



DOCUMENTACIÓN DE TENISCLUBDROID

CIP VIRGEN DE GRACIA

En esta documentación se verán todos los aspectos del desarrollo de la aplicación TenisClubDroid.

Pablo Prado Ruiz

Índice

1. Presentación del proyecto

- 1.1 Explicación resumida
- 1.2 Estudio de mercado
- 1.3 Valor de producto

2 Planificación de tareas y estimación de costes

- 2.1 Planificación y organización de tareas
- 2.2 Estimación de costes y recursos: hardware, software y humanos
- 2.3 Herramientas usadas
- 2.4 Gestión de riesgos

3 Análisis de la solución

- 3.1 Análisis de la solución
- 3.2 Análisis de escenarios

4 Diseño de la solución

- 4.1 Diseño de la interfaz de usuario y prototipos
- 4.2 Diseño de la persistencia de la información
- 4.3 Diseño de la arquitectura del sistema

5 Implementación de la solución

- 5.1 Justificación tecnológica
- 5.2 Aspectos esenciales de la implementación

6 Testeo y pruebas de la solución

- 6.1 Plan de pruebas (unitarias, integración, sistema y usuarios)
- 6.2 Solución a problemas encontrados

7 Lanzamiento y puesta en marcha

- 7.1 Aspectos relevantes del despliegue y puesta en marcha del sistema
- 7.2 Manual de uso

8 Valoración y conclusiones

9 Bibliografía y recursos utilizados

1. Presentación del proyecto

1.1. Introducción

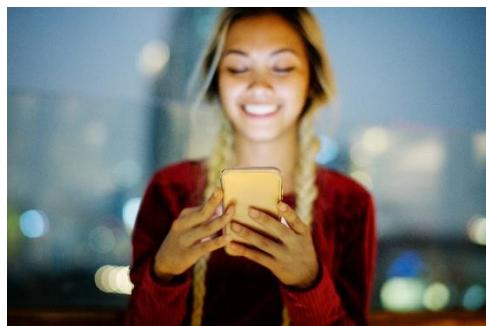
Esta aplicación consistirá en una app para poder reservar y gestionar pistas de un club de tenis. Será multiplataforma y tendrá los datos del club, así como su ubicación para poder ir con google maps. La aplicación también tendrá un listado con contactos con los que podrá quedar para jugar reservando una pista.



En la aplicación habrá un calendario con todas las reservas y partidos que tenga el usuario para facilitar visualmente sus partidos. El usuario podrá subir fotos del partido y quedarán guardadas en el partido y podrán ser vistas en el historial de encuentros.

Habrá una zona de perfil donde se verá una foto del usuario, su nombre de usuario y un código QR, tanto la foto del usuario y su nombre se podrán cambiar, el nombre de usuario solo se podrá cambiar si no hay otro usuario con el mismo. También habrá un chat privado con los distintos usuarios que tenga el usuario agregado.

Finalmente habrá un apartado para poder reservar pistas seleccionando entre las disponibles, elegir entre los extras y finalmente realizar el pago electrónico. Para acceder a las instalaciones presentará su código QR y así quedará guardado sus entradas y salidas.



Para gestionar la aplicación principal habrá una aplicación paralela para pc donde el administrador realizara una gestión de las instalaciones ya que no siempre pueden estar disponibles ya que pueden estar en reformas o mejoras.

También podrá fijar un precio fijado por tramos horarios para cada pista y podrá gestionar los precios por los extras de las pistas. Finalmente podrá gestionar a los socios y crear informes de contabilidad por pistas.

1.2. Estudio de mercado

En este apartado se verán las aplicaciones existentes que son similares a la nuestra y en cierto modo la “competencia”. Las aplicaciones más populares que se asemejan a la nuestra son:

-**Playtomic**: es la app líder en reserva de pistas de pádel y tenis. Con +300.000 usuarios activos al mes, cuenta con más de 2.000 clubes en 19 países distintos.



-**Sporttia**: es una app que permite reservar pistas y actividades a través del móvil, sin necesidad de llamar ni desplazarse hasta el centro deportivo.

Entre sus actividades están el Pádel, baloncesto, fútbol, tenis, voleibol, yoga, senderismo o danza. La aplicación, disponible para Android e iOS, permite incluso alquilar la calle de una piscina, una sauna o una sesión de fisioterapia.



Pese a la competencia, es un mercado muy amplio ya que cada vez más personas prefieren hacer todas las reservas online y así tener bien planificado su deporte favorito y nuestra aplicación será muy útil para ello.

Ahora veremos un análisis DAFO para tener una imagen más amplia de nuestra aplicación. En este DAFO podremos ver las debilidades, amenazas, fortalezas y finalmente las oportunidades de la aplicación.

- **Debilidades:** residen en la novedad del producto, nuestra aplicación se va a enfrentar a varios competidores que ya están asentados en el mercado y tendremos buscar nuestro lugar en él.
- **Amenazas:** las amenazas de la app es la gran variedad de aplicaciones que sirven para lo mismo y que ya están asentadas en el mercado.
- **Fortalezas:** nuestra fortaleza un diseño simple pero que brinda toda la información que el usuario necesita ver sin crear molestias y permite en una sola aplicación tener un control de las pistas reservadas como de nuestros amigos con los que solemos jugar.
- **Oportunidades:** nuestra mayor oportunidad es que el mercado que afecta a nuestra aplicación es muy amplio y cada vez lo va a ser más, pues cada vez menos gente quiere ir en persona para hacer una reserva de una pista.

Con este DAFO hemos visto los puntos flacos a mejorar de la aplicación y nuestros puntos fuertes que tenemos que potenciar para que nuestra aplicación sea todo un éxito.

Finalmente veremos el público al que va dirigida TenisCLubDroid.

Nuestro público comprende a personas mayores de 18 (para poder realizar la reserva) y personas 50 que o bien quieran jugar al tenis con sus amigos o por ejemplo padres que quieren jugar con sus hijos.

1.3. Valor del producto

Existen varias aplicaciones similares como hemos podido ver, pero algunas de las características que diferencian a nuestro producto del resto es el diseño minimalista e intuitivo de la aplicación donde el usuario podrá ver fácilmente su calendario de partidos, los contactos con los que habla y suele quedar para jugar y la facilidad para reservar pistas y pagar electrónicamente.

2. Planificación de tareas y estimación de costes

2.1 Planificación y organización de tareas

En este punto se va a ver la planificación general para el proyecto, con la explicación resumida de cada punto y la fecha estimada para finalizar cada punto. Para la planificación voy a utilizar las herramientas que proporciona GitHub, con ellas se podrá controlar el progreso de la aplicación y ver las partes que están en desarrollo, las que están finalizadas y las que quedan por hacer.

The image shows a GitHub project board titled "Proyecto_Final" updated on 2 Dec 2020. It has three columns: "To do", "In progress", and "Done".

- To do:**
 - Testeo y pruebas de la solución (0 of 2)
 - Lanzamiento y puesta en marcha (0 of 2)
 - Valoración, conclusiones, bibliografía y recursos utilizados (10 opened by Pablogls2)
- In progress:**
 - Ánalisis de la solución (1 of 2)
 - Implementación de la solución (1 of 3)
- Done:**
 - Presentación del proyecto (3 of 3)
 - Diseño de la solución (3 of 3)
 - Planificación de Tareas y Estimación de costes (4 of 4)

La planificación es la siguiente:

1. Presentación del proyecto:

En este apartado se explica de forma resumida el proyecto, su estudio de mercado y valor de producto

The image shows a GitHub issue titled "Presentación del proyecto #3" closed by Pablogls2 on 21 Oct 2020 with 0 comments. The issue contains a checklist and estimated completion dates.

Pablogls2 commented on 21 Oct 2020 • edited

Explicación resumida
 Estudio de mercado
 Valor del producto

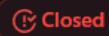
Fecha estimada para terminar: 18/10/2020
Fecha finalizada: 18/10/2020

la fecha estimada para terminar es el 18/10/2020

2. Planificación de tareas y estimación de costes:

En este apartado se ve la planificación y organización de tareas, estimación de costes y recursos, herramientas utilizadas y gestión de riesgos.

Planificación de Tareas y Estimación de costes #5

 Closed

Pablogls2 opened this issue on 21 Oct 2020 · 0 comments



Pablogls2 commented on 21 Oct 2020 · edited

- Planificación y organización de tareas
- Estimación de costes y recursos: hardware, software y humanos
- Herramientas usadas
- Gestión de riesgos

Fecha estimada para terminar: 27/10/2020

la fecha estimada para terminar es el 27/10/2020

3. Análisis de la solución:

En este apartado se ve el análisis de la solución y sus escenarios.

Análisis de la solución #4

 Open

Pablogls2 opened this issue on 21 Oct 2020 · 0 comments



Pablogls2 commented on 21 Oct 2020 · edited

- Análisis de la solución
- Análisis de casos de uso

Fecha estimada para terminar: 19/11/2020

Fecha finalizada: 20/11/2020

la fecha estimada para terminar es el 19/11/2020

4. Diseño de la solución:

En este apartado se ve el diseño de la interfaz de usuario, la persistencia de la información y la arquitectura del sistema.

Diseño de la solución #9



Pablogls2 opened this issue on 21 Oct 2020 · 0 comments



Pablogls2 commented on 21 Oct 2020 · edited

- Diseño de la interfaz de usuario y prototipos
- Diseño de la persistencia de la información
- Diseño de la arquitectura del sistema

Fecha estimada para terminar: 10/11/2020

la fecha estimada para terminar es el 10/11/2020

5. Implementación de la solución:

En este apartado se justificará la tecnología, se verán los aspectos esenciales de la implementación y el desarrollo de la funcionalidad indicada por el tutor.

Implementación de la solución #6



Pablogls2 opened this issue on 21 Oct 2020 · 0 comments



Pablogls2 commented on 21 Oct 2020 · edited

- Justificación tecnológica
- Aspectos esenciales de la implementación
- Desarrollo de la funcionalidad indicada por el tutor/a

Fecha estimada para terminar: 05/01/2020

la fecha estimada para terminar es el 05/01/2021

6. Testeo y pruebas de la solución:

En este apartado se testeará la aplicación y se probarán sus funcionalidades.

Testeo y pruebas de la solución #7

 **Pablogls2** opened this issue on 21 Oct 2020 · 0 comments

Pablogls2 commented on 21 Oct 2020 · edited

- Plan de pruebas
- Solución a problemas encontrados

Fecha estimada para terminar: 7/01/2020

la fecha estimada para terminar es el 07/01/2020

7. Lanzamiento y puesta en marcha:

En este apartado se verán los aspectos relevantes del despliegue y puesta en marcha del sistema.

Lanzamiento y puesta en marcha #8

 **Pablogls2** opened this issue on 21 Oct 2020 · 0 comments

Pablogls2 commented on 21 Oct 2020 · edited

- Aspectos relevantes del despliegue y puesta en marcha del sistema
- Manual de uso

Fecha estimada para terminar: 12/01/2020

la fecha estimada para terminar es el 12/01/2021

8. Valoración, conclusiones, bibliografía y recursos utilizados:

En este apartado se hará una valoración y unas conclusiones del proyecto y se mostrará la bibliografía y recursos.

Valoracion, conclusiones, bibliografia y recursos utilizados

(1) Open Pablogls2 opened this issue on 21 Oct 2020 · 0 comments

Pablogls2 commented on 21 Oct 2020 · edited

Owner

Fecha estimada para terminar: 14/01/2020

la fecha estimada para terminar es el 14/01/2021

2.2. Estimación de costes y recursos: hardware, software y humanos

Para estimar los costes y los recursos del proyecto se deben determinar las personas, equipos y/o materiales necesarios para llevarlo a cabo. La exactitud de la estimación del costo de un proyecto, aumenta según avanza el proyecto, de manera que es un proceso iterativo.

En nuestro proyecto solo va a trabajar una persona (Pablo Prado) y este se encargará de todas las fases del desarrollo.



Pablo prado
Pablogls2

Por lo tanto, los costes son los siguientes:

- **Sueldo del trabajador:** será de 22 € a la hora y se estima que el proyecto se realice en 80h, por lo tanto, el coste total del equipo será de **1760€**.
- **Cuota de autónomos:** se cobrará un 33% de la cuota de autónomos para amortizar gastos, lo cual equivale a **93.39€**.
- **Hardware:** no se necesita un hardware especial para este proyecto por lo que no será necesario nada dedicado a él, aun así, se necesitará el uso de un **SAI** para que en caso de que se vaya la luz no haya riesgos en cuanto hardware o

perdida de software, los precios rondan 115€, pero no se cargará todo al proyecto y se tomará como una inversión, por lo que el hardware tendrá un coste de **60€**.

- **Software:** el software necesario que tiene licencia radica en el uso de **Firebase**, el servicio de Firebase tiene una licencia que tendrá un coste dependiendo del número de usuarios y de información que se consuma, haciendo una estimación de unos 200 usuarios mensuales o más, el uso de varias gigas para almacenamiento, el coste para este proyecto llegaría a los **100€**.

Resumen de gastos:

Coste	Precio (€)
Sueldo de trabajador	1.760
Cuota de autónomo	93,39
Hardware	60
Software	100

El total es de: **2.013,39 €**(con i.v.a)

2.3. Herramientas usadas

En este apartado veremos las herramientas utilizadas por la aplicación, así como su función. Son las siguientes:

-**Android Studio:** es el IDE oficial para la plataforma Android y está basado en el software de IntelliJ, cuenta con integración de sistemas de gestión de versiones como GitHub y tiene muchas funcionalidades para desarrollar en Android. Esta herramienta será con la que se desarrollará la aplicación móvil.



-**Firebase:** es una plataforma en la nube para el **desarrollo de aplicaciones web y móvil**, sus herramientas son variadas y de fácil uso, como ser proporcionar datos en tiempo real y permitir autenticación de usuarios. Esta herramienta será con la que se realice el servidor y base de datos de la aplicación.



-**JustInMind:** es una herramienta muy simple pero muy útil para hacer maquetas de aplicaciones para poder diseñarla sin necesidad de código y así poder ir ajustando la aplicación visualmente para a la hora de hacerla de verdad sea más ágil. Con esta herramienta se va crear una maqueta de la aplicación para que el cliente pueda opinar sobre ella y así poder realizar los cambios que sean necesarios de una forma veloz.



-**ArgoUML**: es una herramienta para la creación de diagramas UML para así poder hacer análisis de proyectos con diagramas de casos de uso, diagramas de clases. Con esta herramienta se van a realizar los diagramas del análisis del proyecto.



ArgoUML

eSoftner

-**NetBeans**: es un IDE para el desarrollo de diferentes proyectos en distintos lenguajes como java, JavaScript... Esta herramienta va ser utilizada para hacer la aplicación de administración del proyecto.



-**GitHub**: es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores. Con esta herramienta se hará el control de versiones de la aplicación, así como con gitflow se realizará la planificación de tareas y con el uso de wikis se documentará el proyecto.



2.4. Gestión de riesgos

Para poder gestionar los riesgos primero debemos saber identificar tipos de riesgos y así poder solucionarlos. Un riesgo se define como cualquier cosa que retrase o impida que avance del proyecto al ritmo esperado o que provoque que se salga del presupuesto estimado. En definitiva, cualquier aspecto del proceso que suponga una desviación del plan previsto y que permita que el proceso fracase.

Riesgos laborales:

Gran parte del trabajo propio del sector de la informática se desarrolla fundamentalmente en oficinas. Este tipo de trabajo implica la exposición a determinadas condiciones ambientales como es el caso del ruido, la temperatura, la humedad, iluminación etc. todo ello con una influencia directa sobre la comodidad y la salud de los trabajadores y trabajadoras.

Algunos de los riesgos más comunes son:

- **El dolor de espalda** y otros trastornos músculo esqueléticos, los trastornos músculo esqueléticos son una de las enfermedades de origen laboral más comunes, normalmente afectan a la espalda, cuello, hombros y extremidades superiores. De ellos se derivan problemas de salud, que van desde pequeñas molestias y dolores, a cuadros médicos más graves.
- **Fatiga visual**, conocida como fatiga ocular, se ha visto incrementada con el uso de las nuevas tecnologías y su incorporación al entorno laboral. La fatiga visual presenta una sintomatología que se evidencia con dolor de cabeza, sensación constante arenilla en los ojos, sequedad etcétera. En el ámbito laboral, la fatiga visual se convierte en un elemento de riesgo o peligro, dado que en el sector de la informática la continua lectura de documentos o en la pantalla de ordenadores sin protectores visuales o con bajos niveles de iluminación.

En este sentido, es importante disponer adecuadamente el puesto de trabajo, específicamente la pantalla, con el ángulo visual, distancia visual etc.... hay que controlar el contraste y el brillo de la citada pantalla, limpiarla y que esta tenga un adecuado nivel de iluminación. Es recomendable igualmente, que los profesionales realicen periódicamente técnicas de relajamiento, para evitar lesiones, así como otros trastornos ya señalados, caso de la fatiga visual.

Riesgos de proyecto:

En cuanto a riesgos de proyecto software, hay muchos otros riesgos que no son de ámbito laboral que pueden presentarse, pero por ello hay que tenerlos presentes para poder solucionarlos de la mejor forma posible. Algunos de los riesgos a los que nos podremos enfrentar son:

- **Riesgos derivados del cliente.** Algunos que los riesgos que se pueden dar son la **falta de colaboración**, para ello se le insistirá para no crear problemas de confusión de lo que pide. Otro riesgo común es que en fases avanzadas del proyecto el cliente demande **nuevos requisitos**, en este caso se acordarán de nuevo los precios y se aceptarán mostrando una actitud empática ya que a todos se nos pueden olvidar cosas.
- **Riesgos derivados de la planificación.** En algún momento del desarrollo de un proyecto se pueden producir **retrasos** dados por bien por problemas técnicos o por una mala planificación, para solucionarlo se deben hacer esfuerzos para arreglar el problema y que así no afecte a otras tareas y revisar la planificación por si hay más errores y así solucionarlos para que no se repitan, además todo el tiempo que este fuera del acordado con el cliente repercutirá sobre el equipo.
- **Riesgos derivados del producto.** Existen muchos riesgos de este ámbito como la posible **pérdida del código**, para prevenirlo se realizarán varias copias de seguridad y se usarán sistemas de control de versiones como GitHub para guardar el código. Otro posible riesgo es una **mala calidad**, para prevenir este problema se debe testear cada parte del proyecto e ir consultado con el cliente para si es lo que él desea y no ir arrastrando errores.

3.Análisis de la solución

3.1. Análisis de la solución

En esta fase estarán comprendidas las tareas que un software va realizar y en las condiciones en las que va actuar. Para ello nos centraremos en QUÉ va a realizar la aplicación y determinar sus restricciones. Los requisitos de la aplicación serán los siguientes:

Requisitos funcionales

Los requisitos funciones definen una función del sistema o sus componentes. Cada función se comporta como un conjunto de entradas, comportamientos y salidas. Dichos requisitos pueden ser cálculos, detalles, técnicos, manipulación de datos, etc.

Los requisitos del sistema móvil son los siguientes:

- RF1. Ver datos del club, geolocalización y galería fotográfica.
- RF2. Ver estado de las pistas en tiempo real.
- RF3. Tendrá una lista de contactos, se podrá seleccionar uno para iniciar el proceso de quedar para jugar.
- RF4. Habrá un calendario de las reservas y partidos del usuario.
- RF5. Tendrá una línea de tiempo donde se verá el historial de encuentros, los usuarios con los que ha jugado el usuario, los resultados y fotografías del partido.
- RF6. Al subir una foto se podrá usar la cámara o seleccionarlas del almacenamiento interno.
- RF7. Habrá una zona de perfil donde el usuario podrá cambiar su descripción, su nombre de usuario (si no está ocupado) y su fotografía de perfil. Además, podrá generar su QR personal y único para compartir con los demás usuarios.
- RF8. Tendrá un buscador que permitirá buscar por usuarios.
- RF9. Se podrá tener una conversación privada con distintos usuarios.
- RF10. Para agregar a un usuario o seguirlo, se podrá buscar su nombre de usuario o simplemente escanear el QR que puede mandar o mostrar por su pantalla.
- RF11. El sistema de creación de cuentas e identificación, será propio y además debe implementar los sistemas de autorización de Google y Twitter.
- RF12. A la hora de reservar se seleccionará la instalación, los extras y realizar el pago electrónico.

RF13. Para entrar a la instalación se presentará el QR de ficha de usuario. Quedando memorizado la entrada y salida.

RF14. Se podrá recibir notificaciones en todo momento.

RF15. Poder restablecer la contraseña del usuario.

Los requisitos del sistema del administrado son:

RF15. Se podrá gestionar las instalaciones.

RF16. Se permitirá poner un precio fijado por tramos horarios.

RF17. Se podrá gestionar los extras por pistas.

RF18. Gestión las reservas.

RF19. Se podrá listar y gestionar los usuarios.

RF20. Se podrán crear informes de contabilidad o información económica por pistas.

Requisitos no funcionales

Los requisitos no funcionales representan restricciones o condiciones que el cliente impone a la aplicación, como el tiempo de entrega del programa, el lenguaje... Los requisitos son los siguientes:

RNF1. Todos los datos se almacenarán remotamente en un servidor.

RNF2. La autorización se hará utilizando el sistema de Firebase.

RNF3. Las aplicaciones móviles deben ser multiplataforma y funcionar en Android o iOS, utilizando Kotlin Native.

RNF4. Los datos se almacenarán en la base de datos de Firebase.

RNF5. El acceso al almacenamiento se hará a través del servicio de Firebase.

Requisitos de información

Los requisitos de información representan entidades e información con las que va a trabajar nuestra aplicación TenisClubDroid.

RI1.Club. Representa al club y tiene los datos del club como el nombre, la ubicación y álbum de fotos.

RI2.Usuario. Representa a la persona que va a interactuar con la aplicación y tiene datos del usuario como el nickName, contraseña, fotografía, descripción, correo electrónico y código QR.

RI3.Partido. Representa los objetos de los partidos que los usuarios van a jugar y guardan información como nickname y correo de ambos contrincantes, el resultado y fotografías del partido.

RI4.Pistas. Representan las pistas del club y tienen datos como los extras que se pueden elegir (luz, sistema de limpieza, tipo de suelo) , estado (disponible o no disponible) y precio.

RI5.Reserva. Representan las reservas realizadas por el usuario, tiene datos como la pista donde se va a jugar, los extras, los identificadores del usuario que ha realizado la reserva y del adversario (si existe), la fecha, el precio y un identificador de la reserva.

3.2. Análisis de escenarios

Casos de uso

En este apartado veremos los casos de uso del sistema, los cuales describirán una acción o actividad, dichos casos de uso son extraídos de los requisitos funcionales de nuestro sistema.

Actores del sistema

Los actores del sistema son todas las entidades externas al sistema e interactúa con el demandando tareas. Incluye a usuarios físicos y no físicos externos al sistema. Los actores de la aplicación son:



- **Usuario Android.** Se trata de un usuario normal con una cuenta registrada que interacciona con la aplicación Android para reservar pistas, quedar con contactos, etc.



- **Usuario no registrado.** Este tipo de usuario no podrá acceder a la aplicación Android hasta que no formalice una cuenta, tras ello se convertirá en un usuario Android.



