Trabajo de Fin de Título

Grado en Ingeniería Informática – Tecnologías de la Información

Aplicación móvil multiplataforma del sitio web Todotorneos.com

Alumno

Javier Sintes Medina

Tutor

Dr. Alexis Quesada Arencibia – Ciencias de la Computación e Inteligencia Artificial

Julio 2020 - Las Palmas de Gran Canaria

Agradecimientos

Agradezco la ayuda prestada y el tiempo dedicado a mi tutor, Dr. Alexis Quesada, que siempre ha estado disponible cuando lo necesitaba y ha aportado tanto en mi formación. Agradezco también a José Luis Macías Rodríguez por su implicación en este proyecto y su colaboración para llevarlo a cabo.

A mis compañeros de clase y amigos Pablo Hernández y Adrián García, por haberme acompañado y apoyado durante estos años y por haber colaborado en el desarrollo de las pruebas de este proyecto.

Por último, agradezco a mi familia por confiar en mí desde el primer día y enseñarme a no rendirme ante la adversidad y a Geno Bernabé por estar siempre a mi lado y ser, junto con mi familia, mis mayores motivaciones.

Resumen

El sitio web todotorneos.com permite la creación y gestión de torneos por parte de un usuario registrado, que ejerce el rol de administrador y es el único responsable del torneo e interactúa directamente con él. Los participantes deben acceder constantemente a la web, concretamente a cada torneo en el que participan, para conocer resultados y fechas de los próximos partidos.

La aplicación para dispositivos móviles propuesta permite la interacción del participante con el torneo, facilitando el acceso a resultados, fechas, clasificación y ofreciendo la posibilidad de establecer resultados de sus partidos, ahorrando gestión al administrador. Además, se hace uso de recordatorios y notificaciones para involucrar al competidor de forma activa y facilitar el acceso a información del mismo.

Abstract

The website todotorneos.com allows the creation and management of tournaments by a registered user, who plays the role of administrator and is the only one responsible of the tournament and interacts directly with it. The participants must constantly access to the website, specifically to each tournament in which they participate, to know results and dates of the next matches.

The proposed mobile application allows the participant to interact with the tournament, facilitating access to results, dates, classification and offering the possibility of establishing results of their games, saving management to the administrator. In addition, reminders and notifications are used to involve the competitor and facilitate access to information about it.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

Introducción

La realización de competiciones no profesionales es una actividad cada vez más común en la sociedad actual, pues fomenta la práctica deportiva y es utilizada para la creación de eventos de ocio o sencillamente con fines lúdicos. Sin embargo, la complejidad de la organización de estas competiciones crece acorde con el número de participantes que participan en ella y más aún si todos ellos no se encuentran en un mismo lugar físico.

Con la intención de solventar este tipo de problemas aparecen distintas herramientas web que permiten organizar una competición de forma sencilla, con capacidad para generar los cruces entre equipos o servir como fuente de información en la que consultar la clasificación, los resultados o los partidos posteriores.

Todotorneos.com es una web que se presenta como una opción para facilitar la administración de competiciones, ofreciendo no solo la gestión y almacenamiento de los datos de esta, sino también la posibilidad de diseñar un sitio web personalizado y aumentar el intercambio de información entre el administrador y los participantes. Esta web lleva operativa más de una década y ha publicado y gestionado más de 42.000 competiciones, lo que aporta confianza a los usuarios que buscan un sistema para gestionar este tipo de eventos.

Las funcionalidades del portal descritas anteriormente se encuentran claramente orientadas a los administradores de competiciones, pues se les facilita en gran medida el trabajo requerido para reunir a los competidores y recabar la información necesaria. Sin embargo, el rol de competidor o participante no es contemplado en este sistema y estos no tienen posibilidad de interactuar en la competición ni de acceder rápidamente a los datos de mayor interés.

La propuesta de este Trabajo de Fin de Título consiste en la implementación de una aplicación multiplataforma para dispositivos móviles que permita la interacción directa del competidor en sus respectivas competiciones. Además, esta permitirá al usuario realizar un seguimiento completo de la competición, haciendo uso de notificaciones y recordatorios que involucrarán de forma directa a los participantes. El sistema no estará solo en manos del administrador, evitando así una gestión centralizada y favoreciendo la colaboración entre los participantes de cada competición.

En el presente documento, se hace referencia indistintamente a competiciones y torneos, y se asume que cada uno de estos está dividido en, al menos, una categoría.

Estructura del documento

El documento se encuentra dividido en los siguientes capítulos:

- Capítulo 1: Estado actual y objetivos

  Se analiza el estado del arte actual, es decir, las aplicaciones orientadas a la organización de competiciones que existen actualmente. También se mencionan los objetivos iniciales del proyecto y los añadidos durante el desarrollo de este.
- Capítulo 2: Competencias cubiertas

  Se detallan las competencias específicas del Grado en Ingeniería Informática cubiertas, mencionando también la justificación de las mismas.
- Capítulo 3: Aportaciones

  Se mencionan las ventajas del proyecto tanto en el entorno socioeconómico como en el personal.
- Capítulo 4: Normativa y legislación

  Se especifican las licencias software utilizadas a lo largo de la realización del proyecto y se analiza el impacto de las mismas. Además, se hace referencia a la normativa y legislación vigente que afecta a la aplicación.
- Capítulo 5: Metodología de trabajo y planificación

  Se presenta la metodología de trabajo escogida para llevar a cabo el proyecto. También se referencia la planificación inicialmente organizada y los ajustes que ha sufrido en el transcurso de esta.
- Capítulo 6: Tecnologías y herramientas

  Se listan las tecnologías y herramientas utilizadas en el desarrollo de la aplicación y se justifica su uso, explicando características de estas y su aportación al proyecto.
- Capítulo 7: Análisis

  Se analizan los actores que harán uso de la aplicación y los requisitos de la misma, haciendo uso del Lenguaje Unificado de Modelado para definirlos y representarlos gráficamente.
- Capítulo 8: Diseño

  Se especifica la arquitectura del sistema requerido para el funcionamiento de la aplicación y la interfaz gráfica de la misma, detallando sus características principales y la estructuración de estas.
- Capítulo 9: Desarrollo

  Se explican detalladamente los aspectos de desarrollo más importantes de la aplicación, haciendo hincapié en la estructura de los ficheros, la obtención de datos desde el servidor de Todotorneos.com y los módulos utilizados.

- Capítulo 10: Pruebas

  Se hace referencia a las pruebas realizadas en una competición real para asegurar el correcto funcionamiento de la aplicación y se contemplan los errores encontrados durante el transcurso de estas.
- Capítulo 11: Resultados, conclusiones y trabajo futuro

  Se evalúa el resultado del proyecto tras las fases de desarrollo y se extraen conclusiones acerca de este. Además, se mencionan algunas de las ampliaciones y mejoras potenciales para la aplicación.
- Bibliografía

  Se listan las distintas fuentes de información consultadas para la redacción del presente documento.
- Anexos

  Se incluye información complementaria del proyecto:
  - Anexo I: Se explica detalladamente cómo usar la aplicación y sus funcionalidades.
  - Anexo II: Se detallan los *endpoints* de Todotorneos.com utilizados para el intercambio de datos.
  - Anexo III: Se comentan los pasos a seguir para instalar la aplicación en un entorno de desarrollo.

1. Estado actual y objetivos

1.1 Estado actual

Actualmente, la preparación y organización de un torneo de aficionados de cualquier deporte no es tarea sencilla cuando el número de participantes es elevado. Normalmente, se recurre a sitios webs que ayudan a administrar el torneo, como todotorneos.com, o incluso se organiza “a mano”, lo cual es una tarea pesada y laboriosa. En cualquier caso, aun con el torneo perfectamente organizado, no es sencillo para los participantes hacer un seguimiento completo del mismo, así como acceder rápidamente a datos relevantes como las fechas de los próximos partidos o los resultados de otros equipos de la competición.

En la Play Store y la App Store, podemos encontrar varias aplicaciones para dispositivos móviles que ofrecen la capacidad de administración y/o seguimiento de torneos, entre las mejores posicionadas y más descargadas encontramos las siguientes:

- Leverade: A nivel de funcionalidad, permite la administración de torneos de diferentes modalidades, ofreciendo también la posibilidad de recibir notificaciones del torneo en cuestión. Sin embargo, las calificaciones son negativas debido a que ciertas funcionalidades son de pago y, como indican en los términos y condiciones de uso, se ofrecen distintos planes que obligan a los administradores de torneos a pagar los servicios. Además, no existe el usuario con rol de competidor y por lo tanto no recibe notificaciones personalizadas, ni puede conocer los torneos en los que participa, pues no hay ningún vínculo con el torneo a excepción de poder seguirlo.
- Competize: Se trata de una aplicación desactualizada (última actualización hace dos años) y que no permite la posibilidad de compartir un torneo y hacer un seguimiento de este, simplemente administrarlo manualmente. Es decir, está totalmente orientada al administrador y tampoco existe el rol de competidor o participante.
- Winner: La mejor puntuada en la App Store, no permite compartir y añadir personas al torneo. La interfaz es intuitiva, pero está orientada únicamente al administrador.
- MyTournament: De nuevo, es una aplicación orientada a administrar el torneo sin la posibilidad de unirse a uno existente o seguirlo.

Como conclusión, podemos destacar la falta de una aplicación móvil que permita realizar un seguimiento total de una competición aficionada en la que se participe, pues siempre

se prioriza la administración y organización del mismo frente a la posibilidad de mostrar datos actuales y hacer uso de notificaciones para mantener a los competidores al día, a excepción de la aplicación Leverade pero cuyos servicios no son gratuitos y el usuario no se vincula a su participación en el torneo. Al omitir estas funcionalidades, los torneos pueden presentar los siguientes inconvenientes:

- Desinformación de los otros resultados y participantes.
- Suspensión de partidos debido al olvido de estos.
- Toda la responsabilidad de la gestión del torneo recae sobre el administrador.
- Desinterés final de los participantes por los motivos anteriores.

1.2. Objetivos

El objetivo de este Trabajo de Fin de Título será el desarrollo de una aplicación móvil multiplataforma que permita el seguimiento de los torneos de las distintas competiciones de los competidores, en colaboración con el sitio web de TodoTorneos.com. Este portal web dispone de una amplia variedad de herramientas para la creación y administración de torneos, pero se encuentra orientada al administrador del mismo.

En la planificación inicial del trabajo, y como referencia el documento TFT01, se fijaron las siguientes funcionalidades como objetivos:

- Validación y registro de usuarios.
- Mostrar las competiciones en las que participa.
- Mostrar información de los torneos en los que participa: sus resultados, resultados del torneo y clasificación.
- Mostrar las fechas de los próximos partidos y programar recordatorios.
- Notificación de resultados.

En el transcurso de la realización del trabajo se han añadido las siguientes funcionalidades como nuevos objetivos:

- Proponer el resultado de un partido disputado por el usuario.
- Aceptar o rechazar propuestas de resultados en los partidos disputados.
- Recibir y aceptar o rechazar invitaciones a torneos.
- Inscribirse en torneos con inscripción abierta.
- Mostrar las distintas categorías de los torneos en los que participa el usuario, aunque no participe en estas.
- Mostrar los resultados de un equipo a lo largo de una categoría concreta.
- Editar perfil del usuario.

En resumen, la aplicación móvil se encuentra totalmente orientada al competidor, de forma que pueda hacer un seguimiento detallado y personalizado de cada torneo en el que participa mientras que el administrador gestiona la competición desde la web.

Ilustración 1. Logo de Todotorneos.com

2. Competencias cubiertas

2.1. TI04

“Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.”

En la planificación de este Trabajo de Fin de Título se ha hecho hincapié en la organización del mismo, para ello se hace uso de técnicas de desarrollo ágil como la creación previa de bocetos o *mockups* para las vistas, la programación de *sprints* o la confección de pilas de producto. Esta metodología se centra en el usuario y en las funcionalidades que se le aportan, lo que denominamos historias de usuario.

2.2. TI06

“Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.”

En este proyecto es fundamental el uso de tecnologías de red, pues el propio acceso a los datos es gestionado constantemente a través del protocolo HTTP por medio del servicio web de Todotorneos, por lo que la aplicación resultante es completamente dependiente de Internet y está basada en la red.

2.3. IS01

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

Además de desarrollar un sistema software orientado al usuario y que resulte fiable para este, se ha dado gran importancia al mantenimiento de la aplicación, al uso de estándares en el lenguaje oportuno y otras buenas prácticas de la ingeniería del software. Además, se han seguido los principios SOLID en la medida de lo posible, que son los siguientes:

- S – *Single Responsibility Principle* (Principio de responsabilidad única)
- O – *Open/Closed Principle* (Principio de abierto/cerrado)
- L – *Liskov Substitution Principle* (Principio de sustitución de Liskov)

- I – *Interface Segregation Principle* (Principio de segregación de la interfaz)
- D – *Dependency Inversion Principle* (Principio de inversión de dependencias)

El seguimiento de estos principios reside en el objetivo de desarrollar un código limpio, eficaz, reutilizable y mantenible, de forma que el software resultado sea robusto y escalable. Lo que es considerado software de calidad.

2.4. IS02

"Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones."

Aunque se ha dado libertad en el diseño de la aplicación, el cumplimiento de los requisitos acordados para cumplir las necesidades del sitio web todotorneos.com ha jugado un papel fundamental en el desarrollo de este proyecto. Sin disponer de un coste que condicione el producto, si ha sido importante considerar las limitaciones de tiempo o del sistema ya desarrollado y las posibilidades que ofrece.

3. Aportaciones

3.1. Entorno socioeconómico

Tras el estudio de la situación actual, en busca de aplicaciones con funcionalidades similares, podemos concluir que este proyecto presenta características innovadoras dentro del mundo del deporte, especialmente del no profesional. Los deportistas aficionados podrán hacer un seguimiento de sus propias competiciones, así como interactuar dinámicamente con el torneo para acceder a nuevas competiciones o guardar resultados, ahorrando trabajo al administrador de este.

