APLICACIÓN MÓVIL MULTIPLATAFORMA DE APOYO A LA ATENCIÓN FISIOTERAPÉUTICA DE NEONATOS

Alumno

Adrián Louro Alonso

Máster Universitario en Ingeniería Informática (Programación de Dispositivos Móviles)

Tutor

Dr. Alexis Quesada Arencibia

Ciencias de la Computación e Inteligencia Artificial

Enero 2019 – Las Palmas de Gran Canaria

Agradecimientos

A mi tutor Dr. Alexis Quesada por haberme acompañado en un nuevo proyecto.

A mi familia y amigos por haberme apoyado durante todos estos años.

Resumen

El Servicio Canario de la Salud dispone de un servicio de fisioterapia para atender a neonatos que requieren asistencia profesional. En él, los tutores recurren a plataformas de mensajería para preguntar a los fisioterapeutas acerca de los ejercicios que deben hacer los bebés y hacerles llegar imágenes o vídeos de las sesiones que realizan con ellos.

Para los tutores, puede ser complicado realizar los ejercicios con los bebés sin material de apoyo y para los fisioterapeutas es complicado realizar el seguimiento sin recibir la información de una manera organizada.

Por ello, proponemos una aplicación móvil que contenga material de apoyo para los tutores a la hora de realizar las terapias y que además permita realizar su seguimiento de forma organizada.

Abstract

The Servicio Canario de la Salud has a physiotherapy service to care for newborns who require profesional assistance. In it, tutors use messaging platforms to ask the physiotherapists about the exercises that babies should do and send them images or videos of the sessions they perform with them.

For tutors, it can be complicated to perform the exercises with the babies without support material and for the physiotherapists it is complicated to monitor the therapy without receiving the information in an organized way.

For this reason, we propose a mobile application that contains support material for the tutors at the time of performing the therapies and that also allows monitoring them in an organized way.

Índice de contenido

Índice de ilustraciones

Índice de tablas

Introducción

El Servicio Canario de la Salud dispone de un servicio de fisioterapia para atender y realizar el seguimiento de neonatos con necesidad de asistencia profesional. Este servicio conlleva la realización de una serie de ejercicios para corregir malos hábitos del bebé que pueden derivar en futuros problemas físicos.

Los fisioterapeutas recomiendan a los tutores que realicen estos ejercicios en casa, por lo que no pueden comprobar in situ si los ejercicios se están realizando correctamente. Esto hace que los tutores recurran a plataformas de mensajería como WhatsApp para hacer llegar a los fisioterapeutas imágenes o vídeos de las distintas sesiones que tienen con los bebés. También, suelen utilizar estas plataformas con frecuencia para preguntar cualquier duda que tienen acerca de los ejercicios.

Este sistema presenta una serie de inconvenientes. Los tutores solo cuentan con material de apoyo impreso para ayudarles con la realización de los ejercicios, los cuales pueden tener cierto grado de dificultad. Además, el seguimiento de la evolución del bebé es bastante tedioso, puesto que los fisioterapeutas reciben una gran cantidad de mensajes, mediante plataformas que no están preparadas para este tipo de servicio, que no pueden visualizar de forma organizada.

En el curso académico 2017/2018, se presentó el Trabajo de Fin de Grado, titulado "*App asistencial para niños prematuros*" y realizado por Cristian Manuel Suárez Vera, cuyo objetivo era desarrollar un primer prototipo de aplicación móvil para facilitar el seguimiento de estas terapias. En este prototipo:

- Un fisioterapeuta asigna ejercicios precargados en una base de datos a un tutor
- Un tutor puede ver ejercicios marcados
- Un tutor puede marcar un ejercicio como "Hecho"
- Un fisioterapeuta puede ver los ejercicios asociados a un tutor y si fueron realizados
- Tutores y fisioterapeutas pueden enviarse mensajes mediante un chat en tiempo real

Sin embargo, no se contempla la figura del neonato, por lo que si éste está a cargo de varios tutores éstos no comparten la información. Además, no se permite que los fisioterapeutas modifiquen la base de datos de ejercicios desde la aplicación. Tampoco se permite que los tutores envíen a los fisioterapeutas información acerca de las distintas sesiones de ejercicios que tienen con los bebés, por lo que el seguimiento de la evolución del bebé sigue sin estar cubierto.

Por ello, proponemos un nuevo sistema que, aprovechando las ventajas de las tecnologías de la información, haga que las terapias del servicio de fisioterapia del Servicio Canario de la Salud se lleven a cabo de forma sencilla y eficiente, evitando todos los problemas del sistema de terapia actual y mejorando el primer prototipo de aplicación móvil desarrollado.

A lo largo del presente documento haremos referencia a tutor o tutores para referirnos a la madre, padre, tutora o tutor legal de un bebé.

Estructura del documento

El documento se encuentra estructurado en diferentes capítulos. A continuación, se explican los distintos capítulos que han sido desarrollados:

- Capítulo 1: Estado actual y objetivos iniciales

  En este capítulo se explica el estado actual del problema y los objetivos que se pretenden alcanzar con el desarrollo de este proyecto.

- Capítulo 2: Competencias específicas cubiertas

  En este capítulo se enumeran y justifican las distintas competencias atribuidas al Máster Universitario en Ingeniería Informática cubiertas por este proyecto.

- Capítulo 3: Aportaciones

  En este capítulo se describen las distintas aportaciones del proyecto tanto en el entorno socioeconómico como a nivel personal.

- Capítulo 4: Normativa y legislación

  En este capítulo se presenta un breve estudio acerca de la normativa y la legislación vigente relevante sobre este proyecto. También, se enumeran y describen las licencias software utilizadas para realizar el proyecto.

- Capítulo 5: Metodología de trabajo y planificación del proyecto

  En este capítulo se explica y justifica la metodología de trabajo seguida, así como la planificación inicial del proyecto y los diferentes ajustes realizados sobre ella.

- Capítulo 6: Tecnologías y herramientas utilizadas

  En este capítulo se hace una breve descripción de las distintas tecnologías y herramientas que han sido utilizadas a lo largo del desarrollo del proyecto.

- Capítulo 7: Análisis

  En este capítulo se explica la fase de análisis del proyecto. Se comentan los distintos actores que hacen uso de la aplicación móvil y se explican los requisitos del sistema que han sido identificados mediante diagramas de casos de uso y la especificación de los casos de uso más relevantes.

- Capítulo 8: Diseño

  En este capítulo se explica la fase de diseño del proyecto. Se explica y justifica la arquitectura del sistema escogida, se habla sobre el diseño de las bases de datos utilizadas y, por último, se explican los distintos aspectos de diseño que se han tenido en cuenta a la hora de implementar la interfaz de usuario de la aplicación móvil.

- Capítulo 9: Desarrollo

  En este capítulo se explica la fase de desarrollo del proyecto. Se explica y justifica la elección de los *frameworks* de programación utilizados, el sistema de control de acceso de la

aplicación, las capas de datos y de acceso a datos de la aplicación, la comunicación entre la aplicación móvil y el servidor y los distintos patrones de diseño que han sido implementados.

- Capítulo 10: Pruebas

  En este capítulo se comentan las distintas pruebas realizadas a lo largo del proyecto para comprobar el correcto funcionamiento del sistema, así como la calidad de la interacción entre los usuarios y la aplicación móvil.

- Capítulo 11: Resultados, conclusiones y trabajo futuro

  En este capítulo se comentan los resultados y conclusiones obtenidas tras la finalización del proyecto y se comentan posibles mejoras y ampliaciones para el mismo.

- Capítulo 12: Bibliografía

  En este capítulo se presenta la bibliografía sobre la que se apoya este documento.

- Anexos

  En este apartado se incluye información complementaria del proyecto realizado.

  - Anexo I: se explica de manera clara y sencilla cómo se utiliza la aplicación móvil.
  - Anexo II: se explica cómo desplegar el sistema en un entorno de desarrollo.
  - Anexo III: contiene el *script* escrito en lenguaje SQL utilizado para generar la base de datos MySQL utilizada por la aplicación móvil.

1. Capítulo 1: Estado actual y objetivos iniciales

1.1. Estado actual

A día de hoy, en el servicio de atención fisioterapéutica a neonatos del Servicio Canario de la Salud, los fisioterapeutas realizan el seguimiento a los recién nacidos tanto en la consulta como desde casa.

Cuando es necesario, indican a los tutores que deben realizar una serie de ejercicios, que ya han sido confeccionados, con los bebés para corregir malos hábitos que puedan derivar en problemas físicos en el futuro. Para facilitar que los tutores realicen los ejercicios con los bebés en casa, los fisioterapeutas les entregan documentación impresa con ilustraciones e información acerca de los ejercicios marcados.

Fuera de la consulta, el seguimiento se lleva a cabo mediante la plataforma de mensajería WhatsApp. Esto tiene una serie de inconvenientes como pueden ser:

- Los tutores reciben material impreso con diferentes ilustraciones referentes a los ejercicios, cuya impresión conlleva una serie de gastos en los centros, además de un impacto negativo en el medio ambiente debido al uso masivo del papel
- En ciertas ocasiones, el material impreso de apoyo a la realización de los ejercicios no es lo suficientemente claro y no evita que los tutores tengan que realizar consultas a los fisioterapeutas
- Los fisioterapeutas reciben consultas de los tutores a través una plataforma que no está destinada al ámbito laboral, lo cual hace que se mezclen conversaciones de ámbito personal y laboral
- El seguimiento de los ejercicios realizados por los tutores no se realiza de manera ordenada, puesto que WhatsApp no permite organizar la información de las distintas sesiones que tienen los tutores con los bebés

1.2. Objetivos

Para solventar los inconvenientes citados anteriormente, el objetivo de este Trabajo de Fin de Título consiste en el desarrollo de una aplicación móvil multiplataforma que permita que los fisioterapeutas hagan llegar a los tutores contenido de apoyo (escrito y multimedia) para que realicen los ejercicios y que el seguimiento de la evolución del bebé se realice de manera cómoda y eficiente.

Gracias a esta aplicación, los fisioterapeutas podrán:

- Crear una base de datos de ejercicios con contenido multimedia
- Hacer llegar distintos ejercicios de la base de datos a los tutores de sus pacientes y establecer ciertos parámetros para que se ajusten a cada caso particular
- Visualizar las sesiones llevadas a cabo por los tutores con los bebés
- Establecer valoraciones o apuntes acerca de dichas sesiones

A su vez, los tutores podrán:

- Ver los ejercicios enviados por los fisioterapeutas
- Hacer llegar a los fisioterapeutas imágenes o vídeos de las distintas sesiones que tienen con los bebés

Además, se implementará un sistema de envío de mensajes en tiempo real entre tutores y fisioterapeutas que permitirá a los tutores realizar cualquier tipo de consulta a los fisioterapeutas, para que estos puedan resolver sus dudas.

De esta manera, buscamos centralizar toda la información asociada al seguimiento de los bebés y evitar que los tutores recurran a otras plataformas de mensajería, las cuales no han sido creadas para este propósito, para contactar con los fisioterapeutas.

2. Capítulo 2: Competencias específicas cubiertas

2.1. TI01

"Capacidad para modelar, diseñar, definir la arquitectura, implantar, gestionar, operar, administrar y mantener aplicaciones, redes, sistemas, servicios y contenidos informáticos."

Durante este proyecto se han abarcado las fases de análisis, diseño y desarrollo de un servicio mediante el uso de las TIC. Se ha diseñado e implementado la arquitectura de un sistema informático, así como una aplicación informática que hace uso de las redes de comunicaciones para interactuar con otros usuarios.

2.2. TI05

"Capacidad para analizar las necesidades de información que se plantean en un entorno y llevar a cabo en todas sus etapas el proceso de construcción de un sistema de información."

En este Trabajo de Fin de Título se ha realizado el estudio de un servicio ya implantado en una organización para actualizarlo mediante el uso de un sistema informático, con el objetivo de mejorar ciertos aspectos que han quedado anticuados tras la aparición de las tecnologías de la información.

2.3. TI10

"Capacidad para utilizar y desarrollar metodologías, métodos, técnicas, programas de uso específico, normas y estándares de computación gráfica."

En este proyecto se hace hincapié en la interacción de los usuarios con la aplicación mediante el diseño de una interfaz gráfica que tiene en cuenta distintos principios y técnicas de diseño de interacción para asegurar que la experiencia del usuario sea lo más gratificante posible.

2.4. TI11

"Capacidad para conceptualizar, diseñar, desarrollar y evaluar la interacción persona–ordenador de productos, sistemas y servicios informáticos."

Para este proyecto se ha diseñado y desarrollado una interfaz gráfica en una aplicación móvil con el objetivo de que los usuarios que hacen uso del servicio puedan interactuar cómodamente con sistema desarrollado. Se han utilizado distintos principios de diseño de interacción para asegurar que la interacción entre los usuarios y el sistema sea cómoda y sencilla.

2.5. TI12

"*Capacidad para la creación y explotación de entornos virtuales, y para la creación y distribución de contenidos multimedia.*"

Uno de los pilares de este proyecto es hacer llegar contenido multimedia mediante una aplicación móvil, en formato de imagen o vídeo, a fisioterapeutas y tutores para facilitar la realización de los ejercicios y el seguimiento de las terapias.

3. Capítulo 3: Aportaciones

3.1. Entorno socioeconómico

El desarrollo de este proyecto ofrecerá una serie de mejoras en la calidad del servicio de fisioterapia de neonatos del Servicio Canario de la Salud.

A los fisioterapeutas, les proporcionará un sistema rápido, organizado y sencillo que les permitirá realizar el seguimiento de sus pacientes sin tener que recurrir a otras plataformas de mensajería que no están especializadas en este ámbito.

En cuanto a los tutores de los neonatos, el poder contar con documentación en formato de vídeo acerca de los ejercicios propuestos por los fisioterapeutas hace que su realización sea más sencilla y correcta.

Tanto un buen seguimiento por parte del fisioterapeuta como una correcta realización de los ejercicios llevada a cabo por los tutores harán que los recién nacidos que tengan la oportunidad de recibir este servicio gocen de mejor salud, puesto que se corregirán problemas que aparecen durante los primeros meses de vida de los bebés y que pueden derivar en alguna complicación en el futuro.

También, al no hacer uso de material de apoyo impreso los centros de salud ahorrarán los recursos necesarios para su impresión. No podemos olvidar que este punto también es un aspecto positivo para la conservación del medio ambiente, ya que se suprime el uso del papel.

En resumen, podemos decir que la realización de este Trabajo de Fin de Título tendrá consecuencias positivas en:

- La salud de las nuevas generaciones
- El ahorro de recursos por parte de los centros de salud
- El medio ambiente

3.2. Personal

A nivel personal, este proyecto ha sido una experiencia muy enriquecedora. Ha sido llevado a cabo desde cero, abarcando las distintas fases del ciclo de vida del software, desde el análisis hasta el desarrollo. Este aspecto es muy importante para cualquier ingeniero de software, ya que las fases previas al desarrollo son de vital importancia para que cualquier proyecto tenga éxito.

Además, la oportunidad de poder habido contar con la ayuda de una profesional que forma parte del servicio de fisioterapia del Servicio Canario de la Salud, Dña. María del Mar Batista Guerra, me ha permitido tener una experiencia que se vive en el día a día de muchos proyectos reales de software, que consiste en tener que desarrollar un producto o servicio, el cual va evolucionando a lo largo del tiempo, para un cliente que lo solicita.

También, he tenido la posibilidad de aprender nuevas tecnologías que desconocía y que tienen un peso importante en el mundo laboral actual, como pueden ser el uso de bases de datos no

relacionales, el diseño e implementación de APIs REST utilizando .NET Core, el uso de la plataforma Firebase para poder integrar un sistema de chat en tiempo real en la aplicación móvil y el uso de Ionic 4 como *framework* para desarrollar dicha aplicación.

4. Capítulo 4: Normativa y legislación

4.1. Licencias de Software

Una licencia de software establece un contrato entre licenciante, el autor o titular de los derechos de distribución y explotación, y licenciatario, que especifica los términos y condiciones a seguir para el uso del software.

A continuación, se describen las licencias asociadas a las herramientas y tecnologías utilizadas para la realización de este proyecto.

4.1.1. GPL

La Licencia Pública General de GNU [1] es la licencia de derecho de autor más ampliamente utilizada en el mundo del software libre y código abierto. Garantiza a los usuarios finales (personas, organizaciones y compañías) la libertad de usar, estudiar, compartir y modificar el software.

Su propósito es doble: declarar que el software cubierto por esta licencia es libre y protegerlo (mediante una práctica conocida como *copyleft*) de intentos de apropiación que restrinjan esas libertades a nuevos usuarios cada vez que la obra es distribuida, modificada o ampliada.

Las herramientas utilizadas que se encuentran bajo esta licencia son:

- MySQL Workbench
- StarUML
- MySQL
- Git

StarUML [2] utiliza esta licencia, con la salvedad de dos excepciones:

- Permitir vincular librerías y componentes comerciales específicos
- Permitir vincular módulos patentados

El código generado por StarUML no se encuentra sujeto bajo esta licencia, por lo que podemos utilizar el código generado por StarUML para realizar código comercial no sujeto a GPL.

MySQL se encuentra desarrollado bajo una licencia dual GPL/Licencia comercial de Oracle Corporation, por lo que los derechos de autor del código están en poder del autor individual. La licencia GNU GPL de MySQL obliga a que la distribución de cualquier producto derivado se haga bajo esa misma licencia.

Si un desarrollador desea incorporar MySQL en su producto, pero desea distribuirlo bajo otra licencia que no sea la GNU GPL, puede adquirir una licencia comercial de MySQL que le permite hacer justamente eso.

4.1.2. Licencia MIT

La licencia MIT [3] es una licencia de software libre permisiva que posee una excelente compatibilidad de licencia. Esta licencia permite reutilizar software dentro de software propietario. Por otro lado, es compatible con muchas licencias *copyleft*, como la GNU General Public License (software con licencia MIT puede integrarse en software con licencia GPL, pero no al contrario). El texto de la licencia no tiene *copyright*, lo que permite su modificación.

Esta licencia permite reutilizar el software tanto para ser software libre como para ser software no libre, permitiendo no liberar los cambios realizados al programa original.

También permite licenciar dichos cambios con licencia BSD, GPL u otra cualquiera que sea compatible (es decir, que cumpla las cláusulas de distribución).

Con esta licencia se tiene software libre. Ejemplos en los que podría interesar su aplicación serían las licencias duales, si se pretende difundir un estándar mediante una implementación de referencia, o si simplemente se pretende que el producto sea libre sin mayores consideraciones.

Las herramientas utilizadas que se encuentran bajo esta licencia son:

- Ionic
- Angular
- .NET Core
- C#

4.1.3. Apache License

Apache License [4] es una licencia de software libre permisiva creada por la Apache Software Foundation (ASF). La licencia Apache requiere la conservación del aviso de derecho de autor y el descargo de responsabilidad, pero no es una licencia *copyleft*, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.

Las herramientas utilizadas que se encuentran bajo esta licencia son:

- Android Studio
- Apache Cordova
- Typescript
- Entity Framework Core

4.1.4. Postman EULA

Se trata de una licencia propietaria de Postman, Inc. Asociada a la herramienta Postman [5]. Establece que el usuario final tiene todos los derechos del contenido que genera mediante el uso del producto, por lo que es el único responsable de cualquier contenido generado mediante el uso del producto a través de su cuenta. Dicho contenido no debe violar ninguna ley aplicable o cualquier derecho, incluyendo el derecho de propiedad intelectual, de una persona.

4.1.5. Licencia de Software de Microsoft para Microsoft Visual Studio Community 2017

Se trata de una licencia propietaria de Microsoft Corporation asociada a la herramienta Microsoft Visual Studio Community 2017 [6].

Esta licencia, en su forma "individual", permite que una persona pueda utilizar este software para desarrollar y probar una aplicación para fines de venta.

También, establece que no se puede:

- Eludir las limitaciones técnicas del software
- Utilizar técnicas de ingeniería inversa, descompilar o desensamblar el software, así como tampoco derivar el código fuente del software, excepto y únicamente en la medida que lo exijan los términos de licencia de terceros que rigen el uso de ciertos componentes de código abierto que se podrían incluir con el software
- Eliminar, minimizar, bloquear o modificar ninguna notificación de Microsoft o sus proveedores en el software
- Utilizar el software de ninguna manera que esté en contra de la ley
- Compartir, publicar, alquilar o dar el software en préstamo, ni entregarlo como oferta independiente para que otros lo utilicen, así como tampoco transmitir el software ni este contrato a terceros.

4.1.6. Licencia de Software de Microsoft para Microsoft Visual Studio Code

Se trata de una licencia propietaria de Microsoft Corporation asociada a la herramienta Microsoft Visual Studio Code [7].

Esta licencia permite utilizar cualquier número de copias del software para desarrollar y probar aplicaciones, además de ser utilizado para hacer demostraciones de las aplicaciones.

También, establece que no se puede:

- Eludir las limitaciones técnicas del software
- Utilizar técnicas de ingeniería inversa, descompilar o desensamblar el software, así como tampoco derivar el código fuente del software, excepto y únicamente en la medida que lo exijan los términos de licencia de terceros que rigen el uso de ciertos componentes de código abierto que se podrían incluir con el software
- Eliminar, minimizar, bloquear o modificar ninguna notificación de Microsoft o sus proveedores en el software
- Utilizar el software de ninguna manera que esté en contra de la ley
- Compartir, publicar, alquilar o dar el software en préstamo, ni entregarlo como oferta independiente para que otros lo utilicen

4.1.7. Microsoft Office: Office 365 Education Plus for students

Es un conjunto de servicios que permite la colaboración entre profesores y estudiantes en las tareas escolares. Está disponible de forma gratuita para los profesores que trabajen en una institución académica y para los alumnos que estén inscritos actualmente en centros educativos, en nuestro caso, la Universidad de Las Palmas de Gran Canaria.

Este servicio permite que los profesores y los alumnos instalen las aplicaciones completas de Office en un máximo de 5 equipos PC o Mac de forma gratuita.

La herramienta Microsoft Word 2016 utilizada para la redacción de este documento se encuentra bajo esta licencia.

4.2. Seguridad de los datos

4.2.1. Ley Orgánica de Protección de Datos

La Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal (LOPD) [8], es una Ley Orgánica española que tiene por objeto garantizar y proteger las libertades públicas y los derechos fundamentales de personas físicas, como el honor, la intimidad y la privacidad tanto familiar como personal, en todo lo referente al tratamiento de los datos personales.

Según esta Ley, se debe garantizar la protección de los datos personales que figuren en todos aquellos elementos relacionados, principalmente aquella información guardada en bases de datos. Los datos afectados por esta Ley son los denominados como datos de carácter personal (DCP), los cuales hacen referencia a personas físicas registradas sobre cualquier soporte informático.