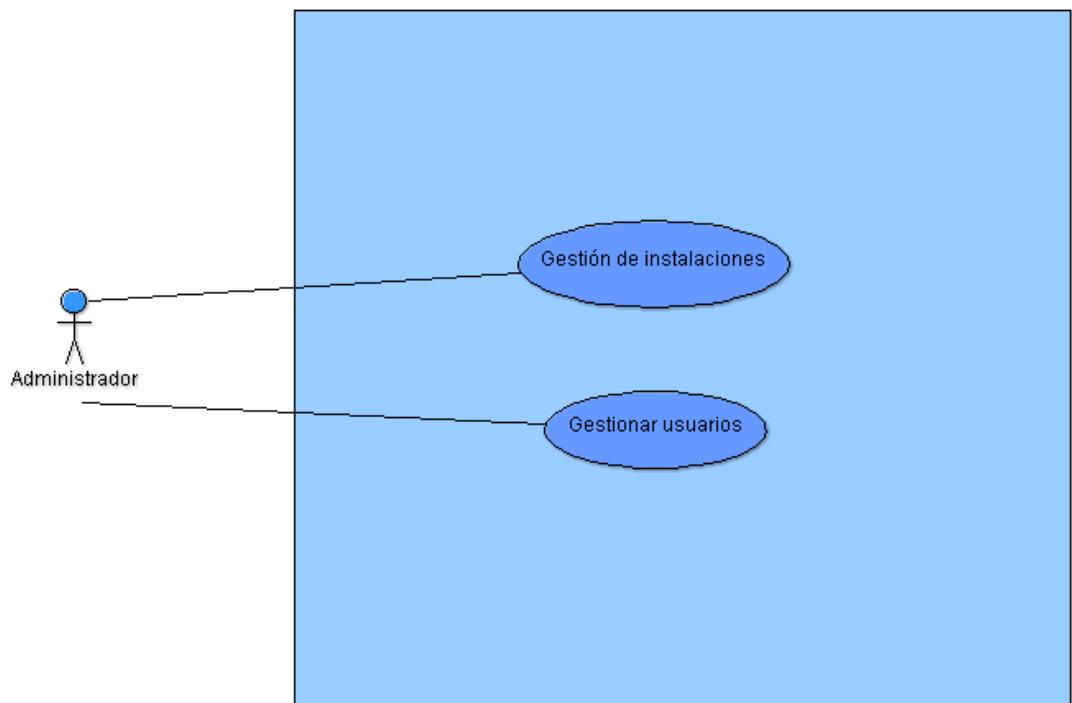
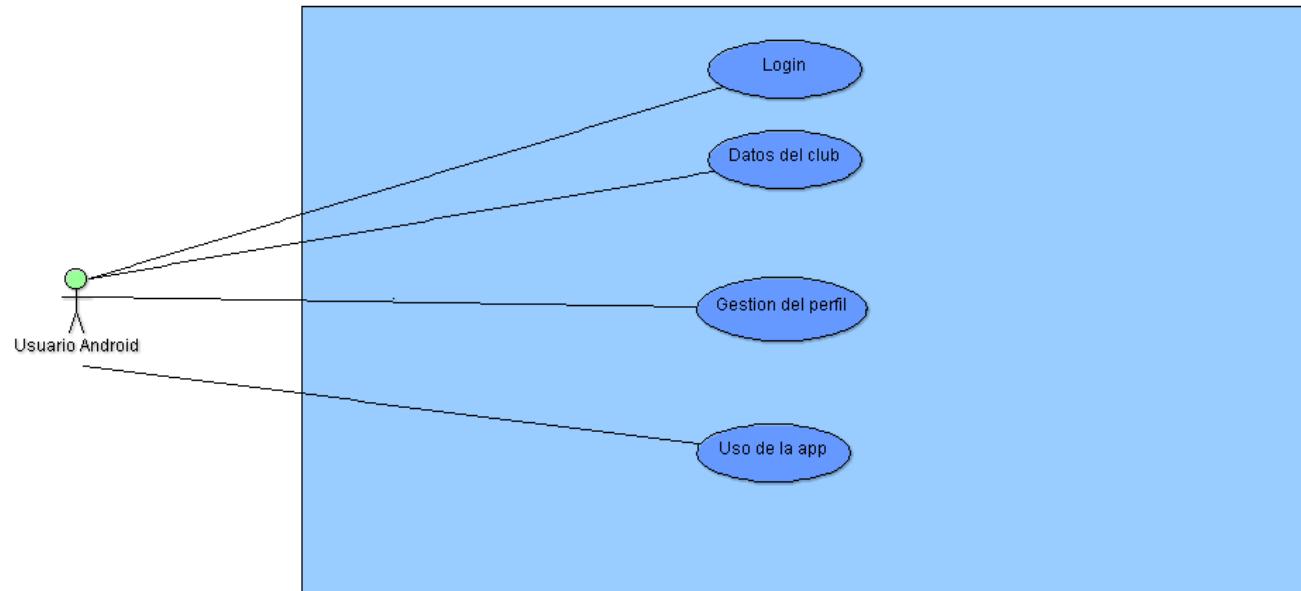
- **Administrador.** Este es un usuario especial que interactuará con la aplicación de administración para gestionar el sistema principal.

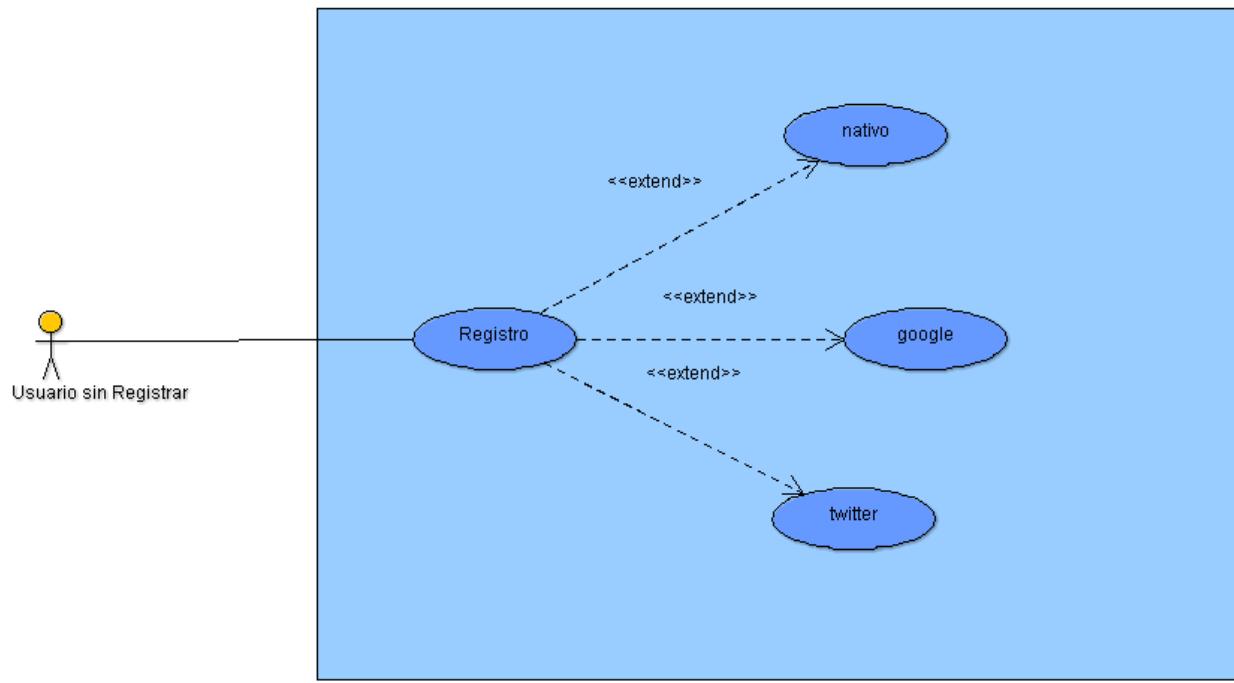
Diagramas de casos de uso

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

El diagrama de casos de uso describe actividades que deberán realizar los actores para realizar una tarea.

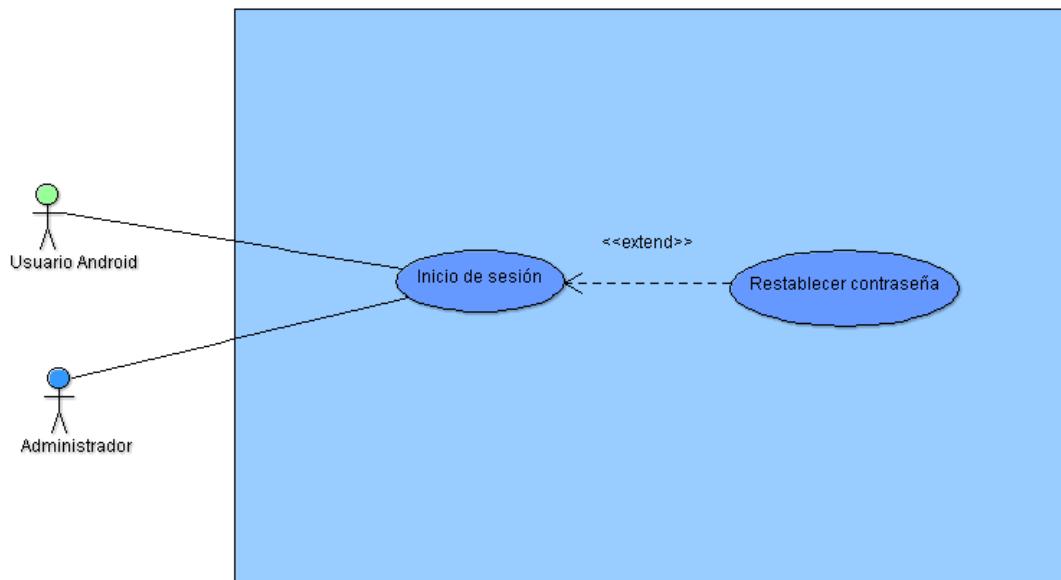
Principales subsistemas funcionales





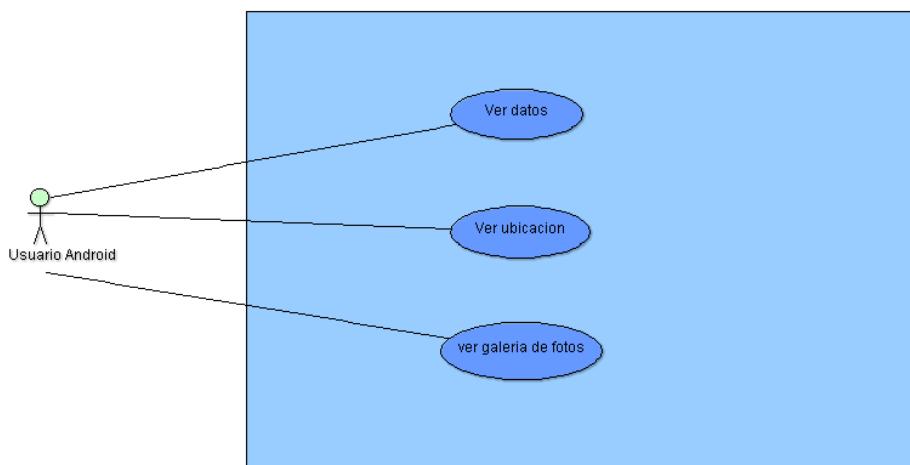
Login

Este subsistema va a estar presente tanto en el usuario Android como en el administrador, pues ambos tendrán que entrar a su aplicación dando sus datos, lo represento como un subsistema para que no se repita en cada uno de los diagramas.



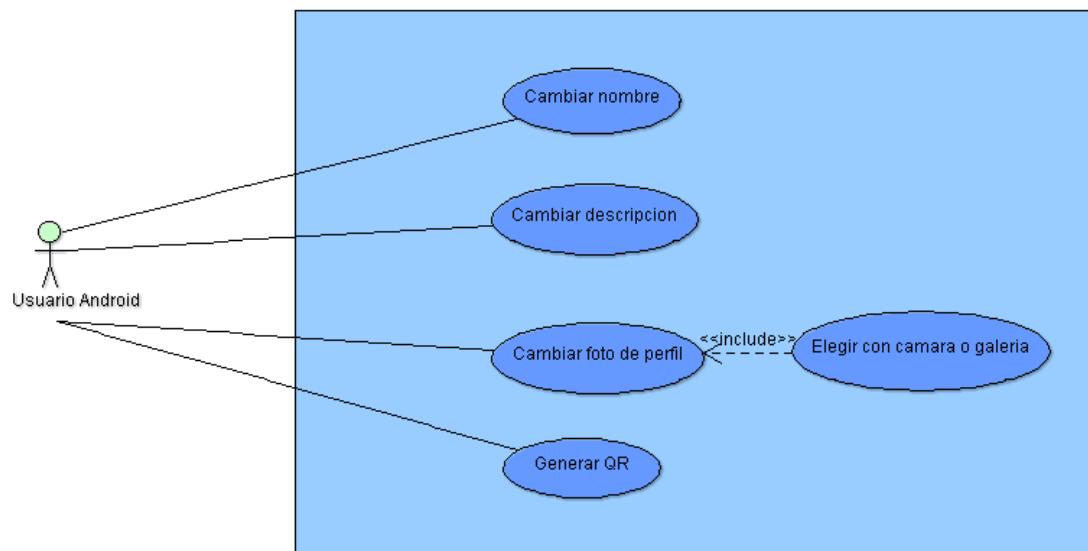
Datos del club

En este subsistema el usuario Android va a poder ver todos los datos del club, así como su localización para poder ir cómodamente con google maps y ver la galería de fotos del club.



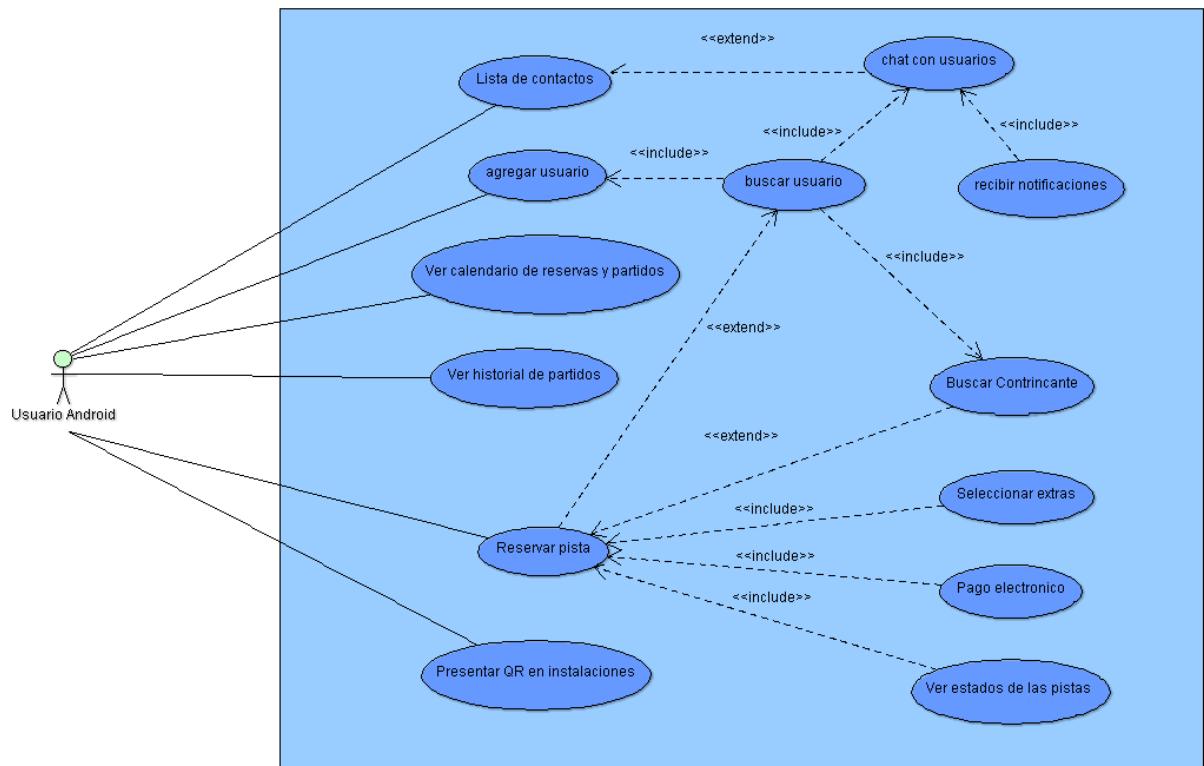
Gestión del perfil

En este apartado el usuario Android podrá cambiar su descripción, su nombre de usuario, su fotografía de perfil y generar su código QR.



Uso de la app

En este subsistema es donde el usuario va a poder consumir lo más importante de la aplicación y va a reservar pistas, buscar nuevos usuarios con los que quedar etc.



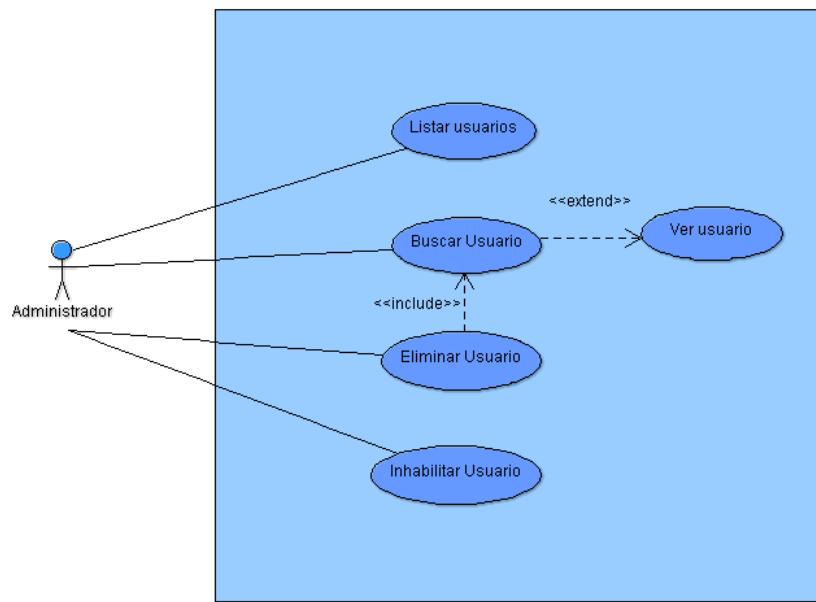
Gestión de instalaciones

En este subsistema el administrador podrá gestionar el estado de las instalaciones ya que pueden no estar disponibles por reformas o mejoras, también va a gestionar el precio fijado por tramos horarios de cada pista y el precio de sus extras como la luz, sistema de limpieza... Y para finalizar ver informes de contabilidad o información económica por cada pista.



Gestionar usuarios

En este subsistema el administrador podrá ver cómodamente todos los usuarios registrados y gestionarlos.



4.DISEÑO DE LA SOLUCIÓN

4.1. Diseño de la interfaz de usuario y prototipos

En este apartado vamos a ver el diseño de la interfaz de usuario y los prototipos de las dos aplicaciones, la primera para dispositivos móviles y la segunda para PC para la administración de la app móvil.

Ambas usan una paleta de colores basadas en el azul, para el dispositivo móvil se ha intentado diseñar siguiendo la metodología de material design, dejando pantallas lo más limpias posibles y sin demasiada información que atosigue al usuario. Para la aplicación de administración se ha intentado hacer lo más útil posible y un poco menos decorada ya que está más enfocada a la administración y a la eficiencia.

El logo de la aplicación es el siguiente:



El tipo de letra usada en todo el proyecto ha sido: Sans Serif

**ABCDEFGHIJKLMN
OPQRSTUVWXYZÀ**
abcdefghijklmnopqrstuvwxyz

Ahora veremos las pantallas de los prototipos explicadas con la herramienta JustInMind.

- **Aplicación móvil.**

Lo primero que verá el usuario al iniciar la aplicación será este SplashScreen con el logo y nombre de la app.



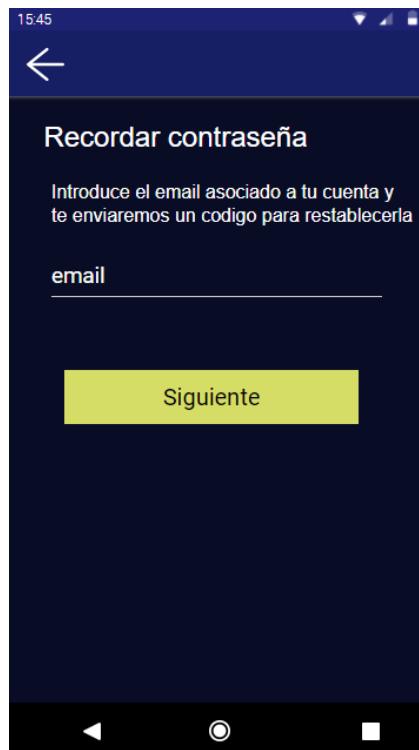
Tras esto, si el usuario tenía una sesión iniciada pasará directamente a la página principal de la aplicación, en el caso contrario tendrá que iniciar sesión con los datos de su cuenta o entrando con su cuenta de Twitter o Google.

-Esta pantalla hace referencia al caso de uso de login



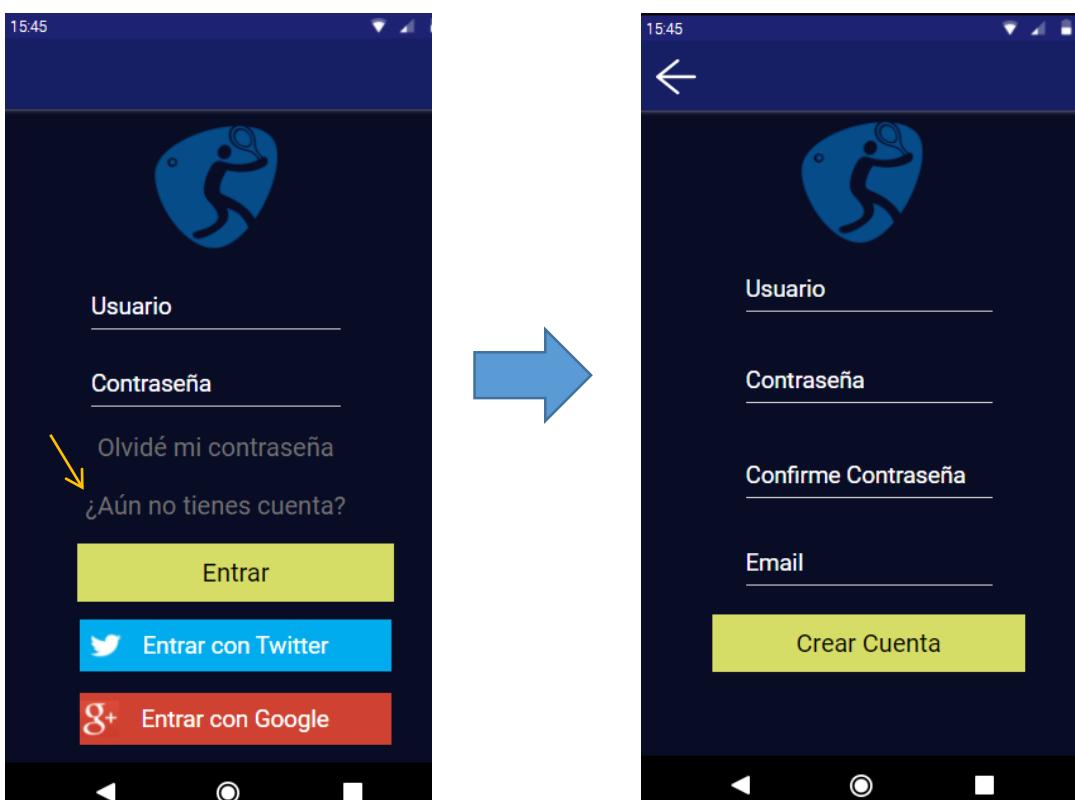
En el caso de que el usuario no recuerde su contraseña será dirigido a la siguiente ventana, donde siguiendo el proceso de restablecer su contraseña podrá cambiarla y será enviado de nuevo al login.

-Esta pantalla hace referencia al caso de uso de Restablecer contraseña



Cuando se trate de un nuevo usuario y no quiera entrar con sus cuentas de twitter o google podrá crearse una cuenta dándole a la opción del login.

-Esta pantalla hace referencia al caso de uso de registro



Esta será la página principal de la aplicación, tiene el nombre, una foto de un tenista y una pequeña bienvenida.



Presionando el ícono de las 3 barras aparecerá el menú lateral por el que podremos movernos por toda la aplicación.



Ahora iremos viendo las distintas partes de la app siguiendo el menú lateral ya comentado.

Club

-Esta pantalla hace referencia al caso de uso de datos del club (con ver nombre, galería de imágenes y ubicación).

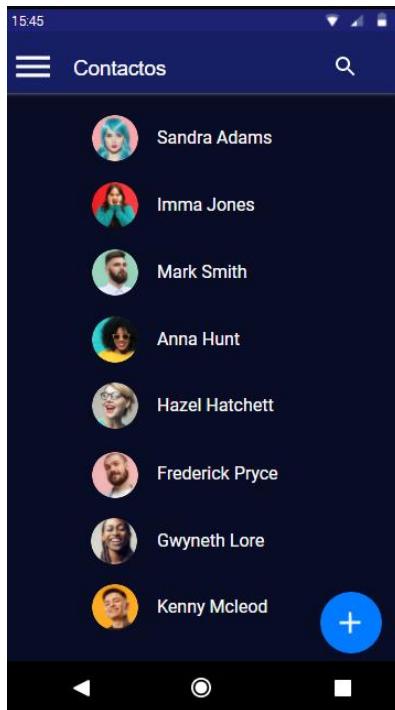
En esta ventana veremos el nombre del club, *Club de ocio Nudos*, su galería de fotos que podremos ver deslizándolas y finalmente un mini mapa que al pulsar en él nos llevara a google maps para poder ir cómodamente.



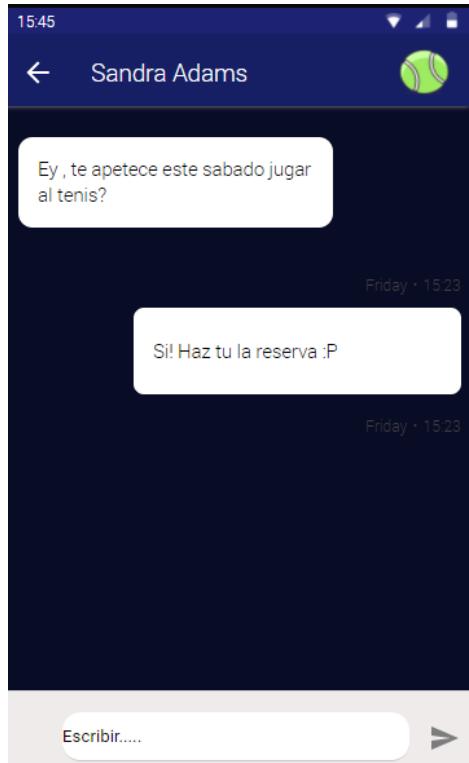
Contactos

-Esta pantalla hace referencia a los casos de uso: listar contactos, buscar usuario, agregar usuario y tener un chat con usuarios y recibir notificaciones

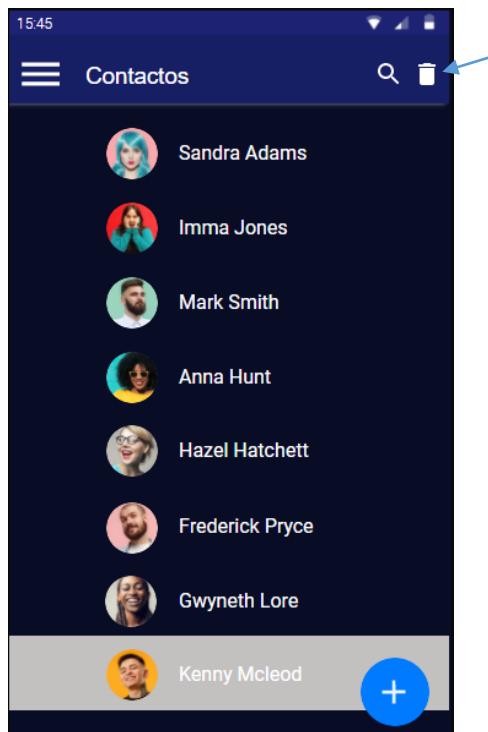
En el apartado de Contactos podremos ver los usuarios que tenemos guardados para poder hablar y quedar para jugar.