El propio hecho de conocer la existencia de la aplicación y del sitio web, puede ser motivo suficiente para generar motivación en los practicantes de deporte aficionados, que ven novedoso el sistema y desean probarlo. Realmente, muchos deportistas no profesionales no encuentran la posibilidad de participar en competiciones con otros deportistas de su edad o nivel y administrar uno por ellos mismos representa una gran dificultad y suelen acabar en intentos fallidos. Es por ello, que la posibilidad de administrar y seguir un torneo de forma simple y eficiente es motivo suficiente para animar a muchos deportistas a organizar competiciones.

Como añadido, y teniendo en cuenta la situación actual, la capacidad de organizar y realizar un seguimiento de los torneos en los que se participa de forma completamente remota y dinámica es un punto a favor. Con la aplicación, se evitan congregaciones de personas a la espera de disputar sus partidos o para conocer el resultado de otros equipos, pues pueden visualizar en todo momento y rápidamente cuándo han de jugar y cuáles son los resultados de cualquier equipo de la categoría.

Por poner un ejemplo, en el Trofeo Rector de la ULPGC, los resultados se comunican hoy en día a través de Whatsapp u otros métodos de contacto, pero solo a los capitanes de cada equipo, convirtiéndolos en los encargados de difundirlo en su equipo. El uso de una aplicación como la desarrollada en este proyecto supondría un gran avance en este tipo de competiciones, pues los participantes tendrían acceso a la totalidad de los datos de un torneo sin la necesidad de preguntar cada vez que deseen hacerlo y eliminando los procesos de contacto entre capitanes y los administradores del torneo, en los cuales pueden producirse errores de comunicación y malentendidos.

3.2. Personal

A nivel personal, la realización de este proyecto me ha aportado multitud de conocimientos útiles para el futuro. A lo largo de la realización de este, me he formado no solo como ingeniero informático sino como persona también, mejorando aptitudes

como la perseverancia y la superación. He experimentado multitud de momentos de cansancio o de ignorancia afrontando un problema concreto, lo que me ha llevado a dar lo mejor de mí y esforzarme, aportándome después momentos de felicidad y satisfacción.

Como futuro profesional, he aprendido a buscar respuestas por mí mismo y superar los obstáculos que se me han presentado. He trabajado con tecnologías de última generación y que sin duda tienen un gran impacto en el entorno tecnológico actual, lo cual es extremadamente importante en la informática, que se encuentra en constante cambio y evolución.

4. Normativa y legislación

4.1. Licencias software

Las licencias de software son contratos entre el autor y el usuario final del propio software. En ellos, se especifican los términos y condiciones requeridos para su uso. Se basan en un conjunto de permisos que el desarrollador le otorga al usuario para distribuir, usar o modificar el producto bajo una licencia específica. También se suele especificar la duración de esta y el territorio en el que se aplica.

A continuación, se revisarán las licencias software asociadas a los productos utilizados en este proyecto.

4.1.1. GPL

La Licencia Pública General de GNU es una de las licencias de autor más utilizadas en software libre y código abierto. Cualquier software bajo esta licencia garantiza a los usuarios finales (personas, organizaciones y compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software.

A su vez, el software cubierto por la licencia GPL queda protegido mediante *copyleft* de intentos de apropiación que restrinjan las libertades a los nuevos usuarios cada vez que la obra es distribuida, modificada o ampliada. El software Git se encuentra bajo la GPL versión 2 [1, 2, 3].

4.1.2. MIT

La licencia MIT debe su nombre al Instituto Tecnológico de Massachutetts, donde se originó. Es una licencia permisiva, impone pocas limitaciones en la reutilización de software y posee una alta compatibilidad de licencias, siendo compatible con muchas licencias *copyleft* como la Licencia Pública General (software con licencia MIT puede integrarse en software con licencia GPL, pero no al contrario) [4, 5].

La licencia MIT es otra de las más importantes y utilizadas en el mundo del software libre. En el desarrollo de este proyecto, el software utilizado bajo esta licencia corresponde a los *frameworks* Angular e Ionic.

4.1.3. Licencia JetBrains

JetBrains ofrece licencias basadas en suscripciones para sus productos, estas pueden renovarse anual o mensualmente y adquirirse para uno, varios o todos los productos. Para el desarrollo de este proyecto se ha hecho uso de uno de sus productos, el IDE Webstorm. La licencia es otorgada de forma gratuita a los estudiantes y el personal docente en instituciones educativas acreditadas [6].

La licencia de estudiante cuenta con algunas limitaciones, como que solo debe usarse con fines educativos y no se puede usar para desarrollar productos o servicios de una organización [7].

4.2 Seguridad de los datos

La Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPDGDD) regula el tratamiento de los datos de carácter personal y se adapta al Reglamento General de Protección de Datos (RGPD). La responsabilidad de estos datos recae sobre Todotorneos.com, ya que es la web la que trata estos datos y los almacena en una base de datos.

Los datos personales tratados en la web son nombre, apellidos y email. El nivel de seguridad correspondiente a estos datos es el básico según lo establecido en la RGPD y Todotorneos.com aplica las medidas necesarias para garantizar el cumplimiento de la legislación vigente.

5. Metodología de trabajo y planificación

5.1 Metodología

En el desarrollo de este proyecto se ha tratado de simular una metodología de desarrollo ágil, en concreto SCRUM. Por supuesto, varios roles importantes en este ámbito como el *SCRUM Master* (maestro de SCRUM) y el *Product Owner* (propietario del producto) no tienen un rol claro en el mismo, sin embargo, se ejecutaron otros procesos de SCRUM como un seguimiento diario (*daily meeting*), la planificación de *sprints* y la creación de un tablón de Trello con las tareas propuestas.

Cada día de trabajo desde el inicio de la elaboración del trabajo, se ha escrito un diario en el que se detallaba el trabajo realizado durante el día y el planificado para el día siguiente, lo cual me permitía comenzar rápidamente con el trabajo diario o revisar el porqué de algunas decisiones tomadas anteriormente.

Uno de los objetivos de esta metodología es ofrecer tras cada iteración un prototipo funcional, o lo que se conoce como producto mínimo viable. Todos los *sprints* tuvieron una duración de dos semanas y, aproximadamente al término de cada uno, se realizaba una reunión con el tutor y el cliente para presentar las novedades y recibir sugerencias de mejora. En función de esta reunión y las sugerencias aportadas en ella se actualizaba y reordenaba el siguiente sprint.

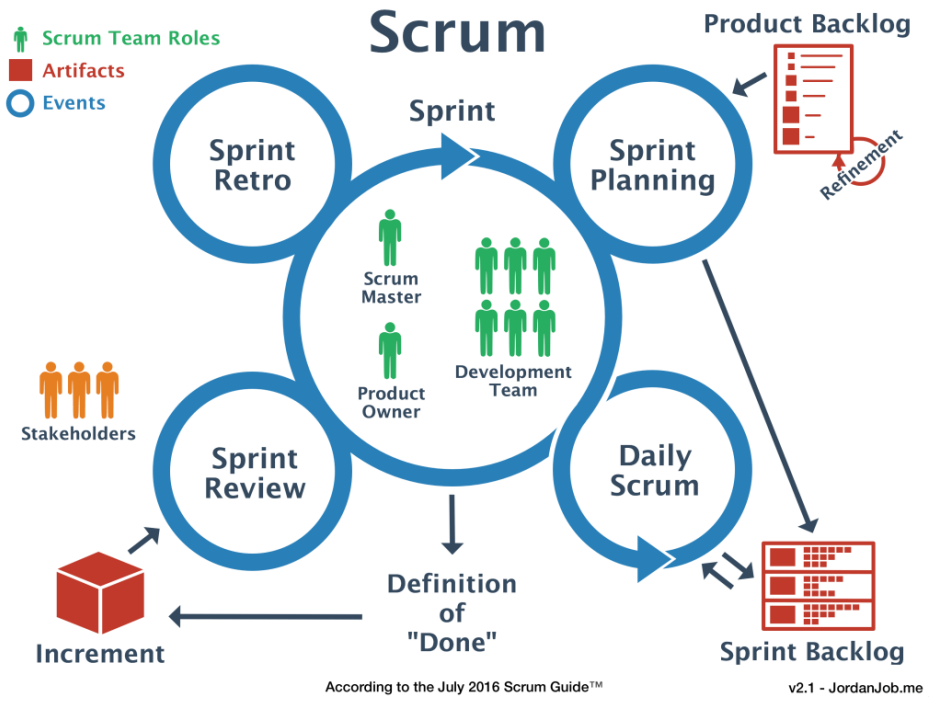

Ilustración 2. Elementos de la metodología Scrum

5.2 Planificación inicial

El plan de desarrollo propuesto en la tabla 1 es el correspondiente a la planificación del TFT01:

Fase	*Duración Estimada (horas)*	*Tareas*
Estudio previo / Análisis	55	Tarea 1.1: Estudio del sitio web a adaptar
		Tarea 1.2: Obtención de requisitos para la aplicación
		Tarea 1.3: Elaboración de plan de desarrollo en “Sprints”.
Diseño / Desarrollo / Implementación	150	Tarea 2.1: Diseño de las vistas o “mockups”
		Tarea 2.2: Aprendizaje de Angular e Ionic
		Tarea 2.3: Desarrollo de la aplicación
Evaluación / Validación / Prueba	50	Tarea 3.1: Pruebas de funcionamiento
		Tarea 3.2: Reuniones de testeo
Documentación / Presentación	45	Tarea 4.1: Realización de la memoria
		Tarea 4.2: Preparación de la presentación

Tabla 1. Planificación inicial

En la planificación inicial, se confeccionó una pila de producto, en la que se detallaban las historias de usuario referentes a la aplicación, es decir, una lista de las funcionalidades a implementar. Estas, como es habitual en SCRUM, se dividieron en distintos *sprints* teniendo en cuenta el tiempo aproximado de cada una de estas funcionalidades y las tareas necesarias para su cumplimiento. Se confeccionaron 6 pilas de *sprint* como fruto de esta división, lo que aproximaba a 10 semanas la finalización del producto deseado. A continuación, se detallan las tareas principales de cada *sprint* y el prototipo resultante de este:

- Sprint 1:
  - Tareas principales: Implementación de las vistas en HTML y CSS y de los servicios de iniciar sesión, registro y obtener torneos.
  - Novedades del prototipo: Capacidad para iniciar sesión y mantenerla iniciada, cerrar sesión, registrarse y listar los torneos en los que participa el usuario.
- Sprint 2:
  - Tareas principales: Profundizar en la mejora de la estética de la interfaz en la lista de torneos, ocultar la barra de navegación al deslizar hacia abajo en una categoría e implementar un menú.

- Novedades del prototipo: Sin nuevas funcionalidades para el usuario, pero se adaptó el estilo a las sugerencias del cliente y se mejoró la accesibilidad y el manejo de la aplicación.

- Sprint 3:
  - Tareas principales: Configurar *Capacitor* para dispositivos móviles, usar IonicStorage para almacenar la información de usuario localmente y configurar las notificaciones *push.*
  - Novedades del prototipo: Aplicación preparada para su uso en dispositivos Android e IOS y con capacidad de recibir notificaciones *push* desde un servidor Firebase.
- Sprint 4:
  - Tareas principales: Obtener los resultados de una categoría, listar las categorías de un torneo y realizar pruebas básicas con el *framework* Jasmine.
  - Novedades del prototipo: Capacidad para mostrar las categorías de cada torneo y acceder a los resultados a lo largo de la temporada de las categorías en las que participa el usuario.
- Sprint 5:
  - Tareas principales: Obtener la clasificación de una categoría, obtener los resultados del usuario en una categoría, añadir notificaciones locales, permitir el cambio de orientación de pantalla para visualizar más datos y añadir una vista de notificaciones en cada categoría.
  - Novedades del prototipo: Aplicación con capacidad para mostrar la clasificación en cualquier categoría de los torneos en los que participa el usuario y los resultados y notificaciones de las categorías en las que participa, así como posibilidad para programar notificaciones locales a la hora del partido o con antelación.

Cada *sprint* constaba de 30 horas de trabajo, por lo que la planificación en 5 *sprints* se ajustaba aproximadamente a lo propuesto en el documento TFT01, en el que se estimaba el desarrollo de la aplicación en 150 horas.

5.3 Ajustes en la planificación

En el transcurso del proyecto se produjeron varios ajustes en la planificación del mismo, ya que existía una dependencia con el servicio web de Todotorneos.com para la obtención de datos reales y estructura de estos. Estos ajustes no supusieron en ningún momento la detención del *sprint*, sino que se reordenaron e intercambiaron las tareas con otros *sprints* o se desarrollaron funcionalidades no planificadas al inicio del proyecto, añadiendo dos nuevos *sprints* al final.

La introducción de nuevas funcionalidades se debe principalmente a la motivación de añadir más posibilidades al usuario y elaborar una aplicación más completa e independiente de la web para los participantes. Además, la experiencia adquirida con las tecnologías utilizadas permitió desarrollar las nuevas características de forma ágil y eficiente y sin influir negativamente en las inicialmente planificadas.

