Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

APLICACIÓN MÓVIL PARA HACER EJERCICIO FÍSICO BASADA EN UN JUEGO

Alumno/Alumna: Minchero, Rosende, Mikel Asier

Director/Directora: Ferro, Vázquez, Armando

Curso: 2019-2020

Fecha: lunes, 4, noviembre, 2019

Índice

Lista de tablas

Lista de ilustraciones

Resumen trilingüe

Resumen

Debido al auge en el sector del fitness y del ejercicio en general en los últimos tiempos, la demanda de aplicaciones móviles relacionadas con el deporte ha aumentado considerablemente. Sin embargo, no abundan aplicaciones que combinen un juego y el entretenimiento que conlleva con la práctica del ejercicio. Esta es la carencia que viene a completar la aplicación desarrollada en este proyecto, un juego basado en cartas con penalizaciones físicas para ejercitar el cuerpo y pasar un buen rato al mismo tiempo.

Laburpena

Azken aldian, fitness eta ariketa fisikoaren sektorean izandako gorakada dela eta, kirolari lotutako aplikazio mugikorren eskaria nabarmen handitu da. Hala ere, ez daude ariketa praktikarekin batera joko bat combinatzen dituzten aplikazio asko. Proiektu honetan garatutako aplikazioa arazo hay betetzeko asmoa du. Hau da, gorputza egikaritzeko, eta aldi berean ondo pasatzeko, zigor fisikoak dituen kartetan oinarritutako joko bat.

Summary

Due to the boom in the fitness and exercise sector in general in recent times, the demand for mobile applications related to sports has increased considerably. However, there are not many applications that combine the entertainment that comes with a game and the practice of exercise. This is the lack that aims to fullfill the application developed in this project, a game based on cards with physical penalties to exercise the body and have a good time at the same time.

Memoria

Introducción

El presente documento muestra la memoria relativa al Trabajo de Fin de Grado (TFG) “Aplicación móvil para hacer ejercicio físico basada en un juego” por parte de Mikel Asier Minchero Rosende, siendo este alumno del grado en Ingeniería en Tecnología de Telecomunicación ofertado por la UPV/EHU (Universidad del País Vasco / Euskal Herriko Unibertsitatea). Dicho proyecto consiste en la realización de una aplicación para dispositivos móviles para hacer ejercicio físico.

El fin último del documento es el de explicar el trabajo realizado a lo largo del desarrollo de este proyecto y poner en conocimiento de los interesados la solución técnica desarrollada.

Se empezará introduciendo las consideraciones iniciales a tener en cuenta del proyecto desarrollado para después explicar en profundidad la solución dada. Posteriormente, se proseguirá con la explicación de la evolución del proyecto a lo largo del tiempo y de la forma en la que ha sido organizado su desarrollo. Por último, se expondrán las conclusiones pertinentes.

Contexto

El gran crecimiento de las redes sociales ha marcado la última década. Aplicaciones como WhatsApp, Instagram, Twitter o Facebook se han vuelto parte del día a día de la mayoría de las personas.

Sin ir más lejos, en España se estima que un 88% de la población usa WhatsApp, un 87% usa Facebook, un 68% usa YouTube, un 54% Instagram y un 50% Twitter. Es decir, prácticamente toda la población tiene un perfil en la red que puede ser visitado y comentado, con fotos y demás información sobre uno mismo. Esto por un lado favorece en gran medida la comunicación entre las personas, pero por otro lado crea un nivel de exposición a nivel público que no existía antes de su creación.

Dicho aumento ha provocado en una gran cantidad de gente (sobre todo la más activa en redes sociales) una tendencia a proyectar lo que se conoce como "buena imagen". Uno de los componentes de dicha imagen consiste en hacer ver que uno se encuentra en un buen estado de forma.

Por otro lado, cada vez hay más información a nuestro alcance en campos como la alimentación, hábitos de vida saludables, ejercicio, etc. Todo el mundo tiene al alcance de la mano la gran variedad de beneficios que aporta llevar un estilo de vida en el que se ejercite el cuerpo de manera habitual.

La combinación de estos dos factores ha provocado un auge en el sector del fitness y el mundo del ejercicio en general. Hoy en día es mucho más habitual que la gente se apunte a un gimnasio o decida hacer algo de deporte de lo que lo era hace 15 años, sea por los factores mencionados anteriormente o porque simplemente le gusta hacer deporte y ejercitarse.

Sin ir más lejos, la facturación del fitness en el mercado español ha aumentado un 7.5% desde 2015 debido al aumento del volumen de clientes, siendo el quinto mayor mercado de fitness en Europa por detrás de Alemania, Reino Unido, Francia e Italia. Y estos datos son únicamente en lo referente a gimnasios, habría que sumarle el incremento de personas que toman otras alternativas como puede ser ir a correr, hacer bici, escalada, apuntarse a algún deporte de equipo, y un largo etcétera.

Dicho auge se ha visto reflejado en todos los ámbitos, desde cadenas multinacionales de gimnasios hasta aplicaciones para dispositivos móviles orientadas al mundo del fitness o del deporte en general. A día de hoy, existe una gran variedad de opciones en cuanto a control de dietas, guías de ejercicios, seguimiento semanal de entrenamiento… Todas estas aplicaciones están orientadas a un perfil de usuario disciplinado que hace ejercicio con regularidad.

No obstante, pese a que cada vez más gente decide hacer ejercicio, hay una ingente cantidad de personas que padecen problemas de sobrepeso u obesidad. La OMS (Organización Mundial de la Salud) define el sobrepeso en adultos con tener un IMC (Índice de Masa Corporal) igual o superior a 25 y a obesidad con tener un IMC igual o superior a 30, siguiendo otros baremos para menores de 19 años.

Dicho problema se ve reflejado en las estimaciones recientes que realizó la OMS a nivel mundial en 2016. En dichas estimaciones, se estableció que el 39% de los adultos (39% hombres y 40% mujeres) padecían de sobrepeso. También se señaló que el 13% de la población adulta (11% de hombres y 15% de mujeres) padecían de obesidad. Esto supone casi el triple de obesos comparado con 1975.

Un IMC elevado es un importante factor de riesgo de enfermedades no transmisibles. Por ejemplo: enfermedades cardiovasculares, diabetes, trastornos del aparato locomotor y algunos cánceres (endometrio, mama, ovarios, próstata, hígado, vesícula biliar, riñones y colon). Por ello, es necesario intentar prevenir y facilitar en la medida de la posible que esta situación vaya a más.

Conociendo esto, existe otro perfil de persona. Está compuesto por gente que reconoce los beneficios que puede aportar el ejercicio físico, pero o no se pone a ejercitarse o deja de hacerlo en caso de haber empezado porque le parece aburrido o tedioso. El clásico ejemplo de esto sería una persona que se apunta a un gimnasio porque quiere ponerse en forma y lo deja un mes después porque se aburre. Por otro lado, hay personas que no pueden o no quieren pagar un gimnasio y dejan de ejercitarse porque piensan que hacer deporte sin el equipamiento de esta clase de centros es una pérdida de tiempo.

Esta clase de persona cuando decide hacer ejercicio de una manera social, tiene las alternativas de siempre. Es decir, jugar un partidillo con los amigos, salir a correr o en bici con ellos, ir al monte, etc. Y son buenas opciones, desde luego. Pero desde el sector de las aplicaciones móviles no existe prácticamente ninguna aplicación que cubra esta necesidad. Faltan aplicaciones que supongan una alternativa y que propongan realizar ejercicio de una forma estimulante y divertida como puede ser un juego que implique realizar ejercicio físico.

Objetivos y alcance del trabajo

Objetivo

El objetivo de este proyecto es desarrollar una aplicación para dispositivos móviles (tanto iOS como Android) basada en un juego que incluya la realización de pruebas físicas. De esta manera, los usuarios podrán hacer ejercicio de una manera distendida y casual con otras personas siempre de una manera no lesiva (que será indicada en la aplicación).

Alcance del trabajo

En primer lugar, será necesario definir qué tecnologías se van a emplear para desarrollar el software en base a las especificaciones del producto. Para ello, se realizará un análisis de alternativas y una vez determinadas las tecnologías, será necesario pasar por una fase de formación en caso de que el alumno no domine las tecnologías seleccionadas.

Una vez el alumno tenga claras las tecnologías que va a utilizar y un conocimiento que le permita emplearlas con soltura y eficacia, se pasará al diseño de la aplicación. En primer lugar, se definirá la arquitectura del proyecto software de tal forma que luego haga uso de cada componente del diseño detallado de una forma eficiente y que cumpla con las especificaciones previamente definidas. Una vez escogida la arquitectura, se pasará al diseño detallado de cada componente de la aplicación.

Finalmente, al conseguir una aplicación funcional, se procederá a realizar la fase de pruebas en la que se tratarán de descubrir la mayor cantidad de errores o problemas de rendimiento con el objetivo de corregirlos y entregarle al usuario final una aplicación lo más libre de errores posible.

Beneficios que aporta el trabajo

A la hora de describir los beneficios que proporciona esta aplicación podemos discernir dos bloques principales. Por un lado, están los beneficios que aporta la realización de ejercicio físico en personas de todas las edades. Por otro, el componente divertido propio de un juego.

Beneficios aportados por hacer ejercicio físico

Realizar de forma regular y sistemática una actividad física ha demostrado ser una práctica muy beneficiosa en la prevención, desarrollo y rehabilitación de la salud, a la vez que ayuda al carácter, la disciplina y a la toma de decisiones en la vida cotidiana.

El ejercicio físico, ya sea de corta o larga duración, contribuye a establecer un bienestar mental, mejorando la autonomía de la persona, la memoria, rapidez de ideas, etcétera, y promoviendo sensaciones como el optimismo o la euforia, al tiempo que se mejora la autoestima de las personas, lo que produce beneficios en diferentes enfermedades como la osteoporosis, la hipertensión o las crisis diabéticas.

Entre los beneficios que aporta hacer ejercicio, diferenciamos dos bloques principales. En primer lugar, tenemos los beneficios biológicos, en segundo lugar, los beneficios psicológicos.

Beneficios biológicos

- La actividad aeróbica o los ejercicios que necesitan de más oxigeno como caminar, trotar, nadar, bailar, esquiar, pedalear, favorecen el sistema cardiovascular, disminuyen la presión sanguínea y mejoran la circulación, lo que reduce el riesgo de ataques cardíacos y accidentes cerebrovasculares.
- Ayuda en la prevención de cáncer de colon, ya que acelera el paso de los desechos por los intestinos. Asimismo, al regular los niveles hormonales, puede contribuir a evitar el cáncer de mama y de próstata.
- El impacto en los huesos es muy positivo. En los niños puede aumentar la densidad ósea, en los adolescentes los fortalece y en la vida adulta retrasa la degeneración. Puede prevenir la osteoporosis.
- Con el ejercicio el sistema inmune se acelera de manera temporal, aumenta su capacidad y defensas para el organismo.
- Realizar actividad física con regularidad ayuda a mantener un nivel saludable de azúcar en la sangre, que no sólo contribuye a controlar el peso, sino a evitar el riesgo de padecer diabetes tipo 2.
- El flujo de oxígeno al cerebro aumenta, por lo que la capacidad de aprendizaje, concentración, memoria y estado de alerta pueden mejorar de manera considerable.
- Beneficia la calidad del sueño. Un estilo de vida activo puede significar un sueño más reparador y profundo, que estimula la concentración en el día, aumenta la productividad y propicia un mejor estado de ánimo.
- En los músculos, no sólo aumenta la oxigenación, tono, fuerza y volumen, también favorece la flexibilidad, la fuerza de los tendones y los ligamentos.
- Con el ejercicio se queman calorías, lo cual ayuda a controlar el balance energético diario, y por ende a controlar el peso corporal.

Beneficios psicológicos

- Aumenta tu rendimiento. Los trabajadores que hacen deporte con regularidad logran eliminar mejor sus niveles de estrés. De esta forma, al desconectar, logran volver a aumentar el rendimiento. Este reinicio es especialmente efectivo si se realiza al mediodía o a primera hora de la mañana.
- Mejora la autoestima. El deporte ayuda a que nos sintamos mejor con nosotros mismos, no solo por el efecto de lograr mejorar nuestro cuerpo. A nivel mental nos dota de validez, capacidad para trabajar por una meta y mejora nuestra autopercepción.
- Mejora las relaciones. Al dotarnos de una mayor confianza en nosotros mismos, nos predispone al contacto social. Nos sentimos con mejores herramientas y disfrutamos más con la interacción social.
- Previene enfermedades cognitivas. Si nuestro cerebro se encuentra siempre estimulado, logramos que a largo plazo no se atrofie. El ejercicio físico tiene este efecto, tanto a nivel corporal como mental. Tener activado el cerebro trae consigo la ventaja de prevenir enfermedades degenerativas, como algunas demencias.
- Aumenta el rendimiento cerebral. El ejercicio tiene el poder de producir más y mejores conexiones entre nuestras neuronas. De esta forma, mejora el rendimiento y la capacidad cerebral. Esto nos predispone a que no haya un deterioro en la memoria y podamos aprender de una forma más rápida y eficaz.
- Aumento de la felicidad. El efecto más directo sobre nuestro bienestar lo tiene al activar la producción de endorfinas. Este neurotransmisor reduce el dolor y genera sensaciones estables y elevadas de felicidad. El ejercicio físico está pautado en trastornos del estado de ánimo, como la depresión. Activa el cuerpo y estimula la energía y la sensación de bienestar.
- Menos ansiedad. La actividad física nos ayuda a calmar el cuerpo y la mente. Tiene dos funciones. Por un lado, acaba calmando la tensión muscular una vez que hemos estimulado el cuerpo con el ejercicio. Por otro lado, nos distrae de preocupaciones durante el tiempo que lo estamos practicando. Esto hace que sintamos liberación y reduzcamos el estrés.
- Sensación de control. Cuando establecemos unos hábitos, independientemente de los que sean, y movilizamos nuestras acciones para lograr una meta, ganamos en sensación de control. Tenemos responsabilidad sobre nuestras acciones y vemos sus resultados. El deporte logra esto a nivel psicológico.

Diversión

Al ser una aplicación basada en un juego, genera una predisposición mayor que, por ejemplo, el hecho de salir a correr solo (hablando siempre de un perfil de persona no propensa a ejercitarse). La aplicación está pensada para ser usada con más gente y generar un entorno competitivo a la par que divertido que haga pasar un buen rato al mismo tiempo que se hace ejercicio. De esta manera, la aplicación se postula como una alternativa más para hacer deporte de cara a todas esas personas a las que no les motiva tanto hacer deporte pero quieren incluirlo en sus vidas.

Especificaciones y requisitos del producto

Perspectiva del producto

Este producto está pensado para ser utilizado en ámbitos en los que haya el suficiente espacio para hacer ejercicio físico, sin ser necesario ningún tipo de equipamiento. Por ejemplo: un parque, una campa, un patio… Es por ello que debe ser una aplicación para dispositivos móviles.

Especificaciones del producto

La aplicación debe cumplir una serie de especificaciones de cara al usuario:

- La aplicación no debe depender de que el usuario tenga conectividad para su funcionamiento.
- La aplicación deberá tener una página que haga de menú principal. Desde ella, el usuario puede seleccionar si desea empezar un partido, ver la lista de jugadores registrados en la aplicación o consultar las reglas del juego. Además, deberá poder activar o desactivar el sonido de la aplicación.
- Una página en la que se muestre la lista de jugadores registrados en la aplicación. Además, debe dar la opción de añadir un nuevo jugador o eliminar los ya existentes.
- Una página para añadir un nuevo jugador. En ella, el usuario deberá introducir un nombre y podrá introducir de manera opcional una imagen asociada a ese jugador. Si decide añadir una imagen, la aplicación le dará la opción de seleccionar una imagen de la galería del dispositivo o sacar una fotografía. Si el usuario decide no añadir una imagen asociada al jugador, se le asociará una imagen genérica.
- Una página en la que se detallen las reglas del juego.
- Una página previa a la página de partido. En ella, se definirán parámetros previos al inicio del juego como número de jugadores, equipo de cada jugador, duración del propio partido y nivel de dificultad (3 niveles).
- Una página destinada al juego en sí. En dicha página, se mostrarán varios elementos, a ser posibles todos en la misma vista, sin que el usuario tenga que deslizar la pantalla hacia arriba o hacia abajo para ver todos los datos:
    - Los nombres de cada equipo y las fotografías de los jugadores de cada equipo.
    - La suma acumulada en cada punto del juego por cada equipo. Tanto este como el punto anterior deberán mostrar la información de la parte superior de la pantalla (correspondiente a uno de los dos equipos) girado 180º, ya que habrá un equipo a cada lado de la pantalla del dispositivo.
    - El marcador del partido.
    - El triunfo de ese juego.
    - Una imagen que represente la baraja de cartas. Cuando el usuario toque la imagen de dicha baraja, una carta al azar de la baraja española se mostrará y se añadirá a la suma de ese equipo para ese punto. Además, realizará una serie de operaciones:

- Al pulsar, se reproducirá un sonido similar al de una raqueta golpeando una bola de tenis. Si se termina un punto, reproducirá otro sonido de público aplaudiendo. Si se termina un set, reproducirá otro sonido distinto de público vitoreando.
- Esa carta no podrá volver a salir hasta que se reinicie la baraja.
- Si era la última carta por mostrar en ese punto, calculará para quién ha sido el punto y se lo sumará en el marcador.
- Si ese punto ha significado un juego, se reflejará en el marcador.
- Si ese juego ha significado un set, se reflejará en el marcador. Y así progresivamente.
- Además, en cada punto, juego, set o partido se mostrará una alarma al usuario indicándole quién ha ganado el punto, juego, set o partido, el marcador dentro de ese propio punto y el "castigo" que tiene que hacer el equipo perdedor. Dicho castigo variará en función del nivel de dificultad seleccionado previamente y consistirá en la realización de una o varias pruebas físicas.

Requisitos

Acceso a periféricos del dispositivo

La aplicación debe ser capaz de sacar fotos mediante la cámara del dispositivo (en caso de que la tenga), acceder a la galería y seleccionar una imagen. Por otro lado, también deberá ser capaz de reproducir un sonido en determinados momentos del juego a través del altavoz del dispositivo (en caso de que lo tenga).

Funcionalidad para almacenar datos de forma local

La aplicación debe ser capaz de almacenar la lista de usuarios que se vayan registrando en la misma de manera local, de tal forma que se pueda acceder posteriormente a esos datos incluso después de haber cerrado la aplicación.

Limitar la orientación de la pantalla del dispositivo a vertical

La aplicación es un juego diseñado para que todos los elementos de un partido se muestren en una misma pantalla, con una distribución concreta que no debe variar con la orientación del teléfono. Siempre deberá mostrarse en orientación vertical.

