UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

Aplicación móvil para interfaz terapéutica de un sistema de telerehabilitación

Autor:

D. Óscar Alejandro García Custodio

Tutor:

Dr. D. Mario Martínez Zarzuela

Valladolid, Julio de 2018

TÍTULO:	Aplicación móvil para interfaz terapéutica de un sistema de telerehabilitación
AUTORA:	D. Óscar Alejandro García Custodio
TUTOR:	Dr. Mario Martínez Zarzuela
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	Dr. Francisco J. Díaz Pernas
VOCAL:	Dr. Míriam Antón Rodríguez
SECRETARIO:	Dr. Mario Martínez Zarzuela
SUPLENTE 1:	Dr. David González Ortega
SUPLENTE 2:	Dr. Carlos Gómez Peña

FECHA:

CALIFICACIÓN:

A mis padres y hermanos, por todo su cariño y su fuerza cuando más lo necesité.

A mis amigos, por ser esa red con la que todo es más fácil.

A Mario, por su ayuda y por darme la oportunidad de formar parte de un proyecto tan prometedor.

Resumen

Debido a los accidentes y operaciones, a diario, una gran cantidad de personas requieren de terapias de rehabilitación para recuperar total o parcialmente las funcionalidades de ciertas partes del cuerpo. Por lo que la eficacia de este proceso afecta directamente a la recuperación del paciente y para ello es necesario herramientas que permitan la movilidad en la gestión, como las aplicaciones móviles.

En primer lugar, es necesario que enunciemos cuales son los objetivos que plantea este proyecto. Para conseguirlos, haremos un estudio de la plataforma de telerehabilitación que utilizaremos y las que hay actualmente desarrolladas. Previo al desarrollo de la solución móvil investigaremos la tendencia actual del mercado para conseguir llegar al mayor número de personas y hacerlo de una manera eficaz. Por último, desarrollaremos la herramienta teniendo presente las consideraciones realizadas y probaremos la usabilidad de la solución con sujetos de prueba representativos.

Para terminar el estudio y desarrollo, se expondrán las conclusiones que hemos sacado en el proceso y propondremos unas posibles líneas futuras para la herramienta y la plataforma.

Palabras clave

Rehabilitación, smartphone, aplicación móvil, plataforma de gestión, Android.

Abstract

Due to accidents and operations, every day, many people in need of rehabilitation therapies to recover all or part of the functionalities of certain parts of the body. So, the effectiveness of this process was directly to the recovery of the patient and for this it is necessary that mobility in management, such as mobile applications.

First, there needed to be a clear enunciation of the objectives of this project. To achieve, we will make a study of the telerehabilitation platform that we will use and those that are currently developed. Prior to the development of the mobile solution, we will investigate the current market trend to reach the largest number of people and do it effectively. Finally, we will develop the tool including the considerations that are listed and then, prove the usability of the solution with representative study subjects.

To finish the study and development, the conclusions that we have drawn in the process will be exposed and we will propose some possible future lines for the tool and the platform.

Keywords

Rehabilitation, smartphone, mobile application, management platform, Android.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Motivación y Objetivos

En España hay un gran número de accidentes tanto laborales como accidentes de tráfico. Entre los años 2010 y 2015, hubo una media de casi 125.000 víctimas en accidentes de tráfico en medios terrestres al año [1], y una media de algo más de 450.000 accidentes laborales al año [2]. Dependiendo de la gravedad de estos y otros accidentes no contemplados en las estadísticas anteriores, estas víctimas deberán recurrir a la "rehabilitación médico-funcional" para recuperar las capacidades motrices.

Tras algunas operaciones quirúrgicas, también es necesario recurrir a la rehabilitación anteriormente mencionada para recuperar la movilidad de articulaciones, fortalecer músculos que pueden haberse visto afectados o recobrar la normalidad en tendones y otras partes del cuerpo.

La rehabilitación puede ser no sólo física, sino también cognitiva. Según el Instituto de Neurología Cognitiva de Argentina (INECO), el deterioro cognitivo leve ha sido largamente reconocido como la pérdida de memoria asociada al aumento de la edad, aunque más recientemente se ha llegado a considerar una patología que afecta al cerebro [3].

El Instituto de Especialidades Neurológicas señala que: "La Rehabilitación Neuropsicológica tiene como objetivo general el mejorar las funciones mentales que han resultado afectadas como consecuencia del daño cerebral, sobre todo en atención, memoria, lenguaje, percepción, psicomotricidad, función ejecutiva y emoción, así como devolver al paciente el nivel de funcionamiento, independencia e integración social más alto posible" [4].

Los dos factores a tener en cuenta más importantes para que la rehabilitación, en especial la cognitiva tenga la mayor eficacia terapéutica posible son: iniciar la rehabilitación precozmente y la constancia en el trabajo cognitivo, es decir la adecuada adhesión al tratamiento. Cuanto más se tarde en iniciar un tratamiento de rehabilitación, más se pueden incrementar los efectos negativos, y si no se tiene una continuidad en el tratamiento, se puede perder los avances realizados.

Una vez que se ha producido la evaluación neuropsicológica de un paciente para determinar las áreas afectadas y las que no lo están, se establece el protocolo a seguir para que se lleve a cabo la rehabilitación. Las técnicas a utilizar pueden ser:

- Restauración de la función dañada: Empleando técnicas de estimulación y entrenamiento cognitivo.
- Compensación de la función perdida: Las actividades tienen un objetivo funcional, se emplean estrategias alternativas o ayudas externas.
- Optimización de las funciones residuales: Donde lo que se busca es fomentar las funciones no afectadas.
- Técnicas de Modificación de conducta cognitivo-conductuales.
- Mixta: utilización de las técnicas anteriores de modo combinado.

• Técnicas de estimulación Cognitiva en Demencias:

- Terapia de Orientación a la realidad.
- Técnica de Reminiscencia.
- Psicomotricidad.
- Técnicas mixtas (restauración, compensación y optimización).

Según el Instituto Nacional de Estadística (INE), la población con discapacidad que ha recibido servicios sanitarios o sociales, en concreto rehabilitación médico-funcional, en un tiempo de estudio de 14 días es de 138700 personas en toda España, siendo en Castilla y León de 5600 personas. Este estudio realizado en 2008, aunque puede resultar un tanto obsoleto y en un periodo de tiempo demasiado corto, puede esbozar ligeramente los resultados reales de este escenario [5].

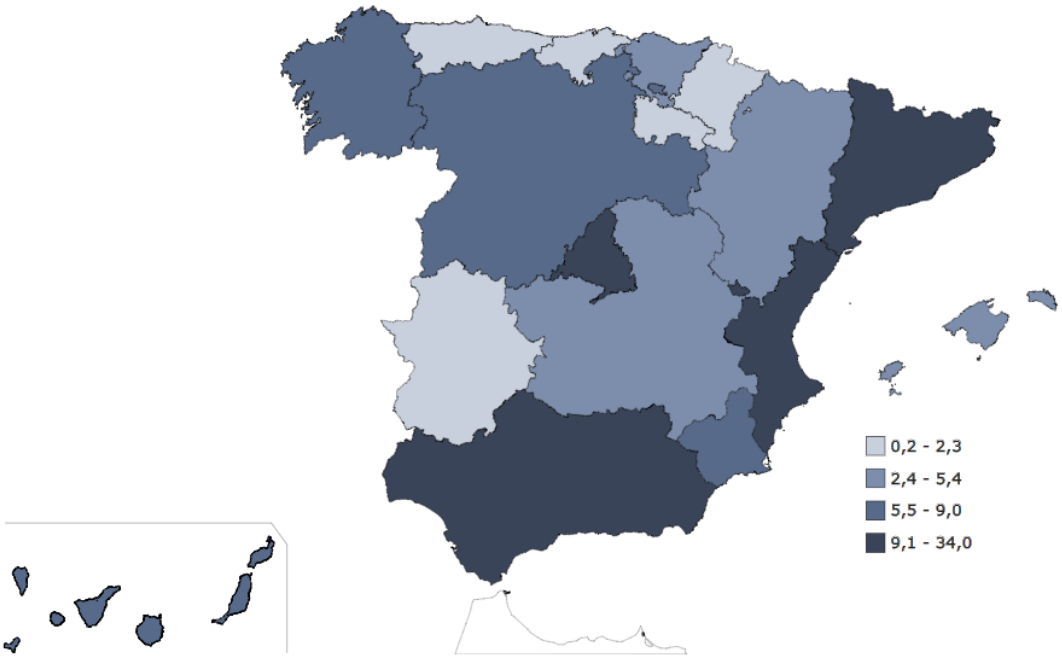

***Ilustración 1**. Mapa de población (en miles de personas) con discapacidad que ha recibido servicios de rehabilitación médico-funcional según la comunidad autónoma*

Si tenemos en cuenta todos los casos expuestos anteriormente, es fácil caer en la cuenta de la importancia que tiene la rehabilitación en la calidad de vida de las personas y la cantidad de casos en los que resulta fundamental.

Hasta ahora, lo más común en materia de rehabilitación era acudir a centros especializados en los que mediante la fisioterapia o ejercicios se conseguían los resultados del tratamiento, en ocasiones añadiendo ejercicios de repetición en casa. Hoy en día hay técnicas más novedosas en las que con la integración de las nuevas tecnologías y los videojuegos, el largo camino que puede llegar a ser una rehabilitación se hace mucho más ameno y llevadero.

Resulta fácil pensar que para un paciente es mucho más interesante el hacer un ejercicio de flexión de rodilla si este lleva consigo una meta añadida a la final que es la

recuperación. Mediante los videojuegos puedes hacer que una serie de repeticiones de flexión vertical de rodilla se convierta en una experiencia de realidad virtual en la que hay que dar toques a una pelota o simular que subes unos escalones.

La implantación es estas nuevas tecnologías abre un mundo de posibilidades para que la recuperación de los pacientes sea más entretenida y se consiga que no abandonen el tratamiento o no lo sigan con la dedicación necesaria, que es uno de los pilares para la rehabilitación que hemos mencionado anteriormente.

Al igual que en las técnicas tradicionales, el encargado de realizar el seguimiento y valorar el progreso siempre será el médico, que, conociendo los ejercicios, será capaz de asignar para cada juego o experiencia interactiva una determinada duración, un determinado nivel o un limitado número de dificultades para el paciente.

En el Grupo de Telemática e Imagen (GTI) de la Universidad de Valladolid trabajan en herramientas de telerehabilitación para discapacidades motoras y cognitivas mediante una plataforma llamada Telekin [6].

Esta novedosa herramienta permite tener usuarios con distintos niveles de permisos. Estos usuarios pueden tener el rol de médico y asignar terapias y actividades a pacientes, o incluso pueden tener el rol de familiar para ver los resultados del paciente al que están vinculados. Además de ser una potente herramienta de telerehabilitación, también pretende serlo de gestión de pacientes.

Actualmente la plataforma Telekin cuenta con una página web diseñada para que los usuarios interactúen con los datos del servidor mediante una interfaz atractiva. Este sitio web está totalmente optimizado para ordenadores y para todos los navegadores más comúnmente utilizados hasta la fecha.

Sin embargo, al ser una herramienta en la que los usuarios requerirán acceder con frecuencia y no necesariamente en un lugar con un ordenador, se hace necesario buscar una alternativa complementaria a este sitio web. Como solución, este proyecto tiene como **objetivo es crear una aplicación móvil para la configuración y seguimiento de terapias, complementaria al sitio web actual**. Esta herramienta deberá cumplir una serie de requisitos que consideraremos imprescindibles. Debemos conseguir diseñar de una solución portátil, crear una herramienta sencilla e intuitiva, y además, que tenga una integración total con la plataforma que existe actualmente.

Desde la aparición del primer teléfono inteligente en 1992, han evolucionado a lo largo de los años, y hoy en día son una herramienta que en España usa el 87% de la población con teléfono móvil de cualquier tipo [7]. Este porcentaje prácticamente se ha duplicado en los últimos 5 años, y tiende a seguir creciendo en los próximos años.

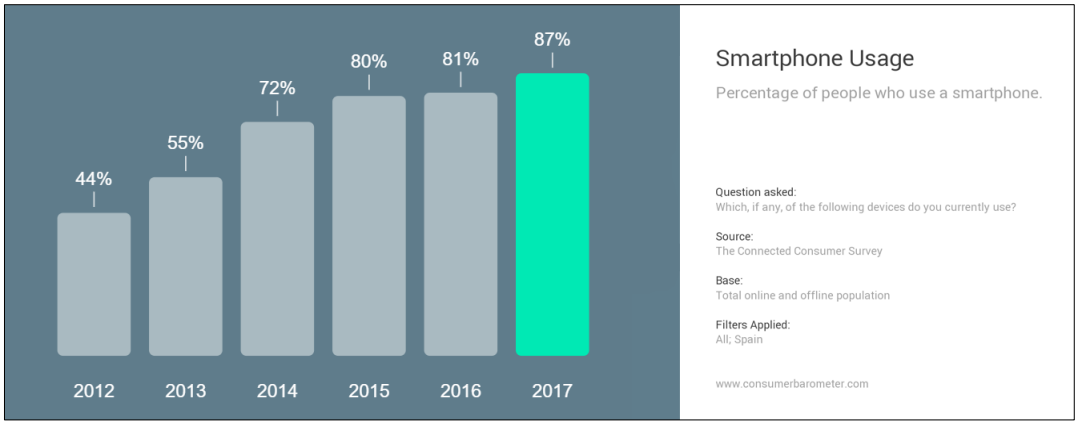

Ilustración 2. Porcentaje de la población con teléfono inteligente en España

A medida que han evolucionado los teléfonos inteligentes, las aplicaciones para estos se han aumentado de manera exponencial, superando en el sistema operativo Android las 3,5 millones de aplicaciones a día de hoy [8]. Las aplicaciones resultan unas herramientas realmente útiles para la interacción de los usuarios con datos en internet. Por lo tanto, la mejor alternativa para que la plataforma Telekin sea accesible desde cualquier sitio y en cualquier momento, es una aplicación.

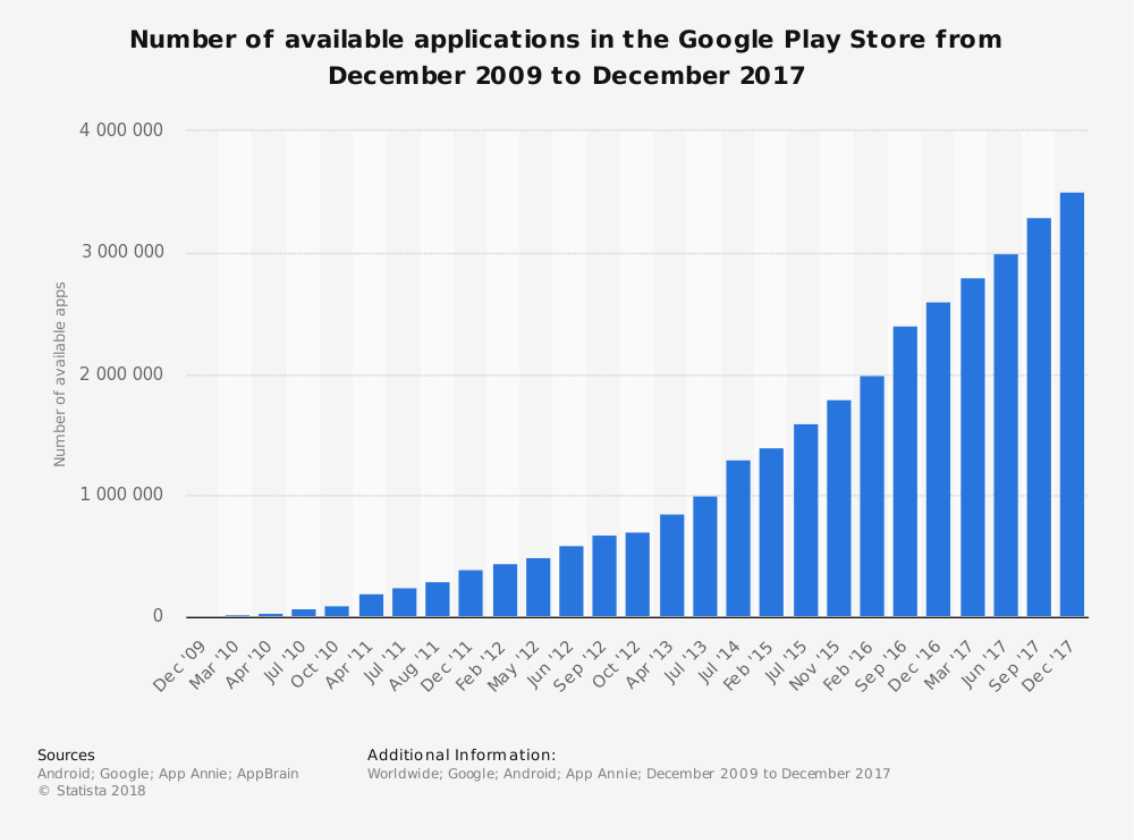

Ilustración 3. Numero de aplicaciones en el Google Play Store desde dic-2009 hasta dic-2017

El objetivo de este trabajo será desarrollar una aplicación para la utilización de la plataforma Telekin. Esta aplicación tendrá que poder satisfacer todas las necesidades de los usuarios con la plataforma.

Diseño de una solución portátil

Uno de los principales objetivos de este trabajo será el de crear una herramienta portátil en la que el usuario pueda acceder en cualquier momento y en cualquier lugar.

Pese a la versatilidad que ofrece un ordenador, un dispositivo móvil, ya sea un *Smartphone* o una *tablet,* tiene una gran autonomía y suficiente potencia para hacer una gran cantidad de tareas. Por otra parte, la alternativa portátil a una aplicación podría ser que la página web que de Telekin se optimizase para dispositivos móviles.

Actualmente esa optimización es parcial, no obstante, se podría considerar una herramienta complementaria, pero no sustitutiva. Una aplicación nativa, hace mejor uso de los recursos, lo que conlleva que los datos se muestren mucho más rápido puesto que no tiene que descargar todo el recurso de la página web. Además, al no tener que descargar tanta información, ya que en el caso de una aplicación solo descarga datos que hay en la base de datos y no lo necesario para visualizarlos correctamente, si el usuario dispone de poco ancho de banda, no le afectará en tan gran medida.

Por último, al tratarse de una plataforma a la que se accederá con mucha frecuencia, del orden de varias veces al día, es mucho más cómodo para el usuario acceder a una aplicación con una sola pulsación en el icono de la aplicación que acceder a una página web en la que tendrá que hacer varias pulsaciones para acceder al navegador y posteriormente al sitio web.

Creación de una herramienta sencilla e intuitiva

Al hacer cualquier aplicación para dispositivos móviles, una de las partes más importantes es el estudio del público al que está destinada dicha aplicación. No es lo mismo hacer un proyecto para niños que para una empresa. Cada grupo de usuarios tienen unas capacidades y unas limitaciones, y es de vital importancia que la aplicación sea sencilla y manejable para este sector.

En principio, la aplicación a realizar solo dará acceso a los usuarios registrados como médico, ya que la plataforma está aún en desarrollo y pueden producirse cambios, no obstante, en futuras implementaciones deberá dar acceso a todos los usuarios sea cual sea su rol. El proyecto final tendrá dos grupos de usuarios finales:

- Médicos y administradores de centro: Son personas de entre 30 y 60 años con estudios superiores que no tienen por qué estar familiarizados con la tecnología, sin embargo, tienen la capacidad para adaptarse a las nuevas herramientas de trabajo.
- Pacientes y familiares: Son personas de cualquier edad y cualquier nivel de estudios. No se puede saber si se familiarizarán fácilmente con la tecnología, lo más probable es que haya usuarios que no.

Una vez analizado el público que tendrá la aplicación, se puede observar que es un sector muy amplio. Por lo que nos tendremos que quedar con las condiciones más restrictivas.

Supondremos que los usuarios tienen un conocimiento del manejo de un teléfono inteligente limitado. No lo consideraremos nulo puesto que el hecho de tener un Smartphone supone una utilización, aunque sea básica.

Lo más importante a la hora de desarrollar la aplicación será que cualquier persona sepa que elemento pulsar para realizar una determinada acción. Esto lo conseguiremos mediante imágenes representativas y evitando una navegación profunda, es decir, que cualquier acción se pueda realizar pasando por un máximo de 3 pasos o pantallas.

También deberemos tener en cuenta las guías de diseño del sistema operativo móvil que elijamos para hacer nuestra aplicación. Es importante seguir estas guías para que un usuario que ha usado otras aplicaciones sepa enfrentarse a una nueva, como por ejemplo el botón de menú típico de Android o el botón de atrás típico de iOS.

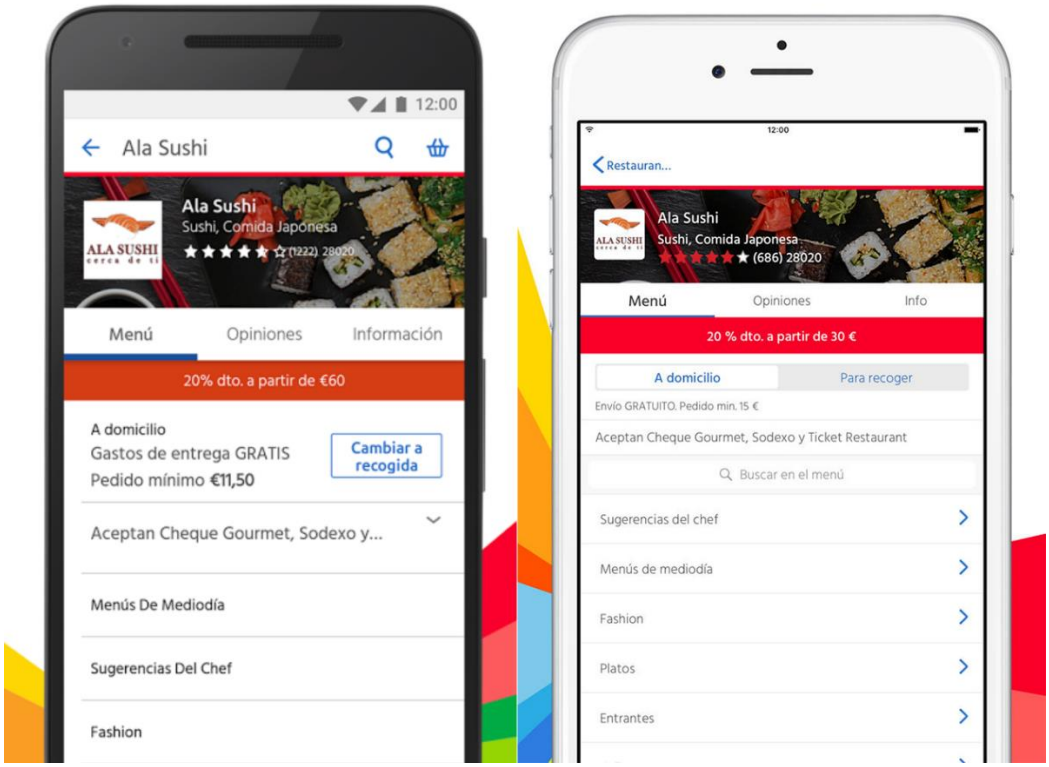

Ilustración 4. Comparativa de diseño entre Android e iOS

Integración total con la plataforma

Puesto que esta aplicación se basará en un sistema ya implementado en una página web, nuestro proyecto debe ser coherente con la plataforma actual. Por tanto, tendremos que encontrar un diseño adecuado que respete las guías del sistema operativo a la vez que no desoriente a un usuario acostumbrado a usar la plataforma web.

Además de tener una apariencia coherente con la página web, nuestra solución debe tener una organización tan similar como sea posible. Tener el mismo número de secciones y acceder a las funcionalidades disponibles desde las mismas pantallas.

