ClassBattle

App didáctica para las aulas de educación secundaria.

Autor
Gregorio Coronado Morón

Consultor
Albert Mata Guerra

03/01/2018

Máster Universitario de Desarrollo de Aplicaciones para dispositivos móviles

UOC
Universitat Oberta de Catalunya

A Ana María, luz en la oscuridad.
A Raquel y Miriam, maestras inagotables.

A Albert, por sus consejos y apoyo.

FICHA DEL TRABAJO FINAL

Título del trabajo:	ClassBattle. App didáctica para las aulas de educación secundaria
Nombre del autor:	Gregorio Coronado Morón
Nombre del consultor:	Albert Mata Guerra
Fecha de entrega (mm/aaaa):	01/2018
Titulación:	*Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles*

Resumen del Trabajo (máximo 250 palabras):

El presente trabajo constituye la documentación que recoge todos los aspectos de estrategias, especificación, análisis, diseño, implantación y pruebas de una aplicación didáctica para teléfonos móviles cuyo uso esta destinado a las aulas de educación secundaria. Se centra en un juego de preguntas y respuestas colaborativo y competitivo al mismo tiempo que pretende mejorar otras alternativas disponibles en el mercado.

Abstract (in English, 250 words or less):

This paper is the documentation that covers all aspects of strategies, specification, analysis, design, implementation and testing of a didactic application for mobile phones whose use is intended for secondary education classrooms. It focuses on collaborative and competitive question-and-answer play while at the same time aiming to improve other alternatives available in the market.

Palabras clave (entre 4 y 8):

smartphone, didáctica, juego, colaborativo, competitivo, preguntas y respuestas

Índice

1 Introducción 1

1.1 Contexto y justificación del trabajo 1

1.2 Objetivos del trabajo 3

1.3 Enfoque y método elegido 4

1.4 Planificación del trabajo 7

1.4.1 Recursos Hardware 7

1.4.2 Recursos Software 7

1.4.3 Tareas a Desarrollar 7

1.5 Breve sumario de productos obtenidos 11

1.6 Breve descripción de los otros capítulos de la memoria 11

2 Fase de Diseño 12

2.1 Fases del DCU 12

2.1.1 Usuarios y contextos de uso 12

2.1.2 Diseño conceptual 13

2.1.3 Prototipado 18

2.1.4 Evaluación 21

2.2 Definición de los Casos de Uso 22

2.3 Diseño de la arquitectura 27

2.3.1 Diagrama de Base de Datos UML 27

2.3.2 Diagrama de Clases UML 29

2.3.3 Diagrama MVC 30

2.4 Revisión de la Temporalización de la Fase 32

3 Fase de Implementación 33

3.1 Creación y Gestión de la Base de Datos 33

3.1.1 Diseño Físico 33

3.1.2 Revisión del Diagrama de Base de Datos UML 35

3.2 Desarrollo de ClassBattle Edición del Profesor 36

3.2.1 Revisión del Diagrama del Modelo UML 36

3.2.2 Vistas y Controladores 38

3.3 Desarrollo de ClassBattle Edición de Estudiante **39**
3.3.1 Revisión del Diagrama del Modelo UML 39
3.3.2 Vistas y Controladores 40
3.4 Pruebas **40**
3.5 Revisión de la Temporalización de la Fase **42**
4. Conclusiones **43**
5. Glosario **45**
6. Bibliografía **46**
7. Anexos **48**
7.1 Hoja de Evaluación del Prototipo **48**
7.2 Hoja de Registro de Fallos de las App **48**
7.3 Puesta en Marcha en MacOS X **50**
7.3.1 Aplicaciones 50
7.3.2 Base de Datos 53
Manual de Usuario **54**

Lista de Figuras

FIGURA 1. DIFERENCIAS ENTRE KAHOOT Y QUIZIZZ [2]. 2
FIGURA 2. CUOTA DE MERCADO DE SISTEMAS OPERATIVOS DE MÓVILES A NIVEL MUNDIAL DESDE NOVIEMBRE DE 2016 A SEPTIEMBRE DE 2017 SEGÚN NETMARKETSHARE [12]. 5
FIGURA 3. DIAGRAMAS DE GANTT DE LAS TAREAS DEL PROYECTO 10
FIGURA 4. MAPA DE NAVEGACIÓN DE CLASSBATTLE TEACHER EDITION 16
FIGURA 5. MAPA DE NAVEGACIÓN DE CLASSBATTLE STUDENT EDITION 16
FIGURA 6. LOGOTIPOS DE LAS APPS CLASSBATTLE TEACHER EDITION Y STUDENT EDITION 18
FIGURA 7. PROTOTIPO DE PANTALLA DE AUTENTICACIÓN 19
FIGURA 8. PROTOTIPO DE PANTALLA DE ENTRAR A JUEGO 19
FIGURA 9. PROTOTIPO DE PANTALLA DE SALA DE ESPERA 20
FIGURA 10. PROTOTIPO DE PANTALLA DE TABLERO 20
FIGURA 11. PROTOTIPO DE PANTALLA DE PULSADOR 21
FIGURA 12. DIAGRAMA DE CASOS DE USO DE CLASSBATTLE (AMBAS EDICIONES) 22
FIGURA 13. DIAGRAMA UML DE LA BASE DE DATOS 28
FIGURA 14. DIAGRAMA UML DE CLASSBATTLE TEACHER EDITION 29
FIGURA 15. DIAGRAMA UML DE CLASSBATTLE STUDENT EDITION 30
FIGURA 16. DIAGRAMA BÁSICO MVC 31
FIGURA 17. DIAGRAMA BÁSICO MVC DE LA PÁGINA DE AUTENTIFICACIÓN 31
FIGURA 18. PLANIFICACIÓN FINAL Y REAL DE LA FASE DE DISEÑO 32
FIGURA 19. ESQUEMA DE EJEMPLO DEL OBJETO GAMERS QUE ALMACENA LOS JUGADORES DE CADA JUEGO 35
FIGURA 20. DIAGRAMA UML DE LA BASE DE DATOS REVISADO. 36
FIGURA 21. DIAGRAMA UML DEL MODELO DE DATOS DE CLASSBATTLE TEACHER EDITION REVISADO. 37
FIGURA 22. DIAGRAMA UML DEL MODELO DE DATOS DE CLASSBATTLE STUDENT EDITION REVISADO. 39
FIGURA 23. PLANIFICACIÓN FINAL Y REAL DE LA FASE DE IMPLEMENTACIÓN 42
FIGURA 24. PÁGINA WEB DE NODE.JS 51
FIGURA 25. APPSTORE DE MACOS SIERRA CON LA PÁGINA DE DESCARGA DE XCODE. 51
FIGURA 26. EJECUCIÓN DE CLASSBATTLE STUDENT EDITION EN UN NAVEGADOR WEB. 52
FIGURA 27. EJECUCIÓN DE CLASSBATTLE TEACHER EDITION EN UN EMULADOR IOS. 53

1 Introducción

1.1 Contexto y justificación del trabajo

El disponer de aparatos de comunicación fácilmente portables, con colores y música atractiva que responden de manera interactiva, fácil e intuitiva a sonidos, imágenes o el tacto, hace que no sea de extrañar que la media de edad en el uso de dispositivos móviles (smartphones, tabletas, etc.) sea cada vez menor.

Al margen de los problemas que puedan surgir por la mala praxis en el uso de estos dispositivos en edades tan tempranas como el cyberbulling o el grooming, suponen un autentico reto para los educadores. Los procesos de enseñanza-aprendizaje clásicos basados en memorizar se vuelven poco efectivos en muchos casos ante un público más que acostumbrado a información audiovisual e interactiva.

Si bien es cierto que las Tecnologías de la Información y Comunicación, en adelante TIC, juegan un papel muy importante en nuestra sociedad desde décadas, su inclusión en las aulas sufre un proceso más lento debido a la necesidad de dotación tanto material como de formación del profesorado.

Partiendo del dato de que “En 2015, un 98% de los jóvenes de 10 a 14 años contaba ya con un teléfono de última generación con conexión a Internet” [3] este Trabajo Fin de Máster, en adelante TFM, pretende desarrollar una app basada en un juego de preguntas y respuestas en equipo que permite a los estudiantes participar en él, así como visualizar su avance desde cualquier smartphone adherido. Dado que la dotación material ya la posee el alumnado, la convierten en una herramienta TIC de fácil implantación para su aplicación en el aula.

Si bien la idea de esta aplicación no es novedosa, ya que existen otras tales como Kahoot [4] o Quizizz [5] a nivel individual o Quizalize [6] o Quizlet Live [7] por equipos, ClassBattle introduce ciertos aspectos que pretenden mejorar la sencillez y efectividad, así como la cooperación y competitividad de la actividad en el aula.

Por poner un ejemplo, tanto en Kahoot como en Quizziz un profesor debe preparar un juego inicialmente con una serie de preguntas y respuestas. Los alumnos pueden incluirse en dicho juego a través de la app y comenzar a responder.

En ambos, el dispositivo del profesor debe conectarse a un proyector. En Kahoot el proyector mostrará las preguntas y las posibles repuestas. Desde el smartphone los alumnos/as deberán elegir una respuesta basada en un símbolo adherido a cada una de ellas. La sesión de aula

es guiada por el profesor ya que todos responden a las mismas preguntas al mismo tiempo.

Por el contrario, en Quizziz el proyector solo muestra la clasificación de los jugadores. Los alumnos/as reciben preguntas aleatorias y deben elegir las posibles respuestas directamente desde su smartphone. El profesor no guía la actividad, es autónoma.

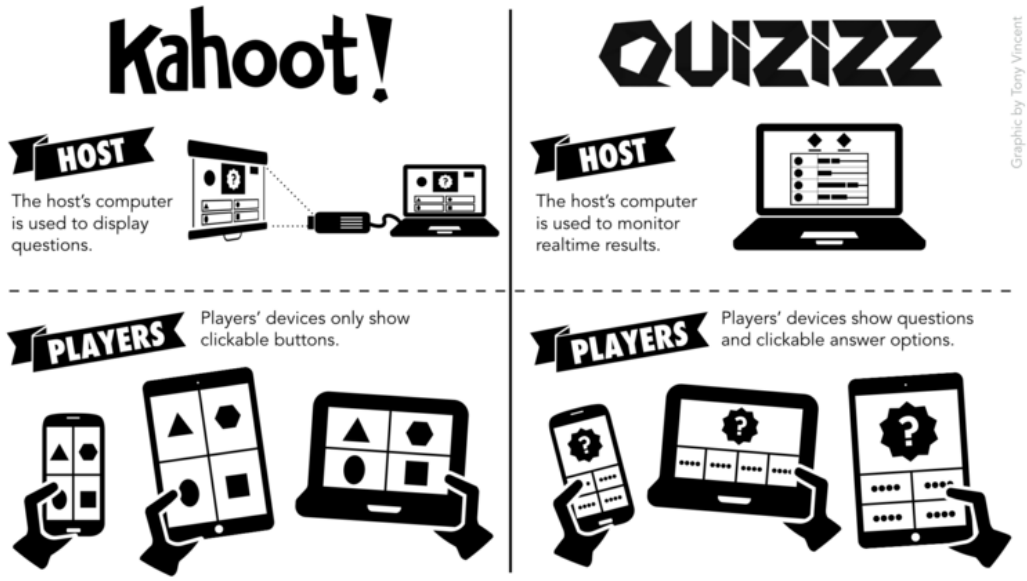

Figura 1. Diferencias entre Kahoot y Quizizz [2].

ClassBattle se trata de un juego sobre un tablero formado por un grupo de casillas. Pueden jugar desde 2 a 4 grupos de 1 o más jugadores en una misma partida desde su smartphone. Todos los grupos disponen de un ficha sobre dicho tablero y todos los jugadores adheridos ven el tablero y las puntuaciones de cada uno.

Una diferencia importante con las apps anteriores es que el juego no contiene previamente las preguntas y respuestas. Estas son lanzadas de viva voz por el profesor. Al no existir un grupo de preguntas predeterminado expuestas en orden (Kahoot) o aleatoriamente (Quizizz) el docente puede guiar el proceso de enseñanza-aprendizaje mucho más abiertamente. Si detecta alguna laguna puede reincidir con preguntas sobre ese mismo tema.

Al realizar la pregunta desde su dispositivo permite que los jugadores puedan pulsar un botón para pedir la palabra y responder. El primero que pulse, dirá la respuesta. Si es correcta, desde su terminal podrá elegir una casilla adyacente a su ficha y "conquistarla". Si falla se producirá rebote, hasta que algún grupo sepa la respuesta o la cambie el profesor.

Al jugar en grupos se favorece la cooperación entre sus miembros aunque también se propicia la competitividad entre distintos grupos.

Cuando un jugador conquista una casilla y la abandona esta puede ser conquistada por otro grupo.

El juego finalizará cuando el profesor lo dictamine (finalice la sesión del aula, por ejemplo). Una vez acabada, ganará aquel que tenga más casillas conquistadas.

Por tanto, como ventajas se pueden destacar que:

- No es necesario proyectar ninguna imagen en el aula, ya que toda la información está en los dispositivos móviles de los jugadores. Elimina el problema de falta de dotación existente en muchos centros.
- Al no existir un guión programado internamente en la app (lo que no quiere decir que el profesor no posea su guión externo) le permite dirigir las preguntas hacia aquellos temas sobre los que es más interesante centrarse y que puede que no tuviese planificados previamente.
- Favorece la cooperación entre los miembros del grupo, provocando un trabajo colaborativo para encontrar la respuesta correcta.
- La gamificación de la actividad favorece la competitividad y el interés por la misma.

Como desventajas se puede recalcar que:

- Existen centros donde el uso de dispositivos móviles está totalmente prohibido
- Se requiere que el alumnado disponga de su propio teléfono móvil que en ocasiones no siempre es así por muchos motivos.

1.2 Objetivos del trabajo

- Crear una aplicación para smartphone denominada ClassBattle fácil de usar, capaz de gestionar una actividad lúdico-educativa en el aula, que favorezca el interés por la materia y la cooperación del alumnado, minimizando a su vez los recursos del aula necesarios.
- La elaboración del presente documento donde se recogen objetivos, requisitos y cualquier aspecto importante relacionado con el análisis, diseño y/o desarrollo de la app propuesta.
- La realización de una presentación virtual por medio de diapositivas y video con la descripción del proyecto y demostración del funcionamiento de los productos obtenidos.
- Dicha aplicación pretende abarcar el mayor número de usuarios posible, por lo que se establece la línea para ser desarrollada para teléfonos móviles con distintos sistemas operativos, siempre y cuando el hardware permita una ejecución fluida de la app. Con todo ello, este TFM solo abarca la creación final y pruebas en sistemas iOS por razones de tiempo, aunque se sientan las bases para el desarrollo en otras plataformas buscando la menor inversión en su desarrollo posterior.

- La aplicación dispone de dos versiones, una versión del profesor y otra versión del estudiante. Separando la funcionalidad de ambas se pretende buscar la simplicidad.
- En ambas versiones es necesario registrarse con el fin de que el sistema identifique correctamente a los usuarios. Internamente es posible modificar información básica sobre el perfil del usuario autenticado.
- La versión del profesor muestra un listado de los juegos que tiene disponibles, pudiendo entrar en cualquiera de ellos o crear uno nuevo. Cada juego dispone de un identificador único que usan los alumnos para poder unirse a él y no a otro.
- Al entrar en un juego nuevo, la aplicación muestra una pantalla de "sala de espera" y se mantendrá allí hasta que se hayan unido 2 o más jugadores. A partir de ese momento el profesor puede dar comienzo al juego.
- Una vez comenzado se muestra un tablero e información de los jugadores tanto en los dispositivos de los alumnos como en el del profesor. El tablero del profesor dispone de un botón titulado "lanzar pulsadores" que activará una vez haya formulado una pregunta. Esto provocará la aparición de un botón gigante en los dispositivos de los alumnos. Cuando alguno pulse sobre su terminal, aparecerá un mensaje en el del profesor que diga: "Turno de Fulanito ¿Acertó? Sí o No". Si este elige el botón "sí", el terminal que pulsó podrá elegir una casilla adjunta para conquistar. Si por el contrario, eligió "no", se vuelven a activar todos los botones de los jugadores para poder ser pulsados de nuevo.
- El tablero del profesor también dispone de un botón que permite abandonar juego. Al pulsarlo este dará por finalizada la sesión y ya no se podrá volver a jugar.
- Desde cada terminal se pueden visualizar las puntuaciones (casillas conquistadas) de cada grupo.

1.3 Enfoque y método elegido

Desde el punto de vista de la Ingeniería del Software la metodología ideal a seguir para el desarrollo de este proyecto, así como el de su futura ampliación, implicaría el seguimiento de una metodología ágil. Sin embargo, las dimensiones reducidas del proyecto a desarrollar así como los diferentes hitos que se marcan para la evaluación de este TFM hacen más viable la utilización de un modelo en cascada, que puede interpretarse como si se tratase de una o dos iteraciones dentro de cualquiera de los métodos de desarrollo ágil si el proyecto continuase su ciclo de vida tras este TFM.

En cuanto al uso de tecnologías, con el fin de que la app sea utilizada en el mayor número de dispositivos y con el menor uso de recursos posible para su desarrollo, se plantea la posibilidad de utilizar un framework que use un conjunto de lenguajes unificados para el desarrollo de un único proyecto en varias plataformas (Write Once, Runs Anywhere).

En la actualidad es posible el uso de frameworks que utilizan tecnologías web o similares destinados al desarrollo de aplicaciones híbridas tales como **PhoneGap** [8] o **Ionic** [9], o nativas por medio de **NativeScript** [10] o **React Native** [11].