Funciones

La función principal de la aplicación es la de un juego que involucre la realización de ejercicio físico. Para ello, existen tres bloques diferenciados dentro de la propia aplicación.

- Partido
  - Creación de partido
  - Partido
- Jugadores
  - Ver jugadores
  - Añadir jugador
  - Eliminar jugador
- Reglas

Por un lado, está el bloque del partido. Dicho bloque incluye primero una página en la que se definen los parámetros previos al partido: número de jugadores, nombre de los equipos, jugadores por equipo, tipo de partido y nivel de dificultad. Después, se va a la pantalla del partido en la que, empleando los datos recogidos en la página de creación de partido, se ejecuta toda la lógica del juego. En dicha pantalla, en los casos en los que proceda, se reproducirá un sonido previamente almacenado en la aplicación.

Por otro lado, tenemos el bloque de los jugadores. Dicho bloque incluye una página en la que se muestran los jugadores registrado de forma local en el propio dispositivo, mostrando su foto asociada dentro de la propia galería del dispositivo y su nombre. Existe también la opción de agregar un nuevo jugador. En esta página, se puede escoger entre sacar una fotografía en el momento o seleccionar una imagen de la galería del dispositivo y se añade un nombre al jugador. Además, existe la opción de eliminar a cualquier jugador del almacenamiento local del dispositivo.

Por último, está el bloque de las reglas, en el que se definen las reglas del juego.

Requisitos de rendimiento

La aplicación debe navegar entre las distintas páginas que la componen y ejecutar el código de cada una de ellas de una manera fluida para el usuario sin colgarse. Además, las funciones que impliquen acceso a alguno de los periféricos del dispositivo, así como el almacenamiento local en el mismo no deben dar problemas.

Atributos de la aplicación

La aplicación debe ser completamente independiente de que el usuario tenga conectividad a la red o no. Es por ello que todos los archivos deben estar almacenados a nivel local en el propio dispositivo. De esta forma, el usuario podrá hacer uso de la aplicación en cualquier circunstancia.

Análisis de alternativas

Esta aplicación está destinada a ser utilizada en cualquier parte, por lo que debe ser una aplicación apta para dispositivos móviles. Para ello, es necesario escoger entre las diversas opciones de desarrollo en función de los criterios que mejor se ajusten al proyecto.

Tipo de aplicación

En primer lugar, hay que decidir el tipo de aplicación: aplicación web, aplicación nativa o aplicación híbrida.

Aplicación web

Las aplicaciones web destacan por su versatilidad. Este tipo de aplicaciones son accesibles para todos los usuarios con independencia del dispositivo que utilicen siempre y cuando tengan conexión a Internet.

Empleando tecnologías como HTML5, SCSS y JQuery se puede proporcionar al usuario una experiencia bastante similar al de una aplicación nativa sin necesidad de que ésta sea instalada en el dispositivo.

La mayor ventaja de este tipo de aplicación radica en que el coste y los plazos de entrega son inferiores respecto a los otros dos tipos de aplicaciones mencionados en este apartado. No es necesario cumplir todas las especificaciones de los diferentes sistemas operativos para lanzarlas. Además, requiere menos conocimientos técnicos.

Por otro lado, este tipo de aplicaciones da bastantes problemas al intentar emplear elementos propios del dispositivo (como por ejemplo la cámara, el micrófono, GPS…). Además, la velocidad de carga será inferior al de una aplicación nativa. Esto es debido a en parte la tecnología utilizada y en parte a la calidad de la conexión a internet del usuario (queda descartado el modo offline).

Aplicación híbrida

Las aplicaciones híbridas combinan la compatibilidad con cualquier dispositivo de las aplicaciones web y un mejor acceso a las funcionalidades propias de éste.

Al igual que sucede en las aplicaciones web, este tipo de aplicaciones tiene como principal ventaja su compatibilidad con cualquier dispositivo (básicamente es una aplicación web encapsulada). Pese a no tener herramientas estandarizadas para acceder a los periféricos del dispositivo, existe la posibilidad de programar manualmente dichas herramientas.

Además, su coste y tiempo de desarrollo (al igual que sucede en una aplicación web) es menor y requiere menos conocimientos técnicos por parte del desarrollador que en una aplicación nativa. El usuario tendría una aplicación instalada en su dispositivo, al igual que una aplicación nativa.

La principal desventaja de este tipo de aplicación frente a una nativa es su menor rendimiento y velocidad de carga.

Aplicación nativa

Debido a que las aplicaciones nativas son desarrolladas con las herramientas y el kit de desarrollo de aplicaciones propias de cada sistema operativo, ofrecen las mejores prestaciones en eficiencia, estabilidad, velocidad, experiencia de usuario… Están desarrolladas y optimizadas para cada sistema operativo.

Tienen mejor integración con las funcionalidades de hardware (cámara, micrófono…) que las demás alternativas de este documento. Además, la implementación de dichas funcionalidades es más sencilla que en los otros casos.

Por otro lado, este tipo de aplicaciones tiene un mayor coste de desarrollo, ya que requiere una mayor inversión de tiempo y recursos. Además, es necesario conocimiento específico de cada lenguaje de programación de cada sistema operativo en el que se quiera emplear la aplicación. Las futuribles actualizaciones realizadas a la aplicación deberán ser realizadas para cada plataforma.

Selección

A la hora de decidir el tipo de aplicación, se tendrán en cuenta 3 criterios:

- Experiencia de usuario: Fluidez, velocidad de carga, periféricos… (1-5)
- Tiempo: El Trabajo de Fin de Grado tiene asignados 12 créditos ECTS (300h). En ese tiempo hay que hacer la documentación del proyecto, la aplicación y las pruebas pertinentes para asegurar el correcto funcionamiento de la misma. (Sí-No)
- Conocimiento previo de las tecnologías empleadas. (1-5)

Tipo de aplicación	Experiencia de usuario	Tiempo	Conocimiento previo de las tecnologías empleadas
Aplicación Web	2	Sí	4
Aplicación Híbrida	4	Sí	3
Aplicación Nativa	5	No	1

Tabla 1 - Tipo de aplicación

Lo ideal sería una aplicación nativa, pero al ser una aplicación que va a ser utilizada tanto en dispositivos Android como en iOS, aprender desde cero a programar para estos dos sistemas operativos en el tiempo estipulado para la realización de este proyecto no es viable para el alumno.

Entre una aplicación web y una aplicación híbrida, la aplicación híbrida es una mejor opción. Es más cómoda para el usuario y es más asequible el aprovechamiento de los periféricos del dispositivo. El tiempo disponible es suficiente para que el alumno aprenda a desarrollar este tipo de aplicaciones desde cero, ya que está basado en tecnologías con las que ha trabajado previamente como JavaScript, SCSS y HTML.

Librería de apoyo JavaScript

Una vez decidido que será una aplicación híbrida, hay que decidir de qué librería de JavaScript ayudarse para hacer la aplicación.

Otra opción sería no usar este tipo de librerías y escribir todo el código JS. Sin embargo, esto supondría añadir una complejidad y margen de error innecesarios para realizar funciones que se pueden cubrir perfectamente mediante métodos ya creados en estas librerías.

JQuery

jQuery es una librería que sirve para acceder y modificar el estado de cualquiera de los elementos de una página.

A través de jQuery y los selectores de SCSS (así como los selectores creados por el propio jQuery) se pueden leer y modificar las propiedades de cualquier elemento de una página, suscribirse a eventos que ocurran en esos elementos, etc. Con jQuery se puede manejar cualquier evento que ocurra en esos elementos de una manera mucho más cómoda que escribiendo el código Javascript íntegro.

JQuery es más ligero que Angular. Está destinado a aplicaciones menos complejas y funciona muy bien con la mayoría de navegadores.

AngularJS

AngularJS es un framework basado en JS. Se pueden agregar fácilmente conceptos de Modelo-Vista-Controlador (con lo que ya he trabajado previamente) a los proyectos con la ayuda de AngularJS.

Es un framework muy robusto, por lo que es una herramienta flexible a la hora de desarrollar aplicaciones web. No solo permite una serie de funciones y mecanismos para acceder a los elementos de la página y modificarlos, sino que también ofrece una serie de mecanismos por los cuales extender el HTML, para hacerlo más semántico, incluso ahorrar líneas de código Javascript para hacer las mismas cosas que se pueden hacer con jQuery. Es más adecuado para aplicaciones complejas.

Cabe destacar que dentro de Angular hay una pequeña implementación de JQuery (jqLite). Esta implementación incluye una librería de acceso al DOM (Document Object Model), así como un API de funciones compatibles con todos los navegadores.

Selección

Al ser una comparativa entre dos opciones, simplemente se indicará cuál de las dos es mejor opción en base a cada criterio. Se tendrán en cuenta 3 criterios:

- Flexibilidad: Variedad de opciones que proporciona la herramienta para cada escenario.
- Eficiencia de código: El empleo de un mínimo de líneas de código para realizar una misma función
- Familiaridad con el modelo de programación empleado

Librería	Flexibilidad	Eficiencia de código	Familiaridad con el modelo de programación empleado
JQuery	Peor	Peor	No
AngularJS	Mejor	Mejor	Sí

Tabla 2 - Librería de apoyo JavaScript

Como se puede observar, claramente AngularJS es una mejor opción. Necesita menos líneas de código para realizar las mismas funciones (lo que resta margen de error y, en definitiva, tiempo), además de ser un framework robusto con multitud de opciones. Por último, el hecho de que trabaje con MVC es un añadido importante, ya que el alumno está familiarizado con este modelo.

Framework de desarrollo de aplicaciones móviles

También es necesario decidir el framework a usar para el desarrollo de la aplicación móvil. En el desarrollo de aplicaciones híbridas destaca Ionic sobre el resto, así que en este apartado se analizarán las versiones 3 y 4 de Ionic.

Ionic 3 vs Ionic 4

En el salto a Ionic 4 destaca el uso de Web Components. Un Web Component es un conjunto de web APIs que permiten crear tags HTML reutilizables y encapsulables.

Desde sus inicios Ionic Framework fue diseñado utilizando Angular. A partir de esta versión 4, Ionic será totalmente independiente del framework base, y aunque se pueda seguir usando con Angular, también se podrán emplear otros frameworks (o ninguno). Aún así, yo utilizaré Angular, así que continuaré la comparación asumiendo que empleo Angular junto a Ionic.

Hay grandes cambios en lo relacionado a la navegación y el Router. Ionic 4 ahora usa el Router de Angular.

Los ciclos de vida (conocidos como lifecycles en inglés) que se usaban en Ionic 3 ya no se usarán más en Ionic 4. Ahora se usan los ciclos de vida de Angular.

Además, ya no es necesario importar las páginas y los servicios en el archivo app.module.ts para hacer uso de ellos. Por cada página habrá un módulo propio de esa página.

Por otro lado, mientras que la versión 3 es una versión que ha estado varios años vigente y tiene una gran cantidad de información a la que recurrir en caso de que surjan problemas con el desarrollo, la versión 4 salió hace aproximadamente un año y en ese sentido parte con desventaja.

Selección

Al ser una comparativa entre dos opciones, simplemente se indicará cuál de las dos es mejor opción en base a cada criterio. Se tendrán en cuenta 3 criterios:

- Conocimiento previo
- Disponibilidad de información: foros y demás fuentes de información ante cualquier problema que surja durante el desarrollo, además de la documentación oficial de Ionic.
- Empleo de la tecnología en el futuro

Versión	Conocimiento previo	Disponibilidad de información	Empleo de la tecnología en el futuro
Ionic 3	Ninguno	Mayor	Menor
Ionic 4	Ninguno	Peor	Mayor

Tabla 3 – Versión de Ionic

En este caso, se va a decidir desarrollar la aplicación en Ionic 4. Ya que el alumno se va a formar desde cero, le es indiferente aprender a manejar la versión 3 o la versión 4. Pese a que Ionic 3 tenga mayor disponibilidad de información, la documentación oficial de Ionic 4 es de bastante calidad. Además, después aproximadamente un año con Ionic 4 operativo, hay suficiente información para llevar el proyecto a cabo. Por último, de cara a futuras modificaciones en la aplicación, es mejor que esté programada en la versión más actualizada de Ionic.

Tecnología de encapsulamiento

Una vez creada la aplicación con las tecnologías mencionadas más arriba, lo único que se tendrá será una aplicación web. Para encapsularla en una aplicación nativa del dispositivo y poder acceder a los periféricos del mismo de forma eficiente es necesario implementar una herramienta adicional. Teniendo en cuenta las elecciones tomadas en los apartados anteriores, las dos principales alternativas son Ionic Capacitor y Apache Cordova.

Apache Cordova

Apache Cordova es un proyecto comunitario que permite crear aplicaciones para varias plataformas móviles empleando una misma base de código (desarrollada en HTML5, JS y SCSS). Fue creado a finales de 2011.

Vendría a cumplir las funciones de un servidor web dentro de nuestro propio dispositivo, proporcionando así la capacidad de crear aplicaciones móviles que es ejecuten localmente en el mismo basados en el diseño y desarrollo web.

Por otro lado, existe una amplísima gama de plugins desarrollados por la comunidad desde el inicio de Cordova que permiten acceso a prácticamente cualquier característica del dispositivo en el que la aplicación va a ser ejecutada.

Sin embargo, tiene el contrapunto de que la velocidad de funcionamiento de la aplicación es un poco más lenta que la de una aplicación nativa y algunas aplicaciones en dispositivos de Apple son rechazadas por dicho motivo.

Ionic Capacitor

Ionic Capacitor se podría considerar como el Cordova desarrollado por el propio Ionic y fue creado hace poco más de un año. Al igual que Cordova, es capaz de convertir una aplicación web en una aplicación nativa y acceder a características propias del dispositivo. Si Cordova hace esto mediante plugins, en Capacitor son denominados APIs.

Por otro lado, cabe destacar que mientras Cordova tiene que esperar a que el dispositivo esté listo antes de realizar llamadas a la funcionalidad nativa seleccionada, Capacitor exportará código JS en el arranque de la aplicación para que esto no sea necesario.

Además, Capacitor se instala de forma local en los proyectos, lo que facilita el mantenimiento de diferentes versiones en proyectos múltiples. Su velocidad de funcionamiento es algo superior al de Cordova.

Por último, cabe destacar que en caso de que una API de Ionic Capacitor no cubra el uso de una característica del dispositivo que estemos utilizando, es capaz de implementar una gran cantidad de plugins del propio Cordova sin comprometer el funcionamiento de la aplicación.

Selección

Al ser una comparativa entre dos opciones, simplemente se indicará cuál de las dos es mejor opción en base a cada criterio. Se tendrán en cuenta 3 criterios:

- Opciones de acceso a características del dispositivo
- Rendimiento
- Empleo de la tecnología en el futuro

Versión	Opciones de acceso a características del dispositivo	Rendimiento	Empleo de la tecnología en el futuro
Apache Cordova	Mayor	Menor	Menor
Ionic Capacitor	Menor	Mayor	Mayor

Tabla 4 - Tecnología de encapsulamiento

En este último apartado, la elección será emplear Capacitor. El único inconveniente de Capacitor frente a Cordova es que será necesario emplear alguno de los plugins de Cordova dentro del propio Capacitor para acceder a alguna de las características del dispositivo no cubiertas por alguna API de Capacitor. Por lo demás, Capacitor ha sido desarrollado por el propio Ionic, lo cual siempre es positivo cuando se trabaja en un proyecto Ionic. Además, cabe destacar que el rendimiento y la fluidez de las aplicaciones en Capacitor son superiores a Cordova y el inconveniente más arriba mencionado de las aplicaciones Apple se vería solventado.

Análisis de riesgos

En este apartado se van a analizar los riesgos presentes en el proyecto. Estos se pueden dividir en riesgos por causas externas al proyecto y riesgos por causas internas dentro del proyecto.

Riesgos internos

- Inexperiencia en el uso de las tecnologías empleadas para desarrollar la aplicación. **(1)**
- El alumno no tiene la cualificación necesaria para poder sacar el proyecto adelante. **(2)**
- Mala planificación en el desarrollo del proyecto. **(3)**

Riesgos externos

- Baja aceptación del producto creado a la hora de comercializarlo. **(4)**
- Competencia con otras aplicaciones similares ya existentes. **(5)**
- Cambios de la propia tecnología empleada. **(6)**

Ponderaciones y representación

Con el fin de analizar el grado de probabilidad y el impacto de cada uno de los riesgos mencionados anteriormente, se le asignará a cada riesgo una probabilidad de que suceda y el impacto que tendría en el proyecto (valores entre 0 y 1, siendo 0 ningún impacto y 1 un impacto total).

Número asociado	1	2	3	4	5	6
Probabilidad	0,9	0,8	0,6	0,6	0,4	0,2
Impacto	1	0,8	0,8	0,9	0,8	0,5

Tabla 5 - Probabilidades e imapctos

PROBABILIDAD \ IMPACTO	0,2	0,4	0,5	0,6	0,7	0,8	0,9	1
0,2	0,04	0,08	**(6) 0,1**	0,12	0,14	0,16	0,18	0,2
0,4	0,08	0,16	0,2	0,24	0,28	**(5) 0,32**	0,36	0,4
0,5	0,1	0,2	0,25	0,3	0,35	0,4	0,45	0,5
0,6	0,12	0,24	0,3	0,36	0,42	**(3) 0,48**	**(4) 0,54**	0,6
0,7	0,14	0,28	0,35	0,42	0,49	0,56	0,63	0,7
0,8	0,16	0,32	0,4	0,48	0,56	**(2) 0,64**	0,72	0,8
0,9	0,18	0,36	0,45	0,54	0,63	0,72	0,81	**(1) 0,9**
1	0,2	0,4	0,5	0,6	0,7	0,8	0,9	1

Tabla 6 - Matriz probabilidad/impacto

Plan de contingencia

1. El proyecto va a demandar el uso de unas tecnologías en concreto. En caso de que el alumno no tenga experiencia trabajando con ellas, se llevará a cabo un proceso de formación exhaustivo que permita que el alumno adquiera las aptitudes necesarias con el uso de las tecnologías empleadas.
2. En caso de que el alumno no tenga la cualificación necesaria para llevar a cabo el proyecto, se le inscribirá en un curso especializado en este tipo de proyectos en la conocida plataforma Udemy en el curso "Build iOS, Android & Web Apps with Ionic & Angular". Al finalizar el curso, el alumno obtendrá un documento que acredite que lo ha finalizado.
3. El plan de trabajo a seguir durante todo el proyecto es una parte esencial en el desarrollo del proyecto, ya que define todo el desarrollo de la aplicación. Para evitar que este plan de trabajo no se lleve a cabo y evitar alargar el proyecto retrasando la fecha de entrega preestablecida, se organizara el trabajo en días y paquetes de trabajo que hay que ir cumpliendo como objetivos a lo largo del paso del tiempo en el proyecto. Para ello, utilizaremos herramientas que nos ayuden a realizar una correcta planificación, como un diagrama de Gantt.
4. Es posible que el producto final que se obtenga de este proyecto tenga una baja aceptación en el mercado y no despierte un gran interés en los posibles clientes. Para evitar esto, previamente el alumno preguntará a varias personas por la calle de distintas edades acerca de si usarían la aplicación.
5. A la hora de sacar la aplicación al mercado, existirá competencia de otras aplicaciones que ofrezcan un contenido parecido. Es por ello que es imprescindible ofrecer un contenido lo más original y auténtico posible.
6. Las tecnologías empleadas en este proyecto están en constante desarrollo. Se actualizan constantemente con el objetivo de mejorar sus prestaciones. Con estos cambios, es posible que el alumno no sepa manejarse con las nuevas versiones, por lo que será imprescindible de cara a futuras actualizaciones formarse mediante plataformas como Udemy.com, o la propia documentación oficial de dichas tecnologías.