Por ello, toda la información recogida en base de datos utilizada por la aplicación se encuentra protegida bajo contraseña, la cual solo podrá ser accedida por el por el administrador de la propia base de datos.

4.2.2 Reglamento General de Protección de Datos

El Reglamento General de Protección de Datos (RGPD) es un nuevo reglamento aprobado por el Parlamento y el Consejo Europeo, cuyo objetivo es unificar los regímenes de todos los Estados Miembros en materia de protección de datos. Está en vigor desde el día 25 de mayo de 2016 pero, sin embargo, su cumplimiento es obligatorio desde el 25 de mayo de 2018. El RGPD no deroga la LOPD, por lo que ambas normativas coexisten.

Este reglamento establece que el consentimiento para tratar datos de carácter personal debe ser inequívoco, libre y revocable y deberá darse mediante un acto afirmativo claro. Por este motivo, la aceptación de la Política de Privacidad es obligatoria para poder darse de alta en el sistema.

Este reglamento también establece que el tratamiento de datos de carácter personal por parte de las organizaciones debe ser transparente, por lo que tienen el deber de informar a los interesados. Por ello, en la Política de Privacidad se debe informar acerca de:

- Datos de contacto del Responsable y Delegado de Protección de Datos
- Los datos que se recaban
- Finalidad del tratamiento
- Legitimación del tratamiento de datos
- El plazo durante el cual se conservarán los datos personales
- Derechos ARCOPOL
- Derecho a reclamar ante la AGPD

4.2.3. Clasificación de los datos

El sistema desarrollado recoge datos de carácter personal acerca de los fisioterapeutas, los tutores de los neonatos y los propios neonatos. A continuación, presentamos una clasificación acerca del nivel de seguridad de los datos de carácter personal según lo establecido por la Agencia Española de Protección de Datos (AGPD).

Datos	Descripción	Nivel
Nombre y apellidos	Nombre y apellidos de fisioterapeutas, tutores y neonatos	BAJO
Número de teléfono	Número de teléfono de tutores	BAJO
Número de colegiado	Número de colegiado de los fisioterapeutas	BAJO
ID historial clínico	Identificador del historial clínico del neonato	BAJO
Fecha de nacimiento	Fecha de nacimiento del neonato	BAJO
Sexo	Sexo del neonato	BAJO
Historial clínico	Información relevante acerca del historial clínico del neonato	ALTO

Tabla 1: Clasificación de los datos de carácter personal según su nivel de seguridad

En relación a los datos anteriores, se garantiza que se recogerán con fines determinados explícitos y legítimos y no se utilizarán para otros fines. Así mismo, se garantiza que los datos son adecuados, pertinentes y no excesivos en relación con esa finalidad, sin exigirse datos de más.

4.2.4. Protección de los datos

De acuerdo con el artículo 96 del Real Decreto 1720/2007, de 21 de diciembre [9]:

"A partir del nivel medio los sistemas de información e instalaciones de tratamiento y almacenamiento de datos se someterán, al menos cada dos años, a una auditoría interna o externa que verifique el cumplimiento del presente título."

"*El informe de auditoría deberá dictaminar sobre la adecuación de las medidas y controles a la Ley y su desarrollo reglamentario, identificar sus deficiencias y proponer las medidas correctoras o complementarias necesarias. Deberá, igualmente, incluir los datos, hechos y observaciones en que se basen los dictámenes alcanzados y las recomendaciones propuestas.*"

Por ello, debido a que el sistema realiza un tratamiento de datos de carácter personal de nivel alto será necesario realizar una auditoría ordinaria, como mínimo, cada dos años.

Al existir datos de nivel alto en el sistema, el artículo 104 nos exige que la transmisión de datos de carácter personal a través de redes de comunicaciones públicas o inalámbricas debe realizarse utilizando mecanismos de cifrado que garanticen que la información no pueda ser inteligible o manipulada por terceros.

Por ello, se ha establecido que todas las comunicaciones entre el servidor y los dispositivos móviles se lleven a cabo mediante el protocolo HTTPS, el cual obliga a que los datos viajen cifrados por la red.

Además, de acuerdo con los artículos 105-114, al realizarse un tratamiento de datos de carácter personal de nivel alto, se deberán cumplir las siguientes obligaciones:

- Se designará, al menos, un responsable de seguridad encargado de coordinar y controlar las medidas de seguridad establecidas, las cuales serán recogidas en un documento de seguridad
- Los ficheros deben ser almacenados en elementos situados en áreas de acceso protegido con puertas dotadas de sistemas de apertura mediante llave u otro dispositivo equivalente
- La generación de copias de los ficheros será llevada a cabo bajo control de personal autorizado en el documento de seguridad
- La destrucción de las copias o reproducciones desechadas debe llevarse a cabo de forma que se evite el acceso a la información de las mismas y que se impida su recuperación posterior
- El acceso a los ficheros se limitará exclusivamente al personal autorizado y se establecerán mecanismos que permitan identificar los accesos a dichos ficheros
- El acceso de personas no autorizadas deberá quedar adecuadamente registrado
- Durante el traslado físico de la información contenida en un fichero deberán adoptarse medidas dirigidas a impedir el acceso o manipulación de dicha información

5. Capítulo 5: Metodología de trabajo y planificación del proyecto

5.1. Metodología de trabajo

La metodología de trabajo utilizada para el desarrollo de este proyecto se basa en el modelo de ciclo de vida del software basado en prototipos, representado en la *Ilustración 1* [10].

Un prototipo puede ser desde un boceto hasta un programa que funciona, el cual puede ser utilizado para validar las funcionalidades implementadas con los clientes y/o usuarios de la aplicación a medida que ésta va siendo desarrollada.

Los prototipos son muy útiles en la identificación de nuevos requisitos y para eliminar aquellos que no son realmente necesarios, es decir, aquellos que no cubren una necesidad de los usuarios del software desarrollado.

Esta metodología pertenece a los modelos de desarrollo evolutivo, es decir, permite desarrollar diferentes versiones de la aplicación de forma iterativa hasta llegar al objetivo final deseado, de tal forma que cada versión es más completa y compleja que la anterior.

Ilustración 1: Metodología de desarrollo basada en prototipos

Básicamente, consiste en construir prototipos de forma iterativa, donde en cada iteración se muestra el prototipo al cliente para validar las funcionalidades implementadas y así obtener *feedback* para la siguiente iteración. De este *feedback* se busca obtener:

- Nuevas funcionalidades para la aplicación
- Funcionalidades que necesitan ser modificadas o mejoradas
- Funcionalidades que deben ser eliminadas al no cubrir una necesidad de los usuarios

Se ha escogido esta metodología porque permite que la aplicación se vaya desarrollando de forma incremental a medida que se va validando con el tutor del proyecto, lo cual facilita el control de los distintos prototipos desarrollados para asegurar la calidad de los mismos.

5.2. Planificación inicial del proyecto

A continuación, se presenta el plan de trabajo definido al comienzo del proyecto desglosado en distintas tareas:

Fases	Duración estimada (horas)	Tareas
Estudio previo / Análisis	60	Tarea 1.1: Estudio del problema y soluciones actuales
		Tarea 1.2: Estudio del trabajo previo realizado
		Tarea 1.3: Obtención de nuevos requisitos del sistema
Diseño / Desarrollo / Implementación	140	Tarea 2.1: Diseño de la arquitectura del sistema
		Tarea 2.2: Diseño de la base de datos
		Tarea 2.3: Aprendizaje de Ionic 4, Angular 6, .NET Core, Entity Framework y LINQ
		Tarea 2.4: Implementación de prototipos de la aplicación móvil
Evaluación / Validación / Prueba	60	Tarea 3.1: Pruebas de funcionamiento
		Tarea 3.2: Reuniones con fisioterapeutas para validar los contenidos de la aplicación móvil
Documentación / Presentación	40	Tarea 4.1: Realización de la memoria
		Tarea 4.2: Realización de la presentación

Tabla 2: Plan de trabajo inicial

Durante la primera fase, "Estudio previo / Análisis", se realizó un estudio del prototipo de aplicación móvil que ya había sido desarrollado y se detectaron los siguientes problemas:

- No se contempla la figura del neonato, por lo que si está a cargo de varios tutores éstos no tienen posibilidad de compartir la información
- Al no contemplarse la figura del neonato, el fisioterapeuta no podría acceder a un historial centralizado del bebé si se diera el caso de que la tutela de éste la tuviera el Estado y sus tutores no fuesen los mismos a lo largo del tiempo
- No se permite que los fisioterapeutas modifiquen la base de datos de ejercicios desde la aplicación
- No se permite que los tutores envíen a los fisioterapeutas información (observaciones, imágenes, vídeos) organizada acerca de las distintas sesiones de ejercicios que tienen con los bebés, por lo que el seguimiento de la evolución del bebé no se simplifica
- No existe la posibilidad de clasificar los ejercicios en distintas categorías
- Al haber implementado toda la API de acceso a datos utilizando Firebase, la cual ofrece un tratamiento limitado de la información, gran parte de la lógica de negocio se lleva a cabo en la aplicación, lo que conlleva un mayor tráfico de datos en la red y un mayor uso de los recursos del dispositivo móvil
- Cualquier persona puede darse de alta en la aplicación como fisioterapeuta

En esta fase, también se organizó una reunión con Dña. María del Mar Batista Guerra para analizar el estado actual del servicio de atención a neonatos. En dicha reunión, se describió dicho servicio y se comentaron los distintos problemas que detectados desde el punto de vista del fisioterapeuta y del tutor del paciente.

Los problemas detectados fueron:

- El material impreso de apoyo a los ejercicios entregado a los tutores no les resuelve todas las dudas
- La impresión del material de apoyo genera un gasto de recursos a los centros de salud que puede evitarse mediante el uso de las nuevas tecnologías
- Los tutores recurren a la plataforma de mensajería WhatsApp para comentar las dudas a los médicos y para mostrarles el avance de los bebés, lo que hace que el seguimiento por parte del fisioterapeuta sea complejo al no disponer de una forma de organizar la información recibida

También, se llegó a la conclusión de que a los fisioterapeutas les beneficiaría más una aplicación móvil que una aplicación web o de escritorio. Esto se debe a que, debido a problemas de conexión en sus puestos de trabajo, los fisioterapeutas suelen recurrir a su *smartphone* durante las consultas.

El desarrollo de una aplicación web que permita trabajar cómodamente desde su equipo de escritorio a los fisioterapeutas que así lo deseen, quedó descartado en esta reunión debido a la imposibilidad de llevar a cabo el desarrollo de la aplicación móvil y la aplicación web en el tiempo asignado a este Trabajo de Fin de Título.

Una vez identificados los problemas y propuesta la solución de desarrollar una aplicación móvil, se estudiaron las funcionalidades que ésta debía tener.

Tras definir las funcionalidades del sistema y los distintos actores que intervendrían en él, se realizó un primer diseño de la arquitectura del sistema y la base de datos a la vez que se estudiaron las distintas tecnologías existentes para escoger las más adecuadas para el desarrollo del proyecto.

Después de escoger las tecnologías a utilizar, se comenzó con la fase de su aprendizaje mediante la realización de tutoriales que proponían el desarrollo de pequeñas aplicaciones.

Finalizada la fase de aprendizaje de las tecnologías, se comenzó a desarrollar la aplicación móvil. De acuerdo con la metodología basada en prototipos explicada anteriormente, su desarrollo se dividió en diferentes iteraciones:

- Iteración 1: configuración básica de los proyectos (aplicación móvil y API REST) e implementación de un sistema de inicio de sesión basado en roles para acceder a la aplicación móvil de manera segura.

- Iteración 2: implementación de las funcionalidades relacionadas con la gestión de las distintas entidades existentes en el sistema (fisioterapeutas, tutores y pacientes). Esto incluye su creación y edición, así como las distintas relaciones entre ellas.

- Iteración 3: implementación de la gestión de ejercicios por parte de los fisioterapeutas. Esto incluye su creación, edición y eliminación.

- Iteración 4: implementación de la asignación de ejercicios a los tutores por parte de los fisioterapeutas y la creación de sesiones para llevar a cabo el seguimiento de los neonatos por parte de los tutores.

- Iteración 5: implementación de las funcionalidades asociadas al administrador del sistema para la gestión de fisioterapeutas y categorías. Esto incluye el alta de fisioterapeutas en el sistema, así como la gestión de las categorías en las que se clasifican los ejercicios.

- Iteración 6: implementación del sistema de chat en tiempo real entre tutores y fisioterapeutas.

5.3. Ajustes en la planificación del proyecto

En general, el transcurso del proyecto fue de acuerdo a lo establecido a su comienzo. Sin embargo, el desarrollo de la aplicación móvil fue más complejo de lo esperado debido a que la versión 4 del *framework* utilizado, Ionic, fue lanzada en fase beta a finales del mes de julio de 2018. Durante el tiempo en el que se ha desarrollado la aplicación móvil, el equipo de Ionic ha realizado distintas actualizaciones del *framework*.

Estas actualizaciones han provocado algunos contratiempos durante el desarrollo de la aplicación, debido a que cada versión de la beta modificaba el comportamiento de algunos componentes.

Además, la documentación oficial de Ionic 4 también se encuentra en fase beta, lo cual ha dificultado también el desarrollo de la aplicación.

Debido a estas dificultades, las horas estimadas para la fase de desarrollo de la aplicación sufrieron un incremento considerable, pasando de las 140 horas estimadas en el inicio del proyecto a las 170 horas dedicadas realmente.

Por ello, se ha tenido que prescindir de un módulo que fue planteado al comienzo del proyecto que incluía la implementación de notificaciones *push* que permitirían a tutores y fisioterapeutas ser notificados cuando:

- Un fisioterapeuta asigna un ejercicio a un tutor
- Un tutor crea una sesión
- Un fisioterapeuta aporta un comentario a una sesión
- Se recibe un mensaje en el chat de la aplicación

6. Capítulo 6: Tecnologías y herramientas utilizadas

6.1. Herramientas

- Microsoft Visual Studio Community 2017: es un entorno de desarrollo integrado (IDE) multiplataforma desarrollado por Microsoft que soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc.

- Postman: es un entorno utilizado para el testeo y la monitorización de APIs REST. Permite, entre otras cosas, enviar de peticiones HTTP y crear grupos de peticiones para generar tests complejos.

- Microsoft Visual Studio Code: es un editor de código fuente multiplataforma desarrollado por Microsoft que incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

- MySQL Workbench: es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

- StarUML: es una herramienta para la creación de diagramas UML.

- Google Chrome: es un navegador web desarrollado por Google que cuenta con herramientas de desarrollo para desarrollar y depurar el *frontend* de aplicaciones web.

- GitHub: es una plataforma de desarrollo colaborativo de software utilizada para alojar proyectos que utilizan el sistema de control de versiones Git.

- Android Studio: es el entorno de desarrollo integrado (IDE) oficial para la plataforma Android. Permite renderizado en tiempo real de aplicaciones, consola de desarrollador, soporte para construcción basada en Gradle, refactorización de código, editor de diseño *drag & drop* para interfaces de usuario y probar aplicaciones utilizando un emulador, entre otras cosas.

6.2. Tecnologías

- MySQL: es un sistema de gestión de bases de datos relacional que está considerada como la base datos de código abierto más popular del mundo, sobre todo para entornos de desarrollo web.

- Ionic 4: es un kit de desarrollo de software (SDK) de código abierto para el desarrollo de aplicaciones móviles híbridas. Ofrece herramientas y servicios para desarrollar aplicaciones

mediante el uso de tecnologías web (CSS, HTML5, Sass). Utiliza Cordova para la creación de aplicaciones nativas.

- Ionic CLI: es una interfaz de línea de comandos para el desarrollo de aplicaciones basadas en Ionic. Ofrece una serie de facilidades como la creación de proyectos y módulos dentro de un proyecto, un servidor de desarrollo, etc.

- Apache Cordova: es un popular entorno de desarrollo de aplicaciones móviles que permite construir aplicaciones para dispositivos móviles utilizando CSS3, HTML5, y JavaScript en vez de utilizar APIs específicas de cada plataforma como Android o iOS.

- Angular 6: es un *framework* para aplicaciones web desarrollado en TypeScript desarrollado por Google que se utiliza para crear aplicaciones web de una sola página (SPA).

- Angular CLI: es una interfaz de línea de comandos para el desarrollo de aplicaciones basadas en Angular. Ofrece una serie de facilidades para crear, depurar y publicar aplicaciones Angular de manera fácil y sencilla.

- TypeScript: es un lenguaje de programación desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases. Está pensado para grandes proyectos, los cuales se traducen a código JavaScript original a través de un compilador de TypeScript.

- HTML: es un lenguaje de marcado para la elaboración de páginas web. Se considera el lenguaje web más importante. Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

- CSS: es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy utilizado para establecer el diseño visual de las páginas web e interfaces de usuario escritas en HTML.

- Sass: es un metalenguaje de hoja de estilos en cascada (CSS) que ofrece diferentes mecanismos avanzados sobre CSS, como el uso de variables, anidamiento, y herencia de selectores.

- .NET Core 2.1: es un *framework* multiplataforma que permite la ejecución de programas desarrollados bajo la plataforma .NET de Microsoft. Se utiliza para la creación de aplicaciones web, de escritorio y móviles.

- Entity Framework Core: es un *framework* ORM (mapeador de objetos-relacional) utilizado como API de acceso a datos para el *framework* .NET Core. Permite el tratamiento de los datos guardados en un sistema de bases de datos relacional mediante programación orientada a objetos.

- LinQ: es un componente de la plataforma Microsoft .NET que agrega capacidades de consulta a datos de manera nativa a los lenguajes .NET.

- C#: es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.

- Firebase: es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles que permite, entre otras cosas, el acceso a información en tiempo real.

- Realtime Database: es una base de datos en tiempo real que proporciona a los desarrolladores de aplicaciones una API que permite que la información de las aplicaciones sea sincronizada y almacenada en la nube de Firebase.

- Git: es un software de control de versiones basado en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Gestiona eficientemente el desarrollo distribuido, ofreciendo a cada programador una copia local del historial de desarrollo.

7. Capítulo 7: Análisis

7.1. Actores

Los diferentes actores que harán uso del sistema son el administrador, los fisioterapeutas y los tutores de los neonatos. A continuación, se explicarán las funcionalidades y características de cada uno de ellos.

7.1.1. Administrador

El administrador es un miembro del servicio de fisioterapia encargado de:

- Dar de alta a los fisioterapeutas en el sistema. Esto es así para tener un control de quién tiene acceso a los datos de los bebés, porque que no es correcto que cualquier persona pueda darse de alta como fisioterapeuta en la aplicación y acceder a información que debe estar protegida.
- Gestionar las categorías utilizadas para organizar los ejercicios subidos por los fisioterapeutas. Esta gestión incluye su creación, edición y eliminación.

Una vez se haya establecido quien, o quienes, tendrán el papel de administrador, se les dará de alta en el sistema creando un usuario en la propia base de datos.

7.1.2. Fisioterapeuta

Los fisioterapeutas son los encargados de llevar las terapias y el seguimiento de los neonatos. Podrán realizar las siguientes funciones:

- Dar de alta a los pacientes en el sistema, así como editarlos.
- Asociar y desasociar tutores a sus pacientes.
- Gestionar una base de datos de ejercicios con contenido multimedia.
- Asignar ejercicios a sus pacientes, los cuales podrán ser vistos por sus tutores.
- Modificar el estado actual de las tareas asignadas a los pacientes.
- Ver las opiniones aportadas por los tutores acerca de los ejercicios.
- Aportar comentarios a las sesiones realizadas por los tutores con sus bebés.
- Responder las cuestiones formuladas por los tutores mediante el chat de la aplicación.

Se ha establecido que sean los propios fisioterapeutas quienes den de alta a los pacientes en el sistema y que, además, sean ellos quienes se encarguen de asociarlos a sus tutores. Esto se debe a que:

- Es más aconsejable que estas tareas queden en manos de una única persona ya que, de ser llevadas a cabo por los tutores, si un bebé está a cargo de varios tutores que quieren utilizar la aplicación, llevarían a confusión al no saber quién debe dar de alta al bebé y quién debe asociarlo al otro tutor

- Hay casos en los que la tutela del bebé la tiene el Estado, por lo que podría tener distintos tutores a lo largo del tiempo

7.1.3. Tutor

Son los tutores de los neonatos. Podrán realizar las siguientes funciones:

- Ver los ejercicios asignados por los fisioterapeutas.
- Escribir opiniones acerca de los ejercicios asignados.
- Gestionar las distintas sesiones que lleven a cabo con los bebés. Esto incluye su creación, edición y eliminación.
- Ver los comentarios aportados por el fisioterapeuta en una sesión.
- Realizar consultas a los fisioterapeutas mediante el chat interno de la aplicación.

Los tutores se darán de alta en la aplicación por su cuenta.

7.2. Requisitos

Para la representación de los requisitos funcionales del sistema nos hemos apoyado en UML [11] (Lenguaje Unificado de Modelado). Se ha escogido este lenguaje debido a que es el lenguaje de modelado de sistemas de software más utilizado en la actualidad.

Una de las características más interesantes de UML es que es un estándar aprobado por la ISO (Organización Internacional de Normalización), lo que hace que sea cualquier diagrama creado con UML pueda ser interpretado por cualquier persona que conozca dicho estándar.

7.2.1. Casos de uso

A continuación, se muestra el diagrama de casos de uso realizado para tener un esquema claro de los diferentes requisitos funcionales identificados e implementados. Dicho diagrama ha sido dividido en distintas imágenes para facilitar su lectura.

Los actores “Administrador”, “Fisioterapeuta” y “Tutor” heredan del actor denominado “Usuario registrado”, por lo que todos los casos de uso asociados a este actor también se aplican a los demás.

Los actores “Fisioterapeuta” y “Tutor” heredan del actor denominado “Conversador”, por lo que todos los casos de uso asociados al “Conversador” también se aplican a ambos.