Todos ellos poseen puntos fuertes y débiles frente a las otros. La experiencia previa en el uso de tecnologías web en Angular e Ionic y el hecho de que no es estrictamente necesario que la app sea nativa hace que el desarrollo se decante por el uso de este último reduciendo así la curva de aprendizaje necesaria. Si bien es cierto que PhoneGap permite la ejecución en un mayor número de plataformas que Ionic, la mayor parte de los sistemas operativos utilizados en la actualidad se centran en iOS y Android [12] lo que convierte a Ionic en una alternativa más que válida, aunque no por ello exenta de dificultades. Veamos por qué.

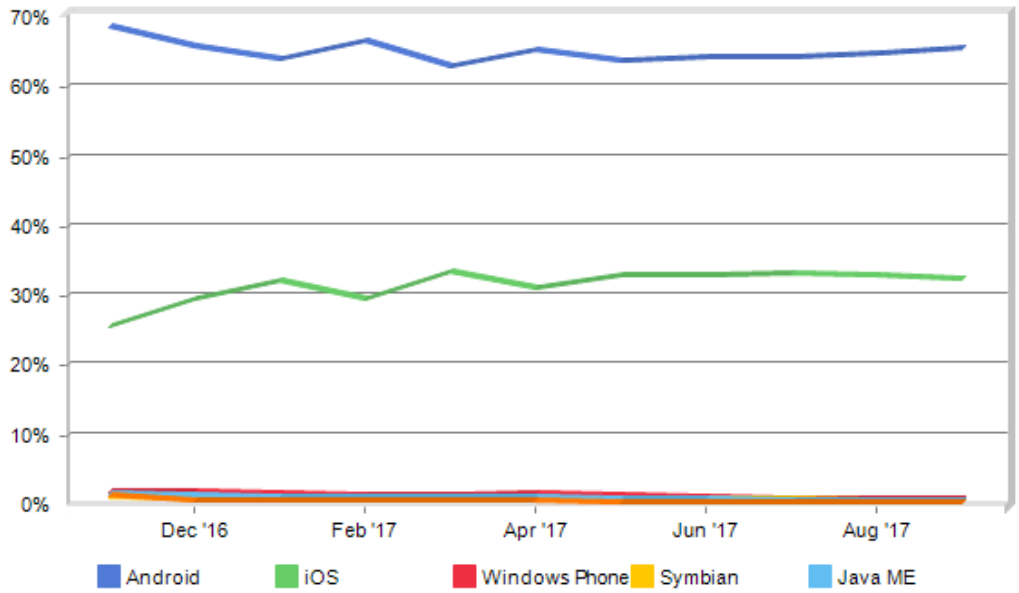

Figura 2. Cuota de mercado de sistemas operativos de móviles a nivel mundial desde noviembre de 2016 a septiembre de 2017 según NetMarketShare [12].

Si se analiza detenidamente los requisitos funcionales de la app es fácil darse cuenta de que es necesario que todos los jugadores se encuentren sincronizados para que el juego sea efectivo. Dicha sincronización se puede realizar vía bluetooth o Wi-Fi por ejemplo. iOS posee un framework específico para ello denominado **MultipeerConnectivity** [13] que permite la comunicación entre dispositivos próximos usando redes Wi-Fi, Wi-Fi direct o Bluetooth, aunque sin embargo sólo puede ser utilizado entre equipos de Apple.
Por otro lado, Google posee una API similar denominada Google Nearby que se divide en 2:

- **Google Nearby Connection [14]**. Para la conexión entre dispositivos cercanos por medio Bluetooth, BLE o Wi-Fi spots. Sólo disponible para dispositivos Android en la actualidad.
- **Google Nearby Messages [15].** Para la conexión entre dispositivos cercanos utilizando Bluetooth, BLE, Wi-Fi y audio ultrasónico aunque requiere de Internet para poder establecer la comunicación. Tiene soporte para iOS y Android en la actualidad.

Si bien esta última parece muy interesante no se encuentra soporte directo para el resto de frameworks comentados, por lo que sería necesario elaborar un plugin específico destinado a ambas plataformas (iOS y Android). La elaboración de dicho plugin supondría una inversión en tiempo que excede del requerido para la elaboración de este TFM.

Dado que al final es necesario almacenar información de los usuarios y el juego en general, se ha optado por el uso de una base de datos en tiempo real para la sincronización y almacenamiento centralizado de la información de la app. Estas bases de datos tienen la capacidad de notificar cambios en los datos a los distintos usuarios conectados a ella, lo que permite a la app que todos los usuarios tengan acceso a la misma información en tiempo real.

Dentro de este tipo de bases de datos podemos destacar **Google Firebase** [16], **RethinkDB** [17] o **MongoDB** [18]. Dado que no se dispone de experiencia previa en ninguna de ellas, se opta por la utilización de Firebase. La elección se centra principalmente por el magnífico soporte que ofrece Google a sus productos, así como la posibilidad futura de utilizar otros servicios asociados como Crash Reporting (para la identificación de fallos), Authentication (Servicio de autenticación de Google) o File Storage (para almacenar ficheros en la nube) directamente sobre Ionic. Además posee un plan gratuito cuando el consumo es pequeño, lo que lo hace ideal para el proyecto presentado en este TFM.

1.4 Planificación del trabajo

1.4.1 Recursos hardware

- iMac 27” (finales 2012) i7 3,4 GHz, 16 GB
- MacBook Pro (principios 2009) Core 2 Duo 2,66 Ghz, 8GB
- MacBook Pro Retina (principios 2015) i5 2,7GHz 8 GB
- iPhone 7 Plus
- iPhone 6 Plus

1.4.2 Recursos software

- MacOS Sierra
- iOS 11
- Xcode 8.3.3
- NodeJS 7.7.2
- Ionic 3.9.2
- Microsoft Office 2011 para Mac
- JustInMind 7.5.0
- WebStorm 2017
- ScreenFlow
- SourceTree 2.4.1
- GIMP
- FinalCut

1.4.3 Tareas a desarrollar

El desarrollo del TFM está planificado para ser ejecutado en 300 horas, divididas en 4 etapas fundamentales:

- Plan de Trabajo
- Diseño
- Implementación
- Entrega Final

A continuación se muestran dichas etapas contemplando adicionalmente las actividades necesarias para ser completadas, así como el número de horas que se prevén para su finalización.

Actividad	Duración (h)
PEC1. Plan de Trabajo	50
Formación sobre TFM	6
Elección del Trabajo	5
Investigación sobre el estado del arte	8
Investigación sobre tecnologías de desarrollo	9
Identificación y planificación de tareas	7
Elaboración de documentación	15
PEC2. Diseño	76
DCU. Usuarios y contextos de uso	8
DCU. Diseño conceptual	8
DCU. Prototipado	22
DCU. Evaluación	11
DT. Definición de casos de uso	9
DT. Diseño de la arquitectura	11
Elaboración de documentación	7
PEC3. Implementación	131
Creación y gestión de la B.D.	15
Desarrollo de ClassBattle. Edición Profesor	52
Desarrollo de ClassBattle. Edición Estudiante	36
Pruebas	18
Elaboración de documentación	10
Entrega Final	43
Empaquetado	6
Finalización de Memoria	9
Presentación	28
Total	**300 h**

De igual forma, se expone el número de horas previstas que se emplearán cada día a lo largo de todo el proyecto. De forma general se prevé dedicar 3 horas al día entre semana y 5 horas al día los fines de semana.

Actividad	Duración (h)
PEC1. Plan de Trabajo	**50**
Formación sobre TFM	6
Elección del Trabajo	5
Investigación sobre el estado del arte	8
Investigación sobre tecnologías de desarrollo	9
Identificación y planificación de tareas	7
Elaboración de documentación	15

PEC2. Diseño	76
DCU. Usuarios y contextos de uso	8
DCU. Diseño conceptual	8
DCU. Prototipado	22
DCU. Evaluación	11
DT. Definición de casos de uso	9
DT. Diseño de la arquitectura	11
Elaboración de documentación	7

PEC3. Implementación	131
Creación y gestión de la B.D.	15
Desarrollo de ClassBattle. Edición Profesor	52
Desarrollo de ClassBattle. Edición Estudiante	36
Pruebas	18
Elaboración de documentación	10

Entrega Final	43
Empaquetado	6
Finalización de Memoria	9
Presentación	28

		Septiembre	Octubre	Noviembre	Diciembre	Enero
		1 al 30	1 al 31	1 al 30	1 al 31	1 al 6
Actividad	**Duración (h)**					
PEC1. Plan de Trabajo	**50**					
PEC2. Diseño	76					
PEC3. Implementación	131					
Entrega Final	43					
	300					

Figura 3. Diagramas de Gantt de las tareas del proyecto

Como puede observarse, la etapa de diseño se basa en el Diseño Centrado en el Usuario (DCU). Aunque en el diagrama se muestra una estructura lineal, esta se plantea iterativa por desconocimiento del número de iteraciones finales.

1.5 Breve sumario de productos obtenidos

- ClassBattle Edición del Profesor para iOS. Permite crear juegos y llevar estadísticas básicas de uso.
- ClassBattle Edición del Estudiante para iOS. Permite incluirse y participar en los juegos creados en la edición del profesor.
- Base de Datos Firebase con la estructura necesaria para el correcto funcionamiento de ClassBattle.
- Documentación propia del TFM
- Presentación del Proyecto.

1.6 Breve descripción de los otros capítulos de la memoria

- **Fase de Diseño**. A lo largo de este apartado se desgranan los diagramas Unified Modeling Language (UML) y decisiones de diseño que sirven para eliminar posibles ambigüedades establecidas en las especificaciones iniciales así como para crear una estructura base sobre la que trabajar en la siguiente fase de desarrollo.
- **Fase de Implementación**. Este apartado describe los pasos seguidos para la codificación final de la aplicación en los entornos establecidos y por medio de las herramientas utilizadas. Se exponen las dificultades encontradas y las soluciones elegidas.
- **Conclusiones**. En este capitulo se desgranan las conclusiones finales obtenidas tras la consecución de este proyecto.
- **Glosario**. Relación de definiciones que permiten aclarar los contenidos expuestos en este documento.
- **Bibliografía**. Listado de referencias utilizadas como apoyo al proyecto.

2 Fase de diseño

Tal y como se ha comentado en apartados anteriores la metodología que sirve de guía en el diseño de nuestro proyecto es el Diseño Centrado en el Usuario (DCU). Por esta razón abordaremos cada una de las fases que componen esta estrategia (usuarios y contextos de uso, diseño conceptual, prototipado y evaluación) sin por ello dejar de lado el diseño técnico (casos de uso y arquitectura de la aplicación).

2.1 Fases del DCU

2.1.1 Usuarios y contextos de uso

Usuario 1

Nombre: Gregorio
Edad: 41 años
Profesión: Profesor

Descripción de la persona

Gregorio es una persona casada y con 2 hijas de 4 y 7 años que acaba de mudarse para este curso académico a un pueblo de la serranía de Málaga de no más de 20.000 habitantes. Trabaja de lunes a viernes de 8.00 a 15.00 en el único Instituto de Educación Secundaria del pueblo y pedanías aledañas. El centro, de reducidas dimensiones, no dispone de grandes inversiones tecnológicas. La situación geográfica del mismo y la del pueblo en general hace que la conexión cableada a Internet sea deficiente, sin embargo, curiosamente se dispone de cobertura 4G. Se trata de una persona muy despistada y habituada al uso de tecnologías que se preocupa por encontrar vías de hacer que los contenidos sean más atractivos al alumnado.

Usuario 2

Nombre: Alicia
Edad: 16 años
Profesión: Estudiante

Descripción de la persona

Alicia vive y estudia en el pueblo en el que Gregorio trabaja. De hecho es alumna suya. Vive con sus padres y dos hermanos a los que ayuda en las tareas del campo. Aunque es inteligente y trabajadora no obtiene buenas calificaciones principalmente porque no dispone de mucho tiempo por las tardes para estudiar en casa. Como la inmensa mayoría

de los adolescentes en España, dispone de un smartphone con conexión a Internet.

Usuario 3

Nombre: Antonio
Edad: 15 años
Profesión: Estudiante

Descripción de la persona

Antonio también es alumno de Gregorio. Vive en un pueblo cercano al del Instituto al que acude cada día entre semana. Es hijo único y vive con sus padres. Es un buen estudiante que suele obtener calificaciones altas. Al igual que Alicia dispone de un smartphone con conexión a Internet y le encanta hacer uso de él. Adora los videojuegos y siempre que puede pasa su tiempo de ocio usando la consola en casa o su móvil.

Usuario 4

Nombre: Luis
Edad: 16 años
Profesión: Estudiante

Descripción de la persona

Luis es el menor de tres hermanos. Todos viven en casa con sus tías. Al igual que Alicia y Antonio, acude cada día al instituto y es alumno de Gregorio. Suele ser un estudiante que aunque no es disruptivo, tiene una falta de motivación completa por los estudios. Aunque el quiere abandonarlos se siente forzado por su familia a continuar. Sus calificaciones son bajas y no suele trabajar en clase. Por las tardes suele salir a la calle con sus amigos y pasarlas enteras sin hacer nada de provecho. Su situación económica y la de su familia no le permite disponer de un smartphone y mucho menos con una tarifa con conexión a Internet.

2.1.2 Diseño conceptual

2.1.2.1 Escenarios

Escenario 1

Después de impartir una unidad de trabajo es necesario repasar los conceptos estudiados. Para ello Gregorio prepara una actividad con ClassBattle eligiendo preguntas que reflejen los conocimientos

adquiridos por sus alumnos y alumnas. Una vez que llega a la clase, realiza cuatro grupos cada uno de ellos con un smartphone con conexión a Internet de cualquiera de los integrantes del juego. El encargado de utilizarlo siempre será aquel alumno/a con mayor dificultad en la materia y es escogido por Gregorio para cada grupo. En concreto en uno de ellos maneja el terminal Alicia y en otro Luis que es compañero de equipo junto con Antonio.

El mero hecho de enfrentarse con sus compañeros y realizar una actividad lúdica motiva a todos los participantes.

Gregorio accede a la edición profesor de la app y crea un juego online. Esta le proporciona un ID del juego que usarán los alumnos para participar en él y pasa a una pantalla de sala de espera donde visualizará todos los jugadores que se van uniendo a la partida.

Por el contrario los alumnos desde su terminal acceden a la versión de estudiante e incluyen un nombre para el juego y el gameID proporcionado por Gregorio. Esto les lleva a una pantalla donde se ven todos los participantes que se están uniendo al juego. Cuando Gregorio observa que todos están conectados pulsa un botón desde su terminal y comienza la partida.

Automáticamente tanto el profesor como los alumnos/as ven un tablero y una ficha que los representa situada sobre él. Junto al tablero, existe una puntuación asociada a cada grupo.

En base a las preguntas que posee Gregorio, lanza la primera de ellas y una vez que está en el aire, pulsa un botón en su terminal que activa unos pulsadores virtuales en los teléfonos de los jugadores/as.

Todos los grupos buscan la respuesta. El grupo de Antonio y Luis la encuentran, aunque el de Alicia también. Ambos grupos activan sus pulsadores pero Antonio lo hace antes que Alicia por lo que se desactiva el del resto de jugadores menos el suyo. Dice la respuesta y acierta, por lo que Gregorio le permite que mueva ficha desde su terminal. Al avanzar conquista una casilla del tablero.

Antonio, que no había prestado mucha atención durante las clases se queda con la idea de la pregunta y la respuesta. Siente la ayuda del grupo y la motivación por ganar. Alicia por otro lado, aunque no ha ganado esta vez, siente la necesidad de volver a responder correctamente a la mayor brevedad posible.

Escenario 2

Durante el desarrollo del juego, Gregorio realiza una pregunta que nadie sabe contestar y formula una nueva relacionada con la anterior. A partir de esa pregunta, ha podido detectar una laguna en sus alumnos/as y redirigirlos para que encuentren la solución. Asimismo, el hecho de que

las preguntas no sean elegidas al azar o completamente predeterminadas ayuda a Gregorio a que se entiendan mejor los conceptos dentro del juego.

Escenario 3

Durante el desarrollo del juego, el teléfono de Antonio y Luis se bloquea. Para continuar reinician el terminal y al entrar a la app introducen el email y contraseña, así como el ID del juego que utilizaron la vez anterior. Dado que se identifican correctamente ingresan nuevamente al juego pudiendo incorporarse en el punto actual del juego.

2.1.2.2 Mapas de navegación

A partir de los escenarios contemplados anteriormente podemos establecer el orden de navegación y funcionalidad básica de las distintas pantallas con las que interactuarán los usuarios de ambas aplicaciones.

A continuación se muestran los mapas de navegación de ClassBattle Teacher Edition y Student Edition respectivamente. Se puede observar que hay algunos nodos coloreados en azul. Se han destacado así para identificar aquellas pantallas que salvo modificaciones mínimas, son muy parecidas o idénticas.

Aunque no se refleja en los diagramas, con el fin de evitar un exceso de líneas que dificulten la legibilidad del mapa, **existe un enlace desde cualquier nodo del árbol hasta la pantalla de autentificación** en el caso de que la sesión de usuario haya caducado.

En ambos casos se ha buscado la simplicidad extrema, eliminando aquellos pasos innecesarios que puedan confundir o retrasar la ejecución de cualquier acción del usuario. La simplicidad es un objetivo principal con el fin de atraer profesorado no muy dado al uso de nuevas tecnologías.