Debido al gran número de funcionalidades propuestas durante el desarrollo de la aplicación y teniendo en cuenta el tiempo disponible, se programaron dos nuevos *sprint*s que terminarían añadiendo las nuevas historias de usuario. Los *sprints* fuera de la planificación inicial, que duraron también 2 semanas, tuvo las siguientes características:

- Sprint 6:
    - Tareas principales: Creación de ventanas modales para proponer un resultado y mostrar los resultados de un equipo, obtener invitaciones a torneos, refactorizar las vistas de resultados para lograr mayor compactación, crear alertas de error en las peticiones, añadir categorías de tipo eliminatoria en la aplicación.
    - Novedades del prototipo: Capacidad para proponer resultados de los partidos en los que participa el usuario, aceptar o rechazar invitaciones a torneos, mostrar los resultados de un equipo en una categoría y hacer uso de las características anteriores con categorías de tipo eliminatoria.
- Sprint 7:
    - Tareas principales: Diseño e implementación de sistema de inscripciones a torneos abiertos, añadir funcionalidades de actualización del perfil de usuario y restablecer la contraseña.
    - Novedades del prototipo: Capacidad para inscribirse en torneos con inscripción abierta, tanto con un equipo nuevo como con uno existente. Posibilidad de restablecer la contraseña en caso de olvido y de modificar nombre, apellidos y contraseña del usuario.

De esta forma, la duración de la implementación de la aplicación superó las 210 horas debido a los dos *sprints* añadidos.

6. Tecnologías y herramientas

6.1 Tecnologías

Para llevar a cabo este proyecto se realizó un análisis previo de las tecnologías más adecuadas para su implantación, ya que hoy en día el desarrollo de aplicaciones móviles ofrece un amplio abanico de posibilidades, cada una con sus ventajas y desventajas. Finalmente, en el transcurso de este TFT se han utilizado las siguientes tecnologías:

Ionic 5: *Framework* gratuito y de código abierto, especialmente orientado al desarrollo de aplicaciones híbridas multiplataforma basado en las populares tecnologías web (HTML, CSS y JavaScript). Este *framework* permite la creación de aplicaciones móviles complejas, escalables y con buen rendimiento. Incluye una amplia variedad de componentes y *plugins* para hacer uso de funciones nativas gracias a Capacitor o Cordova. Aunque Ionic está relacionado estrechamente con Angular, puede usarse también con Vue, React o JavaScript exclusivamente.

Angular 9: *Framework* desarrollado en TypeScript para aplicaciones web, es de código abierto y mantenido por Google. Es uno de los *frameworks* web más populares y el que mejor integra Ionic, su estructura basada en componentes y módulos permite la creación de potentes aplicaciones de una página (*Single Page Application*) [8] que aportan fluidez a la aplicación [9].

Capacitor 2.0: Proyecto capaz de convertir aplicaciones web en nativas, tanto para Android como para iOS. Fue desarrollado por la misma compañía que Ionic e ideado para usarse junto a este, acompañando así a Cordova como únicas opciones para la conversión de aplicaciones web a nativas. Además, permite integrar los *plugins* ya diseñados por Cordova y actualiza muchos otros haciendo uso de las APIs más modernas [10].

Git: Software de control de versiones de código abierto, diseñado para gestionar proyectos, especialmente en casos en los que existan numerosos archivos de código fuente. En este caso, ha sido fundamental contar con el control de versiones, para mantener un seguimiento de las modificaciones e incluso restaurar las anteriores, aunque no se ha requerido el uso de más de una rama.

Ilustración 3. Logos de Angular e Ionic

6.2 Herramientas

Webstorm 2020.1: Entorno de desarrollo integrado (IDE) desarrollado por JetBrains especialmente para JavaScript, incorpora finalización de código inteligente, detección de errores y refactorización entre otras características para este lenguaje. Es comúnmente utilizado para proyectos de NodeJS, React, React Native, Ionic o Angular [11]. La elección de este IDE para la implementación de la aplicación con Angular e Ionic se debe, además de por las funcionalidades comentadas anteriormente,a la navegación intuitiva y la integración con proyectos de Git que presenta el entorno de desarrollo.

Postman: Herramienta utilizada generalmente para probar y consumir API Rest, esta permite escribir pruebas automatizadas, guardar las peticiones y otras funciones adicionales para el intercambio de datos vía HTTP. En el proyecto actual, ha resultado sumamente importante a la hora de probar los distintos *endpoints* que proporciona el servidor de Todotorneos rápidamente, mostrando las respuestas de estos en una interfaz intuitiva y amigable.

AndroidStudio: IDE oficial de la plataforma Android desarrollado por JetBrains y basado en el software de Intellij [12]. Esta herramienta es necesaria para la integración y publicación de cualquier aplicación en un dispositivo móvil Android, ya que permite generar el fichero APK (Android Aplication Package o Paquete de Aplicación Android) y así instalarlo en el dispositivo. Es también de utilidad para realizar pruebas o depurar el funcionamiento de la aplicación en nativo, gracias a la funcionalidad integrada para simular dispositivos Android o para instalarla rápidamente en un dispositivo físico a través de una conexión USB.

XCode: IDE diseñado por Apple para el desarrollo de software en macOS, iOS, watchOS y tvOS, disponible únicamente para el sistema operativo macOS [13]. Es necesario para integrar y publicar software en dispositivos iOS.

7. Análisis

7.1. Actores

El actor principal de esta aplicación corresponde al participante de un torneo o competidor. A continuación, se detallan las distintas funcionalidades de este actor:

- Visualizar los torneos en los que participa, así como un histórico que muestre los que ya han terminado.
- Visualizar resultados y clasificación de cualquier categoría en los torneos en los que participa.
- Visualizar los resultados de su equipo en una categoría en la que participe.
- Visualizar los resultados de cualquier equipo en cualquier categoría de los torneos en los que participe.
- Guardar el resultado de un partido en el que participe.
- Aceptar o rechazar resultados propuestos por el equipo rival en partidos en los que participe.
- Inscribirse a torneos abiertos.
- Aceptar o rechazar invitaciones a torneos.
- Visualizar y editar datos de su perfil.

7.2. Requisitos

Para definir los requisitos se utiliza el Lenguaje Unificado de Modelado (UML) [14], que ofrece un estándar para describir un modelo. Los casos de uso son conceptos clave en UML y ayudan a definir procesos, actores y la interacción entre ellos. Para describir los requisitos del proyecto se usa un diagrama de casos de uso que sirve como notación gráfica de los casos de uso que se dan en la aplicación.

Los casos de uso fueron mayormente extraídos de los objetivos del proyecto, pero también se añadieron progresivamente nuevos casos de uso fruto de las reuniones mantenidas con el gestor de Todotorneos.com.

7.2.1 Diagrama de casos de uso

A continuación, en la ilustración 4 se muestra el diagrama de casos de uso resultante de los casos de uso de un usuario registrado, único actor del modelo y cuyas funcionalidades están dirigidas a los participantes de competiciones.

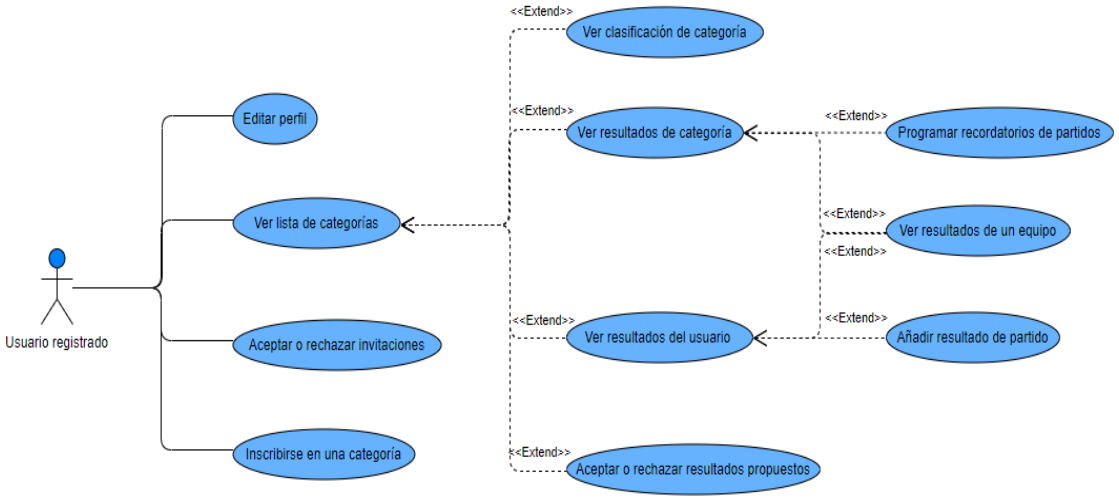

Ilustración 4. Diagrama de casos de uso

7.2.2 Especificaciones de casos de uso

La especificación de los casos de uso se refiere a la descripción de cada una de las partes definidas para lograr su descripción completa. Para ello, es necesario conocer el sistema y cómo se comporta en la interacción con el usuario.

Finalmente, se explicarán detalladamente los casos de uso más complejos de la aplicación:

Caso de uso	**9**	**Añadir resultado de partido**	
Descripción		El usuario añade el resultado de un partido en el que ha participado y cuyo resultado no está definido	
Actores		Usuario	
Precondiciones		El resultado del partido es nulo.	
Flujo normal		Paso	Acción
		1	Se accede a la vista “Mis resultados” de una categoría
		2	Se pulsa en “Añadir resultado”
		3	Se establece el resultado y la información deseada
		4	Se envía el resultado
Variantes			
Extensiones			
Postcondiciones		El resultado del partido es actualizado al propuesto por el usuario	
Excepciones		4a	No se ha rellenado alguno de los campos obligatorios
		4b	El resultado ya ha sido añadido previamente
Observaciones			

Tabla 2. Especificación de casos de uso: Añadir resultado de partido

Caso de uso	10	Programar recordatorio de partido		
Descripción		El usuario programa un recordatorio en forma de notificación para el partido elegido		
Actores		Usuario		
Precondiciones		La fecha del partido está establecida y es posterior a la actual, el partido no tiene recordatorios programados		
Flujo normal		Paso	Acción	
		1	Se accede a la vista “resultados”	
		2	Se pulsa en el icono de campana	
		3	Se escoge la antelación deseada	
		4	Se acepta la configuración establecida	
Variantes				
Extensiones				
Postcondiciones		El dispositivo programa una alerta con la antelación especificada y la campana del partido se vuelve amarilla		
Excepciones				
Observaciones				

Tabla 3. Especificación de casos de uso: Programar recordatorio de partido

Caso de uso 4	Inscribirse en una categoría		
Descripción	El usuario se inscribe en una de las categorías con inscripción abierta		
Actores	Usuario		
Precondiciones	El usuario no está inscrito aún en la categoría escogida		
Flujo normal	Paso	Acción	
	1	Se accede a la opción del menú “Inscribirse”	
	2	Se elige una de las categorías mostradas	
	3	Se pulsa sobre un equipo de entre los mostrados	
	4	Se confirma la inscripción	
Variantes	3a	Inscribirse con un nuevo equipo	
		3a.1	Se pulsa en la opción “Nuevo equipo”
		3a.2	Se escribe el nombre del nuevo equipo
		3a.3	Se envía el formulario
		3a.4	Ir al paso 4
Extensiones			
Postcondiciones	La ventana modal de inscripción se cierra y se actualiza la vista “Mis torneos”, que incluye la nueva categoría		
Excepciones			
Observaciones			

Tabla 4. Especificación de casos de uso: Inscribirse en una categoría

Caso de uso 8	Aceptar o rechazar resultados		
Descripción	El usuario responde a un resultado propuesto por el contrincante, rechazándolo o aceptándolo		
Actores	Usuario		
Precondiciones	Se ha recibido una notificación de resultado añadido		
Flujo normal	Paso	Acción	
	1	Se accede a la vista “Notificaciones” de una categoría	
	2	Se pulsa el botón “Validar” en la notificación de resultado deseada	
	3	Se acepta o rechaza el resultado mostrado	
	4	Se envía la respuesta	
Variantes			
Extensiones	3a	Añadir justificación	El usuario añade un mensaje de justificación cuando rechaza el resultado
Postcondiciones	Se actualiza la notificación de resultado con la respuesta dada y si se rechaza se actualizan también las vistas de “Resultados”, “Mis resultados” y “Clasificación”		
Excepciones	4a	No se ha seleccionado ninguna respuesta	
	4b	El resultado ya ha sido aceptado o rechazado previamente	
Observaciones			

Tabla 5. Especificación de casos de uso: Aceptar o rechazar resultados

8. Diseño

8.1. Arquitectura del sistema

Para la comunicación con el servidor de Todotorneos.com y, por lo tanto, la interacción con su base de datos, se hace uso de la *API Rest* facilitada. El intercambio de datos se realiza utilizando el protocolo HTTP y sus métodos GET y POST, ya que no se usan los comúnmente utilizados PUT/PATCH y DELETE. Como cualquier REST, el protocolo cliente/servidor no tiene estado, es decir, cada petición HTTP contiene la información necesaria para el servidor, de forma que peticiones idénticas generan respuestas idénticas.

A día de hoy, la API Rest gestiona el token de usuario para mantener la sesión iniciada y acceder a peticiones restringidas para el usuario. El token generado en el servidor se facilita en la respuesta al iniciar sesión y se envía como autorización en las peticiones necesarias, bien por medio de los parámetros en GET o del cuerpo de la petición en POST.

8.2. Interfaz gráfica

8.2.1 Características principales

En el diseño de la interfaz gráfica, se tomó como referencia el sitio web de Todotorneos.com en cuestión de estilos y colores, manteniendo así la temática de la web y permitiendo que los usuarios identifiquen fácilmente la relación entre la aplicación móvil y web.

En primer lugar, se diseñaron los bocetos haciendo uso de la herramienta Figma, estos sirvieron de referencia a la hora de implementar el código HTML y CSS en el proyecto. De esta forma, se presentaron también los diseños desarrollados al *cliente*, pues era menester asegurar que se cumpliese con los requisitos deseados. Se muestra una comparativa entre la web y la aplicación móvil en la ilustración 5.

En el diseño de la interfaz gráfica ha sido prioritario el uso de componentes Ionic, pues la mayoría de estos son convertidos en componentes nativos para iOS o Android y, por lo tanto, los usuarios de estos sistemas operativos se encuentran familiarizados con ellos.