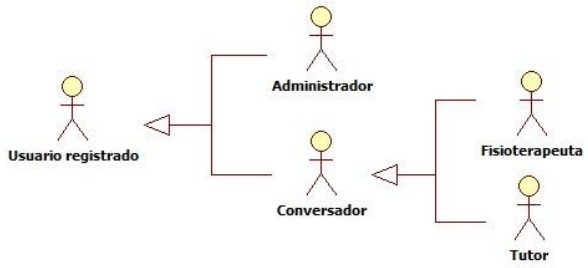

Ilustración 2: Diagrama de casos de uso (actores)

Ilustración 3: Diagrama de casos de uso (usuario registrado)

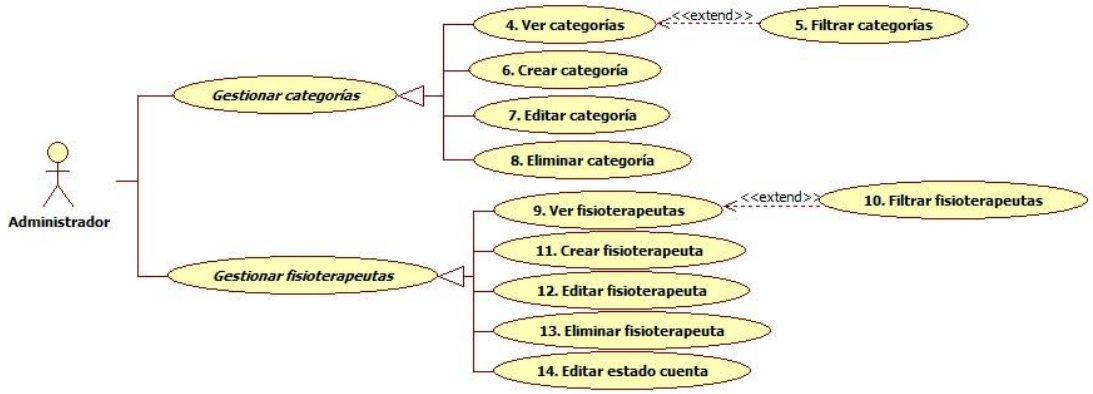

Ilustración 4: Diagrama de casos de uso (administrador)

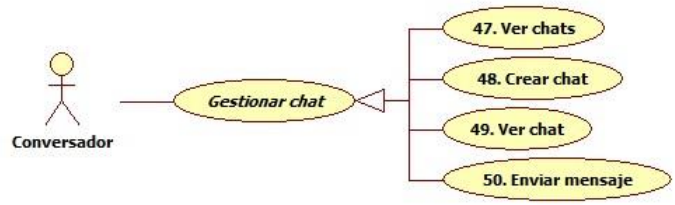

Ilustración 5: Diagrama de casos de uso (conversador)

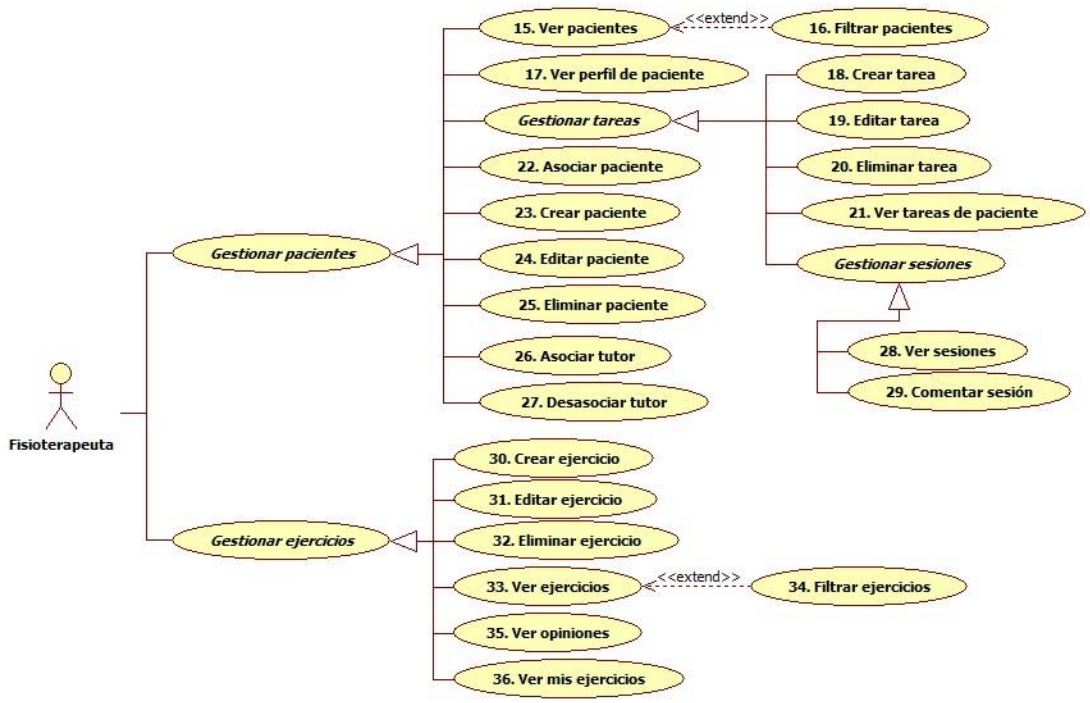

Ilustración 6: Diagrama de casos de uso (fisioterapeuta)

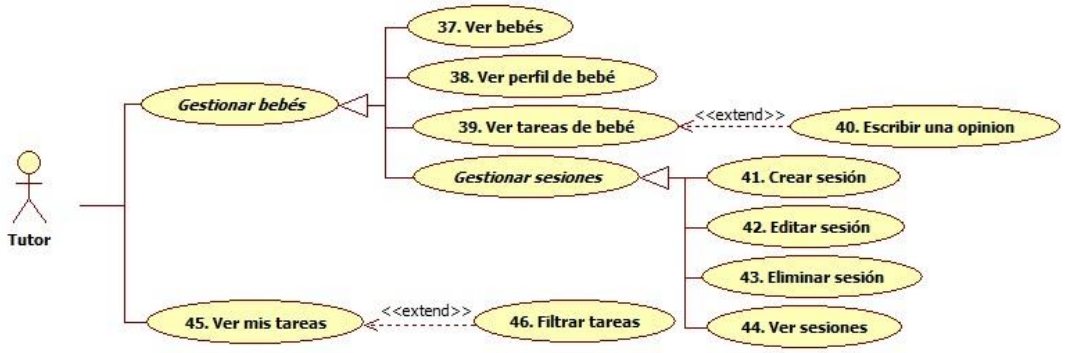

Ilustración 7: Diagrama de casos de uso (tutor)

En la siguiente tabla se muestra una breve descripción de los casos de uso representados anteriormente:

ID	Actor principal	Descripción
1	Usuario registrado	Se muestra el perfil del usuario
2	Usuario registrado	Se edita el perfil del usuario
3	Usuario registrado	Se edita la cuenta asociada al usuario
4	Administrador	Se muestran las categorías existentes
5	Administrador	Se muestran las categorías cuyos nombres coinciden con el filtro establecido
6	Administrador	Se crea una nueva categoría en el sistema
7	Administrador	Se edita una categoría
8	Administrador	Se elimina una categoría del sistema
9	Administrador	Se muestran los fisioterapeutas que se han dado de alta en el sistema

10	Administrador	Se muestran los fisioterapeutas cuyos nombres coinciden con el filtro establecido
11	Administrador	Se da de alta un fisioterapeuta en el sistema y se le envía un correo indicándole una contraseña aleatoria autogenerada
12	Administrador	Se edita el perfil de un fisioterapeuta
13	Administrador	Se elimina un fisioterapeuta del sistema
14	Administrador	Se bloquea o desbloquea la cuenta de un fisioterapeuta, de tal forma que pueda, o no, iniciar sesión en la aplicación
15	Fisioterapeuta	Se muestran los pacientes dados de alta en el sistema asociados al fisioterapeuta
16	Fisioterapeuta	Se muestran los pacientes cuyos nombres coinciden con el filtro establecido
17	Fisioterapeuta	Se muestra el perfil del paciente
18	Fisioterapeuta	Se crea una nueva tarea en el sistema asociada a un paciente
19	Fisioterapeuta	Se edita una tarea
20	Fisioterapeuta	Se elimina una tarea del sistema
21	Fisioterapeuta	Se muestran las tareas asociadas a un paciente
22	Fisioterapeuta	Se asocia un paciente al fisioterapeuta
23	Fisioterapeuta	Se da de alta un paciente en el sistema
24	Fisioterapeuta	Se edita un paciente
25	Fisioterapeuta	Se elimina un paciente del sistema
26	Fisioterapeuta	Se asocia un tutor al paciente
27	Fisioterapeuta	Se desasocia un tutor del paciente
28	Fisioterapeuta	Se muestran las sesiones de un paciente asociadas a una tarea
29	Fisioterapeuta	Se escribe un comentario en una sesión
30	Fisioterapeuta	Se crea un ejercicio en el sistema
31	Fisioterapeuta	Se edita un ejercicio
32	Fisioterapeuta	Se elimina un ejercicio del sistema
33	Fisioterapeuta	Se muestran todos los ejercicios dados de alta en el sistema
34	Fisioterapeuta	Se muestran los ejercicios que coinciden con los filtros especificados (nombre y/o categoría)
35	Fisioterapeuta	Se muestran las opiniones asociadas a un ejercicio del fisioterapeuta
36	Fisioterapeuta	Se muestran los ejercicios del fisioterapeuta
37	Tutor	Se muestran los bebés asociados al tutor
38	Tutor	Se muestra el perfil de un bebé asociado al tutor
39	Tutor	Se muestran las tareas asignadas a un bebé asociado a un tutor
40	Tutor	Se escribe una opinión asociada a un ejercicio
41	Tutor	Se crea una sesión en el sistema asociada a un ejercicio
42	Tutor	Se edita una sesión
43	Tutor	Se elimina una sesión del sistema
44	Tutor	Se muestran las sesiones asociadas a una tarea de un bebé
45	Tutor	Se muestran las tareas asociadas a un tutor a través de sus bebés
46	Tutor	Se muestran las tareas cuyo estado coincida con el filtro seleccionado
47	Conversador	Se muestran los chats en los que participa el usuario
48	Conversador	Se crea un nuevo chat con un fisioterapeuta o un tutor
49	Conversador	Se muestra el historial de mensajes de un chat
50	Conversador	Se envía un mensaje a un fisioterapeuta o un tutor a través del chat

Tabla 3: Resumen de casos de uso

7.2.2. Especificación de casos de uso

En este apartado se realizará la especificación de los casos de la aplicación que contengan algún aspecto que sea necesario aclarar:

CASO DE USO	3	**EDITAR CUENTA**	
Descripción	El usuario desea modificar sus credenciales de acceso a la aplicación móvil.		
Actores	Usuario registrado		
Precondiciones	El usuario debe haber iniciado sesión en el sistema y accedido a la sección “Mi perfil”.		
Flujo normal	**Paso**	**Acción**	
	1	Se establece el email	
	2	Se establece la contraseña actual, por motivos de seguridad	
	3	Se establece la nueva contraseña (opcional)	
	4	Se confirman los cambios realizados	
Postcondiciones	Se requerirán las nuevas credenciales en los siguientes inicios de sesión.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	4	El email está vacío	
	4	El formato del email es incorrecto	
	4	La contraseña actual no es correcta	
	4	El email ya está siendo utilizado por otro usuario	
Observaciones			

Tabla 4: Especificación de casos de uso (editar cuenta)

CASO DE USO	11	**CREAR FISIOTERAPEUTA**
Descripción	El administrador desea dar de alta un nuevo fisioterapeuta en el sistema.	
Actores	Administrador	

Precondiciones	El administrador debe haber iniciado sesión en el sistema y accedido a la vista de “Fisioterapeutas”.		
Flujo normal	**Paso**	**Acción**	
	1	Se establece el nombre del fisioterapeuta	
	2	Se establece el número de colegiado del fisioterapeuta	
	3	Se establece el número de teléfono del fisioterapeuta	
	4	Es establece el email del fisioterapeuta	
	5	Se crea el fisioterapeuta en el sistema	
Postcondiciones	El fisioterapeuta se ha dado de alta en el sistema y se le envía un correo al email establecido con una contraseña aleatoria autogenerada.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	5	El email está vacío	
	5	El formato del email es incorrecto	
	5	El número de colegiado está vacío	
	5	El número de teléfono está vacío	
	5	El email ya está siendo utilizado por otro usuario	
Observaciones			

Tabla 5: Especificación de casos de uso (crear fisioterapeuta)

CASO DE USO	18	**CREAR TAREA**	
Descripción	El fisioterapeuta desea asignar una tarea a un paciente para que la realice con sus tutores.		
Actores	Fisioterapeuta		
Precondiciones	El fisioterapeuta debe haber iniciado sesión en el sistema y accedido a la vista de “Tareas” del paciente.		
Flujo normal	**Paso**	**Acción**	
	1	Se selecciona el ejercicio asociado a la tarea	

	2	Se establece la frecuencia con la que se debe realizar el ejercicio	
	3	Se establece la duración del ejercicio	
	4	Se establece la frecuencia mínima recomendada con la que los tutores deben crear una sesión asociada al ejercicio (frecuencia de *feedback*)	
	5	Se establece información personalizada acerca de la tarea	
	6	Se crea la tarea en el sistema	
Postcondiciones	Se crea la tarea en el sistema y los tutores del paciente podrán acceder a ella.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	6	No se ha escogido ningún ejercicio	
	6	La frecuencia del ejercicio está vacía	
	6	La duración del ejercicio está vacía	
	6	La frecuencia de *feedback* está vacía	
	6	La información personalizada está vacía	
Observaciones			

Tabla 6: Especificación de casos de uso (crear tarea)

CASO DE USO	22	**ASOCIAR PACIENTE**
Descripción	El fisioterapeuta desea asociar a su usuario un paciente que ya ha sido de alta en el sistema para poder hacerle el seguimiento.	
Actores	Fisioterapeuta	
Precondiciones	El fisioterapeuta debe haber iniciado sesión en el sistema y haber accedido a la vista de “Mis pacientes”.	
Flujo normal	**Paso**	**Acción**
	1	Se establece el ID del historial médico del paciente
	2	Se muestra la información perteneciente al paciente para comprobar que el ID del historial médico se ha introducido correctamente

	3	Se solicita confirmación	
	4	Se asocia el paciente al fisioterapeuta	
Postcondiciones	Se asocia el paciente al fisioterapeuta y éste podrá realizar el seguimiento.		
Variaciones	**Paso**	**Acción**	
	3 A	El fisioterapeuta cancela la asociación con el paciente	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	4	El paciente ya estaba asociado al fisioterapeuta	
Observaciones			

Tabla 7: Especificación de casos de uso (asociar paciente)

CASO DE USO	23	**CREAR PACIENTE**	
Descripción	El fisioterapeuta desea dar de alta un paciente en el sistema para poder realizar el seguimiento.		
Actores	Fisioterapeuta		
Precondiciones	El fisioterapeuta debe haber iniciado sesión en el sistema y accedido a la vista de “Mis pacientes”.		
Flujo normal	**Paso**	**Acción**	
	1	Se establece el ID del historial médico del paciente	
	2	Se establece el nombre del paciente	
	3	Se establece el sexo del paciente	
	4	Se establece la fecha de nacimiento del paciente	
	5	Se establece el número de semanas de gestación del paciente	
	6	Se establece un resumen del historial médico del paciente	
	7	Se crea el paciente en el sistema	
Postcondiciones	Se crea el paciente en el sistema y el fisioterapeuta podrá realizarle el seguimiento.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**

Excepciones	7	El ID del historial médico está vacío	
	7	El nombre está vacío	
	7	El sexo está vacío	
	7	La fecha de nacimiento está vacía	
	7	El número de semanas de gestación está vacío	
	7	El resumen del historial médico está vacío	
Observaciones			

Tabla 8: Especificación de casos de uso (crear paciente)

CASO DE USO	26	**ASOCIAR TUTOR**	
Descripción	El fisioterapeuta desea asociar un tutor a un paciente para que pueda acceder a las tareas asignadas.		
Actores	Fisioterapeuta		
Precondiciones	El fisioterapeuta debe haber iniciado sesión en el sistema y accedido a la vista de "Tutores" del paciente.		
Flujo normal	**Paso**	**Acción**	
	1	Se introduce el NIF del tutor	
	2	Se solicita confirmación	
	3	Se asocia el tutor al paciente	
Postcondiciones	Se asocia el tutor al paciente y podrá acceder a las tareas asignadas al bebé.		
Variaciones	**Paso**	**Acción**	
	2 A	El fisioterapeuta cancela la asociación del paciente con el tutor	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	3	El tutor ya estaba asociado al paciente	
Observaciones			

Tabla 9: Especificación de casos de uso (asociar tutor)

8. Capítulo 8: Diseño

8.1. Diseño de la arquitectura del sistema

Se ha optado por una arquitectura multi-nivel, representada en la *Ilustración 2* [12], puesto que ofrece las siguientes características:

- Es fácilmente escalable: los clientes solo contienen la aplicación móvil, desarrollada utilizando HTML, CSS y JavaScript, la cual puede ser ejecutada desde cualquier dispositivo móvil con sistema operativo Android o iOS.
- Ahorra recursos: solo debemos de preocuparnos de los recursos asignados al *backend*.
- Es flexible: permite organizar mejor las distintas capas de software, lo que hace que sea más sencillo modificar una de ellas.
- Es segura: no es necesario aplicar políticas de seguridad complejas en las aplicaciones móviles, ya que la lógica de negocio y los datos están alojados en un único servidor.

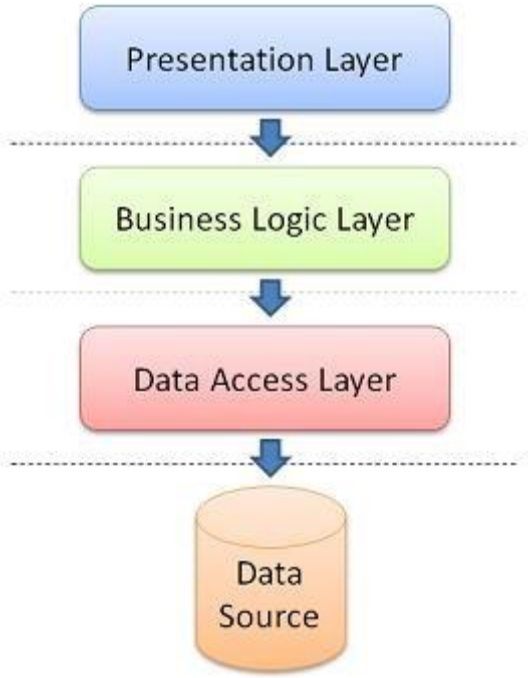

Ilustración 8: Arquitectura multi-nivel

Las aplicaciones cliente contienen la capa de presentación y una capa de red para poder comunicarse con el servidor que contiene la API REST. El *backend* que utiliza la aplicación móvil para comunicarse con la base de datos se encuentra alojado en un servidor web (capas de negocio y de acceso a datos y datos). En la misma máquina se aloja la base de datos.

En el siguiente diagrama se muestra la arquitectura utilizada:

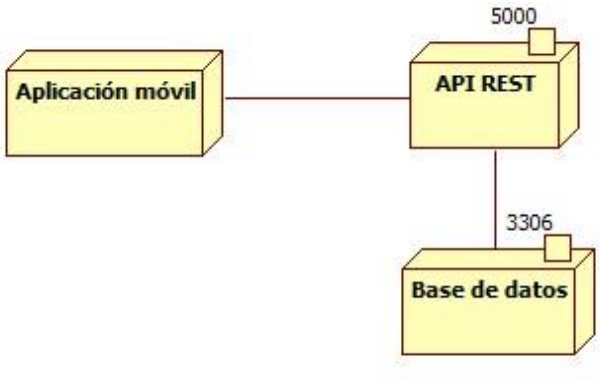

Ilustración 9: Diagrama de despliegue

A continuación, se describirán brevemente los diferentes elementos mostrados en el diagrama anterior:

- Aplicación móvil: aplicación móvil multiplataforma utilizada por administradores, fisioterapeutas y tutores
- Servidor web: servidor donde se aloja el *backend* que comunica la aplicación móvil con la base de datos
- Base de datos: base de datos MySQL donde se almacenan los datos del sistema

8.2. Diseño de la base de datos

8.2.1. Base de datos MySQL utilizada por la *API REST*

Como se ha nombrado anteriormente, se ha utilizado el sistema de gestión de bases de datos relacional MySQL. A continuación, se muestra un diagrama en el que se podrá ver el esquema de la base de datos implementada:

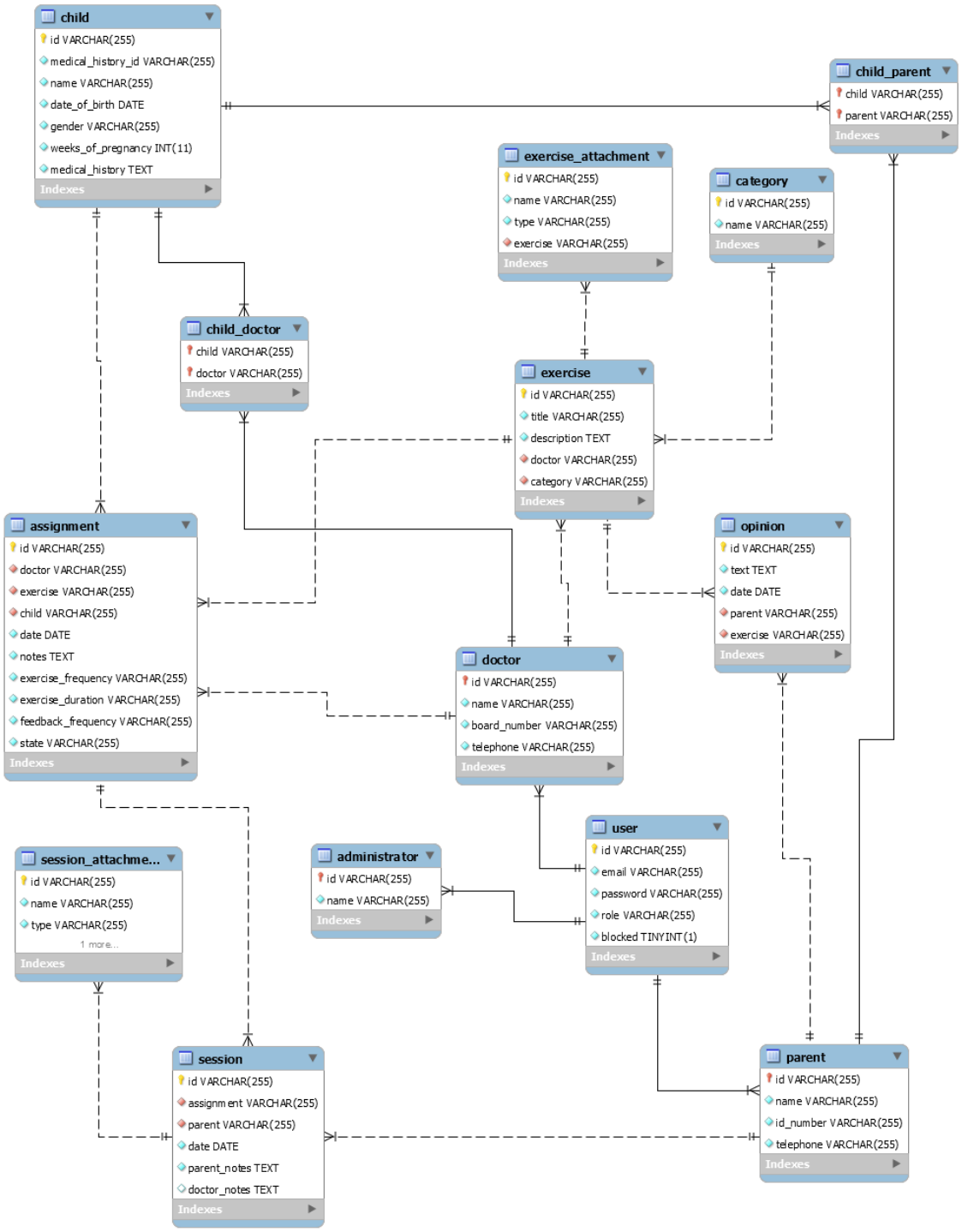

Ilustración 10: Diseño de la base de datos

Cabe destacar que las tablas *administrator*, *doctor* y *parent* heredan de la tabla *user*, la cual contiene todos los campos comunes a los distintos tipos de usuario.