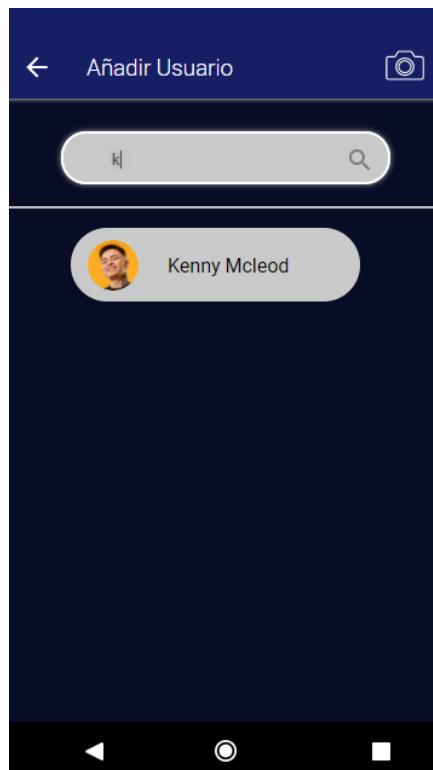
Al pulsar en ellos nos llevará a nuestro chat privado donde podremos hablar e incluso empezar el proceso de reservar una pista. Mediante el campo de escribir y el botón de enviar podremos mandarnos mensajes. Al pulsar sobre el ícono de la pelota de tenis nos lleva directamente a hacer una reserva rellenando al contrincante con el del chat.



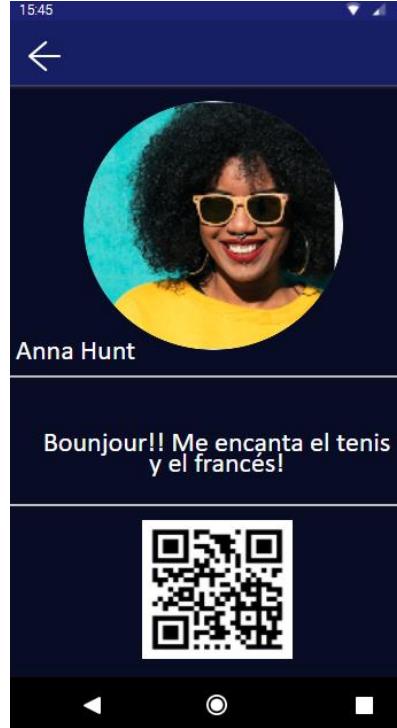
Continuando en la zona de contactos, al mantener pulsado sobre un usuario nos saldrá la opción de eliminarlo.



Después de eliminar a un usuario, si nos arrepentimos o si queremos añadir uno nuevo, podemos pulsar el botón flotante del + para ir a la siguiente pantalla. Donde podremos buscar al usuario por su nombre o mediante su QR



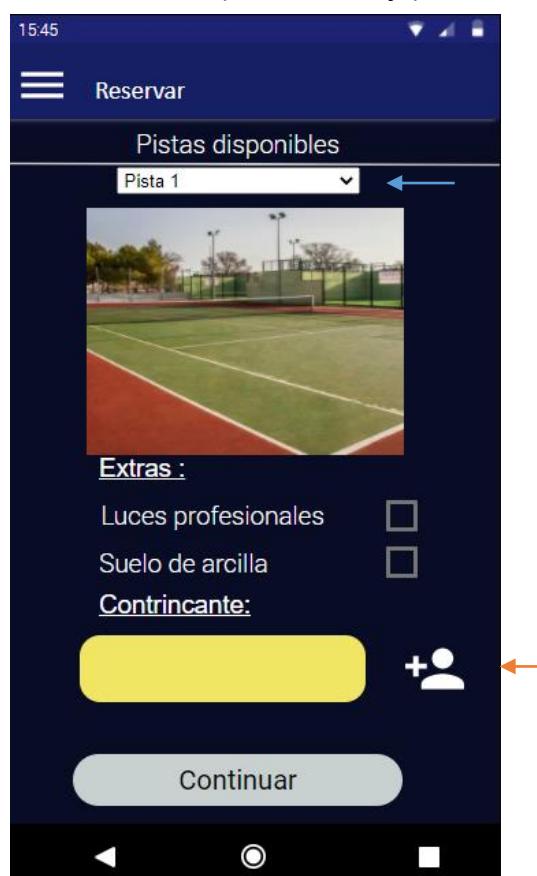
Para finalizar con los contactos, al pulsar sobre la imagen de ellos podremos ver su perfil.



Reservar

-Esta pantalla hace referencia al caso de uso de reservar una pista, con los casos de uso incluidos de elegir extras, pago electrónico, ver estado de las pistas (en tiempo real) y el caso de uso opcional de buscar un contrincante.

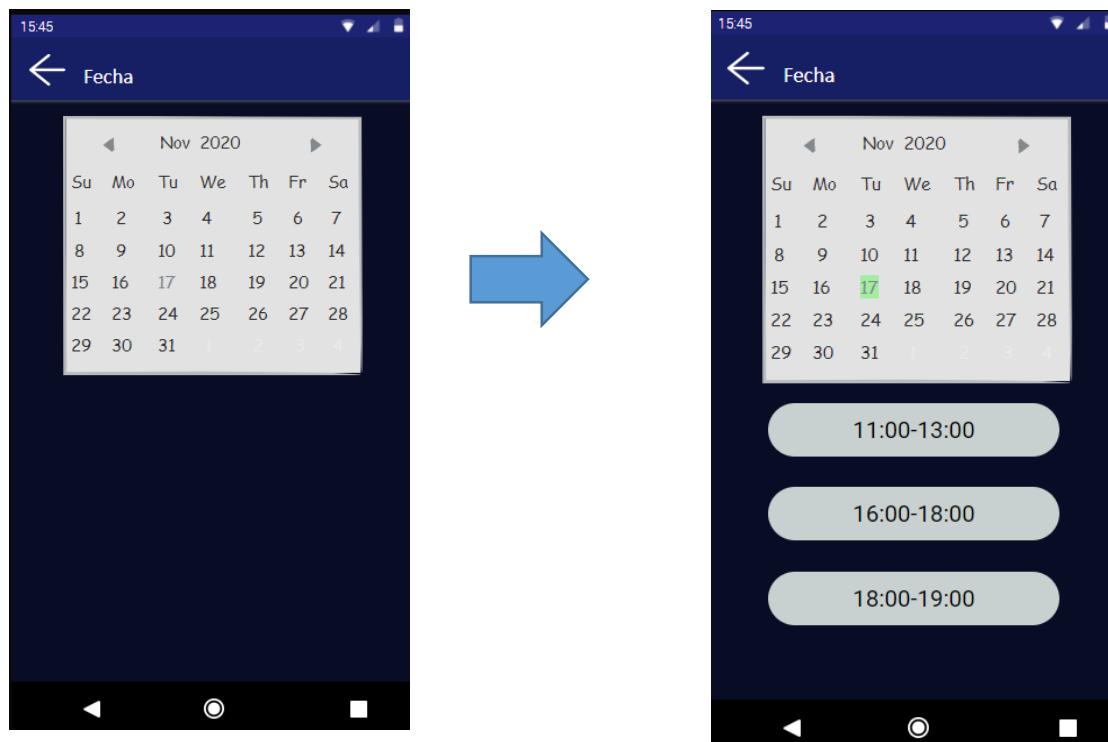
En este apartado se llevarán a cabo las reservas de las pistas, primero podrá elegir la pista (entre las que estén disponibles) mediante el desplegable (flecha azul), apareciendo su foto y sus extras y opcionalmente añadir un usuario contrincante, para ello pulsando el botón de añadir usuario (flecha naranja).



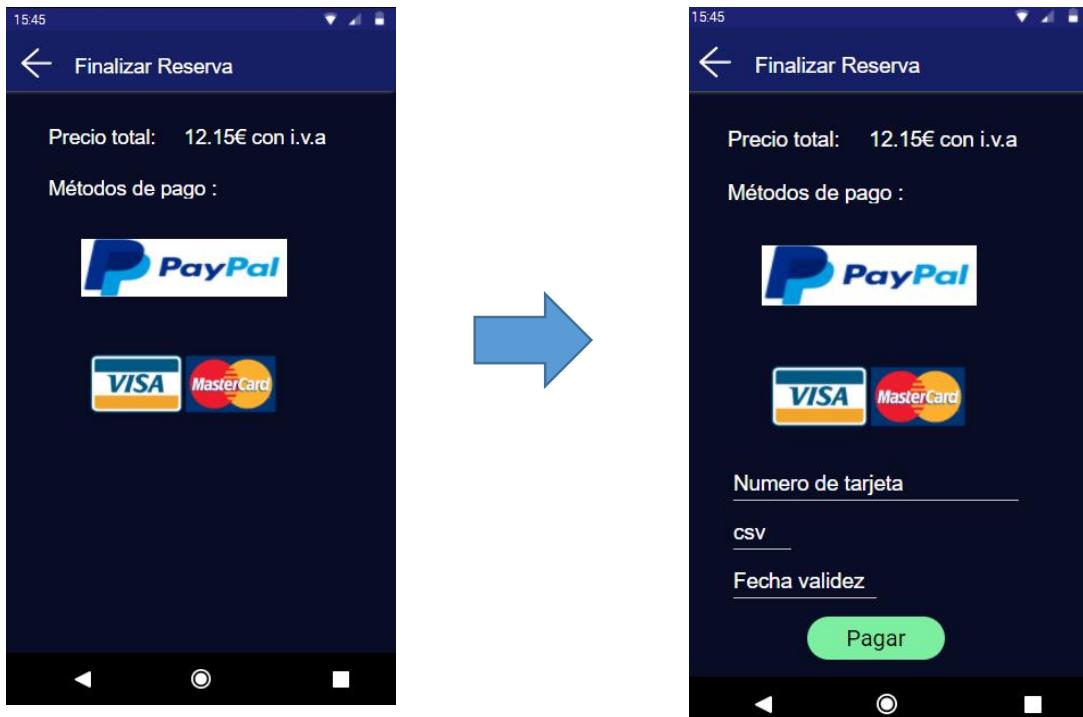
En caso de que haya venido a la reserva mediante el chat con un usuario se rellenará automáticamente, en caso contrario la pantalla de buscar usuario es la siguiente:



Tras la primera ventana podrá elegir la fecha y la hora (entre las disponibles):



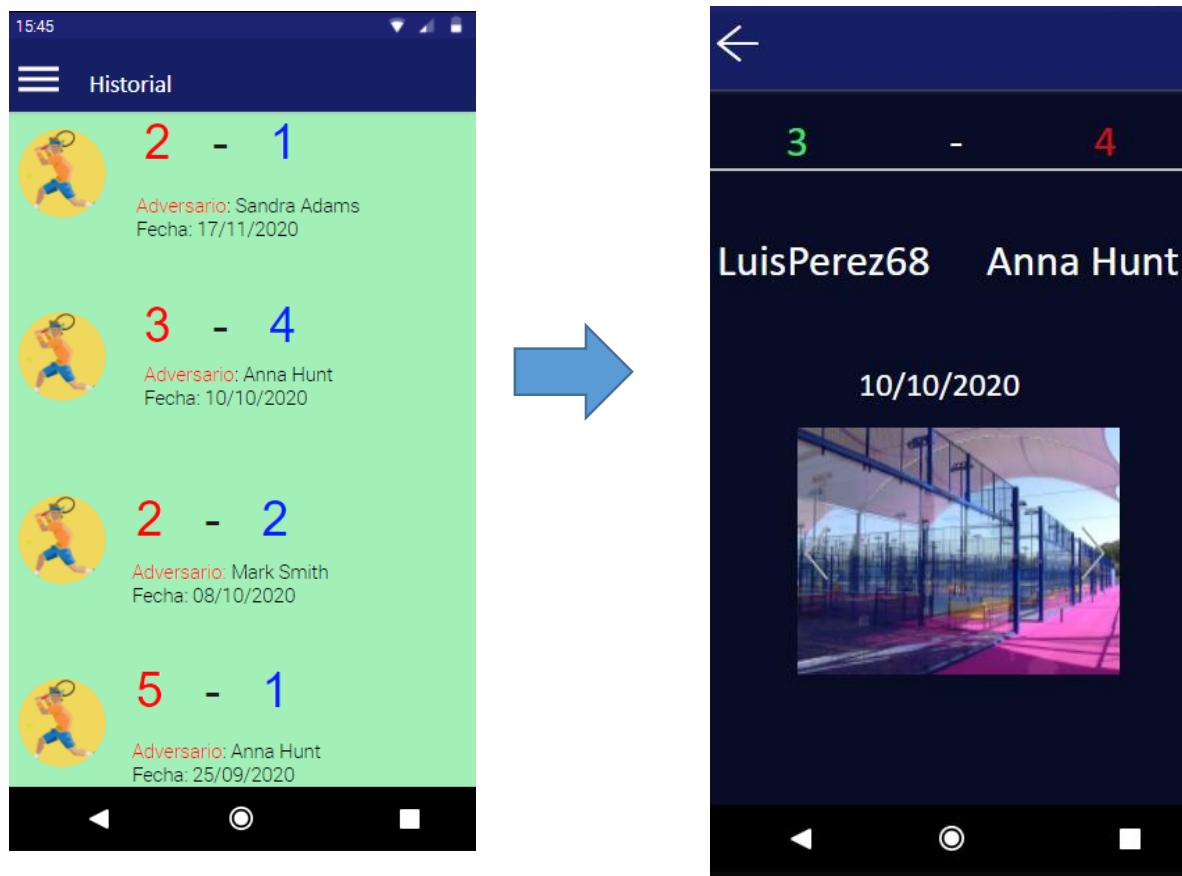
Una vez elegida la fecha y hora se pasará a la ventana de pago, donde se ve el precio total y se dan dos opciones de pago, mediante PayPal, al pulsar sobre él nos llevará a PayPal para hacer la transacción o mediante tarjeta visa o MasterCard, donde nos mostrará los campos para rellenar, una vez pagado nos llevará a la página principal.



Histórial

-Esta pantalla hace referencia al caso de uso de ver el histórico de partidos

En este apartado aparecerán todos los partidos que usuario ha jugado, estarán en forma de lista (Card View), donde a simple vista podrá verse el resultado, siendo el rojo el contrincante y azul el usuario, el nombre del adversario y la fecha. Para ver el partido junto con las fotos tomadas pulsaremos sobre él.

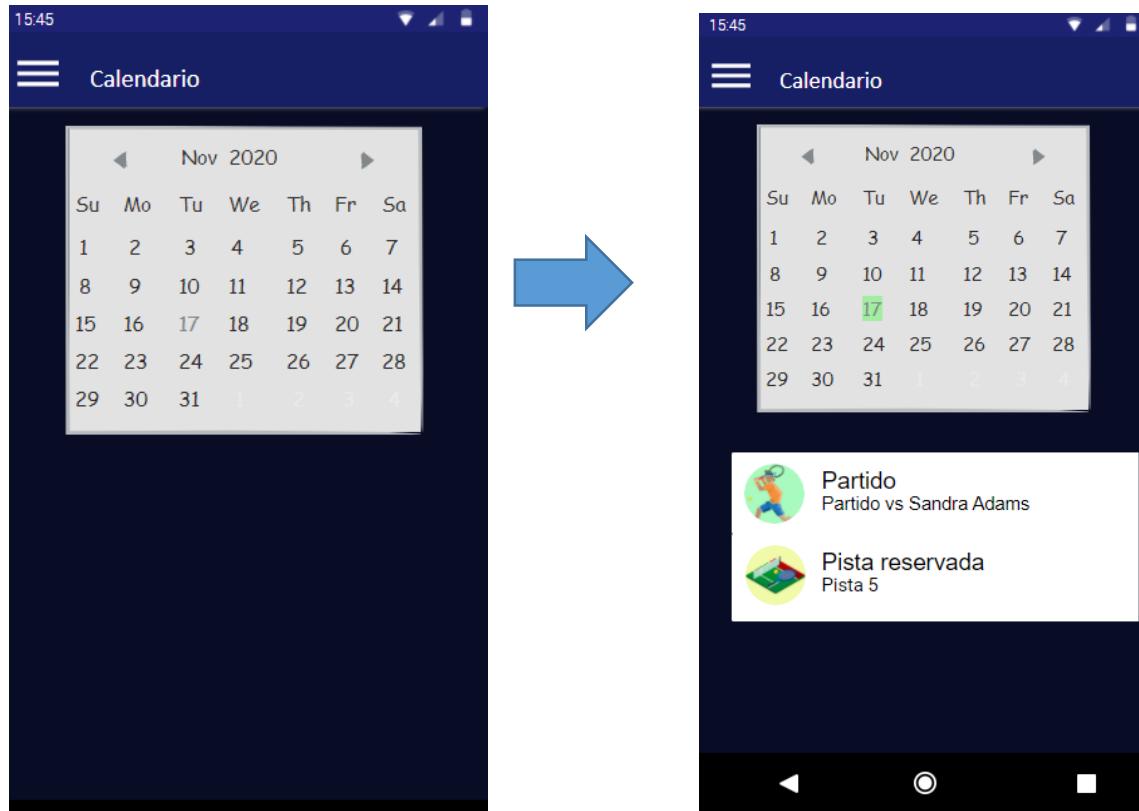


Las fotos, así como la información del partido será rellenada por el usuario cuando le llegue una notificación en la app una vez haya concluido su reserva de la pista y podrá poner la puntuación de los sets y subir fotos mediante su cámara o desde la galería.

Calendario

-Esta pantalla hace referencia al caso de uso de ver calendario de reservas y partidos

En este apartado el usuario tendrá un calendario con los partidos o reservas que tenga, para verlos deberá pulsar en el día del calendario y le aparecerán.



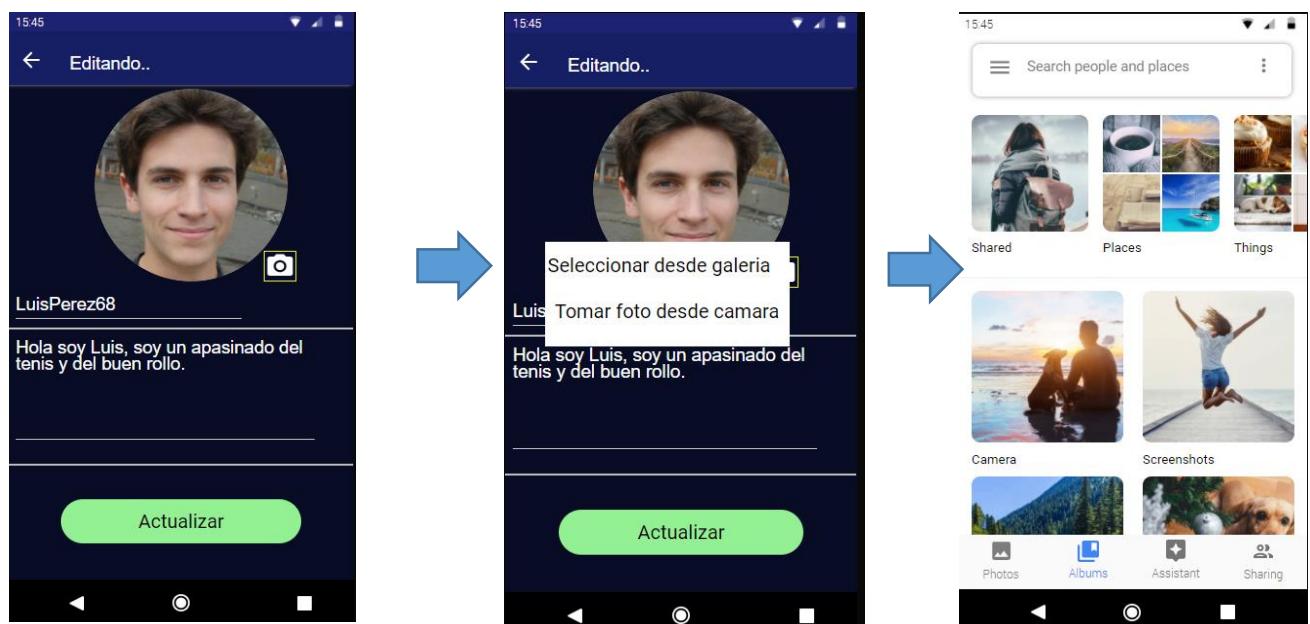
Perfil

-Esta pantalla hace referencia a los casos de uso de cambiar nombre de usuario, cambiar descripción , cambiar foto de perfil , generar qr y elegir con cámara o galería.

En el perfil el usuario podrá ver su foto, su nombre de usuario, su descripción y pulsando en el botón de código QR ver su código QR para enseñárselo a otro usuario. En esta pantalla también está la opción de cerrar sesión (al lado de lápiz)



Para editar el perfil del usuario pulsará el ícono del lápiz y le llevará a la siguiente pestaña, donde pulsando el botón de la cámara le dará a elegir entre elegirla desde la galería o directamente con la cámara, cambiar su nombre (si no está repetido) y modificar su descripción.

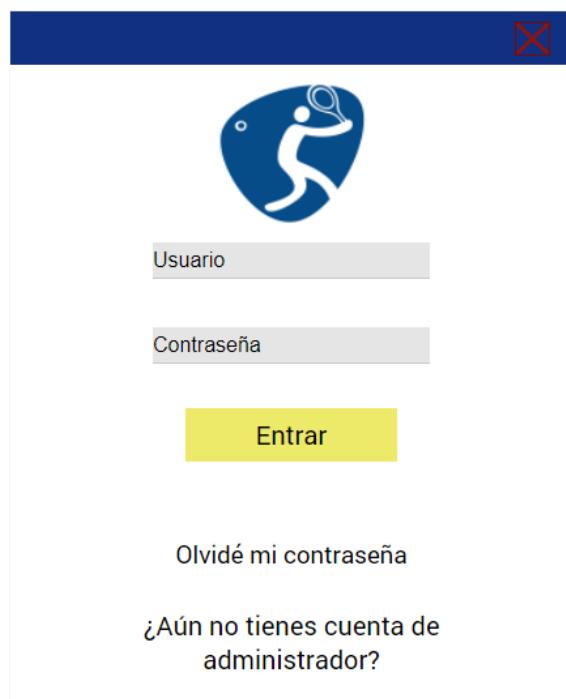


- **Aplicación Administración.**

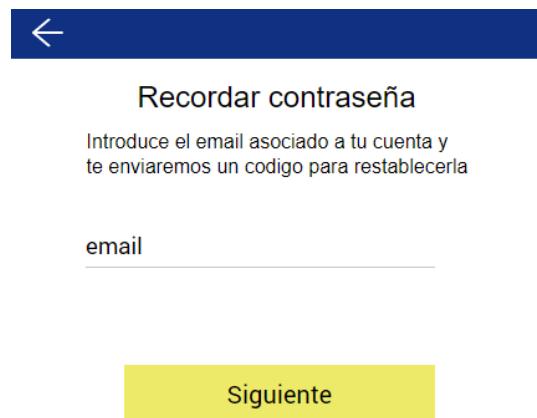
Esta aplicación es la de la administración y solo será usada por usuarios específicos.
Ahora veamos sus pantallas:

Login

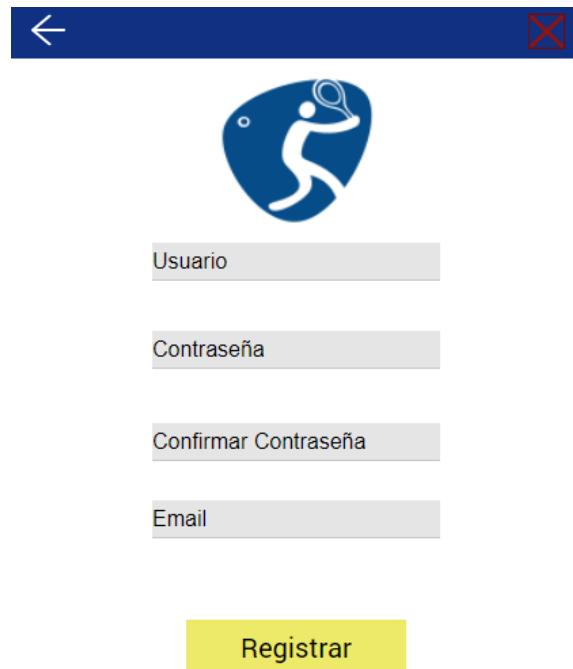
Al igual que en la app móvil el usuario se encontrará con un login en el que introducir sus datos de acceso.