Otra característica relevante es la sencillez y la facilidad de uso, es decir, evitar que una misma vista implemente múltiples funcionalidades y confunda al usuario. Debido a esto, cada funcionalidad implementada en la aplicación se encuentra implementada en una vista asociada, siguiendo así el estilo tradicional de las aplicaciones móviles.

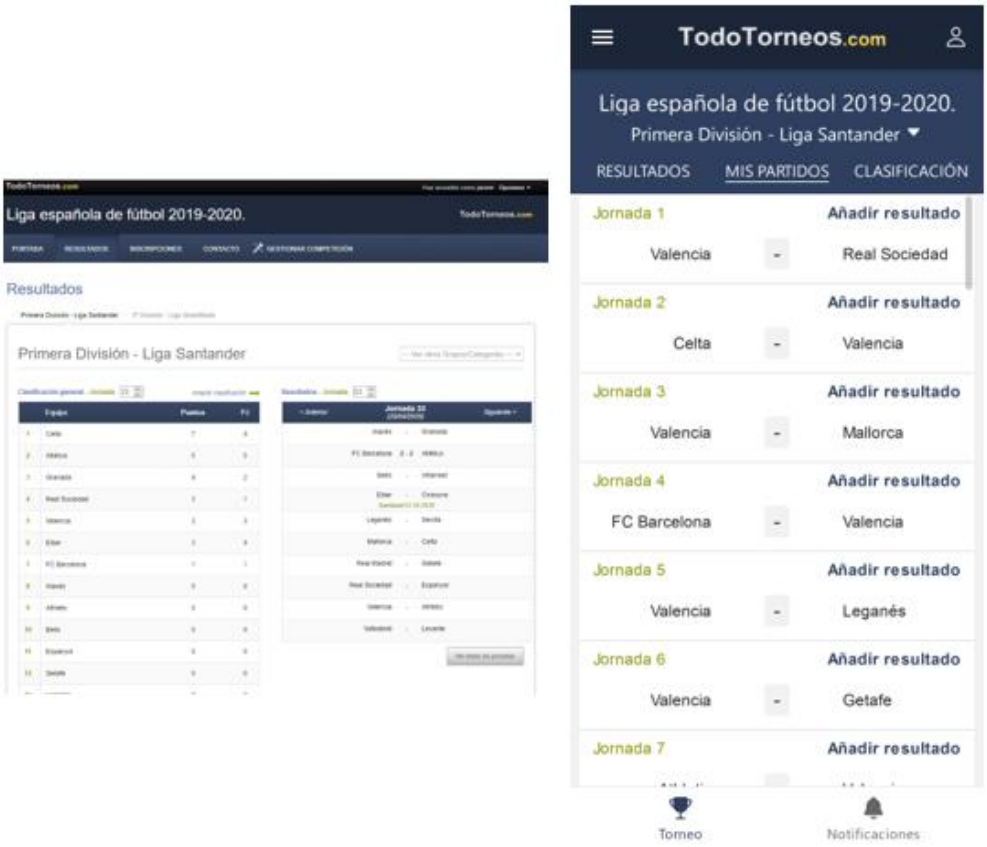

Ilustración 5. Comparativa web vs aplicación

8.2.2 Jerarquía de las vistas

En este proyecto, y como incluye por defecto Ionic 5, se utiliza *lazy loading* [15], esto se resume en cargar únicamente los componentes necesarios en cada momento, y descartar los que no. La alternativa a esto es mantener en el Modelo de Objetos del Documento (DOM) todos los componentes que se van cargando en el flujo de la aplicación, lo cual es una evidente desventaja en cuestiones de rendimiento.

Para favorecer la técnica de *lazy loading* es importante estructurar correctamente los módulos de la aplicación. Cada módulo contiene uno o más componentes que se cargan en DOM cuando son importados, de forma que se descartan los que no se están utilizando actualmente. La aplicación consta de un total de 7 modulos, que son los siguientes:

- App: Módulo principal de la aplicación, encargado de cargar los otros módulos cuando se requieran y de incluir el menú de navegación.
- Login: Contiene el inicio de sesión y la vista de “Restablecer contraseña”.
- Registro: Contiene el registro de usuario.
- Perfil: Muestra los datos del perfil y permite modificarlos.
- Inscribirse: Controla las inscripciones a torneos.

- Torneos: Muestra la lista de torneos del usuario actual y permite acceder a cada uno de ellos.
- Torneo: Contiene información de la categoría y las notificaciones de la misma.

La estructuración de los módulos cargados por el módulo “App” se describe en la ilustración 6, el color verde corresponde a los módulos y el gris a los componentes:

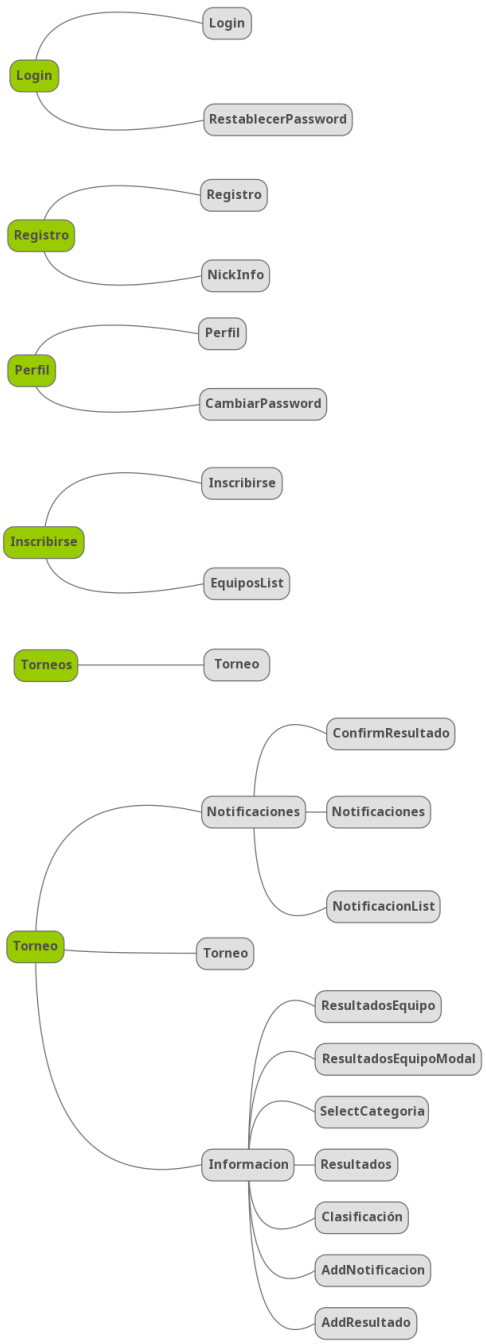

Ilustración 6. Jerarquía de módulos y componentes

9. Desarrollo

En el desarrollo de este proyecto los componentes esenciales han sido Ionic y Angular, pues haciendo uso de estas tecnologías se ha implementado la totalidad de la aplicación. Sin embargo, se ha de tener en cuenta la implicación del servidor de Todotorneos.com y que actúa como API Rest para el intercambio de datos de la web. Esta ha sido diseñada en conjunto con el cliente, pero implementada por él, de esta forma la colaboración y el intercambio de impresiones ha sido fundamental para su elaboración.

9.1. Estructura

Los proyectos de Ionic en general siguen una estructura muy similar a la que presenta Angular. Aunque no hay establecida ninguna estructura definida o estándar oficial, la gran mayoría de aplicaciones de Ionic con Angular se organizan tal y como indica el proyecto de Github titulado “ionic-angular-schematics” [16] en un intento de crear una estructuración de proyectos estándar basados en las mejores prácticas usadas por la comunidad, esta estructura se muestra en la ilustración 7. Por supuesto, la estructura de los proyectos es totalmente flexible y está abierta a cambios en todo momento.

Dentro de cualquier proyecto de Angular encontramos los siguientes directorios:

- `e2e`: Este directorio denominado *End To End* hace referencia a los tests llevados a cabo en el proyecto que se incluyen automáticamente en la creación de un proyecto haciendo uso de la interfaz de línea de comando (CLI) de Angular. Estas pruebas no son unitarias, sino que aseguran el funcionamiento conjunto de los componentes de la aplicación.
- `src`: Es la carpeta destinada al código que identifica la aplicación y la hace única, es decir, todo aquel código que no genera la interfaz de línea de comando de Angular.
- `node_modules`: Como en toda aplicación de NodeJS, esta carpeta incluye todos los módulos indicados en el archivo */package.json* y que son necesarios para el funcionamiento de la misma.
- `www`: Este directorio incluye el contenido de la aplicación en HTML, JavaScript y CSS, es generado haciendo uso del empaquetador Webpack, que puede ser configurado desde el archivo */angular.json*. Esta carpeta es la requerida para usar la aplicación en modo de producción.

A continuación, se detallará la estructura del directorio */src/*, que incluye el contenido más relevante del proyecto.

- `app`: Ubicación de los distintos módulos y componentes de la aplicación. En él se encuentran el módulo *app* y su componente, correspondientes al módulo principal de la aplicación. Este componente es el encargado de enrutar las diferentes páginas implementando *lazy loading*. Además, aquí se ubican también los siguientes directorios:
    - `auth`: Incluye la configuración de los *guards* de Angular, que determinan si un módulo puede ser cargado o no dependiendo de la autorización del usuario.
    - `pages`: Contiene el resto de los módulos y componentes que corresponden a las distintas páginas de la aplicación y que son cargados por `/src/app/app-module.ts`.
    - `shared`: Incluye un módulo que exporta los componentes compartidos en distintos módulos de la aplicación.
    - `services`: Compuesto de los distintos servicios que usan los componentes de la aplicación, estos son clases de Typescript accesibles desde cualquier componente de la aplicación y utilizados para obtener o intercambiar datos entre estos.
    - `directives`: Se incluyen aquí los componentes marcados con el decorador "directive" de Angular, que son clases incluidas en componentes para modificar elementos de los mismos.
    - `helpers`: Este directorio contiene distintas funciones utilizadas varias veces en la aplicación y que sirven como ayuda para evitar repeticiones de código.
- `assets`: Destinado a las imágenes y otros archivos de interés que deben cargarse junto a la aplicación.
- `environment`: Contiene opciones de configuración para las variables de entorno del proyecto. Por defecto incluye un entorno de producción y otro de desarrollo.
- `theme`: Incluye el fichero `variables.scss`, en el que Ionic define las variables de las que hace uso en sus componentes. Se pueden configurar a gusto del desarrollador para modificar los colores principales de la aplicación.

Ilustración 7. Estructura de un proyecto Ionic

9.2. Services

Como indica la documentación oficial de Angular [17], un servicio implementa la lógica que no está asociada a un componente específico y que se comparte entre ellos. En el proyecto actual, los servicios son utilizados mayormente para acceder a los datos facilitados por la API Rest de Todotorneos.com. Entre los servicios encontramos:

- error: Este servicio es utilizado para generar una alerta en caso de error con el mensaje devuelto como respuesta a una petición vía HTTP. El servicio es utilizado por los componentes que hacen uso de peticiones de este tipo. De esta forma, se evita código duplicado y se controlan de forma global las alertas generadas por errores en la aplicación.
- refresh: El servicio *refresh* está destinado a refrescar el contenido de los componentes que se requieran, pudiendo desde este solicitar una recarga del contenido obtenido desde la API Rest.
- notificacion: Este servicio crea y gestiona las respuestas de las solicitudes HTTP referentes a la clase “Notificación”, que se encuentra declarada como interfaz.
- user: Crea y gestiona las respuestas de las solicitudes HTTP referentes a la clase “User”, además de otras funciones como el almacenamiento del *token* en el dispositivo o la obtención de una variable de tipo observable del usuario. Este tipo de variables permiten suscripciones, por lo que son útiles para reaccionar a los cambios en estas, en este caso para gestionar el cambio de usuario.
- torneo: Crea y gestiona las respuestas de las solicitudes HTTP referentes a la clase “Torneo”, como obtener datos del mismo, responder a invitaciones o fijar un resultado.

9.3 Endpoints

Para la interacción con el servidor de Todotorneos.com se hace uso de la API Rest facilitada, esta dispone de distintos *endpoints* para ejecutar las funciones requeridas en el servidor haciendo uso de peticiones HTTP. Estos *endpoints* son fruto de la colaboración con Todotorneos.com y se han elaborado conjuntamente a lo largo del desarrollo del proyecto, de modo que el intercambio de datos se produce en condiciones óptimas para ambas partes y en el formato acordado.

Ilustración 8. Diagrama de interacción cliente-servidor

La API Rest siempre devuelve datos de tipo JSON y los recibe en "form-data" en las peticiones POST. Las funciones proporcionadas por el servidor y accesibles desde el protocolo HTTP son las siguientes:

- Inicio de sesión: Recibe los campos de correo electrónico y contraseña introducidos y responde con los distintos campos de usuario correspondientes al correo y contraseña. En caso de no coincidir con ningún usuario devuelve el error correspondiente.
- Registro: Recibe los campos introducidos correspondientes al registro de usuario y responde con código de estatus 200 o los errores correspondientes, por ejemplo, si existe un usuario con el email o nombre de cuenta introducidos.
- Obtener lista de torneos: Recibe el usuario y su *token* correspondiente, si coincide responde con la lista de torneos en los que se está inscrito el usuario. En caso contrario, responde con error.
- Obtener categoría: Recibe el email del usuario, su *token* y el identificador de la categoría deseada, devuelve detalles de esta como su nombre o el torneo al que pertenece.
- Obtener resultados (liga): Recibe el identificador de la categoría y, opcionalmente, la jornada deseada. Si no se indica jornada responde con los resultados de la jornada activa, en caso contrario responde con los resultados de la jornada indicada.
- Obtener resultados (eliminatoria): Recibe el identificador de la categoría y, opcionalmente, la fase deseada. Si no se indica jornada responde con los resultados de la fase activa, en caso contrario responde con los resultados de la jornada indicada.
- Obtener resultados de un equipo (liga): Recibe el identificador de la categoría y el del equipo deseado. Responde con los resultados del equipo indicado.

