UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

Desarrollo de una App informática para favorecer la adherencia al tratamiento de fisioterapia y farmacología de los pacientes con enfermedad de Párkinson. TFG interdisciplinar.

TRABAJO FIN DE GRADO

Julio - 2021

AUTOR: Tomas Neverdauskas

DIRECTOR: Manuel Quesada Martínez

RESUMEN

La enfermedad del Parkinson (EP) es un trastorno neurodegenerativo progresivo crónico del sistema nervioso central que ocurre principalmente en adultos mayores, caracterizado por manifestaciones motoras como no motoras. Contribuir a la mejora y el seguimiento de los tratamientos de pacientes con EP mejoraría su adherencia. Este trabajo pretende contribuir a este objetivo mediante la asistencia, a través de una aplicación informática interactiva, que facilite el cálculo y seguimiento de planes personalizados para pacientes con EP que combinen, además, elementos interdisciplinares de fisioterapia (realización de ejercicios) y farmacia (ingesta de fármacos).

Se ha realizado un estudio de aplicaciones relacionadas sin encontrar ninguna aplicación integrada que incluya la combinación de tratamientos de fisioterapia y farmacia. Este estudio también ha permitido diseñar una interfaz observando e integrando las buenas prácticas observadas en aplicaciones relacionadas. La investigación del estado del arte de las tecnologías de desarrollo, nos ha permitido definir una arquitectura basada en React Native, Expo y Node.js usando principalmente el lenguaje de programación TypeScript para el desarrollo de una aplicación híbrida, que ha requerido el desarrollo de una base de datos relacional MySQL a la que se accederá a través de una API desarrollada con Flask y usando el lenguaje de programación Python.

El resultado obtenido ha sido un prototipo funcional de una aplicación móvil que permite al usuario introducir determinados sus datos médicos que serán utilizados para calcular y realizar el seguimiento de un tratamiento personalizado. Para facilitar el seguimiento del tratamiento por parte del usuario, e indirectamente mejorar la adherencia al tratamiento, se mostrará dinámicamente la información como un listado diario de ejercicios y fármacos con un horario recomendado para su realización o ingesta. El sistema se ha probado prioritariamente en un sistema Android testeando que se han cumplido la mayoría de los requisitos identificados en la fase de diseño. Se han identificado una serie líneas de trabajo futuro que podrían ser tenidas en cuentas en la siguiente fase del proyecto, en la que se desea comercializar la aplicación aquí desarrollada.

Palabras clave: enfermedad de Parkinson (EP), APP, parkinson, react native, expo, interdiscplinar, TFG, Flask, javascript, typescript, API, fetch, axios, hook.

AGRADECIMIENTOS

En primer lugar, quiero agradecer profundamente a Manuel Quesada Martínez, que aparte de ser un gran profesor universitario ha sido un gran tutor de este Trabajo de Fin de Grado, siendo un ejemplo para seguir en cuanto a profesionalidad y vocación. Quiero agradecerle por la paciencia, su gran experiencia, buenos consejos y momentos que hemos tenido en todas las reuniones que hemos ido realizando a lo largo del proyecto.

En segundo lugar, quiero agradecer a Kenia Ferrer Pomer por lo fácil que ha sido trabajar con ella, lo implicada y profesional que ha sido durante todo el proceso, sin dudar en contestar cualquier duda que me haya ido surgiendo, habiendo ella finalizado su carrera y a Adrián Muñoz Cascajero por su participación y buenos consejos.

Finalmente quiero agradecer a Mari Carmen Lillo y Enrique Barrajón por implicarse tanto en este proyecto y hacer posible la participación de este grupo en el acto de presentación y entrega de premios de la primera edición del proyecto TFG interdisciplinar y gracias a ello, obtener el segundo puesto en dicho evento.

TFG INTERDISCIPLINAR

Este TFG ha sido realizado como parte de la primera promoción del programa de Innovación Docente: Trabajo Fin de Grado Interdisciplinar (TFGi). En el programa se ofrece la posibilidad de que estudiantes de diversas titulaciones se agrupen y de manera colaborativa aborden la resolución integral de un problema o reto real.

En este TFGi han participado tres estudiantes UMH de las siguientes titulaciones: Grado en Ingeniería Informática en Tecnologías de la Información, Grado en Fisioterapia y Grado en Farmacia. El reto que resolver surgió durante las Jornadas de Formación organizadas como parte del programa TFGi, donde se identificó la necesidad de planes personalizados en el tratamiento de pacientes con Enfermedad de Parkinson (EP), y contribuir a su seguimiento con una aplicación informática interactiva que favorezca la adherencia de estos tratamientos.

Durante el desarrollo de este proyecto, los tres alumnos han trabajado conjuntamente en una parte común, dirigiendo posteriormente sus esfuerzos a desarrollar el contenido específico y relacionado con su titulación. Es por ello, que en este documento aparecerán puntualmente y de forma resumida referencias y aspectos de los contenidos trabajados junto al resto de las titulaciones, con el objetivo de plasmar, el proceso interdisciplinar seguido por todo el equipo.

Más información sobre el programa TFGi en:
http://tfgi.edu.umh.es/

ÍNDICE DE CONTENIDO

1. INTRODUCCIÓN

La enfermedad del Parkinson (EP) es un trastorno neurodegenerativo progresivo crónico del sistema nervioso central que ocurre principalmente en adultos mayores, caracterizado tanto por manifestaciones motoras como no motoras. La EP es la segunda enfermedad neurodegenerativa más frecuente seguida del Alzheimer [1]. Aunque la causa principal de la EP sigue siendo desconocida, se han descrito algunos factores que podrían influir en la aparición de la EP, como pueden ser el estrés y las toxinas ambientales [2], las drogas [3] y algunas mutaciones genéticas [4].

Existen diferentes tipos de EP, por ejemplo, la EP idiopática forma parte de un gran grupo de enfermedades llamado Parkinsonismos [5], aunque el pronóstico y tratamiento de las diferentes variantes de la enfermedad puede variar, la mayoría de ellas se cursan con sintomatología similar a la idiopática [6]. Actualmente, la incidencia de esta enfermedad se encuentra principalmente relacionada con la variante del sexo, ya que en hombres es 1.5 veces más frecuente que en mujeres [7]. Por otra parte, es un 1.6 más probable de padecerla en personas mayores de 65 años. Se estima que la prevalencia de tener EP va a aumentar bastante en los próximos 20 años, llegándose a afirmar que el 1% de la población americana a partir de los 60 años la sufrirá [1].

1.1 Tratamiento y sintomatología de pacientes con EP

Contribuir con soluciones que permitan ayudar a los pacientes a mejorar su día a día es un aspecto muy importante. En 2015 la ONU aprobó la Agenda 2030 sobre el Desarrollo Sostenible en la que se incluyen 17 objetivos [8]. Concretamente, el Objetivo de Desarrollo Sostenible (ODS) 3 (“Salud y Bienestar”) promueve que es esencial para el desarrollo sostenible desarrollar iniciativas que permitan garantizar una vida sana y promover el bienestar en todas las edades.

Contribuir a la mejora y el seguimiento de los tratamientos de pacientes con EP permitiría contribuir en gran medida al ODS 3. En este trabajo centramos nuestra atención en el análisis y seguimiento de tratamientos que combinen de manera efectiva principios de fisioterapia y farmacología. Actualmente, la mayoría de los tratamientos para pacientes que sufren la EP son de tipo sintomático, y están dirigidos a equilibrar en lo posible la falta de ciertos neurotransmisores como la dopamina [9].

Aunque existen diferentes tipos de tratamiento podemos clasificarlos en los siguientes grupos: tratamiento conservador y tratamiento quirúrgico. En este trabajo vamos a centrar nuestra atención en los tratamientos del primer grupo:

- **Tratamiento conservador:** este tipo de tratamiento combina las intervenciones farmacológicas y fisioterapéuticas. Mientras que las intervenciones farmacológicas implican la toma de distintos tipos de fármacos, las intervenciones fisioterapéuticas requieren de la realización de diferentes tipos de ejercicios.

En cuanto a las intervenciones farmacológicas, se pueden categorizar los fármacos en dos grandes grupos, los fármacos para el tratamiento de síntomas motores y otros para síntomas no motores. La mayoría de pacientes toman fármacos de los dos tipos, por esto se tiene que controlar su dosificación y combinación para que no se alteren sus efectos [10] [11]. En cuanto a las intervenciones fisioterapéuticas, las intervenciones con mayor evidencia científica reportadas incluyen distintos tipos de ejercicio como ejercicios de equilibrio, estiramiento muscular, entrenamientos de fuerza progresivos, ejercicios de resistencia progresiva, ejercicios en cinta, ejercicios de señales auditivas y visuales, ejercicio aeróbico, ejercicios de doble tarea, ejercicios con videojuegos, hidroterapia, pilates, taichí, danza o tratamiento de puntos gatillos [12].

Uno de los principales objetivos de los tratamientos conservadores es mejorar la vida del paciente. Por ello, para el desarrollo de soluciones efectivas, es interesante focalizar la atención en los distintos síntomas de esta enfermedad. Los principales síntomas de la EP pueden ser divididos en: síntomas motores y síntomas no motores. La evidencia existente sobre las intervenciones del tratamiento está encaminadas principalmente a la mejora de ambos tipos.

- **Síntomas motores:** los principales son la bradicinesia, el temblor y la rigidez asociada a dolor. Este último síntoma suele ir acompañado de deformidades posturales como la flexión de cuello, tronco, codos y rodillas. El control postural dinámico se ve alterado en las primeras fases de la enfermedad mientras que estático se suele alterar en las últimas fases, así como los reflejos posturales. Por tanto, los principales signos motores para diagnosticar EP son: temblor, rigidez, inestabilidad postural y bradicinesia [13][12].

- **Síntomas no motores:** los más frecuentes son la disfunción olfatoria, el trastorno del comportamiento del sueño REM, el estreñimiento, la depresión, la demencia, la incontinencia urinaria y la disfunción sexual [13][12].

Además, otra de las dificultades añadidas de esta enfermedad está relacionada con las fluctuaciones motoras que sufren este tipo de pacientes a lo largo del día. Las fluctuaciones motoras son variaciones en la sintomatología del paciente con EP, dependientes de la medicación, del estado emocional y de la personalidad. Estas consisten en dos estados: el estado ON en el que la sintomatología mejora y el estado OFF en el que la sintomatología empeora [14]. En la Figura 1 (extraída de [15]) se muestra un ejemplo de las fluctuaciones que podría sufrir un paciente a lo largo un día.

Figura 1. Ejemplo de fluctuaciones que puede sufrir un paciente de EP a lo largo del día y su relación con las fases on y off.

1.2 Aplicaciones informáticas para el seguimiento de tratamientos

Una aplicación informática es un programa que está incorporado en un dispositivo TIC, el cual, permite la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones, en forma de voz, imágenes y datos. Con esas características puedes ofrecer al usuario información programada relevante respecto a sus necesidades e interacciones previas con el fin de mejorar el estado del afectado [16]. Las aplicaciones informáticas aparecieron de la mano de ordenadores fijos, que poco a poco han ido expandiéndose a otros ámbitos como los ordenadores portátiles, teléfonos inteligentes o *smartphones*, tabletas, y recientemente, incluso en tecnologías emergentes como los relojes inteligentes o wearables [17].

Frecuentemente a los pacientes con EP se les aconseja un programa domiciliario. Este consiste en que el paciente, con o sin ayuda de un cuidador, llevará a cabo unas recomendaciones de tratamiento coordinado y asesorado por profesionales sanitarios de atención primaria [18]. Además, existen evidencias de que, si se siguen las pautas marcadas por un tratamiento conservador de manera óptima, el efecto del tratamiento aumenta haciendo que la condición física de estos tipos de pacientes mejore. Además, puede disminuir la manifestación de la sintomatología y ayudar a que no se produzcan otras enfermedades con las que conviven las personas con EP [19].

Sin embargo, en el tratamiento domiciliario hay un serio problema y es la falta de adherencia al mismo. Esta provoca un descenso en la calidad de vida de estas personas y un empeoramiento de los síntomas motores y complicaciones motoras [20]. Para facilitar esos problemas se plantea el uso de dispositivos y aplicaciones que asistan tanto a los pacientes de EP, como el personal sanitario que lo supervisa o sus cuidadores. Por ejemplo, en [9] se describen los beneficios de utilizar un dispositivo innovador que permite realizar un seguimiento de los síntomas de personas con EP. Otro ejemplo sería la aplicación *mHealth* [21] cuyo principal objetivo es ofrecer a los paciente el monitoreo y seguimiento de su enfermedad.

En general, las aplicaciones con más impacto serán aquellas que posibiliten recopilar información, recibir diagnóstico y tratamiento, así como utilizar los datos gestionados para favorecer a la prevención. Por ello, el mayor impacto para el paciente tendrá relación con el asesoramiento y el seguimiento después de la visita inicial al médico. Además, se estima que la utilización de aplicaciones móviles podría mejorar la eficiencia de la atención al paciente y minimizar hasta el 30% del tiempo empleado en acceder a la información y analizarla, con un ahorro económico del 15% de los costes de utilización de la atención sanitaria mediante el seguimiento a distancia a través de aplicaciones móviles [22].

Creemos que el uso de este tipo de aplicaciones permitiría reducir la mortalidad prematura debida a enfermedades no transmisibles gracias a la prevención y tratamiento a aquellas personas que puedan utilizar dicha aplicación (ODS 3). Además, una aplicación que permita mejorar la adherencia a un tratamiento contribuirá también al ODS 12 (“Producción y consumo responsable”) ya que, si se incluyen consejos sobre el tratamiento y la posible mejora en la adherencia al mismo, se reduce la mala gestión de

los productos químicos y la generación de desechos químicos mediante una mejor dosificación y empleo de los fármacos, y un menor uso de los transportes [23].

2. OBJETIVOS Y METODOLOGÍA

En esta sección se recogen los principales objetivos y metodología utilizada durante la realización de este trabajo.

2.1 Objetivos generales y específicos

En la página 4 se ha comentado la integración de este TFG como parte del programa de innovación docente de TFG interdisciplinares, y como el reto a resolver surgió durante las Jornadas de Formación organizadas como parte de dicho programa. Dicho reto permitió definir de forma global un **objetivo interdisciplinar común**: *analizar y planificar los tratamientos de farmacología y fisioterapia, y unificarlos de manera que se favorezca su adherencia en personas con EP a través de su seguimiento con una aplicación informática interactiva e integrada*.

En el contexto del Grado de Ingeniería Informática en Tecnologías de la Información en el que se desarrolla el trabajo descrito en este documento, se centrarán los esfuerzos en lograr el siguiente **objetivo general**:

- Desarrollar una aplicación informática partiendo de los requisitos establecidos conjuntamente por los tres participantes del proyecto de tal manera que puedan coexistir sin contradicciones y la información sea clara y precisa mediante la implementación de algoritmos, funciones y condiciones.

A su vez, se definen una serie de **objetivos específicos** concretos y relacionados con la elaboración de la aplicación, que indirectamente nos permitiría alcanzar los objetivos de más alto nivel anteriormente descritos:

1. **Liderar la parte IT del equipo:** interactuar con los compañeros de Fisioterapia y Farmacia ayudándolos a relacionar los conocimientos teóricos/prácticos de sus disciplinas con el desarrollo software. Realizar durante el proceso la extracción de requisitos y el diseño del software que permita su posterior implementación.
2. **Análisis del estado del arte y tecnologías relacionadas:** analizar el estado actual de la cuestión para identificar aplicaciones relacionadas y la tecnología de desarrollo y *Frameworks* más apropiados para implementar la aplicación aquí requerida.

3. **Implementación de la App (preferiblemente para dispositivos móviles):** implementar una versión funcional y accesible utilizando una metodología de desarrollo software que refleje en la mayor medida posible todas las necesidades y requisitos extraídos e identificados en los puntos 1 y 2. Esta versión deberá ser extensible y tener en cuenta el perfil (edad, sexo, conocimientos TICS…) de los usuarios favoreciendo su usabilidad.

Finalmente me gustaría destacar que la elección de este TFG también fue motivada por una serie de **objetivos personales** que enumero a continuación:

- **Similitud con un proyecto real:** participar en un equipo que requiera de interacción entre diferentes perfiles, acercando el proceso de desarrollo e investigación de este trabajo a un proyecto real.
- **Poner en práctica contenidos aprendidos durante la carrera:** poner en práctica lo estudiado en asignaturas sobre desarrollo de aplicaciones móviles. Afianzar estos conocimientos aplicándolos en un proyecto real permitirá profundizar y seguir aprendiendo, dando lugar a su posible aprovechamiento en un futuro. Además, el desarrollo desde cero me permitirá tener libertad para utilizar tecnologías actuales y novedosas.
- **Ventajas de un TFGi:** aprovechar las ventajas que ofrece un TFG interdisciplinar, presentando el proyecto en un concurso con posibilidad de premios y oportunidades de continuar el proyecto a nivel empresarial.

2.2 Metodología

La realización de este TFG ha requerido de un trabajo común e inicial conjunto con los compañeros de las otras dos titulaciones. En la Figura 2 se muestra un resumen de la metodología grupal que se ha seguido para realizar la parte común del proyecto. Como se puede observar en dicha figura, en las fases tempranas se ha requerido de una mayor comunicación entre los tres estudiantes por medio de video llamadas en las cuales el procedimiento consistía en realizar propuestas, debatirlas y decidir si son válidas o no. Debido a la dificultad de realizar reuniones presenciales, en ocasiones se han extraído requisitos puntuales mediante conversaciones de mensajería instantánea.

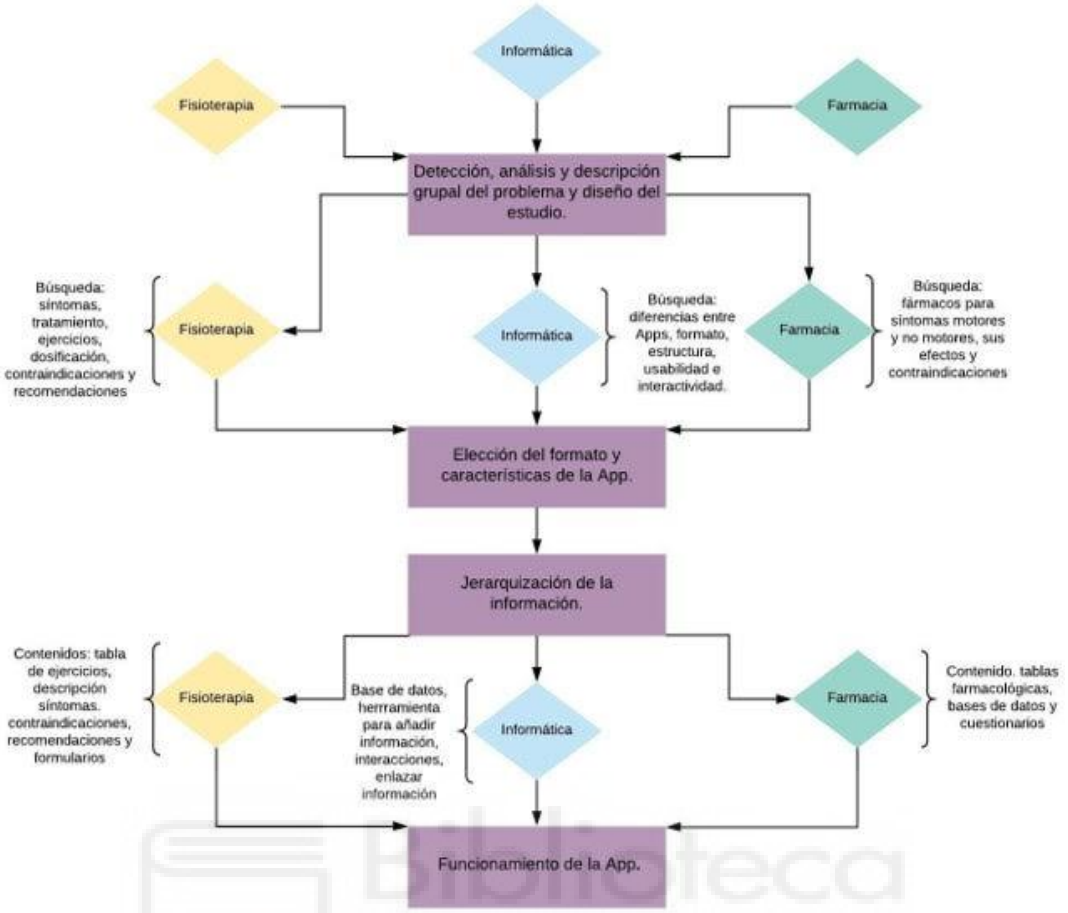


Figura 2. Resumen de la interacción entre las diferentes titulaciones para la realización de la parte común.

El punto de partida de este trabajo parte de la realización por parte del alumnado de fisioterapia y farmacia de borradores que elaboraron de forma individual describiendo los aspectos para tener en cuenta en estas disciplinas. Estos documentos fueron refinados posteriormente y en conjunto a través de las diferentes reuniones organizadas y utilizados como fuente de extracción de requisitos. Prácticamente en todas las reuniones se debatían y se tomaban decisiones respecto al diseño tanto funcional como visual de la aplicación. Merece la pena destacar que los distintos avances fueron plasmándose en documentos en la nube de forma estructurada, semiestructurada o no estructurada, y se categorizaron según la disciplina con la que estaban más relacionados. Por ejemplo, en la Figura 3, Figura 4, Figura 5 y Figura 6 se muestran algunos ejemplos representativos con el objetivo de que sirvan como referencia para mostrar la evolución de los requisitos y la aplicación resultado que serán descritos en la sección 4.

La Figura 3 muestra la descripción de uno de los ejercicios propuestos por la compañera de fisioterapia que deberán posteriormente mostrarse de una manera atractiva para el usuario que utilice la aplicación; en este ejemplo se puede observar que la información no se encuentra estructurada, por ejemplo, aparecen datos sobre la frecuencia del ejercicio en texto plano que nosotros deberemos representar de forma apropiada en nuestro modelo. Una vez estructurados los datos se utilizarán para implementar el algoritmo que calcula un plan personalizado del paciente de un tratamiento conservador.

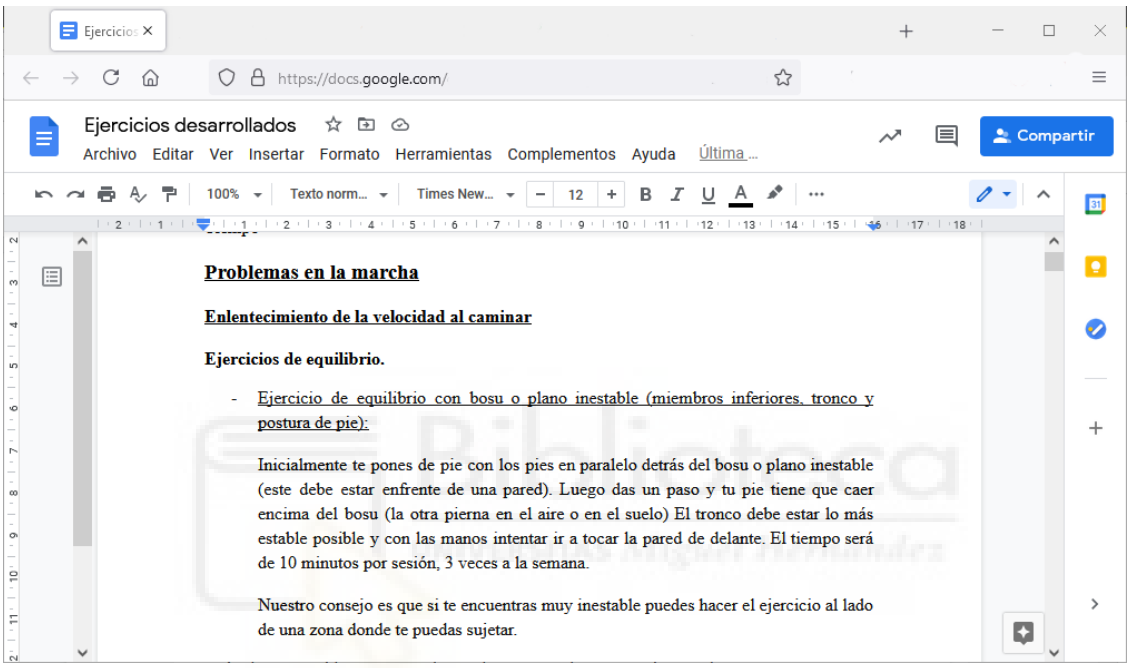


Figura 3. Ejemplo de documento compartido donde se ha codificado información estática que se deberá incluir como parte de la aplicación.

A su vez, en la Figura 4 se muestra como ejemplo una de las preguntas que debemos codificar para registrar información del paciente que también deberá tenerse en cuenta para definir su plan personalizado. La pregunta mostrada en este ejemplo está relacionada con el registro de alergias por parte del paciente. Como se puede observar, en las notas se describen una serie de funcionalidades que deben tenerse en cuenta a la hora de implementar el formulario. Desde el punto de vista de nuestra aplicación, se deberá asistir al usuario en el reconocimiento automático de los fármacos y su relación con principios activos. La información relacionada con estos elementos se nos ha proporcionado de forma semiestructurada por medio de ficheros con extensión .csv (ver Figura 5).