En la siguiente tabla se muestra una breve descripción de las distintas tablas de la base de datos que se corresponden con las entidades del modelo de datos:

Tabla	Descripción
user	Contiene la información referente a las cuentas de usuario de la aplicación.
administrator	Contiene la información de los administradores del sistema. Hereda de la tabla *user*.
parent	Contiene la información de los tutores de los neonatos. Hereda de la tabla *user*.
doctor	Contiene la información de los fisioterapeutas. Hereda de la tabla *user*.
child	Contiene la información de los bebés.
category	Contiene las categorías utilizadas para clasificar los ejercicios.
exercise	Contiene los ejercicios dados de alta por los fisioterapeutas.
exercise_attachment	Contiene las referencias a los archivos asociados a un ejercicio. Estos archivos se almacenan en el sistema de ficheros del servidor.
opinion	Contiene las opiniones de los ejercicios escritas por los tutores.
assignment	Contiene las distintas tareas asignadas a los pacientes.
session	Contiene las sesiones que han tenido los tutores con sus bebés.
session_attachment	Contiene las referencias a los archivos asociados a una sesión. Estos archivos se almacenan en el sistema de ficheros del servidor.

Tabla 10: Descripción de las tablas de entidad de la base de datos

A continuación, se muestra una tabla con una breve descripción de las distintas tablas de la base de datos que se corresponden con tablas que relacionan entidades:

Tabla	Descripción
child_parent	Relaciona tutores con bebés. Esto permite que un tutor pueda estar asociado con múltiples bebés y que un bebé pueda estar asociado con múltiples tutores.
child_doctor	Relaciona pacientes con fisioterapeutas. Esto permite que un fisioterapeuta pueda estar asociado con múltiples pacientes y que un paciente pueda estar asociado con múltiples fisioterapeutas.

Tabla 11: Descripción de las tablas de relaciones de la base de datos

8.2.2. Realtime Database

Realtime Database es la base de datos que se ha utilizado para la implementación del chat en tiempo real. Se trata de una base de datos NoSQL alojada en la nube, donde los datos se almacenan en formato JSON y se sincronizan con todos los clientes en tiempo real.

Cuando utilizamos Realtime Database no es necesario definir ninguna estructura a priori, como se hace en las bases de datos relacionales. El JSON se modifica de forma dinámica y cada objeto puede tener campos variables.

En cuanto al diseño del modelo de datos implementado, se han definido tres colecciones de datos diferentes:

- users: usuarios que utilizan el chat (fisioterapeutas y tutores)
- chats: conversaciones entre un tutor y un fisioterapeuta
- messages: mensajes enviados por un tutor a un fisioterapeuta y viceversa

Los usuarios de Realtime Database se corresponden con los de la base de datos MySQL. Por ello, cada vez que se crea, edita o elimina un usuario en la base de datos MySQL la misma acción se lleva a cabo en la base de datos Realtime Database.

A continuación, se muestra un ejemplo de un JSON que contiene un chat entre un tutor y un fisioterapeuta en el que se han enviado dos mensajes:


```
{
  "chats" : {
    "-LTr6JC_RdV3HSzFSQp7" : {
      "doctor" : "22222222-2222-2222-2222-222222222222",
      "parent" : "11111111-1111-1111-1111-111111111111"
    },
      "-LTr6JC_RdV3HSzFSQp8" : {
      "doctor" : "22222222-2222-2222-2222-222222222222",
      "parent" : "33333333-3333-3333-3333-333333333333"
    }
  },
  "messages" : {
    "-LTr6JC_RdV3HSzFSQu9" : {
      "chat" : "-LTr6JC_RdV3HSzFSQp7",
      "sender" : "11111111-1111-1111-1111-111111111111",
      "text" : "parent1 says hi to doctor1",
      "timestamp" : 1544979211346
    },
      "-LTr6JC_RdV3HSzFSQu4" : {
      "chat" : "-LTr6JC_RdV3HSzFSQp7",
      "sender" : "22222222-2222-2222-2222-222222222222",
      "text" : "doctor1 says hi to parent1",
      "timestamp" : 1544979476346
    }
  },
  "users" : {
    "11111111-1111-1111-1111-111111111111" : {
      "name" : "parent1",
      "role" : "parent"
    },
    "22222222-2222-2222-2222-222222222222" : {
      "name" : "doctor1",
      "role" : "doctor"
    }
  }
}
```

8.3. Almacenamiento de ficheros en el servidor

Se ha decidido guardar las imágenes y los vídeos de los ejercicios y las sesiones en el sistema de ficheros del servidor, en vez de en tablas de la base de datos. Esta decisión se debe a los siguientes motivos [13]:

- Guardar los ficheros en la base de datos complica el código encargado de su persistencia debido a que, en una base de datos, podemos guardar un fichero utilizando distintos tipos de datos. Esto hace que sea necesario crear una capa de software distinta para poder persistir los ficheros utilizando cada uno de estos tipos de datos.
- El nivel de conocimiento necesario para mantener una base de datos es proporcional a su tamaño, ya que dificulta las migraciones y las copias de seguridad.
- Guardar los ficheros en la base de datos impide aprovechar la potencia del almacenamiento en la nube.
- Es más sencillo y eficiente acceder a un fichero, desde una aplicación web o móvil, utilizando una ruta del sistema de ficheros en el cual se encuentra guardado que realizando un acceso a una base de datos.

Para almacenar los ficheros en el servidor en la base de datos se guarda:

- El nombre original del fichero
- El ejercicio o la sesión al que está asociado
- El tipo de fichero (imagen o vídeo)

Los ficheros se guardan en un directorio del sistema de ficheros y se utiliza el identificador del registro de la tabla de la base de datos como nombre del fichero. Para recuperar el fichero del servidor solo es necesario conocer el identificador del registro de la base de datos asociado al fichero y la ruta del sistema de ficheros en la que se encuentra almacenado.

8.4. Diseño de la interfaz gráfica

Para que una aplicación tenga éxito, es muy importante cuidar la interfaz gráfica de usuario. Por ello, es muy importante tener en cuenta los principios de diseño formulados por especialistas en la materia que hacen que una aplicación pueda ser agradable a la vista y fácil de usar, logrando que los usuarios tengan una buena experiencia con ella.

Para mejorar la interacción entre los usuarios y la aplicación móvil desarrollada se han tenido en cuenta algunos de los principios de diseño de interacción formulados por el consultor de usabilidad Bruce Tognazzini [14].

A continuación, se presentan los principios de diseño que se han tenido en cuenta.

8.4.1. Anticipación

Proporcionar toda la información y las herramientas necesarias para realizar una tarea, evitando cambios de pantalla innecesarios.

Para cumplir con este principio, nos aseguramos de que las distintas pantallas contienen toda la información necesaria para poder llevar a cabo una acción determinada, de tal forma que dicha acción pueda realizarse de la manera más rápida posible, evitando cambios de pantalla.

Por ello, las distintas acciones se realizan en una única pantalla que contiene toda la información necesaria para ser realizada fácil y cómodamente.

8.4.2. Consistencia

Los objetos similares deben comportarse de la misma manera (consistentes) y los objetos que no son similares no deben comportarse de la misma manera (inconsistentes).

Se ha tenido en cuenta que todos los objetos que realizan un tipo determinado de acción tengan la misma apariencia y que, además, se comporten de la misma manera. Así, se consigue que cuando el usuario realice una acción sobre un objeto habiendo realizado, anteriormente, una acción similar sobre un objeto similar no experimente un comportamiento inesperado que le haga dudar acerca de si su forma de utilizar dicho objeto es correcta.

Por ello, las distintas vistas de la aplicación móvil cuentan con la misma estructura y los botones que realizan un tipo de acción determinada siempre utilizan el mismo icono.

8.4.3. Legibilidad

El texto debe tener buen contraste con el fondo y un tamaño de fuente suficientemente grande, la información que sea importante debe mostrarse en un tamaño mayor.

En todas las vistas de la aplicación se cuida que el contraste entre las fuentes y el fondo sobre el que se encuentran sea adecuado, utilizando fuentes oscuras sobre fondos claros y viceversa. De esta manera, se consigue que el texto sea fácil de leer.

8.4.4. Simplicidad

Eliminar funcionalidades innecesarias y hacer que sean lo más simple posible.

Para cumplir con este principio, se ha validado con una fisioterapeuta que todas las vistas y funcionalidades de la aplicación móvil aportan valor.

Por ello, se han eliminado algunas vistas (listado de fisioterapeutas para los tutores y listado de tutores para los fisioterapeutas) tras ver que añadían complejidad al menú principal sin aportar ningún valor a la aplicación.

También, se han estudiado con detenimiento los campos de información de todos los formularios de la aplicación, de manera que cualquier información que no sea estrictamente necesaria para su uso sea descartada.

8.4.5. Navegación visible

Reducir la navegación al máximo y hacer que sea clara y natural, procurar que parezca que el usuario está siempre en el mismo sitio.

Para hacer que la navegación sea clara y natural, se han adoptado los estilos de navegación comúnmente utilizados en las plataformas móviles actuales.

Algunos ejemplos son:

- Navegar hacia una nueva vista mediante el gesto de tocar un elemento de los listados que aparecen en la aplicación
- Ubicar un botón para retroceder a la vista previa en la esquina superior izquierda de la barra de herramientas utilizando un icono fácilmente reconocible

8.5.6. Guardar el estado

Debido a que la web utiliza un protocolo sin estado, nosotros debemos guardarlo en su lugar.

Para evitar que el usuario tenga que realizar inicios de sesión cada vez que inicie la aplicación, se almacena un *token* en el dispositivo que le permitirá mantener la sesión iniciada hasta que se supere la fecha de caducidad de dicho *token* o hasta que el usuario decida finalizar la sesión manualmente.

9. Capítulo 9: Desarrollo

9.1. Backend

Para el desarrollo del *backend*, se ha utilizado el *framework* de programación .NET Core de Microsoft.

De acuerdo con una encuesta realizada por la página web Stack Overflow [15], el *framework* .NET Core ha sido el segundo *framework* de programación de *backend* más utilizado en el año 2018. Además, también se encuentra en el segundo puesto de *frameworks* de *backend* que más gustan a los desarrolladores.

Otro motivo para haber escogido este *framework* ha sido que el lenguaje de programación C# es muy similar a otros aprendidos durante los estudios de Grado, por lo que su aprendizaje no ha resultado complicado.

Se ha escogido la versión Core de .NET debido a que permite el desarrollo de aplicaciones multiplataforma, que funcionan en entornos Windows, MacOS y Linux.

9.1.1. Estructura del proyecto

Micosoft Visual Studio, el IDE de Microsoft utilizado para el desarrollo del *backend*, utiliza el concepto de "solución", que se puede definir como un conjunto de proyectos.

La solución desarrollada cuenta con distintos proyectos, de tal forma que cada uno de ellos pueda ser reutilizado. Los proyectos creados son:

- ActionFilters: contiene funciones que se ejecutan antes de un método
- Contracts: contiene las interfaces utilizadas en el proyecto
- CustomExceptionMiddleware: contiene un mecanismo de gestión de errores global para la API
- Entities: contiene las distintas entidades definidas a partir del modelo de datos creado
- LoggerService: contiene un mecanismo de *logging* para mejorar el mantenimiento de la API en un entorno de producción
- PrematureKidsAPI: contiene los distintos métodos de la API que utiliza la aplicación móvil
- Repository: contiene una capa intermedia que permite que los métodos de la API accedan a la base de datos escribiendo la menor cantidad de código posible
- SecurityService: contiene funciones para la gestión de contraseñas

9.1.2. API REST pragmática

La API implementada se basa en un sistema RESTful, cuyas características son:

- Existe un bajo acoplamiento entre cliente y servidor

- Las peticiones no mantienen estado, al igual que el protocolo HTTP
- Se define una interfaz uniforme que hace que cada recurso tenga una única dirección

Además, se han seguido una serie de buenas prácticas recomendadas para realizar APIs RESTful pragmáticas [16], por lo que la API:

- Utiliza estándares web
- Es amigable para el desarrollador y es explorable mediante una barra de direcciones del navegador
- Es sencilla, intuitiva y consistente
- Es flexible, para potenciar la interfaz de usuario con los desarrolladores

Las buenas prácticas que se han utilizado son:

- Separar la API en recursos lógicos. Los recursos son sustantivos que tienen sentido desde una perspectiva del consumidor de la API.
- Los recursos se escriben en plural para mantener URLs consistentes. De esta forma, se evita lidiar con plurales irregulares y se simplifica la labor del consumidor de la API.
- Se devuelven códigos de estado HTTP ya estandarizados:
    - 200 (OK): respuesta estándar para peticiones correctas.
    - 201 (Created): la petición ha sido completada y se ha creado un nuevo recurso.
    - 204 (No Content): la petición se ha completado con éxito, pero la respuesta no tiene ningún contenido.
    - 400 (Bad Request): la petición contiene parámetros erróneos.
    - 401 (Unauthorized): la autentificación ha fallado.
    - 404 (Not Found): recurso no encontrado.
    - 409 (Conflict): la solicitud no pudo ser procesada debido a un conflicto con el estado del recurso actual. Se utiliza cuando, por ejemplo, se intenta crear o actualizar un recurso que contiene un campo que incumple una restricción de unicidad en la base de datos.
    - 500 (Internal Server Error): error interno del servidor.
- Se utilizan peticiones HTTP donde cada método tiene un significado específico para manejar acciones CRUD (crear, leer, actualizar y borrar). Los métodos HTTP se utilizan de la siguiente forma:
    - GET: para devolver un recurso.
    - POST: para crear un recurso.
    - PUT: para actualizar un recurso.
    - DELETE: para eliminar un recurso.
- Se filtran recursos utilizando un único parámetro de consulta por cada campo que implemente el filtro.

A continuación, se muestran una serie de ejemplos acerca de las URLs pragmáticas explicadas anteriormente:

- *GET /exercises* – Devuelve una lista de ejercicios
- *GET /exercises?title=foo* – Devuelve una lista de ejercicios titulados "*foo*"
- *GET /exercises/1* – Devuelve el ejercicio #1
- *POST /exercises* – Crea un nuevo ejercicio
- *PUT /exercises/1* – Actualiza el ejercicio #1

- *DELETE /exercises/1* – Elimina el ejercicio #1
- *GET /exercises/1/opinions* – Devuelve las opiniones asociadas al ejercicio #1
- *POST /doctors/1/children/1* – Se asocia el bebé #1 al fisioterapeuta #1
- *DELETE /doctors/1/children/1* – Se desasocia el bebé #1 del fisioterapeuta #1

9.1.3 Control de acceso y seguridad

Para que el acceso a la información de la aplicación sea seguro, se ha utilizado el estándar RFC7519 [17], conocido como JSON Web Token o JWT. Este estándar define un mecanismo seguro para la transmisión de información entre varias partes mediante un objeto JSON.

El inicio de sesión implementado, utilizando JSON Web Token, funciona de la siguiente manera:

1. El cliente realiza una petición al servidor, enviándole las credenciales introducidas por el usuario, para realizar el inicio de sesión.
2. Si las credenciales son válidas, el servidor genera el objeto JSON y lo encripta utilizando una clave secreta que sólo él conoce.
3. El servidor devuelve el objeto JSON encriptado (*token*) al cliente.

Una vez se ha iniciado sesión, si se quiere acceder a un recurso protegido de la API:

4. El cliente envía el objeto JSON cifrado devuelto por el servidor en la cabecera "*Authorization*" de HTTP con el formato "*Bearer <token>*".
5. El servidor valida que el *token* enviado es correcto. Si lo es, permitirá el acceso al recurso solicitado y, si no lo es, devolverá una respuesta con código 401 (Unauthorized).

Para la validación de las credenciales, llevada a cabo en el paso 2, se ha utilizado la librería *CryptoHelper* [18]. Esta librería permite cifrar contraseñas mediante una función de cifrado PBKDF2 [19] y comprobar si una contraseña cifrada se corresponde con otra sin cifrar. Para que una contraseña sea considerada válida, se ha establecido que debe contener un mínimo de 8 caracteres.

Por último, para que el mecanismo de JSON Web Token sea seguro, las comunicaciones entre la aplicación móvil y el servidor se realizan mediante el protocolo HTTPS, de tal forma que la información viaja cifrada. Esto evita la sustracción de los datos (mediante ataques de tipo *man-in-the-middle* [20]) enviados en el cuerpo de la petición, así como de las cabeceras entre las cuales se encuentra el *token*.

9.1.4. Configuración de la API

La configuración de la API se lleva a cabo en la clase *Startup.cs*, ubicada en el proyecto PrematureKidsAPI. En ella se encuentran dos métodos:

- *Configure*: configuración de la API
- *ConfigureServices*: se añaden servicios al contenedor de servicios

La adición de servicios en el contenedor se realiza mediante el patrón de Inyección de Dependencias, de tal forma que se logra una Inversión de Dependencias. El patrón de Inyección de Dependencias y el principio de Inversión de Dependencias funcionan de la siguiente manera:

- En vez de instanciar objetos dentro de una clase, se le suministran
- La declaración de dichos objetos se realiza mediante interfaces

Así, se logra que las clases dependan de abstracciones y no de concreciones, de tal forma que el código se vuelve más modular y mantenible.

Gracias a la Inyección de Dependencias, los controladores tienen acceso a los distintos servicios que se hayan inyectado en el contenedor.

En el método *ConfigureServices* de la clase *Startup.cs* se realiza la configuración de diferentes servicios como:

- CORS: es un mecanismo para permitir acceder a recursos desde un dominio distinto al del servidor.
- Base de datos: se establece la cadena de conexión con la base de datos.
- Cliente SMTP: se establece la configuración necesaria para el envío de correos, utilizado para enviar a los fisioterapeutas la contraseña que deben utilizar en su primer acceso a la aplicación.
- Autenticación: se establece JSON Web Token como el mecanismo de autenticación y se realiza su configuración.
- Formatos aceptados: se establecen los formatos aceptados por la API. Por defecto, se utiliza el formato JSON, pero también se ha añadido la posibilidad de utilizar el formato XML.
- RepositoryWrapper: es un contenedor de repositorios.
- ActionFilters: son funciones que se pueden ejecutar antes de un método de un controlador de la API.

En el método *Configure* de la clase *Startup.cs* se configuran distintos parámetros de la API como:

- ExceptionMiddleware: mecanismo implementado para lograr una gestión de errores global en la API.
- StaticFiles: se configura la ruta del directorio en el cual se guardarán los ficheros subidos desde la aplicación. En este caso, será en el directorio *wwroot/uploads/*.

9.1.5. ActionFilters

Como ya se ha explicado anteriormente, un ActionFilter es una función que se ejecuta antes de un método de un controlador de la API. Es un mecanismo muy útil que nos permite encapsular código que se repite en distintos métodos.

Los ActionFilters implementados son:

- ValidationFilterAttribute: valida que el modelo de la entidad enviada en el cuerpo de la petición es correcto. Si el modelo no es válido devuelve un error 400 (Bad Request).

- ValidateEntityExistsAttribute: valida que la entidad existe en la base de datos utilizando su identificador, el cual se envía en la URL. Si la entidad no existe en la base de datos devuelve un error 404 (Not Found).

A continuación, se muestra un ejemplo de un método de la API (actualizar un ejercicio) que hace uso de los ActionFilters implementados:

```
[HttpPut("{id}"), Authorize(Roles = "doctor")]
[ServiceFilter(typeof(ValidationFilterAttribute))]
[ServiceFilter(typeof(ValidateEntityExistsAttribute<Exercise>))]
public IActionResult UpdateExercise(Guid id, [FromBody] Exercise exercise)
{
    _repository.Exercise.UpdateExercise(HttpContext.Items["entity"] as Exercise, exercise);
    return NoContent();
}
```

Ilustración 11: ActionFilters en un método de la API

Gracias a los ActionFilters, ubicados encima de la cabecera del método, conseguimos que en un método de tan solo dos líneas se realicen todas estas tareas:

- Comprobar si el ejercicio existe en la base de datos
- Comprobar que todos los campos del ejercicio son válidos
- Actualizar el ejercicio en la base de datos
- Devolver una respuesta con un código de estado según lo que haya ocurrido (400 si el modelo del ejercicio no es válido, 404 si el ejercicio no existe o 204 si se actualizó el ejercicio correctamente)

9.1.6. Clases de entidad

Entity Framework es un mapeador relacional de objetos utilizado por el *framework* .NET que permite trabajar con una base de datos mediante objetos .NET.

Para poder hacer uso de Entity Framework, se debe trasladar el modelo de datos definido en la base de datos a clases de entidad que representan las entidades y las relaciones entre ellas.