- Obtener resultados de un equipo (eliminatoria): Recibe el identificador de la categoría y el del equipo deseado. Responde con los resultados del equipo indicado.
- Obtener clasificación: Recibe el identificador de la categoría y responde con la clasificación de la misma.
- Obtener lista de categorías de un torneo: Recibe el identificador de un torneo y responde con las categorías de este.
- Obtener notificaciones de una categoría: Recibe el email del usuario, su token y el identificador de la categoría. Responde con las notificaciones del usuario en la categoría indicada.
- Leer notificaciones: Recibe los identificadores de las notificaciones leídas y responde con código de estatus 200 o los errores correspondientes.
- Borrar notificaciones: Recibe el identificador de la notificación leída y responde con código de estatus 200 o los errores correspondientes.
- Grabar resultado (liga): Recibe los siguientes campos:
    - Email del usuario
    - *Token* del usuario
    - Identificador de la categoría
    - Identificador del partido
    - Número de jornada
    - Identificador del equipo 1
    - Identificador del equipo 2
    - Resultado 1 del equipo 1
    - Resultado 1 del equipo 2

  Si el partido es de tipo sets se incluyen también los siguientes parámetros:
        - Resultados 2,3,4 y 5 del equipo 1
        - Resultados 2,3,4 y 5 del equipo 1
        - Equipo ganador (1, 2 o 11 si hay empate y sólo en los partidos tipo “sets”)

  Responde con código de estatus 200 o los errores correspondientes.
- Grabar resultado (eliminatoria): Recibe los siguientes parámetros
    - Email del usuario
    - *Token* del usuario
    - Identificador de la categoría
    - Identificador del partido
    - Identificador del equipo 1

    - Identificador del equipo 2
    - Resultados 1,2,3,4,5 del equipo 1
    - Resultados 1,2,3,4,5 del equipo 2
    - Texto de datos adicionales 1
    - Texto de datos adicionales 2

  Responde con código de estatus 200 o los errores correspondientes.
- Validar resultado: Recibe los siguientes parámetros:
    - Email del usuario
    - *Token* del usuario
    - Identificador del torneo
    - Identificador del partido
    - Validación (“OK” o “NOK”)
    - Texto en caso de rechazo del resultado
- Obtener invitaciones: Recibe el email y el token del usuario y responde con las invitaciones pendientes del usuario.
- Responder invitación: Recibe el email, el token del usuario, el identificador de la invitación y la respuesta (“OK” o “NOK”). Responde con código de estatus 200 o los errores correspondientes.
- Obtener torneos con inscripción abierta: Recibe el email y token del usuario y responde con la lista de torneos cuya inscripción es abierta.
- Obtener equipos categoría: Recibe el email y token del usuario junto con el identificador de una categoría. Responde con la lista de equipos de la categoría y los usuarios inscritos en cada uno. Además, devuelve un parámetro que indica si el usuario puede inscribirse o no en la categoría.
- Inscribir jugador: Recibe email y token del usuario junto con la categoría, como último parámetro puede recibir un identificador del equipo, en caso de inscribirse en un equipo ya existente, o el nombre del nuevo equipo en caso de inscribirse con este. Responde con código de estatus 200 o los errores correspondientes.
- Editar perfil: Recibe email y token del usuario junto con el nombre y apellidos actualizados, responde con código de estatus 200 o los errores correspondientes.
- Editar contraseña: Recibe email y token del usuario junto con la contraseña antigua y la actualizada, responde con código de estatus 200 o los errores correspondientes.
- Restablecer contraseña: Recibe un email y responde siempre con un código de estatus 200.

9.4. Notificaciones Push

Las notificaciones de tipo *push*, al contrario que lo que se conoce como notificaciones locales, son generadas en el servidor, que “empuja” la notificación a los dispositivos deseados. Existen distintas herramientas para registrar dispositivos y gestionar las notificaciones, en este caso se ha hecho uso de la plataforma Firebase y su servicio de mensajería en la nube.

La aplicación para iOS y para Android se registran por separado en Firebase, ya que cada una de ellas posee un fichero g*oogle-services.json* distinto, que ha de ser añadido a la aplicación tras la actuación de Capacitor. Esta tecnología incluye la gestión de notificaciones *push* y no es necesario añadir *plugins* (complementos) adicionales, dando la posibilidad de gestionar los permisos de estas, los eventos que emiten: registro exitoso, registro erróneo, recibo de notificación con la aplicación abierta y pulsación de la notificación.

Cuando el usuario inicia la aplicación y acepta el permiso para recibir notificaciones (solo en iOS), se registra el dispositivo en Firebase con un identificador único. Desde este punto, el dispositivo es capaz de recibir notificaciones. Para mantener un registro de los dispositivos registrados y poder enviar notificaciones individuales, la aplicación envía al servidor el identificador otorgado por Firebase junto con el sistema operativo utilizado, de forma que en la base de datos de Todotorneos.com cada usuario contará con uno o varios identificadores correspondientes a los dispositivos en los que ha iniciado sesión y la plataforma en la que corre la aplicación, Android o iOS.

Firebase permite generar las notificaciones utilizando el protocolo HTTP, de esta forma el servidor de Todotorneos.com puede gestionar la emisión de notificaciones masivas al conjunto de usuarios deseado. En estas peticiones se pueden configurar detalladamente las notificaciones que se generan, estableciendo campos básicos como el título, el cuerpo y los datos que incluyen u otros más específicos como el sonido o el icono de la misma. Las notificaciones enviadas a Firebase desde el servidor tienen la estructura de la ilustración 9:

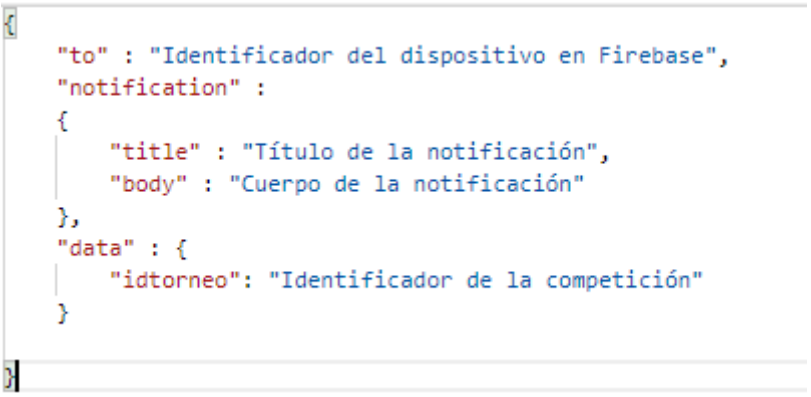

Ilustración 9. Estructura de notificación push en Firebase

Además, se debe añadir la cabecera HTTP *authorization* junto con la clave API de Firebase. La sintaxis completa se encuentra detallada en la documentación oficial [18].

9.5 Plugins

9.5.1. Notificaciones locales

Las notificaciones locales, a diferencia de las de tipo *push*, son generadas en el cliente y almacenadas en el dispositivo cuando se programan. Son utilizadas en la aplicación para programar las alertas de los partidos de interés para el usuario, pudiendo indicar además la antelación con la que se debe generar la alerta.

Para configurar las notificaciones locales se ha hecho uso del complemento de Cordova *local notification.* La instalación del módulo correspondiente permite gestionar las notificaciones y sus eventos de forma similar a las de tipo *push,* pudiendo, por ejemplo, dirigir al usuario a la vista de la categoría relacionada con la notificación cuando se pulsa en ella.

Este complemento requiere de una migración a AndroidX en su versión de Android, pues las librerías de Java utilizadas no están actualizadas y pueden provocar conflictos en el proyecto. Puede realizarse la migración automáticamente en AndroidStudio en *Refactor/Migrate to AndroidX* o manualmente cambiando las librerías desactualizadas. En este caso lo más apropiado es realizar la migración manual y posteriormente actualizar manualmente las dos librerías restantes por su sustituto en AndroidX.

9.5.2. Orientación de pantalla

En la vista de “clasificación”, no se pueden mostrar el nivel de detalle deseado en un dispositivo móvil común, excluyendo las tabletas. Normalmente, las pantallas de estos dispositivos son estrechas y dificulta la visualización de contenido horizontal sin hacer uso de *scroll*.

Con el complemento de Cordova *screen orientation*, la orientación de la pantalla del dispositivo puede cambiarse, e incluso fijarse a una orientación determinada. De esta forma, el usuario puede ver la totalidad de los datos en la clasificación, rotando el dispositivo o pulsando el botón indicado en esta vista. Además, en el resto de las vistas de la aplicación la orientación queda fijada a vertical, o lo que el complemento conoce como *portrait (*retrato).

9.5.3. Almacenamiento

El almacenamiento del *token* del usuario para mantener la sesión iniciada en el dispositivo es una de las características fundamentales de las aplicaciones móviles. Para ello, es necesario el uso de almacenamiento local, es decir, estas credenciales deben almacenarse de forma segura y no volátil en el dispositivo.

Ionic ofrece el complemento de Cordova *storage*, que es una solución gratuita y de código abierto que hace uso del motor de base de datos SQLite para el almacenamiento de clave/valor y objetos de tipo JSON [19]. Utilizando este complemento, se almacenan las credenciales del usuario localmente, la aplicación lee el contenido de la base de datos al iniciarse y recupera estas credenciales si las hubiera. De esta forma, el usuario permanece con la sesión iniciada hasta que esta caduque o la cierre manualmente.

9.5.4 Transiciones nativas

Para hacer uso de transiciones propias de las aplicaciones móviles nativas se utiliza el complemento *native page transitions*. Este permite definir distintos detalles de la transición como la dirección, el retardo, la dirección o el efecto. En la aplicación desarrollada simplemente se hace uso de transiciones de izquierda o derecha, para crear la sensación de jerarquía en las vistas diseñadas.

10. Pruebas

10.1. Integración

En primer lugar, y tras la elaboración de cada *endpoint*, se realizaron pruebas con la herramienta Postman de las distintas funciones de la API Rest. Antes de integrarlas en la aplicación móvil, se garantizaba el correcto funcionamiento de estos métodos y que los datos fueran recibidos en el formato acordado.

Una vez se verificaba la API Rest, se añadía el método correspondiente a la aplicación y se comprobaba la integración con ayuda del inspector del navegador Firefox y las propias funciones de la aplicación.

Finalmente, se ha comprobado que la API Rest implementada por Todotorneos.com funciona según lo acordado y todos los errores han sido solventados.

10.2. Usabilidad

Tras cada iteración, se comprobaba la usabilidad de la aplicación en una reunión con el cliente. Además, después del *sprint 3,* se generó el fichero *APK* y se comprobaban las funcionalidades desde dispositivos móviles Android. De este modo, el cliente comunicaba errores y posibles mejoras que se aplicaban continuamente. Tanto el lado del servidor como el del cliente se implementaban en función del resultado de estas pruebas de usabilidad semanales y, por lo tanto, minimizando el riesgo de errores en iteraciones anteriores.

Para garantizar la usabilidad de la aplicación en una competición real, se creó un torneo de ajedrez tipo liga, que contaba con la participación de cuatro jugadores. Para ello, el torneo fue configurado como abierto y los participantes pudieron utilizar la aplicación para inscribirse en ella sin problemas. Se disputaron 6 jornadas, y durante el transcurso de la competición todos los participantes utilizaban la aplicación para consultar los resultados y clasificación, programar recordatorios para sus partidos y añadir el resultado de estos.

Se detectaron los siguientes errores o posibles mejoras durante la competición:

- Las notificaciones eran las mismas en todos los torneos.
- Se enviaba notificación al propio jugador que rechazaba un resultado.
- Las notificaciones recibidas cuando la aplicación estaba abiertas mostraban en modo de alerta.
- No se mostraba ningún mensaje cuando no había partidos o clasificación.
- Al cancelar un recordatorio no se eliminaba el color del icono.

- Los resultados grandes no cabían en la tabla.
- Introducir un salto de línea en el mensaje de cancelación producía un error de conversión a JSON al recibir las notificaciones.

Los errores eran solventados tras su detección y, tras la jornada 2 de la competición, no se volvieron a detectar nuevos errores. Además, las impresiones de los usuarios fueron muy positivas y ninguno tuvo problemas para hacer uso de la aplicación y sus funcionalidades. Las impresiones de los participantes recogidas durante el torneo de prueba fueron las siguientes:

- Fluidez y sin grandes tiempos de carga.
- Buena combinación de colores y estética.
- Mayor comodidad y organización al distribuir las notificaciones por categoría.
- Interfaz intuitiva con componentes utilizados por otras aplicaciones.
- Buena organización de las funcionalidades.
- Recordatorios de gran utilidad.
- Adaptación correcta a los distintos deportes y modalidades (eliminatoria y liga).

Ilustración 10. Jornada 1 del torneo de prueba

11. Resultado, conclusiones y trabajo futuro

11.1 Resultado y conclusiones

Al término del desarrollo del proyecto, y tras ejecutar las distintas iteraciones planteadas, encontramos como resultado una aplicación móvil multiplataforma que cumple con los objetivos planteados al inicio de la planificación y con los que se han ido añadiendo en el transcurso del desarrollo de esta. Las funcionalidades de la aplicación resultante se pueden resumir en:

- Consultar datos del torneo como resultados y clasificación.
- Programar recordatorios para los partidos del usuario.
- Añadir y validar resultados.
- Recibir notificaciones de los eventos del torneo.

Con estas funcionalidades, los usuarios de Todotorneos.com tendrán a su disposición un sistema fiable para la consulta de las competiciones en las que participen, además de poder realizar un seguimiento completo gracias a las notificaciones y los recordatorios. Se favorece así, tanto la visibilidad de la información como la colaboración entre participantes para mantener el torneo, reduciendo la centralización del sistema y eliminando carga de trabajo al administrador.