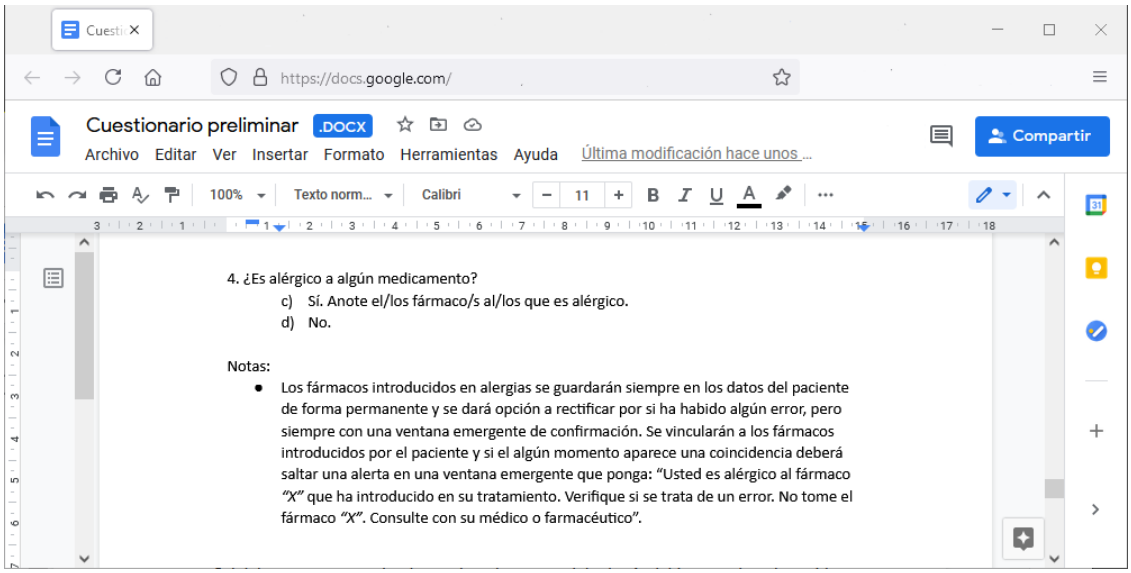
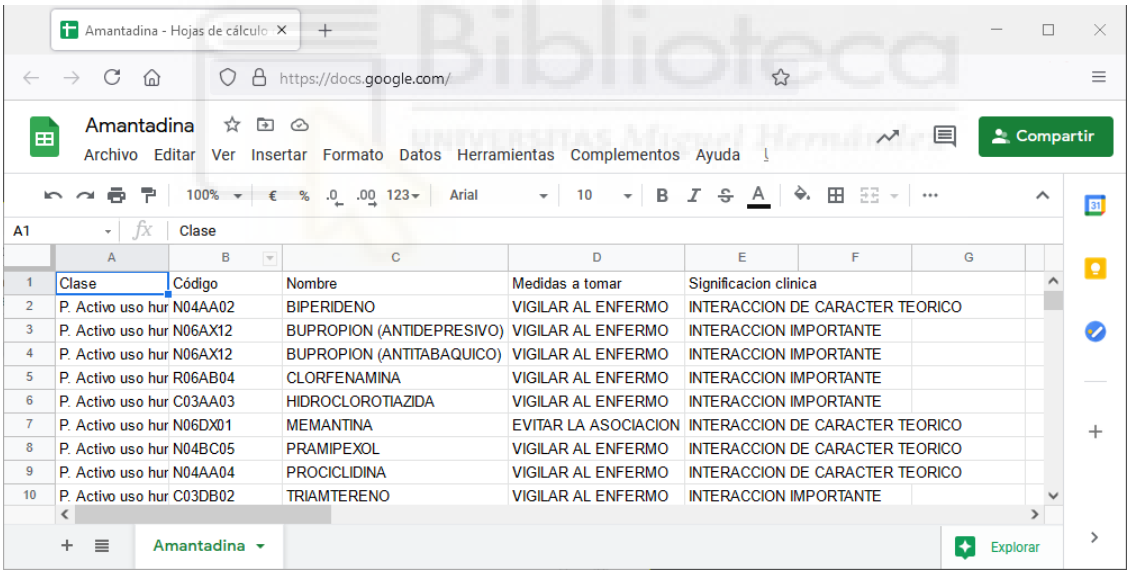


Figura 4. Ejemplo de la descripción de un formulario y las diferentes dependencias según las opciones elegidas.



	Clase	Código	Nombre	Medidas a tomar	Significacion clinica
2	P. Activo uso hur	N04AA02	BIPERIDENO	VIGILAR AL ENFERMO	INTERACCION DE CARACTER TEORICO
3	P. Activo uso hur	N06AX12	BUPROPION (ANTIDEPRESIVO)	VIGILAR AL ENFERMO	INTERACCION IMPORTANTE
4	P. Activo uso hur	N06AX12	BUPROPION (ANTITABAQUICO)	VIGILAR AL ENFERMO	INTERACCION IMPORTANTE
5	P. Activo uso hur	R06AB04	CLORFENAMINA	VIGILAR AL ENFERMO	INTERACCION IMPORTANTE
6	P. Activo uso hur	C03AA03	HIDROCLOROTIAZIDA	VIGILAR AL ENFERMO	INTERACCION IMPORTANTE
7	P. Activo uso hur	N06DX01	MEMANTINA	EVITAR LA ASOCIACION	INTERACCION DE CARACTER TEORICO
8	P. Activo uso hur	N04BC05	PRAMIPEXOL	VIGILAR AL ENFERMO	INTERACCION DE CARACTER TEORICO
9	P. Activo uso hur	N04AA04	PROCICLIDINA	VIGILAR AL ENFERMO	INTERACCION DE CARACTER TEORICO
10	P. Activo uso hur	C03DB02	TRIAMTERENO	VIGILAR AL ENFERMO	INTERACCION IMPORTANTE

Figura 5. Ejemplo de información semiestructurada sobre los distintos medicamentos, principios activos e interacciones.

Finalmente, en la Figura 6 se muestra como ejemplo el esbozo de un plan personalizado asociado al tratamiento conservador que deberá realizar un paciente. En este caso el

ejemplo ha sido representado a través de una hoja de cálculo y es objetivo de este proyecto automatizar este proceso a través de la aplicación desarrollada.

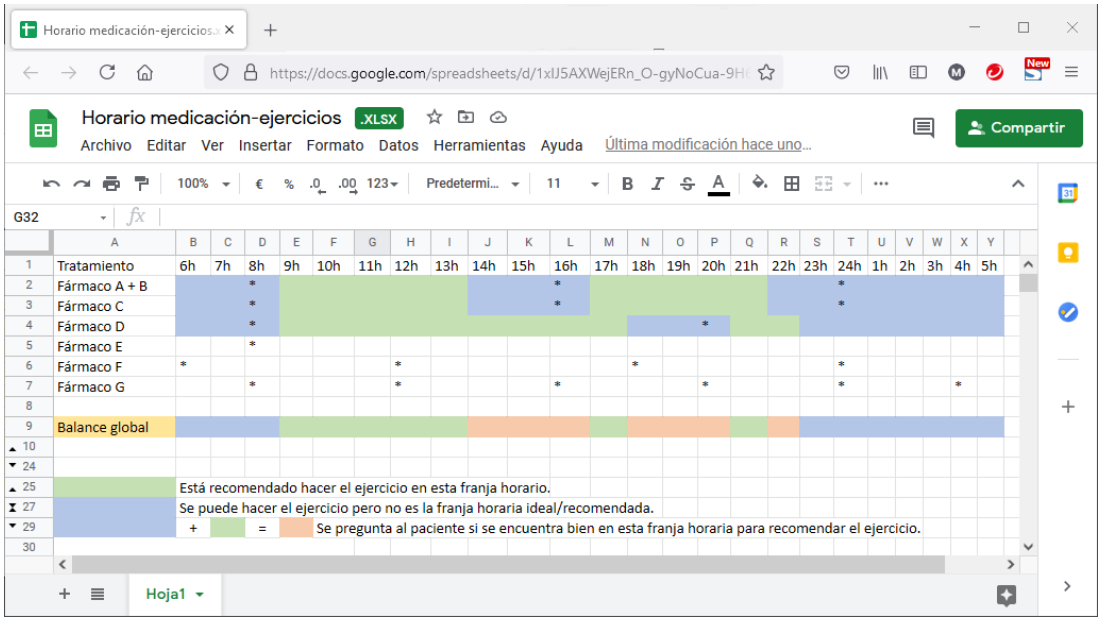


Figura 6. Ejemplo de un plan personalizado para distintos fármacos asociados a un tratamiento y su relación con los ejercicios.

2.2.1 Análisis de los aspectos a incluir en la aplicación

Como se comentará posteriormente en la sección 3.1, parte de los esfuerzos realizados durante las fases descritas anteriormente han requerido realizar una búsqueda de aplicaciones informáticas del ámbito de la salud y relacionadas con la adherencia a tratamientos. Se utilizó como punto de partida el análisis de aplicaciones relacionadas con la adherencia al tratamiento elaborado en [24], en el que el compañero de farmacia realizó una búsqueda sistemática de aplicaciones móviles relacionadas con la adherencia a un tratamiento. Entre las aplicaciones analizadas en este estudio, se encontraron 19 aplicaciones potencialmente útiles. Entre ellas, 7 tenían una utilidad dual (farmacéutica/fisioterapia), 5 sólo de utilidad en fisioterapia y 7 sólo de utilidad farmacéutica (se puede consultar el documento generado fruto de este análisis en [25]).

Esta selección de 19 aplicaciones se completó en este TFG con un segundo análisis centrado en aspectos técnicos de las aplicaciones, y completando la búsqueda a fecha de mayo de 2020. Se buscaron en las principales tiendas de aplicaciones usando como **palabras clave**: "*salud*", "*párkinson*", "*health*", "*healthcare*", "*tratamientos*" y "*medicación*". Dado el elevado número de aplicaciones obtenidas, se utilizó como filtro

los siguientes criterios de inclusión: popularidad y valoración, opiniones de usuarios y profesionales, número de descargas, funcionalidad, innovación, atractivo del diseño y presencia de imágenes en la descripción. Además, se incluyeron otros criterios subjetivos como:

1. Que puedan ser útiles en la EP desde el punto de vista de poner en conocimiento del paciente, cuidador o familiares información general sobre la enfermedad
2. Que sean útiles desde el punto de vista médico, fisioterápico y farmacológico
3. Que ayuden a manejar de forma más eficiente la terapéutica, tanto la farmacológica como la de fisioterapia
4. Que faciliten que la EP tenga el mínimo impacto posible en la calidad de vida del paciente y del cuidador.

Este análisis me permitió investigar aspectos de funcionalidad y las distintas interfaces ofrecidas por cada uno de estos productos que incorporar en nuestro desarrollo; además, he podido generar material que intercambiar con los compañeros de fisioterapia y farmacia en las reuniones anteriormente mencionadas. A partir de estos análisis y con el fin de mejorar la comunicación del resto del equipo, se generaron inicialmente esbozos de la parte visual de la aplicación en papel aglutinando los componentes seleccionados. Estos esbozos evolucionaron a lo largo del proyecto en dos prototipos funcionales, que permitieron a su vez mejorar la extracción de requisitos inicial y su refinamiento durante todo el proceso que se comentará en detalle durante las próximas secciones de este trabajo.

2.2.2 Elección de las tecnologías de desarrollo

Una vez identificados los principales requisitos de la aplicación, debí dirigir mi análisis en ampliar lo aprendido durante el grado en relación con las tecnologías para el desarrollo de aplicaciones móviles. Esta tarea es imprescindible para poder llevar a cabo la labor de desarrollo y es muy importante escoger las tecnologías adecuadas ya que, una elección incorrecta, podría comprometer el avance del proyecto, bien por incrementar el tiempo de desarrollo o por no permitir implementar parte de la funcionalidad requerida

Con el fin de escoger las tecnologías adecuadas, se realizó el análisis de los siguientes elementos que tendrán impacto en el desarrollo de nuestra herramienta:

1. Tipo de aplicación.
2. IDE (Entorno de Desarrollo Integrado).
3. SDK (Kit de Desarrollo de Software).
4. Framework o conjunto de librerías.

Durante el análisis, se analizaron y filtraron las diferentes opciones siguiendo los siguientes **criterios de inclusión y exclusión**:

- Criterios de inclusión:
    - Experiencia previa con el lenguaje de programación.
    - Posibilidad de desarrollo híbrido.
    - Tecnología no obsoleta.
    - Cantidad de extensiones y módulos.
    - Facilidad para enlazar las diferentes tecnologías para el desarrollo.
    - Popularidad de la tecnología.
- Criterios de exclusión:
    - Se descartó realizar una aplicación nativa debido al coste de esfuerzo, tiempo que requiere programa con distintas tecnologías específicas.
    - Desde un primer momento, y fruto de las conversaciones con los compañeros de fisioterapia y farmacia, se descartó el desarrollo web priorizando el desarrollo móvil porque, dada la edad del público objetivo y la necesidad de seguimiento del tratamiento en todo momento.

En relación con este último punto, hemos tomado como referencia el hecho de que en los últimos años ha sido una práctica común la incorporación de las nuevas tecnologías a través de aplicaciones móviles, lo que muestra cierta predilección de los usuarios en el uso de *smartphones* frente a otro tipo de dispositivos. En el ámbito de la salud, el uso de dispositivos móviles para el cuidado de la salud se denomina *mHealth*. En el artículo [26] se describe el potencial de este tipo de aplicaciones y de su eficacia, ya que este tipo de dispositivos presentan ventajas de movilidad y accesibilidad dado que es un dispositivo muy común, tamaño reducido, de uso diario y la mayoría de las personas suelen tenerlo presente, lo cual, favorece su uso.

2.2.3 Scrum: metodología ágil para la gestión del proyecto

Para el desarrollo de este proyecto se aplicó una metodología de desarrollo del software ágil denominada Scrum [27]. Es una metodología ideal para controlar y planificar proyectos con un gran volumen de cambios de última hora y requisitos cambiantes, por lo que consideramos que era ideal dadas las características de este proyecto interdisciplinar.

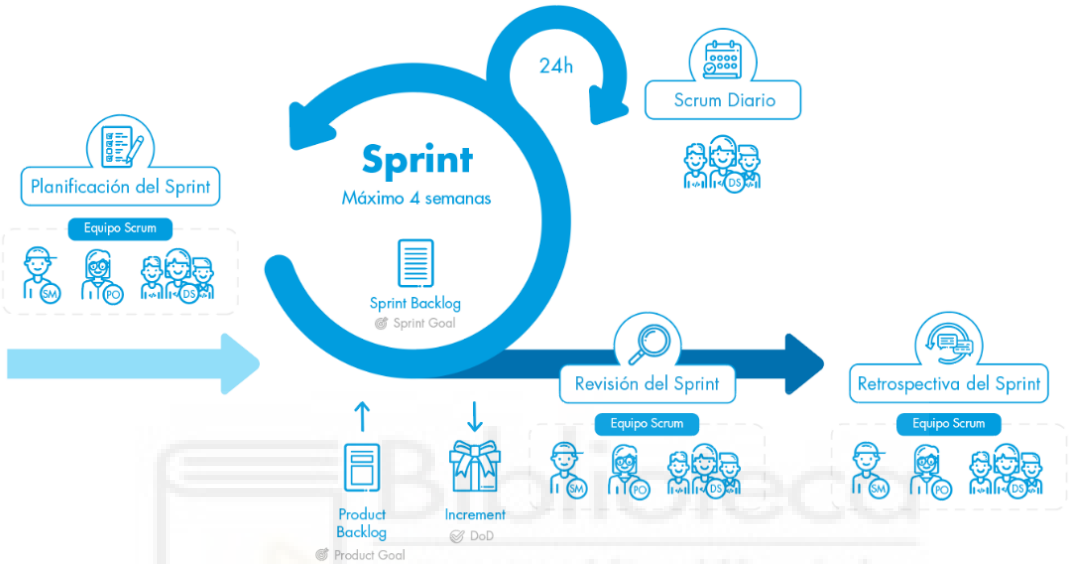


Figura 7. Resumen de las distintas fases propuestas por Scrum.

Scrum es un método para trabajar en equipo a partir de iteraciones o *sprints*. En la Figura **7** se resumen visualmente las distintas fase e interacciones entre los distintos miembros, las cuales se describen a continuación:

- **Reunión para la planificación del *sprint***: se planifica el sprint teniendo en cuenta su duración y se establecen sus objetivos y entregables resultados.
- **Scrum diario:** se ponen en común y se sincronizan las actividades elaborando un plan del día.
- **Trabajo de desarrollo durante el *sprint*:** nos aseguramos de que los objetivos se están cumpliendo, que no se producen cambios que alteran el objetivo del *sprint* y se mantiene un *feedback* constante.
- **Revisión del Sprint:** reunión con los integrantes del proyecto, en la que se estudia y revisa el proyecto. Se definen los aspectos a cambiar, en caso necesario, de mayor valor o probables para planificarlo en el siguiente *sprint*.

- **Retrospectiva del proyecto:** oportunidad del equipo de desarrollo para mejorar su proceso de trabajo y aplicar los cambios en los siguientes *sprints*.

Se puede observar la similitud entre los *sprints* propuestos por esta metodología y la forma de trabajar llevada a cabo también durante el desarrollo de la parte común explicada anteriormente. En nuestro caso, no pudimos trabajar diariamente, por lo que la frecuencia de los *sprints* fue mayor (de media de entre 2 y 3 semanas). Una vez finalizada la parte común del trabajo e identificados los requisitos iniciales, procedí a dedicar esfuerzos en solitario para crear una aplicación totalmente funcional. Merece la pena destacar que, aunque en el desarrollo de software ha sido realizado únicamente por mí y no hay un gran equipo, hemos decidido mantener esta metodología debido a la agilidad que aporta trabajar mediante *sprints* y la facilidad de adaptarse a los cambios propios de un trabajo de esta naturaleza.

3. MATERIALES Y MÉTODOS

3.1 Aplicaciones informáticas sobre EP

Fruto de la aplicación de la metodología explicada en la sección 2.2.1, se ha realizado una búsqueda de aplicaciones disponibles para los sistemas operativos para *smartphones* más populares: Android, iOS y Windows Phone.

La búsqueda se realizó en mayo del 2020. Pese a los resultados de las búsquedas han dado como resultado decenas de aplicaciones relacionadas, finalmente se analizaron en profundidad cinco aplicaciones que permiten un seguimiento básico de tratamientos a través de recordatorios, así como otros niveles más avanzados de funcionalidad. Durante el proceso de selección he tenido en cuenta algunos aspectos relacionados con características deseables para conseguir una aplicación funcional y usable, poniendo en práctica lo aprendido durante mis estudios de los últimos años. Por ejemplo, en el ámbito de la EP, en general el público objetivo de la aplicación (pacientes o cuidadores) será de edad avanzada; en el caso de los pacientes con EP, se trata de usuarios con trastornos de movimiento. Teniendo en cuenta estos factores hay probabilidades altas de problemas de visión y de interacción con los elementos de la aplicación. Por ello, se han incluido en el análisis aspectos en línea con los siguientes factores:

- **Usabilidad**: partiendo de que el perfil del usuario que pretende utilizar la aplicación es específico, el objetivo es permitir al usuario alcanzar objetivos en concreto con la mayor facilidad, eficiencia y satisfacción. Para poder lograr ese objetivo es necesario aportar información clara y concisa, bien organizada, elementos interactivos de gran tamaño y detalles como la ubicación en la que se encuentra el usuario en cada momento [28].

- **Interactividad**: la interactividad describe la relación de comunicación entre un usuario y un sistema informático. El diseño interactivo de las aplicaciones para dispositivos móviles determina el acceso de los usuarios a los contenidos, el cual, es un objetivo deseable en el desarrollo software de aplicaciones [29].

La referencia [24] contiene un estudio sobre el diseño de las aplicaciones móviles y hay apartados en los que comenta la definición de la usabilidad y de su relación con el usuario, mientras que el artículo [29] nos muestra un estudio en profundidad de cómo

intervienen los conceptos de interactividad e interacción entre el usuario y el sistema de información, mencionando qué efecto provoca ese tipo de acciones. Los factores anteriores pueden ser observados en las aplicaciones actuales, y más concretamente a través del análisis de sus interfaces gráficas.

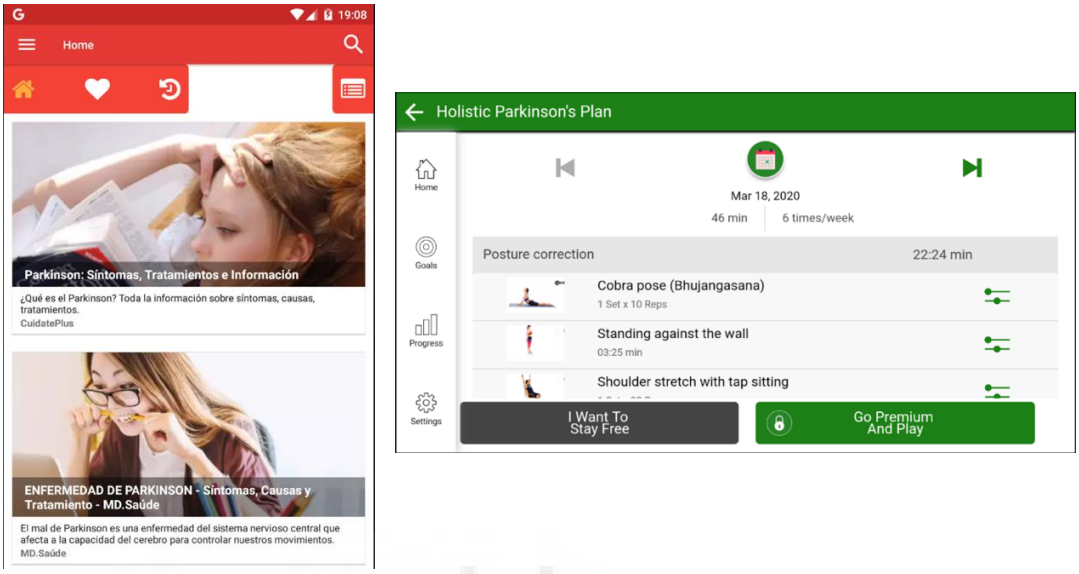


Figura 8. Capturas de pantalla de las aplicaciones "APP Parkinson Symptoms and Treatments" y "9zest"

Una vez seleccionadas estas cinco aplicaciones, se instalaron las aplicaciones mediante un emulador Android y se identificaron los elementos comunes potencialmente útiles. Por ejemplo, se identificaron patrones comunes como la presencia de páginas navegaciones accesibles y con imágenes descriptivas que ayuden a identificar su función dado que el perfil de usuario será de edad avanzada como se comentó en la introducción. A continuación, se describen algunos aspectos destacables de dos de las cinco aplicaciones seleccionadas:

- **App Parkinson Symptoms and Treatments [30]:** la finalidad de esta aplicación es mostrar información relacionada con la EP y sus tratamientos. Como se puede observar en la Figura 8 (izquierda), los elementos son de un tamaño relativamente grande, texto de tamaño intermedio y fotos de gran tamaño.
- **9zest [31]:** se asemeja mucho a la idea de este proyecto ya que ofrece una variedad de ejercicios para el tratamiento del usuario con EP. No es una aplicación centrada en un tratamiento conservador, ya que no incluye aspectos relacionados con medicación ni la relación entre ambas ciencias (farmacología y fisioterapia). En

la Figura 8 (derecha), se pueden observar la selección de elementos de esta aplicación que se utilizarán como referencia, como son el menú vertical o la forma de representar y estructurar la información.

3.2 Tecnologías para el desarrollo de aplicaciones híbridas

Existen diferentes alternativas para implementar aplicaciones para dispositivos móviles: aplicaciones nativas y aplicaciones híbridas (ver Figura 9).

- **Las aplicaciones nativas**: ofrecen el acceso a todas las capacidades del dispositivo (cámara, GPS, acelerómetro y agenda, entre otras), el alto rendimiento. Sin embargo, el precio de todas estas ventajas es alto ya que no es posible reutilizar el código fuente entre plataformas diferentes, y el esfuerzo se multiplica elevando los costos de desarrollo y actualización.

Por otro lado, el desarrollo multiplataforma se contrapone al nativo y se centra en la reutilización de código. Se procura entonces optimizar la relación costo/beneficio compartiendo la misma codificación entre las versiones para las distintas plataformas.

- **Las aplicaciones híbridas:** constituyen un tipo de desarrollo multiplataforma basado en tecnologías web (HTML, JavaScript y CSS). Este tipo de aplicaciones funcionan en un contenedor web especial con acceso a las capacidades del dispositivo a través de una API específica [32].

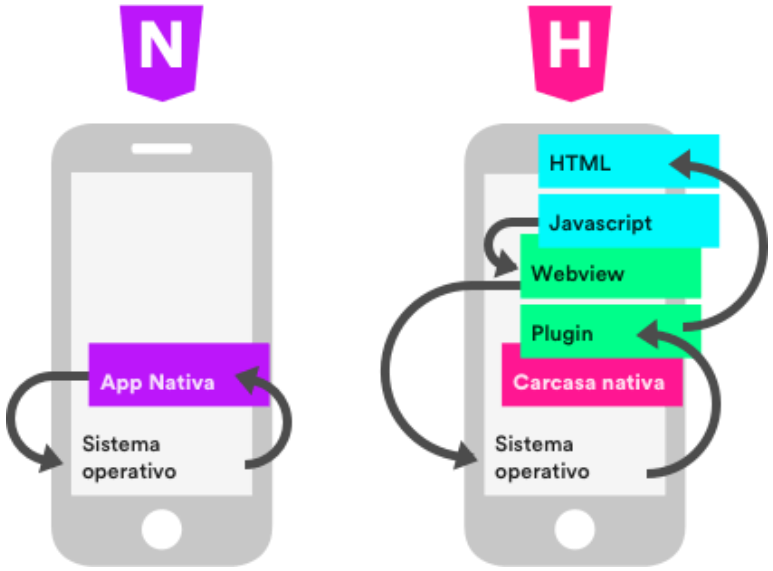


Figura 9. Representación visual de las diferencias entre una aplicación nativa (izquierda) de una aplicación híbrida (derecha). Imagen extraída de [33].