Descripción de la solución propuesta. Diseño (básico o de alto nivel)

Tecnologías utilizadas

Ilustración 1- Tecnologías utilizadas

Durante el desarrollo de la aplicación se han empleado las tecnologías señaladas en la ilustración superior, es decir, Ionic v4, AngularJS, TypeScript, HTML 5, Ionic Capacitor y CSS 3.

HTML 5 se ha empleado para desarrollar la vista que tiene el usuario de la aplicación. Además, se ha empleado CSS 3 para proporcionar estilo a los archivos .html que componen la vista. Por otro lado, se han empleado tags html propios de Ionic v4 a la hora de confeccionar las distintas páginas que ve el usuario.

Además, AngularJS e Ionic v4 se han utilizado conjuntamente para características como el routing (navegar de una página a otra dentro de la propia aplicación) o el paso de información de un componente o página a otro. Pese a que sea un proyecto Ionic, por debajo es como si fuera un proyecto Angular. Es por esto que se definen en Angular todos los tags HTML empleados por Ionic, así como acceso a librerías del mismo, para luego poder usarlos en el resto del proyecto.

Asimismo, cabe señalar que toda la lógica de la aplicación está desarrollada en código TypeScript. Se usa en los componentes, los servicios y los archivos de routing.

Por último, se hace uso de Ionic Capacitor para poder convertir la aplicación (que sin Capacitor es una aplicación web) en una aplicación nativa del dispositivo. Además, se puede acceder a periféricos necesarios para la aplicación como el altavoz, la cámara o almacenamiento interno del dispositivo, así como a restringir la orientación de la distribución en pantalla del mismo.

Diseño de la solución

A continuación, se procede a indicar la solución escogida para desarrollar esta aplicación. Se indicarán las funcionalidades de cada elemento e indicaciones de cómo lleva a cabo dichas funcionalidades. El código perteneciente a cada elemento está en el Anexo II.

Diseño de alto nivel

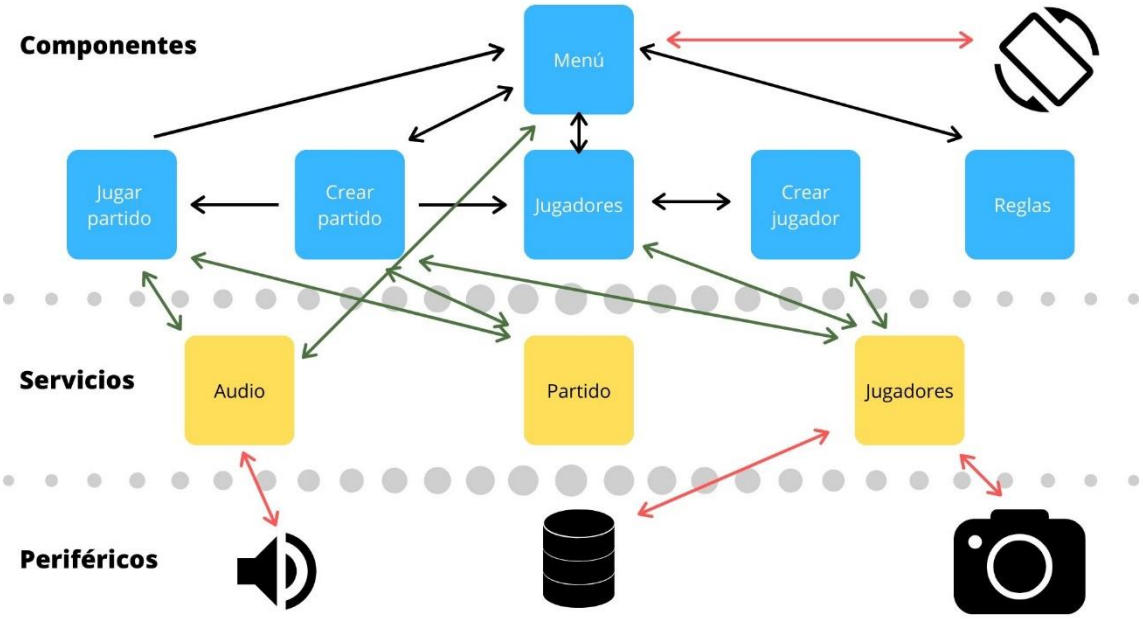

Ilustración 2- Diseño de alto nivel

Como se puede apreciar en la imagen, se pueden diferenciar tres bloques principales.

Por un lado, están los componentes y páginas (bloques de azul). Cada uno de estos corresponde a una vista que puede tener el usuario de la aplicación. De esta manera, habrá una vista que muestre el menú, otra para mostrar la lista de jugadores, otra para añadir un nuevo jugador, otra para definir los parámetros previos a un partido, otra para el propio juego y otra para ver las reglas. Este es el bloque con el que el usuario puede interactuar.

Por otro lado, se encuentran los servicios (bloques de amarillo). Los servicios pueden incluir la lógica de negocio de la aplicación, así como comunicar varios componentes entre sí y permitir la transferencia de datos entre ellos. Los servicios de la aplicación son: uno para gestionar la lista de jugadores registrados en la misma, otro para gestionar lo relacionado con un partido y otro para gestionar el periférico del altavoz del dispositivo.

Asimismo, están los periféricos (objetos de negro). El acceso a dichos periféricos se realiza mediante plugins y APIs que se detallarán más adelante en su apartado correspondiente. Dichos plugins y APIs permiten acceso a dichos periféricos del dispositivo móvil. En este proyecto se hace uso de plugins propios de Cordova para acceder al altavoz y controlar la orientación del dispositivo y APIs de Capacitor para acceder a la cámara de fotos y su galería, así como para almacenar de manera local la lista de jugadores.

Por último, se pueden apreciar interacciones entre los distintos elementos (las flechas). Las flechas negras representan rutas de navegación realizables por el usuario entre los distintos componentes dentro la aplicación. Las flechas verdes representan llamadas a los servicios de los componentes de la aplicación. Las flechas rojas representan llamadas a las APIs y plugins del periférico involucrado.

Resumiendo el funcionamiento de la aplicación, el usuario nada más entrar accederá a un menú desde el que podrá decidir entre ver las reglas del juego, gestionar la lista de jugadores y jugar un partido. En dicho menú se accederá directamente a la orientación del teléfono, fijándola en vertical. Además, también se accederá al servicio de "Audio", para que el usuario pueda decidir mediante un interruptor si activa el sonido de la aplicación o no.

Si decide ver la lista de jugadores, será capaz de eliminar cualquier jugador o añadir uno nuevo. Para ello se hará uso del servicio "Jugadores", tanto para mostrar la lista en un primer momento como para modificarla. Dicho servicio trabaja con la cámara de fotos y la base de datos del dispositivo.

Si decide ir a jugar un partido, primero introducirá unos datos previos a la configuración del partido y entonces pasará a la pantalla del partido. Para ello, a la hora de seleccionar los jugadores del partido se hará uso del servicio "Jugadores". Para cargar parámetros propios del partido y pasárselos a jugar-partido, se hará uso del servicio "Partido". Para reproducir sonidos dentro del propio partido, se hará uso del servicio "Audio". Este servicio trabaja con el altavoz del dispositivo.

Diseño de bajo nivel

Antes de analizar el diseño de bajo nivel empleado en cada uno de los componentes, cabe destacar que no se inflige ninguna ley de derechos de autor en el uso de las imágenes, los audios o los vídeos empleados en esta aplicación.

Routing

En primer lugar, y antes de analizar los demás elementos de la aplicación en detalle, es imprescindible establecer las rutas de los distintos componentes entre los que el usuario va a navegar.

Para ello, empleando las herramientas de routing de Angular Se definirá que la página principal que salga al iniciar la aplicación sea el menú. Además, se deberá establecer que en caso de introducirse una dirección de una página de manera incorrecta el usuario sea redireccionado al menú.

Acceso a periféricos

Como en el proyecto se va a emplear Ionic Capacitor, se dispone de la ventaja de poder emplear tanto plugins desarrollados por la comunidad de Apache Cordova como APIs de Ionic Capacitor para acceder a los periféricos del dispositivo.

Orientación de la pantalla del dispositivo

Para limitar la orientación de la pantalla del dispositivo a vertical se va a hacer uso del plugin de Apache Cordova "Screen Orientation".

Este va a ser el único plugin que va a ser accedido directamente desde un componente sin pasar por un servicio intermedio. Esto es debido a limitar al mínimo el retardo de establecer como orientación vertical y así evitar una experiencia peor por parte del usuario. Además, al ser el menú la primera página que se muestra al iniciar la aplicación, sólo es necesario emplearlo en dicha página para que quede ya esa orientación durante toda la aplicación, mientras que otros periféricos son accedidos por varios componentes distintos, lo que favorece el empleo de un servicio intermedio.

Altavoz

A la hora de hacer uso del altavoz y reproducir sonido, se va a emplear el plugin desarrollado por la comunidad de Apache Cordova "Native Audio".

Este plugin será accedido desde un servicio denominado "Audio", ya que es empleado tanto en el componente del menú como en el de jugar-partido. El funcionamiento de este plugin, básicamente, consiste en cargar un sonido primero, para después reproducirlo.

Almacenamiento interno

Con el objetivo de guardar los usuarios registrados en la aplicación de manera local, se empleará la API de Ionic Capacitor "Storage".

Es una API que no está pensada para aplicaciones que requieran una gran gestión de datos, pero en el caso de este proyecto, en el que su único propósito es guardar una lista de usuarios, es una mejor opción que alternativas como SQLite debido a su ligereza.

Como se ha mencionado, esta API será utilizada para guardar el nombre y la ruta a la fotografía asignada a cada jugador de la lista de jugadores de la aplicación.

Cámara de fotos y galería del dispositivo

Por último, para poder acceder tanto a la cámara del dispositivo como a su galería de imágenes, se hará uso de la API de Ionic Capacitor "Camera".

Esta API requiere permisos de acceso a la cámara y a la galería que el usuario tendrá que aceptar para poder cumplir su función. Será utilizada cuando el usuario añada un nuevo jugador a su aplicación. Para ello, podrá escoger entre no asignarle ninguna fotografía a ese usuario (se le asignará una por defecto), buscar en su galería una imagen adecuada o sacar una foto en ese mismo momento. Para estas dos últimas opciones, es necesario el uso de esta API.

Servicios

Un servicio consiste en un único archivo TypeScript que se encarga de hacer de interfaz entre componentes u otras entidades, así como proporcionar una lógica de negocio. En este caso, existen 3 servicios en la aplicación: jugadores, partido y audio.

Jugadores

El servicio de jugadores es el encargado de proporcionar a los componentes tanto la lista de jugadores actualizada, como la posibilidad de modificarla.

Para ello, hace uso del plugin de la API de almacenamiento interno mencionada previamente y va actualizando un array de Jugadores interno dentro del propio servicio con el contenido del almacenamiento local del dispositivo. Este array es el que le pasa a los componentes cuando la lista de jugadores es requerida.

Por otro lado, también hace uso de la API de la cámara, añadiendo un método para sacar una foto o añadirla de la galería del usuario.

Además, al agregar un nuevo jugador, el componente correspondiente le pasa los datos como nombre y foto, y este servicio se encarga de añadirle una id aleatoria y única.

Partido

El servicio del partido es el encargado de inicializar y pasar al componente del partido la baraja de cartas con todas las cartas que se van a emplear en el juego.

Además, hace de nexo de unión entre el componente crear-partido y jugar-partido, pasándole datos a jugar-partido previamente recogidos en crear-partido. Estos datos son el nombre de los equipos, la dificultad seleccionada, el tipo de partido y el array de jugadores de cada uno de los equipos.

Audio

El servicio de audio, como su propio nombre indica, se encarga de controlar el sonido que emite la aplicación.

Por un lado, contiene un boolean que se va a actualizando con el componente del menú en función de si el usuario activa o desactiva el sonido.

Por otro lado, llama al plugin de sonido mencionado anteriormente. Básicamente lo que hace es establecer un array de audios y cuando se le indica desde el componente de jugar-partido, reproduce el audio del array que se indique.

Componentes

Los componentes están formados por un archivo html (lo que se le muestra al usuario), un archivo scss (lo que da estilo al html), un archivo TypeScript (la lógica del componente). Además, los componentes que son tipo páginas (page), llevan también un archivo module.ts (en el que se insertan librerías y se configuran funciones de routing).

Menú

El componente del menú es el primero al que accede el usuario al arrancar la aplicación. En él puede ver las distintas páginas a las que puede navegar, así como activar o desactivar la opción del sonido en la aplicación.

Para ello, en cuanto se crea el componente del menú, se inicializa un objeto del servicio de audio y se le indica si el sonido está activo o desactivado. Además, también se llama al plugin de la orientación de la pantalla mencionado anteriormente y se bloquea en orientación vertical.

A continuación, se muestra la visión del usuario de este componente:

Ilustración 3 - Menú

Jugadores

El componente de jugadores es el encargado de mostrar al usuario la lista de jugadores registrados en la aplicación y permitirle modificarla si lo desea. Para ello, se llama al servicio de jugadores y se obtiene la lista registrada en el dispositivo para mostrarla por pantalla.

Además, mediante un icono en pantalla que el usuario puede pulsar, se da la posibilidad de eliminar un jugador. Para eliminarlo, se mostrará primero una alarma al usuario para preguntarle si está seguro y después se borrará de la lista y del almacenamiento interno (por medio del servicio).

Por último, también se da la opción de añadir un nuevo jugador. Esto provocará que se abra un módulo (similar a un componente), en el que podrá llevarse a cabo esta gestión.

A continuación, se muestra la visión del usuario de este componente:

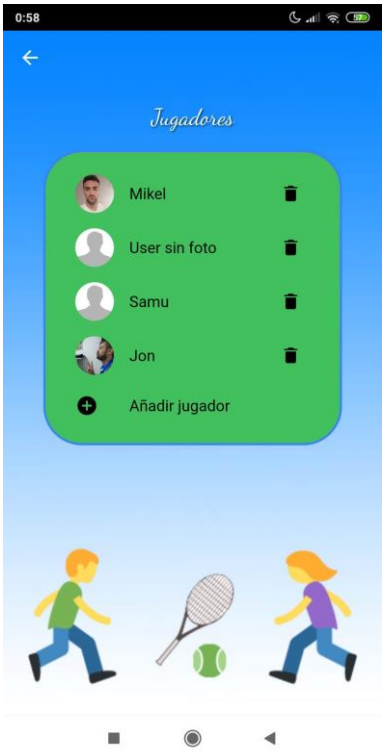

Ilustración 4 - Jugadores

Crear jugador

Este módulo consiste básicamente en un formulario en el que el usuario introduce los datos necesarios para un nuevo jugador. El módulo llama al servicio de jugadores para añadir dicho jugador a la base de datos.

Antes de poder enviar nada, se valida que los datos sean correctos. En caso de que el usuario no quiera asignarle ninguna foto al usuario, se le asignará una foto genérica.

A continuación, se muestra la visión del usuario de este módulo:

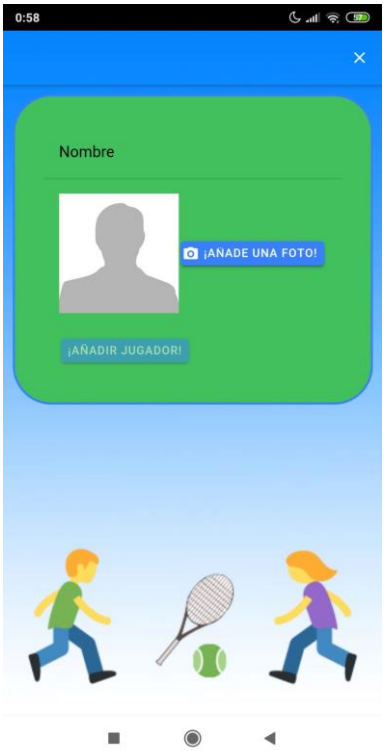

Ilustración 5 - Crear jugador

Crear partido

En este componente se introducirán los parámetros necesarios para iniciar un partido. Consistirá básicamente en dos formularios, primero se renderizará uno y cuando esté acabado, se renderizará el otro.

En el primer formulario se le pedirá al usuario el número de jugadores del partido, el nombre de los equipos y los jugadores que van a pertenecer a cada equipo. Para mostrar los jugadores disponibles, se hará uso del servicio de jugadores. En caso de seleccionar un número de jugadores mayor al registrado en la aplicación, se le indicará al usuario, ofreciéndole un link que le lleve al componente de jugadores. Se asegurará que un mismo jugador no pueda estar en ambos equipos, así como que la longitud del nombre escogido por el usuario para cada equipo no supere los 12 caracteres.

Una vez completado el primer formulario, se mostrará el segundo, en el que seleccionará la dificultad y la duración del partido. El nivel de dificultad actuará como un multiplicador a la hora de asignar castigos durante el partido. Una vez hecho esto, se le pasarán los datos del partido al servicio de partido y se redirigirá al usuario al componente de jugar partido.

