UNIVERSIDAD DE  VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

Desarrollo de una aplicación móvil Android para la gestión de citas de fisioterapia y la asignación de ejercicios

Autor:
D. Daniel Manrique Lucas

Tutor:
D. Míriam Antón Rodríguez

Valladolid, 11 de septiembre de 2017

TÍTULO:	**Desarrollo de una aplicación móvil Android para la gestión de citas de fisioterapia y la asignación de ejercicios**
AUTOR:	**D. Daniel Manrique Lucas**
TUTOR:	**Dña. Míriam Antón Rodríguez**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dña. M.ª Ángeles Pérez Juárez**
VOCAL:	**D. Javier Aguiar Pérez**
SECRETARIO	**D. David González Ortega**
SUPLENTE	**D. Jesús Poza Crespo**
SUPLENTE	**Dña. María García Gadañón**
SUPLENTE	**D. Jaime Gómez Gil**

FECHA:	**21 de septiembre de 2017**
CALIFICACIÓN:	

Resumen y Palabras Clave

La aplicación de las Tecnologías de la Información y las Comunicaciones en el ámbito de la medicina está suponiendo una revolución constante en el paradigma de la atención socio-sanitaria. Dentro de este ámbito de las nuevas tecnologías asociadas al ámbito sanitario, el fenómeno del desarrollo de aplicaciones móviles (*apps*), conocido como *mHealth*, está en constante crecimiento y evolución, convirtiendo día a día a las *apps* en herramientas cada vez más indispensables tanto para pacientes como profesionales sanitarios.

Sin embargo, este uso de la *mHealth* no tiene la misma cobertura de aplicación en todas las áreas de la Medicina. En particular, la fisioterapia es una de ellas en la que más se está invirtiendo actualmente para llevar a cabo la transformación de sus infraestructuras como de la formación de sus profesionales en este nuevo ecosistema digital. Por ello, se está haciendo especial énfasis en el desarrollo de aplicaciones de móviles como motor de esta nueva innovación del sector.

En base al contexto y motivación anterior, se propone este Trabajo Fin de Máster (TFM) como una propuesta de valor añadido para el ámbito de la fisioterapia, mediante el desarrollo de una aplicación móvil para el sistema operativo Android. El objetivo de la misma es servir como una herramienta de apoyo de uso diario que facilite y ayude, tanto a pacientes como a profesionales, en su trabajo diario (gestión de reserva de citas y ejercicios a pacientes, consulta de información de clínicas y tratamientos, etc.) así como la interacción entre los mismos.

Para finalizar, se realizaron pruebas para comprobar el correcto funcionamiento de la aplicación desarrollada y poder evaluar su calidad y facilidad de uso. Como ventajas principales se encuentran el factor de la movilidad que aporta la aplicación y la facilidad de uso en el día a día para la realización de tareas comunes al usuario. Como principales aspectos a tener en cuenta, la necesidad de adaptación de la aplicación a nuevas versiones del sistema operativo, así como la posible inclusión de nuevas funcionalidades que la hagan más completa y atractiva a los usuarios.

Palabras Clave

TIC, *mHealth*, Fisioterapia, Apps, Movilidad, Java, Android, MVP, Base de Datos

Abstract and Keywords

Application of Information Technology and Communications in the field of medicine is causing a constant revolution for the new paradigm of socio-health care. The phenomenon of mobile application development (apps) in the area of new technologies applied to healthcare, better known as *mHealth*, is constantly growing and evolving, which is becoming itself essential to both patients and health professionals.

However, the application coverage of *mHealth* is not the same in all different areas of medicine. In particular, physiotherapy is one of the areas where it is being invested heavily in order to carry out the digital transformation of its infrastructures and the training of its professionals. Particular emphasis is being placed on the development of mobile applications as the engine of this new innovation in the sector.

Based on the context and motivation previously exposed, this Master´s Thesis is proposed as a value-added proposal in the field of physiotherapy through the development of a mobile application for the Android operating system. Its objective is to serve as a support tools for daily use which facilitates and helps both patient and professionals in their daily work and interaction between them (management of appointments and exercises to patients, looking for clinic and treatment information, updates on the latest news and upcoming events related to their clinic and treatments, etc.)

Finally, some tests were performed to evaluate and verify the quality and ease of use of the developed application. As the main advantages of it there should be mentioned the mobility factor and user-friendly format of the application. In contrast, there are some points to take into account such as the necessity to adapt the application with the new releases of the operating system and the possible inclusion of new features to make a more comprehensive and attractive version of the application to the user.

Keywords

ICT, *mHealth*, Physiotherapy, App, Mobility, Java, Android, MVP, Database

Agradecimientos

En este pequeño apartado, me gustaría agradecer a todos aquellos y aquellas que han hecho posible el desarrollo y consecución de este Trabajo Fin de Máster.

En primer lugar, y con mi más sentido afecto, a Carmen, por todo su apoyo, comprensión, cariño, ayuda y estar siempre ahí, tanto en los buenos y no tan buenos momentos en cada día. Porque sé que sin ti no hubiera llegado a ser quién soy…

En segundo lugar, a mi familia, por su sacrificio hacia mí y haberme enseñado siempre a luchar por lo que uno quiere en la vida durante todos estos años.

En tercer lugar, a mi tutora Míriam, por todo su esfuerzo y ayuda en los pasos a seguir para la realización de este proyecto, así como haberme dado la oportunidad de poder llevarlo a cabo.

Finalmente, agradecer a todos los compañeros/as, amigos/as, profesores/as y resto de conocidos de los que he aprendido muchas y muy buenas cosas, sacando lo mejor de mí.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Capítulo 1 - Introducción

Capítulo 1 – Introducción

En este primer capítulo se pretende dar una introducción de las aplicaciones de las TIC al campo sanitario y, en concreto, el desarrollo e implementación de las aplicaciones móviles en el ámbito de la salud, como contextualización de este Trabajo Fin de Máster (TFM). Posteriormente se expondrán los objetivos TFM, así como las fases, medios y métodos utilizados en el mismo para la obtención del resultado final. Finalmente se describirá resumidamente el contenido por capítulo de este Trabajo Fin de Máster.

1.1. Las Tecnologías de la Información y Comunicaciones aplicadas a la Sanidad

Una observación detenida de nuestra sociedad pone de manifiesto un cambio radical en la forma en que accedemos, utilizamos y producimos la información. Adicionalmente, existe un incremento sostenido del volumen de información disponible. Estos cambios son en su mayoría debidos al impacto que han tenido Internet, las Tecnologías de la Información y la Comunicación (TIC) y uno de sus productos más recientes las redes sociales (Fernández Silano, 2013).

Las situaciones descritas causan una profunda transformación en los estilos de vida de nuestra sociedad, reflejados en cambios en nuestra cotidianidad, en la forma de aprender y trabajar; incluso transformando las casas de las familias, por la inclusión de aparatos tecnológicos y llegando a impactar a personas completamente ajenas a la tecnología, o que se supone están fuera de su área de influencia (Fernández Silano, 2013).

Esta evolución en las tecnologías produce nuevas formas de relacionamiento social en espacios virtuales, haciendo que las relaciones personales sean más complejas, pero a su vez, más enriquecedoras, permitiendo mayor grado de intercambio e interacción. La salud y la educación son de las áreas más impactadas por los cambios ocasionados por la emergencia de internet y las TIC, cambios estos identificados como características de la era digital. En el caso de la salud, estas asociaciones han dado lugar a novedosos campos y áreas de desarrollo, que permiten la mejora

de las condiciones de vida y de la atención médica y de salud de nuestras comunidades.

En este aspecto, nos centraremos en dos componentes fundamentales de aplicación de las TIC a la Salud: *eHealth* (ó salud electrónica) y *mHealth* (salud electrónica mediante dispositivos móviles). A continuación, se describirán los mismos en mayor profundidad, haciendo especial énfasis en *mHealth*, puesto que el objetivo principal de este TFM es el desarrollo de una aplicación móvil para el ámbito de la fisioterapia.

1.2. *eHealth*

Entre los términos más utilizados en la literatura para referirse al tema de salud en la era digital se encuentran*: electronic Health, e-Health, eHealth* o eSalud en español. Para este TFM nos referiremos a la misma como *eHealth*.

Según los investigadores del *Center for Global eHealh Innovation*, quienes llevaron a cabo una revisión sistemática de 51 definiciones obtenidas de revistas científicas y artículos originales conseguidos de sitios web relevantes, afirman que no existe consenso entre las definiciones encontradas, la mayoría asocia el término a la provisión de servicios y su íntima relación con la tecnología (Fernández Silano, 2013).

La Organización Mundial de la Salud (OMS) y la Organización Panamericana de la Salud (OPS) definieron la *eHealth* dentro de sus iniciativas políticas y estratégicas como: "La eSalud consiste en el apoyo que la utilización costo-eficaz y segura de las tecnologías de la información y las comunicaciones ofrece a la salud y a los ámbitos relacionados con ella, con inclusión de los servicios de atención de salud, la vigilancia y la documentación sanitaria, así como la educación, los conocimientos y las investigaciones en materia de salud" (Fernández Silano, 2013). Además, la OMS divide la *eHealth* en tres áreas principales:

- ✓ Entrega de información de salud a través de las TIC.
- ✓ Uso de las TIC para mejorar los servicios y formación de salud pública.
- ✓ Uso del comercio electrónico (*eCommerce*) en la gestión sanitaria.

Sin embargo, encuentran que la definición más referenciada fue la de Eysenbach en el 2001 (Fernández Silano, 2013): "eSalud es un campo emergente en la intersección de la informática médica, salud pública y las iniciativas privadas, en referencia a los servicios de salud y la información entregada o mejoradas a través de Internet y las tecnologías relacionadas. En un sentido más amplio, el término caracteriza no sólo un desarrollo técnico, sino también un estado de ánimo, una manera de pensar, una actitud y un compromiso para las redes y el pensamiento global, para mejorar la atención de la salud a nivel local, regional y mundial mediante el uso de tecnología de información y comunicación."

También se puede encontrar otra definición para *eHealth* realizada por el Informe Anual sobre el Desarrollo de la Sociedad de la Información en España 2006 (Martel, 2015): "La eSalud se define como la aplicación de las Tecnologías de la Información y Comunicación en el amplio rango de aspectos que afectan el cuidado de la salud, desde el diagnóstico hasta el seguimiento de los pacientes, pasando por la gestión de las organizaciones implicadas en estas actividades. En el caso concreto de los ciudadanos, la eSalud les proporciona considerables ventajas en materia de información, incluso favorece la obtención de diagnósticos alternativos. En general, para los profesionales, la eSalud se relaciona con una mejora en el acceso a información relevante, asociada a las principales revistas y asociaciones médicas, con la prescripción electrónica asistida y, finalmente, con la accesibilidad global a los datos médicos personales a través de la historia clínica informatizada"

Volviendo a la definición dada por Eysenbach, éste propuso también que la *eHealth*, no solo es electrónica, sino que posee características adicionales como la eficiencia, la mejora de la calidad de atención, basada en la evidencia, admite el empoderamiento de pacientes y consumidores, permite procesos educativos, el intercambio de información, extiende la atención de salud más allá de los límites convencionales, es ética y equitativa (Fernández Silano, 2013).

Esta definición, es una declaración de principios, muestra los fines institucionales que estas organizaciones pretenden con la instauración y apoyo a la *eHealth*. Cónsono con esta definición establecen los siguientes componentes principales (Fernández Silano, 2013):

1. Registro médico electrónico (o historia clínica electrónica, HCE): registro en formato electrónico o digital de la información sobre la historia de salud de cada paciente que puede ayudar a los profesionales de salud en la toma de decisiones y el tratamiento.

2. Telesalud (incluida la telemedicina): consiste en la prestación de servicios de salud utilizando las tecnologías de la información y la comunicación, especialmente donde la distancia es una barrera para recibir atención de salud. Este es, tal vez, uno de los componentes más desarrollado. Cada espacialidad clínica ha desarrollado su quehacer, por ejemplo: la teleradiología, mediante la transmisión de imágenes radiológicas por medios electrónicos; teledermatología, la transmisión de imágenes de lesiones dérmicas, para revisión y diagnóstico por especialistas; teleoftalmología, captura de imágenes de fondo de ojo; entre otras.

3. *mHealth* (o salud por dispositivos móviles): es un término empleado para designar el ejercicio de la medicina y la salud pública con apoyo de los dispositivos móviles, como teléfonos móviles, tabletas, dispositivos de monitoreo de pacientes y otros dispositivos inalámbricos.

4. *eLearning* (incluida la formación o aprendizaje a distancia): consiste en la aplicación de las tecnologías de la información y la comunicación al aprendizaje. Puede utilizarse para mejorar la calidad de la educación, aumentar el acceso a la educación y crear formas nuevas e innovadoras de enseñanza al alcance de un mayor número de personas.

5. Educación continua en tecnologías de la información y la comunicación: desarrollo de cursos o programas de salud profesionales (no necesariamente acreditados formalmente) que facilitan habilidades en tecnologías de la información y la comunicación de aplicación en la salud. Estos incluyen los métodos actuales para el intercambio de conocimiento científico como la publicación electrónica, el acceso abierto, la alfabetización digital y el uso de las redes sociales.

6. Estandarización e interoperabilidad: la interoperabilidad hace referencia a la comunicación entre diferentes tecnologías y aplicaciones de software para el intercambio y uso de datos en forma eficaz, precisa y sólida. Esto requiere del uso de estándares, es decir, de normas, regulaciones, guías o definiciones con especificaciones técnicas para hacer viable la gestión integrada de

los sistemas de salud en todos los niveles. La interoperabilidad es de vital importancia para la comunicación dentro del sistema, porque permitiría el uso de cualquier dispositivo o del recurso que se disponga. Una búsqueda somera en Internet sobre esta área demostrará la importancia de estos componentes en la actualidad, tómese el caso de la Telemedicina y sus diferentes ramas, como se ejemplificó o el movimiento por un registro clínico electrónico único, en niveles locales y nacionales y más recientemente, el gran auge de la *mHealth*, que se tratará al final del artículo.

Por otro lado, se pueden enumerar en la Tabla 1 las ventajas y desventajas del uso de la *eHealth* en el ámbito sanitario o de la medicina:

Ventajas	Desventajas
Facilidad y rapidez de comunicación e intercambio de información con una mayor calidad y cobertura	La aplicación de las TIC al ámbito sanitario puede ser interpretado como una deshumanización de la Medicina
Ahorro de tiempo y dinero en el seguimiento de pacientes y tratamientos	Menor valor a la relación médico-paciente clásica de forma presencial
Constante actualización de los contenidos, procedimientos y métodos de aplicación	Poca penetración en el global de profesionales sanitarios
Mejora de la formación e investigación	Necesidad de inversión por parte de la Administración Pública para su implementación en centros sanitarios
Optimización de los procesos sanitarios	Reticencia de algunos profesionales sanitarios a poder tener un contacto digital directo con los pacientes
Facilidad para campañas de educación sanitarias	Falta de evidencia rentable en tecnologías de salud online
Mejora del pronóstico y seguimiento de las enfermedades	Poca interconexión operativa y de sistemas entre áreas de la salud

Personalización del sistema sanitario	Posibles desigualdades potenciales derivadas de la relación digital, especialmente para personas mayores y discapacitadas
Integración de los sistemas sanitarios a los modelos de desarrollo económico de los países	
Participación del ciudadano en temas sanitarios o de salud Seguridad del paciente en todas las etapas del proceso sanitario	

Tabla 1 - Principales ventajas y desventajas del uso de la eHealth en el ámbito sanitario (Elsevier, Equipo Editorial, 2014)

1.3. *mHealth*

La OMS define sanidad móvil ó *mHealth* como: "La práctica de la medicina y la prestación de servicios sanitarios mediante dispositivos móviles, como teléfonos móviles, dispositivos de seguimiento de pacientes, asistentes digitales personales (PDA, *Personal Digital Assistant*) y otros dispositivos inalámbricos" (Comisión Europea, 2014).

Incluye también aplicaciones tales como las de modo de vida y bienestar que pueden conectarse a dispositivos médicos o sensores (por ejemplo, brazaletes o relojes inteligentes (*smartwatches*). También comprende dispositivos de orientación personal, información sanitaria y recordatorios de medicación mediante el envío de mensajes de texto y la telemedicina inalámbrica (Comisión Europea, 2014).

La *mHealth* constituye un sector emergente y en rápida evolución, que tiene el potencial de participar en la transformación de la atención sanitaria y de incrementar su calidad y su eficacia. Las soluciones de *mHealth* comprenden diferentes soluciones tecnológicas que, entre otras funciones, miden las constantes vitales, como la frecuencia cardíaca, el nivel de glucosa en la sangre, la presión arterial, la temperatura corporal y la actividad cerebral (Figura 1). Como ejemplos destacados de aplicaciones cabe mencionar las herramientas de comunicación, información y motivación, tales como los

recordatorios de medicación o las herramientas que proporcionan recomendaciones dietéticas y para mantenerse en forma.

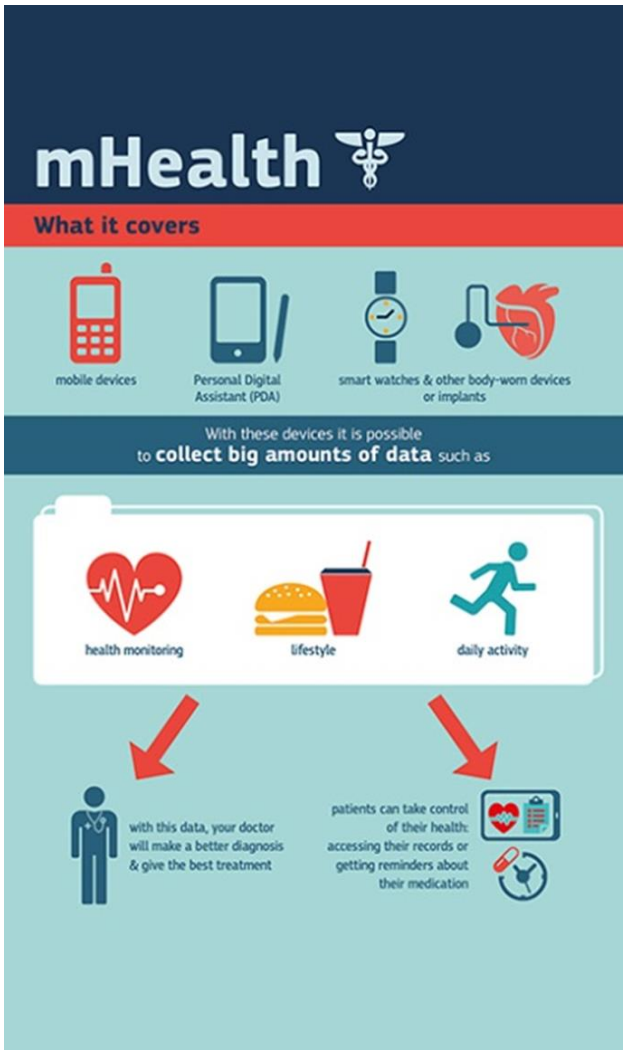

Figura 1 - Infografía sobre la cobertura y propósito de la mHealth (Abdel-lah, 2014)

La expansión de los teléfonos inteligentes ó *smartphones*, así como la de las redes 3G y 4G, ha impulsado la utilización de aplicaciones móviles que ofrecen servicios de atención sanitaria. La disponibilidad de tecnologías de navegación por satélite en dispositivos móviles ofrece la posibilidad de mejorar la seguridad y la autonomía de los pacientes.

Mediante sensores y aplicaciones móviles, la *mHealth* permite la recogida de un considerable número de datos médicos, fisiológicos y relativos al modo de vida, a la actividad diaria y al entorno. Esto podría servir de base para el ejercicio de una práctica sanitaria y actividades de investigación basadas en

resultados comprobados, al tiempo que facilitaría el acceso de los pacientes a su información sanitaria en cualquier lugar y en cualquier momento.

La *mHealth* podría también apoyar la prestación de una atención sanitaria de alta calidad, y permitir un diagnóstico y un tratamiento más precisos. Puede facilitar el trabajo de los profesionales sanitarios con el fin de tratar a los pacientes de manera más eficaz, ya que las aplicaciones móviles pueden favorecer el mantenimiento de un modo de vida sano, dando lugar a una medicación y a un tratamiento más personalizados.

Puede contribuir a la capacitación de los pacientes, ya que estos podrían gestionar su salud de manera más activa, con vidas más independientes en el entorno de sus propios hogares, gracias a la autoevaluación o a soluciones de seguimiento a distancia y de seguimiento de factores ambientales, tales como cambios de la calidad del aire que puedan afectar a ciertas enfermedades.

En este sentido, el objetivo de la *mHealth* no es sustituir a los profesionales sanitarios, que siguen siendo esenciales para proporcionar atención sanitaria, sino que se considera más bien una herramienta de apoyo para la gestión y la prestación de la atención sanitaria. La *mHealth* tiene el potencial de desempeñar un papel crucial para mejorar nuestras vidas. Sin embargo, resulta indispensable garantizar que los ciudadanos pueden utilizar la tecnología con total seguridad.

Paralelo a toda esto, la introducción del concepto de *mHealth* está marcando el comienzo de una revolución en la asistencia sanitaria. Esta revolución debe integrar las 4 "P" de la *mHealth*: *Predictiva, preventiva, personalizada y participativa.* Unas formas tangibles de evidenciar esta revolución son (Diagnostrum, 2015):

1. Están surgiendo nuevas formas de comunicación entre pacientes y profesionales (mensajes de texto vía móvil, correo electrónico, videollamadas, etc.).

2. El aumento constante de aplicaciones para teléfonos móviles ofrece nuevas herramientas al servicio del profesional (soluciones para registros clínicos, calculadoras médicas, herramientas de consulta rápida)

3. Los pacientes cada vez se involucran más en su proceso asistencial, de modo que se hacen expertos en su patología pudiendo llevar a cabo la monitorización de signos o síntomas, que son seguidos y transmitidos vía electrónica al médico. De esta forma finalmente puede tomar de

forma conjunta con el profesional las decisiones sobre su proceso asistencial.

Todo esto permite que se esté transformando a un sistema de salud donde los médicos no solo recetarán medicamentos sino también indicarán el uso de aplicaciones personalizadas que permitan orientar al paciente hacia una participación más directa en el cuidado de su salud.

La *mHealth* se está convirtiendo en uno de los pilares básicos de la sanidad en todas partes, incluidos los países en vías de desarrollo. Y está ocurriendo porque facilita una mejora en la calidad asistencial, una mayor eficacia y un importante ahorro en costes sanitarios: "de casi 100.000 millones de euros en la Unión Europea de aquí al 2017", según un informe de PwC (The App Date, 2016).

Las aplicaciones móviles vinculadas a la salud tienen el potencial de llegar al público general, abordar necesidades específicas y complementar otros desarrollos tecnológicos.

El mercado está evolucionando rápidamente, por lo que existe un sin número de nuevas tecnologías móviles potencialmente disponibles para el sistema de atención de salud. Sin embargo, uno de los grandes hándicaps que nos encontramos ante tal eclosión de tecnología aplicada a la salud es la escasa investigación sistemática sobre el impacto de estas tecnologías en la salud, así como la idoneidad y calidad de muchas de ellas.

De todo el campo de la salud digital, el de la *mHealth* es el que tiene más perspectivas de crecimiento, en ratios de más del 24 por ciento anuales hasta 2022. Cada vez más concienciados por el estilo de vida y el autocuidado de la propia salud, los usuarios tienen en la palma de su mano la posibilidad de acceder a más de 165.000 aplicaciones desde las tiendas de Android y Apple, según QuintilesIMS. Sólo en la última plataforma hay ahora más de 90.000 aplicaciones, el doble que en 2013. Es pronto para decir que el sector de la *mHealth* está viviendo su momento de oro, porque hasta ahora el mercado no ha hecho sino crecer. Basta echar la vista atrás. En 2014, el número global de aplicaciones disponibles era de 100.000, según los datos que acompañaban una consulta de la Comisión Europea, que por aquel entonces creía que la sanidad móvil podría recortar los costes sanitarios de la Unión Europea en 99.000 millones de euros hasta 2017 (Rodríguez, 2016).

IMS Institute señala que el 70% de las apps se dirigen al público en general, a través de los segmentos de bienestar y ejercicio físico (Figura 2). El resto, un 30%, están ideadas para un sector más específico: el de los

profesionales sanitarios y sus pacientes. La funcionalidad más común es la de aportar información (39,8%), seguida, en un porcentaje mucho menor, de proveer de instrucciones de uso (21,4%) y registrar o capturar datos del usuario (18,7%). El principal uso que se les da es el relacionado con la prevención o estilos de vida (alimentación, actividad física, sueño, relajación, control de adicciones, etc.) (The App Date, 2016).

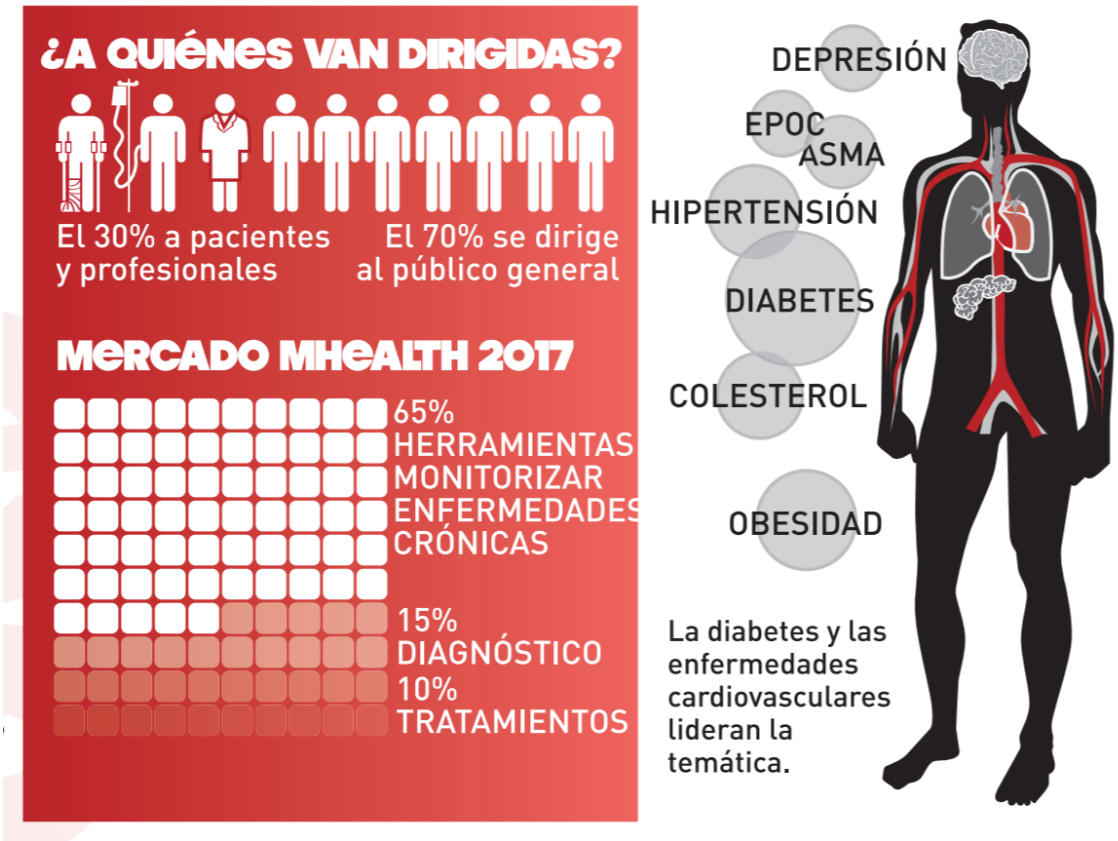

Figura 2 - Enfoque de las aplicaciones móviles en el ámbito de la mHealth (The App Date, 2016)

Pero la *mHealth* ya no se considera como la hermana pequeña de la *eHealth*. Ha cedido ese puesto a una nueva generación de dispositivos que hasta hace poco eran abono para la ciencia ficción: gafas que se conectan a Internet o que permiten combatir el *jetlag*; píldoras que miden el consumo de medicamentos; absorbentes que analizan la orina; camisetas que alertan de los niveles de monóxido de carbono; sujetadores con sensores para detectar el cáncer de mama; relojes y pulseras que registran la temperatura, el ritmo cardíaco, la tensión, la glucosa... Los '*weareables*' o 'prendas electrónicas' son la última gran tendencia dentro del inabarcable campo de la salud digital. Por ahora. Parece que ya tienen

sustituto, posiblemente por los '*hearables'*, sensores colocados en el pabellón auricular que permitirían registrar mediciones de salud de manera muy precisa.

No hay duda de que la *mHealth*: generará un cambio en el modelo asistencial. Minimizará costes, errores diagnósticos, y probablemente permitirá romper esa barrera de comunicación que se establece en muchas ocasiones con el paciente.

1.4. *mHealth* en el ámbito de la fisioterapia

La fisioterapia es una disciplina de la Salud que ofrece una alternativa terapéutica no farmacológica que, en muchos casos, ayuda a paliar los síntomas de múltiples dolencias, tanto agudas como crónicas.

Frecuentemente se relaciona la fisioterapia o a los fisioterapeutas con el masaje, pero su arsenal de técnicas terapéuticas es mucho más amplio.

La fisioterapia y, en concreto, el fisioterapeuta, como agente de salud, trabajan para la prevención, curación y paliación de los problemas musculoesqueléticos y posturales (Colegio Profesional de Fisioterapeutas Comunidad de Madrid, 2015).

El fisioterapeuta desarrolla su labor en cuatro grandes campos (Colegio Profesional de Fisioterapeutas Comunidad de Madrid, 2015):

- ✓ Asistencial: su labor consiste en promover, prevenir, curar y paliar la salud de los pacientes aplicando el abanico de conocimientos adquiridos en su formación continua.

- ✓ Docente: su labor consiste en formar y promover el conocimiento de la Fisioterapia en las escuelas universitarias públicas y privadas, así como en las propuestas formativas convocadas para la formación continua del fisioterapeuta.

- ✓ Investigador: su labor consiste en buscar evidencia científica de los modos de proceder de la fisioterapia, ofreciendo al paciente, en consecuencia, aquellas actuaciones con mayores garantías de éxito, demostradas científicamente por estudios con validez de la comunidad científica.

- ✓ Gestión y Dirección: realizando su labor en la dirección de centros asistenciales, educativos o en colegios profesionales.

Dentro de los ámbitos sanitarios en los que se aplica las TIC, el de la fisioterapia es uno de los sectores en el cuál los profesionales de este sector, los fisioterapeutas, deben encaminar sus esfuerzos hacia una nueva era digital, donde profesionales y expertos en tecnología móvil trabajen en conjunto para realizar investigaciones con las aplicaciones móviles a favor de los usuarios/pacientes. Esta es la principal motivación para el desarrollo de este TFM encaminado a la aplicación de la *mHealth* en este sector mediante el desarrollo de una aplicación móvil.

Así pues, se exponen algunos de los factores clave que los fisioterapeutas deben considerar para la transformación digital del sector (EFisioterapia, 2017):

- ✓ Consciencia digital para visualizar, la dirección que va a tomar el sector de la fisioterapia y el entorno donde se desarrolla la actividad profesional.

- ✓ Transformación de las clínicas y profesionales de la fisioterapia para adaptarse al nuevo marco digital del sector.

- ✓ Completar una formación y una base de conocimientos digitales con el objetivo de crear una cultura digital en todos los miembros del equipo.

- ✓ Uso de servicios y proveedores digitales para dotar de competitividad a las infraestructuras y especialistas del sector.

La aplicación de estos factores, va a permitir que muchos fisioterapeutas puedan ser competitivos en un entorno cada día más digitalizado (EFisioterapia, 2017).

Así pues, la creación y el uso de las aplicaciones móviles (*mHealth)* en el ámbito de la fisioterapia deben responder a las condiciones específicas de cada persona y dar la posibilidad de evaluar el costo-beneficio de la utilización de la aplicación para potenciar la atención por parte de todos los profesionales de la salud, en este caso los fisioterapeutas. Un reto para la

fisioterapia es entonces proyectarse a la era digital usando las tecnologías de la información y la comunicación con el fin de minimizar riesgos y facilitar la adherencia a programas de rehabilitación. Por ello, la *mHealth* en el ámbito de la fisioterapia va a permitir:

- ✓ La innovación en aspectos más cercanos al ámbito clínico como la mejora en el diagnóstico, el análisis masivo de datos (que permitirá por ejemplo abordar mejores políticas de salud pública), la monitorización de pacientes a distancia, etc.

- ✓ El acceso a información de salud en internet y las redes de pacientes y cuidadores están dando lugar a que los usuarios dispongan de mayor información, actualizada y a tener un mayor conocimiento de la misma, teniendo un papel más activo.

- ✓ Beneficiarse del potencial económico y social del nuevo mercado de las aplicaciones móviles y contenidos relacionados con la salud, de gran importancia en programas de prevención y educación en salud, mejora de hábitos de vida, etc.

Por último, cabe destacar un ejemplo ilustrativo del uso de la *mHealth* en el sector de la fisioterapia (Monasterio, 2017):

- ✓ *Fissios* (Google Play, 2017): Cirujanos torácicos, especialistas en Medicina Física y Rehabilitación y fisioterapeutas del Hospital Clínico San Carlos han creado, en este 2017, la aplicación *Fissios*, dirigida a pacientes que tienen programada una cirugía, o que ya han sido intervenidos, con la finalidad de disminuir el riesgo de que desarrollen complicaciones respiratorias postoperatorias, que son muy frecuentes y representan una importante causa de morbimortalidad.

  La herramienta es informativa e interactiva. Une a la realización de ejercicios básicos de fisioterapia respiratoria una serie de recomendaciones encaminadas a disminuir el riesgo de desarrollar estas complicaciones, que conllevan una prolongación de la estancia en el hospital y un aumento del gasto sanitario.

Fissios tiene un diseño básico (Figura 3) e intuitivo con tres pantallas. En la inicial se pueden elegir diferentes funciones de la aplicación, e incluye una cuenta atrás de los días que faltan para la intervención. Otra incluye diez ejercicios de fisioterapia respiratoria explicados con texto e imágenes animadas que guían al paciente a realizar de una manera segura respiraciones eficientes, toser, movilizar secreciones y utilizar el incentivador respiratorio. Por último, una pantalla informativa con consejos preoperatorios – postoperatorios que reforzaran estos ejercicios para preparar mejor al paciente para la cirugía.

Figura 3 - Ejemplo de Fissios como aplicación móvil en el ámbito de la fisioterapia (Monasterio, 2017)

Cada paciente puede personalizar la aplicación introduciendo el nombre y la fecha de la operación. La herramienta envía notificaciones para recordar los días que faltan para la cirugía, los ejercicios que debe realizar y algunos consejos pre y postoperatorios, aumentando la adherencia a las recomendaciones. *Fissios* ya está disponible para Android y Apple de forma gratuita, y, una vez instalada, se puede usar sin necesidad de conexión a internet.

1.5. Objetivos del TFM

El objetivo principal del desarrollo de este TFM es la implementación de una aplicación para dispositivos móviles en el sistema operativo Android que permita, tanto a pacientes y profesionales de una clínica de fisioterapia, interaccionar y gestionar de una forma más fácil y rápida las operaciones propias llevadas a cabo entre los mismos. En particular:

- ✓ Gestión de los ejercicios asignados a los pacientes por parte de los profesionales para facilitar su progreso y mejora a lo largo de su tratamiento
- ✓ Dar la posibilidad al paciente de realizar reservas de citas en diferentes clínicas de forma remota y en cualquier momento
- ✓ Visualizar información acerca de las clínicas de fisioterapia disponibles para sus citas y tratamientos, tanto para pacientes como profesionales
- ✓ Permitir al paciente visualizar información, así como contenido multimedia, sobre los ejercicios asignados durante su tratamiento por un profesional de la clínica.

Por último, se destaca como otro objetivo que se pretende conseguir con la realización de esta aplicación, es disminuir las barreras que supone el desplazamiento del paciente hacia la clínica a la hora de obtener información o realizar gestiones con la clínica de fisioterapia. De esta manera, el paciente puede disponer de un medio, con las únicas necesidades de que se trate de un dispositivo móvil Android con conexión a Internet, para poder realizar todas estas operaciones de forma remota, en cualquier lugar y de manera instantánea.

1.6. Fases y métodos de elaboración del TFM

Para llevar a cabo y lograr los objetivos presentados en el apartado anterior, el TFM ha constado del siguiente diagrama de trabajo (Figura 4), en el que se visualizan las siguientes fases de su desarrollo:

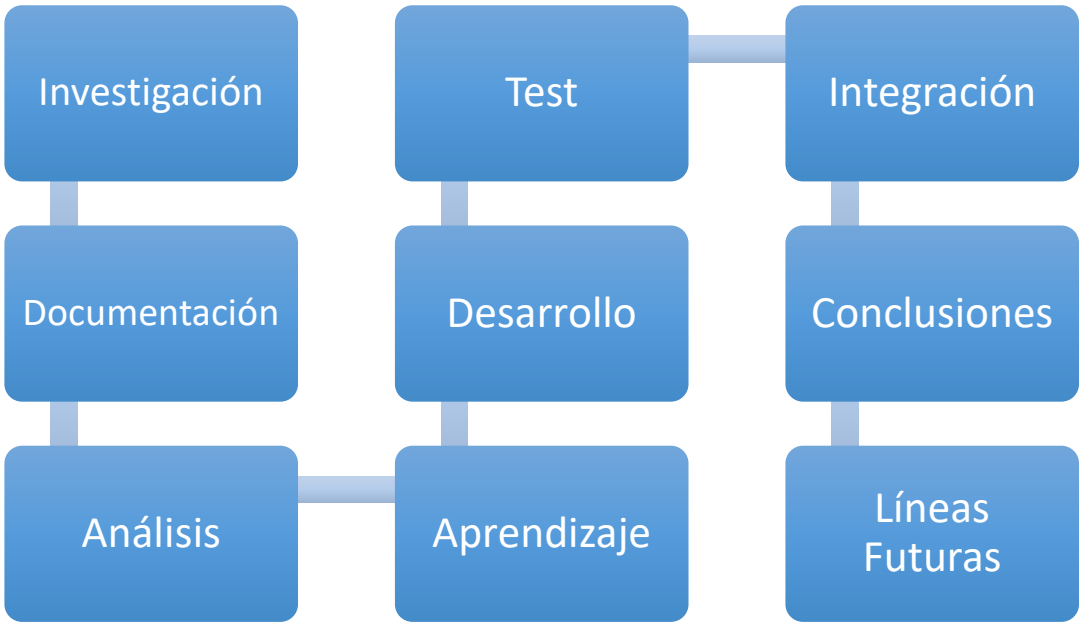

Figura 4 - Diagrama de trabajo del TFM

- ✓ Fase de Investigación: En esta fase se lleva a cabo un estudio del estado de arte del sector de las aplicaciones móviles, el ámbito sanitario y su posible aplicación de las mismas en él. Encontrar una necesidad que pueda aportar valor a los usuarios es el objetivo fundamental en este punto.

- ✓ Fase de Documentación: En ella se consultó toda la bibliografía y recursos digitales para la obtención de toda la información referente para la realización de la aplicación (*eHealth*, *mHealth*, dispositivos móviles, Android, iOS, etc.) así como de la escritura del TFM.

- ✓ Fase de Análisis: Durante esta fase se diseñan y analizan todos los requisitos funcionales y no funcionales que debe implementar la aplicación a desarrollar.

- ✓ Fase de Aprendizaje: En esta fase se aprenden y consolidan los conocimientos técnicos para la programación de la aplicación en

Android, así como el diseño de la arquitectura de la aplicación, su interfaz gráfica y la lógica de la misma en función de los requisitos anteriores.

- ✓ Fase de Desarrollo: Durante la misma se desarrolla la aplicación con el entorno de desarrollo oficial para Android, Android Studio. Se pone en funcionamiento un servidor web con *Apache*, para la conexión y obtención de la información que requiere la aplicación mediante *scripts* en PHP que interaccionan con una base de datos MySQL

- ✓ Fase de Test: Durante la misma se realizan las pruebas y test necesarios de la lógica de la aplicación en un emulador virtual del entorno de desarrollo de Android (Android Studio) con el objetivo de comprobar que el funcionamiento las funcionalidades en base a los requisitos planteados es correcto.

- ✓ Fase de Integración: Esta fase tiene el mismo objetivo que la fase anterior de test, pero con la diferencia de ejecutar y probar la aplicación desarrollada en un entorno real mediante un dispositivo móvil real.

- ✓ Fase de Conclusiones: Esta fase tiene como objetivo la extracción de conclusiones sobre el trabajo desarrollado en todas las fases previas del TFM.

- ✓ Fase de Líneas Futuras: Esta fase tiene el objetivo el establecimiento de unas líneas futuras sobre aspectos a mejorar o añadir nuevas funcionalidades en la aplicación desarrollada para este TFM.

1.7. Medios para la realización del TFM

A continuación, se describen los medios y recursos software utilizados para la realización de la aplicación móvil objetivo de este TFM:

- ✓ Ordenador portátil DELL XPS, con procesador Intel CORE i7, 16 GB de memoria RAM y sistema operativo Windows 10 de 64 bits.
- ✓ Android Studio 2.3 (Entorno integrado oficial de desarrollo de Android)

- ✓ WampServer 3.0.6 64-bit (Servidor Web HTTP Apache 2.4.23, módulo PHP 5.6.25, módulo phpMyAdmin 4.6.4 y motor de base de datos MySQL 5.7.14)
- ✓ MySQL Workbench 5.2 CE (Diseño de la estructura de base de datos de la aplicación)
- ✓ Dispositivo móvil Samsung Galaxy S6 (Para las pruebas de test e integración de la aplicación móvil)

1.8. Estructura del contenido del TFM

A continuación, se describen los capítulos en los que se ha estructurado el contenido de este TFM:

- ✓ Capítulo 1: En este capítulo se realiza una introducción sobre el contenido del TFM, sus objetivos, fases, metodología y medios para su realización.

- ✓ Capítulo 2: En este capítulo se realiza un análisis y comparativa en profundidad de los dispositivos móviles, su historia, las tecnologías actuales de desarrollo de sus aplicaciones, etc.… además de justificar la decisión de la elección de Android como sistema operativo móvil para el desarrollo de la aplicación.

- ✓ Capítulo 3: En este capítulo se describe en profundidad, de manera técnica, el lenguaje de programación Android, su arquitectura y componentes, así como patrones de diseño de arquitecturas para aplicaciones móviles.
- ✓ Capítulo 4: En este capítulo se analizan las características técnicas del proyecto de aplicación desarrollado; entre otras, la base de datos utilizada, sus funcionalidades descritas con diagramas de flujo y su patrón de diseño de arquitectura utilizado.

- ✓ Capítulo 5: En este capítulo se presenta un manual de usuario para la mejor comprensión del funcionamiento y funcionalidades de la aplicación, ilustrado con capturas de la misma.

- ✓ Capítulo 6: En este capítulo se recogen las conclusiones obtenidas del desarrollo de este TFM, así como el establecimiento de líneas futuras para una posible ampliación del mismo.

- ✓ Capítulo de Bibliografía: En él se listan todas las referencias bibliográficas que han sido consultadas para la realización del TFM.

- ✓ Anexo: En él se calculará y detallará el presupuesto económico que ha supuesto la realización de este TFM.

Capítulo 2 - Tecnologías para el desarrollo móvil

Capítulo 2 - Tecnologías para el desarrollo móvil

2.1. Dispositivos móviles

Una gran cantidad de dispositivos electrónicos se clasifican actualmente como dispositivos móviles, desde teléfonos hasta *tablets*, pasando por dispositivos como lectores de RFID[1]. Con tanta tecnología clasificada como móvil, puede resultar complicado determinar cuáles son las características de los dispositivos móviles.

A la hora de dar una definición de lo que realmente es un dispositivo móvil, nos encontramos con que no existe un consenso o idea clara sobre ello. En el día a día, este término mayoritariamente se utiliza para designar a ciertos modelos de teléfonos móviles con amplias prestaciones. Sin embargo, un dispositivo móvil no tiene por qué estar enfocado exclusivamente al ámbito telefónico.

Para poder dar una definición precisa, debemos considerar las siguientes características básicas o fundamentales que debe cumplir cualquier aparato para considerarlo como un dispositivo móvil (Pozo, 2011):

- ✓ Son de reducido tamaño, mejorando su usabilidad y haciéndolos fáciles de transportar (movilidad)
- ✓ Ofrecen unas prestaciones de diferente grado en términos de capacidad de procesamiento, memoria y almacenamiento de datos
- ✓ Incorporan elementos de E/S básicos (e.g. una pantalla, altavoces, algún tipo de teclado, etc.) con lo que el usuario puede interaccionar con el mismo.

[1] RFID (*radio frequency identification*): identificación por radiofrecuencia

Más allá de estas características comunes, los dispositivos móviles forman en la actualidad un grupo sumamente heterogéneo y pueden incorporar casi cualquier componente de hardware y software que amplía y diversifica su función inicial. El más frecuente sin duda es la conexión telefónica (incluyendo servicios como el envío de SMS y MMS) o la conexión a Internet de manera inalámbrica mediante estándares de tecnologías de comunicaciones móviles como el 3G ó 4G LTE (*Long Term Evolution*).

2.2. Introducción a las generaciones de dispositivos móviles

A través del tiempo la telefonía móvil ha evolucionado en etapas que se caracterizan por la capacidad de comunicación entre las compañías telefónicas y los usuarios, además de la información que pueden soportar.

Según esto, podemos clasificar la evolución de los dispositivos móviles en una generación 0 o inicial junto a 5 generaciones denominadas como 1G, 2G, 3G, 4G y 5G (Figura 5). A continuación, se hace una breve introducción de las mismas.

Figura 5 - Generaciones de dispositivos móviles (Villalta, 2014)

2.3. Generación inicial o generación 0

El dispositivo móvil se inicia a principios de la Segunda Guerra Mundial, donde ya se veía que era necesaria la comunicación a distancia. Es por ello que la compañía Motorola lanzó el Handie Talkie H12-16 (Figura 6), el cuál permitía comunicarse a distancia con las tropas mediante una transmisión basadas en ondas de radio que no superaban los 600 Khz de frecuencia (Landa, 2015).

Los primeros estándares más utilizados de esta generación fueron:

- ✓ Estándar PTT (*PushToTalk*): Pulsa para hablar
- ✓ Estándar IMTS (*Improved Mobile Telephone System*): Sistema de telefonía móvil mejorado

Figura 6 - HandieTalkie H12-16 (Landa, 2015)

2.3.1. Primera generación (1G)

A partir de 1973, surgieron los primeros dispositivos de primera generación (Figura 7), los cuales eran de gran tamaño y peso. Funcionaban de manera analógica, lo que conllevaba que la transmisión y recepción de datos se apoyaba sobre un conjunto de ondas de radio que cambiaban de modo continuo.

Figura 7 - Dispositivo móvil de primera generación (Landa, 2015)

El hecho de que fueran analógicos traía consigo una serie de inconvenientes, tales como que solo podían ser utilizados para la transmisión de voz (el uso de mensajería instantánea era algo sólo visible en un futuro muy lejano) o su baja seguridad, la cual hacía posible a una persona escuchar llamadas ajenas con un simple sintonizador de radio o, incluso, hacer uso de frecuencias cargando el importe de las llamadas a otras personas (Landa, 2015).

Los estándares más utilizados eran:

- ✓ NMT: *Nordic Mobile Telephone*
- ✓ AMPS: *Advanced Mobile Phone System*

2.3.2. Segunda generación (2G)

Esta generación no supuso un estándar concreto, sino que marcó el paso de la telefonía analógica a la digital, permitiendo de esta manera el manejo de llamadas, más enlaces simultáneos para el mismo ancho de banda y la integración de otros servicios adicionales al de la voz, destacando de entre ellos el Servicio de Mensajes Cortos (SMS, *Short Message Service) (*Figura 8) (Tudela, 2010).

Figura 8 - Dispositivos móviles de segunda generación (Landa, 2015)

Los estándares más utilizados en esta generación fueron:

- ✓ GSM: *Global System for Mobile Communications* - Sistema Global para Comunicaciones Móviles
- ✓ CDMA: *Code Division Multiple Access* - Acceso Múltiple por División de Código
- ✓ GPRS: *General Packet Radio Service* - Servicio General de Radio por Paquetes

2.3.3. Tercera generación (3G)

A partir del año 2001, se introdujeron los primeros dispositivos móviles que incorporaban una pantalla a LCD (*Liquid Crystal Display*) a color (Figura 9).

Figura 9 - Dispositivos móviles de tercera generación

Igualmente, el usuario pudo asistir al nacimiento de dispositivos que se consideraban como futuristas tales como móviles con cámara fotográfica digital, con posibilidad de grabar videos y mandarlos con un sistema de mensajería instantánea evolucionado, juegos 3d, sonido Mp3 o poder mantener conversaciones por videoconferencia gracias a una tasa de transferencia de datos más que aceptable y a un soporte para Internet correctamente implementado (correo electrónico, descargas, etc.) (Tudela, 2010).

El estándar más utilizado en esta generación fue:

- ✓ UMTS: *Universal Mobile Telecommunications System* - Servicios Universales de Comunicaciones Móviles.

2.3.4. Cuarta generación (4G)

En el año 2010 se lanzaron los primeros servicios 4G basados en la tecnología o estándar LTE en Tokyo, Nagoya y Osaka. Al estar la red 4G basada en el protocolo IP. Esta tecnología puede ser utilizada por módems inalámbricos, *smartphones* (teléfonos inteligentes) y otros dispositivos móviles (Figura 10).

Figura 10 - Dispositivos móviles de cuarta generación (Landa, 2015)

La principal característica de esta red de esta generación es que tiene la capacidad de proveer velocidades de acceso mayores a los 100 Mbps en movimiento y 1 Gbps en reposo manteniendo una calidad de servicio (QoS, *Quality of Service*) de extremo a extremo,

permitiendo ofrecer servicios de cualquier clase en cualquier momento, en cualquier lugar (Tudela, 2010).

2.3.5. Quinta generación (5G)

La siguiente generación de redes de telecomunicaciones (5G) saldrá al mercado en 2020. Más allá de mejorar la velocidad, se espera que las redes 5G liberen un ecosistema masivo de *IoT* (*Internet of Things)* en el que las redes puedan cubrir las necesidades de comunicación de miles de millones de dispositivos conectados, con la combinación adecuada entre velocidad, latencia y costo (Figura 11).