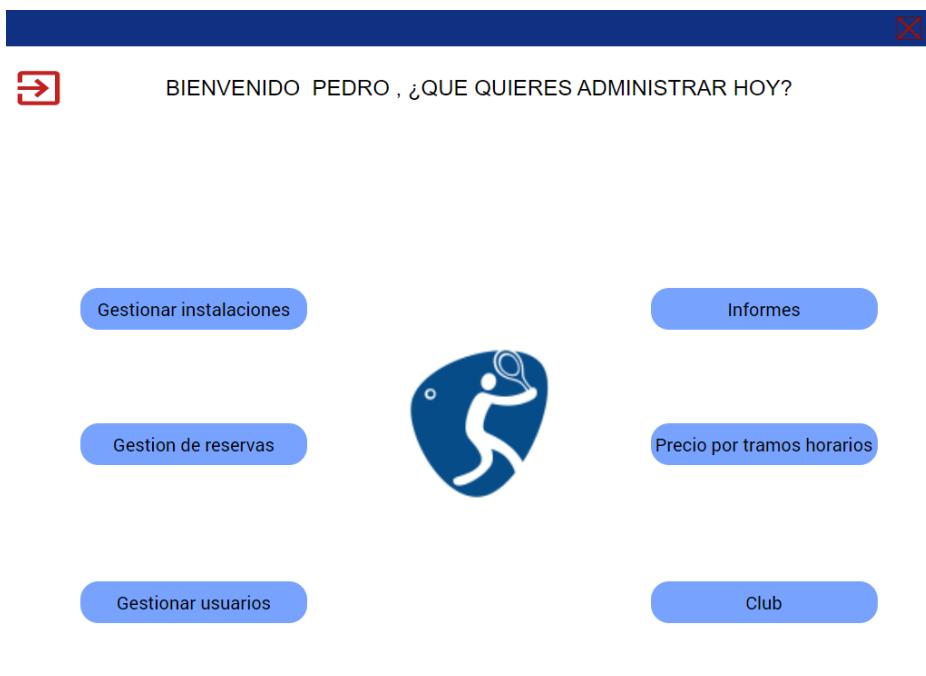
En caso de haber olvidado su contraseña pulsará en la opción correspondiente y seguirá los pasos (los mismos que en la app móvil).



Si aún no tiene cuenta como administrador podrá crearla fácilmente pulsando en la opción correspondiente.



Una vez introduzca de forma correcta sus datos de acceso pasará a la página principal de la aplicación.



Esta pantalla tiene los accesos (mediante botones) a las distintas partes de la aplicación. Para cerrar su sesión podrá darle al botón de la izquierda rojo.

Gestionar instalaciones

-Esta pantalla hace referencia a los casos de uso de crear pista, modificar pista, eliminar pista, ver pista, buscar pista y modificar precio de extras

<X

Gestionar Instalaciones

Q

Crear Pista
Borrar Pista

Nombre	Foto	Estado	Precio base/h	
Pista 1		Disponible	12€	<input type="checkbox"/>
Pista 2		Disponible	10€	<input type="checkbox"/>
Pista 3		No Disponible	14€	<input type="checkbox"/>
Pista 4		Disponible	14€	<input type="checkbox"/>
Pista 5		No Disponible	9€	<input type="checkbox"/>
Pista 6		Disponible	10€	<input type="checkbox"/>

En este apartado se mostrarán todas las pistas que tiene el club, mostrando su nombre, foto, estado y precio base por hora. Tiene un buscador en caso de que tenga demasiadas pistas y así poder buscar la que se desea de forma rápida. Ahora veremos las distintas funciones de esta pantalla:

- Para crear una **nueva pista** pulsaremos en el botón de “Crear Pista” y aparecerá una ventana de creación, podrá ponerle un nombre, una foto (al pulsar sobre el botón de cambiar saldrá el explorador de Windows), el estado (disponible o no disponible), su precio base, y añadir extras con su precio.

Gestionar Instalaciones

Nombre	Foto	Estado	Precio base/h	Extras
Pista 1		Disponible	€	
Pista 2		Disponible	€	
Pista 3		Disponible	€	
Pista 4		Disponible	€	
Pista 5		Disponible	€	
Pista 6		Disponible	10€	

Buscar pistas

NOMBRE:

FOTO:

Cambiar

Estado: Disponible

Precio base/h : €

Extras:

Extra	Precio	Añadir
-------	--------	--------

Eliminar

Crear Pista

Borrar Pista

- Para **ver** o **modificar** una pista pulsaremos sobre el nombre de la pista deseada, al hacerlo podremos ver los datos sin más o modificarlos y pulsar el botón verde para guardar los cambios.

Gestionar Instalaciones

Buscar pistas	NOMBRE: PISTA 1
Nombre	FOTO:
Pista 1	
Pista 2	Cambiar
Pista 3	Estado: Disponible ▾
Pista 4	Precio base/h : 12
Pista 5	Extras:
	Extra Precio
	Luz profesional 2 <input type="checkbox"/>
	Limpeza extrema 4 <input type="checkbox"/>
	Añadir Eliminar

- Para **eliminar** una o varias pistas marcaremos su casilla (flecha) y pulsaremos **borrar pista**.

Gestionar Instalaciones

Buscar pistas 

Nombre	Foto	Estado	Precio base/h	
Pista 1		Disponible	12€	<input checked="" type="checkbox"/> 
Pista 2		Disponible	10€	<input type="checkbox"/> 
Pista 3		No Disponible	14€	<input type="checkbox"/> 

Crear Pista 
Borrar Pista 

Gestión de reservas

-Esta pantalla hace referencia a crear informes de reservas

Gestión de Reservas

Filtros:

<input type="button" value="Fecha Inicio"/>	<input type="button" value="Fecha Final"/>	<input type="button" value="Pista"/>	<input type="button" value="Precio <"/>	<input type="button" value="Precio >"/>	<input type="button" value="Ver"/>	<input type="button" value="Crear PDF"/>
---	--	--------------------------------------	--	--	------------------------------------	--

Pista	Usuario	Extras	Fecha	Precio
-------	---------	--------	-------	--------

En esta pantalla se podrán ver y crear PDF sobre las reservas de la aplicación, para ello crearemos nuestro filtro personalizado pudiendo filtrar entre fechas, por pistas y por precio inferior o superior. Una vez tengamos el filtro hecho pulsaremos “ver” y si hay

reservas que coincidan con el filtro se rellenará la tabla, una vez rellena se podrá pasar a PDF.

Pista	Usuario	Extras	Fecha	Precio
Pista 5	LuisPerez68	No	25/10/2020	12
Pista 1	SandraAdams	Luz profesional	01/11/2020	15

Gestión de usuarios

-Hace referencia a los casos de uso de listar usuarios, buscar usuarios, eliminar usuarios, ver usuarios e inhabilitar usuarios.

Foto	Correo	Nombre	Contraseña	
	luisperez68@gmail.com	LuisPerez68	a6f9sd4sadf9fgÑFY	<input type="checkbox"/>
	annaHunt@gmail.com	AnnaHunt	Psd8sd4sadf9fgS5s	<input type="checkbox"/>
	kenny123@gmail.com	KennySmith	789sd4sadf9fgSFD	<input type="checkbox"/>

< 1 >

En este apartado se podrán ver a todos los usuarios de la aplicación, podrá ir pasando entre las páginas de usuarios con las flechas de abajo o buscar al usuario que quiera

con el buscador. Se podrá mirar el correo, nombre y una contraseña resumida del usuario. Activando el checkbox del usuario se podrá borrar o inhabilitar del sistema.

Informes

-Hace referencia a crear informes de pistas

The screenshot shows a dark blue header with a back arrow and a close button. Below it, the word "Informes" is centered. A "Filtros:" section contains two dropdown menus: "Pista 1" and "Todos". To the right are two buttons: "Ver" and "Crear PDF". Below this is a table header row with columns labeled "Pista", "Usuario", "Fecha", and "Precio".

De forma similar a las reservas en este apartado se podrán crear informes de contabilidad, el filtro aquí será la pista y el mes (pudiendo elegir todo el año), una vez hecho el filtro se podrá ver en la tabla cada registro de la pista con el usuario, fecha y precio y después crear un PDF.

The screenshot shows a dark blue header with a back arrow and a close button. Below it, the word "Informes" is centered. A "Filtros:" section contains two dropdown menus: "Pista 2" and "Febrero". To the right are two buttons: "Ver" and "Crear PDF". Below this is a table showing four rows of data, each with columns "Pista", "Usuario", "Fecha", and "Precio".

Pista	Usuario	Fecha	Precio
Pista 2	LuisPerez68	25/02/2020	12
Pista 2	SandraAdams	01/02/2020	15
Pista 2	Kenny Smith	01/02/2020	12
Pista 2	LuisPerez68	08/02/2020	10

Precio por tramo horario

Hace referencia al caso de uso de poner precio fijado por tramos

Fijar Precio por tramos horarios

Inicio	Fin	Precio €
12:00	13:00	5
16:00	18:00	8
20:00	21:00	9

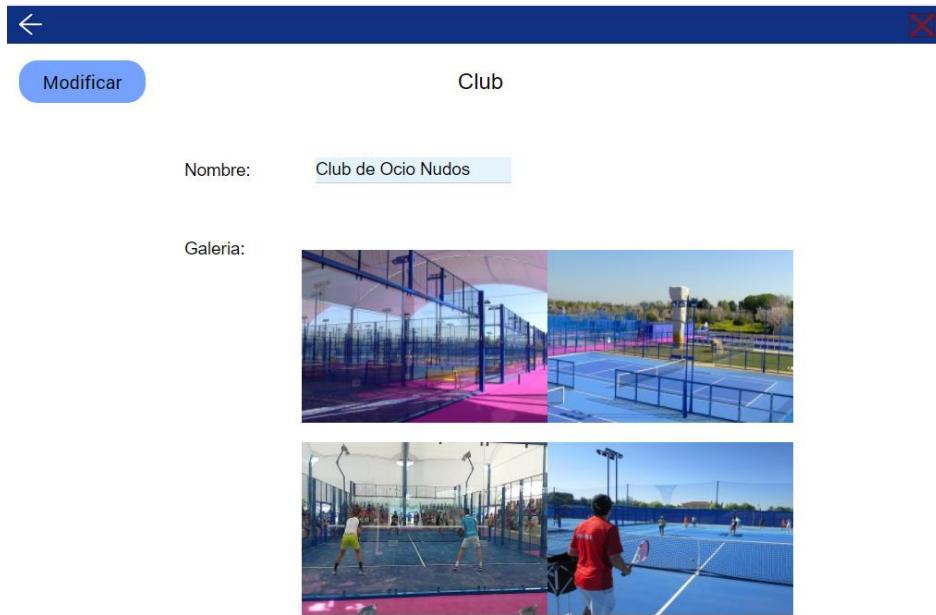
Añadir Tramo



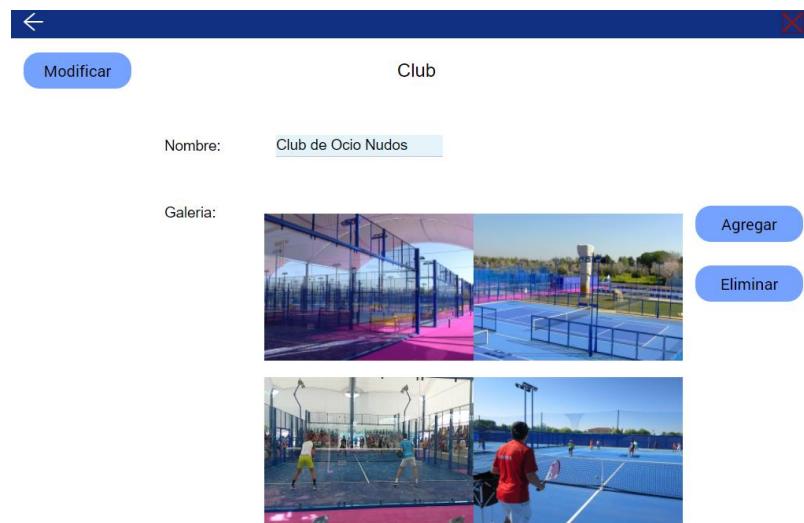
En este apartado el administrador podrá marcar unos precios dependiendo del tramo horario en el que se encuentre la reserva para encarecerla. Para crear un tramo horario elegirá una hora inicial, una final y el precio, al pulsar el botón de añadir se añadirá a la tabla y se aplicará en la aplicación móvil.

Club

-Hace referencia a los casos de uso de crear club y modificar club



En este apartado el administrador podrá modificar los datos del club o simplemente verlos. Para modificarlos pulsará el botón de modificar para hacer todos los campos editables y aparecerán dos nuevos botones para añadir o eliminar fotografías del club, una vez modificado para poder guardar los cambios volveremos a pulsar el botón de modificar.

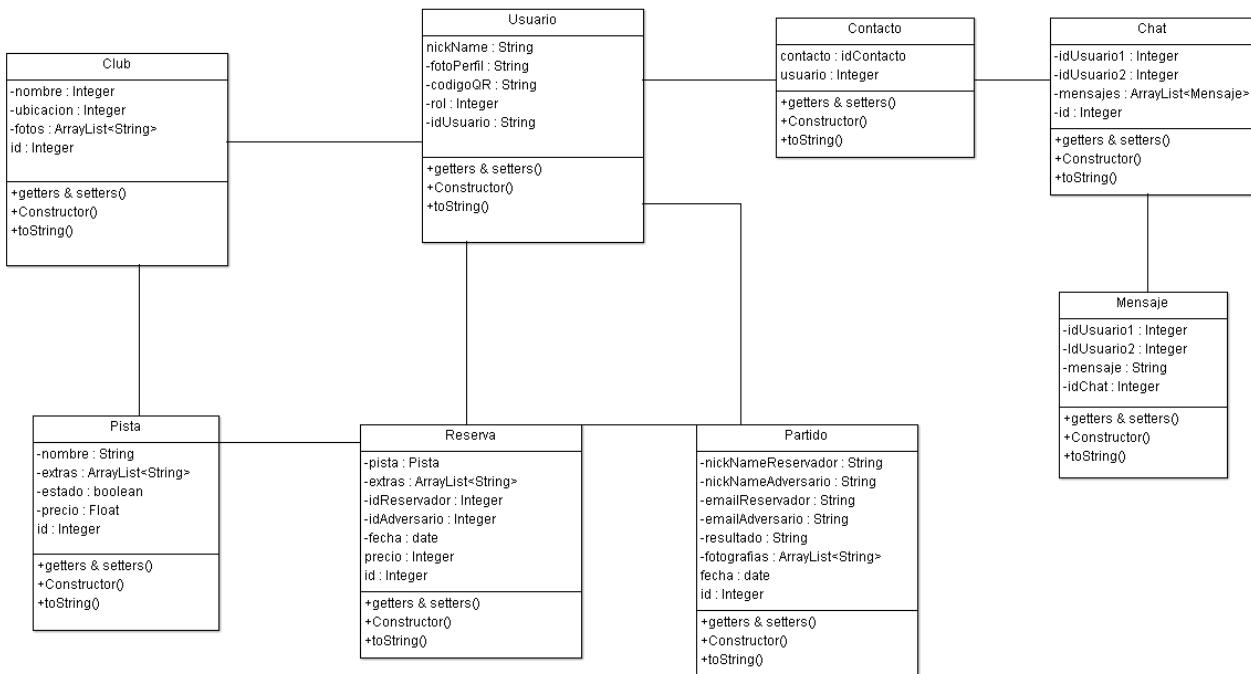


Ubicacion: 38°58'26.9"N 3°57'33.4..."

4.2 Diseño de la persistencia de la información

Diagrama de Clases

Con los requisitos de información y los casos de uso se relacionan las entidades importantes en clases. De este modo se muestra la estructura del sistema con sus clases, atributos, métodos y relaciones entre ellos. El diagrama de clases es el siguiente:



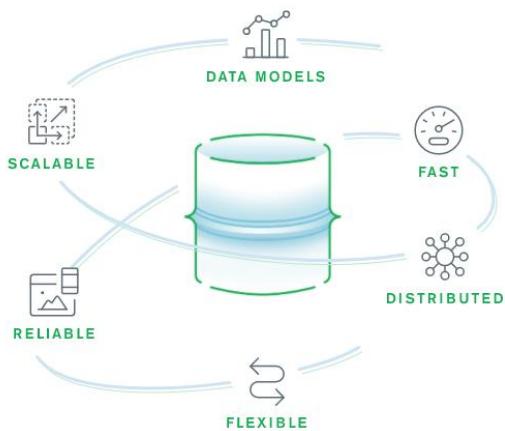
Con el diagrama podremos ver las distintas relaciones entre las clases, que son:

- Un club tiene un nombre, una ubicación (guardando su latitud y longitud) y una serie de fotografías, además tiene distintas pistas y además tiene unos usuarios.
- Un usuario (que guardara datos como su nombre de usuario, fotografía, email...) pertenece a un club, puede hacer reservas (donde elige una pista) y ver sus partidos (que pertenecen a una reserva), además puede tener una serie de contactos (otros usuarios que podrá buscar mediante nombre de usuario o utilizando su código qr) y tener un chat con ellos para mandar y recibir mensajes.

Diseño de la persistencia de la información

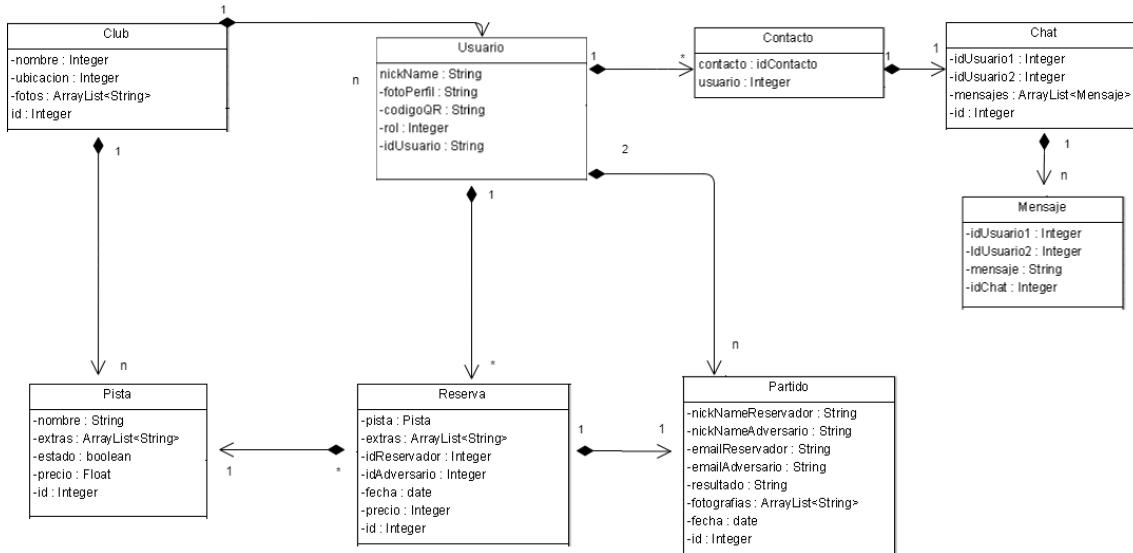
Con los requisitos de información y el diagrama de clase podemos extraer las entidades que necesitan ser persistentes. En este proyecto voy a utilizar un modelo no relacional, es decir NoSQL, utilizando Firebase como base de datos, los motivos de mi elección son los siguientes beneficios:

The benefits of NoSQL database



- **Modelado de datos:** al no ser SQL no tiene que seguir sus reglas por lo que al modelar los datos se pueden adaptar más a los casos de usos y los requisitos de la aplicación. Por ejemplo, nuestro club tendrá un álbum de fotos que en un modelo SQL tendría que hacer uso de varias tablas y relaciones y en el modelo NoSQL se podrá simplemente usar una Lista de imágenes.
- **Rendimiento:** las bases de datos NoSQL suelen funcionar mejor que las bases de datos SQL, y en concreto **Firebase** puede realizar de forma sencilla la visualización de datos en tiempo real para la aplicación, y al estar asociada con Google le da un añadido de confianza en cuanto eficiencia y seguridad.
- **Flexibilidad:** al utilizar archivos JSON los campos se pueden variar de un documento a otro de una forma sencilla y de esta forma si con el tiempo se tienen que modificar no supondrá un desafío para el equipo de desarrollo como sí lo tendría al usar un modelo SQL. Además, los modelos NoSQL se adaptan mejor a las arquitecturas de microservicios (nuestro caso).

Una vez vista la justificación del modelo de persistencia de la información veremos el diagrama, al ser NoSQL será muy parecido al diagrama de clases.



Como se puede apreciar en el diagrama un club puede tener indefinidos usuarios e indefinidas pistas.

Un **usuario** podrá tener indefinidas reservas, estas **reservas** guardan el identificador del usuario que la ha realizado y además en caso de haber introducido un contrincante también guardara su identificador para acceder a sus datos. Además, un usuario podrá tener indefinidos **contactos** (o amigos) con los que poder quedar para jugar o para “chatear”, este **chat** guardará los identificadores de ambos usuarios que serán referencias para poder guardar los mensajes de forma correcta. Finalmente, el usuario podrá ver los **partidos** que ha jugado y ver la información de ellos.

La **reserva** tendrá un partido (el número de sets puede ser los que el usuario quiera, no tiene por qué atenerse a un número definido, de este modo puede ser un partido al mejor de 5 o X) y guardara una referencia a la **pista** que se ha elegido con sus datos.

El **partido** tendrá referencias del jugador (o jugadores) que ha jugado el partido y guardara datos como el email, el resultado, fotografías y la fecha.

4.3 Arquitectura del sistema

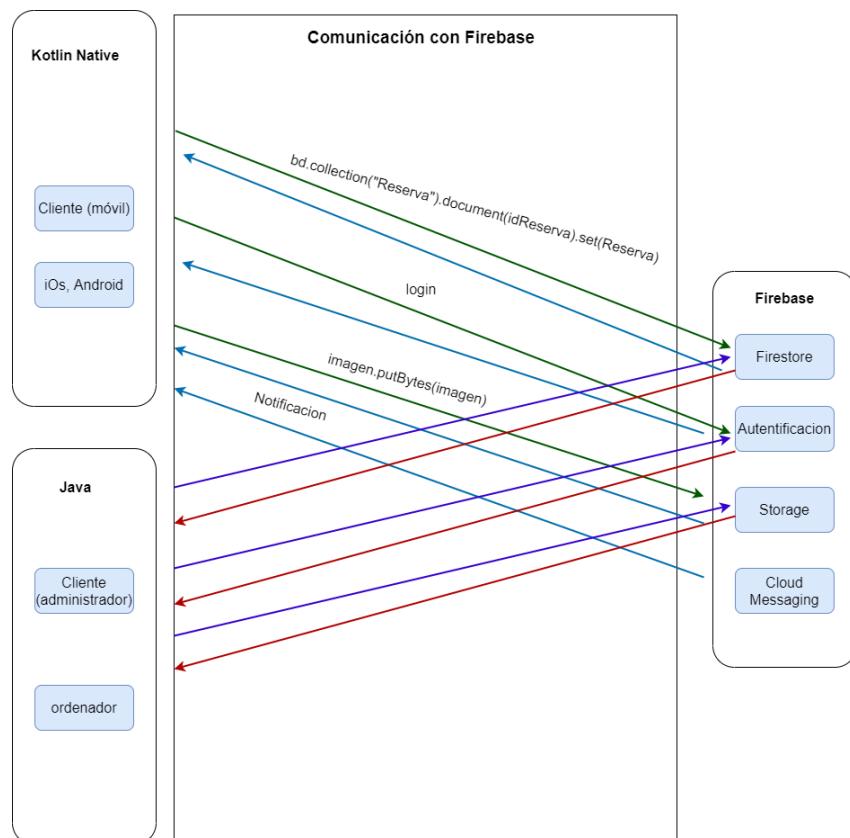
La arquitectura de un sistema de software es la organización o la estructura de los componentes importantes del sistema que interactúan mediante interfaces, con componentes compuestos de interfaces y componentes cada vez más pequeños.

En el caso de TenisClubDroid tendremos dos clientes diferentes, uno enfocado para el **usuario móvil** (iOs o Android) y otro cliente enfocado al **usuario administrador**, cada uno estará implementado con una tecnología diferente, el primero con Kotlin Native y el segundo con Java. Pero aunque sean distintos ambos comparten el uso de Firebase como servidor y base de datos. Al compartirlo ambos utilizarán los mismos métodos que la biblioteca de Firebase proporciona y de este modo podrá acceder a cada servicio que ofrece Firebase.

Por ejemplo, cuando el usuario (cualquiera de los dos tipos) quiera acceder a la aplicación hará una llamada al servicio de **Autenticación** de Firebase, en caso de que se quiera recuperar o crear una reserva llamará al servicio de **Firestore**, cuando quiera guardar las fotos del partido o su propia foto de perfil hará una petición al servicio de **Storage** y finalmente cuando al usuario le lleguen notificaciones de un chat hará uso del **Cloud Messaging**.

Diagrama de la arquitectura:

Aunque a simple vista parezca liso, es sencillo de entender teniendo en cuenta que aunque los dos clientes son diferentes comparten todos los métodos de Firebase. Las flechas verdes y moradas representan las peticiones del usuario y las azules y rojas la respuesta del servidor.



5. Implementación de la solución

5.1 Análisis tecnológico

En este apartado veremos un análisis de las posibles alternativas que se hubieran adaptado a las necesidades del proyecto, y al final veremos cuál ha sido la escogida y los motivos. Para ello veremos tanto las posibles tecnologías para el desarrollo de la aplicación móvil, como de la aplicación de administración y finalmente la del servidor.

- **Tecnologías para la aplicación móvil.**

Como un requisito de la solución era que fuera compatible tanto para sistemas iOS como para sistemas Android la mejor solución es una tecnología que permita desarrollar de forma hibrida para agilizar el desarrollo y no aumentar los costes y tiempos del proyecto. Las tecnologías o frameworks que se adaptan a esta metodología son:

Flutter

Flutter es una mezcla entre un framework y un SDK (Software Development Kit) que está construido en C y C++ para programar en lenguaje Dart.