A continuación, se muestra la visión del usuario de este componente:

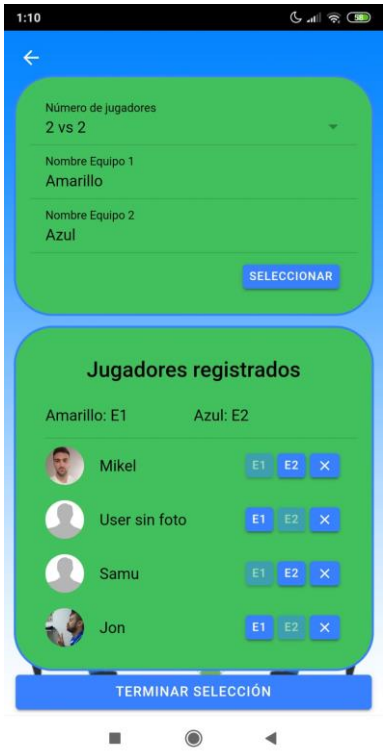

Ilustración 6 - Crear Partido. Formulario 1

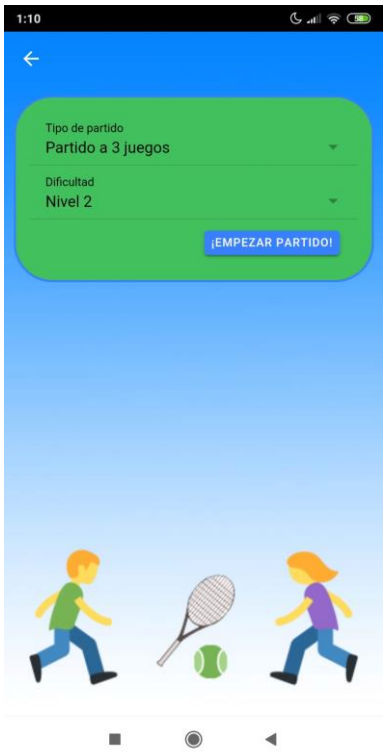

Ilustración 7 - Crear partido. Formulario 2

Jugar partido

Este es el componente en el que más tiempo va a pasar el usuario y que contiene toda la lógica del juego propiamente dicho. Este componente llama al servicio partido para recibir los parámetros definidos en el componente crear-partido e inicializar el juego. Además, también inicializa el servicio de audio y carga tres sonidos en él. Un sonido para cada carta, otro para cada punto y otro para cada juego.

La distribución en pantalla permite ver para los dos equipos (uno a cada lado de la pantalla) la carta sacada, la carta que supone un triunfo, los miembros de cada equipo, la suma de cada punto de cada equipo y un marcador, así como la opción de salir del juego.

En el TypeScript, se define toda la lógica del juego, la suma de las puntuaciones, la determinación del ganado, etc. Se pueden consultar las normas del juego en el anexo I. Además, las penalizaciones por perder se muestran a modo de alarmas para el usuario, indicando el tipo de ejercicio a realizar (de forma aleatoria), así como un link de un vídeo explicativo para realizar la técnica correcta de dicho ejercicio y evitar lesiones.

A continuación, se muestra la visión del usuario de este componente:

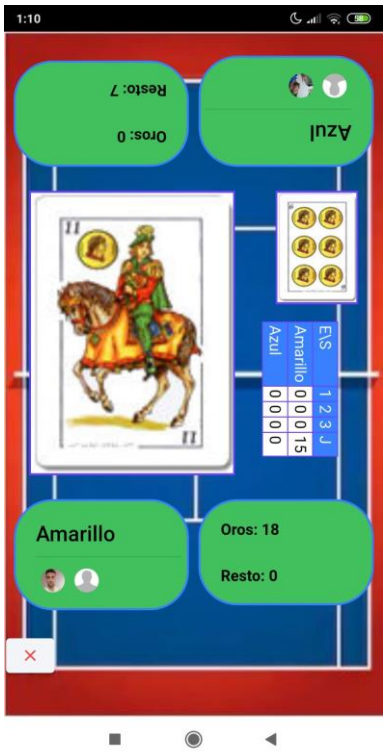

Ilustración 8 – Partido

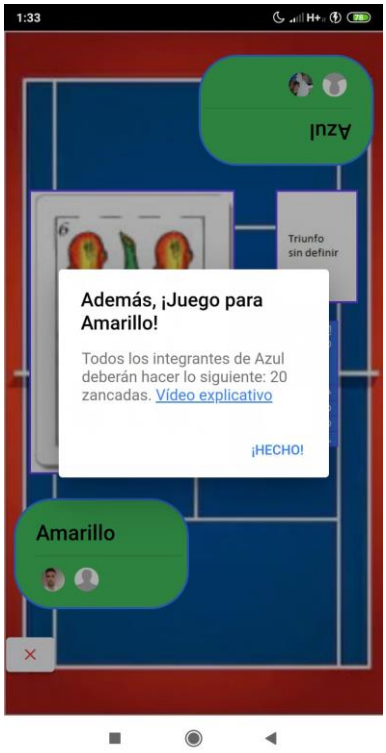

Ilustración 9 - Partido. Alarma final de juego

Reglas

Por último, este componente indica las normas del juego al usuario para que las pueda consultar en cualquier momento. No incluye ninguna llamada a ningún servicio.

A continuación, se muestra la visión del usuario de este componente:

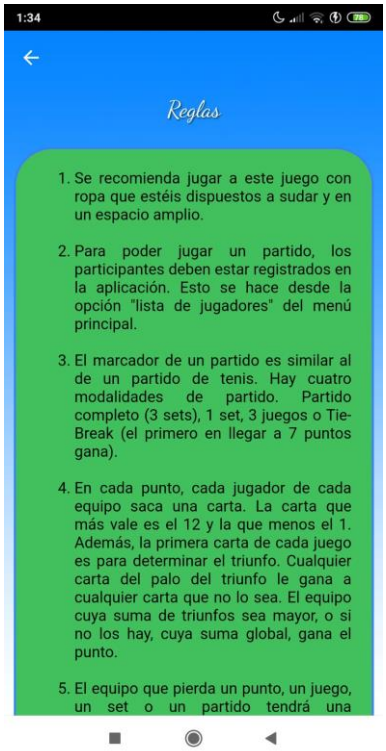

Ilustración 10 - Reglas

Metodología seguida en el desarrollo del trabajo

Descripción de tareas y fases

El conjunto de tareas a realizar durante el desarrollo del trabajo se ha separado en 7 paquetes de trabajo muy diferenciados:

1. Análisis de ingeniería. Este es el primer paquete de trabajo a realizar. En él se definen las tecnologías que se van a emplear en el desarrollo de la aplicación en base a las especificaciones previamente definidas. Esto se realiza mediante un análisis de alternativas.
2. Formación. Una vez seleccionadas las tecnologías a emplear, el alumno debe adquirir conocimientos suficientes en cada una de ellas que le permitan desarrollar una aplicación que cumpla con las especificaciones definidas.
3. Diseño. Antes de empezar a escribir el código de la aplicación, es imprescindible que el alumno defina claramente cuál va a ser la estructura de los diferentes componentes de la aplicación y cómo van a interactuar entre ellos.
4. Desarrollo y pruebas unitarias. Una vez definido el diseño, el siguiente paso es desarrollar el código de cada uno de los componentes de la aplicación. Paralelamente, según se vaya terminando el código de cada componente, se realizarán pruebas a cada uno de ellos para minimizar potenciales problemas que puedan dar.
5. Integración y pruebas de convergencia. Después de tener los componentes de la aplicación terminados y probados de manera unitaria, es necesario asegurar su capacidad para pasarse datos e interactuar entre ellos de manera correcta. Para ello, se realizarán pruebas orientadas a este tipo de interacciones también para minimizar potenciales errores que puedan surgir.
6. Validación. Tras asegurar la correcta integración de los componentes de la aplicación, es el momento de comparar las especificaciones iniciales y el resultado obtenido. De esta comparación se extraerán las conclusiones del proyecto.
7. Documentación. Por otro lado, durante la realización del proyecto es necesario ir elaborando el presente documento en el que están reflejados los aspectos más importantes del mismo.

Diagrama de Gantt

En el siguiente diagrama de Gantt se observa la distribución temporal seguida para la realización de los distintos paquetes de trabajo mencionados en el apartado anterior. Se estima una media de 7 horas diarias de dedicación al proyecto durante 49 días.

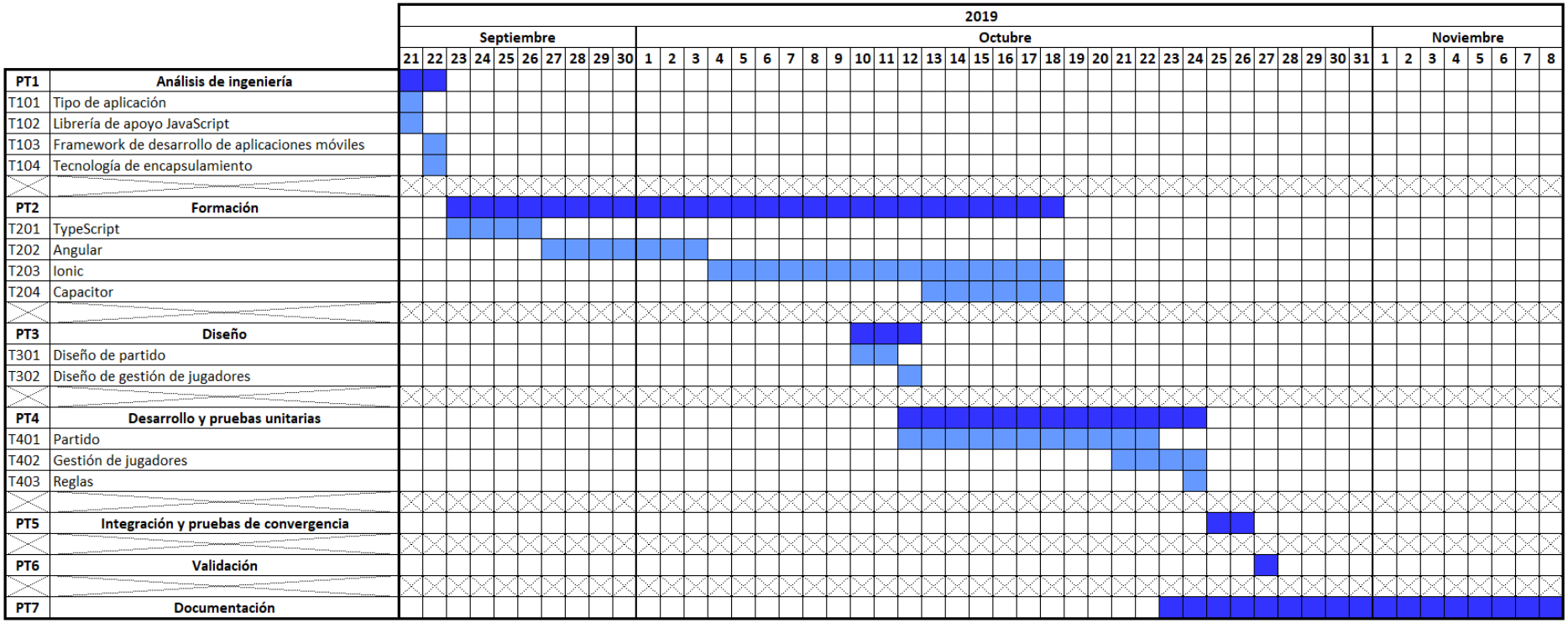

Ilustración 11- Diagrama de Gantt

Descripción de los resultados

Para poder analizar el resultado del proyecto, primero se deberá someter a la aplicación a una serie de pruebas que otorguen validez a cada uno de los elementos que la componen.

Casos de prueba

Prueba de gestión de información

El objetivo de esta prueba es descubrir errores a la hora de consultar o modificar la información guardada en el dispositivo. Para ello se comprueba que:

1. Se obtiene la lista de jugadores correctamente y de forma actualizada.
2. Se elimina a un jugador del almacenamiento interno correctamente.
3. Se añade a un jugador al almacenamiento interno correctamente.

Esta prueba se completó satisfactoriamente sin errores.

Prueba de periféricos

El objetivo de esta prueba es asegurar el correcto funcionamiento de los distintos métodos empleados para usar los periféricos del dispositivo. Para ello se comprueba que:

1. Lo sonidos son cargados y reproducidos correctamente.
2. Se puede acceder a la funcionalidad de la cámara del dispositivo.
3. Se puede acceder a la galería local del dispositivo.

Esta prueba se completó satisfactoriamente sin errores.

Prueba de características de la interfaz

Esta prueba consiste en descubrir errores en las características de la propia interfaz del usuario comprobando:

1. Que la fuente sea adecuada.
2. Que el color de la fuente sea legible, combinado con el fondo.
3. Que las imágenes se vean correctamente.

Esta prueba se saldó sin errores.

Prueba de usabilidad

Esta prueba consiste en comprobar que la aplicación muestra la vista de usuario de una forma fluida y sin errores de contenido. Para ello, se comprueba:

1. La navegación entre las páginas de la aplicación es intuitiva, fluida y sin errores.
2. El correcto funcionamiento de los formularios al introducir datos en él.
3. Los datos que se muestran en pantalla están actualizados con el almacenamiento interno del dispositivo.
4. Comprobar que el sistema de alarmas dentro del juego es claro para el usuario.

Esta prueba mostró que la primera vez que se iniciaba la aplicación en un dispositivo, los primeros segundos tardaba en navegar de una manera fluida.

Prueba de errores semánticos

El fin de esta prueba es el de descubrir errores semánticos que puedan llevar a la confusión al usuario. Para ello se comprueba que:

1. La información mostrada en pantalla sea clara y concisa.
2. La información esté bien organizada.
3. Textos más elaborados como las reglas del juego no den pie a confusiones y tengan una fácil comprensión.

Esta prueba se completó satisfactoriamente sin errores.

Prueba de integración del sistema

El fin de esta prueba es el de asegurarse que los distintos elementos del proyecto se comunican correctamente entre sí. Para ello se comprueba que:

1. El envío de datos entre servicios y componentes funciona adecuadamente.
2. Los métodos de los servicios llamados desde los componentes son ejecutados correctamente.

Esta prueba se completó de una forma satisfactoria y sin errores.

Prueba alfa

Una vez realizadas todas las pruebas anteriores, se lleva a cabo la prueba alfa. Esta consiste en entregar la aplicación a usuarios reales y probarlas en un escenario real con el desarrollador controlando las acciones de los usuarios y registrando errores y problemas de uso. Esta prueba se realizó con 5 personas y el desarrollador y la aplicación respondió correctamente y sin incidencias.

Resultados finales

Como se puede observar en las pruebas, el único inconveniente son los primeros segundos de la primera vez que se usa la aplicación, en la que ésta funciona a una velocidad algo menor. A partir de ahí, ya funciona de una manera fluida y se cumplen todos los requisitos y especificaciones definidas previamente en todos sus usos en el dispositivo.

Presupuesto

A continuación, se desglosan los distintos factores a tener en cuenta para realizar un presupuesto para este proyecto:

- El equipo de desarrollo lo forma una única persona (el alumno), que trabaja una media de 7 horas diarias valoradas a 10€/h. Esto supone un total de 7 horas durante 49 días, es decir, 343 horas en total.
- Para la fase de formación, se emplea la documentación oficial de las distintas tecnologías empleadas en el proyecto (gratuita), así como el curso de Udemy.com "Build iOS, Android & Web Apps with Ionic & Angular", (valorado en 179.99€).
- Al ser una única persona y ser trabajo a realizar íntegramente en un ordenador, no será necesario que el alumno alquile ningún local en el que trabajar, podrá hacerlo desde su propio domicilio.
- Como es una aplicación multiplataforma, el alumno necesita un ordenador "MacBook" para poder desarrollar la aplicación tanto para Android como para iOS (requisito indispensable para esta última). El precio de un MacBook Pro adecuado para el desarrollo de este proyecto está valorado en 1255,55€.
- Todas las imágenes, audios, vídeos, iconos y demás elementos externos empleados en la aplicación no tienen licencias que restrinjan su uso, por lo que no es necesario abonar nada por emplearlos.
- Con el fin de ofrecer la aplicación en ambas plataformas (iOS y Android), será necesario que el alumno obtenga las correspondientes cuentas de desarrollador tanto en la Google Play Store (25$) como en la Apple Store (99$ al año). En el caso de la licencia de Apple, se contratará por un año y en caso de que el producto funcione, se procederá a renovarse por otro año.

Con estos factores, el presupuesto para la realización de esta aplicación será el siguiente:

Concepto	Unidades	Coste unitario	Nº Unidades	Coste total
Ingeniería	h	10,00 €	343	3.430,00 €
Formación	cursos	179,99 €	1	179,99 €
MacBook Pro	ordenadores	1.255,55 €	1	1.255,55 €
Cuenta desarrollador Android	licencia	25,00 €	1	25,00 €
Cuenta desarrollador Apple	licencia	99,00 €	1	99,00 €
			Total	4.989,54 €

Tabla 7 - Presupuesto

Respecto a los ingresos, la aplicación se ofrecerá tanto en la Google Play Store como en la Apple Store de manera gratuita para llegar a la mayor cantidad de público posible. Los ingresos vendrán generados por la inclusión de publicidad de terceros.

Conclusiones

Este proyecto consigue crear una aplicación móvil intuitiva y divertida para que los usuarios puedan hacer deporte en un entorno casual sin necesitar ningún tipo de material. Además, incluye varios niveles de dificultad, lo que la hace apta tanto para personas que hagan ejercicio habitualmente con un nivel físico alto como para personas que no hagan deporte de manera habitual, pero les guste hacerlo de vez en cuando.

Es cierto que ya encontramos algunas aplicaciones pensadas para el fitness y el deporte en general. Sin embargo, la gran mayoría están orientadas a llevar un control de la alimentación, técnica correcta de realización de ejercicios, rutinas, cronómetros, indicadores de rutas, etc. Esta aplicación supone una alternativa totalmente distinta que combina toda la diversión y el ocio que proporciona un juego con otras personas con los beneficios para la salud que otorga el ejercicio físico. Además, se incluyen vídeos explicativos para el usuario con el fin de prevenir cualquier tipo de lesión. Es por todo esto que se prevé que la aplicación tenga una buena acogida.