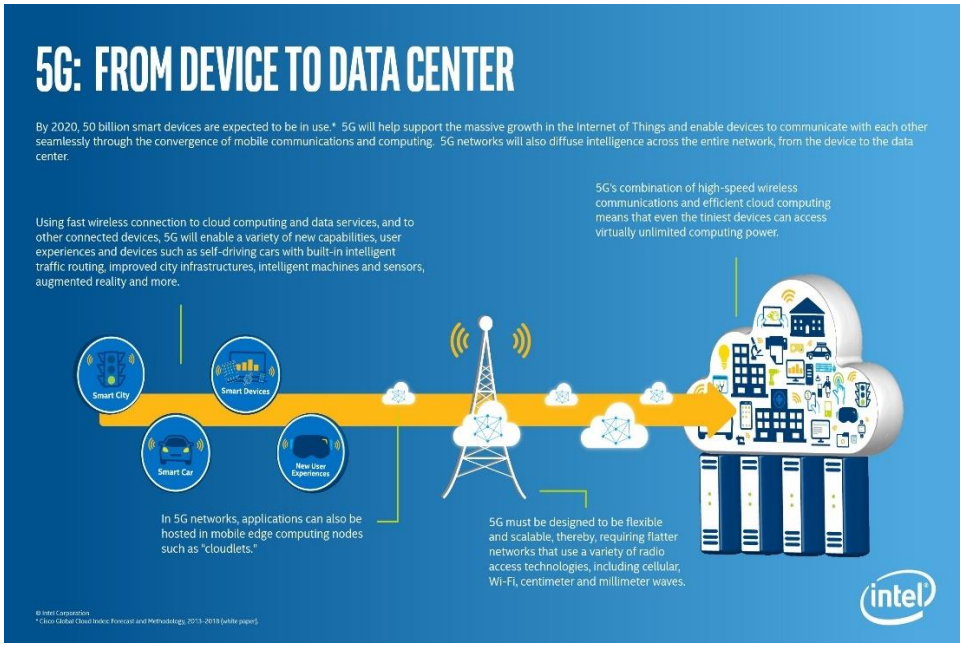

Figura 11 - Estándar 5G aplicado al IoT (Landau, 2015)

La principal evolución en comparación con 4G y 4.5G (LTE avanzado) de hoy en día es que más allá de las mejoras en la velocidad de los datos, los nuevos casos de uso del *IoT* y de comunicación crítica van a requerir nuevos tipos de rendimiento mejorado. Por ejemplo, la "baja latencia" es lo que provee interactividad en tiempo real para los servicios que utilizan la nube: esto es clave para el éxito de los vehículos autónomos, por ejemplo. Además, el bajo consumo de energía es el factor que va a permitir

que los objetos conectados funcionen por meses o años, sin la necesidad de ayuda humana.

2.4. Clasificación de dispositivos móviles

Debido a la dificultad que supone definir qué es un dispositivo móvil, la clasificación que se pueda hacer de los mismos está condicionada a diferentes criterios y a veces no existe un consenso amplio para clasificar un dispositivo móvil en una determinada categoría.

En la década de los 90, tras la aparición de estos primeros dispositivos, establecer clasificaciones más o menos rigurosas era posible debido a que cada aparato estaba claramente definido para una función determinada o para un público concreto. El aumento de las prestaciones y funcionalidades que en la actualidad puede ofrecer cualquier dispositivo móvil dificulta el poder agruparlo dentro de un conjunto determinado.

Por ejemplo, un *smartphone* representa una evolución de un teléfono móvil tradicional, esto es, su cometido es ofrecer comunicación telefónica; sin embargo, cuenta con otros servicios adicionales como la conexión a Internet y aplicaciones, servicios propios de un ordenador, cámara de fotos y de vídeo o la posibilidad de reproducir películas o videojuegos.

Considerando los puntos anteriores, para poder establecer una clasificación se propone utilizar como criterio principal la funcionalidad para la cual se ha diseñado el dispositivo móvil. Sin embargo, la clasificación en una categoría concreta no implica que el dispositivo no pueda ofrecer más funcionalidades o características propias de otras categorías. Así pues, los dispositivos móviles se pueden clasificar en las siguientes categorías (Tudela, 2010):

- ✓ Dispositivos de comunicación:

  Se define como aquel cuya funcionalidad u objetivo principal es ofrecer capacidades de comunicación, principalmente de manera telefónica. Además, ofrecen servicios como el envío de mensajes SMS y MMS. En esta categoría se incluiría el tradicional teléfono móvil como precursor dentro de los dispositivos móviles, y el *smartphone*, el cual amplía considerablemente las prestaciones o funcionalidades del mismo mediante, por ejemplo, la inclusión de pantalla táctil, conexión a internet o ejecución de aplicaciones móviles.

- ✓ Dispositivos de computación:

  Se definen como aquellos que ofrecen amplias capacidades de procesamiento de datos, contando con una pantalla y teclado como accesorios, guardando similitudes con un ordenador de sobremesa. Dentro de este grupo podemos encontrar los siguientes:

  - PDA (*Personal Digital Assistant*):
    Muy populares a finales de los años 90 y que permitían al usuario disponer de una especie de agenda u organizador más completo en relación a lo ofrecido por los teléfonos móviles tradicionales, e incluso la visualización de documentos o acceso a Internet.

  - *Laptop* ó ordenador portátil:
    Son los dispositivos móviles que mayores prestaciones ofrecen, aunque a cambio de mayores tamaños, pesos y precios.

  - Calculadoras gráficas

- ✓ Reproductores multimedia:

  Es aquel diseñado específicamente para proporcionar al usuario la reproducción de uno o varios formatos de datos digitales, ya sean en formato de audio, vídeo o imágenes. En esta categoría encontramos reproductores de MP3, DVD portátiles, los eBooks, etc. Poseen un tamaño reducido y, junto a los teléfonos móviles tradicionales y *smartphones*, son los más extendidos.

- ✓ Grabador multimedia:

  Su función principal es la grabación de datos en un determinado formato digital, principalmente de audio y vídeo. En esta categoría encontramos las cámaras fotográficas digitales o las cámaras de video digital.

- ✓ Consola portátil:

  Se trata de un dispositivo móvil cuya única función es la de proporcionar al usuario una plataforma de juego. A día de hoy representan un amplísimo sector dentro del entretenimiento digital, con un gran volumen de ventas debido a su popularización en la sociedad. Algunos ejemplos de esta categoría son la Nintendo DS de Nintendo, o la PSP de Sony.

Como resumen, en la Figura 12 podemos observar algunos de los ejemplos mencionados de los distintos dispositivos móviles que podemos encontrar en el mercado:

Figura 12 - Ejemplos de dispositivos móviles (Tudela, 2010)

2.5. Tipos de desarrollos de aplicaciones móviles

Teniendo en cuenta el desarrollo de aplicaciones móviles, podemos ver cómo hay una tendencia creciente a pensar que las aplicaciones *web* son el futuro al que se dirige este complejo ecosistema, por ser la más sencilla de cuantas alternativas se proponen. Para ello, existen tanto defensores como detractores del desarrollo de las aplicaciones web, pero, a día de hoy, se pueden considerar diversas soluciones para el desarrollo de aplicaciones móviles que hacen plantear que tal vez aquella más diversificada, como el desarrollo de aplicaciones *web* mencionado anteriormente, no sea la mejor.

Podemos distinguir tres tipos de aplicaciones orientadas a ser utilizadas en dispositivos móviles como *smartphones* y *tablets*, entre otros (Guzmán, 2013):

- ✓ Aplicaciones nativas:

  Se trata de aplicaciones desarrolladas en un lenguaje de programación clásico (*Java/Kotlin* en el caso de Android, y *Objective-C/Swift* en el caso de iOS), con una serie de bibliotecas y *frameworks* de funciones de bajo nivel para cada entorno. Estas aplicaciones se compilan y se instalan en el dispositivo y no necesitan de un navegador para ser ejecutadas. Están diseñadas especialmente para funcionar en dispositivos con un sistema operativo y firmware. Los costes de desarrollo y mantenimiento y sus posibles actualizaciones son más altos en las apps nativas que en otras alternativas. El proceso de conceptualización y desarrollo de una arquitectura a través del análisis del comportamiento de los consumidores permite una experiencia para el usuario mucho mejor; además de un control mayor de las funciones de seguridad que para servicios recurrentes, como la banca móvil, son perfectas. A continuación, se enumeran algunas de sus principales ventajas:

  - Concebidas para su descarga y uso integral en dispositivos móviles, este tipo de aplicaciones permiten una máxima calidad en funciones gráficas, velocidad de ejecución, y acceso a todos los recursos del teléfono (sensores, NFC, GPS, cámara, memoria, etc.)

  - El usuario habitualmente puede acceder a los contenidos de la aplicación, aunque no disponga de conexión a Internet. La interfaz no tiene que cargarse junto con el resto de datos.

  - Se pueden distribuir en los mercados de aplicaciones propios de cada plataforma (AppStore para iOS, y Google Play para Android)

  - Tienen una estructura más segura ya que se evitan las conexiones constantes a la red, propias de las aplicaciones web.

- ✓ Aplicaciones *web* y/o *web* de diseño adaptable (*Responsive Design*):

  Las aplicaciones *web* se ejecutan desde un navegador, aunque en ocasiones este puede estar integrado en la propia aplicación. Se desarrollan en el mismo lenguaje de programación que una *web* como HTML5, JavaScript, CSS3, etc. Algunas de sus principales características son:

  - Se puede acceder a ellas desde cualquier dispositivo móvil con navegador e internet y funcionan en todos ellos.
  - No es necesario instalarlas con carácter previo en el dispositivo.
  - Se pueden integrar fácilmente con los servicios habituales del sitio web, de modo que las actualizaciones y los contenidos por suscripción se vuelven más accesibles y sencillos.

  Las páginas *web* de diseño adaptable (*Responsive Design*) son una solución añadida que se puede clasificar dentro del marco de las apps web. Son sitios webs configurados y diseñados para ser visualizados en cualquier dispositivo. Su configuración se adapta a cualquier pantalla, de modo que los contenidos se agrupan, expanden y se adaptan para una óptima experiencia de usuario. No está claro, de forma general que sean mejores o peores que el desarrollo de las apps nativas. Dependerá del tipo de servicio que se quiera ofrecer, del presupuesto, de la necesidad de calidad gráfica, e incluso de las capacidades técnicas que tengamos accesibles, formación de los programadores, etc.

- ✓ Soluciones hibridas:

  Una app híbrida es una aplicación nativa que para determinados servicios o pantallas se ejecutan en un *WebView*, es decir, en un navegador *web* incrustado en la aplicación. Para el usuario, puede ser casi imperceptible que pantallas son nativas y cuales son accesos a la web.

Dependerá de la consistencia en el diseño de las mismas. Algunas de sus principales características son:

- Se desarrollan en el lenguaje propio de cada entorno, y se incrustan llamadas a páginas *web* las que a su vez se desarrollan en el mismo lenguaje de programación que una *web* clásica (HTML5, JavaScript y CSS3, entre otros).

- Combinan una infraestructura de desarrollo que funciona en todos los dispositivos móviles al mismo tiempo que podemos utilizar todas las tecnologías y ventajas que nos aporta un Smartphone.

- Algunos elementos se descargan directamente de la web.

- Con una app híbrida nos beneficiamos de todas las ventajas de una app nativa y nos aseguramos la madurez del proyecto y el desarrollo de un diseño adaptable a la visualización multipantalla.

Teniendo en cuenta todo lo anterior, ninguna solución es perfecta. A día de hoy se puede considerar dos grandes opciones: HTML5 vs. Apps nativas. El HTML5 se presenta como una fuerte apuesta porque su distribución se realiza a través de "*open web*" de modo que no es necesario el uso de plataformas de distribución de aplicaciones (*app stores*). Además, los desarrolladores juegas un rol crucial; tendrán más libertad ya que actualmente existen limitaciones originadas por las plataformas dominantes del mercado móvil, como en el caso de iOS. Existen muchas opciones de desarrollo de una aplicación y resulta complicado decidir cuál es la solución adecuada. El desarrollo de apps nativas ofrece ventajas adicionales al HTML5 y viceversa. Depende del contexto, de las necesidades del negocio y de la estrategia, entre otros.

Finalmente, en la Figura 13 se muestra un cuadro resumen de la comparativa de los tres tipos de desarrollos de aplicaciones móviles predominantes en la actualidad.

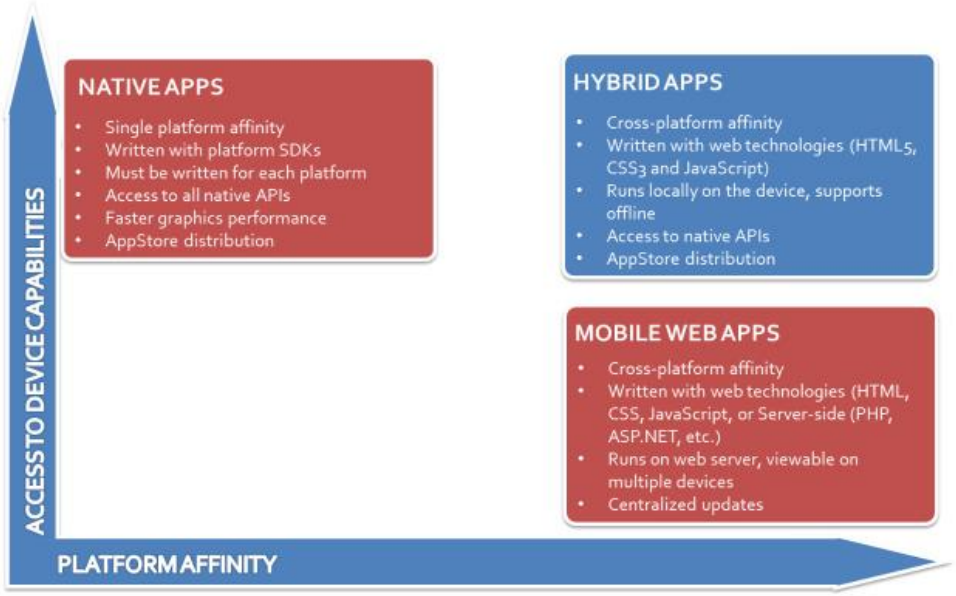

Figura 13 - Cuadro resumen de los diferentes tipos de desarrollo de aplicaciones móviles más utilizados en la actualidad (Gutiérrez, 2013)

2.6. Sistemas operativos para desarrollo nativo de aplicaciones móviles

El sistema operativo destinado a correr en un dispositivo móvil necesita ser fiable y tener una gran estabilidad, ya que incidencias habituales y toleradas en ordenadores personales como reinicios o caídas no tienen cabida en un dispositivo de estas características.

Además, ha de adaptarse adecuadamente a las consabidas limitaciones de memoria y procesamiento de datos, proporcionando una ejecución exacta y excepcionalmente rápida al usuario. Estos sistemas han de estar perfectamente testeados y libres de errores antes de incorporarse definitivamente a la línea de producción. Las posibilidades que existen en un ordenador estándar de realizar actualizaciones e incluso reinstalar mejores versiones del sistema para cubrir fallos o deficiencias son más limitadas en un dispositivo móvil. Es posible incluso que un aparato de esta naturaleza deba estar funcionando ininterrumpidamente durante semanas e incluso meses antes de ser apagado y reiniciado, a diferencia de lo que ocurre con un ordenador personal.

El consumo de energía es otro tema muy delicado: es importante que el sistema operativo haga un uso lo más racional y provechoso posible de la batería, ya que esta es limitada y el usuario siempre exige una mayor autonomía.

Todos estos aspectos de los dispositivos móviles, entre otros muchos, han de ser tenidos en cuenta a la hora de desarrollar un sistema operativo competente en el mercado, atractivo para los fabricantes y que permita al usuario sacar máximo provecho de su terminal (Tudela, 2010).

En base a lo anterior, se presentan los sistemas operativos más utilizados en la actualidad para dispositivos móviles, Android y iOS, con una breve introducción de su historia y evolución, así como sus principales características.

2.6.1. Android

Android es un sistema operativo y una plataforma software, basado en Linux para dispositivos móviles. Además, también usan este sistema operativo (aunque no es muy habitual) *tablets*, *netbooks*, reproductores de música e incluso ordenadores personales. Android permite programar en un entorno de trabajo (*framework*) de Java, aplicaciones sobre una máquina virtual *Dalvik* (una variación de la máquina de Java con compilación en tiempo de ejecución).

Una característica diferenciadora respecto a otros sistemas operativos es que, al igual que Linux, se trata de una plataforma de código abierto, lo que permite a fabricantes, operadores y desarrolladores dar mayor funcionalidad a su smartphone. Además, Android es un sistema gratuito y multiplataforma; por multiplataforma entendemos que el sistema operativo puede ser usado en distintas plataformas, y por plataforma entendemos que es una combinación de hardware y software usada para ejecutar aplicaciones (en su forma más simple consiste únicamente de un sistema operativo, una arquitectura, o una combinación de ambos). Android es gratuito al poder ir instalado sin ningún coste en cualquier dispositivo móvil (Manuel Báez).

Todo esto hace de Android ser líder indiscutible en el mercado de smartphones a nivel mundial, con un nivel de cuota de mercado del 87,5 %. A esto último habría que sumar sus más de 2000 millones de personas en todo el mundo que utilizan este sistema operativo de Google, así como

los 1,5 millones de activaciones de dispositivos Android diarias (Smith, 2017).

Aunque Android es el sistema operativo más utilizado en dispositivos móviles a día de hoy, lo cierto es que no siempre fue así. La compañía lanzó su primera versión de Android en 2003, y desde entonces han estado luchando por conseguir el mayor número de usuarios posibles. A continuación, se realiza un repaso por la historia y evolución de Android, a través de sus diferentes versiones (Stelapps, 2017).

2.6.2. Historia y evolución

Android Inc. fue fundada en octubre de 2003 por Andy Rubin, Rich Miner, Nick Sears y Chris White. Sus intenciones iniciales eran las de desarrollar un sistema operativo inteligente, que tuviese en cuenta la localización y los gustos de su propietario y actuase en consecuencia.

Al principio, este sistema operativo estaba enfocado a cámaras de fotografía digitales, pero al darse cuenta de que el mercado de los dispositivos móviles no estaba tan explotado como podría ser, decidieron dividir los esfuerzos y producir también un sistema operativo para móviles que compitiese con Windows Mobile y Symbian, muy fuertes por aquel entonces.

En 2005, tras dos años de trabajo, Google se interesó por Android Inc. y su trabajo y decidió adquirir la compañía, para así poder tomar partido en el desarrollo y la toma de decisiones relacionadas con el sistema operativo que estaban desarrollando. Andy Rubin, Rich Miner y Chris White decidieron seguir con el proyecto, además de algunos trabajadores que tenía la compañía, y pasaron a trabajar para Google.

Lo cierto es que poco se sabía por aquel entonces de lo que Google y Android Inc. estaban desarrollando, aunque debido a la naturaleza de Android Inc., se comenzó a especular que Google podría sacar al mercado un sistema operativo para móviles en un futuro próximo.

La salida al mercado del iPhone, por parte de Apple en el año 2007, fue un toque de atención para Google. En ese momento, contaban con un prototipo de sistema operativo que funcionaría en un móvil con un teclado QWERTY, muy similar a una BlackBerry.

Comenzó entonces una carrera a contrarreloj para cambiar parte del código del sistema operativo y poder adaptarlo a un móvil con pantalla táctil, para así poder competir con Apple, el cual había ganado muchísima popularidad con el iPhone.

Un poco más adelante ese mismo año, la *Open Handset Alliance*, formada por compañías como Google, Sony, HTC, Samsung, T-Mobile, Sprint Nextel, Texas Instruments o Qualcomm anunciaron que estaban trabajando en un mismo objetivo: un sistema operativo abierto para una plataforma móvil.

Fue entonces, en ese mismo momento, en el que Android presentó su primero producto (en una primera fase *Alpha* de desarrollo): un sistema operativo bajo su nombre basado en Linux 2.6, conocido la versión de Android 1.0 Apple Pie.

Desde esa primera versión hasta la actualidad, se describe la evolución de las versiones de Android junto con sus principales características (Stelapps, 2017):

- ✓ Android 1.0 Apple Pie (2008)

Android 1.0 Apple Pie era una versión muy sencilla de Android que no preocupó en exceso a sus competidores en aquel momento, aunque lo cierto es que incluí-a algunas de las características que aún se utilizan en versiones de Android posteriores y que son características de este sistema operativo como:

- Un menú desplegable para las notificaciones.
- La posibilidad de contar con widgets (accesos directos) en el escritorio.
- Integración con Google Mail (Gmail), Google Contacts y Google Calendar.
- Navegador, Google Maps, Google Talk, soporte para cámaras y reproductor de Youtube.
- Un mercado de aplicaciones conocido como Android Market, con un catálogo de aplicaciones completamente gratuitas.

✓ Android 1.1 Banana Bread (2009)

En febrero de 2009, poco después del primer lanzamiento de Android al mercado, Google nos sorprendía con el lanzamiento de Android 1.1 Banana Bread. Esta versión de Android apenas introdujo cambios, y fue más bien una versión dedicada a arreglar fallos y *bugs* de Android 1.0 Apple Pie. Como gran característica que se añadió en Android 1.1 Banana Bread fue algo que potenció el sistema operativo: las actualizaciones automáticas.

✓ Android 1.5 Cupcake (2009)

Para dar salida a esta nueva herramienta, Google lanza en abril de 2009 su siguiente versión: Android 1.5 Cupcake. A esta altura fue cuando comenzó a demostrarse el hecho de que Google iba a utilizar nombres de postres para sus versiones de Android, además de que la salida sería en orden alfabético.

En Android 1.5 Cupcake nos encontramos con algunas novedades muy utilizadas aún a día de hoy:

- Teclado táctil desplegable QWERTY con predicción de escritura en la pantalla.
- La introducción de un widget de escritorio de Google para realizar búsquedas rápidas por Internet.
- Bluetooth.
- Una interfaz mejorada para grabar y reproducir vídeos.
- SDK, el cual permitió el desarrollo de aplicaciones por parte de terceros.

✓ Android 1.6 Donut (2009)

En septiembre de 2009 aparece Android 1.6 Donut, una pequeña actualización que incluía un buen número de mejoras en la interfaz Android. Pero, además, Android 1.6 Donut realizó cambios notorios en el núcleo del sistema operativo:

- Se añadió un soporte para CDMA/EVDO, VPN y 802.1x, el cual amplió el alcance de mercado de Android.

- Se modificó el sistema operativo para añadir compatibilidad con diferentes resoluciones de pantallas.
- Se añadió un motor de búsqueda de Internet y en el dispositivo integrado.
- Se re-diseñó la interfaz de la aplicación de la cámara.
- Nuevo diseño de Android Market.

Con el soporte de Android 1.6 Donut, salió al mercado el Motorola Droid, pensado para competir con el iPhone. Su éxito fue rotundo, y en una semana se vendieron más de 250.000 terminales en Estados Unidos. Aquí es cuando la popularidad de Android como sistema operativo para dispositivos móviles se hizo patente.

✓ Android 2.x Eclair (2009)

Siguiendo con el ritmo acelerado de Android, en noviembre de 2009 salió a la luz Android 2.x Eclair (versiones 2.0 y 2.1, siendo esta última la más popular), pensada y adaptada para terminales con un tamaño un poco mayor. Además, se introdujeron algunos cambios interesantes en la forma de funcionamiento del sistema operativo:

- Soporte para multicuentas de usuario en un mismo terminal.
- GPS gratuito a través de Google Maps Navigation.
- Navegador de Internet actualizado con soporte para HTML5.
- Introducción de la función “Text to Speech”.
- Zoom digital en la cámara.
- Pantalla de desbloqueo nueva.

✓ Android 2.2 Froyo (2010)

En mayo de 2010 veía la luz la siguiente versión del sistema operativo: Android 2.2 Froyo. El primer teléfono en recibir la actualización fue el HTC Nexus One, dejando claro que Google iba a dar prioridad a sus terminales Nexus en las actualizaciones de software. Android 2.2 Froyo incluía algunos cambios interesantes:

- Pantalla de inicio completamente rediseñada.
- Galerí-a de imágenes nueva.
- La opción de hacer *tethering*.
- Pantalla de desbloqueo mediante código PIN.
- Grabación de ví-deo en calidad 720p.
- Mejora en la velocidad del sistema operativo.

✓ Android 2.3 Gingerbread (2010)

En diciembre de 2010 Google lanzó la última actualización del año: Android 2.3 Gingerbread. Esta versión de Android supuso una revolución que la llevó a ser la versión de Android más utilizada en el mundo. Los cambios de Android 2.3 Gingerbread fueron los siguientes:

- Nuevo diseño estético más moderno.
- Compatibilidad con pantallas de mayor tamaño y resoluciones más altas.
- Soporte para conexión NFC.
- Teclado mejorado.
- Soporte para cámaras de fotografía frontales.
- Sustitución de sistema de archivos YAFFS por el sistema ext4.

✓ Android 3.x Honeycomb (2011)

A estas alturas, Android ya era un sistema operativo para móviles ampliamente reconocido, y con la mejora y ampliación del mercado de *tablets*, se sacó una versión específica para este tipo de terminal: Android 3.x Honeycomb. Esta versión simplemente adaptó la interfaz de los teléfonos móviles a las *tablets*, y solamente era compatible con *tablets*. Posteriormente, se actualizaron a Android 3.1 Honeycomb y Android 3.2 Honeycomb, con mejoras en fallos y *bugs*.

En el CES 2011 se presentó la tablet Motorola Xoom, siendo esta la primera tablet equipada con Android 3.x Honeycomb, considerada como una de las mejores tablets del momento.

✓ Android 4.1 Jelly Bean (2012)

Llegados a este punto, el lanzamiento de versiones de este sistema operativo es más pausado, y en 2012 sólo sale a la luz Android 4.1 Jelly Bean. Eso sí, los cambios que introdujo en el sistema operativo y su flexibilidad le han colocado como una de las versiones de Android más utilizada del momento. Entre sus características más notables están:

- Introducción de Google Now, el asistente de voz de Google.
- Introducción de navegador Google Chrome.
- Mejora del rendimiento del sistema.
- Búsqueda mediante el uso de la voz mejorada.
- Mejora de las notificaciones interactivas del escritorio.
- Re-ajuste del tamaño de los widgets.
- Mejora en la predicción de escritura.
- Posibilidad de dictado de voz.

✓ Android 4.2 Jelly Bean (2012)

En noviembre de 2012 se actualiza la versión de Android Jelly Bean de la 4.1 a la 4.2, introduciendo mejoras en el rendimiento y pequeños cambios en la interfaz como los siguientes:

- Nuevo panel de control.
- Posibilidad de acceso a widgets desde la pantalla de bloqueo.
- Soporte para *streaming* de vídeo y audio.
- Soporte para más de un usuario en el mismo terminal.
- Captura de fotografías de 360º.
- Introducción de Gesture Mode para personas invidentes.
- Re-diseño del reloj y de sus widgets.

✓ Android 4.3 Jelly Bean (2013)

A mediados de año, en Julio de 2013, Google anuncia que sacará al mercado una última versión de Android 4.3 Jelly Bean, pensada para dar soporte a juegos y corregir algunos fallos. Lo cierto es que, pese a

no introducir demasiados cambios, su uso está muy extendido. Como principales novedades se destacan:

- Mejora en el soporte multiusuario y multiperfil.
- Compatible con TRIM.
- Bluetooth Smart.
- Introducción de Google Games.
- Localización de WiFi mejorada.
- Soporte para OpenGL ES 3.0.

✓ Android 4.4 KitKat (2013)

En septiembre de 2013 se anunció la muy esperada versión de Android 4.4 KitKat. La parte positiva de Android 4.4 KitKat es que está pensada para ser la versión unificadora de Android. Como principales novedades se destacan:

- Compatibilidad con terminales desde 512 MB de memoria RAM.
- Reducción del consumo de batería.
- Reducción de requisitos de hardware para funcionar.

✓ Android 5.0 Lollipop (2014)

Tras la pequeña revolución que supuso Android Kit Kat llega Android Lollipop (Android L), la nueva versión del sistema operativo para smartphones más popular que copa el mercado y que es ya el referente para decenas de fabricantes. Android tiene unas exigencias enormes para seguir liderando la tabla de sistemas operativos móviles y sin duda es imprescindible reinventarse cada cierto tiempo, si con Android Kit Kat Google ya nos demostró que había empezado un nuevo camino, con Android L no hizo más que mejorar a su predecesor de forma notable en lo siguiente:

- Nuevo diseño tanto a nivel de sistema operativo como de aplicaciones mediante la creación y unificación de normas de diseño para todos los desarrolladores bajo el nombre de *Material Design*.

- Mejoras de las notificaciones pudiendo ser consultadas desde la pantalla de bloqueo de forma individual, así como su interacción con las mismas.
- Mejoras de rendimiento y en fluidez gracias a *Project Volta*, la sustitución de la máquina virtual de Java *Dalvik* por ART (*Android Runtime*) y una nueva y rediseñada multitarea.
- Soporte total a sistemas de 64 bits además de incluir las librerías OpenGL ES 3.1, dando soporte para poder utilizarse como plataforma de juegos.
- Integración con los dispositivos *weareables* (*smartwatches*) mediante el sistema operativo Android adaptado a este tipo de dispositivos llamado Android Wear.

✓ Android 6.0 Marshmallow (2015)

El nombre de esta nueva versión es Android Marshmallow (Android M), y trae consigo cambios relevantes en experiencia de usuario y en la gestión de batería, pero manteniendo el diseño de la versión Android Lollipop. El cambio en cuanto a diseño en la versión Lollipop fue bastante significativo y alabado por la crítica. Por ello, Google decidió mantener prácticamente el mismo diseño para centrarse en solucionar otros problemas de esta última versión y añadir características nuevas de gran interés. A continuación, se enumeran las principales:

- Renovación de la pantalla de bloqueo, haciendo más accesible los comandos de voz de Google.
- Nuevo cajón de aplicaciones vertical con buscador.
- Nuevo control de volumen con modo silencio.
- Control y administración de los permisos individuales de cada aplicación para garantizar la seguridad del usuario.
- Facilidad para el establecimiento de aplicaciones predeterminadas.
- Optimización en el consumo de batería mediante el sistema *Doze*.
- Soporte para memorias USB.

✓ Android 7.0 Nougat (2016)

Android 7.0 Nougat se presenta como una de las actualizaciones más importantes hasta la fecha que recibe el sistema operativo de Google, con más de 250 mejoras que harán al sistema más personalizable, más productivo, más seguro, más potente y más eficiente. A continuación, se destacan sus principales novedades:

- Pantalla divida, permitiendo la ejecución de dos aplicaciones a la vez
- Cambio rápido entre aplicaciones
- Notificaciones agrupadas con mayor interacción y la opción de respuesta rápida desde las mismas.
- Control más individualizado de las notificaciones
- Incorporación de ajustes rápidos en la barra de notificaciones
- Mejora del Sistema *Doze*, obteniendo incluso un ahorro de batería con el móvil en movimiento
- *Project Svelte,* para optimización del consumo de memoria RAM por parte de las aplicaciones en segundo plano
- *Direct Boot*, para conseguir que los dispositivos se inicien más rápido
- Mejora en el tiempo de instalación de las aplicaciones con ART
- Modo economizador en el consumo de datos móviles
- Soporte multilocal de idiomas
- Soporte para desarrollo de aplicaciones con la última versión de Java, Java 8
- Soporte para la nueva API de bajo nivel para gráficos de última generación

✓ Android 8.0 Oreo (fecha apróx. para finales de Agosto del 2017)

Aunque todavía a fecha de escritura del TFM no ha sido lanzado públicamente, Android 8.0 Oreo será la nueva versión del sistema operativo en los próximos meses. Entre sus principales novedades destacan (Garzón, 2017):

- *Project Treble*: Google pretende dividir el sistema operativo en módulos para reducir el tiempo que les toma a los fabricantes actualizar sus dispositivos
- Límite de apps en segundo plano
- Mejores notificaciones
- *Instant apps*, permitiendo al usuario de disfrutar de una versión simple de la aplicación sin tener que instalarla

A modo de resumen visual en la siguiente Fig., se observa un timeline de las diferentes versiones de Android mencionadas anteriormente, a excepción de Android 8.0 Oreo aún por salir.

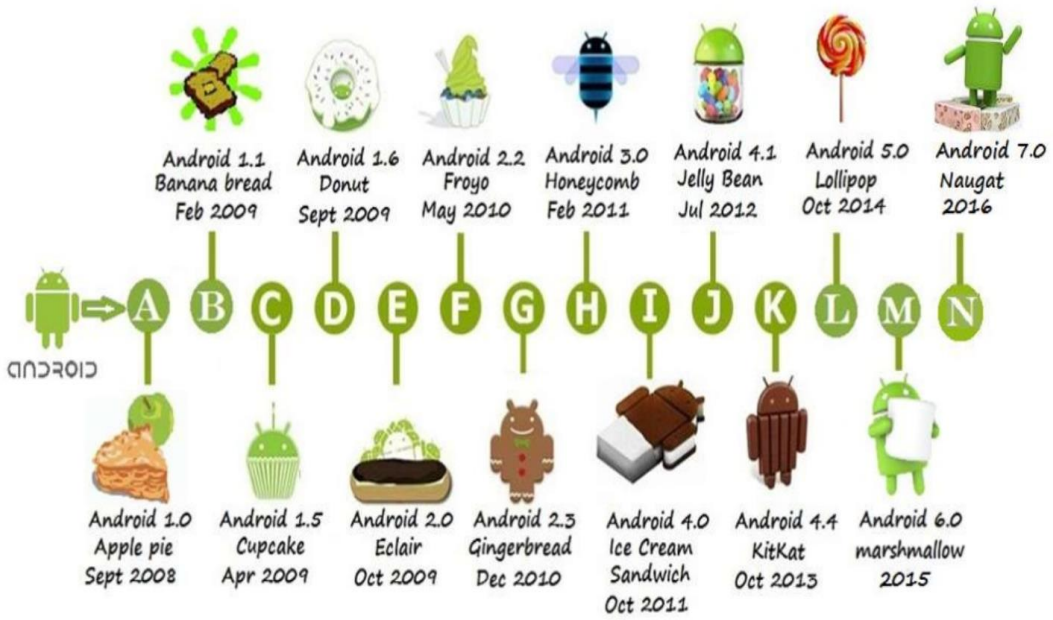

Figura 14 - Evolución de las diferentes versiones del sistema operativo Android (SlideShare, 2016)

2.6.2.1. Cuota de mercado

Android se consolida como líder indiscutible en el mercado de smartphones a nivel mundial, con un nivel de cuota de mercado del 87,5 %. A esto último habría que sumar sus más de 2000 millones de personas en todo el mundo que utilizan este sistema operativo de Google, así como los 1,5 millones de activaciones de dispositivos Android diarias (Smith, 2017).

En cuanto a sus versiones, en las Figuras, se puede observar que la versión más presente en los dispositivos móviles con Android es Marshmallow, con apróx. un 32%., seguida muy cerca de Lollipop con un 28% apróx (Android Developer, 2017).

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.7%
4.1.x	Jelly Bean	16	2.7%
4.2.x		17	3.8%
4.3		18	1.1%
4.4	KitKat	19	16.0%
5.0	Lollipop	21	7.4%
5.1		22	21.8%
6.0	Marshmallow	23	32.3%
7.0	Nougat	24	12.3%
7.1		25	1.2%

Datos recopilados durante un período de 7 días hasta 8/8/2017.
No se muestran versiones con una distribución inferior al 0,1%.

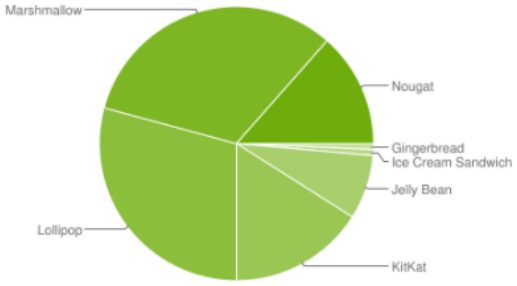

Figura 15 - Cuota de mercado de las versiones de Android (Android Developer, 2017)

2.6.2.2. Ventajas y desventajas

Android es uno de los mejores y más extendidos sistemas operativos para móviles a nivel mundial, pero como cualquiera de ellos, posee una serie de ventajas y desventajas que afectan al usuario a la hora de su elección, como las siguientes (BeMovil, 2016):

- ✓ Puntos a favor

  - **Código abierto**: Gracias al *open source* o código abierto, existen hoy en día más de 100.000 aplicaciones disponibles para teléfonos Android.

- **Multitarea**: Los sistemas Android son capaces de hacer funcionar a la vez varias aplicaciones, gestionándolas, dejándolas en modo suspensión si no se utilizan e incluso cerrándolas si llevan un periodo determinado de inactividad para evitar un consumo tan excesivo de batería.

- **Interfaz atractiva para el usuario**: Google Play Store existen diversos *launchers* cuyo objetivo es sustituir la apariencia por defecto que muestran las distintas versiones de Android por una más atractiva con las funcionalidades y opciones más personalizables para el usuario. Los múltiples widgets contribuyen a ampliar la comodidad del usuario.

- El **sistema de notificaciones de Android** es sensiblemente superior al sistema de iOS, ya que es mucho más organizado, veloz y simple.

- Una **gama más amplia de precios** para escoger: Aunque el lanzamiento del iPhone SE ha acercado a los dispositivos de Apple a la gama media, los terminales Android permiten escoger entre un rango de precios más amplio y adaptado a los diferentes bolsillos. En torno a los 100 euros se pueden encontrar móviles de gama media-baja.
- **Transmisión de archivos por NFC**: Traspasar archivos de un teléfono a otro con tecnología NFC es actualmente en Android, una acción cómoda y rápida, cuya ejecución se realiza aproximadamente en un minuto. Los iPhone integran esta tecnología, pero con el objetivo de ejecutar las opciones de Apple Pay, que todavía no se encuentra disponible en todos los países, aunque si está experimentando un gran proceso de crecimiento.

- **Amplia disponibilidad para una enorme cantidad de equipos de diferentes marcas**, mientras que iOS, sólo se puede utilizar en equipos de la marca Apple.

- El **sistema de sincronización de Android en la nube** demuestra una elevada eficiencia, ya que, al estar ligado a Google, los servicios web cuentan con un gran sistema de trabajo.

✓ Puntos a mejorar

- **La fragmentación**: Android está totalmente fragmentado provocando problemas de incompatibilidad con algunas aplicaciones de las tiendas de aplicaciones que funcionan en determinadas versiones de Android. Además, a la hora de adaptar los fabricantes sus versiones personalizadas del sistema operativo, se encuentras con muchas dificultades para su implementación, teniendo que emplear una gran cantidad de tiempo y recursos debido al gran número de marcas, tipos de dispositivos y versiones en uso de Android.

  Este es el principal problema de Android y que se está tratando de solventar en las versiones más actuales con una modularización del sistema operativo para facilitar su implementación y personalización a los fabricantes de dispositivos, así como una mayor unificación a la hora de establecer la versión mínima del sistema operativo con la cual pueden funcionar la mayor parte de las aplicaciones.

- El hecho de tener varias aplicaciones abiertas hace que el **consumo de la batería** aumente.

- La duración de la batería es sensiblemente menor que los dispositivos iOS en general.

2.6.3. iOS

IOS es un sistema operativo para dispositivos móviles desarrollado por la empresa *Apple* y que se distribuye ya instalado y de forma exclusiva en los dispositivos comercializados por esta empresa (además del iPod Touch, iPad y el Apple TV), no permitiéndose la instalación de este sistema operativo en ningún otro tipo de dispositivo (considerando su software de código cerrado).

El término IOS no lo adoptó *Apple* hasta la versión 4 de este sistema operativo, en junio de 2010, anteriormente se denominaba oficialmente *iPhone OS* (Varilas, 2016).

En cuanto a su arquitectura, iOS deriva de OS X (renombrado como MacOS en la WWDC 2016). A su vez OS X está basado en Darwin BSD, por lo que podemos decir que iOS es un sistema operativo tipo Unix.

Podemos dividir al sistema en cuatro capas de abstracción: el núcleo del sistema operativo, la capa de Servicios, la capa de Medios y la capa de Cocoa Touch. iOS tiene mucho en común con OS X siendo la última capa (Cocoa Touch) la que más diferencias tiene pues OS X está pensado para usarse mediante ratón y teclado (Cocoa) y iOS está pensado para manejarse mediante una interfaz táctil (Cocoa Touch).

Así pues, la arquitectura iOS está basada en capas (Figura 16), donde las capas más altas contienen los servicios y tecnologías más importantes para el desarrollo de aplicaciones, y las capas más bajas controlan los servicios básicos (Varilas, 2016).

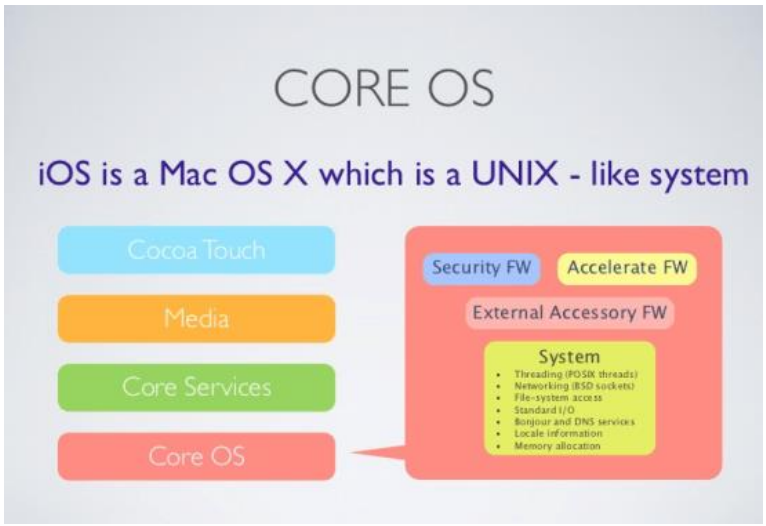

Figura 16 - Arquitectura de capas del sistema operativo iOS (Varilas, 2016)

2.6.3.1. Historia y evolución

El sistema operativo IOS vio la luz en junio del año 2007 con la salida al mercado del primer teléfono móvil de *Apple*, el *iPhone,* un teléfono inteligente o *smartphone* en inglés, que es un teléfono con capacidades de computadora de bolsillo ya que permite la instalación de aplicaciones informáticas. Este primer *iPhone* supuso toda una revolución en el mundo de la telefonía móvil ya que fue el primer teléfono móvil con pantalla *multitouch,* que permite al usuario utilizar varios puntos para interactuar con la pantalla y no únicamente un punto como hasta ese momento. También creo tendencia en cuanto a usabilidad y mejora de la experiencia del usuario en el uso de teléfonos móviles inteligentes.

En ese momento las únicas aplicaciones que se podían tener eran las típicas que ahora vienen preinstaladas Mapas, Reloj, Calculadora, Fotos, iPod, … No fue hasta la segunda versión (iPhone OS 2.0) cuando por fin se pudieron instalar aplicaciones de terceros y cuando vimos el nacimiento de la AppStore (Varilas, 2016).

Posteriormente *Apple* incluyó este mismo sistema operativo en sus dispositivos *iPod Touch*, un ordenador de bolsillo sin capacidad de teléfono que fue lanzado al mercado en septiembre de 2007. En enero de 2010 también se incluyó este sistema operativo en los dispositivos iPad, la tableta de Apple y en los iPad Mini, una tableta con una pantalla más reducida, a finales del año 2012.

De esta manera, El iPhone ha tenido una evolución espectacular en los últimos 8 años, pero en parte se debe también a iOS, que no ha parado de crecer y cambiar hasta la actualidad, con servicios tan comunes como Siri, FaceTime, la Apple Store, iCloud…

A continuación, se describe la evolución de las versiones de iOS, desde esa primera versión en 2007 hasta la actualidad, junto con sus principales características (Rus, 2016) :

- ✓ iPhone OS 1 (2007)

El primer sistema operativo móvil de Apple se presentó el 9 de enero de 2007, fecha en la que Steve Jobs mostró al mundo el iPhone. Sus principales y novedosas características eran:

- Gestos *multi-touch*,
- Apuesta por una interfaz con interacción plenamente visual
- Búsqueda en Internet con Safari y una app para YouTube.

Esta primera versión recibió una importante actualización en enero de 2008, con la posibilidad de "personalizar" el sistema moviendo las apps en pantallas diferentes. Además, se ofrecía a los usuarios de un iPod touch nuevas apps como Mail, Mapas, Notas, Bolsa y Tiempo. Como curiosidad, la actualización era gratuita, pero si disponías de un iPod Touch resultaba ser de pago.

- ✓ iPhone OS 2 (2008)

Después del lanzamiento del SDK (*Software Development* Kit) del iPhone para desarrolladores, en marzo de 2008, Apple nombró a su sistema operativo iPhone OS. La segunda versión, iPhone Os 2, venía preinstalada en los iPhone 3G y traía lo que para muchos ha sido el motor de iOS: la *App Store*.

Por otro lado, pudimos ver cómo los Mapas por fin recibían compatibilidad con el GPS y el Mail actualizaciones *push*. Una vez más, la actualización era gratuita para los usuarios del iPhone, al contrario que para los usuarios del iPod Touch (eso sí, las futuras actualizaciones menores serían gratuitas).

- ✓ iPhone OS 3 (2009)

iPhone OS 3 fue la tercera versión de iOS y se lanzó durante la WWDC de 2009. Las novedades más importantes fueron Control por voz, mensajes multimedia, búsqueda con *Spootlight*, un modo horizontal y funciones de copiar, cortar y pegar (no existían hasta entonces).

En marzo de 2010, cuando se presentó el primer iPad, Apple decidió dejar atrás el nombre iPhone OS para pasar a iOS. Una

vez más, los usuarios de iPod Touch tuvieron que pagar para poder actualizar, pero fue la última versión de iOS en ser de pago.

- ✓ iOS 4 (2010)

iOS 4, la versión de iOS que venía preinstalada no solo en el iPhone 4 sino también en el iPad 2. Ha sido uno de los saltos importantes de iOS, pues por fin se introdujeron los fondos de pantalla, la multitarea, las carpetas de apps, FaceTime y iBooks para iPad.

iOS 4 era una muestra de madurez del sistema, la multitarea ofrecía a los usuarios muchas más posibilidades, y el iPhone 4 así como el iPad 2 tenían la potencia suficiente para ejecutar muchas tareas a la vez. Una vez más, todo está sincronizado entre hardware y software.

- ✓ iOS 5 (2011)

Junto al iPhone 4s Apple mostró iOS 5, que trajo como protagonista a Siri, el asistente virtual que ha marcado tendencia durante estos últimos cinco años. También tuvimos por primera vez un Centro de Notificaciones, junto a las aplicaciones preinstaladas como iMessage y Recordatorios.

También fue importante iOS 5 por ser la primera versión compatible con las actualizaciones OTA. Es decir, a partir de esta versión se podía actualizar el sistema sin disponer de un cable y un ordenador con iTunes, directamente desde Internet *over-the-air*. iCloud y la integración con Twitter son otras dos novedades importantes.

- ✓ iOS 6 (2012)

El iPhone 5 y el iPad mini fueron los dispositivos que introdujeron a iOS 6. Pero eso no fue lo más destacable de iOS 6, sino la eliminación de los servicios de Google. Ahora YouTube ya no venía preinstalado y sobre todo los mapas ya no provenían de Google Maps, sino de un nada maduro servicio de mapas creado por Apple.

iOS 6 mantenía el skeumorfismo como estilo de diseño en un 2012 que pedía a gritos un cambio de estilo adaptado a los tiempos actuales. Por otro lado, vimos la aparición de Passbook, soporte para LTE e integración con Facebook. No fue suficiente, el protagonismo (para mal) fue de Apple Maps.

- ✓ iOS 7 (2013)

Con el desastre de Apple Maps rodaron cabezas en el equipo directivo de Apple, entre ellas la de Scott Forestall, que fue sustituido por Jonathan Ive como *Senior Vice President* de Diseño. En consecuencia, vimos el cambio más radical de todos en iOS con el abandono del skeumorfismo a favor de un colorido sistema basado en las transparencias y los degradados.

El iPhone 5s, el iPhone 5c, el iPad Air y el iPad mini 2 son los que venían equipados con iOS 7. Este nuevo sistema operativo basado en las transparencias introdujo el Centro de Control, AirDrop, una nueva app de fotos, iTunes Radio y CarPlay entre otras aplicaciones.

- ✓ iOS 8 (2014)

Si en algo se centró iOS 8 fue en arreglar algunos errores notables de iOS 7 y en añadir mejoras para que fuese un sistema no tan limitado. Las extensiones y la posibilidad de conectar apps entre sí mediante el menú compartir hicieron que iOS 8 fuese el motivo para que la gente dejase de lado el ordenador a la hora de trabajar y coger un iPad.

También pudimos ver cómo llegaron Apple Pay, Salud, HandOff, QuickType, Compartir en Familia, iCloud Drive, teclados de terceros y Apple Music. En definitiva, una actualización del sistema que buscaba corregir los errores de iOS 7 y sobre todo abrir el sistema para que los desarrolladores aprovechasen todas las posibilidades.

- ✓ iOS 9 (2015)

En iOS 9 se introdujo todo un cambio en el uso del sistema gracias a la compatibilidad con el 3D Touch del iPhone 6s y el iPhone 6s Plus. Fue el momento de actualizar Notas con nuevas funciones, sustituir Newsstand por News, Añadir mejoras en Passbook que pasó a llamarse Wallet y mejorar Mapas con direcciones y tráfico.

Por otro lado, con el lanzamiento del iPad Pro se aprovechó para añadir Split View, Picture-in-Picture y atajos en teclados de terceros. Además, el sistema ahora podía aprovechar mejor la autonomía del dispositivo gracias al Modo ahorro de batería que limita las conexiones y características del sistema para no consumir tanta energía.

- ✓ iOS 10 (2016)

Si por algo se caracteriza la décima versión del sistema operativo móvil de Apple es por abrirse a los desarrolladores. Por fin estos tienen acceso a Siri, al 3D Touch y a apps directas del sistema como Teléfono. Esto hace que se pueda, por ejemplo, enviar mensajes de WhatsApp con Siri o recibir las llamadas de Skype directamente en la app Teléfono.

El centro de widgets ahora es más intuitivo y la pantalla de bloqueo más útil. Además de que las notificaciones ahora están enriquecidas para realizar más acciones directamente desde ellas. Como es habitual, se sigue viendo mejoras en Mapas, Música y otras apps, pero sobre todo ha sido importante el cambio de Mensajes,

que se ha vitaminado gracias a la llegada de los *stickers*, las apps de terceros y nuevos efectos al enviar los mensajes.

Por último, en iOS 10 se da la posibilidad de desinstalar las apps de Apple.

- ✓ iOS 11 (2017)

Apple ha presentado una nueva versión de iOS durante la WWDC 2017, algo que ya es tradición en su conferencia anual para desarrolladores. Esta vez, el sistema operativo está lleno de pequeñas mejoras, incluyendo un nuevo centro de control, una renovada Siri y soporte para realidad aumentada.

Como mejoras menores cabe destacar las siguientes: soporte nativo a códigos QR (en China y otros territorios donde son muy utilizados), renovada aplicación de podcasts, sincronización inmediata de iMessage, el envío de dinero usando Apple Pay a través de la app de Mensajes, nuevo AirPlay 2, nuevo HomeKit 2 y un rediseño completo de la App Store.