En el directorio “Models”, ubicado en el proyecto “Entities”, se han implementado las distintas entidades definidas en el modelo de datos, el cual puede verse definido en el diagrama de clases que se muestra a continuación:

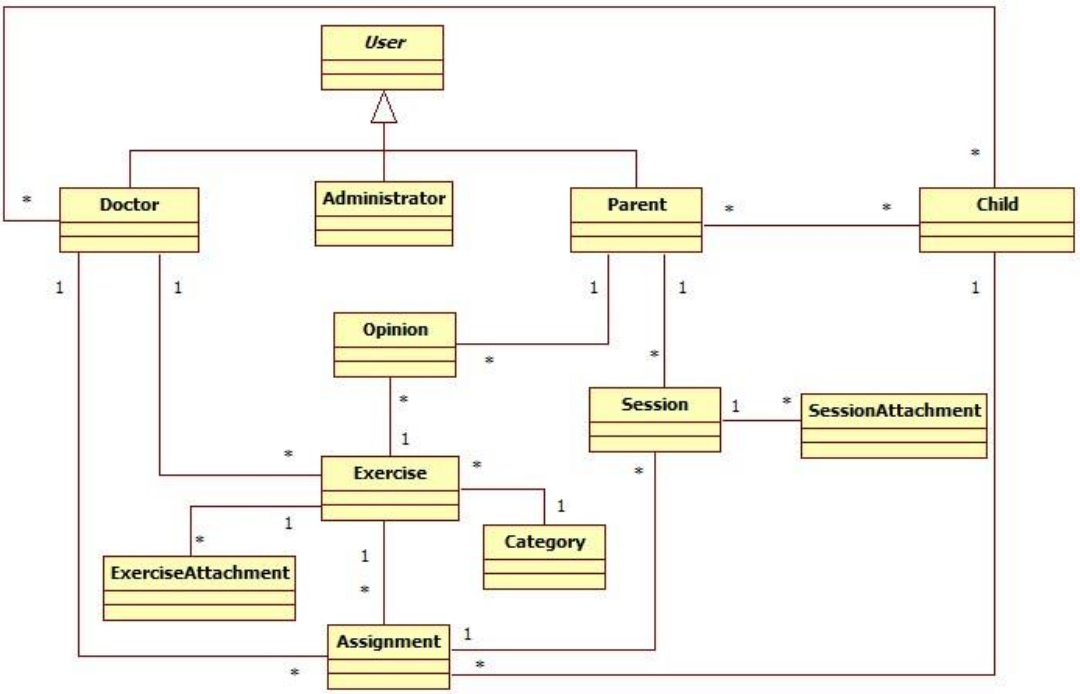

Ilustración 12: Diagrama de clases de entidad

Como se puede ver, las clases “Administrator”, “Parent” y “Doctor” heredan de la clase “User”. Sin embargo, no ha sido posible implementar la herencia debido a que la versión Core de Entity Framework no admite la herencia TPT (*table-per-type*), la cual consiste en crear una tabla diferente por cada clase heredada.

Además, entre las entidades implementadas, aparecen las clases “ChildDoctor” y “ChildParent”. Estas clases representan las relaciones *muchos-a-muchos* entre pacientes-fisioterapeutas y tutores-bebés. Son necesarias puesto que la versión Core de Entity Framework precisa de ellas para el correcto funcionamiento de las relaciones *muchos-a-muchos*.

9.1.7. Capa de acceso a base de datos

Para que la interacción entre los controladores de la API y la base de datos sea lo más sencillo posible, se ha utilizado el patrón Repositorio. Este patrón permite crear una capa de abstracción entre el acceso a los datos y la capa de lógica de negocio de la aplicación. La lógica de acceso a los datos se separa en un conjunto de clases denominadas repositorios, las cuales tienen la responsabilidad de interactuar con la base de datos, leyendo y escribiendo datos en ella.

Utilizando este patrón se consigue que el código sea mucho más limpio y más fácil de mantener y reutilizar.

En el diagrama que se muestra a continuación, se mostrará un ejemplo del uso del patrón Repositorio:

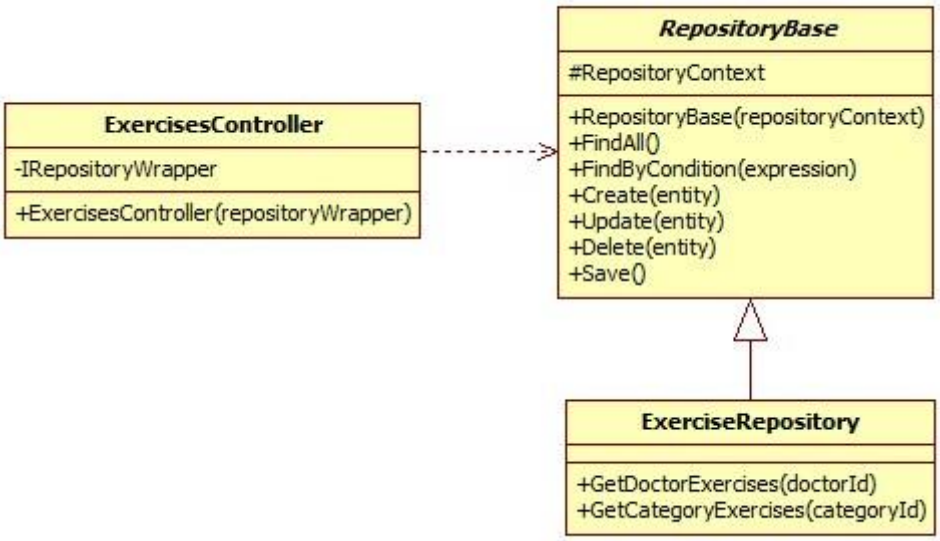

Ilustración 13: Patrón Repositorio

En la implementación del patrón Repositorio realizada:

- *RepositoryBase* es una clase abstracta que contiene los métodos para las operaciones CRUD (crear, leer, actualizar y eliminar).
- Todos los repositorios heredan de *RepositoryBase* y existe un repositorio por cada clase de entidad definida.
- Cada subclase de *RepositoryBase* contiene métodos específicos para interactuar con la base de datos.
- Si un método de un controlador requiere interactuar con la base de datos, hará una llamada al método del repositorio correspondiente, el cual contiene toda la lógica asociada a la interacción con la base de datos.

Para no tener que declarar un campo por cada repositorio utilizado en los controladores de la API, se ha creado la clase *RepositoryWrapper*, que consiste en un contenedor de repositorios que permite que, mediante Inyección de Dependencias, una instancia de *RepositoryWrapper* sea inyectada en el constructor de los controladores de la API. Este contenedor incluye instancias de los distintos repositorios asociados a cada clase de entidad. Se ha implementado un patrón *Singleton* para que no coexistan varias instancias de un mismo repositorio en el servidor.

9.2. Frontend

Para el desarrollo del *frontend*, se ha utilizado el *framework* de programación Ionic en su versión 4, acompañado del *framework* de programación Angular en su versión 6. Ionic permite, con un único desarrollo, la creación de aplicaciones móviles multiplataforma que funcionan tanto en la plataforma Android como en la plataforma iOS.

El desarrollo se realiza, principalmente, en HTML, CSS y TypeScript/JavaScript (el código se programa en TypeScript y se convierte en JavaScript gracias a un transpilador), lenguajes muy extendidos en la comunidad de desarrolladores.

TypeScript aprovecha todo el potencial de JavaScript, añadiéndole tipado estático a las variables y otras características como la creación de objetos mediante clases.

De acuerdo con una encuesta realizada por la página web Stack Overflow [15], el *framework* Angular ha sido el primer *framework* de programación de *frontend* para programación web más utilizado en el año 2018. Además, JavaScript ha sido el lenguaje de programación más utilizado en el mismo año y TypeScript ha sido el cuarto lenguaje que más gusta a los programadores.

9.2.1. Estructura del proyecto

Ionic proporciona un esqueleto bien definido en los proyectos que lo utilizan. Prácticamente todo el código de la aplicación se encuentra bajo el directorio */src/*. La estructura de este directorio es la siguiente:

- */src/app/*: en este directorio se encuentran todos los componentes de la aplicación.
- */src/assets/*: este directorio se utiliza para guardar ficheros estáticos como imágenes, iconos, etc.
- */src/environments/*: en este directorio se definen y configuran diferentes entornos de desarrollo.
- */src/theme/*: este directorio contiene un archivo con distintas variables que permiten establecer el tema de la interfaz gráfica de la aplicación.

Dentro del directorio */src/app/*, se encuentran el módulo y el componente raíz de la aplicación (*AppModule* y *AppComponent*). Además, se han definido los siguientes directorios:

- */src/app/private/*: en este directorio se encuentran los distintos módulos que deben ser cargados una vez el usuario haya iniciado sesión en la aplicación. Hay un módulo para el administrador, uno para los fisioterapeutas, otro para los tutores y un último módulo que es compartido por todos ellos.
- */src/app/public/*: este directorio contiene los distintos módulos que deben ser cargados siempre, aunque el usuario no haya iniciado sesión.
- */src/app/services/*: es el directorio que contiene distintos servicios utilizados en distintos componentes de la aplicación.
- */src/app/shared/*: módulo que contiene los *pipes* desarrollados.

Para aumentar el rendimiento de la aplicación, se ha utilizado la técnica de *lazy loading*, que consiste en cargar los componentes a medida que se van necesitando. Para hacer esto, el *router* de la aplicación va activando/desactivando los distintos módulos privados según el rol del usuario que haya iniciado sesión. La forma de hacer esto es mediante la propiedad *canActivate* asociada a cada ruta que representa un módulo.

Para definir cuándo debe activarse un módulo, se han implementado una serie de servicios, los cuales serán explicados en el siguiente punto.

9.2.2. Servicios

Según la documentación oficial de Angular [21], la lógica que no está asociada a una vista específica y que se quiere compartir entre distintos componentes se debe implementar en un servicio. Los servicios pueden ser utilizados en las clases de los componentes mediante Inyección de Dependencias.

A continuación, se explicarán brevemente los diferentes servicios implementados:

- DateService: servicio que contiene funciones de apoyo para el tratamiento de fechas.
- FirebaseChatService: este servicio contiene funciones para interactuar con la base de datos Realtime Database de Firebase.
- HttpService: contiene las funciones necesarias para realizar las peticiones HTTP a la API REST.
- AuthGuardService: es el servicio utilizado para activar diferentes módulos cuando el usuario se ha autenticado en la aplicación. Para activar los módulos según el rol del usuario también existen los servicios *AdministratorAuthGuard*, *DoctorAuthGuard* y *ParentAuthGuard*, donde cada uno indicará si se pueden activar, o no, los distintos módulos asociados a dicho rol.
- AuthenticationService: servicio que contiene funciones de apoyo para controlar la autenticación del usuario en la aplicación. Algunas de las funciones que contiene este servicio son aquellas que nos permiten guardar, recuperar o descifrar el objeto JSON (*token*) devuelto por la API REST.

9.2.3. Pipes

En Angular, un *pipe* es una clase que nos permite transformar datos de entrada en otros datos de salida. Son muy útiles en las plantillas de los distintos componentes para procesar la información que se muestra al usuario.

Los *pipes* que se han implementado son:

- SearchPipe: filtra los elementos de una lista que no cumplen una condición. Se utiliza cuando el usuario aplica un filtro sobre una lista de elementos.
- SortPipe: ordena los elementos de una lista según el campo indicado.

9.2.4. Plugins nativos

Para poder acceder a funcionalidades nativas del dispositivo móvil, Ionic utiliza el *framework* Cordova, el cual nos permite encapsular una aplicación web en una aplicación móvil nativa.

Para acceder a componentes nativos del sistema operativo sobre el cual se ejecuta la aplicación, se deben utilizar *plugins* de Cordova. Los *plugins* que se han utilizado para el desarrollo de la aplicación móvil son los siguientes:

- Camera: utilizado para acceder a la galería de imágenes y vídeos del dispositivo

- FileTransfer: utilizado para la subida de archivos (vídeos e imágenes asociados a ejercicios y sesiones) al servidor
- PhotoViewer: utilizado para mostrar imágenes a pantalla completa
- StreamingMedia: utilizado para la reproducción de vídeos

9.2.5. Firebase

Para poder interactuar con la base de datos Realtime Database de Firebase, con el objetivo de poder utilizar el chat a tiempo real con el que cuenta la aplicación, se debe instalar la librería *Firebase* [22] en el proyecto.

Primero, se debe inicializar el objeto que nos permitirá interactuar con la API de Firebase en el constructor del componente raíz de la aplicación (*AppComponent*). Para inicializar dicho objeto, se debe obtener el objeto JSON asociado al proyecto de Firebase que contiene la configuración requerida para poder interactuar con la API.

Para obtener este objeto JSON hay que acceder al proyecto creado en "*Firebase console*" (https://console.firebase.google.com) y hacer *click* en "Agregar app".

Además, para que el acceso a la base de datos de Firebase sea seguro, se han configurado sus reglas de acceso de tal forma que un usuario que no haya iniciado sesión en la aplicación no pueda leer ni escribir en ella. A continuación, se muestran las reglas definidas en la base de datos:

```
"rules": {
      ".read": "auth != null",
      ".write": "auth != null"
}
```

Para acceder a la base de datos de Firebase con credenciales de acceso desde la aplicación móvil se llevan a cabo las siguientes acciones:

1. Desde la aplicación móvil, se solicita a la API REST un *token* de acceso a Firebase
2. En la API REST, se genera un *token* utilizando una clave privada obtenida en la consola del proyecto de Firebase y se devuelve a la aplicación móvil
3. Con el *token* devuelto, la aplicación móvil realiza un inicio de sesión en Firebase y ya puede leer y escribir en la base de datos

10. Capítulo 10: Pruebas

10.1. Pruebas de integración

Las pruebas realizadas se han realizado en un entorno de desarrollo, utilizando equipos personales. Esto se debe a que no se dispone de un servidor en el cual desplegar el *backend* de la aplicación para poder simular un entorno de producción.

Sin embargo, se han realizado diferentes pruebas de integración para comprobar que un fisioterapeuta y un tutor, cuyos dispositivos móviles se encuentran en distintas redes, que a su vez son distintas a la del servidor, son capaces de interaccionar entre ellos de manera satisfactoria.

Para comprobar que esto es así:

- Se han desplegado la API REST y la base de datos en un equipo y se ha realizado una redirección de puertos hacia el par [IP,puerto] que identifica a la API REST, de tal forma que los dispositivos móviles puedan acceder a ella desde una red externa.
- Se han conectado dos *smartphones*, en los que se ha instalado la aplicación móvil, a distintas redes, uno mediante conexión WiFi y el otro utilizado una red de datos móviles.

Una vez hecho esto, se ha comprobado que:

- Desde la aplicación, las peticiones realizadas a la API REST se realizan correctamente.
- Una acción llevada a cabo por un fisioterapeuta se ve reflejada inmediatamente en la aplicación que está siendo utilizada por un tutor y viceversa.
- La comunicación mediante el chat implementado, utilizando Firebase, funciona correctamente y es en tiempo real.

10.2. Pruebas de usabilidad

Para garantizar una buena experiencia de usuario con la aplicación es de vital importancia realizar pruebas de usabilidad constantemente, puesto que la usabilidad es un factor clave para que la interacción persona-máquina sea correcta.

Para ello, ha sido de vital importancia la fase de revisión del prototipo en cada iteración realizada. Durante estas fases de revisión, se han realizado pruebas de usabilidad con usuarios potenciales de la aplicación para ajustar las funcionalidades desarrolladas a las necesidades de los usuarios.

Para realizar las pruebas de usabilidad, se ha contado con la ayuda de la fisioterapeuta Dña. María del Mar Batista Guerra. Sin embargo, no se han podido realizar demostraciones de la aplicación con tutores que participen o hayan participado el servicio de fisioterapia, lo cual aportaría información que sería muy importante para mejorar la aplicación.

En las demostraciones, se ha comprobado que los usuarios:

- Encuentran que el menú de navegación principal está bien estructurado

- Navegan por la aplicación correctamente
- La información que encuentran en cada vista es correcta, es decir, que ni falta ni sobra información
- Entienden los distintos mensajes de error mostrados en la aplicación cuando introducen datos incorrectos y son capaces de solventar dichos errores
- Son conscientes de cuándo los datos son guardados en el sistema
- Identifican la función de los distintos iconos utilizados en la interfaz gráfica

11. Capítulo 11: Resultados, conclusiones y trabajo futuro

11.1. Resultados y conclusiones

Tras la realización de este proyecto, podemos concluir en que todos los objetivos propuestos al comienzo del mismo han sido cumplidos.

Los fisioterapeutas, utilizando la aplicación móvil, pueden:

- Crear una base de datos de ejercicios con contenido multimedia
- Asignar ejercicios personalizados a los tutores
- Visualizar y valorar las sesiones llevadas a cabo por los tutores con sus bebés

A su vez, los tutores pueden:

- Ver los ejercicios asignados por los fisioterapeutas
- Hacer llegar imágenes y vídeos de las distintas sesiones a los fisioterapeutas

Además, el sistema de chat en tiempo real de la aplicación permite que la resolución de dudas sea más rápida y que no sea necesario utilizar otros sistemas de mensajería.

Con todos los objetivos cumplidos, los problemas detectados durante la fase de análisis también han sido resueltos:

- Los tutores ya no reciben material impreso de apoyo para la realización de los ejercicios, lo cual evita el malgasto de recursos que conlleva la impresión de dicho material y ayuda a la conservación del medio ambiente
- Los fisioterapeutas cuentan con un sistema centralizado y especializado donde poder encontrar toda la información necesaria para llevar a cabo el seguimiento de los neonatos de manera cómoda y sencilla

De esta manera, la aplicación ayuda a que el servicio de fisioterapia para la atención de neonatos sea de mayor calidad y que, además, sea menos costoso para los centros de salud al no tener que recurrir a la impresión de documentos.

A nivel personal, el haber llevado a cabo un proyecto desde cero ha sido una experiencia muy enriquecedora al poder haber realizado las fases de análisis y diseño sobres las cuales se ha hecho gran hincapié en la intensificación de Ingeniería del Software del Grado. El haber tenido que interaccionar con un cliente real de la aplicación, ha sido gran una oportunidad para poner en práctica los conocimientos adquiridos para la obtención de requisitos.

También, el haber desarrollado un sistema en el cual interviene una aplicación móvil, hace que el objetivo de haber cursado la especialidad de Programación de Dispositivos Móviles del Máster se haya cumplido.

Además, hay que destacar positivamente todo el conocimiento adquirido acerca de las tecnologías utilizadas, .NET Core para el desarrollo del *backend*, Ionic 4 para el desarrollo de la aplicación móvil y Firebase para la implementación del chat en tiempo real.

Por último, hay que destacar que ha sido muy gratificante haber podido contribuir en la mejora de un servicio que pertenece a un campo tan importante como es el de la salud, el cual tiene un impacto muy positivo en la calidad de vida de las nuevas generaciones.

11.2. Trabajo futuro

Todo proyecto, aun cumpliendo todos los objetivos propuestos al comienzo del mismo, siempre es susceptible a mejoras y/o ampliaciones. A día de hoy, con el sistema desarrollado se han alcanzado todos los objetivos que fueron propuestos al comienzo del proyecto. Sin embargo, se podrían mejorar algunas funcionalidades ya implementadas e implementar nuevas funcionalidades que aumentaran la utilidad del sistema.

A continuación, se describen algunas posibles mejoras sobre las funcionalidades ya implementadas, además de nuevas funcionalidades que harían que el sistema sea más completo:

- Permitir que en el chat se puedan hacer referencias (mediante enlaces) a tareas o sesiones
- Implementar un sistema de validación de cuentas de correo electrónico para los tutores mediante el envío de un email con un enlace de confirmación
- Mejorar la pantalla de inicio de la aplicación (logo, imágenes, texto descriptivo, etc.)
- Aplicar un tema de colores a la aplicación para mejorar la estética de la interfaz de usuario
- Permitir subir documentos en formato PDF asociados a un ejercicio
- Permitir a los tutores subir imágenes y vídeos utilizando la cámara del dispositivo
- Implementar un sistema para la recuperación de contraseñas
- Hacer que las llamadas a la API utilicen paginación cuando la cantidad de información devuelta pueda ser muy elevada, de tal forma que se reduzca el consumo de datos
- Implementación de notificaciones *push* que permitan a tutores y fisioterapeutas ser notificados cuando:
    - Un fisioterapeuta asigna un ejercicio a un tutor
    - Un tutor crea una sesión
    - Un fisioterapeuta aporta un comentario a una sesión
    - Se recibe un mensaje en el chat de la aplicación
    - Un tutor debe tener una sesión con un bebé

12. Capítulo 12: Bibliografía

[1] Wikipedia. "GNU General Public License", [en línea]. 17 de mayo de 2017. Disponible en: https://es.wikipedia.org/wiki/GNU_General_Public_License

[2] StarUML. "About StarUML", [en línea]. Disponible en: http://staruml.sourceforge.net/v1/license.php

[3] Wikipedia. "Licencia MIT", [en línea]. 17 de junio de 2017. Disponible en: https://es.wikipedia.org/wiki/Licencia_MIT

[4] Wikipedia. "Apache License", [en línea]. 4 de junio de 2017. Disponible en: https://es.wikipedia.org/wiki/Apache_License

[5] Postman, Inc. "Postman EULA", [en línea]. 25 de mayo de 2018. Disponible en: https://www.getpostman.com/terms/Postman_EULA_May_2018.pdf

[6] Microsoft Corporation. "Términos de la licencia de software de Microsoft, Microsoft Visual Studio Community 2017", [en línea]. Disponible en: https://visualstudio.microsoft.com/es/license-terms/mlt553321/

[7] Microsoft Corporation. "Microsoft software license terms, Microsoft Visual Studio Code", [en línea]. Disponible en: https://code.visualstudio.com/license

[8] Agencia Estatal Boletín Oficial del Estado. "Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal", [en línea]. 13 de diciembre de 1999. Disponible en: https://www.boe.es/buscar/pdf/1999/BOE-A-1999-23750-consolidado.pdf

[9] Agencia Estatal Boletín Oficial del Estado. "Reglamento de desarrollo de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal", [en línea]. 19 de enero de 2008. Disponible en: https://www.boe.es/eli/es/rd/2007/12/21/1720/dof/spa/pdf

[10] Aruquipa, Oscar. "Análisis comparativos de los distintos ciclos de vida en el software", [en línea]. 4 de octubre de 2012. Disponible en: http://oscararuquipacolque.blogspot.com.es/2012_10_01_archive.html

[11] Wikipedia. "Lenguaje unificado de modelado", [en línea]. 20 de junio de 2017. Disponible en: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

[12] Normén, Fredrik. "Using Web Services in a 3-tier architecture", [en línea]. 5 de noviembre de 2008. Disponible en: https://weblogs.asp.net/fredriknormen/using-web-services-in-a-3-tierarchitecture

[13] Software Engineering Stack Exchange. "Storing files in the database", [en línea]. 31 de mayo de 2012. Disponible en: https://softwareengineering.stackexchange.com/questions/150669/isit-a-bad-practice-to-store-large-files-10-mb-in-a-database

[14] Tognazzini, Bruce. "First Principles of Interaction Design", [en línea]. 5 de marzo de 2014. Disponible en: http://asktog.com/atc/principles-of-interaction-design/

[15] Stack Overflow. "Developer Survey Results 2018", [en línea]. 2018. Disponible en: https://insights.stackoverflow.com/survey/2018/

[16] Armentano, Lucila. "Buenas prácticas para el Diseño de una API RESTful Pragmática", [en línea]. 12 de febrero de 2017. Disponible en: https://elbauldelprogramador.com/buenaspracticas-para-el-diseno-de-una-api-restful-pragmatica/

[17] Jones, Michael B.; Bradley, John; Sakimura, Nat. "JSON Web Token (JWT)", [en línea]. Mayo de 2018. Disponible en: https://tools.ietf.org/html/rfc7519

[18] .NET Foundation. "CryptoHelper", [en línea]. 7 de marzo de 2018. Disponible en: https://www.nuget.org/packages/CryptoHelper/

[19] Wikipedia. "PBKDF2", [en línea]. 30 de diciembre de 2018. Disponible en: https://en.wikipedia.org/wiki/PBKDF2

[20] Wikipedia. "Ataque de intermediario", [en línea]. 25 de mayo de 2018. Disponible en: https://es.wikipedia.org/wiki/Ataque_de_intermediario

[21] Angular Team. "Architecture overview", [en línea]. Disponible en: https://angular.io/guide/architecture

[22] npm, Inc. "Firebase - App success made simple", [en línea]. 27 de diciembre de 2018. Disponible en: https://www.npmjs.com/package/firebase

Anexo I: Manual de usuario

General

Barra de herramientas

La barra de herramientas, ubicada en la parte superior de cada vista, cuenta con una serie de botones que son utilizados con frecuencia para llevar a cabo las distintas tareas dentro de la aplicación.