Tal y como cabría esperar, los datos que se representen en la aplicación deben ser exactamente los mismos que se ven en la página web. Siempre que se añadan o se modifiquen datos en la aplicación deben reflejarse tanto es esta como el sitio web. Por tanto, deberán compartir la misma base de datos, la cual deberá ser accesible desde nuestra aplicación. Debido a esto, este proyecto consistirá no solo en hacer la aplicación, sino también en presentar los datos que se envíen a la base de datos del sistema Telekin, y preparar los que se envíen a la aplicación móvil.

Capítulo 2. Revisión de la tecnología

Lo primero antes de empezar a trabajar en el proyecto, es tomar una serie de decisiones sobre el desarrollo de la aplicación. Tendremos que decidir qué tecnología vamos a usar, tenemos varias alternativas:

	Aplicación Nativa	Aplicación Híbrida	Aplicación Web
Rendimiento	Alto	Medio	Bajo
Tiempo/Coste de desarrollo	Alto	Medio	Bajo
Multiplataforma	No	Si	Si
Acceso al dispositivo	Total	Medio	Bajo
Portabilidad de código	Baja	Alta	Alta

Tabla 1. Comparativa de tecnologías móviles

Con la información que tenemos en la tabla 1, podemos ver como no hay una tecnología indiscutiblemente mejor que otra, dependiendo de nuestro caso, nos puede venir mejor una u otra.

En cuanto al rendimiento, nos interesa que sea el mejor posible, no obstante, debemos tener en cuenta el tiempo de desarrollo, que en este caso es limitado. Respecto a los datos que aparecen en la tabla en cuanto al tiempo y coste de desarrollo, hay que tener en cuenta que se refieren al desarrollo de un proyecto, es decir, en el caso híbrido el tiempo es menor puesto que el código es reutilizable para hacer la aplicación para otras plataformas.

Teniendo en cuenta todos los datos, lo más sencillo es descartar una aplicación web ya que incluso pueden dar algunos problemas a la hora se subirlas a los *app stores*.

Lo más acertado en nuestro caso, será realizar una aplicación nativa para una plataforma (Android o iOS), para tener el máximo rendimiento para un tiempo de desarrollo similar. Al tener acceso total al dispositivo y sus funcionalidades, podremos implementar ahora o en actualizaciones futuras características totalmente integradas con el teléfono, ya sean notificaciones *push*, uso de cámara, etc.

Comparación entre Android e iOS

Actualmente el mercado de los teléfonos inteligentes está dominado por dos gigantes tecnológicos, Android e iOS. Según estadísticas, en el primer trimestre de 2017 Android estaba presente en el 86,1% de los teléfonos del mercado, un 2% más que el año anterior. Por otro lado, iOS obtenía un 13,7% del mercado, un 1,1% menos que el año anterior, y por último, el resto de sistemas operativos del mercado solo se quedan con un 0,2%, un 0,9% menos que en el primer trimestre de 2016 [9].

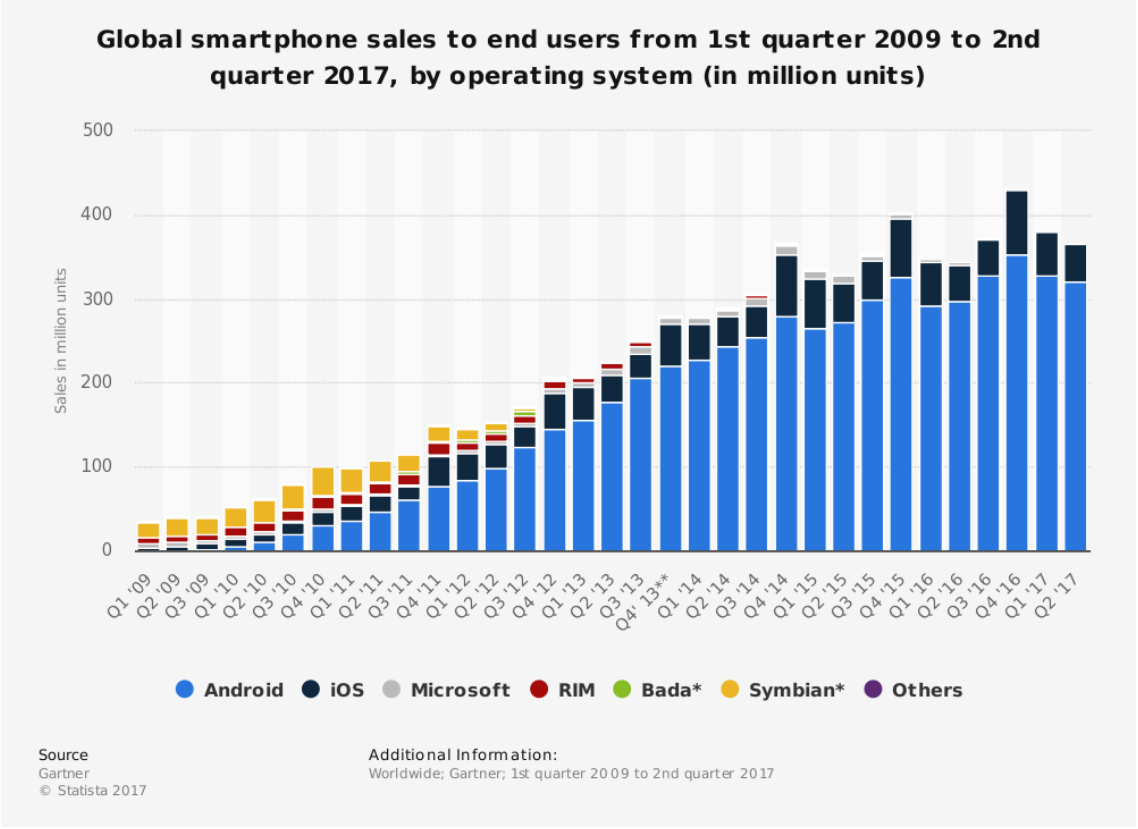

Ilustración 5. Ventas de teléfonos inteligentes desde 2009 hasta el segundo trimestre de 2017

Como podemos observar en la Ilustración 5, desde 2009, el crecimiento de Android e iOS han desbancado a todos los demás sistemas operativos quedándose con todo el mercado. Además, el hecho de que las empresas no apuesten por hacer sus aplicaciones en otros sistemas operativos que puedan surgir, hace que los clientes descarten una alternativa a los dos.

La evidente diferencia de la cuota de mercado en el que Android domina holgadamente es debido a que es un sistema operativo soportado por una gran variedad de fabricantes. Mientras que iOS es un sistema operativo que solo lo tienen los iPhone, Android lo tienen dispositivos de un rango de precios desde 50€ hasta 1000€. Es por este motivo por el que este sistema operativo llega a muchos más clientes, que no se pueden permitir el precio de un dispositivo de la marca Apple.

Las principales diferencias entre estos dos sistemas operativos son:

- **Actualizaciones**: Cuando Apple saca una nueva versión de iOS, todos sus teléfonos reciben esa actualización, sin embargo, cuando Android saca una nueva versión de su software, depende de cada fabricante el sacar la actualización para sus terminales. Generalmente las empresas solo introducen actualizaciones a los terminales de mayor calidad de su línea de productos, aunque cualquiera podría ser actualizado. La falta de estas actualizaciones hace que algunos terminales puedan quedarse obsoletos respecto a otros.

- **Seguridad**: Aunque hay personas que aseguran que iOS es menos sensible a ataques, no hay evidencias de que haya menos probabilidades de ser atacado por un *malware* con un sistema operativo o con otro. Una de las diferencias más importante entre estos dos sistemas operativos es que Android es un entorno abierto e iOS es un entorno cerrado. Cuando hablamos de entonos cerrado nos referimos a que las aplicaciones solo pueden ser descargadas desde la plataforma oficial "iTunes", las cuales han sido examinadas para descartar que tengan *malware*. El mercado oficial de aplicaciones de Android "Google Play" o "Play Store" no analiza por defecto las aplicaciones que han sido subidas a la plataforma, por lo que la seguridad es menor.

  Al tener esa seguridad en iOS, la única alternativa para descargar aplicaciones es iTunes, sin embargo, en Android hay otros *markets* alternativos en los que quizás no haya que pagar una licencia de desarrollador.

- **Licencia**: Para publicar aplicaciones en las plataformas oficiales, hace falta pagar unas licencias de desarrollador. Aunque Android permite instalar aplicaciones fuera de su *market*, este es el mejor lugar donde subir las aplicaciones, puesto que todos los teléfonos lo tienen instalado. En el caso de Android la licencia es un pago único de $25 USD, una vez hecho el desarrollador puede subir todas las aplicaciones que desee [10]. Para iOS la licencia de desarrollador básica cuesta $99 USD al año, y al igual que su competidor, una vez pagado puedes subir cualquier aplicación. Apple tiene una alternativa para empresas por $299 USD al año con más características [11].

- **Personalización de la IU**: En Android la interfaz de usuario (IU) es totalmente personalizable y la mayoría de empresas que venden estos teléfonos suelen incluirle una capa de personalización propia, a veces con *widgets* o aplicaciones propias que mejoran la experiencia de usuario. Por otro lado, como todos los teléfonos con iOS son del mismo fabricante, no tienen posibilidad de personalización, puesto que ya está optimizado para esos dispositivos.

- **Rendimiento**: Teniendo en cuenta que no existe un teléfono inteligente con el mismo *hardware* con ambos sistemas operativos, no hay datos exactos de la superioridad de uno sobre el otro. Históricamente, los iPhone han tenido componentes de menos potencia que sus competidores Android, un ejemplo claro es la memoria RAM. Sin embargo, los procesadores no son comparables y tendríamos que recurrir a herramientas como los *benchmark*. Como demuestra la Ilustración 6 [12], los dispositivos iPhone tienen un rendimiento mucho superior a sus competidores. No obstante, estos test no son del todo

fiables y son fácilmente manipulables, por lo que debemos ser muy cuidadosos a la hora de interpretarlos.

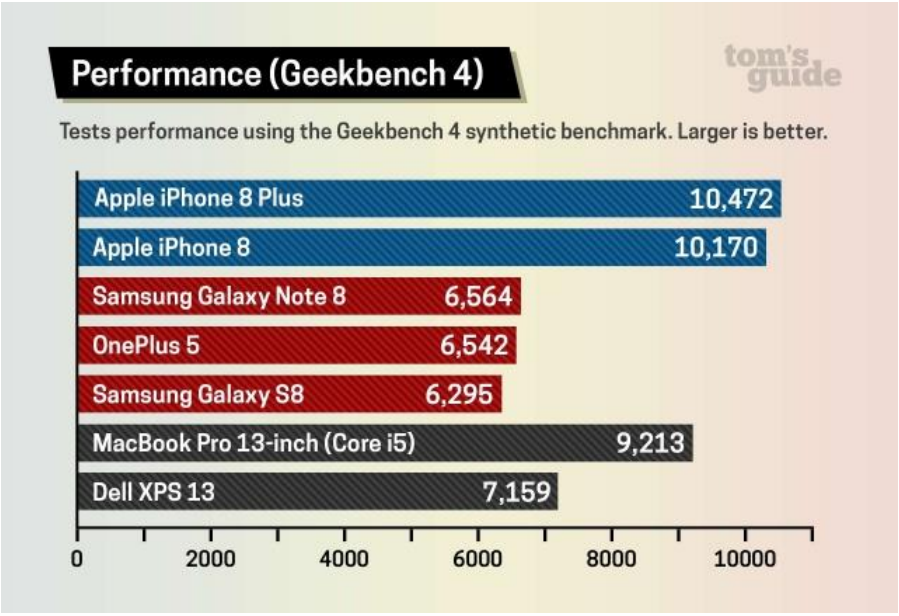

Ilustración 6. Comparativa de rendimiento entre los teléfonos inteligentes más potentes de 2017

Elección de la plataforma

Tras examinar las dos principales opciones que tenemos para realizar nuestra aplicación, la mejor plataforma en la que hacerla parece ser Android.

El motivo principal para tomar esta decisión es la gran diferencia de consumidores de teléfonos Android, que como hemos dicho anteriormente es de algo más del 85%. Para cualquier aplicación, cuanto mayor sea el mercado al que está destinado nuestro producto, más debemos priorizar ese sector.

Por otro lado, la licencia de desarrollador es más económica y hace que el gasto final de la aplicación sea ligeramente menor, sobre todo a largo plazo. Además, para desarrollar en iOS, se necesita un ordenador de la marca Apple, el cual no disponemos y su precio encarecería notablemente el presupuesto.

Por último, teniendo en cuenta que una de las limitaciones de este proyecto es la limitación de tiempo (300 horas), de hacer la aplicación para iOS habría que destinar parte de ese tiempo a aprender a usar el lenguaje Swift o Objective-C. Al haber cursado una asignatura en la que se programan aplicaciones Android, teniendo los conocimientos básicos del lenguaje, se podrá garantizar un acabado mejor del proyecto.

Análisis de las soluciones que hay en el mercado actual

"La Telerehabilitación es el uso y práctica de servicios para la rehabilitación a través de las TIC (Tecnologías de la información y comunicación) e Internet" [13].

“De las tecnologías aplicadas a la salud (e-Health), posiblemente la Telerehabilitación es la que consigue uno de los índices de eficacia más evidentes, considerando eficacia como el grado de optimización de los recursos sanitarios y la mejora de la calidad de vida de los pacientes” [14].

La mayoría de los servicios se dividen en dos categorías: la evaluación clínica y la terapia clínica. Actualmente, una gran cantidad de las soluciones empresariales correspondientes a la telerehabilitación se basan en la rehabilitación a distancia haciendo uso de las TIC para realizar el seguimiento y para indicar que ejercicios se deben realizar, normalmente con ejemplos gráficos en fotografías y vídeos. Estas plataformas suelen también ofrecer consultas on-line para realizar el seguimiento del paciente. Esto es de gran ayuda para aquellas personas que no se pueden desplazar, tanto por limitaciones físicas como por limitaciones de transporte. Además, cabe destacar la evidente reducción de costes a largo plazo (a corto plazo habría que preparar todo el material de las sesiones de rehabilitación).

Esta visión de la telerehabilitación es en la que se basa Telekin, pero además añade un valor mucho mayor. Además del seguimiento de la rehabilitación, asigna una serie de actividades y realiza el seguimiento de estas y de las sesiones. Pudiendo en el futuro, mediante inteligencia artificial poder incluso modificar las terapias dependiendo de los resultados de las sesiones.

Este tipo de rehabilitación es menos común en entornos empresariales, puesto que requiere un desarrollo de las soluciones y que el personal sanitario sea capaz de utilizar la herramienta. Debido a esta comunicación necesaria entre equipos de ingeniería y equipos sanitarios estas soluciones son menos comunes que las que explicábamos al principio de la sección. Además del incremento del coste a corto plazo para el desarrollo de las actividades y las plataformas.

Por tanto, podríamos clasificar la telerehabilitación en dos grandes grupos, cuya principal diferencia es la monitorización o no de los ejercicios de las sesiones. En el grupo de la rehabilitación sin seguimiento de las actividades entrarían las empresas con servicios de telerehabilitación mediante videos y consultas. En el otro grupo, en el que entraría Telekin, se realizan mediciones o seguimientos de las sesiones para controlar más eficazmente el progreso.

Existen numerosos métodos para realizar el control de los ejercicios realizados por el paciente, Telekin utiliza el sistema de Microsoft Kinect para detectar al paciente y sus movimientos.

La empresa multinacional española Indra, en 2008, inició una línea de desarrollo de telerehabilitación con el objetivo de ayudar a las personas con lesión medular o de ictus a afrontar la rehabilitación. De aquí surge Toyra, que es un producto de rehabilitación del miembro superior mediante un sistema de realidad virtual y captación de movimiento en tiempo real, realizado en colaboración con el Hospital Nacional de Parapléjicos de Toledo y la Fundación Rafael del Pino. Este proyecto en 2014 evolucionó al sistema TRAM, que además añadía rehabilitación en las extremidades inferiores [15].

El proyecto TRAM (TeleRehabilitación Audiovisual Motora) es un proyecto de desarrollo experimental realizado por Divisa iT en colaboración con Indra Sistemas S.A y financiado por el Ministerio de Industria, Energía y Turismo. Su objetivo principal es realizar el seguimiento continuo del estado del paciente y evolución, tanto en el centro sanitario como en el entorno domiciliario [16]. Para ello utiliza sensores inerciales de movimientos, Kinect para Windows y *joysticks* o *gamepads* con al menos 3 botones. Además, utilizan las tecnologías J2EE, Tomcat, MySQL, MS Visual C++ y OpenGL [17].

Ilustración 7. Diseño del proyecto Toyra/TRAM.

Otro ejemplo de plataforma de telerehabilitación es Play For Health o P4H. Esta herramienta desarrollada desde mediados de 2009 por el área de salud de la fundación Bit en colaboración con el Servicio de Salud de las Islas Baleares y el servicio de rehabilitación del hospital Son Llàtzer [18]. Play For Health pretende mejorar la forma de realizar la rehabilitación convencional mediante el uso de elementos tales como una webcam, unos guantes, una pista de baile, la Wibalance, el Wimote o la Kinect. El uso de todo este material permite la rehabilitación de multitud de zonas y aspectos, ya sea el equilibrio, la coordinación, el control postural, la movilidad fina o la de miembros inferiores o superiores. Además, cuenta con una plataforma para la gestión de la herramienta.

Uno de los sectores que más han apoyado la telerehabilitación son las universidades. Además de la Universidad de Valladolid, con su herramienta Telekin, que comentaremos más en detalle en el próximo capítulo, hay otras plataformas para la rehabilitación usando las nuevas tecnologías.

La UPV/EHU (Universidad del País Vasco o Euskal Herriko Unibertsitatea) ha diseñado y desarrollado un sistema de telerehabilitación llamado KiReS (Kinect Rehabilitation System). KiReS es un sistema basado en Kinect para Windows que permite a los fisioterapeutas definir terapias de rehabilitación personalizadas a los pacientes, pudiendo crear nuevos ejercicios simplemente realizando esos ejercicios en frente del sistema y registrándolos [19]. De esta forma, no se trata de telerehabilitación basada en

juegos, sino en ejercicios, lo cual lo hace más simple y permite al médico añadir sus propias sesiones de ejercicios.

Un equipo de investigadores del Labhuman-I3BH de la Universitat Politècnica de València (UPV) ha desarrollado junto médicos del Servicio de Neurorrehabilitación de los hospitales NISA Valencia al Mar y Sevilla-Aljarafe, y la empresa de tecnología de realidad virtual, Bienetec S.L. ha desarrollado un nuevo módulo del sistema de rehabilitación BioTrak, un nuevo sistema de telerehabilitación para pacientes afectados por daño cerebral que permite hacer terapia desde casa. BioTrak está basado en tecnología Cloud y permite llevar a cabo una evaluación continua y a distancia de la evolución de los pacientes mediante un ordenador portátil que el paciente puede conectar a su televisión, una cámara de profundidad (tipo Kinect) y una plataforma de captación de movimiento (como la Wii Balance Board) en la que realiza los ejercicios programados por los terapeutas diariamente [20].

Como podemos observar en los ejemplos proporcionados, actualmente hay alternativas en cuanto a la rehabilitación. No obstante, la mayoría de ellas parecen ser más un proyecto de investigación puesto en práctica en centros reales que un producto comercial que sacar al mercado.

Las últimas plataformas de rehabilitación que están surgiendo a día de hoy son las basadas en la realidad virtual, la realidad aumentada y la denominada realidad mixta. Pero aun así, lo que parece que será un referente en el futuro es la telerehabilitación basada en videojuegos, tal y como hace Telekin o el instituto Kennedy Krieger [21].

Capítulo 3. Estudio de la herramienta Telekin

Antes de empezar a programar la aplicación, vamos a analizar el sistema Telekin para saber claramente en que consiste y como debemos empezar a diseñar la solución.

Telekin es un sistema de telerehabilitación para discapacidades motoras y cognitivas, es decir, que usa una serie de servicios para la rehabilitación a través de las TIC (Tecnologías de la Información y Comunicación) e Internet.

"El objetivo principal de este sistema es el de estimular a los pacientes para realizar la terapia y que esta sea adecuada a sus patologías. Además, se quiere ofrecer a los especialistas (médicos, terapeutas) una herramienta para realizar un seguimiento del progreso de sus pacientes dondequiera que hagan los ejercicios, ya sea en casa o en el centro médico" [22].

El proyecto Telekin está dividido en módulos, los cuales han sido programados para que sean independientes unos de otros, de esta manera en caso de necesitar añadir algún módulo nuevo, los demás módulos no necesitarán ninguna modificación. Además, de esta forma resulta mucho más sencillo detectar fallos y minimizar el impacto de estos en caso de suceder.

Los módulos actuales que se encuentran en el sistema son:

- **Módulo web de familiares:** Este módulo muestra los datos del paciente al que está vinculado dicho familiar, además de la información de las sesiones realizadas por dicho paciente.
- **Módulo web del médico:** Los usuarios de este módulo hacen uso de una interfaz en la que pueden crear pacientes, archivarlos, modificarlos o incluso borrarlos. También pueden modificar la terapia de los pacientes y añadir sesiones nuevas seleccionando las actividades. Se pueden visualizar las sesiones antiguas en forma de gráfico y valorar el progreso del paciente a lo largo del tiempo.
- **Módulo web del Administrador:** Esta parte del sistema consiste en una interfaz web para que el administrador realice las acciones propias de su rol, que incluyen las acciones de los otros niveles. Entre estas acciones se encuentra la gestión de usuarios, tanto crearlos como eliminarlos o editarlos (nombre, alias, nivel de seguridad, etc.) también puede asignar pacientes a un médico en concreto. El administrador puede hacer uso de un configurador de juegos en el que cambiar ciertas propiedades de la herramienta. Además, también puede borrar, añadir y modificar los rangos articulares de las actividades.

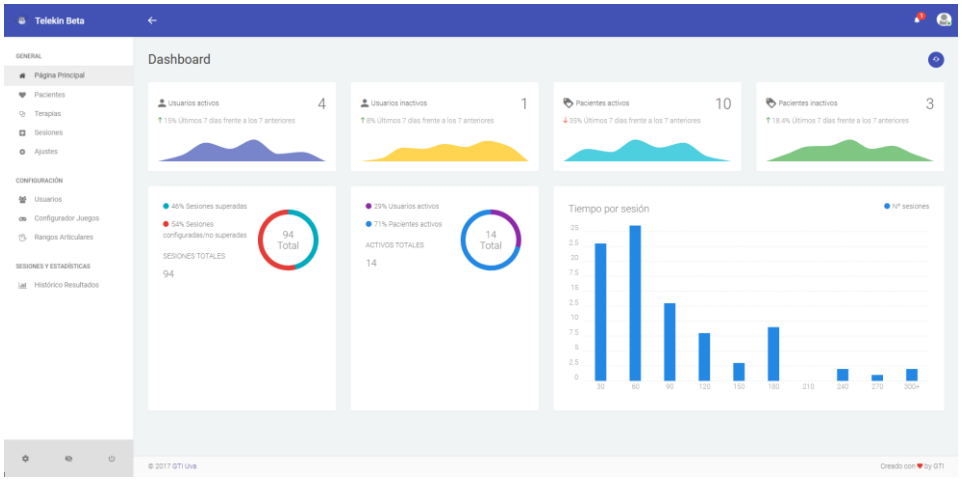

Ilustración 8. Módulo web del administrador

- **Módulo del juego:** Este módulo consiste en un entorno en 3D que contiene las actividades a realizar por el paciente. Una vez que el paciente está activado, puede acceder a un menú en el que el usuario elige su cuenta y las actividades que va a realizar de las que el médico le ha asignado. Estas actividades pueden ser ejercicios en los que se mide el progreso de la rehabilitación o juegos que hay que completar realizando una serie de movimientos.

Los movimientos que realizan los pacientes en el módulo del juego son capturados mediante dos dispositivos:

- **Microsoft Kinect v2:** Este dispositivo fabricado por Microsoft y que actualmente no se fabrica es el encargado de capturar el movimiento del todo el cuerpo del paciente para las actividades. Este dispositivo cuenta con una cámara 1080p que capta los movimientos de hasta 6 personas a la vez [23].