2.6.3.2. Cuota de mercado

iOS se posiciona en la segunda posición en el mercado de smartphones a nivel mundial, con un nivel de cuota de mercado del 12,1 %, bastante lejos de la posición predominante que tiene Android como líder del mercado (Apple Developer, 2017).

En cuanto a sus versiones, en la Figura 17 se puede observar que la versión más presente en los dispositivos móviles con iOS es iOS 10, con un 87% (Apple Developer, 2017).

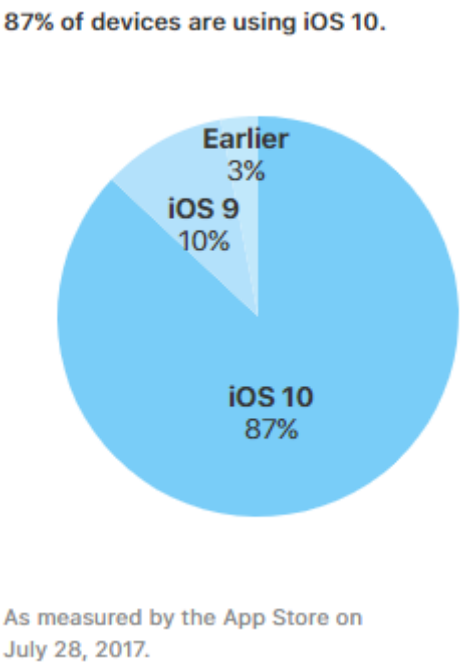

Figura 17 - Cuota de mercado de las versiones de iOS (Apple Developer, 2017)

2.6.3.3. Ventajas y desventajas

iOS, estandarte del gigante tecnológico Apple, es conocido por todos es conocida la calidad de sus materiales, la sofisticación de sus dispositivos o las alabanzas hacia su intuitivo y limpio diseño. Sin embargo, al igual que Android, posee una serie de ventajas y desventajas que afectan al usuario a la hora de su elección, como las siguientes (BeMovil, 2016):

- ✓ Puntos a favor

  - **Elevada seguridad**: El sistema operativo iOS siempre ha presumido de su eficiencia frente a las amenazas cibernéticas externas. Además, con la última actualización, el código de seguridad con el sistema típico de 4 dígitos se ha actualizado a 6 dígitos.

  - **Mayor filtro y exclusividad dentro del mercado de aplicaciones**: Todas las apps para iOS atraviesan un proceso de revisión manual por parte del equipo de

Apple. Este filtro permite un mercado de aplicaciones más cuidado y con un mayor índice de calidad que en la competencia. Las encontrarás personalizadas para cada idioma y país. Además, iOS permite tener un mayor control sobre las aplicaciones, ya que cada vez que una app quiere hacer algo, tiene que pedirte permiso para ello.

- **Interfaz intuitiva:** Los dispositivos de iOS focalizan todos sus elementos en la mejora de la experiencia de usuario, con unas opciones muy sencillas de configuración, pocos botones y un menú de navegación que se caracteriza por la intuición.

- El **asistente personal SIRI** se caracteriza por su gran capacidad para la resolución de tareas cotidianas con apoyo virtual. Con la última actualización del sistema se ha vuelto un 40% más rápido y preciso.

- **Integración entre software y hardware**: Debido a que Apple fabrica y diseña tanto el hardware como el software de sus dispositivos, este hecho les dota de una optimización y gran fiabilidad en su funcionamiento que les hace tener un valor diferenciador frente al resto. Además, a pesar de tener un modelo antiguo, Apple permite la compatibilidad con sistemas operativos anteriores, por lo que se puede disfrutar de las novedades más recientes creadas para todos sus dispositivos.

- Elaboración de todos sus terminales con algunos de los **mejores materiales del mercado**.

- **Elevada duración de la batería**, visiblemente superior a los dispositivos Android. Esto les proporciona una gran autonomía.

- La **sincronización entre los diversos dispositivos de Apple** ha sido, desde su creación, una de las principales

banderas de la marca. Servicios como iTunes, iTouch o iCloud sirven para sincronizar de forma cómoda y sencilla diferentes servicios**.**

- ✓ Puntos a mejorar

  - **Precios mucho más elevados** que sus competidores: Muchos dispositivos de gama media-alta de Android tienen prestaciones similares y unos costes mucho más competitivos que Apple.

  - **Menor personalización, variedad y especialización con respecto a Android:** con Android se puede elegir entre un gran abanico de modelos, de diversos tamaños y precios, y orientados hacia los perfiles de usuario más variados**.** Por ejemplo, en la tienda de aplicaciones de Android, en Google Play Store, existen diversos *launchers* cuyo objetivo es sustituir la apariencia por defecto que muestran las distintas versiones de Android por una más atractiva con las funcionalidades y opciones más personalizables para el usuario. Hacer todo esto en iOS es mucho más limitado.

  - **Carece de la libertad de tener un sistema operativo de código abierto:** los dispositivos Android pueden instalar aplicaciones procedentes de cualquier fuente, ya sea de la tienda oficial o de otras, mientras que tanto los iPhone como el resto de dispositivos pertenecientes a Apple se encuentran con la limitación de hallarse obligados a utilizar únicamente las aplicaciones de la App Store**.**

  - **La comunidad de usuarios de Android es la más grande del mundo,** contando con más recursos en línea que la de iOS, tanto a nivel de desarrollo del sistema operativo como de aplicaciones**.**

2.6.4. Otros sistemas operativos

En este apartado, se va a realizar un breve resumen de lo que han sido o fueron posibles alternativas al duopolio que en la actualidad mantienen los sistemas operativos de Android y iOS en el terreno de los dispositivos móviles, alcanzando casi el 97% de la cuota de mercado, siendo Windows Phone el siguiente en la lista con apenas un 1% (Herrmann, 2017). Debido a la irrelevancia a día de hoy de estas posibles alternativas, no se hará un repaso profundo de las mismas sino un acercamiento del porqué de su fracaso y su estado actual (Herrmann, 2017).

- ✓ Ubuntu Mobile

  Ubuntu Mobile ha quedado reducido prácticamente a la nada. El fabricante español de dispositivos BQ tuvo una serie de smartphones y tabletas en su cartera de productos, que, a través de múltiples funciones de sincronización, colaboraron con el crecimiento conjunto del hardware para escritorio y móvil. Sin embargo, esa convergencia nunca se llegó a producir.

  La lista de dispositivos con Ubuntu Mobile siempre fue escasa. BQ y Meizu mostraron su interés, pero hasta ahí llegó la cosa. Penetrar en el mercado se consigue de otra manera. Los dispositivos ofrecían desde el punto de vista del usuario poco valor añadido en comparación con dispositivos Android económicos. Las especificaciones eran las mismas, incluso muchos dispositivos estaban disponibles en versión Android. Y si el cliente tiene la opción, lo normal es que elija el dispositivo con el que se puede instalar sus aplicaciones favoritas.

- ✓ Cyanogen OS

  La historia de Cyanogen OS era una de los más interesantes de todas las alternativas a Android que se han discutido en los últimos años. Esto se debió, entre otras cosas, según el jefe de proyecto Kirt McMaster, a que competir contra Google en un proyecto en paralelo era un suicidio. A través de una mejor

capacidad de modificación del sistema, Cyanogen Inc. quería ganarse a más desarrolladores de hardware y software que Google con Android. Sin embargo, el sistema debía ser compatible con Android, para que Ubuntu no tuviera problemas con las aplicaciones. Pero debido a que muchas negociaciones no se desarrollaron como se esperaba y a que McMaster rechazó ofertas, como la que al parecer le hizo Nintendo, el proyecto se fue a pique y con él su software.

- ✓ Firefox OS

Firefox se inició con una visión fantástica: en lugar de aplicaciones específicas para cada sistema operativo, todo se desarrollaría como un sitio web, basado en HTML y otros estándares web del W3C. Lo más destacado: lo que se desarrollase una vez, funcionaría en todos los navegadores y en todos los demás sistemas operativos. Por desgracia, esta suposición no funcionó en absoluto, y el proyecto fue cancelado.

- ✓ Windows Phone

Microsoft tiene un montón de dinero con el que probablemente debe estar financiando a su filial móvil durante años. La cuota de mercado de este sistema operativo, con un concepto del que no se puede decir que sea equivocado, sigue siendo extremadamente baja (alrededor de un 1%). No obstante, hacer desaparecer los límites entre sistemas operativos móviles y de escritorio es una oportunidad única para el todavía gigante del software.

- ✓ Tizen

En principio se trataba de un sistema operativo libre. Sin embargo, debido a que al parecer nadie más que Samsung está interesado en él, y a causa de la falta de cooperación, la calidad del código fuente (según expertos en seguridad) está por detrás de la competencia, así que Tizen probablemente no tiene en el punto de mira los smartphones, al menos en Europa, como habíamos supuesto hace poco.

Por lo tanto, Tizen puede permanecer de momento solo en smartwatches, como el Gear S3, y en la cocina en forma de frigoríficos inteligentes, antes de que este sistema plagado de vulnerabilidades de seguridad vuelva inseguros nuestros smartphones.

- ✓ Sailfish OS

El servicio de software finlandés Jolla acaba de firmar un acuerdo en China de gran importancia. Este sistema operativo móvil, altamente escalable, también basado en Unix, funciona en relojes, smartphones, televisores y otros dispositivos.

Jolla tiene un enfoque diferente al de Google y permite a los países licenciar toda la plataforma del sistema operativo. La marca Sailfish no se inmiscuye en ese sentido, y los clientes, aparte de esto, tienen la opción de sustituir u omitir componentes completos de la estructura del sistema. Por lo tanto, tiene una flexibilidad similar a la de un sistema completamente independiente, pero necesita mucho menos esfuerzo para ponerlo en uso, debido a que el núcleo básico continúa desarrollándose, mejorándose y manteniéndose seguro como un proyecto de código abierto con Jolla y sus otros clientes.

Por la naturaleza de la concesión de licencias, Jolla es una alternativa invisible para iOS y Android, sin embargo, no debe subestimarse como sistema genérico. Es muy posible que en esos mercados de sus socios gane una gran relevancia, de forma que se pueda expandir hacia el oeste y amenazar realmente un día al duopolio de Android e iOS.

Como conclusión, en la actualidad, nos encontramos ante un duopolio de sistemas operativos para *smartphones*. La vista se centra ahora en mercados como China o Rusia, donde se prueban y maduran otras alternativas.

2.7. Tecnologías de desarrollo de lado del servidor

En este apartado, se hará un breve resumen de la tecnología más utilizada actualmente como lenguaje de programación del lado de servidor como PHP.

2.7.1. PHP

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente.

En este sentido, PHP es el acrónimo recursivo de *Hipertext Preprocesor*. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y abundante documentación. Un esquema básico de su funcionamiento se muestra en la siguiente Figura 18 (Herraez, 2015):

Figura 18 - Esquema básico de funcionamiento de PHP (Herraez, 2015)

PHP tiene una serie de características más importantes a tener en cuenta, que mencionaremos a continuación (Herraez, 2015):

- ✓ Gratuito: Cualquiera puede descargar a través de la página principal de PHP y de manera gratuita, un módulo que hace que nuestro servidor web comprenda los scripts realizados en este lenguaje.

- ✓ Independiente de plataforma: Puesto que existe un módulo de PHP para casi cualquier servidor web. Esto hace que cualquier sistema pueda ser compatible con el lenguaje y significa una ventaja importante, ya que permite portar el sitio desarrollado en PHP de un sistema a otro sin prácticamente ningún trabajo.

- ✓ De código abierto: PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP

- ✓ Rapidez: PHP, en el caso de estar montado sobre un servidor Unix, es más rápido que ASP, dado que se ejecuta en un único espacio de memoria y esto evita las comunicaciones entre componentes COM que se realizan entre todas las tecnologías implicadas en una página ASP

- ✓ Seguridad: el hecho de que en muchas ocasiones PHP se encuentra instalado sobre servidores Unix, que son de sobra conocidos como más veloces y seguros que el sistema operativo donde se ejecuta las ASP, Windows NT o 2000. Además, PHP permite configurar el servidor de modo que se permita o rechacen diferentes usos, lo que puede hacer al lenguaje más o menos seguro dependiendo de las necesidades de cada cual.

- ✓ Interfaces para una gran cantidad de integración de base de datos: PHP dispone de una conexión propia a todos los sistemas de base de datos. Puede conectarse directamente a las bases de datos de MySQL, PostgreSQL, mSQL, Oracle, dbm, filePro, Hyperwave, Informix, Internase y Sybase, entre otras.

Esto se debe a que PHP utiliza ODBC (*Open Database Connectivity Standard*).

- ✓ Sencillez en el aprendizaje: PHP es relativamente sencillo de aprender, además se pueden encontrar infinidad de manuales en internet

2.8. Tecnologías de bases de datos

En este apartado, se hará una introducción a las bases de datos (BBDD) relaciones, al lenguaje de consulta de BBDD, SQL (*Structured Query Language*) así como un motor de bases de datos basado en el lenguaje SQL como MySQL.

2.8.1. Bases de datos relacionales

El modelo de datos relacional organiza y representa los datos en forma de tablas o relaciones. Relación es un término que viene de la matemática y representa una simple tabla de dos dimensiones, consistente en filas y columnas de datos. Tiene los componentes siguientes:

- ✓ Estructura de datos: Es una colección de objetos abstractos formados por datos. Dominios, tuplas, atributos y relaciones.

- ✓ Operadores: Conjunto de operadores, con reglas bien definidas, que permiten manipular las estructuras de datos. Además del cambio de esquema, los primitivos del álgebra relacional para manipulación de datos, es decir, unión, diferencia, producto cartesiano, proyección y selección.

- ✓ Definiciones de integridad: Colección de conceptos y reglas que permite expresar qué valores de datos pueden aparecer válidamente en nuestro esquema. Las claves y la posibilidad de tener valores nulos. También se incluyen aquí dos reglas de integridad, llamadas:

- Integridad de claves primarias.
- Integridad referencial.

En el modelo de datos relacional la forma en que se almacenan los datos no importa, por lo que es más fácil para un usuario entender y utilizar la BBDD. La información puede ser recuperada o almacenada mediante consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

Además, durante el diseño de una base de datos relacional se realiza un proceso de normalización, en el que cada relación se describe en términos de dependencia. Este proceso se realiza para evitar la redundancia de los datos, evitando así problemas de actualización de los datos, y para proteger la integridad de los datos.

2.8.2. Lenguaje SQL

SQL es el lenguaje estándar ANSI/ISO de definición, manipulación y control de bases de datos relacionales. Es un lenguaje declarativo: sólo hay que indicar qué se quiere hacer. En cambio, en los lenguajes procedimentales es necesario especificar cómo hay que hacer cualquier acción sobre la base de datos.

SQL es un lenguaje muy parecido al lenguaje natural; concretamente, se parece al inglés, y es muy expresivo. Por estas razones, y como lenguaje estándar, el SQL es un lenguaje con el que se puede acceder a todos los sistemas relacionales comerciales.

El lenguaje SQL tiene varios aspectos diferentes (Herraez, 2015):

- ✓ Lenguaje de manipulación de datos (LMD): Este subconjunto de SQL permite a los usuarios formular consultas e insertar, eliminar y modificar filas.

- ✓ Lenguaje de definición de datos (LDD): Este subconjunto de SQL soporta la creación, eliminación y modificación de definiciones de tablas y vistas. Se pueden definir restricciones de integridad para las tablas, bien en el momento de crearlas, bien posteriormente.

- ✓ Disparadores y restricciones de integridad avanzadas: SQL incluye soporte para los disparadores, que son acciones ejecutadas por el SGBD siempre que las modificaciones de la base de datos cumplen las condiciones especificadas en el disparador.

- ✓ SQL incorporado y SQL dinámico: Las características de SQL incorporado permiten llamar al código SQL desde lenguajes anfitriones como C o COBOL. Las características de SQL dinámico permiten que se creen (y ejecuten) consultas en el momento de la ejecución.

- ✓ Ejecución cliente-servidor y acceso a bases de datos remotas: Estas órdenes controlan el modo en que los programas de aplicación clientes pueden conectarse con los servidores de bases de datos de SQL o tener acceso a los datos de las bases de datos a través de la red.

- ✓ Gestión de transacciones: Diversas órdenes permiten que los usuarios controlen de manera explícita aspectos del modo en que se deben ejecutar las transacciones.

- ✓ Seguridad: SQL ofrece mecanismos para control de acceso de los usuarios a los objetos de datos, como tablas y vistas.

2.8.3. Motor de BBDD MySQL

MySQL es un sistema de bases de datos que se puede usar tanto en la web como en el servidor, fácil de usar, y sirve tanto para pequeñas aplicaciones como para aplicaciones más grandes y potentes.

Es un sistema de bases de datos multiplataforma, y además es gratuito y distribuido oficialmente por Oracle.

La estructuración de las diferentes bases de datos en MySQL es muy sencilla, la información de almacena mediante tablas, donde cada uno de los campos que contiene la tabla se estructura en forma de columnas, y cada elemento que se introduzca en la base de datos en forma de filas.

Además, determinadas sentencias de MySQL pueden ser embebidas en código PHP y HTML para diseñar aplicaciones Web dinámicas que incorporan la información de las tablas de MySQL a páginas Web.

Por esta razón MySQL es uno de los sistemas de bases de datos más conocidos y utilizados en la actualidad, webs como Facebook, o Google utilizan este sistema de bases de datos.

Algunas de las características de MySQL son las siguientes:

- ✓ Velocidad: MySQL es rápido.
- ✓ Facilidad de uso: Es un sistema de base de datos de alto rendimiento, pero relativamente simple y es mucho menos complejo de configurar y administrar que sistemas más grandes.
- ✓ Coste: Es gratuito.
- ✓ Capacidad de gestión de lenguajes de consulta: MySQL comprende SQL, el lenguaje elegido para todos los sistemas de bases de datos modernos.
- ✓ Capacidad: Pueden conectarse muchos clientes simultáneamente al servidor. Los clientes pueden utilizar varias bases de datos simultáneamente. Además, está disponible una amplia variedad de interfaces de programación para lenguajes como C, Perl, Java, PHP y Python.
- ✓ Conectividad y seguridad: MySQL está completamente preparado para el trabajo en red y las bases de datos pueden ser accedidas desde cualquier lugar de Internet. Dispone de control de acceso.
- ✓ Portabilidad: MySQL se puede utilizar en una gran cantidad de sistemas Unix diferentes, así como bajo Microsoft Windows.
- ✓ Distribución abierta: Puede obtener y modificar el código fuente de MySQL.

2.9. Tecnologías de servicios web

A la hora de desarrollar servicios web, se tiene que tomar la decisión de que arquitectura será la más apropiada para el sistema en el cuál se desplegarán y el uso se pretenda darles. En esta apartado se presentan os voy a presentar las características de SOAP (*Simple Object Access Protocol*) y REST (*Representational State Transfer*), dos técnicas de arquitectura software orientadas a *webservices*.

2.9.1. REST

REST define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes.

REST emergió en los últimos años como el modelo predominante para el diseño de servicios. De hecho, REST logró un impacto tan grande en la web que prácticamente logró desplazar a SOAP y las interfaces basadas en WSDL por tener un estilo bastante más simple de usar.

Una implementación concreta de un servicio web REST sigue cuatro principios de diseño fundamentales (Seta, 2008):

- ✓ <u>Utilización de los métodos HTTP de manera explícita:</u>

  Una de las características claves de los servicios web REST es el uso explícito de los métodos HTTP, siguiendo el protocolo definido por RFC 2616.

  REST hace que los desarrolladores usen los métodos HTTP explícitamente de manera que resulte consistente con la definición del protocolo. Este principio de diseño básico establece una asociación uno-a-uno entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP. De acuerdo a esta asociación se tienen los siguientes métodos HTTP:

- POST para crear un recurso en el servidor
- GET para obtener un recurso
- PUT para cambiar el estado de un recurso o actualizarlo
- DELETE para eliminar un recurso

✓ Sin mantenimiento del estado:

Los servicios web REST necesitan ser escalables para poder satisfacer una demanda en constante crecimiento. Se usan *clusters* de servidores con balanceadores de carga y alta disponibilidad, *proxies* y *gateways*, que permita transferir peticiones de un equipo a otro para disminuir el tiempo total de respuesta de una invocación al servicio web.

El uso de servidores intermedios para mejorar la escalabilidad hace necesario que los clientes de servicios web REST envíen peticiones completas e independientes; es decir, se deben enviar peticiones que incluyan todos los datos necesarios para cumplir el pedido, de manera que los componentes en los servidores intermedios puedan redireccionar y gestionar la carga sin mantener el estado localmente entre las peticiones.

Una petición completa e independiente hace que el servidor no tenga que recuperar ninguna información de contexto o estado al procesar la petición. Una aplicación o cliente de servicio web REST debe incluir dentro del encabezado y del cuerpo HTTP de la petición todos los parámetros, contexto y datos que necesita el servidor para generar la respuesta. De esta manera, el no mantener estado mejora el rendimiento de los servicios web y simplifica el diseño e implementación de los componentes del servidor, ya que la ausencia de estado en el servidor elimina la necesidad de sincronizar los datos de la sesión con una aplicación externa.

Así pues, podemos distinguir un servicio web con estado o sin él de la siguiente manera:

- La Figura 19 nos muestra un servicio con estado, del cual una aplicación realiza peticiones para la página siguiente en un conjunto de resultados multi-página, asumiendo que el servicio mantiene información sobre la última página que pidió el cliente. En un diseño con estado, el

servicio incrementa y almacena en algún lugar una variable pagina

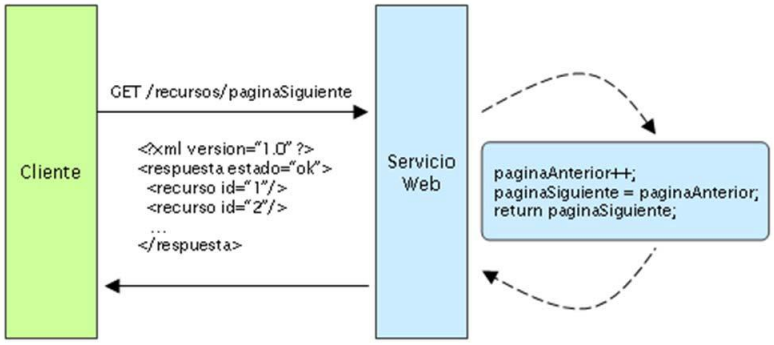

Figura 19 - Esquema de un servicio web con estado (Seta, 2008)

- Por otro lado, los servicios sin estado son mucho más simples de diseñar, escribir y distribuir a través de múltiples servidores. Un servicio sin estado no sólo funciona mejor, sino que además mueve la responsabilidad de mantener el estado al cliente de la aplicación. En un servicio web REST, el servidor es responsable de generar las respuestas y proveer una interfaz que le permita al cliente mantener el estado de la aplicación por su cuenta. Por ejemplo, en el mismo ejemplo de una petición de datos en múltiples páginas, el cliente debería incluir el número de página a recuperar en vez de pedir "la siguiente", tal como se muestra en la siguiente Figura 20

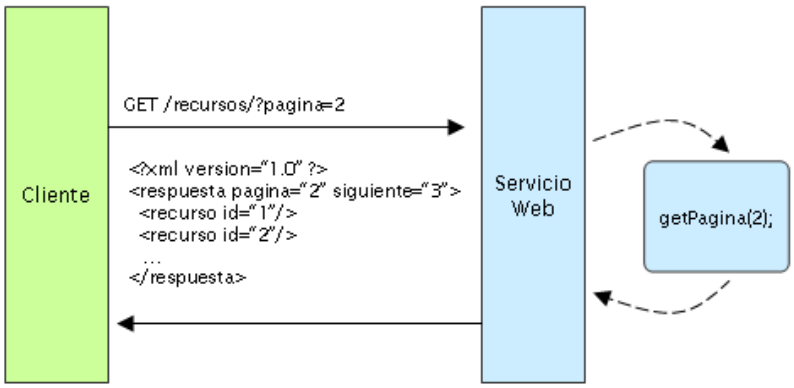

Figura 20 - Esquema de un servicio web sin estado (Seta, 2008)

- ✓ Expone URIs (*Universal Resource Identifier*) con forma de directorios:

Desde el punto de vista del cliente de la aplicación que accede a un recurso, la URI determina qué tan intuitivo va a ser el *webservice* REST, y si el servicio va a ser utilizado tal como fue pensado al momento de diseñarlo. La tercera característica de los servicios web REST es justamente sobre las URIs.

Las URI de los servicios web REST deben ser intuitivas, hasta el punto de que sea fácil adivinarlas. Pensemos en las URI como una interfaz autodocumentada que necesita de muy poca o ninguna explicación o referencia para que un desarrollador pueda comprender a lo que apunta, y a los recursos derivados relacionados.

Una forma de lograr este nivel de usabilidad es definir URIs con una estructura al estilo de los directorios. Este tipo de URIs es jerárquica, con una única ruta raíz, y va abriendo ramas a través de las subrutas para exponer las áreas principales del servicio. De acuerdo a esta definición, una URI no es solamente una cadena de caracteres delimitada por barras, sino más bien un árbol con subordinados y padres organizados como nodos.

- ✓ Representaciones de los recursos en XML, JSON o ambos:

La representación de un recurso en general refleja el estado actual del mismo y sus atributos al momento en que el cliente de la aplicación realiza la petición. Las representaciones del recurso son simples "fotos" en el tiempo. Esto podría ser una representación de un registro de la base de datos que consiste en la asociación entre columnas y tags XML, donde los valores de los elementos en el

XML contienen los valores de las filas. O, si el sistema tiene un modelo de datos, la representación de un recurso es una fotografía de los atributos de una de las cosas en el modelo de datos del sistema. Estas son las cosas que serviciamos con servicios web REST.

La última restricción al momento de diseñar un servicio web REST tiene que ver con el formato de los datos que la aplicación y el servicio intercambian en las peticiones/respuestas. Acá es donde realmente vale la pena mantener las cosas simples, legibles por humanos, y conectadas.

Los objetos del modelo de datos generalmente se relacionan de alguna manera, y las relaciones entre los objetos del modelo de datos (los recursos) deben reflejarse en la forma en la que se representan al momento de transferir los datos al cliente.

2.9.2. SOAP

Es un protocolo que permite la comunicación entre aplicaciones a través de mensajes por medio de Internet. Es independiente de la plataforma, y del lenguaje. Está basado en XML y es la base principal de los *webservices*.

SOAP es el primer protocolo de su tipo que ha sido aceptado prácticamente por todas las grandes compañías de software del mundo. Compañías que en raras ocasiones cooperan entre sí están ofreciendo su apoyo a este protocolo. Algunas de las mayores compañías que soportan SOAP son Microsoft, IBM, SUN, Microsystems, SAP y Ariba.

Algunas de las principales características de SOAP son (Herraez, 2015):

- ✓ No está asociado con ningún lenguaje: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista, pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación.

- ✓ No se encuentra fuertemente asociado a ningún protocolo de transporte: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede

transportarse utilizando cualquier protocolo capaz de transmitir texto.

- ✓ No está atado a ninguna infraestructura de objeto distribuido La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.

- ✓ Aprovecha los estándares existentes en la industria: Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado SOAP no define un medio de trasporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.

- ✓ Permite la interoperabilidad entre múltiples entornos: las aplicaciones se ejecutan en plataformas con estándares que pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecutan en otras plataformas. Por ejemplo, una aplicación de escritorio que se ejecute en una PC puede comunicarse con una aplicación del *backend* ejecutándose en un mainframe capaz de enviar y recibir XML sobre HTTP.

2.10. Elección de tecnologías

Dado que el objetivo de este proyecto es el desarrollo de una aplicación para dispositivos móviles como *smartphones*, se va a resumir las tecnologías elegidas, en función de lo explicado anteriormente, para llevar a cabo el desarrollo de este proyecto.

2.10.1. Sistema operativo para dispositivos móviles

El sistema operativo elegido para el desarrollo de la aplicación móvil es Android. Su elección se realiza con la ayuda de las características propias de este sistema operativo en comparativa con las de su rival más directo,

iOS, ambos presentados en los apartados 2.6.1 y 2.6.3. El resto de sistemas operativos presentados en el apartado 2.6.4 se han descartado debido a su irrelevancia en la actualidad como alternativa a los dos sistemas operativos anteriores en términos de cuota de mercado.

Las razones para la elección de Android en favor de iOS son las siguientes:

- ✓ Lenguaje de programación a utilizar

  Android permite la programación de sus aplicaciones tanto con Java como con Kotlin, recientemente adoptado oficialmente por Google para el desarrollo de aplicaciones Android. Por otra parte, para la programación en iOS es necesario el uso de los lenguajes Objective C ó Swift, este último más utilizado en la actualidad.

  La curva de aprendizaje para alguien sin experiencia en el desarrollo de aplicaciones es más corta para el caso de lenguajes como Java debido a sus numerosas aplicaciones y conocimiento a gran escala, así como a la comunidad de usuarios que hay detrás del mismo a nivel mundial. Además, por la proximidad de conocimientos que se adquieren en el presente Máster de Ingeniería de Telecomunicaciones sobre la cuál versa este proyecto, es más factible en términos de horas de aprendizaje utilizar Java, lo cual convierte a Android en la mejor elección.

- ✓ Coste y accesibilidad de medios

  El coste asociado para poder empezar a programar en Android es notablemente inferior que para iOS.

  Mientras que para Android con cualquier ordenador y la instalación de su IDE (*Integrated Development Environment*) oficial, Android Studio, cualquier persona puede empezar a dar sus primeros pasos en la tecnología.

  Sin embargo, para poder programar en iOS, resulta imprescindible contar con un ordenador propio de Apple (como un Mac), así como el pago anual de una licencia de desarrollo como *developer* en el caso de querer publicar aplicaciones en la propia AppStore.

Además de esto, hay que añadir la diferencia de coste para poder tener un dispositivo con el cual poder probar las propias aplicaciones desarrolladas, siendo mucho mayor para iOS (al sólo ser exclusivos de Apple) que para Android.

✓ Cuota de mercado

Como ya se ha mencionado anteriormente sobre las respectivas cuotas de mercado, tanto para Android como para iOS en los apartados 2.6.2.1 y 2.6.3.2, resulta mucho más atractivo poder desarrollar una aplicación que pueda tener un público objetivo numeroso y que pueda ser accesible a multitud de dispositivos. Este es el caso de Android muy por delante de iOS.

Asimismo, en cuanto a comunidad de usuarios y desarrollo en ambas plataformas, resulta que para Android podemos disponer de muchos más ejemplos prácticos de desarrollo, tanto a nivel de sistema operativo como de aplicaciones, que en iOS.

En la siguiente Figura 21, podemos observar como el número de aplicaciones disponibles para Android en Google Play supera al de la AppStore para iOS.

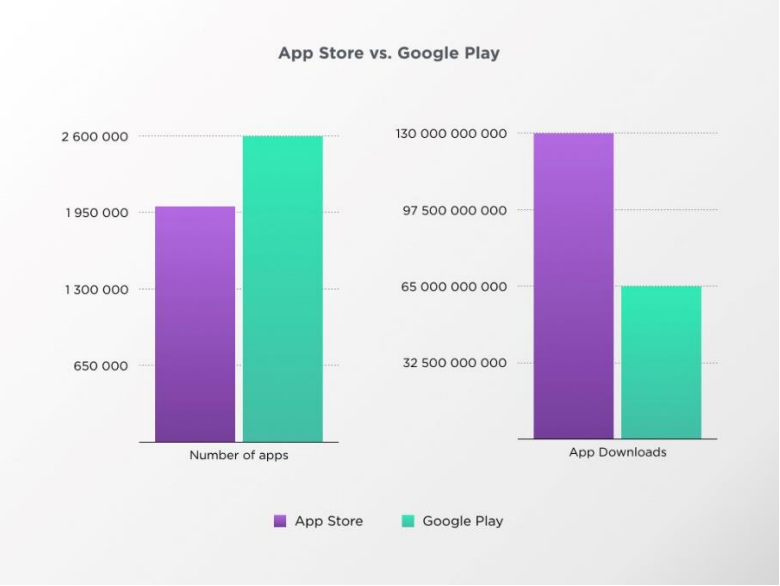

Figura 21 - Comparativa entre los mercados de aplicaciones para los sistemas Android y iOS (Birmacher, 2017)

2.10.2. Lenguaje de servidor y BBDD

Una vez realizado un estudio de las diferentes tecnologías existentes, finalmente se ha optado por el uso PHP junto a MySQL. El hecho de que ASP tenga una fuerte dependencia de Microsoft, debido a que requiere de un servidor web de Microsoft, ha descartado su uso.

PHP es un lenguaje totalmente extendido, con gran fiabilidad y de carácter gratuito, con amplia documentación y manuales en su web que hace que la búsqueda de información sobre sus funcionalidades sea sencilla. Además, es un lenguaje relativamente sencillo e intuitivo, por lo que finalmente ha sido la opción utilizada.

En lo relativo a la elección de la tecnología de BBDD, se ha optado por el motor de BBDD de MySQL, ya que se trata de una de las opciones más extendida en la actualidad, además de poseer una serie de ventajas ya comentadas previamente en 2.8.3. Su carácter gratuito, ser una herramienta de código libre y el hecho de que sitios web tan importantes como Google y Facebook usen esta tecnología como motor de base de datos en sus respectivos sitios web, hacen de esta opción la más adecuada para los propósitos de este proyecto.

Capítulo 3 - Android

Capítulo 3 - Android

3.1. Introducción

Android es un sistema operativo creado para ser independiente de cualquier tipo de arquitectura de hardware en los dispositivos móviles. Esta característica hace que sea tan atractivo ante los fabricantes y desarrolladores.

Adicionalmente su portabilidad, flexibilidad y seguridad les da un añadido a las personas interesadas en los sistemas de código abierto.

Pero antes de empezar a programar en este sistema operativo, es necesario tener una visión más detalla y profunda de los siguientes conceptos relativos a Android:

- ✓ La arquitectura del *framework* desde un punto de vista técnico.
- ✓ Los componentes básicos de las aplicaciones.
- ✓ La estructura básica, organización y recursos de un proyecto de una aplicación enmarcados dentro de su IDE oficial, Android Studio.
- ✓ El proceso de compilación y generación de la aplicación final mediante la herramienta *Gradle*.
- ✓ Patrones de diseño de arquitecturas para aplicaciones móviles en Android más utilizados a día de hoy.

Así pues, estos puntos anteriores serán el contenido del presente capítulo, cuyo objetivo es profundizar en el marco técnico del sistema operativo de Android.

3.2. Arquitectura del *framework*

Android ofrece un completo *framework* de aplicaciones que permite crear apps y juegos innovadores para dispositivos móviles en un entorno de lenguaje Java ó Kotlin.

Dicho *framework* se trata de una pila de software de código abierto basado en Linux creada para una variedad amplia de dispositivos y factores de forma. En la siguiente Figura se muestran los componentes principales del *framework* de Android (Android Developer, 2017):

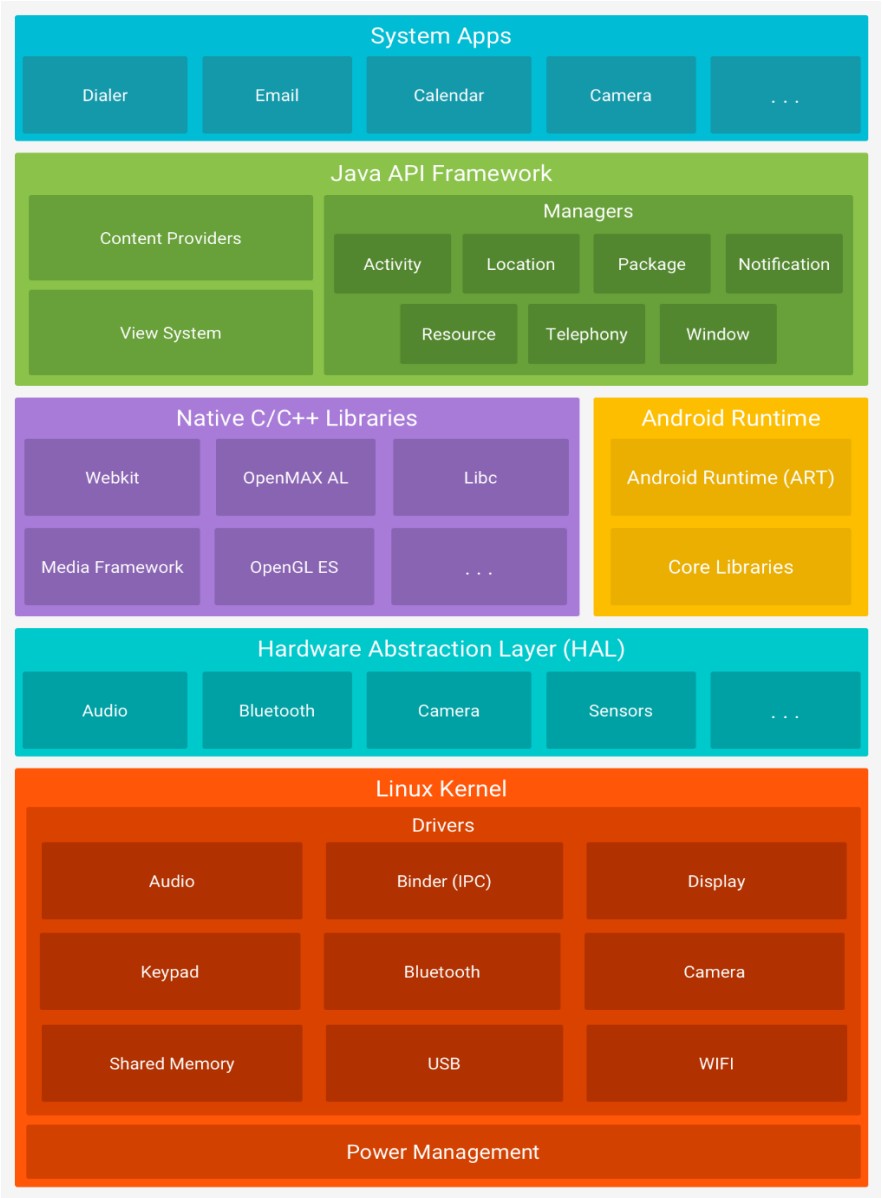

Figura 22 - Arquitectura del framework de Android (Android Developer, 2017)

Examinado la figura anterior desde abajo, se observan los siguientes componentes:

- ✓ *Kernel* de Linux:

  La base del *framework* de Android es el *kernel* de Linux. Por ejemplo, el sistema en tiempo de ejecución de Android (ART), más conocido como la máquina virtual de Android, se basa en el *kernel* de Linux para funcionalidades subyacentes, como la generación de subprocesos y la administración de memoria de bajo nivel.

  El uso del *kernel* de Linux permite que Android aproveche funciones de seguridad claves y, al mismo tiempo, permite a los fabricantes de dispositivos desarrollar controladores de hardware para un *kernel* conocido.

- ✓ Capa de abstracción de hardware (HAL, *Hardware Abstraction Layer*)

  La capa de abstracción de hardware (HAL) brinda interfaces estándares que exponen las capacidades de hardware del dispositivo al *framework* de la API (*Application Programming Interface*) Java de nivel más alto.

  La HAL consiste en varios módulos de biblioteca y cada uno de estos implementa una interfaz para un tipo específico de componente de hardware, como el módulo de la cámara o de bluetooth.

  Cuando el *framework* de una API realiza una llamada para acceder a hardware del dispositivo, el sistema Android carga el módulo de biblioteca para el componente de hardware en cuestión.

- ✓ Sistema en tiempo de ejecución ó máquina virtual ART (Android Runtime)

  Para los dispositivos con Android 5.0 (nivel de API 21) o versiones posteriores, cada app ejecuta sus propios procesos con sus propias instancias del tiempo de ejecución de Android (ART).

El ART está escrito para ejecutar varias máquinas virtuales en dispositivos de memoria baja ejecutando archivos DEX, un formato de código de bytes diseñado especialmente para Android y optimizado para ocupar un espacio de memoria mínimo. Crea cadenas de herramientas, como Jack, y compila fuentes de Java en código de bytes DEX que se pueden ejecutar en la plataforma Android.

Estas son algunas de las funciones principales del ART:

- ✓ Compilación *Ahead-Of-Time* (AOT), en la cual el *bytecode* (código fuente de la aplicación compilado) se traducen código máquina, durante la instalación de la aplicación, antes de su ejecución.

- ✓ Recolección de elementos no usados (mediante el *garbage collector*) optimizada.

- ✓ Mejor compatibilidad con la depuración, como un generador de perfiles de muestras dedicado, excepciones de diagnóstico detalladas e informes de fallos, y la capacidad de establecer puntos de control para controlar campos específicos.

Antes de Android 5.0 (nivel de API 21), *Dalvik* era el sistema en tiempo de ejecución del sistema operativo. Si la app a desarrollar se ejecuta bien en el ART, también debe funcionar en *Dalvik* por retro compatibilidad, pero es posible que no suceda lo contrario.

Dalvik, a diferencia de ART, utiliza la compilación JIT (*Just-In-Time)*, en la cual el *bytecode* (código fuente de la aplicación compilado) se traduce en código máquina en tiempo de ejecución de la aplicación.

Por último, en Android también se incluye un conjunto de bibliotecas de tiempo de ejecución centrales que proporcionan la mayor parte de la funcionalidad del lenguaje de programación Java; se incluyen algunas funciones del lenguaje Java 8, que el *framework* de la Java API usa.

- ✓ Bibliotecas C/C++ nativas

Muchos componentes y servicios centrales del sistema Android, como el ART y la HAL, se basan en código nativo que requiere bibliotecas nativas escritas en C y C++.

La plataforma Android proporciona la API del *framework* de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las apps. Por ejemplo, puedes acceder a OpenGL ES a través de la Java OpenGL API del *framework* de Android para agregar a tu app compatibilidad con los dibujos y la manipulación de gráficos 2D y 3D.

Si se quiere desarrollar una app que requiere C o C++, es posible usar el NDK (*Native Development Kit*) de Android para acceder a algunas de estas bibliotecas de plataformas nativas directamente desde tu código nativo.

- ✓ *Framework* de la API Java

Todo el conjunto de funciones del SO Android está disponible mediante API escritas en el lenguaje Java. Estas API son los cimientos que necesitas para crear apps de Android simplificando la reutilización de componentes del sistema y servicios centrales y modulares, como los siguientes:

- ✓ Un sistema de vista enriquecido y extensible que puedes usar para compilar la IU de una app; se incluyen listas, cuadrículas, cuadros de texto, botones e incluso un navegador web integrable.

- ✓ Un administrador de recursos que te brinda acceso a recursos sin código, como cadenas de texto localizadas, gráficos y archivos de diseño.

- ✓ Un administrador de notificaciones que permite que todas las apps muestren alertas personalizadas en la barra de estado.

- ✓ Un administrador de actividad que administra el ciclo de vida de las apps y proporciona una pila de retroceso de navegación común.

- ✓ Proveedores de contenido que permiten que las apps accedan a datos desde otras apps, como la app de Contactos, o compartan sus propios datos.

Los desarrolladores tienen acceso total a las mismas API del *framework* que usan las apps del sistema Android.

- ✓ Aplicaciones del sistema

En Android se incluye un conjunto de apps centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, entre otros elementos. Las apps incluidas en la plataforma no tienen un estado especial entre las apps que el usuario elije instalar; por ello, una app externa se puede convertir en el navegador web, el sistema de mensajería SMS o, incluso, el teclado predeterminado del usuario (existen algunas excepciones, como la app *Settings* del sistema).

Las apps del sistema funcionan como apps para los usuarios y brindan capacidades claves a las cuales los desarrolladores pueden acceder desde sus propias apps. Por ejemplo, si en tu app se intenta entregar un mensaje SMS, no es necesario que compiles esa funcionalidad tú mismo; como alternativa, puedes invocar la app de SMS que ya está instalada para entregar un mensaje al receptor que especifiques.

3.3. Componentes básicos de las aplicaciones de Android

El *framework* de aplicaciones de Android permite crear aplicaciones interesantes e innovadoras usando un conjunto de componentes reutilizables. Este apartado explica cómo se pueden construir los componentes que definen los bloques de una aplicación y cómo conectarlos usando *intents* (Android Developer, 2017).

3.3.1. View

Las vistas (*views*) son los componentes básicos con los que se construye la interfaz gráfica de la aplicación, análogo por ejemplo a los controles de Java o .NET. De inicio, Android dispone de una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear algunos personalizados.

3.3.2. Activity

Una *Activity* es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción, como marcar un número telefónico, tomar una foto, enviar un correo electrónico o ver un mapa. A cada actividad se le asigna una ventana en la que se puede dibujar su interfaz de usuario. La ventana generalmente abarca toda la pantalla, pero en ocasiones puede ser más pequeña que esta y quedar "flotando" encima de otras ventanas.

Una aplicación generalmente consiste en múltiples actividades vinculadas de forma flexible entre sí. Normalmente, una actividad en una aplicación se especifica como la actividad "principal" que se presenta al usuario cuando este inicia la aplicación por primera vez. Cada actividad puede a su vez iniciar otra actividad para poder realizar diferentes acciones. Cada vez que se inicia una actividad nueva, se detiene la actividad anterior, pero el sistema conserva la actividad en una pila (la "pila de actividades"). Cuando se inicia una actividad nueva, se la incluye en la pila de actividades y capta el foco del usuario. La pila de actividades cumple

con el mecanismo de pila "el último en entrar es el primero en salir" (estructura LIFO, *Last Input First Output*), por lo que, cuando el usuario termina de interactuar con la actividad actual y presiona el botón Atrás, se quita de la pila (y se destruye) y se reanuda la actividad anterior. En la Figura 23, se visualiza este comportamiento con una línea de tiempo que muestra el progreso entre actividades junto con la pila de actividades actual en cada momento.

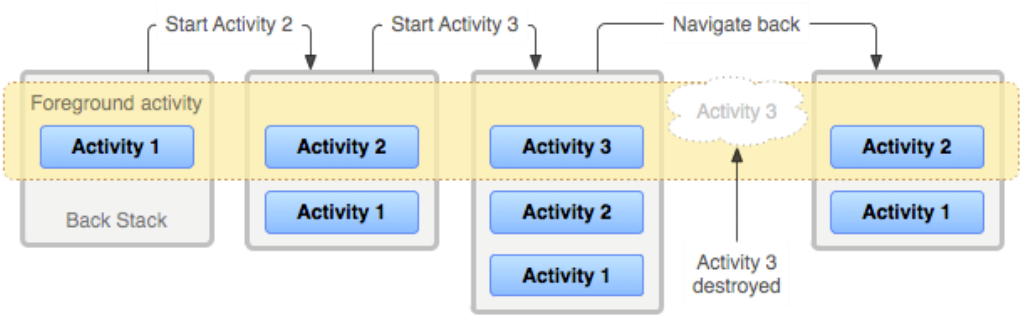

Figura 23 - Comportamiento de la pila de actividades en Android (Android Developer, 2017)

Cuando se detiene una actividad porque se inicia otra, se notifica el cambio de estado a través de los métodos *callback* del ciclo de vida de la actividad. Existen diferentes métodos *callback* que podría recibir una actividad como consecuencia de un cambio de estado (ya sea que el sistema la esté creando, deteniendo, reanudando o destruyendo) y cada callback te da la oportunidad de realizar una tarea específica que resulta adecuada para ese cambio de estado. Todas estas transiciones de estado forman parte del ciclo de vida de la actividad, que se describe de manera visual en la siguiente Figura 24 (los bloques rectangulares representan los métodos *callback* que se pueden implementar para realizar operaciones cuando la actividad cambie de estado).

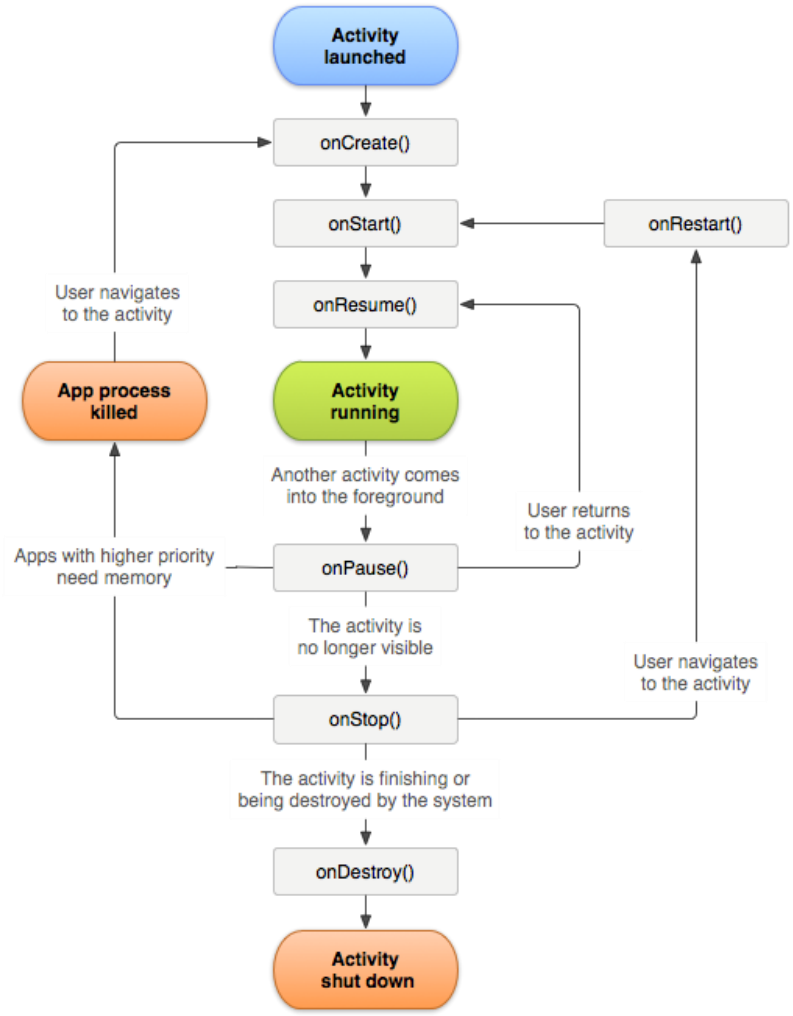

Figura 24 - Ciclo de vida de una actividad en Android (Android Developer, 2017)

3.3.3. Fragments

Android introduce los fragmentos en Android 3.0 (nivel de API 11), principalmente para admitir diseños de IU más dinámicos y flexibles en pantallas grandes, como las de las *tablets*. Como la pantalla de una *tablet* es mucho más grande que la de un teléfono, hay más espacio para combinar e intercambiar componentes de la IU. Los fragmentos admiten esos diseños sin la necesidad de que administres cambios complejos en la jerarquía de vistas. Al dividir el diseño de una actividad en fragmentos, puedes modificar el aspecto de la actividad durante el tiempo de ejecución y conservar esos cambios en una pila de actividades administrada por la actividad.

Un *Fragment* representa un comportamiento o una parte de la interfaz de usuario en una *Activity*. Puedes combinar múltiples fragmentos en una sola actividad para crear una IU multipanel y volver a usar un fragmento en múltiples actividades. Puedes pensar en un fragmento como una sección modular de una actividad que tiene su ciclo de vida propio, recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando (algo así como una "subactividad" que puedes volver a usar en diferentes actividades).