Bibliografía

- desarrolloweb.com "AngularJS vs jQuery". Disponible en: https://desarrolloweb.com/articulos/angularjs-vs-jquery.html
- programacion.net "AngularJS vs jQuery, ¿cuáles son sus mayores diferencias?". Disponible en: https://programacion.net/articulo/angularjs_vs_jquery-_cuales_son_sus_mayores_diferencias_1610
- innovaportal.com "Apps Híbridas vs Nativas vs Generadas. ¿Qué decisión tomar?". Disponible en: https://www.innovaportal.com/innovaportal/v/696/1/innova.front/apps-hibridas-vs-nativas-vs-generadas-que-decision-tomar
- medium.com "Ionic 4 vs Ionic 3 – Todo lo que necesitas saber sobre Ionic 4". Disponible en: https://medium.com/learn-ionic-framework/ionic-4-vs-ionic-3-todo-lo-que-necesitas-saber-sobre-ionic-4-5235927c6dd9
- medium.com "El sueño de crear una aplicación iOS, Android, Electron y Web (PWA) desde una sola base de código". Disponible en: https://medium.com/williambastidasblog/el-sue%C3%B1o-de-crear-una-aplicaci%C3%B3n-ios-android-electron-y-web-pwa-desde-una-sola-base-de-c%C3%B3digo-ab66ad903919
- codeway.me "¿Cuál es la diferencia entre Apache Cordova e Ionic?". Disponible en: https://codeday.me/es/qa/20190315/311965.html
- wikipedia.org "Apache Cordova" Disponible en: https://es.wikipedia.org/wiki/Apache_Cordova.
- udemy.com "Build iOS, Android & Web Apps with Ionic & Angular" Disponible en: https://www.udemy.com/course/ionic-2-the-practical-guide-to-building-ios-android-apps/
- palco24.com "El negocio de los gimnasios en España crece un 2.5% en 2018, hasta 2291 millones de euros". Disponible en: https://www.palco23.com/fitness/el-negocio-de-los-gimnasios-en-espana-crece-un-25-en-2018-hasta-2291-millones-de-euros.html
- expansion.com "Las redes sociales en cifras: ha llegado la madurez". Disponible en: https://www.expansion.com/economia-digital/innovacion/2019/07/15/5d276365e5fdeab7578b46f2.html
- who.int "Obesidad y sobrepeso". Disponible en: https://www.who.int/es/news-room/fact-sheets/detail/obesity-and-overweight
- webconsultas.com "Beneficios del ejercicio físico". Disponible en: https://www.webconsultas.com/ejercicio-y-deporte/vida-activa/beneficios-del-ejercicio-fisico-869
- elperiodico.com "8 beneficios psicológicos de hacer deporte". Disponible en: https://www.elperiodico.com/es/ser-feliz/20190101/beneficios-psicologicos-hacer-deporte-7217004
- ionicframework.com "Ionic Documentation". Disponible en: https://ionicframework.com/docs
- capacitor.ionicframework.com "Capacitor Documentation". Disponible en: https://capacitor.ionicframework.com/docs/
- angular.io "Angular Documentation". Disponible en: https://angular.io/docs

- typescriptiang.org “TypeScript Documentation”. Disponible en: https://www.typescriptlang.org/docs/home.html
- w3schools.com “CSS Tutorial”. Disponible en: https://www.w3schools.com/css/
- amazon.com “Búsqueda de MacBook Pro”. Disponible en: https://www.amazon.es/macbook-pro-Port%C3%A1tiles-Inform%C3%A1tica/s?k=macbook+pro&i=computers&rh=n%3A938008031%2Cp_n_feature_browse-bin%3A1482669031&dc&qid=1573088143&rnid=8178865031&ref=sr_nr_p_n_feature_fifteen_browse-bin_2
- youtube.com “Cómo hacer abdominales. Plancha abdominal perfecta”. Disponible en: https://www.youtube.com/watch?v=5JMakqxdT5c
- youtube.com “Mountain Climbers para quemar grasa”. Disponible en: https://www.youtube.com/watch?v=IfNxtI5oMVM
- youtube.com “Flexiones – Desarrollar pectorales, hombros, brazos”. Disponible en: https://www.youtube.com/watch?v=VxSbYM5X6xk&t=
- youtube.com “Cómo hacer sentadillas correctamente – Ejercicio de glúteos y piernas”. Disponible en: https://www.youtube.com/watch?v=YOPq7v0mcuI
- youtube.com “Zancadas traseras | 8 claves para una técnica perfecta”. Disponible en: https://www.youtube.com/watch?v=yOQKob8VDBI
- youtube.com “Explicación paso a paso de cómo hacer un burpee para principiantes”. Disponible en: https://www.youtube.com/watch?v=A933f34Wj3Y
- youtube.com “Ejercicios de piernas: Jumping Jacks”. Disponible en: https://www.youtube.com/watch?v=95j1mH27eXc
- ionicframework.com “Native Audio”. Disponible en: https://ionicframework.com/docs/native/native-audio
- ionicframework.com “Ionic Native – Screen Orientation”. Disponible en: https://ionicframework.com/docs/v3/native/screen-orientation/
- capacitor.ionicframework.com “Storage - Capacitor”. Disponible en: https://capacitor.ionicframework.com/docs/apis/storage
- capacitor.ionicframework.com “Camera - Capacitor”. Disponible en: https://capacitor.ionicframework.com/docs/apis/camera
- canva.com Herramienta de diseño empleada en el proyecto. Disponible en: https://www.canva.com/
- freesound.org Audios empleados en el proyecto. Disponibles en:
    - https://freesound.org/people/Rani%20Shoket/sounds/63246/
    - https://freesound.org/people/jayfrosting/sounds/333404/
    - https://freesound.org/people/singintime/sounds/170629/

ANEXO I: Manual de usuario y reglas del juego

Manual de usuario

Al arrancar la aplicación, esta se inicializará en el menú principal de la aplicación. Aquí el usuario podrá ir a la pantalla de jugar un partido, a la pantalla de ver la lista de jugadores y a la pantalla donde consultar las reglas del juego. Además, podrá activar o desactivar el sonido de la aplicación.

En la lista de jugadores, se muestran los usuarios registrados en la aplicación. El usuario podrá eliminar cualquiera de ellos pulsando en el icono que simula una papelera. Además, podrá añadir un nuevo jugador. Al pulsar en añadir un nuevo jugador, se desplegará una pantalla en la que el usuario introducirá el nombre del nuevo jugador y tendrá la opción de asociarle una foto (tanto sacar una foto en el momento como usar una de la galería).

Si el usuario selecciona jugar partido desde el menú principal, primero configurará parámetros para poder iniciar el partido. En primer lugar, seleccionará el número de jugadores y el nombre de ambos equipos y pulsará “seleccionar”. Una vez hecho esto, se mostrará la lista de jugadores registrados de la aplicación con 3 botones a su lado. Un botón será para añadir a ese jugador a un equipo, el otro botón para añadirlo al otro equipo y el último botón para eliminarlos de cualquier equipo en el que estén inscritos. Una vez el usuario haya colocado los jugadores en sus equipos a su gusto, pulsará en “Terminar selección”.

A continuación, se mostrará un formulario en el que escoger el tipo de partido y el nivel de dificultad. Después, se pulsará “Empezar partido” y se pasará a la pantalla del partido.

En dicha pantalla se puede observar los integrantes de ambos equipos, su puntuación acumulada en este punto, la carta que va saliendo mientras se va jugando, el triunfo y el marcador del partido.

Por último, si el usuario desde el menú principal ha seleccionado “reglas”, se le mostrará una pantalla en el que se le detallan las reglas del juego.

Reglas del partido

1. El marcador de un partido es similar al de un partido de tenis. Hay cuatro modalidades de partido. Partido completo (3 sets), 1 set, 3 juegos o Tie-Break (el primero en llegar a 7 puntos gana).
2. En cada punto, cada jugador de cada equipo saca una carta. La carta que más vale es el 12 y la que menos el 1. Además, la primera carta de cada juego es para determinar el triunfo. Cualquier carta del palo del triunfo le gana a cualquier carta que no lo sea. El equipo cuya suma de triunfos sea mayor, o si no los hay, cuya suma global, gana el punto.
3. El equipo que pierda un punto, un juego, un set o un partido tendrá una penalización en forma ejercicio físico.

ANEXO II: Diseño de bajo nivel. Código

Descripción

Routing

Se establece un array de rutas en el que se asocia un nombre a cada página de la aplicación para referenciarlo más tarde. También se establece que en caso de que haya una url desconocida se redirija a Menú.

Servicios

Jugadores

En este servicio se importan los plugin de capacitor para la cámara, la galería y Storage en primer lugar.

Se crean métodos para añadir jugadores, eliminar jugadores y definir la lista de jugadores almacenados en el servicio y un método get para obtener la lista como un observable al que otros componentes puedan suscribirse. En cada uno de estos métodos, se modifica el atributo del servicio que es una lista de jugadores y luego se hace un set de esa misma lista dentro del almacenamiento interno.

También se crea el método para sacar foto y darle la opción al usuario de seleccionar una fotografía de la galería. De aquí se devuelve el path de la imagen seleccionada dentro del dispositivo, sea de la galería o una foto sacada en el momento.

Partido

Este servicio tiene como atributo un array de objetos Carta que constituyen una baraja española con todas las cartas. Además también incluyen objetos Carta de “no quedan cartas” y de “no hay triunfo”, cartas que serán solicitadas por el componente jugar-partido. Para proporcionárselas, se definen métodos get para todas las cartas mencionadas anteriormente.

También se define un método para guardar en atributos locales del servicio una serie de parámetros, que llamará el componente crear-partido. Luego, el componente jugar-partido hará uso de esos atributos.

Audio

En este servicio se importará el plugin de sonido. Además, se definirá un boolean que hará las veces de interruptor para que el sonido se reproduzca o no y un array de sonidos. Se tendrá también un método que sirva para añadir sonidos al array y otro método para reproducir un sonido concreto de ese mismo array.

Componentes

Todos los componentes constan de tres archivos diferentes: un archivo HTML, un archivo TypeScript (TS) y un archivo SCSS.

Menú

Este componente implementa el plugin para restringir la orientación de pantalla. Se utiliza en el lifecycle hook de Angular OnInit, es decir, nada más se crea la instancia de este componente, se ejecuta el código dentro del método ngOnInit. De esta forma, la orientación queda sellada desde el primer momento para toda la aplicación. Además, también se crea una instancia del servicio de audio.

Por otro lado, además de incluir las rutas en el html a otros componentes, se ejecuta el método actualizaSonido para cambiar el boolean del servicio de audio con cada interacción del usuario con el botón de sonido.

Jugadores

En este componente se crea una instancia del servicio Jugadores. Este servicio se emplea para suscribirse a la lista de jugadores de dicho servicio, y así mostrar en pantalla al usuario la lista de jugadores registrados en la aplicación de manera actualizada. Además, se incluye el método de eliminar un jugador (que se ejecuta cuando el usuario pulsa en el icono de la basura), que llama al método de eliminar un jugador del servicio.

Por otro lado, cuando se presione en añadir jugador, se hará uso de un objeto modalController previamente inicializado como atributo para abrir el módulo crear-jugador.

Crear jugador

Este módulo inicializa una instancia del servicio Jugadores nada más empezar. Muestra en pantalla un formulario para que el usuario lo cumplimente y cuando pulsa en el botón de añadir foto, llama al método sacar foto del servicio jugadores.

Una vez el usuario ha rellenado los campos del formulario correctamente (lo comprueba), le permite pulsar el botón de enviar. Para añadir el jugador, se le pasa al método del servicio de Jugadores de añadir jugador el path de la imagen seleccionada por el usuario (se le pasa una genérica si no ha seleccionado ninguna) y el nombre del jugador. Después, mediante la herramienta de routing de Angular, se fuerza la vuelta a la vista del componente Jugadores.

Crear partido

Este componente inicializa una instancia del servicio Partido y otra del servicio Jugadores. Su vista consiste en dos formularios, renderizados por un mismo boolean, de tal manera que primero se realiza un formulario, y al ser validado se muestra el segundo.

En el primer formulario emplea el servicio de jugadores para mostrar la lista de jugadores de la aplicación y que el usuario decida los jugadores para cada equipo. Dichos jugadores se guardarán por equipos en un array propio de este componente. Mediante métodos como estaEquipoContrario y AddJugador, se asegurará que un mismo jugador no esté en dos equipos al mismo tiempo. El botón para completar el primer formulario no será clickable hasta que el boolean CompruebaNumJugadores esté a true, lo que garantiza que en cada equipo hay el número de jugadores que debe haber.

El segundo formulario establece la dificultad y la duración del partido mediante varias opciones desplegables entre las que el usuario seleccionará la más adecuada. Cuando pulse el botón de Comenzar partido, se ejecutará el método empiezaPartido, en el que se hará un set a los atributos del servicio Partido en base a los formularios rellenados por el usuario. Luego, el componente jugar-partido usará estos datos para iniciar el juego.

Jugar partido

Este es el componente principal de la aplicación y el que más código tiene detrás.

Por un lado, se ha usado el tag ion-grid para distribuir todos los elementos de la vista de usuario de la manera deseada, empleando propiedades CSS para girar algunos elementos y que sean visibles tanto por los usuarios de un lado de la pantalla como los del otro.

Respecto al TypeScript, en primer lugar, se crean instancias para los servicios de Audio y Partido. Se recogen del servicio Partido los parámetros recogidos en el componente Crear-partido y se usan para configurar las variables internas del componente para el desarrollo del juego. Además, se obtiene del servicio partido la baraja de cartas al completo.

En lo referente a la lógica del juego, en el momento en el que un usuario toca la imagen de la carta , se ejecuta el método onCarta:

1. En primer lugar se comprueba si es la última carta de la baraja (en ese caso, se reinicia la baraja mediante el servicio partido y se muestra una alerta al usuario).
2. Se saca una carta al azar de la baraja, se muestra en pantalla y se retira del array de cartas (para que no pueda volver a salir hasta que se reinicie la baraja).
3. Se comprueba si esa carta es la primera del juego actual (en caso de serlo, se define como carta de triunfo).
4. Se reproduce un sonido de una raqueta golpeando una pelota.
5. Se suma el valor de la carta a la suma en este punto del equipo.
6. Se comprueba si ya han golpeado todos los miembros de cada equipo en este juego. En caso de que así sea:
    a. Se dejan 200 milisegundos de pausa para que el usuario tenga tiempo de ver la última carta que se ha sacado.
    b. Se calcula el ganador del punto y se le pasa por parámetro al método sumarPunto.
    c. En el método sumarPunto, se le suma el punto al equipo correcto y se comprueba si además ha sido juego, break, set o partido. En función de lo que haya sido, se llama al método castigo. Éste va mostrando mediante una o más alarmas sucesivas las pruebas que tienen que realizar los perdedores del punto.
    d. Se reinician las puntuaciones de cada equipo para el punto siguiente, y si es necesario también el sacador.

Reglas

Este componente está desarrollado únicamente en SCSS y HTML y se indican en texto las normas del juego.

Objetos predefinidos

Para asegurar un funcionamiento eficiente en la aplicación , se ha decidido crear dos clases propias del proyecto como son la clase Jugador y la clase Carta.

Jugador

La clase Jugador es la clase que se emplea para mapear a cada jugador registrado en la aplicación y consta de 3 campos. Una id única propia de cada jugador, un nombre y el path a la foto asignada a ese usuario.

Carta

La clase Carta es la clase empleada para mapear cada carta. Esta clase contiene el palo al que pertenece la carta, el valor numérico de la misma y el path a la imagen asociada a esa carta.

Código

Routing

```
import { NgModule } from '@angular/core';

import { PreloadAllModules, RouterModule, Routes } from '@angular/router';


const routes: Routes = [

  { path: '', redirectTo: 'menu', pathMatch: 'full' },

  { path: 'menu', loadChildren: './menu/menu.module#MenuPageModule' },

  { path: 'reglas', loadChildren: './reglas/reglas.module#ReglasPageModule' },

  { path: 'jugadores', loadChildren: './jugadores/jugadores.module#JugadoresPageModule' },

  { path: 'crear-partido', loadChildren: './partido/crear-partido/crear-
partido.module#CrearPartidoPageModule' },

  { path: 'jugar-partido', loadChildren: './partido/jugar-partido/jugar-
partido.module#JugarPartidoPageModule' },

  ];


@NgModule({

  imports: [

    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })

  ],

  exports: [RouterModule]

})

export class AppRoutingModule { }
```

Servicios

Jugadores

```
import { Injectable, OnInit } from '@angular/core';
import { Jugador } from './jugador.model';
import { BehaviorSubject } from 'rxjs';
import { take } from 'rxjs/operators';

import { Plugins, Capacitor, CameraResultType, CameraSource } from '@capacitor/core';

@Injectable({
 providedIn: 'root'
})
export class JugadoresService {

 private listaJugadores = new BehaviorSubject<Jugador[]>([]);

 constructor() { }

 get jugadores() {
  return this.listaJugadores.asObservable();
 }
 addJugador(nom: string, foto: string) {
  this.listaJugadores.
   pipe(take(1)).
   subscribe(j => {
    const newJugador = new Jugador(this.calculaNuevaId(j), nom, foto);
    this.listaJugadores.next(j.concat(newJugador));
   });
  this.setLista();
 }
```

```
calculaNuevaId(lista: Jugador[]) {
  let id = 1;
  let existe;
  do {
    existe = false;
    lista.filter(j => {
      if (j.id === id) {
        existe = true;
        id++;
      }
    });
  } while (existe);
  return id;
}
deleteJugador(id: number) {
  this.listaJugadores.
    pipe(take(1)).
    subscribe(listaJ => {
      this.listaJugadores.next(
        listaJ.filter(j => {
          return j.id !== id;
        })
      );
    });
  this.setLista();
}
async setLista() {
  const { Storage } = Plugins;
  let aux: Jugador[] = [];
  this.listaJugadores.
```

```
    pipe(take(1)).
    subscribe(j => {
      aux = j;
    });
  await Storage.set({
    key: 'lista',
    value: JSON.stringify(aux)
  });
  this.getLista();
}
async getLista() {
  const { Storage } = Plugins;
  const ret = await Storage.get({ key: 'lista' });
  let lista = JSON.parse(ret.value);
  if (lista === null) { // En caso de que la base de datos no exista
    await Storage.set({
      key: 'lista',
      value: '[]'
    });
    lista = '[]';
  }
  this.listaJugadores.
    pipe(take(1)).
    subscribe(() => {
      this.listaJugadores.next(lista);
    });
}
async sacaFoto() {
  const { Camera } = Plugins;
  if (!Capacitor.isPluginAvailable('Camera')) {
```

```
    return;
   }
   const imagen = await Camera.getPhoto({
    quality: 50,
    source: CameraSource.Prompt, // Con prompt hacemos que el usuario escoja entre sacar
foto o buscar en galería
    correctOrientation: true,
    resultType: CameraResultType.Uri
   });
   const imageUrl = imagen.webPath;
   console.log('Url imagen servicio: ' + imageUrl);
   return imageUrl;
  }
}
```