Por esta razón, a la hora de elegir la forma de navegar entre las distintas pantallas se ha optado por evitar una página principal desde la que se tenga acceso a todas las opciones del programa. La idea se ha centrado en guiar al usuario por las distintas partes según las necesidades que le vayan surgiendo. Esto facilita su uso y evita confusiones a la hora de utilizarse.

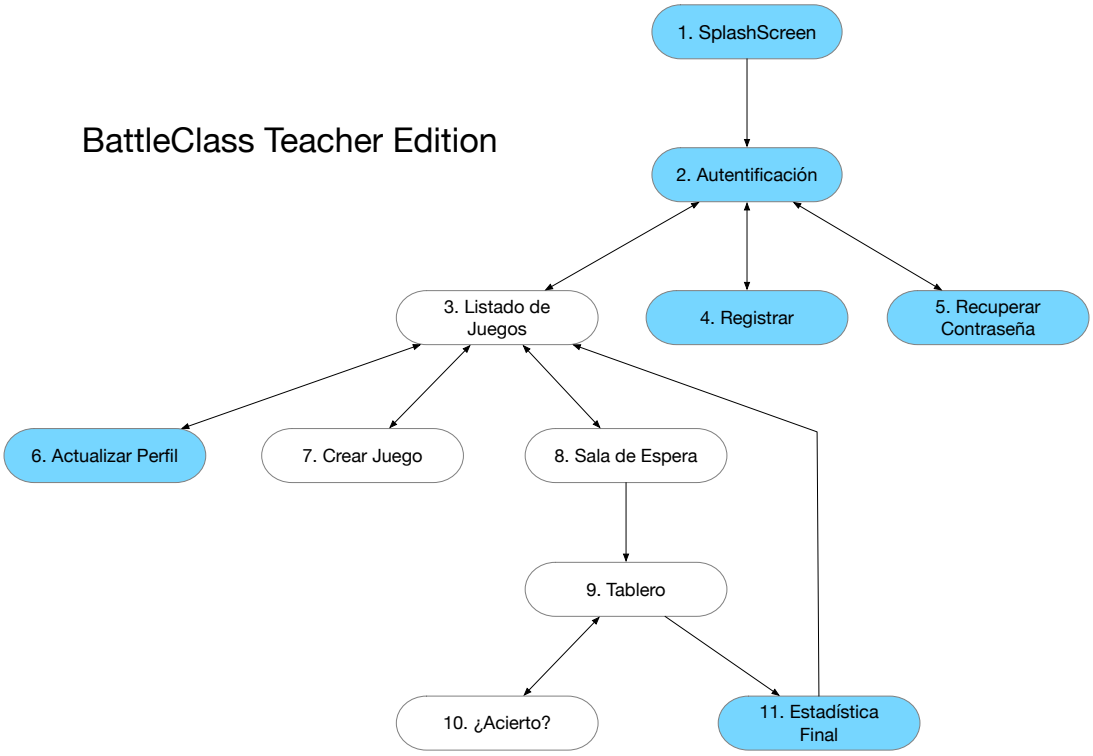

Figura 4. Mapa de navegación de ClassBattle Teacher Edition

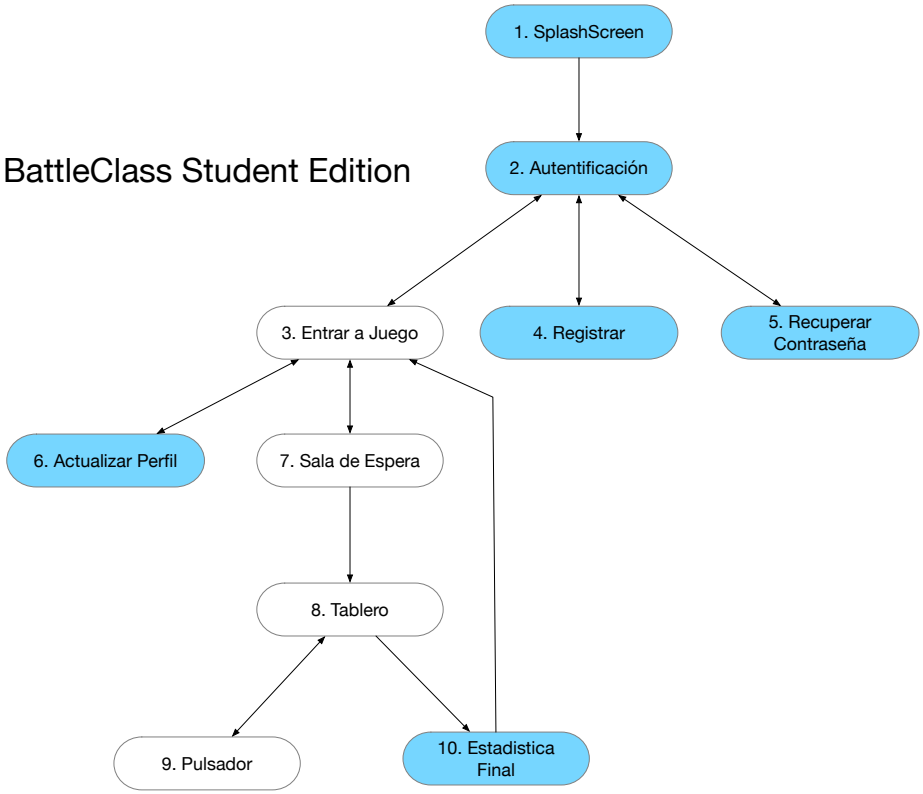

Figura 5. Mapa de navegación de ClassBattle Student Edition

Una vez que se ha visualizado la Splash Screen (1) de la app y que sirve para entretener al usuario mientras carga, este debe autenticarse (2) para ser identificado en el sistema. Si no posee ninguna cuenta siempre tiene la opción de registrarse (4) o si ha olvidado la contraseña puede recuperarla (5).

Una vez identificado el usuario la navegación será diferente dependiendo de si se trata de un profesor o de un estudiante. Si en primer lugar analizamos al estudiante, este al entrar accederá a una pantalla (3) desde la cual podrá escribir los datos de un juego al que desee unirse o acceder a una página de actualización (6) sobre datos básicos de su perfil (contraseña o nombre). En el caso de querer acceder a un juego, si los datos son correctos podrá visualizar una página de sala de espera (7) donde podrá visualizar los jugadores que se van uniendo a la partida. No dispone de ningún botón y sólo avanzará a la pantalla siguiente (8) cuando el profesor desde su aplicación lo permita.

Una vez iniciado el juego el estudiante se encontrará con un tablero (8). Cada vez que el profesor lance una pregunta, se abrirá una pantalla con un pulsador (9) para activarlo en caso de saber la respuesta. Si el profesor determina que la respuesta a su pregunta ha sido correcta se volverá a la pantalla del tablero (8) y podrá escoger una casilla que conquistar. Si no, se volverán a activar los pulsadores de todos los jugadores. Si estando en la pantalla 9 algún otro estudiante acierta la pregunta primero, se pasará a la pantalla 8 pero no se podrá conquistar ninguna casilla. Tan sólo podrá ver el movimiento que realiza su contrincante.

Una vez que el profesor o el estudiante lo determine el juego terminará, pasando a una pantalla de estadísticas (10). Si la decisión es del alumno y el profesor aun no ha cerrado el juego puede volver a incorporarse siguiendo el proceso desde la pantalla de entrar a juego (3).

En el caso de la edición del profesor la navegación es ligeramente distinta aunque comparte muchas pantallas con el mismo contenido. Al entrar accederá a una pantalla (3) desde la cual visualizará todos los juegos que tiene dados de alta. Desde allí podrá acceder a una pantalla para crear uno nuevo (7), editar su perfil, al igual que el estudiante (6), o seleccionar el inicio de un juego creado previamente. Esto último llevará al profesor a una pantalla de sala de espera (8). Difiere de la de estudiante porque esta proporciona un identificador de juego que debe facilitar a sus alumnos/as y un botón que sólo podrá activar cuando haya dos o más jugadores conectados y esperando en la sala de espera.

Una vez haya comenzado el juego la app accederá al tablero (9). La principal diferencia con la de los estudiantes es que posee un botón para lanzar una pregunta y que activará los pulsadores en los dispositivos del alumnado. Cuando algún estudiante active su pulsador mostrará una pantalla (10) en la que aparecerá su alias, habilitando dos botones que

servirán para indicar si el jugador acertó o no. En caso de haber acertado se vuelve a la pantalla del tablero y habrá que esperar a que el usuario elija una casilla para conquistar. Si no acertó, la app se quedará en la misma pantalla esperando nuevamente a que se nos indique que usuario ha pulsado, repitiéndose el proceso hasta que se acierte la pregunta.

2.1.3 Prototipado

Basándonos en el diseño conceptual se ha desarrollado un prototipo horizontal para ambas aplicaciones, que ha sido retocado como resultado de la etapa de evaluación que se comentará posteriormente. Si bien se comenzó con una implementación a bajo nivel, la simplicidad que se ha buscado en las distintas pantallas ha provocado la elaboración final de un prototipo de alto nivel. Dicho prototipo se ha llevado a cabo por medio de Hypertext Markup Language (HTML) y Syntactically Awesome Stylesheets (SCSS) bajo el framework de Ionic, con el fin de reducir tiempos y costes en la etapa de implementación.

Dado que el publico objetivo se trata de un público inminentemente joven en su mayoría o de edad adulta, pero acostumbrada a trabajar con material para adolescentes, las apps se visten de distintos colores atractivos y suaves. Letras grandes y claras, así como frases desenfadadas.

Como claro ejemplo de esta estrategia es la opción de un logotipo formado por un lápiz y una regla con mango al estilo de dos espadas cruzadas preparadas para la batalla.

Figura 6. Logotipos de las apps ClassBattle Teacher Edition y Student Edition

En cuanto a las pantallas elegidas para formar parte del prototipo horizontal, se han elegido aquellas fundamentales que forman parte común (aunque requieran pequeñas modificaciones) tanto de la app destinada a los estudiantes como a los profesores. Autenticación, Entrar a Juego, Sala de Espera o el Tablero. Adicionalmente se ha incluido el prototipo de la pantalla del Pulsador.

Figura 7. Prototipo de Pantalla de Autenticación

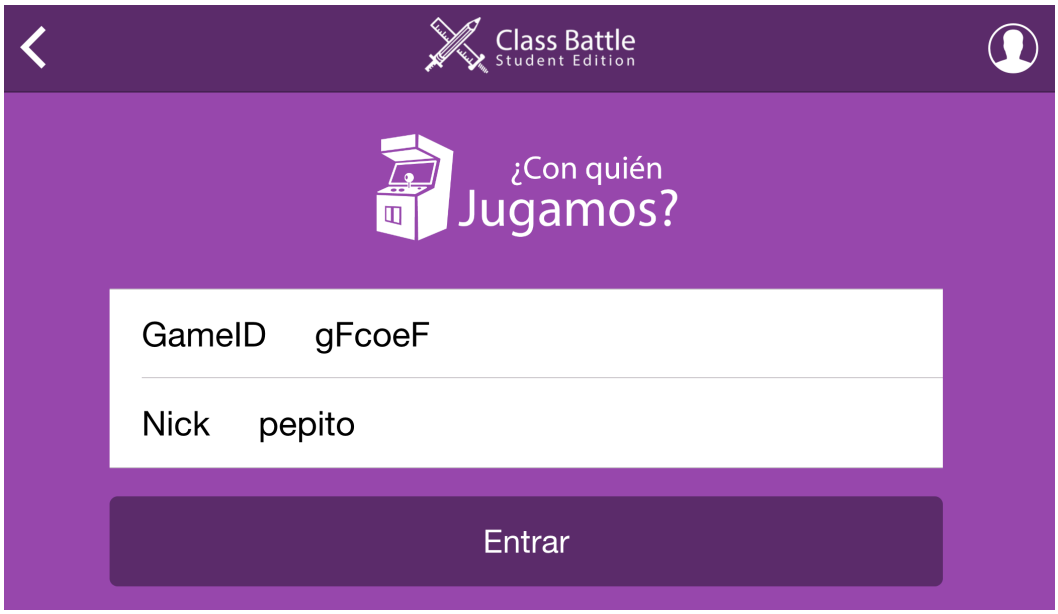

Figura 8. Prototipo de pantalla de Entrar a Juego

Figura 9. Prototipo de pantalla de Sala de Espera

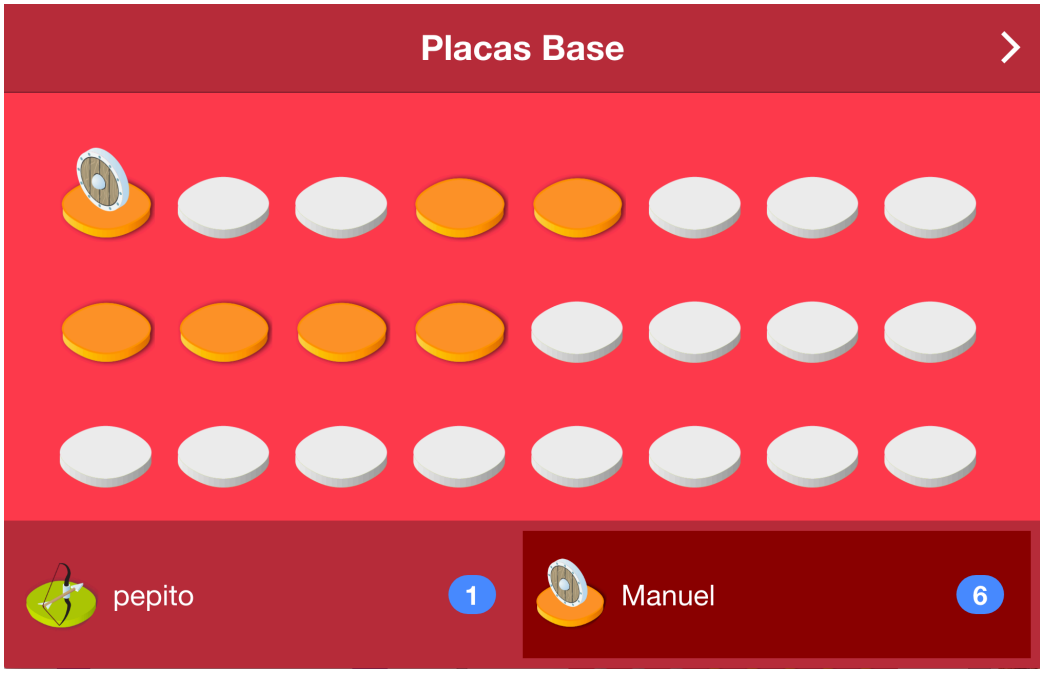

Figura 10. Prototipo de pantalla de Tablero

Figura 11. Prototipo de pantalla de pulsador

2.1.4 Evaluación

Dado que se dispone de la posibilidad de acceder de forma relativamente fácil a usuarios potenciales de nuestra app, se plantea como estrategia de evaluación el **test con usuarios**. Para tal fin se convoca de manera voluntaria a 7 alumnos/as y 2 profesores/as de la etapa de secundaria con la única información de que van a valorar una aplicación móvil.

En términos generales, de este encuentro de aproximadamente una hora de duración, se detectaron y solucionaron entre otros problemas utilizando esta técnica:

- Que el mapa de navegación inicial provocaba más pasos de los deseados por los usuarios tanto en la versión de estudiante como la de profesor, ya que inicialmente se planificó un menú principal desde donde se escogían las diferentes opciones.
- Que en la pantalla de Entrar a juego era necesario especificar que significa GameID y Nick
- Que se requería un tablero de mayor tamaño, ya que la versión inicial entraba en una única pantalla, lo que dificultaba su visualización y pulsación.

El prototipo se mostró a los usuarios como imágenes estáticas en un móvil iPhone 7 Plus. Ante la imposibilidad de utilizar un prototipo vertical para el test, las consultas iban acompañadas de ligeras explicaciones acerca del funcionamiento de la app al pulsar o realizar determinadas acciones propuestas por los presentes. En cualquier caso, sin comentar el objetivo de la app todos dedujeron fácilmente que debían registrarse y conocían perfectamente el proceso a seguir. De igual forma, coincidieron que por la estética y las frases se trataba de un juego. Al contrario de lo que inicialmente se podía pensar, se relacionó fácilmente el gameID con

el identificador del juego creado por el profesor. Esto es debido a su similitud con otras plataformas como Kahoot que utilizan la misma técnica con el mismo tipo de público. En general, una vez evaluado el prototipo establecieron en una escala del uno (completamente de acuerdo) al cinco (completamente en desacuerdo) las siguientes calificaciones que se pueden considerar positivas.