Un fragmento siempre debe estar integrado a una actividad y el ciclo de vida del fragmento (Figura 25) se ve directamente afectado por el ciclo de vida de la actividad anfitriona. Por ejemplo, cuando la actividad está pausada, también lo están todos sus fragmentos, y cuando la actividad se destruye, lo mismo ocurre con todos los fragmentos. Sin embargo, mientras una actividad se está ejecutando (está en el estado del ciclo de vida reanudada), puedes manipular cada fragmento de forma independiente; por ejemplo, para agregarlos o quitarlos. Cuando realizas una transacción de fragmentos como esta, también puedes agregarlos a una pila de actividades administrada por la actividad; cada entrada de la pila de actividades en la actividad es un registro de la transacción de fragmentos realizada. La pila de actividades le permite al usuario invertir una transacción de fragmentos (navegar hacia atrás) al presionar el botón Atrás.

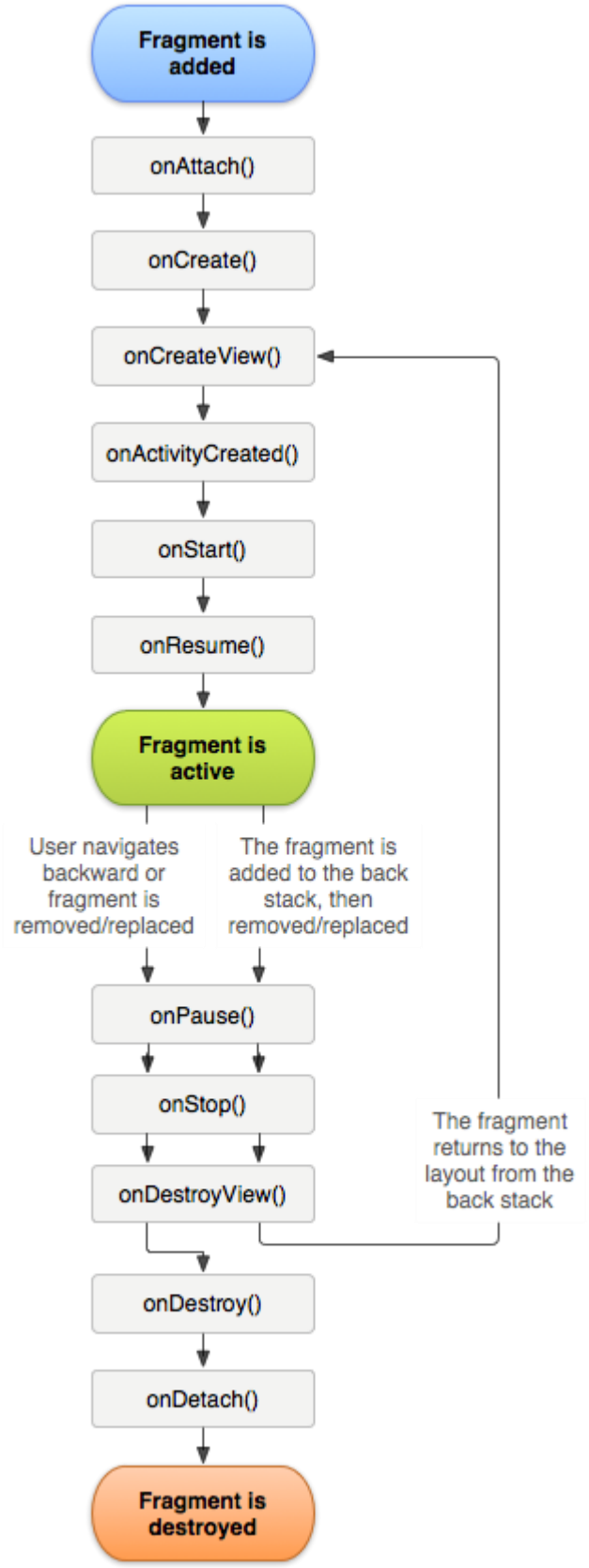

Figura 25 - Ciclo de vida de un fragmento en Android (Android Developer, 2017)

3.3.4. Loaders

Introducidos en Android 3.0, los *loaders* son unos componentes que simplifican la carga asincrónica de datos en una actividad o un fragmento. Tienen las siguientes características:

- ✓ Están disponibles para cada *Activity* y *Fragment*.
- ✓ Proporcionan carga asincrónica de datos.
- ✓ Controlan el origen de los datos y entregan resultados nuevos cuando el contenido cambia.
- ✓ Se reconectan automáticamente al último cursor del cargador cuando se recrean después de un cambio de configuración. Por lo tanto, no necesitan consultar los datos nuevamente

3.3.5. Services

Un *Service* es un componente de una aplicación que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario. Otro componente de la aplicación puede iniciar un servicio y continuará ejecutándose en segundo plano, aunque el usuario cambie a otra aplicación. Además, un componente puede enlazarse con un servicio para interactuar con él e incluso realizar una comunicación entre procesos (IPC, *Inter-Process Communication*). Por ejemplo, un servicio puede manejar transacciones de red, reproducir música, realizar I/O de archivos o interactuar con un proveedor de contenido, todo en segundo plano.

Un servicio en Android puede adoptar esencialmente dos formas:

- ✓ Servicio iniciado:

  Un servicio está "iniciado" cuando un componente de aplicación (como una actividad) lo inicia. Una vez iniciado, un servicio puede ejecutarse en segundo plano de manera indefinida, incluso si se

destruye el componente que lo inició. Por lo general, un servicio iniciado realiza una sola operación y no devuelve un resultado al emisor. Por ejemplo, puede descargar o cargar un archivo a través de la red. Cuando la operación está terminada, el servicio debe detenerse por sí mismo.

- ✓ Servicio de enlace:

  Un servicio es de "de enlace" cuando un componente de la aplicación se vincula a él. Un servicio de enlace ofrece una interfaz cliente-servidor que permite que los componentes interactúen con el servicio, envíen solicitudes, obtengan resultados e incluso lo hagan en distintos procesos con la comunicación entre procesos (IPC). Un servicio de enlace se ejecuta solamente mientras otro componente de aplicación está enlazado con él. Se pueden enlazar varios componentes con el servicio a la vez, pero cuando todos ellos se desenlazan, el servicio se destruye.

3.3.6. Content Providers

Los proveedores de contenido (c*ontent providers*) administran el acceso a un conjunto estructurado de datos. Encapsulan los datos y proporcionan mecanismos para definir la seguridad de los datos. Los proveedores de contenido son la interfaz estándar que conecta datos en un proceso con código que se ejecuta en otro proceso.

Su funcionamiento se basa en administrar el acceso a un repositorio central de datos. Un proveedor forma parte de una aplicación de Android y a menudo proporciona su propia IU para trabajar con los datos. No obstante, los proveedores de contenido están principalmente orientados a que los usen otras aplicaciones que acceden al proveedor usando un objeto de cliente del proveedor. Juntos, proveedores y clientes de proveedores ofrecen una interfaz estándar y uniforme para los datos que también manipula la comunicación dentro del proceso y el acceso seguro a datos.

Para ello, un proveedor de datos presenta datos a aplicaciones externas en forma de una o más tablas que son similares a las tablas de una base de datos relacional. Una fila representa una instancia de algún tipo de datos que recopila el proveedor, y cada columna de la fila

representa una pieza individual de los datos recopilados para una instancia.

Android incluye proveedores de contenido que administran datos como video, audio, imágenes e información de contacto personal. Con algunas restricciones, cualquier aplicación de Android puede acceder a esos proveedores.

3.3.7. Broadcast Receiver

Un *broadcast receiver* es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: "Batería baja", "SMS recibido", "Tarjeta SD insertada", …) o por otras aplicaciones (cualquier aplicación puede generar mensajes (*intents*, en terminología Android) *broadcast*, es decir, no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo).

3.3.8. Widgets

Los *widgets* son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal (*home screen*) del dispositivo Android y recibir actualizaciones periódicas. Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal.

3.3.9. Intent

Un *intent* es el elemento básico de comunicación entre los distintos componentes Android descritos anteriormente. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones. Mediante un *intent* se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.

3.3.10. Procesos y subprocesos

Cuando un componente de la aplicación se inicia y la aplicación no tiene ningún otro componente ejecutándose, el sistema de Android inicia un nuevo proceso de Linux para la aplicación con un único subproceso de ejecución.

De manera predeterminada, todos los componentes de la misma aplicación se ejecutan en el mismo proceso y subproceso (que se denomina "subproceso principal"). Si el componente de una aplicación se inicia y ya existe un proceso para la aplicación (porque otro componente de la aplicación existe), el componente se inicia dentro de ese proceso y usa el mismo subproceso de ejecución. Sin embargo, es posible configurar que diferentes componentes de la aplicación se ejecuten en procesos separados y puedes crear subprocesos adicionales para cualquier proceso.

3.4. Estructura y recursos de un proyecto de una aplicación con Android Studio

Todas las aplicaciones Android se componen de una estructura general; conformada por **librerías de código, archivos de recursos y vistas, código fuente y el *Android Manifest*** (Cardenas, 2015)**.**

Así pues, resulta de gran importancia conocer los directorios que forman nuestro proyecto dentro del IDE oficial Android Studio, ya que estos contienen los recursos que vamos a utilizar a medida que desarrollamos nuestra aplicación. Pero, antes de pasar a explorar dichos directorios, hace falta dar un breve resumen de lo que supone Android Studio para el desarrollo de Android, así como sus principales características.

3.4.1. IDE Android Studio

Android Studio fue anunciado en el año 2013 en la conferencia de Google I/O. Fue creado para reemplazar a Eclipse, la plataforma que se usaba para la creación de aplicaciones y que todavía hoy en día es utilizada por muchos programadores. De esta manera, con Android Studio, Google consigue su propio IDE para el desarrollo de aplicaciones, pudiendo

instalar todo el SDK para desarrollar apps específicas adaptadas a la mayor parte de versiones (Sgoliver, 2014).

Un breve resumen del mismo es el siguiente:

- ✓ Nace el 16 de mayo de 2013 en la Google I/O.
- ✓ Primera versión estable en diciembre de 2014.
- ✓ Disponible para Windows / Mac / Linux.

Con el lanzamiento de su propio IDE oficial para Android, Google se beneficia de tener su propio creador de aplicaciones para esta plataforma y actualmente ya cuenta con la versión Android Studio 3.0 *Beta*. Obviamente, al ser oficial de Google cuenta con muchas ventajas, entre otras, la de tener siempre un software actualizado y con numerosas novedades.

En la actualidad Android Studio es la plataforma que se postula como el más completo IDE para desarrollar aplicaciones Android con muchas características que destacan de los otros programas usados para este trabajo, está basado en IntelliJ y puede ser descargado de forma gratuita a través de la licencia de Apache 2.0.

Cuenta con una estructura simple que permite organizar los proyectos de manera que facilite su ubicación y su publicación, como también un entorno para desarrollar más potente, fácil e intuitivo. Permite ver el desarrollo a tiempo real de las aplicaciones y las pantallas en las que será usada la aplicación, y a su vez nos ofrece plantillas para diferentes elementos para programar como el uso de mapas (AndroidStudioFAQs, 2016).

En cuanto a sus principales características de Android Studio se destacan las siguientes (Sgoliver, 2014) (AndroidStudioFAQs, 2016):

- ✓ Soporte para programar aplicaciones para Android *Wear*.
- ✓ Herramienta *Lint*: Se trata de un analizador de código estático, con lo que permite detectar el código no compatible entre arquitecturas diferentes o código confuso.
- ✓ Utiliza *ProGuard* para poder optimizar y reducir el código del proyecto al exportar a APK, para dispositivos de gama con limitaciones.
- ✓ Nuevo diseño del editor con un soporte para la posible edición de temas.

- ✓ Actualizaciones frecuentes (diferentes canales de desarrollo).
- ✓ Nueva interfaz específica para el desarrollo en Android.
- ✓ Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de acabar la aplicación.
- ✓ Vista previa con renderizado en tiempo real de los diseños de los *layouts* de la aplicación, en diferentes tipos de proyectos y resoluciones.
- ✓ Posibilita la opción del control de versiones mediante las herramientas como Mercurial, Git, Github o Subversion.
- ✓ Permite la importación de los proyectos realizados desde Eclipse.
- ✓ Asociación automática de carpetas y archivos con su papel en la aplicación
- ✓ Proceso de compilación rápido y generación del archivo final de la aplicación (APK, *Android Application Package*).
- ✓ Ejecución y depuración de la aplicación en tiempo real desde un emulador o dispositivo móvil real.
- ✓ Como posible desventaja de este IDE, es que los requisitos mínimos de funcionamiento son un poco elevados, por lo que resulta necesario de un ordenador con unas prestaciones medias-altas para poder aprovechar al máximo todas sus funcionalidades.

3.4.2. Estructura de un proyecto en Android Studio

Google siempre ha tratado de hacer más sencilla la tarea de añadir recursos a los proyectos de Android. Uno de los pasos más significativos es el uso del lenguaje XML, lo que evita exceso de código inútil en nuestros ficheros java, y hace más clara la comprensión del código fuente. La modularización en directorios (como se puede apreciar en la siguiente Figura 26) y el uso de este lenguaje ha logrado que cada vez más personas aprendan a programar en Android (Leira, 2015a).

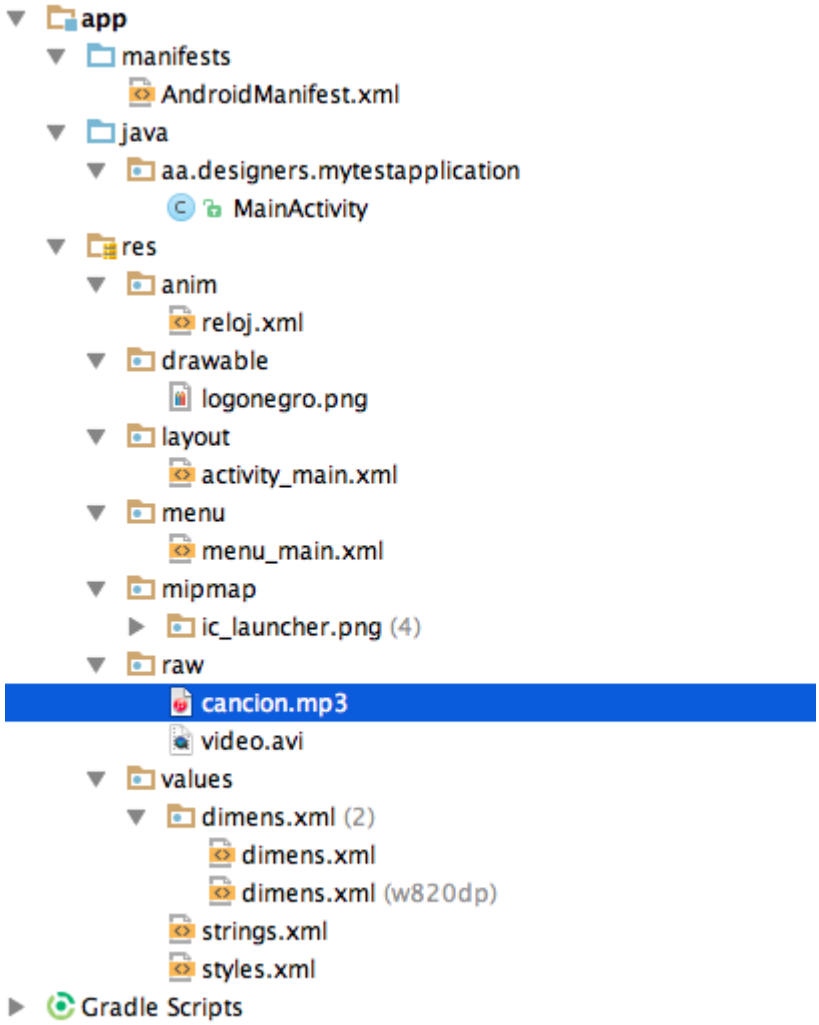

Figura 26 - Estructura de un proyecto en Android Studio mediante la vista Android (Leira, ¿Cúal es la estructura de un proyecto en Android Studio?, 2015a)

En base a la figura anterior, podemos describir, de una manera resumida, los diferentes directorios que componen la estructura del proyecto (Sgoliver, 2014) (Leira, 2015a):

- ✓ ***manifests***: aquí se puede encontrar el archivo de configuración AndroidManifest.xml en la raíz de todos nuestros proyectos (directorio "/src/main/AndroidManifest.xml"). En este archivo podemos configurar las opciones básicas (como el nombre del paquete, número de versión y nombre de nuestra aplicación), así como la definición de otros componentes de la aplicación (como *activities* ó *services*) o el establecimiento de los permisos de la aplicación con el dispositivo móvil (Leira, 2015b).

- ✓ ***java:*** en este directorio encontraremos los ficheros.java con el código fuente de la aplicación. Su ubicación es "/src/main/java/".

- ✓ ***res:*** contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, *layouts*, cadenas de texto, etc. Los diferentes tipos de recursos se pueden distribuir entre las siguientes subcarpetas (sus ubicaciones respectivas son "/src/main/res/<nombre de la subcarpeta>"):

    - ***anim:*** contienen la definición de las animaciones utilizadas por la aplicación en formato XML.

    - ***color:*** contiene ficheros XML de definición de listas de colores según estado.

    - ***drawable:*** contiene las imágenes y otros elementos gráficos usados por la aplicación. Para poder definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo se suele dividir en varias subcarpetas:

        - /drawable (recursos independientes de la densidad)
        - /drawable-ldpi (densidad baja)
        - /drawable-mdpi (densidad media)
        - /drawable-hdpi (densidad alta)
        - /drawable-xhdpi (densidad muy alta)
        - /drawable-xxhdpi (densidad muy muy alta)

    - ***layout:*** Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Para definir distintos *layouts* dependiendo de la orientación del dispositivo se puede dividir también en subcarpetas:

        - /layout (vertical)
        - /layout-land (horizontal)

    - ***menu:*** contiene la definición XML de los menús de la aplicación.

    - ***mipmap:*** Contiene los iconos de lanzamiento de la aplicación (el icono que aparecerá en el menú de aplicaciones del

dispositivo) para las distintas densidades de pantalla existentes. Al igual que en el caso de las carpetas /drawable, se dividirá en varias subcarpetas dependiendo de la densidad de pantalla:

- /mipmap-mdpi
- /mipmap-hdpi
- /mipmap-xhdpi
- etc…

- ***raw:*** Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.

- ***values:*** Contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), *arrays* de valores (arrays.xml), tamaños (dimens.xml), etc.

- ***gradle:*** Los ficheros de compilación gradle de Android Studio no son necesarios puesto que son generados automáticamente por Android Studio.

  De todos ellos, el más relevante es el llamado *build.gradle*. Contiene información necesaria para la compilación del proyecto, por ejemplo, la versión del SDK de Android utilizada para compilar, la mínima versión de Android que soportará la aplicación, referencias a las librerías externas utilizadas, etc.

  Respecto a la mínima versión (o API mínima) a utilizar, ésta implicará que nuestra aplicación se pueda ejecutar en más o menos dispositivos. De esta forma, cuanto menor sea ésta, a más dispositivos podrá llegar nuestra aplicación, pero más complicado será conseguir que se ejecute correctamente en todas las versiones de Android. Para hacernos una idea del número de dispositivos que cubre con cada versión podemos hacer uso de la ayuda en *el set*-up inicial de un nuevo proyecto que nos ofrece Android Studio, dentro de la configuración de las plataformas y APIs de

Android a utilizar, que mostrará el porcentaje de dispositivos que ejecutan actualmente cada versión de Android.

En un proyecto pueden existir varios ficheros *build.gradle*, para definir determinados parámetros a distintos niveles. Por ejemplo, en nuestro proyecto podemos ver que existe un fichero *build.gradle* a nivel de proyecto, y otro a nivel de módulo dentro de la carpeta /app. El primero de ellos definirá parámetros globales a todos los módulos del proyecto, y el segundo sólo tendrá efecto para cada módulo en particular.

3.5. Proceso de compilación y generación de la aplicación final mediante Gradle

Gradle es un sistema de compilación automatizado que reúne en un uno solo las mejores prestaciones de otros sistemas de compilación. Está basado en JVM (*Java Virtual Machine*), lo que significa que puedes escribir tu propio script en Java, así Android Studio lo entenderá y lo usará.

Lo mejor de *Gradle* es que es un *plugin*, lo que facilita su actualización y su exportación de un proyecto a otro. Esto significa que puedes tener tu propio lenguaje de programación y automatizar el proceso de compilación en un solo paquete (de la misma manera que un archivo *jar* en caso de Java) y poder distribuirlo al resto del mundo.

De esta manera, Google ha creado uno de los sistemas de compilación más avanzados del mercado para permitir a todos los usuarios escribir sus propios scripts sin necesidad de aprender ningún nuevo lenguaje, disminuyendo así la curva de aprendizaje y permitiendo llegar a un mayor público la programación en Android (Leira, ¿Qué es gradle en Android Studio?, 2015c).

Entre las ventajas que aporta *Gradle* se encuentran (Leira, 2015c):

- ✓ Permite reutilizar fácilmente código.
- ✓ Hace sencilla la tarea de configurar y personalizar la compilación.

- ✓ Permite la distribución sencilla de código al resto del mundo, y fomenta el trabajo en equipo.

- ✓ Gestiona las dependencias de forma potente y cómoda (está basado en Maven).

- ✓ Permite la compilación desde la consola de comandos, lo que nos puede hacer más sencilla la tarea de compilación en sistemas sin el entorno de desarrollo montado.

- ✓ Facilita enormemente la creación de diferentes versiones de la aplicación, por ejemplo, para hacer múltiples versiones para móviles o *tablets*, diferentes *flavors* para versiones de pago o gratuitas, etc...

El sistema de compilación de Android compila recursos y código fuente de la app y los empaqueta en APK que puedes probar, implementar, firmar y distribuir. Android Studio usa *Gradle* para automatizar y administrar el proceso de compilación, y al mismo tiempo te permite definir configuraciones de compilación personalizadas y flexibles. Cada configuración de compilación puede definir un conjunto de código y recursos propios, y reutilizar las partes comunes a todas las versiones de tu app. El complemento de Android para *Gradle* funciona con el paquete de herramientas de compilación para proporcionar procesos y ajustes configurables específicos para la compilación y prueba de aplicaciones de Android.

Gradle y el complemento de Android se ejecutan independientemente de Android Studio. Esto significa que es posible compilar las aplicaciones para Android desde Android Studio, la línea de comandos de tu máquina o en máquinas en las que no esté instalado Android Studio (como servidores de integración continua). El resultado de la compilación es el mismo si compilas un proyecto desde la línea de comandos, en una máquina remota o usando Android Studio.

La flexibilidad del sistema de compilación de Android permite realizar configuraciones de compilación personalizadas sin modificar los archivos de origen principales de tu app.

Respecto al proceso de compilación de Android, además de ser muy flexible con el uso de *Gradle*, incluyen muchas herramientas y procesos que convierten el proyecto de aplicación en un paquete para aplicaciones de Android (APK) (Android Developer, 2017).

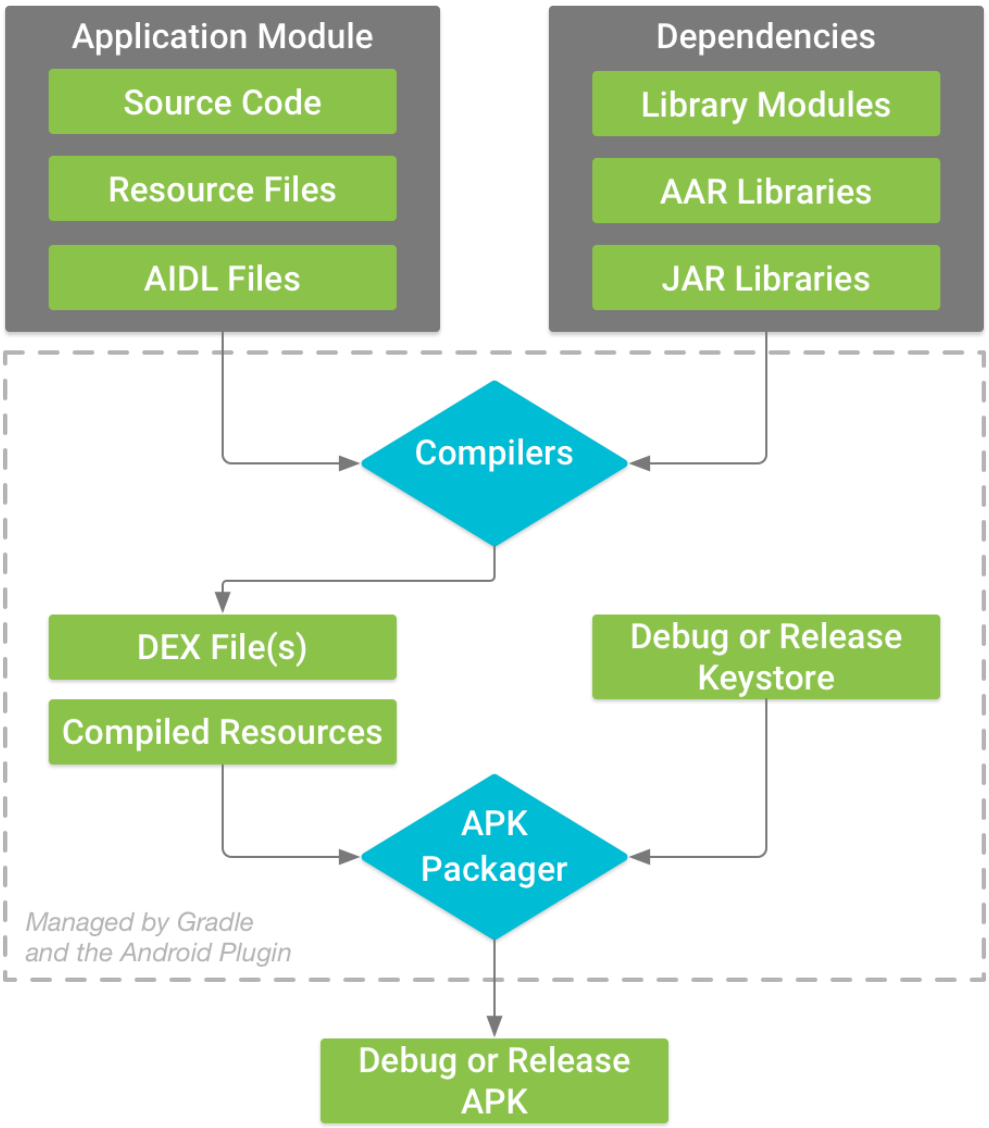

Figura 27 - Proceso de compilación típico de un módulo de aplicación para Android (Android Developer, 2017)

El proceso de compilación para un módulo de aplicación para Android, como se muestra en la Figura 27, sigue estos pasos generales (Android Developer, 2017):

1. Los compiladores convierten el código fuente en archivos DEX (*Dalvik Executable*, ejecutable de *Dalvik*), los cuales incluyen el *bytecode,* que se ejecutará en la máquina virtual *Dalvik* o ART de los dispositivos Android, y todo lo demás en recursos compilados.

2. El empaquetador de APK combina los archivos DEX y los recursos compilados en un solo APK. No obstante, antes de que la aplicación pueda instalarse e implementarse en un dispositivo Android, se debe firmar el APK.

3. El empaquetador de APK firma tu APK usando el *keystore* de depuración o lanzamiento:

    a. Si se trata de una versión de depuración de la aplicación (es decir, una app que para pruebas y generación de perfiles), el empaquetador firmará la app con el *keystore* de depuración. Android Studio configura automáticamente nuevos proyectos con un *keystore* de depuración.

    b. Si compilas una versión de lanzamiento de la aplicación, el empaquetador firmará la app con el *keystore* de lanzamiento.

4. Antes de generar el APK final, el empaquetador usa la herramienta *zipalign* para optimizar la aplicación de modo que use menos memoria cuando se ejecute en un dispositivo.

Al final del proceso de compilación, se obtiene un APK de depuración o un APK de lanzamiento de la aplicación con el cuál se podrá usar a fin de implementar, probar o lanzar la aplicación para usuarios externos

3.6. Patrones de diseño de arquitecturas de aplicaciones móviles

A lo largo de los últimos años, han surgido nuevos patrones de diseño para las aplicaciones móviles que pretender dar una respuesta acerca de cuál es la mejor organización ó arquitectura para una aplicación en forma de componentes lógicos. Para el caso de Android, tanto los desarrolladores de esta tecnología como su comunidad han evolucionado desde el uso del conocido patrón de diseño MVC (*Model-View-Controller*) hasta patrones más modulares y más fáciles de testear como MVP (*Model-View-Presenter*) ó MVVM (*Model-View-ViewModel*). En este apartado se realizará una breve introducción a los mismos, resaltando sus principales características.

3.6.1. Model-View-Controller (MVC)

El patrón de diseño MVC separa la arquitectura de la aplicación en tres componentes lógicos: Modelo, Vista y Controlador (Figura 28). Esta arquitectura supone una separación de responsabilidades, desacoplando la interfaz de usuario (la vista) del modelo de dominio y la lógica del controlador. Como ventajas, esto supone que el mantenimiento y el testeo de este tipo de aplicaciones se convierte en mucho más simple y fácil de llevar a cabo (Maxwell, 2017) (Prajesh, 2016).

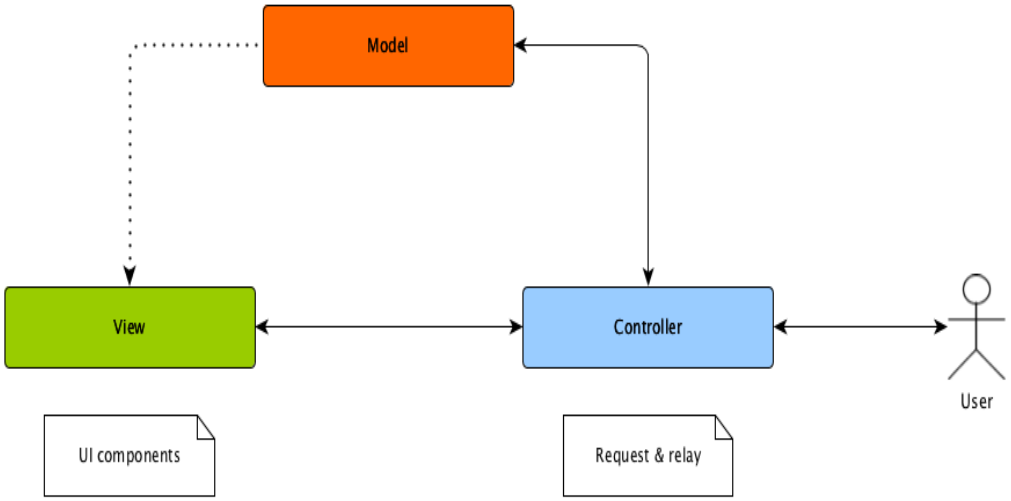

Figura 28 - Patrón de diseño MVC (Prajesh, 2016)

✓ Modelo

El modelo representa un conjunto de clases que describen la lógica de negocio (*business logic*), es decir, el modelo de negocio junto con las operaciones sobre los datos del modelo. También define las reglas de negocio para los datos, describiendo se puede modificar el modelo de datos.

✓ Vista

La vista representa los componentes de la interfaz gráfica (UI, *User interface*). Solamente es responsable de mostrar los datos que recibe del controlador. En resumen, "transforma" el modelo de datos en la interfaz gráfica.

✓ Controlador

El controlador es responsable de procesar las peticiones que le llegan como consecuencia de llamadas de red, a BBDD o cómo interacciones del usuario con la aplicación, entre otras. Así pues, recibe las peticiones del usuario a través de la vista, luego procesa los datos introducidos por el usuario con la ayuda del modelo de datos y envía los resultados vuelta a la vista. Típicamente, actúa como coordinador/mediador entre la vista y el modelo.

3.6.2. Model-View-Presenter (MVP)

El patrón de diseño MVP es muy similar al patrón MVC anterior en el cuál el controlador ha sido reemplazado por el presentador. Al igual que en el MVC, en el MVP se separa la arquitectura de la aplicación en tres componentes lógicos: Modelo, Vista y Presentador (Figura 29) (Maxwell, 2017) (Prajesh, 2016).

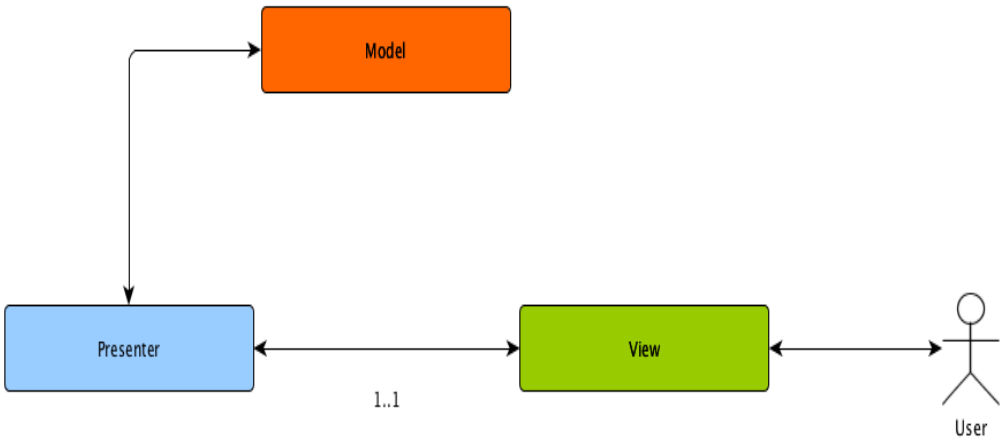

Figura 29 - Patrón de diseño MVP (Prajesh, 2016)

✓ Modelo

El modelo representa un conjunto de clases que describen la lógica de negocio (*business logic*), es decir, el modelo de negocio junto con las operaciones sobre los datos del modelo. También define las reglas de negocio para los datos, describiendo cómo se puede modificar el modelo de datos.

✓ Vista

La vista representa los componentes de la interfaz gráfica (UI, *User interface*). Solamente es responsable de mostrar los datos que recibe del controlador. En resumen, "transforma" el modelo de datos en la interfaz gráfica.

✓ Presentador

El presentador es responsable de procesar todos los eventos de la UI por cuenta de la vista. Así pues, recibe las peticiones del usuario a través de la vista, luego procesa los datos del usuario con la ayuda del modelo de datos y envía los resultados vuelta a la vista. Sin embargo, al contrario que en MVC con la vista y controlador, la vista y presentador en MVP están completamente desacoplados uno del otro y se comunican mediante sus interfaces respectivas. Además, el presentador no gestiona las peticiones entrantes de red como controlador.

✓ Aspectos clave sobre MVP

- El usuario interactúa con la vista,
- Existe una relación uno a uno entre la vista y el presentador, con lo que una única vista se puede asociar con un único presentador.
- La vista tiene una referencia al presentador, pero la vista no tiene ninguna referencia al modelo de datos, algo que sí existe en el patrón MVC.
- Proporciona una comunicación bidireccional entre vista y presentador.

3.6.3. Model-View-ViewModel (MVVM)

El patrón de diseño MVVM (Figura 30) soporta un modo bidireccional de aprovisionamiento de datos entre la vista y el componente modelo-vista. Esto permite la propagación automática de los cambios, junto con el estado del modelo-vista, a la vista. Típicamente, el modelo-vista utiliza el patrón observador (*observer pattern*) para notificar los cambios en el modelo-vista al modelo de datos (Maxwell, 2017) (Prajesh, 2016).

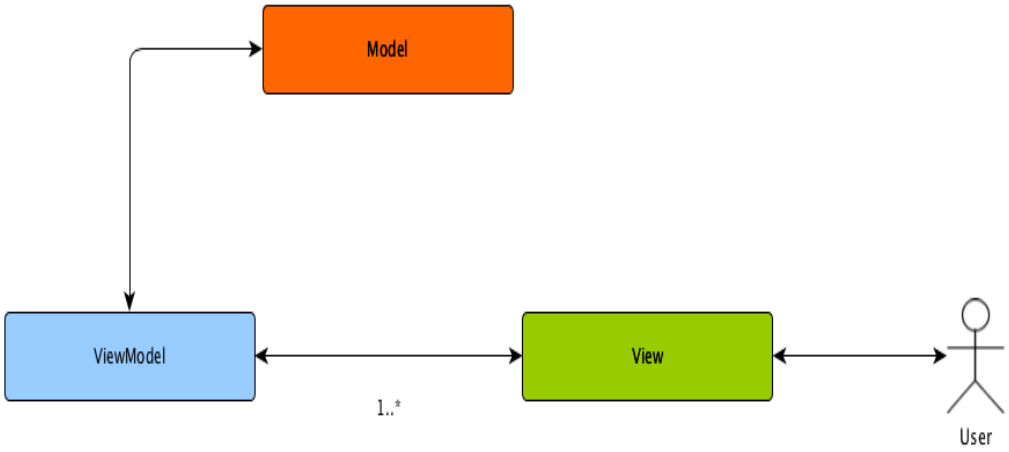

Figura 30 - Patrón de diseño MVVM (Prajesh, 2016)

✓ Modelo

El modelo representa un conjunto de clases que describen la lógica de negocio (*business logic*), es decir, el modelo de negocio junto con las operaciones sobre los datos del modelo. También define las reglas de negocio para los datos, describiendo se puede modificar el modelo de datos.

✓ Vista

La vista representa los componentes de la interfaz gráfica (UI, *User interface*). Solamente es responsable de mostrar los datos que recibe del controlador. En resumen, "transforma" el modelo de datos en la interfaz gráfica.

✓ Modelo-Vista

El modelo-vista es responsable de presentar los métodos, comandos y otras propiedades que ayudan a mantener el estado de la vista, manipular el modelo de datos como consecuencia de las acciones o eventos en la vista, así como lanzar eventos en la misma.

✓ Aspectos clave sobre MVVM

- El usuario interactúa con la vista,
- Existe una relación varios a uno entre la vista y el modelo-vista, con lo que varias vistas se pueden asocia con un único modelo-vista.
- La vista tiene una referencia al modelo-vista, pero la modelo-vista no tiene ninguna referencia ni información sobre la vista.
- Proporciona un modo bidireccional de aprovisionamiento de datos entre la vista y el componente modelo-vista.

Capítulo 4 – Descripción técnica de la aplicación de fisioterapia

Capítulo 4 - Descripción técnica de la aplicación de fisioterapia

En este apartado se realizará una descripción más detallada a nivel técnico de la aplicación de fisioterapia desarrollada para este Trabajo Fin de Máster, como ya se comentó en los objetivos del capítulo 1. Para ello se describirá:

- ✓ La BBDD que utiliza, su estructura y sus tablas que la componen, así como la vinculación de las mismas entre sí. Esta BBDD es la que se utiliza la aplicación para obtener la información del modelo de datos, a través de peticiones de red a un servidor *web*.

- ✓ Un resumen del proceso de comunicación que lleva a cabo la aplicación para obtener y modificar la información almacenada en la base de datos anterior.

- ✓ La arquitectura utilizada para la aplicación según los patrones de diseño vistos anteriormente en el apartado 3.6.

- ✓ Las funcionalidades de la aplicación con ayuda de diagramas de flujo.

4.1. Base de datos

La BBDD utilizada para esta aplicación de fisioterapia consta de 8 tablas de datos, tal y como se visualizan en la siguiente Figura 31. Además, esta BBDD tiene una estructura racional con vinculación entre sus tablas.

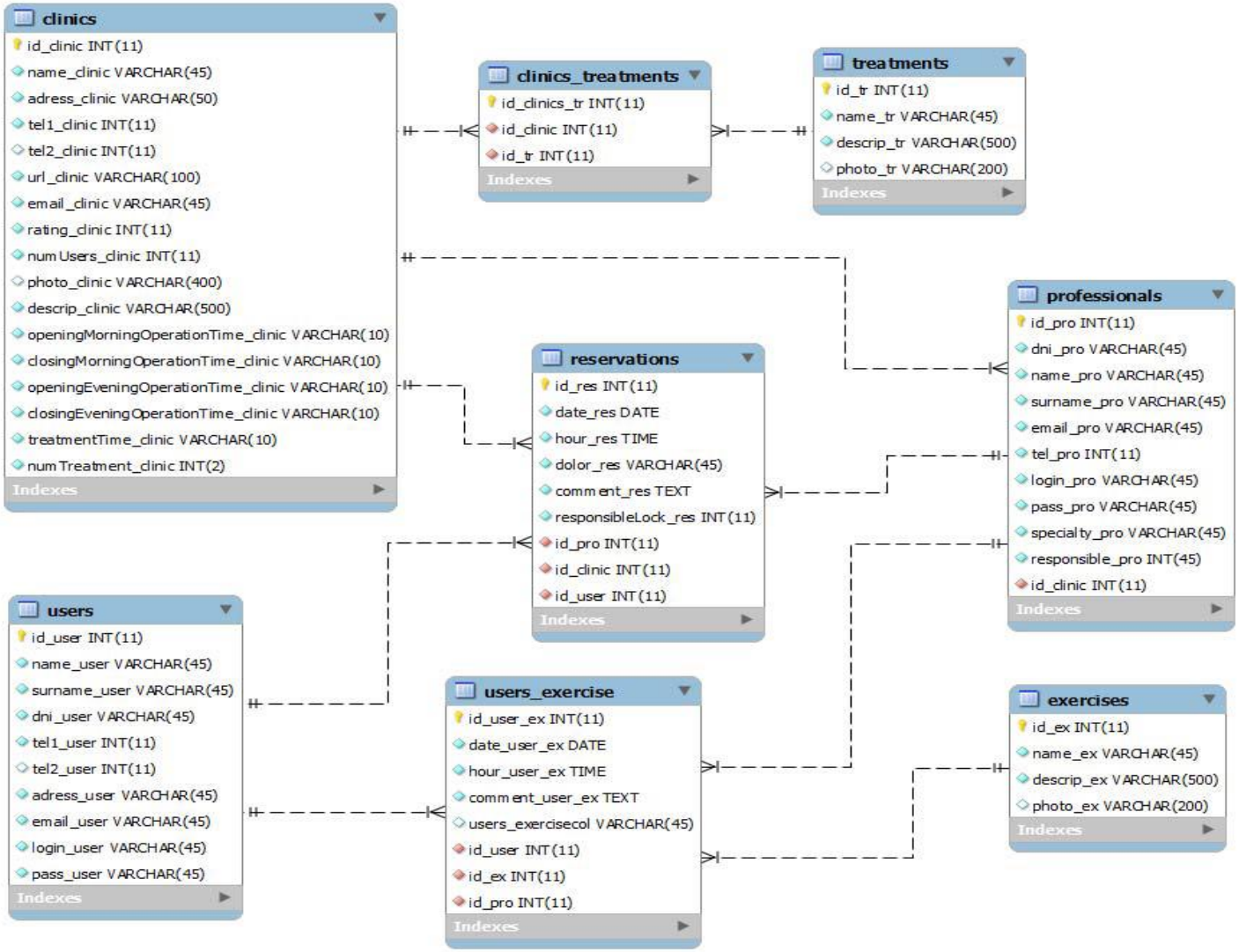

Figura 31 - Estructura de la BBDD de la aplicación

A continuación, se describen cada una de las tablas que componen la BBDD, así como sus campos y su significado dentro del contexto de la aplicación.

4.1.1. Tabla *Clinics*

En esta tabla se almacenarán todos los datos de las clínicas de fisioterapia registradas para su uso y consulta en la aplicación. Su estructura es la siguiente según la Tabla 2:

Tabla *Clinics*				
Campo	**Atributo (Longitud máx.)**	**¿Nulo?**	**Valor predeterminado**	**Atributo(s) extra(s)**
id_clinic	int (11)	No	Ninguno	Auto incremental
name_clinic	Varchar (45)	No	Ninguno	-
Address_clinic	Varchar (50)	No	Ninguno	-
tel1_clinic	int (11)	No	Ninguno	-
tel2_clinic	Int (11)	No	Ninguno	-
url_clinic	varchar (100)	No	Ninguno	-
email_clinic	Varchar (45)	No	Ninguno	-
rating_clinic	Int (11)	No	Ninguno	-
numUsers_clinic	Int (11)	No	Ninguno	-
photo_clinic	Varchar (400)	No	Ninguno	-
descrip_clinic	Varchar (500)	No	Ninguno	-
openingMorningOperationTime_clinic	Varchar (10)	No	Ninguno	-

closingMorningOperationTime_clinic	Varchar (10)	No	Ninguno	-
openingEveningOperationTime_clinic	Varchar (10)	No	Ninguno	-
closingEveningOperationTime_clinic	Varchar (10)	No	Ninguno	-
treatmentTime_clinic	Varchar (10)	No	Ninguno	-
numTreatment_clinic	Int (11)	No	Ninguno	-

Tabla 2 - Estructura de la tabla Clinics de la BBDD

- ✓ id_clinic: es el identificador de cada clínica, el campo que va a diferenciar una clínica de otra.
- ✓ name_clinic: indica el nombre de la clínica.
- ✓ adress_clinic: indica la dirección de la clínica.
- ✓ tel1_clinic: da la información del primer teléfono de la clínica.
- ✓ tel2_clinic: da la información del segundo teléfono de la clínica.
- ✓ url_clinic: campo utilizado para guardar la página web de la clínica.
- ✓ email_clinic: en este campo se guarda el correo electrónico de contacto de la clínica.
- ✓ rating_clinic: campo utilizado para indicar la valoración que tiene cada clínica.
- ✓ numUsers_clinics: guarda el número de usuarios que han reservado en la clínica, necesario para establecer el rating.
- ✓ photo_clinic: es un campo de texto, debido a que no se guarda, la imagen como tal en la base de datos, sino la *url* donde está alojada la fotografía.
- ✓ descrip_clinic: campo utilizado para mostrar una breve descripción de los servicios que ofrecen las clínicas a los pacientes.
- ✓ opening (Morning, Evening) OperationTime_clinic: indica la hora en la que la clínica inicia la apertura, tanto por la mañana (*Morning*), como por la tarde (*Evening*).
- ✓ closing (Morning, Evening) OperationTime_clinic: indica la hora en la que la clínica cierra, tanto por la mañana (*Morning*), como por la tarde (*Evening*).
- ✓ treatmentTime_clinic: campo que indica el tiempo aproximado que tarda un profesional en realizarle un tratamiento a un cliente.
- ✓ numTreatment_clinic: indica el número de tratamientos que realiza una clínica en una misma franja horaria.

4.1.2. Tabla *Treatments*

En esta tabla se almacenarán todos los datos de los tratamientos disponibles para actividades de fisioterapia, para su uso y consulta en la aplicación. Su estructura es la siguiente según la Tabla 3:

Tabla *Treatments*				
Campo	**Atributo (Longitud máx.)**	**¿Nulo?**	**Valor predeterminado**	**Atributo(s) extra(s)**
id_tr	int (11)	No	Ninguno	Auto incremental
name_tr	varchar (45)	No	Ninguno	-
descrip_tr	varchar (500)	No	Ninguno	-
photo_tr	varchar (200)	No	*Null*	-

Tabla 3 - Estructura de la tabla Treatments de la BBDD

- ✓ id_tr: es el identificador único de cada tratamiento.
- ✓ name_tr: indica el nombre del tratamiento.
- ✓ descrip_tr: campo para la descripción del tratamiento (en qué consiste, aplicaciones…).
- ✓ photo_tr: campo que contiene la *url* donde está alojada la fotografía relativa al tratamiento.

4.1.3. Tabla *Clinics_Treatments*

El objetivo de esta tabla es la de relacionar, en un orden de varios a varios, clínicas y tratamientos. Es decir, una clínica puede ofrecer varios tratamientos y un tratamiento puede ser ofrecido en varias clínicas; de ahí la necesidad de esta tabla con dos tipos de claves (identificador para la clínica dónde se imparte un tratamiento y el identificador de este último). Su estructura es la siguiente según la Tabla 4:

Tabla *Clinics_Treatments*				
Campo	**Atributo (Longitud máx.)**	**¿Nulo?**	**Valor predeterminado**	**Atributo(s) extra(s)**
id_clinic_tr	int (11)	No	Ninguno	Auto incremental
id_clinic	int (11)	No	Ninguno	-
id_tr	int (11)	No	Ninguno	-

Tabla 4 - Estructura de la tabla Clinics_Treatments de la BBDD

- ✓ id_clinic_tr: campo clave, con el que se idéntica a cada elemento perteneciente a esta tabla.
- ✓ id_clinic: es el identificador único de la clínica a la que pertenece el tratamiento con identificador id_tr.
- ✓ id_tr: es el identificador único de cada tratamiento.

4.1.4. Tabla *Users*

Esta tabla contiene la estructura para el almacenamiento de la información de un usuario registrado en la aplicación. Su estructura es la siguiente según la Tabla 5:

Tabla *Users*				
Campo	**Atributo (Longitud máx.)**	**¿Nulo?**	**Valor predeterminado**	**Atributo(s) extra(s)**
id_user	int (11)	No	Ninguno	Auto incremental
name_user	varchar (45)	No	Ninguno	-
surname_user	int (11)	No	Ninguno	-
dni_user	varchar (45)	No	Ninguno	-
tel1_user	int (11)	No	Ninguno	-
tel2_user	int (11)	No	*Null*	-

address_user	varchar (45)	No	Ninguno	-
email_user	varchar (45)	No	Ninguno	-
login_user	varchar (45)	No	Ninguno	-
pass_user	varchar (45)	No	Ninguno	-

Tabla 5 - Estructura de la tabla Users de la BBDD

- ✓ id_user: es el identificador único para cada usuario.
- ✓ name_user: almacena el nombre del paciente.
- ✓ surname_user: almacena el apellido del paciente.
- ✓ dni_user: se almacena el documento nacional de identidad del paciente.
- ✓ tel1_user: almacena el primer teléfono de contacto del usuario.
- ✓ tel2_user: almacena el segundo teléfono de contacto del usuario.
- ✓ address_user: recoge la dirección del domicilio del paciente.
- ✓ email_user: recoge la información del correo electrónico del paciente.
- ✓ login: es un campo que va a permitir distinguir a un usuario u otro, es único por lo que no puede haber dos usuarios con el mismo *login*.
- ✓ password: es la clave de ingreso a la aplicación de cada usuario, la que lleva a permitir acceder a la aplicación.

4.1.5. Tabla *Exercises*

Esta tabla contiene la estructura para el almacenamiento de la información de los ejercicios registrados en la aplicación a realizar por los usuarios. Su estructura es la siguiente según la Tabla 6:

Tabla *Exercises*				
Campo	**Atributo (Longitud máx.)**	**¿Nulo?**	**Valor predeterminado**	**Atributo(s) extra(s)**
id_ex	int (11)	No	Ninguno	Auto incremental
name_ex	varchar (45)	No	Ninguno	-
descrip_ex	varchar (500)	No	Ninguno	-

photo_ex	varchar (200)	Si	*Null*	-
url_video_ex	text	Si	*Null*	-

Tabla 6 - Estructura de la tabla Exercises de la BBDD

- ✓ id_ex: es el identificador único de cada ejercicio.
- ✓ name_ex: indica el nombre del ejercicio.
- ✓ descrip_ex: campo para la descripción del ejercicio (en qué consiste, número de repeticiones…).
- ✓ photo_ex: campo que contiene la *url* donde está alojada la fotografía relativa al ejercicio, si lo hubiese.
- ✓ photo_ex: campo que contiene la *url* del vídeo asociado al ejercicio si lo hubiese.

