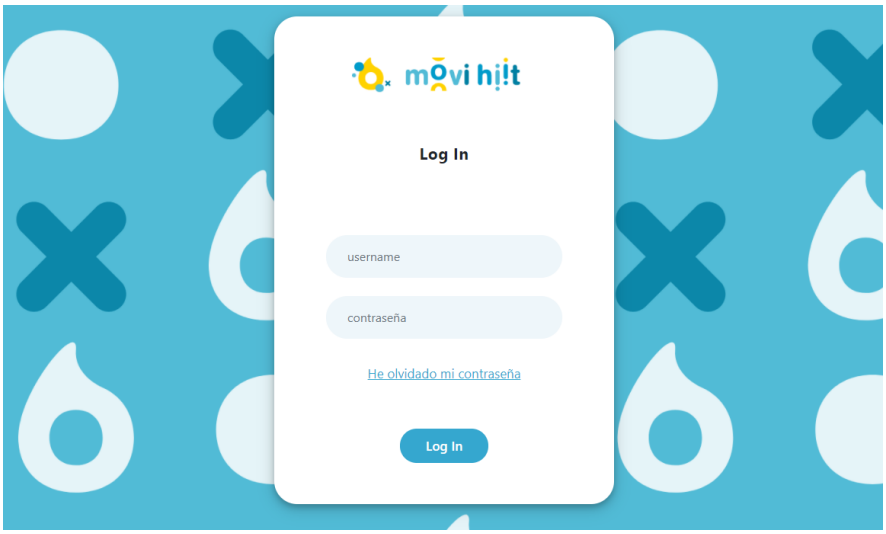

Figura B.1: Página de Log In de MOVI-HITT

Una vez introducidos los datos que se nos solicitan para el acceso, haremos click en el botón de Log In y esperaremos a que el sistema verifique nuestra identidad para poder acceder. En el caso de que alguno de los campos sea incorrecto, se mostrará una alerta indicando que ha habido algún problema con los credenciales introducidos.

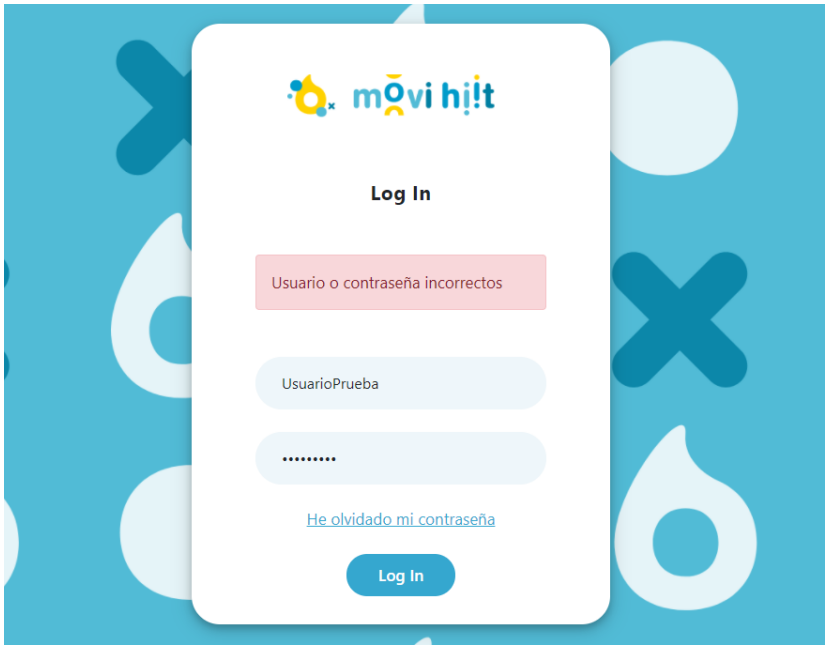

Figura B.2: Ejemplo de Log In incorrecto

En caso de que hayamos olvidado nuestra contraseña, solo tendremos que hacer uso de la opción que se facilita para la recuperación de contraseñas: *He olvidado mi contraseña.*

Si hemos hecho todo de forma correcta, habremos accedido al página web principal de la plataforma. Las páginas a las que podremos acceder, variarán dependiendo del rol que tengamos asignado en la cuenta: administrador o usuario.

B.2. MENÚ DE NAVEGACIÓN

Si estamos usando la plataforma en su versión de ordenador, el menú principal se mostrará en el lateral izquierdo de la pantalla. Los elementos que podemos encontrar son los siguientes:

- **Información del Usuario**: se encuentra en la cima del menú lateral. Aquí encontraremos el nombre del colegio/administrador actual junto con un icono con su inicial y el nombre de usuario antecedido por una @.
- **Paneles de selección**: estos paneles indican las páginas web por las que podremos navegar en la plataforma. Un indicador amarillo sobre una de las selecciones nos señala que esa página es la que se esta mostrando actualmente.
- **Cierre de Sesión**: el ultimo elemento del menú lateral, ubicado en la parte baja de éste, consiste en un botón para finalizar el uso de la plataforma. Nos devuelve al menú de Log In.

Siempre que se accede a la plataforma seremos redirigidos a la página principal [HOME]. Podremos desplazarnos haciendo uso de los paneles de selección.