- Botón de salvado: guarda los cambios en la base de datos
- Botón de borrado: borra un elemento en la base de datos
- Botón de navegación: permite retroceder a la vista anterior

Ilustración 14: Barra de herramientas con botones de navegación, salvado y borrado

Menú

En la parte izquierda de la barra de herramientas también podemos encontrar el botón del menú lateral.

Ilustración 15: Barra de herramientas con botón de menú lateral

Cada rol de usuario tendrá un menú distinto según las funcionalidades a las que puede acceder. Tras presionar el botón de menú, se desplegará un menú lateral que permitirá al usuario acceder a dichas funcionalidades.

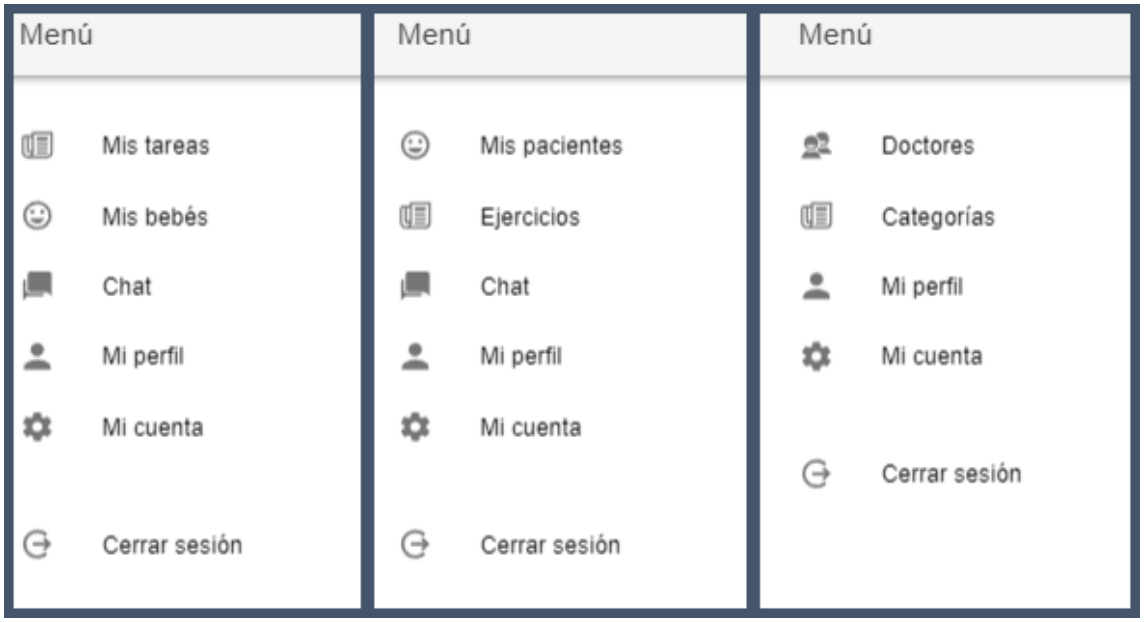

Ilustración 16: Menús laterales (tutores, fisioterapeutas y administrador)

Formularios

Todos los formularios de la aplicación cuentan con control de formularios. Esto hace que no se puedan guardar cambios en la base de datos si algún campo del formulario no es válido.

Cuando algún campo no es válido, se desactiva el botón de salvado de la barra de herramientas y se muestra un texto informativo para ayudar al usuario a corregir el error.

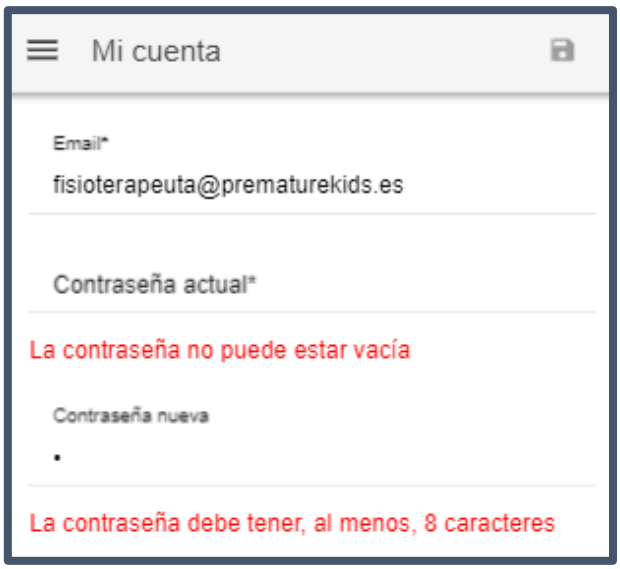

Ilustración 17: Formulario inválido

Además, se notifica al usuario cada vez que se lleva a cabo una escritura en la base de datos mediante un *toast*.

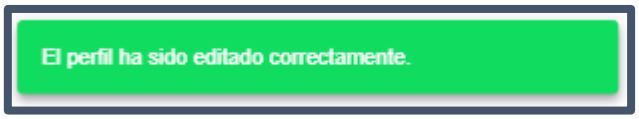

Ilustración 18: Notificación mediante toast

Vistas comunes

Pantalla de inicio

Pantalla principal de la aplicación.

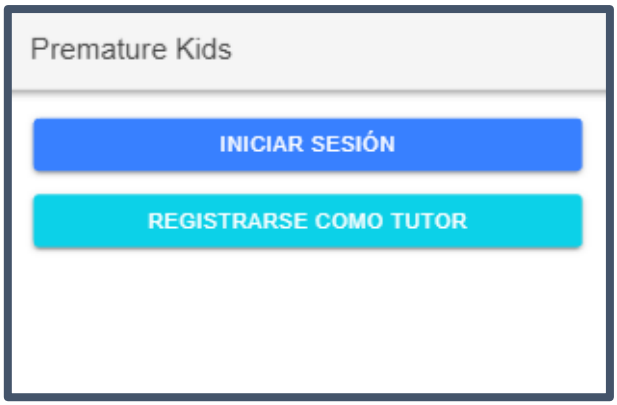

Ilustración 19: Pantalla de inicio

Inicio de sesión

Formulario de inicio de sesión, en el cual los usuarios podrán introducir sus credenciales de acceso para poder acceder a las funcionalidades privadas de la aplicación.

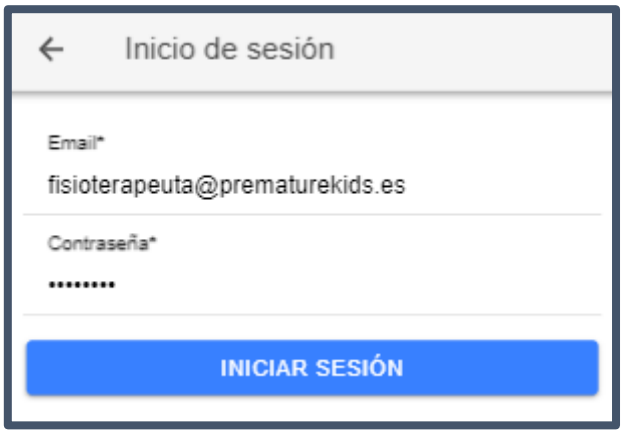

Ilustración 20: Formulario de inicio de sesión

Mi perfil

Los usuarios pueden editar la información de su perfil presionando el botón de salvado.

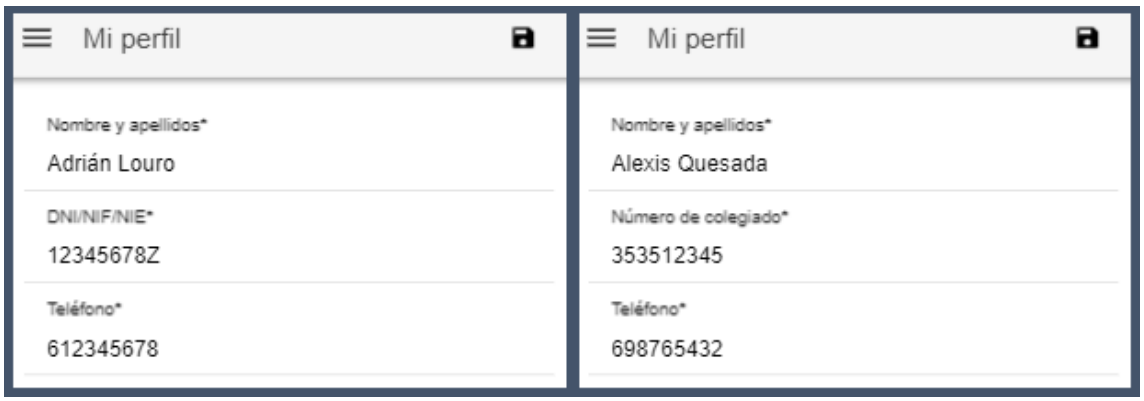

Ilustración 21: Formulario para la edición del perfil del usuario (tutor y fisioterapeuta)

Mi cuenta

Los usuarios pueden editar sus credenciales de acceso a la aplicación (email y contraseña) presionando el botón de salvado. Para poder hacerlo, es obligatorio introducir la contraseña actual por motivos de seguridad.

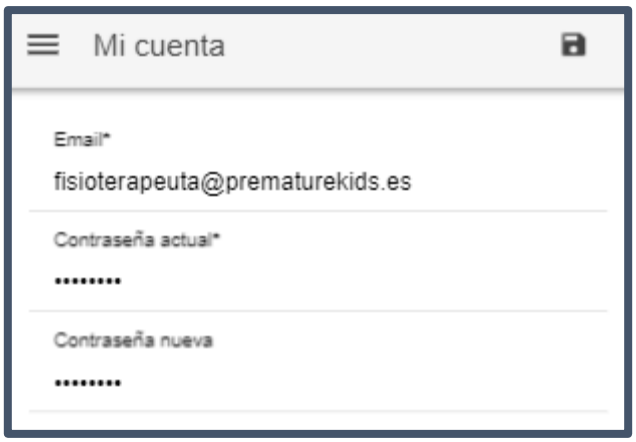

Ilustración 22: Formulario para la edición de las credenciales de acceso

Chats

Vista accesible por fisioterapeutas y tutores. Se muestra un listado con los chats del usuario. Se puede filtrar el listado utilizando la barra de búsqueda. Presionando el botón flotante se accede a la vista de contactos (para la creación de chats). Presionando un elemento de la lista se accede al chat.

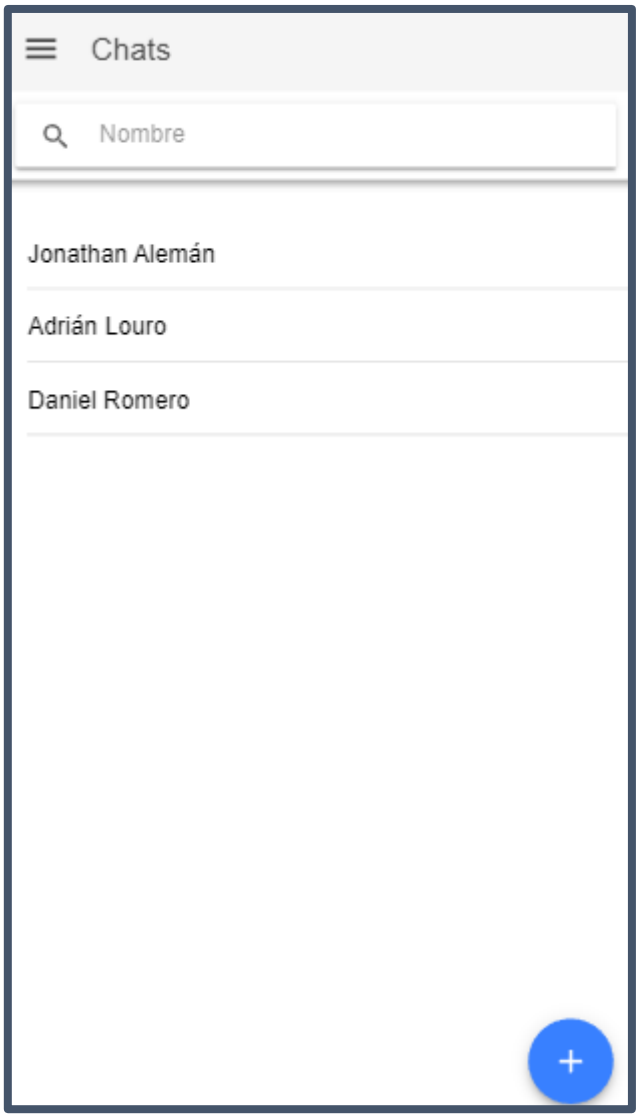

Ilustración 23: Listado de chats

Contactos

Vista accesible por fisioterapeutas y tutores. Se muestra un listado con los contactos del usuario. Se puede filtrar el listado utilizando la barra de búsqueda. Presionando un elemento de la lista se crea un chat con ese contacto o se accede al chat si este ya existe.

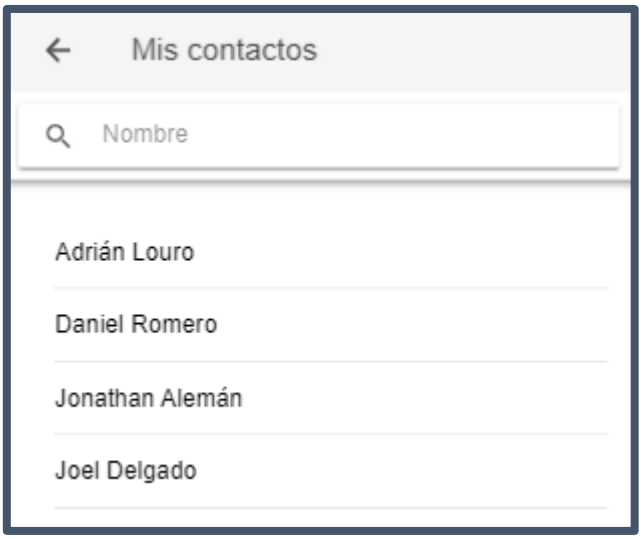

Ilustración 24: Listado de contactos

Chat

Vista accesible por fisioterapeutas y tutores. Se muestra el historial de mensajes del chat y se permite enviar mensajes.

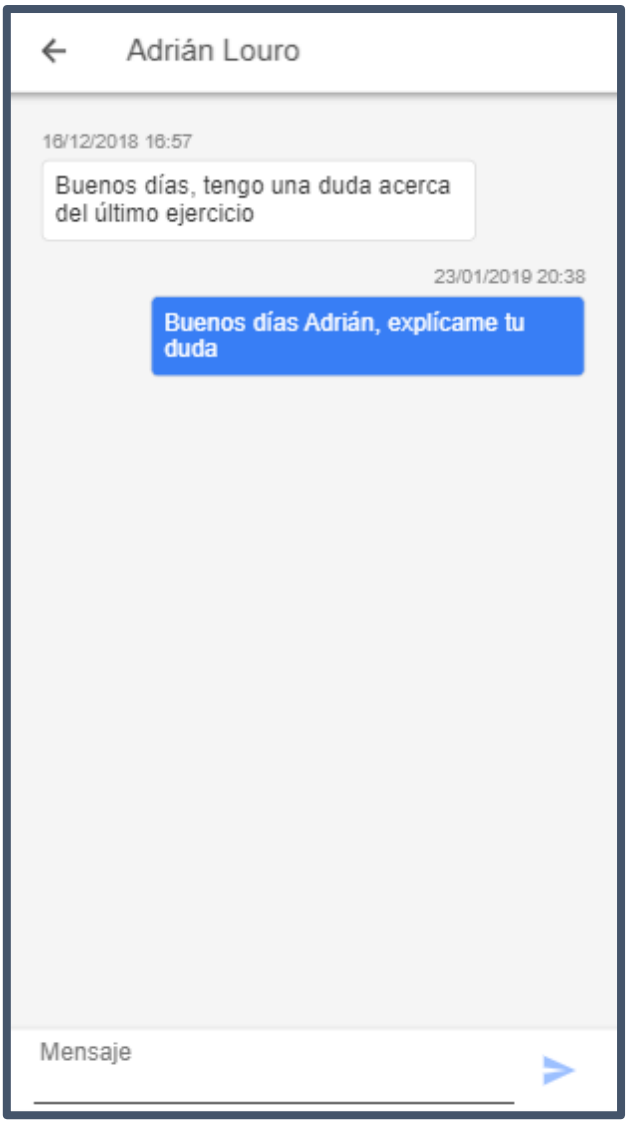

Ilustración 25: Chat entre fisioterapeuta y tutor

Tarea

Vista accesible por fisioterapeutas y tutores. Se muestran tres pestañas:

- Información: información asociada a la tarea. El fisioterapeuta que creó la tarea puede editarla presionando el botón de salvado y eliminarla presionando el botón de borrado.

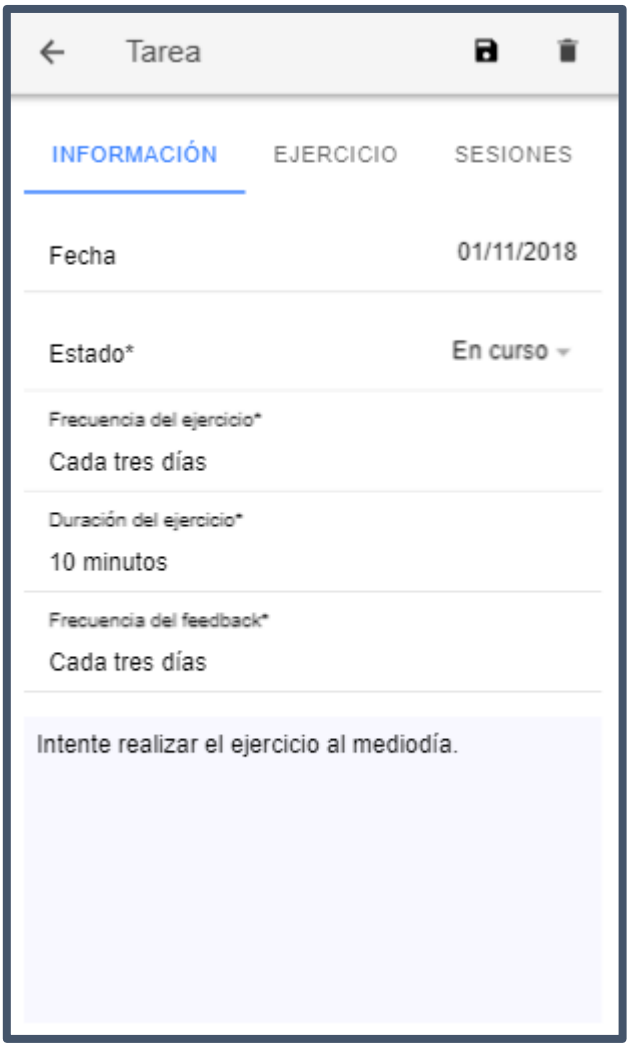

Ilustración 26: Vista de información de una tarea

- Ejercicio: se muestra la información y el contenido multimedia del ejercicio asociado a la tarea. El tutor podrá añadir una opinión acerca del ejercicio presionando “Escribir una opinión”.

Ilustración 27: Vista del ejercicio asociado a una tarea

- Sesiones: se muestra un listado de las sesiones asociadas a la tarea. Presionando un elemento de la lista, se accede a la vista de la sesión. Un tutor podrá añadir una nueva sesión presionando el botón flotante.

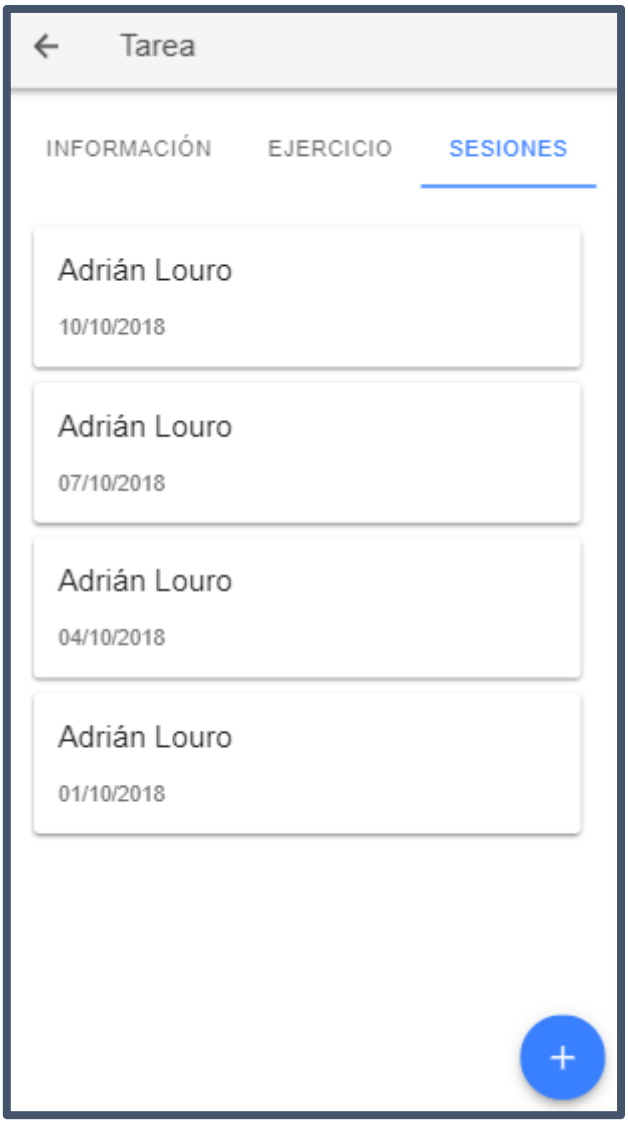

Ilustración 28: Listado de sesiones asociadas a una tarea

Sesión

Vista accesible por tutores y fisioterapeutas. Se muestran dos pestañas:

- Información: se muestra la información de la sesión. Un tutor puede editar dicha información presionando el botón de salvado o eliminar la sesión presionando el botón de borrado. Un fisioterapeuta puede aportar un comentario a la sesión.