Esta aplicación tendrá una utilidad real, ya que ha sido diseñada con intención de ser publicada en la App Store (iOS) y en la Play Store (Android) y utilizada por los usuarios de Todotorneos.com. Por ello, la aplicación desarrollada se encuentra preparada para un entorno de producción y su despliegue en las plataformas requeridas.

El resultado del proyecto ha sido satisfactorio, ya que cumple con los requisitos iniciales del cliente y los añadidos posteriormente. Además, el código utilizado sigue en la medida de lo posible los principios de código limpio, facilitando la legibilidad del mismo y la escalabilidad y mantenimiento del software.

En conclusión, el desarrollo de este Trabajo de Fin de Título realiza aportaciones tanto al entorno socioeconómico como al personal. Este proyecto, además de ser de utilidad para los usuarios, me ha permitido desarrollar por primera vez una aplicación móvil, pasando además por todas las fases del proceso de desarrollo de software e interactuando con un cliente real, lo que permite a este trabajo reunir las condiciones necesarias para convertirse en una excelente preparación para el mundo laboral.

11.2 Trabajo futuro

El software desarrollado, a pesar de cumplir con los objetivos planteados, está orientado a ser ampliado y mejorado, por ello ha sido fundamental priorizar la legibilidad y simplicidad del código, de forma que cualquiera con conocimientos en las tecnologías utilizadas sea capaz de añadir nuevas funcionalidades y mejorar las existentes.

Por supuesto, y dada la naturaleza de la aplicación, para muchos de estos avances será necesaria la colaboración de Todotorneos.com, ya que es el único punto de acceso a los datos de la web. A continuación, se listan algunas de las potenciales mejoras y ampliaciones planteadas durante el desarrollo de la aplicación:

- Posibilidad de filtrar torneos abiertos por localización o deporte: En el futuro, y para cuando exista un gran número de torneos abiertos, será de utilidad poder filtrarlos por la localización de la competición o la modalidad deportiva que se practica. Actualmente, las competiciones abiertas son escasas y la API no devuelve los datos necesarios para realizar este filtrado.
- Posibilidad de aceptar o rechazar fechas de partidos y de proponer alternativas: De forma similar a la funcionalidad de añadir resultados, se podrían proponer fechas para los partidos, dando la posibilidad al rival de aceptarla o rechazarla y acordar otra diferente. Este proceso agilizaría la gestión de la competición y reduciría aún más la dependencia del administrador.
- Chat general de la competición: La inclusión de un chat general o foro facilitaría la comunicación entre los participantes, otorgándoles la capacidad de intercambiar impresiones o de acordar decisiones relacionadas con la competición.
- Zona de palmarés de los jugadores: Una nueva opción del menú principal podría ser un palmarés o estadísticas totales del usuario. Esta opción reflejaría el número de competiciones en las que ha participado, las victorias, derrotas, deportes y otras estadísticas del interés del usuario.
- Uso de JSON Web Token (JWT) para evitar el envío de un token manual: Actualmente, la autenticación en los *endpoints* se realiza a través de un parámetro en la petición. Esto, sin dejar de funcionar correctamente, podría mejorarse enviándose a través de la cabecera *Authentication* del protocolo HTTP. Además, el uso de JWT reduciría el continuo intercambio de datos, facilitando un solo *token* que servirá como autenticación y a su vez de resumen de los datos del usuario, que se cifrarán con un algoritmo, pudiendo acceder a ellos en cualquier momento.

- Formulario de contacto o de ayuda, tanto con el administrador del torneo como con el centro de ayuda de Todotorneos.com: En caso de duda o cualquier problema con la aplicación que pueda sufrir el usuario, sería recomendable incluir un formulario de ayuda o de contacto. Esto es realmente útil sobre todo durante los primeros usos de la aplicación, ya que se encontrará altamente vulnerable a errores no descubiertos aún. Por otra parte, la posibilidad de contactar con el administrador de un torneo específico sería también de gran utilidad para solventar, por ejemplo, errores de datos mal introducidos por el usuario.
- Integración de tests unitarios completos: La aplicación desarrollada integra pruebas unitarias con el *framework* Jasmine utilizado comúnmente con Angular, pero estas pruebas solo garantizan la correcta creación de componentes, simulando para ello los distintos servicios y complementos utilizados. Para aumentar la garantía del *software*, se podrían implementar pruebas de las funciones más importantes de cada componente, facilitando así su posterior ampliación y mejora.

Bibliografía

[1] Wikipedia, «GNU General Public License,» [En línea]. Available: https://es.wikipedia.org/wiki/GNU_General_Public_License. [Último acceso: 2 Junio 2020].

[2] GNU, «Licenses,» [En línea]. Available: https://www.gnu.org/licenses/licenses.es.html. [Último acceso: 2 Junio 2020].

[3] Git, «About,» [En línea]. Available: https://git-scm.com/about. [Último acceso: 2 Junio 2020].

[4] Wikipedia, «Licencia MIT,» [En línea]. Available: https://es.wikipedia.org/wiki/Licencia_MIT. [Último acceso: 2 Junio 2020].

[5] Wikipedia, «MIT License,» [En línea]. Available: https://en.wikipedia.org/wiki/MIT_License. [Último acceso: 2 Junio 2020].

[6] JetBrains, «What is our licensing model,» [En línea]. Available: https://sales.jetbrains.com/hc/en-gb/articles/206544679-What-is-our-licensing-model-. [Último acceso: 2 Junio 2020].

[7] JetBrains, «Licencias educativas gratuitas,» [En línea]. Available: https://www.jetbrains.com/es-es/community/education/#students. [Último acceso: 2 Junio 2020].

[8] Wikipedia, «Single-page application,» [En línea]. Available: https://es.wikipedia.org/wiki/Single-page_application. [Último acceso: 28 Mayo 2020].

[9] Wikipedia, «Angular (framework),» [En línea]. Available: https://es.wikipedia.org/wiki/Angular_(framework). [Último acceso: 28 Mayo 2020].

[10] Ionic Framework, «Capacitor vs Cordova,» [En línea]. Available: https://ionicframework.com/resources/articles/capacitor-vs-cordova-modern-hybrid-app-development. [Último acceso: 28 Mayo 2020].

[11] JetBrains, «Webstorm,» [En línea]. Available: https://www.jetbrains.com/es-es/webstorm/. [Último acceso: 2020 Junio 8].

[12] Wikipedia, «Android Studio,» [En línea]. Available: https://es.wikipedia.org/wiki/Android_Studio. [Último acceso: 8 Junio 2020].

[13] Wikipedia, «Xcode,» [En línea]. Available: https://es.wikipedia.org/wiki/Xcode. [Último acceso: 8 Junio 2020].

[14] Wikipedia, «Lenguaje unificado de modelado,» [En línea]. Available: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado. [Último acceso: 6 Junio 2020].

[15] Angular, «Lazy loading ngmodules,» [En línea]. Available: https://angular.io/guide/lazy-loading-ngmodules. [Último acceso: 6 Junio 2020].

[16] Github, «ionic-angular-schematics,» [En línea]. Available: https://github.com/Robinyo/ionic-angular-schematics. [Último acceso: 6 Junio 2020].

[17] Angular, «Architechture services,» [En línea]. Available: https://angular.io/guide/architecture-services. [Último acceso: 7 Junio 2020].

[18] Firebase, «Protocolo HTTP de Firebase Cloud Messaging,» [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/http-server-ref?hl=es. [Último acceso: 11 Junio 2020].

[19] Ionic Framework, «Storage,» [En línea]. Available: https://ionicframework.com/docs/angular/storage. [Último acceso: 15 Junio 2020].

Anexo I: Manual de usuario

Creación de una cuenta

Para comenzar a utilizar la aplicación y acceder al contenido de las competiciones, es necesario haberse registrado desde la misma e iniciar sesión con email y contraseña (ilustración 11).

Ilustración 11. Inicio de sesión

En caso de no disponer aún de una cuenta, podremos acceder al formulario de creación pulsando en “Regístrate” desde la vista de inicio de sesión. En ella introduciremos el nombre, apellidos, email, nick y contraseña que se asociarán a la cuenta (ilustración 12).

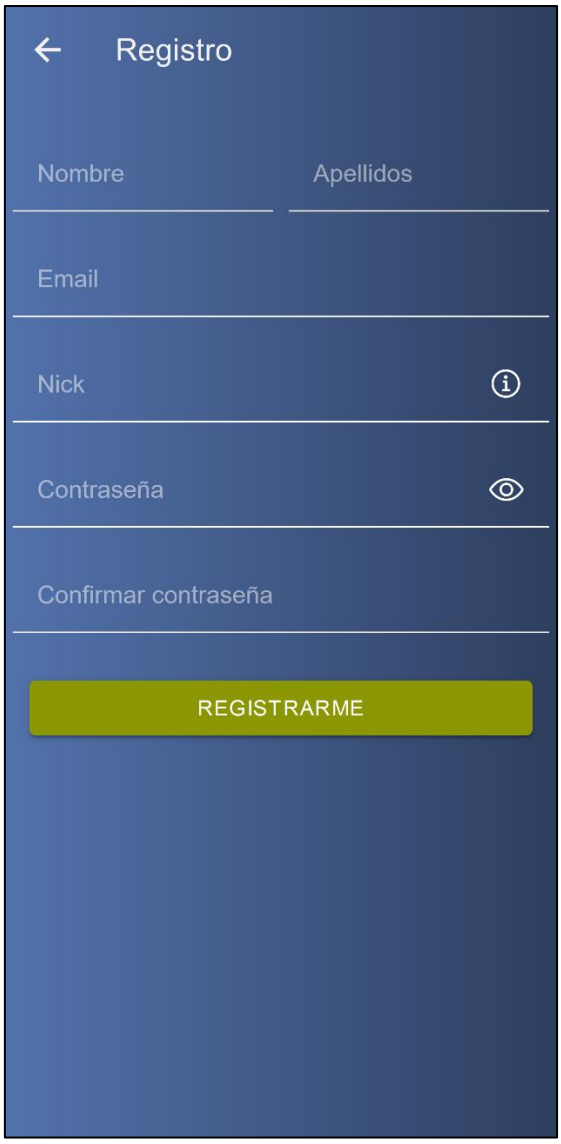

Ilustración 12. Registro

Tras enviar el formulario, se debe confirmar el correo electrónico y habremos completado el proceso de registro.

Inscripción en competiciones abiertas

Una vez creada la cuenta, se mostrará la vista principal y desde donde se accede a las distintas competiciones. Al principio, la lista de competiciones se encontrará vacía y se mostrará una opción de redirección a la vista de inscripciones, tal y como se muestra en la ilustración 13.

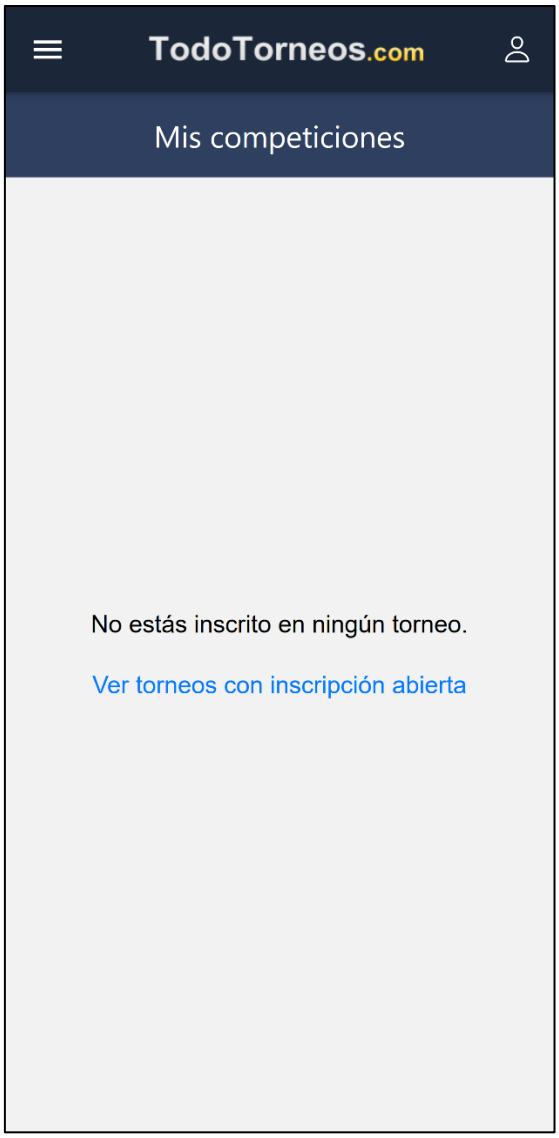

Ilustración 13. Lista de torneos vacía

Al pulsar en el enlace, se navegará a la página de inscripción en torneos abiertos, mostrada en la ilustración 14.