Ilustración 9. Actividad con Kinect

- **Leap Motion:** Este dispositivo desarrollado por la empresa de mismo nombre es el encargado de capturar el movimiento de las manos del paciente.

Ilustración 10. Actividad con Leap Motion

Capítulo 4. Diseño y desarrollo de la solución

Partimos de una base de datos de donde sacaremos los datos de los usuarios, pacientes, terapias, etc. Esta es la única herramienta que tendremos a nuestra disposición y será suficiente para realizar toda la aplicación:

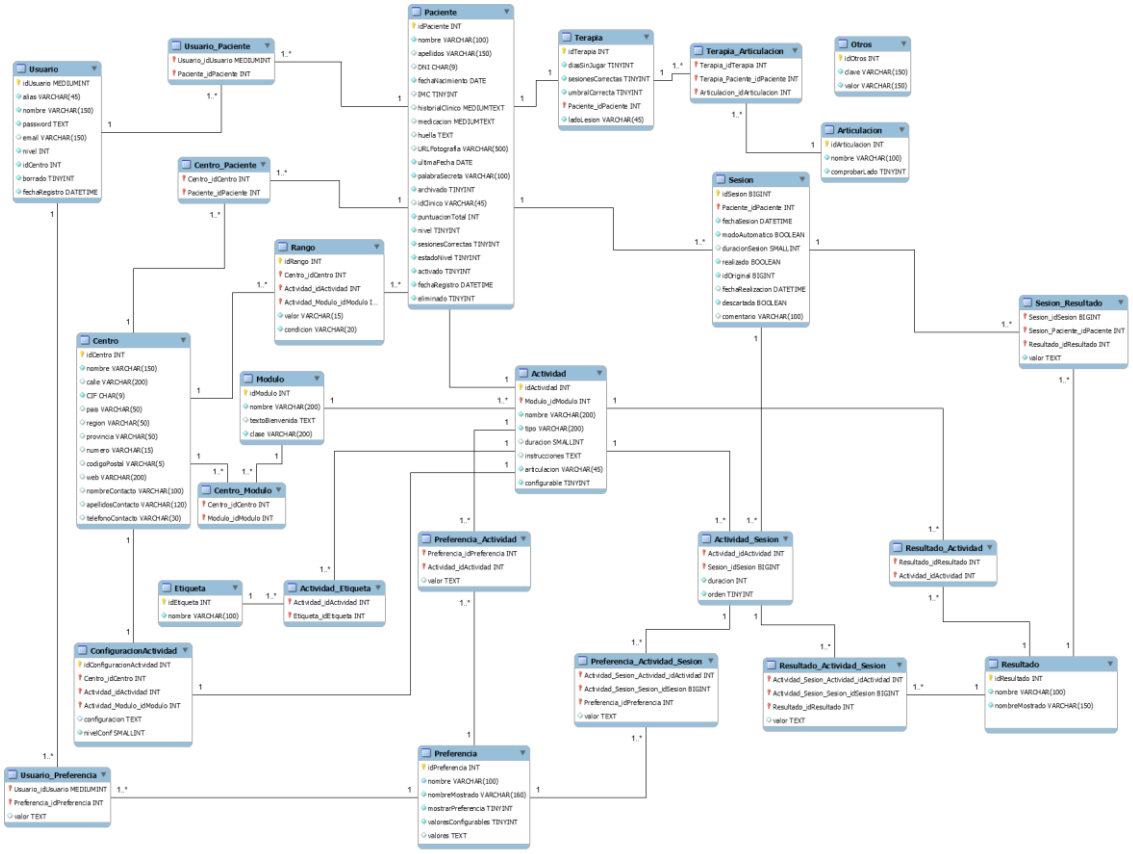

Ilustración 11. Diagrama de la base de datos de Telekin

Además, dispondremos de una copia de la base de datos que introduciremos en un servidor local, este será el que usemos en todo el proceso de desarrollo de la aplicación. Una vez que la terminemos, haremos una migración desde el servidor local al servidor del GTI donde se encuentran los datos reales de los pacientes.

Una vez que tenemos claro qué es lo que vamos a hacer, hay que centrarse en cómo lo vamos a hacer. Para la programación de aplicaciones Android hay diversas alternativas:

- App Inventor: Esta herramienta fue creada por Hal Abelson junto con un grupo de estudiantes. App Inventor está administrada por el Instituto Tecnológico de Massachusetts (MIT) y es un entorno de programación visual intuitivo para hacer aplicaciones basada en bloques [24]. Es una herramienta en línea que va por su segunda versión y está disponible en el enlace http://ai2.appinventor.mit.edu/.

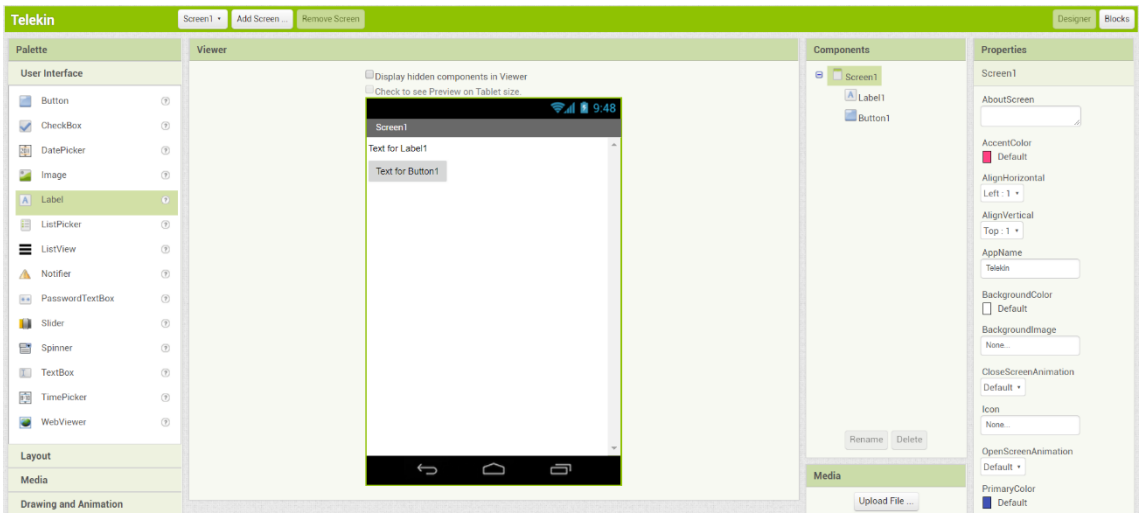

Ilustración 12. Interfaz de App Inventor

- Eclipse: El proyecto Eclipse fue originalmente creado por IBM en noviembre de 2001 y apoyada por un consorcio de vendedores de *software* [25]. A día de hoy, Eclipse es una de las principales alternativas para programar en un gran número de lenguajes, entre los que destaca java. Ha sido la herramienta principal de programación de aplicaciones hasta diciembre de 2014, cuando fue lanzado un nuevo programa llamado Android Studio.

- Android Studio: Es la herramienta oficial y recomendada para la programación de aplicaciones Android [26]. Fue presentada en mayo de 2013 y ahora mismo va por su versión 3.0.1, actualizada en noviembre de 2017 [27]. Este programa está totalmente enfocado a la programación de aplicaciones Android y cuenta con una serie de herramientas que ayudan a un desarrollo más eficiente.

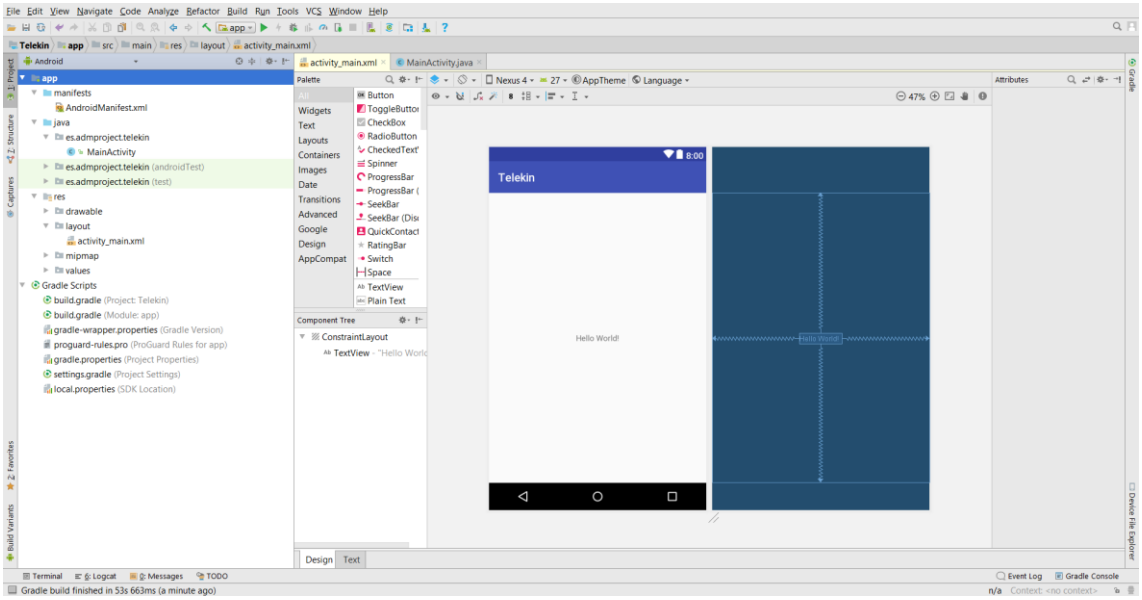

Ilustración 13. Interfaz de Android Studio

Para el desarrollo de nuestra aplicación usaremos Android Studio, ya que es la herramienta recomendada por Google y además contamos con experiencia con este

programa. Esta plataforma no nos pondrá ningún problema a la hora de programar y podremos acceder a todas las funcionalidades que necesitemos del teléfono.

Al iniciar Android Studio, nos aparece un asistente que nos hará una serie de preguntas para crear la aplicación vacía. Estas preguntas son:

- Nombre de la aplicación: Telekin
- Dominio de la compañía: gti.tel.uva.es
- Localización de proyecto: C:\Users\Oscar\AndroidStudioProjects\Telekin
- Nombre del paquete: es.uva.tel.get.telekin

El nombre del paquete se crea automáticamente con el nombre de la aplicación y el dominio de la compañía, aunque se puede modificar. El nombre del paquete debe ser único en el Google Play Store, por lo que, introduciendo el nombre de la compañía y el nombre de la aplicación dentro de ella, no debería haber ninguna aplicación igual. La localización del proyecto es el directorio local donde vamos a trabajar.

Al darle al botón “Siguiente” en el asistente nos pregunta que versión mínima del SDK (Kit de Desarrollo de Software) de Android queremos que soporte nuestra aplicación. Actualmente solo permite elegir desde la API (Interfaz de Programación de Aplicaciones) 14 en adelante. Cuando creamos la aplicación, permitía elegir cualquiera de las anteriores, pero escogimos la API 15: Android 4.0.3 (IceCreamSandwich) puesto que ya entonces alcanzaba aproximadamente el 100% de los dispositivos Android tal y como podemos ver en la Ilustración 14. Hay que tener en cuenta que la API 15 salió el 16 de diciembre de 2011 [28], y sabiendo que los dispositivos con la versión anterior (Android 2.3 Gingerbread) más potentes fueron actualizados, es normal que actualmente no haya dispositivos con versiones anteriores. Los teléfonos inteligentes tienen una esperanza de vida menor a los dos años [29], por lo que podemos estar totalmente seguros de que no estamos ignorando ningún dispositivo.

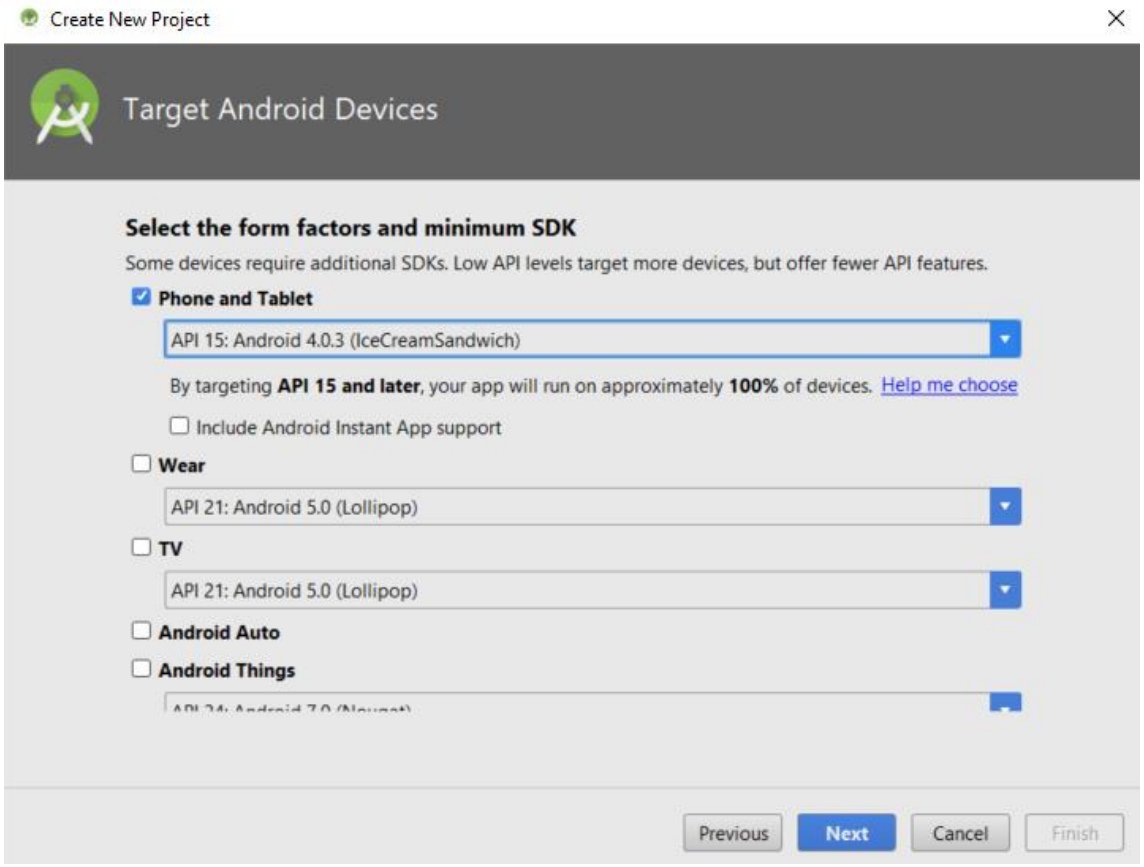

Ilustración 14. Asistente de creación de aplicaciones, elección del SDK mínimo

Por último, el asistente nos pregunta si queremos añadir una actividad al proyecto. En caso de no querer añadir ninguna, hemos terminado y pulsamos el botón de "Finalizar". Si queremos añadir una actividad, podemos elegir entre las siguientes:

- Basic Activity
- Bottom Navigation Activity
- Empty Activity
- Fullscreen Activity
- Google AdMob Ads Activity
- Google Maps Activity
- Login Activity
- Master/Detail Flow
- Navigation Drawer Activity
- Scrolling Activity
- Settings Activity
- Tabbed Activity

Una vez que salgamos del asistente ya tendremos creado nuestro proyecto. Ahora el siguiente paso es asegurar que nuestra aplicación no se pierda en caso de algún

fallo, o asegurar que podemos recuperar una versión anterior. Para ello añadiremos un control de versiones al proyecto, como por ejemplo Git. De entre todos los controles de versiones utilizaremos este puesto que es el que ya se utiliza en el Grupo de Telemática e Imagen (GIT). Para añadir Git a nuestro proyecto tan solo tenemos que ir a *VCS* > *Import into Version Control* > *Create Git Repository*. Seleccionamos la ruta de nuestro proyecto en local y después en la consola de Android Studio escribimos lo siguiente:

```
git remote add origin http://sigflot.gti.tel.uva.es:443/TELEK
IN/TelekinAppAndroid.git
```

Para añadir los ficheros al repositorio basta con pulsar el botón derecho encima y seleccionar *Git* > *Add*. Una vez que hayamos añadido todos los ficheros que queramos tener en el repositorio podemos subir los cambios accediendo al menú *VCS* > *Commit Changes*… En la ventana emergente que nos aparece podemos añadir un comentario a esa versión y seleccionar los ficheros que queramos incluir, por defecto se seleccionarán todos aquellos que hayan sido modificados desde el *commit* anterior. Para terminar, basta con pulsar el botón "Commit" y los cambios se guardarán.

Cuando se realiza una aplicación, normalmente se suele realizar primero el diseño de la aplicación y posteriormente la programación, de esta forma conseguiremos ir mucho más rápido a la hora de crear las vistas y ahorraremos tiempo. Para el diseño de las vistas de la aplicación utilizaremos una herramienta en línea llamada NinjaMock, que es de pago, pero tiene una versión gratuita que podremos utilizar [30]. Esta aplicación web nos permite realizar diseños de una forma muy sencilla y esquemática.

Actividad de inicio de sesión

Empezaremos por la pantalla inicial, será una pantalla de "*login*" en la que el usuario pondrá su alias y su contraseña para acceder a la plataforma. Esta pantalla deberá ser muy sencilla, sin demasiados objetos que aparten la atención del contenido principal.

Ilustración 15. Diseño de la pantalla inicial de Telekin

En cuanto a la elección de colores, los que usemos en esta pantalla serán los que usemos en toda la aplicación, para que esta tenga consistencia y no de la sensación de haber cambiado de aplicación. En Android se definen 3 colores principales para una aplicación:

- PrimaryColor: Como bien da a entender el nombre, es el color principal de la aplicación, y, alrededor de él girarán todos los demás colores.
- DarkPrimaryColor: Este color debe ser muy parecido al principal pero más oscuro, se usará automáticamente para la barra de notificaciones del *Smartphone* (donde aparece la hora, notificaciones y batería). Además, resulta muy atractivo combinar los dos tonos de color principal para botones, títulos y demás elementos de la interfaz.
- AccentColor: Este color suele ser uno totalmente distinto a los anteriores, y con él se pretende que destaque mucho con respecto al resto de elementos. Suele aparecer en las barras de progreso o botones flotantes.

Para la elección de los colores haremos uso de la herramienta on-line “material design pallete” [31]. Gracias a esta página web, podemos seleccionar un color con el que nos muestra el código hexadecimal del *PrimaryColor* y el *DarkPrimaryColor*, en nuestro caso hemos seleccionado el “Light Blue”. Una vez seleccionado un color, si seleccionamos otro, este aparecerá como el *AccentColor*, en nuestro caso no hemos

seleccionado ninguno puesto que utilizaremos sólo el azul y el blanco para nuestra interfaz. Con estos colores conseguiremos la seriedad y el estilo sobrio que la aplicación necesita. Por tanto, utilizaremos el blanco como *AccentColor*.

El título de la aplicación, al ser lo más representativo, deberá tener una tipografía especial. A la hora de elegir el tipo de letra tendremos en cuenta tanto las asociaciones perceptuales como las asociaciones directas que provoca la fuente para que sea coherente y congruente con el producto, que en este caso es una plataforma para la telerehabilitación. El tipo de letra provoca en el lector una serie de asociaciones tanto emocionales como semánticas que hacen que interprete el producto antes de usarlo, por lo que una evaluación negativa en un determinado contexto puede provocar el rechazo por parte de los usuarios [32].

La decisión principal a la hora de elegir un tipo de letra es seleccionar un tipo, ya sea "Sans Serif" o "Serif". Las diferencias entre estos dos tipos son muy obvias en la Ilustración 16. Elegiremos el tipo "Sans Serif" puesto que es una alternativa que da la sensación de ser más sencilla, limpia y menos seria que "Serif".

Ilustración 16. Diferencias entre fuentes Sans Serif y Serif

Para buscar tipografías gratuitas, la opción más segura es "Google Fonts". Que además cuenta con un buscador que permite descartar tipos de letras que no nos interesan, como el ya descartado Serif, Handwriting (al ser una herramienta médica no daría la impresión de seriedad), Display (las tipografías no suelen ser demasiado sobrias) o Monospace (son tipos de letra del estilo a las consolas de comandos de los ordenadores). De todas las tipografías disponibles, elegimos la llamada "Dosis" [33], puesto que es sencilla, sobria pero no seria, y algo alargada. Además, su descarga y uso es totalmente libre.

Android de forma nativa no permite añadir fuentes externas, por lo que añadiremos una librería llamada "Calligraphy". Con esta sencilla herramienta podemos

cambiar el tipo de letra de un solo elemento con el atributo “fontPath” o cambiarlo en toda la aplicación dentro del fichero “style.xml” [34].

Para hacer un poco más dinámica y profesional esta pantalla vamos a introducir una pequeña animación de entrada para la caja donde irán las credenciales y para el título de la aplicación.

Para hacer las animaciones en nuestra aplicación utilizaremos una librería externa que hará mucho más sencilla la tarea. Esta librería se llama ViewAnimator y permite hacer animaciones con unas pocas líneas de código, además, permite encadenar animaciones para que, al terminar una, empiece otra. ViewAnimator tiene un buen reconocimiento en la web Github y tiene un mantenimiento que asegura no quedarse obsoleta.

Para añadir librerías externas en Android Studio tenemos que abrir el fichero “build.gradle (Module: app)” y en el apartado “dependences” añadir ViewAnimator con:

```
compile 'com.github.florent37:viewanimator:1.0.5'
```

Cuando el usuario pulsa el botón de “ENTRAR”, el alias y la contraseña que ha introducido deberán enviarse al servidor y este deberá responder con todos los datos del usuario y los pacientes que tiene este usuario. Cuando hagamos la conexión con el servidor no basta con hacer una petición desde el *UI thread* o hilo de la interfaz gráfica de usuario. El sistema operativo Android está diseñado para hacer una tarea simultáneamente por hilo, cuando hacemos una petición HTTP, el hilo se bloquea hasta que recibe la respuesta. Por otra parte, si el *UI thread* se bloquea más de 3 segundos, el sistema operativo interpreta que la aplicación se ha bloqueado y la cierra. Para hacer esta conexión tenemos varias alternativas: AsyncTask, Volley y Retrofit.

- AsyncTask: Esta alternativa permite el uso adecuado del *UI thread* o hilo de la interfaz gráfica de usuario de una forma sencilla. La clase “AsyncTask” permite realizar operaciones en segundo plano y publicar los resultados de estas en el hilo de la interfaz de usuario sin tener que recurrir al uso de *threads* o *handlers*. Se define una tarea asíncrona que corre en un hilo en segundo plano y el resultado es publicado en el *UI thread*. Una tarea asincrónica está definida por 3 tipos genéricos (Params, Progress y Result) y 4 pasos (onPreExecute, doInBackground, onProgressUpdate y onPostExecute) [35].
- Volley: Es una herramienta creada por Google para realizar peticiones HTTP de manera sencilla y rápida. Volley ofrece grandes ventajas, entre las más importantes [36]:
    - Programación automática de solicitudes de red.
    - Múltiples conexiones de red simultáneas.
    - Almacenamiento en memoria caché transparente de memoria y disco con coherencia de caché HTTP estándar.
    - Soporte para la priorización de solicitudes.

- Solicitud de cancelación API. Puede cancelar una sola solicitud o puede establecer bloques o ámbitos de solicitudes para cancelar.
- Facilidad de personalización, por ejemplo, para reintento y rebobinado.
- Una ordenación sólida que facilita el llenado correcto de su IU con datos obtenidos de forma asíncrona de la red.
- Herramientas de depuración y rastreo.

- Retrofit: Es un cliente REST para Android, desarrollada por Square, muy simple y fácil de usar. Soporta peticiones de los 5 tipos HTTP básicos (GET, POST, PUT, DELETE y HEAD). Es muy flexible y ofrece grandes ventajas como la alternativa anterior. También permite la conversión de datos, de tal forma que podemos recibir la información en formato JSON o XML entre otros [37].