Pregunta	Calificación (Media)
La app es fácil de usar	1.5
Siempre sé que estoy en la app	1
Es difícil aprender a utilizarla	3.8
El diseño me parece atractivo	1.6
Me gustaría utilizarla en clase	1.2
La recomendaría a otros compañeros o profesores	1.2

2.2 Definición de los casos de uso

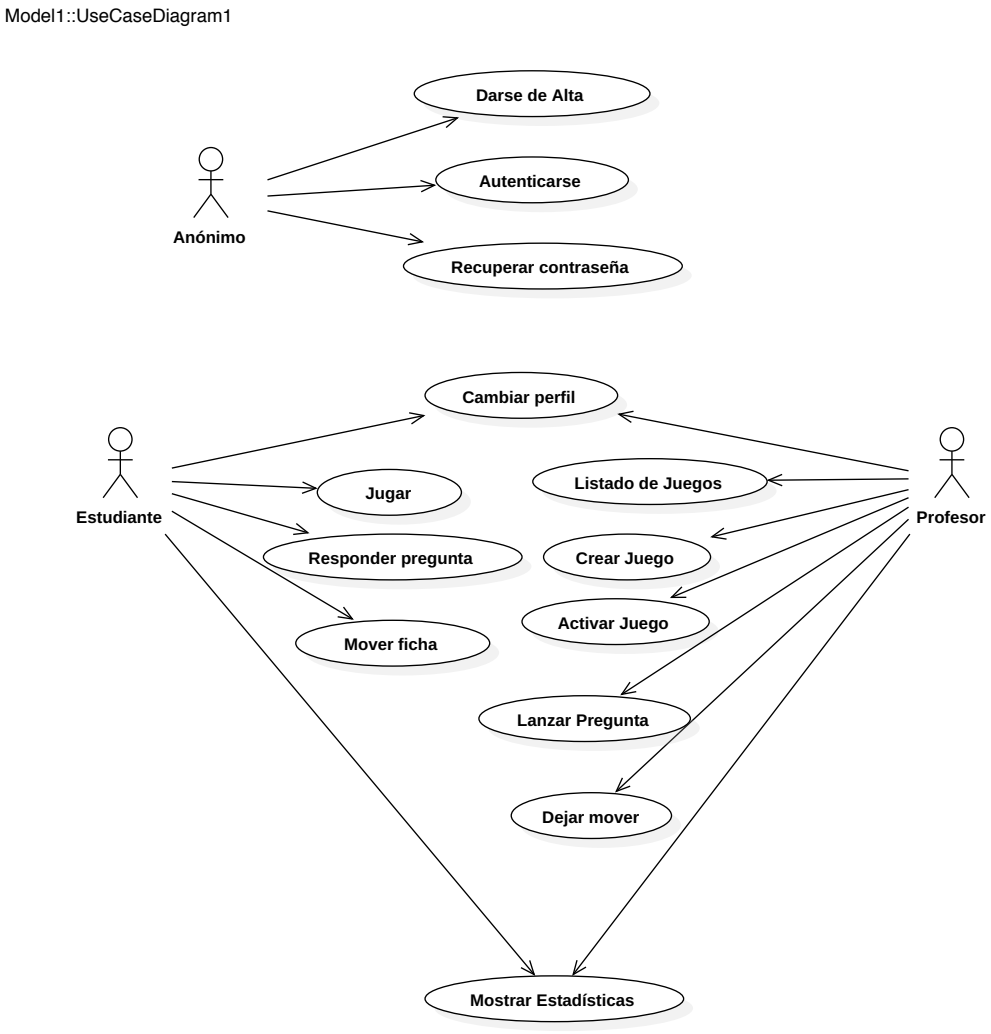

Figura 12. Diagrama de casos de uso de ClassBattle (ambas ediciones)

Aunque el proyecto lo conforman dos aplicaciones: ClassBattle Teacher Edition (en adelante BTE) y ClassBattle Student Edition (en adelante BSE), ambas trabajan sobre los mismos datos por lo que se encuentran tan íntimamente relacionadas que sus casos de uso se pueden estudiar como una única unidad.

En nuestro caso podemos distinguir 3 actores distintos:

- **Anónimo**. Usuarios que aún no se han identificado en el sistema
- **Estudiante**. Aquellos usuario que desean jugar en ClassBattle
- **Profesor**. Aquellos usuarios que organizan y dirigen un juego en ClassBattle.

A continuación se detallan cada uno de los casos de uso identificados en el sistema y que se muestran gráficamente en la figura 12.

Nombre	Darse de Alta
Actores	Anónimo
Objetivo	Conseguir una cuenta de usuario en el sistema.
Precondiciones	No estar identificado en el sistema.
Postcondiciones	Nueva cuenta creada si el proceso se realiza con éxito. Será de tipo profesor si se usó BTE o estudiante si se usó BSE.
Flujo	1. El actor abre la app BSE o BTE. 2. Se muestra la pantalla de autenticación 3. Pulsa sobre el botón “Crear una nueva Cuenta” 4. Se le conduce a una nueva pantalla donde debe proporcionar email, contraseña, repetir la contraseña y su nombre 5. Pulsa sobre el botón “Darse de alta” 6. Si todo es correcto se informa que debe confirmar su alta mediante un correo recibido en su cuenta y se le conduce a la pantalla de autenticación. 7. El actor abre su email y pincha sobre el correo recibido desde la app 8. Pulsa sobre el enlace proporcionado.

Nombre	Autenticarse
Actores	Anónimo
Objetivo	Identificarse como usuario dentro del sistema y adquirir el rol de profesor o estudiante
Precondiciones	El usuario no está identificado en el sistema.
Postcondiciones	El actor cambia de rol a estudiante o profesor y pasa a estar identificado si estaba previamente dado de alta en el sistema.
Flujo	1. El actor abre la app BSE o BTE. 2. Se le solicita un email y una contraseña 3. Introduce dicha información 4. Si dicha pareja de datos estaban registradas previamente en el sistema podrá acceder a la app.

Nombre	Recuperar Contraseña
Actores	Anónimo
Objetivo	Establecer una nueva contraseña porque se ha olvidado o se desea cambiar por motivos de seguridad
Precondiciones	Tener una cuenta previa en el sistema
Postcondiciones	El actor dispondrá de una nueva contraseña
Flujo	1. El actor abre la app BSE o BTE 2. Se muestra pantalla de autenticación 3. Pulsa sobre el botón “¿Olvidé mi contraseña?” 4. Se le conduce a una nueva pantalla en la que se le requiere su email 5. Si todo es correcto se le conduce a la pantalla de autenticación. 6. Recibe un email con un enlace que al pulsarlo le lleva a una página donde puede modificarlo.

Nombre	Cambiar perfil
Actores	Estudiante o Profesor
Objetivo	Cambiar la información personal proporcionada por el usuario inicialmente
Precondiciones	Estar identificado dentro del sistema y encontrarse en la pantalla de “Listado de Juegos” si es un profesor o “Entrar a juego” si es alumno.
Postcondiciones	Información cambiada si el proceso se realiza con éxito.
Flujo	1. El actor pulsar sobre el icono de perfil. 2. Se muestra una nueva pantalla donde puede modificar su nombre y contraseña 3. Una vez terminado pulsa sobre el botón “Actualizar datos” 4. Si todo es correcto la información es modificada y se le conduce a la pantalla anterior.

Nombre	Listado de Juegos
Actores	Profesor
Objetivo	Mostrar los juegos creados por el profesor identificado en el sistema
Precondiciones	Estar identificado dentro del sistema
Postcondiciones	Se visualiza la lista de juegos del profesor
Flujo	1. Identificarse correctamente en el sistema

Nombre	Crear Juego
Actores	Profesor
Objetivo	Dar de alta un nuevo juego dentro del sistema
Precondiciones	Estar identificado dentro del sistema y encontrarse en la pantalla de “Listado de Juegos” de BTE
Postcondiciones	Nuevo juego creado si el proceso se realiza con éxito
Flujo	1. El actor pulsar sobre el icono ‘+’. 2. Se muestra una nueva pantalla donde se le solicita un nombre y una descripción. 3. Una vez terminado pulsa sobre el botón “Actualizar datos” 4. Si todo es correcto el juego es creado con éxito

Nombre	Activar Juego
Actores	Profesor
Objetivo	Permitir el registro de jugadores
Precondiciones	Estar identificado dentro del sistema y encontrarse en la pantalla de “Listado de Juegos” de BTE
Postcondiciones	El juego seleccionado estará activo y disponible para que posteriormente los actores estudiantes puedan jugar en él.
Flujo	1. Pulsar en el botón “comenzar” del juego disponible en el listado. 2. El sistema muestra la pantalla de “Sala de Espera” proporcionando un ID de juego (GameID) 3. El actor debe proporcionar el GameID a los jugadores por cualquier medio.

Nombre	Jugar
Actores	Estudiante
Objetivo	Añadirse como jugador en un juego creado por un actor profesor
Precondiciones	Estar identificado dentro del sistema y encontrarse en la pantalla de “Entrar a Juego” de BSE
Postcondiciones	Estar registrado en el juego y acceder a la pantalla de sala de espera si todo es correcto
Flujo	1. Conseguir el GameID proporcionado por un actor profesor. 2. Introducir dicho GameID y así como el nick que se usará en el juego en el formulario que proporciona la app. 3. Pulsar el botón entrar. 4. Si todo es correcto y el número de jugadores no excede de 4 se pasará a la pantalla de sala de espera.

Nombre	Lanzar pregunta
Actores	Profesor
Objetivo	Lanzar una pregunta y establecer un mecanismo para elegir al estudiante más rápido que la conoce
Precondiciones	Estar identificado dentro del sistema y encontrarse en la pantalla de “Tablero” de BTE
Postcondiciones	Pulsadores en todos los terminales de los jugadores
Flujo	1. Lanzar una pregunta al aire 2. Pulsar el icono “lanzar pregunta” 3. Se abren pantallas de “Responder Pregunta” en todos los terminales de los jugadores 4. Se abre la pantalla “¿Acertó?” en el terminal del profesor.

Nombre	Responder pregunta
Actores	Estudiante
Objetivo	Escoger al alumno más rápido que conoce la respuesta
Precondiciones	Estar identificado dentro del sistema, encontrarse en la pantalla de “Responder Pregunta” de BSE.
Postcondiciones	Saber que alumno ha ganado si todo es correcto
Flujo	1. Pulsar sobre el botón disponible en la pantalla mientras esté activo 2. Si ha sido el más rápido la pantalla lo informa desactivando los pulsadores de los demás jugadores 3. Lanzar la pregunta al aire 4. Si la pregunta es correcta todos los jugadores pasan a la pantalla del tablero, si no se vuelven a activar todos los botones.

Nombre	Dejar mover
Actores	Profesor
Objetivo	Permitir que el estudiante que acierte una pregunta pueda realizar un movimiento en el tablero
Precondiciones	Estar identificado dentro del sistema, encontrarse en la pantalla de “¿Acertó?” de BTE y que un estudiante haya respondido correctamente usando la pantalla “Responder Pregunta”
Postcondiciones	El estudiante tiene permiso para conquistar una nueva casilla del tablero
Flujo	1. En la pantalla aparece el Nick del estudiante que pulsó en primer lugar el pulsador 2. El estudiante lanza la pregunta al aire 3. Si acertó el profesor pulsa el botón acierto y tanto el como los estudiantes pasan a la pantalla del tablero. 4. Si falla el profesor pulsa el botón de fallo, continua en la misma pantalla y se vuelven a activar todos los pulsadores de los jugadores.

Nombre	Mover ficha
Actores	Estudiante
Objetivo	Conquistar una celda del tablero
Precondiciones	Estar identificado dentro del sistema, encontrarse en la pantalla de “Tablero” de BSE y que el profesor haya permitido mover ficha a un estudiante
Postcondiciones	El estudiante posee una nueva celda del tablero aumentando su puntuación
Flujo	1. De todas las celdas próximas a la ficha del jugador en el tablero marca una y la conquista.

Nombre	Mostrar Estadísticas
Actores	Estudiante o Profesor
Objetivo	Mostrar los datos finales del juego
Precondiciones	Estar identificado dentro del sistema, encontrarse en la pantalla de “Tablero”
Postcondiciones	Información de la puntuación de cada jugador
Flujo	1. Pulsar salir.

2.3 Diseño de la arquitectura

2.3.1 Diagrama de base de datos UML

Tal y como se comentó en apartados anteriores, para conseguir el almacenamiento centralizado de los datos manejados por las aplicaciones desarrolladas en este TFM y lo que es más importante, para habilitar la sincronización entre el profesor y los distintos estudiantes que desean jugar a través de ellas se hará uso de una base de datos en tiempo real.

A continuación se muestra su diagrama UML. Aunque este tipo de diagramas suelen ser bastante auto explicativos, debemos destacar varios atributos que son claves a la hora de sincronizar a los participantes. En primer lugar el atributo **estado** de la clase **juego**. Esta clase representa todos los juegos activos en el sistema y en concreto, tal y como indica su nombre, este atributo representa el estado en el que se encuentra (no inicializado, en sala de espera, jugando y terminado). Por otro lado, el atributo **pulsadores** almacena si el profesor ha activado los pulsadores de los jugadores o no o por ejemplo, la relación **turno** entre estudiante y juego representa el usuario que pulsa primero su pulsador ante una pregunta.

Model::Main

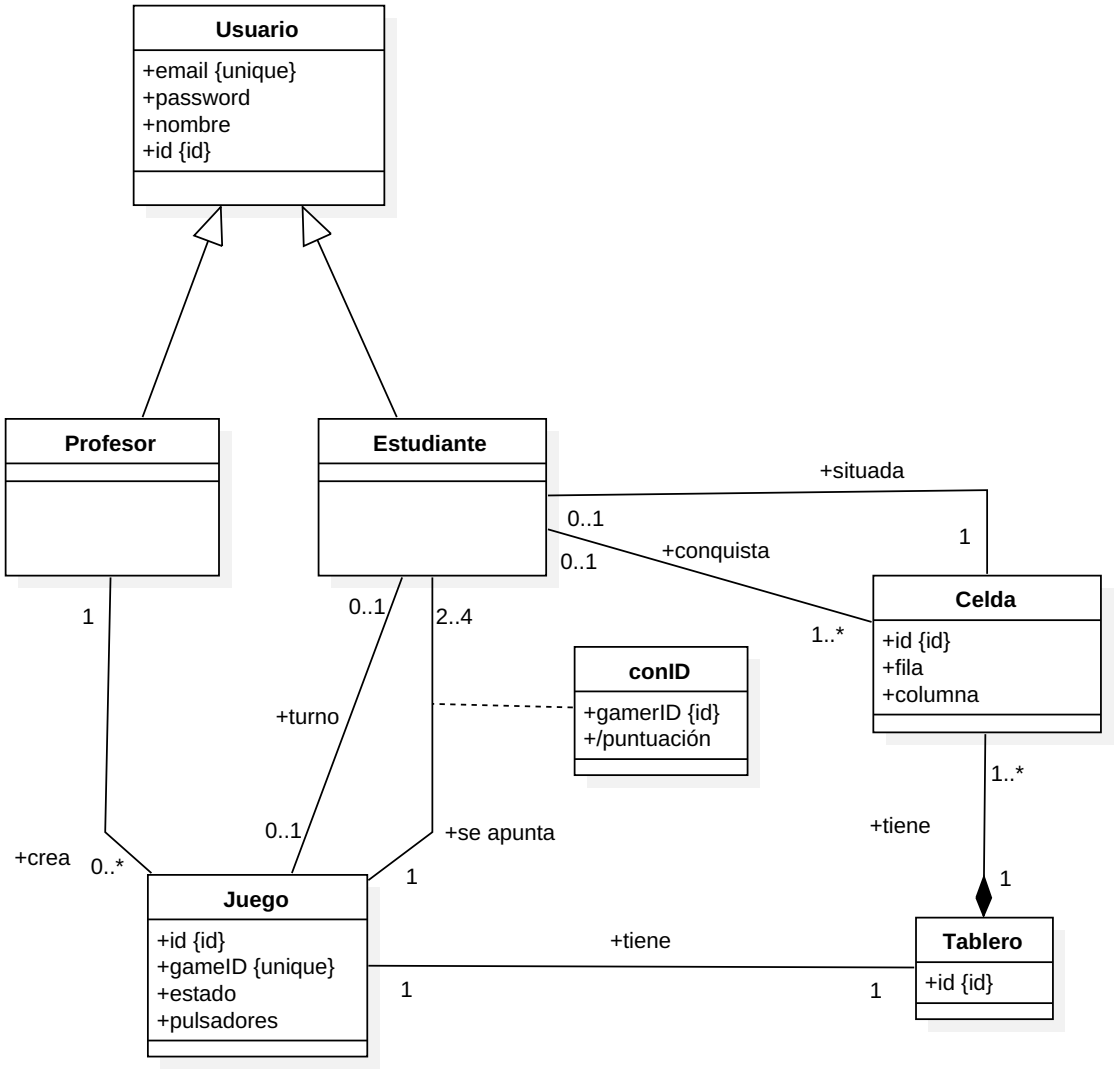

Figura 13. Diagrama UML de la base de datos

2.3.2 Diagrama de clases UML

Al igual que el diagrama de la base de datos se han construido los diagramas de clases tanto de BTE como de BSE. Estos diagramas constituyen los modelos que almacenarán los datos dentro de las apps para ser utilizados de manera inmediata. Asimismo, constituyen el armazón sobre el que se construirán ambas aplicaciones, haciendo de puente entre las especificaciones establecidas en lenguaje natural y el lenguaje de programación que se usa en la implementación.