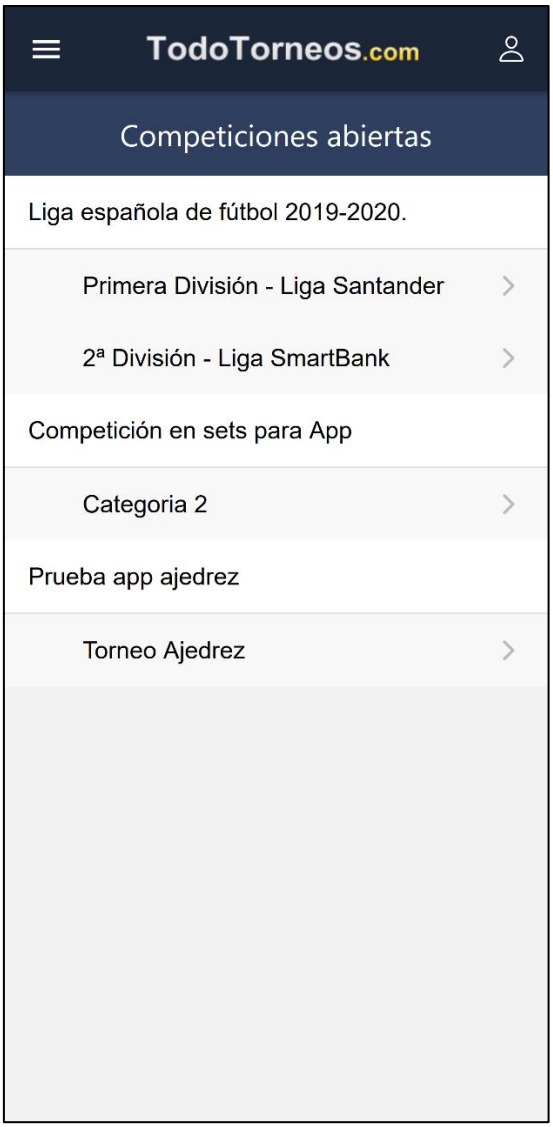

Ilustración 14. Lista de competiciones abiertas

La inscripción en una competición podrá hacerse tanto con un equipo existente como con un equipo nuevo. Para inscribirse con un nuevo equipo, habrá que añadir el nombre de este, para hacerlo con uno existente bastará con pulsar en él (ilustración 15).

Ilustración 15. Ventana de inscripción

Vista principal

Cuando el usuario esté inscrito en alguna competición, estas se mostrarán en forma de lista de la forma mostrada en la ilustración 16.

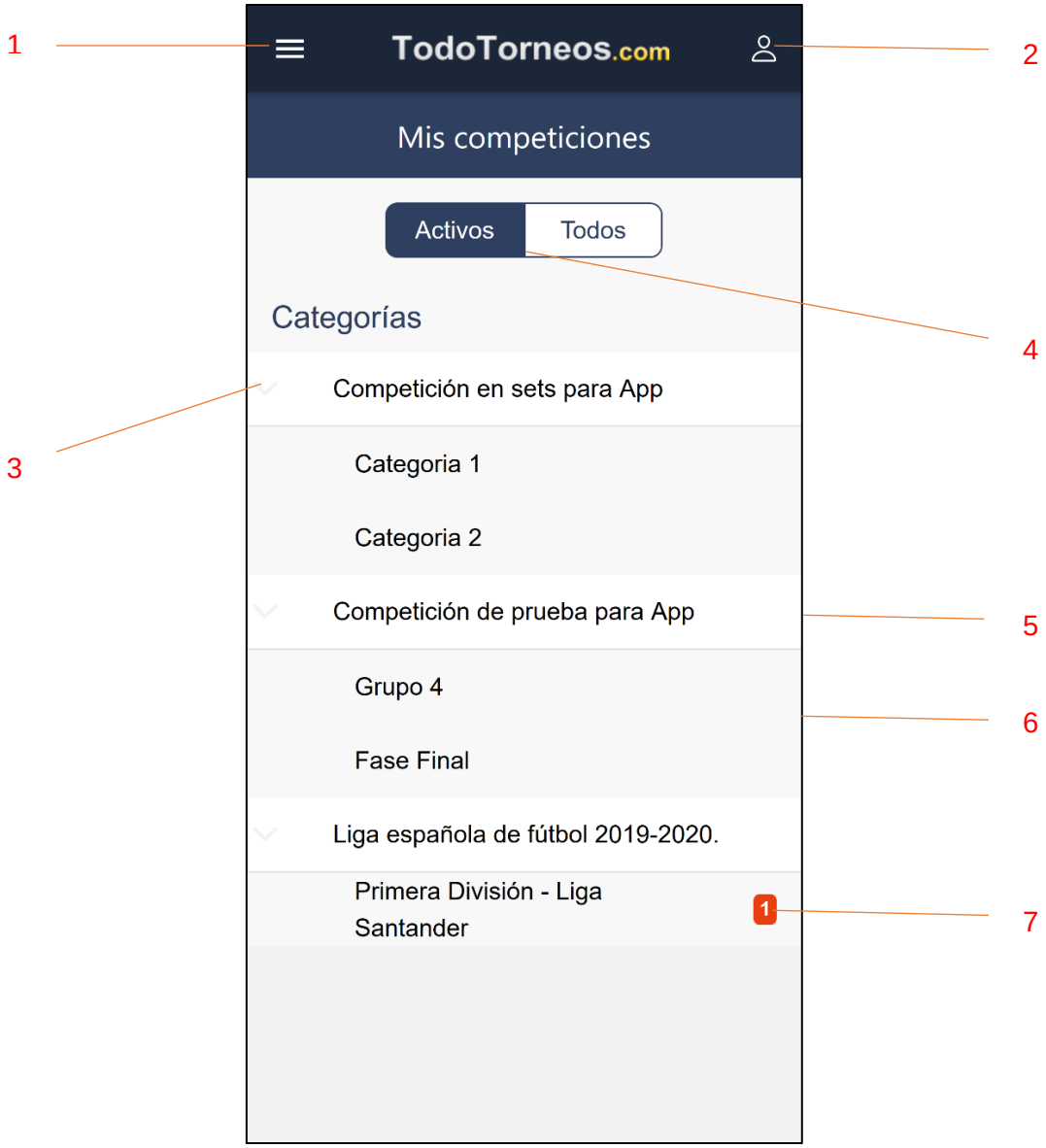

Ilustración 16. Elementos de la vista principal

A continuación, se indicarán también los distintos elementos que componen esta vista:

1. Menú principal de la aplicación.
2. Menú de usuario.
3. Botón de pliegue y despliegue de competiciones.
4. Filtro de competiciones.
5. Competición.
6. Categorías de la competición.
7. Número de notificaciones no leídas en la categoría.

Menú principal

El menú principal de la aplicación es utilizado para navegar entre las distintas funcionalidades que esta ofrece. Por ahora, la aplicación incluye la consulta de las competiciones del usuario y la inscripción en competiciones abiertas (ilustración 17).

Ilustración 17. Menú principal

Menú de usuario

El usuario dispone de su propio menú, en el que se incluyen las funcionalidades propias de este. En este caso, el menú permite acceder al perfil y cerrar la sesión activa, como muestra la ilustración 18.

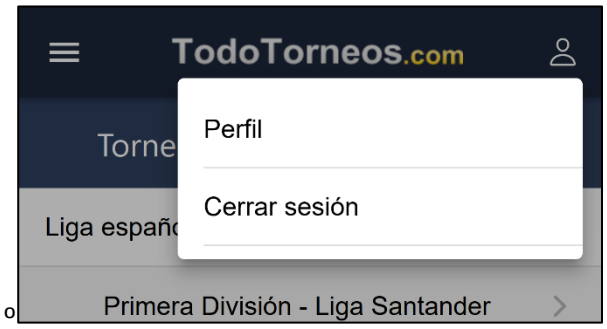

Ilustración 18. Menú de usuario

Consulta de una categoría

Resultados y clasificación

Al acceder a una categoría, se mostrarán por defecto los partidos del usuario en dicha categoría (ilustración 19).

Ilustración 19. Resultados del usuario

Deslizando hacia la izquierda o derecha o pulsando en la opción deseada en el menú superior se podrá acceder a las vistas de “Clasificación” (ilustración 20) y “Resultados” (ilustración 21), en las que se muestran los datos generales de la categoría.

Ilustración 20. Clasificación de una categoría

La clasificación mostrará más campos si el dispositivo móvil se orienta en horizontal, también se puede forzar el cambio de orientación pulsando en el icono de rotación.

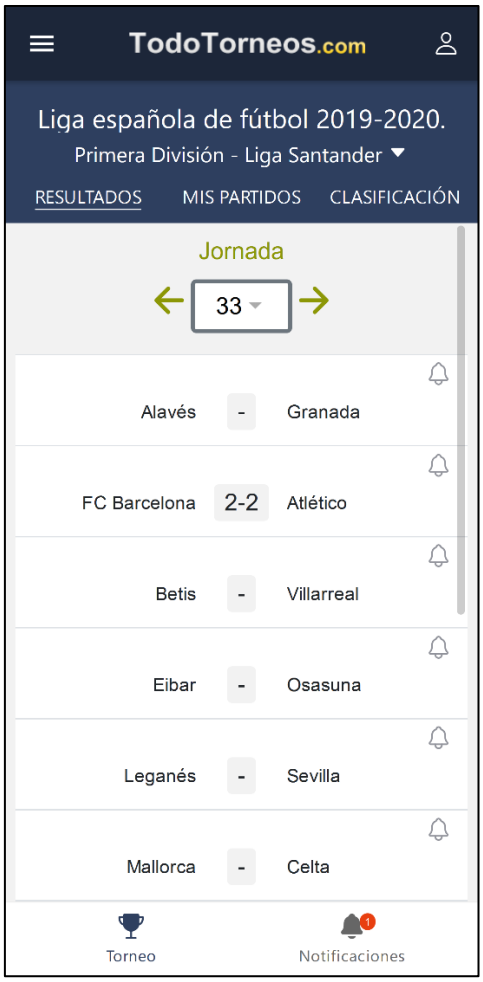

Ilustración 21. Resultados de una categoría

Para navegar entre jornadas o fases (eliminatoria), pueden usarse las flechas izquierda y derecha o usar el selector de jornada/fase pulsando sobre el número de la jornada/fase actual.

Se pueden consultar los datos de otras categorías del mismo torneo pulsando en la flecha del título de la categoría, lo que abrirá un selector como el que muestra la ilustración 22:

Ilustración 22. Selector de categorías

Tanto en la vista de “Resultados” como en la de “Mis partidos”, se pueden consultar los partidos de un equipo concreto a lo largo de la categoría pulsando en su nombre. Esto abrirá una ventana modal con todos los partidos del equipo elegido ordenados por jornada.

Notificaciones

El menú inferior es utilizado para navegar entre las vistas de datos del torneo y las notificaciones del mismo, junto al icono de notificaciones se muestra el número de notificaciones no leídas. La vista de notificaciones clasifica en “leídas” y “no leídas” las notificaciones recibidas (ilustración 23).

Ilustración 23. Notificaciones de una categoría

Se pueden eliminar las notificaciones deseadas deslizando hacia la izquierda la notificación y pulsando en el icono de papelera.

Añadir resultados

Los usuarios pueden guardar el resultado de sus partidos desde la vista de “Mis partidos”, pulsando el botón de “Añadir resultado” en los partidos cuyo resultado esté aún vacío se abrirá la ventana modal de la ilustración 24.

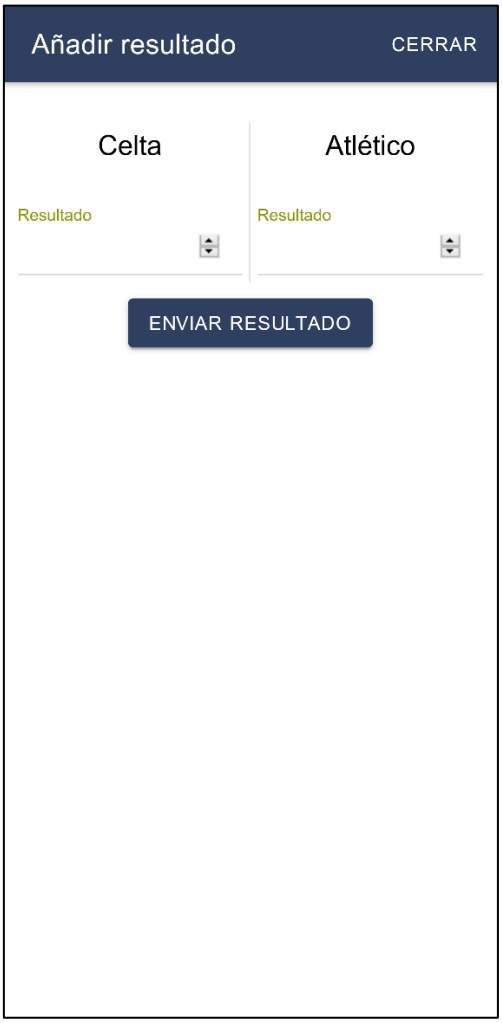

Ilustración 24. Ventana de añadir resultado

Los campos a enviar en este formulario dependerán de la modalidad de la categoría. Una vez añadido, el resultado se verá reflejado en la clasificación y en los resultados y se enviará una notificación al equipo rival. Si este lo rechaza, el resultado será eliminado.

Validar resultados

Cuando se recibe una propuesta de resultado el usuario será notificado, para validarla se accede a la vista de notificaciones de la categoría o directamente pulsando sobre la notificación, que abrirá esta vista. Al pulsar el botón “Validar” en la notificación deseada se abrirá una ventana en la que podremos elegir si aceptar o rechazar el resultado propuesto por el rival. En este último caso, se podrá adjuntar también un mensaje de justificación del rechazo, como muestra la ilustración 25.

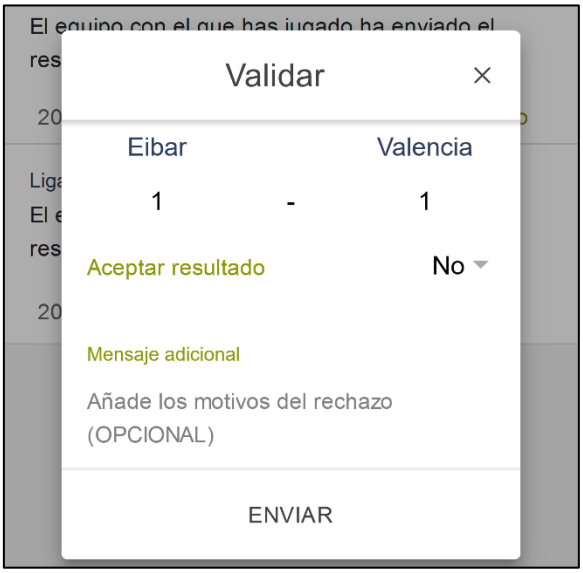

Ilustración 25. Validación de resultado

Programar recordatorios

Desde la vista de “Resultados”, se pueden programar recordatorios de los partidos pulsando en el icono de campana en el partido deseado. Para programarlos, es requisito indispensable que el administrador del torneo haya especificado fecha y hora en la web de la competición. La ventana emergente permite especificar la antelación al partido con la que se programará el recordatorio, como se puede ver en la ilustración 26.