De estas tres herramientas, la primera que podríamos descartar es el AsincTask, debido principalmente a lo básica que resulta esta alternativa. De utilizarla, habría que implementar manualmente herramientas complementarias como por ejemplo la recepción de los datos en el formato elegido.

Las otras dos alternativas resultan muy útiles y fáciles de usar. Sin embargo, a la hora de medir el rendimiento, podemos ver en la Ilustración 17 [38] cómo Retrofit es más rápido que Volley.

	One Discussion	Dashboard (7 requests)	25 Discussions
AsyncTask	941 ms	4,539 ms	13,957 ms
Volley	560 ms	2,202 ms	4,275 ms
Retrofit	312 ms	889 ms	1,059 ms

Ilustración 17. Comparativa de tiempos entre AsyncTask, Volley y Retrofit

Por tanto, utilizaremos la herramienta Retrofit para el intercambio de datos entre la aplicación cliente y el servidor.

Para integrar la última versión de Retrofit (v2.4) en nuestro proyecto tan sólo es necesario incluir la dependencia en el Gradle como ya hicimos anteriormente con el ViewAnimator:

```
compile 'com.squareup.retrofit2:retrofit:2.4.0'
compile 'com.squareup.retrofit2:converter-gson:2.4.0'
compile 'com.google.code.gson:gson:2.8.0'
```

La primera dependencia es el paquete de Retrofit, la segunda es el conversor GSON para los datos en JSON, y el último es el paquete GSON. Si cualquiera de las tres dependencias falta, la aplicación no funcionará.

Las conexiones serán de tipo POST puesto que llevan datos sensibles que deben estar ocultos cuando se envíe al servidor. Una conexión de tipo GET lleva los datos en la URL del tipo “http://URL?tipo1=valor1&tipo2=valor2”.

La clase principal de Retrofit en nuestro proyecto toma el nombre de “RESTHelper”. Esta clase cuenta con un patrón singleton que se encarga de que solo haya un único objeto del tipo “RESTHelper”:

```
private static RESTHelper mInstance = null;
public static RESTHelper getInstance() {
    if(mInstance == null)
        mInstance = new RESTHelper();
    return mInstance;
}
```

En el constructor de nuestra clase principal de Retrofit se definen la URL del servidor al que se harán las consultas y el conversor (en este caso GSON). Además, se crea un servicio al que le pasaremos una interfaz con cada una de las páginas a las que se conectará la aplicación. Con cada conexión se especificará:

- Página a la que hacer la conexión (Ej.: iniciarSesion.php).
- Tipo de conexión (Ej.: POST).
- Parámetros que se enviarán al servidor (Ej.: Usuario y contraseña).
- Objeto que recogerá lo que devuelva el servidor (Ej.: objeto Usuario).

```
private RESTHelper() {
  super();
  Retrofit telekinRetrofitInstance = new Retrofit.Builder()
            .baseUrl(ConstantsHelper.TELEKIN_BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();
 telekinService = telekinRetrofitInstance
            .create(TelekinAPIEndpointInterface.class);
 }
```

En cuanto a las conexiones contenidas en TelekinAPIEndpointInterface serán del tipo:

```
@FormUrlEncoded
@POST("iniciarSesion.php")
Call<LoginResponse> iniciarSesion(@Field("user") String user,
                                  @Field("pass") String pass);
```

Por último, en la clase “RESTHelper” incluiremos los métodos de cada conexión encargados de recoger los datos e interpretar si la conexión ha sido satisfactoria o se ha producido algún tipo de error:

```
public void iniciarSesion(final TelekinLoginListener listener,
  String user, String pass){
  Call<LoginResponse> call = telekinService.iniciarSesion(user,
          pass);
  call.enqueue(new Callback<LoginResponse>() {
    @Override
    public void onResponse(Call<LoginResponse> call,
      Response<LoginResponse> response) {
      if (response != null && response.body() != null ){
        if(response.body() != null && response.body().isOK()) {
          listener.loginOK(response.body().getUsuario()
                          ,response.body().getPacientes());
          return;
        }
        try {
          listener.loginKO(response.body().getError().get(0));
        } catch (Exception ex) {
           ex.printStackTrace();
           listener.loginKO(ex.toString());
        }
      }
    }

    @Override
    public void onFailure(Call<LoginResponse> call, Throwable t){
       listener.loginKO(t.getLocalizedMessage());
     }
  });
}
```

En nuestro caso haremos uso de un *listener* llamado "TelekinLoginListener" que es una interfaz que hará accesible los métodos loginOK y loginKO desde la clase desde la que se llame al método "iniciarSesion".

Una vez hecho todo lo anterior, nuestra aplicación tiene casi todo lo necesario para recibir los datos del servidor. Solo falta añadir el permiso de internet en el "AndroidManifest.xml", si este permiso no está, no será posible conectarse con la base de datos, no obstante, no se producirá ningún error que cierre la aplicación puesto que Retrofit hará la gestión de excepciones. Añadimos:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Como se puede apreciar en la Ilustración 15, hay un *CheckBox* con el texto "Recuerdame", que el usuario deberá marcar si desea que en los siguientes inicios de sesión no le pida las credenciales. Esta preferencia se guardará en la memoria local del teléfono junto con el nombre de usuario, la contraseña y todos los datos del usuario que nos devuelva el servidor. Como cabría esperar, solo se almacenarán si el inicio de sesión ha sido satisfactorio. Para el almacenamiento de esta información utilizaremos la clase

SharedPreferences. Esta clase nos permite almacenar información del tipo clave-valor, que nos resulta muy útil para estos datos (ej. Usuario - user123) [39].

Otra alternativa para almacenar los datos sería hacer una base de datos local, pero la consulta, creación y modificación de esta llevaría mucho más tiempo que nuestra elección. Además, solo almacenaremos los datos de un solo usuario a la vez, es decir, si se sale de la sesión se borran los datos del usuario y deberá iniciar sesión otra vez. La sesión solo se guardará si el usuario se sale de la aplicación sin haber hecho un "*logout*".

Cuando se inicie la aplicación y esta tenga un usuario almacenado con la opción de recordarlo, la interfaz de usuario no mostrará el cuadro en el que se rellenan las credenciales y en su lugar mostrará un "ProgressBar" circular para mostrar al usuario que está recibiendo los datos del servidor. Siempre que esté el cuadro de inicio de sesión no estará el *ProgressBar* y viceversa.

Para evitar que, en caso de tener una muy buena conexión de internet, esta pantalla inicial reciba los datos muy rápido y desaparezca casi sin verla, pondremos un retardo mínimo de 2 segundos (solo en el caso del usuario almacenado). Si la pantalla desaparece demasiado rápido, da la sensación al usuario de que hay información que no ha visto o que la aplicación no ha funcionado como debería. El retardo es lo suficientemente pequeño para no resultar molesto y lo suficientemente grande para que el usuario sepa lo que está pasando.

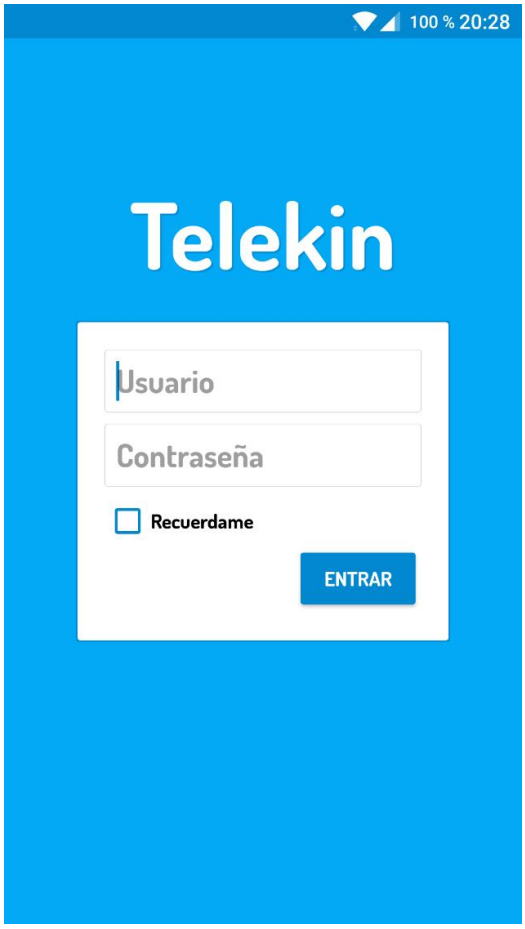

Ilustración 18. Diseño final de la pantalla de inicio de sesión

Como cabría esperar, en el cuadro de edición de texto donde irá la contraseña, irá codificado con los típicos caracteres "*" propios de este tipo de formularios.

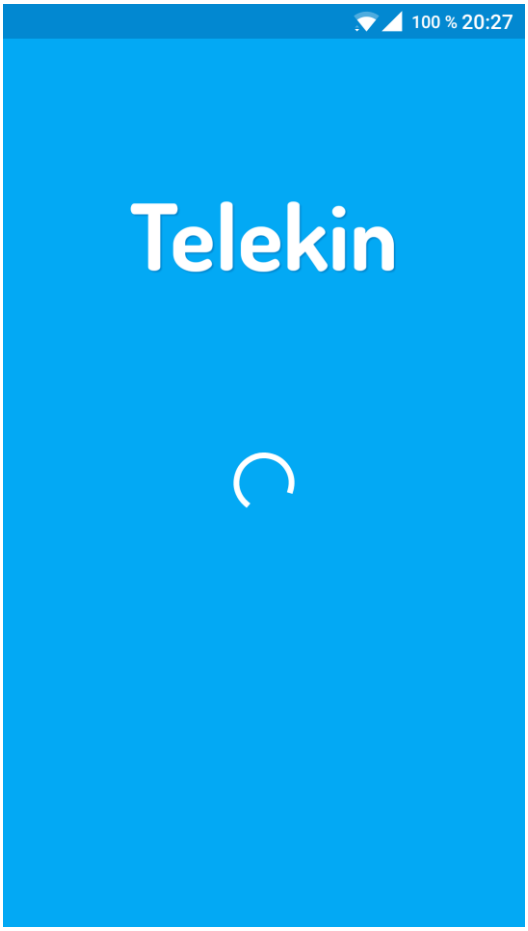

Ilustración 19. Diseño final de la pantalla de inicio de sesión cuando esta está guardada

En la parte del servidor, deberemos crear un servicio que responda con los datos del usuario y los pacientes asociados a este cuando reciba una petición con el nombre de usuario y su contraseña. Estos parámetros se recibirán por POST de la siguiente forma:

```
$usuario = $_POST['user'];
$password = $_POST['pass'];
```

Con estos datos haremos dos consultas a la base de datos, una para obtener todos los datos del usuario y los datos de todos los pacientes asociados al usuario registrado, tanto los de los pacientes activos como los archivados, los borrados no serán necesarios puesto que estos no pueden ser recuperados por un usuario de nivel médico. Las consultas a la base de datos para obtener esta información son:

```
SELECT * FROM Usuario WHERE alias='$usuario' and password='$password'
SELECT * FROM Paciente WHERE idPaciente IN (
      SELECT paciente_idPaciente
      FROM  Usuario_Paciente
      usuario_idUsuario IN (
```

```
            SELECT idUsuario FROM Usuario WHERE alias='$usuario' and
password='$password')
      )"
```

Con todos estos datos, podemos enviarlos a la aplicación cliente en formato JSON donde los pacientes se enviarán como un array.

Actividad de pantalla inicial

Una vez que los datos fueron verificados en el servidor, este respondió con los datos y fueron correctamente recibidos, se pasará a la siguiente *Activity* de la aplicación, que será un *dashboard* donde se verán algunas estadísticas de la aplicación. Con vistas a que en implementaciones futuras se muestren unos *logs* con las últimas actividades registradas tanto por los pacientes como por el médico.

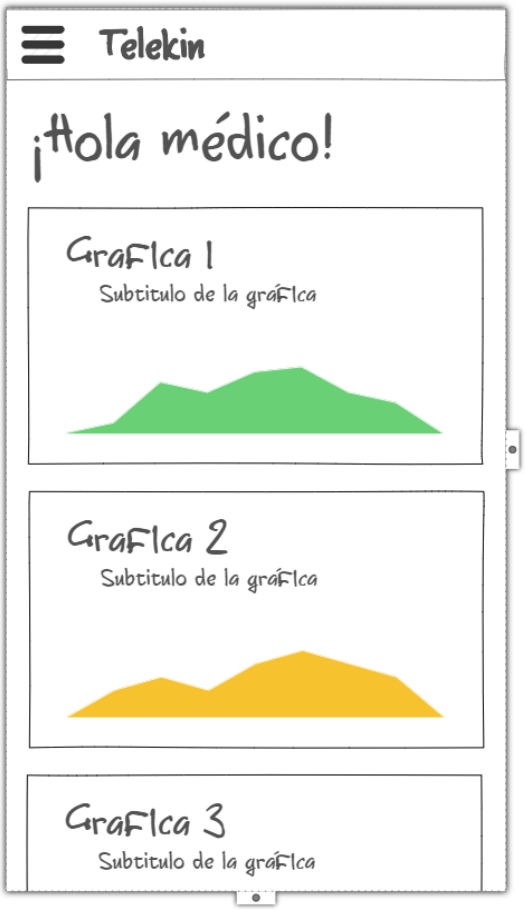

Ilustración 20. Diseño de la pantalla de bienvenida de Telekin

La página principal de la aplicación será muy sencilla y limpia, de tal forma que los usuarios puedan ver claramente la información. Para empezar, la pantalla tendrá un saludo al usuario y varias gráficas que mostrarán la tendencia de los pacientes. Estas gráficas mostrarán los pacientes asociados, los inactivos, los activos y los activados, y la comparación de los 7 últimos días con los 7 anteriores, tal y como aparece en la versión web.

Actualmente las gráficas son simples imágenes, con ellas conseguimos que los datos sean más atractivos al lector y cobren un mayor significado. En implementaciones futuras estas gráficas serán reales y reflejarán datos veraces.

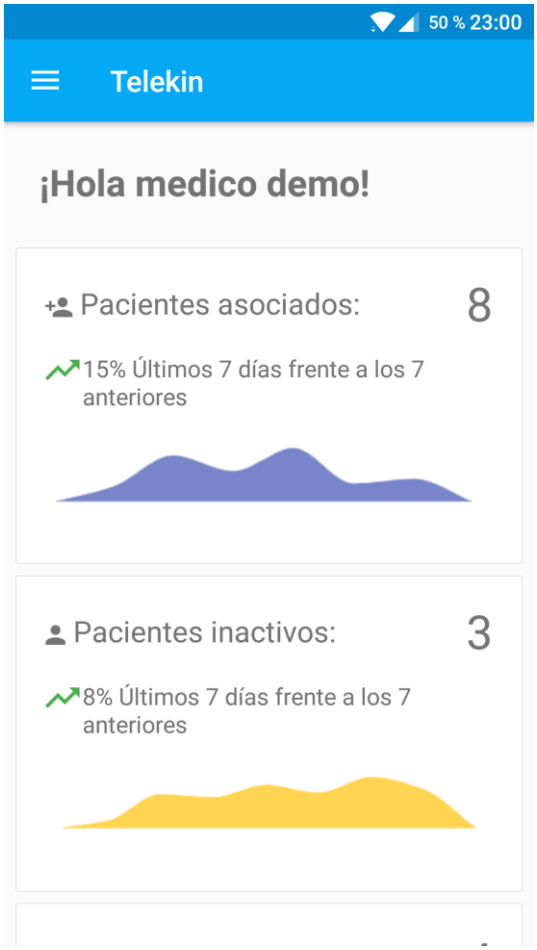

Ilustración 21. Diseño final de la pantalla principal de Telekin

Junto a los títulos de gráfica y a la descripción hay iconos que representan el contenido de las mismas y la tendencia de los datos respectivamente. Estas pequeñas imágenes representativas han sido añadidas mediante una librería llamada "*Iconify*". Gracias a esta librería se pueden añadir una serie de iconos de forma muy sencilla y sin tener que descargarlos previamente. Añadiremos *Iconify* al *Gradle* como hicimos con las librerías anteriores:

```
compile 'com.joanzapata.iconify:android-iconify-fontawesome:2.2.2'
compile 'com.joanzapata.iconify:android-iconify-material:2.2.2'
```

Con estas dos dependencias nuevas, podremos añadir iconos de *Fontawesome* y *Material Design* de forma sencilla. *Iconify* permite además añadir otras fuentes desde las que añadir iconos, pero nosotros nos centraremos en estas dos puesto que *FontAwesome* es utilizada en la plataforma web de Telekin y *Material Design* es lo más utilizado en el sistema operativo Android.

Las ventajas principales de usar esta librería es ahorrar tiempo en el desarrollo, puesto que no hace falta ir a la fuente a descargar los iconos, y conseguir una visualización totalmente óptima de los iconos, sea cual sea el tamaño y la densidad de la pantalla del dispositivo que lo muestra. Al ser iconos "*vector*" nos aseguramos de que en ningún momento se vean pixelados o distorsionados [40].

Menú lateral desplegable

Como vemos en la Ilustración 21, en la esquina superior izquierda hay tres líneas horizontales que en Android es la forma de representar un menú desplegable. Al pulsar este botón aparece un menú de navegación con las secciones principales de la aplicación. Estas secciones varían dependiendo del nivel de privilegios de usuario, puesto que esta aplicación se centra en los usuarios de tipo "médico" las secciones son:

- Página principal – Esta es la pantalla principal de la aplicación de donde hemos accedido a este menú. Puesto que este menú estará presente en las secciones principales, este elemento deberá estar presente para siempre que se quiera volver a ella.
- Pacientes – En esta sección se podrá ver la lista de los pacientes asociados. Este elemento de la aplicación es muy importante, desde él se podrán realizar las acciones más importantes de la plataforma. Estas acciones son ver los datos del paciente, ver su terapia, ver sus sesiones, editar los datos del paciente y asignarle sesiones. Además, los pacientes podrán ser activados, desactivados, archivados, recuperados y eliminados.

  Un paciente archivado no se mostrará en la lista principal y no se podrán realizar acciones sobre él excepto recuperarlo. Los pacientes que sean recuperados pasarán de la lista de archivados a la lista de pacientes principal, mientras que los archivados podrán ser eliminados y no se recuperarán a no ser que lo haga un usuario con permisos de administrador. Para que un paciente pueda realizar los ejercicios de telerehabilitación, antes debe estar activo, los pacientes que hayan sido activados serán visibles para la aplicación de Kinect. Una vez que un paciente haya realizado los ejercicios pertinentes de la sesión, este podrá ser desactivado.
- Terapias – Desde esta sección se pueden consultar los detalles de las terapias de cada uno de los pacientes, y permite modificarlas. Telekin está diseñado para que los pacientes tengan únicamente una terapia, y esta pueda tener diferentes sesiones con distintas articulaciones.
- Sesiones – Aquí se podrá ver una lista de todas las sesiones de todos los pacientes, tanto las finalizadas, como las que aún no han sido realizadas. Además, también se podrán eliminar las sesiones no realizadas y descartar las que han sido completadas. Las sesiones completadas mostrarán un gráfico de los resultados de la sesión y el progreso de la misma dependiendo del tipo de ejercicio realizado.

- Ajustes – Desde esta sección se podrán modificar algunos aspectos relacionados con la aplicación. Esta sección obtendrá los ajustes de la base de datos del servidor. En futuras implementaciones habrá ajustes locales que enviarán los valores al servidor, como por ejemplo las políticas de notificaciones cuando se introduzcan los *logs* en el sistema.
- Cerrar sesión – Al pulsar sobre este elemento del menú, se regresará a la pantalla de inicio de sesión anterior. Si fue pulsada la opción de recordar al usuario, esta será desechada y deberá volver a introducir las credenciales para acceder a la plataforma de nuevo.

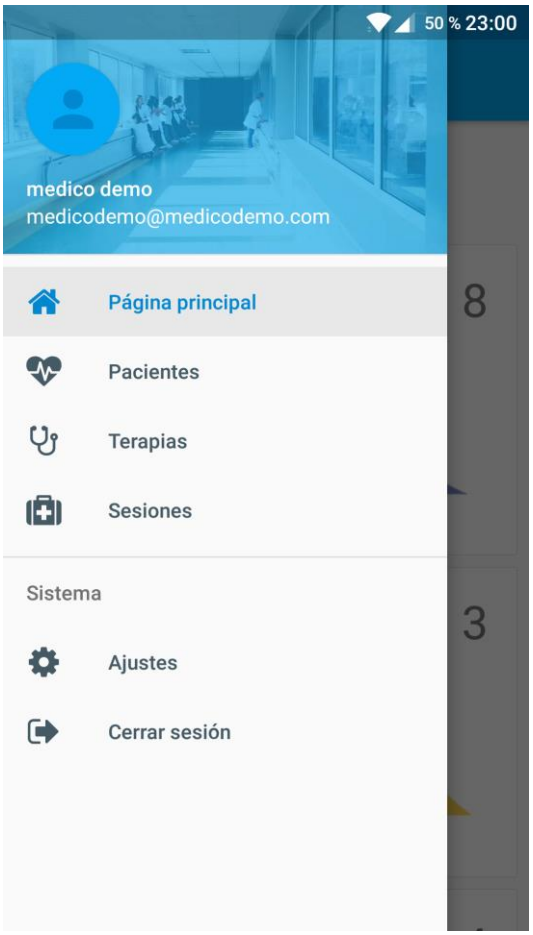

Ilustración 22. Diseño final del menú desplegable

La Ilustración 22 nos enseña como las secciones serán separadas en dos grupos principales, el principal en el que se accederá a las acciones que muestran y modifican información de la plataforma, y el de las acciones que tienen que ver con la utilización de la aplicación, a este último grupo lo llamaremos "sistema". Además, en la parte superior del menú tendremos una fotografía del usuario y justo debajo su nombre y apellidos sobre su dirección de correo electrónico.

Para poder hacer este menú recurriremos a una librería externa llamada *MaterialDrawer*. Este menú desplegable toma la base del *DrawerLayout* nativo de Android, pero añade muchas más funcionalidades. Aunque actualmente no sacamos todo el partido a este panel, al haber añadido esta librería, en futuras implementaciones

se podrán añadir etiquetas a las secciones para advertir de notificaciones, crear un *MiniDrawer* para la versión tablet o añadir varias cuentas. Gracias a *MaterialDrawer* podremos asegurarnos el poder implementar características más avanzadas y útiles en el futuro [41].

Para la implementación de este panel desplegable, vamos a introducir el código que lo genera en una clase especial que se llamará "*MaterialDrawer*". El objetivo de esta clase es crear el *NavigationDrawer* mediante un constructor al que le indiquemos una serie de parámetros. Como hay varias pantallas que crearán este menú, es importante que se pueda reutilizar el código y no haya que construir varias clases muy similares, lo que provocará es que al aumentar tanto el número de líneas de código, los errores aumenten. Para crear un NavigationDrawer en nuestra aplicación tendremos que llamar a la clase de la siguiente manera:

```
MaterialDrawer(act, toolbar, listaPacientes, itemSelected);
```

En el constructor anterior, los parámetros son:

- act – Este parámetro es del tipo *Activity* y sirve para acceder a los elementos de la interfaz de usuario que creamos en la actividad y para acciones típicos de una actividad de Android.
- toolbar – Este parámetro es del tipo *Toolbar* y es la barra de tareas superior típico de las aplicaciones de *smartphones*, especialmente en los de Android. Esta variable es necesaria para la creación del MaterialDrawer, aunque nosotros no lo usaremos para nada más que la instanciación del mismo.
- listaPacientes – Este parámetro es del tipo *List<Paciente>*. Como es esta clase la que lanzará las nuevas actividades, tendremos que enviar la lista de pacientes a las nuevas pantallas. No obstante, en la sección de pacientes pediremos al servidor nuevamente la lista.
- itemSelected – Este parámetro es del tipo *double* y nos sirve para decir al menú cual es el apartado en el que estamos para que lo destaque e indique al usuario donde se encuentra.