Partido

```
import { Injectable } from '@angular/core';
import { Carta } from './carta.model';
import { Jugador } from 'src/app/jugadores/jugador.model';

@Injectable({
  providedIn: 'root'
})
export class PartidoService {
  private listaCartas: Carta[] = [
    new Carta(
      'Espadas',
      1,
      'assets/images/cartas/espadas/1.jpg'
    ),
    new Carta(
      'Espadas',
      2,
      'assets/images/cartas/espadas/2.jpg'
    ),
    new Carta(
      'Espadas',
      3,
      'assets/images/cartas/espadas/3.jpg'
    ),
    new Carta(
      'Espadas',
      4,
      'assets/images/cartas/espadas/4.jpg'
    ),
```

```
new Carta(
  'Espadas',
  5,
  'assets/images/cartas/espadas/5.jpg'
),
new Carta(
  'Espadas',
  6,
  'assets/images/cartas/espadas/6.jpg'
),
new Carta(
  'Espadas',
  7,
  'assets/images/cartas/espadas/7.jpg'
),
new Carta(
  'Espadas',
  10,
  'assets/images/cartas/espadas/10.jpg'
),
new Carta(
  'Espadas',
  11,
  'assets/images/cartas/espadas/11.jpg'
),
new Carta(
  'Espadas',
  12,
  'assets/images/cartas/espadas/12.jpg'
),
```

```
new Carta(
  'Copas',
  1,
  'assets/images/cartas/copas/1.jpg'
),
new Carta(
  'Copas',
  2,
  'assets/images/cartas/copas/2.jpg'
),
new Carta(
  'Copas',
  3,
  'assets/images/cartas/copas/3.jpg'
),
new Carta(
  'Copas',
  4,
  'assets/images/cartas/copas/4.jpg'
),
new Carta(
  'Copas',
  5,
  'assets/images/cartas/copas/5.jpg'
),
new Carta(
  'Copas',
  6,
  'assets/images/cartas/copas/6.jpg'
),
```

```
new Carta(
  'Copas',
  7,
  'assets/images/cartas/copas/7.jpg'
),
new Carta(
  'Copas',
  10,
  'assets/images/cartas/copas/10.jpg'
),
new Carta(
  'Copas',
  11,
  'assets/images/cartas/copas/11.jpg'
),
new Carta(
  'Copas',
  12,
  'assets/images/cartas/copas/12.jpg'
),
new Carta(
  'Bastos',
  1,
  'assets/images/cartas/bastos/1.jpg'
),
new Carta(
  'Bastos',
  2,
  'assets/images/cartas/bastos/2.jpg'
),
```

```
new Carta(
  'Bastos',
  3,
  'assets/images/cartas/bastos/3.jpg'
),
new Carta(
  'Bastos',
  4,
  'assets/images/cartas/bastos/4.jpg'
),
new Carta(
  'Bastos',
  5,
  'assets/images/cartas/bastos/5.jpg'
),
new Carta(
  'Bastos',
  6,
  'assets/images/cartas/bastos/6.jpg'
),
new Carta(
  'Bastos',
  7,
  'assets/images/cartas/bastos/7.jpg'
),
new Carta(
  'Bastos',
  10,
  'assets/images/cartas/bastos/10.jpg'
),
```

```
new Carta(
  'Bastos',
  11,
  'assets/images/cartas/bastos/11.jpg'
),
new Carta(
  'Bastos',
  12,
  'assets/images/cartas/bastos/12.jpg'
),
new Carta(
  'Oros',
  1,
  'assets/images/cartas/oros/1.jpg'
),
new Carta(
  'Oros',
  2,
  'assets/images/cartas/oros/2.jpg'
),
new Carta(
  'Oros',
  3,
  'assets/images/cartas/oros/3.jpg'
),
new Carta(
  'Oros',
  4,
  'assets/images/cartas/oros/4.jpg'
),
```

```
new Carta(
  'Oros',
  5,
  'assets/images/cartas/oros/5.jpg'
),
new Carta(
  'Oros',
  6,
  'assets/images/cartas/oros/6.jpg'
),
new Carta(
  'Oros',
  7,
  'assets/images/cartas/oros/7.jpg'
),
new Carta(
  'Oros',
  10,
  'assets/images/cartas/oros/10.jpg'
),
new Carta(
  'Oros',
  11,
  'assets/images/cartas/oros/11.jpg'
),
new Carta(
  'Oros',
  12,
  'assets/images/cartas/oros/12.jpg'
)
```

```
];
private cartaSinTriunfo: Carta = new Carta(
  '',
  0,
  'assets/images/cartas/SinTriunfo.jpg'
);

private cartaSinBaraja: Carta = new Carta(
  'Reinicio',
  0,
  'assets/images/cartas/Reinicio.jpg'
);
jEquipo1: Jugador[];
jEquipo2: Jugador[];
tipoP: string;
nomE1: string;
nomE2: string;
dificultad: number;

get baraja() {
  return this.listaCartas;
}

get sinTriunfo() {
  return this.cartaSinTriunfo;
}

get sinBaraja() {
  return this.cartaSinBaraja;
}
```

```
  generaPartido(equipo1: Jugador[], equipo2: Jugador[], tipo: string, n1: string, n2: string, dif:
number) {
    this.jEquipo1 = equipo1;
    this.jEquipo2 = equipo2;
    this.tipoP = tipo;
    this.nomE1 = n1;
    this.nomE2 = n2;
    this.dificultad = dif;
  }

  constructor() { }
}
```

Audio

```
import { Injectable } from '@angular/core';
import { Platform } from '@ionic/angular';
import { NativeAudio } from '@ionic-native/native-audio/ngx';

interface Sound {
  key: string;
  asset: string;
  isNative: boolean;
}

@Injectable({
  providedIn: 'root'
})
export class AudioService {

  private sounds: Sound[] = [];
  private audioPlayer: HTMLAudioElement = new Audio();
  private forceWebAudio = true;
  private enableSound = true;

  constructor(private platform: Platform, private nativeAudio: NativeAudio) {

  }

  setEnableSound(enable: boolean) {
    this.enableSound = enable;
  }
  preload(k: string, a: string): void {
```

```
    const aud = new Audio();
    aud.src = a;

    this.sounds.push({
      key: k,
      asset: a,
      isNative: false
    });

  }

  play(k: string): void {

    if (this.enableSound) {
      const soundToPlay = this.sounds.find((sound) => {
        return sound.key === k;
      });

      this.audioPlayer.src = soundToPlay.asset;
      this.audioPlayer.play();
    }
  }

}
```

Componentes

Menú

HTML

```
<ion-content>
 <ion-grid>
  <ion-row>
   <ion-col class="ion-text-center">
    <h1>
     <ion-text>fiTennis</ion-text>
    </h1>
   </ion-col>
  </ion-row>
  <ion-row class="ion-text-center">
   <ion-col size-xs="10" offset-xs="1" size-sm="8" offset-sm="2">
    <ion-list lines="none" y>
     <ion-item
      [routerLink]="['/', 'crear-partido']"
      class="borde"
      detail="false"
     >
      <ion-icon name="tennisball" slot="start"></ion-icon>
      <ion-label>¡Jugar!</ion-label>
     </ion-item>
     <ion-item
      [routerLink]="['/', 'jugadores']"
      class="borde"
      detail="false"
     >
      <ion-icon name="people" slot="start"></ion-icon>
      <ion-label>Lista de jugadores</ion-label>
     </ion-item>
```

```
        <ion-item [routerLink]="['/', 'reglas']" class="borde" detail="false">
          <ion-icon name="book" slot="start"></ion-icon>
          <ion-label>Reglas</ion-label>
        </ion-item>
        <ion-item lines="none" color="transparent">
          <ion-icon name="volume-mute" slot="end"></ion-icon>
          <ion-toggle
            slot="end"
            (ionChange)="actualizaSonido()"
            checked
          ></ion-toggle>
        </ion-item>
      </ion-list>
    </ion-col>
  </ion-row>
 </ion-grid>
</ion-content>
```

TS

```
import { Component, OnInit } from '@angular/core';

import { ScreenOrientation } from '@ionic-native/screen-orientation/ngx';
import { AudioService } from '../audio/audio.service';

@Component({
 selector: 'app-menu',
 templateUrl: './menu.page.html',
 styleUrls: ['./menu.page.scss'],
})
export class MenuPage implements OnInit {
```

```
  private sonidoActivo: boolean;
  constructor(
    private screenOrientation: ScreenOrientation,
    private audioSrv: AudioService
  ) { }

  ngOnInit() {
    this.actualizaSonido();
    this.screenOrientation.lock(this.screenOrientation.ORIENTATIONS.PORTRAIT);
    this.sonidoActivo = true;
  }
  actualizaSonido() {
    this.sonidoActivo = !this.sonidoActivo;
    this.audioSrv.setEnableSound(this.sonidoActivo);
  }

}
```

SCSS

```
.borde {
  padding-left: 0;
  border-radius: 50px;
  border: 2px solid var(--ion-color-primary);
  margin: 15px auto;
  color: black;
  --ion-background-color: rgb(67, 192, 94);
  // rgb(67, 192, 94)
}
ion-list {
  background: transparent;
}
```

```
h1 {
  color: white;
  font-family: "Rouge Script", cursive;
  font-size: 36px;
  font-weight: normal;
  line-height: 48px;
  margin: 40px 0 20px;
  text-align: center;
  text-shadow: 1px 1px 2px #082b34;
}
```

Jugadores

HTML

```
 <ion-toolbar color="translucent">
   <ion-buttons slot="start">
     <ion-back-button defaultHref="/menu" color="light"></ion-back-button>
   </ion-buttons>
 </ion-toolbar>
<ion-content fullscreen>
 <ion-grid class="ion-text-center">
   <ion-row>
     <ion-col class="ion-text-center">
       <h2>
         <ion-text>Jugadores</ion-text>
       </h2>
     </ion-col>
   </ion-row>
   <ion-row>
     <ion-col size-xs="10" offset-xs="1" size-sm="8" offset-sm="2">
       <ion-list>
         <ion-item *ngFor="let jugador of listaJugadores" lines="none">
           <ion-avatar slot="start"
             ><ion-img [src]="jugador.imagen"></ion-img
           ></ion-avatar>
           <ion-label>{{ jugador.nombre }}</ion-label>
           <ion-button (click)="onDeleteJugador(jugador.id)" fill="clear">
             <ion-icon name="trash" slot="icon-only" ></ion-icon>
           </ion-button>
         </ion-item>
         <ion-item lines="none" (click)="onCrearJugador()">
           <ion-icon name="add-circle" slot="start"></ion-icon>
```

```
          <ion-label>Añadir jugador</ion-label>
        </ion-item>
      </ion-list>
    </ion-col>
  </ion-row>
 </ion-grid>
</ion-content>
```

TS

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Jugador } from './jugador.model';
import { JugadoresService } from './jugadores.service';
import { ModalController, AlertController } from '@ionic/angular';
import { CrearJugadorComponent } from './crear-jugador/crear-jugador.component';
import { Subscription } from 'rxjs';

@Component({
 selector: 'app-jugadores',
 templateUrl: './jugadores.page.html',
 styleUrls: ['./jugadores.page.scss'],
})
export class JugadoresPage implements OnInit, OnDestroy {

 listaJugadores: Jugador[];
 private jugadoresSub: Subscription;
 constructor(private jugadoresSrv: JugadoresService, private modalCtrl: ModalController,
private alertCtrl: AlertController) { }

 ngOnInit() {
  this.jugadoresSrv.getLista();
  this.jugadoresSub = this.jugadoresSrv.jugadores.subscribe(jugadores => {
```

```
    this.listaJugadores = jugadores;
  });
}

ngOnDestroy() {
  if (this.jugadoresSub) {
    this.jugadoresSub.unsubscribe();
  }
}

onCrearJugador() {
  this.modalCtrl
    .create({ component: CrearJugadorComponent })
    .then(modalEl => {
      modalEl.present();
      return modalEl.onDidDismiss();
    });
}
onDeleteJugador(id: number) {
  this.alertCtrl.create({
    header: 'Eliminar jugador',
    message: '¿Estás seguro de que quieres eliminar a este jugador?',
    buttons: [
      {
        text: 'Sí',
        handler: () => {
          this.jugadoresSrv.deleteJugador(id);
        }
      }, {
        text: 'No',
```

```
        role: 'cancel'
      }]
    })
     .then(alertEl2 => {
      alertEl2.present();
     });

  }
}
```

SCSS

```
h2 {
  color: white;
  font-family: "Rouge Script", cursive;
  font-size: 24px;
  font-weight: normal;
  line-height: 48px;
  margin: 0px 0px 0px;
  text-align: center;
  text-shadow: 1px 1px 2px #082b34;
 }
 ion-list{
  border-radius: 40px;
  border: 2px solid var(--ion-color-primary);
  padding: 15px;
  --ion-background-color: rgb(67, 192, 94);
  ion-label{
   color: black;
  }
  ion-icon{
   color: black;
```

```
    }
}
```

Crear jugador

HTML

```
<ion-header>
  <ion-toolbar color="translucent">
    <ion-buttons slot="primary">
      <ion-button (click)="onCancel()" color="light">
        <ion-icon name="close"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content class="ion-text-center" fullscreen>
  <form (ngSubmit)="onAddJugador()" #f="ngForm">
    <ion-grid>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-list>
            <ion-item>
              <ion-label position="floating">Nombre</ion-label>
              <ion-input
                type="text"
                ngModel
                name="nombre"
                required
                maxlength="15"
              ></ion-input>
            </ion-item>
            <ion-item lines="none">
              <ion-img class="image" [src]="urlImagen"></ion-img>
              <ion-button color="primary" (click)="onAddFoto()">
```

```
            <ion-icon name="camera" slot="start"></ion-icon>
            <ion-label>¡Añade una foto!</ion-label>
          </ion-button>
        </ion-item>
        <ion-item lines="none">
          <ion-button
            type="submit"
            color="primary"
            expand="block"
            [disabled]="!f.valid"
            >¡Añadir jugador!</ion-button
          >
        </ion-item>
      </ion-list>
    </ion-col>
  </ion-row>

  </ion-grid>
 </form>
</ion-content>
```

TS

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { ModalController } from '@ionic/angular';
import { JugadoresService } from '../jugadores.service';
import { Jugador } from '../jugador.model';
import { NgForm } from '@angular/forms';

@Component({
 selector: 'app-crear-jugador',
 templateUrl: './crear-jugador.component.html',
```

```
  styleUrls: ['./crear-jugador.component.scss'],
})
export class CrearJugadorComponent implements OnInit {

  @ViewChild('f', { static: false }) form: NgForm;
  urlImagen = '/assets/images/SinFoto.jpg';
  constructor(
    private modalCtrl: ModalController,
    private jugadoresSrv: JugadoresService
  ) { }

  ngOnInit() {
  }


  onAddJugador() {
    if (!this.form.valid) {
      return;
    }
    console.log('Url imagen componente: ' + this.urlImagen);
    this.jugadoresSrv.addJugador(this.form.value['nombre'], this.urlImagen);
    this.onCancel();
  }

  onCancel() {
    this.modalCtrl.dismiss(null, 'cancel');
  }
  async onAddFoto() {
    this.urlImagen = await this.jugadoresSrv.sacaFoto();
    console.log('Url imagen método addFoto: ' + this.urlImagen);
```

```
  }
}
```

SCSS

```
.picker {
    width: 30 rem;
    max-width: 80%;
    height: 20 rem;
    max-height: 30vh;
    margin: auto;
    display: flex;
    justify-content: center;
    align-items: center;
}
.image{
    width: 100%;
    height: 100%;
    object-fit: cover;
}
ion-list{
    border-radius: 40px;
    border: 2px solid var(--ion-color-primary);
    padding: 15px;
    --ion-background-color: rgb(67, 192, 94);
}
ion-item{
    margin: 15px;
}
```

Crear partido

HTML

```
<ion-toolbar color="translucent">
  <ion-buttons slot="start">
    <ion-back-button defaultHref="/menu" color="light"></ion-back-button>
  </ion-buttons>
</ion-toolbar>

<ion-content class="ion-text-center" fullscreen>
  <ion-grid *ngIf="mostrarJugadores">
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <form
          #fJ="ngForm"
          (ngSubmit)="compruebaJugadoresSuficientesDisponibles()"
          *ngIf="mostrarJugadores"
        >
          <ion-list>
            <ion-item>
              <ion-label position="floating">Número de jugadores</ion-label>
              <ion-select [ngModel]="'1'" name="numJ" interface="popover">
                <ion-select-option value="1">1 vs 1</ion-select-option>
                <ion-select-option value="2">2 vs 2</ion-select-option>
                <ion-select-option value="3">3 vs 3</ion-select-option>
              </ion-select>
            </ion-item>
            <ion-item>
              <ion-label position="floating">Nombre Equipo 1</ion-label>
              <ion-input
                type="text"
```

```
        ngModel="Equipo 1"
        name="nomE1"
        required
        maxlength="10"
       ></ion-input>
      </ion-item>
      <ion-item>
       <ion-label position="floating">Nombre Equipo 2</ion-label>
       <ion-input
        type="text"
        ngModel="Equipo 2"
        name="nomE2"
        maxlength="10"
        required
       ></ion-input>
      </ion-item>
      <ion-item lines="none">
       <ion-button
        type="submit"
        color="primary"
        expand="block"
        slot="end"
        >Seleccionar</ion-button
       >
      </ion-item>
     </ion-list>
    </form>
   </ion-col>
  </ion-row>
  <ion-row>
```

```
<ion-col size-md="4" offset-md="4">
 <ion-list>
  <h3>Jugadores registrados</h3>
  <ion-item *ngIf="mostrarBotonesJugadores">
   <ion-label> {{ nomEq1 }}: E1 </ion-label>
   <ion-label> {{ nomEq2 }}: E2 </ion-label>
  </ion-item>
  <ion-item *ngFor="let jugador of listaJugadores" lines="none">
   <ion-avatar slot="start"
    ><ion-img [src]="jugador.imagen"></ion-img
   ></ion-avatar>
   <ion-label>{{ jugador.nombre }}</ion-label>
   <ion-button
    (click)="addJugadorEquipo(jugador, true)"
    *ngIf="mostrarBotonesJugadores"
    [disabled]="estaEquipo(jugador.id, true)"
   >
    <ion-label>E1</ion-label>
   </ion-button>
   <ion-button
    (click)="addJugadorEquipo(jugador, false)"
    *ngIf="mostrarBotonesJugadores"
    [disabled]="estaEquipo(jugador.id, false)"
   >
    <ion-label>E2</ion-label>
   </ion-button>
   <ion-button
    (click)="borrarDeEquipo(jugador.id)"
    fill="solid"
    *ngIf="mostrarBotonesJugadores"
```

```
        >
          <ion-icon name="close"></ion-icon>
        </ion-button>
      </ion-item>
    </ion-list>
    <ion-button
      (click)="muestraTipoPartido()"
      *ngIf="mostrarBotonesJugadores"
      color="primary"
      expand="block"
      [disabled]="!compruebaNumJugadores()"
      >Terminar selección</ion-button
    >
   </ion-col>
  </ion-row>
</ion-grid>

<ion-grid>
  <ion-row>
   <ion-col size-sm="6" offset-sm="3">
    <form
      (ngSubmit)="empiezaPartido()"
      #fP="ngForm"
      *ngIf="!mostrarJugadores"
    >
      <ion-list>
        <ion-item>
          <ion-label position="floating">Tipo de partido</ion-label>
          <ion-select [ngModel]="'3'" name="tipoP" interface="popover">
            <ion-select-option value="tb">Tie-Break</ion-select-option>
```

```
        <ion-select-option value="3"
          >Partido a 3 juegos</ion-select-option
        >
        <ion-select-option value="s">Partido a 1 set</ion-select-option>
        <ion-select-option value="p"
          >Partido completo</ion-select-option
        >
      </ion-select>
    </ion-item>
    <ion-item>
      <ion-label position="floating">Dificultad</ion-label>
      <ion-select [ngModel]="'2'" name="dif" interface="popover">
        <ion-select-option value="1">Nivel 1</ion-select-option>
        <ion-select-option value="2">Nivel 2</ion-select-option>
        <ion-select-option value="3">Nivel 3</ion-select-option>
      </ion-select>
    </ion-item>
    <ion-item lines="none">
      <ion-button
        type="submit"
        color="primary"
        expand="block"
        slot="end"
        [disabled]="!fP.valid"
        >¡Empezar partido!</ion-button
      >
    </ion-item>
  </ion-list>
 </form>
</ion-col>
```

```
  </ion-row>
 </ion-grid>
</ion-content>
```