Las aplicaciones híbridas ofrecen una serie de ventajas para los programadores, ya que pueden realizar un único desarrollo que automáticamente podrá generar aplicaciones para las distintas plataformas. Por lo tanto, permiten la reutilización de código en las distintas plataformas, simplifican el acceso al hardware del dispositivo y permiten generar contenido que pueda ser distribuido a través de las diferentes tiendas de aplicaciones. Sin embargo, también presentan desventajas, por ejemplo, el uso de componentes no nativos en la interfaz perjudica la experiencia de usuario, y la ejecución se ve ralentizada por la carga asociada a los componentes web que utilizan en lugar de elementos nativos de cada plataforma.

Una vez analizadas las ventajas e inconvenientes de los distintos tipos de aplicaciones, en este proyecto se ha decidido implementar una aplicación híbrida por el ahorro de tiempo que supone, ya que, de otra forma se debería programar la misma aplicación en dos lenguajes de programación distintos con sus respectivos entornos; esto implicaría aprender más lenguajes de programación, ejecución del código en diversos dispositivos con la necesidad de mayor depuración del código. Esos factores, según mi criterio, harían el proyecto demasiado complejo, ya que, ese tipo de desarrollos suelen estar compuestos por equipos expertos en diferentes sistemas. Además, otra de las ventajas del desarrollo híbrido es que, para poder desarrollar aplicaciones para iOS, generalmente necesitas tener un equipo físico de Apple del cual yo no dispongo.

Merece la pena destacar también que, aunque se consideró realizar una página web adaptable usando HTML/CSS/JS y Responsive Web Design (RWD), finalmente nos decantamos por utilizar tecnologías de desarrollo de aplicaciones móviles por diversos factores, como podrían ser la gran variedad de funcionalidades que ofrece, la comunidad con sus diversos foros y la afinidad del desarrollador por este tipo de tecnologías de desarrollo. Una vez decidido el tipo de aplicación, en este apartado se van a explicar detalladamente las tecnologías y *Frameworks* para el desarrollo de aplicaciones híbridas que han sido escogidas.

3.2.1 **React Native + Expo + Node JS**

En la Figura 10 se muestran los logotipos de los diferentes *framework* seleccionados y que se explicarán a lo largo de esta sección. Un *framework* es un esquema de reutilización del software conformado por componentes y relaciones entre estos, por

ejemplo: la abstracción de clases, objetos o componentes que la conforman; además, provee diferentes componentes de conexión a base de datos, como controladores para conexión directa (MySQL, SQL Server, Oracle) o de manera general, mediante el estándar ODBC (Open Data Base Connectivity) [34]. Además, este tipo de entornos abastecen de un conjunto de herramientas que proporciona el fabricante, normalmente una plataforma de hardware, un sistema operativo (SO) o un lenguaje de programación, con el objetivo de ayudar a los desarrolladores de software a crear aplicaciones para una plataforma, sistema o lenguaje de programación específicos.

Figura 10. Logotipos de las distintas tecnologías seleccionadas para el desarrollo de la aplicación móvil

- **React Native** [35] es un framework y un proyecto de código libre desarrollado por Facebook en 2015. Su meta es permitir a los desarrolladores escribir un código de alta calidad para aplicaciones nativas para sistemas operativos iOS y Android utilizando tecnologías web comunes [36]. Las principales ventajas que proporciona esta tecnología son las siguientes: (1) reduce muchas de las ineficiencias del desarrollo nativo, (2) permite simplemente refrescar la aplicación después de cada cambio del código sin realizar los pasos tradicionales (escritura, compilado, desplegado y depurado), y (3) permite utilizar tecnologías comunes para el desarrollo multiplataforma.

Existen un conjunto de herramientas y servicios creados alrededor de React Native y plataformas nativas:

- **Expo** [37]**:** es un framework y una plataforma para aplicaciones universales de React. Expo ayuda al programador a desarrollar, construir, implementar e iterar rápidamente en iOS, Android y aplicaciones web desde la misma base de código JavaScript/TypeScript.

En este proyecto hemos escogido la combinación de estos dos Frameworks por las características anteriormente mencionadas, ya que puedes desarrollar una aplicación híbrida sin tener que programar en Java que es el lenguaje de Android y en Swift que es el lenguaje de iOS. Además, el uso del SDK ofrecido por Expo proporciona acceso a la funcionalidad del dispositivo y del sistema, como contactos, cámara y ubicación GPS u otra funcionalidad específica. Otro de los beneficios de utilizar Expo es su popularidad, que se refleja en la actualización periódica de su SDK, para el que cada mes hay una nueva versión que incluye una variedad de correcciones de errores, características y mejoras [38]. Para el desarrollo de este proyecto se ha usado la versión 41 (última disponible cuando se inició el desarrollo) de Expo SDK. A parte de las características mencionadas, el uso del SDK permite importar módulos de código en JavaScript que facilitan y aumentan la velocidad de desarrollo. Su uso es también imprescindible para que React Native y Expo puedan funcionar en conjunto. Además, ambas tecnologías tienen dependencias de versiones, es decir, si usas Expo SDK 41, la versión de React Native tiene que ser 0.63.3 por lo que es la que hemos utilizado también.

Dado que nuestra aplicación va a requerir del uso de un entorno de ejecución en tiempo real. Un entorno de ejecución en tiempo real es el encargado de proporcionar funcionalidad necesaria para que los programas puedan ejecutarse. Actúan como un pequeño sistema operativo para mediar entre el programa de la aplicación y el sistema operativo. Cuando el programa se ejecuta, envía instrucciones al procesador y la memoria del ordenador, y accede a los recursos del sistema. Así, el sistema en tiempo de ejecución incluye el hardware, el espacio de almacenamiento, las variables del entorno, las interacciones del usuario y los componentes de software [39]. En este trabajo utilizaremos el siguiente:

- **Node JS** [40]**:** está ideado como un entorno de ejecución en tiempo real de JavaScript orientado a eventos asíncronos y fue diseñado para crear aplicaciones en red escalables. La idea principal de Node.js es usar el modelo de entrada y salida sin bloqueo y controlado por eventos para seguir siendo liviano y eficiente frente a las aplicaciones en tiempo real de uso de datos que se ejecutan en los dispositivos.

Este entorno de ejecución es un requerimiento para que Expo pueda funcionar, por lo tanto, es totalmente necesario ya que es el encargado de ejecutar el código JavaScript en el servidor.

3.2.2 TypeScript: lenguaje para programar la aplicación

En este apartado se van a describir los lenguajes de programación utilizados, así como por qué y cómo ha evolucionado su uso a lo largo de todo el proceso de desarrollo. La elección del lenguaje está altamente influenciada por la elección de tecnologías descrita en la sección anterior. Es decir, React Native y Expo están escritos en JavaScript, eso quiere decir, que todos sus módulos los cuales se importan en el código de la aplicación también. Por este motivo, JavaScript ha sido el lenguaje utilizado principalmente para el desarrollo de nuestra aplicación híbrida.

JavaScript es un lenguaje de programación interpretado y se define como orientado a objetos el cual permite implementar funciones complejas. Además, hemos decidido utilizar las siguientes extensiones:

- **JSX:** es la extensión del lenguaje de React que permite escribir marcando directamente con instrucciones que serán remplazadas por llamadas directas a la API como React.createElement o cualquier otra a la que se dirija.
- **TypeScript y TSX:** TypeScript es una extensión de JavaScript destinada a facilitar el desarrollo de aplicaciones JavaScript a gran escala, aunque requiere de configuraciones adicionales en el desarrollo. La intención es que TypeScript proporcione una transición fluida para los programadores de JavaScript; los modismos de programación JavaScript bien establecidos son compatibles sin ninguna reescritura o anotaciones importantes [41]. TSX es similar a JSX, pero invocando a funciones específicas de TypeScript.

Aunque inicialmente se priorizó el uso de JavaScript puro debido a la experiencia previa adquirida durante el Grado, a medida que avanzó el desarrollo del proyecto JavaScript perdió protagonismo en favor de TypeScript. Para poder programar en TypeScript se requiere de cierto conocimiento previo de JavaScript que pude adquirir durante las primeras fases del desarrollo y me dio la confianza para dar el salto y aprender este nuevo lenguaje a pesar de tener una mayor complejidad. Además, el uso de TypeScript estuvo también motivado por la presencia de una gran variedad de código en

proyectos de React Native escrita en TypeScript que ha podido ser parcialmente reutilizado, lo que motivó nuestro interés por aprender a utilizar este lenguaje de programación. De hecho, algunas funcionalidades identificadas como requisitos requerían de ejemplos y referencias en este lenguaje. Otro de los aspectos clave fue el descubrimiento de los Hooks, los cuales se describirán más adelante. Tras aprender y profundizar en todo lo relacionado a este lenguaje incluso algunos de los problemas o fallos que se habían ocasionado en el desarrollo con JavaScript fueron reprogramados en TypeScript y de esa manera fueron solucionados.

3.3 Desarrollo de la base de datos: MySQL

Inicialmente se planteó almacenar los datos dentro de la aplicación mediante archivos tipo JSON o implementar una base de datos NoSQL como podría ser MYSQL Lite o similar. Sin embargo, como el planteamiento de este proyecto consistía en tener un sistema de información complejo y teniendo en cuenta la futura proyección del proyecto, se consideró que era mejor tener todos los datos externalizados para poder realizar análisis de datos y poder incluso integrarlos en otras utilidades que se comentarán más adelante. Por este motivo decidimos implementar la base de datos usando un esquema relacional con MySQL.

En el ANEXO II: esquema inicial de datos se puede consultar como el ejemplo el diagrama E/R en las primeras fases del proyecto. Usar este tipo de diagrama fue interesante ya que nos permitió acercar a los compañeros con bajos conocimientos técnicos a la aplicación y nos permitió usarlo como ayuda para la extracción de algunos requisitos. A medida que fue avanzando el proyecto el esquema evolucionó acorde con las necesidades de la aplicación, y con el fin de adaptarlo a la estructura de datos de nuevos requisitos, lo que supuso, por ejemplo, la necesidad de crear vistas de datos que se explican más adelante.

El uso de MySQL ha requerido tener que tomar la siguiente decisión de diseño para implementar el acceso a la BBDD. Se ha utilizado SQL (Structured Query Language) para implementar el acceso a la BBDD. SQL tiene como fin de acceder a bases de datos relacionales que son administradas por sistemas gestores de bases de datos [42]. Este lenguaje ha sido utilizado para la creación y administración de la base de datos relacional de la aplicación. Principalmente las funcionalidades que se han usado, han sido las

siguientes: tablas de datos, relaciones entre tablas, inserción de los datos, creación y gestión de vistas y realización de consultas.

3.3.1 Acceso a los datos mediante API: Flask-RESTful y PyCharm

Las tecnologías descritas en la sección anterior no disponían de funcionalidad directa para la comunicación con la base de datos, debido a ello, ha sido necesaria la implementación de una API que sirva para la comunicación intermedia entre la app y base de datos. Las APIs (Interfaces de Programación de Aplicaciones), son construcciones disponibles en los lenguajes de programación que permiten a los desarrolladores crear funcionalidades complejas de una manera simple. Estas abstraen el código más complejo para proveer una sintaxis más fácil de usar en su lugar. La API sería la intermediaria entre diversos componentes (en nuestro caso la aplicación móvil y la BBDD) para mandar y recibir consultas y respuestas de datos, siendo el intérprete de ellas.

En este proyecto, se ha utilizado un tipo de API denominada REST (Representational State Transfer) API [43]. Ofrece la posibilidad de realizar peticiones POST, GET, PUT, DELETE basadas en el protocolo HTTP para su consulta. Además, existen un tipo concreto de API REST denominada RESTful API. Para que un API REST se pueda considerar RESTful debe de cumplir los siguientes principios [43]: (1) Arquitectura cliente-servidor, (2) peticiones sin estados, (3) almacenable en caché, (4) interfaz uniforme, (5) sistema por capas y (6) código bajo demanda (opcional).

Entre las diferentes opciones disponibles para la implementación del API hemos elegido Flask-RESTful [44]. Es una extensión para Flask, siendo Flask un framework, que agrega soporte para construir rápidamente APIs REST. Es una abstracción ligera que funciona con sus bibliotecas/ORM (Object Relational Mapping) existentes. Se ha utilizado en este proyecto para realizar peticiones de datos de todo tipo tanto para consultar información como para modificarla.

En la Figura 11 se puede observar una descripción gráfica del proceso de comunicación cliente – servidor y cómo interactúa la REST API entre ambos, siendo nuestro cliente la aplicación móvil y el servidor nuestro servidor de bases de datos. En la parte del cliente podemos observar que para las consultas puede mandar peticiones HTTP de tipo GET, POST, PUT, DELETE y en la respuesta se reciben datos de tipo JSON (JavaScript Object Notation) o XML (Extensible Markup Language). Finalmente, en la

parte de servidor de bases de datos la REST API manda la petición del cliente y la manda al servidor para después recibir su respuesta y mandarla al cliente.

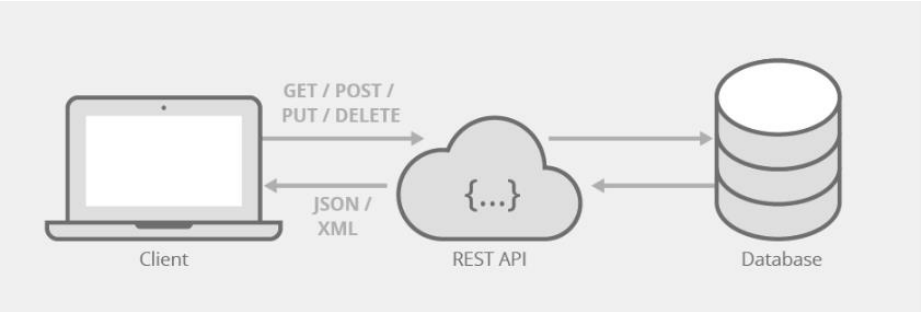


Figura 11. Ejemplo de comunicación con la BBDD a través de un API REST

En este proyecto hemos utilizado POSTMAN [45] para interactuar y probar la APIs mediante su interfaz gráfica, estas pruebas nos han permitido depurar el código antes de hacer uso de este desde la aplicación móvil. Este software ha sido utilizado principalmente para hacer pruebas para comprobar de que las peticiones a la API se realizan correctamente, ver las respuestas de este, el formato de los datos de las peticiones necesarias, y aprender en más profundidad todo lo relacionado a ello.

Finalmente, Python ha sido utilizado para programar la parte de REST API junto a un editor de código fuente denominado PyCharm. Python es un lenguaje de programación orientado a objetos, es fácil de usar e incluye una gran biblioteca estándar que admite muchas tareas de programación comunes [46].

3.4 Entornos de desarrollo (IDEs) y emuladores

En este apartado resumimos otros elementos necesarios para realizar las tareas de desarrollo, programación y prueba de la aplicación. En cuanto a la edición de código y la gestión de la BBDD hemos utilizado:

- **Visual Studio Code** [47]**:** es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes de programación. Este editor ha sido seleccionado porque cumple todas las necesidades del proyecto y estaba habituado a utilizarlo previamente, lo cual, ha sido un factor extra para tomar la decisión.

- **MySQL Workbench** [48]**:** es una herramienta visual unificada para arquitectos, desarrolladores y administradores de bases de datos. Proporciona modelado de datos, desarrollo SQL y herramientas de administración integrales para la configuración y administración del servidor. En este proyecto ha sido la herramienta utilizada como contenedor de información en la cual se ha generado la estructura de datos y creado el servidor de BBDDs cuyo acceso se implementará a través de nuestra API RESTful.

Aunque en teoría las tecnologías elegidas ofrecen la posibilidad de ejecutar la aplicación desarrollada en los diferentes sistemas operativos mencionados, existen ciertos componentes que requieren de depuración para garantizar tal efectividad, los cuales, no han sido posibles de comprobar ya que el no disponer de un ordenador de la marca Apple ha dificultado la prueba de la versión para IOS. En nuestro caso, debemos destacar que hemos definido un entorno de pruebas centrado en la aplicación generada para los sistemas Android. Para la ejecución de la aplicación, inicialmente, se utilizó el siguiente emulador:

- **Android Studio Emulator:** es el emulador de dispositivos móviles virtuales incluido en el IDE Android Studio [49]. Esta herramienta nos ha permitido configurar un dispositivo virtual que permite ejecutar la aplicación desarrollada y poder visualizar el resultado de la compilación del código.

Sin embargo, tras aumentar la complejidad del código, los tiempos de ejecución empezaron a incrementarse demasiado y en las fases finales de desarrollo se desplegó la aplicación directamente a un dispositivo móvil real conectado al ordenador mediante cable USB con el modo desarrollador activado. También merece la pena destacar que, aunque Android Studio incluye un potente editor de códigos y las herramientas para desarrolladores, hemos preferido utilizar Visual Studio Code para las distintas tareas de programación y depuración como se ha comentado anteriormente.

3.5 Propuesta de solución

El análisis realizado previamente nos ha permitido identificar los diferentes elementos que serán necesarios para implementar la aplicación. En la Figura 12 se muestra un

diagrama que resumen la arquitectura software que será implementada, relacionando cada una de las partes con la tecnología utilizada.

Durante el funcionamiento de la aplicación, los datos constantemente interactuarán entre el cliente y servidor siendo la REST API intermediaria de ellas. Durante la fase de desarrollo se ha seguido el siguiente flujo de trabajo: (1) el código implementado en Visual Studio y correspondiente con la parte del cliente debe ser compilado y ejecutado usando el entorno de ejecución Node JS, y (2) el resultado de la ejecución será visible tanto en el smartphone conectado al ordenador como en el dispositivo virtual emulado mediante Android Studio, a través del que se podrá interactuar y probar la aplicación.

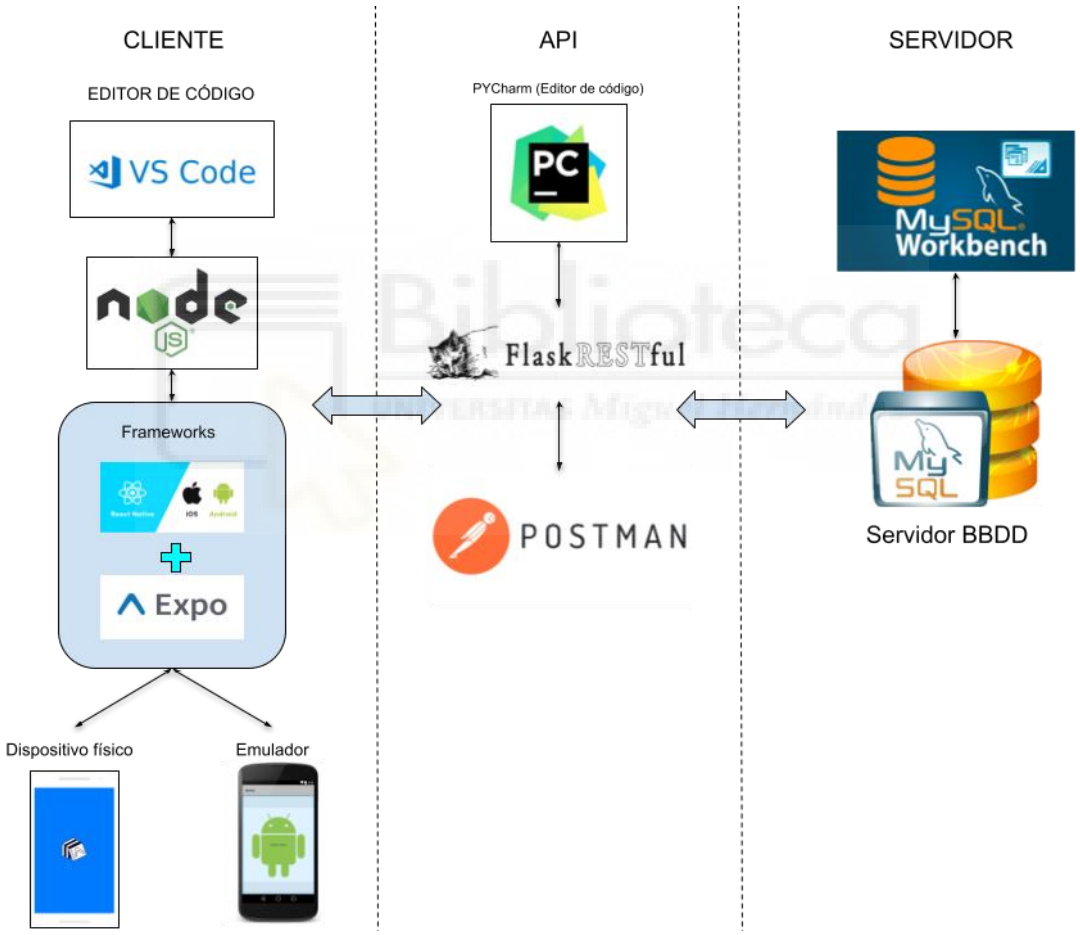


Figura 12. Propuesta de solución que incluye un resumen de los distintos elementos de la aplicación y las herramientas y tecnologías utilizadas para implementarlas

Si durante la prueba de la aplicación, alguna de la funcionalidad requiere la consulta de la BBDD, en el código del cliente se incluirán sentencias que permitan implementar la llamada a nuestra API mediante los módulos ofrecidos por Expo y React Native.

Como se ha mencionado anteriormente, se ha desplegado el API de acceso a los datos utilizando PyCharm, el cual, está ejecutado en tiempo real para escuchar las peticiones en todo momento y conectado al servidor de la base de datos mediante la utilización de las funcionalidades aportadas por el framework de FLASK. La implementación del API de acceso a los datos se ha realizado de forma independiente, es decir, antes de ser invocada desde la aplicación móvil se han realizado pruebas de los métodos implementados a través de POSTMAN. Esto ha permitido, por ejemplo, que en el caso de que alguna petición fuese errónea o estuviese mal implementada devolviendo un código de error 500 se puedan realizan pruebas en para identificar el problema y analizar la respuesta del servidor.

Además, previamente se ha utilizado MySQL Workbench para construir el esquema de bases de datos utilizando su editor visual que ha permitido construir las tablas y sus relaciones. Otras sentencias de consulta e inserciones de datos también han sido ejecutadas y testeadas a través de esta herramienta.

4. RESULTADOS Y DISCUSIÓN

En este apartado se van a describir los resultados obtenidos tras todas las fases de diseño e implementación de la aplicación, así como las dificultades que han presentado a lo largo de todo el proceso.

4.1 Análisis y diseño de software

En este apartado se va a mostrar un diagrama de casos de uso con el fin de representar visualmente las necesidades de la aplicación y resumir los requerimientos identificados a lo largo del proyecto.

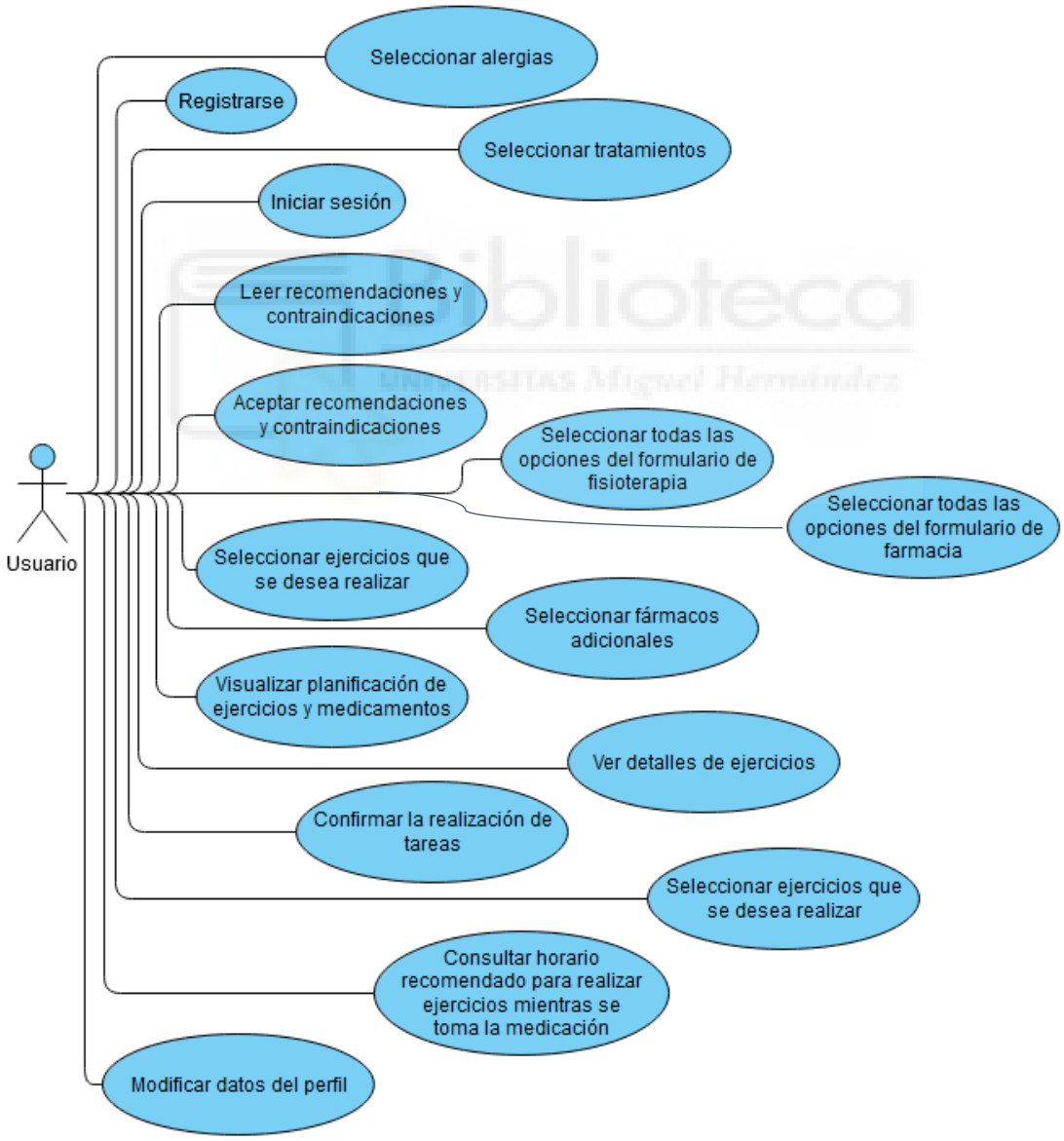


Figura 13. Diagrama de casos de uso.