Permite desarrollar código a una gran velocidad gracias a su “recarga caliente”. Esto permite aplicar los cambios de forma dinámica, en menos de un segundo, sin perder el contexto de la aplicación. Además, se compila en código ensamblado directamente nativo. Así da un gran rendimiento a la hora de usar una aplicación desarrollada con este nuevo framework.

Algunas de sus ventajas son:

- **Recarga caliente:** al hacer algún cambio en el código se podrán ver los efectos reflejados inmediatamente, sin tener que compilar la aplicación de nuevo y sin perder el contexto en el que estábamos.
- **Desarrollo multiplataforma:** no es necesario construir por separado para las dos plataformas: Android y IOS. Flutter ya genera un código base que sirve para ambas plataformas.

Pero como todo, tiene sus desventajas:

- **Dart necesario:** para poder usar Flutter es necesario aprender el lenguaje de programación Dart.
- **Framework muy joven:** y aún no tiene una gran comunidad detrás, por lo que se deberán afrontar los problemas que nos encontraremos con menos ayuda que en otros frameworks.

- **Librerías limitadas:** las bibliotecas a las que pueden acceder están muy limitadas. No siempre proporcionan todas las funcionalidades que necesita el desarrollador.

React Native

React Native es un framework JavaScript para crear aplicaciones reales nativas para iOS y Android, basado en la librería de JavaScript React para la creación de componentes visuales, cambiando el propósito de los mismos para, en lugar de ser ejecutados en navegador, correr directamente sobre las plataformas móviles nativas.

Algunas de sus ventajas son:

- **Desarrollo rápido:** su tiempo de desarrollo es muy corto, el framework ofrece muchos componentes que pueden acelerar el proceso de desarrollo.
- **Desarrollo multiplataforma:** al igual que otros frameworks y tecnologías que vamos a ver React Native permite desarrollar un código que se podrá utilizar tanto en iOS como Android.
- **Amplia comunidad:** es un framework muy utilizado en todo el mundo y se pueden encontrar muchos foros de desarrolladores para solucionar problemas o consultar dudas.

Sus desventajas son las siguientes:

- **Compatibilidad:** aunque es muy utilizado aún sigue en estado de beta por lo que se pueden encontrar problemas de compatibilidades con los distintos paquetes de JavaScript y con las herramientas de depuración de código.
- **Conocimiento de Native:** algunas de las características y módulos necesitan que el desarrollador conozca native y algunas funcionalidades como por ejemplo las notificaciones (push notifications) no están aún soportadas.

Kotlin Native

Se trata de una tecnología que **permite compilar el código escrito en Kotlin directamente a binarios nativos**, esto indica que ese código puede ser ejecutado sin la necesidad de una máquina virtual. Es, en esencia, un *backend* basado en LLVM para el compilador de Kotlin y la implementación nativa de la librería estándar de Kotlin. Actualmente ofrece soporte para plataformas como iOS, MacOS, Android, Windows, Linux y WebAssembly.

Las ventajas de Kotlin Native son:

- **Desarrollo multiplataforma:** permite un desarrollo multiplataforma para una gran variedad de sistemas como ya se ha mencionado anteriormente.
- **Fácil de aprender:** a diferencia de otras tecnologías no depende de ningún otro lenguaje, además si se viene de programar java la curva de aprendizaje es mínima.
- **Buen rendimiento:** Kotlin Native proporciona un rendimiento excelente a la hora de programar las aplicaciones ya que no hace uso de bridge entre el código compartido y las plataformas nativas.

Algunas de sus desventajas son:

- **Es nueva:** aunque se está apostando mucho por ella aún está en estado experimental y esto puede suponer un reto para los desarrolladores.

Conclusión:

Vistos los principales frameworks/tecnologías para el desarrollo de aplicaciones multiplataforma voy a escoger Kotlin Native porque pese a que es reciente y aún se está desarrollando y experimentando se asemeja a Java (lenguaje que conozco) y me va a permitir no tener que invertir mucho tiempo en aprender otro lenguaje y además ofrece un buen rendimiento y cada vez va teniendo más comunidad.

- Tecnologías para la aplicación de administración.

Para esta aplicación he tenido menos condicionamientos ya que no va a ser móvil por lo que puede ser solo para ordenador. Las tecnologías que más me han atraído para este desarrollo son:

Java

Java es un lenguaje de programación orientado a objetos que se incorporó al ámbito de la informática en los años noventa. La idea de **Java** es que pueda realizarse programas con la posibilidad de ejecutarse en cualquier contexto, en cualquier ambiente, siendo así su portabilidad uno de sus principales logros.

Algunas de sus ventajas son:

- **Multiplataforma:** la principal característica de Java es que es independiente de la plataforma (multiplataforma). Esto significa que cuando estás programando en Java, no necesitas conocer a priori el tipo de ordenador o el sistema operativo para el que estás programando.
- **Fácil de usar:** el lenguaje Java es relativamente fácil de aprender comparado con otros.
- **Cómodo:** hoy en día existen excelentes editores (IDEs) que aportan multitud de ayudas a la programación, haciendo que el desarrollo sea más fluido y cómodo.

Algunas desventajas son:

- **Rendimiento:** al tratarse de un lenguaje interpretado, el rendimiento en la ejecución de programas suele ser un poco menor.
- **Sintaxis:** su sintaxis comparada con C# o Python parece para algunos bastante engorrosa.

C#

C# es un lenguaje de programación desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Sus ventajas son:

- Microsoft actualmente utilizan c# como lenguajes programación principal, de desarrollo aplicaciones.
- Es multiparadigma, no tengo que concentrarme solamente en programar orientado a objetos, puedo programar en Dinámico, programar con eventos, me permite programar lo mejor de estos cada uno.

Sus desventajas son:

- Pese a tener compatibilidades con una gran variedad de servicios y bases de datos no tiene una librería oficial para utilizar servicios como Firebase (tiene algunas como FileSharp o FirebaseSharp pero al no ser oficiales siempre tienes que depender del soporte de la comunidad).
- Al estar desarrollado por Windows desplegar una aplicación hecha con c# puede ser costoso implementarla en otros sistemas operativos.

Python

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

Algunas de sus ventajas son:

- **Simplificado y rápido:** Este lenguaje simplifica mucho la programación. Es un gran lenguaje para scripting, si se requiere algo rápido (en el sentido de la ejecución del lenguaje), con unas cuantas líneas ya está resuelto.
- **Ordenado y limpio:** El orden que mantiene Python, es de lo que más le gusta a sus usuarios, es muy legible, cualquier otro programador lo puede leer y trabajar sobre el programa escrito en Python. Los módulos están bien organizados, a diferencia de otros lenguajes.

Sus desventajas son:

- **Hosting:** la mayoría de los servidores no tienen soporte a Python, y si lo soportan, la configuración es un poco difícil.
- **Librerías incluidas:** algunas librerías que trae por defecto no son del gusto de amplio de la comunidad, y optan a usar librerías de terceros.

Conclusión:

Para el desarrollo de la aplicación de administración me he decantado por Java porque, aunque otras tecnologías como C# me han atraído, el hecho de que tengo más experiencia con Java y que por ejemplo para utilizar Firebase como mi servidor y base de datos es más adecuada.

- **Tecnologías para el servidor.**

Para el servidor hay muchas opciones y todas tienen sus puntos fuertes y sus débiles, pero como uno de los requisitos o valoraciones es que todo funcione en tiempo real (sin tener que recargar los datos), he contemplado las siguientes:

AWS

Amazon Web Services es un conjunto de herramientas y servicios de cloud computing de Amazon. Este servicio se lanzó oficialmente en 2006 y para junio de 2007 AWS ya contaba con una base de usuarios de aproximadamente 180 mil personas. La tendencia general para las plataformas en la nube es la de ofrecer la mayor cantidad posible de herramientas y servicios, para que así se pueda crear todo un entorno de computación en una misma nube.

Algunas ventajas son:

- **Low cost:** Amazon ha pensado esta plataforma con la mentalidad low cost. Cuantos más usuarios la usen, más económico es.
- **Se factura por lo que se utiliza:** incluso si te equivocas y haciendo pruebas contratas algo involuntariamente, para notarlo en la factura se deberían subir bastantes gigas de información involuntariamente.

- **Es escalable:** cada usuario puede contratar lo que quiera y configurar su servicio como quiera. No hay un solo modo de utilizar AWS, sino múltiples y casi ilimitados.

Pero no todo lo que brilla es oro, algunas desventajas son:

- **No apto para amateurs:** el sistema es muy complejo si no tienes experiencia previa y a veces es necesario conocer y utilizar demasiados plugins. En este sentido hay plataformas más sencillas de usar.
- **Difícil planificación presupuestaria y de coste:** si bien para algunos casos el cobro por uso puede ser una ventaja, en otros no. Cuando la planificación de costos es estricta puede ser muy difícil determinar cuál será el cobro total de los servicios adquiridos, sobre todo cuando el flujo de trabajo es más intenso.

Firebase

Firebase de Google es una plataforma en la nube para el desarrollo de aplicaciones web y móvil. Está disponible para distintas plataformas (iOS, Android y web), con lo que es más rápido trabajar en el desarrollo. Su función esencial es hacer más sencilla la creación de tanto aplicaciones webs como móviles y su desarrollo, procurando que el trabajo sea más rápido, pero sin renunciar a la calidad requerida. Sus herramientas son variadas y de fácil uso, considerando que su agrupación simplifica las tareas de gestión a una misma plataforma.

Sus ventajas son:

- **Real time:** es muy recomendable para aplicaciones que necesiten compartir datos en tiempo real.
- **Funcionalidades:** sus funcionalidades, además de ser variadas, se complementan muy bien y se pueden gestionar de forma sencilla desde un único panel. Además, no es necesario usar todas estas opciones para la aplicación, pudiendo elegir solo aquellas que más nos interesen.
- **Soporte:** su soporte es gratuito vía email, sin importar si el desarrollador utiliza la versión gratuita o de pago.
- **Escalabilidad:** los inicios son gratuitos, pero permite ir adaptándose a las necesidades de la aplicación con diferentes planes de pago.

- **Seguridad:** con los certificados SSL proporciona gran seguridad para los usuarios.
- Permite a los desarrolladores restarle atención al backend y a las infraestructuras complejas para centrarse completamente en otros aspectos

Sus desventajas son:

- **Pruebas en la nube limitadas:** el servicio de realizar testing de tu apk en la nube, es algo que nos ha gustado a todos. Pero tenemos restricciones en la versión gratuita. Solamente puedes hacer pruebas en 15 dispositivos por día, de los cuales, solo 5 dispositivos son físicos, los demás son virtualizados.
- **Precio:** Firebase ofrece muchas funciones (por no decir, casi todas) con la versión gratuita. Pero, no por tener todas las funcionalidades, significa que no haya límites. Como se mencionó anteriormente habrá varias situaciones donde habrá límites, como las conexiones de usuarios simultáneos. Para tener ilimitado deberemos pagar.

Springboot

Spring Boot permite compilar nuestras aplicaciones Web como un archivo .jar que podemos ejecutar como una aplicación Java normal (como alternativa a un archivo .war, que desplegaríamos en un servidor de aplicaciones como Tomcat).

Esto lo consigue integrando el servidor de aplicaciones en el propio .jar y levantándolo cuando arrancamos la aplicación. De esta forma, podemos distribuir nuestras aplicaciones de una forma mucho más sencilla, al poder configurar el servidor junto con la aplicación. Esto también es muy útil en arquitecturas de micro servicios, puesto que permite distribuir nuestras aplicaciones como imágenes Docker que podemos escalar horizontalmente.

Algunas de sus ventajas son:

- **Tiempo:** reduce el tiempo de desarrollo y aumenta la eficiencia a la hora de trabajar en equipo.
- **Depuración:** proporciona un servidor HTTP embebido como Tomcat o Jetty que para desarrollar y testear la aplicación es muy útil.
- **Plugins:** tiene una gran variedad de plugins que le permite trabajar con la mayoría de bases de datos tanto SQL como No-SQL

Sus desventajas son:

- No proporciona un sistema claro y fácil para poder realizar cambios en tiempo real en la base de datos.
- Tienes que entender el sistema de Spring que usa por debajo Spring boot para algunas opciones avanzadas para modificarlas.

Conclusión:

Para el servidor (y base de datos) utilizaré Firebase, me he decantado por esta opción porque es una plataforma muy completa y con grandes y numerosos beneficios, como la posibilidad de tener una base de datos en tiempo real. Es moderadamente sencilla de utilizar e internet está repleto de ejemplos y tutoriales sobre cómo utilizarla y al ser de Google tiene una gran comunidad.

5.2 Aspectos esenciales de la implementación

En este apartado veremos los aspectos esenciales de la implementación, es decir las partes de la aplicación más importantes o que al menos han causado dificultad y son remarcables. El código desarrollado para tanto Android como iOS mediante Kotlin Native es el siguiente (todos los aspectos que se muestran son los basados en Android).

El primer aspecto que vamos a ver es el acceso a la aplicación, este acceso se podrá hacer mediante google o mediante el sistema de autenticación implementado por nosotros y basado en Firebase.

En este código podemos ver la implementación del **sistema de autenticación con Firebase**, donde mediante la variable **auth** (de tipo FirebaseAuth) accedemos a la función de **signInWithEmailAndPassword** (para acceder con el email y contraseña del usuario). Además, puesto que en nuestra app cuando un usuario se loguea se queda guardada su sesión (durante un periodo de tiempo) permitiéndole volver a acceder sin necesidad de volver a meter sus **credenciales**, para ello guardamos las preferencias del usuario y guardamos en ellas su email y contraseña (flecha azul)

```
private lateinit var auth: FirebaseAuth;
auth = FirebaseAuth.getInstance()

btnLogin.setOnClickListener(View.OnClickListener { it: View!  
    //se comprueba que el correo y contraseña esten bien rellenos  
    if (!etLoginEmail.text.toString().isEmpty() && !etLoginPassword.text.toString()  
        .isEmpty())  
    {  
  
        //se accede al acceso de Firebase  
        auth.signInWithEmailAndPassword(  
            etLoginEmail.text.toString(),  
            etLoginPassword.text.toString())  
        .addOnCompleteListener { it: Task<AuthResult!>  
            //si se ha completado se guardan las preferencias y nos vamos a la pantalla principal de la app  
            if (it.isSuccessful)  
            {  
                //Guardado de datos de sesión  
                val prefs: SharedPreferences.Editor = getSharedPreferences(  
                    "com.example.tenisclubdroid.PREFERENCE_FILE_KEY",  
                    Context.MODE_PRIVATE  
                ).edit()  
                prefs.putString("email", etLoginEmail.text.toString())  
                prefs.putString("password", etLoginPassword.text.toString())  
                prefs.apply()  
  
                val intent = Intent(packageContext: this, MainActivity::class.java)  
                startActivity(intent)  
                finish()  
            } else {  
                Toast.makeText(context: this, text: "Datos incorrectos!", Toast.LENGTH_SHORT).show()  
            }  
        }  
    } else {  
        Toast.makeText(context: this, text: "Rellene todos los campos!", Toast.LENGTH_SHORT).show()  
    }  
}
```

La próxima vez que usuario se meta en la app tendrá sus credenciales guardadas y con el método de **cargarSesion()** se cogerán los datos del email y contraseña y accederá la página principal de la aplicación.

```
private fun cargarSesion() {
    val prefs: SharedPreferences =
        getSharedPreferences("com.example.tenisclubdroid.PREFERENCE_FILE_KEY", Context.MODE_PRIVATE)
    val email = prefs.getString( key: "email", defValue: null)
    val password = prefs.getString( key: "password", defValue: null)

    if (email != null && password != null) {
        val intent = Intent( packageContext: this, MainActivity::class.java)
        startActivity(intent)
    }
}
```

El acceso con google actúa como login y a la vez registro. Pues, aunque el usuario no tiene que introducir datos la aplicación si recogerá los datos que se necesitan para que pueda interactuar con la app como el nombre de usuario, su identificador y su foto de perfil (que tiene de su cuenta de google). Para ello con la configuración (dentro del botón) y la credencial del usuario que nos proporciona google podemos acceder a la aplicación con el método de **signInWithCredential** , para más adelante guardar los dichos datos del usuario en la **base de datos** de Firebase llamando la url de la base de datos de los usuarios (flecha azul) y mediante una clase POJO guardar al usuario en ella (flecha verde) (mejor explicado en el registro).

```
private val GOOGLE_SIGN_IN = 100

btnLoginGoogle.setOnClickListener(View.OnClickListener { it: View!

    val googleConf =
        GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN).requestIdToken(
            "870878529434-moi22k960dm0v0jdl02a0hs13efkoc6h.apps.goog..."
        ).requestEmail().build()

    val googleClient = GoogleSignIn.getClient(this, googleConf)
    startActivityForResult(googleClient.signInIntent, GOOGLE_SIGN_IN)

})
```

```

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == GOOGLE_SIGN_IN) {
        val task = GoogleSignIn.getSignedInAccountFromIntent(data)
        try {
            val account = task.getResult(ApiException::class.java)

            if (account != null) {
                val credential = GoogleAuthProvider.getCredential(account.idToken, null)
                FirebaseAuth.getInstance().signInWithCredential(credential)
                    .addOnCompleteListener { it: Task<AuthResult!>
                        if (it.isSuccessful) {
                            //se recoge el usuario
                            val user = FirebaseAuth.getInstance().currentUser
                            //se cogen sus datos
                            val nickname = user?.displayName.toString()
                            val id = user?.uid.toString()
                            val foto = user?.photoUrl.toString()
                            val u = Usuario(nickname, foto, descripcion: "Tu descripción", rol: 0, id)
                            //se guarda en la base de datos
                            FirebaseAuth.getInstance().currentUser?.let { it1 ->
                                FirebaseDatabase.getInstance( url: "https://tenisclubdroid-default-rtdb.firebaseio.com/" )
                                    .getReference( path: "usuarios" ).child(
                                        it1.uid
                                    ).setValue(u).addOnCompleteListener { it: Task<Void!>
                                }
                            }
                        }
                    }
            }

            val intent = Intent( packageContext: this, MainActivity::class.java)
            startActivity(intent)
            finish()

        } else {
            //Toast.makeText(this, "Error de acceso", Toast.LENGTH_SHORT).show()
        }
    }
}

} catch (e: ApiException) {
    Toast.makeText( context: this, text: "Error, revisa tu internet", Toast.LENGTH_SHORT).show()
}
}

```

Otro aspecto importante de la aplicación es el **registro** de un usuario mediante el sistema de Firebase, aquí el usuario creará su cuenta con su nombre de usuario (único), su correo y su contraseña.

Lo primero que se hace es comprobar que el usuario ha llenado correctamente los campos del email, usuario, contraseña y la confirmación de la contraseña, en caso contrario se le hará saber al usuario con mensajes de tipo Toast y poniendo campos en color rojo.

Una vez comprobados llamaremos al método de registrar (flecha azul)

```

//se comprueba que todos los campos esten rellenos
if (!email.isEmpty() && !contra.isEmpty() && !confirm_contra.isEmpty() && !usuario.isEmpty()) {
    //se comprueba que el email tiene un formato correcto
    if (comprobarEmail(email.trim())) {
        etRegistroEmail.setBackgroundTintList(activity?.applicationContext?.let { it1 ->
            ContextCompat.getColorStateList(
                it1, R.color.background_tint_azul
            )
        })
    }
    if (contra.trim().length >= 6) {
        etRegistroContra.setBackgroundTintList(activity?.applicationContext?.let { it1 ->
            ContextCompat.getColorStateList(
                it1, R.color.background_tint_azul
            )
        })
    }
}

//se comprueba que la contraseña es correcta
if (contra.trim().equals(confirm_contra)) {

    etRegistroConfirmContra.setBackgroundTintList(activity?.applicationContext?.let { it1 ->
        ContextCompat.getColorStateList(
            it1, R.color.background_tint_azul
        )
    })
}


//se han pasado los filtros y se crea la cuenta con el email y la contraseña
registrar(usuario, email, contra)

} else {

```

Al registrar también se hace la comprobación de que el nombre del usuario sea único, para ello crearemos una variable de tipo FirebaseDatabase (la base de datos de Firebase) y unas de tipo DatabaseReference, una para registrar (crear) al usuario en la base de datos y otra para guardar el nombre cogido del usuario (para que no se vuelva a utilizar) y también para comprobar que el introducido no está siendo usado.

```

private lateinit var nombresCogidos: DatabaseReference
private lateinit var database: FirebaseDatabase
database =
    FirebaseDatabase.getInstance("https://tenisclubdroid-default-rtdb.firebaseio.com")

databaseReference = database.reference.child("usuarios")
nombresCogidos = database.reference.child("NombresCogidos")

```

Primero se comprueba que el nombre no este cogido utilizando la función sobreescrita asíncrona de onDataChange para que busque si en la base de datos de los nombres cogidos existe ya el que el usuario a introducido, en caso de que este cogido se le notificara al usuario y en caso contrario con el método de Firebase **createUserWithEmailAndPassword** (flecha verde) se guardará en la parte de Firebase de autenticación,si todo ha salido bien con el identificador del usuario que se ha creado y el método de **FirebaseDatabase.getInstance(la url de la bbdd).getReference (documento de los usuarios).child (identificador).setValue(el usuario)** (flecha azul) se

guardará en la base de datos (documento) de los usuarios con los datos introducidos además de una descripción por defecto y una imagen también por defecto. Una vez hecho también se guardará en la base de datos de los nombres cogidos el nickname del usuario para que sea único (flecha naranja).

```

private fun registrar(usuario: String, email: String, contra: String) {
    nombresCogidos.addListenerForSingleValueEvent(object : ValueEventListener {
        override fun onDataChange(dataSnapshot: DataSnapshot) {
            Log.e( tag: "repetido", msg: "" + dataSnapshot.child(usuario).key)
            if (!dataSnapshot.hasChild(usuario)) {
                // dato =dataSnapshot.child(sUsername).key.toString()
                FirebaseAuth.getInstance().createUserWithEmailAndPassword(
                    email,
                    contra
                ).addOnCompleteListener { it: Task<AuthResult!>
                    if (it.isSuccessful) {
                        val id =
                            FirebaseAuth.getInstance().currentUser?.uid.toString()
                        val foto_preweseta = "https://firebasestorage.googleapis.com/v0/b/tenisclubdroid.appspot.com/o/usuario.jpg?alt=media&token=b34680bc-64a9-4598-a734-8180aa"
                        //public Usuario(String nickname, String fotoPerfil, String descripcion, int rol)
                        var u = Usuario(usuario, foto_preweseta, descripcion: "Tu descripción", rol: 0, id)
                    }
                }
            }
        }
    })
}

FirebaseAuth.getInstance().currentUser?.let { it1 ->
    FirebaseDatabase.getInstance( url: "https://tenisclubdroid-default-rtbd.firebaseio.west1.firebaseio.database.app/")
        .getReference( path: "usuarios").child(
            it1.uid
        ).setValue(u).addOnCompleteListener { it: Task<Void!>
            if (it.isSuccessful) {
                nombresCogidos.child(usuario).setValue(id)
                val toast1 = Toast.makeText(
                    context,
                    text: "Guardado", Toast.LENGTH_SHORT
                )
                toast1.show()
            }
        }
}

```

Una vez registrado volverá a la pantalla de login

```

        }
    }
}
val intent = Intent(activity, LoginActivity::class.java)
activity?.startActivity(intent)
} else {
    val toast1 = Toast.makeText(
        context,
        text: "MAL", Toast.LENGTH_SHORT
    )
    toast1.show()
}
}
else{
    Toast.makeText(
        context,
        text: "Usuario repetido ,escoja otro",
        Toast.LENGTH_SHORT
    ).show()
    Log.e( tag: "repetido2", dataSnapshot.child(usuario).key.toString())
}
}

override fun onCancelled(databaseError: DatabaseError) {
    Toast.makeText(
        context,
        text: "ERROR",
        Toast.LENGTH_SHORT
    ).show()
}
})
}

```

El siguiente aspecto tiene que ver cuando el usuario **olvida su contraseña**, pondrá su email y el servicio de Firebase le mandará un correo para que pueda restablecerla. Para ello primero comprobaremos que el email está lleno y que es un email (al menos que cumple el formato), una vez hecho con la variable auth de tipo FirebaseAuth llamará al método de **sendPasswordResetEmail** (*su email*), si se ha mandado con éxito se le notificará al usuario mediante un Toast y si por el contrario no (por un problema de internet o porque el correo no exista) le notificará con otro Toast.

```
private lateinit var auth: FirebaseAuth

btnContraOlvidadaEnviar.setOnClickListener(View.OnClickListener { it: View! }

    val email = etContraOlvidadaEmail.text.toString()

    if (!email.isEmpty()) {
        email.trim()

        if (comprobarEmail(email)) {
            auth = FirebaseAuth.getInstance()
            auth.sendPasswordResetEmail(email).addOnCompleteListener { it: Task<Void!>
                if (it.isSuccessful) {
                    val toast1 = Toast.makeText(
                        context,
                        text: "se le ha enviado un correo ", Toast.LENGTH_SHORT
                    )
                    toast1.show()
                    val intent = Intent(activity, LoginActivity::class.java)
                    activity?.startActivity(intent)
                } else {
                    val toast1 = Toast.makeText(
                        context,
                        text: "ha ocurrido un error ", Toast.LENGTH_SHORT
                    )
                    toast1.show()
                }
            }
        }
    }
}
```

Finalmente veremos el perfil del usuario, donde podrá **ver sus datos** (cogidos de la base de datos) y también **podrá modificarlos**, de esta forma quedará reflejada en la documentación un **CRUD** con Firebase (el usuario no podrá borrarse a si mismo pero el sistema al cambiar el usuario de nickname elimina de la base de nombres cogidos al antiguo para dejarlo libre)

En el Fragment de Perfil se **mostrarán los datos del usuario** y para ello al igual que en los anteriores habrá que hacer uso de variables de Firebase del tipo FirebaseDatabase y DatabaseReference para coger los datos del usuario de la bbdd de Firebase. Con la referencia del usuario (cogida con su identificador único) se creará un ValueEventListener que actuará de “query” para buscar al usuario entre todos los que hay para poder coger con la snapshot de tipo DataSnapshot su nickname, descripción, la dirección de la imagen (guardada en el storage de Firebase). Mediante la librería de Picasso se mostrará en el perfil la imagen del usuario y además los datos se guardarán en la clase POJO de usuario para más adelante poder pasárselo al Fragment de edición y así no tener que repetir código ni llamadas a la base de datos innecesarias.

```
//se coge el usuario por su uid
database =
    FirebaseDatabase.getInstance( url: "https://tenisclubdroid-default-rtdb.europe-west1.firebaseio.database.app/" )

databaseReference = database.reference.child( pathString: "usuarios")
val id_usuario = FirebaseAuth.getInstance().currentUser?.uid

val referencia_usuario = databaseReference.child(id_usuario!!)

referencia_usuario.addValueEventListener(object : ValueEventListener {

    override fun onDataChange(snapshot: DataSnapshot) {
        tvPerfilNickName.setText(snapshot.child( path: "nickName").value.toString())
        tvPerfilDescripcion.setText(snapshot.child( path: "descripcion").value.toString())
        fotoUrl = snapshot.child( path: "fotoPerfil").value.toString()
        Picasso.get().load(fotoUrl).transform(ImagenRedonda()).into(ivPerfilFoto)
        usuario = Usuario(
            tvPerfilNickName.text.toString(),
            fotoUrl,
            tvPerfilDescripcion.text.toString(),
            rol: 0,
            id_usuario
        )
    }

    override fun onCancelled(error: DatabaseError) {
        TODO( reason: "Not yet implemented")
    }
})
```

Los últimos aspectos relevantes se llevan a cabo en el Fragment de editar el perfil, pues aquí se **modifican** (sobrescriben, pero de forma más sencilla) y **eliminan** datos de la base la base de datos de Firebase. Para modificar los datos del usuario primero se comprueba que estén los campos correctamente rellenos y si el usuario se ha cambiado su nickname se comprobará (como se explicó en el registro) que no esté cogido, a diferencia del registro aquí también hay que comprobar si el nombre que esta repetido pertenece al usuario porque en este caso es debido a que no lo está modificando sino simplemente está cambiando otros datos, una vez hecho llamará al método para **subirUsuario()** para modificarlo.