TS

```
import { Component, OnInit, ViewChild, OnDestroy } from '@angular/core';
import { JugadoresService } from 'src/app/jugadores/jugadores.service';
import { Jugador } from 'src/app/jugadores/jugador.model';
import { Router } from '@angular/router';
import { NgForm } from '@angular/forms';
import { AlertController } from '@ionic/angular';
import { PartidoService } from '../jugar-partido/partido.service';
import { Subscription } from 'rxjs';

@Component({
 selector: 'app-crear-partido',
 templateUrl: './crear-partido.page.html',
 styleUrls: ['./crear-partido.page.scss'],
})
export class CrearPartidoPage implements OnInit, OnDestroy {

 listaJugadores: Jugador[];
 jugadoresEquipo1: Jugador[] = [];
 jugadoresEquipo2: Jugador[] = [];
 mostrarJugadores = true;
 mostrarBotonesJugadores = false;
 totalJugadores: number = 0;
 @ViewChild('fJ', { static: false }) formJ: NgForm;
 @ViewChild('fP', { static: false }) formT: NgForm;
 private jugadoresSub: Subscription;
 nomEq1: string;
```

```
nomEq2: string;

constructor(
  private jugadoresSrv: JugadoresService,
  private partidoSrv: PartidoService,
  private router: Router,
  private alertCtrl: AlertController
) { }

ngOnInit() {
  this.jugadoresSrv.getLista();
  this.jugadoresSub = this.jugadoresSrv.jugadores.subscribe(jugadores => {
    this.listaJugadores = jugadores;
  });
}

ngOnDestroy() {
  if (this.jugadoresSub) {
    this.jugadoresSub.unsubscribe();
  }
}

reiniciar() {
  this.mostrarJugadores = true;
  this.mostrarBotonesJugadores = false;
  this.totalJugadores = 0;
  this.jugadoresEquipo1 = [];
  this.jugadoresEquipo2 = [];
}
colocaJugadores() {
```

```
  // Setear jugadores equipo 1 y jugadores equipo 2. Vigilar que no sean ni más ni menos de los necesarios


  this.totalJugadores = +this.formJ.value['numJ'];
  if (this.jugadoresEquipo2.length === this.totalJugadores && this.jugadoresEquipo1.length === this.totalJugadores) {
    this.mostrarJugadores = false;
  }
 }
 empiezaPartido() {
  this.partidoSrv.generaPartido(this.jugadoresEquipo1, this.jugadoresEquipo2, this.formT.value['tipoP'], this.nomEq1, this.nomEq2, +this.formT.value['dif']);
  this.reiniciar();
  this.router.navigate(['/jugar-partido']);
 }
 addJugadorEquipo(j: Jugador, equipo1: boolean) {
  // Borrar si está en el equipo contrario
  if (this.estaEquipo(j.id, !equipo1)) {
    this.borrarDeEquipo(j.id);
  }


  if (equipo1 && !this.estaEquipo(j.id, equipo1)) {
    this.jugadoresEquipo1.push(j);
    return;
  }
  if (!equipo1 && !this.estaEquipo(j.id, equipo1)) {
    this.jugadoresEquipo2.push(j);
    return;
  }
 }
 estaEquipo(id: number, equipo1: boolean) {
```

```
  let aux = false;
  if (equipo1) {
    this.jugadoresEquipo1.filter(j => {
      if (j.id === id) {
        aux = true;
      }
    });
  } else {
    this.jugadoresEquipo2.filter(j => {
      if (j.id === id) {
        aux = true;
      }
    });
  }
  return aux;
}
borrarDeEquipo(id: number) {
  this.jugadoresEquipo1 = this.jugadoresEquipo1.filter(j => {
    return (j.id !== id);
  });
  this.jugadoresEquipo2 = this.jugadoresEquipo2.filter(j => {
    return (j.id !== id);
  });
}
compruebaNumJugadores() {
  if (this.jugadoresEquipo2.length === this.totalJugadores / 2 && this.jugadoresEquipo1.length
=== this.totalJugadores / 2) {
    return true;
  }
  return false;
```

```
}
compruebaJugadoresSuficientesDisponibles() {
  let aux: number;
  aux = 2 * (+this.formJ.value['numJ']);
  if (aux > this.listaJugadores.length) { // No suficientes
    this.reiniciar();
    this.alertCtrl.create({
      header: 'Jugadores insuficientes',
      message: 'No tienes suficientes jugadores registrados para poder jugar este partido',
      buttons: [{
        text: 'Ok',
        role: 'cancel'
      },
      {
        text: 'Editar lista de jugadores',
        handler: () => {
          this.router.navigate(['jugadores']);
        }
      }]
    })
      .then(alertEl2 => {
        alertEl2.present();
      });
    return;
  }
  this.totalJugadores = aux;
  this.nomEq1 = this.formJ.value['nomE1'];
  this.nomEq2 = this.formJ.value['nomE2'];
  this.mostrarBotonesJugadores = true;
  return;
```

```
  }
  muestraTipoPartido() {
    this.mostrarJugadores = false;
  }
}
```

SCSS

```
ion-list{
    border-radius: 40px;
    border: 2px solid var(--ion-color-primary);
    padding: 15px;
    --ion-background-color: rgb(67, 192, 94);
}
```

Jugar partido

HTML

```
<ion-content>
 <ion-grid>
  <ion-row class="ion-align-items-end">
   <ion-col class="girar180" size ="6">
     <ion-list *ngIf="existeTriunfo" class="fondoVerde">
       <ion-item  lines="none">
        <ion-label
          >{{ triunfoActual.palo }}: {{ sumaTriunfosEquipo2 }}</ion-label
        ></ion-item
       >
       <ion-item lines="none">
        <ion-label
          >Resto: {{ sumaEquipo2 }}</ion-label
        >
       </ion-item>
     </ion-list>
   </ion-col>
   <ion-col class="girar180" size="6">
     <ion-list class="fondoVerde">
       <ion-item>
        <h3>{{nombreEquipo2}}</h3>
       </ion-item>
       <ion-item lines="none">
        <ion-avatar *ngFor="let jug2 of jugadoresEquipo2">
          <ion-img [src]="jug2.imagen"></ion-img
        ></ion-avatar>
       </ion-item>
     </ion-list>
```

```
  </ion-col>
</ion-row>
<ion-row>
  <ion-col size="8">
    <ion-item
      lines="none"
      (click)="onCarta()"
      class="bordeCarta"
      color="transparent"
    >
      <ion-img [src]="cartaActual.imagen" style="width:100%"></ion-img>
    </ion-item>
  </ion-col>
  <ion-col size="4">
    <ion-item lines="none" class="bordeCarta" color="transparent">
      <ion-img [src]="triunfoActual.imagen" style="width:100%"></ion-img>
    </ion-item>
    <ion-item class="marcador" lines="none" color="transparent">
      <div class="row header">
        <div class="col"><ion-text color="light">E\S</ion-text></div>
        <div class="col"><ion-text color="light">{{nombreEquipo1}}</ion-text></div>
        <div class="col"><ion-text color="light">{{nombreEquipo2}}</ion-text></div>
      </div>
      <div class="row">
        <div class="col"><ion-text color="light">1</ion-text></div>
        <div class="col">
          <ion-text>{{ juegosEquipo1[0] }}</ion-text>
        </div>
        <div class="col">
          <ion-text>{{ juegosEquipo2[0] }}</ion-text>
```

```
  </div>
 </div>
 <div class="row">
  <div class="col"><ion-text color="light">2</ion-text></div>
  <div class="col">
   <ion-text>{{ juegosEquipo1[1] }}</ion-text>
  </div>
  <div class="col">
   <ion-text>{{ juegosEquipo2[1] }}</ion-text>
  </div>
 </div>
 <div class="row">
  <div class="col"><ion-text color="light">3</ion-text></div>
  <div class="col">
   <ion-text>{{ juegosEquipo1[2] }}</ion-text>
  </div>
  <div class="col">
   <ion-text>{{ juegosEquipo2[2] }}</ion-text>
  </div>
 </div>
 <div class="row">
  <div class="col"><ion-text color="light">J</ion-text></div>
  <div class="col">
   <ion-text>{{ puntos[puntosEquipo1] }}</ion-text>
  </div>
  <div class="col">
   <ion-text>{{ puntos[puntosEquipo2] }}</ion-text>
  </div>
 </div>
</ion-item>
```

```
  </ion-col>
</ion-row>
<ion-row class="ion-align-items-start">
  <ion-col size="6">
    <ion-list class="fondoVerde">
      <ion-item>
        <h3>{{nombreEquipo1}}</h3>
      </ion-item>
      <ion-item lines="none">
        <ion-avatar *ngFor="let jug1 of jugadoresEquipo1">
          <ion-img [src]="jug1.imagen"></ion-img
        ></ion-avatar>
      </ion-item>
    </ion-list>
  </ion-col>
  <ion-col size="6">
    <ion-list *ngIf="existeTriunfo" class="fondoVerde">
      <ion-item lines="none">
        <ion-label
          >{{ triunfoActual.palo }}: {{ sumaTriunfosEquipo1 }}</ion-label
        ></ion-item
      >
      <ion-item lines="none">
        <ion-label
          >Resto: {{ sumaEquipo1 }}</ion-label
        >
      </ion-item>
    </ion-list>
  </ion-col>
</ion-row>
```

```
  </ion-grid>
  <ion-button (click)="volverMenu()" fill="solid" color="light">
    <ion-icon name="close" color="danger"></ion-icon>
  </ion-button>
</ion-content>
```

TS

```
import { Component, OnInit } from '@angular/core';
import { Carta } from './carta.model';
import { PartidoService } from './partido.service';
import { AlertController } from '@ionic/angular';
import { Jugador } from 'src/app/jugadores/jugador.model';
import { Router } from '@angular/router';
import { AudioService } from 'src/app/audio/audio.service';

@Component({
  selector: 'app-jugar-partido',
  templateUrl: './jugar-partido.page.html',
  styleUrls: ['./jugar-partido.page.scss'],
})
export class JugarPartidoPage implements OnInit {

  private turnoEquipo1: boolean;
  private sacaEquipo1: boolean;
  existeTriunfo: boolean;
  private isTieBreak: boolean;
  private contador: number;
  juegosEquipo1: number[];
  juegosEquipo2: number[];
  setsEquipo1: number;
  setsEquipo2: number;
```

```
setActual: number;
private tipoPartido: string;
nombreEquipo1: string;
nombreEquipo2: string;
private dificultad: number;


private baraja: Carta[];
cartaActual: Carta;
triunfoActual: Carta;
private cartaSinBaraja: Carta;
private cartaSinTriunfo: Carta;


jugadoresEquipo1: Jugador[] = [];
jugadoresEquipo2: Jugador[] = [];

sumaEquipo1: number;
sumaTriunfosEquipo1: number;
sumaEquipo2: number;
sumaTriunfosEquipo2: number;

puntosEquipo1: number;
puntosEquipo2: number;
tragosEquipo1: number;
tragosEquipo2: number;

puntos: string[] = [
  '0',
  '15',
```

```
  '30',
  '40',
  'AD',
  '1',
  '2',
  '3',
  '4',
  '5',
  '6'
 ];
 pruebasN1: string[] = [
  'Cada miembro del equipo perdedor hará 10 flexiones.',
  'Cada miembro del equipo perdedor hará 10 sentadillas.',
  'Cada miembro del equipo perdedor hará 10 zancadas.',
  'Cada miembro del equipo perdedor hará 20 segundos de plancha.',
  'Cada miembro del equipo perdedor hará 30 segundos a sprint.',
  'Cada miembro del equipo perdedor hará 30 segundos haciendo climbers.',
  'Cada miembro del equipo perdedor hará 30 burpees',
  'Cada miembro del equipo perdedor hará 10 jumping jacks'
 ];
 pruebasN2: string[] = [
  'Cada integrante del equipo perdedor deberá hacer 5 flexiones con un miembro del equipo ganador sobre su espalda.',
  'Los miembros del equipo perdedor echan un carrera en el espacio que ellos decidan. El perdedor, hace 10 flexiones, 10 sentadillas y 15 segundos de plancha.',
  'Los miembros del equipo perdedor hacen todos los jumping jacks que puedan en 30 segundos. El que menos haya hecho, hace 10 flexiones.',
  'Los miembros del equipo perdedor hacen todos los burpees que puedan en 30 segundos. El que menos haya hecho, sprinta durante 15 segundos.',
  'Los miembros del equipo perdedor aguantan en plancha hasta que no puedan más. El primero en caer hace 10 flexiones y (en caso de ser 3 jugadores en el equipo) el segundo en caer hace 5 flexiones.',
```

'Los miembros del equipo perdedor aguantan haciendo climbers hasta que no puedan más. El primero en caer hace 10 sentadillas y (en caso de ser 3 jugadores en el equipo) el segundo en caer hace 5 sentadillas.',

'Los miembros del equipo perdedor hacen todas las sentadillas que puedan en 30 segundos. El que menos haya hecho, hace 10 flexiones.',

'Los miembros del equipo perdedor compiten en una prueba de salto de longitud. El que menos distancia salte, hará 10 flexiones.'

];

ejercicios: string[] = [

'plancha. Vídeo explicativo',

'climbers. Vídeo explicativo',

'sprint.',

'flexiones. Vídeo explicativo',

'sentadillas. Vídeo explicativo',

'zancadas. Vídeo explicativo',

'burpees. Vídeo explicativo',

'jumping jacks. Vídeo explicativo'

];