4.1.6. Tabla *Users_Exercise*

El objetivo de esta tabla es la de relacionar, en un orden de varios a varios, usuarios y ejercicios. Es decir, un usuario puede tener varios ejercicios asignados y un ejercicio puede estar asignado a varios usuarios; de ahí la necesidad de esta tabla con dos tipos de claves (identificador para el ejercicio que tiene asignado un usuario y el identificador de este último). Su estructura es la siguiente según la Tabla 7:

Tabla *Exercises*				
Campo	**Atributo (Longitud máx.)**	**¿Nulo?**	**Valor predeterminado**	**Atributo(s) extra(s)**
id_user_ex	int (11)	No	Ninguno	Auto incremental
id_user	int (11)	No	Ninguno	-
id_ex	int (11)	No	Ninguno	-
date_users_ex	date	Si	*Null*	-
hour_user_ex	time	Si	*Null*	
id_pro	int (11)	No	Ninguno	

comment_user_ex	text	Si	*Null*	
user_exercisecol	tinyint (1)	No	0	

Tabla 7 - Estructura de la tabla Exercises de la BBDD

- ✓ id_user_ex: identificador de relación entre usuario y ejercicio.
- ✓ id_user: identificador de usuario.
- ✓ id_ex: identificador de ejercicio.
- ✓ date_user_ex: indica la fecha de disponibilidad del vídeo asociado al ejercicio asignado si así fuese seleccionado.
- ✓ hour_user_ex: indica la hora de disponibilidad del vídeo asociado al ejercicio asignado si así fuese seleccionado.
- ✓ comment_user_ex: indica información que el profesional puede considerar interesante, como se debería de realizar el ejercicio, aclaraciones…
- ✓ users_exerciseCol: indica si el ejercicio esta realizado.

4.1.7. Tabla *Professionals*

Esta tabla contiene la estructura para el almacenamiento de la información de un profesional personal de una clínica de fisioterapia o de un encargado de la clínica, el cual está capacitado para realizar una serie de actividades a mayores que el profesional (esto se describe mediante un atributo booleano en la tabla llamado *responsible_pro*). Su estructura es la siguiente según la Tabla 8:

Tabla *Professionals*				
Campo	**Atributo (Longitud máx.)**	**¿Nulo?**	**Valor predeterminado**	**Atributo(s) extra(s)**
id_pro	int (11)	No	Ninguno	Auto incremental
name_pro	varchar (45)	No	Ninguno	-
surname_pro	varchar (45)	No	Ninguno	-
dni_pro	varchar (45)	No	Ninguno	-

tel_pro	int (11)	No	Ninguno	-
email_pro	varchar (45)	No	Ninguno	-
speciality_pro	varchar (45)	No	Ninguno	-
id_clinic	int (11)	No	Ninguno	-
login_pro	varchar (45)	No	Ninguno	-
pass_pro	varchar (45)	No	Ninguno	-
responsable_pro	varchar (45)	No	Ninguno	-

Tabla 8 - Estructura de la tabla Professionals de la BBDD

- ✓ id_pro: identificador del profesional, único para cada especialista.
- ✓ name_pro: contiene el nombre del profesional.
- ✓ surname_pro: contiene el apellido del profesional.
- ✓ dni_pro: contiene el número del documento de identidad del profesional.
- ✓ tel_pro: contiene el teléfono de contacto del profesional.
- ✓ email_pro: almacena el correo electrónico de contacto con el especialista.
- ✓ login_pro: almacena el *login* de inicio de sesión de cada profesional.
- ✓ pass_pro: almacena la clave de acceso a los servicios de la parte de profesional o encargado.
- ✓ speciality_pro: hace referencia a la especialidad de cada profesional, pudiendo haber profesionales expertos en fisioterapia deportiva, otros expertos en una técnica en concreto, etc.…
- ✓ id_clinic: identificador de la clínica en la que está trabajando el profesional o encargado.
- ✓ responsible_pro: atributo que indica si se trata de un profesional (valor “0”) ó de trata de un encargado (valor “1”), pudiendo este último tener privilegios para realizar tareas de asignación de pacientes a profesionales, etc.

4.1.8. Tabla *Reservations*

Esta tabla contiene la estructura para el almacenamiento de la información de reservas realizadas por los usuarios con los profesionales de las clínicas de fisioterapia. Cabe destacar que esta tabla implica varias relaciones entre otras debido al carácter de la información que almacena. Su estructura es la siguiente según la Tabla 9:

Tabla *Reservations*				
Campo	**Atributo (Longitud máx.)**	**¿Nulo?**	**Valor predeterminado**	**Atributo(s) extra(s)**
id_res	int (11)	No	Ninguno	Auto incremental
date_Res	date	No	Ninguno	-
hour_res	time	No	Ninguno	-
id_clinic	int (11)	No	Ninguno	-
id_user	int (11)	No	Ninguno	-
dolor_res	varchar (45)	No	Ninguno	-
comment_res	text	No	Ninguno	-
id_pro	int (11)	No	Ninguno	-
responsibleLock_res	int (11)	No	Ninguno	-

Tabla 9 - Estructura de la tabla Reservations de la BBDD

4.2. Relaciones entre las tablas de la base de datos

Una vez descritas cada una de las tablas de la base de datos para la aplicación de fisioterapia, resulta de ayuda conocer cómo se relacionan algunas de ellas con el objetivo de aclarar cómo se estructura la información que posteriormente utilizará la aplicación en cada una de sus funcionalidades.

Entre las 8 tablas existentes, se pueden observar cinco relaciones bien diferenciadas, explicadas a continuación.

4.2.1. Relación entre las tablas *Clinics* y *Treatments*

De forma gráfica en la Figura 32 se puede observar una relación entre ambas tablas, con una cardinalidad de varios a varios, ya que como se ha mencionado anteriormente, una clínica puede ofrecer varios tratamientos y cada uno de ellos puede ser ofrecido en varias clínicas. Debido a que la aplicación puede consultar los tratamientos de una clínica determinada, se hace necesaria describir esta relación entre ambas tablas anteriores mediante la tabla *clinics_treatments*.

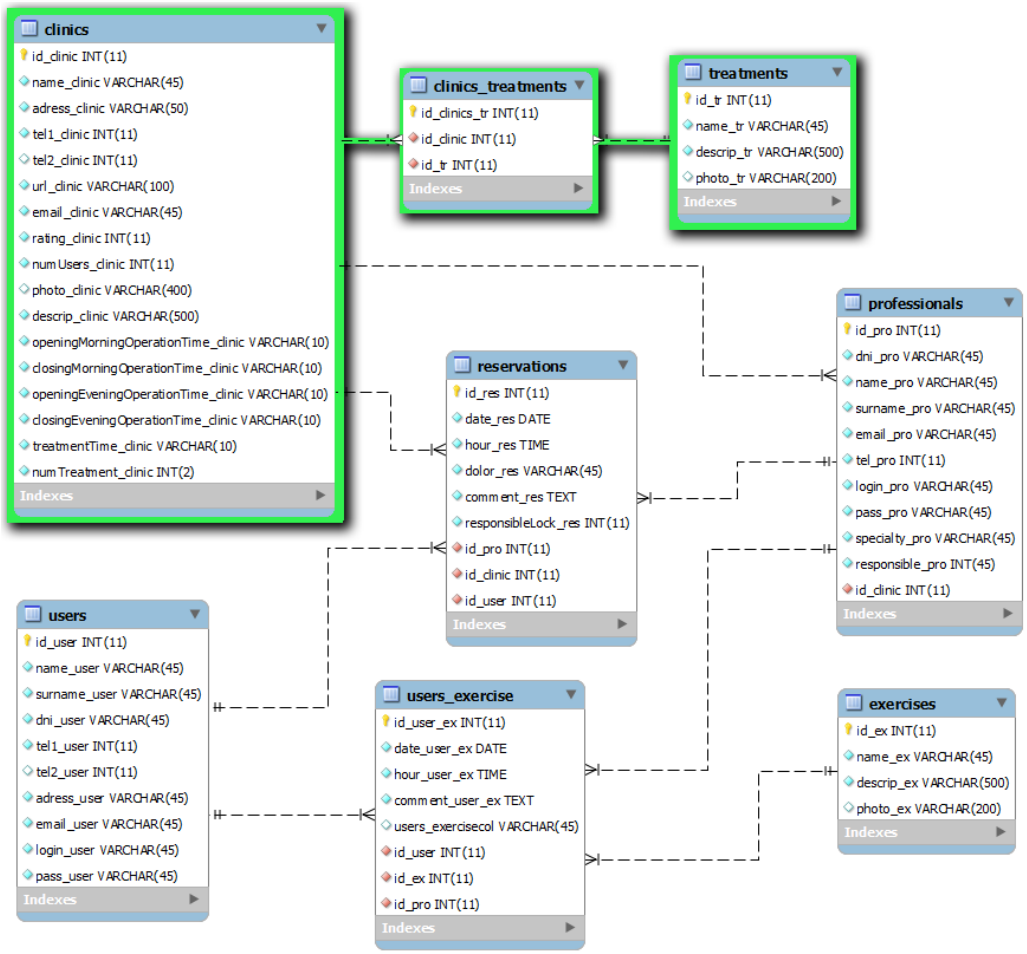

Figura 32 - Relación entre las tablas Clinics y Treatments

4.2.2. Relación entre las tablas *Professionals, Exercises* y *Users*

De forma gráfica en la Figura 33 se puede observar una relación entre todas las tablas.

En primer lugar, se observa una cardinalidad de varios a varios entre las tablas *Users* y *Exercises*; ya que como se ha mencionado anteriormente, un usuario puede tener asignado varios ejercicios y un ejercicio puede estar asignado a varios usuarios. Debido a que la aplicación puede consultar los ejercicios de un usuario determinado, se hace necesaria describir esta relación entre ambas tablas anteriores mediante la tabla *users_exercise*.

En segundo lugar, ambas tablas anteriores están relacionadas con la tabla *professionals*, ya que la asignación de un ejercicio a un usuario es realizada por un profesional.

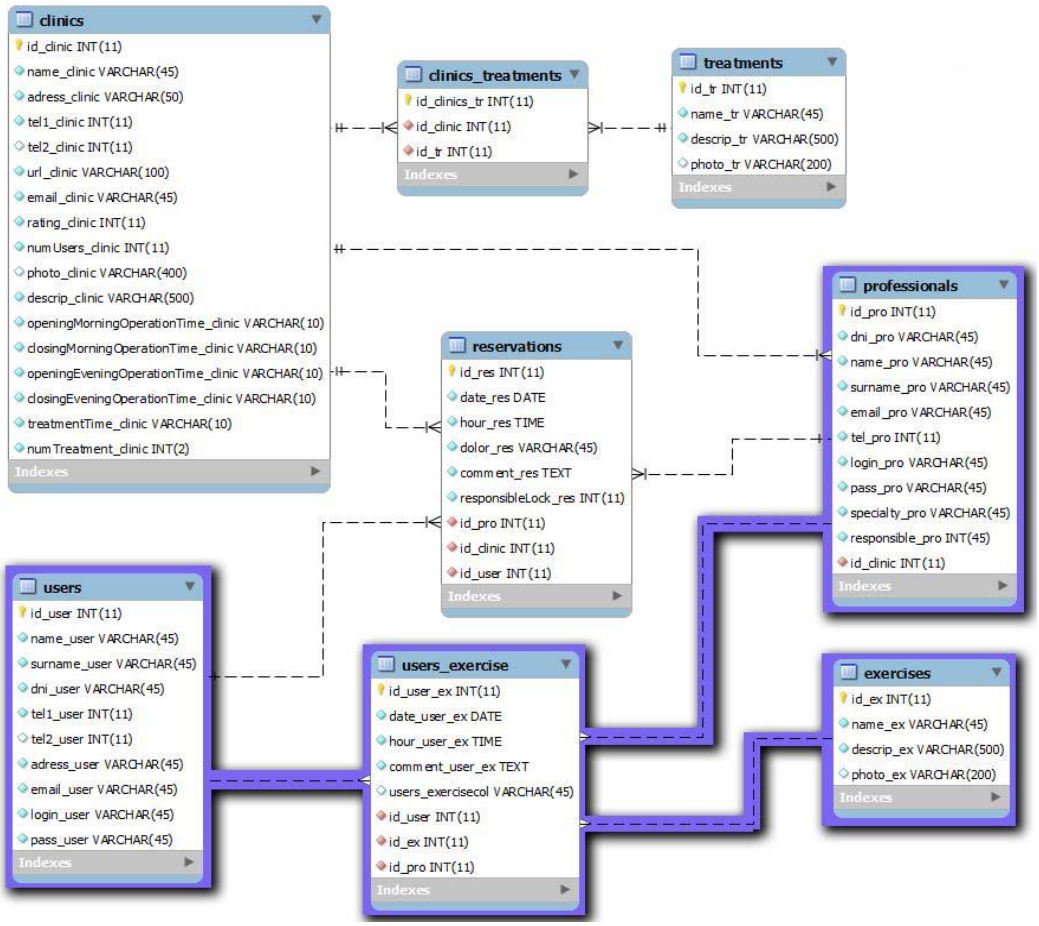

Figura 33 - Relación entre las tablas Professionals, Users y Exercises

4.2.3. Relación entre las tablas *Clinics, Reservations* y *Professionals*

De forma gráfica en la Figura 34Figura 33 se puede observar una relación entre todas las tablas.

En primer lugar, se observa la relación entre las tablas *Clinics* y *Professionals*. Esta relación tiene una cardinalidad de uno a varios, ya que una clínica puede emplear a varios profesionales, pero solamente un profesional puede estar trabajando en una clínica. Así pues, en la tabla *Professionals* se añade el identificador de clínica.

En segundo lugar, la misma relación de cardinalidad (uno a varios) se observa entre las tablas *Clinics* y *Reservations*, ya que una clínica puede tener varias reservas, pero una reserva sólo puede estar asociada a una clínica. Así pues, en la tabla *Reservations* se añade el identificador de clínica.

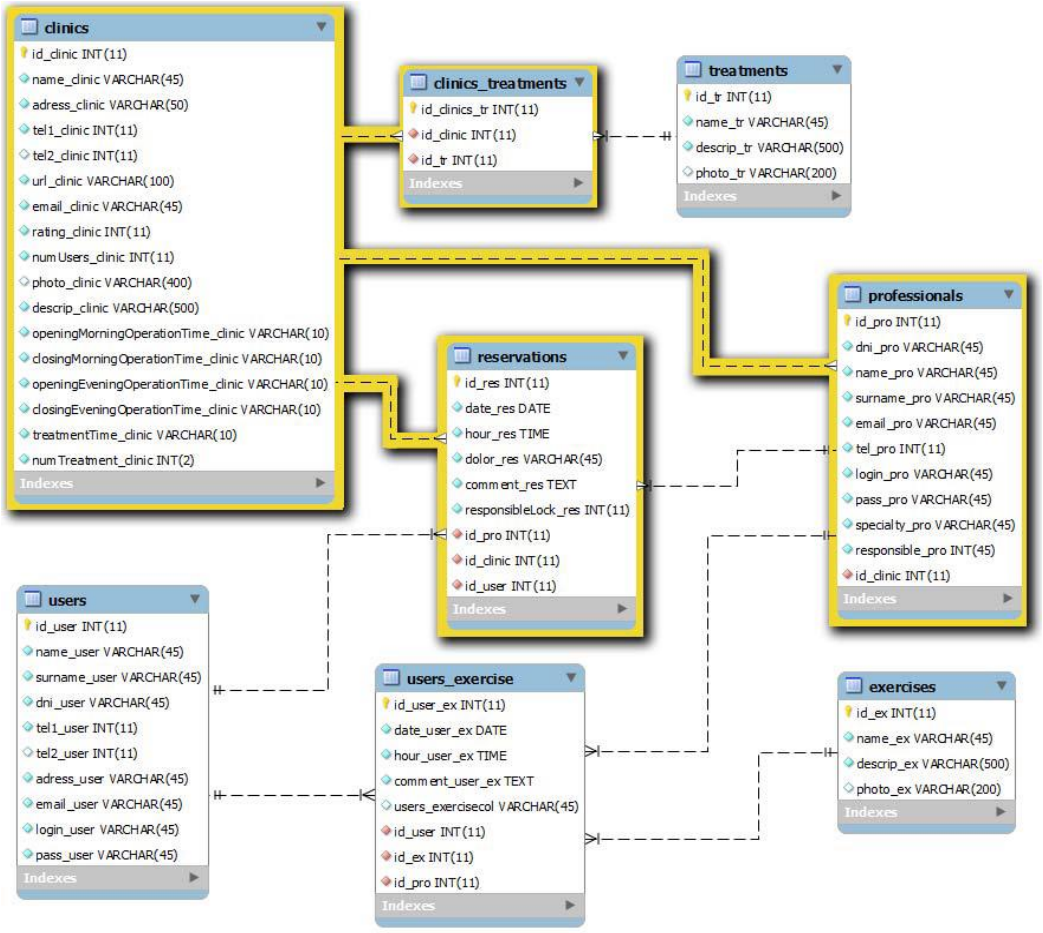

Figura 34 - Relación entre las tablas Clinics, Professionals y Reservations

4.2.4. Relación entre las tablas *Professionals, Reservations y Users_Exercise*

De forma gráfica en la Figura 35Figura 33 se puede observar una relación entre todas las tablas.

En primer lugar, se puede observar una relación entre las tablas de *Professionals* y *Reservations* de cardinalidad 1 a varios, ya que una reserva sólo puede estar asignada a un profesional (que asigna determinados ejercicios a un usuario) pero un profesional puede tener asignadas varias reservas.

En segundo lugar, se observa la otra relación entre las tablas *Professionals* con la tabla *Users_Exercise*, con la misma cardinalidad de 1 a varios, ya que un profesional puede asignar varios ejercicios a un usuario, pero solamente un ejercicio de cada usuario es asignado por un solo profesional.

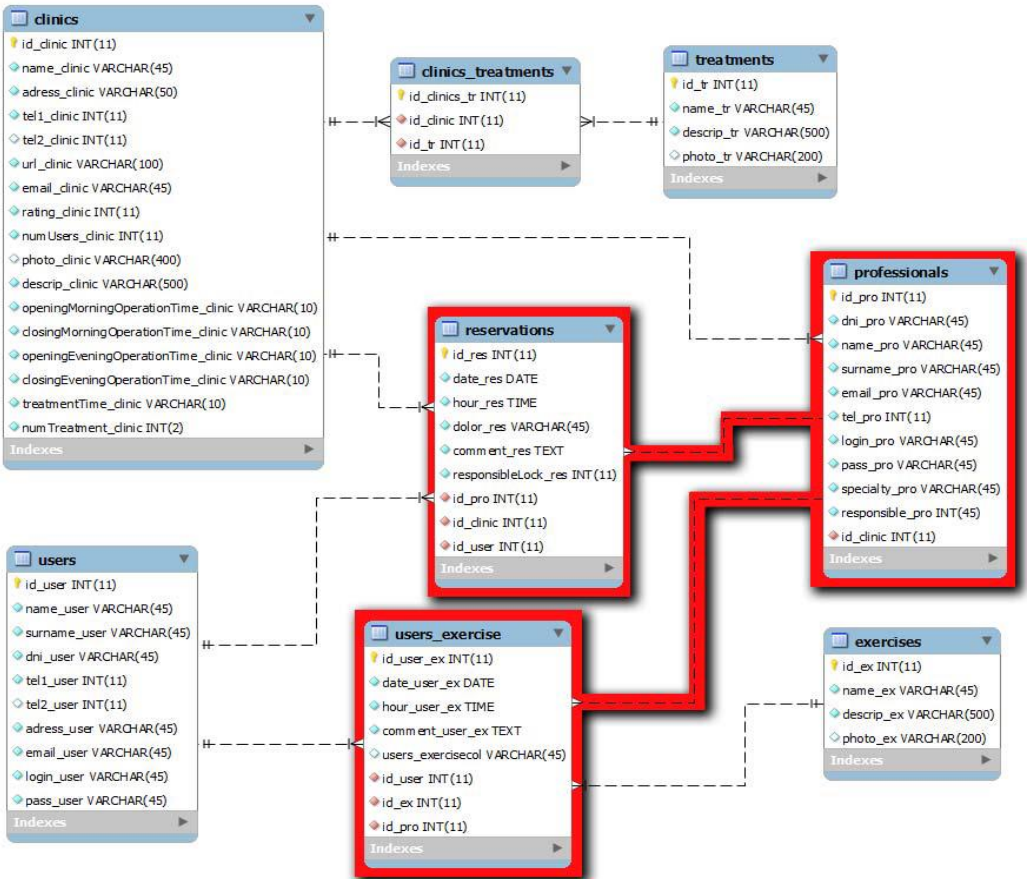

Figura 35 - Relación entre las tablas Professionals, Reservations y Users_Exercise

4.2.5. Relación entre las tablas *Professionals, Reservations, Clinics y Users*

De forma gráfica en la Figura 36Figura 34Figura 33 se puede observar una relación entre todas las tablas.

Se puede observar la relación de la tabla *Reservations* con el resto, ya que necesita, al estar implementado de esta manera, el identificador de la clínica en la cual se ha realizado la reserva, el identificador de usuario el cual ha realizado la reserva y el identificador del profesional asignado por el encargado para la reserva realizada por el usuario. Todas estas relaciones entre la tabla *Reservations* y el resto son de cardinalidad de uno a varios, ya que una reserva solo puede realizarse por un único usuario, asignada a un único profesional y estar planificada para una sola clínica. Sin embargo, un usuario, clínica o profesional pueden tener asignados varias reservas.

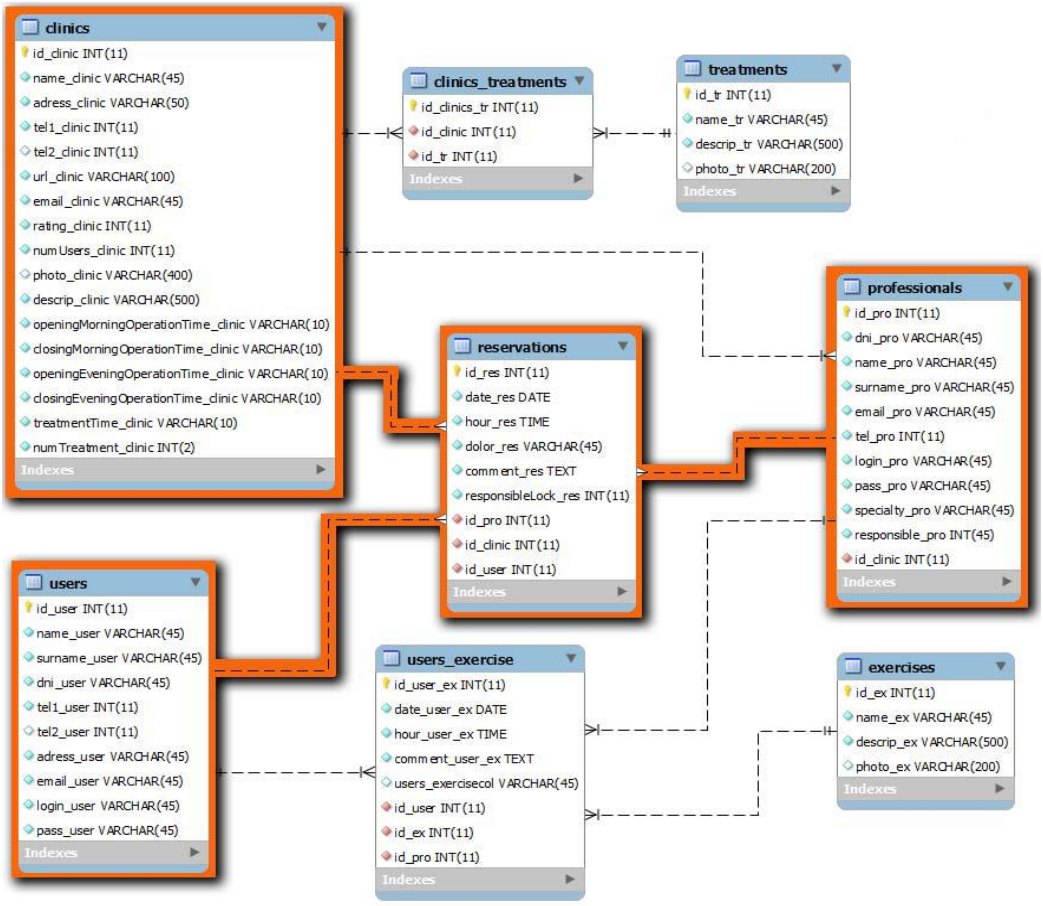

Figura 36 - Relación entre las tablas Professionals, Reservations, Clinics y Users

4.3. Interacción de la aplicación con la base de datos. Modelo de comunicación cliente/servidor.

En este apartado se pretende describir brevemente cuál es la interacción de la aplicación (un extremo de la comunicación) con el lugar donde reside la información para su consulta y modificación, la base de datos (el otro extremo de la comunicación). Se utiliza un modelo de comunicación cliente/servidor, ampliamente conocido.

Para ello, la aplicación enviará peticiones de red HTTP POST a un servidor en el cual unos *webservices*, en forma de scripts escritos en lenguaje PHP, se encargarán de realizar las operaciones pertinentes en la BBDD para obtener o modificar la información del modelo de datos requerida en la petición de red. Esto sería el proceso de comunicación unidireccional desde la aplicación a base de datos.

Por otra parte, en el proceso contrario de comunicación unidireccional desde la base de datos a la aplicación (con el fin de que ésta obtenga la respuesta con la información requerida en la petición de red inicial) serán los *webservices* ejecutándose en el servidor web, los scripts en PHP, los encargados de devolver la información (tras las pertinentes operaciones sobre la base de datos) en el cuerpo de las respuestas HTTP en formato JSON (*JavaScript Object Notation).*

El esquema de toda la interacción de la aplicación con la base de datos se puede resumir en la siguiente Figura 37.

Figura 37 - Esquema del modelo de cliente/servidor de comunicación de la aplicación (Portela, 2015)

4.4. Patrón de diseño de arquitectura de la aplicación

Entre los patrones de diseño de arquitecturas de aplicación para Android, vistos en el apartado 3.6, se ha elegido el patrón MVP para el diseño e implementación de la arquitectura de la aplicación. Este patrón ofrece como ventajas un desacoplamiento entre los componentes lógicos en los que se divide la aplicación (modelo, vista y presentador), lo que permite una mayor mantenibilidad de la misma y facilidad a la hora de realizar *tests*.

Así pues, para la aplicación, podemos distinguir cuál es la relación entre los componentes de la arquitectura MVP con los componentes propios de Android de la aplicación:

- ✓ Modelo de datos:
  Este componente está formado por todas las clases Java que modelan las tablas de información de la base de datos de la aplicación, y almacenan sus datos durante la ejecución de la misma. Así mismo, este modelo de datos proporciona una serie de interfaces mediante las cuáles el presentador podrá acceder a sus métodos para obtener dicha información de la base de datos mediante la implementación de un patrón repositorio de datos (*Repository Pattern*).

- ✓ Presentador:
  Este componente también está diseñado en clases Java y es el encargado principal de toda la lógica y funcionamiento de la aplicación, así como de obtener la información necesaria para ello, su gestión y mantenimiento. Ofrece dos interfaces, tanto al modelo de datos para que éste le pueda notificar las respuestas de las peticiones de red de información a la base de datos, y otra a las vistas de la aplicación, para que ésta le pueda notificar al presentador de las interacciones y eventos del usuario en las diversas funcionalidades de la misma. Cabe destacar que éste es el único componente de la arquitectura que tiene comunicación con las vistas y el modelo de datos de la aplicación, ya que entre éstas últimas no existe comunicación alguna, debiendo gestionar el presentador todas las decisiones en cuánto a actualización de interfaz gráfica, navegación entre pantallas de la aplicación gestión del modelo de datos.

- ✓ Vistas:

  Estos componentes son propios del lenguaje Android y están conceptualizados mediante Actividades y Fragmentos. Una actividad sería la visualización y gestión de una pantalla o varias de la aplicación, mientras que los fragmentos serían pequeñas porciones de esas pantallas con su propio diseño y gestión. El principal objetivo de estas vistas es la visualización al usuario de las funcionalidades de la aplicación mediante una interfaz gráfica, su interacción con las mismas y la notificación de esas interacciones al presentador de la aplicación para realizar el flujo de funcionamiento de las funcionalidades. Ofrece también una interfaz al presentador para que éste le notifique cuando las vistas deben actualizar su aspecto y contenido, de acuerdo a lo que le notifique el presentador.

Con el objetivo de comprender mejor lo anterior, se puede observar, en la Figura 38, la comunicación mediante las interfaces de los componentes en el patrón MVP, para un ejemplo de aplicación de notas.

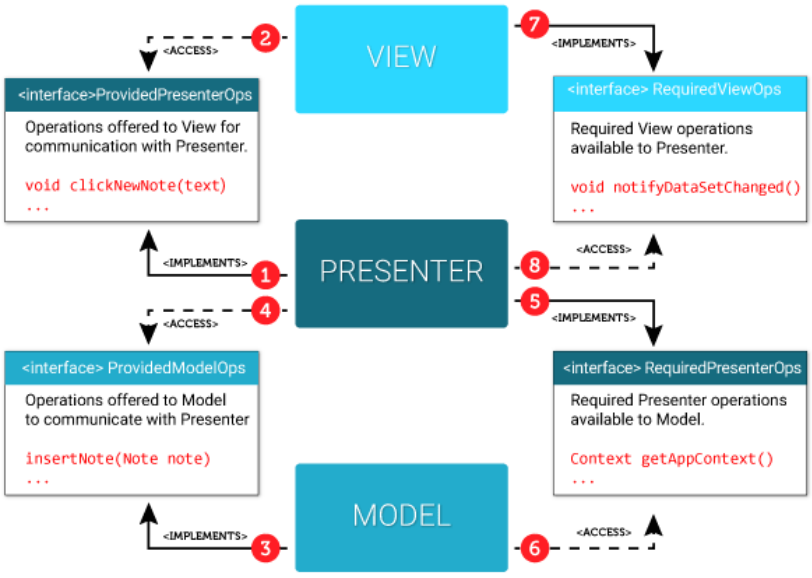

Figura 38 - Ejemplo de interfaces de comunicación en la arquitectura MVP (Megali, 2016)

4.5. Funcionalidades técnicas de la aplicación

En este apartado se describen las funcionalidades propias de la aplicación desde un punto de vista técnico.

Cabe recordar que hay tres tipos de usuarios/perfiles con los que se puede usar esta aplicación: usuario normal como paciente, profesional o encargado de clínica. Para este proyecto de aplicación solamente están implementadas las funcionalidades más relevantes, es decir, todas las que puede hacer uso de la aplicación como paciente, así como algunas del perfil de un profesional, como asignar ejercicios a pacientes (así como vídeos de los mismos) o el listado de tratamientos disponibles en la clínica.

Para realizar la descripción de la funcionalidad se describirá cada una de ellas mediante la ayuda de dos tipos de diagramas de flujo, uno para explicar desde el punto de vista técnico la clase que modela cada una de las funcionalidades a describir, y otro en que se detalla el flujo de operaciones que componen dicha funcionalidad y están gestionadas por la clase que modela la misma. Puede haber casos de funcionalidad en las que el primero de los mismos es suficiente para describir la misma.

A continuación, se describen las funcionalidades pertenecientes al perfil de usuario como paciente (a excepción de la funcionalidad de *login*, común para todos los perfiles):

4.5.1. Funcionalidad de *Login*

El objetivo de esta funcionalidad es la de autentificar de manera correcta un usuario registrado en la aplicación, ya sea con un perfil como paciente, profesional o encargado. Se trata de la primera pantalla que se ve en la aplicación nada más iniciarla, a no ser que el usuario ya se encuentre autentificado con anterioridad.

Si el usuario se autentifica correctamente se le muestra la pantalla principal, personalizada para cada tipo de perfil autentificado con las funcionalidades disponibles en función del mismo. Para ello, la aplicación envía una petición HTTP POST con la información de nombre de usuario y clave de acceso para su posterior comprobación al *webservice* correspondiente. Será este mismo el que le devuelva a la aplicación una respuesta con un código de éxito o error y, en el caso de la primera

situación, se envía además un código de perfil que identifica el tipo de usuario autentificado (paciente, profesional o encargado). En función de este código de perfil, se muestra la pantalla principal de la aplicación personalizada para cada tipo de perfil en función del tipo de operaciones/funcionalidades permitidas para el mismo. En caso contrario, cuando la respuesta devuelve un código de error, se muestra un mensaje de error al usuario informándole de que el usuario con los datos introducidos de autentificación no se encuentra registrado en el sistema.

A continuación, se presentan los diagramas de flujo de la clase que gestiona la funcionalidad y de las operaciones que se realizan durante el flujo de la misma (Figura 39 y Figura 40).

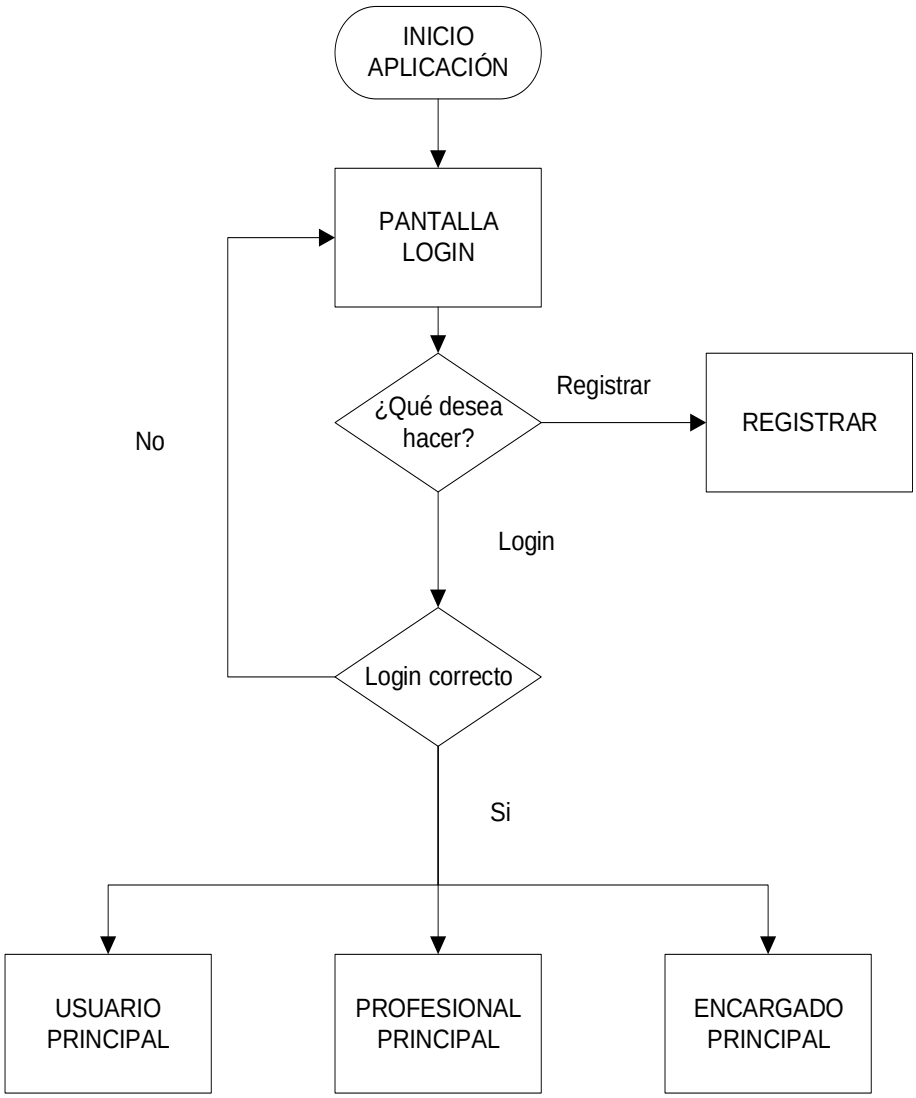

Figura 39 - Diagrama de flujo de la clase que gestiona la funcionalidad de Login

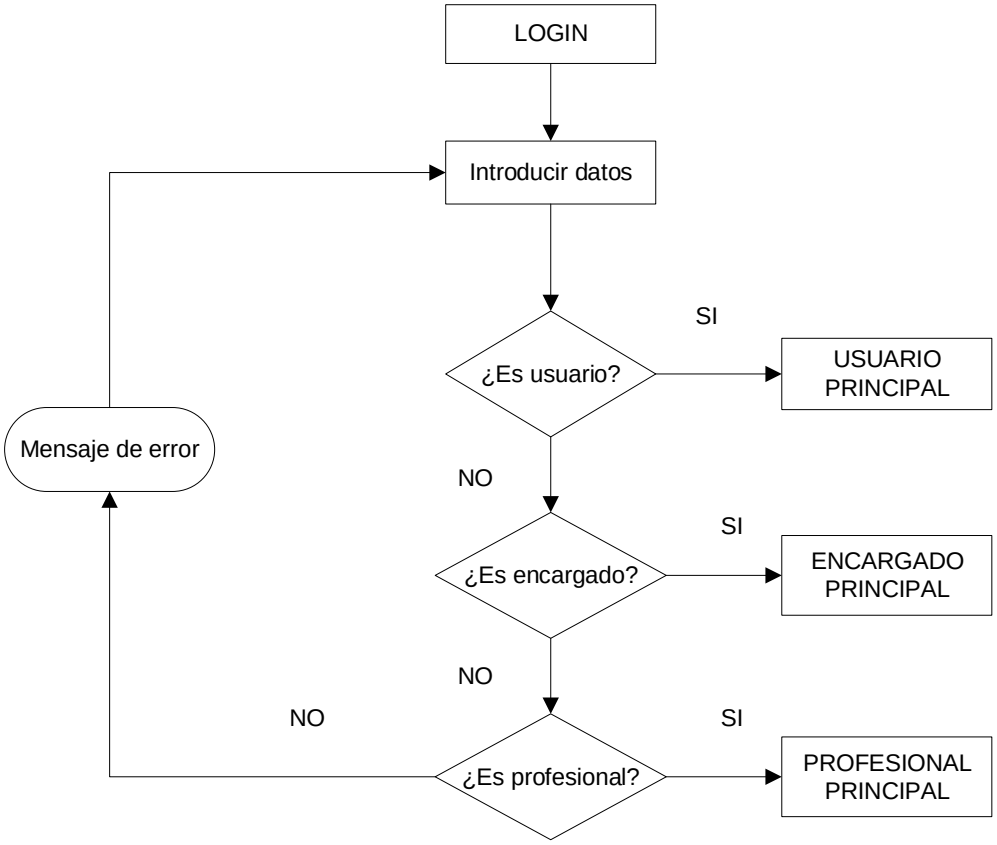

Figura 40 - Diagrama de flujo de la funcionalidad de Login

4.5.2. Funcionalidad de Registro de usuario

El objetivo de esta funcionalidad es la de registrar en el sistema un usuario con el perfil de paciente para que pueda hacer uso de las funciones propias de la aplicación con este tipo de perfil. Esta funcionalidad se integra en la misma primera pantalla que la funcionalidad de *login*.

Para realizar el registro, el usuario tiene que cumplimentar una serie de campos obligatorios y opcionales. Una vez la aplicación ha validado el formato y tipo de datos cumplimentados (obligatorios y alguno opcional) son los correctos, ésta realiza una petición HTTP POST al *webservice* correspondiente con dichos datos. La respuesta puede ser exitosa, devolviendo el usuario a la pantalla de *login* con los nuevos datos de nombre de usuario y clave de acceso elegidos para el registro. En caso de error, la respuesta puede devolver un código de error que indique que el nombre de usuario elegido para el registro ya está registrado. Para esta situación, la aplicación le informa al usuario de que debe de elegir un nuevo nombre de usuario.

A continuación, se presenta el diagrama de flujo de las operaciones que se realizan durante el flujo de la funcionalidad, en la Figura 41. El diagrama de flujo de la clase que gestiona esta funcionalidad es el mismo que en la Figura 39.

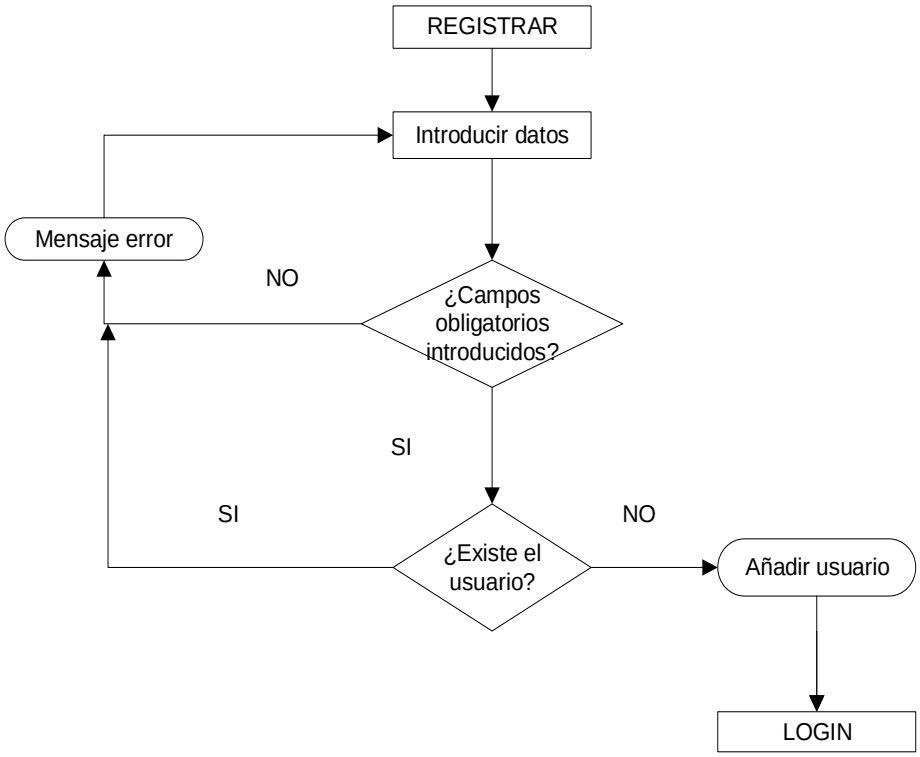

Figura 41 - Diagrama de flujo de la funcionalidad de Registro de usuario

4.5.3. Funcionalidades de la pantalla principal

Esta pantalla es la principal de la aplicación, desde la cual, con los diferentes perfiles de usuario posibles de autentificarse en la aplicación, cada uno de ellos puede realizar diferentes operaciones. En este apartado, se enumerarán las funcionalidades principales a las que es posible acceder con los perfiles de paciente y profesional de clínica, entre ellas, la común de poder cerrar sesión para poder autentificarse con otro perfil diferente:

- ✓ Perfil de usuario ó paciente: Dentro del mismo el paciente puede realizar consultas sobre las distintas clínicas de fisioterapia disponibles, toda su información como localización, contacto,

cómo llegar a la misma, su lista de tratamientos, etc. Además, el usuario es capaz de realizar reservas en cada una de las mismas, especificando el motivo de la misma y fecha deseadas, así como poder consultarlas posteriormente e incluso cancelarlas. También, el paciente es capaz de revisar los ejercicios que el profesional de clínica le ha asignado previamente, con la posibilidad de visualizar vídeos de los mismo si el profesional así lo hubiese considerado. En la Figura 42 se puede observar el diagrama de flujo de la clase que gestiona las funcionalidades de la pantalla principal de la aplicación en este perfil.

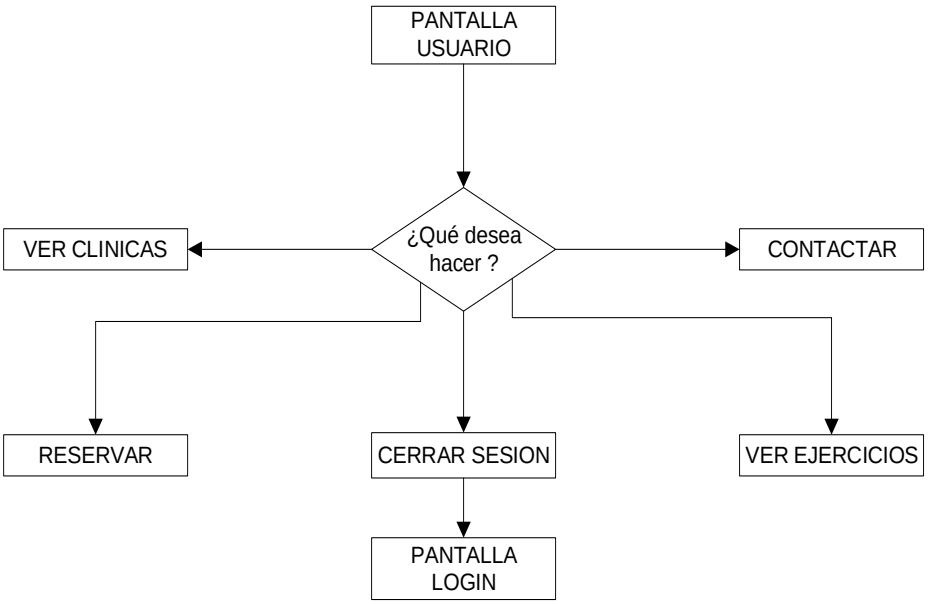

Figura 42 - Diagrama de flujo de la clase que gestiona las funcionalidades de la pantalla principal con perfil de paciente

- ✓ Perfil de profesional de clínica: Para este perfil el profesional de clínica puede consultar los datos de la clínica para la que trabaja, así como sus tratamientos y poder contactar con la misma. Adicionalmente, es posible realizar la asignación de ejercicios de fisioterapia a sus pacientes asignados, con la posibilidad de añadir vídeos de ejercicios (para aquellos ejercicios que dispongan de los mismos) para que los pacientes puedan disponer de ellos en una fecha determinada en función del criterio del profesional. En la Figura 43 se puede observar el diagrama de flujo de la clase que gestiona las funcionalidades de la pantalla principal de la aplicación en este perfil.

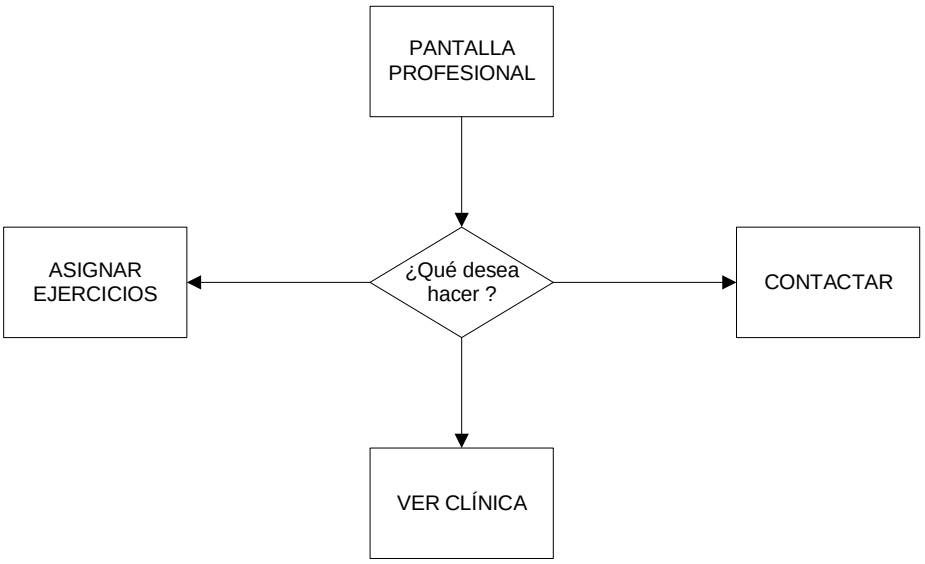

Figura 43 - Diagrama de flujo de la clase que gestiona las funcionalidades de la pantalla principal con perfil de profesional de clínica

A continuación, se describirán cada una de las funcionalidades y pasos de las mismas necesarios para realizarlas. Las funcionalidades de cerrar sesión y contactar no se describen porque son muy intuitivas y no requieren de interacción de la aplicación con el servidor. Únicamente la funcionalidad de Cerrar Sesión elimina el registro de usuario autentificado de la aplicación y se muestra la pantalla inicial de *login*; y la de Contactar abre la aplicación de correo electrónico para la composición de un *email* para obtener más información acerca de la clínica seleccionada en el caso del paciente, o de la clínica de trabajo en caso del profesional.

4.5.4. Funcionalidad de Ver Clínica(s)

Esta funcionalidad está presente tanto para el perfil de paciente como de profesional de clínica. La diferencia entre ambos perfiles radica en que para el perfil de paciente el usuario puede ver todas las clínicas de fisioterapia disponibles en la aplicación para consultar toda su información de ubicación, contacto, así como su lista de tratamientos. En cambio, para el perfil de profesional de clínica, el propósito de la funcionalidad es poder obtener la información de la clínica en la que trabaja el propio profesional para su posterior consulta. En resumen, se trata de la misma funcionalidad para ambos perfiles, pero mostrando un número diferente de clínicas según los mismos.

Para ello, se realiza una petición HTTP POST al *webservice* correspondiente para la obtención de dichos datos, sin que sea necesario el envío de dato alguno. En el caso de respuesta exitosa (código HTTP de la respuesta del servidor 200), ésta incluirá una lista de todas las clínicas disponibles que se mostrarán por pantalla (esta lista podría ser vacía, en cuyo caso se informaría al usuario con un mensaje por pantalla). Una vez tenemos la lista de las clínicas disponibles, pulsando en cada una de ellas se obtienen los detalles de la misma (en una nueva pantalla con los detalles propios de la clínica) mediante una petición similar a la de obtención del listado de clínicas. En la respuesta de la petición se reciben los datos de contacto, ubicación de la misma, la posibilidad de lisar sus tratamientos, etc.… que se presentan al usuario para que le sea posible consultarlos e interaccionar con los mismos. Por último, en la pantalla de detalle de cada clínica se ofrece la posibilidad de compartir la misma mediante una opción en pantalla.

A continuación, se presenta el diagrama de flujo de las operaciones que se realizan durante el flujo de la funcionalidad, en la Figura 44. El diagrama de flujo de la clase que gestiona esta funcionalidad se presenta en la Figura 45.