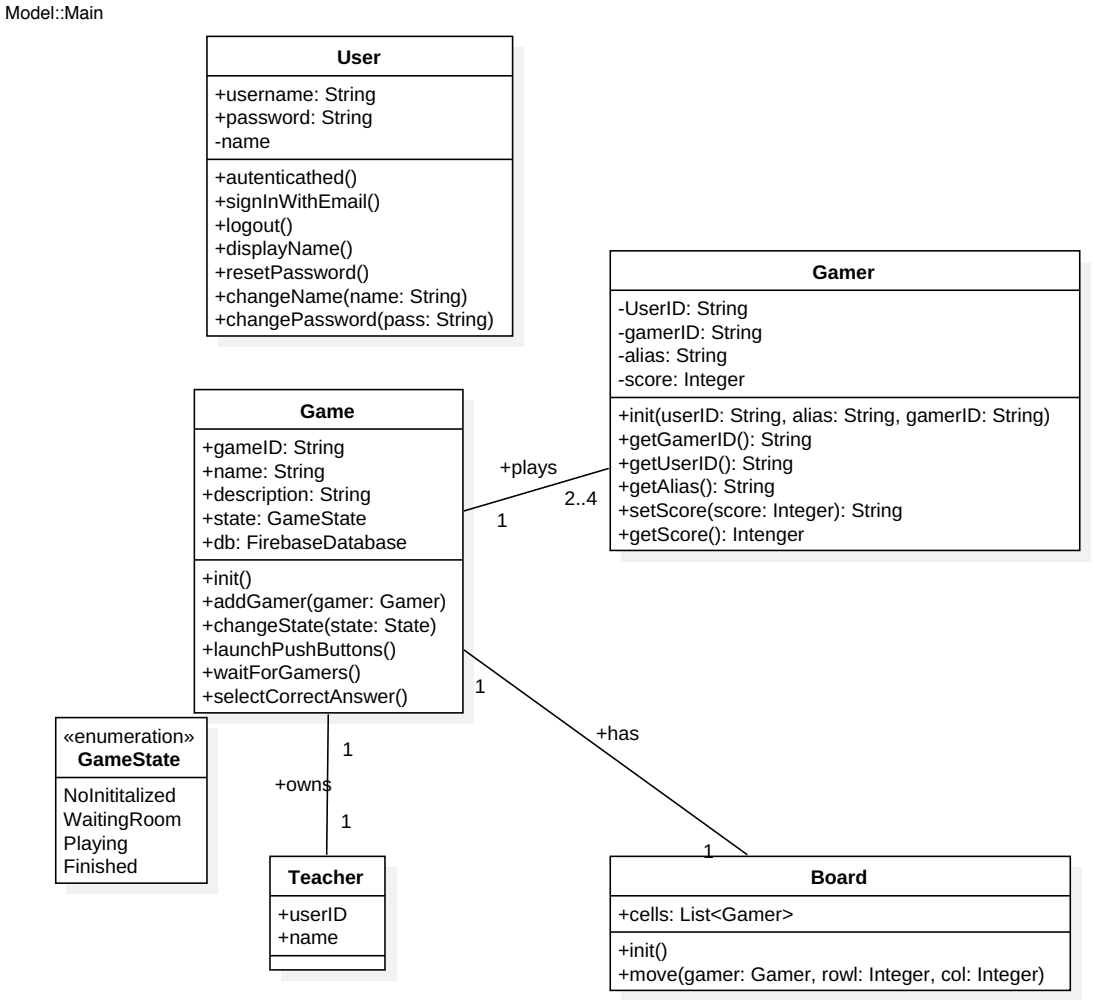

Figura 14. Diagrama UML de ClassBattle Teacher Edition

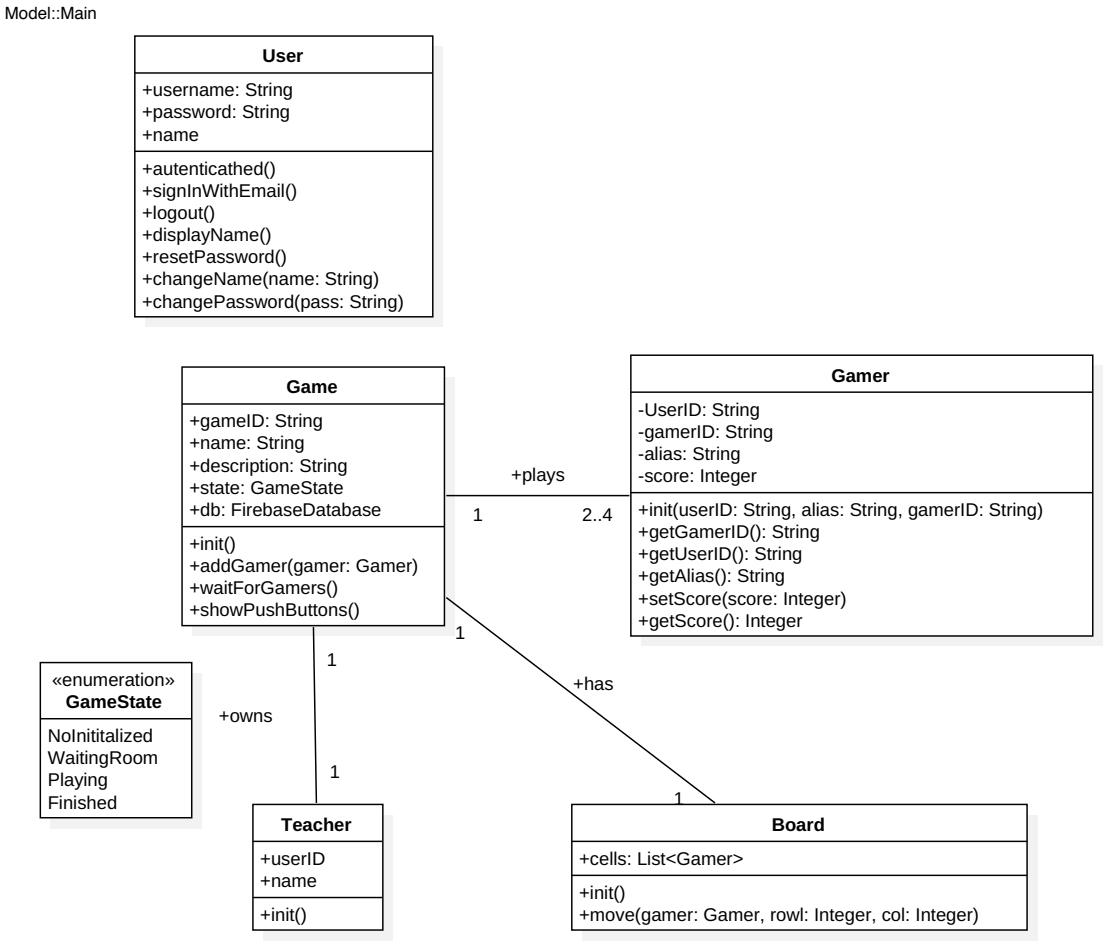

Figura 15. Diagrama UML de ClassBattle Student Edition

2.3.3 Diagrama MVC

El framework Ionic usado en la fase de desarrollo se basa en el patrón de programación MVC (Modelo Vista Controlador). Este patrón establece tres entidades que colaboran entre sí para interactuar correctamente con el usuario y que coinciden con cada una de las palabras que constituyen su nombre:

- **Modelo**. Representa la información con la que el sistema trabaja.
- **Controlador**. Actúa de intermediario entre el Modelo y la Vista.
- **Vista**. Muestra el modelo en un formato adecuado para que el usuario pueda entenderlo e interactuar con él.

Con este fin, por cada pantalla de la app, Ionic dispone de tres ficheros elementales:

- Una clase en lenguaje Typescript que actúa de controlador.
- Un fichero HTML que actúa a modo de vista y que es decorado con un fichero CSS o SCSS.

Figura 16. Diagrama básico MVC

Adicionalmente y para completar los tres elementos necesarios del patrón MVC, en Ionic es necesario crear otras clases que sirvan de modelo y gestionen los datos que deben mostrarse en la pantalla. En concreto, los diagramas de clases UML mostrados en apartados anteriores constituyen precisamente eso: las clases que sirven de modelo para el desarrollo de ClassBattle.

Es lógico pensar por tanto, que los diagramas UML anteriores no representan todas las clases del sistema y hay que tener en cuenta que para que todo funcione correctamente en Ionic es necesario añadir nuevas clases que hagan de controladores y ficheros HTML y SCSS que actúen a modo de vistas.

Cómo ejemplo de funcionamiento del patrón, se muestra a continuación un diagrama que refleja una parte del funcionamiento de la pantalla de autenticación.

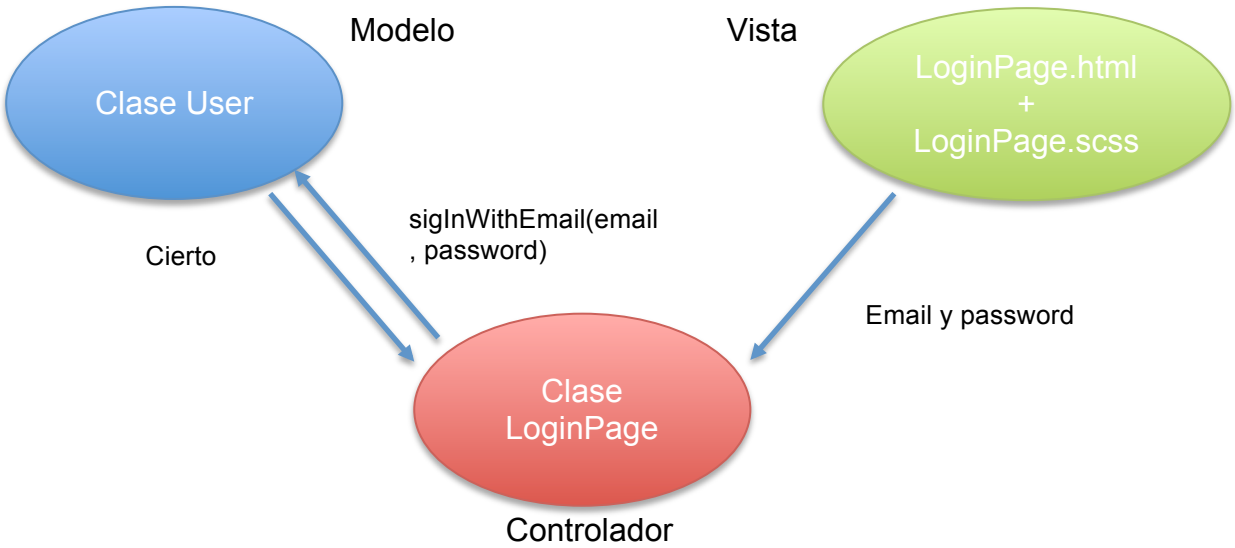

Figura 17. Diagrama básico MVC de la página de autentificación

En este ejemplo, cuando el usuario introduce su email y su contraseña en la pantalla de Autentificación y pulsa el botón “entrar”, el controlador (una clase denominada LoginPage) recoge los datos de los campos del

formulario incluidos en la vista (un documento HTML y una hoja de estilos con el mismo nombre que el controlador), formados por el email y el password del usuario. A continuación invoca el método de la clase user pasándole el email y password que se encontraban en la vista. Esta clase forma parte del modelo y como tal nos da acceso a los datos con el fin de saber si los datos de nuestro usuario forman parte del sistema. Pertenezca o no al mismo, devolverá un mensaje al controlador indicando el resultado.

Como puede observarse el controlador se encarga de hacer de intermediario entre el modelo y la vista, separando los datos de la presentación, con las ventajas programáticas y de diseño que todo esto conlleva.

2.4 Revisión de la temporalización de la fase

Si bien es cierto que no se ha cumplido estrictamente la temporalización inicialmente establecida en la fase del plan de trabajo, si que se ha aproximado bastante. De hecho, durante esta fase tan sólo se ha producido un exceso de 5,5 horas sobre el computo total estimado. Esto ha sido debido principalmente por:

- La elaboración de un prototipo algo más detallado con el fin de que el test con usuarios fuese más efectivo y a su vez se pueda ganar más tiempo a la fase de implementación.

- Problemas ajenos con la elaboración de este documento.

PEC2. Diseño	81,5																					
DCU. Usuarios y contextos de uso	8	2	2	2	3																	
DCU. Diseño conceptual	8							4	4													
DCU. Prototipado	22								1	3	3	3	3				3	5	1			
DCU. Evaluación	11													4	3	3	1					
DT. Definición de casos de uso	9				2	3	3	1														
DT. Diseño de la arquitectura	11																		4	5	2	
Elaboración de documentación	8,5																				2,5	3	3

Figura 18. Planificación final y real de la fase de diseño

3 Fase de implementación

3.1 Creación y gestión de la base de datos

Tal y como se comentó en el apartado de introducción de este documento, el sistema gestor de base de datos (SGBD) elegido para la elaboración ClassBattle ha sido Google Firebase. Este SGBD no se basa en el almacenamiento de los datos en tablas interrelacionadas tal y como ocurre en los sistemas relacionales tradicionales si no que almacenan la información en forma de objetos utilizando la notación JSON (JavaScript Object Notation), con la característica adicional de que permite trabajar en tiempo real. Esto conlleva que las aplicaciones que se encuentren conectadas a ella, pueden recibir una notificación de cualquier modificación que se realice en cualquier dato almacenado en su interior. Es la aplicación la encargada de suscribirse a los datos sobre los que desea ser informado en caso de creación, borrado o modificación.

Mientras que existen una serie de estrategias definidas y establecidas para obtener el diseño físico de las bases de datos relacionales a partir del modelo de datos en formato UML, en el caso de las bases de datos JSON esto no está tan claro, principalmente porque la filosofía de trabajo cambia ligeramente. Sin embargo sí que existe el objetivo principal de buscar la simplicidad al máximo en cada objeto con la idea de provocar un intercambio mínimo de datos entre el cliente y el SGBD. Esto conlleva que se evite en la medida de lo posible el anidamiento de unos objetos dentro de otros. Adicionalmente, dado que este tipo de bases de datos no poseen un esquema previo que defina su estructura, hay que prestar mucha atención para evitar crear jerarquías de datos inconsistentes. Si bien esta característica es una ventaja ideal en las metodologías ágiles, no deja de ser un talón de Aquiles que puede derivar en problemas con mucha mayor facilidad que lo que cabe esperar de un SGBD relacional tradicional.

3.1.1 Diseño físico

Partiendo de que en el caso de Google Firebase existe una ubicación u objeto raíz (/) desde donde cuelgan o se almacenan cada uno de los objetos principales del sistema, nuestro diagrama UML se ha implementado de la siguiente forma:

- /users: Contiene los objetos de tipo “**usuario**”
- /games: Contiene los “**juegos**” activos en el sistema
- /gamers: Contiene los objetos “**estudiantes**” que se “**apuntan**” a un juego

- /boards: Contiene los "**tableros**" asociados a cada juego dentro del sistema
- /question: Contiene el "**turno**" para ver quien puede responder y en el caso de acertar, contendrá también el usuario que "**conquistará**" la casilla.

Descripción de cada objeto principal

/users

id: { username, password, email, teacher }

- id: identificador único del usuario del sistema.
- username: Nombre del usuario.
- email: Email del usuario.
- teacher: Verdadero o falso, determina si el usuario es un profesor o un estudiante.

/games

gameID: { nombre, descripcion, creador, status }

- gameID: Identificador único del juego en el sistema.
- nombre: nombre del juego
- descripción: descripción del juego
- creador: id del usuario que "**crea**" el juego.
- status: estado en el que se encuentra el juego. (1 – No inicializado, 2 – en la sala de espera, 3 – En juego y 4 – Finalizado).

/gamers

gameID: { userID : { id, name, score }}

- gameID: Identificador único del juego en el sistema.
- userID: Identificador del jugador que "**se apunta**" al juego identificado en gameID
- id: Número que lo identifica dentro del juego
- name: Nombre que se utilizará dentro del juego
- score: Puntuación dentro del juego.

/boards

gameID: { cells, lastmovements}

- gameID: Identificador único del juego en el sistema.
- cells: Celdas del tablero del juego que "**tiene**" gameID y que "**contienen**" las casillas "conquistadas" por los jugadores
- lastmovements: Almacena la posición de jugadores "**situados**" en el tablero.

/question
gameID: { asking, player }

- gameID: Identificador único del juego en el sistema.
- asking: Contiene si se ha realizado una pregunta. (0 – no hay pregunta, 1 – hay pregunta, 2 – Alguien ha contestado). Constituye el atributo "**pulsadores**" del modelo de datos.
- player: Almacena el jugador que posee el turno para mover porque ha respondido correctamente a la pregunta.

Obsérvese que las palabras marcadas entre comillas se refieren a objetos y relaciones del modelo de datos inicialmente establecido.

Figura 19. Esquema de ejemplo del objeto gamers que almacena los jugadores de cada juego

3.1.2 Revisión del diagrama de base de datos UML

En un mundo ideal todo podría hacerse de forma correcta al primer intento, desgraciadamente esto no es así. Si se analizan detalladamente los objetos declarados en el punto anterior es posible darse cuenta de que aunque en el modelo se establece almacenar la contraseña de cada usuario en la base de datos, esto finalmente no se ha llevado a cabo. La razón es que Google proporciona una herramienta llamada Google Auth que gestiona esta tarea y que trabaja conjuntamente con Google Firebase proporcionando un framework integrado para la gestión de usuarios de forma completamente segura dentro de cualquier app. La existencia de estas herramientas ha llevado a que no fuese necesaria su inclusión directa en el objeto usuario, aunque no se ha retirado del modelo por motivos de claridad.

Al margen de esta situación fue necesario revisar el modelo inicial con el fin de que los usuarios pudiesen cambiar su nombre según el juego. La idea se centra en que es posible que el usuario preste su terminal para que otro jugador o jugadores puedan participar en el juego. De ahí que el objeto que almacena los jugadores (gamers) en cada juego, posea una propiedad name que inicialmente no se contempló y que puede contener

un valor completamente distinto del nombre del usuario que se autentificó inicialmente.