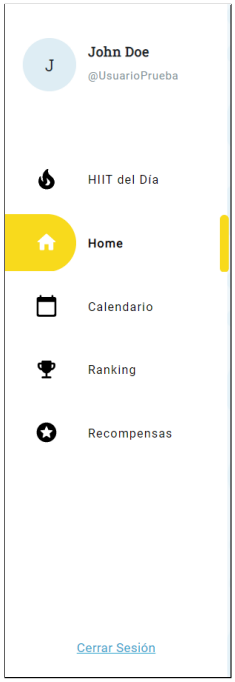

Figura B.3: Menú lateral de selección de página para Usuarios

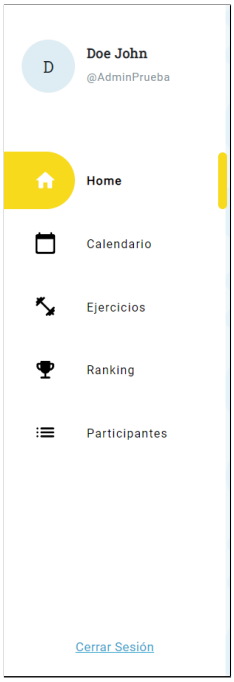

Figura B.4: Menú lateral de selección de página para Administradores

En el caso de estar utilizando la versión móvil, el menú nos aparecerá arriba de la pantalla. Cada uno de los iconos representa las páginas a través de las cuales podemos navegar.

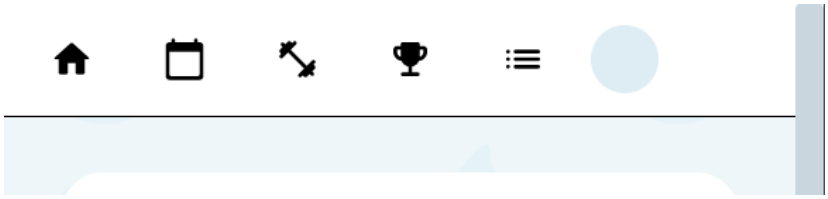

Figura B.5: Menú principal en la versión móvil

Para facilitar encontrar la información deseada al lector se han separado las explicaciones de uso de las páginas en dos categorías: Páginas de Usuario y Páginas de Administrador.

B.3. PÁGINAS DE USUARIO

B.3.1. HIIT del Día

En esta página se van a reproducir los HIITs diarios que toquen realizar. Una vez el juego haya cargado nos aparecerá un botón en mitad de la pantalla para iniciar.

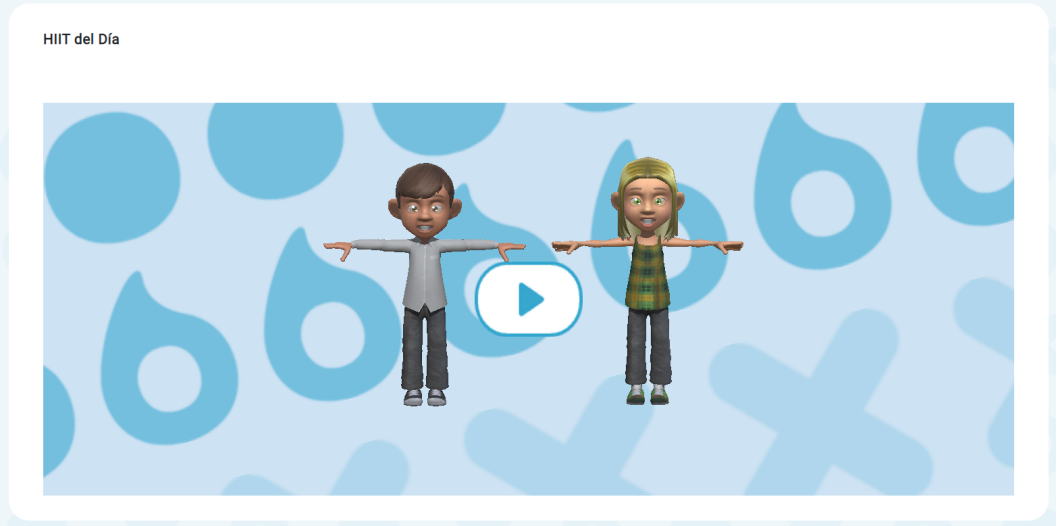

Figura B.6: Botón de inicio del HIIT.

Durante la ejecución tendremos arriba a la derecha disponibles en todo momento dos botones. El primero para poder pausar y reanudar el HIIT en cualquier momento que lo necesitemos. Y el segundo para pasar a Pantalla Completa o regresar a la vista normal.

Figura B.7: Botones de pausa y pantalla completa

Una vez los Avatares explican que ejercicios tocan y como se hacen, comenzarán los ejercicios. Mientras estemos haciendo los ejercicios tendremos a nuestra disposición un temporizador (arriba izquierda) que nos indica los segundo que faltan para cambiar de intensidad. Abajo tendremos también disponible una barra de progreso que indica por cual ejercicio vamos y cuanto falta para terminar.

Una vez hemos finalizado los ejercicios, el HIIT se ha acabado y se recibirá una recompensa, un HEALTHY-HIIT. Esto es un punto que os ayudará a desbloquear recompensas mensuales y también os permitirá subir puestos en la clasificación.