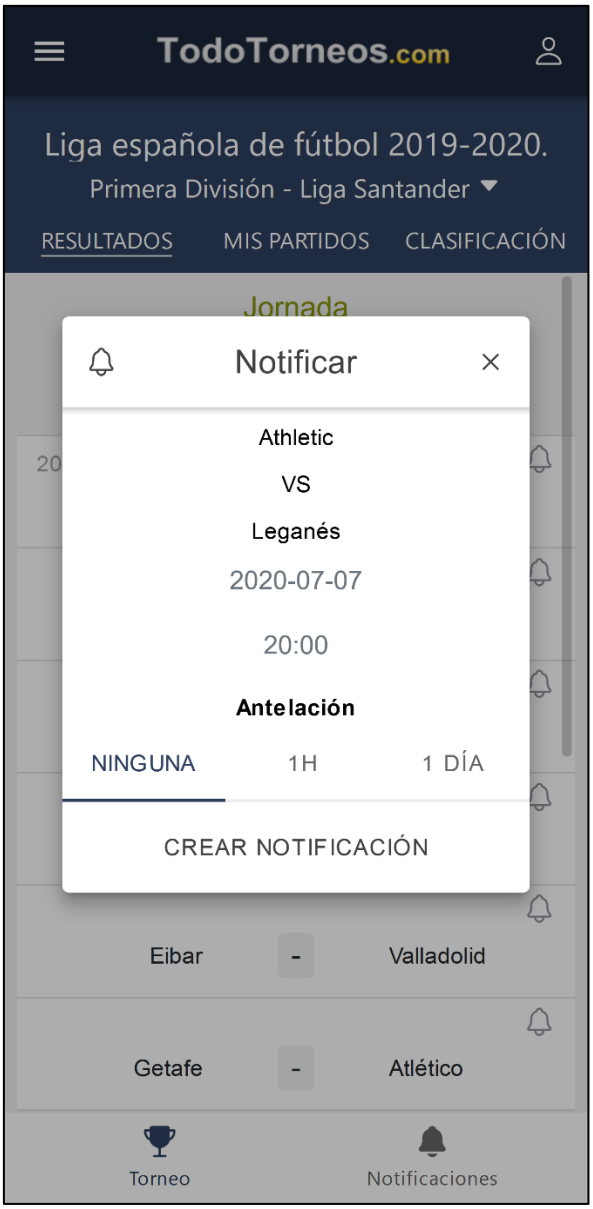

Ilustración 26. Creación de recordatorio

El icono de campana aparecerá coloreado de amarillo en los partidos cuyo recordatorio se encuentre activado.

Editar perfil

Se pueden modificar el nombre, apellidos y contraseña del usuario accediendo al perfil desde el menú de usuario. Desde la página de “Perfil”, se podrá editar tanto el nombre como los apellidos y posteriormente enviar los nuevos datos (ilustración 27).

Ilustración 27. Editar perfil

Para modificar la contraseña es necesario pulsar en “Cambiar la contraseña”, lo cual abrirá la ventana modal mostrada en la ilustración 28 y permitirá fijar una nueva comprobando previamente la actual.

Ilustración 28. Cambiar contraseña

Restablecer contraseña

En caso de no recordar la contraseña actual y no poder acceder a la aplicación, se podrá acceder a la funcionalidad de restablecer contraseña desde la página de inicio de sesión. La ventana emergente de la ilustración 29 permitirá introducir el correo electrónico del usuario y se le enviará un correo con los pasos para establecer la nueva contraseña.

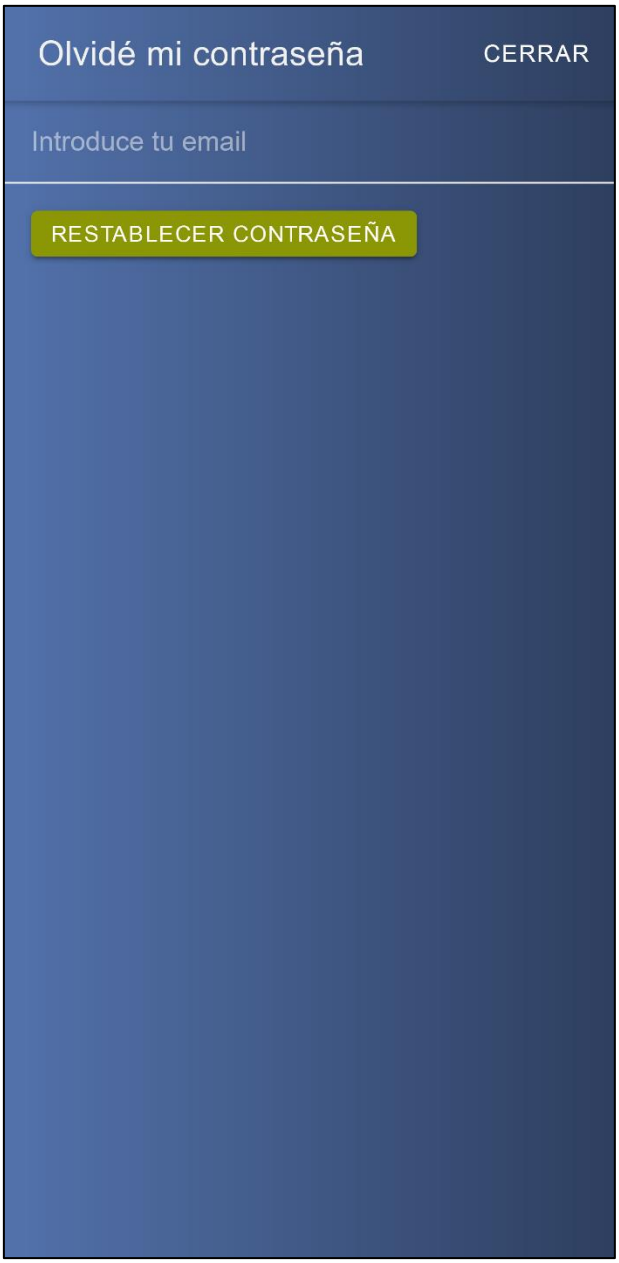

Ilustración 29. Restablecer contraseña

Anexo II: Endpoints

1. Inicio de sesión

- Método: POST
- Endpoint**:** login
- Solicitud: {usuario, contraseña, so, id}
- Respuesta: {nombre, apellidos, nick, email, token} o código de estatus de error.

2. Registro

- Método: POST
- Endpoint**:** registro
- Solicitud: {nombre, apellidos, email, nick, contraseña}
- respuesta: Respuesta: Código de estatus 200 o de error.

3. Editar contraseña

- Método: POST
- Endpoint**:** cambiapwd
- Solicitud: {usuario, token, antigua, nueva}
- Respuesta: Código de estatus 200 o de error.

4. Restablecer contraseña

- Método: POST
- Endpoint**:** recordarpwd
- Solicitud: {usuario}
- Respuesta: Código de estatus 200 o de error.

5. Editar perfil

- Método: POST
- Endpoint: cambiaperfil
- Solicitud: {usuario, token, nombre, apellidos}
- Respuesta: Código de estatus 200 o de error.

6. Obtener lista de torneos

- Método: GET
- Endpoint: gettorneoslist
- Solicitud: usuario, token
- respuesta: [{id, nombre, categorias[{id,nombre, tipo, numnotificaciones}]

7. Obtener categoría

- Método: GET
- Endpoint: getcategoria
- Solicitud: usuario, token, idCategoria,
- respuesta: [{nombre, id, nombreTorneo, idTorneo, idEquipo, nombreEquipo, modalidadVisual, tipo, jornadaActiva, fases}]

8. Obtener resultados (liga)

- Método: GET
- Endpoint: getrdosliga
- Solicitud: torneo, jornada?
- respuesta: {jornada,totaljornadas, observacionesjornada, modalidadvisual, resultados: [ ]}

9. Obtener resultados (eliminatoria)

- Método: GET
- Endpoint: getrdoseliminatoria
- Solicitud: torneo, fase?
- Respuesta: {fase, totalfases, observacionesfase, resultados: [ ]}

10. Obtener resultados de un equipo (liga)

- Método: GET
- Endpoint: getrdosligaequipo
- Solicitud: torneo, idequipo, usuario

- respuesta: {jornadaactiva, totaljornadas, puedeactualizar, modalidadvisual, resultados: [ ]}

11. Obtener resultados de un equipo (eliminatoria

- Método: GET
- Endpoint: getrdoseliminatoriaequipo
- Solicitud: torneo,idequipo
- respuesta: {resultados: [ ]}

12. Obtener clasificación

- Método: GET
- Endpoint: getclasifica
- Solicitud: torneo
- respuesta: {jornada, posiciones: [ ]}

13. Obtener lista de categorías de un torneo

- Método: GET
- Endpoint: getcategorias
- Solicitud: torneo
- respuesta: [{id,nombre,tipo}]

14. Obtener notificaciones de una categoría

- Método: GET
- Endpoint: getnotificaciones
- Solicitud: {usuario, token, idCategoria}
- respuesta: notificacion[{idpartido, idtorneo]

15. Leer notificaciones

- Método: POST
- Endpoint: readnotificaciones

- Solicitud: {usuario, token, notificaciones: [id]}
- Respuesta: Código de estatus 200 o de error.

16. Borrar notificaciones

- Método: POST
- Endpoint: borranotificaciones
- Solicitud: {usuario, token, notificacion}
- Respuesta: Código de estatus 200 o de error.

17. Obtener invitaciones

- Método: GET
- Endpoint: getinvitaciones
- Solicitud: {usuario, token}
- respuesta: {id, titulo, descripcion}

18. Responder invitación

- Método: POST
- Endpoint: respondeinvitacion
- Solicitud: {usuario, token, idinvitacion,respuesta(OK/NOK)}
- Respuesta: Código de estatus 200 o de error.

19. Grabar resultado (liga)

- Método: POST
- Endpoint: grabardoliga
- Solicitud: {usuario, token, idtorneo, idpartido, njornada, idequipo1, idequipo2, equipoganador, txtrdo11, txtrdo12, txtrdo13, txtrdo14, txtrdo15, txtrdo21, txtrdo22, txtrdo23, txtrdo24, txtrdo25}
- Respuesta: Código de estatus 200 o de error.

20. Grabar resultado (eliminatoria)

- Método: POST
- Endpoint: grabardoeliminatoria

- Solicitud: {usuario, token, idtorneo, idpartido, idequipo1, idequipo2, equipoganador, txtrdo11, txtrdo12, txtrdo13, txtrdo14, txtrdo15, txtdatos1, txtrdo21, txtrdo22, txtrdo23, txtrdo24, txtrdo25, txtdatos2}
- Respuesta: Código de estatus 200 o de error.

21. Validar resultado

- Método: POST
- Endpoint: validardo
- Solicitud: {usuario, token, idpartido, idcategoria, validar (OK, NOK), textonok}
- Respuesta: Código de estatus 200 o de error.

22. Obtener torneos con inscripción abierta

- Método: GET
- Endpoint: gettorneosinscripcion
- Solicitud: {usuario,token}
- Respuesta: [{nombretorneo, categorias: [{id, nombre}]}]

23. Obtener equipos de una categoría

- Método: GET
- Endpoint: getequiposcategoria
- Entrada: {usuario,token,idcategoria}
- Respuesta: {permitirinscripcion, equipos: [{id, nombreequipo}]}

24. Inscribir jugador

- Método: POST
- Endpoint: inscribejugador
- Solicitud: {usuario,token,idcategoria,idequipo} o { usuario, token, idcategoria, nombreequipo}
- Respuesta: Código de estatus 200 o de error.

Anexo III: Manual de instalación

1. Descargar el código fuente de Github
2. Descargar e instalar la versión LTS de NodeJS
3. Ejecutar *npm install* en el directorio

Los comandos utilizados a continuación se ejecutan con *npx* para acceder a los módulos incorporados en el proyecto, esto se hace para evitar la instalación de dichos módulos. En caso de querer instalarlos globalmente, los módulos necesarios serán las interfaces de línea de comando (CLI) de Angular y de Ionic:

- *npm install* -g *@angular/cli*
- *npm install -g @ionic/cli*

La aplicación se encontrará instalada y preparada para su prueba en un entorno de desarrollo en diferentes plataformas. Los requisitos previos para la ejecución de aplicaciones en ambas plataformas móviles se encuentran en la documentación oficial de Capacitor: https://capacitorjs.com/docs/getting-started/dependencies. A continuación se indican los pasos de instalación en las distintas plataformas:

- Web: Ejecutar npx ng serve.
- iOS:
    1. Ejecutar *npx ionic cap run ios*.
    2. Ejecutar la aplicación en XCode.
- Android:
    1. Ejecutar *npx ionic cap run android* (si AndroidStudio no se abre automáticamente hay que abrir manualmente el directorio *android* generado).
    2. Migrar el proyecto a AndroidX en AndroidStudio con la opción *Refactor/Migrate to AndroidX.*
    3. Sustituir las librerías:
        - *android.support.v4.media.app.NotificationCompat.MediaStyle* por *androidx.media.app.NotificationCompat.MediaStyle.*
        - *android.util.ArraySet<E>* por *androidx.collection.ArraySet*
    4. Ejecutar la aplicación en AndroidStudio.