4.1.1 Requerimientos y casos de uso

En la Figura 13 se muestra un diagrama de caso de uso elaborado teniendo en cuenta todas las acciones que se le ofrecerán al usuario, y extraídas del resumen de los requerimientos establecidos. Se puede observar que existe un único actor que es el usuario que va a utilizar su dispositivo móvil para realizar ciertas acciones. Aunque el desarrollo completo de cada caso de uso se puede encontrar en el ANEXO I, a continuación, describimos algunos de ellos:

- **Registrar usuario:**
    - **Información de sesión:** en primer lugar, el usuario tiene que registrarse para poder iniciar sesión. Una vez registrado podrá acceder a la pantalla de inicio desde la que podrá leer las recomendaciones y contraindicaciones y posteriormente deberá de aceptar que han sido leídas.
    - **Información médica:** una vez realizadas las acciones anteriores, podrá rellenar dos formularios (uno de fisioterapia y otro de farmacia) que asistirán al usuario a la hora de introducir datos médicos que serán usados para crear automáticamente un plan personalizado según su tratamiento.
- **Creación de un plan personalizado para el paciente:**
    - **Algoritmo automático:** se deberá diseñar un algoritmo que permita crear un plan personalizado teniendo en cuenta las características de cada usuario. Se materializarán en este algoritmo el resultado de la investigación realizada por los compañeros.
    - **Interactividad y seguimiento**: una vez calculado el plan, se le mostrará al usuario de una forma amigable que le facilite su seguimiento. Además, se le permitirá registrar eventos y contratiempos, que podría ser utilizados en tiempo real para recalcular el plan. Por ejemplo, el usuario podrá consultar la programación diaria en el calendario pudiendo consultar los detalles de cada ejercicio y seleccionar las tareas completadas.

Teniendo en cuenta lo anterior, el perfil del usuario estará compuesto tanto por los datos personales insertados en el formulario de registro como por su información médica. Todos estos datos podrán ser eliminados y modificados desde la pantalla de perfil.

Se debe tener en cuenta que los formularios para introducir los datos médicos alcanzan cierta complejidad debido a la cantidad de dependencias entre las diferentes preguntas. Por ello, se deberá asistir al usuario para mostrarle en cada pantalla únicamente la información sensible de ser completada. A continuación, describimos algunos requisitos relacionados con los dos formularios definidos en la aplicación:

- **Formulario de fisioterapia:** el usuario deberá registrar una serie de cuestiones relacionadas con su enfermedad y posibles ejercicios que podrían ser incluidos para completar su tratamiento. Al final del proceso se lo ofrecerán al usuario el listado de ejercicios de los que se podrán consultar sus detalles clicando sobre un botón de información, y el usuario podrá seleccionar algunos de ellos.

- **Formulario de farmacia:** el usuario tendrá que seleccionar los tratamientos que tiene asignados por su médico y otros fármacos adicionales esté tomando sin prescripción médica. Se incluirá un asistente que a partir del texto introducido por el usuario sugiera los fármacos a seleccionar registrados en una base de datos. Además, se deberán indicar otros aspectos como posibles alergias a medicamentos, así como otras cuestiones como si tiene demencia, depresión o si se está embarazada y/o en periodo de lactancia en el caso de que el usuario sea mujer.

4.1.2 Diseño de la base de datos: modelo entidad/relación

En la Figura 14 se muestra el modelo entidad relación (E/R) que refleja la base de datos desarrollada en nuestra aplicación. Este esquema está compuesto por tablas y vistas, las cuales están relacionadas entre sí. Las líneas discontinuas del modelo representan las relaciones entre las tablas, los rectángulos azules a las tablas y los rectángulos amarillos a las vistas.

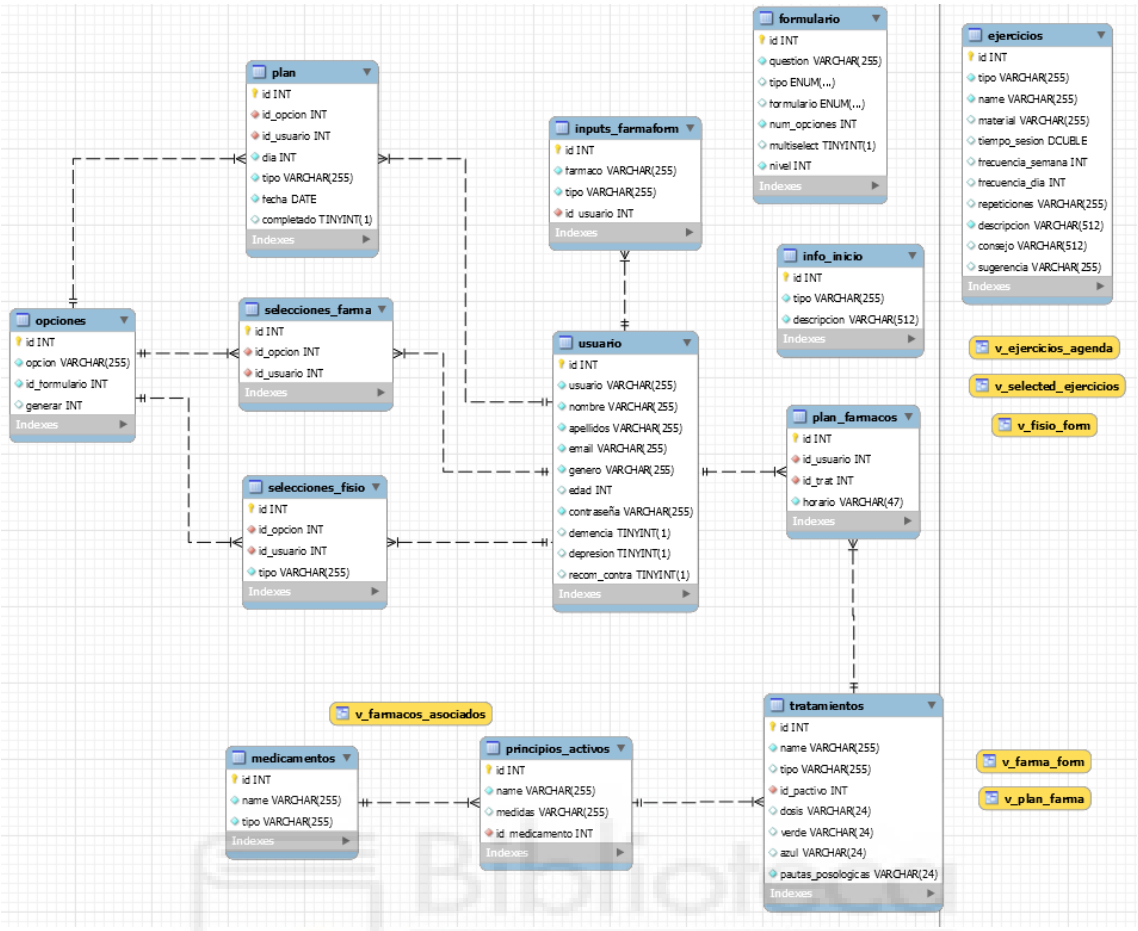


Figura 14. Esquema entidad/relación con el modelo de datos de la herramienta

Las vistas, aunque no tengan líneas discontinuas para representar relaciones, están relacionadas con varias tablas e incluyen condiciones para generar una tabla con los datos necesarios. Cada vista está diseñada para facilitar las necesidades de la aplicación a la hora de representar datos, de tal forma en la que se muestren únicamente los datos necesarios, los cuales, deben de cumplir ciertas condiciones. A continuación, se describe brevemente la funcionalidad de cada una de las vistas:

- **`V_fisio_form`:** esta vista sirve para tener en la misma tabla tanto las cuestiones de la tabla `formulario` como las opciones del cuestionario de fisioterapia de la tabla `opciones`. Ello facilita la carga de datos ya que simplemente se hace una consulta simple para obtener todos los datos de la vista teniendo en cuenta las relaciones entre ambas tablas y el tipo de formulario. La finalidad de esta vista consiste en cargar y mostrar todas las cuestiones con sus opciones relacionadas en la pantalla del formulario de fisioterapia.

- **`V_selected_ejercicios:`** esta vista está diseñada para organizar las opciones seleccionadas en el formulario de fisioterapia. Está relacionada con las tablas: `selecciones_fisio`, `opciones` y `formulario`. Siendo la condición de que el campo tipo de la tabla `selecciones_fisio` sea "bloque" ya que existe otro tipo para otra utilidad que se mencionará más adelante. Se utiliza para cargar los ejercicios en la pantalla de ejercicios dependiendo de los bloques de ejercicios seleccionados en el formulario.

- **`V_ejercicios_agenda:`** está compuesta por las tablas `selecciones_fisio`, `ejercicios` y `plan`. La finalidad de esta vista es obtener todos los datos seleccionados siendo la condición de que en la tabla `selecciones_fisio` el tipo sea "ejercicios" para cargar los ejercicios seleccionados de cada bloque en el calendario con sus respectivos datos para la ventana modal de muestra de detalle de cada ejercicio.

- **`V_farma_form:`** vista similar a `V_fisio_form`, compuesta por la tabla `formulario` y opciones con la condición de que el tipo de formulario sea de tipo "medicina". La finalidad es obtener y cargar los datos de todas las cuestiones y opciones en el formulario de farmacia.

- **`V_plan_farma:`** esta vista es un poco diferente de las demás ya que contiene valores binarios y cadenas de caracteres separados por comas siendo una letra un color para generar la planificación del horario de recomendaciones para realizar ejercicios. Está compuesta por la tabla de `tratamientos`, `plan_farmacos`, `principios_activos`, `medicamentos` e `inputs_farmaform`.

A modo de ejemplo, se muestran en la Figura 15 la información obtenida al realizar una consulta a la vista `V_ejercicios_agenda` extraída desde nuestra aplicación:

id_op	id_usuario	tipo	name	material	tiempo	frecuen	frecue	repetici	descripcion	consejo	sugerencia	dia	fecha	completado
1	3	Ejercicios de equilibrio	Ejercicios de equilibrio con b…	NULL	10	3	NULL	NULL	Inicialmente te pones de pie con…	Nuestro consejo es que si te encuentras muy in…	NULL	1	2021…	0
2	3	Ejercicios de equilibrio	Ejercicios de equilibrio bota…	NULL	5	3	3	NULL	Primero te pones de pie. Vas an…	NULL	NULL	1	2021…	1
3	3	Ejercicios de equilibrio	Ejercicios de equilibrio con o…	NULL	5	2	2	NULL	Inicialmente te pones de pie en …	NULL	Nuestra sugerencia …	1	2021…	0
4	3	Ejercicios de equilibrio	Ejercicio de resistencia prog…	NULL	15	3	NULL	NULL	Inicialmente te sientas en una sil…	NULL	NULL	1	2021…	0
5	3	Ejercicios de equilibrio	Ejercicio fuerza/resistencia …	El material nec…	10	3	NULL	NULL	Primero te pones detrás de la ca…	Nuestro consejo es que puedes ir aumentando l…	Nuestra sugerencia …	1	2021…	0
6	3	Ejercicios de equilibrio	Ejercicios de equilibrio para …	NULL	5	3	3	NULL	Inicialmente te pones de pie. Lu…	NULL	NULL	1	2021…	0
1	1	Ejercicios de equilibrio	Ejercicios de equilibrio con b…	NULL	10	3	NULL	NULL	Inicialmente te pones de pie con…	Nuestro consejo es que si te encuentras muy in…	NULL	NULL	NULL	NULL
2	1	Ejercicios de equilibrio	Ejercicios de equilibrio bota…	NULL	5	3	3	NULL	Primero te pones de pie. Vas an…	NULL	NULL	NULL	NULL	NULL
3	1	Ejercicios de equilibrio	Ejercicios de equilibrio con o…	NULL	5	2	2	NULL	Inicialmente te pones de pie en …	NULL	Nuestra sugerencia …	NULL	NULL	NULL
4	1	Ejercicios de equilibrio	Ejercicio de resistencia prog…	NULL	15	3	NULL	NULL	Inicialmente te sientas en una sil…	NULL	NULL	NULL	NULL	NULL
5	1	Ejercicios de equilibrio	Ejercicio fuerza/resistencia …	El material nec…	10	3	NULL	NULL	Primero te pones detrás de la ca…	Nuestro consejo es que puedes ir aumentando l…	Nuestra sugerencia …	NULL	NULL	NULL
6	1	Ejercicios de equilibrio	Ejercicios de equilibrio para …	NULL	5	3	3	NULL	Inicialmente te pones de pie. Lu…	NULL	NULL	NULL	NULL	NULL
7	1	Estiramiento muscular	Estiramiento para los isquio…	NULL	0.5	4	5	NULL	Inicias el estiramiento sentado e…	NULL	NULL	NULL	NULL	NULL
8	1	Estiramiento muscular	Estiramiento para el psoas	NULL	0.5	4	5	NULL	Primero te pones en posición de …	NULL	NULL	NULL	NULL	NULL
9	1	Estiramiento muscular	Estiramiento para el glúteo …	NULL	0.5	4	5	NULL	Inicias el estiramiento acostado …	NULL	NULL	NULL	NULL	NULL
10	1	Entrenamientos de	Ejercicio fuerza/resistencia …	El material nec…	10	3	NULL	NULL	Primero te pones detrás de la ca…	Nuestro consejo es que puedes ir aumentando l…	Nuestra sugerencia …	NULL	NULL	NULL

Figura 15. Ejemplo que muestra el contenido de la vista `V_ejercicios_agenda`

4.1.3 **Prototipado de la GUI**

Como se comentó en la sección, una herramienta que ha resultado muy útil en la fase de diseño del software ha sido la elaboración de prototipos para mostrar las características a incluir en la GUI (*Graphic User Interface*) de la aplicación. Durante el desarrollo del proyecto se realizaron principalmente dos prototipos.

En el primer prototipo de interfaz (ver Figura 16) se diseñaron los elementos básicos, con el fin de tener una referencia para el desarrollo posterior y mostrar a los integrantes del grupo la idea inicial de la aplicación en relación con lo acordado en las reuniones. Como resultado de ello se crearon unas pantallas de inicio de sesión y registro relativamente comunes, pero visualmente atractivas. Posteriormente se diseñó la pantalla de inicio junto a la pantalla del calendario con un menú de navegación lateral, con la intención de reflejar que el calendario va a ser interactivo e informativo. Para la elaboración de este prototipo se utilizó la herramienta online Moqups [50].

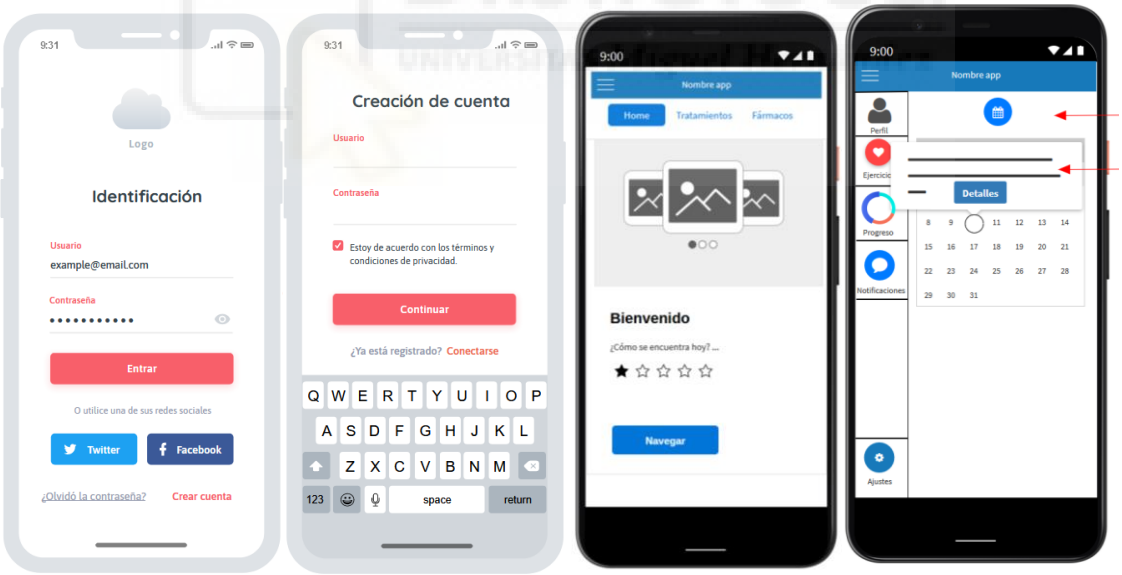


Figura 16. Capturas de las pantallas login, creación de cuenta, home y calendario pertenecientes al primer prototipo desarrollado

El prototipo inicial fue de gran ayuda por ser una representación visual inicial para tener un punto de partida a la hora de debatir sobre la interfaz con los integrantes del proyecto. Sin embargo, una vez avanzamos en las fases de diseño y nos adentramos en las fases de desarrollo fue necesaria la realización de un segundo prototipo más avanzado.

La elaboración de este segundo diseño de interfaz se realizó con intención de ofrecer una idea más desarrollada de lo que sería la aplicación más cercana al diseño final, con muchas más pantallas, botones, objetos visuales y elementos funcionales como podría ser formularios, videos, etc.

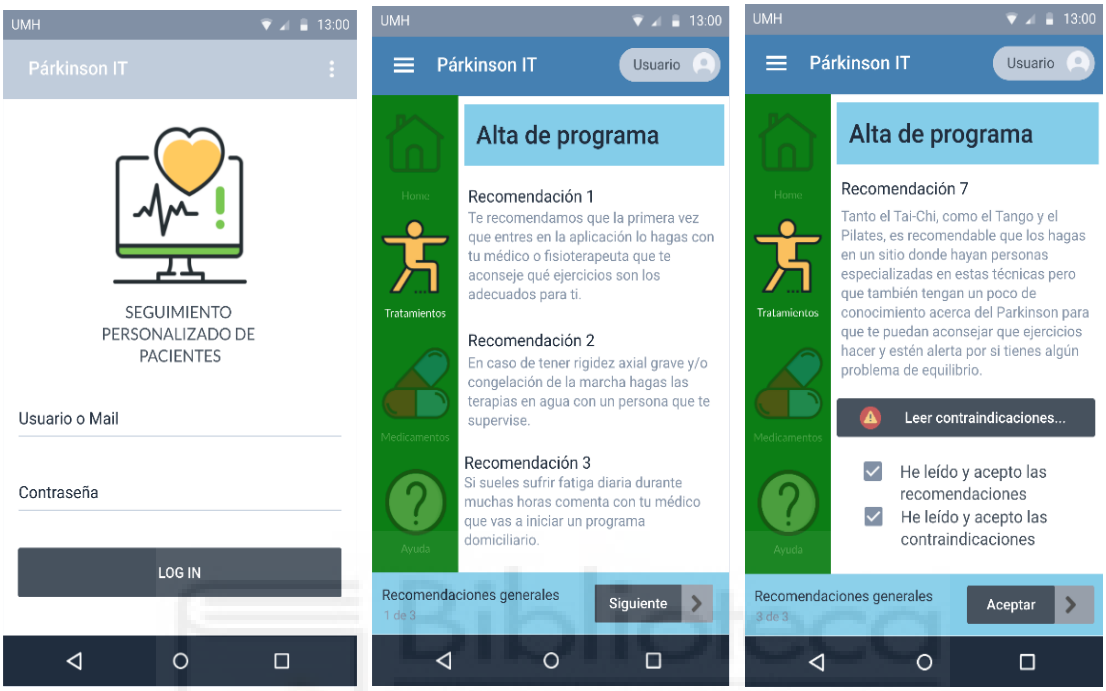


Figura 17. Captura de las pantallas Login y Home del segundo prototipo

Por ejemplo, en la Figura 17 se puede observar la evolución de la pantalla de registro y el Home con información de las recomendaciones y contraindicaciones. Tras aceptar esos datos desaparece la ventana emergente y solamente aparece la información.

A continuación, en la Figura 18 se muestra un ejemplo del formulario de fisioterapia con varias preguntas y con sus respectivas opciones seleccionables. Finalmente, la Figura 19 muestra un extracto de la pantalla de seguimiento desde la que el paciente podrá consultar el calendario con la organización de su plan personalizado y una pantalla de detalle de un ejercicio con su descripción y un video demostrativo.

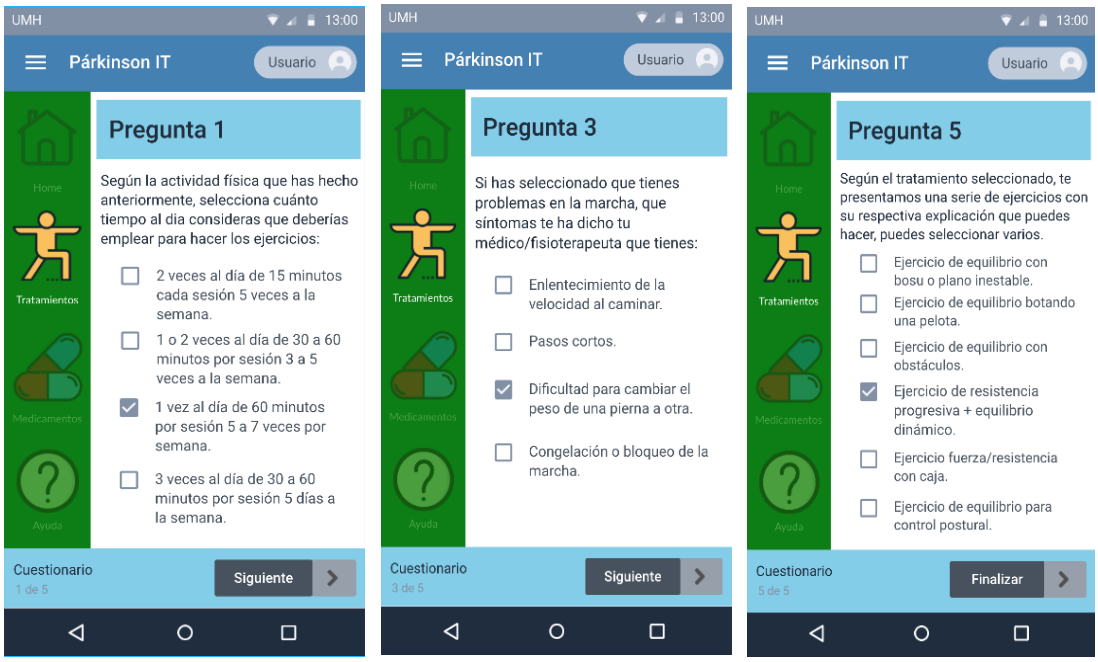


Figura 18. Captura de una selección de 3 pantallas del formulario de fisioterapia del segundo prototipo

Figura 19. Captura de las pantallas calendario y detalle de ejercicios del segundo prototipo

Para la realización del segundo prototipo se utilizó como punto de partida los aspectos acordados en las reuniones posteriores siendo la referencia el anterior prototipo y los aspectos visuales investigados y acordados en la fase intermedia. Para finalizar este

segundo prototipo, se crearon las distintas vistas con el programa de edición de imágenes Gimp [51], a partir de una interfaz gráfica base implementada con las tecnologías anteriormente mencionadas.

4.2 Implementación

En este apartado se resumirán algunos aspectos relacionados con decisiones de diseño y problemas que han surgido durante la fase de implementación de este proyecto. Se debe tener en cuenta que en esta sección se describen algunos aspectos relevantes con el objetivo de describir el uso de la tecnología React Native y el lenguaje TypeScript en algunas partes del proyecto, por lo que algunas partes no serán descritas para no extender la longitud de este documento (por ejemplo, cómo se ha implementado el registro y autenticación del usuario…).