Tambien cabe destacar que si el usuario ha cambiado de nickname se **eliminará** del documento de nombresCogidos de la base de datos el viejo nickname para que vuelva a quedar libre. Para ello cogiendo la referencia del documento de los nombres cogidos y buscando el viejo nickname se elimina con **removeValue()**

```
btnPerfilActualizar.setOnClickListener(View.OnClickListener { it: View!>

    //se comprueba que estan los campos llenos
    if (!etEditarNickName.text.toString().isEmpty() && !etEditarDescripcion.text.toString()
        .isEmpty())
    ) {

        val nickname = etEditarNickName.text.toString()

        nombresCogidos.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(dataSnapshot: DataSnapshot) {
                Log.e("tag: "repetido", msg: "" + dataSnapshot.child(nickname).key)
                if (!dataSnapshot.hasChild(nickname) || dataSnapshot.child(nickname).value == usuario.idUsuario) {
                    nombresCogidos.child(nickname).setValue(usuario.idUsuario)
                    if (!etEditarNickName.text.toString()
                        .equals(nombresCogidos.child(usuario.nickName)))
                } {

                    nombresCogidos.child(usuario.nickName).removeValue()
                }
                subirUsuario()
            } else {
                Toast.makeText(
                    activity?.baseContext,
                    text: "Nombre de usuario no valido",
                    Toast.LENGTH_SHORT
                )
            }
        }.show()
    }
}
```

El método subirUsuario sirve para guardar los cambios como para cambiar la foto de perfil del usuario. Al coger la foto de la galería o cámara cogera la dirección (fotoUri) de la imagen y la subirá a FirebaseStorage en la carpeta de imágenes y después llamará al método para guardar los cambios del usuario.

```

private fun subirUsuario() {
    if (fotoUri == null) {
        guardar(usuario.fotoPerfil)
    } else {
        val filename = UUID.randomUUID().toString()
        val ref = FirebaseStorage.getInstance().getReference( location: "/imagenes/$filename")

        ref.putFile(fotoUri!!).addOnSuccessListener { it: UploadTask.TaskSnapshot!
            ref.downloadUrl.addOnSuccessListener { it: Uri!
                guardar(it.toString())
            }
        }
    }
}

```

Con el método de guardar (que recibe la url de la foto de perfil) se creará una referencia de la base de datos (currentUserDb) que tiene el usuario (mediante su identificador), se creará un usuario con la clase POJO para hacerlo mas cómodamente y con setValue() se guardará en la base de datos de Firebase y volverá al Fragment de perfil.

```

private fun guardar(fotoUrl: String) {
    val fUser: FirebaseUser = FirebaseAuth.getInstance().currentUser!!
    val currentUserDb = databaseReference.child(fUser.uid)
    //se crea el usuario con los datos
    val user = Usuario(
        etEditarNickName.text.toString().trim(),
        fotoUrl,
        etEditarDescripcion.text.toString().trim(),
        rol: 0,
        fUser.uid.toString()
    )
    //se guardan los cambios
    currentUserDb.setValue(user)
    Toast.makeText(activity?.baseContext, text: "Perfil Actualizado", Toast.LENGTH_SHORT).show()

    val fm = fragmentManager
    val perfil = PerfilFragment()
    val transaction = fm!!.beginTransaction()
    transaction.replace(R.id.nav_host_fragment, perfil)
    transaction.addToBackStack( name: null)
    transaction.commit()
}

```

6. Testeo y pruebas

6.1 Plan de pruebas (unitarias, integración, sistema y usuarios)

El plan de pruebas consiste en unos casos de pruebas generales a los aspectos más susceptibles a fallar y que se tienen que controlar. Estas pruebas serán de los siguientes tipos:

- **Unitarias**

Las pruebas unitarias o unit testing son una forma de comprobar que un fragmento de código funciona correctamente. Es un procedimiento más de los que se llevan a cabo dentro de una metodología ágil de trabajo.

- **De Integración**

Las pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias y lo que prueban es que todos los elementos unitarios que componen el software, funcionan juntos correctamente probándolos en grupo. Se centra principalmente en probar la comunicación entre los componentes y sus comunicaciones ya sea hardware o software.

- **De sistema**

Las pruebas del sistema tienen como objetivo ejercitarse profundamente el sistema comprobando la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica.

- **De usuarios**

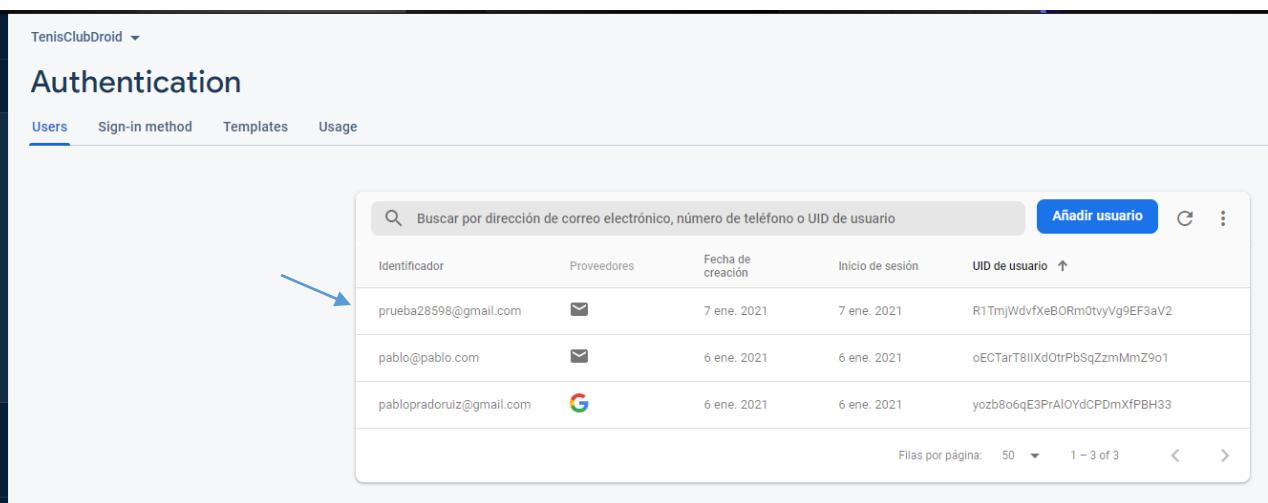
La prueba de usabilidad por parte del usuario es una técnica usada en el diseño de interacciones centrado en el usuario para evaluar un producto mediante pruebas con los usuarios mismos. Esto puede ser visto como una práctica de usabilidad irreemplazable, dado que entrega información directa de cómo los usuarios reales utilizan el sistema.

- Casos de prueba Unitarias:
 - Caso de prueba en Login
 - Caso de prueba en Registro
 - Caso de prueba en Perfil

➤ Caso de prueba en Login:

En el login hay muchas opciones, la primera que vamos a probar es la opción de entrar con la cuenta creada una vez el usuario se registra. Para ello usaré una cuenta asociada al correo de pruebas : prueba28598@gmail.com

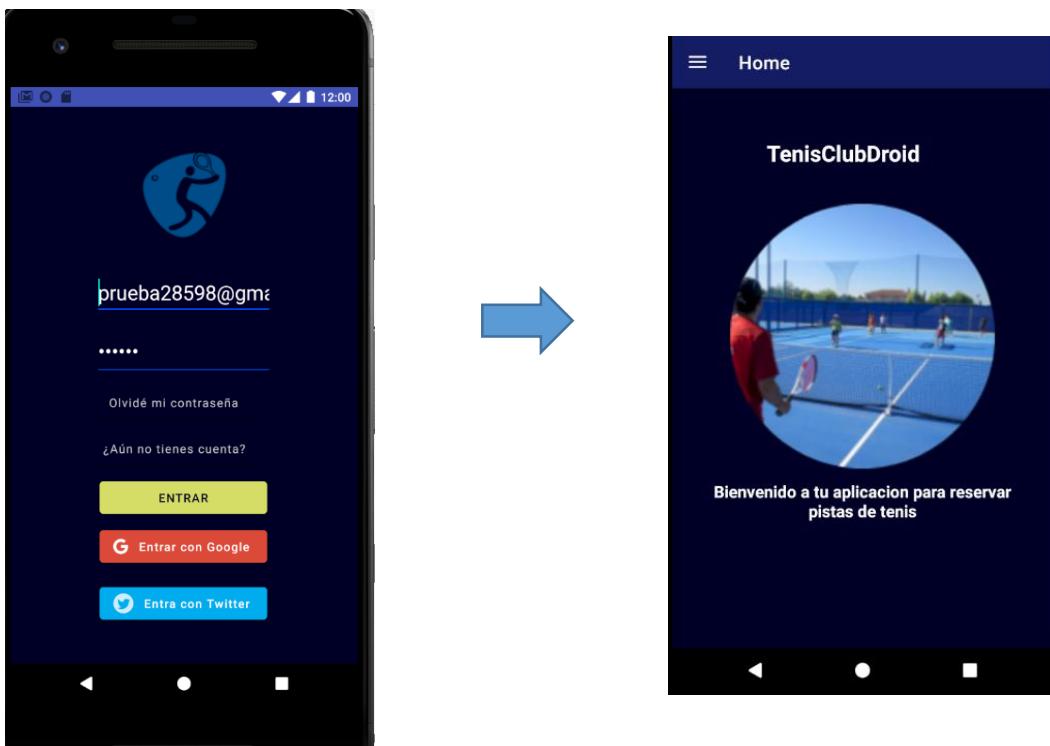
Como se puede ver en la imagen el usuario con el correo de pruebas



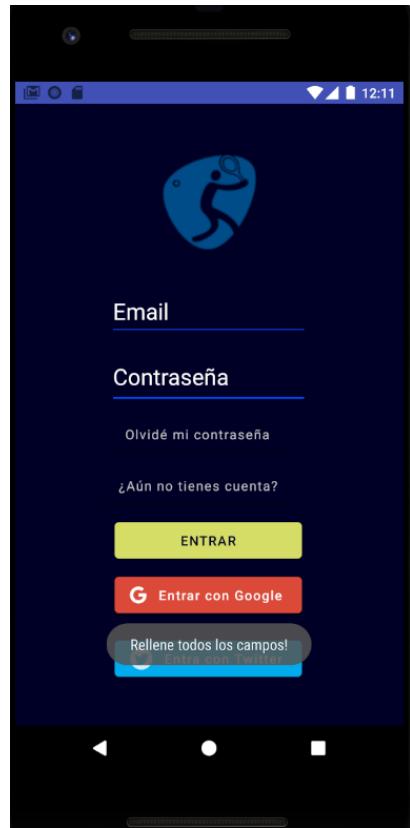
The screenshot shows the Firebase Authentication console under the 'TenisClubDroid' project. On the left, there's a sidebar with various services: Compilación, Authentication (selected), Cloud Firestore, Realtime Database, Storage, Hosting, Functions, and Machine Learning. The main area is titled 'Authentication' and shows the 'Users' tab selected. A search bar at the top allows searching by email, phone number, or UID. Below it is a table with columns: Identificador, Proveedores, Fecha de creación, Inicio de sesión, and UID de usuario. Three users are listed: 'prueba28598@gmail.com' (created via email, Jan 7, 2021, session Jan 7, 2021, UID R1TmjWdvfxeB0Rm0tvYg9EF3aV2), 'pablo@pablo.com' (created via email, Jan 6, 2021, session Jan 6, 2021, UID oECTarT8IIXd0trPbSqZzmMmZ9o1), and 'pabloraduriz@gmail.com' (created via Google, Jan 6, 2021, session Jan 6, 2021, UID yozb8o6qE3PrAlOYdCPDmXfPBH33). A blue arrow points from the text 'Como se puede ver en la imagen el usuario con el correo de pruebas' to the 'prueba28598@gmail.com' row in the table.

Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario
prueba28598@gmail.com	✉	7 ene. 2021	7 ene. 2021	R1TmjWdvfxeB0Rm0tvYg9EF3aV2
pablo@pablo.com	✉	6 ene. 2021	6 ene. 2021	oECTarT8IIXd0trPbSqZzmMmZ9o1
pabloraduriz@gmail.com	G	6 ene. 2021	6 ene. 2021	yozb8o6qE3PrAlOYdCPDmXfPBH33

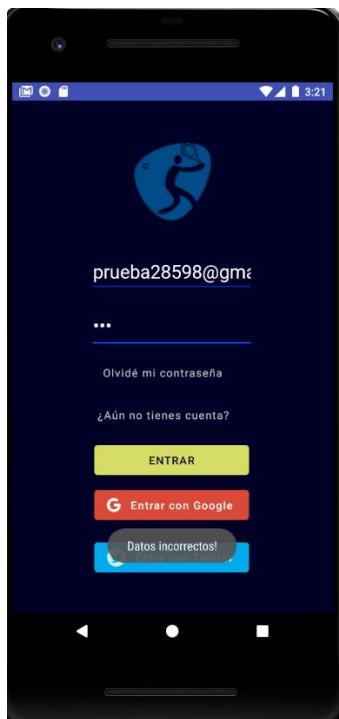
- Al poner los datos correctos el usuario al pulsar “entrar” accede a la aplicación.



- Ahora probaremos a no meter datos y pulsar el botón para entrar

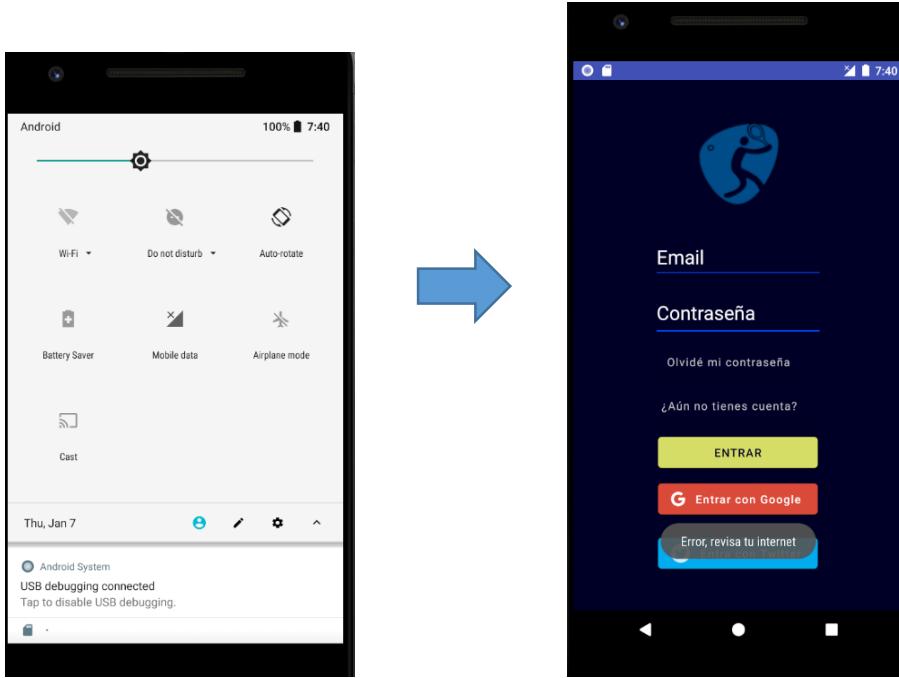


Al hacerlo ha saltado un mensaje para el usuario para que rellene todos los campos. De misma forma si el usuario solo rellena uno de los campos le dirá el mismo mensaje.



-Probaremos a poner datos incorrectos. Cuando ponemos la contraseña incorrecta o el email incorrecto por motivos de seguridad no se especifica cual está mal y simplemente se muestra un mensaje de “Datos incorrectos”.

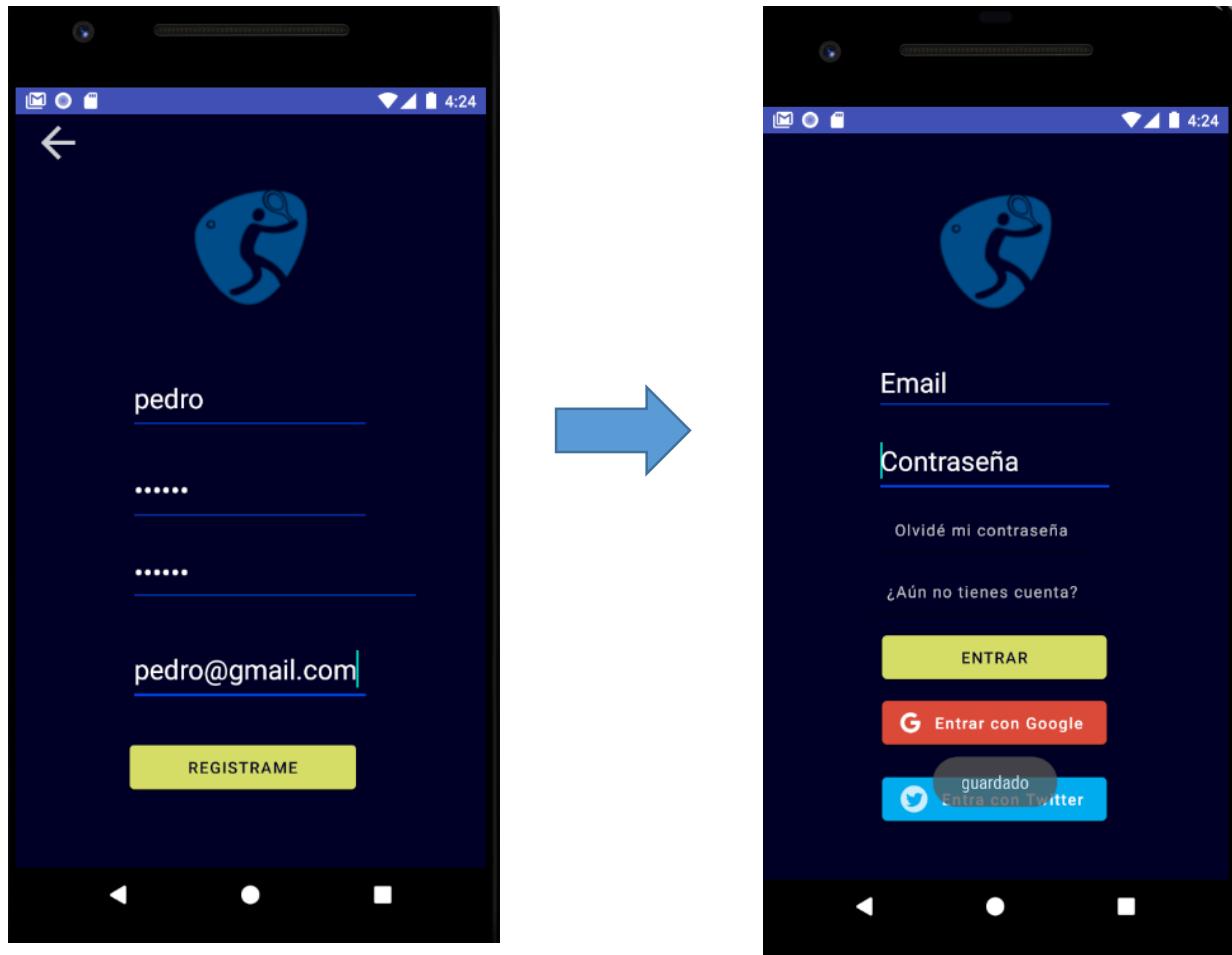
La última prueba del login es de la función de entrar con google, esta acción es muy difícil que falle y solo cuando el usuario no tenga internet activo es cuando puede fallar, en ese caso ocurrirá lo siguiente.



➤ Caso de prueba Registro

En el registro se deben controlar muchas cosas como que todos los campos estén llenos, que el campo del email tenga un formato de email, que la contraseña esté bien formada (6 caracteres y que el nickname no esté cogido por otro usuario.

- La primera prueba será hacer un registro siguiendo las normas y ver si efectivamente lo registra:



El usuario se ha registrado y se ha quedado guardado en la base de datos de Firebase y en la parte de autenticación de Firebase para que pueda acceder a la aplicación.

Authentication				
Users	Sign-in method	Templates	Usage	
<input type="text"/> Buscar por dirección de correo electrónico, número de teléfono o UID de usuario	Añadir usuario			
Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario ↑
pedro@gmail.com		7 ene. 2021	7 ene. 2021	02NVhTz4aeWE3l48s45sAFST4...

Realtime Database

Datos Reglas Copias de seguridad Uso

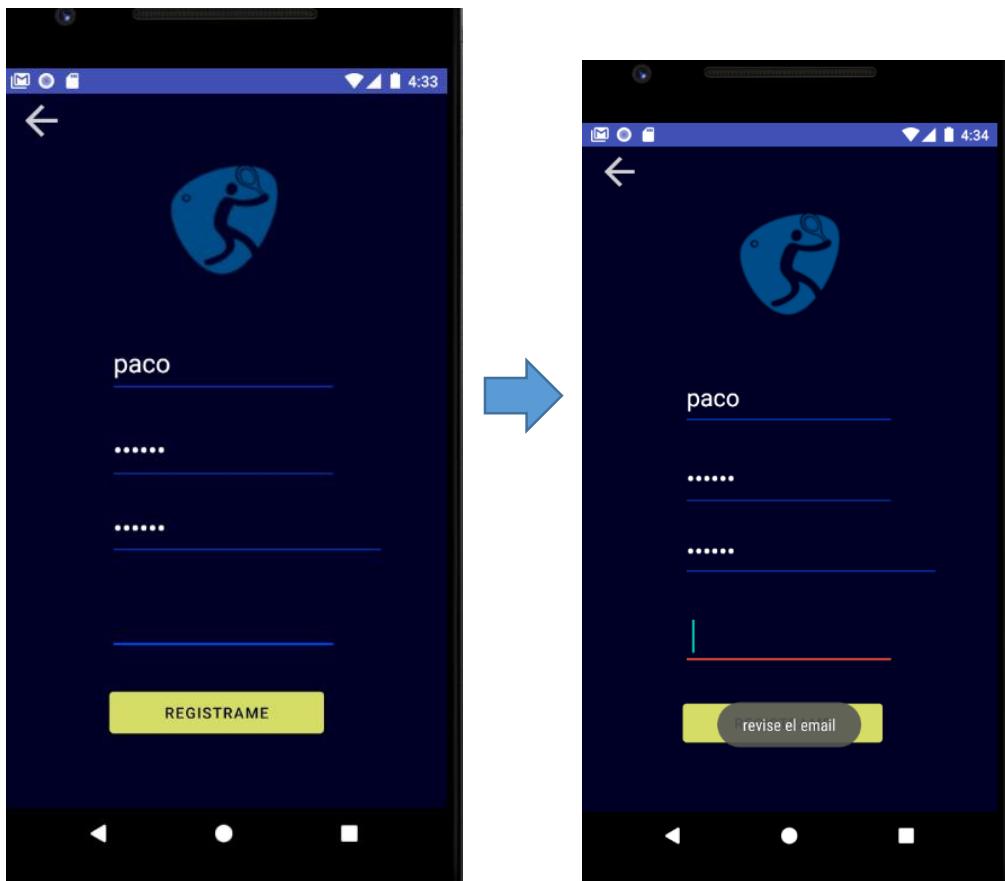
https://tenisclubdroid-default-rtbd.firebaseio.com/.json

```
private_key: "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhki\nprivate_key_id: "11a6704750738b29e5c1a6b5ad34a1aa5fc81t\nproject_id: "tenisclubdroid"\ntoken_uri: "https://oauth2.googleapis.com/tok\ntype: "service_account"\nusuarios\nO2NVhTZs4eWE3I48s45sAFST42P2\ndescripcion: "Tu descripcion"\nfotoPerfil: "https://storage.googleapis.com/v0/b/t..."\nidUsuario: "O2NVhTZs4eWE3I48s45sAFST42P2"\nnickName: "pedro"\nrol: 0
```

- Ahora probaremos a registrar sin llenar los campos. Al hacerlo nos mostrara un mensaje informandonos del problema.