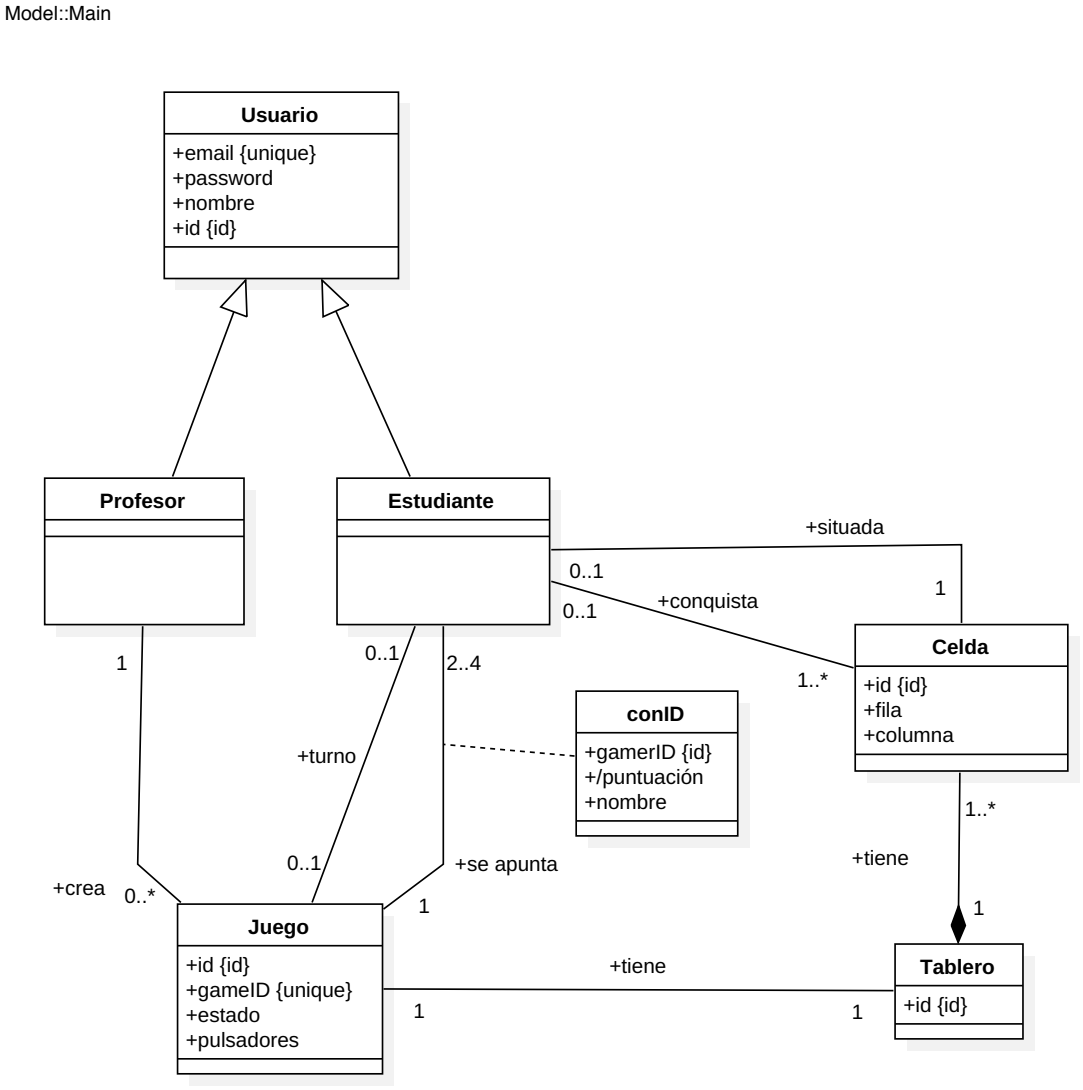

Figura 20. Diagrama UML de la base de datos revisado.

3.2 Desarrollo de ClassBattle edición del profesor

3.2.1 Revisión del diagrama del modelo UML

Al igual que ocurrió en la etapa de implementación de la base de datos, a la hora de empezar a codificar la aplicación destinada a los profesores surge la necesidad de añadir algún elemento más al modelo. Si bien la estructura principal es correcta, se hace imprescindible la inclusión de un método que mantenga sincronizado el tablero tanto en la base de datos

como en la memoria local del dispositivo. De ahí que posteriormente se añada uno adicional a la clase **Game** llamado updateBoard().

Una vez hecha esta modificación, el modelo se transcribe prácticamente igual al código con la salvedad de que:

- Finalmente no se crea un nuevo tipo denominado GameState si no que se introduce un número para identificar el estado del juego, simplificando el código y los accesos a la base de datos. Se mantiene en el modelo por razones de claridad.
- La clase Game se implementa con el nombre de **GameProvider** con el fin de seguir la nomenclatura que establece Ionic para las clases que definen objetos que se pueden compartir entre páginas.
- No es necesario crear la clase Board y se implementa directamente como una matriz, aunque se mantiene en el modelo por las mismas razones que GameState.

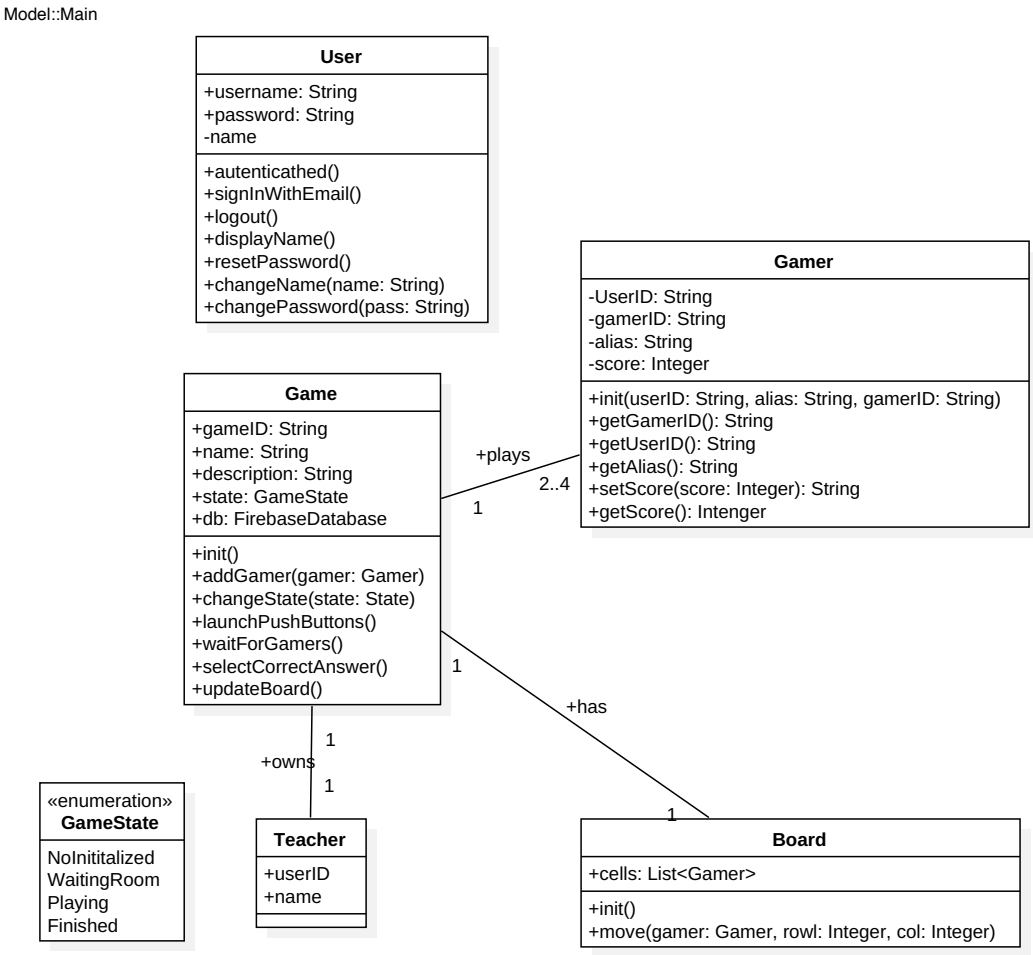

Figura 21. Diagrama UML del modelo de datos de ClassBattle Teacher Edition revisado.

Aunque el modelo a simple vista pueda parecer simple, **la complejidad de llevarlo al código ha sido bastante elevada**. La razón principal se

centra en la necesidad de comunicar de manera síncrona como mínimo tres aplicaciones (un profesor y dos estudiantes) utilizando métodos asíncronos sobre una zona de memoria compartida que en nuestro caso ha sido Firebase.

Para añadir mayor dificultad al proyecto, la app del profesor debe ser desarrollada primero, a sabiendas de que una no puede vivir sin la otra. Esto hace que el desarrollo se haga un poco “a ciegas”, siempre a expensas de simular directamente y de forma manual el comportamiento de su app homóloga sobre la base datos con el fin de poder comprobar que la funcionalidad deseada de la versión del profesor es correcta.

La inclusión en la temporalización del desarrollo de la versión del profesor como primera opción frente a la estudiante no es meramente casual. Al ser el director de orquesta, es la que dirige el juego y por tanto es la que nos permite detectar posibles errores de especificación con mayor antelación, tal y como así ha sucedido.

3.2.2 Vistas y controladores

En cuanto a los controladores y vistas empleados, se desarrollan 10 de cada uno de ellos, cuyos nombres y funcionalidades se detallan a continuación:

- **add-game**. Encargado de gestionar la creación de nuevos juegos
- **board-game**. Encargado de mostrar el tablero en pantalla y comunicarse con GameProvider para la gestión y desarrollo del juego en curso.
- **edit-profile**. Permite la modificación de los datos de perfil del usuario identificado en ese momento dentro de la app.
- **forgot-password**. Tiene la tarea recuperar la contraseña del usuario en caso de pérdida o de sospecha de haber sido descubierta.
- **games-list**. Muestra un listado de los juegos creados en el sistema por el profesor dado de alta en la app.
- **login**. Permite la identificación del usuario dentro de la app
- **sign-up**. Destinado a permitir el registro de usuarios como profesores.
- **statistics**. Muestra un listado de las jugadores y puntos obtenidos durante el juego.
- **turn**. Se encarga de mostrar el jugador que posee el turno en este momento porque fue el primero en pulsar el botón de juego.
- **waiting-room**. Pantalla que muestra un listado de los usuarios.

Para cada uno de los nombres anteriores, y siguiendo el flujo de trabajo de Ionic se crean al menos 3 ficheros:

- Un HTML con la información a mostrar. (Vista)

- Un SCSS con la información del estilo gráfico que se aplicará al HTML (Vista)
- Un TS con la lógica del controlador y que hará de intermediario entre el modelo y el fichero HTML que constituye parte de la vista.

En la elaboración de las vistas se ha procurado en todo momento que la información se ajustara correctamente a los distintos tipos de pantalla disponibles para dispositivos iPhone más actuales en el momento de la implementación (modelos iPhone SE y 5, iPhone 6, 7, 8 y X e iPhone 6 Plus, 7 Plus y 8 Plus).

3.3 Desarrollo de ClassBattle edición de estudiante

3.3.1 Revisión del diagrama del modelo UML

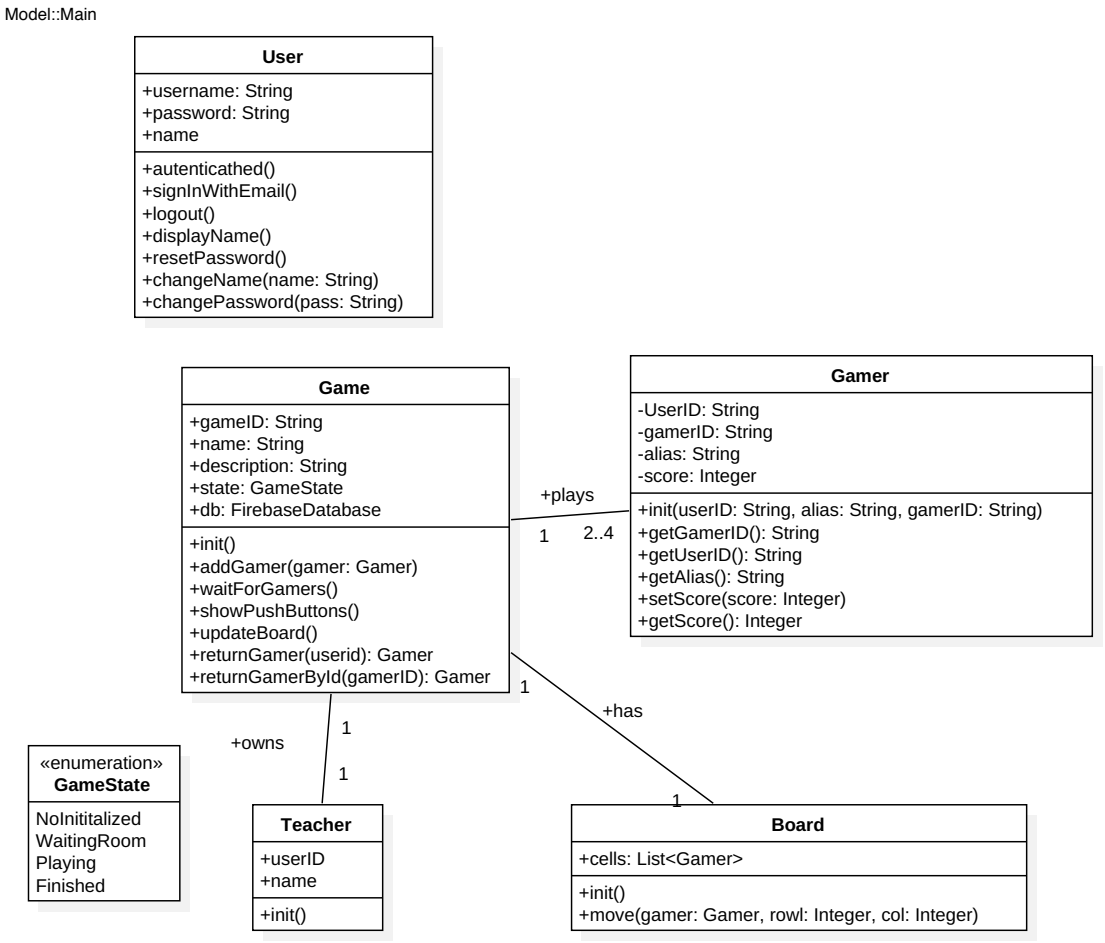

Figura 22. Diagrama UML del modelo de datos de ClassBattle Student Edition revisado.

Tal y como ha ido sucediendo en los desarrollos anteriores, la edición de estudiante de ClassBattle tampoco ha estado exenta de modificaciones en el modelo inicial. En este caso, adicionalmente al método **updateBoard**() que también se añadió en la versión del profesor a la clase GameProvider, se hace

necesario incluir otros métodos. En concreto a esta misma clase se le incluyen dos métodos (**returnGamer** y **returnGamerID**) para recuperar los jugadores almacenados dentro de la clase.

Al llevar el modelo al código, se cumplen las mismas estrategias seguidas en la versión del profesor, es decir:

- Finalmente no se crea un nuevo tipo denominado GameState si no que se introduce un número para identificar el estado del juego, simplificando el código y los accesos a la base de datos. Se mantiene en el modelo por razones de claridad.
- La clase Game se implementa con el nombre de **GameProvider** con el fin de seguir la nomenclatura que establece Ionic para las clases que definen objetos que se pueden compartir entre páginas.
- No es necesario crear la clase Board y se implementa directamente como una matriz, aunque se mantiene en el modelo por las mismas razones que GameState.

3.3.2 Vistas y controladores

En este caso también se dispone de 10 vistas y controladores:

- **board-game**. Encargado de mostrar el tablero en pantalla y comunicarse con GameProvider para la gestión y desarrollo del juego en curso.
- **button**. Muestra el botón que permite dar acceso al turno de respuesta.
- **edit-profile**. Permite la modificación de los datos de perfil del usuario identificado en ese momento dentro de la app.
- **enter-game**. Pantalla que da acceso a un juego proporcionando un identificador de juego y un nombre para el jugador.
- **forgot-password**. Tiene la tarea recuperar la contraseña del usuario en caso de pérdida o de sospecha de haber sido descubierta.
- **login**. Permite la identificación del usuario dentro de la app
- **sign-up**. Destinado a permitir el registro de usuarios como profesores.
- **statistics**. Muestra un listado de las jugadores y puntos obtenidos durante el juego.
- **turn**. Se encarga de mostrar el jugador que posee el turno en este momento porque fue el primero en pulsar el botón de juego.
- **waiting-room**. Pantalla que muestra un listado de los usuarios pendientes de entrar al juego.

3.4 Pruebas

Dado que se dispone de la posibilidad de hacer uso de las aplicaciones en el entorno real de una clase de secundaria, la aplicación es testeada

en cuatro sesiones situadas en diferentes días y llevadas a cabo por dos profesores del centro educativo I.E.S. Serranía de Alozaina (Málaga), ambos distintos del desarrollador del proyecto. Para ello se proporciona una plantilla en la que en caso de detectar un fallo se debe indicar:

- Texto del fallo mostrado.
- Pantalla en la que se produce.
- Descripción del estado del juego en el momento del fallo.
- Nivel del fallo (leve – permite continuar con el juego o grave – Impide continuar con el desarrollo del juego).

Dado que no se dispone de un número de terminales iOS suficiente para realizar las pruebas y tampoco de un certificado de desarrollador, parte de la ejecución se lleva a cabo por medio del simulador web que incluye Ionic y otra por medio de terminales iPhone con iOS 11 haciendo uso de la app Ionic View. Esta última app permite ejecutar una app subida a los servidores de Ionic sin necesidad de tener que instalarla físicamente en el smartphone.