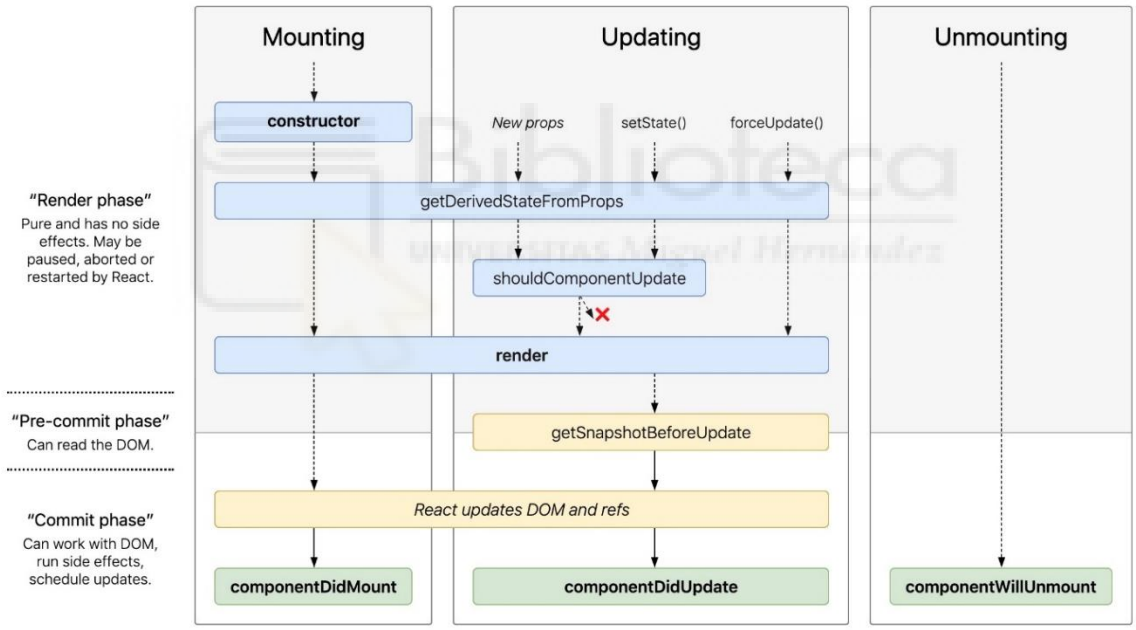


Figura 20. Ciclo de vida de los componentes y métodos de estado de React Native. Imagen extraída de [52]

4.2.1 Aspectos destacables de React Native y TypeSripts (Hooks)

Programar en un marco de frameworks y tecnologías concreto requiere analizar las y conocerlas bien para poder integrarse y desarrollar con ellas. En el caso de React Native, todos sus componentes heredan de la clase React y tienen sus propias fases. Cuando se

crea e inserta una instancia de un componente en el DOM, obtiene propiedades o accesorios. Tras ello comienza el ciclo de vida. En la Figura 20 se muestra visualmente las distintas fases de este proceso que se describen brevemente a continuación:

- **Montaje:** se crea una instancia del componente y se inserta en el DOM.
- **Actualización:** se ejecuta cuando el componente recibe algún cambio o actualización del código.
- **Desmontaje**: se ejecuta cuando el componente ya no es necesario.

Una parte muy importante en este framework es la gestión y el manejo de errores. La propuesta de React Native es la siguiente: cuando hay un error durante la renderización de un componente se invoca la ejecución del método correspondiente que podría codificarse en un método de ciclo de vida o en el constructor de cualquier componente secundario.

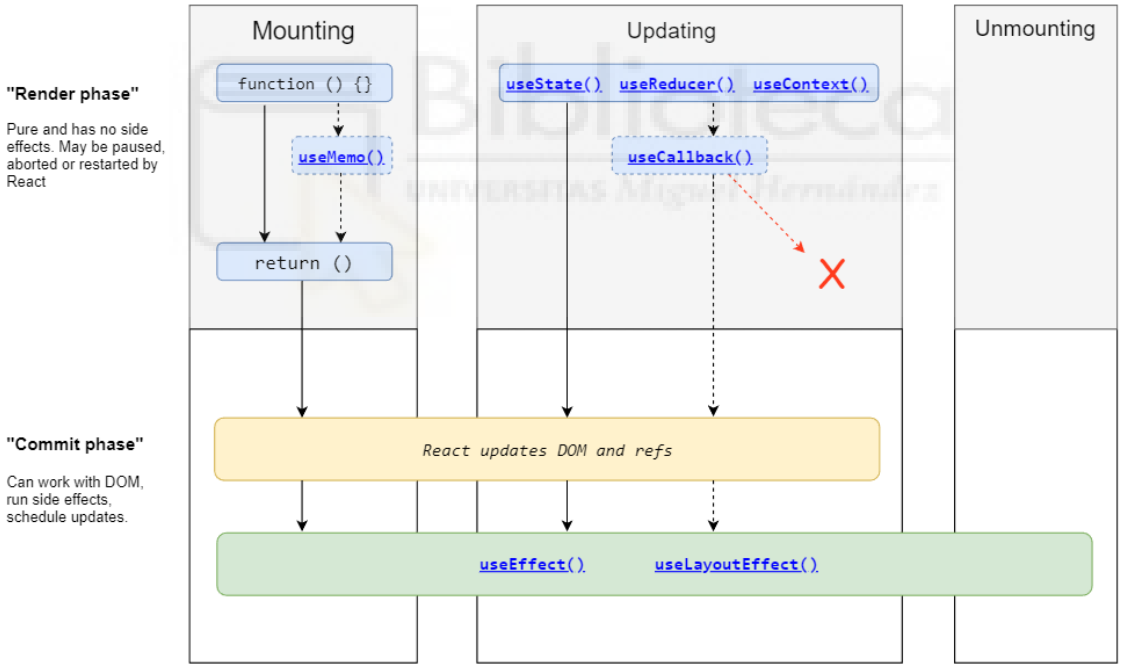


Figura 21. Gráfico resumen del ciclo de vida de los Hooks en React. Imagen extraída de [53]

A su vez, la implementación de código en TypeScript requiere del conocimiento de ciertas funcionalidades como los Hooks, los cuales, facilitan mucho la vida a la hora de programar. Los Hooks son una característica relativamente nueva en React Native que permite utilizar los estados y otras características sin necesidad de tener que crear las

clases. Los estados son asignaciones de datos en variables durante un ciclo de renderizado. Los Hooks son unos componentes del contenedor con el fin de administrar los estados y solicitudes al servidor. Los estados se propagan a otros contenedores a través de los Props. En la Figura 21 se puede observar el ciclo de vida con el uso de Hooks. En este proyecto se han definido y utilizado los siguientes Hooks:

- **`useState`:** permite añadir el estado de React a un componente de función. Conceptualmente, los componentes son como las funciones de JavaScript. Aceptan entradas arbitrarias (llamadas "props") y devuelven a React elementos que describen lo que debe aparecer en la pantalla.

  Normalmente contiene dos funciones, una para acceder a la información y otra para asignar la información a la variable de estado. El procedimiento es el siguiente: (1) declarar el hook, (2) asignar valores al tipo de dato definido y (3) acceder a la información para manipularla a placer.

- **`useRef`:** devuelve un objeto `ref` mutable cuya propiedad `.current` se inicializa con el argumento pasado (`initialValue`). El objeto devuelto se mantiene durante la vida completa del componente. Se ha usado principalmente para acceder a valores anteriores de la actualización de estado de una variable.

- **`useEffect`:** se ejecuta cuando el renderizado es completado, generalmente es usado para realizar peticiones a la API.

En los siguientes ejemplos, vamos a mostrar algunos fragmentos de código extraídos del proyecto que ilustran el uso de estos Hooks en diversos componentes usados por nuestra aplicación. A continuación, se incluyen algunos ejemplos utilizados para definir la estructura de datos para cargar la tabla del formulario de farmacia.

En la Figura 23, Figura 24 y Figura 25 se muestra el proceso de asignación de valores de estado a variables constantes definidas, las cuales, parten de una estructura de datos predefinida, también conocida interfaz.

```
interface MedicForm {
    id_form: number,
    question: string,
    tipo: string,
    formulario: string,
    opc_id: number,
    opcion: string,
    num_opciones: number,
    generar: boolean,
    nivel: number,
    multiple: boolean
}


const preguntas: MedicForm[] = [];
```

Figura 22. Ejemplo de declaración de una interfaz y su asignación a una variable auxiliar.

A continuación, en la Figura 23 se puede observar como la constante es declarada del tipo `MedicForm` para definir qué tipo de datos van a ser asignados a la misma. A su vez está compuesto por una variable y una función. La variable es `medicform`, es dónde va a estar ubicada la información asignada mediante la función `setForm` que va a ser llamada posteriormente de la declaración. Una vez definido el tipo de la variable se realiza una asignación para definir que va a ser una variable que use los estados, siendo la asignación `React.useState`:

```
const [medicform, setForm]: [MedicForm[], (posts: MedicForm[]) => void] = React.useState(
    preguntas
);
```

Figura 23. Ejemplo de la declaración de la variable del Hook useState

Una vez realizada la declaración, en el siguiente fragmento de código de la Figura 24 se puede observar cómo se obtiene la información de la base de datos y esa información resultante se asigna a la variable `medicfrom` mediante la función `setForm` sin necesidad de adaptación de formatos de datos:

```
useEffect(() => {
  query.default.getData2("V_farma_form").then(data => {
      setForm(data);
  });
```

Figura 24. Ejemplo de la asignación de valores del Hook useState a través de la función setForm, siendo en este caso una interfaz el componente

Finalmente, se pueden manipular y/o utilizar esa información para un fin determinado, en este caso, en la Figura 25 se itera por todos los registros hasta que se cumplen las condiciones dadas. En el caso de que esas condiciones se cumplan se devuelve un componente, es decir, un elemento visual el cual aparecerá en la página relacionada con el código, en este caso, la página del formulario de farmacia.

```
{medicform.map((value, index) => {

  if(index == 0 || index >= 1 && medicform[index - 1].question != medicform[index].question) {
      if(value.opc_id == null){
          return(
```

Figura 25. Ejemplo de acceso a valores del Hook useState

A su vez, en la Figura 26 se muestra un ejemplo de función utilizando el hook `useRef`. Concretamente, para actualizar las variables de estado inmediatamente y forzar un renderizado ya que hay casos en los que se necesitan varios renderizados para poder contemplar el cambio de estado de esa variable inmediatamente.

```
const doUpdate = useRef(() => {
    setTimeout(() => {
      setAceptado(true);
    }, 0);
    forceRender({});
  }).current;
```

Figura 26. Ejemplo de uso del Hook useRef.

Para finalizar, en la Figura 24 se puede observar la declaración previa de `useEffect` para que se almacenen los valores obtenidos de la base de datos una vez completado el ciclo de renderizado inicial.

4.2.2 Acceso a la base de datos a través del API

En ese apartado, se va a detallar un ejemplo para obtener información de una tabla o vista desde la aplicación móvil, lo que requiere a su vez la consulta al método del servicio RESTful desarrollado. Ese procedimiento está desarrollado de la siguiente manera:

1. **Creación de llamada asíncrona de acceso a datos:** se debe crear de función asíncrona dentro del fichero `utils.ts`. Esta función se ha parametrizado para construir la URL de acceso al API mediante la concatenación de texto. Una vez construida la URL de acceso al API, se realiza la petición mediante `axios` que es un cliente HTTP proporcionado pornode.js. La respuesta se obtiene como un objeto JSON que será procesado posteriormente. En la Figura 27 se muestra como ejemplo la consulta de los datos asociados a un usuario.

```
async function getDataByUser(table: string, user: number){
  const url = "http://192.168.1.222:5006/getbyuser?table=";
  const peticion = url + table + "&user="+user
  const promise = await axios.get(peticion).then((response) => response.data)

  return promise
}
```

Figura 27. Ejemplo de función asíncrona que realiza una consulta al API mediante el cliente HTTP axios

2. **Uso de la llamada asíncrona para acceder a los datos:** una vez creada la función asíncrona se puede importar en cualquier archivo y utilizar esa función para obtener los datos de cualquier tabla que tenga una columna de usuario. Como ejemplo, se muestra en la Figura 28 el código implementado para obtener los ejercicios seleccionados por el usuario a través de la vista `V_ejercicios_agenda`. Como se puede observar, se indica el identificador del usuario mediante una variable global y tras obtener los datos se pueden almacenar donde se desee y manipularlos libremente.

Obviamente, para que la llamada tenga una respuesta debe existir una función en la API que realice la consulta a la BBDD implementada.

```
query.default.getDataByUser('V_ejercicios_agenda', ENV_id_usuario).then((response) => {
  if(response[0] == null){
     setFisioForm(false);
  }
  else{
      setFisioForm(true);
      setEjercicios(response);
  }
});
```

Figura 28. Ejemplo de uso de la llamada asíncrona para acceder a la vista de ejercicios

```
@app.route('/getbyuser', methods=['GET'])
def getbyuser():
    table = str(request.args.get('table'))
    user = str(request.args.get('user'))
    if request.args:
        try:
            conn = mysql.connect()
            cursor = conn.cursor(pymysql.cursors.DictCursor)
            cursor.execute("SELECT * FROM %s WHERE id_usuario = %s" % (table, user))
            empRows = cursor.fetchall()
            response = jsonify(empRows)
            response.status_code = 200
            return response
        except Exception as e:
            print(e)
        finally:
            cursor.close()
            conn.close()
    else:
        return "No query string received", 200
```


Figura 29. Implementación de la función en Flask para obtener los datos de la base de datos en la API RESTful

En el código de la Figura 29 se muestra un fragmento de código del servicio RESTful usando Flask. En la figura se puede observar cómo se define la primera cabecera `/getbyuser` y el modo de acceso a la misma. Esta cabecera representa la implementación de uno de los métodos implementados en nuestra API. A su vez, este método hace uso de los parámetros `table` y `user`, cuyas instancias se obtienen desde la petición, a través de la URL, y mediante el uso del objeto `request`. A continuación, se construye un cursor hacia la conexión establecida, y se compone la cadena de texto con el código SQL de la consulta. Tras esto, se ejecutará la consulta, cuyo será convertido en un objeto tipo JSON que será devuelto como parte de la respuesta. Finalmente se cierra la conexión establecida y en el caso de que haya algún error imprime el código de error.

Aunque en la Figura 27 se muestra un ejemplo de una función de consulta, se han implementado otras funciones en las que también se almacenan y se borran datos como se muestra en la Figura 30.

```
async function insertFarmaForm(values: number[]) {
  const url = "http://192.168.1.222:5006/addselecciones_farma";
  await fetch(url, {
    method: 'POST',
    headers: {
      Accept: 'application/json',
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      values: values,
    })
  });
}

async function borrarSelFarma(value: number) {
  const url = "http://192.168.1.222:5006/deletesel_far/"+value;
  await fetch(url, {
    method: 'DELETE',
    headers: {
      Accept: 'application/json',
      'Content-Type': 'application/json'
    }
  });
}
```

Figura 30. Ejemplo de peticiones asíncronas de escritura y borrado de datos mediante los métodos POST y DELETE.

4.2.3 Implementación de los formularios

La implementación de los formularios ha requerido a la base de datos ser diseñada de manera que permita representar las dependencias que existen entre las diferentes cuestiones de los formularios como se mostró previamente en la Figura 4. Existen otras dependencias del tipo, una cuestión se mostrará únicamente si se selecciona una opción determinada en la pregunta anterior.

Esto ha requerido modelar la información de los formularios de tal manera que se pueda explotar esta información estructurada como parte de la aplicación móvil y su interfaz gráfica. Se ha planificado de tal manera en la que hay 3 pantallas, siendo cada pantalla un nivel de dependencia. Por ejemplo, si un usuario selecciona un problema de equilibrio, se deben codificar una serie de síntomas asociados con él como la dificultad para cambiar de peso de una pierna a otra, o el dar pasos cortos. A su vez, estos síntomas tienen asociados una serie ejercicios, agrupados en un bloque, que permiten mejorar dicho

problema. En nuestra base de datos hemos modelado estas relaciones definiendo una jerarquía con en tres niveles de dependencia. El primer nivel representará el problema, el segundo nivel los síntomas y el tercer nivel los bloques de ejercicios. En la Figura 31 se pueden observar las relaciones entre las diferentes preguntas y opciones del formulario de fisioterapia.

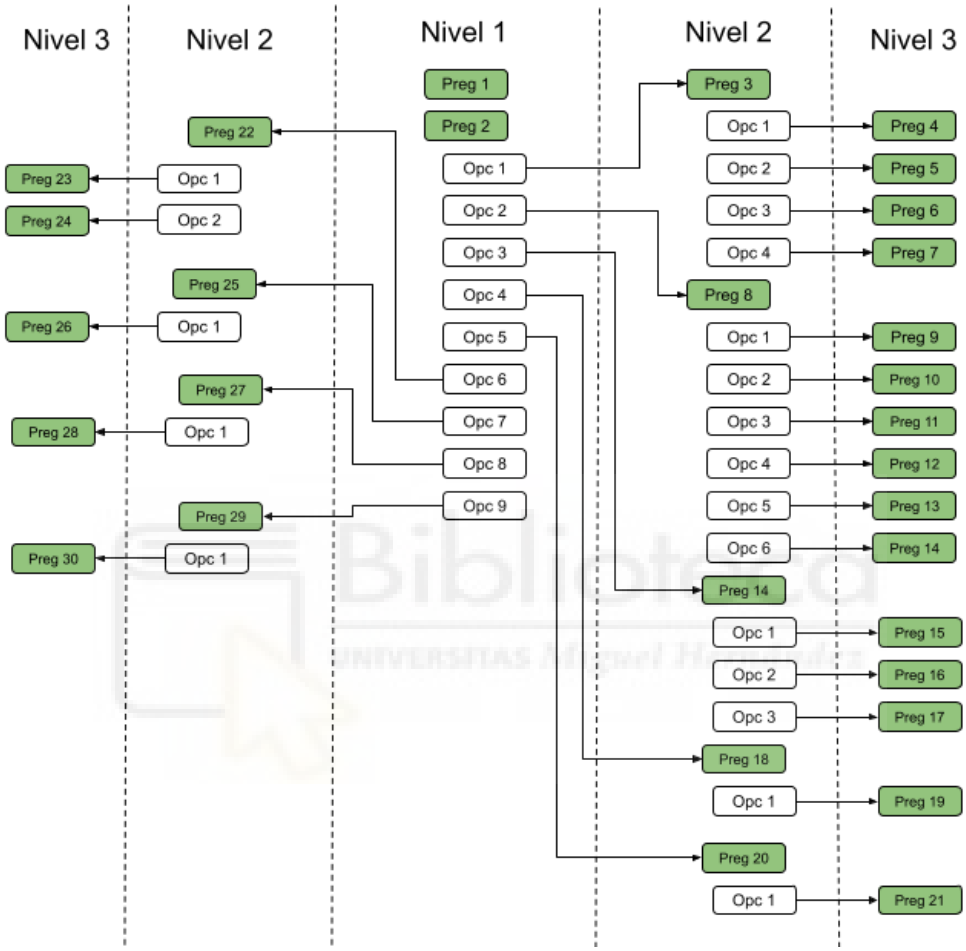


Figura 31. Esquema dependencias pregunta/opción para el formulario de fisioterapia

Además, en algunas de las preguntas existe funcionalidad específica que hemos tenido que contemplar e implementar para que se representen en la vista de forma adecuada. Por ejemplo, como hemos mencionado anteriormente, en el formulario de farmacología hay preguntas en las que se debe asistir al usuario. Por ejemplo, en la Figura 32, se muestra el algoritmo que utilizamos para mostrar los fármacos asociados a un tratamiento que ha seleccionado el usuario, es decir, tras introducir o seleccionar un tratamiento no muestra el nombre comercial del fármaco sino la pauta posológica, ya que, la base de datos está diseñada de tal manera que los fármacos con su respectivo nombre comercial tienen

asociados una serie de pautas posológicas. Para mostrar las pautas posológicas de cada tratamiento se ha diseñado un algoritmo en el que se identifican esas dependencias y se muestra la información mencionada. Tras la selección y el cálculo de las dependencias se muestra esa información en una alerta para informar al usuario de los componentes del tratamiento.

```
const onSelectedItemsChange = (selectedItems) => {
  setSelectedItems(selectedItems);
  for(var i=0; i < selectedItems.length; i++){
    if(i+1 == selectedItems.length && num_selecciones <= selectedItems.length){
      const nom_trat = itemsDW1[selectedItems[i]-1]["name"];
      var z=0;
      const aux: string[] = [];

      for(var x=0; x < farmacos_asoc.length && farmacos_asoc[x]["nom_trat"] != null; x++){
        if(nom_trat == farmacos_asoc[x]["nom_trat"] ){
          aux[z] = farmacos_asoc[x]["nom_pa"]
          z++;
        }
      }

      Alert.alert("ADVERTENCIA",
      "El tratamiento [ "+ tratamientos[i]["name"] + " ] está compuesto por los siguientes fármacos:" + "\n" + "[ " + aux + " ]"
      );
    }
  }
  setNumSelecciones(selectedItems.length);
};
```

Figura 32. Algoritmo para mostrar fármacos asociados al tratamiento

4.2.4 Algoritmos para la creación del plan personalizado

Esta es una de las partes más importantes de la aplicación. Ninguna de las aplicaciones relacionadas analizadas en la sección 3.1 ofrecen al usuario la posibilidad de calcular y realizar el seguimiento de un plan personalizado que combinen tratamientos combinados con fármacos y ejercicios. Por ello, en este algoritmo deberemos tener en cuenta la información registrada por los formularios interactivos de fisioterapia y farmacología.

Como parte del formulario de fisioterapia se le ofrece al usuario un conjunto de ejercicios que formarán parte del tratamiento personalizado. Como se mostró en la Figura 3 los ejercicios tienen asociados una estimación del tiempo requerido para realizarlos y las frecuencia diarias y semanales. Como parte del tratamiento, en el formulario se deberá interactuar con el paciente de manera que seleccione ejercicios acordes a los recomendados por nuestra aplicación. Concretamente, en la primera cuestión del formulario de fisioterapia se preguntará al usuario por la frecuencia de realización de ejercicios, y dinámicamente se calculará un rango mínimo y máximo de tiempo total, el cual, será tenido en cuenta para guiar el proceso de selección de ejercicios por parte del usuario hasta que se alcance este tiempo total.

La interacción entre el usuario y los ejercicios se ha modelado como una de las preguntas del formulario en la que dinámicamente ejecutaremos el siguiente algoritmo que guíe al usuario en la selección de estos hasta alcanzar las restricciones de tiempo anteriormente mencionadas:

1. **Tiempo total de ejercicios seleccionado:** tras seleccionar los ejercicios se realiza el cálculo del tiempo total escogido para posteriormente compararlo con el tiempo total de la frecuencia del plan escogida y se inserta la programación por días dentro de la BBDD.
2. **Creación de plan diario:** para realizar este cálculo hay que tener bastantes factores a cuenta y preparar los valores de la frecuencia de cada ejercicio.

Una vez realizados los pasos anteriores, se utilizará la información seleccionada para realizar los cálculos necesarios que permitan establecer una programación diaria:

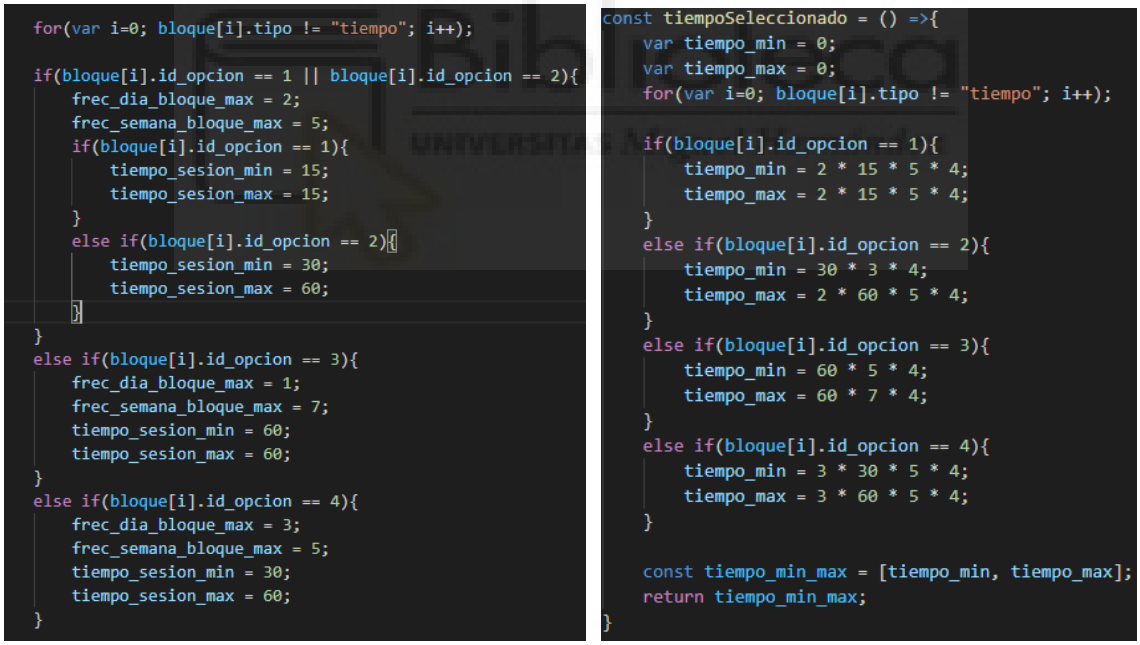
3. **Algoritmo de programación diaria:** consiste en ir acumulando el tiempo de cada ejercicio e ir apilando los ejercicios para cada día sin superar la frecuencia máxima de dicho ejercicio. Por ejemplo, si la frecuencia diaria seleccionada es de 60 minutos, cada vez que la suma de los ejercicios supere los 60 minutos se incrementa el día y se insertan los ejercicios correspondientes al día en la programación. Existen algunos ejercicios que no disponen de frecuencia diaria o tiempo, sino del número de repeticiones; para esos casos se definieron valores por defecto ya que suelen ser ejercicios de estiramiento o por el estilo.

El procedimiento anterior deberá completarse con los pasos necesarios para calcular no solo la programación diaria, si no la **programación distribuida a lo largo de la semana** en caso de que los ejercicios seleccionados no puedan programarse en un único día. Para ello se iterará sobre la lista de ejercicios seleccionados realizando las siguientes acciones:

1. Establecer el tiempo del ejercicio.
2. Multiplicar el tiempo del ejercicio por la frecuencia máxima.
3. Incrementar el día en el caso de que el tiempo acumulado entre el número de días calculados sea superior al tiempo máximo del plan seleccionado multiplicado por el número de repeticiones máximas del plan seleccionado.