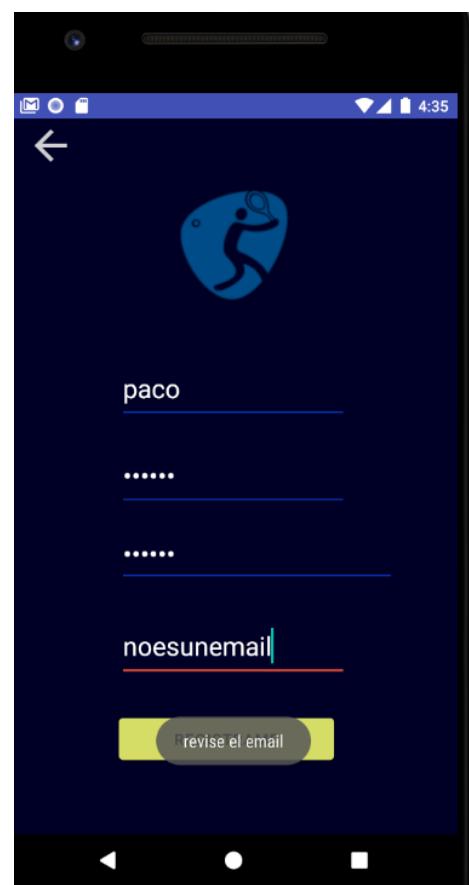


- Ahora probaremos a introducir todos los campos menos el email

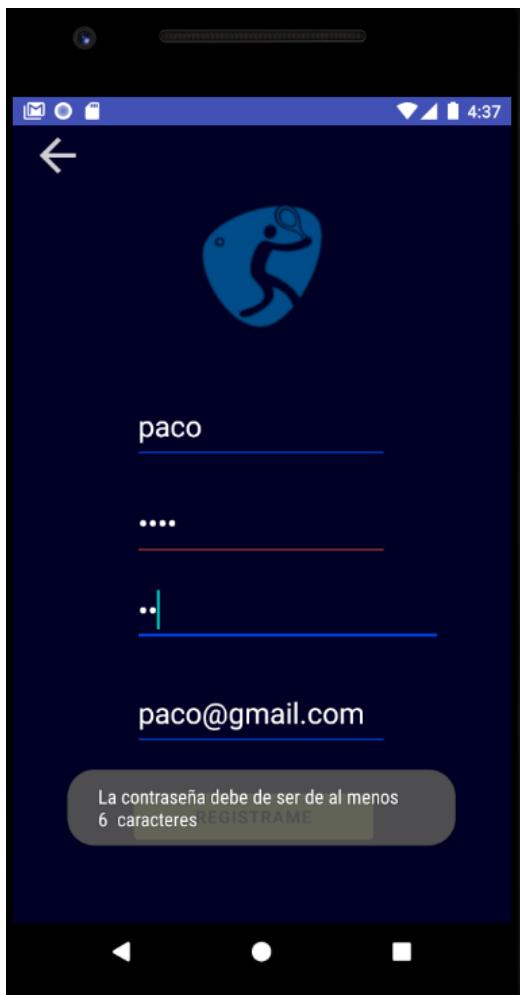


Al hacerlo nos marcará en rojo el campo y nos mostrará un mensaje de error para informarnos.

- Si probamos a rellenarlo con un email falso (con un formato erróneo) también se marcará en rojo y con el mismo mensaje.



- Ahora probaremos a introducir una contraseña demasiado corta.



Al hacerlo se nos marcará de rojo la contraseña y nos mostrara un mensaje informativo.

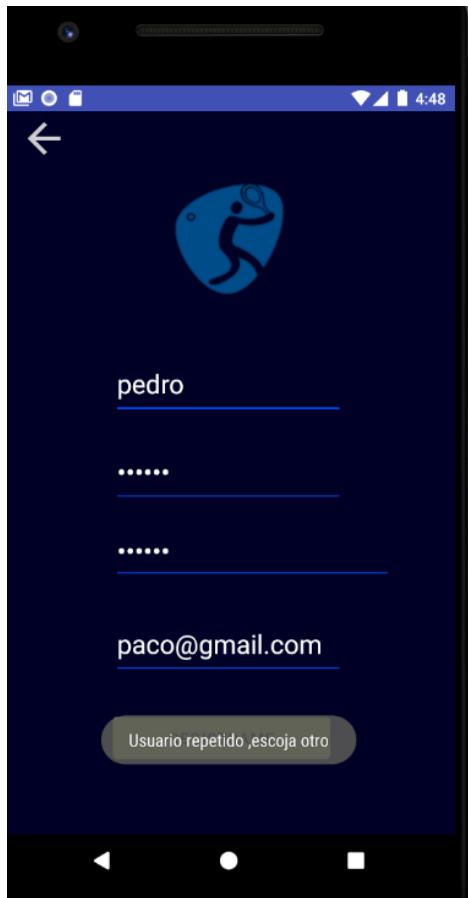
- Cuando introduzcamos mal la contraseña en la confirmación, nos la borrará y marcará en rojo junto con un mensaje para revisarla.



- Finalmente, solo nos queda comprobar que el nickname del usuario no esté cogido por otro. Para comprobarlo vamos a meter el de “pedro” que creamos previamente en el login.

tenisclubdroid-default-rtdb

```
tenisclubdroid-default-rtdb
  |-- NombresCogidos
    |-- pablop: "WpnoEP0VSmaPomqPukJdv971xnS2"
    |-- pedro: "O2NVhTZs4eWE3I48s45sAFST42P2"
    |-- prueba: "R1TmjWdvfXeB0Rm0tvyVg9EF3aV2"
    |-- prueba prueba: "peDpU5NuCZQ7092FFI6DbxhdhKC2"
```



Como puede verse en el documento de la base de datos de nombresCogidos se encuentra el de pedro y por ello al introducirlo le muestra un mensaje de usuario repetido.

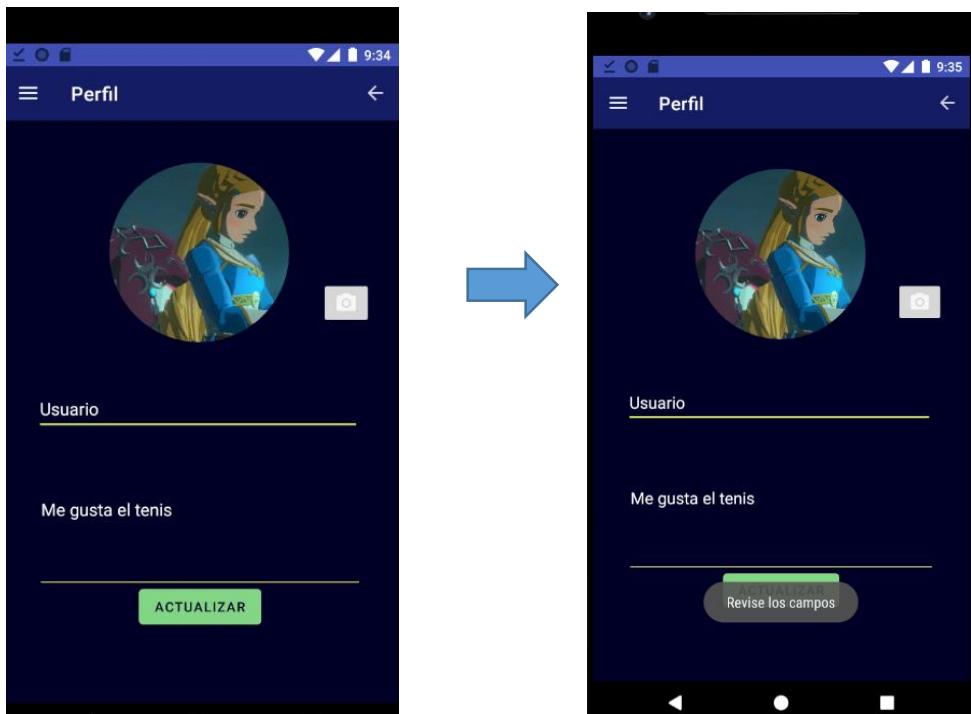
➤ Caso de prueba en perfil

En el perfil se cargan los datos del usuario desde la base de datos de Firebase, no hay muchas formas de que falle pues el código este hecho siguiendo la documentación que proporciona Firebase.

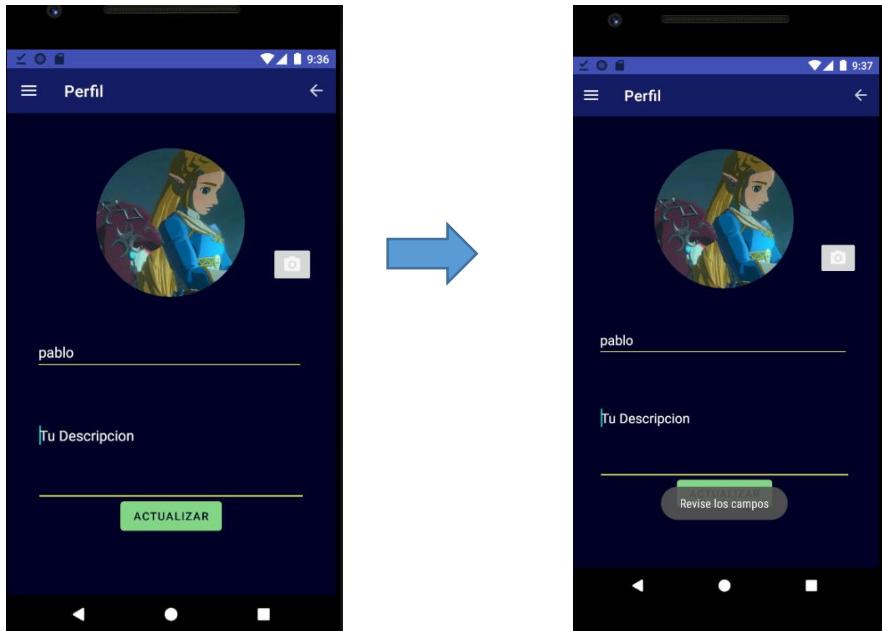


En el perfil también accedemos a editar el perfil para que el usuario pueda cambiar de foto de perfil, nickname o su descripción.

➤ Vamos a probar a dejar vacío el campo del nickname



Como podemos ver nos aparece un mensaje de revisar los campos y lo mismo pasa al dejar vacío el campo de descripción.



- Ahora vamos a actualizar el nombre del usuario a uno repetido

Los nombres cogidos son los siguientes:

```
tenisclubdroid-default-rtdb
  NombresCogidos
    pablo: "oECTarT8IIXd0trPbSqZzmMmZ9o1"
    pablop: "WpnoEP0VSmaPomqPukJdv971xnS2"
    pedro: "O2NVhTZs4eWE3I48s45sAFST42P2"
    prueba: "R1TmjWdvfXeBORm0tvyVg9EF3aV2"
```

Vamos a intentar asignarle el nombre de pedro



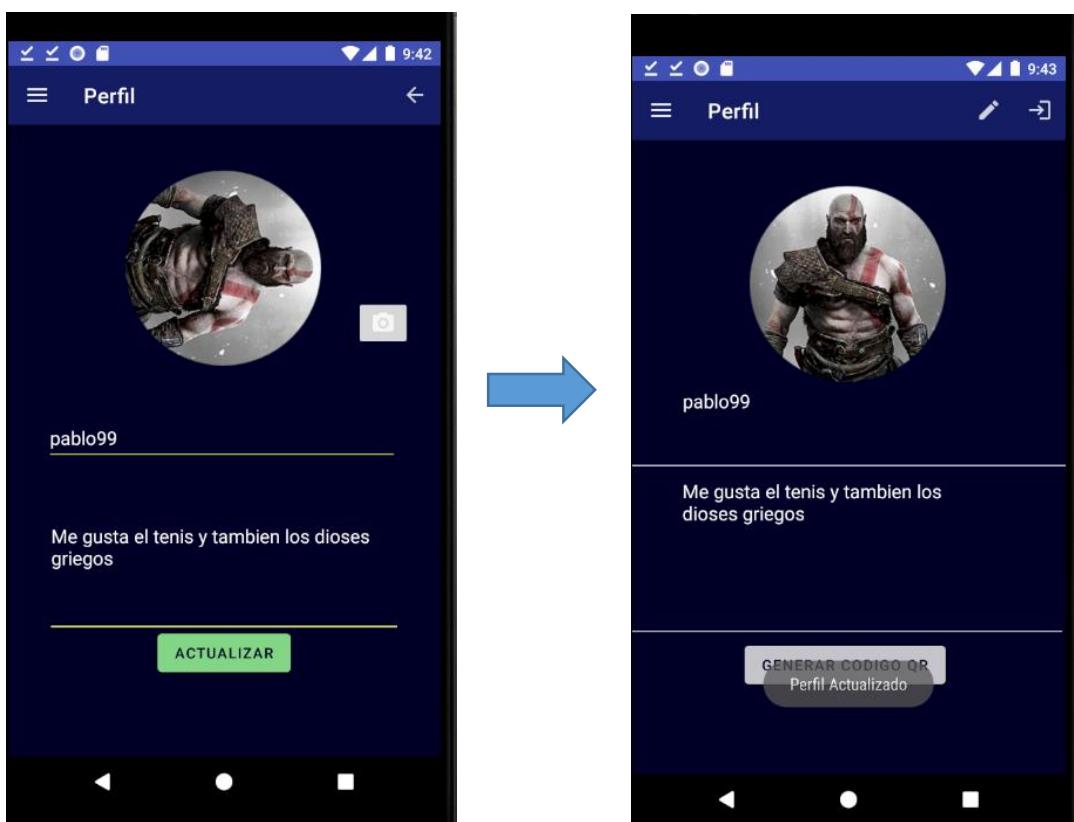
Como se puede ver no nos deja actualizar y nos sale un mensaje de usuario no valido.

- Finalmente vamos a actualizar el perfil con un nickname libre y añadiendo mas descripción y una nueva imagen.

Este es el usuario antes del cambio:

```
- oECTarT8IIXd0trPbSqZzmMmZ9o1
  |-- descripcion: "Me gusta el tenis"
  |-- fotoPerfil: "https://firebasestorage.googleapis.com/v0/b/ten...)"
  |-- idUsuario: "oECTarT8IIXd0trPbSqZzmMmZ9o1"
  |-- nickName: "pablo"
  |-- rol: 0
+-- deDpU5NuCZO7O92FFl6DbxhdhKC2
```

Con el cambio:

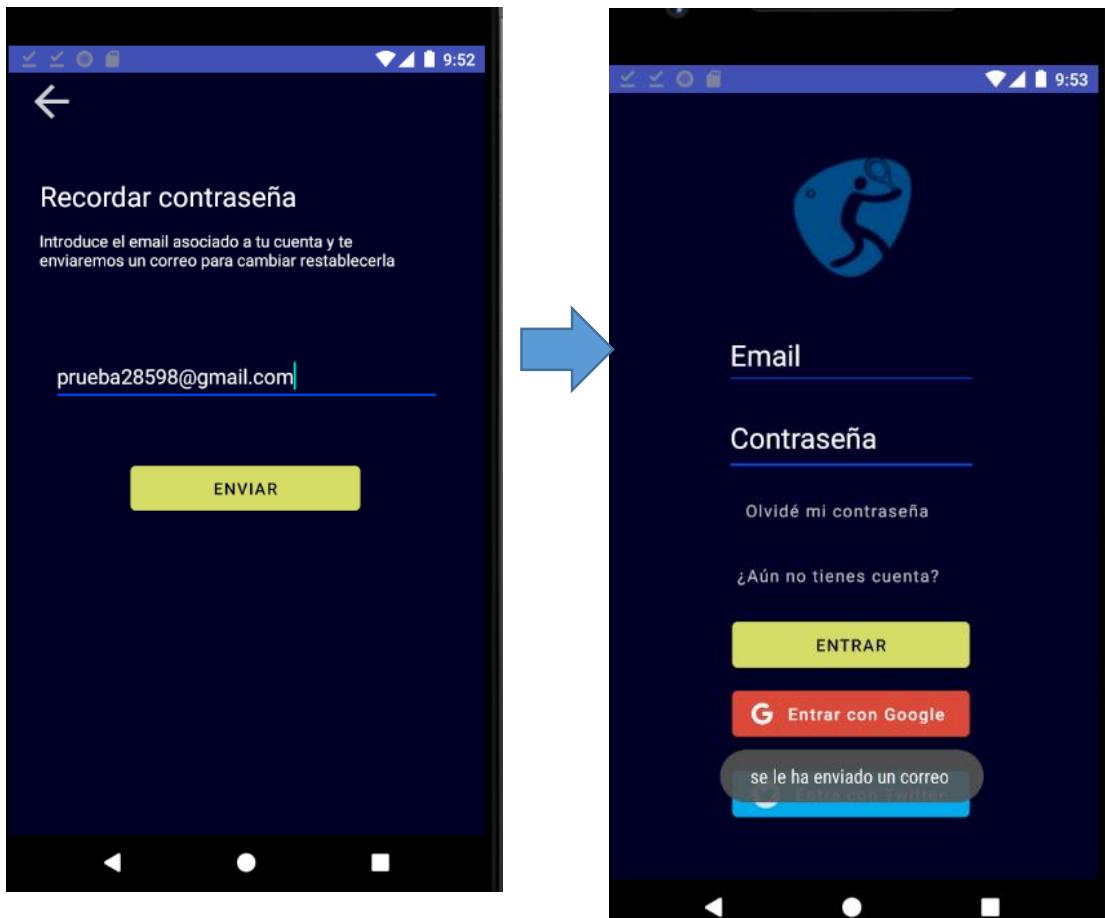


Y en la base de datos:

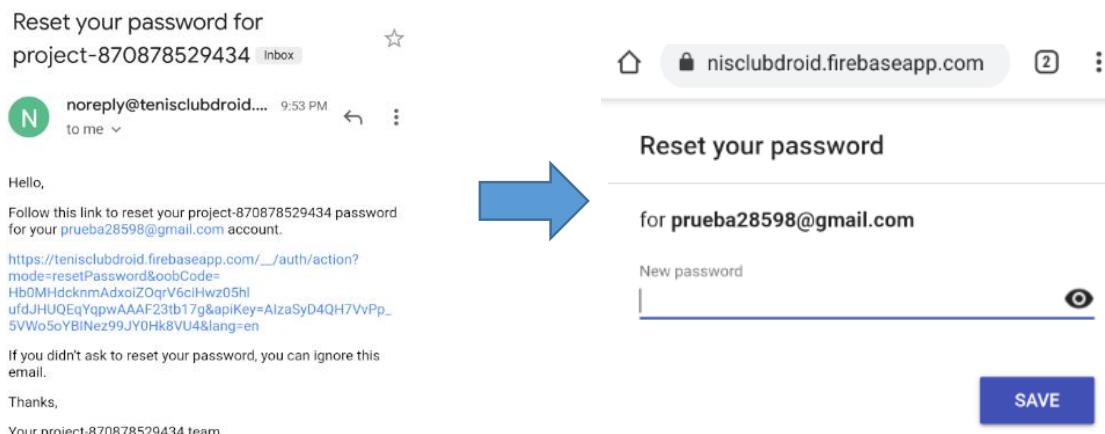
```
- oECTarT8IIXd0trPbSqZzmMmZ9o1
  |-- descripcion: "Me gusta el tenis y tambien los dioses griegos"
  |-- fotoPerfil: "https://firebasestorage.googleapis.com/v0/b/ten...)"
  |-- idUsuario: "oECTarT8IIXd0trPbSqZzmMmZ9o1"
  |-- nickName: "pablo99"
  |-- rol: 0
```

- Caso de prueba de Integración

En este caso de prueba vamos a comprobar que cuando un usuario quiera restablecer su contraseña le llega un correo para ello. Para ello vamos a usar la siguiente cuenta de prueba.



Como podemos ver al usuario le ha llegado el correo y si pulsa el enlace le llevará a una página para cambiar la contraseña.



➤ Caso de prueba de usuarios

La aplicación ha sido probada por distintos usuarios no especializados. Por motivos de la situación actual estos usuarios han sido mis padres y mi hermano. Mi hermano tiene 26 años (dentro de nuestro target de público) y no ha tenido ningún problema para utilizar la aplicación y la ha encontrado intuitiva, mis padres se salen un poco más del target y tampoco son muy buenos con las tecnologías, pero siguiendo el manual de usuario han sido capaces de utilizar la aplicación.

6.2 Solución a problemas encontrados

La mayoría de problemas han surgido por el hecho de tener que formarme para utilizar Kotlin Native junto a Firebase. Pero tampoco ha sido una gran dificultad.

El problema que se salió de eso fue el hecho de controlar que un nickname no estuviese repetido. Fueron un par de días de mirar fijamente a la pantalla e investigar mucho por internet, el problema no era en sí como controlarlo pues era sencillo crear un documento nuevo en la base de datos para ir guardando los nicknames que se utilizaban. Mi problema surgió del método de Firebase que tenía que utilizar (`onDataChange()`) puesto que yo quería dejar el código bonito y dejar un método para comprobarlo, fue más tarde cuando me di cuenta que ese método es asíncrono y tardaba un poco más en realizarse de lo que la app lo hacía.

Asique mi solución (de las muchas posibles que hay) fue meter la lógica de registrar / modificar un usuario dentro de ese método, de esta forma no había inconvenientes en que fuera asíncrono.

7. Lanzamiento y puesta en marcha

7.1 Aspectos relevantes del despliegue y puesta en marcha del sistema

En este punto vamos a ver los aspectos más relevantes del despliegue y puesta en marcha de nuestra aplicación. Nuestra aplicación tiene 3 partes distintas que realizan un despliegue diferente, son la app móvil (para Android e iOS), la aplicación de escritorio (para administrar) y finalmente la parte de backend del servidor y base de datos.

Para el despliegue de nuestra app móvil tenemos que seguir dos caminos distintos, el primero para poder subirla y ponerla en la PlayStore de Google (que es donde más recepción puede tener) y el segundo para poder subirla en la AppStore de Apple (la única para esta plataforma).

➤ Para desplegarla en Android deberemos:

1. Crear una cuenta de desarrollador, para ello habrá que acceder a Google Play y abonar 25 dólares
2. Acceder la consola de Google Play (de desarrollador), ir al apartado de crear aplicación.
3. Seleccionar un idioma predeterminado y poner el título de la aplicación
4. Especificamos que nuestra app es una aplicación y especificar si es de pago o gratuita.
5. Configurar la ficha de la PlayStore con el icono de la app y más detalles sobre ella.
6. Generemos nuestra APK (en nuestro caso Android Studio ofrece esa opción de una forma cómoda).
7. Si hemos especificado que es de pago estableceremos el precio y la distribución.
8. Finalmente enviaremos la aplicación para que sea revisada y nos notificarán cuando esté lista para subir a la PlayStore.

➤ Para desplegarla en iOS deberemos:

1. Nos deberemos de registrar como desarrolladores de Apple e introducir nuestra tarjeta de crédito
2. Crearemos un certificado de Apple (para ello utilizaremos Xcode, el IDE de Apple)
3. Registrar un dispositivo móvil (móvil o Tablet) en la cuenta de desarrollador.
4. Generaremos un id para la app y un perfil de aprovisionamiento.
5. Publicaremos la app a través de Itunes Connect.
6. Crearemos el diseño de la página de descripción de la app y subiremos el archivo build que se ha generado en el paso 2.
7. Ahora tendremos que esperar unos días hasta que revisen nuestra app y esté en la AppStore.

Para el despliegue de la aplicación de escritorio tendremos menos problemas, puesto que hay muchas formas de crear un archivo .exe (junto con su instalador) desde java. Uno de los métodos más utilizados es mediante el software Launch4j. Seguiremos el wizard que proporciona el programa y de una forma sencilla y cómoda podremos tener preparada para la distribución en el sistema del administrador nuestra aplicación. Para mas información:

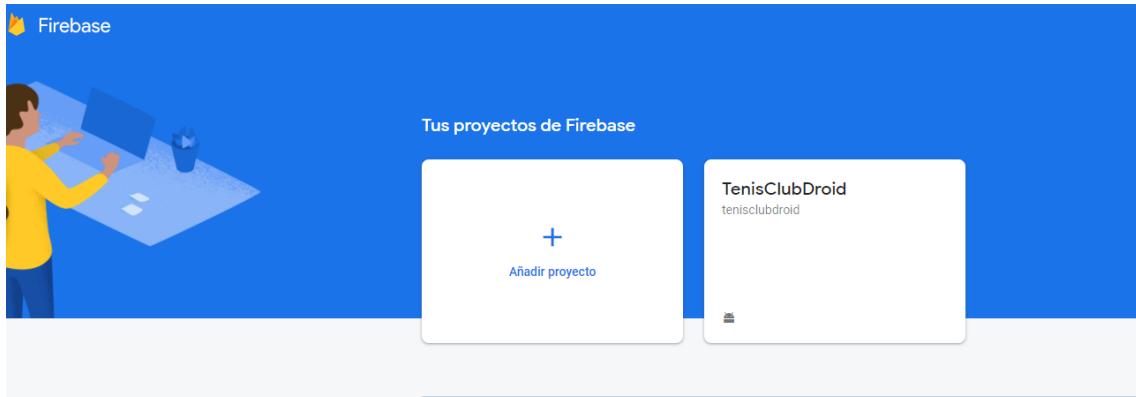
<http://launch4j.sourceforge.net/>

https://www.youtube.com/watch?v=8p-H_SDCbQU&feature=emb_logo&ab_channel=FCOo.O

Finalmente trataremos el despliegue de nuestro servidor. Al estar todo nuestro backend basado en Firebase la puesta en marcha es invisible, puesto que desde el momento que empezamos a utilizar Firebase para el desarrollo ya estaba desplegado.