Las clases principales de la librería MaterialDrawer que vamos a utilizar y las formas de implementarlas son las siguientes:

```
result = new DrawerBuilder()
  .withActivity(act)
  .withToolbar(appbar)
  .withSelectedItem(itemSelected)
  .addDrawerItems(
      new PrimaryDrawerItem()
        .withName(R.string.item1)
        .withIcon(icon)
        .withTextColor(color)
```

```
        .withIdentifier(1),
      new SectionDrawerItem().withName("Section"),
      new PrimaryDrawerItem()
        .withName(R.string.item2)
        .withIcon(icon)
        .withTextColor(color)
        .withIdentifier(9)
  )
  .withOnDrawerItemClickListener(new
     Drawer.OnDrawerItemClickListener(){
       @Override
       public boolean onItemClick(View view, int position,
          IDrawerItem drawerItem) {
          if (drawerItem != null) {
             Intent intent = null;
             switch ((int) drawerItem.getIdentifier()){
               case 1:
                 act.finish();
                 intent = new Intent(act, NewActivity.class);
                 intent.putParcelableArrayListExtra("ListaPacientes",
                   listaPacientes);
                 act.startActivity(intent);
                 break;
               case 2:
                 //...
                 break;
             }
          }
          return false;
       }
     })
  .build();
```

Actividad de administración de pacientes

Pulsando en la sección “Pacientes” del menú desplegable accederemos a la actividad donde se listan todos los pacientes asociados al usuario que no han sido archivados o borrados.

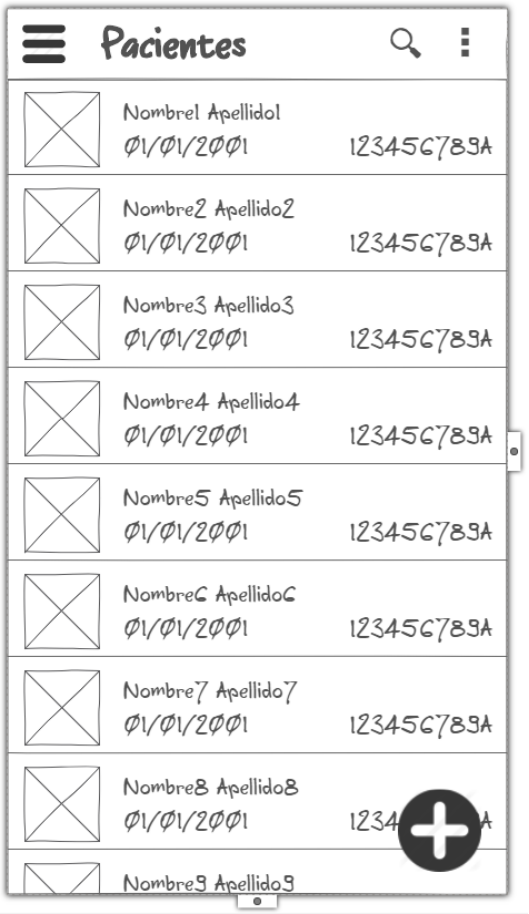

Ilustración 23. Diseño de la pantalla de listar pacientes

Cada elemento de la lista estará formado por una fotografía del paciente, su nombre completo, su fecha de nacimiento y su DNI. Como esto es una lista para identificar a los pacientes, no consideraremos añadir el identificador clínico, puesto que es un dato que no suele ser conocido, o al menos, recordado.

La actividad cuenta con un buscador, en el que, pulsando la lupa de la barra de acción superior, se puede encontrar un paciente por su nombre, apellidos, dni o identificador clínico. A la derecha del botón del buscador podemos ver tres puntos verticales que representan un menú desplegable típico en Android. Al presionar este menú veremos la opción de acceder a la lista de pacientes archivados.

En la actividad que lista los pacientes archivados, veremos la lista con los nombres completos y las fotografías de pacientes activados. Al pulsar sobre cualquiera de ellos los recuperaremos. A la derecha de cada elemento de la lista aparecerá un icono con un cubo de basura (universalmente representativo para eliminar un elemento). Al pulsar cualquiera de las dos opciones, aparecerá una ventana emergente para confirmar que se desea realizar la acción seleccionada.

Tal y como aparece en la Ilustración 23, en la esquina inferior derecha habrá un botón flotante (en Android FloatingActionButton o FAB) con un símbolo “+”. Con este botón se accederá a una nueva actividad en la que se podrá añadir un nuevo paciente.

Como hay una considerable cantidad de acciones que se pueden hacer sobre un paciente (ver, activar, desactivar, editar o archivar), sería un error poner botones asociados a cada elemento de la lista, tanto de forma visual como de manera práctica. Como alternativa, usaremos la funcionalidad de Android de mantener pulsado un elemento para que aparezca un menú emergente con las opciones disponibles. Además, como la acción más común será la de ver los detalles del paciente seleccionado, al hacer una pulsación simple, se accederá directamente a esta pantalla.

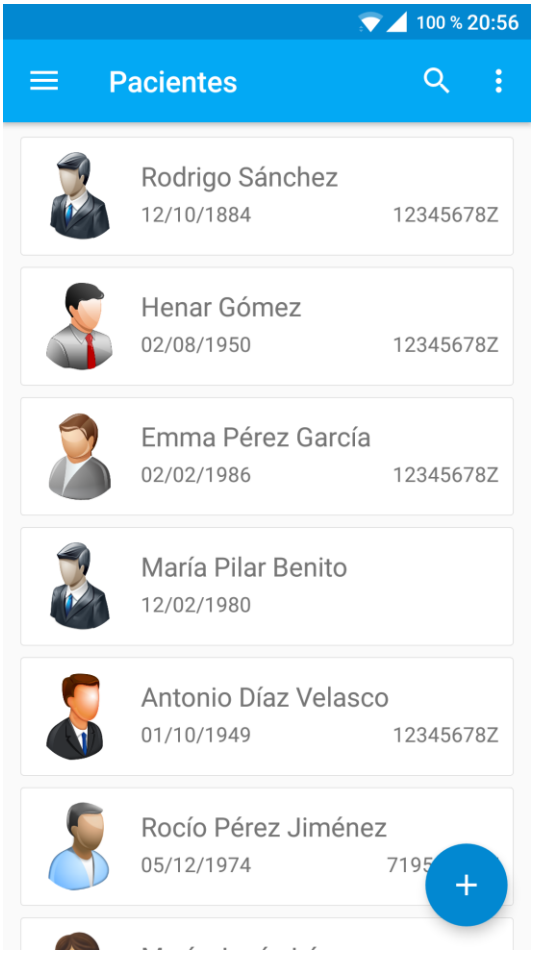

Ilustración 24. Diseño final de la pantalla de listar pacientes

Cuando accedemos a esta actividad, el método *onResume* del ciclo de vida de las actividades Android lanza una petición al servidor usando Retrofit para recibir la lista de pacientes que hay en ese momento en el servidor. Esta petición se hace en el *onResume* para conseguir que al volver a ella cuando esté en segundo plano recojamos cualquier actualización que se pueda haber producido, lo que no podríamos conseguir en el método onCreate.

Las actividades en Android tienen un ciclo de vida que pasa por varios estados desde su creación hasta su destrucción [42]:

- **onCreate()**: Este método es obligatorio, es llamado por el sistema para crear la actividad. Es el lugar donde se deben inicializar los componentes y la interfaz de usuario. Este método tiene como parámetro un objeto *Bundle* que

contiene el estado previo de la actividad. A continuación se llama al método *onStart()*.

- **onStart()**: Es llamado justo antes de que la actividad se haga visible para el usuario. Justo después se llama al método *onResume().*
- **onRestart()**: Es llamado después de que la actividad haya sido parada y justo antes de que vuelva a ser arrancada. Es seguido siempre de la invocación del método *onStart()*.
- **onResume()**: Es llamado justo antes de que la actividad comience a interactuar con el usuario. En este punto la actividad está en la cima de la pila de actividades y recibe la entrada del usuario. A continuación de él, el sistema sólo puede llamar al método *onPause()* del ciclo de vida.
- **onPause()**: Es llamado cuando el usuario va a dejar la actividad, tanto si es destruida como si va a pasar a segundo plano. Debe interrumpir todas las tareas de la actividad que consumen CPU y liberar recursos que no se vayan a usar mientras la actividad no se reanude.
- **onStop()**: Es llamado cuando la aplicación ya no está visible para el usuario. Puede ir seguido de una llamada al método *onRestart* (si la actividad vuelve a interactuar con el usuario) o a *onDestroy* (si la actividad va a ser destruida).
- **onDestroy()**: Esta es la última llamada que recibirá la actividad antes de ser destruida. Se puede haber llamado desde el método finish() o desde el sistema para recuperar recursos [43].

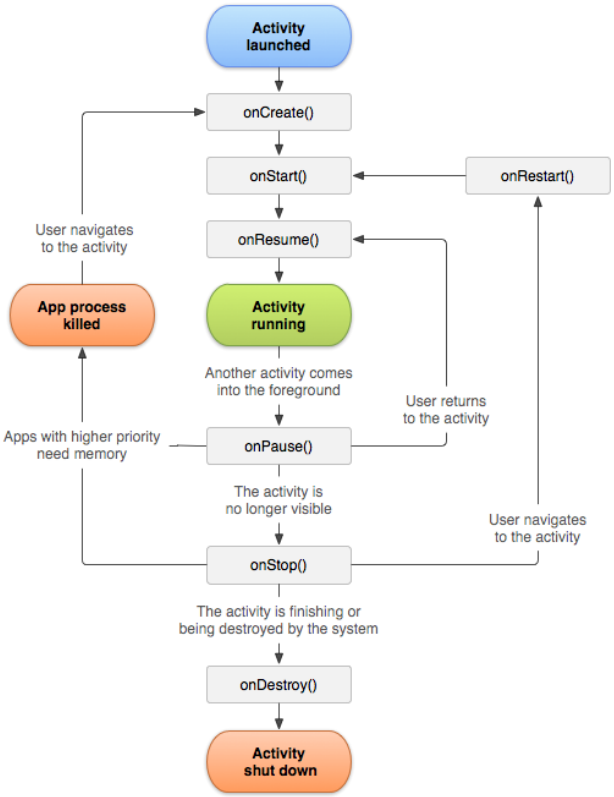

Ilustración 25. Ciclo de vida de una actividad Android

Cuando pulsamos el botón de añadir un paciente, accedemos a la actividad de crear un nuevo paciente. Esta nueva pantalla tendrá un formulario estándar el cual rellenará el usuario médico. En este formulario habrá sólo 6 campos (nombre, apellidos, DNI, identificador clínico, fecha de nacimiento y palabra secreta), de estos, el DNI y el identificador clínico no serán obligatorios. Los usuarios identificarán los parámetros opcionales puesto que no tendrán un asterisco en el título. Una vez creado el paciente, se podrá modificar los datos en cualquier momento.

Al pulsar el botón de enviar, se realizarán unas comprobaciones en la aplicación para asegurar que los datos obligatorios estén rellenos, es decir, que tengan una longitud mayor que cero. Además, gracias a una clase auxiliar que hemos creado a la que hemos llamado "Utilities" comprobaremos la validez del DNI y del formato de la fecha de nacimiento introducida. Para el número de DNI, comprobaremos que tiene una longitud de 9 caracteres, los 8 primeros números y el último una letra (ya sea mayúscula o minúscula). Nos quedamos con la parte numérica y le hacemos el módulo de 23, es decir, nos quedamos el resto resultante de dividir el número entre 23. La cifra obtenida está relacionada con una letra, y esta debe ser la del último carácter del DNI.

Utilizaremos Retrofit para enviar los datos ya comprobados al servidor para su almacenamiento. Además de los datos rellenados en el formulario, también mandaremos el identificador de usuario del médico que crea al paciente. Con este identificador asignaremos el paciente al médico.

En el lado del servidor también se realizarán comprobaciones para asegurar que los campos obligatorios están presentes. Se harán dos inserciones en la base de datos, una para añadir al paciente en la tabla "Paciente" y otra en la tabla "Usuario_Paciente" para la relación paciente-médico:

```
INSERT INTO `Paciente` VALUES (null, '".$nombre."', '".$apellidos."',
'".$dni."',  '".$fNacimiento."',  null,  null,  null,  null,  null,
'".$fechaActual."', '".$palabraSecreta."', '0', '0', '".$idClinico."',
'0', '1', '0', '0', '0', '".$fechaActual."')

INSERT    INTO    `Usuario_Paciente`    VALUES    ('".$idUsuario."',
'".mysqli_insert_id($con)."')
```

Los datos que requiere la tabla "Paciente" son: Identificador de paciente, nombre, apellidos, DNI, fecha de nacimiento, índice de masa corporal (actualmente no utilizado), historial clínico (actualmente no utilizado), medicación (actualmente no utilizado), huella (actualmente no utilizado), fotografía (actualmente elegida aleatoriamente en el servidor), fecha de la última sesión, palabra secreta, si está archivado (0=no, 1=sí), si está eliminado (0=no, 1=sí), identificador clínico, puntuación total, nivel del paciente, sesiones correctas completadas, estado del nivel, si está activado (0=no, 1=sí) y fecha de registro.

En la tabla de "Usuario_Paciente", hay que introducir el identificador de usuario, el cual recibimos por POST en la consulta, y el identificador del paciente, el cual obtenemos con la función PHP "mysqli_insert_id" que nos proporciona el ID auto-

incremental de la consulta anterior. A esta función hay que ponerle como parámetro la variable de sesión "$con". Gracias a esta forma de obtener el identificador, nos ahorramos una consulta a la base de datos para obtener el ID del paciente, lo que mejora el rendimiento y la escalabilidad.

Si todo ha ido como se esperaba y el paciente ha sido insertado correctamente se contesta a la aplicación con un estado "OK" y una frase "Paciente añadido correctamente.". En caso de error se enviará el estado "KO" y se explicará el motivo del error.

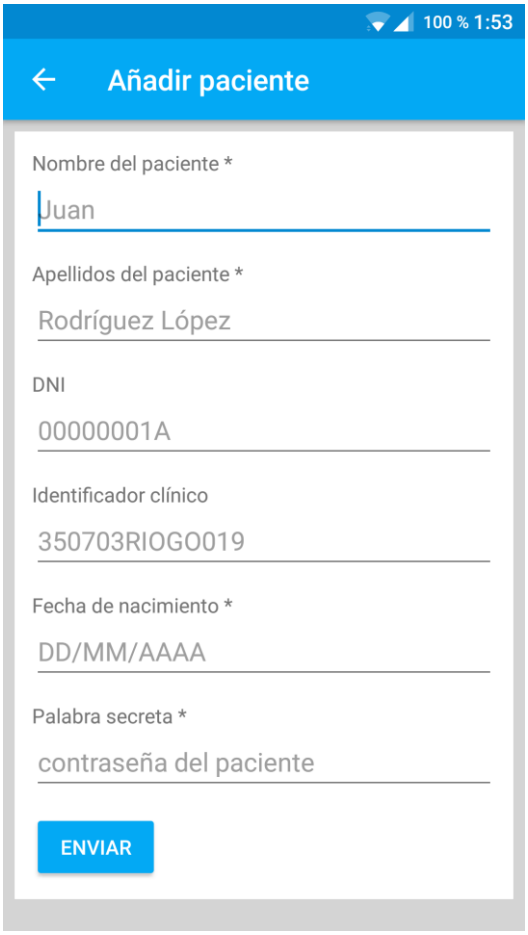

Ilustración 26. Diseño final de la pantalla de añadir un paciente

Al pulsar el botón de enviar que podemos ver en la Ilustración 1, si todo va bien se creará el paciente y se regresará a la pantalla de la lista de los pacientes, la cual se recargará (puesto que la llamada al servidor estaba en el método onResume y la actividad ha pasado de segundo plano a primer plano, por tanto se ejecuta).

Como hemos visto anteriormente, en cada elemento de la lista de pacientes se pueden realizar varias acciones cuando se mantiene pulsado uno de ellos. Para la acción de "Ver paciente" accederemos a una nueva actividad donde veremos los datos y acciones que se pueden realizar sobre él, en "Editar paciente" accederemos a una actividad similar a la de añadir a un paciente en la que los campos están rellenos con los valores actuales del paciente, en "Archivar paciente" no se accede a ninguna actividad al igual que para el caso de "Activar paciente" y "Desactivar paciente". Para estas

acciones se enviará una petición al servidor y la contestación se recibirá en esta misma actividad.

Gracias a Retrofit y al método “accionPaciente” podremos enviar al servidor acciones sobre el paciente tales como borrarlo, archivarlo, recuperarlo, activarlo o desactivarlo. Para ello enviaremos al servidor una petición con el nombre de la acción a realizar y el identificador del paciente. En la parte del servidor se hará la modificación de los datos del paciente de acuerdo a la acción seleccionada. Esto implica que sólo se hará una consulta a la base de datos por cada acción.

```
if($accion == "Archivar"){
    $query = "UPDATE Paciente SET archivado=1 WHERE
              idPaciente='".$idPaciente."'";
}else if($accion == "Recuperar"){
    $query = "UPDATE Paciente SET archivado=0 WHERE
              idPaciente='".$idPaciente."'";
}else if($accion == "Borrar"){
    $query = "UPDATE Paciente SET eliminado=1 WHERE
              idPaciente='".$idPaciente."'";
}else if($accion == "Activar"){
    $query = "UPDATE Paciente SET activado=1 WHERE
              idPaciente='".$idPaciente."'";
}else if($accion == "Desactivar"){
    $query = "UPDATE Paciente SET activado=0 WHERE
              idPaciente='".$idPaciente."'";
}
```

Si todo ha ido correctamente, la aplicación recibirá una contestación en formato JSON con un estado OK y una frase confirmando el éxito en la operación. En caso contrario se recibirá un estado KO junto a la explicación del problema.

Como antes hemos mencionado, para acceder a los datos de un paciente desde la pantalla de la lista de pacientes se puede hacer una pulsación larga y seleccionar la opción “Ver paciente” o simplemente hacer una pulsación simple. Desde la pantalla se lanza una nueva actividad al que se envía los datos del paciente seleccionado.

La pantalla de información del paciente es quizás la más importante de la aplicación, desde ella se podrá modificar casi todo lo relacionado con ese paciente y añadir y ver sus sesiones. Por ello debe ser diseñada con cuidado y evitar que se vea la información demasiado junta y pueda desconcertar al usuario.

Ilustración 27. Diseño de la pantalla de información del paciente

Tal y como vemos en la Ilustración 27, esta pantalla contará con una barra superior, pero en este caso no tendrá un menú desplegable, sino que tendrá un botón de retroceso para volver a la lista de pacientes. También habrá en la misma barra de herramientas, pero al lado opuesto 3 botones de acción: editar paciente, archivar paciente y activar o desactivar paciente (dependiendo del estado en el que esté).

Separaremos por secciones las partes principales de la interfaz:

- Datos de usuario: Donde aparecerán los datos del paciente.
- Datos de terapia: Donde aparecerá el progreso del paciente en la terapia.
- Datos de sesiones: Donde aparecerán las sesiones del paciente, tanto las finalizadas como las que aún no se han realizado.

Por último, añadiremos un botón flotante con el símbolo “+” para añadir una nueva sesión al paciente. Este botón ya aparece en la versión web de Telekin y con esta apariencia conseguiremos que un usuario que está acostumbrado a utilizar esa plataforma pueda rápidamente repetir esta acción en su dispositivo móvil.

Ilustración 28. Diseño final de la pantalla de información del paciente

En los botones de acción superiores que vemos en la Ilustración 28 arriba a la derecha, tenemos asociadas las acciones antes mencionadas. En el icono del lápiz se accede a la actividad de edición de usuario tal y como se podía hacer con una pulsación larga en uno de los elementos de la lista de pacientes. En el icono del archivador se envía al servidor la acción de archivar el paciente, esta acción pedirá una confirmación en una ventana emergente. Al ser una operación algo sensible, al requerir una confirmación evitaremos que se pueda archivar un paciente por una pulsación accidental. Por último, el botón de activar y desactivar al paciente enviará también la acción al servidor de la misma manera que vimos páginas atrás, pero esta vez no necesitará confirmación debido a que esta operación no tiene grandes repercusiones. Sin embargo, le asociaremos una pequeña vibración para saber claramente cuando se ha pulsado.

```
Vibrator v = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
v.vibrate(100);
```

Además debemos añadir un permiso especial al Android Manifest:

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

Aunque cuando se pulsa la campana cambia el icono para indicar si está activado o desactivado el paciente, es la única acción de la pantalla que no abre una nueva pantalla o actividad.

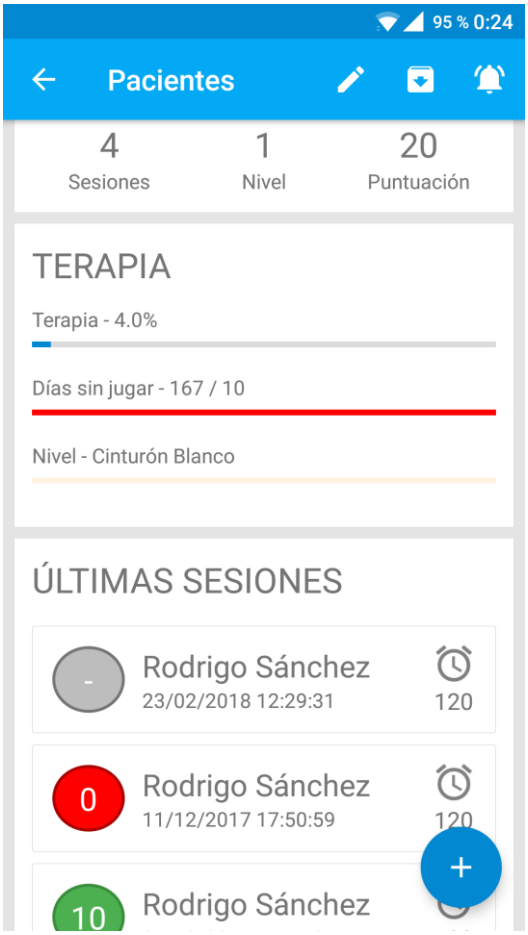

Ilustración 29. Diseño final de la pantalla de información del paciente (2)

En la sección referente a la terapia de nuestra pantalla de paciente, tenemos los datos principales en forma de barras de progreso. Primero podemos ver el progreso general de la terapia junto a su porcentaje en número. En segundo lugar, aparece los días que lleva el paciente sin jugar, siempre que el número de días sea menor que el límite establecido para el paciente aparecerá la barra de progreso en verde. Si el usuario ha superado los días permitidos, la barra aparecerá totalmente llena y en color rojo. Por último, aparecerá el nivel del paciente en forma de cinturón. Esta es la forma que se ha elegido en la plataforma para que el progreso sea más divertido y menos lineal. En nuestro caso, los cinturones se relacionan con los niveles de la siguiente manera:

- Nivel 1 = Cinturón Blanco
- Nivel 2 = Cinturón Amarillo
- Nivel 3 = Cinturón Verde
- Nivel 4 = Cinturón Azul
- Nivel 5 = Cinturón Negro

El objetivo es llegar al máximo nivel, que es el cinturón negro. Una vez alcanzado con la supervisión del médico, el paciente habrá superado la rehabilitación.

La última sección de la pantalla de la información del paciente es la referente a ver las sesiones. Por defecto se cargan las 5 últimas sesiones, si hay más, se podrán ver pulsando un botón en la parte inferior que cargará otras 5. Las sesiones irán cargando de 5 en 5 hasta que no haya más y el botón quede deshabilitado.

Las sesiones se mostrarán con el círculo en gris y con un guion en lugar de la nota si aún no ha sido realizada. Aparecerá en rojo si no ha sido superada, en amarillo si está próximo al suspenso y en verde si se ha superado ampliamente. Las sesiones deshabilitadas se mostrarán en un tono más blanco que las demás para indicar que no se tienen en cuenta.

Al pulsar cualquiera de ellas se irá a la pantalla que muestra los detalles y los resultados (en caso de haberlos) de la misma.