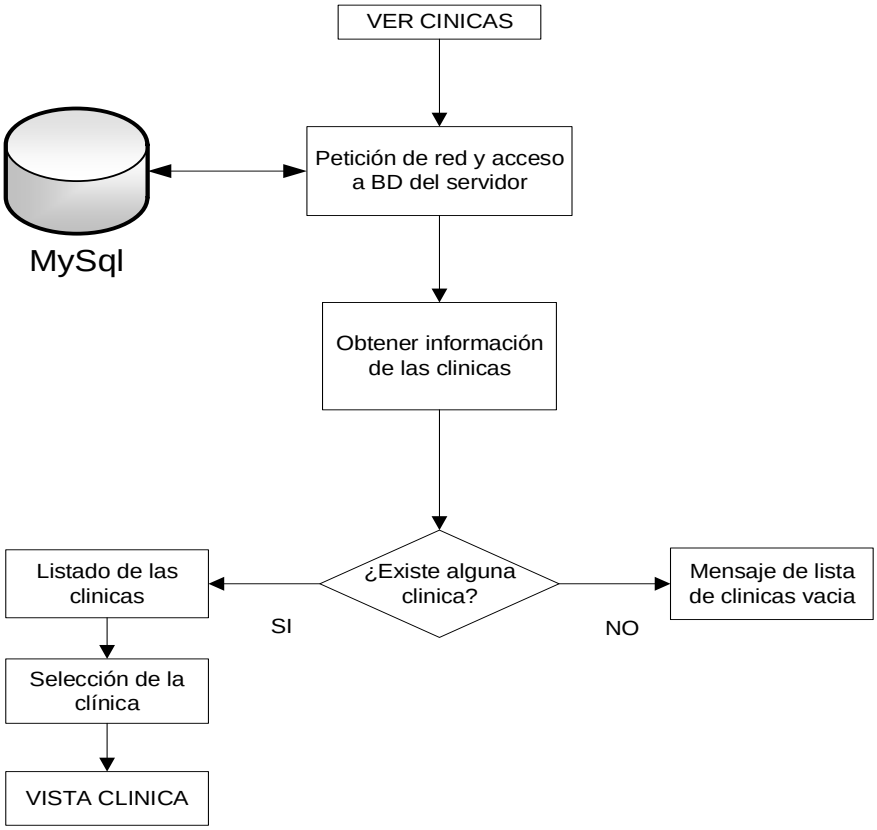

Figura 44 - Diagrama de flujo de la funcionalidad de Ver Clínicas

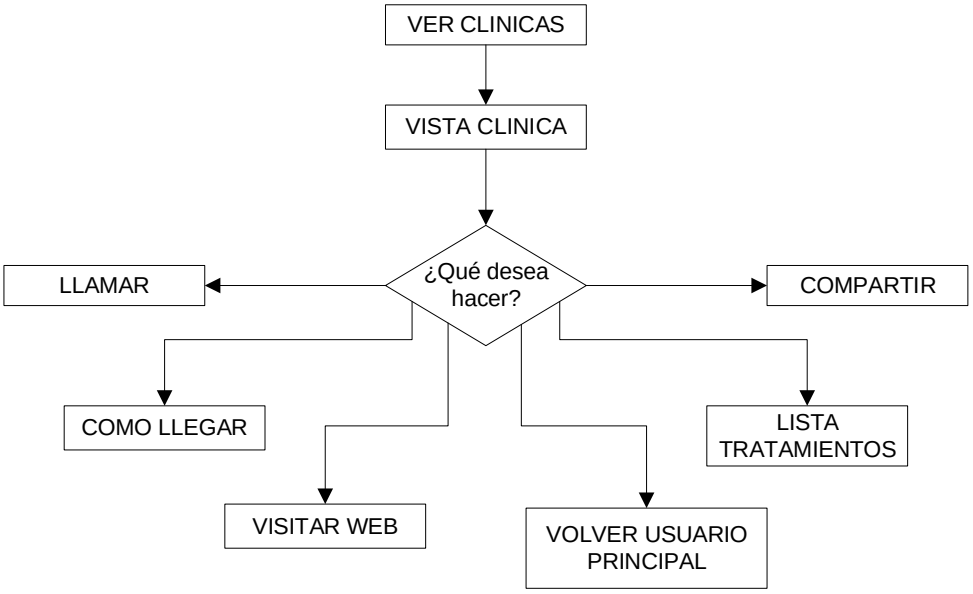

Figura 45 - Diagrama de flujo de la clase que gestiona la funcionalidad de Ver Clínicas

4.5.5. Funcionalidad de Ver Tratamientos de una Clínica

Para llegar a esta funcionalidad, se parte de la anterior (Ver Clínicas) en la cual se selecciona una clínica de la lista (en caso de no estar vacía) y entre la información disponible de la misma existe una opción para poder visualizar los tratamientos fisioterapéuticos que se realizan en la clínica seleccionada.

Para ello, se realiza una petición HTTP POST al *webservice* correspondiente para la obtención de la lista de tratamientos de la clínica, enviándose como dato de entrada el identificador de la clínica seleccionada. En el caso de respuesta exitosa, se obtiene una lista con toda la información de los diversos tratamientos disponibles en la clínica, como su nombre, foto y descripción. Esta lista se muestra por pantalla y en el caso de querer obtener más información del tratamiento seleccionado en dicha lista, se muestra una nueva pantalla de detalle con información como la descripción del mismo. En el caso de que obtengamos una lista vacía de tratamientos, se informaría al usuario con un mensaje por pantalla. Por último, en la pantalla de detalle de cada tratamiento se

ofrece la posibilidad de compartir el mismo mediante una opción en pantalla.

A continuación, se presenta el diagrama de flujo de las operaciones que se realizan durante el flujo de la funcionalidad, en la Figura 46. El diagrama de flujo de la clase que gestiona esta funcionalidad se presenta en la Figura 47.

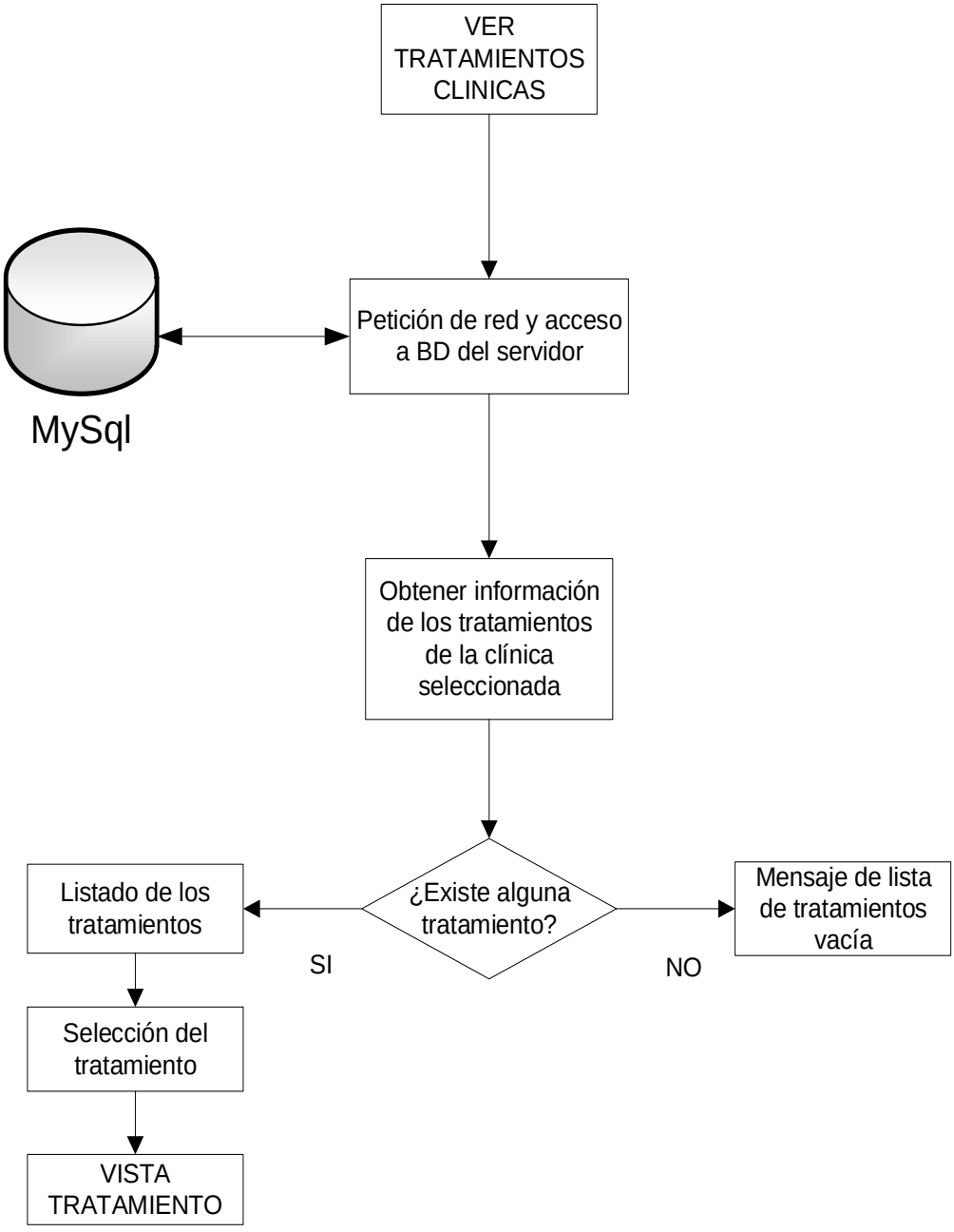

Figura 46 - Diagrama de flujo de la funcionalidad de Ver Tratamientos de una Clínica

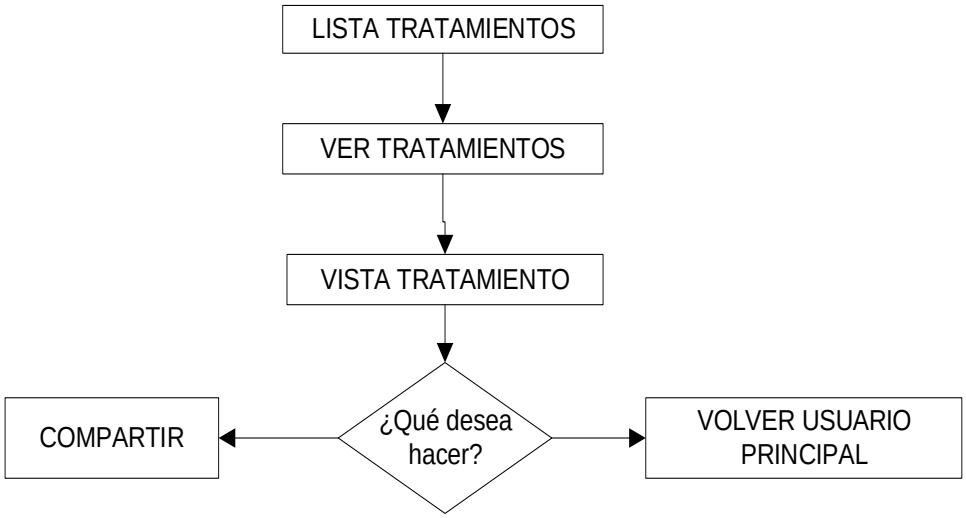

Figura 47 - Diagrama de flujo de la clase que gestiona la funcionalidad de Ver Tratamientos de una Clínica

4.5.6. Funcionalidad de Reservar cita en una Clínica

Esta funcionalidad es accesible para los usuarios con perfil de paciente y se puede seleccionar desde la pantalla principal del usuario con dicho perfil. Tiene como objetivo el poder ofrecer al usuario la posibilidad de reservar o concertar una cita con la clínica que el paciente elija, mediante la introducción del motivo de la consulta (localización del dolor y el tipo o descripción del mismo) y la fecha deseada para la misma.

Para realizar todo este proceso, el paciente selecciona en su menú principal la opción de reservar cita. A continuación, se le mostrará una nueva pantalla en la cual la aplicación mostrará todo el listado de clínicas disponibles para reservar cita mediante una petición HTTP POST al *webservice* correspondiente (la misma petición que en el apartado 4.5.4 de visualizar las clínicas). Una vez listadas las clínicas por pantalla, el usuario debe elegir la clínica para la cual quiere reservar la cita. Una vez seleccionada, se muestra una segunda pantalla del flujo de la funcionalidad en forma de formulario de recogida de datos para el establecimiento de la cita. En este formulario se necesita que el usuario introduzca la localización y descripción del dolor que motiva la necesidad de la cita, así como una fecha y hora deseados para la misma (esta fecha y hora siempre será futura a la actual ya que la aplicación no permite seleccionar fechas pasadas o igual a la actual). Todos estos campos son obligatorios de rellenar. Una vez completados los mismos, el usuario

puede proceder a reservar la cita. En este punto es cuando la aplicación realiza una nueva petición HTTP POST al *webservice* correspondiente, mediante el envío de los datos anteriores del formulario rellenados por el paciente, para comprobar la disponibilidad de la cita solicitada por el mismo en la clínica seleccionada anteriormente.

Para este proceso de comprobación de disponibilidad de la cita se comprueban varios aspectos en el lado del servidor:

- ✓ En primer lugar, se comprueba si la fecha y hora introducida por el paciente en el formulario de datos de reserva de la cita existe y es posterior a la actual. Esta última condición siempre será verdadera ya que es la aplicación la que permite únicamente la selección de fechas futuras a la actual para la reserva de la cita.

- ✓ En segundo lugar, se comprueba si existe un horario definido para la clínica seleccionada. En caso contrario se establecen unos valores por defecto (Rango de mañana de 10:00-14:00 y de tarde de 17:00-20:30).

- ✓ En tercer lugar, se comprueba si el día escogido por el paciente resulta ser laborable o festivo para la operativa de la clínica. Si el día seleccionado por el paciente resulta ser festivo, el servidor siempre avanzará esa fecha hasta el próximo día laborable más cercano.

- ✓ En cuarto lugar, tras la comprobación de la fecha, se comprueba la hora introducida por el paciente. Ésta siempre debe de estar dentro de la franja horaria de apertura y cierre de la clínica seleccionada en los pasos previos. En el caso de que así no sea, se procederá a avanzar al siguiente rango horario. En el caso de que el avance de rango horario implique un nuevo día de selección se procedería a volver al paso anterior para la comprobación de la fecha.

- ✓ Por último, el servidor dispone de un *array* de *slots* temporales en función del horario de apertura y cierre de la clínica y el tiempo que conlleva cada cita/tratamiento. En ese *array* se pueden comprobar que *slots* están ya reservados y cuáles están disponibles para la cita. Cabe mencionar que un *slot* temporal sólo estará reservado completamente cuando todos los profesionales

de la clínica disponibles para la realización de tratamientos tengan concertada una cita con un paciente en ese mismo periodo de tiempo del *slot*, que será el tiempo medio de duración del tratamiento. Tanto el número de tratamientos a realizar simultáneamente como el tiempo medio de cada tratamiento son campos disponibles en la base de datos en la tabla *Clinics*.

Una vez realizadas las comprobaciones anteriores en el lado del servidor, éste envía una respuesta a la petición de solicitud de cita, en la que se propone la fecha más cercana posible a la actual (o la misma solicitada por el paciente en caso de que estuviese disponible) y se le pregunta al usuario por su conformidad. En caso afirmativo, se registra la reserva de la cita mediante una petición HTTP POST al *webservice* correspondiente con la fecha y hora propuesta por el servidor para la clínica seleccionada (en caso de registro con éxito se volvería a la pantalla principal del paciente y en caso de error se volvería a la misma pantalla del formulario de reserva). En caso de disconformidad con la fecha propuesta por el servidor, se volvería a la pantalla del formulario en la que el usuario tendría que elegir una nueva fecha y hora distinta a la anterior para la posible cita y se volvería a iniciar el proceso de confirmación y comprobación de la disponibilidad de la cita comentados anteriormente.

A continuación, se presenta el diagrama de flujo de las operaciones que se realizan durante el flujo de la funcionalidad, en la Figura 48. El diagrama de flujo de la clase que gestiona esta funcionalidad se presenta en la Figura 49.

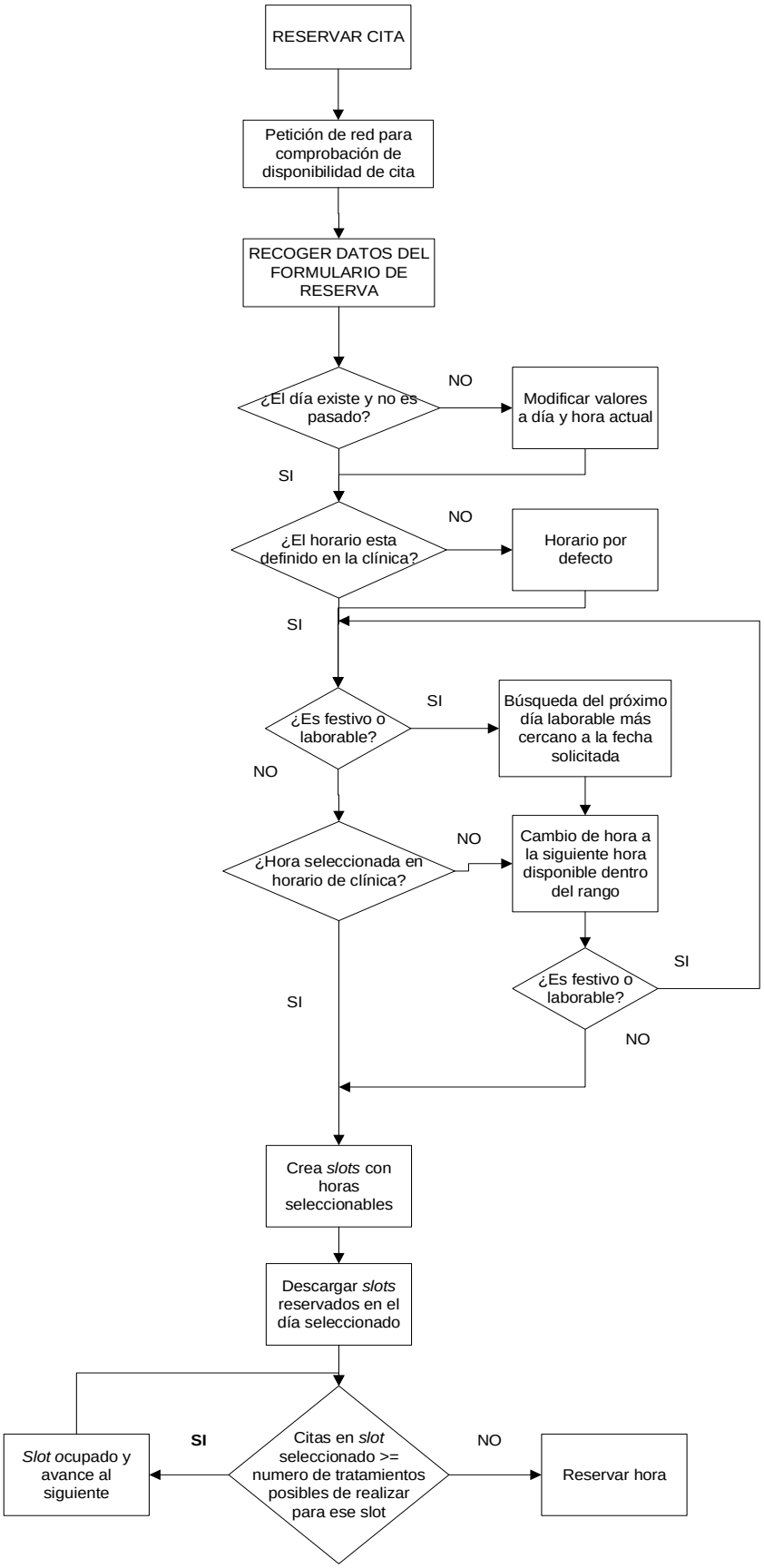

Figura 48 - Diagrama de flujo de la funcionalidad de Reservar cita en una Clínica

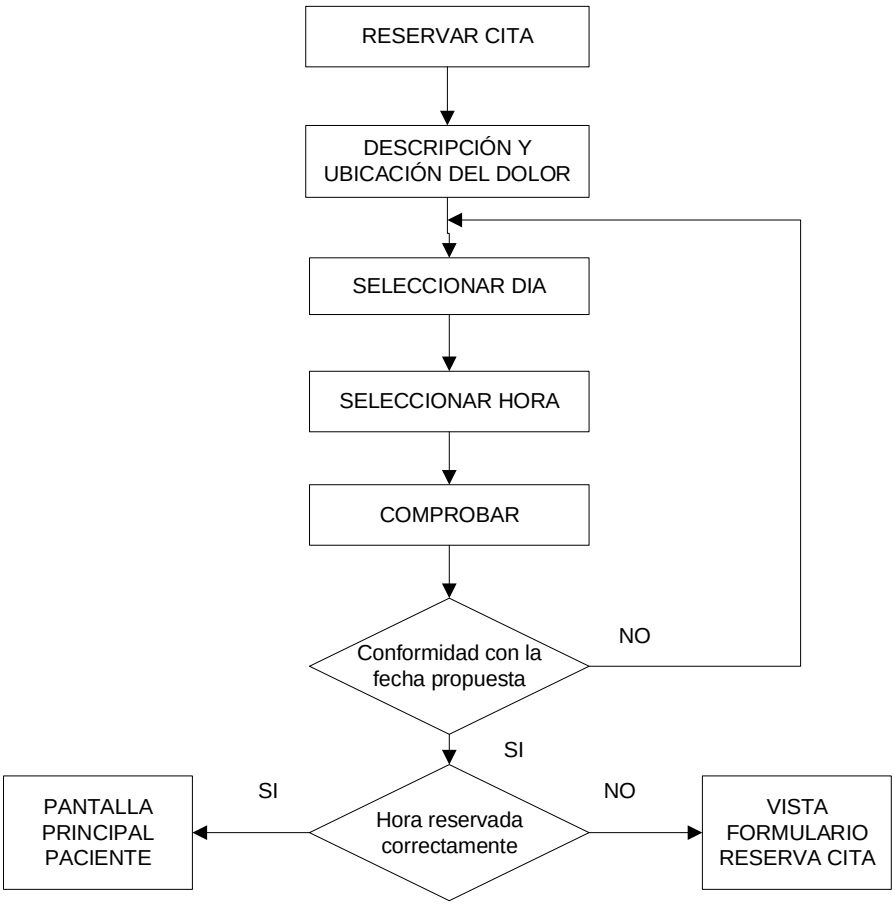

Figura 49 - Diagrama de flujo de la clase que gestiona la funcionalidad de Reservar una cita en una Clínica

4.5.7. Funcionalidad de Ver Ejercicios asignados a un Paciente

Esta funcionalidad es similar a la de ver los tratamientos de una clínica en cuánto al flujo de operaciones que se llevan a cabo, salvo la diferencia s de que los tratamientos están asociados a una clínica y los ejercicios a un paciente, que los ejercicios del paciente pueden tener asociados un enlace a un vídeo de instrucciones sobre el mismo que el usuario podrá visualizar de forma externa de la aplicación, y que esta funcionalidad es exclusiva del perfil de usuario de paciente. Por lo tanto, no se va a describir en profundidad la funcionalidad debido a la similitud con la funcionalidad del apartado 4.5.5 mencionada anteriormente.

A continuación, se presenta el diagrama de flujo de las operaciones que se realizan durante el flujo de la funcionalidad, en la Figura 50. El diagrama de flujo de la clase que gestiona esta funcionalidad se presenta en la Figura 51.

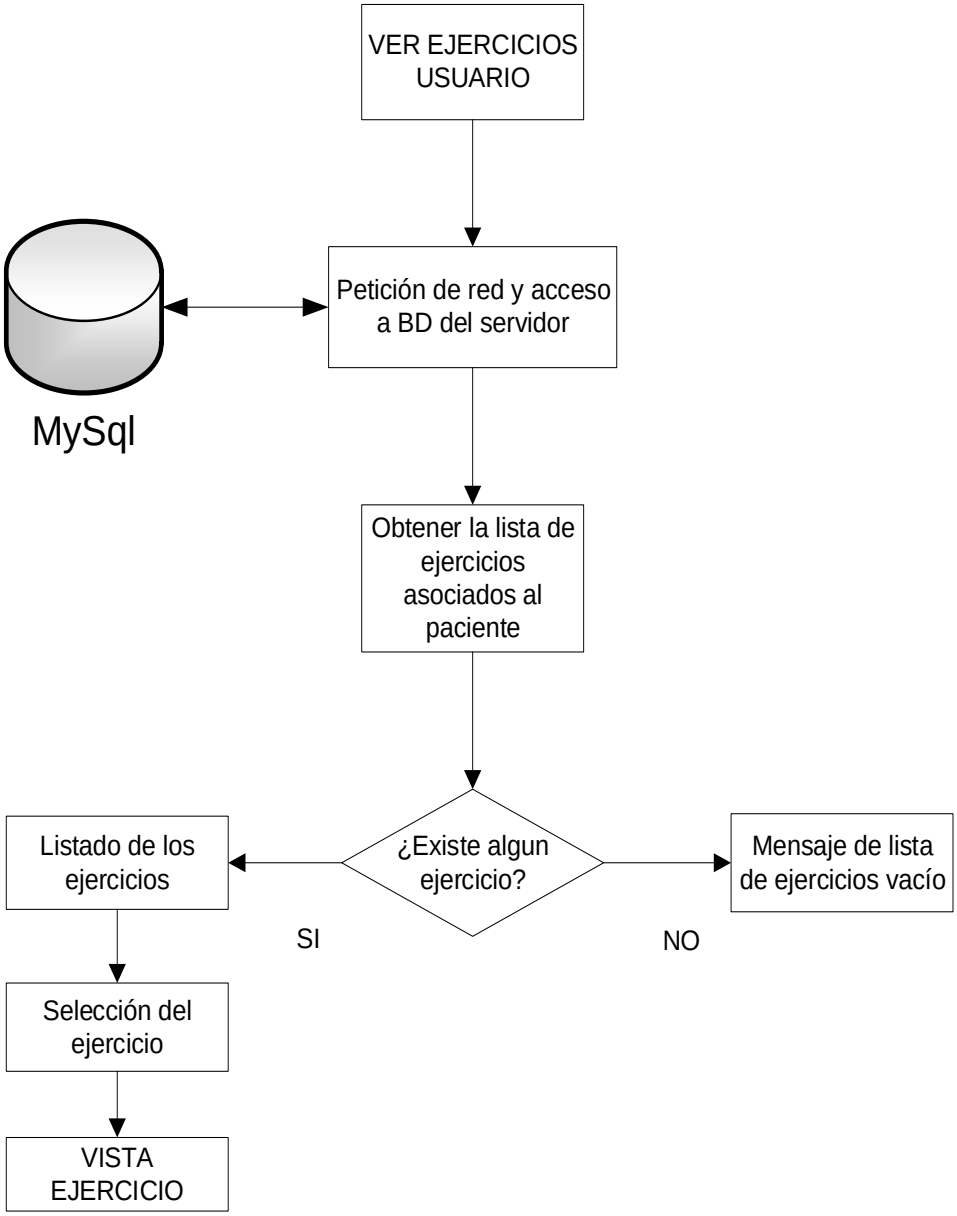

Figura 50 - Diagrama de flujo de la funcionalidad de Ver los ejercicios asociados a un Paciente

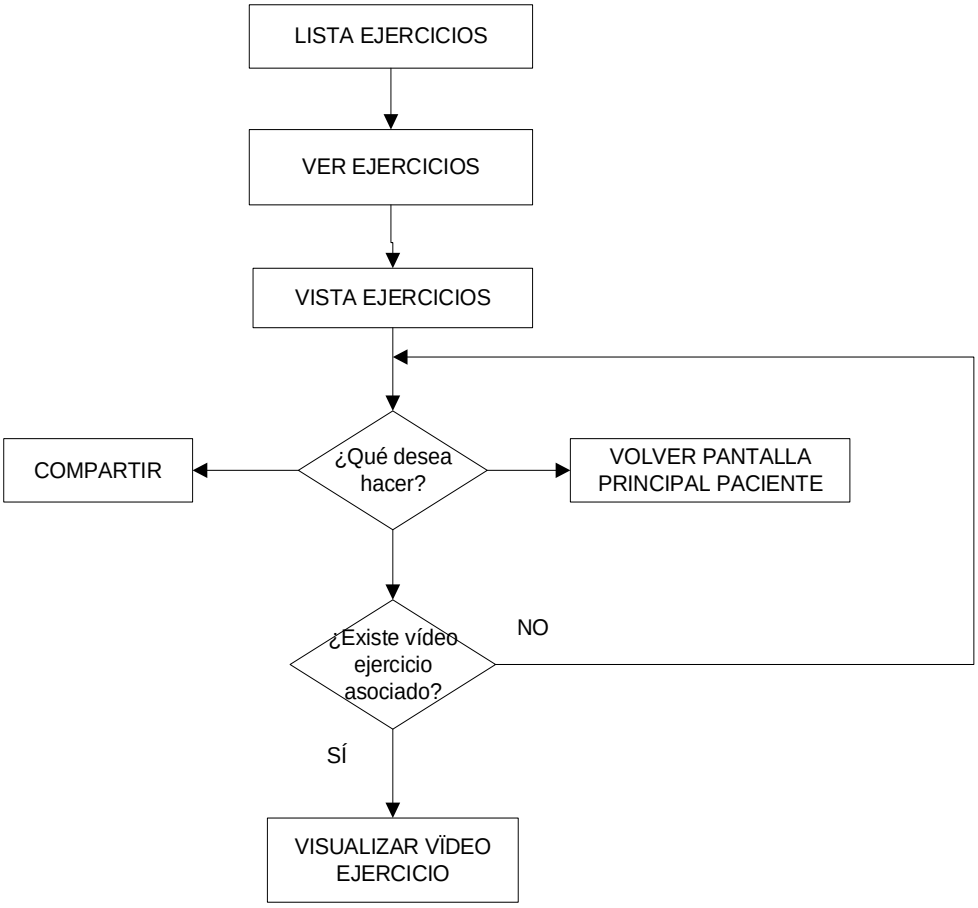

Figura 51 - Diagrama de flujo de la clase que gestiona la funcionalidad de Ver ejercicios asignados a un Paciente

4.5.8. Funcionalidad de Gestionar Reservas de un Paciente

Esta funcionalidad tiene como objetivo permitir al paciente visualizar las reservas realizadas en las diferentes clínicas, con la información de la fecha y hora de las mismas, así como la posibilidad de cancelar las mismas si así lo desea. Solamente pueden acceder a la misma los perfiles de usuario de tipo pacientes.

Para obtener la información de todas las reservas asociadas al usuario se realiza una petición HTTP POST al *webservice* correspondiente con la información del identificador de usuario asociadas a las mismas. En caso de respuesta exitosa, se devuelve una lista (pudiendo estar vacía, en cuyo caso se mostraría un mensaje sobre ello al usuario) con las reservas realizadas. Las mismas contienen información acerca de la clínica asociada y su fecha y hora de la cita. Al lado de esa información aparecerá un botón

de cancelación que le informará al usuario si desea cancelarla. En caso afirmativo, se realiza una petición HTTP POST al *webservice* correspondiente para eliminar el registro de la reserva de la base de datos del servidor. En caso contrario, se vuelve a la pantalla del listado de reservas de usuario.

A continuación, se presenta el diagrama de flujo de las operaciones que se realizan durante el flujo de la funcionalidad, en la Figura 52Figura 50. El diagrama de flujo de la clase que gestiona esta funcionalidad se presenta en la Figura 53.

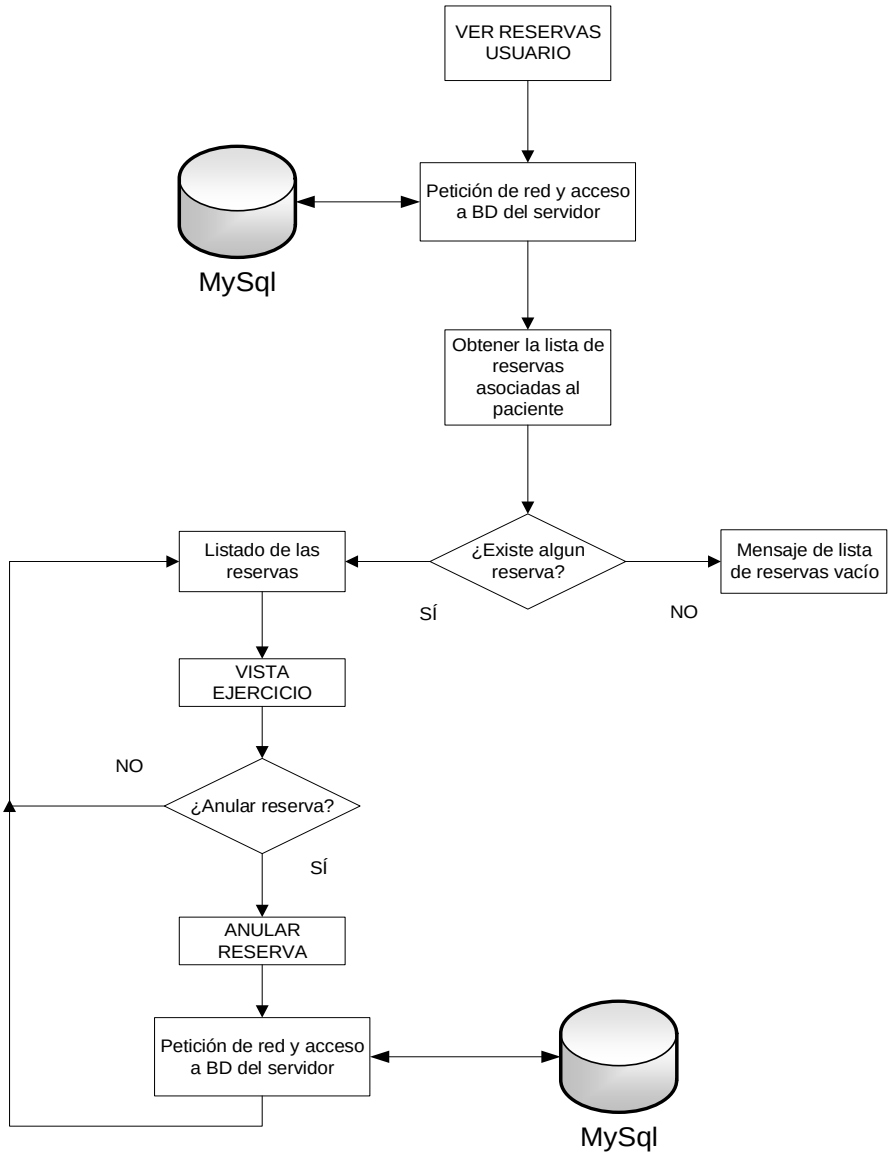

Figura 52 - Diagrama de flujo de la funcionalidad de Gestionar Reservas de un Paciente

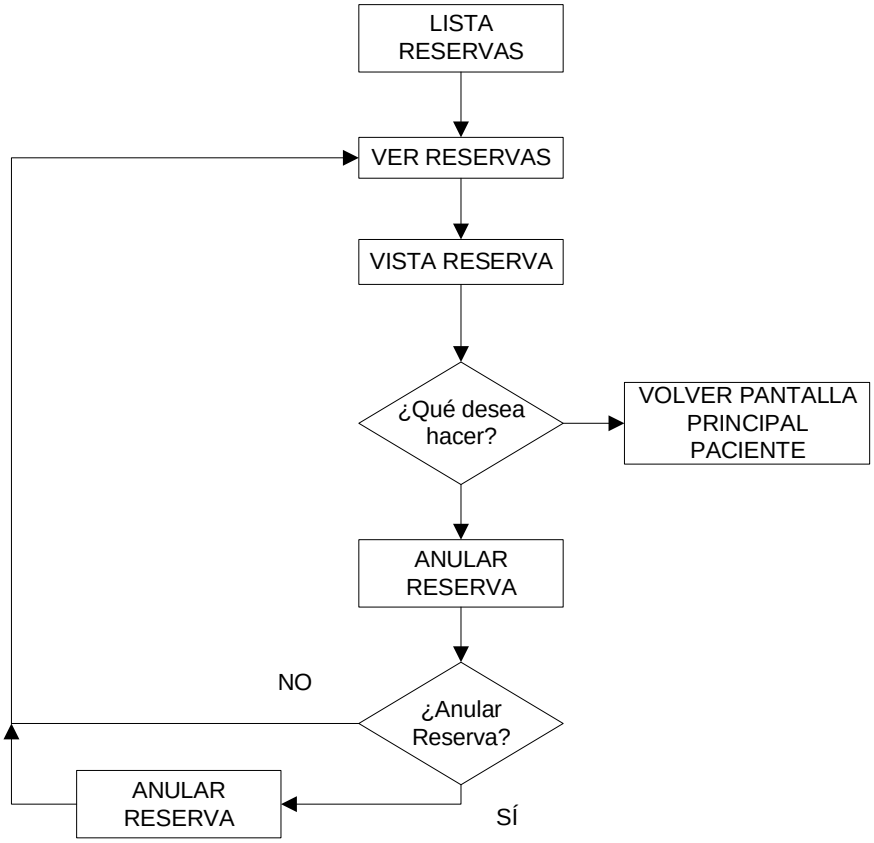

Figura 53 - Diagrama de flujo de la clase que gestiona la funcionalidad de Gestionar Reservas de un Paciente

4.5.9. Funcionalidad de Asignar Ejercicios a un Paciente

Esta funcionalidad tiene como objetivo por parte del profesional de clínica asignar una serie de ejercicios a cada uno de sus pacientes (asignados previamente por el encargado de la clínica) para su tratamiento de fisioterapia, con el añadido de poder asignar un vídeo de instrucciones de los mismos para que estén disponibles al paciente en una fecha determinada elegida por el profesional. Solamente se puede acceder a esta funcionalidad como perfil de usuario profesional de clínica.

Para ello, este flujo constará de dos partes:

- ✓ En primer lugar, se mostrará una pantalla con la lista de los pacientes asignados al profesional de clínica autentificado. Para ello se realizará una petición HTTP POST al *webservice* correspondiente, el cuál devolverá en caso de respuesta exitosa una lista de los pacientes asignados (pudiendo ésta estar vacía, en ese caso se mostraría un mensaje al profesional). Una vez se

muestre la lista por pantalla con la información del nombre de cada paciente y el tipo de dolor asociado (proveniente de la reserva realizada por el paciente anteriormente). El profesional debe seleccionar un paciente para asignarle los ejercicios.

- ✓ En segundo lugar, se mostrará una nueva pantalla con toda la lista de ejercicios disponibles a realizar. Se incluye tanto una fotografía de los mismos, como su descripción y nombre. Adicionalmente, si el profesional así lo desea y lo considera, puede añadir unos comentarios a los ejercicios que puedan servir de ayuda al paciente. También, en aquellos ejercicios en que esté disponible un enlace a un vídeo de instrucciones sobre los mismos, el profesional puede habilitar que se muestren al usuario en una fecha determinada para ayudar al usuario con su seguimiento del tratamiento asignado. Estos dos últimos campos (comentarios y enlace de vídeo) son opcionales de añadir por parte del profesional. Para terminar, el profesional elige tantos ejercicios como considere y los asigna al paciente seleccionado. Esto último se realiza mediante otra nueva petición HTTP POST al *webservice* correspondiente para su correcto registro y asignación al paciente. En caso de éxito se mostrará un mensaje al profesional acerca de ello y se redirige a la pantalla principal del profesional de clínica. En caso contrario se volverá a la pantalla de selección de ejercicios para un nuevo intento.

A continuación, se presenta el diagrama de flujo de las operaciones que se realizan durante el flujo de la funcionalidad, en la Figura 54Figura 50. El diagrama de flujo de la clase que gestiona esta funcionalidad se presenta en la Figura 55.

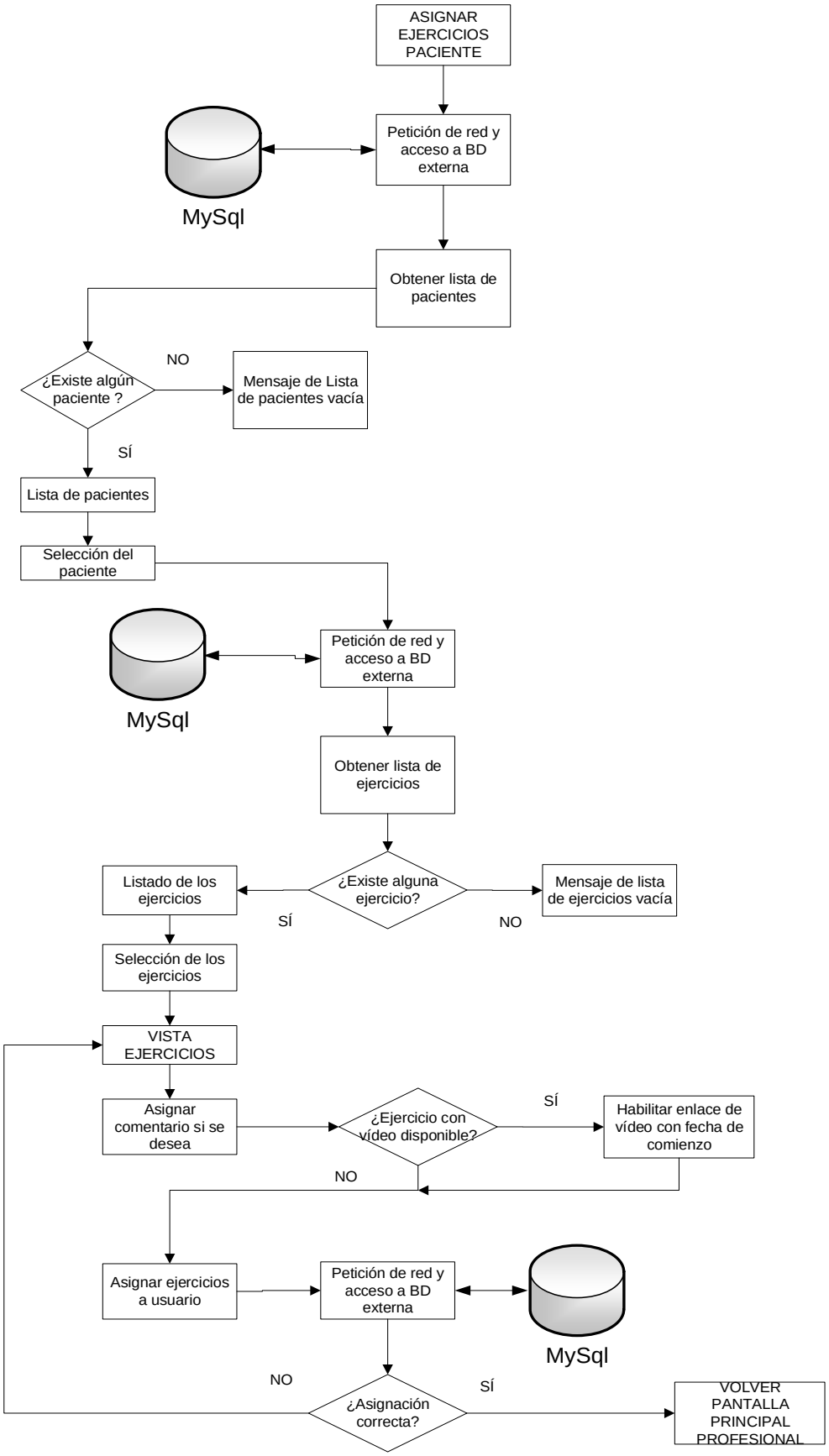

Figura 54 - Diagrama de flujo de la funcionalidad de Asignar Ejercicios a un Paciente

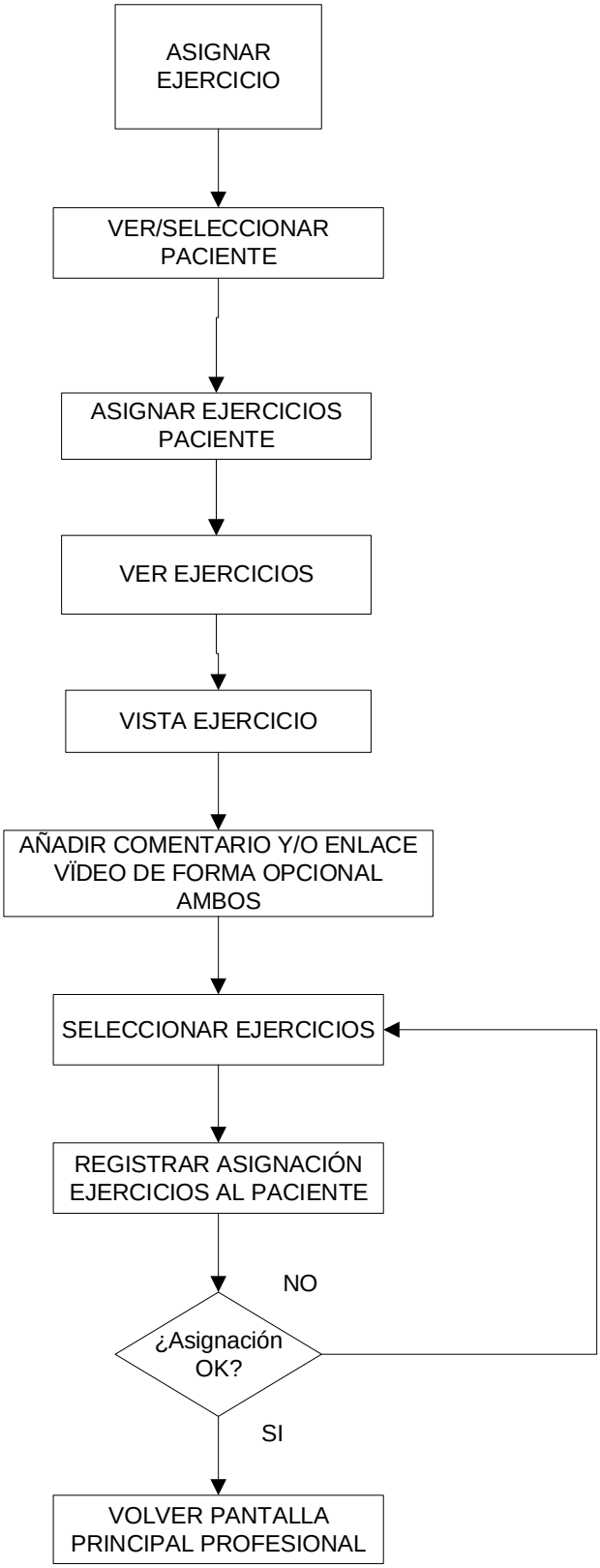

Figura 55 - Diagrama de flujo de la clase que gestiona la funcionalidad de Asignar Ejercicios a un Paciente

Capítulo 5 – Manual de usuario de la aplicación de fisioterapia

Capítulo 5 – Manual de usuario de la aplicación de fisioterapia

En este apartado se va a comentar de forma resumida una posible guía de usuario para el manejo de la aplicación de fisioterapia. Este manual se centrará en las funcionalidades disponibles para cada uno de los dos perfiles de usuarios posibles, junto a unas capturas de pantalla para que sea de mayor ayuda poder visualizar en qué pantalla se van realizando cada una de las mismas. Este apartado será de carácter explicativo e ilustrativo y no técnico, a diferencia del capítulo anterior.

Como últimos aportes a añadir antes de comenzar, el idioma de los contenidos de la base de datos de la aplicación se encuentra en español mientras que el idioma de la aplicación está establecido nativamente en inglés.

5.1. *Login* de usuario

Esta es la primera pantalla que el usuario verá de la aplicación sin que antes se haya autentificado o estuviera algún otro usuario ya logueado. El fondo de esta pantalla es dinámico ya que en ella se reproduce en segundo plano un vídeo relacionado con el ámbito de la fisioterapia con el objetivo de hacer la interfaz gráfica más atractiva al usuario.

Una vez en esta pantalla, el usuario puede observar como existen dos campos a introducir para realizar la autentificación del usuario, su nombre de usuario y su clave de acceso (éste último se introduce de forma que no sea visible el contenido; si se desea visible existe un botón a la derecha del campo de clave de acceso que permite alternar entre un tipo de visualización "encriptado" y "sin encriptar"). También se puede observar, en la parte inferior, como existe una barra que alterna entre tres colores:

- ✓ Azul: Cuando el usuario no ha introducido ningún dato en los campos anteriores. Si se pulsa sobre la misma el usuario podrá realizar el registro de un nuevo usuario (Figura 56).
- ✓ Rojo: Cuando el usuario ha introducido algún dato en los campos, pero no están completados (Figura 57).