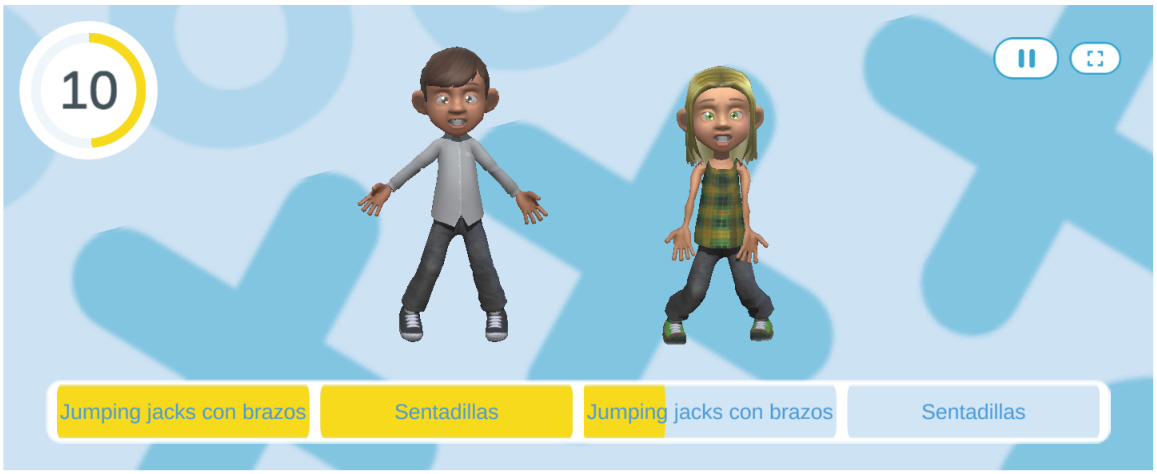

Figura B.8: Pantalla de ejecución del HIIT

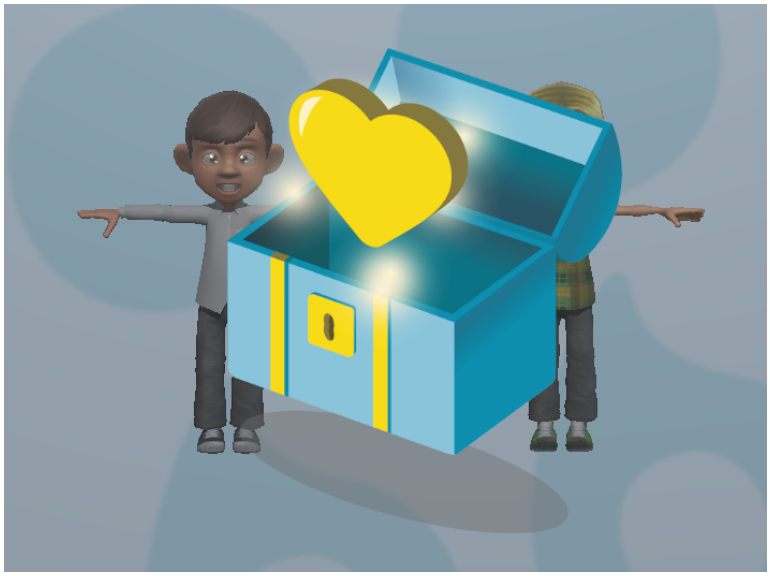

Figura B.9: Momento en el que se recibe la recompensa

B.3.2. Home

Aquí podremos encontrar toda la información que necesitamos de un solo vistazo. Esta página nos informará del HIIT del Día junto con los ejercicios que toca hacer. También nos dirá la planificación que tenemos para esa semana y un pequeño ranking donde aparecerán las tres primeras aulas que más HIITs hayan hecho.

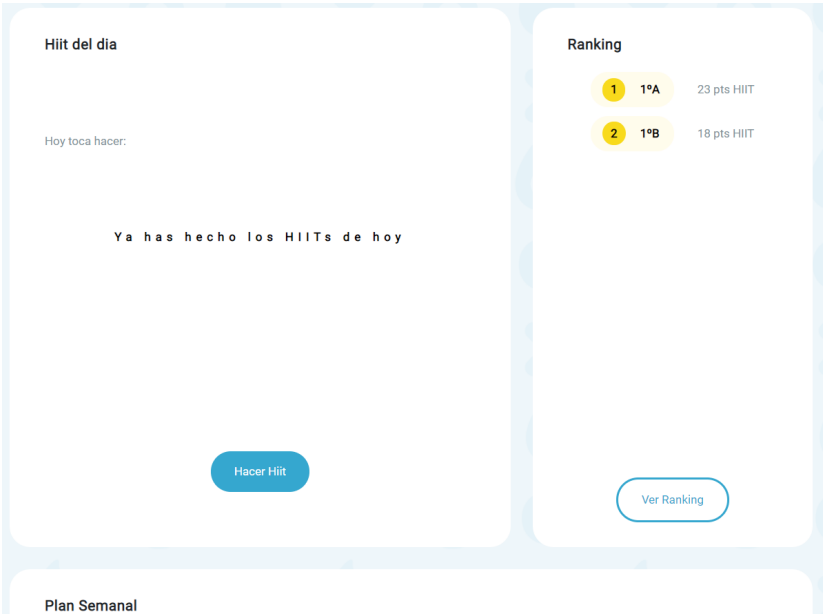

Figura B.10: Página Home de los Usuarios

B.3.3. Calendario

En esta página encontraremos un calendario mensual con la programación para cada uno de los meses. Es posible indagar en el resto de meses haciendo uso de las flechas que aparecen a cada lado del mes.