Durante la primera sesión de pruebas, a parte de varios fallos leves, se detectaron varios graves. Principalmente generados por la falta de previsión de cambios en la latencia de las comunicaciones entre los dispositivos. El desarrollo y pruebas puntuales sobre un mismo equipo nos puede sumergir en un mundo irreal en que las comunicaciones no tienen retardo y todo puede ocurrir al mismo tiempo. Esta primera sesión puso de manifiesto pequeños errores en la codificación de los métodos de GameProvider derivados precisamente del desarrollo en un entorno sin esperas. Entre otros errores cabe destacar uno que se produce de manera aleatoria cuando se ejecuta la app en el entorno web (que en nuestro caso, no debe ocurrir ya que el destino final es una app iOS) y que es ajeno al proyecto en sí. Ionic, al igual que cualquier software, no está exento de fallos y en ocasiones, a pesar de que se le solicita simular un entorno iOS, muestra un Android. Dado que la app no está diseñada específicamente para esta plataforma, muestra la información de manera incorrecta, aunque el funcionamiento y comportamiento sigue siendo el mismo. De igual forma Ionic View provoca fallos debidos a su propio entorno de ejecución ya que por ejemplo, al contrario de lo que ocurre en el emulador, no funcionan los plug-ins que controlan el statusbar o los cambios de orientación, permitiendo ver la app en vertical, cuando esto no debe ocurrir. Lo que si es cierto, es que ninguno de estos errores comentados anteriormente y que se producen en la versión web y la de Ionic Viewer, son visibles en el emulador de iOS.

Tras realizar los cambios pertinentes en el código se realizó una segunda sesión llevada a cabo por el mismo profesor y grupo de alumnos de la primera. En ella se detectan nuevamente fallos, pero esta vez de tipo leve. En ningún caso se produjo un error que impidiese el desarrollo del juego. Entre los leves se detecta que no se ponía límite al tamaño de los nombres a usar y si bien no supone un problema de almacenamiento en el SGBD, si que es un problema a la hora de mostrarlo, ya que puede

romper el diseño de la pantalla del tablero o la de la sala de espera, por ejemplo.

Una vez realizados los cambios oportunos, se pudo comprobar que en las sucesivas sesiones realizadas por otro profesor diferente al de las dos primeras sesiones no se produjo ningún otro fallo adicional al propio de Ionic comentado en la primera sesión. Evidentemente esto no nos guarda de que el software esté exento de errores, sobre todo porque aunque Ionic garantiza que sus aplicaciones pueden ejecutarse en entornos con iOS SDK 8 o superior, tan sólo lo hemos podido testear en iOS 11 o entornos web. Asimismo y aunque no se puedan contemplar como fallos, a raíz de poder testear las app en un entorno real, se han generado una serie de posibles ampliaciones y/o mejoras cuya misión son la de cubrir necesidades que se dejaron de lado, bien por la limitación temporal impuesta en este TFM o porque no se tuvieron en cuenta inicialmente. Dichas modificaciones serán comentadas más adelante en el apartado de conclusiones.

3.5 Revisión de la temporalización de la fase

Al contrario que en la fase de diseño, en esta si que se ha producido una desviación en tiempo más pronunciada. De las 131 horas planificadas inicialmente se ha pasado a 149, siguiendo la distribución que se muestra en la figura 27.

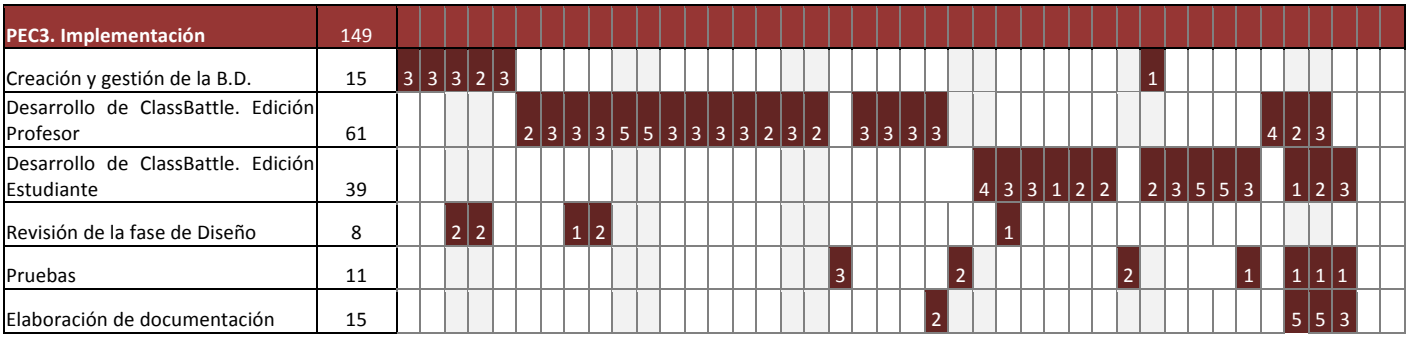

Figura 23. Planificación final y real de la fase de implementación.

Entre las razones más importantes se encuentran:

- El hecho de que se había pasado por alto una posible revisión de la fase de diseño, con el tiempo adicional que ello conlleva.
- No se había contemplado la posibilidad de dedicar tanto tiempo a revisar y modificar la documentación anterior, aumentando el tiempo dedicado a esta tarea
- La complejidad asociada a las comunicaciones asíncronas, provocando un aumento de las horas dedicadas al desarrollo de ambas apps.

Sin embargo, se reduce el número de horas dedicadas a las pruebas, pasando de 18 a 11. Esto no quiere decir que se haya probado menos de lo deseado, sino que dicha tarea sufrió una sobrevaloración en tiempo.

4. Conclusiones

Tras la realización del análisis y pruebas de ClassBattle, se ha obtenido una buena aceptación entre el grupo de profesores/as y alumnos/as que se han ofrecido a colaborar en el proyecto. Sin embargo, tras su uso han surgido multitud de posibilidades de expansión y propuestas de mejora. Entre ellas se pueden destacar:

- Creación de una versión Android y Web.
- Almacenaje de estadísticas por cada usuario/grupo que juega.
- Posibilidad de incluir otros tipos de juegos con la misma dinámica pero que no se basen en un tablero.
- Posibilidad de elegir entre una gama más amplias de fichas al comenzar el juego o incluso la posibilidad de que el usuario diseñe la suya propia.
- Permitir al profesor el almacenamiento de una batería de preguntas tipo, pudiendo elegir entre ellas la próxima en lanzarse o incluso añadirlas bajo demanda y mientras el juego esté activo.

En cuanto a la posible expansión a otras plataformas, el hecho de que se haya utilizado Ionic hace que el proceso sea relativamente sencillo. En concreto, pruebas con el código generado para iOS sobre el emulador de Android o sobre un navegador web a pantalla completa, muestran que el lanzamiento en estas plataformas requieren a priori sólo una serie de cambios mínimos. Dichos cambios no se contemplaron como objetivos de este proyecto para evitar una dilatación en el tiempo que pudiera exceder los límites de este TFM. De hecho el tiempo final dedicado al mismo, sólo para la versión iOS, ha superado el que se planificó inicialmente.

Este proyecto en sí ha supuesto un reto que ha permitido por un lado satisfacer una necesidad profesional del autor y por otro, poner de manifiesto gran parte de los conocimientos adquiridos durante la ejecución del Máster al que pertenece este TFM. Ni que decir tiene que también ha sido necesario el aprendizaje de muchos otros conceptos y procedimientos que estando relacionados con la materia impartida no se habían tratado a lo largo del curso. Este es el caso de Google Firebase, Google Auth o el funcionamiento profundo de Ionic por poner un ejemplo, por lo que la elaboración de este TFM ha servido para complementar todo lo aprendido. Adicionalmente ha servido para desempolvar conceptos adquiridos hace años sobre Ingeniería del software. Disciplina del mundo de la informática tan necesaria como difícil, donde se mezclan ciencia, experiencia y arte.

5. Glosario

C

cyberbulling
Acoso psicológico entre iguales por medios telemáticos. 1

D

DCU **Véase Diseño Centrado en el Usuario**
Diseño Centrado en el Usuario
Filosofía de diseño de productos que tiene como misión final que estos resuelvan necesidades específicas de los usuarios, con la mejor experiencia de uso y mayor satisfacción posible. 12, 13

F

framework
Plataforma que define un estructura conceptual y tecnológica para la organización y desarrollo de software. 5, 6, 19, 32, 38, 46

G

grooming
Técnicas utilizadas para ganarse la confianza de un menor de edad con el objetivo de obtener beneficios sexuales. 1

H

HTML ***Véase*** **Hypertext Markup Language**
HyperText Markup Language 33
Hypertext Markup Language
Lenguaje de marcado usado para estructurar documentos electrónicos. Es el estandar actaulemente utilizado para la elaboración de páginas web. 19

I

Ingeniería del Software
Disciplina perteneciente al campo de la informática cuyo objetivo es obtener software de calidad con el menor uso de recursos posible. 4

J

JavaScript Object Notation
Formato ligero de representación e intercambio de datos 36
JSON **Véase JavaScript Object Notation, Véase JavaScript Object Notation**

S

SGBD ***Véase*** **sistema gestor de bases de datos**
sistema gestor de base de datos
Conjunto de programas que permiten el almacenaje y procesamiento de información dentro de una base de datos 36
smartphone
Teléfono móvil con pantalla táctil que funciona a modo de pequeño computador. 1
Splash Screen
Pantalla inicial que suele aparecer durante el proceso de carga de una aplicación 18
Syntactically Awesome Stylesheets
Lenguaje de hoja de estilos avanzados, que mejora el estandar cascade stylesheets. 19

T

Tecnologías de la Información y Comunicación
Conjunto de tecnologías que permiten el acceso, creación, procesamiento y transmisión de la información. 1
TFM **Véase Trabajo Fin de Máster**
TIC ***Véase*** **Técnologías de la Información y Comunicación**
Trabajo Fin de Máster
Proyecto que se realiza en la fase final del plan de estudios de un Máster universitario con el objetivo de evaluar las competencias asociadas al titulo ofertado. 1

U

UML Véase Unified Modeling Language.
Unified Modeling Language
Lenguaje estándar para la creación de esquemas, diagramas y documentación relativa a los desarrollos de software. 12

6. Bibliografía

[1] J.M. Méndez & M. Delgado. Las TIC en centros de Educación Primaria y Secundaria de Andalucía. Un estudio de casos a partir de buenas práctica. Digital Education Review - Number 29, June 2016- http://greav.ub.edu/der/

[2] Class Quiz Games with Quizizz (an Alternative to Kahoot). https://learninginhand.com/blog/quizizz. Disponible el 26 de Septiembre de 2017.

[3] Informe ditrendia: Mobile en España y en el Mundo 2016. http://www.amic.media/media/files/file_352_1050.pdf. Disponible el 26 de Septiembre de 2017.

[4] Kahoot. https://kahoot.it/. Disponible el 26 de Septiembre de 2017.

[5] Quizziz. https://quizizz.com/join/. Disponible el 26 de Septiembre de 2017.

[6] Quizalize. https://www.quizalize.com/. Disponible el 26 de Septiembre de 2017.

[7] Quizlet. https://quizlet.com/live. Disponible el 26 de Septiembre de 2017.

[8] Phonegap. https://phonegap.com/. Disponible el 26 de Septiembre de 2017.

[9] Ionic. https://ionicframework.com/. Disponible el 26 de Septiembre de 2017.

[10] Native Script. https://www.nativescript.org/. Disponible el 26 de Septiembre de 2017.

[11] Facebook. https://facebook.github.io/react-native/. Disponible el 26 de Septiembre de 2017.

[12] NetMarketShare. https://www.netmarketshare.com/. Disponible el 26 de Septiembre de 2017.

[13] Apple Developer. https://developer.apple.com/documentation/multipeerconnectivity. Disponible el 26 de Septiembre de 2017.

[14] https://developers.google.com/nearby/connections/overview. Disponible el 26 de Septiembre de 2017.

[15] https://developers.google.com/nearby/messages/overview. Disponible el 26 de Septiembre de 2017.

[16] Google Firebase. https://firebase.google.com. Disponible el 26 de Septiembre de 2017.

[17] Rethink. https://www.rethinkdb.com/. Disponible el 26 de Septiembre de 2017.

[18] MongoDB. https://www.mongodb.com/. Disponible el 26 de Septiembre de 2017.

[19] NodeJS. https://nodejs.org/en/. Disponible el 26 de Septiembre de 2017.

7. Anexos

A continuación se muestran las hojas de formulario utilizados para la recogida de información necesaria tanto en la evaluación del prototipo como la fase de pruebas llevadas acabo, así como las instrucciones para la puesta en marcha de ClassBattle y un manual de usuario básico.

7.1 Hoja de evaluación del prototipo

Para finalizar la sesión de evaluación de los prototipos vistos queremos **agradecerte tu colaboración** y pedirte un último favor. Marca con una X la puntuación que das a cada una de las afirmaciones que se te plantean a continuación. Considera que un **1 es completamente de acuerdo** y un **5 completamente en desacuerdo**.

Pregunta	**Calificación** (1 – de acuerdo y 5 – desacuerdo)				
	1	2	3	4	5
La app es fácil de usar					
Siempre sé que estoy en la app					
EEs difícil aprender a utilizarla					
El diseño me parece atractivo					
La recomendaría a otros compañeros o profesores					

7.2 Hoja de registro de fallos de las apps

La siguiente hoja sirve para recoger **cualquier fallo** que encuentres en la aplicación que estas evaluando. Aparte de **agradecerte enormemente** la labor que estás desarrollando, es importante que sigas los siguientes pasos para que esta sesión sea lo más efectiva posible.

- Salvo tu nombre (en el campo nombre del usuario), el resto de campos son **obligatorios**. Aunque si lo especificas siempre puede ser más fácil solucionarlo en caso de dudas.
- Intenta describir cada información **lo mas detallada posible**.
- Es importante que identifiques que un fallo **no es sólo algo** que sale mal en pantalla, si no también algo que esperabas que debía de pasar y no ocurrió o lo hizo de otra forma diferente.
- Usa una de estas hojas por cada fallo que encuentres.

Muchas gracias por tu colaboración.

Nombre del usuario: _____
Fecha:_____

¿Qué aplicación estás usando?

ClassBattle Teacher Edition ☐ ClassBattle Student Edition ☐

¿En qué pantalla se produce el fallo?

¿En qué momento ha ocurrido? ¿Qué estabas haciendo? ¿Y el resto de compañeros?

¿Ha mostrado algún texto de error? ¿Cuál?

¿Se puede continuar con el Juego?.
Si ☐ No ☐

7.3 Puesta en marcha en MacOS X

Las apps desarrolladas y testadas en este proyecto han utilizado de forma base las versiones de las aplicaciones y frameworks que se muestran a continuación en un ordenador de la marca Apple, modelo iMac o MacBook Pro.

```
    @ionic/cli-utils  : 1.13.0
    ionic (Ionic CLI) : 3.13.0

global packages:

    cordova (Cordova CLI) : 7.0.1

local packages:

    @ionic/app-scripts : 3.0.0
    Cordova Platforms  : ios 4.4.0
    Ionic Framework    : ionic-angular 3.7.1

System:

    ios-sim : 6.1.2
    Node    : v7.7.2
    npm     : 4.1.2
    OS      : macOS Sierra
    Xcode   : Xcode 9.0 Build version 9A235

Browser:
    Safari : 11.0.2
    Chrome : 63.0.3239.84
```

En el siguiente apartado se muestran los pasos a seguir para obtener un entorno de desarrollo similar y que permita la ampliación del proyecto o su ejecución para pruebas.

7.3.1 Aplicaciones

Para poder modificar y correr el código fuente es necesario en primer lugar descargar e instalar **npm** desde la página del desarrollador Nodejs [19]. Para evitar problemas a la hora de trabajar con el entorno **es recomendable** que se instalen las mismas versiones de software que las indicadas en el apartado anterior.

Figura 24. Página web de node.js

A continuación descargaremos e instalaremos el entorno de desarrollo Xcode 9 desde la AppStore de macOS Sierra.

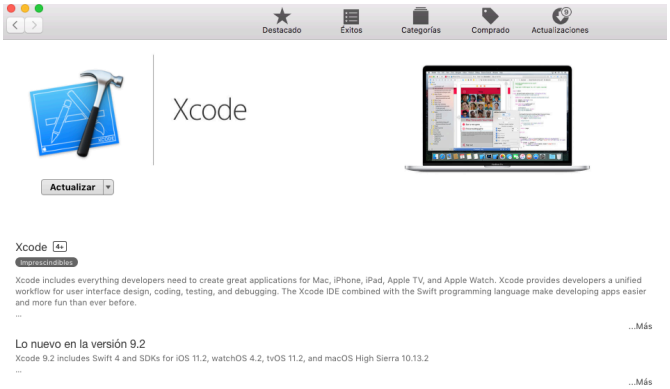

Figura 25. AppStore de macOS Sierra con la página de descarga de Xcode.

Posteriormente desde el terminal hay que ejecutar el comando con privilegios de administrador:

```
> sudo npm install –g cordova@7.0.1 ionic@3.13.0
```

Esta instrucción instala el framework cordova e ionic de una sola vez.