El botón flotante con el signo "+" lleva a la sección más sofisticada y una de las partes más importantes de la aplicación. La actividad que se crea permitirá al usuario crear una nueva sesión para el paciente seleccionado. Esta parte de la herramienta debe ser capaz de listar 2 tipos de actividades que se descargarán del servidor, y a las de uno de los tipos, tiene que permitir agregar ciertos valores. Una vez elegidas las actividades de la sesión, el usuario debe poder modificar el orden de estas, ya que será el orden en el que se le presenten al paciente a la hora de realizar sus ejercicios.

A la hora de diseñar esta parte de la aplicación hemos tenido en cuenta la premisa que impusimos en un principio de no realizar una navegación muy profunda, es decir, que haya que pasar por una gran cantidad de pantallas para realizar una acción. La idea principal es crear una actividad con dos pestañas o *tabs* donde se puedan seleccionar las actividades de la sesión, y una segunda actividad donde meter los datos y ordenar las actividades. Sin embargo, Android presenta una limitación a la hora de acceder a datos introducidos por un usuario en una lista de un recyclerView. Cuando ponemos algún *EditText* en los ítems de un *RecyclerView*, al recorrer la lista para obtener los valores que introdujo el usuario solo podemos acceder a los que se están mostrando en ese momento en la pantalla.

Por tanto, resulta imposible hacerlo en 2 pantallas, habrá que separar las operaciones y crear la sesión en tres pasos.

Ilustración 30. Diseño de las pantallas de creación de una sesión

La interfaz quedaría tal y como aparece en la Ilustración 30, donde en todas las pantallas podemos ver como hay una barra de herramientas superior donde aparece el título de la sección y una flecha para volver a la pantalla anterior.

En la primera actividad del proceso de creación de una sesión, hemos creado una actividad con dos pestañas. Al acceder a la pantalla se enviará una petición al servidor vía *Retrofit* con las articulaciones a rehabilitar relacionadas con el paciente, y este contestará con la lista de actividades asociadas a dichas articulaciones. Las enviará en formato JSON divididas en dos arrays, el de las actividades correspondientes al goniómetro y el de las otras actividades.

En la parte del servidor tenemos un script php que hace dos consultas, una para cada tipo de actividad. Para distinguir qué actividades son de cada tipo habrá que ver si contiene preferencias, puesto que las actividades del goniómetro no son configurables y ese campo de la base de datos está vacío. Como no hay un campo que especifique de que tipo es la actividad, las consultas serán:

```
$query = "SELECT * FROM Preferencia pref
   INNER JOIN Preferencia_Actividad pact ON
   (pact.Preferencia_idPreferencia = pref.idPreferencia)
   INNER JOIN Actividad act ON
   (act.idActividad = pact.Actividad_idActividad)
   WHERE ";
$articulacionesExploded = explode(",", $articulaciones);
for($i=0;$i<count($articulacionesExploded);$i++){
   $query = $query."act.articulacion LIKE
            '%".$articulacionesExploded[$i]."%'";
      if($i<(count($articulacionesExploded)-1)){
         $query = $query." OR ";
      }
}
```

```
$query = $query."ORDER BY idActividad ASC";


$query2 = "SELECT * FROM Actividad WHERE (";
for($i=0;$i<count($articulacionesExploded);$i++){
   $query2 = $query2."articulacion LIKE
             '%".$articulacionesExploded[$i]."%'";
   if($i<(count($articulacionesExploded)-1)){
      $query2 = $query2." OR ";
   }
}
if(count($idsActividades)>0){
   $query2 = $query2.") AND (";
}
for($i=0;$i<count($idsActividades);$i++){
   $query2 = $query2."idActividad != ".$idsActividades[$i]."";
   if($i<(count($idsActividades)-1)){
      $query2 = $query2." AND ";
   }
}
$query2 = $query2.")";
```

Una vez recibida la información de la base de datos en nuestra aplicación, crearemos un *TabLayout* y un *ViewPager*. El *TabLayout* es la parte de la interfaz que gestiona las pestañas, mientras que el *ViewPager* gestiona en contenido de las mismas. Ambas están estrechamente relacionadas y dependen la una de la otra. Para su creación, el *ViewPager* requiere de un adaptador y el *TabLayout* requiere de un *ViewPager* como parámetro.

En cada una de las pestañas que tenemos crearemos una lista, e introduciremos en cada una la lista correspondiente a su categoría. Además, incluiremos un *OnClickListener* en cada elemento para que al pulsarlo se añada a una lista nueva de actividades seleccionadas y se cambie el fondo al color verde para indicar visualmente al usuario que ha sido marcado ese ítem. En caso de estar marcado el elemento y hacer una pulsación en él, este volverá al color habitual y se eliminará de la lista de actividades seleccionadas.

El último elemento de la primera actividad del proceso de creación de una sesión es un botón flotante con una flecha que servirá para avanzar. Al pulsarlo se ejecutará un *intent* que iniciará la siguiente actividad enviándole como parámetros la lista de actividades seleccionadas, el paciente seleccionado y la terapia del paciente. Todos estos datos serán necesarios para enviárselos al servidor cuando creemos la sesión.

Ilustración 31. Diseño final de la primera pantalla de creación de una sesión

La segunda pantalla del proceso será una lista en la que podremos cambiar el orden de los elementos. Para poder realizar estas acciones importaremos una nueva librería llamada "AdvancedRecyclerView". Esta herramienta permite cambiar el orden de los elementos y también deslizarlos hacia la derecha o la izquierda (swipe), todo esto de una forma muy sencilla. Esta librería está disponible a partir de la API 14 (Android 4.0 – IceCreamSandwich) y se añade al *Gradle* como las demás librerías [44].

Como al mirar la pantalla el usuario solo verá una lista, hay que indicarle de alguna forma que en esta lista puede cambiar el orden de los elementos y como tiene que hacerlo. Para ello vamos a utilizar un Snackbar, que es una barra de texto que aparece en la parte inferior de la pantalla y desaparece pasado un tiempo establecido. Esta Snackbar aparecerá al crearse la actividad e indicará al usuario que debe mantener presionado un ítem y después arrastrarlo para cambiarlo de posición.

Cuando el usuario haya colocado las actividades en el orden deseado, deberá pulsar el botón flotante de la esquina inferior derecha que lanzará la actividad de la última pantalla del proceso de creación de una sesión. A esta última pantalla se le enviará nuevamente los datos del paciente seleccionado, su terapia y la lista de actividades, esta vez, ordenada.

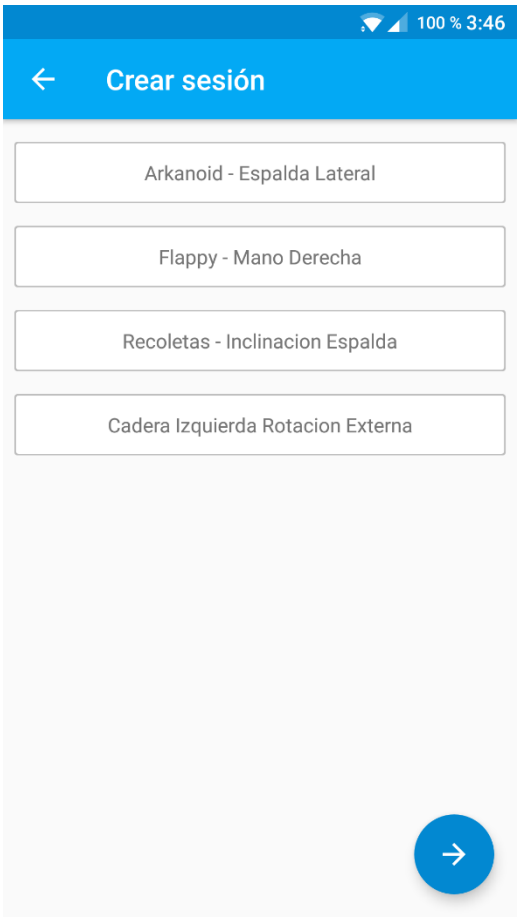

Ilustración 32. Diseño final de la segunda pantalla de creación de una sesión

Con la lista de objetos "Actividad" que recibimos en la actividad, buscaremos las que tengan preferencias, y crearemos tantos campos para introducir datos como sea preciso. Las actividades de tipo goniómetro, al no tener preferencias, tan solo se mostrarán al usuario con el título para que sepa que están incluidas y la posición que ocupan. En las actividades con preferencias, se mostrarán campos de introducción de números (*EditText* de tipo "*Number*") excepto cuando la preferencia tenga una serie de opciones seleccionables, en cuyo caso se mostrarán en un *Spinner* o lista desplegable.

Puesto que como hemos dicho anteriormente, no podemos introducir los elementos en una lista de un RecyclerView o ListView para mostrarlos, los mostraremos en un simple LinearLayout vertical en el que añadiremos los elementos por código.

En la pantalla también aparecerá un interruptor o *switch* el cual sirve para activar o desactivar en la sesión el paso automático de las escenas. Esto es una característica que necesita la plataforma y debe ser enviada junto con los datos de las actividades.

También podemos ver otro botón flotante tal y como aparecía en las pantallas anteriores, pero esta vez, la flecha que hay dibujada es diferente. Esta nueva flecha tiene la forma de un avión de papel y da la sensación al usuario de que no va a avanzar a otra pantalla, sino que este botón enviará la sesión. Al pulsarlo se realizará una comprobación para asegurarse de que los campos de cada una de las actividades no están vacíos, en caso de no ser así aparecerá un cuadro de error.

Si los campos han sido correctamente rellenados, aparecerá un cuadro emergente para añadir, si el usuario lo desea, un comentario. Esta funcionalidad es totalmente opcional y se podrá dejar vacío y pulsar el botón “Enviar” para que finalmente se envíe la sesión al servidor.

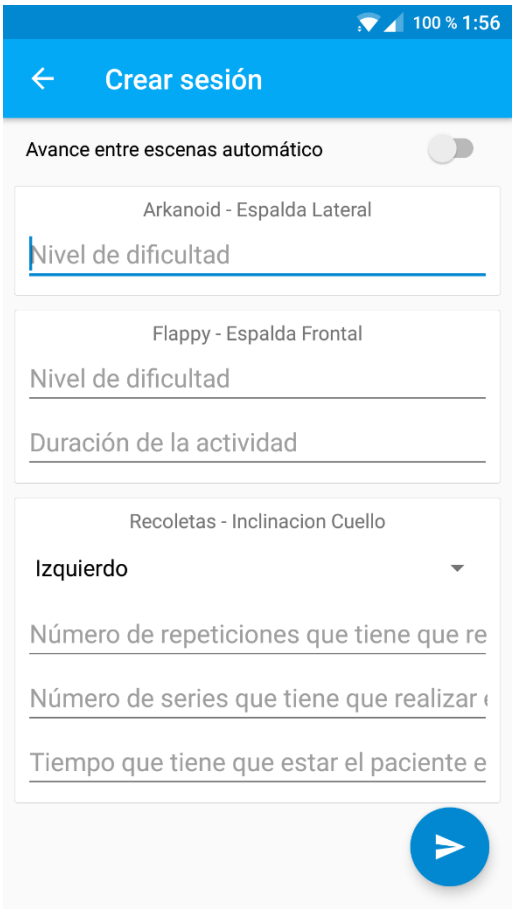

Ilustración 33. Diseño final de la última pantalla de creación de una sesión

En la sesión enviada a la base de datos se incluirá:

- El identificador de usuario: Identificador único de cada usuario que ha sido almacenado en las preferencias de la aplicación.
- El identificador de paciente: Identificador único de cada paciente que se obtiene de la lista de pacientes.
- El modo automático: Un *boolean* que toma el valor true si el modo automático está activado y el valor false en caso contrario.
- El comentario: Un *String* con el comentario introducido por el usuario. Se envía tanto si se ha rellenado como si se ha dejado en blanco, y este se introducirá en la base de datos.
- Las actividades: Un *String* que contiene la información de las actividades de la sesión, las preferencias y sus valores (con sus identificadores es suficiente). Los datos estarán separados por delimitadores. Las actividades se separarán por el carácter “|”, los pares clave-valor de las

preferen cias serán separados por el carácter “/”, y las preferencias con su valor se separarán con el carácter “=”. En el servidor, esta información será decodificada de la misma forma.

Para registrar correctamente la sesión en la base de datos se deberán realizar 3 inserciones a la base de datos más una consulta adicional para obtener el valor de la tabla “Usuario_Preferencia” para una preferencia asociada al usuario que realizó la petición:

```
$query = "INSERT INTO `Sesion` VALUES (null, '" .$idPaciente."',
'".date("Y-m-d H:i:s")."', '".$modoAutomatico."', 120, 0, 0, null, 0,
'".$comentario."')";

$queryDuracion = "SELECT valor FROM Usuario_Preferencia WHERE
Usuario_idUsuario='".$idUsuario."' AND Preferencia_idPreferencia=2";

$query2 = "INSERT INTO `Actividad_Sesion` VALUES ";
$actividadesExploded = explode("|", $actividades);
for($i=0;$i<count($actividadesExploded);$i++){
   $actividadExploded = explode("/", $actividadesExploded[$i]);
   $query2 = $query2."('".$actividadExploded[0]."', '".$idSesion."',
      '".$duracion."', '".$i."')";
   if($i<(count($actividadesExploded)-1)){
      $query2 = $query2.",";
   }
}

$query3 = "INSERT INTO `Preferencia_Actividad_Sesion` VALUES ";
$contadorGoniometro = 0;
for($i=0;$i<count($actividadesExploded);$i++){
   $actividadExploded = explode("/", $actividadesExploded[$i]);
   if(count($actividadExploded)>1){
      $idPreferencias = explode(",", $actividadExploded[1]);
      for($j=0;$j<count($idPreferencias);$j++){
         $prefContent = explode("=", $idPreferencias[$j]);
         $query3 = $query3."('".$actividadExploded[0]."',
            '".$idSesion."','".$prefContent[0]."',
            '".$prefContent[1]."'),";
      }
   }else{
      $contadorGoniometro++;
   }
}
$query3 = substr($query3, 0, -1);
```

El servidor responderá a la aplicación con un estado “OK” y un mensaje de éxito si no se produce ningún error al crear la nueva sesión. Si hay algún problema en el registro, el servidor contestará con un estado “KO” y un mensaje de error indicando el fallo que se ha producido.

En la parte de la aplicación, al recibir la contestación con el estado “OK”, se destruirán las actividades de creación de la sesión que estaban en segundo plano y

volveremos a la actividad de visualización de los detalles del paciente. Además, se mostrará un cuadro indicando que la sesión fue correctamente registrada. Si se produjo algún error en la creación de la sesión, aparecerá una ventana informativa y no se destruirá ninguna actividad, puesto que puede ser un problema puntual (por ejemplo, no disponer de conexión a internet en ese momento) y el usuario puede reintentar la operación.

Actividad de administración de terapias

Al pulsar el botón de "Terapias" del menú desplegable lateral accederemos a una actividad que lista los pacientes asociados al usuario. Esta lista de usuarios se obtendrá de forma local de la que ya descargó en actividades anteriores, puesto que se asume que no cambiará el número de pacientes mientras dura la sesión, puesto que es el propio médico el que tiene abierta la sesión en el dispositivo móvil. En el caso de querer actualizar la lista se podrá hacer accediendo a la sección de pacientes. De esta forma ahorraremos conexiones a la base de datos y la aplicación será más fluida para el usuario.

En la parte superior de la actividad aparecerá un icono de una lupa que al pulsarlo se desplegará una barra de búsqueda, donde podremos buscar a los pacientes por su nombre, apellidos, DNI o identificador clínico, al igual que en la actividad de la sección anterior. Al contrario que la actividad de visualización de pacientes, en esta pantalla no se podrán realizar acciones sobre los pacientes tales como borrarlos, editarlos o modificar sus datos personales, y tampoco permitirá añadir pacientes desde esta lista. Tan sólo permite una pulsación simple sobre cada elemento de la lista, lo cual nos llevará a la actividad de visualización de la terapia del paciente seleccionado.

Al pulsar el elemento deseado de la lista, se creará una nueva actividad a la que se pasará como parámetro el objeto del tipo Paciente. La interfaz será muy sencilla y similar a la de visualización de datos de pacientes, de esta forma se consigue uniformidad y sobriedad para que el usuario asocie este tipo de pantalla a la obtención de información. En caso de que el usuario no tenga asignada una terapia, aparecerá una alerta y obligará al usuario a crearle una terapia al paciente.

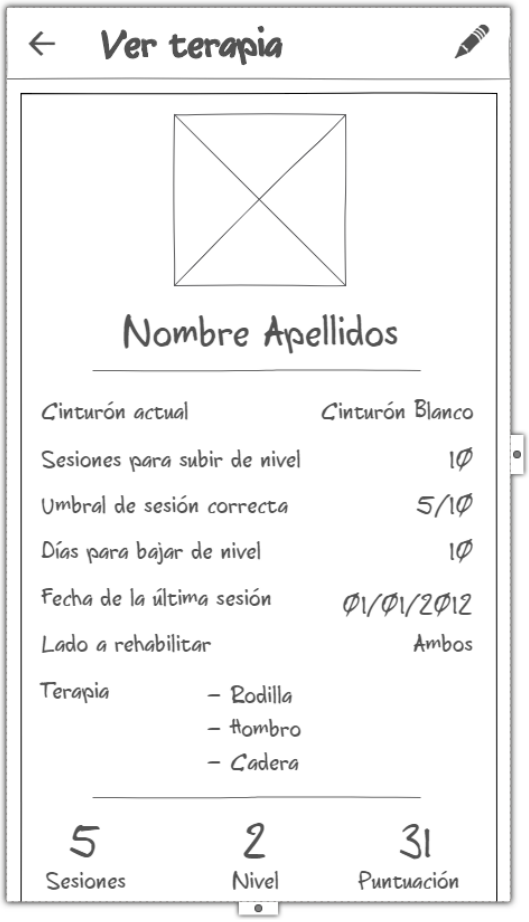

Ilustración 34. Diseño de la pantalla de información de terapia

En esta nueva actividad y con ayuda de Retrofit enviaremos una petición a nuestro servidor con el identificador del paciente seleccionado. En el servidor se realizará una sola consulta a la base de datos para sacar toda la información relativa a la terapia del paciente:

```
$query = "SELECT * FROM Terapia t
    INNER JOIN Terapia_Articulacion ta ON (t.idTerapia =
        ta.Terapia_idTerapia)
    INNER JOIN Articulacion a ON (ta.Articulacion_idArticulacion =
        a.idArticulacion)
    WHERE t.Paciente_idPaciente='$idPaciente'";
```

Al recibir la terapia en la aplicación, se mostrará la información ordenada tal y como describimos en la Ilustración 34. Para hacer más descriptiva la información ofrecida, añadiremos un icono representativo al lado izquierdo de los títulos de cada uno de los atributos de la terapia mostrada, de forma similar a como lo hicimos en la pantalla de información de paciente. Para mostrar estas imágenes usaremos de la misma manera la librería Iconify y como fuente de los iconos FontAwesome.

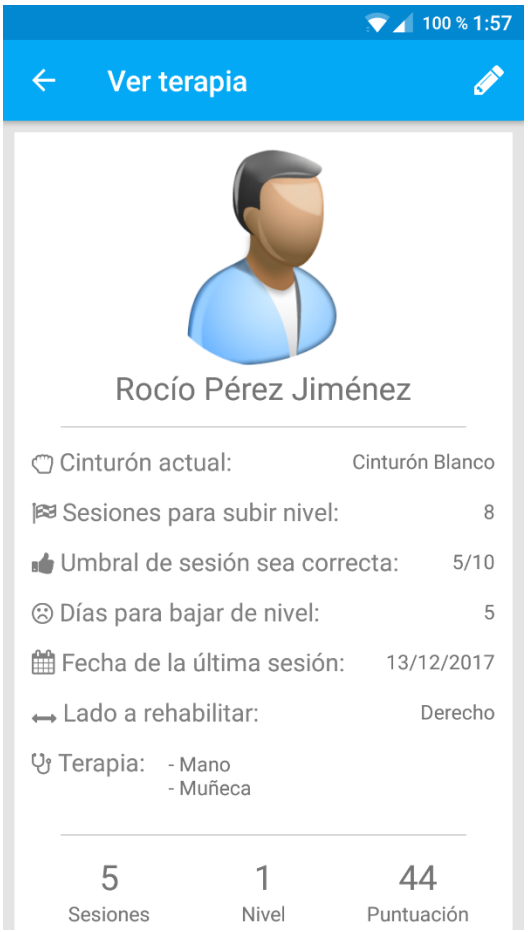

Ilustración 35. Diseño final de la pantalla de información de terapia

En la barra de herramientas superior que hay en la Ilustración 35 podemos ver el símbolo de un lápiz que indica que al pulsar sobre este botón accederemos a una pantalla en la que podremos editar la terapia de este paciente. El *intent* que crea esta nueva actividad le enviará como atributos el objeto del paciente seleccionado y su objeto terapia. Cuando el paciente no tiene una terapia asignada, directamente accederá a esta pantalla para crearla.

Esta nueva pantalla que permite la modificación de los parámetros de la terapia del paciente la haremos de tal forma que no consista tan solo en rellenar campos de texto, de esta manera será más dinámica y conseguiremos mayor atención del usuario.

Los atributos que se podrán modificar en la terapia de los pacientes son:

- El número de sesiones que el paciente debe realizar correctamente para poder subir de nivel y con ello cambiar de cinturón. Esto será un campo de texto sólo numérico, puesto que no hay un máximo de sesiones permitidas.
- El umbral de la sesión para que la sesión sea considerada como correcta. Este parámetro lo configurará el usuario con una *Seekbar*, que es una barra de progreso configurable que tiene como valor mínimo 0 y como valor máximo 10.

- Días que el paciente puede estar sin realizar alguna sesión antes de que el sistema le baje de nivel. Esto será un campo de texto sólo numérico, puesto que no hay un máximo de días permitidos.
- Lado que el paciente tiene que rehabilitar en las articulaciones seleccionadas. Este atributo se seleccionará con *RadioButtons*, que son opciones seleccionables excluyentes, es decir, solo se puede escoger una de todas las disponibles.
- Las articulaciones que el paciente tiene que rehabilitar. Este parámetro lo captaremos mediante *CheckBoxes*, que son opciones seleccionables no excluyentes, por lo que el usuario podrá seleccionar las que considere oportuno.

En las terapias habrá articulaciones que requerirán un que haya un lado especificado y otras que no, sin embargo, esta opción será obligatoria, y será necesario que al menos una articulación sea seleccionada, puesto que si el paciente se encuentra en el sistema es porque requiere de rehabilitación.

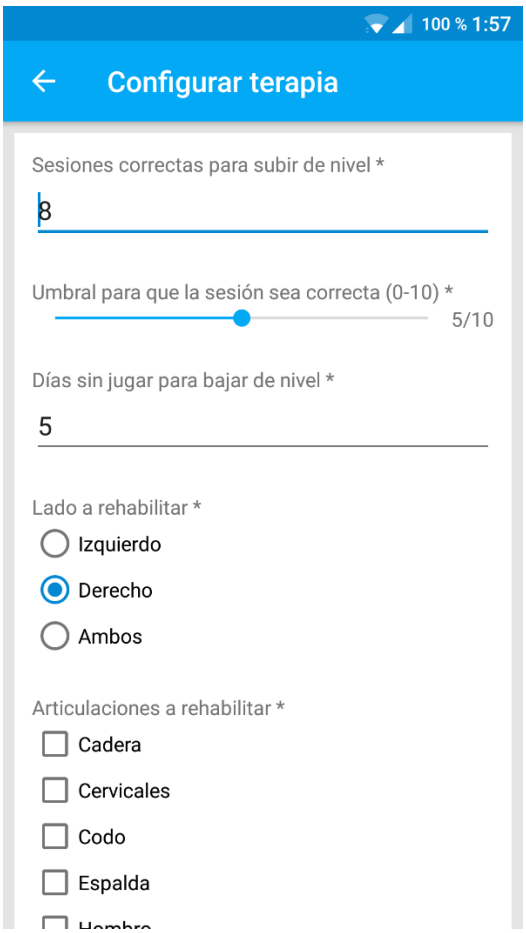

Ilustración 36. Diseño final de la pantalla de edición de terapia

Al final de la pantalla de edición de terapia hay un botón azul con el texto "Enviar" que mediante *Retrofit* manda al servidor la información modificada de la terapia. El fichero que se encarga de la recepción de estos datos en el lado del servidor lo

utilizaremos para añadir una terapia nueva o para editar una existente. Se enviarán como parámetros de esta petición:

- El identificador del paciente.
- El identificador de la terapia.
- El número de días sin jugar antes de bajar de nivel.
- El número de sesiones correctas para subir de nivel.
- El umbral para que la sesión sea correcta.
- El lado de la articulación o articulaciones a rehabilitar.
- El identificador de las articulaciones separadas por comas.