constructor(

private partidoSrv: PartidoService,

private alertCtrl: AlertController,

private router: Router,

private audioSrv: AudioService

) { }

```
ngOnInit() {
  this.cartaSinBaraja = this.partidoSrv.sinBaraja;
  this.cartaSinTriunfo = this.partidoSrv.sinTriunfo;
  this.reiniciarPartido();

  this.audioSrv.preload('celebraPunto', '/assets/audio/celebraPunto.wav');
  this.audioSrv.preload('celebraJuego', '/assets/audio/celebraJuego.wav');
  this.audioSrv.preload('sonidoCarta', '/assets/audio/sonidoCarta.wav');
}

reiniciarPartido() {
  this.recargarBaraja();
  this.reiniciaTriunfo();
  this.turnoEquipo1 = true;
  this.sacaEquipo1 = true;
  this.existeTriunfo = false;

  this.jugadoresEquipo1 = this.partidoSrv.jEquipo1;
  this.jugadoresEquipo2 = this.partidoSrv.jEquipo2;
  this.tipoPartido = this.partidoSrv.tipoP;
  this.nombreEquipo1 = this.partidoSrv.nomE1;
  this.nombreEquipo2 = this.partidoSrv.nomE2;
  this.dificultad = this.partidoSrv.dificultad;

  this.reiniciaContador();

  if (this.tipoPartido === 'tb') {
    this.isTieBreak = true;
  } else {
    this.isTieBreak = false;
```

```
  }
  this.juegosEquipo1 = [0, 0, 0];
  this.juegosEquipo2 = [0, 0, 0];
  this.setsEquipo1 = 0;
  this.setsEquipo2 = 0;
  this.setActual = 0;

  this.sumaEquipo1 = 0;
  this.sumaTriunfosEquipo1 = 0;
  this.sumaEquipo2 = 0;
  this.sumaTriunfosEquipo2 = 0;

  this.puntosEquipo1 = 0;
  this.puntosEquipo2 = 0;
  this.tragosEquipo1 = 0;
  this.tragosEquipo2 = 0;
}
reiniciaContador() {
  this.contador = 2 * this.jugadoresEquipo1.length;
}

reiniciaTriunfo() {
  this.triunfoActual = this.cartaSinTriunfo;
  this.existeTriunfo = false;
}
recargarBaraja() {
  this.baraja = this.partidoSrv.baraja;
  this.cartaActual = this.cartaSinBaraja;
}
```

```
 eliminarCarta(cartaParaEliminar: Carta) {
  this.baraja = this.baraja.filter(cartaBaraja => {
   return (cartaBaraja.palo !== cartaParaEliminar.palo || cartaBaraja.valor !==
cartaParaEliminar.valor);
  });
 }

 onCarta() {
  // Comprueba si es la última carta y reinicia
  if (this.baraja.length === 0) {
   this.alertCtrl.create({
    header: '¡No quedan cartas!',
    message: 'No te preocupes, ahora barajamos otra vez :)',
    buttons: [{
     text: 'Ok',
    }]
   })
    .then(alertEl => {
     alertEl.present();
    });
   this.recargarBaraja();
   return;
  }

  // Sacamos carta al azar y la retiramos de la baraja
  this.cartaActual = this.baraja[Math.floor(Math.random() * (this.baraja.length - 1))];
  this.eliminarCarta(this.cartaActual);

  // Comprobar si es la primera carta del juego para definirlo como triunfo
  if (!this.existeTriunfo) {
```

```
    this.triunfoActual = this.cartaActual;
    this.existeTriunfo = true;
    return;
   }

   // Sumar carta a los puntos de un equipo
   this.audioSrv.play('sonidoCarta');
   this.sumarCarta(this.cartaActual);
   this.contador--;

   // Ver si el punto ha acabado. Si ha acabado, mostrar resultado del punto.
   if (this.contador === 0) {
    this.delay(200).then(() => {
     this.audioSrv.play('celebraPunto');
     const ganador = this.resultadoPunto();
     this.sumarPunto(ganador);
     this.limpiarPuntuaciones();
     this.reiniciaContador();
    });
   }
  }
  async delay(ms: number) {
   await new Promise(resolve => setTimeout(() => resolve(), ms));
  }

  sacaPruebaAlAzar(nivel: boolean) { // True N2, False N1
   if (!nivel) {
    return this.pruebasN1[Math.floor(Math.random() * (this.pruebasN1.length - 1))];
   } else {
    return this.pruebasN2[Math.floor(Math.random() * (this.pruebasN2.length - 1))];
```

```
  }
}
sacaEjercicioAlAzar() {
  const i = Math.floor(Math.random() * (this.ejercicios.length - 1));
  let prueba = '';
  if (i < 3) {
    if (i === 2) {
      prueba += 'segundos a ' + this.ejercicios[i];
    } else {
      prueba += 'segundos haciendo ' + this.ejercicios[i];
    }
  } else {
    prueba += this.ejercicios[i];
  }
  return prueba;
}
sumarPunto(resultado: boolean) {
  if (!this.isTieBreak) {
    if (resultado) {
      // Casos en los que ganas el juego
      if (this.puntosEquipo1 === 4) {
        this.juegosEquipo1[this.setActual]++;
        this.castigo(resultado, true, this.cambiarJuego());
        return;
      }
      if (this.puntosEquipo1 === 3 && this.puntosEquipo2 < 3) {
        this.juegosEquipo1[this.setActual]++;
        this.castigo(resultado, true, this.cambiarJuego());
        return;
      }
```

```
  //Deuce -> Ventaja
  if (this.puntosEquipo1 === 3 && this.puntosEquipo2 === 3) {
    this.puntosEquipo1++;
    this.castigo(resultado, false, false);
    return;
  }

  //Ventaja en contra -> Deuce
  if (this.puntosEquipo1 === 3 && this.puntosEquipo2 === 4) {
    this.puntosEquipo2--;
    this.castigo(resultado, false, false);
    return;
  }
  //Resto de casos
  this.puntosEquipo1++;
  this.castigo(resultado, false, false);
 } else {
  // Casos en los que ganas el juego
  if (this.puntosEquipo2 === 4) {
    this.juegosEquipo2[this.setActual]++;
    this.castigo(resultado, true, this.cambiarJuego());
    return;
  }
  if (this.puntosEquipo2 === 3 && this.puntosEquipo1 < 3) {
    this.juegosEquipo2[this.setActual]++;
    this.castigo(resultado, true, this.cambiarJuego());
    return;
  }
```

```
    //Deuce -> Ventaja
    if (this.puntosEquipo2 === 3 && this.puntosEquipo1 === 3) {
      this.puntosEquipo2++;
      this.castigo(resultado, false, false);
      return;
    }

    //Ventaja en contra -> Deuce
    if (this.puntosEquipo2 === 3 && this.puntosEquipo1 === 4) {
      this.puntosEquipo1--;
      this.castigo(resultado, false, false);
      return;
    }
    //Resto de casos
    this.puntosEquipo2++;
    this.castigo(resultado, false, false);
  }
} else {
  if (resultado) {
    // Primer punto del tie-break
    if (this.puntosEquipo1 === 0) {
      this.puntosEquipo1 = 5;
      this.castigo(resultado, false, false);
      return;
    }

    // Último punto del tie-break
    if (this.puntosEquipo1 === 10) {
      this.juegosEquipo1[this.setActual]++;
      this.castigo(resultado, true, this.cambiarJuego());
```

```
      return;
    }
    this.puntosEquipo1++;
    this.castigo(resultado, false, false);
   } else {
    // Primer punto del tie-break
    if (this.puntosEquipo2 === 0) {
      this.puntosEquipo2 = 5;
      this.castigo(resultado, false, false);
      return;
    }

    // Último punto del tie-break
    if (this.puntosEquipo2 === 10) {
      this.juegosEquipo2[this.setActual]++;
      this.castigo(resultado, true, this.cambiarJuego());
      return;
    }
    this.puntosEquipo2++;
    this.castigo(resultado, false, false);
   }
  }
 }

 muestraResultado() {
  return this.nombreEquipo1 + ': ' + this.sumaTriunfosEquipo1 + ' ' + this.triunfoActual.palo + ',
' + this.sumaEquipo1 + ' resto.<br/>' +
   this.nombreEquipo2 + ': ' + this.sumaTriunfosEquipo2 + ' ' + this.triunfoActual.palo + ', ' +
this.sumaEquipo2 + ' resto.<br/><br/>';
 }
```

```
textoCastigo(tipo: number) { // 1 es punto, 2 es juego, 3 es set y 4 es partido
  let mensaje = ' deberán hacer lo siguiente: ';
  const ejercicio = this.sacaEjercicioAlAzar();
  if (tipo === 1) { // Punto
    mensaje += '5 ' + ejercicio;
  }
  if (tipo === 2) { // Juego
    const n = this.dificultad * 10;
    mensaje = mensaje + n + ' ' + ejercicio;
  }
  if (tipo === 3) { // Set
    const n = this.dificultad * 12;
    mensaje = mensaje + n + ' ' + ejercicio;
  }
  if (tipo === 4) { // Partido
    if (this.dificultad < 3) {
      const n = this.dificultad * 12;
      mensaje = mensaje + n + ' ' + ejercicio;
    } else {
      const n = this.dificultad * 15;
      mensaje = mensaje + n + ' ' + ejercicio;
    }
  }

  return mensaje;
}
castigo(resultado: boolean, hayJuego: boolean, haySet: boolean) {
  // Castigo punto
  let cabeceraFin = '¡Punto para ';
  let mensajeFin = this.muestraResultado();
```

```
if (resultado) {
  cabeceraFin += this.nombreEquipo1 + '!';
} else {
  cabeceraFin += this.nombreEquipo2 + '!';
}
if (this.dificultad === 3) {
  mensajeFin += 'Todos los integrantes de ';
  if (!resultado) {
    mensajeFin += this.nombreEquipo1;
  } else {
    mensajeFin += this.nombreEquipo2;
  }
  mensajeFin = mensajeFin + this.textoCastigo(1);
}
// Pausa para que el jugador vea la carta
this.alertCtrl.create({
  header: cabeceraFin,
  message: mensajeFin,
  buttons: [
    {
      text: '¡Hecho!',
      role: 'cancel',
      handler: () => {
        if (hayJuego) { // Castigo juego
          this.reiniciaTriunfo();
          if (this.tipoPartido !== 'tb') {
            let cabeceraFinJ = '';
            let mensajeFinJ = 'Todos los integrantes de ';
            if (resultado !== this.sacaEquipo1) { // Sin Break
```

```
  cabeceraFinJ += 'Además, ¡Juego para ';
 } else { // Con break
  cabeceraFinJ += 'Además, ¡Break para ';
 }
 if (resultado) {
  cabeceraFinJ += this.nombreEquipo1 + '!';
 } else {
  cabeceraFinJ += this.nombreEquipo2 + '!';
 }
 if (!resultado) {
  mensajeFinJ += this.nombreEquipo1;
 } else {
  mensajeFinJ += this.nombreEquipo2;
 }
 if (resultado !== this.sacaEquipo1) {
  mensajeFinJ = mensajeFinJ + this.textoCastigo(2);
 } else {
  mensajeFinJ = mensajeFinJ +
    this.textoCastigo(2) + ' <br/>Además deberán: <ul><li type="disc">' +
this.sacaPruebaAlAzar(false) + '</li></ul>';
 }

 this.alertCtrl.create({
  header: cabeceraFinJ,
  message: mensajeFinJ,
  buttons: [
   {
    text: '¡Hecho!',
    role: 'cancel',
    handler: () => {
```

```
if (haySet) {// Castigo set
 let cabeceraFinS = 'Y encima, ¡Set para ';
 let mensajeFinS = 'Todos los integrantes de ';
 if (resultado) {
  cabeceraFinS += this.nombreEquipo1 + '!';
 } else {
  cabeceraFinS += this.nombreEquipo2 + '!';
 }
 if (!resultado) {
  mensajeFinS += this.nombreEquipo1;
 } else {
  mensajeFinS += this.nombreEquipo2;
 }
 mensajeFinS = mensajeFinS +
   this.textoCastigo(3) + ' <br/>Además deberán: <ul><li type="disc">' +
this.sacaPruebaAlAzar(true) + '</li></ul>';


 this.alertCtrl.create({
  header: cabeceraFinS,
  message: mensajeFinS,
  buttons: [
   {
    text: '¡Hecho!',
    role: 'cancel',
    handler: () => {
     this.finPartido(resultado);
    }
   }]
 })
```

```
            .then(alertEl2 => {
              alertEl2.present();
            });
          } else {
            this.finPartido(resultado);
          }
        }
      }]
    })
      .then(alertEl2 => {
        alertEl2.present();
      });
  } else {
    this.finPartido(resultado);
  }
 } else { // Cambiar sacador en Tie-Break
  if (this.isTieBreak && (this.puntosEquipo1 + this.puntosEquipo2) % 2 === 1) {
    this.sacaEquipo1 = !this.sacaEquipo1;
    this.cambiarTurno();
    const cabeceraFinTB = '¡Cambio de sacador!';
    let mensajeFinTB = 'Saca ';
    if (this.sacaEquipo1) {
      mensajeFinTB += this.nombreEquipo1;
    } else {
      mensajeFinTB += this.nombreEquipo2;
    }
    this.alertCtrl.create({
      header: cabeceraFinTB,
      message: mensajeFinTB,
      buttons: [
```

```
            {
              text: '¡Ok!',
              role: 'cancel'
            }]
          })
            .then(alertEl2 => {
              alertEl2.present();
            });
        }
      }
    }
  }]
 })
  .then(alertEl2 => {
   alertEl2.present();
  });
}

sumarCarta(carta: Carta) {
 if (this.turnoEquipo1) {
  if (carta.palo === this.triunfoActual.palo) {
   this.sumaTriunfosEquipo1 += carta.valor;
  }
  else {
   this.sumaEquipo1 += carta.valor;
  }
 } else {
  if (carta.palo === this.triunfoActual.palo) {
   this.sumaTriunfosEquipo2 += carta.valor;
  }
```

```
    else {
      this.sumaEquipo2 += carta.valor;
    }
  }
  this.cambiarTurno();
}

resultadoPunto() {
  if (this.sumaTriunfosEquipo1 > this.sumaTriunfosEquipo2) {
    return true;
  }
  if (this.sumaTriunfosEquipo1 < this.sumaTriunfosEquipo2) {
    return false;
  }
  if (this.sumaEquipo1 > this.sumaEquipo2) {
    return true;
  }
  if (this.sumaEquipo1 < this.sumaEquipo2) {
    return false;
  }
  if (this.sacaEquipo1) {
    return true;
  } else {
    return false;
  }

}
limpiarPuntuaciones() {
  this.sumaEquipo1 = 0;
  this.sumaEquipo2 = 0;
```

```
  this.sumaTriunfosEquipo1 = 0;
  this.sumaTriunfosEquipo2 = 0;
}

cambiarTurno() {
  this.turnoEquipo1 = !this.turnoEquipo1;
}

reiniciaJuego() {
  this.sacaEquipo1 = !this.sacaEquipo1;
  this.cambiarTurno();
  this.puntosEquipo1 = 0;
  this.puntosEquipo2 = 0;
}
cambiarJuego() {
  this.audioSrv.play('celebraJuego');
  this.reiniciaJuego();
  // Gana set equipo 1 sin Tie-Break
  if (this.juegosEquipo1[this.setActual] === 6 && this.juegosEquipo2[this.setActual] <= 4 ||
    this.juegosEquipo1[this.setActual] === 7 && this.juegosEquipo2[this.setActual] <= 5) {
    this.setsEquipo1++;
    this.setActual++;
    return true;
  }
  // Gana set equipo 1 con Tie-Break
  if (this.juegosEquipo1[this.setActual] === 7 && this.juegosEquipo2[this.setActual] === 6) {
    this.setsEquipo1++;
    this.setActual++;
    this.isTieBreak = false;
    return false;
```

```
    }
    // Gana set equipo 2 sin Tie-Break
    if (this.juegosEquipo2[this.setActual] === 6 && this.juegosEquipo1[this.setActual] <= 4 ||
      this.juegosEquipo2[this.setActual] === 7 && this.juegosEquipo1[this.setActual] <= 5) {
      this.setsEquipo2++;
      this.setActual++;
      return true;
    }
    // Gana set equipo 2 con Tie-Break
    if (this.juegosEquipo2[this.setActual] === 7 && this.juegosEquipo1[this.setActual] === 6) {
      this.setsEquipo2++;
      this.setActual++;
      this.isTieBreak = false;
      return false;
    }
    // Pasamos a Tie-Break
    if (this.juegosEquipo2[this.setActual] === 6 && this.juegosEquipo1[this.setActual] === 6) {
      this.isTieBreak = true;
      return false;
    }
    return false;
  }
  partidoAcabado() {
    if (this.tipoPartido === 'tb' && (this.juegosEquipo1[this.setActual] === 1 ||
this.juegosEquipo2[this.setActual] === 1) || // Tie-break
      this.tipoPartido === '3' && (this.juegosEquipo1[this.setActual] === 3 ||
this.juegosEquipo2[this.setActual] === 3) || // 3 juegos
      this.tipoPartido === 's' && (this.setsEquipo1 === 1 || this.setsEquipo2 === 1) || // Fin
partido set
      this.tipoPartido === 'p' && (this.setsEquipo1 === 2 || this.setsEquipo2 === 2) // Fin partido
completo
```

```
  ) {
    return true;
  }
  return false;
 }
 finPartido(ganador: boolean) {
  if (this.partidoAcabado()) {
    this.reiniciarPartido();
    let cabeceraFin = '¡Victoria de ';
    let mensajeFin = 'Todos los integrantes de ';
    if (ganador) {
      cabeceraFin += this.nombreEquipo1 + '!';
    } else {
      cabeceraFin += this.nombreEquipo2 + '!';
    }
    if (!ganador) {
      mensajeFin += this.nombreEquipo1;
    } else {
      mensajeFin += this.nombreEquipo1;
    }

    mensajeFin = mensajeFin +
      this.textoCastigo(4) + ' <br/>Además deberán: <ul><li type="disc">' +
this.sacaPruebaAlAzar(true) + '</li></ul>';
    this.alertCtrl.create({
      header: cabeceraFin,
      message: mensajeFin,
      buttons: [{
        text: 'Reiniciar partido'
      },
```

```
    {
      text: 'Volver al menú',
      role: 'cancel',
      handler: () => {
        this.router.navigate(['menu']);
      }
    }]
  })
    .then(alertEl2 => {
      alertEl2.present();
    });
 }
}
volverMenu() {
 this.alertCtrl.create({
  header: '¿Estás seguro de que quieres volver al menú?',
  message: 'Si sales se perderá el progreso del partido',
  buttons: [
    {
      text: 'Sí',
      handler: () => {
        this.reiniciarPartido();
        this.router.navigate(['menu']);
      }
    }, {
      text: 'No',
      role: 'cancel'
    }]
 })
  .then(alertEl2 => {
```

```
      alertEl2.present();
    });
  }
}
```

SCSS

```
ion-content{
  --background: url("/assets/images/fondos/FondoPartido.jpg") 0 0/100% 100% no-repeat;
}
ion-list {
  margin: 0px;
}
ion-avatar {
  height: 25px;
  width: 25px;
}
ion-label{
  font-weight: bold;
}

.girar90 {
  writing-mode: vertical-rl;
}
.giraFoto{
  transform: rotate(270deg);
}
.girar180 {
  display: block;
  transform: rotate(180deg);
}
```

```
.bordeCarta {
  ion-img {
    border: 2px solid var(--ion-color-tertiary);
  }
}
ion-avatar{
  margin: 5px;
}
.marcador {
  writing-mode: vertical-rl;
  font-size: 16px;
  padding: 2px;
}
.col {
  border: solid 1px var(--ion-color-tertiary);
  border-bottom-style: none;
  border-right-style: none;
  padding: 3px;
  background-color: white;
}
.row:last-child .col {
  border-bottom: solid 1px var(--ion-color-tertiary);
}
.col:first-child {
  border-right: solid 1px var(--ion-color-tertiary);
  background-color:  var(--ion-color-primary); // Cabecera de sets y J de juego
}
.header .col {
  background-color: var(--ion-color-primary);
}
```

```
.fondoVerde{
  padding-left: 0;
  border-radius: 40px;
  border: 2px solid var(--ion-color-primary);
  padding: 5px;
  margin: 15px auto;
  --ion-background-color: rgb(67, 192, 94);
}
```

Reglas

HTML

```
<ion-toolbar color="translucent">
  <ion-buttons slot="start">
    <ion-back-button defaultHref="/menu" color="light"></ion-back-button>
  </ion-buttons>
</ion-toolbar>


<ion-content fullscreen>
  <ion-grid class="ion-text-center">
    <ion-row>
      <ion-col class="ion-text-center">
        <h2>
          Reglas
        </h2>
        <ion-list>
          <ion-item lines="none">
            <ion-text>
              <ol>
                <li>Se recomienda jugar a este juego con ropa que estéis dispuestos a sudar y en un espacio amplio.</li>
                <br/><li>Para poder jugar un partido, los participantes deben estar registrados en la aplicación. Esto se hace desde la opción "lista de jugadores" del menú principal.</li>
                <br/><li>El marcador de un partido es similar al de un partido de tenis. Hay cuatro modalidades de partido. Partido completo (3 sets), 1 set, 3 juegos o Tie-Break (el primero en llegar a 7 puntos gana).</li>
                <br/><li>En cada punto, cada jugador de cada equipo saca una carta. La carta que más vale es el 12 y la que menos el 1. Además, la primera carta de cada juego es para determinar el triunfo. Cualquier carta del palo del triunfo le gana a cualquier carta que no lo sea. El equipo cuya suma de triunfos sea mayor, o si no los hay, cuya suma global, gana el punto.</li>
                <br/><li>El equipo que pierda un punto, un juego, un set o un partido tendrá una penalización en forma ejercicio físico.</li>
```

```
            <br/><li>Recordamos que el partido se puede ganar a los puntos (llegando hasta el final del partido) o por KO (que uno de los participantes diga que ya no puede más).</li>
          </ol>
        </ion-text>
      </ion-item>
    </ion-list>
  </ion-col>
 </ion-row>
</ion-grid>
</ion-content>
```

TS

```
import { Component, OnInit } from '@angular/core';

@Component({
 selector: 'app-reglas',
 templateUrl: './reglas.page.html',
 styleUrls: ['./reglas.page.scss'],
})
export class ReglasPage implements OnInit {

 constructor() { }

 ngOnInit() {
 }

}
```

SCSS

```
h2 {
 color: white;
 font-family: "Rouge Script", cursive;
 font-size: 24px;
```

```
  font-weight: normal;
  line-height: 48px;
  margin: 0px 0px 0px;
  text-align: center;
  text-shadow: 1px 1px 2px #082b34;
}

ion-list{
  padding-left: 0;
  border-radius: 40px;
  border: 2px solid var(--ion-color-primary);
  padding: 5px;
  margin: 15px auto;
  --ion-background-color: rgb(67, 192, 94);
}
ion-text{
  text-align: justify;
}
```

Objetos predefinidos

Jugador

```
export class Jugador {

   public id: number;

   public nombre: string;

   public imagen: string;

   constructor(idJ: number, nombreJ: string, imagenJ: string) {

      this.id = idJ;

      this.nombre = nombreJ;

      this.imagen = imagenJ;

   }

}
```

Carta

```
export class Carta {

   constructor(

      public palo: string,

      public valor: number,

      public imagen: string

      ){ }

}
```