Una vez hecho esto, es necesario descomprimir cualquiera de los dos proyectos y desplazarse desde el terminal hasta la carpeta en la que se encuentran los ficheros desempaquetados. Una vez allí hay que ejecutar el comando:

```
> sudo npm install
```

Esta instrucción descarga la versión de todos los plug-ins necesarios para la correcta ejecución de la aplicación. A partir de aquí tenemos varias opciones:

- Ejecutar la app en modo simulación a través del navegador

```
> ionic serve -t ios
```

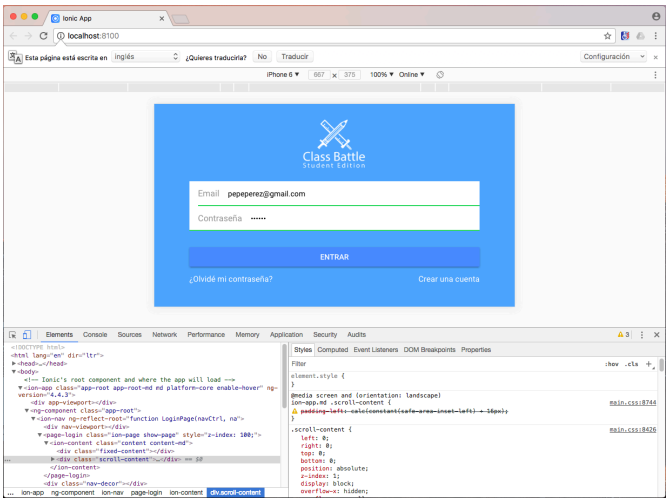

Figura 26. Ejecución de ClassBattle Student Edition en un navegador web.

Para que se visualice correctamente hay que poner el navegador en **modo depuración** e indicarle que se está visualizando en un dispositivo **iPhone**, en otro caso aunque funciona, **no se visualizará correctamente**.

- Crear los ficheros de aplicación

```
> ionic cordova build ios --release --prod
```

Al ejecutar este comando se crea en la carpeta platforms/ios/build/emulator/ un fichero .app. Dicho fichero contiene la app que se puede ejecutar en el simulador.

- Ejecutar en modo simulación en el Simulator de Xcode.

  - Estando en la carpeta de la app es necesario instalar ios-sim. Esta operación debe realizarse una sola vez.

```
> cd platforms/ios/cordova && npm install ios-sim@latest
> cd ../../../
> cordova platform rm ios
> cordova platform add ios@4.4.0
```

  - Una vez hecho esto ya es posible ejecutar la app en el emulador de iOS de Xcode que queramos escribiendo el siguiente comando y sustituyendo el parámetro target por el dispositivo que deseemos. En este ejemplo hemos utilizado un iPhone 6.

```
> cordova emulate ios -target=”iPhone-6”
```

Figura 27. Ejecución de ClassBattle Teacher Edition en un emulador iOS.

7.3.2 Base de Datos

Para acceder a la consola de gestión de la base de datos es necesario acceder a la siguiente URL:

https://console.firebase.google.com/project/classbattle-ba6af/database/classbattle-ba6af/data

Utilizando el siguiente usuario y contraseña:

Usuario: gcoronadomorontfm@gmail.com
Contraseña: uoc1234*

ClassBattle

Manual de Usuario

UOC
Universitat Oberta de Catalunya

Índice

1 ¿Qué es ClassBattle? 3
2 Requisitos 3
3 Darse de alta en el sistema 3
4 Ingresar en el sistema 5
5 Crear un juego nuevo (sólo Teacher Edition) 5
6 Comenzar juego (sólo Teacher Edition) 6
7 Unirse a una partida (sólo Student Edition) 8
8 Conocer el GameID durante el juego (sólo Teacher Edition) 10
9 Activar pulsadores de respuesta (sólo Teacher Edition) 10
10 Responder preguntas (sólo Student Edition) 10
11 Determinar la corrección de una respuesta (sólo Teacher Edition) 11
12 Mover ficha en el tablero (sólo Student Edition) 12
13 Cambiar nombre del usuario 12

1 ¿Qué es ClassBattle?

ClassBattle es un juego de preguntas y respuestas a través de dispositivos móviles para su desarrollo en un aula. Consta de dos aplicaciones:

- **ClassBattle Student Edition**. Diseñada para jugar, es la que deben utilizar los alumnos.
- **ClassBattle Teacher Edition**. Diseñada para organizar y dirigir un juego, es la que deben utilizar los profesores.

El juego consiste en un tablero en la que cómo máximo pueden participar 4 jugadores o grupos de jugadores representados por una ficha. Tras una pregunta al aire del profesor se activan los pulsadores para saber que alumno conoce la respuesta en primer lugar. Si se responde de forma correcta, se puede mover la ficha una posición anexa en el tablero. Es posible mover sobre casillas previamente conquistadas por otros jugadores. Ganará aquel que posea más casillas en su poder cuando el profesor determine que se acaba el juego.

2 Requisitos

- iPhone 5, 5s, 6, 6 Plus, 6s Plus, 7, 7 Plus, 8, 8 Plus, X.
- Sistema iOS 8 o posterior
- Conexión a Internet

3 Darse de alta en el sistema

Una vez que se ha accedido a cualquiera de las dos apps es necesario crear una cuenta que nos identifique dentro de las mismas. Para ello basta con pulsar el botón **Crear una cuenta**.

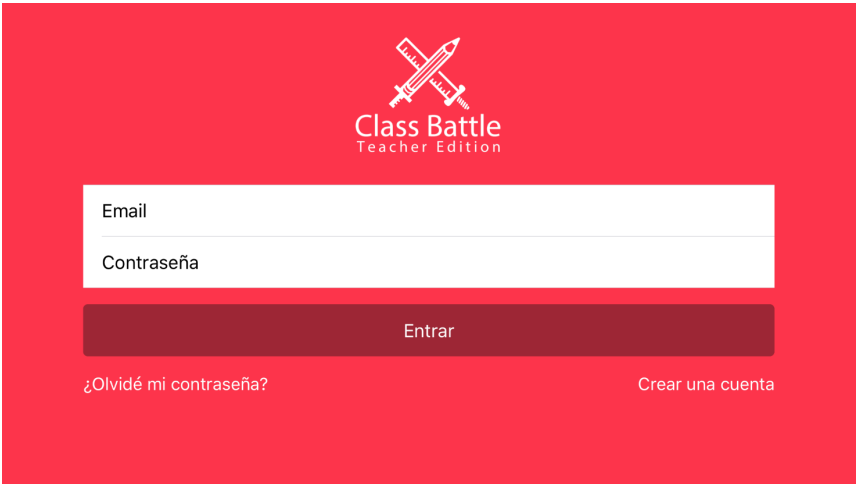

Figura A.1. Pantalla de login de ClassBattle Teacher Edition.

A continuación se nos mostrará una pantalla en la que deberemos proporcionar nuestro nombre, un email que nos identificará como usuario y una contraseña.

Figura A.2. Pantalla de creación de cuenta de usuario

4 Ingresar en el sistema

Una vez que dispongamos de un email y contraseña creado por nosotros mismos, debemos proporcionarlo en la primera pantalla. Al hacerlo accederemos a otra pantalla con un listado de juegos en la Teacher Edition o una página de acceso a juego en la Student Edition.

5 Crear un juego nuevo (sólo Teacher Edition)

Desde la pantalla con el listado de juegos creados, es posible añadir un nuevo juego. Para ello basta con pulsar el símbolo + situado en la parte superior derecha.

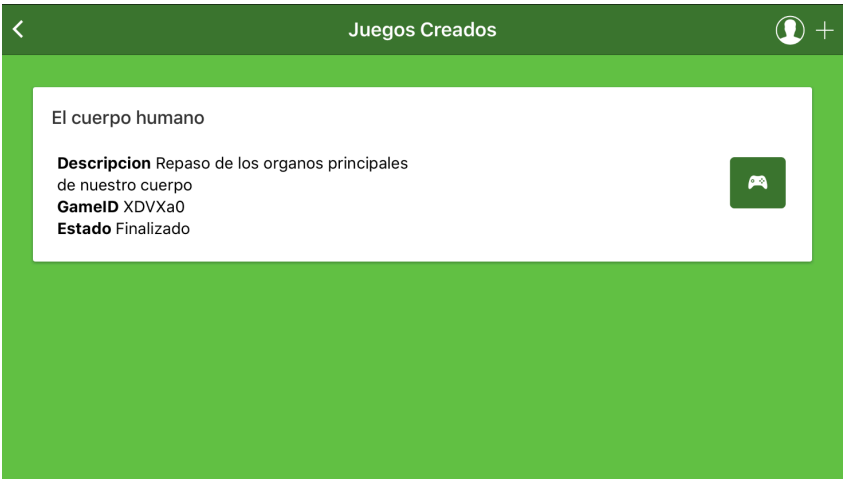

Figura A.3. Pantalla del lista de juegos disponibles

Esto nos abrirá una nueva pantalla en la que deberemos proporcionar un nombre y una descripción.

Figura A.4. Pantalla de creación de juego

6 Comenzar juego (sólo Teacher Edition)

Desde la pantalla del listado de juego debe pulsarse sobre el botón formado por un mando de videojuego. Esto nos mandará a una pantalla de espera en el que se mostrará un código llamado GameID que debe de proporcionarse a los alumnos para que puedan unirse a la partida.

Figura A.5. Pantalla de espera vacía para comienzo del juego BattleClass Teacher Edition

Se dispondrá de un botón **Comenzar** que estará deshabilitado hasta que 2 o más jugadores hayan decidido unirse a la partida.

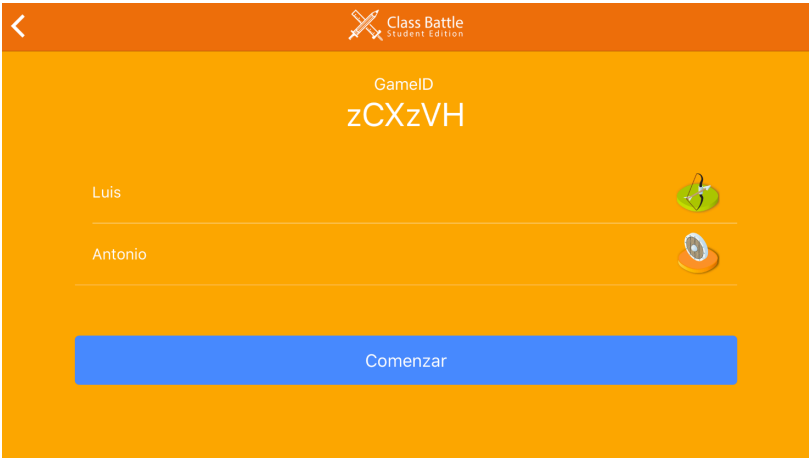

Figura A.6. Pantalla de espera llena para comienzo del juego BattleClass Teacher Edition.

Figura A.7. Tablero de juego en la versión Teacher Edition

Cuando se determine que todos los jugadores se encuentran dentro deberemos pulsar sobre el botón **Comenzar**. Esto nos llevará a la pantalla de tablero del profesor.

7 Unirse a una partida (sólo Student Edition)

Una vez identificados se nos mostrará una pantalla en la que debemos especificar el GameID que nos proporcione el profesor y el nombre que queremos que nos identifique durante la ejecución del juego.

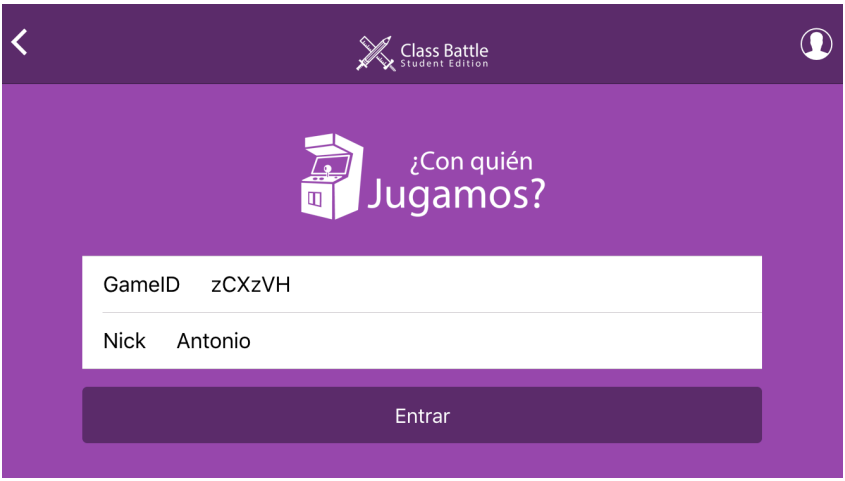

Figura A.8. Pantalla de inclusión a juego en la versión Student Edition

Al pulsar sobre el botón Entrar accederemos a una pantalla donde esperaremos a que todos los jugadores se unan a la partida.

Figura A.9. Pantalla de espera en la versión Student Edition

Aquí no podremos hacer nada, tan sólo esperar a que el juego sea activado desde la Teacher Edition. Cuando esto ocurra pasaremos a la pantalla de tablero del estudiante.

Figura A.10. Tablero de juego en la versión Student Edition.

8 Conocer el GameID durante el juego (sólo Teacher Edition)

Desde el tablero del profesor es posible conocer el GameID del juego en curso pulsando sobre el icono **<···>**.

9 Activar pulsadores de respuesta (sólo Teacher Edition)

Cuando el profesor quiera que los alumnos respondan una pregunta debe lanzarla al aire y posteriormente pulsar el botón de doble diálogo situado en la parte superior izquierda del tablero de profesor. Esto activará en los dispositivos de los alumnos unos pulsadores. En el momento en que se detecte una pulsación, se mostrará el nombre y la ficha del que lo hizo.

10 Responder preguntas (sólo Student Edition)

Cuando el profesor lo determine se mostrará en el dispositivo un botón gigante.

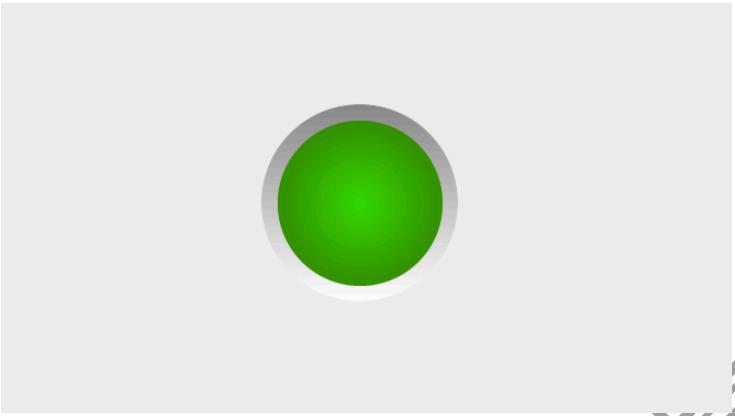

Figura A.11. Pulsador de la versión Student Edition.

Púlsalo lo más rápido que puedas si sabes la respuesta a la pregunta que el profesor planteó anteriormente. El que lo haga antes tendrá el derecho a responder.

En el momento que el sistema determine el primero en pulsar mostrará en la pantalla su nombre y su ficha. En ese instante guarda tu posible respuesta (si la conoces y no has sido el primero), puede ser que el que tenga el turno falle y nuevamente el profesor active los pulsadores.

Figura A.12. Pantalla de permiso para responder en la versión Student Edition

11 Determinar la corrección de una respuesta (sólo Teacher Edition)

Cuando un alumno/a activa uno de sus pulsadores se mostrará su nombre y dos botones. Esos botones determinan si ha acertado (mano verde hacia arriba) o fallado (mano roja hacia abajo). Si se pulsa la verde se permite mover al jugador sobre el tablero. Si se pulsa la roja se volverán a activar los pulsadores en los dispositivos del alumnado.

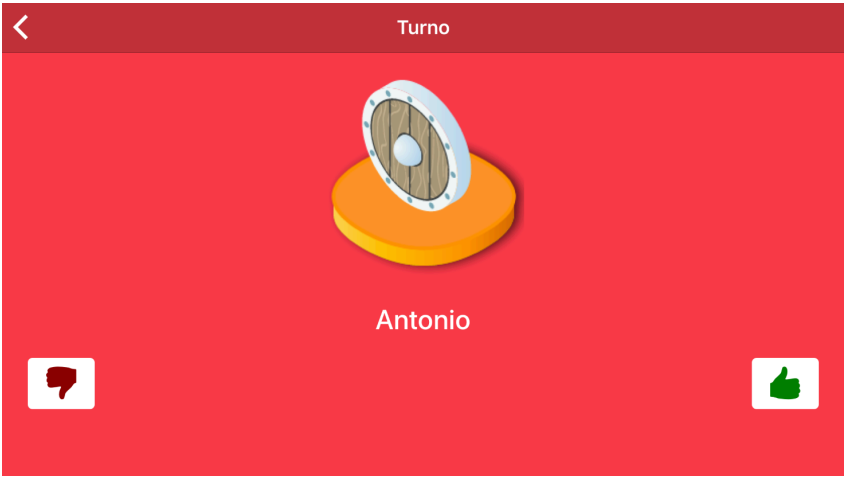

Figura A.13. Pantalla de corrección de respuesta en la versión Teacher Edition

12 Mover ficha en el tablero (sólo Student Edition)

Si el profesor ha determinado que una de nuestras respuestas es correcta, desde la pantalla del tablero podremos pulsar sobre cualquier casillas próxima a nuestra ficha para conquistarla. Da igual que esté vacía (color gris) o pertenezca al color de otro jugador. En ningún caso podemos mover sobre una ficha de otro jugador, tan sólo sobre una casilla en la que no haya ninguna otra ficha.

13 Cambiar nombre del usuario

Desde la pantalla de unirse a una partida en la Student Edition o en la del listado de juegos en la Teacher Edition, es posible acceder a una pantalla donde podemos cambiar el nombre que utilizamos a la hora de registrarnos. Este nombre sólo se usa a nivel informativo. Para ello

debes pulsar sobre el icono de la sombra de un hombre incluida en un circulo situada en la parte superior izquierda de la pantalla.