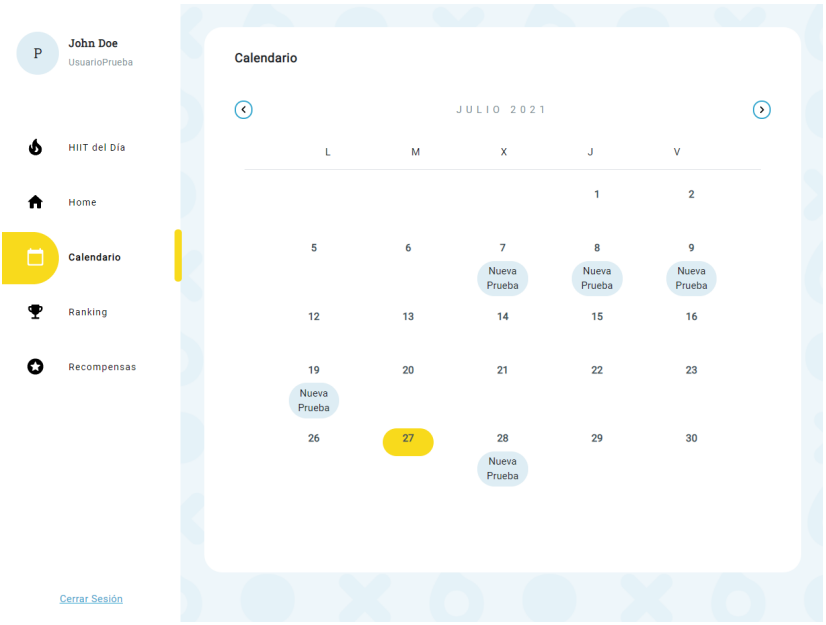

Figura B.11: Página Calendario de los Usuarios

B.3.4. Ranking

Aquí podremos consultar nuestra posición con respecto al resto de aulas. Al contrario del ranking de la página Home, aquí aparecen todas las aulas con sus puntos y posiciones correspondientes.

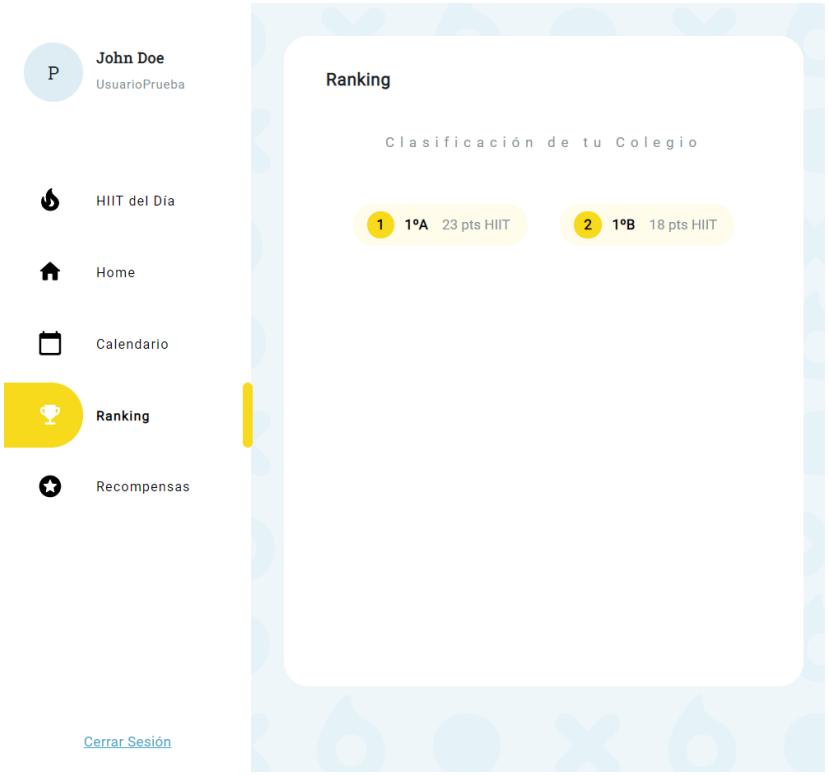

Figura B.12: Página del Ranking de los Usuarios

B.3.5. Recompensas

En la página de las recompensas podremos consultar que recompensas tenemos desbloqueadas y cuanto nos falta para poder desbloquear una. Las recompensas son mensuales por lo que habrá que hacer una cantidad de HIITs para poder conseguirla.

Figura B.13: Página de Recompensas del Usuario

B.4. PÁGINAS DE ADMINISTRADOR

B.4.1. Home

Esta es la página principal que nos va a mostrar información sobre el sistema. Se podrá consultar la planificación semanal establecida, el ranking con las tres mejores aulas de todos los colegios y los últimos HIITs que se han añadido al sistema.

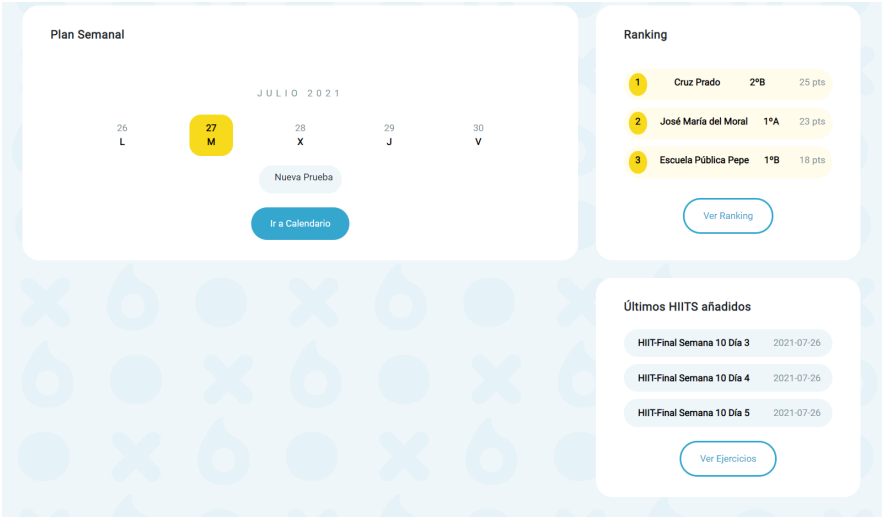

Figura B.14: Página Home de los Administradores

B.4.2. Calendario

En esta página podremos consultar un calendario con la planificación mensual de los HIITs. El calendario nos mostrará el mes actual pero podremos consultar cualquier otro mes diferente haciendo uso de las flechas en la parte superior del calendario.