Para realizar la correcta modificación de la terapia necesitaremos hacer 3 consultas a la base de datos, la primera podrá ser de inserción de la terapia (si la terapia es nueva) o de modificación de ella (si se ha editado):

```
if($idTerapia==0){
   $query = "INSERT INTO `Terapia` VALUES (null, '".$diasSinJugar."',
   '".$sesionesCorrectas."','".$umbralCorrecta."','".$idPaciente."',
   '".$ladoLesion."')";
}else{
   $query = "UPDATE Terapia SET diasSinJugar='".$diasSinJugar."',
   sesionesCorrectas='".$sesionesCorrectas."', umbralCorrecta =
   '".$umbralCorrecta."', ladoLesion = '".$ladoLesion."' WHERE
   idTerapia = '".$idTerapia."'";
}
```

Las otras dos consultas a la base de datos nos permitirán borrar e insertar de nuevo la relación de un paciente con su terapia y las articulaciones de la misma en la tabla denominada “Terapia_Articulacion”.

```
if($idTerapia==0){
  $idTerapia = mysqli_insert_id($con);
}
$query2 = "DELETE FROM Terapia_Articulacion WHERE
  Terapia_Paciente_idPaciente='".$idPaciente."'";

$query3 = "INSERT INTO `Terapia_Articulacion` VALUES ";
$articulacionesExploded = explode(",", $articulacionesId);
for($i=0;$i<count($articulacionesExploded);$i++){
  $query3 = $query3."('".$idTerapia."', '".$idPaciente."',
    '".$articulacionesExploded[$i]."')";
  if($i<(count($articulacionesExploded)-1)){
    $query3 = $query3.",";
  }
}
```

Si la inserción de la terapia en las tablas de la base de datos se realiza de manera exitosa, se devolverá a la aplicación una respuesta con el estado “OK” y la terapia que se acaba de insertar, en caso de que haya habido algún problema se enviará un estado “KO” y una breve descripción del problema.

Cuando la respuesta llega a la aplicación, si ha ido todo bien, finalizará la actividad y mostrará por pantalla un mensaje diciendo que se ha modificado correctamente la terapia. En caso de error, se mantendrá la actividad y permitirá volver a enviarla por si ha sido un problema como la falta de conexión a internet de forma puntual.

Actividad de visualización de sesiones

La actividad de visualización se sesiones mostrará una lista ordenada por fecha de todas las sesiones realizadas y no realizadas por los pacientes asociados al médico usuario registrado. Para acceder a esta sección de la aplicación basta con que pulsemos el elemento “Sesiones” del menú lateral desplegable de las actividades principales. Esta actividad solo mostrará la lista de sesiones y una barra de herramientas superior en la que aparecerá el botón del menú desplegable y un botón con una lupa para abrir la barra de búsqueda de la lista.

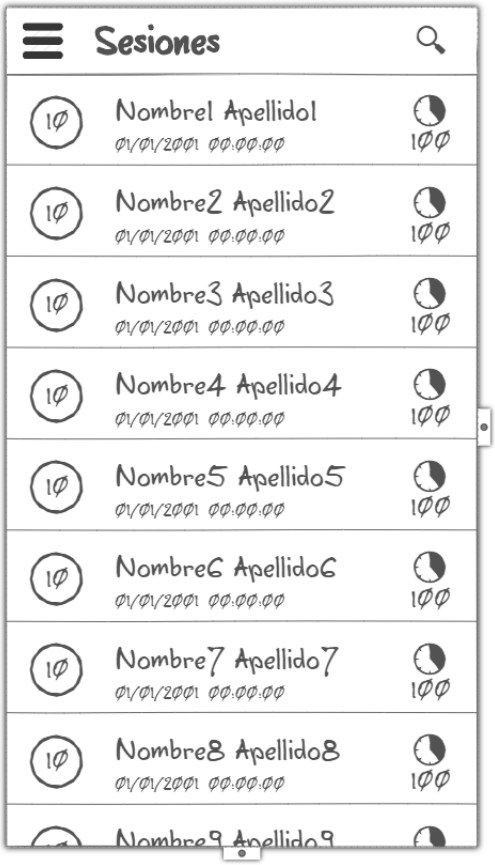

Ilustración 37. Diseño de la pantalla de la lista de sesiones

Cada elemento de la lista muestra el nombre del paciente que ha realizado o tiene que realizar la sesión, su nota (siempre que haya finalizado la misma), la fecha en

la que se realizó la sesión o la fecha en la que se creó en caso de no haber sido realizada, y el tiempo en el que ha sido realizada la sesión.

Cuando se crea esta actividad, se envía una petición al servidor tan solo con el identificador del usuario, con este identificador se obtendrán los pacientes que tiene asociados el médico y con esto sus sesiones:

```
$query = "SELECT idSesion, idPaciente, fechaSesion, fechaRealizacion,
nombre, apellidos, valor, duracionSesion, descartada, comentario FROM
Paciente pac
  INNER JOIN Sesion ses ON (pac.idPaciente = ses.Paciente_idPaciente)
  INNER JOIN Sesion_Resultado sres ON (ses.idSesion =
    sres.Sesion_idSesion)
  WHERE pac.idPaciente IN (
    SELECT paciente_idPaciente FROM Usuario_Paciente upac WHERE
    upac.Usuario_idUsuario='".$idUsuario."')
  ORDER BY fechaRealizacion DESC";

$query2 = "SELECT idSesion, idPaciente, fechaSesion, fechaRealizacion,
nombre, apellidos, duracionSesion, descartada, comentario FROM  Paciente
pac
  INNER JOIN Sesion ses ON (pac.idPaciente = ses.Paciente_idPaciente)
  WHERE pac.idPaciente IN (
    SELECT paciente_idPaciente FROM Usuario_Paciente upac WHERE
    upac.Usuario_idUsuario='".$idUsuario."' AND ses.realizado=0)
  ORDER BY fechaSesion DESC";
```

Se realizarán tan solo dos consultas a la base de datos, una para obtener las sesiones ya realizadas por los pacientes y otra para las sesiones sin hacer. Estos dos grupos de sesiones se enviarán en dos vectores de datos distintos, y será la propia aplicación cliente la que se encargue de hacer una única lista de sesiones. Esto lo hará con una lista fija que irá recorriendo y añadiendo elementos de la otra en función de la fecha de realización o de creación de la sesión.

```
public ArrayList<SesionItem> mezcladorDeSesiones(ArrayList<SesionItem>
sesionItems, ArrayList<SesionItem> sesionSinHacer){
   ArrayList<SesionItem> listaSesiones = new ArrayList<>();
   listaSesiones.addAll(sesionItems);
   int pos=0;
   for (int i=0; i<sesionSinHacer.size();i++){
      for(int j=0;j<listaSesiones.size();j++){
         if(listaSesiones.get(j).getFechaRealizacion() != null){
            if(listaSesiones.get(j).getFechaRealizacion().before(
            sesionSinHacer.get(i).getFechaSesion())){
               listaSesiones.add(j,sesionSinHacer.get(i));
               break;
            }
         }else{
            if(listaSesiones.get(j).getFechaSesion().before(
            sesionSinHacer.get(i).getFechaSesion())){
```

```
                listaSesiones.add(j,sesionSinHacer.get(i));
                break;
            }
        }
    }
    return listaSesiones;
}
```

En la lista de sesiones se mostrarán las sesiones realizadas y superadas; realizadas y no superadas; realizadas y descartadas; y no realizadas. Las que no han sido superadas se mostrarán con el círculo donde figura la nota en rojo, si está cerca de la nota de umbral en amarillo, y si está superada ampliamente en verde. Las sesiones no realizadas tendrán el círculo de color gris y en lugar de una nota, habrá un guion. Por otro lado, las sesiones descartadas aparecerán en un color mucho más claro para que el usuario pueda deducir que esta sesión está deshabilitada simplemente con mirarla.

Ilustración 38. Diseño final de la pantalla de lista de sesiones

Al pulsar cada uno de los elementos de la lista se creará una nueva actividad en la que se mostrará la información completa de la sesión. En ella podremos ver una barra de herramientas al igual que en todas las actividades de la aplicación, con el botón de regreso y un botón adicional para descartar la sesión. En caso de que la sesión esté ya descartada, este botón estará deshabilitado.

La pantalla de dividirá en varias cajas, la primera contendrá información relativa a la sesión (el nombre y apellidos del paciente, la fecha de registro de la sesión, la fecha de realización y el comentario del médico). Además, habrá una caja por cada una de las actividades de la sesión. Las cajas de actividad tendrán el nombre de la actividad que se realizó, y una gráfica con los datos de la sesión o el progreso con respecto a sesiones anteriores (dependiendo del tipo de actividad).

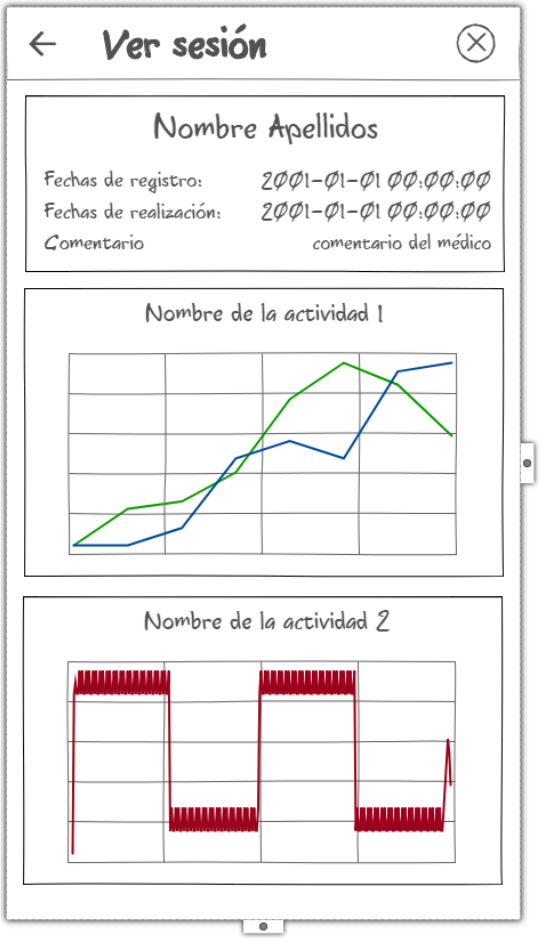

Ilustración 39. Diseño de la pantalla de información de sesión

Al iniciar esta actividad, el método *onCreate* del ciclo de vida de las actividades Android recogerá la información básica de la sesión (identificadores, fechas, paciente, duración y nota) que le envió la pantalla anterior y hará una petición al servidor para recibir toda la información de esta sesión en concreto y así poder construir las gráficas de datos.

Enviamos al servidor una petición POST con *Retrofit* con el identificador del paciente, el identificador de la sesión y la fecha de realización de la sesión.

```
$query = "SELECT * FROM Actividad act
  INNER JOIN Actividad_Sesion act_ses ON (act_ses.Actividad_idActividad
    = act.idActividad)
  INNER JOIN Resultado_Actividad_Sesion res_as ON
    (res_as.Actividad_Sesion_Sesion_idSesion = act_ses.Sesion_idSesion
    AND res_as.Actividad_Sesion_Actividad_idActividad =
```

```
    act_ses.Actividad_idActividad)
  INNER JOIN Resultado res ON (res.idResultado =
    res_as.Resultado_idResultado)
  WHERE res_as.Actividad_Sesion_Sesion_idSesion='".$idSesion."'
  ORDER BY act_ses.orden ASC";

$query2 = "SELECT * FROM Sesion ses
  INNER JOIN Actividad_Sesion act_ses ON (act_ses.Sesion_idSesion =
    ses.idSesion)
  INNER JOIN Resultado_Actividad_Sesion res_as ON
    (res_as.Actividad_Sesion_Sesion_idSesion = act_ses.Sesion_idSesion
    AND res_as.Actividad_Sesion_Actividad_idActividad =
    act_ses.Actividad_idActividad)
  INNER JOIN Resultado res ON (res.idResultado =
    res_as.Resultado_idResultado)
  WHERE ses.Paciente_idPaciente='".$idPaciente."' AND
    ses.fechaRealizacion < '".$fechaRealizacion."' AND (";
    for($i=0;$i<count($idsActividades);$i++){
      $query2 = $query2."act_ses.Actividad_idActividad =
      ".$idsActividades[$i]."";
      if($i<(count($idsActividades)-1)){
        $query2 = $query2." OR ";
      }
    }
    $query2 = $query2.") ORDER BY ses.fechaRealizacion ASC";

$query3 = "SELECT * FROM Actividad act
  INNER JOIN Actividad_Sesion act_ses ON (act_ses.Actividad_idActividad
    = act.idActividad)
  WHERE act_ses.Sesion_idSesion='".$idSesion."' ORDER BY act_ses.orden
    ASC";
```

Con la primera de las consultas obtenemos los datos de la sesión seleccionada, con la segunda captamos la información de las sesiones anteriores (válido para dibujar las gráficas de las actividades de goniómetro). La tercera consulta solo se producirá si la primera no dio ningún resultado, lo que significa que la sesión no está realizada.

La respuesta que envíe el servidor estará en formato JSON y dividida en dos partes, la sesión actual y las sesiones anteriores. Tendrá el siguiente formato:

```
    {
        "estado": "OK",
        "sesionActual": [
            {
                "idActividad": "",
                "nombreActividad": "",
                "tipoActividad": "",
                "duracionActividad": "",
                "orden": "",
                "resultado": [
                    {
                        "nombre": "",
                        "valor": ""
```

```
                    }
                ]
            },
            {
                "idActividad": "",
                "nombreActividad": "",
                "tipoActividad": "",
                "duracionActividad": "",
                "orden": "",
                "resultado": [
                    {
                        "nombre": "",
                        "valor": ""
                    },
                    {
                        "nombre": "",
                        "valor": ""
                    }
                ]
            },
        ],
        "sesionesAnteriores": [
            {
                "fechaRealizacion": "",
                "actividades": [
                    {
                        "idActividad": "",
                        "resultado": [
                            {
                                "idResultado": "",
                                "nombre": "",
                                "valor": ""
                            }
                        ]
                    }
                ]
            }
        ]
    }
```

Los valores de las actividades pueden ser un valor simple que indique un ángulo del goniómetro (el cual será representado junto a los valores de sesiones anteriores de esa misma actividad) o puede ser una serie de valores que indican el registro de los ángulos de interés durante el ejercicio, que se dividen en series y cada valor de la serie está separado del siguiente por el símbolo “,”.

La aplicación, al recibir los datos de las actividades, rellenará el primer cuadro con la información general y mediante la librería externa “MPAndroidChart”. Esta librería nos ofrece una sencilla solución para la creación de gráficas que se mantiene actualizada y tiene una gran variedad de gráficas que pueden ser muy interesantes para futuras implementaciones tanto de la aplicación como de la plataforma web, como es el caso de los “PieCharts” (para la representación de datos generales de los pacientes) o

los “RadarCharts” (para la representación de la progresión de los pacientes en diferentes destrezas) [45]. Para añadir esta herramienta añadiremos al fichero “Gradle” la siguiente línea:

```
compile 'com.github.PhilJay:MPAndroidChart:v3.0.3'
```

La aplicación tendrá dos métodos distintos para representar un tipo de gráfica u otra, puesto que el tratamiento de los datos debe ser distinta. Si el nombre de la actividad es “O_medicion”, se interpretará como que es la medición del goniómetro y se deben de buscar valores de la misma actividad en sesiones con fecha anterior a la realizada y que no hayan sido descartadas.

```
generarMedicionGraph(tituloGrafica, datainfo, data, dates);
```

En este caso, el título de la gráfica se obtendrá del tipo de actividad realizada, la variable “datainfo” contiene el nombre que recibirán los datos representados en la leyenda, mientras que las otras dos variables son vectores que contienen la información que debe ser representada y las fechas de realización.

Si el nombre de la actividad es “O_muestreo”, sabremos que los datos recogidos son del tipo de muestreo de ángulos durante un ejercicio, por lo que en este caso no hace falta recurrir a información de sesiones anteriores, sino que hay que separar las series y los valores de estas mediante el método “Split”.

```
generarMuestreoGraph(tituloGrafica, dataGraphs);
```

En el método que hemos creado para la representación de los datos solo enviaremos el título de la gráfica al igual que en el otro caso, y los datos. Los datos serán una lista de vectores, donde cada vector es una serie, y la lista puede tener varias series que representará en el mismo gráfico. Cada serie representada será dibujada con un color distinto.

Por otro lado, con el botón de la barra superior de herramientas podremos descartar una sesión realizada o borrar una que no ha sido completada aún. Al pulsar sobre él, se envía al servidor una petición para modificar o eliminar la sesión mediante el identificador de la sesión y la acción a realizar:

```
if($accion == "Descartar"){
   $query = "UPDATE sesion SET descartada=1 WHERE idSesion =
   '".$idSesion."'";
}else if($accion == "Borrar"){
   $query = "DELETE FROM sesion WHERE idSesion='".$idSesion."'";
}
```

La aplicación recibirá un estado “OK” y una frase de éxito en caso de que se haya realizado correctamente la acción. En caso de error, se enviará un estado “KO” y una frase explicando el motivo del error.

Ilustración 40. Diseño final de la pantalla de visualización de sesión

Gracias a la flexibilidad de la librería que hemos utilizado, el usuario podrá hacer zoom en las gráficas tanto de manera vertical como horizontal. En el caso de las actividades de muestreo de ejercicios, al llegar a un determinado nivel de proximidad de los datos, aparecerá el valor numérico de los mismo de forma similar a como aparece en los puntos de los ejercicios de medición del goniómetro que aparecen en la Ilustración 40.

Actividad de ajustes

En el menú lateral desplegable hay una sección “Ajustes” en la que al acceder se creará una actividad sencilla que mostrará las preferencias configurables de la plataforma. Lo primero que hará esta actividad será enviar una petición a la base de datos para recibir las preferencias que están asociadas al usuario para que posteriormente pueda modificarlas.

Para aumentar la simplicidad de la aplicación, la solicitud de los ajustes se hará con el mismo método que la modificación de las mismas. El servidor recibirá el identificador de usuario y una variable “ajustes”. Si la variable está vacía, el servidor hará una *query* para obtener los datos. Si la variable tiene información del tipo clave-valor

(idpreferencia1=valor1,idpreferencia2=valor2,…), se borrarán las preferencias y se añadirán estas nuevas.

```
if($ajustes == ""){
  $query = "SELECT * FROM Usuario_Preferencia u_pref
    INNER JOIN Preferencia pref ON (u_pref.Preferencia_idPreferencia =
      pref.idPreferencia)
    WHERE u_pref.Usuario_idUsuario='".$idUsuario."'";
}else{
  $query = "DELETE FROM Usuario_Preferencia WHERE Usuario_idUsuario =
    '".$idUsuario."'";

  $query2 = "INSERT INTO `Usuario_Preferencia` VALUES ";
  $ajustesExploded = explode(",", $ajustes);
  for($i=0;$i<count($ajustesExploded);$i++){
    $ajustesValores = explode("=", $ajustesExploded[$i]);
    $query2 = $query2."('".$idUsuario."', '".$ajustesValores[0]."',
      '".$ajustesValores[1]."')";
    if($i<(count($ajustesExploded)-1)){
      $query2 = $query2.",";
    }
  }
}
```

Al almacenar las preferencias en el servidor, tenemos la ventaja de poder añadir o cambiarlas y que el usuario no tenga que actualizar la aplicación, además de poder cambiar estos parámetros en la plataforma web o en la aplicación y tenerlos sincronizados.

Finalización del desarrollo

Una vez que la fase de desarrollo está acabada, crearemos el fichero APK que se instalará en los dispositivos. Para ello, deberemos acceder en el menú superior del AndroidStudio a la pestaña "*Build*" o pulsar "Alt+B" y pulsar en la opción "*Build APK(s)*".

Cuando se considere que la aplicación está lista para ser publicada en el mercado de aplicaciones de Google, deberemos generar un fichero APK firmado. Esto lo haremos en el menú anterior pero esta vez seleccionando la opción "*Generate Signed APK…*". Nos aparecerá un cuadro de diálogo en el que habrá que seleccionar una ubicación para el fichero de firma (sin este fichero no será posible actualizar la aplicación en el futuro, por lo que es muy importante almacenarlo en un lugar seguro), una contraseña para el fichero de la firma, un alias para la firma y una contraseña para la firma en sí.

En el siguiente paso, el asistente nos preguntará por una ubicación para el fichero APK firmado, el tipo de "*Build*" el cual debe ser "*Release*" y una versión de firma. Una vez rellenado todos los campos podemos pulsar el botón de finalizar y nuestro archivo para instalar la aplicación en cualquier dispositivo estará listo.

Capítulo 5. Pruebas

Para realizar esta sección del documento, vamos a realizar un estudio para determinar la utilidad, facilidad de uso e impacto de la aplicación en usuarios de un perfil similar al sector específico al que está destinada.

Teniendo en cuenta que la aplicación actualmente está destinada a los usuarios del tipo "médico", el sector que escogeremos para realizar las pruebas serán personas de ambos sexos con una edad comprendida entre los 30 y los 60 años, con estudios superiores. Todos los elementos del estudio han tenido experiencia previa con el sistema operativo Android.

Puesto que esta aplicación se basa en la plataforma Telekin, el sector seleccionado para realizar el estudio recibirá antes de empezar unas nociones de cómo funciona la herramienta. Posteriormente se les pedirá realizar una serie de acciones sin haber tenido contacto anterior con la aplicación o la web. Como la aplicación realizada es un complemento de la plataforma web, las últimas pruebas se realizarán habiendo explicado detalladamente cómo funciona el sitio web, y se volverán a repetir las pruebas realizadas anteriormente.

Este estudio se realizará de forma que los usuarios serán anónimos, solo se recogerá su edad, sexo y resultados. Cada elemento del experimento se denominará como "sujeto X" siendo la X el orden en el que se realizan los ejercicios.

Las pruebas que se realizarán serán:

- Prueba 1: creación de un paciente.
- Prueba 2: activación de un paciente.
- Prueba 3: recuperación de un paciente.
- Prueba 4: creación de una sesión.
- Prueba 5: modificación de una terapia.
- Prueba 6: visualización de una terapia en concreto.

Pruebas pre-formación:

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Prueba 6
Sujeto 1	Superada	No superada	No superada	No superada	Superada	Superada
Sujeto 2	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 3	Superada	Superada	Superada	No superada	Superada	Superada
Sujeto 4	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 5	Superada	Superada	No superada	No superada	Superada	Superada
Sujeto 6	Superada	Superada	Superada	No superada	Superada	Superada
Sujeto 7	Superada	Superada	Superada	Superada	Superada	Superada

Sujeto 8	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 9	Superada	Superada	Superada	No superada	Superada	Superada
Sujeto 10	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 11	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 12	Superada	Superada	No superada	No superada	Superada	Superada
Sujeto 13	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 14	Superada	Superada	Superada	No superada	Superada	Superada
Sujeto 15	Superada	Superada	Superada	No superada	Superada	Superada

Tabla 2. Resultados de las pruebas pre-formación

Pruebas post-formación:

	Prueba 1	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**	**Prueba 6**
Sujeto 1	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 2	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 3	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 4	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 5	Superada	Superada	Superada	No superada	Superada	Superada
Sujeto 6	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 7	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 8	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 9	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 10	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 11	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 12	Superada	Superada	Superada	No superada	Superada	Superada
Sujeto 13	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 14	Superada	Superada	Superada	Superada	Superada	Superada
Sujeto 15	Superada	Superada	Superada	Superada	Superada	Superada

Tabla 3. Resultados de las pruebas post-formación

Como se puede observar, tras saber la disposición de las herramientas, los sujetos han superado la mayoría de las pruebas de forma satisfactoria.