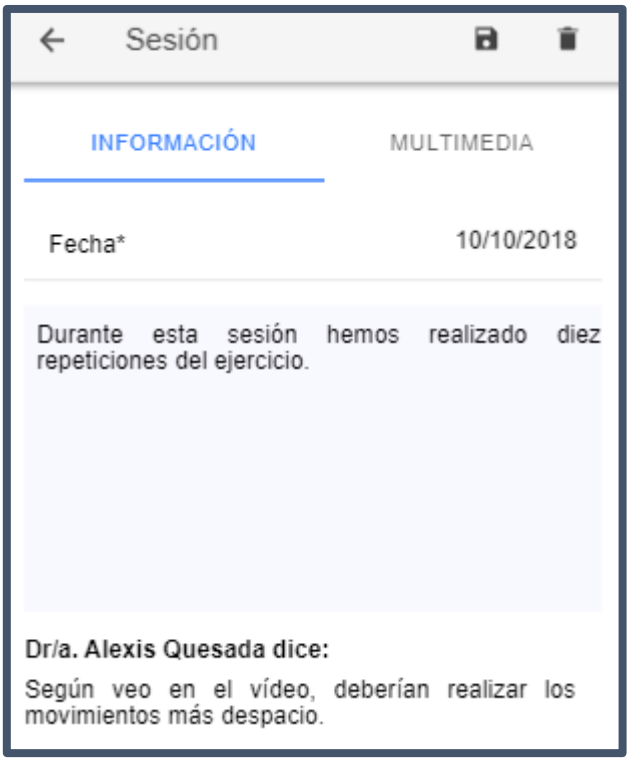

Ilustración 29: Vista de información de una sesión

- Multimedia: se muestra el contenido multimedia (vídeo e imágenes) asociado a la sesión. El tutor que creó la sesión puede subir y eliminar el vídeo y las imágenes de la sesión. Presionando una imagen, se puede ver en pantalla completa.

Ilustración 30: Vista de contenido multimedia de una sesión

Bebé/paciente

Vista accesible por tutores y fisioterapeutas. Se muestran tres pestañas:

- Tareas: se muestra un listado de tareas asociadas al paciente. Presionando un elemento de la lista se accede a la vista de la tarea. Un fisioterapeuta puede crear una nueva tarea asociada al paciente presionando el botón flotante.

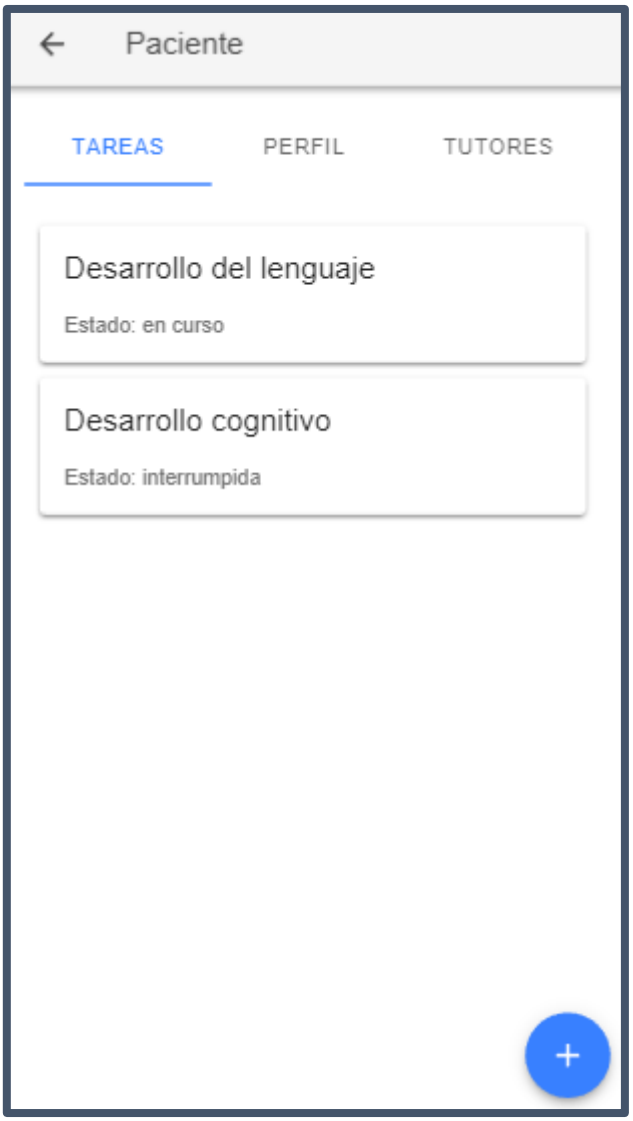

Ilustración 31: Vista de tareas de un paciente

- Perfil: se muestra la información asociada al paciente. Un fisioterapeuta puede editar la información del perfil del paciente presionando el botón de salvado. Además, puede eliminar al paciente del sistema presionando el botón de borrado. La información asociada al historial médico del neonato solo es accesible por el fisioterapeuta.

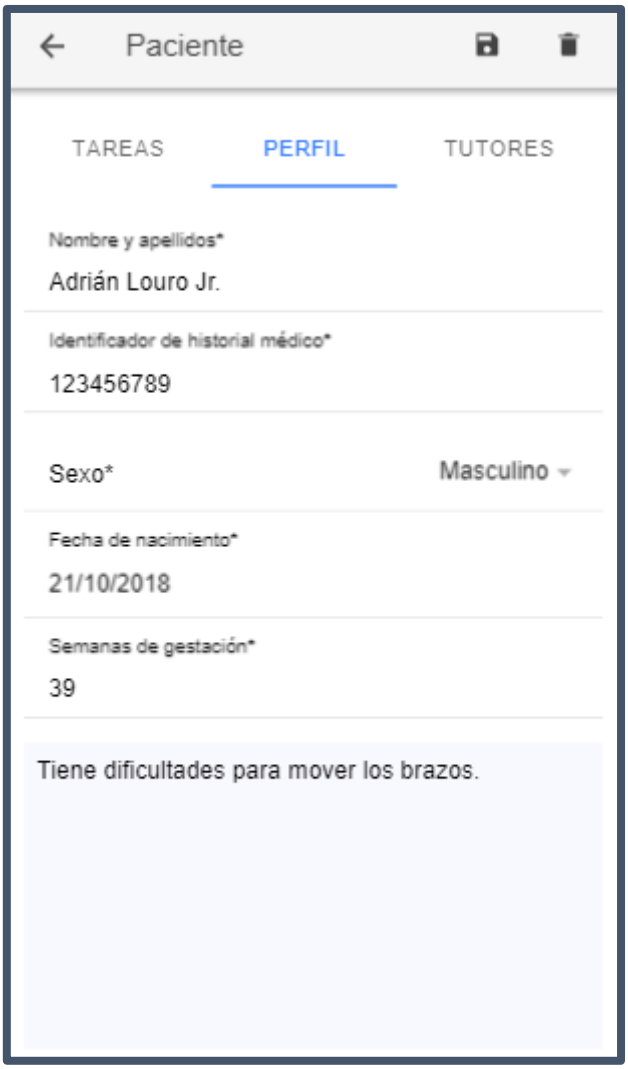

Ilustración 32: Perfil de un paciente

- Tutores: se muestra un listado con los tutores del neonato. Un fisioterapeuta puede desasociar un tutor del paciente presionando el icono que se encuentra a la derecha de la información del tutor. Además, puede asociar un nuevo tutor al paciente presionando el botón flotante.

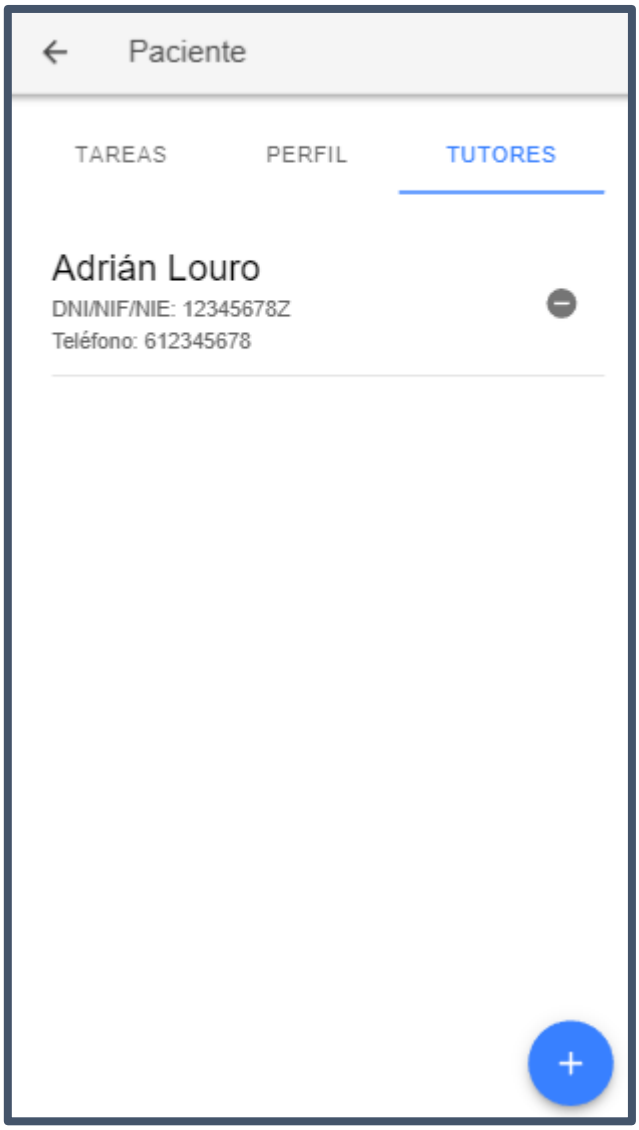

Ilustración 33: Vista de tutores de un paciente

Administrador

Fisioterapeutas

Se muestra un listado con los fisioterapeutas dados de alta en la aplicación. Se puede filtrar el listado utilizando la barra de búsqueda. Presionando el botón flotante se accede a la vista de creación de fisioterapeutas. Presionando un elemento de la lista se accede al perfil del fisioterapeuta.

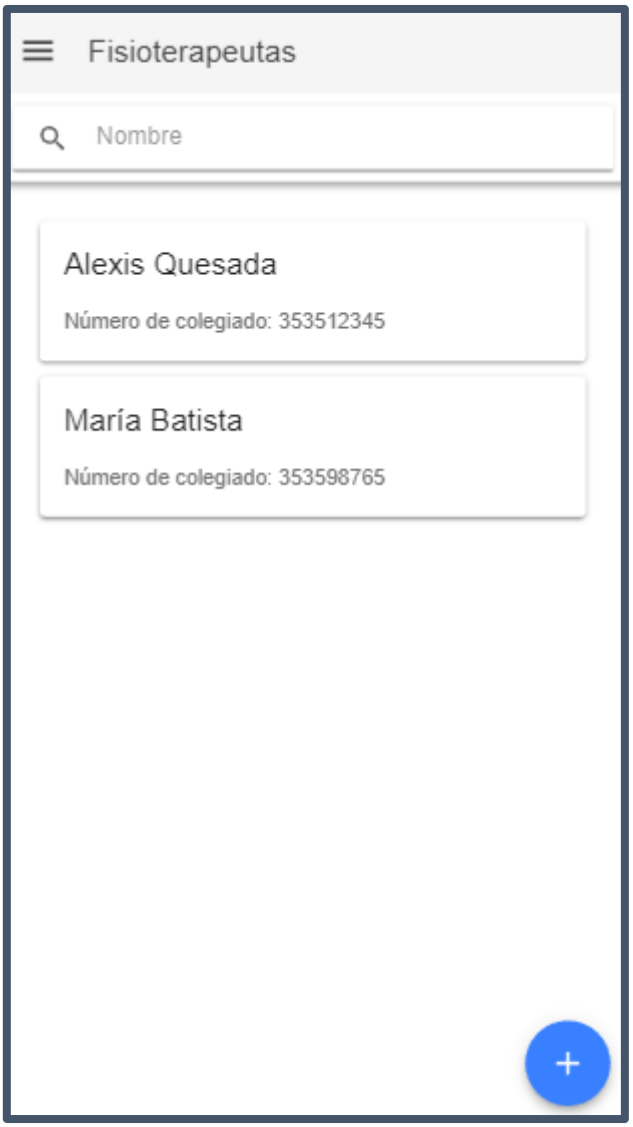

Ilustración 34: Listado de fisioterapeutas dados de alta en el sistema

Añadir fisioterapeuta

Se introducen los datos del fisioterapeuta y se presiona el botón de salvado para guardar los cambios. Una vez creado, se envía automáticamente un correo al email introducido con una contraseña generada aleatoriamente.

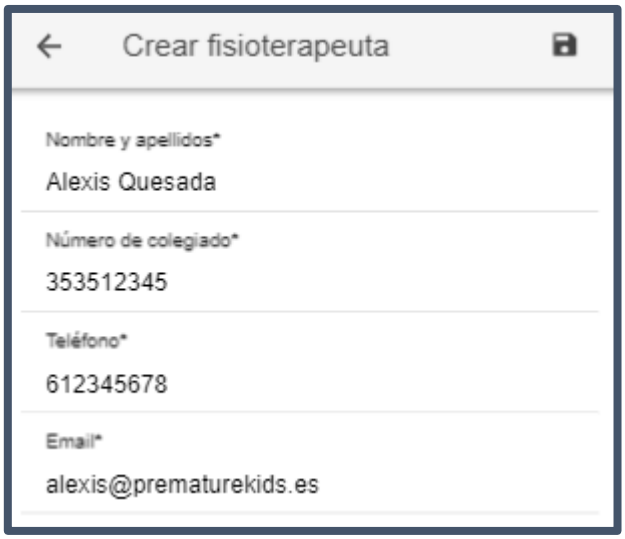

Ilustración 35: Formulario de creación de un fisioterapeuta

Perfil del fisioterapeuta

Se puede editar la información del fisioterapeuta presionando el botón de salvado. Además, se puede eliminar al fisioterapeuta presionando el botón de borrado y bloquear/desbloquear su acceso a la aplicación presionando el botón “Bloquear usuario” o “Desbloquear usuario”.

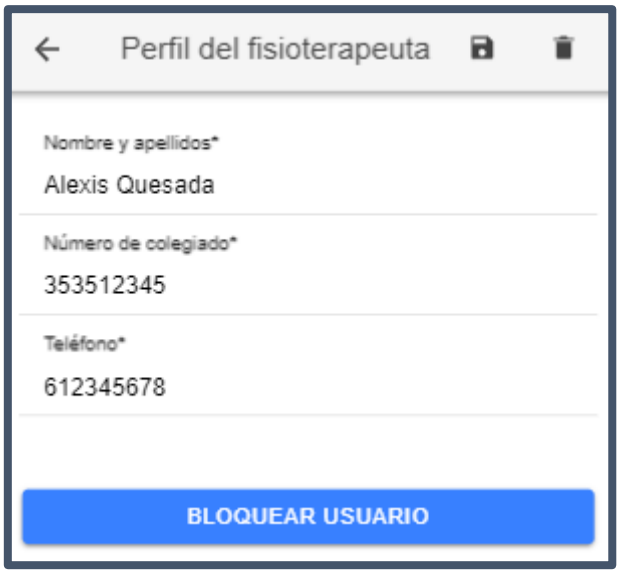

Ilustración 36: Perfil del fisioterapeuta

Categorías

Se muestra un listado con las categorías dadas de alta en la aplicación. Se puede filtrar el listado utilizando la barra de búsqueda. Presionando el botón flotante se accede a la vista de creación de categorías. Presionando un elemento de la lista se accede a la vista de edición/eliminación de la categoría.

Ilustración 37: Listado de categorías dadas de alta en el sistema

Añadir categoría

Se introduce el nombre de la categoría y se presiona el botón de salvado para guardar los cambios.

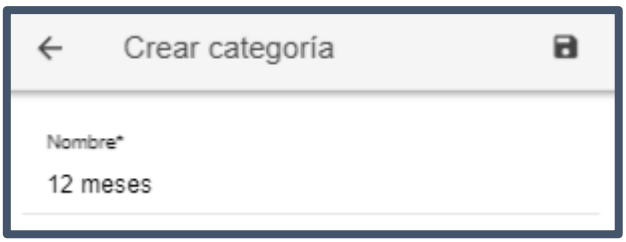

Ilustración 38: Formulario para la creación de categorías

Categoría

Se puede editar el nombre de la categoría presionando el botón de salvado. Además, se puede eliminar la categoría presionando el botón de borrado.

Ilustración 39: Vista de una categoría

Tutor

Registro

Se introducen los datos de acceso y del perfil y se aceptan los "Términos y Condiciones de Uso" y la "Política de Privacidad", los cuales pueden leerse presionando en sus respectivos enlaces. Una vez hecho esto, se presiona en el botón "Registrarse" para finalizar el registro.

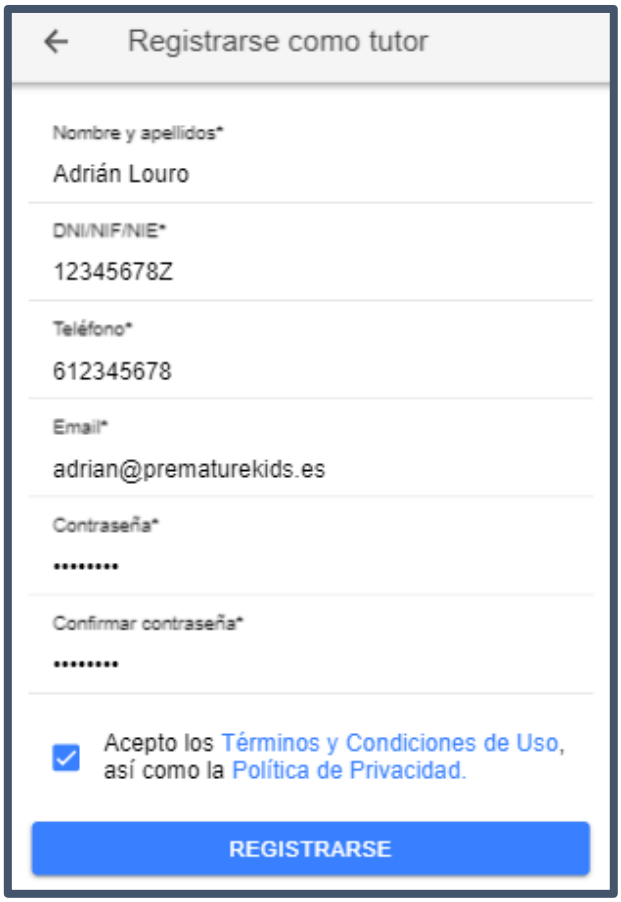

Ilustración 40: Formulario de registro para tutores

Mis tareas

Se muestra un listado de las tareas asignadas a todos sus bebés, clasificadas según el estado en el que se encuentra la tarea. Presionando un elemento de la lista, se accede a la vista de la tarea.

Ilustración 41: Tareas asociadas a un tutor, clasificadas según su estado

Mis bebés

Se muestra un listado con los bebés asociados al tutor. Presionando un elemento de la lista se accede al perfil del bebé.

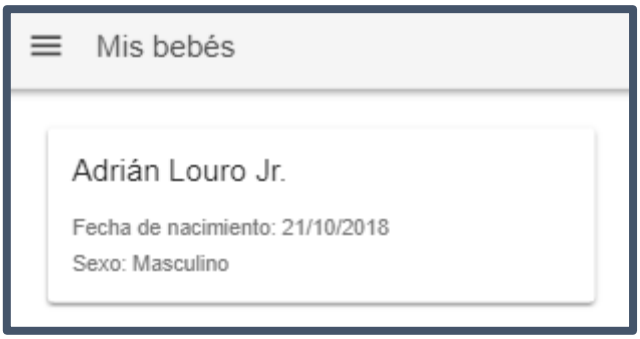

Ilustración 42: Listado de bebés asociados a un tutor

Añadir sesión

Crear una sesión asociada a una tarea. Se introducen los datos de la sesión y se presiona el botón de salvado para guardar los cambios. Una vez creada la sesión, se pregunta al usuario si desea subir algún vídeo o imagen y, si responde afirmativamente, se le redirige a la pestaña "Multimedia" de la página de la sesión recién creada.

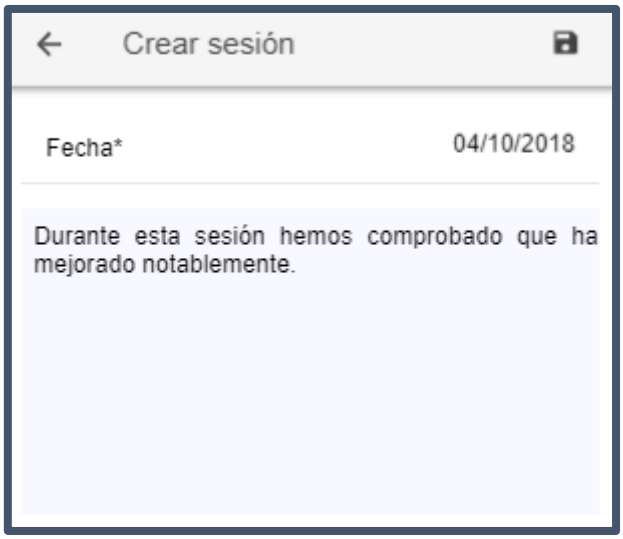

Ilustración 43: Formulario para la creación de sesiones

Opinión

Opinión asociada a un ejercicio. Para crear o editar una opinión, se introduce el texto deseado y se presiona el botón de salvado. Para eliminarla, se presiona el botón de borrado.

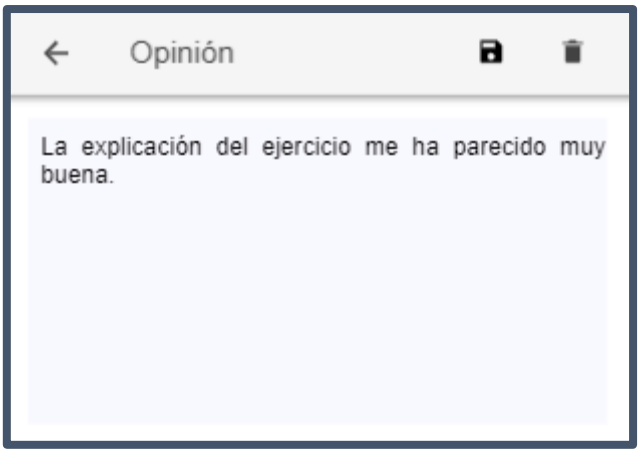

Ilustración 44: Formulario para escribir una opinión acerca de un ejercicio

Fisioterapeuta

Mis pacientes

Se muestra un listado con los pacientes asociados. Se puede filtrar el listado utilizando la barra de búsqueda. Presionando el botón flotante se accede a la vista de creación/asociación de pacientes. Presionando un elemento de la lista se accede al perfil del paciente.