4. Multiplicar el tiempo del día de la iteración por la frecuencia semanal del ejercicio mientras que no supere la frecuencia semanal del plan.
5. Acumular el tiempo de cada iteración para determinar el tiempo semanal.
6. Insertar el ejercicio del plan al día correspondiente en la base de datos.
7. Finalmente, cuando termina el bucle multiplicar por 4 el tiempo total calculado para una semana para obtener el tiempo total mensual.

En la Figura 33 y Figura 34 se muestran a modo de ejemplo fragmentos de la implementación realizada de los algoritmos anteriormente descritos. Como se puede observar, estos algoritmos únicamente realizan una distribución de los ejercicios por días, pero en el plan personalizado del ejemplo de la Figura 6 se puede observar que también deberá tenerse en cuenta la medicación y el tratamiento (seleccionado en el formulario de farmacología) ya que esto incluirá a la hora de determinar los vectores de programación horaria.



```
for(var i=0; bloque[i].tipo != "tiempo"; i++);

if(bloque[i].id_opcion == 1 || bloque[i].id_opcion == 2){
    frec_dia_bloque_max = 2;
    frec_semana_bloque_max = 5;
    if(bloque[i].id_opcion == 1){
        tiempo_sesion_min = 15;
        tiempo_sesion_max = 15;
    }
    else if(bloque[i].id_opcion == 2){
        tiempo_sesion_min = 30;
        tiempo_sesion_max = 60;
    }
}
else if(bloque[i].id_opcion == 3){
    frec_dia_bloque_max = 1;
    frec_semana_bloque_max = 7;
    tiempo_sesion_min = 60;
    tiempo_sesion_max = 60;
}
else if(bloque[i].id_opcion == 4){
    frec_dia_bloque_max = 3;
    frec_semana_bloque_max = 5;
    tiempo_sesion_min = 30;
    tiempo_sesion_max = 60;
}
```

```
const tiempoSeleccionado = () =>{
    var tiempo_min = 0;
    var tiempo_max = 0;
    for(var i=0; bloque[i].tipo != "tiempo"; i++);

    if(bloque[i].id_opcion == 1){
        tiempo_min = 2 * 15 * 5 * 4;
        tiempo_max = 2 * 15 * 5 * 4;
    }
    else if(bloque[i].id_opcion == 2){
        tiempo_min = 30 * 3 * 4;
        tiempo_max = 2 * 60 * 5 * 4;
    }
    else if(bloque[i].id_opcion == 3){
        tiempo_min = 60 * 5 * 4;
        tiempo_max = 60 * 7 * 4;
    }
    else if(bloque[i].id_opcion == 4){
        tiempo_min = 3 * 30 * 5 * 4;
        tiempo_max = 3 * 60 * 5 * 4;
    }

    const tiempo_min_max = [tiempo_min, tiempo_max];
    return tiempo_min_max;
}
```

Figura 33. A la izquierda captura del fragmento de código utilizado para la preparación de valores según la opción seleccionada en la pregunta del formulario. A la derecha captura del fragmento de código utilizado para definir el rango del tiempo en un periodo de un mes mínimo y máximo.

```
query.default.getDataByUser('V_ejercicios_agenda', ENV_id_usuario).then(selecciones => {
    var dia = 1;

    for(var i=0; i < selecciones.length; i++){
        tiempo[i] = 0;
        if(selecciones[i].tiempo_sesion == null){
            tiempo[i] += 5;
        }
        else{
            tiempo[i] += selecciones[i].tiempo_sesion;
        }
        if(selecciones[i].frecuencia_dia != null){
            const aux = tiempo_acum_dia;
            if(selecciones[i].frecuencia_dia <= frec_dia_bloque_max){
                tiempo[i] = tiempo[i] * selecciones[i].frecuencia_dia;
                while(aux / dia > tiempo_sesion_max * frec_dia_bloque_max){
                    dia++;
                }
            }
            else{
                tiempo[i] = tiempo[i] * frec_dia_bloque_max;
                tiempo_acum_dia += tiempo[i];
                while(aux / dia > tiempo_sesion_max * frec_dia_bloque_max){
                    dia++;
                }
            }
        }
        if(selecciones[i].frecuencia_semana != null){
            if(selecciones[i].frecuencia_semana <= frec_semana_bloque_max){
                tiempo[i] = tiempo[i] * selecciones[i].frecuencia_semana;
            }
            else{
                tiempo[i] = tiempo[i] * frec_semana_bloque_max;
            }
        }
        tiempo_acum_dia += tiempo[i];
        query.default.insertPlan([selecciones[i].id_opcion, ENV_id_usuario, dia, "ejercicios", getCurrentDateplus(dia)]);
    }
    setTT(tiempo_acum_dia*4);
    forceRender({});
});
```

Figura 34. Código del algoritmo para realizar la asignación de un día del plan personalizado en función de los ejercicios seleccionados por el usuario

Se debe por tanto tener en cuenta lo siguiente:

- **Asignación horaria según tratamientos seleccionados:** como se puede observar en el código de la Figura 35, en cada iteración del bucle se seleccionan dos tratamientos y se solapan dependiendo del color asociado al índice de la iteración en un vector auxiliar para definir tres vectores uno para cada color representado en la Figura 6 siendo verde recomendado, naranja no recomendado, pero apto y azul no recomendado. En el caso de que hayan más de dos tratamientos se solapan los valores partiendo de las asignaciones de los tres vectores definidos en la primera iteración.

  Después se construye un único array, asignando en cada índice del mismo, un color dependiendo del orden, siendo la prioridad en el siguiente orden: naranja, azul y verde. Por ejemplo, si los tres vectores en el índice uno tiene el valor uno, tiene que aparecer naranja, en el caso de que haya azul y verde tiene que persistir el naranja y si solamente está el verde se almacena en el índice el color verde.

Por último, y como se muestra en la Figura 36, por cada tratamiento seleccionado en el formulario se buscan sus fármacos asociados y por cada fármaco se insertan en la base de datos, quedando por tanto calculado y registrado el plan personalizado que posteriormente se le mostrará al usuario a modo de calendario para que le sea fácil seguirlo. El resultado almacenado se puede observar en la **Figura 37**.

```
if(tratamientos != null && tratamientos != undefined){
  var azul:number[] = [0];
  var naranja:number[] = [0];
  for(var i=0; i < selectedItems.length; i++){
    for(var x=0; x<24; x++){
      naranja[x] = 0;
      if(i==0){
        if(tratamientos[i]["azul"][x] == "1" && tratamientos[i+1]["azul"][x] == "1" || tratamientos[i]["azul"][x] == "1" && tratamientos[i+1]["azul"][x] == "0"
          || tratamientos[i]["azul"][x] == "0" && tratamientos[i+1]["azul"][x] == "1"){
          azul[x] = 1;
          if(tratamientos[i]["verde"][x] == "1" || tratamientos[i+1]["verde"][x] == "1") naranja[x] = 1
        }
        else azul[x] = 0;
      }
      else{
        if(tratamientos[i+1]["azul"][x] == "1" && azul[x] == 1 || tratamientos[i+1]["azul"][x] == "1" && azul[x] == 0 ||  tratamientos[i+1]["azul"][x] == "0" && azul[x] == 1){
          azul[x] = 1;
          if(tratamientos[i]["verde"][x] == "1" || tratamientos[i+1]["verde"][x] == "1") naranja[x] = 1
        }
        else azul[x] = 0;
      }
    }
  }
```

Figura 35. Algoritmo para solapar los índices de horarios de los tratamientos en dos vectores auxiliares

```
const arr_plan:string[] = [""];
for(var i = 0; i < 24; i++){
  if(naranja[i] == 1){
    arr_plan[i] = "n";
  }
  else if(azul[i] == 1){
    arr_plan[i] = "a";
  }
  else arr_plan[i] = "v";
}

for(var i=0; i < selectedItems.length; i++){
  var nom_trat: any = farmacos_asoc.find((fa:any) => fa.id == selectedItems[i]);
  farmacos_asoc.map((value) =>{
    if(value["nom_trat"] == nom_trat["nom_trat"]){
      query.default.insertPlanFarmacia([ENV_id_usuario.toString(), value["id"], arr_plan.toString()]);
    }
  })
}
```

Figura 36. Algoritmo para apilar los resultados obtenidos del algoritmo de la Figura 35 mediante un orden de prioridad

name	horario	pautas_posologicas	pa_name	med_name	id_usuario
trat1	a,a,a,a,a,a,a,a,v,v,v,v,v,n,n,n,v,n,n,n,v,n,a,a	00000001000001000000000001	ACARBOSA	Acebutolol	14
trat1	a,a,a,a,a,a,a,a,v,v,v,v,v,n,n,n,v,n,n,n,v,n,a,a	00000001000001000000000001	ACARBOSA	Acebutolol	10
trat2	a,a,a,a,a,a,a,a,v,v,v,v,v,n,n,n,v,n,n,n,v,n,a,a	00000001000001000000000001	ACETILSALICILICO	Acebutolol	14
trat2	a,a,a,a,a,a,a,a,v,v,v,v,v,n,n,n,v,n,n,n,v,n,a,a	00000001000001000000000001	ACETILSALICILICO	Acebutolol	10
trat3	a,a,a,a,a,a,a,a,v,v,v,v,v,n,n,n,v,n,n,n,v,n,a,a	00000001000001000000000001	ALOGLIPTINA	Acebutolol	14
trat1	a,a,a,a,a,a,a,a,v,v,v,v,v,n,n,n,v,n,n,n,v,n,a,a	00000001000000000000010000	ÁCIDO ACETILSALICÍTICO	Aspirina	14
trat1	a,a,a,a,a,a,a,a,v,v,v,v,v,n,n,n,v,n,n,n,v,n,a,a	00000001000000000000010000	ÁCIDO ACETILSALICÍTICO	Aspirina	10

Figura 37. Datos resultantes del cálculo del plan de fármacos

4.2.5 Seguimiento del plan a través del calendario

Otro de los restos de este proyecto ha sido disponer la información manera que sea fácilmente entendible por el usuario a través de la aplicación desarrollada. Por ejemplo, una vez conseguida automatización para calcular los planes se nos presentó el reto de representar la información del plan de forma fácilmente accesible al usuario. Esta necesidad fue plasmada, por ejemplo, en los prototipos de la Figura 19 en la que se proponía mostrar la información del plan asociada a un calendario, de manera que el usuario tenga visible y filtrado los fármacos que debe ingerir y los ejercicios que debe realizar en un día concreto, y por defecto se mostrará el día actual.

- **Carga de los ejercicios en el calendario:** para cargar los ejercicios en la vista del calendario requiere preparar un objeto con los datos necesarios provenientes del plan previamente registrado en la base de datos y establecer un tamaño para el contenedor en el que se va a mostrar, es decir, la anchura del área dónde se ubica el texto y las imágenes relacionadas al ejercicio cargado.


```
useEffect(() => {

  for(var i=0; i < ejercicios.length; i++){
    const strTime = timeToString(ejercicios[i]["fecha"]);
    if(!items[strTime]){
      items[strTime] = [];
      for(var x=0; x < ejercicios.length; x++){
        let ejers: any = ejercicios[x];
        items[strTime].push({
          name: ejers.name,
          completado: ejers.completado,
          id: ejers.id_opcion,
          fecha: strTime,
          tipo: "ejercicio",
          height: 190,
          duracion: ejers.tiempo_sesion,
        });
      }
```

Figura 38. Fragmento de código donde se muestra la carga de los ejercicios en el calendario

En el código de la Figura 38 se puede observar que, por cada ejercicio existente, mientras que el objeto mencionado para una fecha no esté vacío se cargan los datos. Se ha implementado así para poder mostrar el plan duplicado según su frecuencia (un día, semanal o mensualmente) ya que, si el usuario selecciona por

ejemplo 10 ejercicios, el total del tiempo podría ser acumulado en dos días y se podrían rellenar los días restantes de la planificación usando el mismo patrón siempre y cuando la frecuencia semanal máxima lo permita. Como se puede observar en esta captura para implementar esta funcionalidad se hace uso del Hook `useEffect`.

- **Carga de los fármacos en el calendario:** para cargar los fármacos se sigue un planteamiento similar al de los ejercicios. No obstante, se requiere de código adicional para disponer en el calendario el horario asociado con el consumo de cada fármaco. Por ejemplo, en el fragmento de código mostrado en la Figura 39 se puede observar que, partiendo de la programación horaria binaria que se puede observar en la columna `"pautas_posologicas"` de la Figura 37, cuando el valor es igual a uno de cada índice del vector se concatena en una cadena de texto relacionada a la hora recomendada y se cargan los datos en el objeto.

```
var num_horas = 0;
for(var z=0; z < farmacos.length; z++){
  var farms:any = farmacos[z];
  var hora: string[] = [];
  var hora_aux = "";
  for(var y=0; y < farms.pautas_posologicas.length; y++){
    if(farms.pautas_posologicas[y] == 1){
      var aux = y+1;
        hora_aux = " " +aux+":00";
    }
    hora[num_horas] = hora_aux;
    num_horas++;
  }

  const uniqueHoras = [...new Set(hora)];
  var ind: any;
  for(var n=1; n < uniqueHoras.toString().length; n++){
    ind = uniqueHoras.toString()[n];
    if(!isNaN(ind)) break;
  }

  items[strTime].push({
    cabecera: uniqueHoras.toString().slice(n),
    name: farms.pa_name,
    height: 200,
    completado: false,
    tipo: "medicamento",
    fecha: strTime
  });
}
```

Figura 39. Fragmento de código donde se muestra la carga de los fármacos en el calendario

4.2.6 Otros aspectos relevantes

Además de los aspectos destacados anteriormente, el uso de las tecnologías React Native y la programación mediante TypeScript, me ha permitido aprender conceptos muy útiles que me han permitido profundizar en diferentes aspectos de desarrollo. Trabajar con estas tecnologías implica una serie de peculiaridades a las que hay que amoldase a la hora de programar, y a continuación destaco algunas de las que más han llamado mi atención:

- **Parametrización del contenido de las vistas según el usuario:** la información cargada en las pantallas mediante las peticiones a la API dependerá del id del usuario que ha iniciado sesión. Cuando se navega a una pantalla se genera una consulta de todos los datos relacionados al usuario. Por ejemplo, algunos componentes como los contenedores de texto de la pantalla de inicio son inicializados vacíos y al iniciar sesión y realizar la llamada asíncrona de la consulta de datos, estos campos se actualizan con la información del usuario en cuestión.

- **Sistema de navegación y listener focus:** comprender el sistema de navegación ha sido un gran reto ya que fue una de las partes de desarrollo que se presentaron en las primeras fases de desarrollo, en las cuales, no dominaba aún las tecnologías ni los lenguajes de programación utilizados. Además, durante la evolución del desarrollo este sistema ha ido evolucionando conforme iban apareciendo más pantallas y por lo tanto aumentaba la complejidad de este.

  El sistema de navegación propuesto por estas herramientas contiene tres tipos de contenedores de navegación distintos que proporcionan funcionalidades variadas. En el esquema de la Figura 40 se pueden observar tres elementos separados y relacionados con flechas. Los rectángulos con el fondo en blanco representan a las pantallas, los rectángulos azulados representan a los contenedores normales, el elemento verde representa al contenedor del menú navegacional inferior y la figura magenta representa al contenedor del menú desplegable.

  A su vez, el listener focus es una pieza clave para actualizar datos en tiempo real. Es una función de navegación para ejecutar ciertas acciones cada vez que se navega de una pantalla a otra. Es totalmente necesaria ya que, si los datos han sido

modificados en otras pantallas, los cuales son usados en la pantalla que se ha navegado, es necesario actualizarlos para mostrar la información actual y con las herramientas anteriores no se dispone de esa funcionalidad. En el ejemplo de la Figura 41 se muestra que tras la finalización del montaje de todos los componentes en el caso de que se haya navegado a la pantalla se realiza de nuevo una consulta a la base de datos para reasignar los valores en el caso de que hayan sido actualizados.

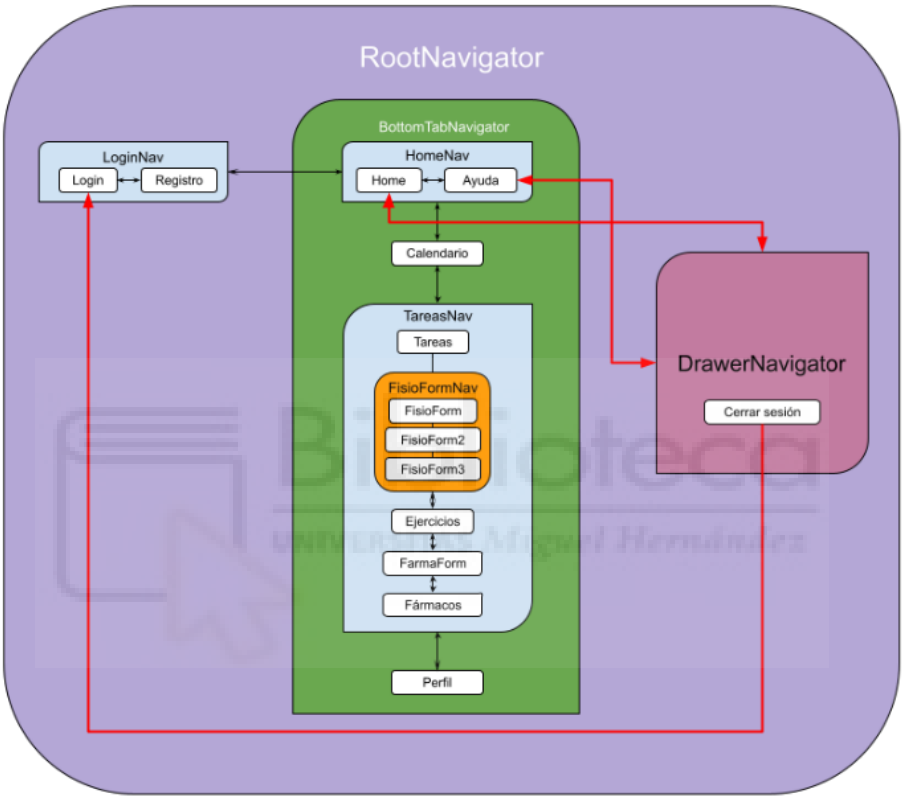


Figura 40. Esquema de navegación.

```
useEffect(() => {
  navigation.addListener('focus', () => {
    query.default.getData2('info_inicio').then(data => {
        setInicio(data);
    })
    query.default.getUser(ENV_usuario).then(data => {
      setAceptado(data[0].recom_contra);
    });
})

}, []);
```

Figura 41. Ejemplo uso listener focus.

- **Gestión de llamadas asíncronas y temporizadores:** las funciones asíncronas suelen ocasionar la ejecución desorganizada de peticiones al servidor. Debido a ello se presentan problemas de incongruencias de datos ya que, por ejemplo, si se realiza una eliminación del plan anterior mientras se está insertando los datos del nuevo plan, puede ocasionar que en el plan nuevo solamente se inserten la mitad de los datos. La solución ha sido poner una espera de un tiempo en el parámetro de la función siendo el valor en milisegundos para que se ejecuten primero unas instrucciones y luego otras junto al hook de `useRef` que se puede observar, por ejemplo, en la Figura 26

- **Interfaces:** las interfaces sirven para estructurar los datos declarando los valores por los que va a estar compuesto y su formato. En algunos casos su uso ha sido muy útil para poder manejar estructuras complejas de datos, sobre todo en las fases tempranas del aprendizaje de TypeScript, ya que facilita el almacenamiento directo desde el objeto al Hook `setState`. Pero conforme fui aumentando mis conocimientos dejé de realizar esa práctica ya que me manejaba con los tipos de datos y sus respectivas asignaciones. Por ejemplo, la interfaz de la Figura 42 sirve para definir el formato de datos provenientes de los fármacos introducidos en el formulario de farmacia.

```
interface InputsFarma {
  id: number,
  farmaco: string,
  tipo: string,
  id_usuario: number
}
```

Figura 42. Ejemplo de interfaz para representar información de los fármacos.

- **Screens:** las pantallas (denominadas Screens en inglés) son los ficheros de código donde se sitúa la estructura y el funcionamiento de cada elemento de navegación. Existen diversas formas para programar el código relacionado a ellas pudiendo ser las pantallas declaradas como funciones, constantes o clases. Antes de modificar mi proyecto para que soporte el lenguaje de TypeScript, principalmente utilizaba la declaración de las pantallas como clases, aunque había muchas referencias de código en pantallas declaradas como funciones. Tras investigar

observé que estas se usaban en versiones anteriores de la tecnología y es una práctica antiguada, por lo que dejé de utilizarlas. Finalmente, tras descubrir TypeScript, empecé a utilizar las pantallas como constantes ya que en la actualidad es lo forma de proceder más frecuente.

- **Creación de una librería de funciones:** he creado mi propia librería de funciones de utilidad denominada "`utils.ts`". Por ejemplo, en ella se encuentran todas las consultas necesarias para la obtener los datos de la base de datos. También se han incluido otras funciones de interpretación de texto o condicionales para generar mensajes de error a la hora de insertar texto en los formularios o la pantalla de inicio de sesión.

```
export const emailValidator = (email: string) => {
  const re = /\S+@\S+\.\S+/;

  if (!email || email.length <= 0) return 'La dirección de correo no puede estar vacía';
  if (!re.test(email)) return 'Por favor, introduzca una dirección de correo válida';
  return '';
};
```

Figura 43. Ejemplo de validador para los campos de tipo mail del formulario de registro

```
const onItemPress = (item) => {
  var color:string = "white"
  if(item.completado == true) color = "lightgreen";
  var aux = items;
  if(item.tipo == "ejercicio"){
    return(
      <TouchableOpacity
        style={[styles.item, {height: item.height, backgroundColor:color, padding:"5%"}]}
        onPress={() => {

              Object.keys(aux[item.fecha]).map((key) => (
                Object.keys(aux[item.fecha][key]).map((key2) => (
                  key2 == "id" && aux[item.fecha][key][key2] == item.id ? (
                    aux[item.fecha][key]["completado"] = !aux[item.fecha][key]["completado"],
                    query.default.updateCompletadoEj(item.completado, item.id, ENV_id_usuario)
                  ): null
                  ))
              ))
              forceRender({});
        }}
        onLongPress={() => {
          setModalVisible({show: true, id: item.id});
          forceRender({});
        }}
      >
```

Figura 44. Ejemplo de código que gestiona los eventos necesarios para que el usuario pueda marcar las distintas tareas de un plan como completadas.

- **Interactividad en la vista a través de mensajes de error de inputs:** es muy importante mostrar mensajes de error para que el usuario puede interactuar con la aplicación correctamente.

  Para ello se han diseñado mensajes de error de tipo *alert* o texto de color rojo que aparece debajo de los componentes. Por ejemplo, en la función de la Figura 43 se implementa el validador del correo electrónico. Se encarga de informar al usuario si el formato del correo electrónico es erróneo o si está vacío.

  Otro aspecto en el que se destacan los aspectos interactivos es a la hora de implementar la pantalla del calendario (ver ejemplo en la sección 4.3.7). Por ejemplo, se ha debido gestionar el evento de clic de los distintos contenedores que la forman para actualizar el estado de las tareas del plan, y controlar los eventos de carga para inicializarlas con los datos de la BBDD. Por ejemplo, en el código de la Figura 44 se puede observar cómo al dispararse el evento de clic, se itera por los distintos niveles del objeto y mediante el código identificador se cambia el color del contenedor en cuestión, y se actualiza inmediatamente en la base de datos para que los datos persistan en el caso de que el usuario decida navegar hacia otra pantalla. También se puede observar cómo se incluye una función para detectar un evento de clic largo para mostrar una ventana modal con la información detallada del ejercicio asociado al contenedor.

  Otro ejemplo de interactividad estaría relacionado con la funcionalidad de cambio de contraseña. Se incluye los *placeholders* informativos (ver ejemplo en Figura 45) y el proceso es asistido para comprobar que la nueva contraseña no coincide con la anterior y de que la repetición de la contraseña coincide con la previamente insertada.

  También se comprueba de que los campos no estén vacíos y en el caso de que todo se cumpla se actualiza el estado de los componentes de inserción de texto para que si se vuelve a desplegar el formulario de cambio de contraseña aparezca vacío.

4.3 Ejemplo de uso de la aplicación

En esta sección se describe un ejemplo de uso de la aplicación resultante. En ella se pude apreciar la similitud respecto a los prototipos iniciales, aunque en esta última versión se pueden apreciar ciertos cambios fruto de la evolución del proyecto y sus requisitos.

Figura 45. Ejemplo de las pantallas de login y registro

4.3.1 Login y registro de usuarios

Como se muestra en la Figura 45, al iniciar la aplicación el usuario se encontrará con la pantalla inicial desde la que iniciar sesión o registrarse. Ambas pantallas se encuentran enlazadas en el mismo contenedor de navegación.

Cuando se introducen los datos solicitados y hacer clic se realiza una consulta a la base de datos y si son correctos se procede a la navegación a la página principal, en caso contrario aparece un mensaje informando al usuario del tipo de error (ver Figura 46).