- ✓ Verde: Cuando el usuario ha introducido todos los campos correctamente. Si se pulsa sobre la misma se procederá a realizar el *login* de usuario. (Figura 58)

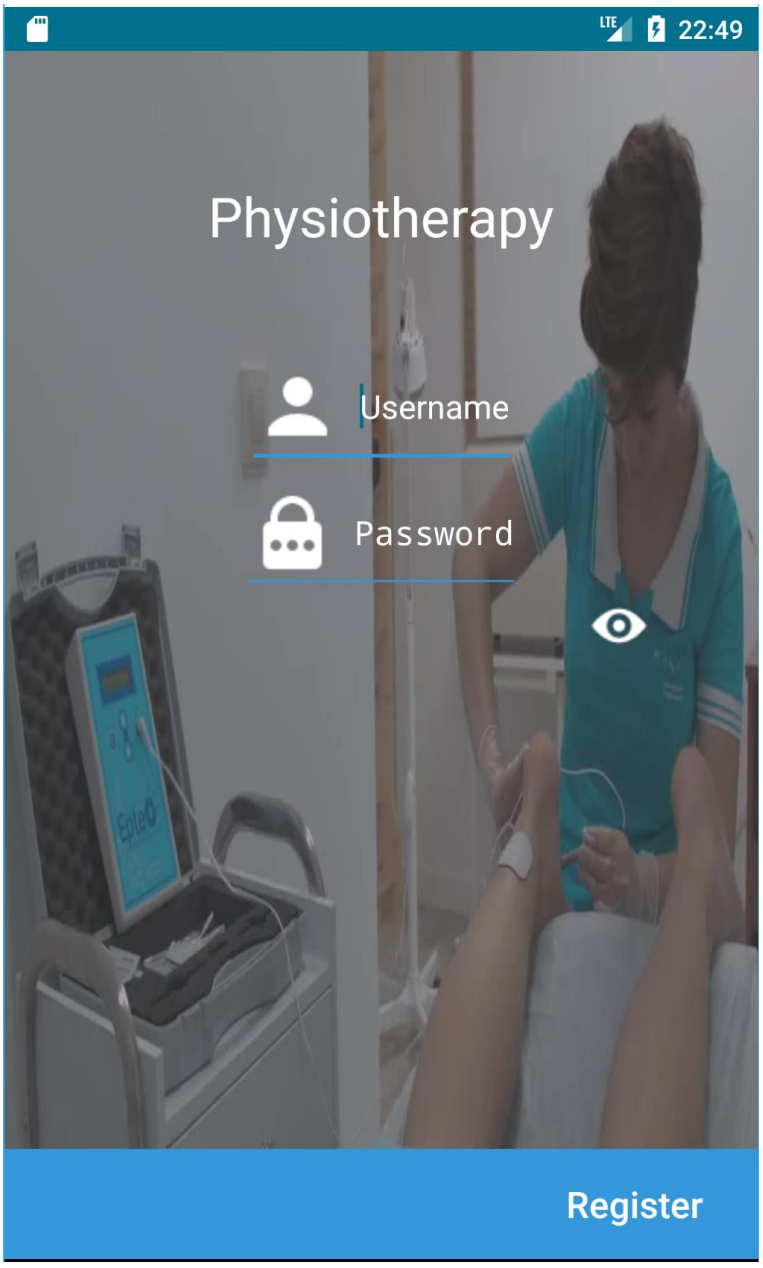

Figura 56 - Pantalla de Login con la barra de acción inferior en color azul

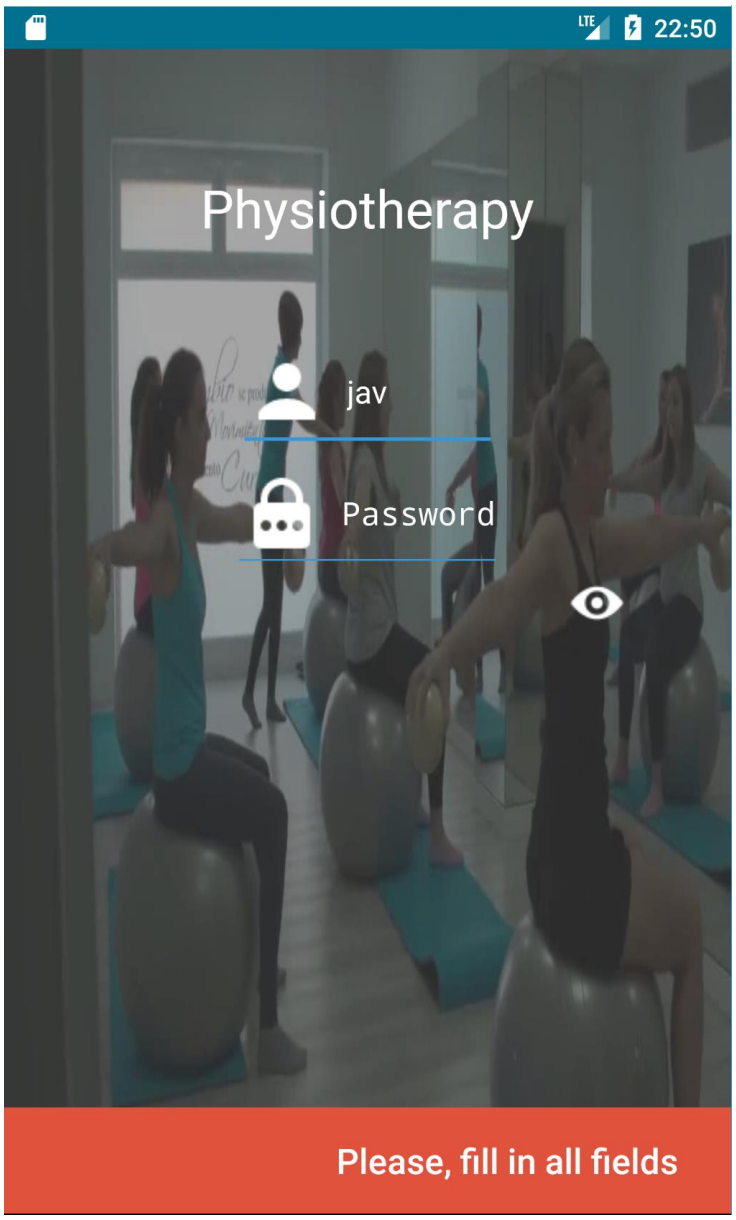

Figura 57 - Pantalla de Login con la barra de acción inferior en color rojo

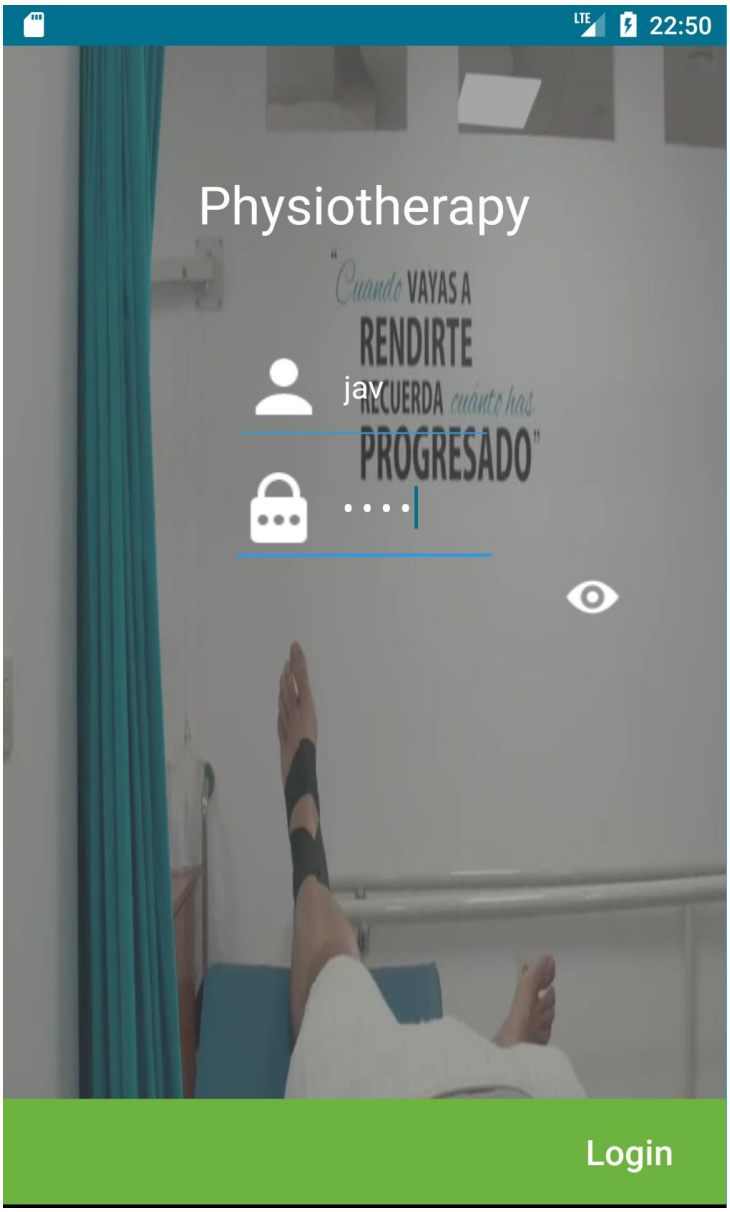

Figura 58 - Pantalla de Login con la barra de acción inferior en color verde

Una vez se procede a realizar el *login*, si éste resulta exitoso se procederá a mostrar la pantalla de inicio de la aplicación con las funcionalidades propias del perfil autentificado en la aplicación. En caso contrario, se informará del error al usuario de que el nombre de usuario y/o clave de acceso introducidos no son correctos (Figura 59).

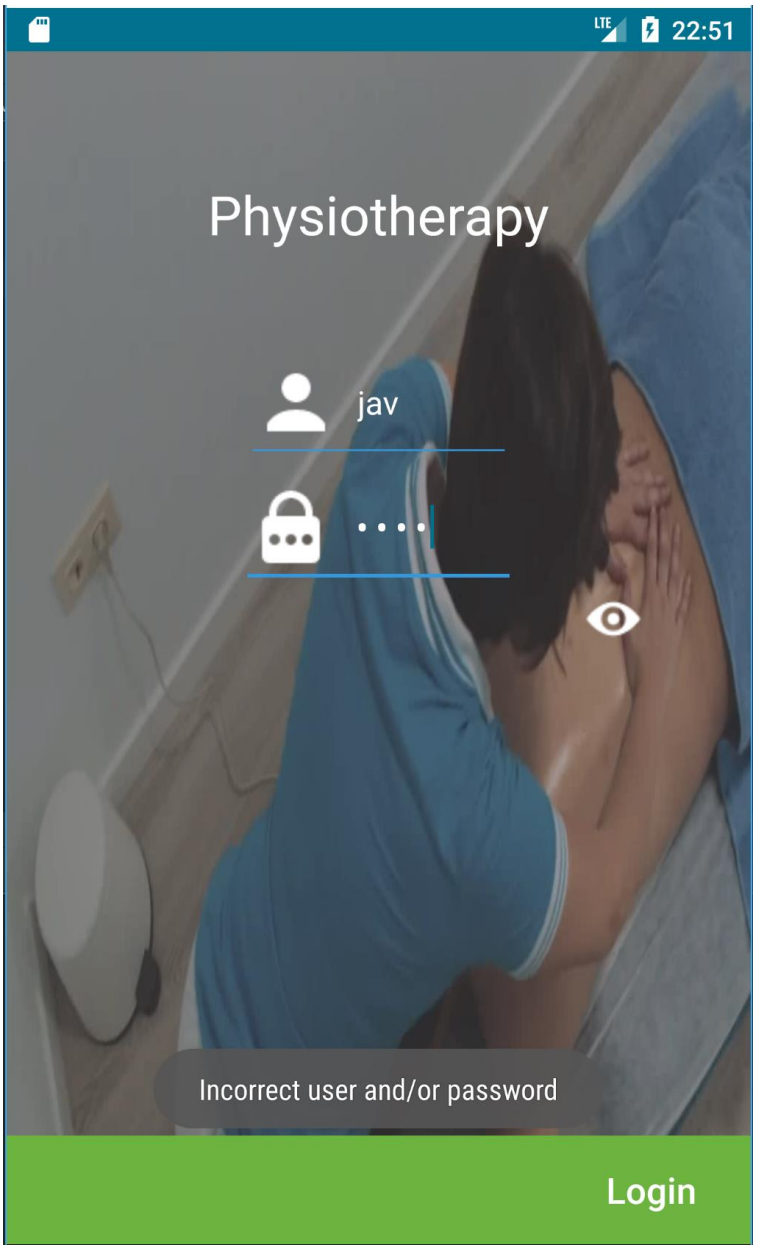

Figura 59 - Pantalla de Login con mensaje de error de datos, tras intentar autentificarse, al haber introducido datos incorrectos

5.2. Registro de nuevo usuario

Para llegar a esta pantalla se ha de pulsar la barra de acción inferior de la pantalla de *login*, cuando no se ha introducido ningún dato en la misma.

Una vez pulsada se muestra la pantalla de registro de nuevo usuario (Figura 60), en la que se pide al usuario que rellene una serie de campos obligatorios de información del mismo (entre otros, nombre, apellidos, DNI, dirección, teléfono principal y secundario (opcional), email, nombre de

usuario y clave de acceso). A medida que el usuario va rellenando los mismos, el contenido de estos campos se valida para comprobar que se ajusta al formato de datos esperado. En caso de no ser así, se muestra un mensaje informativo al usuario debajo del campo en el que está introduciendo los datos (Figura 61).

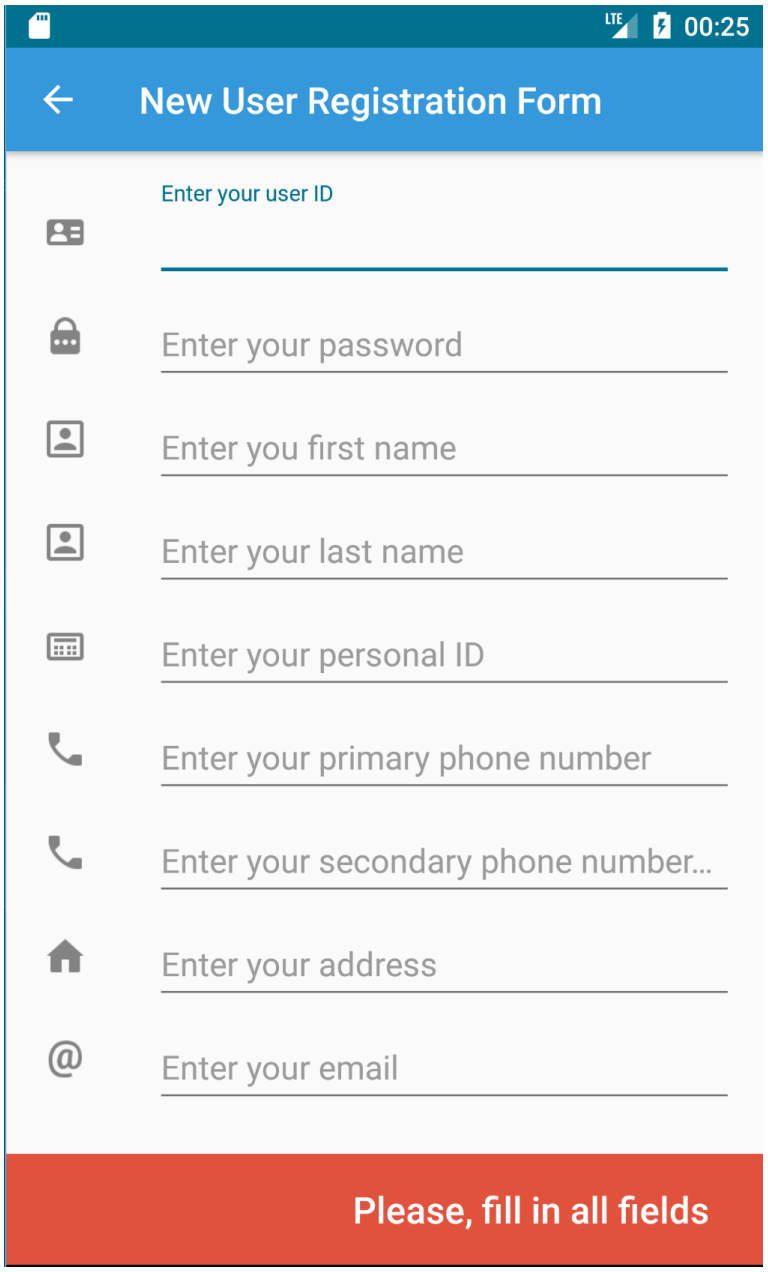

Figura 60 - Pantalla de Registro de nuevo usuario

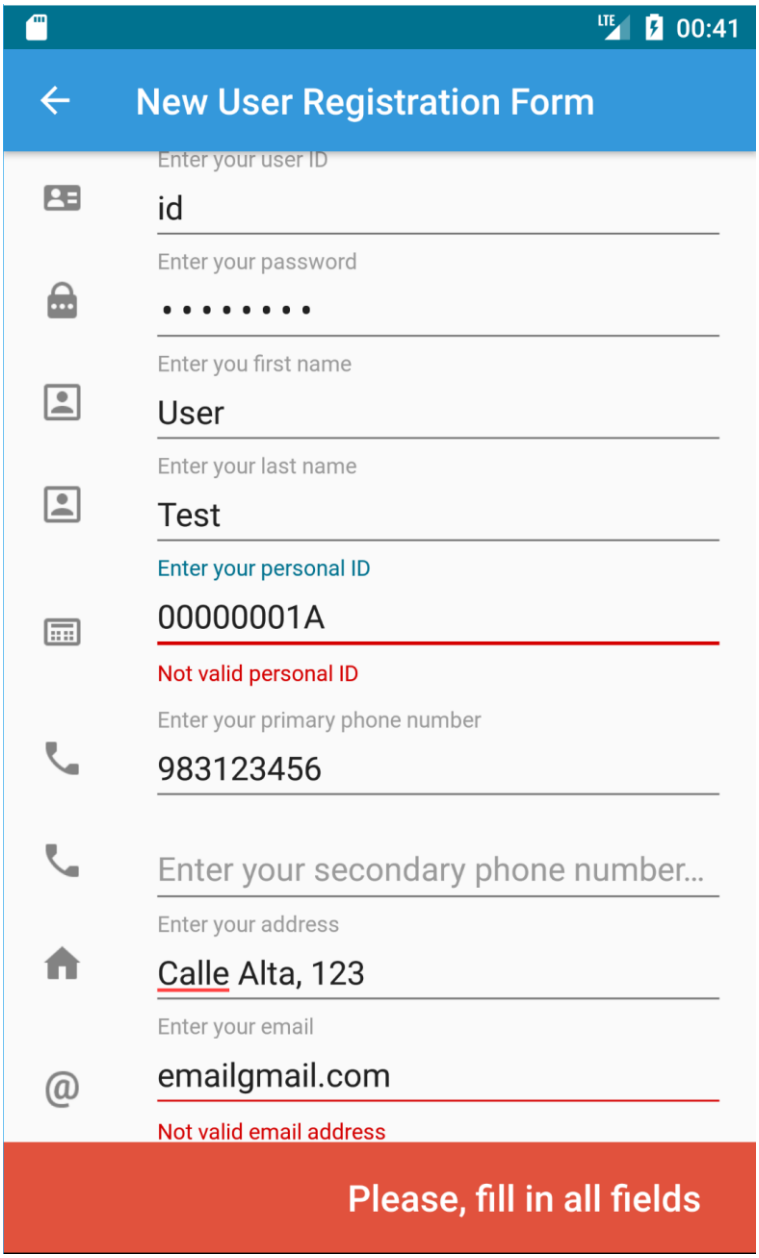

Figura 61 - Pantalla de Registro de un nuevo usuario con validación de los campos a introducir

Una vez introducidos todos los datos del formulario de nuevo usuario, la barra de acción inferior pasa de rojo a color verde para permitir registrar el nuevo usuario (Figura 62). Si se realiza con éxito esta última acción tras pulsar en la barra verde, se mostrará un mensaje al usuario de confirmación de registro del nuevo usuario, redirigiéndose la aplicación a la pantalla de *login* con la nueva información del usuario para poder autentificarse. En caso contrario, pudiendo estar ya registrado el nombre de usuario elegido, se le notificará al usuario con un mensaje para que elija uno nuevo y vuelva a intentar realizar el registro del nuevo usuario.

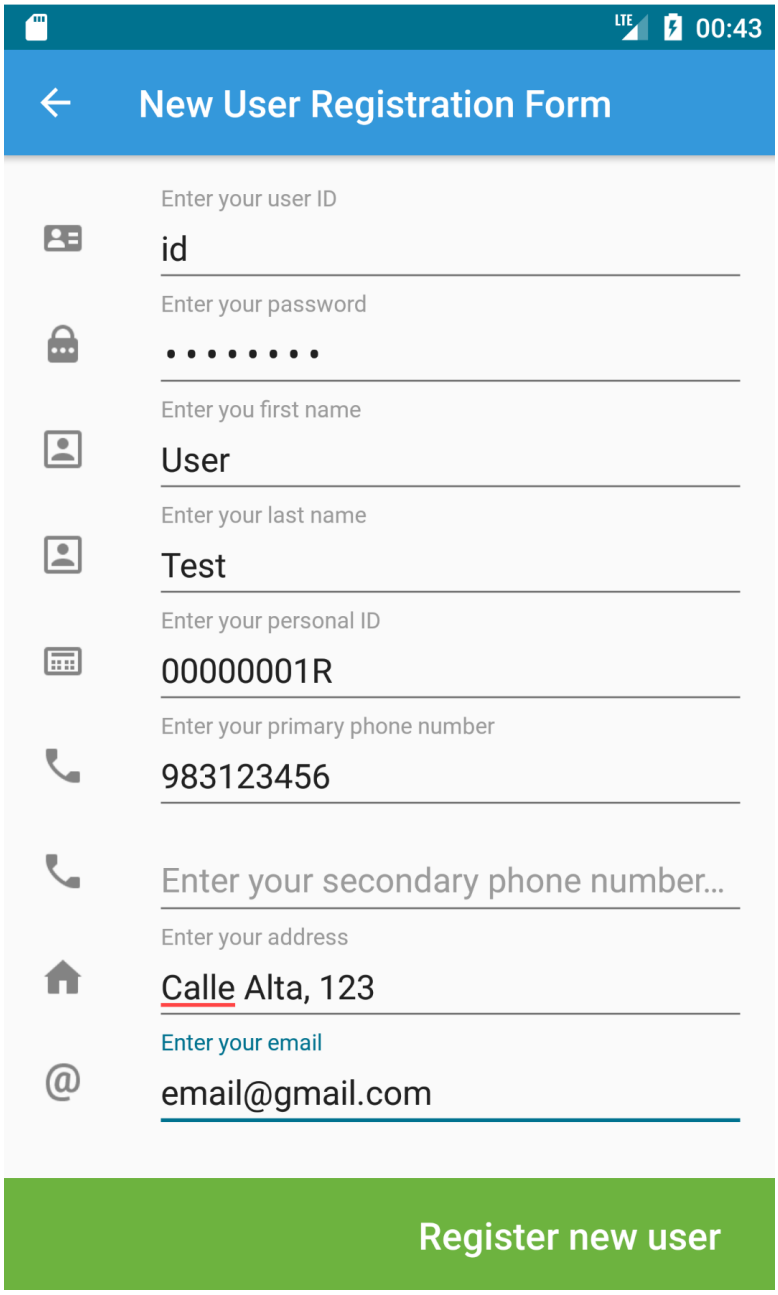

Figura 62 - Pantalla de Registro de un nuevo usuario con todos los campos a introducir validados correctamente

5.3. Pantalla principal de perfil de usuario paciente

Una vez el usuario se ha autentificado correctamente desde la pantalla de *login*, se llega a esta pantalla principal de la aplicación para un perfil de usuario tipo paciente. En la Figura 63 se puede observar esta pantalla con las funcionalidades que ofrece al paciente para realizar en la aplicación:

Figura 63 - Pantalla principal del perfil de usuario tipo paciente

- ✓ A través del deslizamiento lateral desde la izquierda de la pantalla o pulsando en el botón de las tres líneas en la barra superior, se observa un menú lateral con información del nombre de usuario autentificado y la opción de desconectarse y eliminar su autentificación de la aplicación (*log-off*). Este menú se puede ver en la Figura 64.

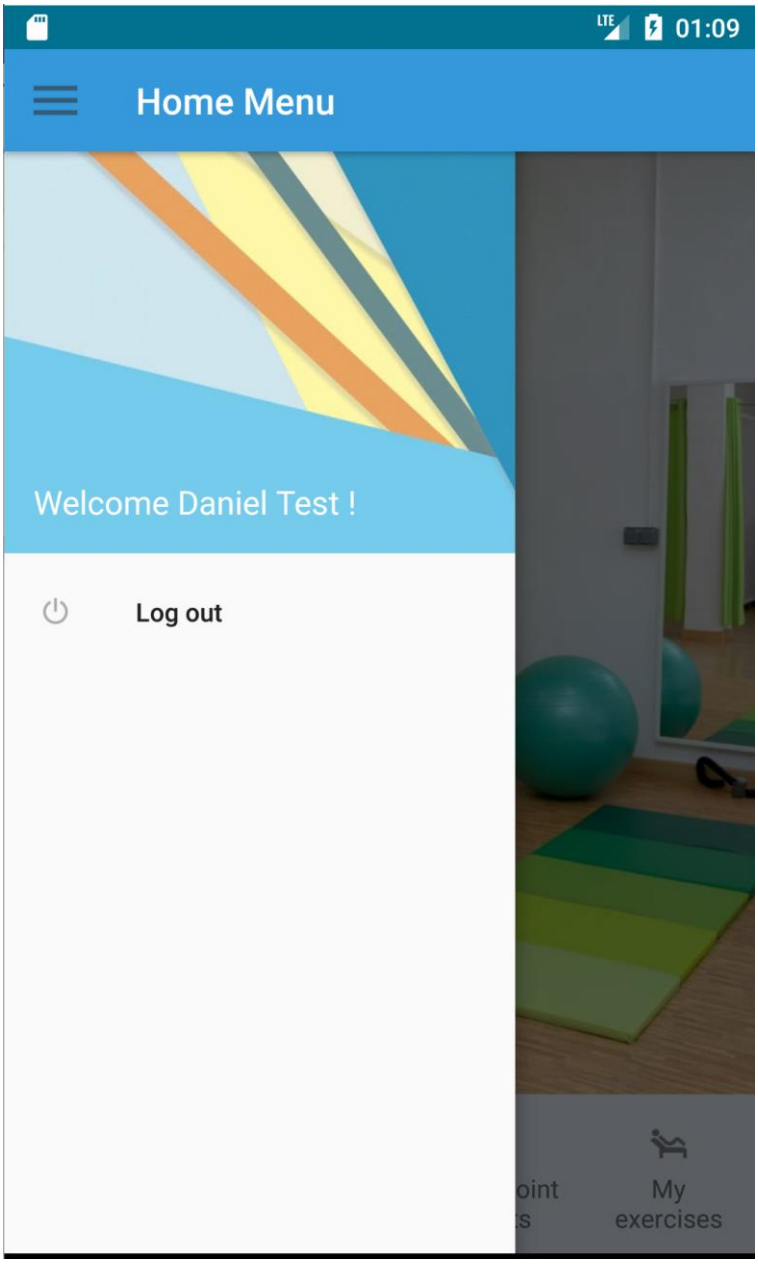

Figura 64 - Menú lateral de la pantalla principal de perfil de tipo usuario paciente

- ✓ *Clinics*: Permite al usuario observar las clínicas de fisioterapia disponibles en la aplicación, con toda su información de ubicación, contacto y lista de sus tratamientos (Figura 63).

- ✓ *Make an appointment*: Permite al usuario reservar una cita en la clínica que el paciente quiera seleccionar (Figura 63).

- ✓ *My appointments*: Permite al paciente observar sus citas con las clínicas realizadas y poder cancelarlas (Figura 63).

- ✓ *My exercises*: Permite al paciente observar los ejercicios asignados por su profesional de clínica, junto a la información relacionada con los mismos (Figura 63).

5.4. Ver clínicas (perfil de usuario paciente)

El usuario podrá consultar toda la información de las clínicas de fisioterapia disponibles en la aplicación a través de un flujo de pantallas tras seleccionar la opción de *Clinics* en la pantalla principal como perfil paciente.

La primera de las pantallas muestra una lista de las clínicas disponibles de la aplicación. En caso de no haber ninguna, se mostraría la pantalla en blanco con un mensaje informativo de tal situación. En esta pantalla, para cada una de las clínicas se muestra parte de su información como su logo, nombre, dirección, teléfono y valoración del 1 al 5 (Figura 65).

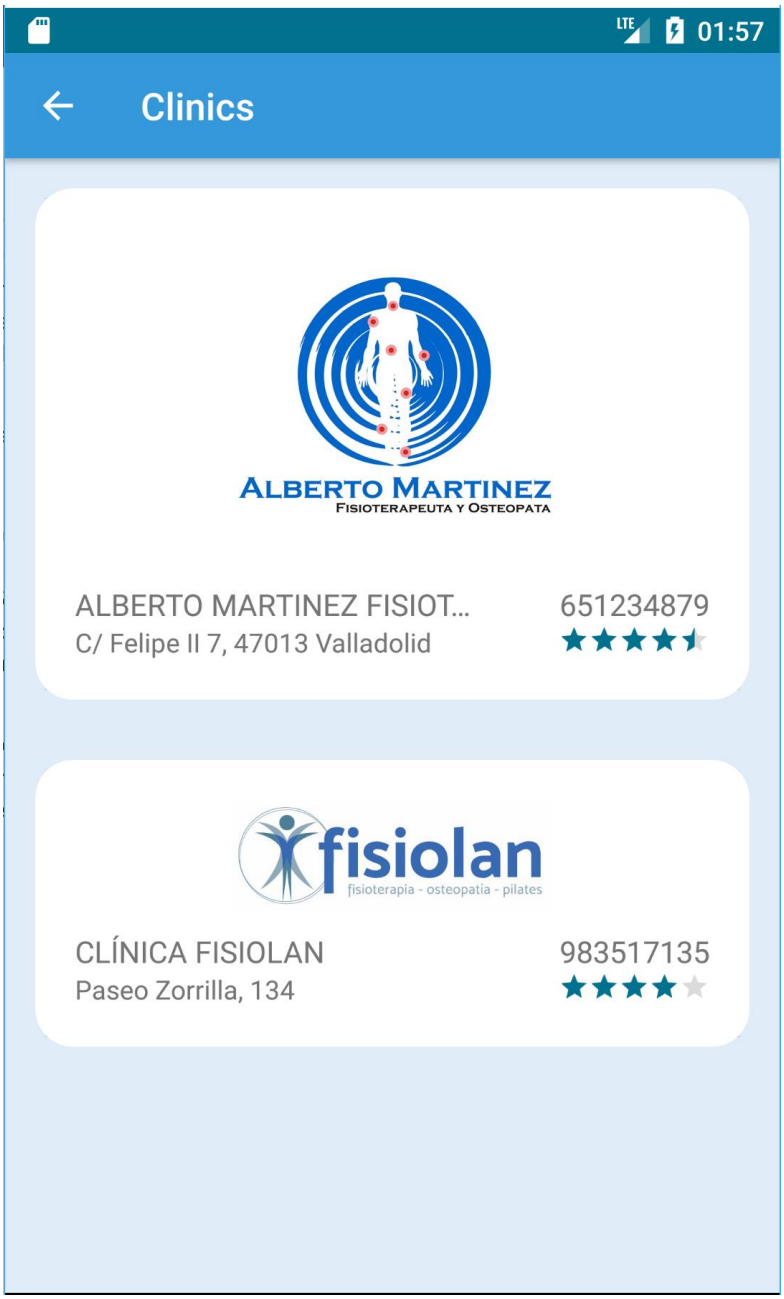

Figura 65 - Pantalla de listado de clínicas de la aplicación como perfil de usuario tipo paciente

Una vez que el usuario ha seleccionado una de las clínicas de la lista, se detalla más información de la misma, en una nueva pantalla (Figura 66), como su dirección *web* o su correo electrónico, además de su dirección y teléfono de contacto.

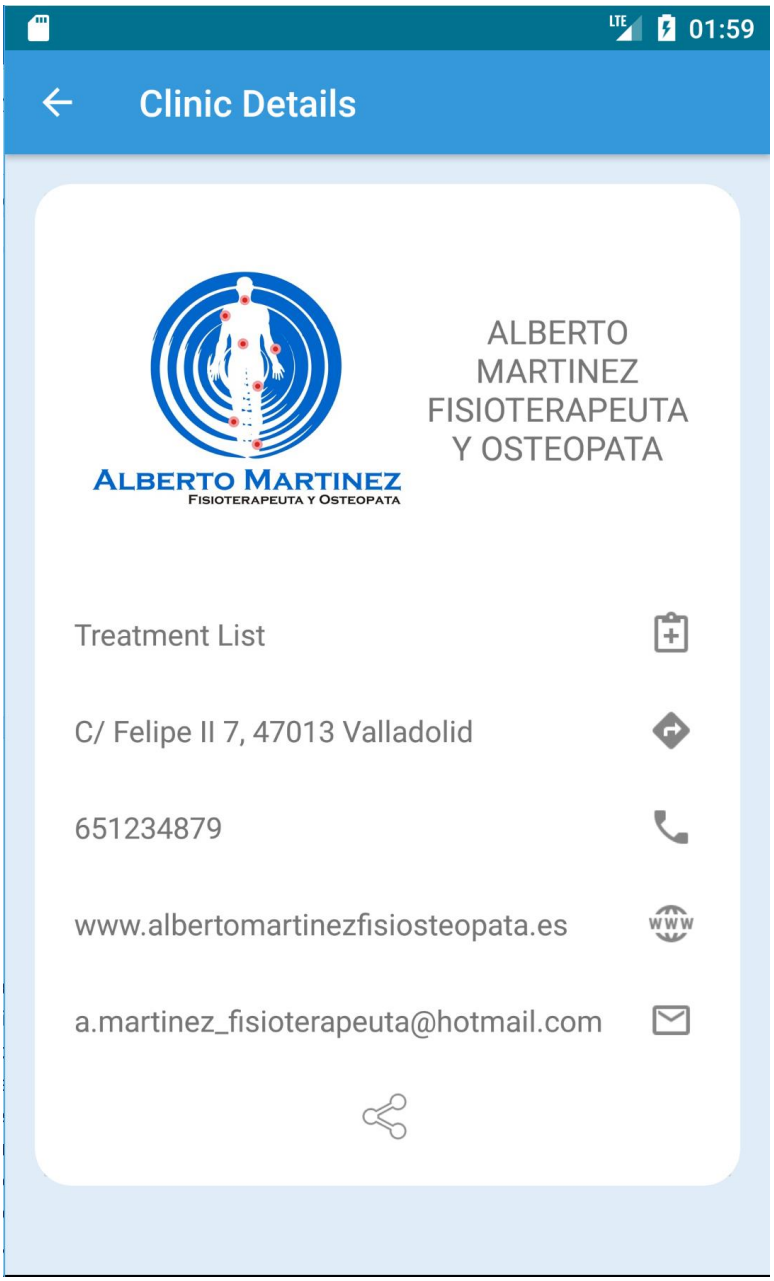

Figura 66 - Pantalla de detalle de una clínica seleccionada de la aplicación como perfil de usuario tipo paciente

Cabe destacar que con estos cuatro últimos campos de información sobre la clínica, el paciente puede interaccionar con los mismos; es decir, si el paciente pulsa por ejemplo sobre su dirección se le enviará a la aplicación de mapas que pueda tener instalada en su dispositivo para que le guíe y pueda ver el recorrido hasta la misma clínica; si el paciente pulsa sobre su correo electrónico se abrirá una pantalla en su aplicación de cliente de correo para contactar con la clínica; si el paciente pulsa sobre el teléfono se le marcará el mismo en la aplicación de llamadas sin realizarse la llamada y, finalmente, si el paciente pulsa sobre la dirección *web* de la clínica, se abrirá dicha página *web* en el navegador del dispositivo.

También se ofrece, en esta pantalla de detalles poder visualizar la lista de tratamientos que ofrece la clínica, pulsando sobre dicha opción. Tras ello, se abrirá una nueva pantalla (Figura 67) con una lista de los mismos. En el caso de esta lista sea vacía, se mostraría la pantalla en blanco con un mensaje informativo de tal situación. En cada uno de los tratamientos listados aparecerá una foto del mismo y su nombre.

Figura 67 - Pantalla de listado de tratamientos de una clínica de la aplicación como perfil de usuario tipo paciente

Si se quiere más detalle acerca del tratamiento, el paciente puede pulsar sobre uno de ellos, abriéndose una nueva pantalla en la que se muestra más información del mismo como su descripción (Figura 68).

Figura 68 - Pantalla de detalle de un tratamiento de una clínica seleccionada de la aplicación como perfil de usuario tipo paciente

Tanto en las pantallas de detalle de la clínica seleccionada como del tratamiento se da la opción de compartir un breve resumen de los mismos con otras aplicaciones del teléfono mediante un icono en la parte inferior de la vista de información de cada uno de ellos.

5.5. Reservar una cita (perfil de usuario paciente)

El usuario podrá realizar la reserva de una cita en una clínica que seleccione de entre las disponibles en la aplicación a través de un flujo de pantallas tras seleccionar la opción de *Make an appointment* en la pantalla principal como perfil paciente.

En la primera pantalla de este flujo se muestra la pantalla de la Figura 65, con el único objetivo de que el usuario seleccione la clínica para la cual desea reservar.

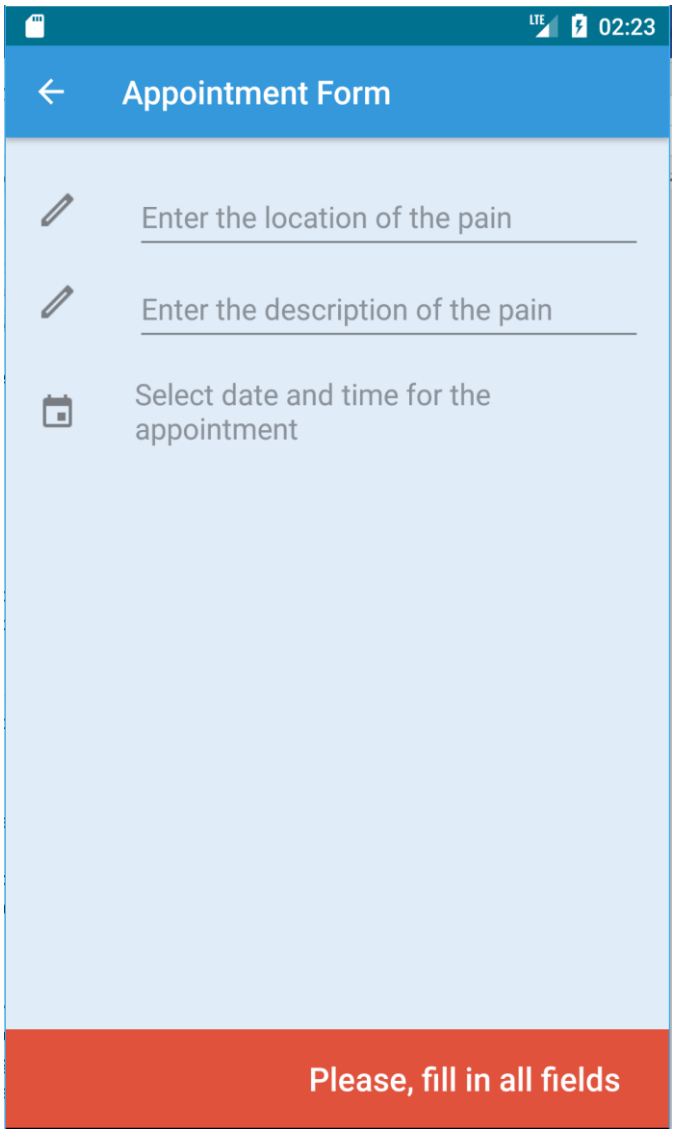

Figura 69 - Pantalla de introducción de datos para la Reserva de una cita en una clínica seleccionada de la aplicación con perfil de usuario tipo paciente

Tras seleccionar la clínica, se abre una nueva pantalla en la que el paciente tiene que introducir una serie de campos para realizar la reserva (Figura 69). Estos campos son todos obligatorios. Se trata de introducir la motivación de la cita en términos de localización y descripción del dolor, así como fecha deseada para la misma. Tras la introducción de los mismos, la barra de acción inferior se pondrá de color verde para que se pueda realizar la reserva de la cita (Figura 70).

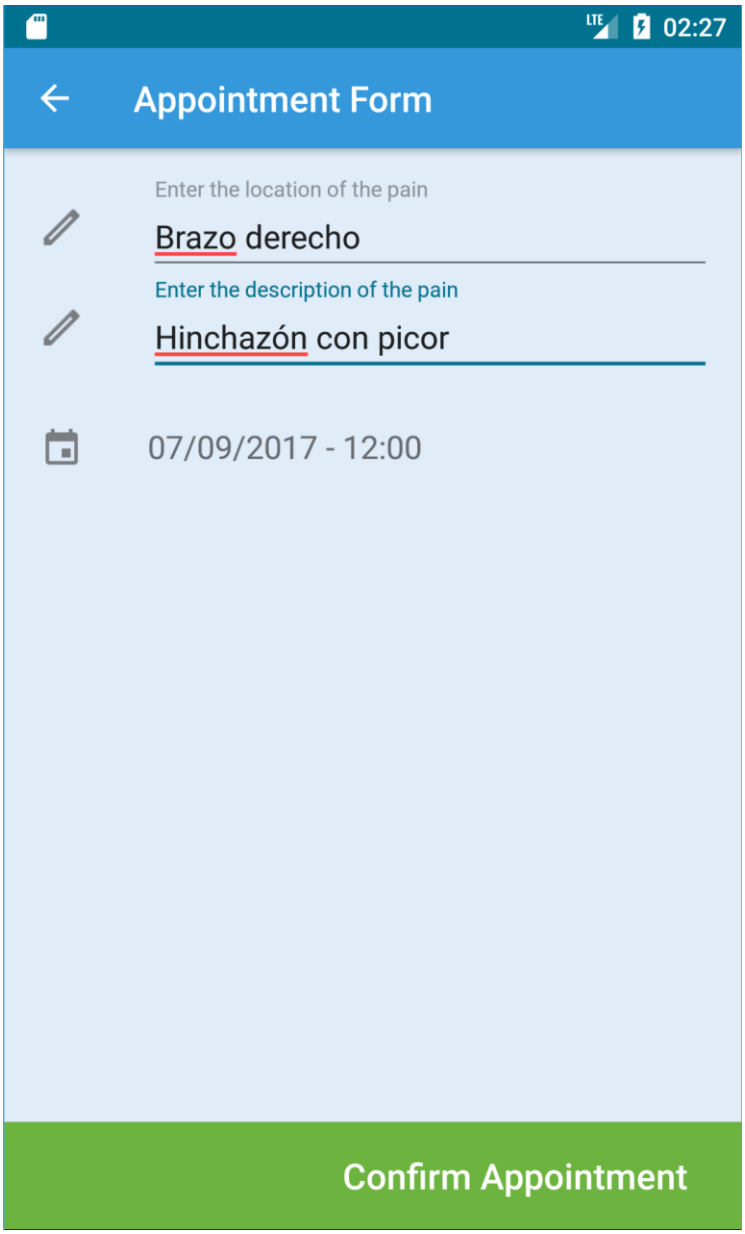

Figura 70 - Pantalla con datos introducidos para la Reserva de una cita en una clínica seleccionada de la aplicación con perfil de usuario tipo paciente

Si pulsamos sobre la barra verde obtendremos un mensaje de confirmación con la fecha más próxima o igual a la introducida anteriormente disponible por el sistema para la reserva de la cita (Figura 71). Si se selecciona que sí, se procede a registrar la reserva, informando al paciente del resultado y se volvería a la pantalla inicial. En caso contrario, el paciente permanecería en la misma pantalla que se encuentra con el objetivo de cambiar la fecha deseada para la cita.

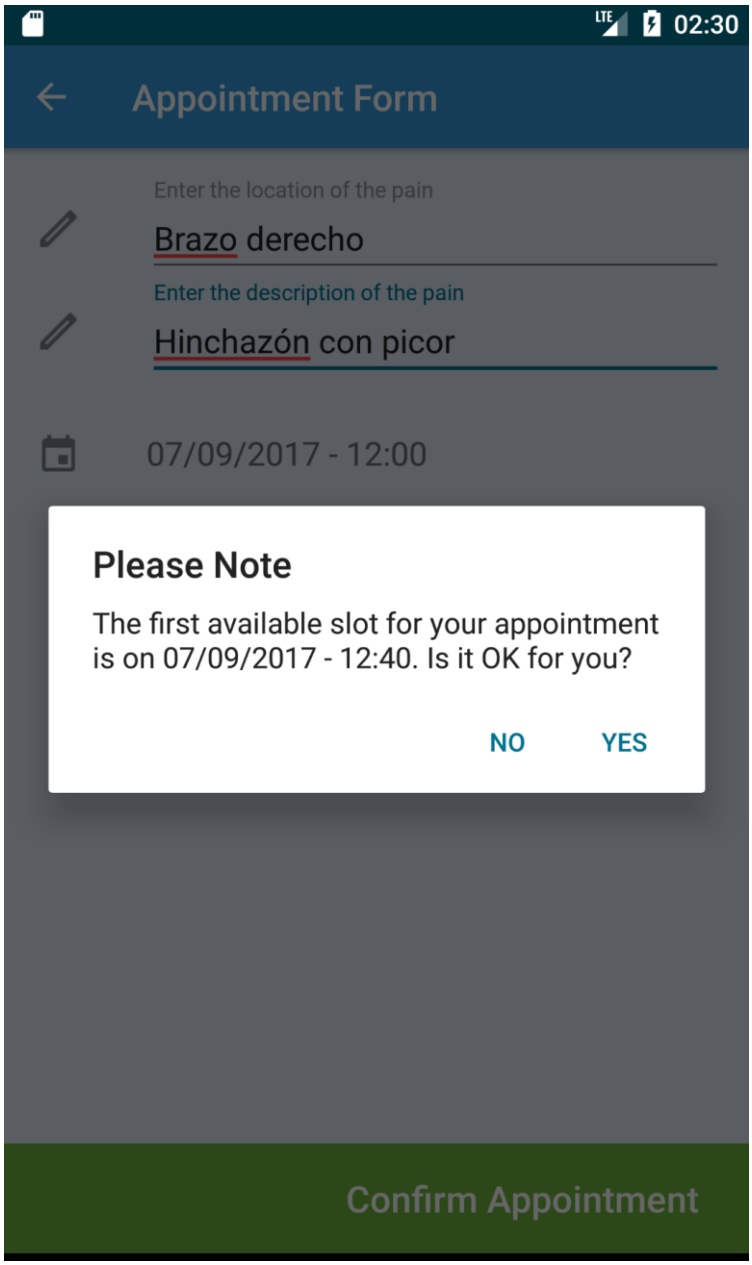

Figura 71 - Mensaje de confirmación de disponibilidad horaria para la reserva de una cita en una clínica seleccionada de la aplicación con perfil de usuario tipo paciente

5.6. Gestionar mis reservas (perfil de usuario paciente)

El paciente podrá visualizar las reservas realizadas en las diferentes clínicas, así como la posibilidad de anularlas.

Para ello, tras seleccionar la opción de *My appointments* en la pantalla principal como perfil paciente, se abre una nueva pantalla al paciente en el que podrá visualizar una lista de todas sus reservas realizadas (Figura 72). En caso de no haber ninguna, se mostraría la pantalla en blanco con un mensaje informativo de tal situación.

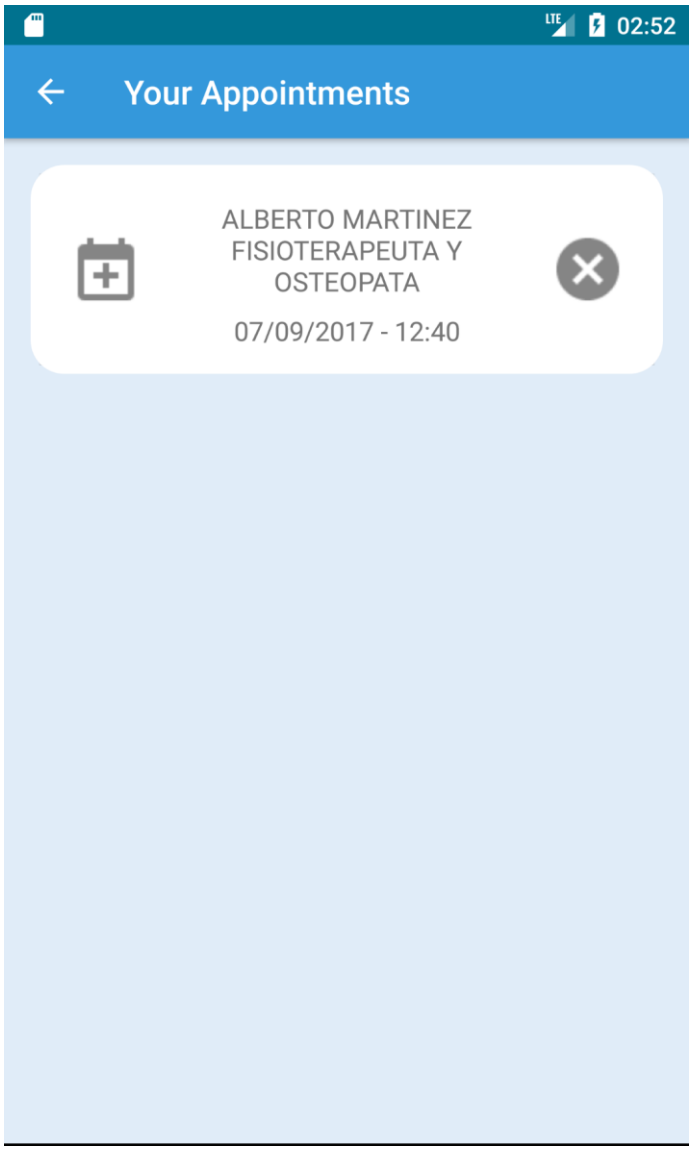

Figura 72 - Pantalla del listado de reservas realizadas en la aplicación con perfil de usuario tipo paciente

Por cada reserva realizada de la lista visualizada se muestra información de la clínica asociada a la misma, así como la fecha de la cita en la misma. A la derecha se muestra un botón de cancelar que da la posibilidad al paciente de anular la misma. Si así de desea, se mostrará al paciente un mensaje de confirmación de si desea anularla (Figura 73) y, en caso afirmativo, se procede a su anulación, informando al paciente del resultado y volviendo a la pantalla principal. En caso contrario, se vuelve a la misma pantalla de lista de reservas.

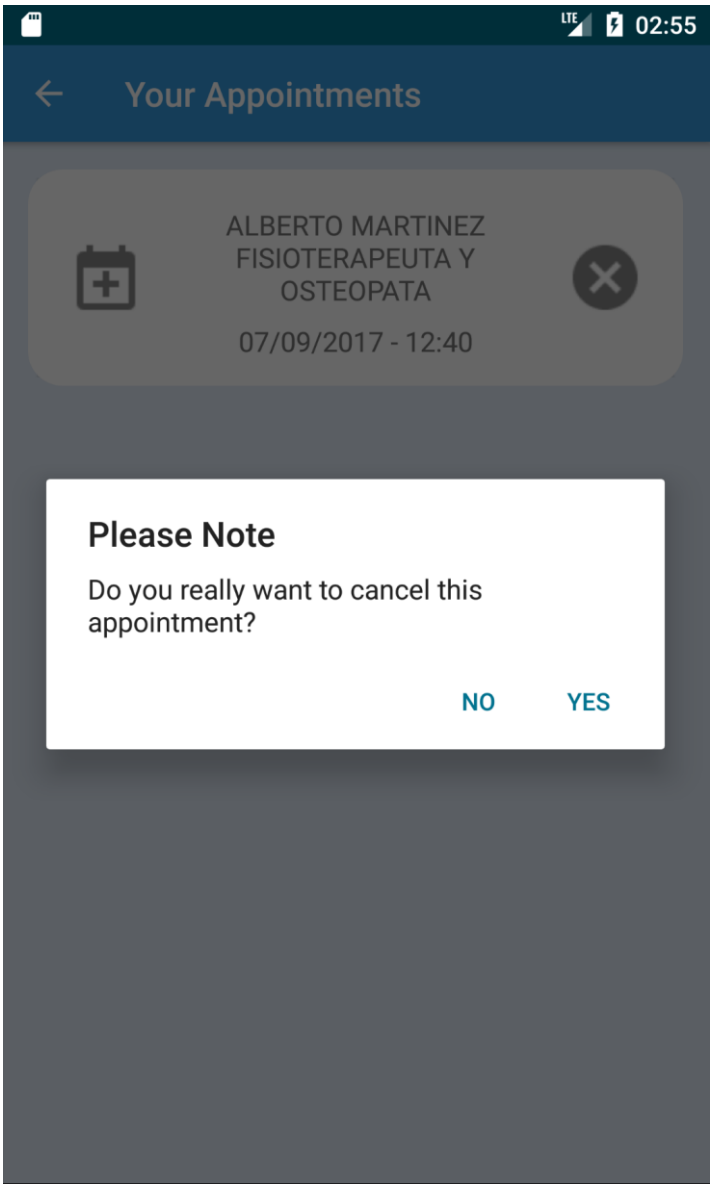

Figura 73 - Mensaje de confirmación de anulación de una reserva en la aplicación con perfil de usuario tipo paciente

5.7. Ver mis ejercicios asignados (perfil de usuario paciente)

El usuario puede visualizar los ejercicios que tiene asignador por un profesional de una clínica mediante la selección de la opción de *My Exercises* en la pantalla principal como perfil paciente.