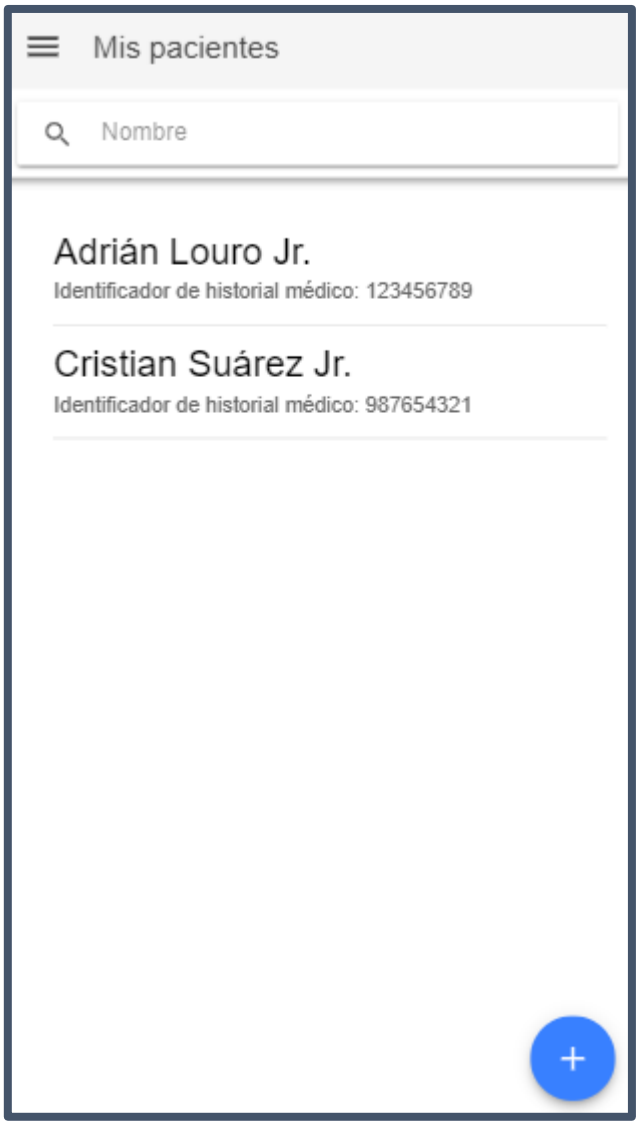

Ilustración 45: Listado de pacientes asociados a un fisioterapeuta

Crear/asociar paciente

Se introduce el identificador del historial médico del paciente. Si ya estaba dado de alta en la aplicación, se muestra su información en el formulario. Si no está dado de alta, se introduce manualmente el resto de los datos. Para guardar los cambios, se presiona el botón de salvado.

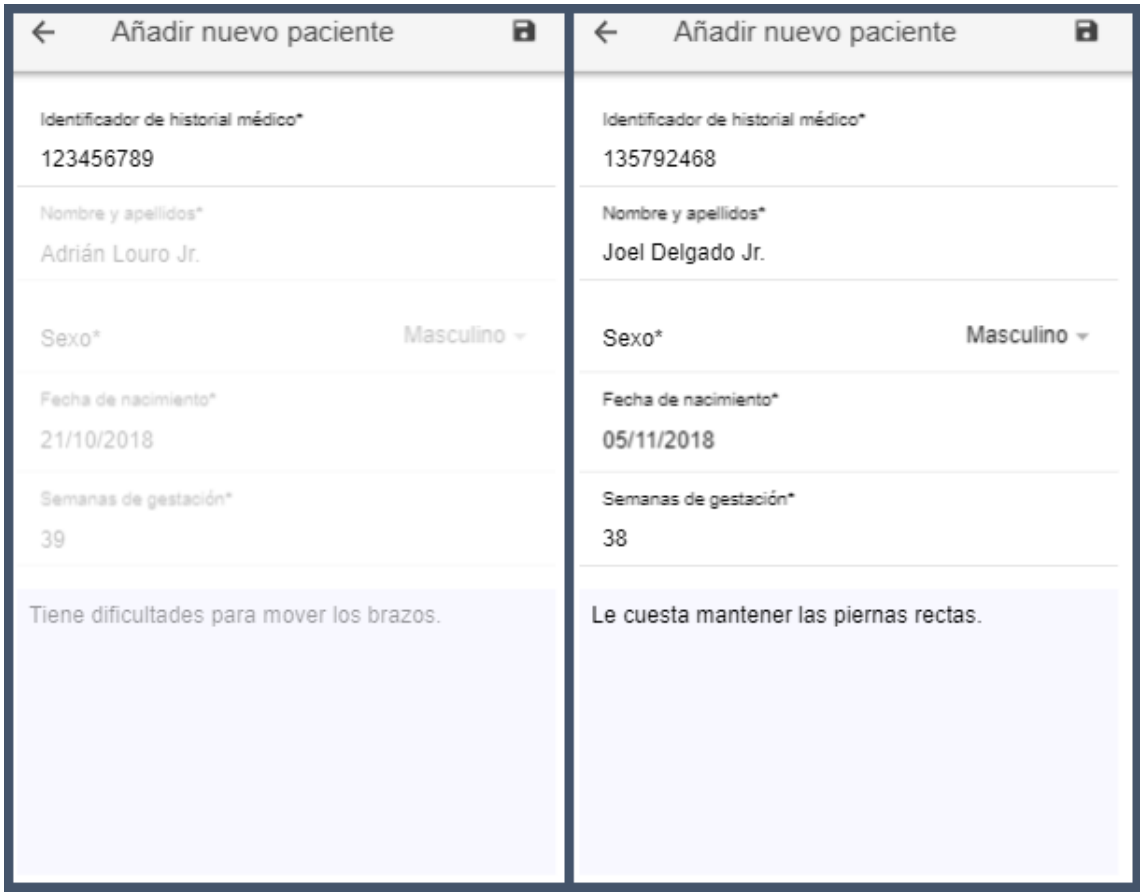

Ilustración 46: Vista para asociar pacientes existentes o dar de alta nuevos pacientes.

Ejercicios

Se muestra un listado con todos los ejercicios dados de alta en la base de datos. Se puede filtrar el listado utilizando la barra de búsqueda y el desplegable de categorías. Además, se podrá filtrar el listado de ejercicios según si han sido creados, o no, por el usuario. Presionando el botón flotante se accede a la vista de creación de ejercicios. Presionando un elemento de la lista se accede a la vista del ejercicio.

Ilustración 47: Listado de ejercicios existentes en el sistema

Añadir ejercicio

Se introducen los datos del ejercicio y se presiona el botón de salvado para guardar los cambios. Una vez creado el ejercicio, se pregunta al usuario si desea subir algún vídeo o imagen y, si responde afirmativamente, se le redirige a la pestaña “Multimedia” de la página del ejercicio recién creado.

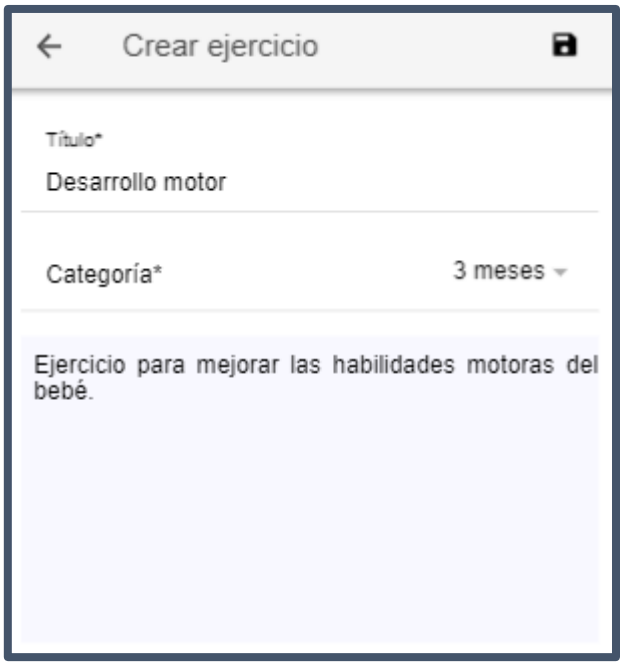

Ilustración 48: Formulario para la creación de un ejercicio

Añadir tarea

Crear una tarea asociada a un paciente. Se introducen los datos de la tarea y se presiona el botón de salvado para guardar los cambios.

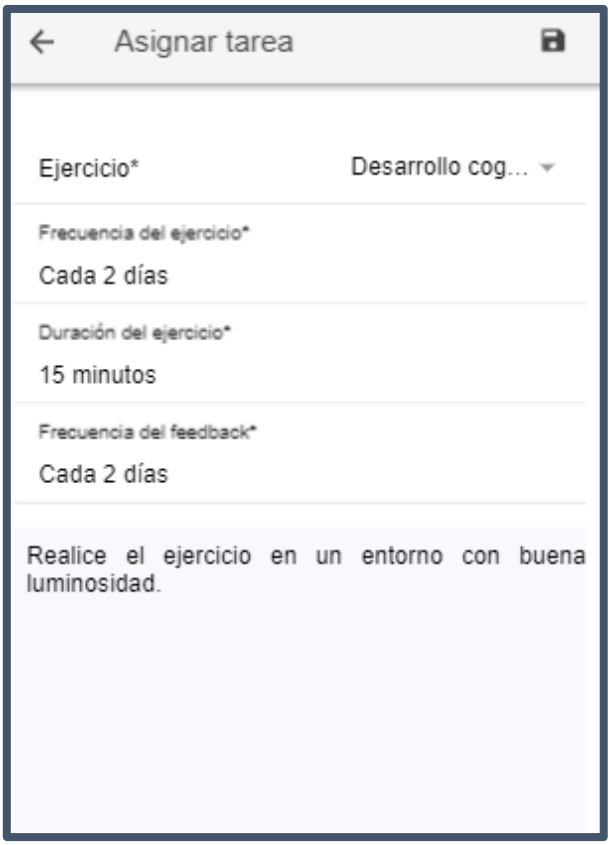

Ilustración 49: Formulario para la creación y asignación de una tarea a un paciente

Añadir tutor

Asociar un tutor a un paciente. Se introduce el DNI/NIF/NIE del tutor. Si dicho tutor se ha dado de alta en la aplicación, se mostrará y, al presionar en él, se solicitará confirmación para asociarlo al paciente.

Ilustración 50: Vista para asociar un tutor a un paciente

Ejercicio

Se muestran tres pestañas:

- Información: información del ejercicio. Si el ejercicio pertenece al fisioterapeuta, se puede editar presionando el botón de salvado y eliminar presionando el botón de borrado.

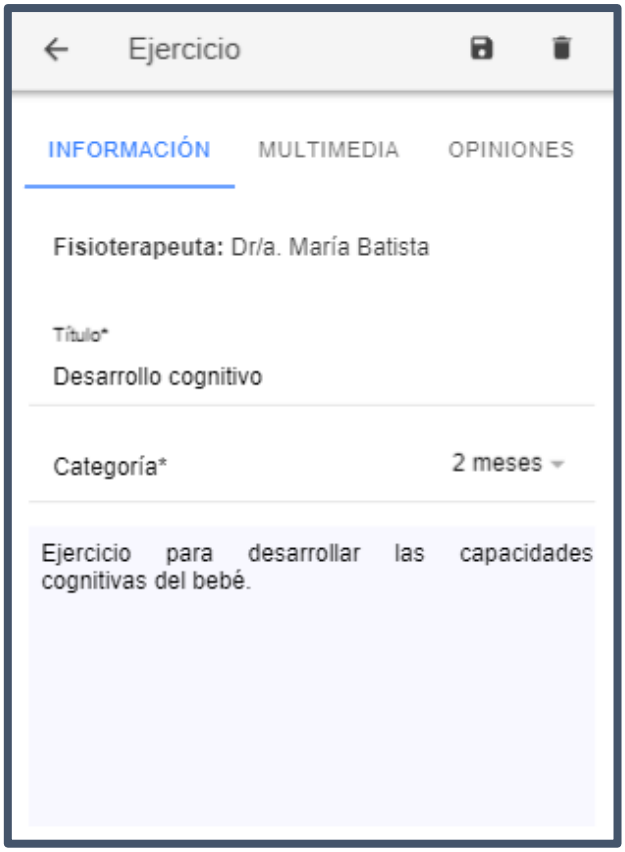

Ilustración 51: Vista de un ejercicio

- Multimedia: se muestra el vídeo y las imágenes asociadas al ejercicio. Si el ejercicio pertenece al fisioterapeuta, se puede subir y eliminar el vídeo y las imágenes asociadas al ejercicio. Presionando una imagen, se puede ver en pantalla completa.

Ilustración 52: Contenido multimedia de un ejercicio

- Opiniones: se muestra el listado de opiniones escritas por los tutores.

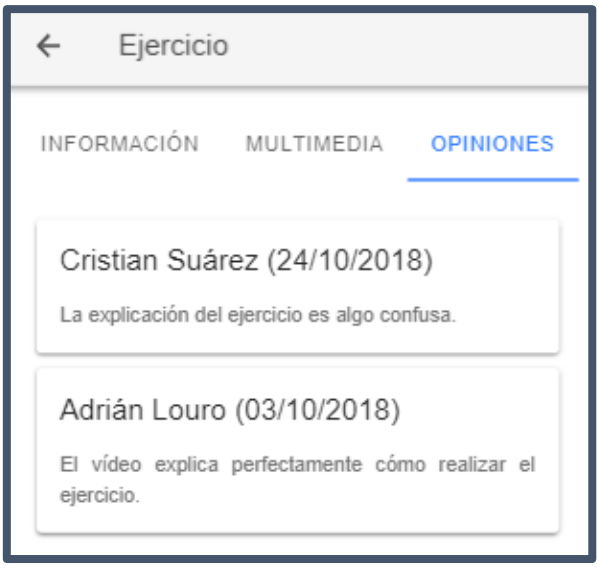

Ilustración 53: Opiniones de un ejercicio

Anexo II: Manual de instalación

A continuación, se explican los pasos a seguir para poder desplegar el sistema en un entorno de desarrollo.

Backend

1. Instalar un sistema de gestión de bases de datos MySQL en su versión 5.0 o superior
2. Ejecutar el script proporcionado en el *ANEXO III* para crear el esquema de la base de datos y un usuario que tendrá acceso al mismo
3. Descargar e instalar Microsoft Visual Studio 2017 para poder abrir y ejecutar la API REST
4. Configurar la URL en la cual se desplegará la API REST: *PrematureKidsAPI > Depurar > Dirección URL de la aplicación*
5. Establecer la cadena de conexión (parámetro *connectionString*) con la base de datos en el archivo *PrematureKidsAPI/appsettings.json*
6. Configurar el servicio de envío de emails (parámetro *Smtp*) en el archivo *PrematureKidsAPI/appsettings.json*
7. Establecer el directorio donde se generarán los archivos del servicio de log (parámetros *internalLogFile* y *fileName*) en el archivo *PrematureKidsAPI/nlog.config*
8. Establecer el directorio donde se encuentra la clave privada necesaria para poder generar el *token* de acceso a Firebase en el archivo *PrematureKidsAPI/Controllers/UsersController.cs* y ubicar la clave privada en dicho directorio
9. Ejecutar la API REST:
    a. Utilizando Microsoft Visual Studio, presionando *F5* o *Debug > Start Debugging*
    b. Utilizando el símbolo del sistema para desarrolladores de Visual Studio 2017, ejecutando el siguiente comando en la raíz del proyecto:

```
dotnet run --urls [URL de acceso]
```

Frontend

1. Descargar e instalar la última versión LTS de Node.JS
2. Instalar Ionic CLI y Cordova CLI ejecutando los comandos:

```
npm i ionic@latest
npm i -g cordova
```

3. Descargar las dependencias del proyecto ejecutando el siguiente comando en la raíz del proyecto:

```
npm install
```

4. Configurar la URL de la API REST (parámetro *apiUrl*) en el archivo *src/app/services/http.service.ts*
5. Para probar la aplicación:
    - En un navegador web, ejecutar el siguiente comando en la raíz del proyecto:

```
ionic serve
```

- En un dispositivo móvil Android:
    i. Activar la depuración USB en el dispositivo
    ii. Conectar el dispositivo al equipo mediante el puerto USB
    iii. Ejecutar el siguiente comando en la raíz del proyecto:

```
ionic cordova run android --debug
```

- En un dispositivo móvil iOS:
    i. Realizar la configuración establecida en el apartado "*iOS Devices*" de la documentación oficial proporcionada por el equipo de Ionic (https://ionicframework.com/docs/intro/deploying/)
    ii. Ejecutar el siguiente comando en la raíz del proyecto:

```
ionic cordova run ios --device
```

Anexo III: Script para la base de datos

```
DROP DATABASE IF EXISTS prematurekids;

CREATE DATABASE prematurekids
      DEFAULT CHARACTER SET utf8
      DEFAULT COLLATE utf8_general_ci;

CREATE USER IF NOT EXISTS prematurekids;

GRANT ALL ON prematurekids.* to 'prematurekids'@'localhost' IDENTIFIED
BY 'prematurekids';

CREATE TABLE prematurekids.user(
      id VARCHAR(255),
      email VARCHAR(255) NOT NULL UNIQUE,
      password VARCHAR(255) NOT NULL,
      role VARCHAR(255) NOT NULL,
      blocked TINYINT(1) NOT NULL DEFAULT 0,
      PRIMARY KEY(id)
);

CREATE TABLE prematurekids.administrator(
      id VARCHAR(255),
      name VARCHAR(255) NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(id) REFERENCES user(id) ON DELETE CASCADE
);

CREATE TABLE prematurekids.parent(
      id VARCHAR(255),
      name VARCHAR(255) NOT NULL,
      id_number VARCHAR(255) NOT NULL UNIQUE,
      telephone VARCHAR(255) NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(id) REFERENCES user(id) ON DELETE CASCADE
);

CREATE TABLE prematurekids.doctor(
      id VARCHAR(255),
      name VARCHAR(255) NOT NULL,
      board_number VARCHAR(255) NOT NULL UNIQUE,
      telephone VARCHAR(255) NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(id) REFERENCES user(id) ON DELETE CASCADE
);

CREATE TABLE prematurekids.child(
      id VARCHAR(255),
      medical_history_id VARCHAR(255) NOT NULL UNIQUE,
      name VARCHAR(255) NOT NULL,
      date_of_birth DATE NOT NULL,
```

```
      gender VARCHAR(255) NOT NULL,
      weeks_of_pregnancy INT NOT NULL,
      medical_history TEXT NOT NULL,
      PRIMARY KEY(id)
);

CREATE TABLE prematurekids.child_parent(
      child VARCHAR(255) NOT NULL,
      parent VARCHAR(255) NOT NULL,
      PRIMARY KEY(child, parent),
      FOREIGN KEY(child) REFERENCES child(id) ON DELETE CASCADE,
      FOREIGN KEY(parent) REFERENCES parent(id) ON DELETE CASCADE
);

CREATE TABLE prematurekids.child_doctor(
      child VARCHAR(255) NOT NULL,
      doctor VARCHAR(255) NOT NULL,
      PRIMARY KEY(child, doctor),
      FOREIGN KEY(child) REFERENCES child(id) ON DELETE CASCADE,
      FOREIGN KEY(doctor) REFERENCES doctor(id) ON DELETE CASCADE
);

CREATE TABLE prematurekids.category(
      id VARCHAR(255),
      name VARCHAR(255) NOT NULL UNIQUE,
      PRIMARY KEY(id)
);

CREATE TABLE prematurekids.exercise(
      id VARCHAR(255),
      title VARCHAR(255) NOT NULL,
      description TEXT NOT NULL,
      doctor VARCHAR(255) NOT NULL,
      category VARCHAR(255) NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(doctor) REFERENCES doctor(id) ON DELETE CASCADE,
      FOREIGN KEY(category) REFERENCES category(id) ON DELETE CASCADE
);

CREATE TABLE prematurekids.exercise_attachment(
      id VARCHAR(255),
      name VARCHAR(255) NOT NULL,
      type VARCHAR(255) NOT NULL,
      exercise VARCHAR(255) NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(exercise) REFERENCES exercise(id) ON DELETE CASCADE
);

CREATE TABLE prematurekids.opinion(
      id VARCHAR(255),
      text TEXT NOT NULL,
      date DATE NOT NULL,
```

```
      parent VARCHAR(255) NOT NULL,
      exercise VARCHAR(255) NOT NULL,
      PRIMARY KEY(id),
      UNIQUE(parent,exercise),
      FOREIGN KEY(parent) REFERENCES parent(id) ON DELETE CASCADE,
      FOREIGN KEY(exercise) REFERENCES exercise(id) ON DELETE CASCADE
);

CREATE TABLE prematurekids.assignment(
      id VARCHAR(255),
      doctor VARCHAR(255) NOT NULL,
      exercise VARCHAR(255) NOT NULL,
      child VARCHAR(255) NOT NULL,
      date DATE NOT NULL,
      notes TEXT NOT NULL,
      exercise_frequency VARCHAR(255) NOT NULL,
      exercise_duration VARCHAR(255) NOT NULL,
      feedback_frequency VARCHAR(255) NOT NULL,
      state VARCHAR(255) NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(doctor) REFERENCES doctor(id) ON DELETE CASCADE,
      FOREIGN KEY(exercise) REFERENCES exercise(id) ON DELETE CASCADE,
      FOREIGN KEY(child) REFERENCES child(id) ON DELETE CASCADE
);

CREATE TABLE prematurekids.session(
      id VARCHAR(255),
      assignment VARCHAR(255) NOT NULL,
      parent VARCHAR(255) NOT NULL,
      date DATE NOT NULL,
      parent_notes TEXT NOT NULL,
      doctor_notes TEXT,
      PRIMARY KEY(id),
      FOREIGN KEY(parent) REFERENCES parent(id) ON DELETE CASCADE,
      FOREIGN KEY(assignment) REFERENCES assignment(id) ON DELETE
CASCADE
);

CREATE TABLE prematurekids.session_attachment(
      id VARCHAR(255),
      name VARCHAR(255) NOT NULL,
      type VARCHAR(255) NOT NULL,
      session VARCHAR(255) NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(session) REFERENCES session(id) ON DELETE CASCADE
);
```