Los pasos de dicho “despliegue” fueron:

Crear nuestro proyecto con un nombre y seguir los pasos que te da la web.



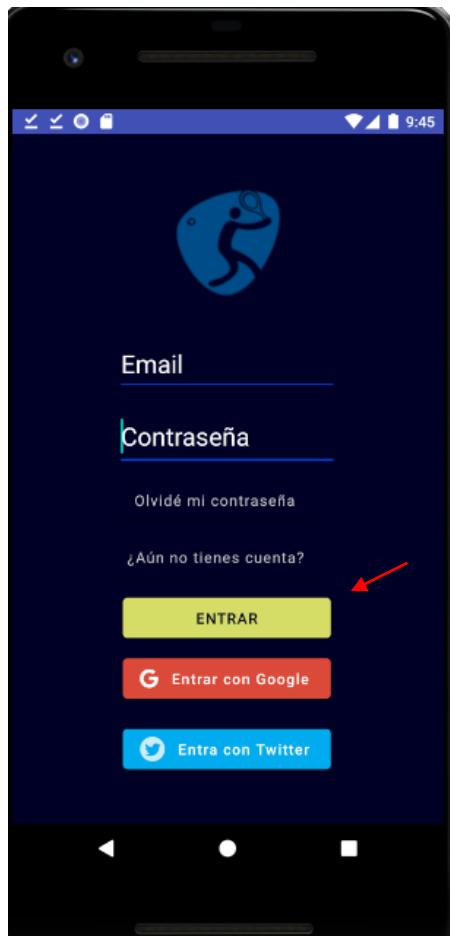
Una vez hecho ya tendremos lo más importante del backend, ya solo nos quedará configurar una base de datos y habilitar los servicios de autentificación que hemos utilizado (email y google)

The image contains two screenshots of the Firebase console. The top screenshot shows the main dashboard for the project "TenisClubDroid". It includes a summary bar with "TenisClubDroid" and "Plan Spark", a "Analytics" section showing "Usuarios activos diarios" (3 users, +200%), and a line chart for "Re" (0). The sidebar on the left lists various services: Información general, Compilación, Authentication, Cloud Firestore, Realtime Database, Storage, Hosting, Functions, and Machine Learning. The bottom screenshot shows the "Realtime Database" screen. It has tabs for Datos, Reglas, Copias de seguridad, and Uso. The main area displays the database structure under "tenisclubdroid-default-rtbd": "NombresCgidos" and "usuarios". A detailed view of the "usuarios" node is shown, listing fields like auth_provider_x509_cert_url, auth_uri, client_email, client_id, client_x509_cert_url, private_key, private_key_id, project_id, token_uri, type, and type: service_account. At the bottom, it says "Ubicación de la base de datos: Bélgica (europe-west1) [beta]".

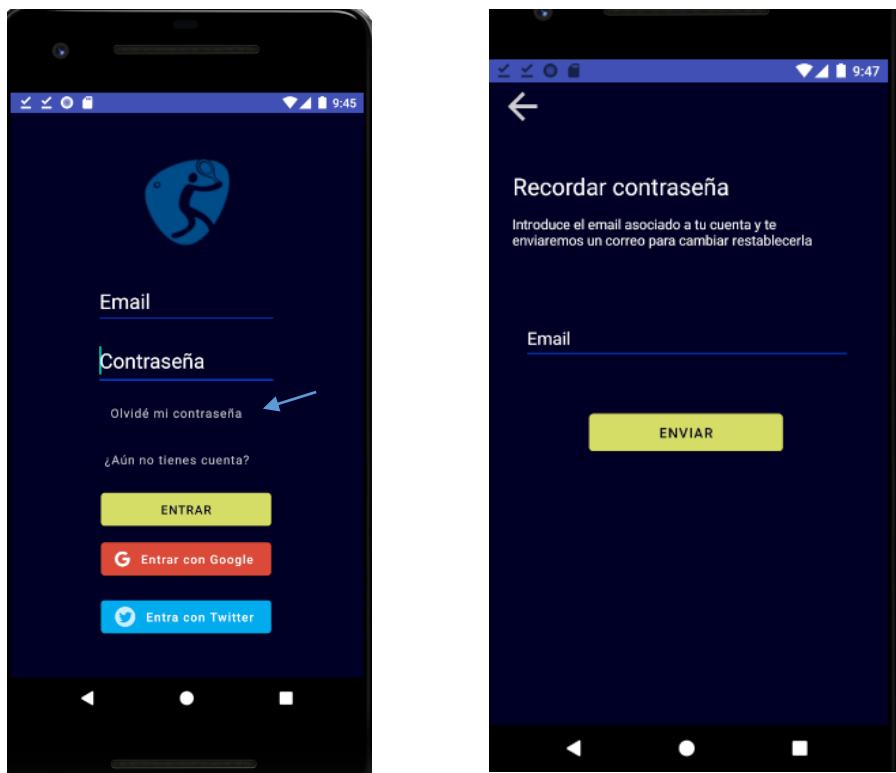
7.2 Manual de usuario

En este punto vamos a enseñar el manual de usuario para que, aunque la app sea utilizada por alguien inexperto siguiendo dicho manual sea capaz de explotar el potencial de TenisClubDroid.

Lo primero que se va a encontrar el usuario (tras un splashscreen con el logo y nombre) será esta pantalla de inicio de sesión. En ella tendrá que introducir sus datos de acceso, correo electrónico y contraseña, y pulsará el botón de entrar (flecha roja).



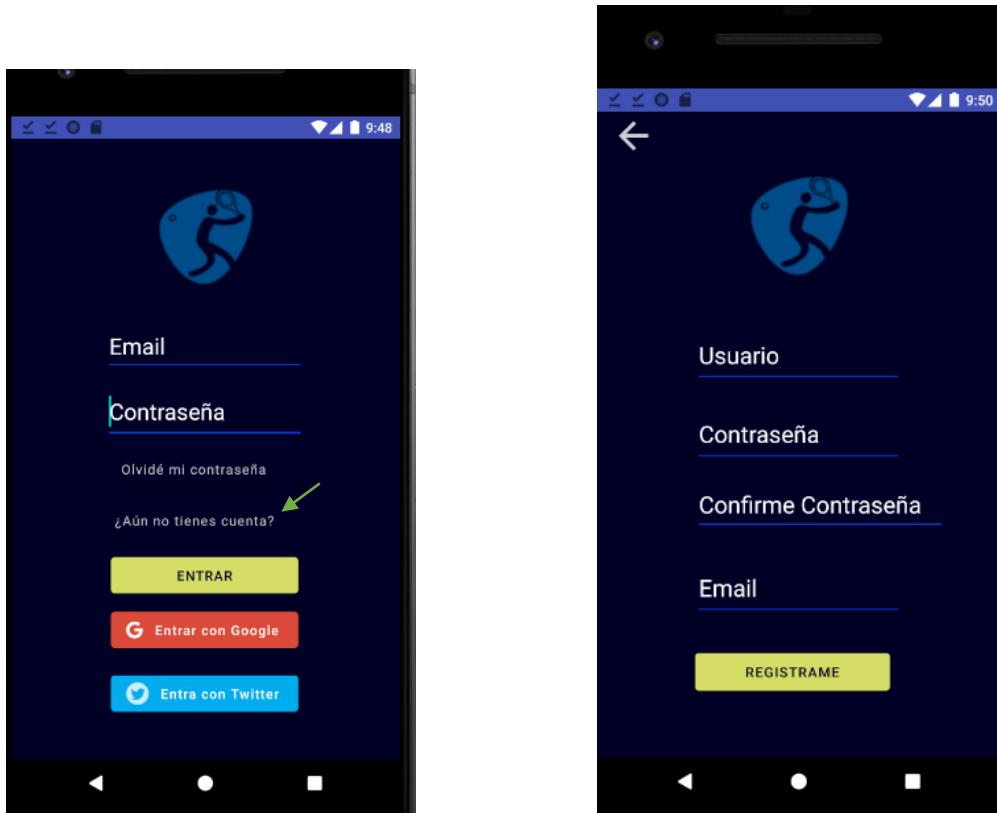
En caso de que no recuerde su contraseña pulsará en “Olvidé mi contraseña” (flecha azul). Al hacerlo saldrá la siguiente pantalla donde deberá de poner su email y si todo ha ido bien le llegará a ese correo unas instrucciones para restablecer su contraseña.



El usuario también puede acceder mediante su cuenta de google. Para ello simplemente tendrá de pulsar “Entrar con Google”



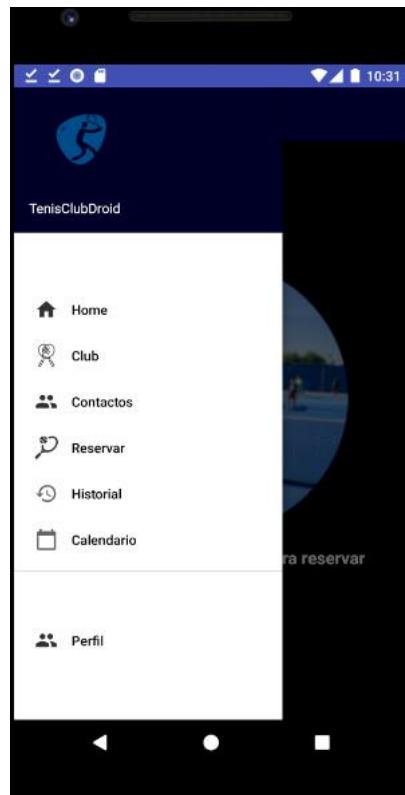
En caso de no tener cuenta pulsará “¿Aun no tienes cuenta?” (flecha verde) y se le mostrará la siguiente pantalla, rellenando todos los campos sin que tenga nombre de usuario o email repetido (por otro usuario) podrá darse de alta en nuestra app.



Una vez accede a la aplicación llegará a la pantalla de inicio de la aplicación.



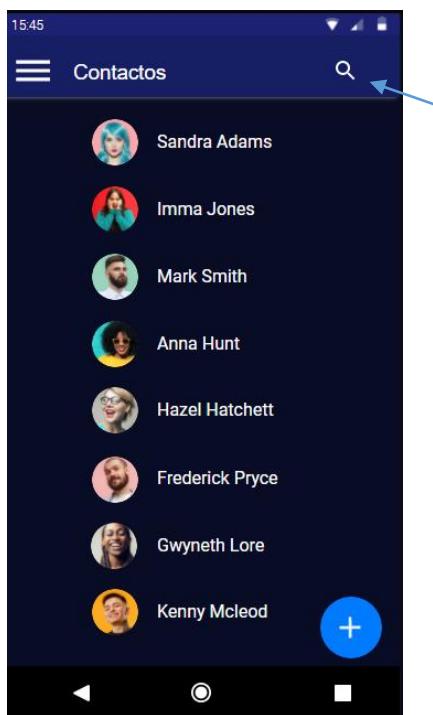
Una vez en la app podrá moverse por la aplicación (solo disponible Home, Club, Contactos, Reservar y Perfil)



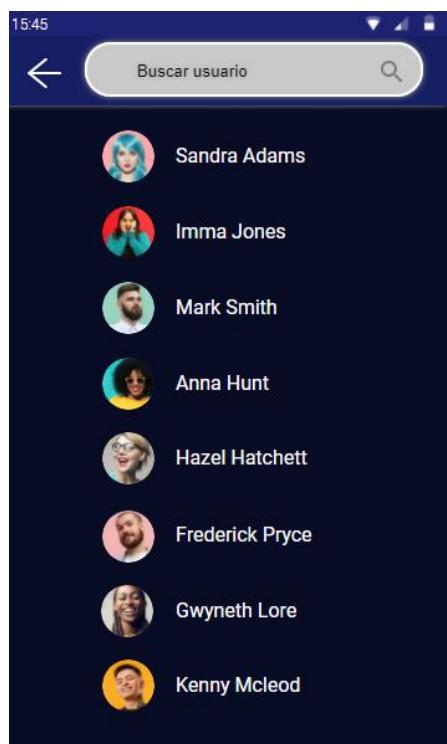
En el apartado de **club** se puede deslizar las fotos para ver toda la galería y pulsando en el mapa le llevará a google maps y le mostrará la ubicación para poder ir al club.



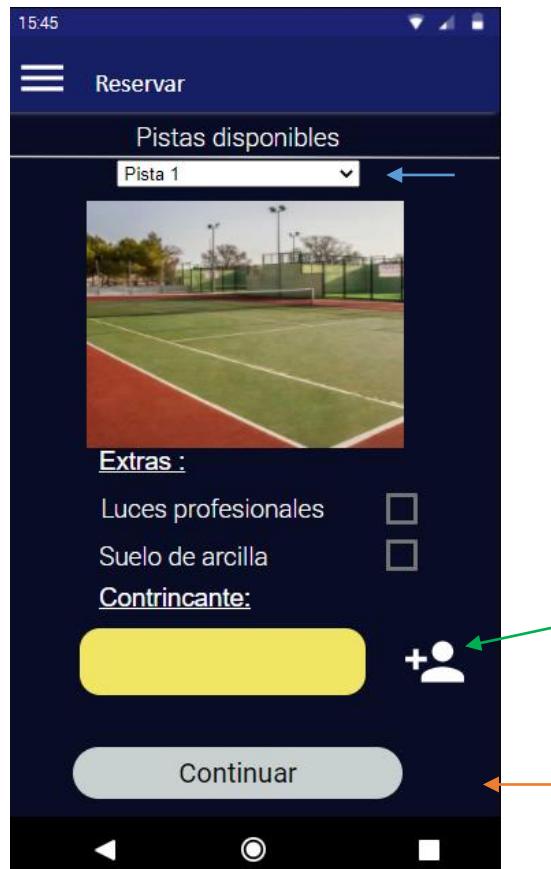
En el apartado de **Contactos** podremos ver los usuarios que tenemos guardados.



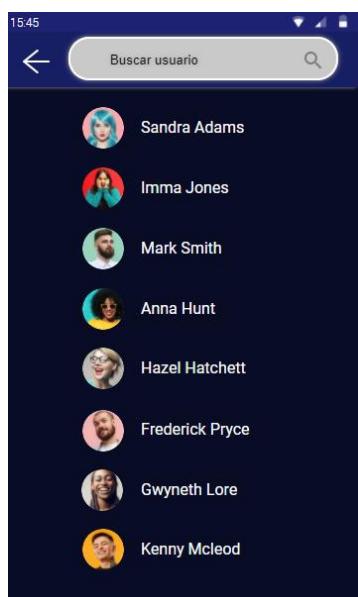
Al pulsar en la lupa saldrá un buscador para que puedas filtrar tus contactos.



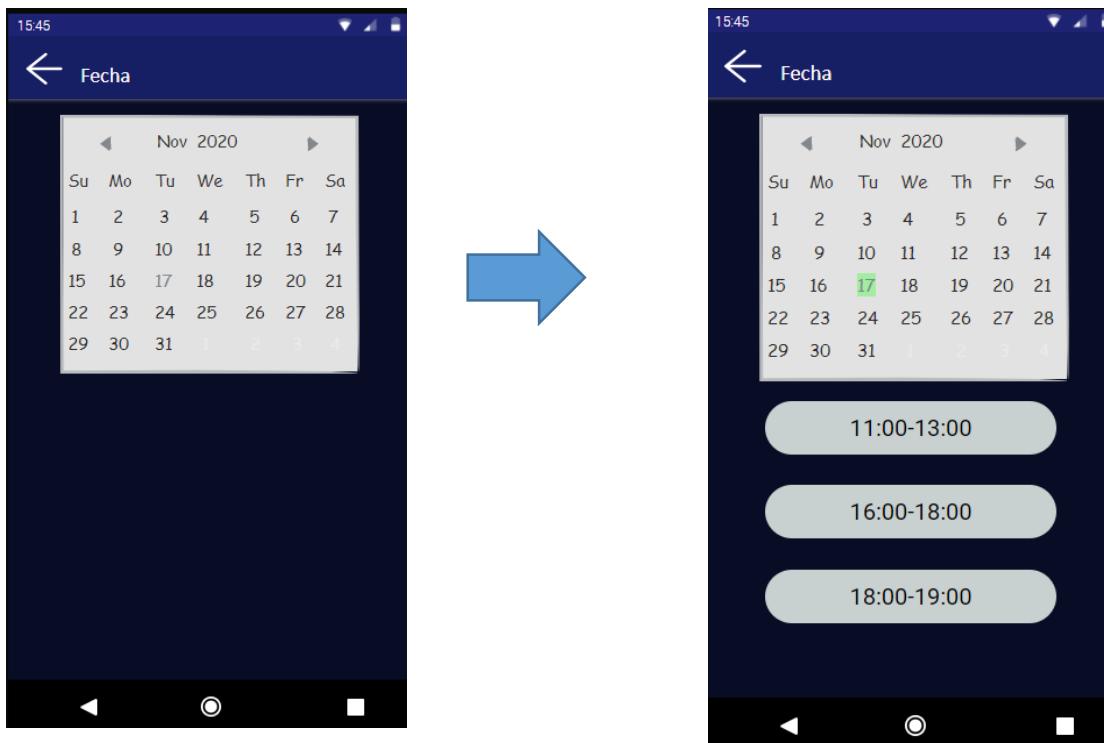
En el apartado de **reservar** se llevarán a cabo las reservas de las pistas, primero podrá elegir la pista (entre las que estén disponibles) mediante el desplegable (flecha azul), apareciendo su foto y sus extras y opcionalmente añadir un usuario contrincante (flecha verde), para ello pulsando el botón de añadir usuario (flecha naranja).



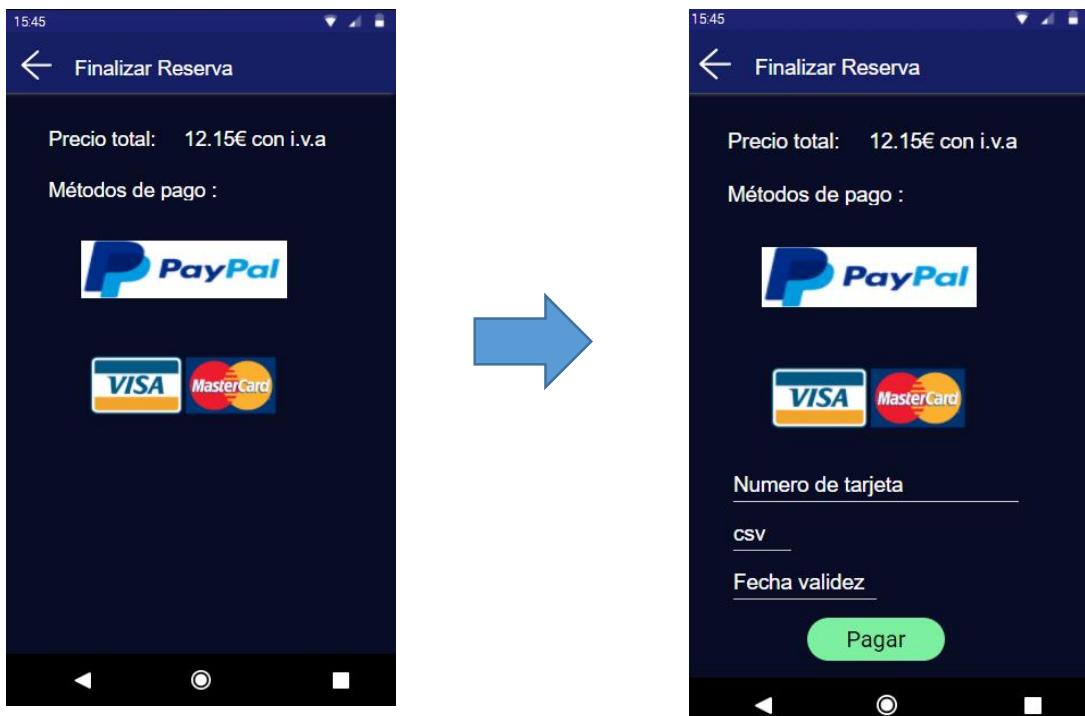
Al pulsar en añadir un contrincante saldrá el buscador



Tras la primera ventana podrá elegir la fecha y la hora (entre las disponibles):



Una vez elegida la fecha y hora se pasará a la ventana de pago, donde se ve el precio total y se dará opción a pagar mediante tarjeta visa o MasterCard, donde nos mostrará los campos para rellenar, una vez pagado nos llevará a la página principal.



Finalmente, el usuario podrá ver su perfil, con su imagen, nombre de usuario y perfil. Al pulsar el botón de cerrar sesión (flecha roja) podrá salir al login de nuevo. Para editar su perfil le dará al botón del lápiz (flecha verde) podrá editar su perfil.



Pulsando el de la cámara podrá elegir una foto desde la galería, podrá cambiar su nombre de usuario (si no está usado) y su descripción. Al pulsar actualizar se guardarán sus cambios.



8.Valoracion y conclusiones

Para la valoración quiero ver varios aspectos para crear una imagen global del proyecto:

Durante el desarrollo de este proyecto he encontrado varias dificultades que he tenido que ir solventando. Lo que más me costó fue acostumbrarme a tener que hacer una documentación “completa”, ya que durante el curso fuimos haciendo, en la asignatura de “Programación multimedia y dispositivos móviles”, mini documentaciones de las aplicaciones que íbamos creando, nunca había hecho una estimación de costes o una planificación al uso. Pero más o menos gracias a los consejos de mi tutor y con la ayuda de google he ido saliendo del paso.

Otras de las dificultades ha sido tener que realizar la aplicación móvil para dos plataformas distintas, como son iOS y Android, y no poder utilizar java para el desarrollo como hemos hecho durante el ciclo. Pero creo que, aunque al principio me costó un poco acostumbrarme a Kotlin Native con el tiempo fui viviendo que era más ágil a la hora de programar y al utilizar kotlin es muy parecido en cuanto a lógica a java y solo cambiaba la sintaxis. Asíque, aunque fue una dificultad ahora que he terminado creo que ha sido provechoso y de todas maneras en este sector siempre hay que ir renovándose asíque ha sido una forma de experimentarlo sin salir al mundo del trabajo.

En este proyecto también he podido aplicar los conocimientos que he ido adquiriendo durante el ciclo como el pensamiento abstracto para programar obtenido de Programación de servicios y procesos y también de Programación multimedia y dispositivos móviles. Para todo lo relacionado a las bases de datos ha sido útil el conocimiento de Acceso a datos (aunque me haya decantado por una bbdd no relacional) y para todas las interfaces he tenido a mano lo aprendido en Desarrollo de Interfaces. También ha sido útil para tener conceptos claros tanto FOL como Empresa e iniciativa emprendedora. De otros modulos también he podido aprovechar cosas pero son mas particulares.

Como conclusión, creo que pese a las dificultades ha sido un viaje muy formativo en el que he aprendido muchas tecnologías y formas de pensar que me va a ser muy útil en mi futura carrera laboral.

9.Bibliografia

<https://firebase.google.com/docs?hl=es>

<https://docs.github.com/es/free-pro-team@latest/github/managing-your-work-on-github/about-project-boards>

<https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow>

<https://support.gitkraken.com/git-workflows-and-extensions/git-flow/>

<https://blog.axosoft.com/gitflow/>

<https://www.gladysgbegnedji.com/estimar-recursos-de-las-actividades/>

<https://www.semantic-systems.com/semantic-noticias/articulos-tecnologicos/gestion-de-riesgos-en-proyectos-de-implantacion-de-software/>

<https://ocw.unican.es/pluginfile.php/274/course/section/196/Lista%20de%20Riesgos%20en%20proyectos%20SW.pdf>

https://es.wikiversity.org/wiki/Gesti%C3%B3n_de_riesgos_de_proyectos_software

https://es.wikiversity.org/wiki/Gesti%C3%B3n_de_riesgos_de_proyectos_software

<https://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html>

https://es.wikipedia.org/wiki/Prueba_de_integraci%C3%B3n

<https://es.stackoverflow.com/>

https://www.youtube.com/watch?v=vjVyUKVNd3U&list=PLavhGrYBf3VlvEDwxunm8Mn2wjFj5VHwf&index=1&t=240s&ab_channel=CodeAndroid

https://www.youtube.com/watch?v=lS3EOKshtyM&list=PLavhGrYBf3VlvEDwxunm8Mn2wjFj5VHwf&index=2&t=1421s&ab_channel=LearningWorldz

https://www.youtube.com/watch?v=dpURgJ4HkMk&ab_channel=MoureDevbyBraisMoure

<https://stackoverflow.com/questions/37390864/how-to-delete-from-firebase-realtime-database>

<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagramas-de-clases-con-uml/>

<https://www.mongodb.com/nosql-explained>

<https://eaminds.com/2018/08/03/modelando-nosql-data-bases/>