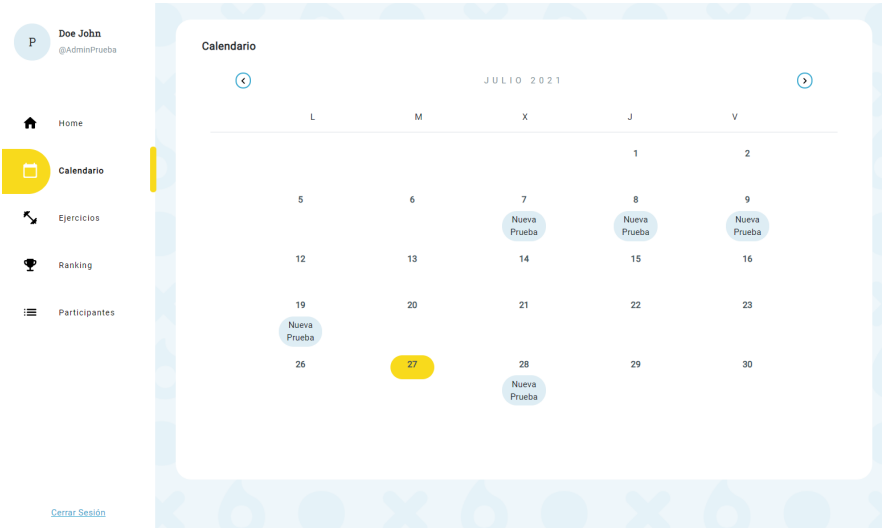

Figura B.15: Página Calendario de los Administradores

B.4.3. Ejercicios

En esta página se va a poder realizar una gestión de los HIITs que hay almacenados y visualizar también los ejercicios que hay creados para los dos avatares

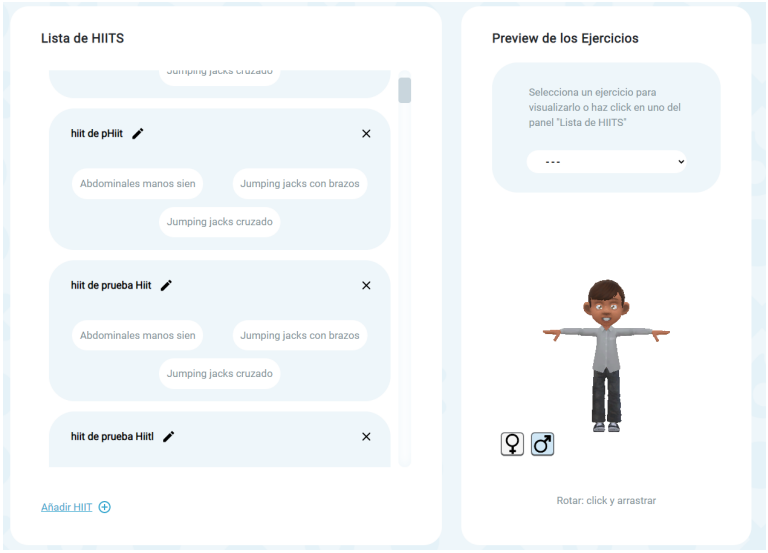

Figura B.16: Página para la gestión de Ejercicios

Si queremos añadir un nuevo HIIT, basta con pulsar en el boton *Añadir HIIT* en la parte de abajo del primer panel. Se nos abrirá un formulario con todos los datos que hay que suministrarle. Una vez estén rellenos los datos, pulsamos en el botón de Crear y ya estaría añadido el nuevo HIIT.