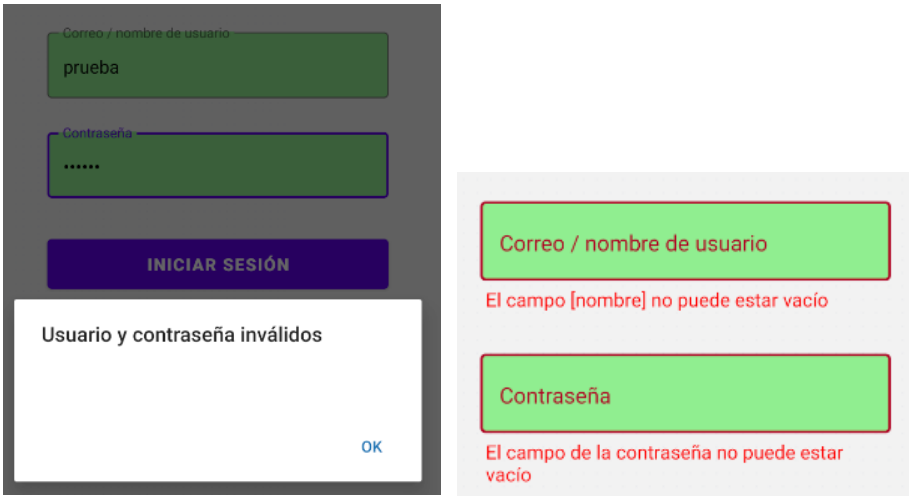


Figura 46. Ejemplo de mensajes informando al usuario de posibles errores al completar los formularios de inicio de sesión.

Figura 47. Captura de pantalla de la página principal de la aplicación (Home).

4.3.2 **Página principal (Home)**

Si el inicio de sesión es correcto, se mostrará al usuario la pantalla principal o Home. En ella se muestran una serie de recomendaciones y contraindicaciones para informar al usuario de todos los aspectos necesarios que debe de saber respecto a los tratamientos y fármacos. Tras leer y desplazarse hacia la parte inferior de la pantalla aparece un botón para confirmar que ha sido leída esa información. Una vez aceptado, clicando en el botón, la pantalla de inicio se cambia por un texto por defecto mostrando otra información útil para el usuario con el mismo formato.

Figura 48. Captura de pantalla de la página desde donde se gestionan las diferentes tareas que puede realizar el usuario. Por ejemplo, acceder a los formularios de fisioterapia y farmacología.

4.3.3 **Página de tareas**

Esta pantalla (ver Figura 48) consiste en informar al usuario del estado de las acciones que tiene que realizar o las que hayan sido realizadas. Como hemos comentado, para que se pueda generar un plan personalizado el usuario deberá haber interactuando como

mínimo una primera vez con los formularios para registrar su información. Además, si a lo largo del tiempo sus condiciones cambian podría editarlos y el plan personalizado se adaptaría a la nueva situación de su enfermedad. Además, como se muestra en la Figura 48 derecha, en función de si los formularios han sido completados o no aparecen como completados o viceversa. También muestra el número de ejercicios seleccionados y no permite la selección de los ejercicios físicos hasta que el formulario de fisioterapia haya sido completado.

Figura 49. Ejemplo extraído del formulario de fisioterapia.

4.3.4 **Tarea para completar el formulario de fisioterapia**

Las pantallas relacionadas con el formulario de fisioterapia están compuestas por listas desplegables a excepción de la primera cuestión. Estas listas ofrecen al usuario la selección de múltiples opciones. Tras la selección aparecen las opciones seleccionadas debajo del componente.

Como se puede observar en el ejemplo de la Figura 49, aparecen cuestiones relacionadas a los problemas que puede tener el usuario; en la siguiente pantalla (nivel 2) se realizan preguntas relacionadas a esos problemas siendo las cuestiones más específicas en relación con el problema anterior. Finalmente, tras indicar los problemas específicos aparecen cuestiones relacionadas a bloques de ejercicios, para que el usuario seleccione el conjunto de ejercicios que le apetece realizar para posteriormente mostrar todos los ejercicios de esos bloques seleccionados en la pantalla del listado de ejercicios.

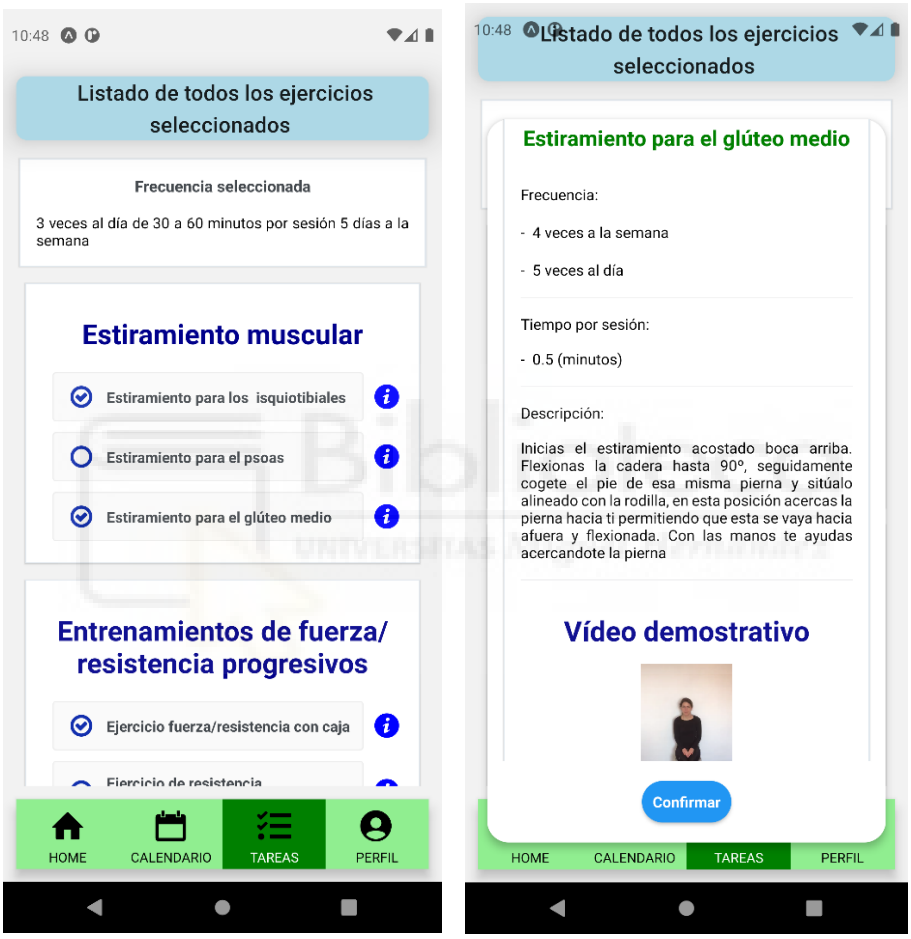


Figura 50. Ejemplo donde se muestra el listado de ejercicio seleccionados y sus detalles.

4.3.5 Tarea para seleccionar ejercicios del tratamiento

Desde esta pantalla se podrán seleccionar y consultar el conjunto de ejercicios que se realizarán como parte del tratamiento, así como información sobre cómo realizar los mismos. Como se puede observar en la Figura 50, en primer lugar, se muestra la frecuencia seleccionada en la primera pregunta del formulario de fisioterapia ya que es aspecto clave para poder guiar al usuario en la selección de estos. Después se muestran

listas seleccionables de ejercicios agrupados por los bloques ofrecidos por el formulario de fisioterapia. Además, junto a cada ejercicio se muestra un botón de información, el cual tras clicarlo muestra una ventana modal con la información detallada del ejercicio adjunto.

Una vez seleccionados todos los ejercicios requeridos por el plan, el usuario deberá clicar el botón de confirmación que hay en la parte posterior de la pantalla. Tras clicarlo aparecerá una ventana emergente solicitando la confirmación y tras aceptarla la aplicación calculará automáticamente el plan, organizando los ejercicios por días en función de la frecuencia seleccionada, y redirigirá al usuario a página del calendario. Merece la pena destacar que el cálculo del plan personalizado tendrá en cuenta si se dispone de información de uno o ambos formularios; por ejemplo, si el usuario no ha completado todavía el formulario de farmacia se calculará y se mostrará únicamente la programación relacionada con los ejercicios.

4.3.6 **Tarea para completar el formulario de farmacia**

Este formulario está compuesto por componentes desplegables simples, siendo los simples de selección única o complejos, los cuales, incluyen la opción de seleccionar múltiples opciones y un buscador para filtrar los fármacos o tratamientos con el fin de facilitar la selección.

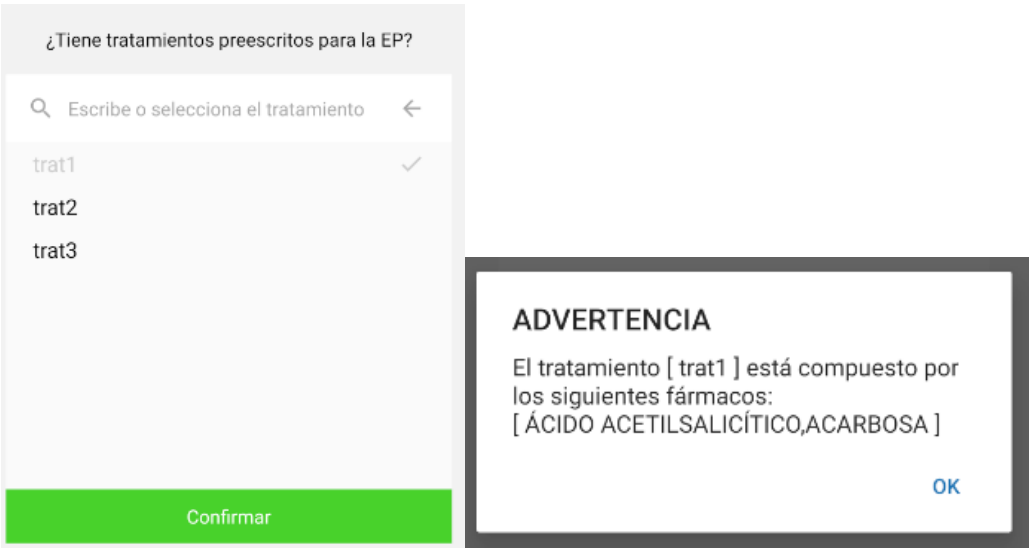


Figura 51. Ejemplo extraído del formulario de farmacia relacionado con la selección de tratamientos.

En primer lugar, el usuario debe de seleccionar los tratamientos que tiene prescritos por su médico y que se encontrarán almacenados en la BBDD. Por ejemplo, en la **Figura 51** izquierda se ofrecen al usuario los distintos tratamientos definidos por defecto, los cuales tendrán asociados una serie de fármacos. Además, en la información del tratamiento estará registrada parte de la información necesaria para calcular el plan de seguimiento explicado en distintas partes de este documento, en este caso, la programación horaria para la toma de esos fármacos. Cuando se selecciona un tratamiento, aparece una ventana emergente indicando los fármacos asociados, con su respectivo principio activo como se puede observar también en la parte derecha de la **Figura 51** .

En este formulario también existen otro tipo de cuestiones en las que el dominio de la respuesta serán los distontos medicamentos o principios activos. Por ejemplo, en la Figura 52 se muestra como ejemplo dos cuestiones en las que se requiere que el usuario especifique los medicamentos frente a los que representa ciertas alergias, así como sustancias que habitualmente ingiere. La gestión de los medicamentos se muestra un modal con un listado de fármacos (ver Figura 52 izquierda) a partir de la información registrada en nuestra base de datos. Sin embargo, en el caso de las sustancias, se registrarán como texto libre (ver Figura 52 derecha) ya que podrían ser de diversa índole.

Por último, otras preguntas tendrán como objetivo inferir a partir de las opciones seleccionadas si el usuario padece algún tipo de problema. Por ejemplo:

- **Demencia:** entre otras condiciones, el usuario debe haber seleccionado al menos dos de los síntomas relacionados con esta situación.
- **Depresión:** entre otras condiciones, el usuario debe haber seleccionado al menos cinco de los síntomas relacionados con esta situación.
- **Embarazo y lactancia:** en este caso se le pregunta al usuario por ambas situaciones a través de dos preguntas que pueden ser contestadas como sí o no.

Una vez finalizado el formulario, se mostrará una alerta determinada para cada condición informando de la posibilidad de la enfermedad o alguna advertencia. Por ejemplo, en la Figura 53 se muestra un ejemplo de una advertencia por deterioro cognitivo.

Figura 52. Ejemplo de la pantalla modal de sustancias y el modal de alergias.

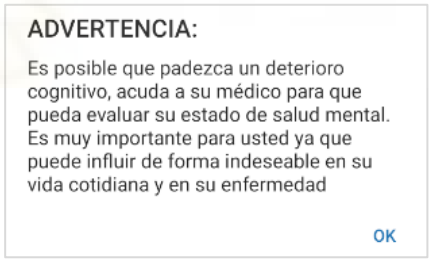


Figura 53. Ejemplo de advertencia de deterioro.

4.3.7 Calendario para el seguimiento del tratamiento

Como hemos comentado anteriormente, una de las fortalezas de la aplicación es: (1) en primer lugar ser capaces de calcular el plan de seguimiento personalizado a partir de la información obtenida a través de los formularios, y (2) mostrar esa información al usuario de manera práctica para que pueda realizar su seguimiento en su día a día. La pantalla del calendario se ha diseñado e implementado principalmente para conseguir el segundo punto indicado. Parte de esta funcionalidad se puede observar en la Figura 54.

En la parte izquierda se muestra la planificación de un día, se puede navegar en el listado mediante con el gesto de desplazamiento hacia abajo o arriba. Además, en la parte superior del calendario se incluye un desplegable (parte derecha de la figura) que ofrece la posibilidad de navegación entre los días del mes programado mediante un clic en el día deseado.

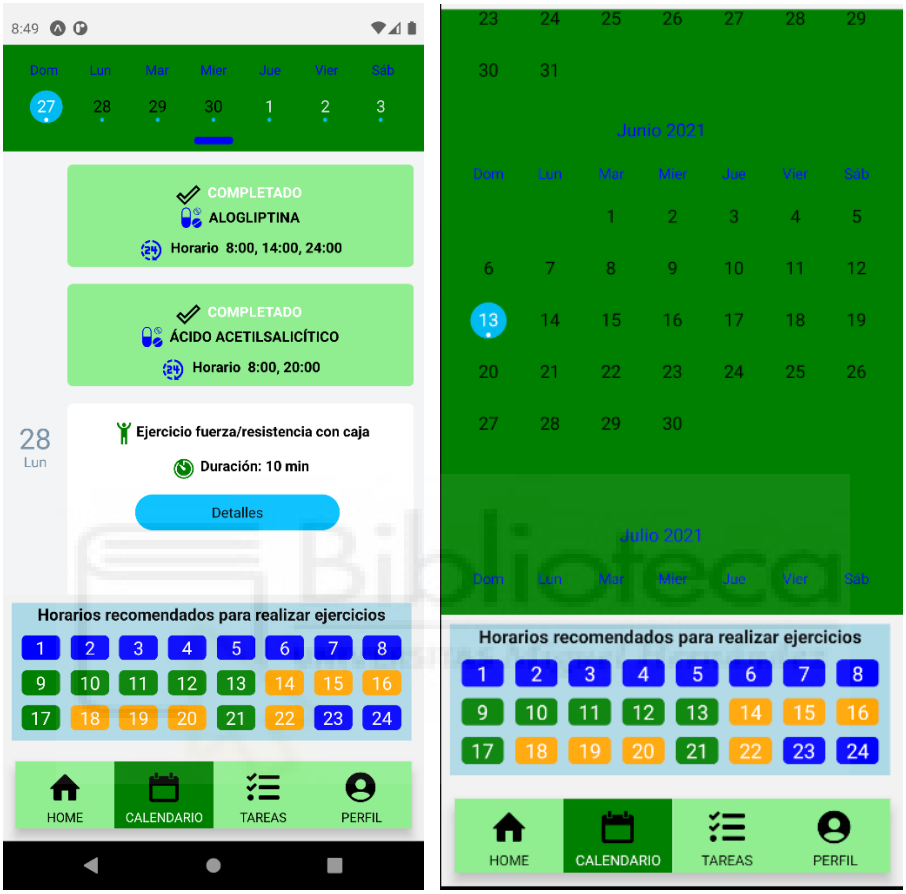


Figura 54. Captura de la pantalla del calendario. A la izquierda se muestra la vista diaria, y a la derecha se muestra la posibilidad de cambiar a otros días.

Siguiendo con la descripción de la Figura 54, la información mostrada contiene el listado de ejercicios y pautas posológicas programados diariamente y un horario calculado y distribuido por colores indicando las horas recomendadas para realizar ejercicios en función del tratamiento:

- **Verde**: está recomendado hacer el ejercicio en esta franja horaria
- **Naranja**: no es una franja horaria recomendada, pero en el caso de que se encuentre bien el usuario puede realizar los ejercicios.
- **Azul**: se puede hacer el ejercicio, pero no es la franja horaria ideal/recomendada.

A su vez, para cada ejercicio se mostrará su nombre y su duración, y para cada fármaco se mostrará su nombre y el horario en el que se deben de tomar. Además, se puede visualizar toda la información respectiva al ejercicio al realizar un clic largo, es decir, mantener clicado el elemento durante un segundo aproximadamente; tras realizar esa acción aparece una ventana modal similar a la mostrada anteriormente vistas de selección de ejercicios del formulario de fisioterapia.

Con el fin de facilitar la tarea de seguimiento por parte del paciente se puede clicar en el recuadro del fármaco o ejercicio y tras ello, se actualizará el estado de la tarea como completada y se representará en la vista con el cambio del color de fondo y la aparición de un título superior indicando de que ha sido completada.

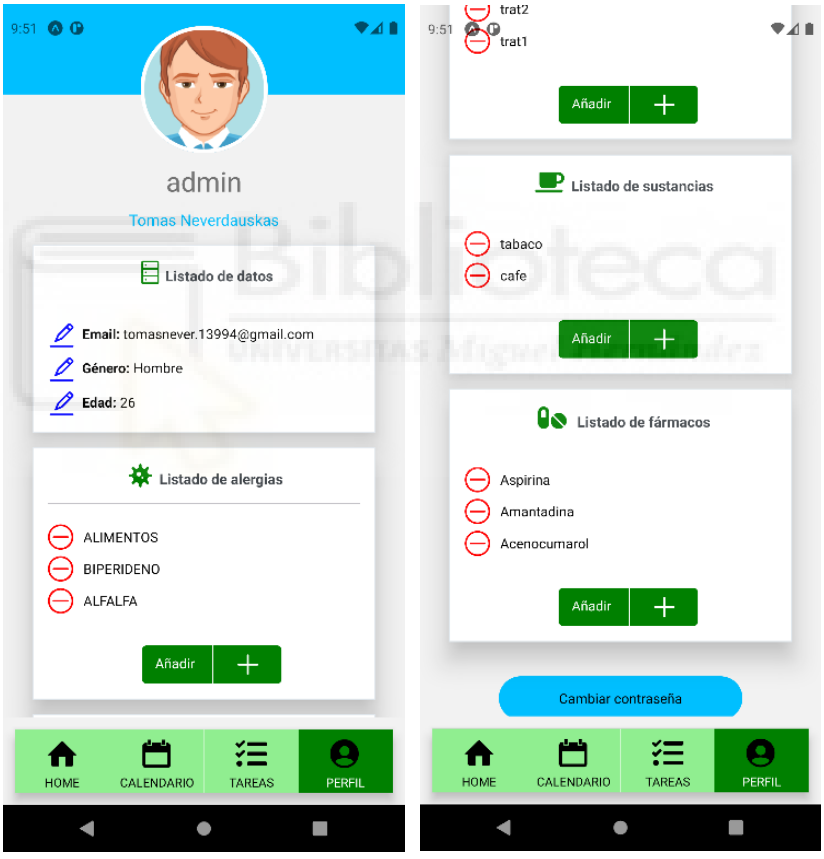


Figura 55. Pantalla de perfil.

4.3.8 **Información y manipulación del perfil**

En la aplicación juega un papel muy importante el perfil del usuario (ver Figura 55), ya que de él dependerá el resto del contenido mostrado al usuario. Además, de la información médica asociada y registrada a través de los formularios, también existirá

información personal básica que será especificada en el momento de crear la cuenta. Los datos personales del usuario pueden ser modificados, como podría ser el correo electrónico, contraseña, etc. Además, toda la información relacionada con los fármacos, sustancias y alergias puede ser también modificada (aunque el propio formulario también permite rectificar las inserciones de forma interactiva). Como se muestra en la Figura 56 modificar este tipo de información requiere la confirmación por parte del usuario.

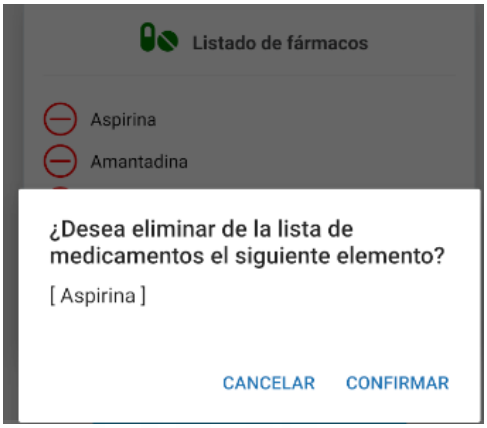


Figura 56. Ejemplo del mensaje de confirmación para eliminar un fármaco asociado a un perfil.

Finalmente, hay que destacar que para cerrar la sesión del usuario existe el navegador de menú lateral, el cual se puede abrir mediante un desplazamiento del dedo desde cualquier parte de la pantalla en el externo izquierdo o dando al botón de tres rayas en el pie de la pantalla de inicio (ver Figura 57).

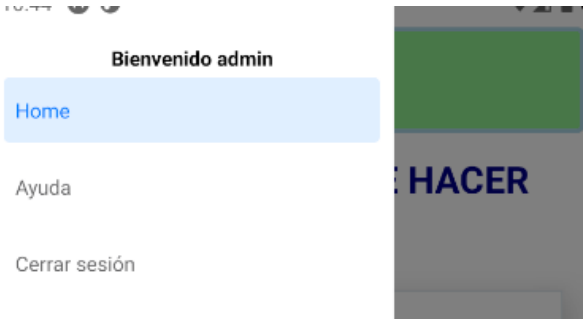


Figura 57. Captura del menú lateral desde el que se puede cerrar la sesión.

5. CONCLUSIONES Y TRABAJO FUTURO

El reto inicial propuesto al inicio de este trabajo pretendía contribuir a la adherencia a tratamientos de pacientes con Enfermedad de Parkison (EP), asistiendo a estos usuarios con herramientas y recursos accesibles a través de una aplicación informática interactiva que favorezca el seguimiento de planes personalizados combinando la ingestión de fármacos y la realización de ejercicios.

Una vez finalizado el proyecto, podemos concluir que se ha conseguido diseñar un prototipo funcional de una aplicación móvil para personas con EP que permite la integración de contenidos relacionados con los síntomas, la medicación, sus efectos y la creación de planes personalizados que permiten el seguimiento de tratamientos conservadores en los que se combinan ejercicios de fisioterapia y la ingesta de fármacos. Además, se diseñó un calendario semanal con la medicación y los ejercicios diarios recomendados que facilita el seguimiento del tratamiento a los usuarios.

El estudio de aplicaciones relacionadas nos permitió constatar que, pese a existir diferentes tipos de aplicaciones relacionadas con la adherencia a tratamientos y otras específicas destinadas a pacientes de EP, no existe ninguna específica en la que se integre información de tratamientos conservadores para pacientes con EP combinando tratamientos de fisioterapia y farmacia. Por ello, consideramos que la herramienta aquí desarrollada podría ser una contribución potencialmente interesante. Además, pese a no existir aplicaciones idénticas, el análisis de diferentes tipos de aplicaciones relacionadas nos ha permitido seleccionar aspectos de diseño comunes que hemos usado como referencia para diseñar nuestra aplicación.

Desde el punto de vista de las arquitecturas de desarrollo software, el análisis del estado del arte nos ha permitido seleccionar como tecnologías de desarrollo React Native, Expo y Node.js para la implementación de una aplicación híbrida usando principalmente el lenguaje TypeScript. También hemos implementado una base de datos en MySQL y la hemos dotado de acceso desarrollando una API REST en Python con Flask. Finalmente, hemos utilizado un entorno de desarrollo y pruebas que nos han permitido probar la aplicación desarrollada en sistemas Android.

Desde el punto de vista de la funcionalidad alcanzada por nuestro prototipo, la aplicación informática desarrollada permita: la gestión y edición de usuarios, integración

de dos formularios interactivos que facilitan al usuario su realización ya que las preguntas estarán condicionadas a las respuestas de cada uno de los participantes, integración de un calendario con agenda interactivo y dinámico, sistema de navegación con tres tipos de contenedores, control de errores, gran variedad de componentes, actualización inmediata de datos y objetos visuales para toda la información mostrada de forma interactiva dentro de la aplicación

Desde el punto de vista técnico, implementar una aplicación de este tipo ha supuesto un reto al enfrentarnos al diseño e implementación de un sistema de datos complejo compuesto por tres tecnologías y tecnologías de apoyo adicionales, diferentes lenguajes de programación, uso e integración de un sistema gestor de bases de datos con múltiples tablas y vistas, integración de un entorno en tiempo de ejecución multiplataforma, desarrollo de funciones, algoritmos, interfaces y múltiples tipos de variables de datos… A modo de resumen sobre los diferentes artefactos software alcanzados se han creado:

- 13 tablas de datos, 7 vistas y más de 100 inserciones de datos estáticas.
- 18 funciones que forma parte del API de acceso a los datos.
- 26 funciones para mandar peticiones desde la aplicación a la API.
- 7 contenedores de navegación interconectados mediante 14 pantallas.
- …

5.1 Trabajo futuro

Aunque hemos conseguido implementar un prototipo funcional que ofrece muchas funcionalidades, durante el desarrollo de este trabajo han aparecido nuevos retos y necesidades que hemos identificado como trabajo futuro que permitirán seguir con este proyecto a nivel empresarial.

- **Mejoras del algoritmo para el cálculo del plan personalizado:** modificar el algoritmo de planificación para que se adapte los estados del paciente ON/OFF. Esto requeriría una replanificación automática del plan de seguimiento cada vez que el usuario cambie el estado en el que se encuentra.