Tras seleccionar dicha opción, se muestra una nueva pantalla con una lista de ejercicios asignados al paciente por su profesional de clínica (Figura 74). En caso de no haber ninguno asignado, se mostraría la pantalla en blanco con un mensaje informativo de tal situación.

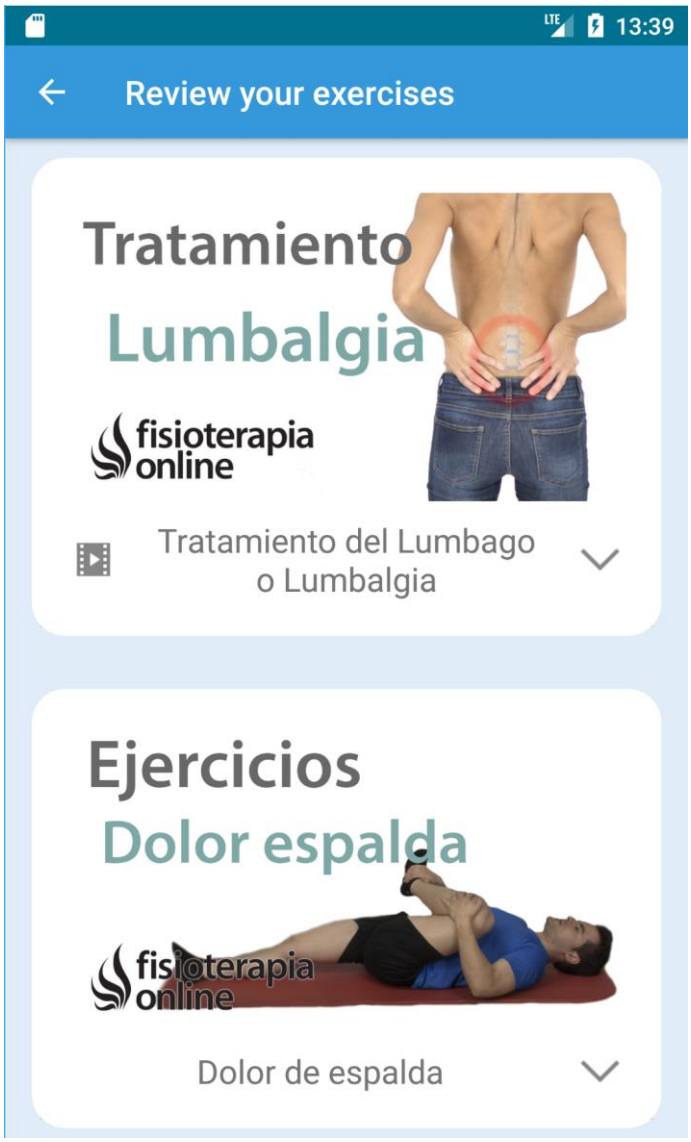

Figura 74 - Pantalla con la lista de ejercicios asignados al paciente en la aplicación con perfil de usuario tipo paciente

Por cada ejercicio asignado, se muestra información del mismo como su nombre, una fotografía del mismo y un botón en forma de flecha, a la derecha del nombre del ejercicio, que permite la expansión de la vista del ejercicio con el objetivo de mostrar más detalles del ejercicio asignado como su descripción, unos posibles comentarios añadidos por parte del profesional de clínica o un botón con un enlace a un vídeo de instrucciones del mismo para su visualización (Figura 75). Para esto último, con el fin de saber si un ejercicio tiene un vídeo asignado, no es necesario expandir el contenido del mismo mediante el botón de expansión puesto que se indicará a la izquierda del nombre el ejercicio con un icono de vídeo asociado la disponibilidad del mismo.

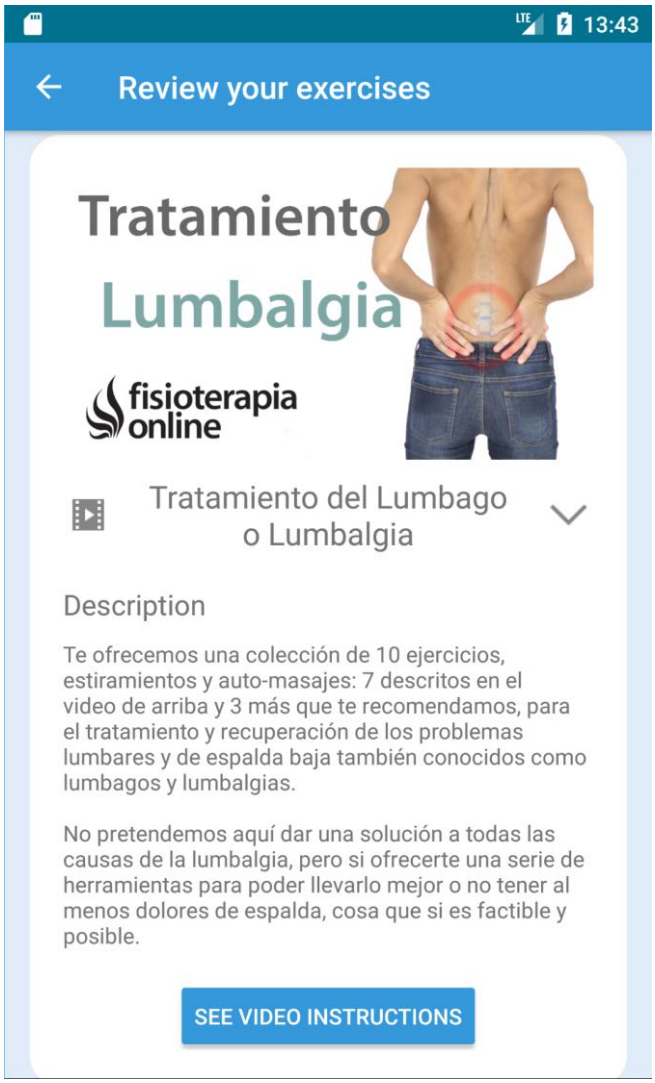

Figura 75 - Pantalla con detalles sobre un ejercicio de la lista de ejercicios asignados a un paciente en la aplicación con perfil de tipo usuario

Por último, si el paciente desea ver el vídeo disponible asociado al ejercicio, simplemente con pulsar en el botón que se encuentra bajo la expansión del contenido del ejercicio, se abrirá en su dispositivo el enlace del mismo (mediante su navegador *web* o plataformas como *YouTube)* para su posterior reproducción.

5.8. Pantalla principal de perfil de usuario profesional de clínica

Una vez el profesional de clínica se ha autentificado correctamente desde la pantalla de *login*, se llega a esta pantalla principal de la aplicación para un perfil de usuario tipo profesional de clínica. En la Figura 76 se puede observar esta pantalla con las funcionalidades que ofrece al profesional para realizar en la aplicación:

Figura 76 - Pantalla principal del perfil de usuario tipo profesional de clínica

- ✓ A través del deslizamiento lateral desde la izquierda de la pantalla o pulsando en el botón de las tres líneas en la barra superior, se observa un menú lateral con información del nombre de usuario autentificado y la opción de desconectarse y eliminar su autentificación de la aplicación (*log-off*). Este menú se puede ver en la Figura 77.

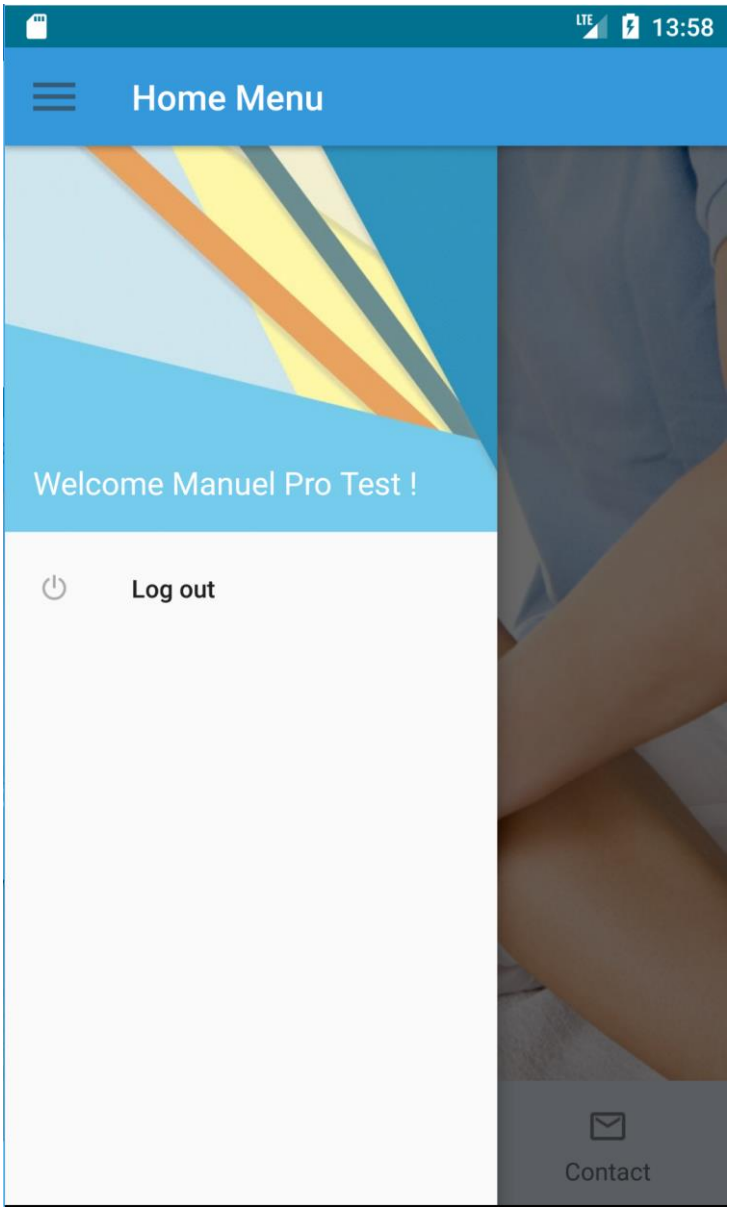

Figura 77 - Menú lateral de la pantalla principal de perfil de tipo usuario paciente

- ✓ *My Clinic*: Permite al profesional obtener la información de su centro de trabajo, su ubicación, contacto y lista de sus tratamientos (Figura 76).

- ✓ *Assign Exercises*: Permite al profesional asignar ejercicios de fisioterapia a pacientes previamente asignados por el encargado de la clínica (Figura 76).

- ✓ *Contact*: Permite al profesional ponerse en contacto con la clínica de trabajo mediante un correo electrónico a la misma (Figura 76).

5.9. Ver clínica de trabajo (perfil de usuario tipo profesional de clínica)

El profesional de clínica podrá consultar toda la información de la clínica de fisioterapia en la cual trabaja a través de un flujo de pantallas tras seleccionar la opción de *My Clinic* en la pantalla principal como perfil profesional de clínica.

Esta funcionalidad dispone de las mismas pantallas que la funcionalidad del apartado 5.4, con la única diferencia que, en vez de mostrarse todas las clínicas de la aplicación en la lista, solamente se muestra la clínica en la cual trabaja el profesional. Todo el funcionamiento, flujo de pantallas y visualización de los detalles de la clínica son iguales que los mostrados en el apartado anterior mencionado.

5.10. Asignar ejercicios a pacientes (perfil de usuario tipo profesional de clínica)

El profesional de clínica puede asignar ejercicios a los pacientes que tiene asignados a través de un flujo de pantallas que comienzan mediante la selección de la opción de *Assign Exercises* en la pantalla principal como perfil profesional de clínica.

Tras seleccionar dicha opción, se muestra una nueva pantalla con una lista de los pacientes asignados previamente al profesional por el encargado de la clínica, en función de las reservas realizadas por estos primeros (Figura 78). En caso de no haber ninguno asignado, se mostraría la pantalla mensaje informativo de tal situación.

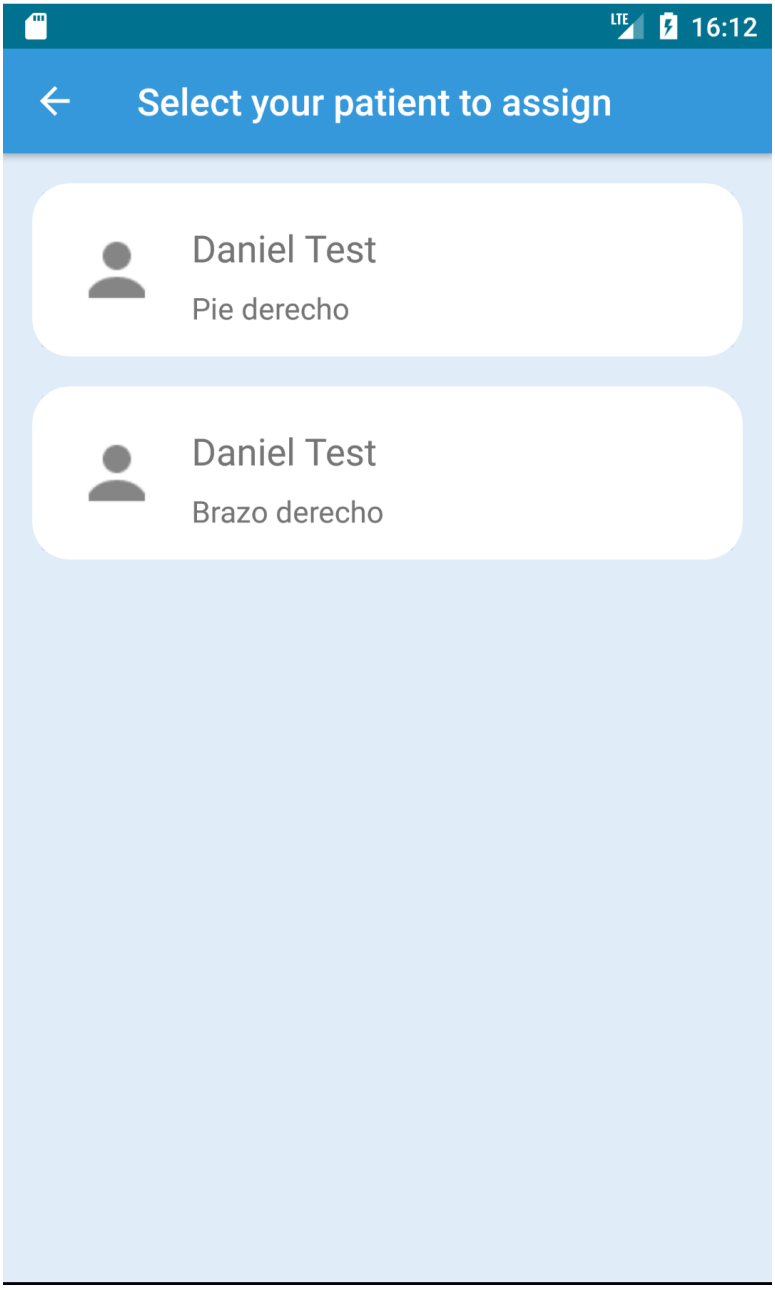

Figura 78 - Pantalla del listado de pacientes asignados a un profesional de clínica en la aplicación con perfil de usuario tipo profesional de clínica

En esta pantalla de selección de paciente, el profesional selecciona cuál es el que quiere seleccionar para asignar los ejercicios.

Una vez seleccionado el paciente, se muestra una nueva pantalla se muestra con una lista de ejercicios posibles de asignar al paciente por su profesional de clínica (Figura 79). En caso de no haber ninguno asignado, se mostraría la pantalla con un mensaje informativo de tal situación.

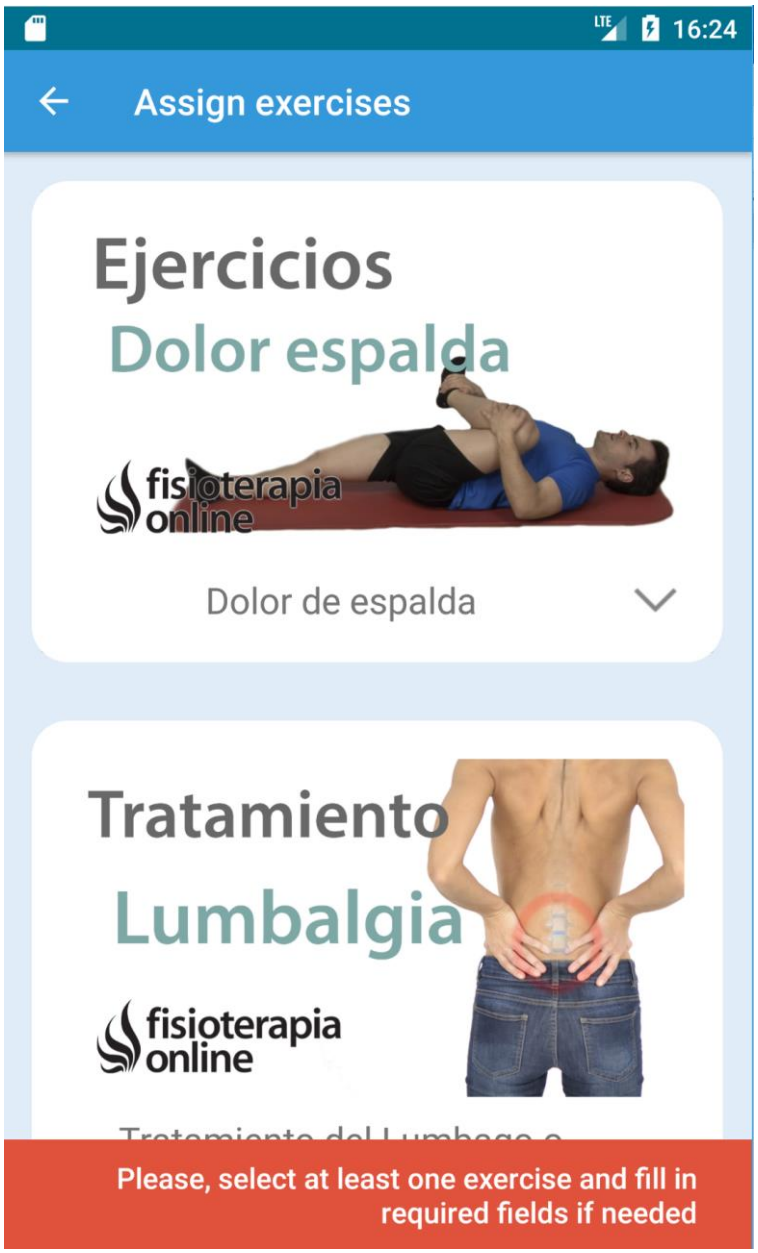

Figura 79 - Pantalla del listado de ejercicios posibles de asignar a un paciente en la aplicación con perfil de usuario tipo profesional de clínica

Por cada ejercicio posible de asignarse, se muestra información del mismo como su nombre, una fotografía del mismo y un botón en forma de flecha, a la derecha del nombre del ejercicio, que permite la expansión de la vista del ejercicio con el objetivo de mostrar más detalles del ejercicio asignado como su descripción, la posibilidad de añadir unos comentarios relativos al ejercicio por parte por parte del profesional de clínica y un *checkbox* para habilitar la asignación de un enlace a un vídeo de instrucciones del mismo para su visualización, para aquellos ejercicios que lo tengan disponible (Figura 80). Para esto último, una vez seleccionado, se muestra un

botón adicional en el que el profesional tiene que seleccionar una fecha y hora de disponibilidad del vídeo en el ejercicio asignado para el paciente. Tanto la asignación de comentarios como el vídeo son opcionales por el profesional. Así pues, para cada ejercicio a asignar por el profesional, este selecciona (la vista del ejercicio quedará resaltada con un borde azul oscuro) y los personaliza en función de los criterios del paciente y sus tratamientos. Cabe destacar que es posible seleccionar y añadir varios ejercicios a cada paciente.

Por último, una vez tengamos seleccionado al menos un ejercicio para asignar al paciente, la barra de acción inferior pasará de color rojo a verde (Figura 81). Si pulsamos sobre la barra verde, se procede a registrar la asignación de ejercicios, informando al profesional de clínica del resultado y se volvería a la pantalla inicial. En caso contrario, el profesional permanecería en la misma pantalla que se encuentra con la posibilidad de modificar su selección de ejercicios a asignar.

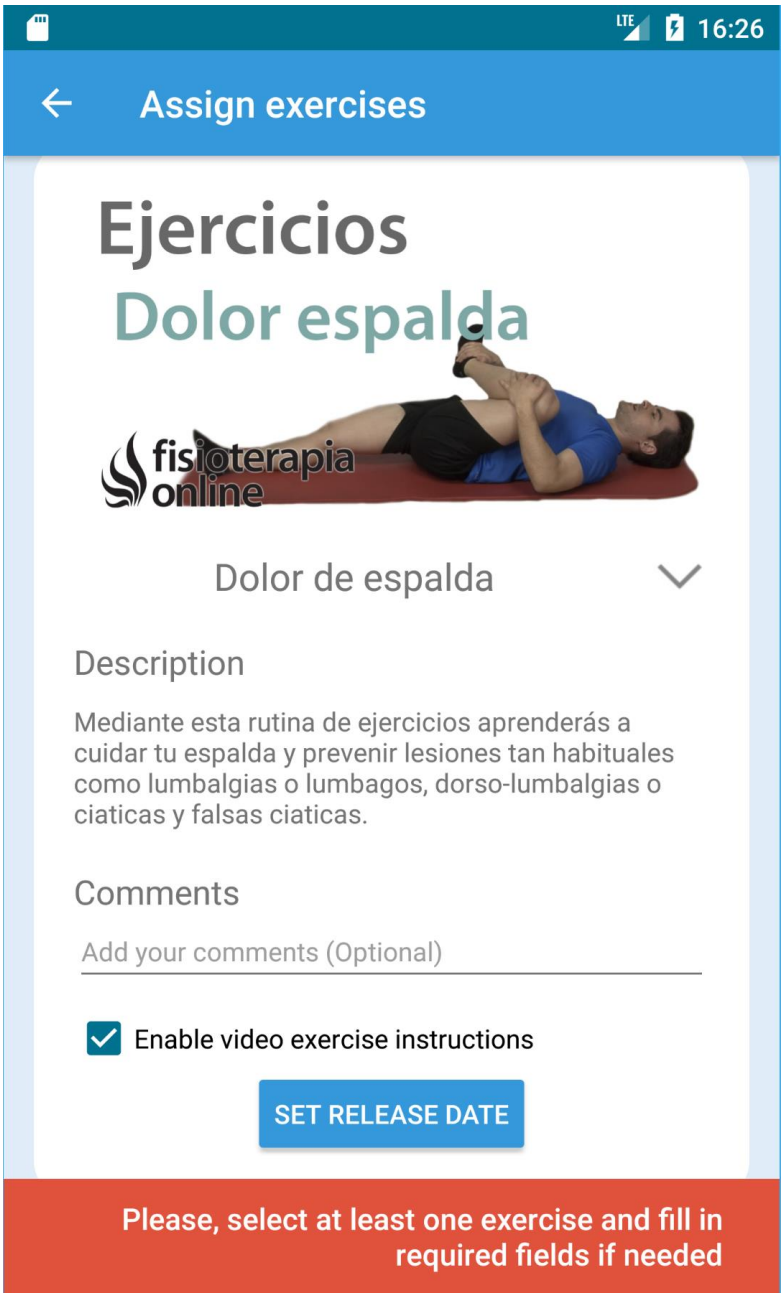

Figura 80 - Pantalla con detalles sobre un ejercicio posible de asignar a un paciente en la aplicación con perfil de tipo profesional de clínica

Figura 81 - Pantalla con selección de ejercicios a asignar a un paciente en la aplicación con perfil de usuario tipo profesional de clínica

Capítulo 6 – Conclusiones y Líneas futuras

Capítulo 6 – Conclusiones y Líneas Futuras

En este último capítulo se exponen las conclusiones obtenidas del desarrollo de este TFM, así como unas posibles líneas futuras a seguir para la mejora y ampliación del mismo.

6.1. Conclusiones

A día de hoy, no existe duda alguna de que la aplicación de las TIC al ámbito sanitario ha supuesto una revolución total de una forma significativa con la intención no sólo de quedarse sino para cambiar el paradigma de la atención socio-sanitaria en su conjunto.

La posibilidad de acceder a cualquier tipo de información y disponer de la misma de forma realmente abundante, a través de la revolución tecnológica que ha supuesto internet y las nuevas tecnologías y herramientas asociadas, es uno de los avances más revolucionarios y significativos realizados por el ser humano en los últimos tiempos.

Precisamente, es en el sector de la salud donde esto último ha sido clave para la transformación del sector y el conocimiento médico en los últimos 20 años. Dentro de este ámbito de las nuevas tecnologías asociadas a la salud, conocido como *eHealth,* se encuentra la revolución de las *apps* móviles para este sector, que están haciendo que tanto el paciente como el profesional adquieran otro papel mucho más clave en su interacción y en la sociedad. Esta nueva realidad es la que se denomina *mHealth*, en la que la conjunción de internet y movilidad se aplica al ámbito de la salud.

El hecho de compartir el conocimiento y hacerlo cada vez de forma más ilustrativa y comprensible a través de un dispositivo tecnológico totalmente versátil y autónomo, es algo que genera un espacio de comunicación abierto entre ambas partes, médico y paciente, que permite tanto entender como compartir experiencias en el sector que más preocupa al ser humano, como es el de la salud.

Dentro de este contexto presentado anteriormente es donde se enmarca el desarrollo de este TFM, con el desarrollo de una aplicación móvil para Android para el ámbito de la fisioterapia. Tras haber analizado el estado del arte de la *mHealth* en diversos sectores sanitarios, es en este sector de la fisioterapia uno de los que están sufriendo una mayor transformación digital, tanto a nivel de infraestructuras como de conocimiento digital de sus profesionales. Por lo tanto, una posible propuesta que aporte un valor diferenciado en el mismo es el desarrollo de la aplicación móvil. Con ello se pretende que la *mHealth*, aplicada a este sector, sea uno de sus aspectos más relevantes a día de hoy con el objetivo de facilitar la comunicación e interacción entre sus pacientes y profesionales.

A lo largo del desarrollo del TFM, se puede apreciar cómo, tras el establecimiento del objetivo principal del mismo, se ha ido describiendo los posibles sistemas operativos a elegir para el desarrollo de la misma. Finalmente, teniendo en cuenta aspectos como cuota de mercado del mismo, personalización y recursos necesarios para el desarrollo de aplicaciones en el sistema operativo, se eligió Android como aquel que cumplía y satisfacía todo ellos de manera significativa y con grandes cifras de mercado.

Tras los análisis previos y una vez ya decididos los recursos necesarios para poder realizar este TFM, se ha conseguido aportar ese valor añadido al ámbito de la fisioterapia mediante la realización de una aplicación que facilita la realización de las tareas propias de gestión de pacientes de los profesionales de clínicas fisioterapéuticas, así como ofrecer a los mismos pacientes la posibilidad de realizar un seguimiento de su tratamiento con la asignación de ejercicios para su recuperación y la posibilidad de reservas citas de una forma rápida, en cualquier momento y en cualquier lugar.

Para conseguir poder disponer de diversas funcionalidades a distintos perfiles de usuario en la aplicación, ésta ha sido diseñada permitiendo la autentificación de cada uno de los mismos mediante sus credenciales. De esta manera la aplicación es capaz de identificar el perfil de usuario autentificado (paciente o profesional) para ofrecerle las funcionalidades propias del mismo de una manera segura e intuitiva.

Como aspectos finales de este TFM, una vez realizadas las pruebas para comprobar que la aplicación cumplía con los requisitos iniciales establecidos, se elaboró un manual de usuario a título ilustrativo que pretende ser una guía de referencia para todo aquel usuario que pretenda usarla, mostrando de forma sencilla el funcionamiento de la misma.

Por último, y como opinión a título personal, la realización de este TFM me ha supuesto un reto ambicioso, ya que profesionalmente me dedico al desarrollo de aplicaciones móviles y para mí el hecho de poder realizar una aplicada en el ámbito de la salud me ha dado a conocer y a mejorar muchos de mis conocimientos en este ámbito. Ha sido una experiencia que me ha servido para asentar mis bases técnicas en este ámbito. Como contrapartida, el hecho de tener que realizar una aplicación móvil en solitario, sin un grupo de trabajo definido, me ha supuesto una mayor dificultad que en el ámbito profesional, debido a la complejidad de desarrollar una aplicación para móviles Android desde el inicio y por cuenta de uno propio. Sin embargo, esto me ha ayudado a aprecian aún más el valor propio y real que tiene, tanto a nivel personal como profesional, el trabajo en grupo en cualquier ámbito.

6.2. Líneas futuras de ampliación

Esta primera versión de la aplicación cubre una serie de funcionalidades, mencionadas a lo largo del TFM, principalmente enfocadas para los pacientes y profesionales de las clínicas de fisioterapia. Sin embargo, se ha descubierto la necesidad de poder añadir nuevas funcionalidades, así como de mejorar las realizadas, con el objetivo de realizar una aplicación más completa.

Además, dentro del contexto del desarrollo de aplicaciones móviles, siendo este un campo en constante evolución y aparición de nuevas funcionalidades tanto a nivel de funcionamiento, optimización y visuales, es muy recomendable que la aplicación vaya evolucionando y adaptándose tanto a los nuevos estándares de diseño como nuevas versiones del sistema operativo Android.

A continuación, se describen de forma resumida algunas de las líneas futuras posibles de implementar como ampliación y mejora de la aplicación en función de nuevas funcionalidades o mejoras de las actuales:

- ✓ Nuevas funcionalidades para la aplicación, por ejemplo:

  - Posibilidad de modificación de sus datos de perfil, con posibilidad de añadir una foto al mismo.

  - Posibilidad de modificar su nombre de usuario y contraseña, así como de recordar esta última en el caso que el usuario, a la hora de autentificarse, lo haga erróneamente más de un número concreto de intentos.

  - Posibilidad de poder marcar como completados los ejercicios asignados ya completados por el paciente.

  - Posibilidad de modificar o borrar un ejercicio asignado a un paciente por parte del profesional.

  - Posibilidad de autentificación de usuario como encargado de la clínica con funcionalidades como dar de alta/baja a profesionales de clínica o asignar reservas a profesionales.

  - Añadir un apartado o pantalla de últimos avisos o noticias, para estar informado de manera instantánea de las últimas noticias o novedades como, por ejemplo, aviso de cancelación de una reserva por algún motivo en concreto, cambio de horarios de la clínica, asignación de nuevos ejercicios como paciente, etc.

  - En relación al punto anterior, todos esos avisos o noticias a añadir se podrían completar con notificaciones en el dispositivo móvil (configurables por el usuario) con el objetivo de que el usuario pueda ser notificado en el menor tiempo posible sin la necesidad de abrir la aplicación.

  - Añadir una gestión del modelo de datos de la aplicación mediante una base de datos, para en el caso de que no se disponga de una conexión a internet estable, se puedan consultar los datos guardados anteriormente, de otras peticiones previas al servidor, en la base de datos interna de la aplicación.

- Posibilidad de añadir funciones de geolocalización para mostrar información de la clínica más cercana en primer lugar, con información de la distancia restante a la misma.

✓ Mejoras de funcionalidades para la aplicación, por ejemplo:

- Añadir *tests* unitarios o de integración para garantizar una mayor cobertura de *testing* del funcionamiento de la aplicación.

- Adaptación de la interfaz gráfica de usuario a los nuevos diseños o componentes visuales derivados de las nuevas versiones del sistema operativo lanzadas por Google. Este punto es especialmente relevante puesto que, además de la interfaz gráfica, se debe de actualizar la aplicación para que sea compatible con dichas nuevas versiones e ir restringiendo la mínima versión de funcionamiento a aquella más antigua con una cuota de mercado actual significativa

- Registro de sesiones del usuario en los servicios *web* para el guardado del estado actual de la misma, ya que actualmente sólo se está haciendo a nivel de aplicación.

Bibliografía

Bibliografía

Abdel-lah, L. D. (22 de Diciembre de 2014). *¿Qué es la mHealth?* Obtenido de EspidiDoctor: http://www.espididoctor.com/que-es-la-mhealth/#popup/1/

Alfonso. (8 de Febrero de 2012). *iOS vs Android ¿Por dónde empiezo?* Recuperado el 10 de Septiembre de 2017, de Sozpic: https://www.sozpic.com/desarrollo-ios-vs-android-por-donde-empiezo/

Android Developer. (Septiembre de 2017). *Componentes de la aplicación*. Recuperado el 10 de Septiembre de 2017, de Android Developer: https://developer.android.com/guide/components/index.html

Android Developer. (Agosto de 2017). *Estadísticas Android*. Recuperado el 10 de Septiembre de 2017, de Android Developer: https://developer.android.com/about/dashboards/index.html

Android Developer. (Septiembre de 2017). *Introducción a Android*. Recuperado el 10 de Septiembre de 2017, de Android Developer: https://developer.android.com/guide/index.html

Android Developer. (Septiembre de 2017). *Sistema de Compilación de Android*. Recuperado el 10 de Septiembre de 2017, de Android Developer: https://developer.android.com/studio/build/index.html

AndroidStudioFAQs. (8 de Diciembre de 2016). *Android Studio: ventajas, desventajas y principales características*. Recuperado el 10 de Septiembre de 2017, de AndroidStudioFAQs: https://androidstudiofaqs.com/conceptos/ventajas-desventajas-android-studio

AndroidStudioFAQs. (31 de Mayo de 2016). *La historia de Android Studio*. Recuperado el 10 de Septiembre de 2017, de AndroidStudioFAQs: https://androidstudiofaqs.com/conceptos/android-studio-historia

Apple Developer. (6 de Septiembre de 2017). *Support*. Recuperado el 10 de Septiembre de 2017, de Apple: https://developer.apple.com/support/app-store/

BeMovil. (26 de Julio de 2016). *Ventajas e Inconvenientes de Android*. Recuperado el 10 de Septiembre de 2017, de BeMovil Blog: http://www.bemovil.es/blog/ventajas-inconvenientes-android/

BeMovil. (2 de Agosto de 2016). *Ventajas e Inconvenientes del Sistema Operativo iOS*. Recuperado el 10 de Septiembre de 2017, de BeMovil Blog: https://www.bemovil.es/blog/ventajas-sistema-operativo-ios/

Birmacher, B. (27 de Enero de 2017). *State of App Development in 2016*. Recuperado el 10 de Septiembre de 2017, de Blog Bitrise: http://blog.bitrise.io/2017/01/27/state-of-app-development-in-2016.html

Cardenas, C. E. (2015). *Arquitectura de una aplicación Android*. Recuperado el 10 de Septiembre de 2017, de Platzi: https://platzi.com/blog/arquitectura-android-app/

Colegio Profesional de Fisioterapeutas Comunidad de Madrid. (2015). *Información Fisioterapia*. Recuperado el 10 de Septiembre de 2017, de Colegio Profesional de Fisioterapeutas Comunidad de Madrid: https://www.cfisiomad.org/pages/informacion_fisioterapia.aspx

Comisión Europea. (10 de Abril de 2014). *Libro Verde sobre sanidad móvil.* Recuperado el 10 de Septiembre de 2017, de http://ec.europa.eu/transparency/regdoc/rep/1/2014/ES/1-2014-219-ES-F1-1.Pdf

Diagnostrum. (21 de Agosto de 2015). *¿Qué es mHealth?* Recuperado el 10 de Septiembre de 2017, de Diagnostrum: http://blog.diagnostrum.com/2015/08/21/que-es-mhealth-2/

Diana Cristina Angarita Rodríguez, J. N. (2017). *Uso de dispositivos móviles en fisioterapia.* (R. C. Salud, Ed.) Recuperado el 10 de Septiembre de 2017, de http://scielo.sld.cu/pdf/ics/v28n2/rci1064.pdf

EFisioterapia. (27 de Marzo de 2017). *Factores clave que los fisioterapeutas deben considerar en el ecosistema digital*. Recuperado el 10 de Septiembre de 2017, de EFisioterapia: https://www.efisioterapia.net/articulos/factores-clave-que-fisioterapeutas-deben-considerar-ecosistema-digital

Elsevier, Equipo Editorial. (24 de Julio de 2014). *Atención Familiar y Comunitaria*. Recuperado el 10 de Septiembre de 2017, de Conecta Elsevier: https://www.elsevier.es/corp/conecta/atencion-familiar-y-comunitaria/la-ehealth-es-una-herramienta-mas-dentro-de-nuestra-practica-clinica-asistencial/

Fernández Silano, M. (2013). *La Salud 2.0 y la atención de la salud en la era digital.* (R. M. Mirasalda, Ed.) Recuperado el 10 de Septiembre de 2017, de

http://revistas.utp.edu.co/index.php/revistamedica/article/viewFile/8483/5675

García, D. (18 de Julio de 2012). *Aplicaciones Móviles, ¿Nativo, Web, Híbrido?* Recuperado el 10 de Septiembre de 2017, de Pixmat Studios Blog: http://www.pixmatstudios.com/blog/aplicaciones-moviles-nativo-web-hibrido/#.WZ8cKeC0P_Yhttps://blogthinkbig.com/aplicaciones-web-nativas-hibridas/

Garzón, J. (21 de Agosto de 2017). *Android 8.0 Oreo ofrece más fluidez, actualizaciones más rápidas y otras novedades*. Recuperado el 10 de Septiembre de 2017, de CNET: https://www.cnet.com/es/analisis/android-o/primer-vistazo/

Gemalto. (2016). *Presentación de las redes 5G.* Recuperado el 10 de Septiembre de 2017, de http://www.gemalto.com/brochures-site/download-site/Documents/tel-5G-networks-QandA-es.pdf

Goga, A. (14 de Junio de 2016). *De iOS 1.0 a iOS 10: Cómo ha evolucionado el sistema operativo de Apple?* Recuperado el 10 de Septiembre de 2017, de ArturoGoga: https://www.arturogoga.com/especial-de-ios-1-0-a-ios-10-como-ha-evolucionado-el-sistema-operativo-de-apple/

Google Play. (2017). *Fissios*. Recuperado el 10 de Septiembre de 2017, de Google Play: https://play.google.com/store/apps/details?id=com.tribalyte.fissiosapp&hl=es

Gutiérrez, A. F. (20 de Febrero de 2013). *Aplicaciones web vs. aplicaciones nativas vs. aplicaciones híbridas*. Recuperado el 10 de Septiembre de 2017, de Blogthinkbig.com: https://blogthinkbig.com/aplicaciones-web-nativas-hibridas

Guzmán, E. (3 de Abril de 2013). *Estrategia Móvil, ¿Desarrollo de una app nativa, web app o soluciones híbridas?* Recuperado el 10 de Septiembre de 2017, de OMExpo: http://www.omexpo.com/es/node/5067

Herraez, J. M. (13 de Abril de 2015). *TFG Aplicación Android para pacientes de fisioterapia.* Recuperado el 10 de Septiembre de 2017, de UvaDoc: https://uvadoc.uva.es/bitstream/10324/11535/1/TFG-G%201113.pdf

Herrmann, E. (Mayo de 2017). *Solo Android o iOS: se nos acaban las alternativas*. Recuperado el 10 de Septiembre de 2017, de AndroidPit: http://www.androidpit.es/android-o-ios-se-nos-acaban-las-alternativas

IT Reseller Tech & Consulting. (24 de Mayo de 2017). *Android acapara el 86% del mercado mundial de smartphones*. Recuperado el 10 de Septiembre de 2017, de IT Reseller Tech & Consulting: http://www.itreseller.es/en-cifras/2017/05/android-acapara-el-86-del-mercado-mundial-de-smartphones

Karpouzis, T. (24 de Agosto de 2015). *Android Architecture*. Recuperado el 10 de Septiembre de 2017, de AndroidPub: https://android.jlelse.eu/android-architecture-2f12e1c7d4db

Landa, H. (30 de Agosto de 2015). *Evolución de Dispositivos Móviles*. Recuperado el 10 de Septiembre de 2017, de Line: https://line.do/es/evolucion-de-dispositivos-moviles/pb7/vertical

Landau, D. M. (14 de Agosto de 2015). *How 5G will Power the Future Internet of Things*. Recuperado el 10 de Septiembre de 2017, de iQ Intel: https://iq.intel.com/how-5g-will-power-the-future-internet-of-things/

Leira, A. (2015a). *¿Cúal es la estructura de un proyecto en Android Studio?* Recuperado el 10 de Septiembre de 2017, de AndroidStudioFAQs: https://androidstudiofaqs.com/conceptos/cual-es-la-estructura-de-un-proyecto-en-android-studio

Leira, A. (2015b). *¿Qué es el androidmanifest en Android Studio?* Recuperado el 10 de Septiembre de 2017, de AndroidStudioFAQs: https://androidstudiofaqs.com/conceptos/que-es-el-androidmanifest

Leira, A. (2015c). *¿Qué es gradle en Android Studio?* Recuperado el 10 de Septiembre de 2017, de AndroidStudioFAQs: https://androidstudiofaqs.com/conceptos/que-es-gradle-en-android-studio

Manuel Báez, Á. B. (s.f.). *Introducción a Android.* (U. C. Madrid, Ed.) Recuperado el 10 de Septiembre de 2017, de http://www.it-docs.net/ddata/18.pdf

Martel, Y. (2015). *Las TIC sanitarias: eHealth*. Recuperado el 10 de Septiembre de 2017, de OPINNO: https://www.opinno.com/es/contenido/las-tic-sanitarias-ehealth

Maxwell, E. (26 de Enero de 2017). *The MVC, MVP, and MVVM Smackdown*. Recuperado el 10 de Septiembre de 2017, de Realm: https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/

Megali, T. (8 de Abril de 2016). *How to Adopt Model View Presenter on Android*. Recuperado el 10 de Septiembre de 2017, de EnvatoTuts+:

https://code.tutsplus.com/tutorials/how-to-adopt-model-view-presenter-on-android--cms-26206

Monasterio, A. (6 de Junio de 2017). *App Fissios*. Recuperado el 10 de Septiembre de 2017, de El Blog de Fisioterapia: http://www.blogdefisioterapia.com/app-fissios/

Ortiz, S. J. (3 de Mayo de 2016). *La Evolución de iOS: Un Paseo desde iOS 1 hasta iOS 9*. Recuperado el 10 de Septiembre de 2017, de iPadizate: https://www.ipadizate.es/2016/05/03/evolucion-ios-1-ios-9/

Pérez, D. (18 de Mayo de 2017). *Las enormes cifras de Google: 2.000 millones de usuarios Android y más*. Recuperado el 10 de Septiembre de 2017, de El Androide Libre: https://elandroidelibre.elespanol.com/2017/05/cifras-de-google-2000-millones-usuarios-android.html

Portela, E. (2 de Abril de 2015). *Introducción a PHP*. Recuperado el 10 de Septiembre de 2017, de Blog Portelab: https://portelab.blogspot.com.es/2015/04/01-caracteristicas-de-php-introduccion.html

Pozo, J. D. (2011). *Introducción a los dispositivos móviles.* (U. O. Catalunya, Ed.) Recuperado el 10 de Septiembre de 2017, de https://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_dispositivos_moviles/Tecnologia_y_desarrollo_en_dispositivos_moviles_(Modulo_2).pdf

Prajesh, S. (28 de Enero de 2016). *Android: MVC, MVP, MVVM*. Recuperado el 10 de Septiembre de 2017, de Tothenews Blog: http://www.tothenew.com/blog/androidmvc-mvp-mvvm/

Rodríguez, C. B. (25 de Noviembre de 2016). *El paciente, la semilla del cambio para vencer barreras y fomentar el uso de la eSalud*. Recuperado el 10 de Septiembre de 2017, de ElGlobal: http://www.elglobal.net/suplementos-y-especiales/autocuidado/el-paciente-la-semilla-del-cambio-para-vencer-barreras-y-fomentar-el-uso-de-la-esalud-XA626617

Rus, C. (17 de Octubre de 2016). *La evolución de iOS desde sus orígenes: una carrera para ser el mejor sistema operativo móvil de la historia*. Recuperado el 10 de Septiembre de 2017, de AppleSfera: https://www.applesfera.com/ios/la-evolucion-de-ios-desde-sus-origenes-una-carrera-para-ser-el-mejor-sistema-operativo-movil-de-la-historia

Seta, L. D. (13 de Noviembre de 2008). *Introducción a los servicios web RESTful*. Recuperado el 10 de Septiembre de 2017, de Blog Dos Ideas: https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful

Sgoliver. (11 de Agosto de 2012). *Componentes de una aplicación Android*. Recuperado el 10 de Septiembre de 2017, de Blog Sgoliver: http://www.sgoliver.net/blog/componentes-de-una-aplicacion-android/

Sgoliver. (20 de Diciembre de 2014). *Entorno de desarrollo Android (Android Studio)*. Recuperado el 10 de Septiembre de 2017, de Sgoliver: http://www.sgoliver.net/blog/entorno-de-desarrollo-android-android-studio/

Sgoliver. (28 de Diciembre de 2014). *Estructura de un proyecto Android (Android Studio)*. Recuperado el 10 de Septiembre de 2017, de Sgoliver: http://www.sgoliver.net/blog/estructura-de-un-proyecto-android-android-studio/

SlideShare. (30 de Septiembre de 2016). *Android vs Windows*. Recuperado el 10 de Septiembre de 2017, de Slideshare: https://www.slideshare.net/dineshsahu44/android-vs-windows%20

Smith, C. (13 de Agosto de 2017). *75 Amazing Android Statistics and Facts (August 2017)*. Recuperado el 10 de Septiembre de 2017, de DMR: https://expandedramblings.com/index.php/android-statistics/

Stelapps. (2017). *Historia de Android*. Recuperado el 10 de Septiembre de 2017, de Stelapps: http://stelapps.com/historia

The App Date. (2016). *Informe 50 Mejores Apps de Salud en Español.* (T. A. Intelligence, Ed.) Recuperado el 10 de Septiembre de 2017, de http://boletines.prisadigital.com/Informe-TAD-50-Mejores-Apps-de-Salud.pdf

Tudela, J. A. (4 de Febrero de 2010). *PFC Desarrollo de aplicaciones para dispositivos móviles sobre la plataforma Android de Google.* Recuperado el 10 de Septiembre de 2017, de https://e-archivo.uc3m.es/bitstream/handle/10016/6506/PFC_Jaime_Aranaz_Tudela_2010116132629.pdf?sequence=1

Universidad Politécnica de Valencia. (14 de Diciembre de 2012). *Introducción - ¿Qué es Android?* Recuperado el 10 de Septiembre de 2017, de Blog Historia de la Informática: http://histinf.blogs.upv.es/2012/12/14/android/

Universidad Politécnica de Valencia. (Septiembre de 2017). *Componentes de una aplicación*. Recuperado el 10 de Septiembre de 2017, de Universidad Politécnica de Valencia: http://www.androidcurso.com/index.php/modulo-fundamentos/31-unidad-1-vision-general-y-entorno-de-desarrollo/149-componentes-de-una-aplicacion

Varilas, V. (25 de Abril de 2016). *¿Qué es y para que sirve? iOS*. Recuperado el 10 de Septiembre de 2017, de Pacmac: http://pacmac.es/que-es-y-para-que-sirve-ios/

Villalta, P. A. (Octubre de 2014). *Generación de las Tecnologías Móviles*. Recuperado el 10 de Septiembre de 2017, de Android Desarrollo de App Para Móviles: http://www.androidestudio.com/2014/10/generaciones-de-las-tecnologias-moviles.html

Anexo – Presupuesto económico

Anexo – Presupuesto económico

En este anexo se estimará el cálculo del presupuesto económico que ha supuesto este trabajo. Se analizarán todos los factores que han influido en la aplicación y se calculará un presupuesto final.

Para la realización de este trabajo se ha hecho uso de programas de software libre y soporte informático ya adquirido previamente, esto ha implicado que el gasto inicial de la aplicación en material de trabajo haya sido nulo. Por lo que el único coste de la aplicación será el de la mano de obra.

Hay muchos factores que influyen en el coste de una aplicación. En primer lugar, se ven los parámetros más importantes que influyen en el precio de una aplicación:

- Complejidad de la aplicación: existen aplicaciones sencillas que se desarrollan en pocas horas y juegos o redes sociales complejas que requieren decenas de miles de horas de trabajo y un equipo amplio multidisciplinar.

- Tipos de contenidos: estáticos o que se puedan actualizar dinámicamente.

- Acceso a datos: ¿va a necesitar la aplicación conectarse a servidores para realizar búsquedas, actualizar su información en tiempo real y mostrar los resultados?

- Geoposicionamiento: ¿incluye información dependiente de la localización del usuario? Por ejemplo, se podría mostrar información de la tienda más cercana.

- Realidad aumentada: por ejemplo, ¿queremos mostrar una imagen en 3D de una televisión cuando estamos viéndola en un catálogo impreso?

- Pasarela de pago: la aplicación, ¿tiene opciones de pago o se utiliza para vender productos? ¿Necesita integrar el pago a través de las tiendas de apps u otras pasarelas de pago?

- Registro de usuarios: ¿es necesario incluir un registro de usuarios? ¿Qué datos se deben recoger?

- Envío de notificaciones *push*: en general, es interesante incluir la posibilidad del envío de mensajes personalizados a los usuarios a través de la aplicación que han descargado. ¿Es un requisito? ¿Qué tipo de notificaciones? ¿Es necesario crear una interfaz de gestión de notificaciones o se debe integrar con un sistema existente?

- Integración con otros sistemas: la complejidad del desarrollo aumenta considerablemente en el caso de tener que integrar la aplicación con sistemas existentes de la empresa (gestor de contenidos, bases de datos, gestión de usuarios, envío de notificaciones push, etc.) y puede haber mucha variación en función del sistema existente.

- Diseño gráfico: evidentemente no es lo mismo un diseño sencillo con menús y páginas tipo ficha informativa que aplicaciones que incluyan juegos.

- Plataformas: el número de plataformas en las que deba funcionar la aplicación es determinante para calcular el coste.

- Aplicaciones nativas/híbridas: las aplicaciones se pueden hacer de forma nativa o de forma híbrida que permiten la generación de aplicaciones multiplataforma con un único desarrollo. En el caso de aplicaciones nativas se consigue una mayor calidad y rendimiento con un coste mayor,

mientras que las aplicaciones híbridas ofrecen menor rendimiento, pero el coste es sensiblemente inferior.

- Coste/hora del programador de apps móviles: el coste de una hora de programación puede variar desde los 20 € de un programador junior sin experiencia hasta los más de 120 € que cobran las consultoras por programadores experimentados.

En función de los factores descritos anteriormente y ateniendo a las características de la aplicación, se puede estimar que el coste total de la aplicación es de 7500 €, ya que se han estimado 300 horas de trabajo real de programación. Se ha tomado como referencia el salario por hora de un programador *junior* con experiencia (entre 20 y 30 €/hora) y se ha fijado en 25 €/hora.