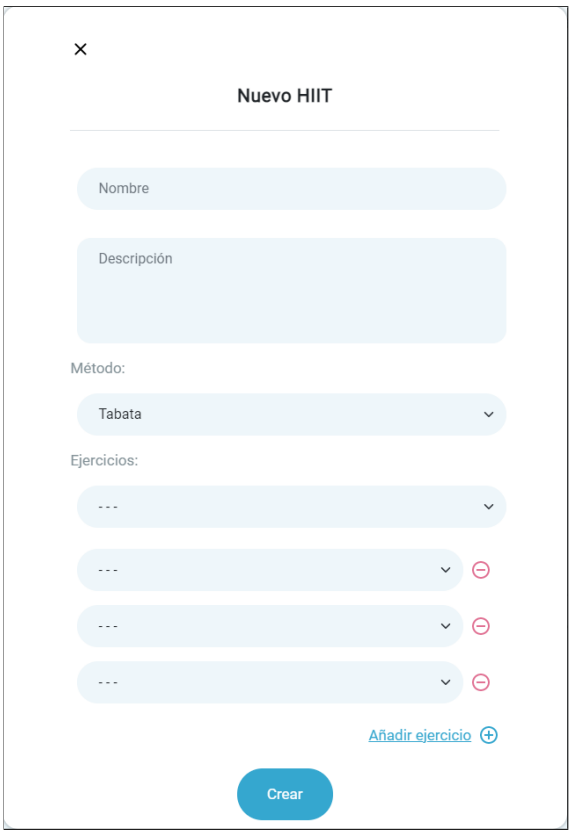

Figura B.17: Formulario para añadir un nuevo HIIT

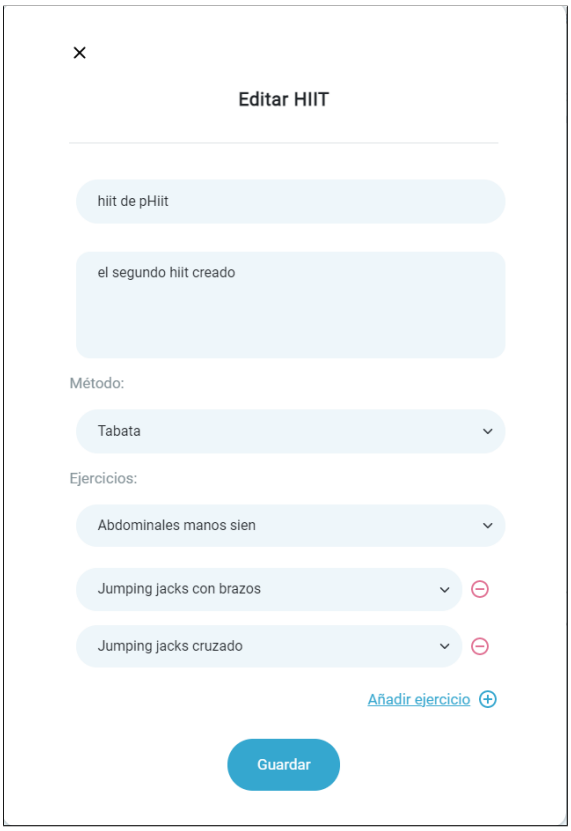

Figura B.18: Formulario para editar un HIIT

Cada HIIT mostrado en la lista nos va a mostrar los ejercicios que contiene y dos iconos: un lápiz y una equis. La equis sirve para eliminar un HIIT del sistema y nos pedirá confirmación de la acción. Para editar un HIIT pulsaremos en el icono del lápiz y se nos abrirá un nuevo formulario donde podremos editar los diferentes campos de información del HIIT. Al finalizar pulsamos en Guardar para fijar los cambios.

Otra de las posibilidades que ofrece esta página es la de poder visualizar los ejercicios para cualquier de los dos Avatares (Chico o Chica). Se nos ofrece una lista desplegable para seleccionar uno de los ejercicios y poder así visualizarlo en el Avatar que esté seleccionado. Los botones de abajo a la izquierda nos permitirán cambiar entre uno y otro. También podremos rotar al Avatar para poder ver el ejercicio desde varios ángulos. Solo hace falta pinchar y arrastrar.

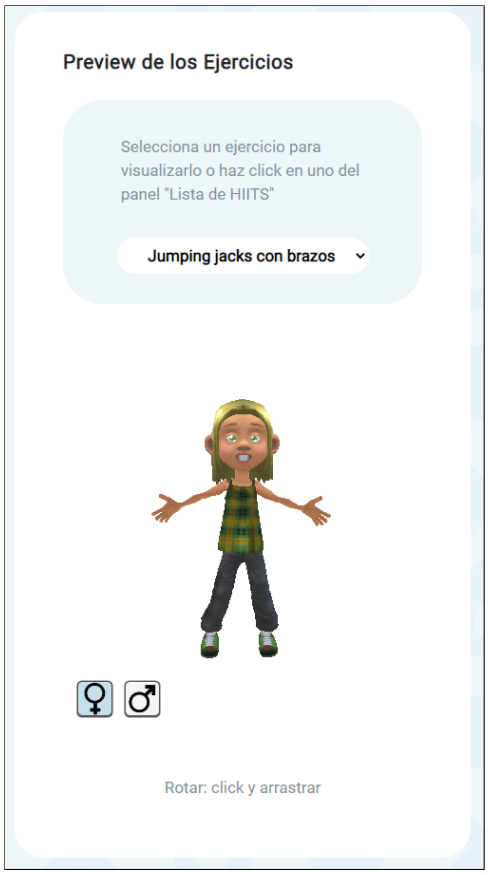

Figura B.19: Preview de los ejercicios con el avatar femenino

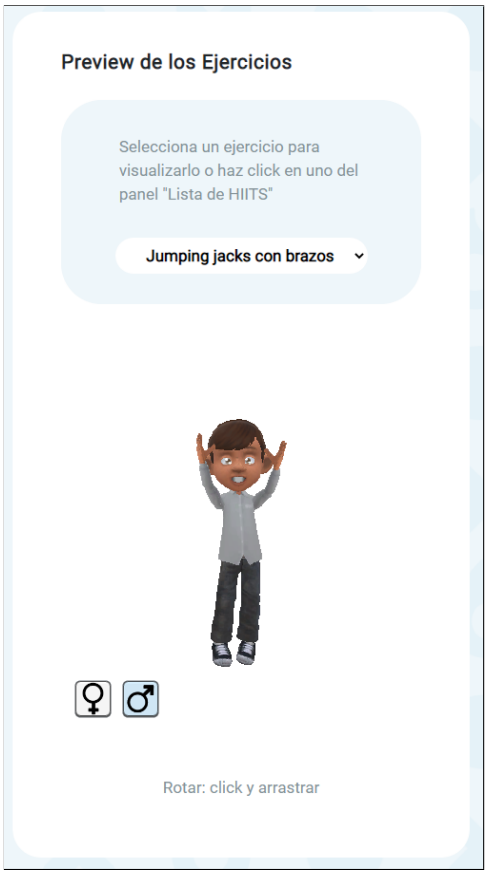

Figura B.20: Preview de los ejercicios con el avatar masculino

B.4.4. Ranking

En esta página se tendrá acceso a un Ranking General donde aparecerán todas las aulas de los colegios participantes ordenadas según los puntos que hayan conseguido. También será posible consultar el ranking de un colegio en particular.

B.4.5. Participantes

En la página de participantes, podremos consultar la información de cualquiera de los colegios que estén participando en el programa y dentro de ellos consultar más en detalle las clases que lo forman. Se podrá visualizar las recompensas que han desbloqueado y los puntos que llevan.

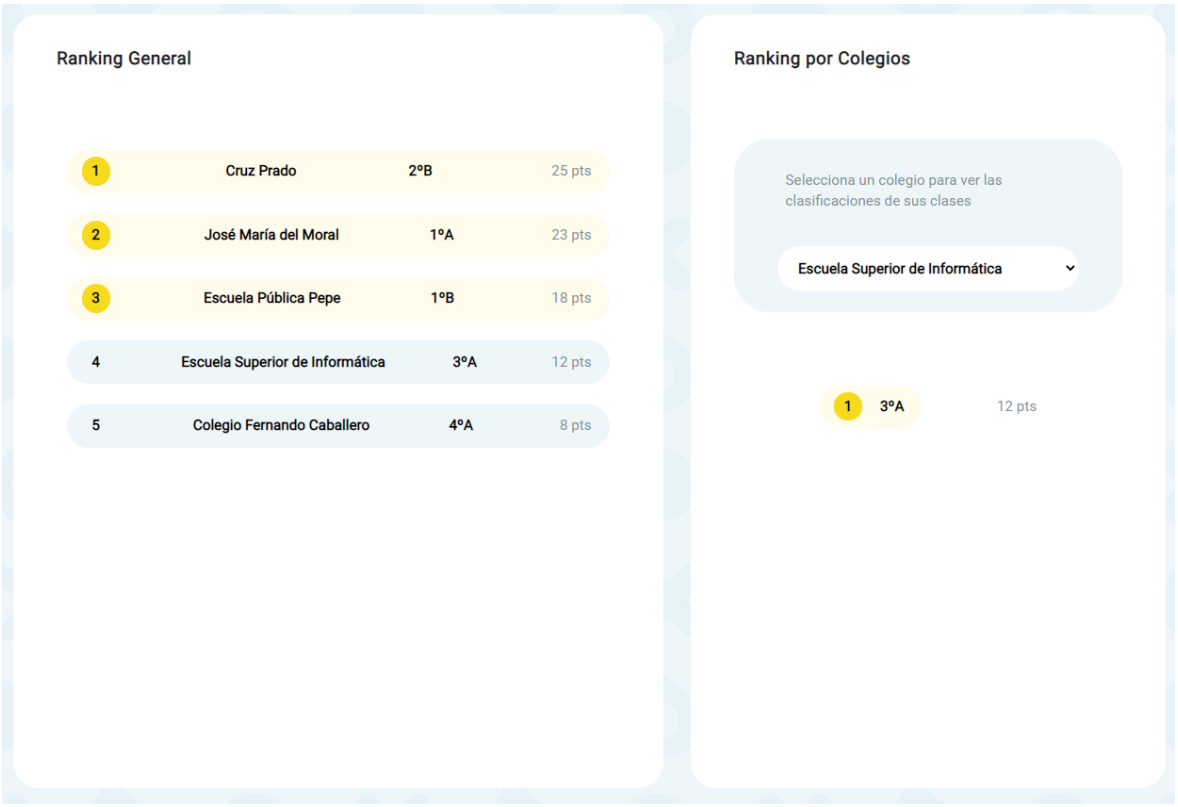

Figura B.21: Página de Ranking para los Administradores

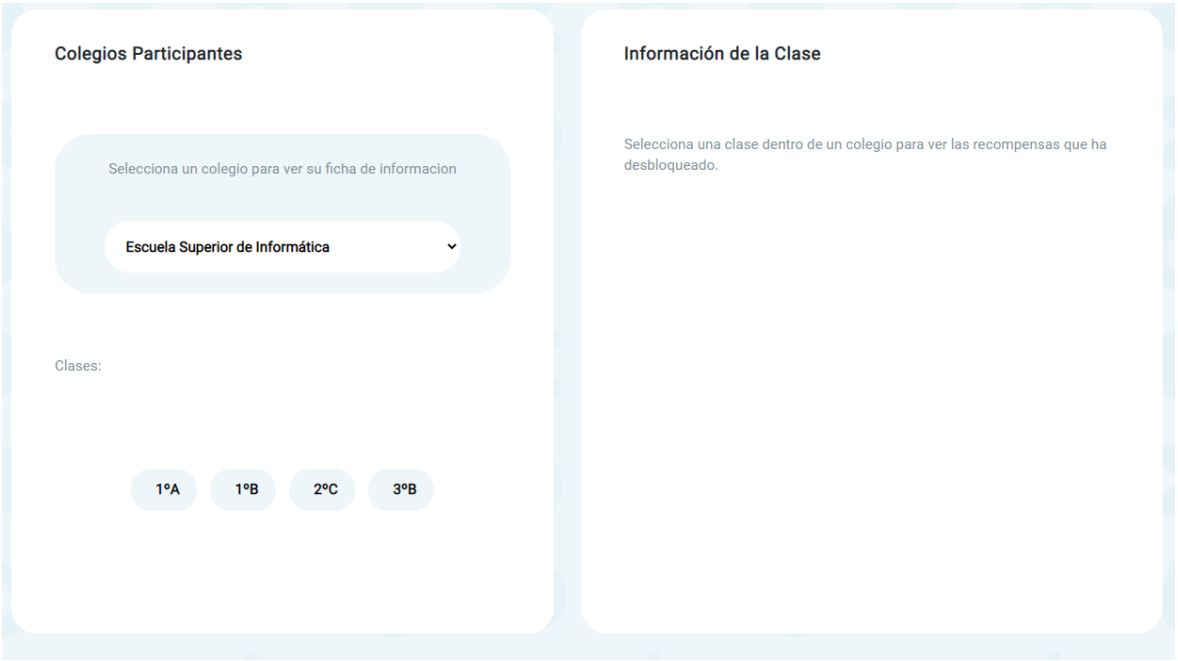

Figura B.22: Página de los Participantes del programa

B.5. RESUMEN

Para finalizar, se recoge un resumen de las páginas webs existentes junto con los iconos que han sido utilizados para representar cada una de ellas. También se expone la información sobre a qué páginas tiene acceso cada tipo de usuario y una breve descripción de cada una.

Tabla B.1: Iconos y accesibilidad a las páginas web

Icono	Nombre	Acceso	Descripción
	Home	Usuario Administrador	Página principal que muestra un resumen de la situación en la plataforma.
	HIIT del Día	Usuario	Es la página donde se reproducirá el HIIT diario.
	Calendario	Usuario Administrador	Página que muestra un calendario con la planificación mensual de las actividades.
	Ejercicios	Administrador	Página para la gestión de HIITs y examinación de ejercicios.
	Ranking	Usuario Administrador	Muestra un ranking de puntuaciones de los distintos colegios.
	Recompensas	Usuario	Página que muestra las recompensas que ha desbloqueado un usuario.
	Participantes	Administrador	Página para consultar información sobre los distintos participantes del programa.

ANEXO C

Diagrama de Clases

En este anexo podemos encontrar la Figura C.1 correspondiente con el diagrama de Clases del módulo de representación 3D.

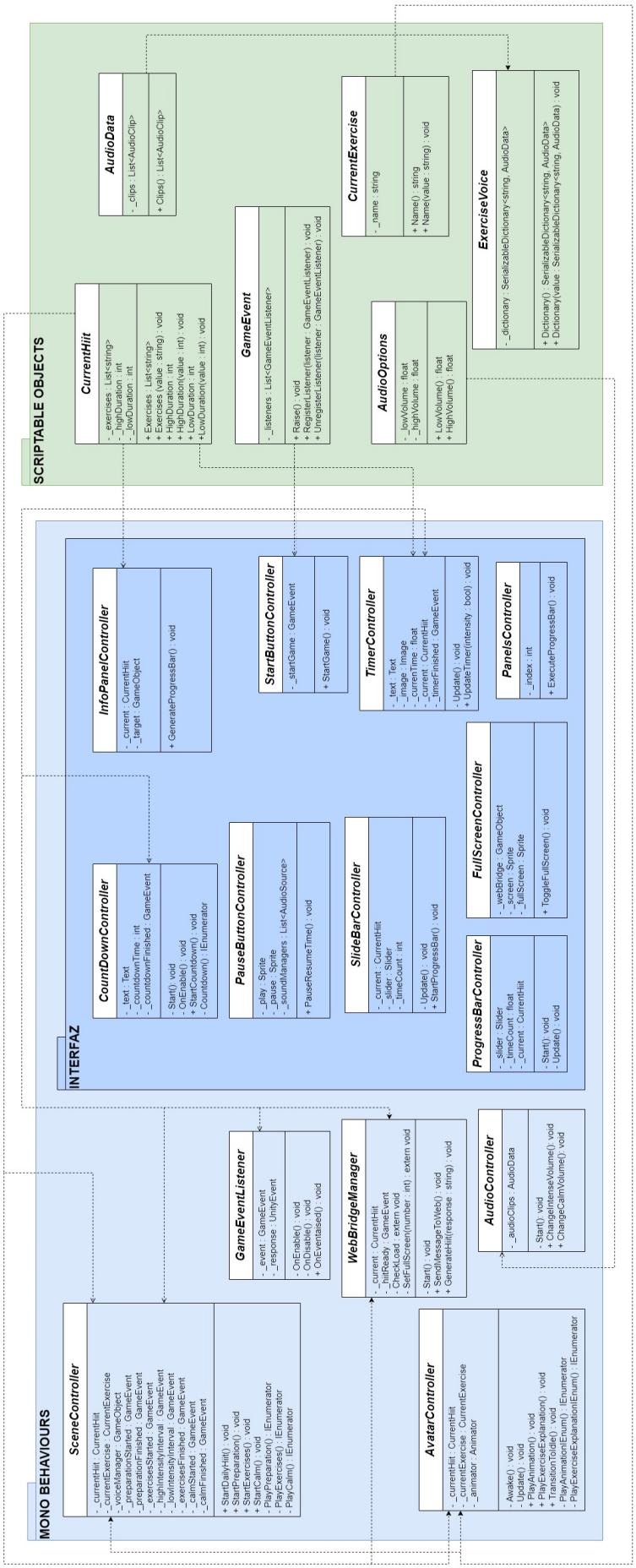

Figura C.1: Diagrama de clases: módulo de representación 3D