Antes de saber el funcionamiento de la plataforma, el 86,67% de las pruebas realizadas fueron superadas. Después de aprender el funcionamiento de la plataforma web, el porcentaje de pruebas superadas se incrementó hasta un 97,78%.

Después de conocer básicamente el manejo de la plataforma, todos los usuarios han sabido hacer 5 de las 6 pruebas sin ningún problema. La prueba número 4, que

consiste en creación de una sesión, es la más complicada y tan solo 2 personas han tenido dificultades en realizarla. Sin embargo, con unas nociones más de la utilización de la plataforma en general, se podrían realizar todas las acciones de manera satisfactoria.

Por tanto, se ha conseguido el objetivo de las pruebas, asegurar que la aplicación es apta para que el público de la aplicación la pueda usar sin ninguna dificultad.

Capítulo 6. Conclusiones y líneas futuras

Del estudio de la plataforma de tele-rehabilitación y desarrollo de esta aplicación podemos sacar como conclusiones:

- La importancia de las nuevas tecnologías en el campo de la medicina. – Hoy en día la tecnología avanza a pasos agigantados. Como hemos visto en los primeros capítulos del documento, la necesidad de la rehabilitación es vital para la recuperación de las funciones de una persona tras un accidente u operación. La tecnología nos permite que podamos realizar estas tareas tan delicadas y minuciosas de una forma más sencilla y que se puedan seguir de una manera exhaustiva. Con la herramienta que tratamos en este proyecto, el paciente puede realizar las tareas de rehabilitación desde su propia casa gracias a ciertos medios, entre ellos el médico que lo controla de forma remota, la herramienta Kinect de Microsoft y la aplicación web o móvil que hemos desarrollado a lo largo de todo el documento.

  Con la tecnología de esta aplicación el médico sabrá a ciencia cierta si el paciente está realizando los ejercicios, cuál es su progresión, y en base a esta modificar y adaptar los ejercicios al ritmo del paciente. Esto permite una recuperación más rápida y segura, ya que evita riesgos para el paciente si un ejercicio resulta demasiado avanzado para su nivel físico en un momento determinado.

  Por último, aunque normalmente no se tiene en cuenta, la tecnología y más en concreto esta herramienta Telekin puede hacer que algo complicado a nivel psicológico y generalmente lento, se convierta en un reto entretenido y estimulante para el paciente. Las rehabilitaciones son procesos muy delicados y a veces a ojos de los pacientes no se ve la evolución en cortos periodos de tiempo. Telekin permite que este proceso tenga recompensas y logros reales que ayuden al paciente a visualizar el objetivo (que es la recuperación total) y a verlo mucho más alcanzable.

- La gran superioridad del sistema operativo Android respecto al resto en cuanto a número de ventas. – Las estadísticas que mostramos anteriormente son claras, el 86,1% de los smartphones en 2017 tenían el sistema operativo de Google. Al hacer una aplicación nativa parece lógico querer llegar al mayor número de “clientes” haciendo una única aplicación, y con las limitaciones que presentan las aplicaciones multiplataforma, una aplicación Android es una buena forma de empezar.

  Teniendo en cuenta el porcentaje de usuarios de Android y que los potenciales clientes de la mayoría de las aplicaciones no tienen una preferencia de un sistema operativo u otro, lo más recomendable sería empezar a desarrollar una aplicación Android y como siguiente paso una aplicación iOS. Como entre estos dos sistemas operativos se ocupa el 99,8% del mercado de los teléfonos inteligentes, resultaría totalmente ineficiente

dedicar tiempo en el desarrollo de una aplicación para otra plataforma para cubrir un mercado tan pequeño.

- La importancia de una planificación adecuada de un proyecto. – Cuando se realiza un proyecto hay que dedicar una buena parte del tiempo en pensar las funcionalidades que debe tener la solución, tener en cuenta los requisitos y los medios que disponemos para la elaboración del producto. Dependiendo del tipo de proyecto, esta fase puede durar más que la de desarrollo, pudiendo ser incluso meses.

  Actualmente, las metodologías de trabajo que más se utilizan en entornos laborales son las metodologías ágiles. Las metodologías ágiles son nuevos sistemas de gestión que apuestan por una gestión dinamizada y muy coordinada de los procesos para llevar a un nivel óptimo el uso de los recursos invertidos.

  En concreto, la metodología Scrum permite abordar proyectos complejos desarrollados en entornos dinámicos y cambiantes de un modo flexible. Está basada en entregas parciales y regulares del producto final en base al valor que ofrecen a los clientes. De esta forma, el cliente tiene una visión más real del producto y se pueden atajar los problemas de manera más rápida [46].

  En nuestro caso, no hemos utilizado esta metodología de trabajo, pues es ideal para acometer proyectos desarrollados en entornos complejos que exigen rapidez en los resultados y flexibilidad, sin embargo, nuestro escenario es distinto.

  Puesto que la aplicación desarrollada se basa en una herramienta web ya hecha no hemos necesitado de ese tiempo de desarrollo. No obstante, teniendo en cuenta este trabajo de fin de grado como un proyecto profesional, fue necesario hacer 3 grandes cambios en determinadas partes de la aplicación. Todos ellos por cambios en la funcionalidad para adaptarla mejor a las necesidades del público objetivo de la aplicación.

- La trascendencia de un correcto diseño para la experiencia de usuario. – En todos los productos que se comercializan un aspecto fundamental es el diseño. Un buen diseño es lo que diferencia un producto profesional y de confianza de uno que no lo es. Para un producto software, como es el caso tratado, el buen diseño es fundamental para causar una buena impresión al usuario. Para el caso de las aplicaciones, este apartado puede ser decisivo, pues si el usuario que ha instalado la aplicación decide que el diseño es mediocre, puede que la desinstale y no se plantee volverla a descargar en el futuro, aunque haya sido actualizada.

  Cuando se realiza un diseño para una aplicación hay que tener presente cual va a ser el público objetivo de la misma, y adaptar la interfaz a sus necesidades y expectativas. Por ejemplo, no tiene sentido hacer una

aplicación para niños con un diseño con el que se haría una aplicación de información sobre mercados financieros.

Por otra parte, hay que estudiar bien la tecnología utilizada e integrar los diseños típicos de esta tecnología en la aplicación. De esta forma, para el usuario será mucho más sencillo entender e interpretar la información. Además, será mucho más sencilla de utilizar por primera vez si las acciones típicas de ciertos componentes se realizan de la misma manera que en la mayoría de aplicaciones.

Tanto un diseño demasiado sencillo como uno demasiado recargado puede ser un error. Un diseño sencillo en exceso puede hacer que la aplicación esté demasiado vacía y los elementos muy dispersos. Por otro lado, un diseño muy complicado y recargado puede hacer que el usuario se vea superado por tanta información tan junta y no sepa cómo gestionarla.

Además, en nuestro caso en concreto, la aplicación realizada depende de una plataforma web ya realizada y asentada. Si el diseño de la aplicación es muy distinto y la distribución de secciones y acciones es muy diferente, el usuario creerá que está haciendo cosas distintas y quizás no esté dispuesto a aprender a usar dos plataformas. El objetivo principal de una aplicación es siempre facilitar la vida a los usuarios.

- Coste real de una aplicación completa. – Como conclusión final, podemos estimar cuanto puede ser el coste de producción de una aplicación de estas características, teniendo en cuenta los gastos de personal y los gastos de equipamiento (aunque este puede ser amortizado para otros proyectos). Para la estimación del coste de personal tendremos en cuenta el tiempo invertido en el desarrollo del proyecto y no en la formación previa.

	€/Unidad	Unidades	Subtotal (€)
Ordenador portátil	1.000,00€	1	1.000,00€
Horas de trabajo	15,00€	350	5.250,00€
Subtotal			6.250,00€
Gastos indirectos (15%)			937,50€
Total			**7187,50€**

Tabla 4. Estimación presupuestaria

El gasto principal del proyecto es el gasto de personal. El gasto de equipamiento podría ser incluso algo menor, puesto que el programa utilizado (Android Studio) no requiere de un ordenador con unas altas prestaciones.

La mayoría de las aplicaciones y páginas web suelen actualizar sus productos cada cierto periodo de tiempo, tanto para corregir errores como para añadir nuevas funcionalidades o mejorar las que ya hay. Si un producto no se actualiza y evoluciona

con la tendencia del mercado, puede hacer que los usuarios de estas plataformas decidan buscar una alternativa más moderna. Por lo que es muy interesante plantear las mejoras que podrían llevarse a cabo en este proyecto a largo plazo.

Implementación de otros tipos de usuarios

La aplicación realizada a lo largo de este documento consiste en una herramienta de gestión de la plataforma creada por el GTI de la Universidad de Valladolid. Teniendo en cuenta la extensión del trabajo, solo ha sido posible la implementación de las acciones asociadas a los usuarios del tipo médico.

La evolución lógica de la aplicación es ampliarla para que implemente a los otros tipos de usuarios. La aplicación ha sido realizada con vistas a futuras implementaciones y han sido planificadas las expansiones como estos nuevos tipos de usuarios. Actualmente el menú desplegable lateral tiene como campos ocultos las secciones correspondientes al usuario administrador, como la gestión de usuarios (añadirlos, modificarlos, borrarlos, cambiarle los privilegios, etcétera), el configurador de juegos (donde se podrán modificar ciertos parámetros de las actividades) y la sección de los rangos articulares. Para el usuario del tipo "familiar" se ocultarán las secciones referentes a la modificación de terapias y pacientes, de tal forma que solo pueda ver las sesiones y el progreso del paciente al que está relacionado.

La implementación del usuario familiar es la más sencilla, puesto que, en principio, no hay que añadir ninguna nueva funcionalidad a la aplicación, tan sólo habría que ocultar acciones a este tipo de usuario. Por otro lado, el administrador sí que requiere de algo más de desarrollo. Habrá que crear el contenido de las tres secciones comentadas anteriormente y asegurarse de que el resto de la aplicación muestra todos los usuarios asociados a los médicos que administra. En esto último no debería haber problema puesto que el *backend* fue configurado acorde a las necesidades de la aplicación completa.

El trabajo de ampliación de usuarios será algo más sencillo ahora, puesto que la fase de diseño y planificación han sido realizadas. Además, ya hay ciertas funcionalidades comunes ya insertadas, como Retrofit, GSON o Iconify.

Expansión a otras plataformas

Como ya mencionamos en otras ocasiones en el documento, aunque el sistema operativo Android es el líder en cuanto a cuota de mercado en el mundo de los teléfonos inteligentes, existe otra alternativa que ocupa casi el resto de usuarios de *smartphones*. El sistema operativo de la empresa Apple "iOS" está presente en algo menos de un 15% de los teléfonos actuales. Aunque el porcentaje es bajo, cuando hablamos de millones de terminales en todo el mundo, el mercado de iOS resulta ser enorme.

Es por esto por lo que el siguiente paso es realizar la aplicación para el sistema operativo de Apple. En el desarrollo de aplicaciones iOS existen dos lenguajes de

programación en los que hacer nuestros trabajos, *Objective-C* y *Swift*. Actualmente el más utilizado es *Swift*, que surgió en 2014 como respuesta a las carencias del antiguo lenguaje *Objective-C*. Swift es más fácil de interpretar, más sencillo de mantener, más seguro y eficiente. Por ello cada vez son más los desarrolladores que se actualizan a este nuevo lenguaje de programación.

También cabe destacar que Swift es el lenguaje que Apple recomienda para la creación de aplicaciones para su sistema operativo. Hay que recordar que esta compañía se suele mostrar reticente a permitir el desarrollo de aplicaciones en productos que no sean de su marca o en lenguajes que no sean los que ellos permiten. Por ello, cuando se realice la aplicación Telekin para *iPhones* e iPads deberá desarrollarse desde un ordenador de Apple (lo cual implica un desembolso considerable de dinero). Además, debemos recordar que la licencia para publicar aplicaciones en el *market* de la marca (iTunes) es de 100 USD al año.

Si consideramos los gastos que conlleva el desarrollo iOS, lo más razonable es realizar la aplicación para Android y, una vez asentada en el mercado, lanzar la aplicación iOS.

Por otro lado, Telekin nació del uso de la herramienta de Microsoft "Kinect" para la rehabilitación de pacientes. Actualmente se ha suspendido la fabricación del adaptador y sensor Kinect, pero la tecnología Kinect sigue existiendo en productos como HoloLens, el asistente de voz de Cortana, el sistema de Id. facial biométrico de Windows *Hello* y una interfaz de usuario con reconocimiento de contexto [47].

Microsoft está trabajando con Intel para proporcionar una opción para los desarrolladores que quieren hacer la transición desde Kinect para la plataforma Windows. No obstante, seguirá proporcionando compatibilidad con Kinect para Windows SDK a través de foros en línea y soporte técnico de pago. Microsoft anima a los desarrolladores a realizar la transición desde el hardware de Kinect a las cámaras de profundidad *RealSense* de Intel.

Otras posibles implementaciones

Además de todas las líneas futuras mencionadas anteriormente, hay algunas implementaciones que se pueden realizar en la aplicación desarrollada en este documento y en el proyecto, que mejorarían notablemente la experiencia de usuario:

- Asociación de una fotografía a los usuarios. – En el menú desplegable lateral de la aplicación, se pretende mostrar en la parte superior la información básica del usuario de la sesión actual. Como la tabla de la base de datos correspondiente a los usuarios no tiene contemplado la asignación de una fotografía, en la aplicación se muestra una imagen por defecto hasta que esta funcionalidad sea añadida.
- Utilización de la cámara. – Teniendo en cuenta que la aplicación funciona en teléfonos y *tablets*, que son dispositivos con cámara, y que, como se ha

expuesto en el punto anterior, sería interesante añadir fotos a los usuarios, se podría implementar un botón de acción en la creación de usuarios y pacientes en el que se accediese a la cámara para realizar una fotografía. La fotografía se podría modificar al igual que cualquiera de los datos del usuario y se almacenaría en la tabla correspondiente. Actualmente se asigna una fotografía aleatoriamente a los nuevos pacientes creados. Esta funcionalidad simplificaría tremendamente la creación de los usuarios, puesto que en un ordenador habría que tener el fichero de la fotografía o conectar un hardware que lo pudiese hacer.

- Implementación de logs. – Una herramienta muy útil sería el registro de las acciones asociadas a los pacientes de un médico. De tal forma que cuando un paciente empiece o termine una sesión, aparezca registrado en una lista junto con el momento en el que se produjo la acción y si procede, información adicional. Esta lista de últimos movimientos podría mostrarse en la página principal de la aplicación junto con las gráficas que ya figuran en ella.

- Creación de notificaciones. – Cuando se hubiese hecho la implementación de los logs, el siguiente paso podría ser la inserción de notificaciones al médico para informarle de los pacientes que hayan finalizado una sesión (y esta opción se pudiese activar o desactivar para cada paciente). Mediante las notificaciones "*push*", se podría implementar esta funcionalidad. La herramienta más utilizada para las notificaciones de este tipo es "*Firebase*", creada por Google y recomendada en el desarrollo Android. Además, *Firebase* cuenta con una gran cantidad de productos que sería interesante introducir en la aplicación, como herramientas de monitorización de rendimiento, de almacenamiento o de predicciones de comportamiento.

Bibliografía

[1] «Instituto Nacional de Estadística,» [En línea]. Available: http://www.ine.es/jaxi/Datos.htm?path=/t38/bme2/t10/a102/l0/&file=1500051.px. [Último acceso: 14 Febrero 2018].

[2] «Instituto Nacional de Estadística,» [En línea]. Available: http://www.ine.es/jaxi/Datos.htm?path=/t38/bme2/t22/a063/l0/&file=0202005.px. [Último acceso: 14 Febrero 2018].

[3] «Instituto de Neurología Cognitiva,» [En línea]. Available: http://www.ineco.org.ar/deterioro-cognitivo/. [Último acceso: 14 Febrero 2018].

[4] «Instituto de Especialidades Neurológicas,» [En línea]. Available: http://www.iensa.es/especialidades/neuropsicologia/item/37-rehabilitacion-neuropsicologica/37-rehabilitacion-neuropsicologica. [Último acceso: 14 Febrero 2018].

[5] «Instituto Nacional de Estadistica,» [En línea]. Available: http://www.ine.es/jaxi/Datos.htm?path=/t15/p418/a2008/hogares/p02/modulo2/l0/&file=03003.px. [Último acceso: 2018 Febrero 14].

[6] «Grupo de Telemática e Imagen,» [En línea]. Available: http://gti.tel.uva.es/. [Último acceso: 5 Febrero 2018].

[7] «Consumer Barometer with Google,» [En línea]. Available: https://www.consumerbarometer.com/en/trending/?countryCode=ES. [Último acceso: 5 Febrero 2018].

[8] «AppBrain,» [En línea]. Available: http://www.appbrain.com/stats/number-of-android-apps. [Último acceso: 16 Febrero 2018].

[9] «Gartner,» [En línea]. Available: https://www.gartner.com/newsroom/id/3725117. [Último acceso: 26 Enero 2018].

[10] «Play Console Help,» [En línea]. Available: https://support.google.com/googleplay/android-developer/answer/6112435. [Último acceso: 26 Enero 2018].

[11] «Support - Apple Developer,» [En línea]. Available: https://developer.apple.com/support/compare-memberships/. [Último acceso: 26 Enero 2018].

[12] «iClarified,» [En línea]. Available: http://www.iclarified.com/62604/iphone-8-is-the-worlds-fastest-smartphone-destroys-samsung-galaxy-s8-and-note-8-in-benchmarks-charts. [Último acceso: 27 Enero 2018].

[13] «Telerehabilitación.net,» [En línea]. Available: http://www.telerehabilitacion.net/buscar-articulos-2/. [Último acceso: 10 junio 2018].

[14] «El Mundo,» [En línea]. Available: http://www.elmundo.es/economia/2014/12/03/547f3307ca47410c138b4573.html. [Último acceso: 10 Junio 2018].

[15] «Toyra,» [En línea]. Available: http://www.toyra.org/que-toyra/. [Último acceso: 10 Junio 2018].

[16] «TRAM,» [En línea]. Available: https://tram.proxiasuite.com/es/proyecto. [Último acceso: 10 Junio 2018].

[17] «Indra,» [En línea]. Available: https://www.indracompany.com/es/indra/tram-telerehabilitacion-audiovisual-motora. [Último acceso: 10 Junio 2018].

[18] «Play For Health,» [En línea]. Available: http://www.play4health.com. [Último acceso: 10 Junio 2018].

[19] «Semantic Scholar,» [En línea]. Available: https://www.semanticscholar.org/paper/KiReS%3A-A-Kinect-based-telerehabilitation-system-Ant%C3%B3n-Go%C3%B1i/4826f55f3b065ffd6d8f410190b2af671d8e364c. [Último acceso: 10 Junio 2018].

[20] «Universidad Politecnica de Valencia,» [En línea]. Available: http://www.upv.es/noticias-upv/noticia-5546-telerrehabilita-es.html. [Último acceso: 11 Junio 2018].

[21] «Telerehabilitacion.net,» [En línea]. Available: http://www.telerehabilitacion.net/elerehabilitacion-en-el-kennedy-krieger-institute/. [Último acceso: 11 Junio 2018].

[22] «Grupo de Telemática e Imagen,» [En línea]. Available: http://gti.tel.uva.es/juegos-serios-para-rehabilitacion-fisica-yo-cognitiva/telekin-sistema-de-telerehabilitacion-para-discapacidades-motoras-y-cognitivas/. [Último acceso: 18 Febrero 2018].

[23] «Engadget,» [En línea]. Available: https://www.engadget.com/2013/05/21/microsofts-new-kinect-is-official/. [Último acceso: 18 Febrero 2018].

[24] «MIT App Inventor,» [En línea]. Available: http://appinventor.mit.edu/explore/about-us.html. [Último acceso: 22 Febrero 2018].

[25] «Eclipse,» [En línea]. Available: https://www.eclipse.org/org/. [Último acceso: 22 Febrero 2018].

[26] «Android Developers,» [En línea]. Available: https://developer.android.com/training/index.html. [Último acceso: 22 Febrero 2018].

[27] «Android Developers,» [En línea]. Available: https://developer.android.com/studio/releases/index.html. [Último acceso: 22 Febrero 2018].

[28] «Android Developers Blog,» [En línea]. Available: https://android-developers.googleblog.com/2011/12/android-403-platform-and-updated-sdk.html. [Último acceso: 22 Febrero 2018].

[29] «Kantar Worldpanel,» [En línea]. Available: https://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=1361. [Último acceso: 22 Febrero 2018].

[30] «NinjaMock,» [En línea]. Available: https://ninjamock.com/home/index. [Último acceso: 22 Febrero 2018].

[31] «Material Design Pallete,» [En línea]. Available: https://www.materialpalette.com. [Último acceso: 9 Marzo 2018].

[32] «NickKolenda,» [En línea]. Available: https://www.nickkolenda.com/font-psychology/. [Último acceso: 9 Marzo 2018].

[33] «Google Fonts,» [En línea]. Available: https://fonts.google.com/specimen/Dosis?selection.family=Dosis. [Último acceso: 9 Marzo 2018].

[34] «Github,» [En línea]. Available: https://github.com/chrisjenx/Calligraphy. [Último acceso: 28 Abril 2018].

[35] «Android Developers,» [En línea]. Available: https://developer.android.com/reference/android/os/AsyncTask. [Último acceso: 7 Abril 2018].

[36] «Android Developers,» [En línea]. Available: https://developer.android.com/training/volley/. [Último acceso: 7 Abril 2018].

[37] «Retrofit,» [En línea]. Available: http://square.github.io/retrofit/. [Último acceso: 7 Abril 2018].

[38] «INSTRUCTURE TECH BLOG,» [En línea]. Available: http://instructure.github.io/blog/2013/12/09/volley-vs-retrofit/. [Último acceso: 7 Abril 2018].

[39] «Android Developers,» [En línea]. Available: https://developer.android.com/reference/android/content/SharedPreferences. [Último acceso: 7 Abril 2018].

[40] «GitHub - Android Iconify,» [En línea]. Available: https://github.com/JoanZapata/android-iconify. [Último acceso: 5 Mayo 2018].

[41] «GitHub - MaterialDrawer,» [En línea]. Available: https://github.com/mikepenz/MaterialDrawer. [Último acceso: 5 Mayo 2018].

[42] «Android Developers,» [En línea]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle. [Último acceso: 12 Mayo 2018].

[43] «Android Developers,» [En línea]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle. [Último acceso: 12 Mayo 2018].

[44] «GitHub - AdvancedRecyclerView,» [En línea]. Available: https://github.com/h6ah4i/android-advancedrecyclerview. [Último acceso: 25 Mayo 2018].

[45] «GitHub - MPAndroidChart,» [En línea]. Available: https://github.com/PhilJay/MPAndroidChart. [Último acceso: 26 Mayo 2018].

[46] «Proyectos ágiles,» [En línea]. Available: https://proyectosagiles.org/que-es-scrum/. [Último acceso: 6 Junio 2018].

[47] «Microsoft Developer,» [En línea]. Available: https://developer.microsoft.com/es-es/windows/kinect. [Último acceso: 9 Junio 2018].

[48] «Aspace,» [En línea]. Available: https://aspace.org/noticia/365/la-tele-rehabilitacion-audiovisual-motora-tram-que-permite-la-rehabilitacion-a-distancia-se-probara-con-personas-con-paralisis-cerebral. [Último acceso: 10 Junio 2018].