- **Mejoras de funcionalidad:** sería interesante implementar un sistema de notificaciones, enlazar el calendario de la App con Google Calendar implementar el reconocimiento por voz entre otras mejoras.

- **Mejorar la gestión de fármacos:** enlazar la aplicación con una base de datos real de todos los fármacos y principios activos relacionados con la enfermedad del Párkinson.

- **Aumentar las pruebas en entornos IOS y navegadores webs convencionales:** por ejemplo, al probar la versión web de la aplicación, se ha realizado la comprobación de la funcionalidad y efectivamente, funciona correctamente, aunque hay algunos aspectos visuales que aparecen más estirados y deben ser adaptados. Se debería comprobar este tipo de aspectos al desplegar la aplicación en sistemas diferentes dispositivos IOS.

- **Mejorar la generación y edición de contenido:** diseñar imágenes para algunos datos informativos para mejorar la interfaz y que sea más atractiva. Además, se deberá mantener actualizado todo el contenido relacionado con los diferentes tratamientos y ejercicios, así como los vídeos asociados a cada ejercicio. En la versión actual de la aplicación, la gestión de esta información se ha realizado estáticamente a través de sentencias SQL para manipular la BBDD, en la siguiente versión se debería implementar funcionalidad que permita gestionar toda esta información de una forma más amigable.

- **Desplegar y publicar la aplicación en las distintas tiendas:** actualmente los datos de la aplicación están almacenados en servidores privados ejecutados en el ordenador en el cual se han desarrollado e instalado todas las tecnologías requeridas. Eso implica que la aplicación no funciona fuera de la red privada en la que está dicho equipo por lo que se debería desplegar la aplicación en un entorno real para que cualquier persona pueda utilizarla. Esta tarea implicaría la contratación de un servicio que al suponer un coste mensual hemos descartado.

Finalmente, la mayor ambición de este proyecto sería desarrollar nueva funcionalidad para que cualquier profesional sanitario (médico, farmacéutico, fisioterapeuta…) pueda consultar el seguimiento del tratamiento por parte del paciente. En esta parte se incluiría la opción de chatear con el profesional o realizar videollamadas. Además, como hemos comentado antes, sería interesante que el propio profesional pueda insertar y manipular la información clínica relacionada con los tratamientos y los fármacos y ejercicios que tiene que debería ingerir y realizar el paciente a lo largo del tiempo.

6. BIBLIOGRAFÍA

[1] T. B. Sherer, S. Chowdhury, K. Peabody, and D. W. Brooks, "Overcoming obstacles in Parkinson's disease," *Mov. Disord.*, vol. 27, no. 13, pp. 1606–1611, Nov. 2012, doi: 10.1002/mds.25260.

[2] C. A. Taylor *et al.*, "Environmental, medical, and family history risk factors for Parkinson's disease: a New England-based case control study.," *Am. J. Med. Genet.*, vol. 88, no. 6, pp. 742–9, Dec. 1999, [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/10581500.

[3] S. Ceccatelli, "Mechanisms of neurotoxicity and implications for neurological disorders.," *J. Intern. Med.*, vol. 273, no. 5, pp. 426–8, May 2013, doi: 10.1111/joim.12053.

[4] A. H. V. Schapira, "Etiology and Pathogenesis of Parkinson Disease," *Neurol. Clin.*, vol. 27, no. 3, pp. 583–603, Aug. 2009, doi: 10.1016/j.ncl.2009.04.004.

[5] S. Alkhuja, "Parkinson disease: research update and clinical management.," *South. Med. J.*, vol. 106, no. 5, p. 334, May 2013, doi: 10.1097/SMJ.0b013e318290f72a.

[6] R. C. of Physicians, "Parkinson's Disease: National Clinical Guideline for Diagnosis and Management in Primary and Secondary Care," 2006.

[7] C. Marras *et al.*, "Prevalence of Parkinson's disease across North America," *npj Park. Dis.*, vol. 4, no. 1, p. 21, Dec. 2018, doi: 10.1038/s41531-018-0058-0.

[8] Santiago de Cuba, "Los Objetivos de Desarrollo Sostenible y la academia," vol. 22, pp. 838–848, 2018, [Online]. Available: http://scielo.sld.cu/scielo.php?pid=S1029-30192018000800838&script=sci_arttext&tlng=en.

[9] Lorena García, "Dispositivo para realizar un seguimiento de los síntomas a personas con párkinson," 2019, [Online]. Available: https://www.consalud.es/saludigital/158/dispositivo-para-realizar-un-seguimiento-de-los-sintomas-a-personas-con-parkinson_63968_102.html.

[10] Neri-Nani GA, “Síntomas motores de la enfermedad de Parkinson. Rev Neurol Neurocir Psiquiat,” vol. 45(2), pp. 45–50, 2017.

[11] C. K, “Enfermedad de Parkinson.”

[12] D. J. Keus S, Munneke M, Graziano M, Paltamaa J, Pelosin E, “European Physiotherapy Guideline for Parkinson’s Disease,” 2014, [Online]. Available: https://www.parkinsonnet.nl/app/uploads/sites/3/2019/11/eu_guideline_parkinson_guideline_for_pt_s1.pdf.

[13] Y. R. R. Jorge Michel Rodríguez Pupo, Yuna Viviana Díaz Rojas, Yesenia Rojas Rodríguez and R. A. Rodríguez, “Actualización en enfermedad de Parkinson idiopática,” 2013, [Online]. Available: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1560-43812013000200007.

[14] S. A. Factor, D. S. Higgins, and J. Qian, “Primary progressive freezing gait: a syndrome with many causes.,” *Neurology*, vol. 66, no. 3, pp. 411–4, Feb. 2006, doi: 10.1212/01.wnl.0000196469.52995.ab.

[15] “Opicapona para el Parkinson disponible en Alemania y Reino Unido.”

[16] A. M. G. Hernández, “TICS - Tecnologías de Información y Comunicación,” [Online]. Available: https://www.monografias.com/trabajos89/tics-tecnologias-informacion-y-comunicacion/tics-tecnologias-informacion-y-comunicacion.shtml.

[17] J. D. M. Pozo, “Tecnologia Y Desarrollo En Dispositivos Moviles (Modulo 2),” pp. 11–26.

[18] X. Xu, Z. Fu, and W. Le, “Exercise and Parkinson’s disease.,” *Int. Rev. Neurobiol.*, vol. 147, pp. 45–74, 2019, doi: 10.1016/bs.irn.2019.06.003.

[19] C. J. de Goede, S. H. Keus, G. Kwakkel, and R. C. Wagenaar, “The effects of physical therapy in Parkinson’s disease: a research synthesis.,” *Arch. Phys. Med. Rehabil.*, vol. 82, no. 4, pp. 509–15, Apr. 2001, doi: 10.1053/apmr.2001.22352.

[20] S. Kumar *et al.*, “Mobile health technology evaluation: the mHealth evidence

workshop.,” *Am. J. Prev. Med.*, vol. 45, no. 2, pp. 228–36, Aug. 2013, doi: 10.1016/j.amepre.2013.03.017.

[21] M.-C. J. Alonso-Arévalo J, “Mobile health applications: potential, regulation and security,” vol. 3, pp. 1–13, 2017.

[22] J. A. M.-C. Julio Alonso-Arévalo, “Aplicaciones móviles en salud: potencial, normativa de seguridad y regulación,” vol. 28, 2017, [Online]. Available: http://www.acimed.sld.cu/index.php/acimed/article/view/1136/690.

[23] J. A. S. Perales, “De los Objetivos del Milenio al desarrollo sostenibleNaciones Unidas y las metas globales,” vol. 7, pp. 49–84, 2014, [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=4942588.

[24] L. Dayer, S. Heldenbrand, P. Anderson, P. O. Gubbins, and B. C. Martin, “Smartphone medication adherence apps: Potential benefits to patients and providers,” *J. Am. Pharm. Assoc.*, vol. 53, no. 2, pp. 172–181, Mar. 2013, doi: 10.1331/JAPhA.2013.12202.

[25] A. M. Cascajero, “Aplicaciones con utilidad farmacológica,” *2020*, [Online]. Available: https://drive.google.com/open?id=1cevsHoqheXWIXN_poTz24UT2tEqNUlQT.

[26] S. Kumar *et al.*, “Mobile Health Technology Evaluation,” *Am. J. Prev. Med.*, vol. 45, no. 2, pp. 228–236, Aug. 2013, doi: 10.1016/j.amepre.2013.03.017.

[27] L. James, Michael and Walter, “Scrum reference card,” *CollabNet Inc*, 2010, [Online]. Available: https://www.collab.net/sites/default/files/uploads/CollabNet_scrumreferencecard.pdf.

[28] J. K. O’Toole, W. Alvarado-Little, and C. J. W. Ledford, “Communication with Diverse Patients,” *Pediatr. Clin. North Am.*, vol. 66, no. 4, pp. 791–804, Aug. 2019, doi: 10.1016/j.pcl.2019.03.006.

[29] M. E. Minguell, “Interactividad e interacción,” vol. 1, pp. 15–25, 2002, [Online]. Available: https://relatec.unex.es/article/view/2.

[30] Karo.otoya@gmail.com, “Parkinson : Sintomas Y Tratamiento - FAQ,” *2018*, [Online]. Available: https://play.google.com/store/apps/details?id=com.proyectoultra24&hl=es&gl=US.

[31] 9zest, “9zest,” 2018, [Online]. Available: https://play.google.com/store/apps/details?id=com.ninezest.fixhealth&hl=es&gl=US.

[32] P. M. Thomas, Pablo Javier Galdamez, Nicolás and Delía, Lisandro Nahuel Corbalán, Leonardo César Pesado, “Dispositivos móviles: desarrollo y análisis de rendimiento de aplicaciones multiplataforma,” 2016, [Online]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/53448/Documento_completo.pdf?sequence=1.

[33] A. Martínez, “App híbrida o App Nativa? Según para qué,” 2017, [Online]. Available: https://cuatroochenta.com/app-hibrida-o-app-nativa-segun-para-que/.

[34] J. Ortega, Dinarle; Guevara, María; Benavides, “UN FRAMEWORK DE PROGRAMACIÓN WEB,” vol. 15, pp. 144–171, 2016, [Online]. Available: https://www.redalyc.org/pdf/784/78457627004.pdf.

[35] Facebook Inc, “React Native,” [Online]. Available: https://reactnative.dev/docs/getting-started.

[36] B. Eisenman, “Learning react native: Building native mobile apps with JavaScript,” 2015, [Online]. Available: https://books.google.es/books?hl=es&lr=&id=274fCwAAQBAJ&oi=fnd&pg=PR2&dq=react+native&ots=tGrheKi8m3&sig=_Lx6MEtAH4Jo6DEloGT1yIkz9jI#v=onepage&q=react native&f=false.

[37] Expo, “Introduction to Expo,” [Online]. Available: https://docs.expo.io/.

[38] Expo, “API Reference,” 2021, [Online]. Available: https://docs.expo.io/versions/latest/?redirected.

[39] Ionos, “Qué es un runtime environment,” 2020, [Online]. Available:

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-un-runtime-environment/.

[40] O. Foundation, “About Node.js,” [Online]. Available: https://nodejs.org/en/about/.

[41] G. Bierman, M. Abadi, and M. Torgersen, “Understanding TypeScript,” 2014, pp. 257–281.

[42] A. R. Melton, Jim and Simon, “Understanding the new SQL: a complete guide,” 1993, [Online]. Available: https://books.google.es/books?hl=es&lr=&id=wyhXvU0Eyg0C&oi=fnd&pg=PP2&dq=sql&ots=MvQGLxMQ7e&sig=YzqpJXQPsqa-6Z9lbfILZB1OCFQ#v=onepage&q=sql&f=false.

[43] Restfulapi.net, “REST API Tutorial,” [Online]. Available: https://restfulapi.net/.

[44] D. Farrell, “Python REST APIs With Flask, Connexion, and SQLAlchemy,” [Online]. Available: https://realpython.com/flask-connexion-rest-api/.

[45] I. Postman, “POSTMAN,” [Online]. Available: https://learning.postman.com/docs/getting-started/introduction/.

[46] P. S. F. M, Chris, “BeginnersGuide/Overview,” [Online]. Available: https://wiki.python.org/moin/BeginnersGuide/Overview.

[47] Microsoft, “Getting Started,” [Online]. Available: https://code.visualstudio.com/docs.

[48] Oracle, “MySQL Workbench,” [Online]. Available: https://www.mysql.com/products/workbench/.

[49] Developers.android, “Cómo ejecutar apps en Android Emulator,” 2020, [Online]. Available: https://developer.android.com/studio/run/emulator?hl=es-419.

[50] S.C Evercoder Software S.R.L, “Herramienta de Maquetas, Esquemas & Prototipos UI,” 2021, [Online]. Available: https://moqups.com/es/.

[51] S. K. Peter Mattis, “Gimp,” [Online]. Available: http://www.gimp.org.es/.

[52] M. Haldar, “Understanding React v16.4+ New Component Lifecycle Methods,” 2018, [Online]. Available: https://blog.bitsrc.io/understanding-react-v16-4-new-component-lifecycle-methods-fa7b224efd7d.

[53] S. Weber, “useState vs. useRef: Similarities, differences, and use cases,” 2021, [Online]. Available: https://blog.logrocket.com/usestate-vs-useref/.

ANEXO I: casos de uso desarrollados

En las siguientes tablas se incluyen los casos de uso desarrollados en el apartado de análisis y diseño de software. Se han desarrollado los siguientes casos de uso:

Tabla 1. Registro de usuario

Nombre:	Registro
Descripción:	Consiste en introducir el nombre, nombre de usuario, email, contraseña y seleccionar el género para la inserción de esos datos en la base de datos.
Actores:	Usuario
Precondiciones:	-
Flujo de eventos:	1. El usuario debe de situarse en la pantalla de registro. 2. Una vez situado debe de rellenar el formulario con los datos requeridos. 3. Cuando todos los datos son rellenados debe de pulsar el botón de registrarse.

Post-condiciones:	El nombre de usuario no debe de existir en los registros de la base de datos, todos los datos deben de ser rellenados, el correo electrónico debe de tener el formato correcto.

Tabla 2. Inicio de sesión

Nombre:	Inicio de sesión
Descripción:	Consiste en introducir el nombre de usuario y la contraseña para navegar a la pantalla de inicio
Actores:	Usuario
Precondiciones:	El usuario tiene que estar registrado.
Flujo de eventos:	1. El usuario debe de situarse en la pantalla de inicio de sesión. 2. Una vez situado debe de rellenar el formulario con los datos requeridos. 3. Cuando todos los datos son rellenados debe de pulsar el botón de inicio de sesión.
Post-condiciones:	El nombre de usuario debe de existir en los registros de la base de datos y la contraseña debe de coincidir con los datos relacionados de ese usuario en la base de datos.

Tabla 3. Recomendaciones y contraindicaciones

Nombre:	Confirmación de lectura de las recomendaciones y contraindicaciones
Descripción:	Consiste en que el usuario se desplace hacia la parte inferior de la pantalla de inicio, habiendo leído toda la información mostrada a lo largo del recorrido y clicar el botón de aceptar

	recomendaciones y contraindicaciones para que posteriormente sea registrada esa acción en la base de datos y se muestre información distinta en la pantalla de inicio.
Actores:	Usuario
Precondiciones:	El usuario tiene que estar registrado y debe de haber iniciado sesión.
Flujo de eventos:	1. El usuario debe de situarse en la pantalla de inicio o "Home". 2. Una vez situado debe de desplazarse hacia la parte inferior de la pantalla leyendo toda la información. 3. Tras situarse en la parte inferior y identificar el botón de aceptar debe de clicarlo.
Post-condiciones:	Ninguna.

Tabla 4. Formulario de fisioterapia

Nombre:	Rellenar formulario de fisioterapia.
Descripción:	Consiste en seleccionar todas las opciones con sus respectivas cuestiones que van apareciendo conforme se navega a través de las pantallas del formulario.
Actores:	Usuario
Precondiciones:	El usuario tiene que estar registrado y debe de haber iniciado sesión.
	1. El usuario debe de situarse en la pantalla tareas mediante el clicado del botón de tareas en el navegador inferior. 2. Una vez situado en la pantalla de tareas tiene que clicar sobre el botón de formulario de fisioterapia.

Flujo de eventos:	3. En la primera pantalla del formulario debe de seleccionar una frecuencia a la que desea realizar los ejercicios a lo largo de la semana y uno o varios problemas de carácter general que pueda tener relacionados a su enfermedad. 4. Tras seleccionar las opciones debe de clicar sobre el botón de confirmar para que aparezcan las opciones seleccionadas debajo de cada cuestión pudiendo ser eliminadas mediante un clic sobre el icono en forma de cruz. 5. Para navegar al siguiente nivel de cuestiones debe de clicar en el botón situado en la parte inferior con el nombre de "siguiente". 6. Tras la navegación se mostrarán una serie de cuestiones dependiendo de cuántas opciones se hayan seleccionado previamente con el fin de que el usuario seleccione los síntomas relacionados a los problemas de carácter global previos siendo necesaria la selección de las opciones listadas. 7. Una vez seleccionadas las opciones de todas las cuestiones deberá de repetir el procedimiento anterior clicando al botón con el nombre "siguiente" en la parte inferior de la pantalla. 8. Tras navegar a la última pantalla el usuario deberá de escoger los bloques de ejercicios que va a querer realizar en su tratamiento. 9. Finalmente deberá de clicar sobre el botón de "aceptar" para finalizar el formulario.
Post-condiciones:	Es necesario que seleccione por lo menos una opción de cada cuestión en cada nivel del formulario para poder continuar al siguiente nivel.

Tabla 5. Selección de ejercicios

Nombre:	Seleccionar los ejercicios para el tratamiento.

Descripción:	Consiste en seleccionar múltiples ejercicios del listado mostrado por cada bloque de ejercicio con el fin de programar el plan dependiendo de la frecuencia seleccionada en el formulario de fisioterapia.
Actores:	Usuario
Precondiciones:	El usuario tiene que estar registrado, debe de haber iniciado sesión y tiene que haber rellenado el formulario de fisioterapia.
Flujo de eventos:	1. El usuario debe de situarse en la pantalla tareas mediante el clicado del botón de tareas en el navegador inferior. 2. Una vez situado en la pantalla de tareas tiene que clicar sobre el botón de formulario de ejercicios. 3. Tras navegar a la pantalla de ejercicios debe de marcar cada ejercicio que deseé realizar pudiendo previamente consultar los detalles de ese ejercicio mediante el botón de información situado a la derecha del nombre de ejercicio. 4. Una vez seleccionados todos los ejercicios deseados debe de clicar sobre el botón de confirmar selecciones situado en la parte inferior de la pantalla. 5. Tras clicar sobre el botón de confirmar selecciones el usuario debe de clicar sobre el botón de aceptar de la ventana emergente que aparece tras la acción anterior.
Post-condiciones:	Es necesario que el usuario seleccione una cantidad mínima de ejercicios, siendo el requisito de que la suma del tiempo de todos los ejercicios esté dentro del límite diario de tiempo en función de la frecuencia seleccionada en el formulario de fisioterapia.

Tabla 6. Formulario de farmacia

Nombre:	Rellenar el formulario de farmacia
Descripción:	Consiste en seleccionar los tratamientos, fármacos y sustancias que el usuario consume durante su tratamiento y una de las opciones de cada cuestión para establecer de si el usuario padece de demencia, depresión, está embarazada y/o en periodo de lactancia en el caso de ser mujer.
Actores:	Usuario
Precondiciones:	El usuario tiene que estar registrado y debe de haber iniciado sesión.
Flujo de eventos:	1. El usuario debe de situarse en la pantalla tareas mediante el clicado del botón de tareas en el navegador inferior. 2. Una vez situado en la pantalla de tareas tiene que clicar sobre el botón de formulario de farmacia. 3. Tras navegar a la pantalla del formulario debe de desplegar el listado de tratamientos mediante un clic en el botón debajo de la primera cuestión pudiendo escribir el nombre del tratamiento para filtrar entre los que haya disponibles o simplemente desplazarse y seleccionar la opción deseada en el listado mostrado. 4. Tras seleccionar una o varias opciones en cada opción se mostrará una ventana emergente indicando el fármaco o conjunto de fármacos que contiene ese tratamiento siendo necesaria la confirmación de para poder continuar mediante el clic en el botón de ok de la ventana. 5. Tras seleccionar las opciones deseadas deberá de pulsar sobre el botón de confirmar del listado en la parte inferior o en el icono de flecha que apunta hacia la izquierda situada en la parte derecha del buscador para que se vuelva a plegar el listado y continuar con la siguiente cuestión.

	6. Las dos siguientes cuestiones son otras listas desplegables que tras clicar en el recuadro y seleccionar la opción afirmativa aparece un listado de fármacos, los cuales el usuario tiene que seleccionar y posteriormente confirmar en el caso de que padezca alguna alergia a esos fármacos o esté consumiendo fármacos sin prescripción adicionales. 7. A continuación, aparece una cuestión que tras seleccionar la opción afirmativa hay que introducir el texto de cada sustancia que toma separado por coma en el caso de que sea más de una. 8. Después el usuario debe de seleccionar las opciones de las cuestiones siguientes, las cuales, son de una sola selección para determinar si el usuario pueda padecer de demencia, depresión, está en periodo de embarazo y/o lactancia, los cuales en caso afirmativo se almacenarán en la tabla de usuario de la base de datos. 9. Finalmente, el usuario tiene que confirmar de que ha rellenado el formulario clicando sobre el botón "confirmar" de la parte inferior de la pantalla.
Post-condiciones:	Es necesario de que el usuario rellene todas las cuestiones a excepción de seleccionar los tratamientos, fármacos adicionales, alergias y/o sustancias.

Tabla 7. Marcar tarea realizada

Nombre:	Completar tarea del calendario como completada
Descripción:	Consiste en marcar como completada cualquier tarea en el listado de tareas de un día determinado en el calendario.
Actores:	Usuario

Precondiciones:	El usuario tiene que estar registrado, debe de haber iniciado sesión, tiene que haber rellenado el formulario de fisioterapia, seleccionado un conjunto de ejercicios y haber rellenado el formulario de farmacia.
Flujo de eventos:	1. El usuario debe de situarse en la pantalla de calendario mediante el clicado del botón de tareas en el navegador inferior. 2. Una vez situado en la pantalla tiene que identificar la tarea que ha sido realizada el día actual mediante un clic sobre el contenedor de la descripción corta de la tarea.
Post-condiciones:	Ninguna.

Tabla 8. Ver detalle de ejercicio

Nombre:	Visualizar la información detallada de un ejercicio en el calendario.
Descripción:	Consiste en consultar información en más detalle del ejercicio con el fin de informarse de su realización, frecuencia, materiales necesarios, consejos y vídeo demostrativo.
Actores:	Usuario
Precondiciones:	El usuario tiene que estar registrado y haber iniciado sesión.
Flujo de eventos:	1. El usuario debe de situarse en la pantalla de calendario mediante el clicado del botón de tareas en el navegador inferior. 2. Tras navegar a la pantalla debe de identificar el ejercicio que desea consultar más información. 3. Una vez identificado puede realizar un clic largo de 1 segundo aproximadamente sobre el contenedor de la

	descripción corta del ejercicio o clicando sobre el botón de "detalles". 4. Tras realizar una de las acciones del paso anterior se abrirá una ventana modal, dependiendo del contenido deslizable hacia abajo, con un botón de aceptar en la parte inferior para cerrarla.
Post-condiciones:	Ninguna.

Tabla 9. Horario recomendado para realizar ejercicios

Nombre:	Consultar el horario recomendado para realizar ejercicios
Descripción:	Consiste en ver la programación horaria de 24 horas en la cual aparecen 3 colores distintos, siendo verde recomendado, azul no ideal y naranja no recomendado con el fin de saber a qué hora es recomendable o no realizar ejercicios.
Actores:	Usuario
Precondiciones:	El usuario tiene que estar registrado, debe de haber iniciado sesión, tiene que haber rellenado el formulario de fisioterapia, seleccionado un conjunto de ejercicios y haber rellenado el formulario de farmacia.
Flujo de eventos:	1. El usuario debe de situarse en la pantalla de calendario mediante el clicado del botón de tareas en el navegador inferior. 2. Una vez situado en la pantalla, en la parte inferior del calendario aparece una sección en la cual hay 24 cuadrados enumerados y de distintos colores. 3. Tras identificar el componente el usuario debe de mirar los 24 cuadrados y identificar los colores verde o azul para saber

	la hora recomendada o no ideal pero permisiva para realizar los ejercicios del listado.
Post-condiciones:	Ninguna.

Tabla 10. Cierre de sesión

Nombre:	Cerrar sesión
Descripción:	Consiste en salir de la aplicación para volver a la pantalla de inicio de sesión.
Actores:	Usuario
Precondiciones:	El usuario tiene que estar registrado y haber iniciado sesión.
Flujo de eventos:	1. El usuario debe de situarse en la pantalla de inicio o "Home" 2. Una vez situado debe de desplegar el menú lateral mediante un desplazamiento del dedo desde el extremo izquierdo hacia la derecha o clicar sobre el icono de las tres rayas en la parte superior izquierda denominada header. 3. Tras abrir el menú lateral tiene que clicar sobre el botón de cerrar sesión.
Post-condiciones:	Ninguna.

ANEXO II: esquema inicial de datos

Al principio se diseñó un esquema E/R partiendo de las especificaciones de los integrantes del grupo, pero conforme el desarrollo iba avanzando y evolucionando fueron surgiendo diferentes necesidades y se plantearon diferentes formas de organización de datos. En este anexo se adjunta el planteamiento inicial del esquema-entidad relación: