Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Aplicación móvil para la consulta de información sanitaria de pacientes renales

Autor: José López García
Tutora: Laura María Roa Romero

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Aplicación móvil para la consulta de información sanitaria de pacientes renales

Autor:
José López García

Tutora:
Laura María Roa Romero
Catedrática de Universidad

Departamento de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2015

Trabajo fin de Grado: Aplicación móvil para la consulta de información sanitaria de pacientes renales

Autor: José López García

Tutora: Laura María Roa Romero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2015

El Secretario del Tribunal

A mis padres, familia, amigos y profesores

Agradecimientos

Con la conclusión de este proyecto pretendo cerrar una importante etapa de mi vida, llena de momentos dulces y amargos a partes iguales. Y como no podía ser de otra manera, lo hago agradeciendo a todos los que contribuyeron a que esto fuera posible.

Doy las gracias a mis padres por todo lo que me han dado, y por haber sacrificado tantas cosas en su vida para que yo pueda disfrutar de un título universitario. Gracias a mi hermana Claudia, por regalarme momentos amenos y porque va a ser una gran enfermera. Gracias a Selua por todos los buenos momentos que hemos pasado durante los años univesitarios y pre-universitarios. Gracias a mi primo Rubén, porque para mí es un hermano y porque con él he pasado muchos de los mejores momentos de mi infancia. Y, por supuesto, gracias a mi abuela Mª Josefa, porque junto a ella he pasado muchos de los mejores momentos de mi vida, y porque sólo ha sabido regalar gestos cariñosos a todos sus nietos.

Gracias a todos mis amigos de la Residencia Militar de Estudiantes San Hermenegildo, porque ellos me enseñaron el concepto de la amistad.

Y no podía faltar mi agradecimiento hacia todos mis profesores de la Escuela Superior de Ingenieros de la Universidad de Sevilla, los cuales me enseñaron muchas y muy interesantes técnicas ingenieriles. En especial, gracias a mi tutora Laura María Roa Romero, por toda la ayuda prestada, y a Jorge Calvillo, por haberme orientado tanto y tan bien durante el desarrollo de este Trabajo Fin de Grado.

A todos vosotros: gracias.

José López García

Sevilla, 2015

Resumen

Las tecnologías móviles ayudan cada vez más a acortar la distancia entre médicos y pacientes, al permitir consultar a los profesionales de la salud sin necesidad de desplazamientos. Entre los mayores beneficiarios se encuentran los enfermos crónicos que, para llevar una vida lo más normal posible, necesitan vigilar ciertas constantes relacionadas con su enfermedad. A través de aplicaciones móviles los médicos pueden aconsejar a los pacientes sobre las dosis necesarias en cada momento, según los datos que le envían los interesados sobre su actividad y medicación actual. Para personas con movilidad reducida es posible establecer una comunicación interactiva con sus médicos, vía ordenador o teléfono móvil e incluyendo una cámara para observar al paciente, como si se tratara de una consulta normal.

El presente trabajo trata de dar un paso adelante en la materialización del deseo de acercar aún más las tecnologías móviles al campo de la sanidad. Se aborda el desarrollo de una aplicación móvil para la consulta de información sanitaria de pacientes renales monitorizados en su hogar, cuyo principal objetivo es facilitar el seguimiento médico de los pacientes renales durante los tiempos inter-consulta. De esta manera, el médico podrá conocer en tiempo real la situación de sus pacientes entre dos consultas sucesivas pudiendo actuar de manera inmediata en caso de ser necesario.

Abstract

Mobile technologies are increasingly helping to bridge the gap between doctors and patients (for example allowing patients to consult their doctors without actually having to leave their own home). Chronic patients are among the greatest beneficiaries of mobile technology because they need to monitor certain medical variables in their daily life. Through mobile applications physicians can advise their patients about necessary dosage at all times, according to the data they send about their medical activity and current medication. Also, disabled people can establish an interactive communication with their doctors via computer or a mobile phone, maybe including a camera for a better medical assistance.

This work is a step forward in materializing the desire of bringing mobile technologies to the field of health even more. In this project, a mobile application is developed so to allow physicians to supervise remotely selfcare information of their renal patients at home. This way, the physician can know in real time the status of their patients between two successive consultations, allowing him to take immediate action if necessary.

Índice

Agradecimientos **ix**

Resumen **xi**

Abstract **xii**

Índice de Tablas **xv**

Índice de Figuras **xvii**

1 Introducción **1**
1.1 Contexto *1*
1.2 Fundamentos médicos del trabajo *2*
1.3 Justificación del Proyecto *3*
1.3.1 Evolución de los smartphones 3
1.3.2 Sistema Android 3
1.4 Objetivos *8*
1.5 Planificación del trabajo *8*

2 Materiales y métodos **11**
2.1 Sistema e-Nefro *11*
2.2 Materiales empleados en el desarrollo técnico del proyecto *12*
2.3 Metodología a seguir para el desarrollo del sistema *14*

3 Resultados **17**
3.1 Introducción a la aplicación *17*
3.2 Descripción del servicio web e interacción con la app *19*
3.3 Alcance de la aplicación y diseño de la interfaz *20*
3.3.1 Diseño de la estructura lógica de navegación 21
3.3.2 Diseño de la interfaz 23
3.4 Implementación y desarrollo de la aplicación y sus menús *26*
3.4.1 Menú de acceso 26
3.4.2 Menú central (MCentralActivity) 28
3.4.3 Menú de lista de pacientes (MListaActivity) 31
3.4.4 Menú paciente (MPacActivity) 32
3.4.5 Menú de consulta de citas (MCitasActivity) 37
3.4.6 Menú de consulta de mensajes (MMensActivity) 39
3.4.7 Menú de consulta de incidencias (MUIncidActivity) 40
3.4.8 Menú de configuración (MConfigActivity) 41
3.4.9 Tabla de errores 41

4 Conclusiones y líneas futuras **43**
4.1 Conclusiones *43*
4.2 Futuras líneas de ampliación y mejora *44*

5 Referencias **48**

Anexo A: Tabla de métodos del servidor **50**

ÍNDICE DE TABLAS

Tabla 1. Comparativa de la velocidad para diferentes métodos 14
Tabla 2. Posibles errores durante el uso de la aplicación. 41
Tabla 3. Tabla de métodos del servidor. 51

ÍNDICE DE FIGURAS

Figura 1. Diálisis peritoneal ambulatoria continua. 3
Figura 2. Versiones de Android más usadas actualmente. 4
Figura 3.Crecimiento de aplicaciones Android disponibles. 5
Figura 4. Captura de pantalla Medscape. 6
Figura 5. Captura de pantalla Epócrates. 6
Figura 6. Captura de pantalla Skyscape. 7
Figura 7. Plan de trabajo. 9
Figura 8. Fragmento de código JSON. 12
Figura 9. Ilustración del proceso de recolección remota. 18
Figura 10. Recolección remota mejorada (Capítulo 4). 19
Figura 11. Módulo MPac. 22
Figura 12. Esquema global de la aplicación. 23
Figura 13. Menú de acceso (izquierda) y menú central (derecha). 24
Figura 14. Lista de pacientes (izquierda), filtro de búsqueda (centro) y menú de paciente (derecha). 24
Figura 15. Lista de pacientes. 25
Figura 16. Lista de mensajes. 25
Figura 17. Menú de acceso original (izquierda), menú de acceso rediseñado (derecha). 26
Figura 18. Proceso de diseño del menú de acceso. 27
Figura 19. Diferencia entre un hilo secundario secuencial (izquierda) y el hilo principal (derecha). 28
Figura 20. Estructura de una clase POJO. 28
Figura 21. Menú central (MCentral). 29
Figura 22. ¿Qué ocurriría si el médico solicita la información antes de que ésta esté disponible? 30
Figura 23. Barra de progreso de autenticación y descarga (izquierda), error personalizado (derecha). 31
Figura 24. Diseño del menú Lista de Pacientes. 32
Figura 25.Menú de paciente (MPac). 33
Figura 26. Se puede acceder a MPac desde casi cualquier menú. 33
Figura 27. Menú de registro de variables (MPacMVar). 35
Figura 28. Diseño adoptado para acomodar la información de un registro de variable. 35
Figura 29. Diseño final que muestra varios registros de variables (MPacMVarMVars). 36
Figura 30. Menú de incidencias para un paciente (MPacMInc). 37
Figura 31. Menú de plan de cuidados (MPacMPCuid). 37
Figura 32. Menú de citas (MCitas). 38
Figura 33. Menú de mensajes (MMens). 40
Figura 34. Menú de inciencias globales, de todos los pacientes (MUIncid). 41
Figura 35. Estadística [24] sobre uso de diferentes sistemas operativos en España. 44

Figura 36. Método de descarga con almacenamiento caché y temporizador interno. 45
Figura 37. Posible mejora en la manera de mostrar los pacientes. 46
Figura 38. Técnica multi-pane layout de Android. 46

1 INTRODUCCIÓN

1.1 Contexto

La realización de este trabajo fin de grado se enmarca bajo un proyecto de investigación actualmente en desarrollo por el Grupo de Ingeniería Biomédica, el proyecto eNefro.

El proyecto eNefro es un esfuerzo multicéntrico entre la Universidad de Sevilla y diversos hospitales nacionales. El equipo investigador está formado por especialistas en Ingeniería Biomédica así como un equipo médico integrado por cuatro hospitales a nivel nacional, cuyo propósito es establecer una arquitectura extensible y adaptable para la asistencia remota de pacientes en pre-diálisis y diálisis peritoneal (DP).

Siendo el objetivo de e-Nefro el de hacer frente a los inconvenientes de la diálisis peritoneal domiciliaria, cabe mencionar que su principal limitación es la necesidad por parte del médico de disponer de un ordenador para poder consultar la información de sus pacientes. Esta limitación es precisamente el motor que impulsa la creación de este proyecto.

El estado actual de desarrollo tecnológico de las TICs permite dar lugar a un proyecto que conjugue los conceptos de tecnología, movilidad, y atención médica inmediata. Concretamente, esto es posible gracias al crecimiento exponencial que está experimentando el uso de los dispositivos smartphone en la sociedad actual. Algunos beneficios directamente derivados del desarrollo de este proyecto son los siguientes:

- La movilidad de las profesiones médicas se vería en gran parte aliviada. El médico podría acceder en cualquier momento y lugar a los registros de un determinado paciente, pudiendo llevar a cabo la acción más apropiada cuando proceda.
- Los pacientes podrían informar a sus médicos cuando lo considerasen necesario. Por ejemplo, ante una incidencia o un efecto adverso consecuencia del actual plan de cuidados.
- Los tiempos entre consultas se utilizarían de manera más productiva. El hecho de que un determinado paciente pueda estar en contacto con su médico a través de una interfaz móvil aporta seguridad al tratamiento y, sobre todo, tranquilidad a ambas partes.

Con el antiguo sistema, donde no existía un control médico inter-consulta, los médicos desconocían el estado de sus pacientes hasta la próxima revisión. Durante dicho tiempo es posible que pudiera ocurrir alguna incidencia sobre la que se debiera actuar lo más tempranamente posible. El principal motivo que impulsa la creación de este proyecto es, precisamente, la necesidad de eliminar estos tiempos muertos y permitir al médico monitorizar el estado de sus pacientes en tiempo real.

Puede afirmarse, por tanto, que el objetivo final del proyecto es facilitar a los médicos el acceso a los historiales más recientes de sus pacientes. Así, se reduciría el riesgo ante posibles incidentes que pudieran ocurrir durante los tiempos inter-consulta. Además de conocer en tiempo real y en cualquier lugar e instante la situación de sus pacientes, los médicos podrían actuar en caso de ser necesario, efectuando por ejemplo un cambio en el plan de cuidados, o demandando al paciente la medida de un determinado parámetro fisiológico.

Sin embargo el proyecto no está exento de inconvenientes, los cuales a su vez establecen las bases de posibles líneas futuras de ampliación. Por ejemplo, sólo los médicos usuarios de dispositivos Android podrán beneficiarse de la aplicación móvil. Desarrollos para dispositivos iPhone, Windows Phone o BlackBerry no pueden contemplarse debido principalmente a las severas limitaciones de tiempo.

Por último cabe explicar cómo complementa este proyecto a la arquitectura e-Nefro.

Desde el punto de vista técnico. El alcance de este proyecto es únicamente el desarrollo de la aplicación móvil, la cual a su vez se alimenta de una base de datos almacenada en un servidor remoto. Este proceso ocurre a través de un servicio web, el cual se desarrolla de manera totalmente independiente a este proyecto. De esta manera, el desarrollo del código del servidor ocurre de forma paralela y ajena al proyecto. Ambas partes, cliente y servidor, buscan complementarse lo más fielmente posible para una mejor experiencia del usuario final.

Desde el punto de vista del usuario. En este proyecto se tiene en cuenta únicamente al médico como usuario de la aplicación, lo que significa que el paciente no interviene en ningún momento en el proceso de diseño. El médico sólo podrá solicitar información a la base de datos, pero no podrá escribir información en ella. El paciente se comunicará con el médico a través de otra aplicación desarrollada de manera independiente a este proyecto, y que no guarda relación alguna con el mismo.

1.2 Fundamentos médicos del trabajo

La Enfermedad Renal Crónica (ERC), al igual que otras enfermedades crónicas de gran prevalencia como la hipertensión arterial y la diabetes mellitus, son claros ejemplos de la necesidad de unificación de criterios y coordinación entre los diferentes profesionales implicados en su atención, desde el laboratorio clínico pasando por la atención primaria hasta la atención especializada. En particular, la ERC se ha convertido en una patología que ha pasado de ser una enfermedad grave que afectaba a pocos individuos y que debía ser atendida por nefrólogos, a una patología común de gravedad variable, que precisa de su conocimiento por otras especialidades y por las autoridades sanitarias.

La ERC es un problema emergente en todo el mundo. En España, según los resultados del estudio EPIRCE (Epidemiología de la Insuficiencia Renal Crónica en España), se estimó que aproximadamente el 10% de la población adulta sufría de algún grado de ERC, existiendo diferencias importantes de edad: afecta al 3,3% de la población de entre 40-64 años, al 22% de los mayores de 64, y al 40% de los mayores de 80 [1] [2].

La diálisis peritoneal (DP) es un procedimiento que permite depurar líquidos y electrolitos en pacientes que sufren de insuficiencia renal aguda. Está basada en el hecho fisiológico de que el peritoneo es una membrana natural semipermeable, que permite pasar agua y distintos solutos desde los capilares sanguíneos peritoneales al líquido dializado.

La técnica de la diálisis peritoneal permite infundir en la cavidad peritoneal un líquido de composición similar al líquido extracelular, dejándolo un tiempo en el interior del peritoneo. Más tarde se producirá la difusión y osmosis de tóxicos y electrolitos desde la sangre hasta el líquido introducido. La Figura 1 muestra de manera esquemática la técnica de la diálisis.

La diálisis peritoneal domiciliaria se introduce como una alternativa a la diálisis peritoneal convencional, donde el paciente debía acudir hasta 5 veces por semana al centro de salud para llevar a cabo la rutinaria limpieza de sangre. La diálisis domiciliaria se introdujo alrededor de los años 1970, aunque no fue sino a partir de la década de los 90 cuando comenzó a adoptarse esta alternativa como solución definitiva a la entonces inevitable dependencia con el centro médico. A día de hoy es la opción preferida por los pacientes renales, porque pueden disfrutar de un estilo de vida más cómodo y menos dependiente del centro de salud.

No cabe duda de que la diálisis domiciliaria introduce múltiples beneficios, tanto para los pacientes (un estilo de vida más regular) como para los centros de salud (aprovechamiento de personal médico e instalaciones de una manera más eficaz). Sin embargo, también introduce algunos inconvenientes que se pasan a comentar a continuación. El objetivo último de este trabajo fin de grado es el de hacer frente a dichos inconvenientes.

- El profesional encargado del seguimiento médico de un determinado paciente renal pierde información sobre lo que ocurre entre dos consultas sucesivas. Entre dichas consultas es el mismo paciente el que se practica la diálisis peritoneal en su propio domicilio, por lo que se desconoce por completo cómo se lleva a cabo la diálisis y, sobre todo, no se lleva un control sobre las variables fisiológicas del paciente durante este tiempo. Esto puede dar lugar a situaciones potencialmente peligrosas.
- El paciente no puede informar de una manera rápida y eficaz a su médico sobre posibles incidencias, bien por falta de tiempo, bien por falta de medios, bien por desconocimiento de la peligrosidad inherente a una incidencia específica a la que el paciente no le dio demasiada importancia.
- Existe una sensación psicológica de inseguridad por parte del paciente al no estar supervisado por su médico durante los tiempos inter-consulta.

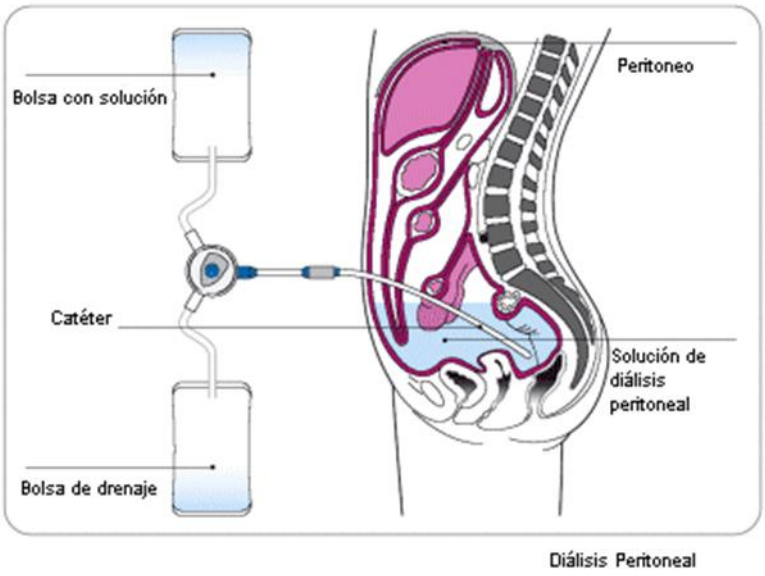


Figura 1. Diálisis peritoneal ambulatoria continua.

1.3 Justificación del Proyecto

1.3.1 Evolución de los smartphones

Desde que en 1993 aparece el IBM Simon hasta la llegada del Samsung Galaxy S6 en 2015, la innovación tecnológica en las telecomunicaciones ha sido constante. Los primeros smartphones eran lentos, sin cámara, sus pantallas eran pequeñas y con poca resolución, la batería duraba poco, etc. Y ahora tenemos teléfonos extremadamente rápidos, con velocidades de conexión de 4G LTE, pantallas táctiles de alta resolución y con procesadores muy potentes.

El primer sistema operativo apareció en 1996 bajo el nombre de Palm OS y fue precursor de los grandes avances tecnológicos hoy existentes, hasta que en 2001 nació Symbian, de Nokia. PalmOS, cuya cuota de mercado disminutó notablemente, se vio en gran parte desplazado por Symbian, el cuál ofrecía una interfaz gráfica más amigable y una velocidad de operación más alta. Otros sistemas operativos que pasaron a la historia son Imode, Maemo y Meego.

En la actualidad Android e iOS se posicionan claramente como los sistemas operativos dominantes, seguidos por Windows Phone, Blackberry OS, WebOS y Bada. Permiten convertir el smartphone en una consola de videojuegos, GPS o reproductor de música. Asimismo permiten intercambiar mensajes usando aplicaciones de mensajería instantánea (una de las funciones más usadas actualmente) e incluso permiten realizar operaciones financieras de una manera bastante segura.

Android fue creado inicialmente para teléfonos en 2008. En febrero de 2011 apareció la versión 3.0 para tablets y ocho meses después apareció la versión 4.0, la cual unificó los dos sistemas, smartphones y tablets, bajo las mismas reglas de desarrollo y operación.

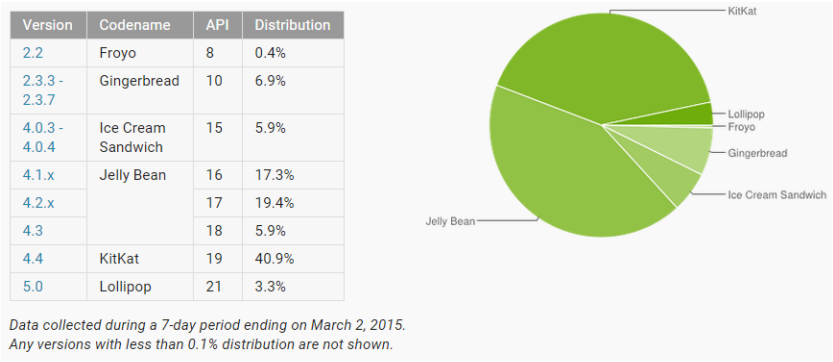
1.3.2 Sistema Android

Se ha decidido desarrollar la aplicación para la plataforma Android. El motivo de esta elección fue que la mayoría de personas, aproximadamente un 78%, utiliza dispositivos Android. Se podría extender el trabajo para el resto de plataformas (iOS, Windows Phone o BlackBerry) utilizando la API de Xamarin, aunque por limitaciones de tiempo se ha descartado esta posibilidad.

Desde su aparición en noviembre de 2007, el sistema operativo Android ha ido evolucionando en forma de diferentes versiones: desde la versión beta (12 de noviembre de 2007) hasta la más actual versión 21, también

conocida como Android Lollipop, que se dio a conocer el 3 de noviembre de 2014. Dependiendo de para qué versiones de Android se desee desarrollar, el código vendrá condicionado de una manera u otra. El siguiente paso lógico en el desarrollo de la aplicación sería escoger una versión a partir de la cual trabajar, teniendo en cuenta que mientras más versiones se soporte, mayores serán las probabilidades de que un determinado médico(y por tanto sus pacientes) puedan aprovecharse de la aplicación. En la Figura 2, extraída de la página oficial de desarrolladores Android, puede consultarse cuáles son las versiones más utilizadas a fecha del 2 de marzo de 2015.Puesto que las versiones Froyo y Gingerbread aún están en uso, se contemplarán durante el proceso de desarrollo de la aplicación. Lógicamente esto restringirá el acceso a las bondades que ofrecen las versiones más recientes (como el novedoso concepto de Material Design de Lollipop). No obstante esto no será un problema para el tipo de aplicación de que se trata.

Finalmente habrá de evaluarse el sistema en dos planos diferentes: el plano técnico y el plano de funcionalidad y usabilidad. Para el primero podrá hacerse las pruebas pertinentes a nivel local, desde la perspectiva del desarrollador. Para la funcionalidad y usabilidad podrá acudirse a un médico y pacientes ficticios. Para mejorar la usabilidad aún más, así como mejorar en lo posible la aplicación desde la perspectiva del usuario, podrá idearse unas encuestas de satisfacción a rellenar por médicos reales usuarios de la aplicación.



Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x		17	19.4%
4.3		18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%

Data collected during a 7-day period ending on March 2, 2015.
Any versions with less than 0.1% distribution are not shown.

Figura 2. Versiones de Android más usadas actualmente.

1.3.2.1 Principales diferencias entre Android y otros sistemas operativos

Hay muchas diferencias entre Android y otros sistemas, siendo la mayor de estas diferencias la filosofía de desarrollo, que se compone de los siguientes puntos:

1. Android es un sistema abierto, por lo que es posible saber en todo momento cuál es el código subyacente desarrollado por Google, la compañía que está detrás del mismo. Por otra parte, iOS y Windows Phone, sus principales competidores, no ofrecen la posibilidad de código abierto al ser dichos Sistemas Operativos de código propietario.
2. Servicios de mapas. Google Maps, tiene a su disposición el servicio Street View frente a iOS que, sin embargo, utiliza en sus mapas vistas tridimensionales. Frente a los dos gigantes, Windows Phone ofrece menos funcionalidades.
3. Imágenes y publicación en redes sociales, navegación, notificaciones y chat. Aunque a diferencia de iOS6, en Android se necesitan aplicaciones de terceros para publicar en Facebook y Twitter, desbanca claramente a Windows Phone en lo que a navegadores, notificaciones y chat se refiere.
4. Personalización y seguridad. La seguridad de los dos grandes es muy elevada; sin embargo, rootear un Android es una acción legal y permitida por Google, mientras que hacer Jailbreak a un iOS, no. Así que el primero permite al usuario tocar y personalizar en profundidad las opciones internas del teléfono o tableta, e instalar todo tipo de aplicaciones.

1.3.2.2 Desarrollo de aplicaciones para Android

Android pone a disposición de sus usuarios un mercado de aplicaciones conocido como Play Store. Una de las revoluciones que ofrece, es que cualquier usuario puede desarrollar su propia aplicación y ofrecerla al resto del mundo tras pagar una cuota única de registro de 25 euros. A diferencia de la App Store de Apple, Android es mucho más permisivo a la hora de dejar que desarrolladores cuelguen sus aplicaciones en la Play Store. Este hecho se refleja en la gran cantidad de aplicaciones actualmente disponibles en la Play Store (ver Figura 3), superando los 1.3 millones en la última estadística [3] llevada a cabo en agosto de 2014.

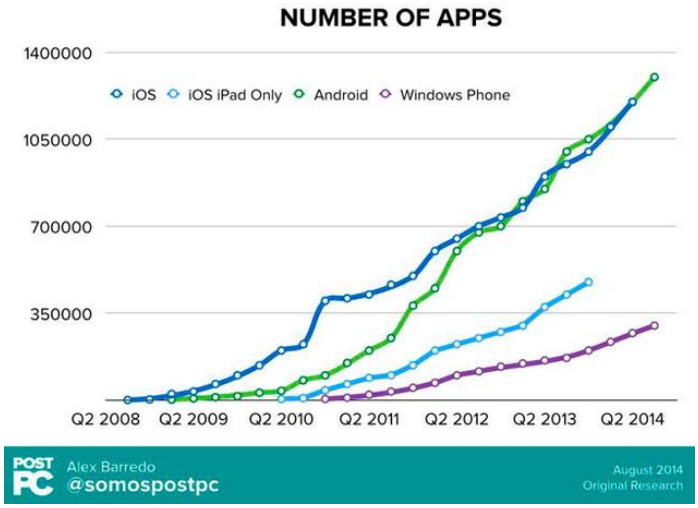


Figura 3.Crecimiento de aplicaciones Android disponibles.

Habitualmente, las aplicaciones para este sistema se desarrollan en el lenguaje Java con las librerías ofrecidas por el Android Software Development Kit (SDK), pero también pueden crearse a través de un Kit de Desarrollo Nativo o extensiones en C o C++, Google App Inventor, un entorno visual para novatos y varios marcos de la web. También es posible usar las bibliotecas Qt gracias al proyecto "Necesitas SDK". El requisito primordial para la realización de apps Android es un conocimiento sólido del lenguaje Java, a partir del cual es posible entender todas las clases ofrecidas por Google en su kit SDK.

1.3.2.3 Aplicaciones móviles en el campo de la sanidad creadas para el sistema Android

Desde el campo de la medicina nos están llegando aplicaciones que, sin pretender sustituir a los profesionales, sirven de apoyo documental a éstos, ayudan a la detección de diversas patologías y su diagnóstico y al diseño de tratamientos farmacológicos y además, gozan del valor añadido que ofrece su movilidad. A continuación se citan [4] algunas de las aplicaciones médicas que gozan de más popularidad actualmente:

- Medscape. Creada por WebMD, aparece en este sistema en enero de 2011. Dirigida a los profesionales de la salud y estudiantes de medicina, contiene más de 7.000 referencias de medicamentos, más de 3.500 referencias clínicas de enfermedades, más de 2.500 imágenes clínicas y vídeos de procedimientos, y una herramienta correctora de interacción entre medicamentos, entre otras utilidades. En la Figura 4 se muestra una captura de la aplicación.

Figura 4. Captura de pantalla Medscape.

- Epocrates. Creada por la compañía del mismo nombre, se trata de una guía de acceso rápido con información sobre enfermedades, diagnósticos y medicamentos. Calcula dosis e informa de interacciones. En la Figura 5 se muestra una captura de pantalla de la aplicación.

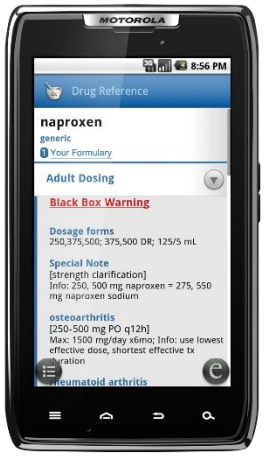


Figura 5. Captura de pantalla Epócrates.

- Skyscape. Desarrollada por la compañía estadounidense Skyscape Medpresso, ofrece a los profesionales de atención de la salud y estudiantes acceso a una selección sólida de calculadoras médicas (Arquímedes), alertas de noticias médicas actualizadas periódicamente, libros de texto, referencias farmacológicas y monografías de enfermedades. En la Figura 6 se ofrece una captura de pantalla de la aplicación.

Figura 6. Captura de pantalla Skyscape.

- Calculate by QxMD. La calculadora clínica más completa que rueda en el Android Market. Muy útil para proveedores de atención primaria, generalistas y estudiantes, de QxMD Medical Software Inc.
- AHRQ ePSS. Aplicación diseñada por el U.S. Department og Health & Human Services (HHS), la cual ayuda a los prestadores de atención primaria a identificar las situaciones y servicios médicos preventivos para sus pacientes. Posee una buena fuente de información provista por el U.S. Preventive Services Task Force (USPSTF). Brinda una consulta rápida, ágil y con una información de buena calidad en cuanto a actividades diarias médicas. Actualmente cuenta con entre 50.000 y 100.000 descargas
- MedPage Today. Aplicación de noticias médicas neoyorkina que entrega noticias médicas en el dispositivo móvil relevantes para el interés particular del usuario. Actualmente cuenta con un número de descargas entre 100,000 y 500,000.
- Visible Body. Uno de los mejores atlas que se pueda encontrar en forma de aplicación. Obtuvo el premio a la excelencia de la Asociación de Ilustradores Médicos de 2011. Ofrece muy buena calidad de imágenes del cuerpo humano en 3D.

Por supuesto, hay otras aplicaciones médicas que son muy reconocidas a escala nacional, en el sentido de la interacción médico-paciente. Una de ellas es la App Doctoralia, que permite tanto a médicos como a pacientes gestionar las citas médicas, encontrar especialistas en función del seguro contratado, e incluso permite a los pacientes evaluar a los profesionales sanitarios que le atendieron. Premiada en el App Circus 2012.

Hoy en día los profesionales de la salud tienen la posibilidad de realizar diagnósticos diferenciales a partir de una condición médica en Diagnosaurius, obtener referencias a patologías comunes y raras y su descripción en Eponyms y cualquier persona puede controlar su ritmo cardíaco con Cardiógrafo - Cardiograph, de Macropinch. El sistema Android está permitiendo el acceso a nuevas aplicaciones de gran utilidad en especialidades tales como la neurología, anestesiología, farmacología, medicina interna, oftalmología, dermatología, etc. y está cambiando los paradigmas médicos y de investigación.

1.4 Objetivos

El objetivo del trabajo es desarrollar una aplicación móvil para teléfonos Android que permita a los nefrólogos poder consultar la información sanitaria de sus pacientes desde cualquier momento y lugar.

Para la consecución de los objetivos debe tenerse en cuenta las siguientes consideraciones:

- El médico debería poder conocer el estado de sus pacientes durante los tiempos entre consultas, así como ejecutar la acción más apropiada en caso de ser necesario. Por ejemplo, el médico debería poder conocer las nuevas incidencias de todos sus pacientes sin tener que esperar a la próxima cita.
- Tanto médicos como pacientes deberían poder ahorrarse consultas o citas innecesarias y no programadas como consecuencia de, por ejemplo, un sencillo cambio en el plan de cuidados.
- El médico debería poder conocer el historial de variables fisiológicas de un determinado paciente y actuar en consecuencia, por ejemplo dando nuevas instrucciones a dicho paciente o incluso programando una nueva cita.
- Para que el proyecto pueda dar el mejor resultado posible es necesario concienciar a los pacientes de la importancia de seguir las instrucciones que su médico le proponga a través de la aplicación.

1.5 Planificación del trabajo

A continuación se va a enumerar los diferentes procesos que conforman el plan de trabajo en orden cronológico e indicando el tiempo estimado para cada fase. También se ha representado gráficamente en la Figura 7.

1. Estudio de necesidades. Durante esta fase se pretende explicar de manera concisa los objetivos del proyecto y a qué necesidades responde.
2. Fase de prediseño (1 día). En este proceso se pretende exponer gráficamente los diferentes menús que mostrará la aplicación una vez desarrollada.
3. Estudio del alcance de la aplicación y funcionalidades (0.5 días). Este proceso intenta aclarar cuáles serán las funcionalidades del proyecto final, y cuáles podrían considerarse como funcionalidades extras que se implementarían en líneas futuras de ampliación del mismo.
4. Diseño de la estructura lógica de navegación (0.5 días). Aquí se pretende diseñar cómo se interrelacionarán los diferentes menús que conformarán la aplicación final.
5. Diseño de la arquitectura MVC (1 día). En este proceso se trata de diseñar la arquitectura Modelo-Vista-Controlador de la aplicación final. De manera resumida, el MVC enuncia que cualquier objeto dentro de la aplicación debería pertenecer a una de las tres capas: modelo, vista o controlador. Dentro de la capa de Modelo entraría todo objeto que contribuyera a la lógica de la aplicación Esto incluiría, por ejemplo, las diferentes clases necesarias para pedir información a la base de datos sobre un determinado paciente. En la capa de Vista estaría cualquier objeto visual con el que el médico actuara directamente, por ejemplo un botón o una interfaz gráfica completa. Finalmente, cualquier objeto dentro de la capa Controlador se encargaría de enlazar las dos capas anteriores.
6. Diseño de cada menú de la aplicación (14 días). En esta fase, que ocupa de manera aproximada dos semanas, se desarrollará cada una de las ventanas gráficas que componen la aplicación. Para ello se empleará el lenguaje de programación XML. Todo lo que aquí se diseñe entrará en la capa de Vista (ver el proceso número 5).
7. Desarrollo de la aplicación (20 días). Aquí deberá diseñarse todos los objetos que entren en las capas de Modelo y Controlador (ver el proceso número 5). Esto incluye todas las líneas de código necesarias para que la aplicación funcione correctamente.
8. Testeo y búsqueda de errores (1 día). Durante esta fase se pondrá a prueba la aplicación y se buscará y documentará cualquier error que pueda surgir durante la operación de la misma.
9. Corrección de errores (2 días). Se pretende en este proceso corregir todos los errores anotados en la fase

anterior.

10. Inclusión de nuevas funcionalidades. En caso de disponer de más tiempo extra, podrá incluirse alguna funcionalidad extra a la aplicación.
11. Desarrollo del manual de usuario (2 días). Finalmente se pretende desarrollar un manual de usuario donde se explique cómo utilizar la aplicación.

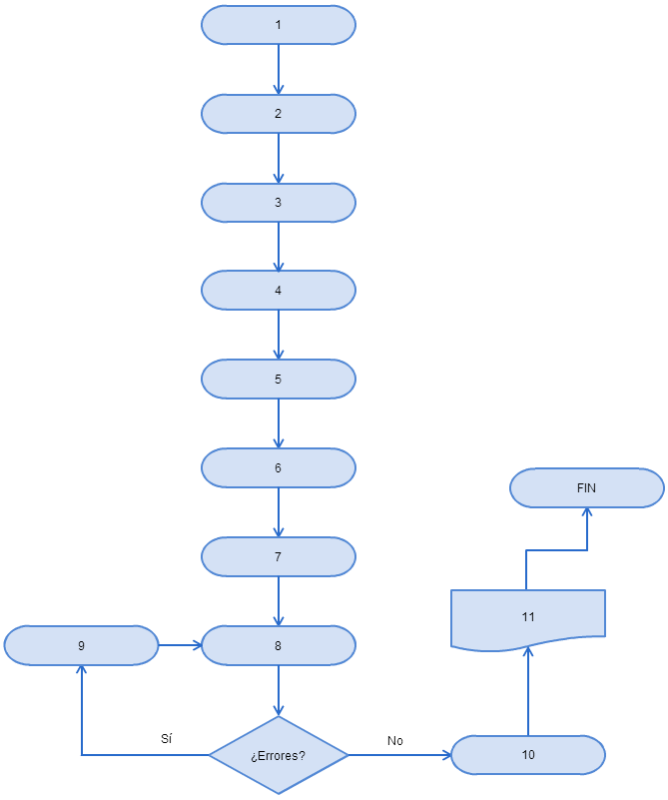


Figura 7. Plan de trabajo.

2 MATERIALES Y MÉTODOS

El objeto de este capítulo es triple. En primer lugar, se expone el Sistema e-Nefro: arquitectura y limitaciones que impone durante el desarrollo de este trabajo. En segundo lugar, en Materiales empleados, se ofrece una explicación de cada una de las herramientas que ha sido necesario utilizar durante el desarrollo de esta aplicación y para qué se han utilizado. El último apartado de este Capítulo 2, al que se ha denominado Metodología a seguir para el desarrollo del sistema, informa sobre los pasos que se van a seguir para el desarrollo de la aplicación, y servirá como nexo de unión al Capítulo 3.

2.1 Sistema e-Nefro

Como ya se introdujo en el Capítulo 1, el proyecto e-Nefro es un esfuerzo multicéntrico entre la Universidad de Sevilla y diversos hospitales nacionales. En dicho proyecto se investiga sobre la mejor forma de optimizar los cuidados en el hogar de los pacientes en prediálisis y diálisis peritoneal, con la intención de potenciar los autocuidados, la seguridad del paciente en su domicilio, y el seguimiento supervisado y continuo de dichos pacientes.

El sistema e-Nefro es la interfaz tecnológica que se ofrece como solución a las ambiciones del proyecto e-Nefro. Dicha interfaz ofrece los medios necesarios para el intercambio de información entre cuidadores y pacientes. Consta principalmente de una base de datos y un servicio web, el cual actúa como intermediario entre la base de datos y aquellas partes interesadas en acceder y/o modificar dicha información. De hecho, la aplicación desarrollada en este proyecto es una de estas partes interesadas, la cual interactuará con la interfaz del sistema e-Nefro en un intento de responder a una de las ideas clave del proyecto e-Nefro: ofrecer a los cuidadores la posibilidad de consultar toda la información de sus pacientes desde un smartphone con acceso a internet.

Antes de introducir las herramientas a utilizar durante el desarrollo del proyecto, conviene entender primero cómo afecta la arquitectura e-Nefro a la aplicación.

Para empezar, este proyecto se puede categorizar como una aplicación de sólo lectura, y esta es la primera y más importante de las limitaciones presentadas. El sistema e-Nefro impone la condición de que la aplicación sólo podrá leer los datos ofrecidos por el servicio web, lo cual afectará al desarrollo de los menús de la aplicación de manera notable.

Por supuesto, para leer la información de la base de datos, primero hay que mostrarse como un usuario válido ante el servicio web. Esto se consigue a través de una de las técnicas de seguridad más antiguas que existen: el par usuario-clave. A cada usuario médico se le asocia un nombre de usuario y una clave únicos, que usarán para acceder a los datos de sus pacientes. Este hecho de tener que autenticarse para usar la aplicación, hace que sea necesario el diseño del menú launcher de la aplicación: el menú de acceso.

La información devuelta por el servidor estará en formato JSON. Se utiliza este formato por su rapidez en el manejo de grandes cantidades críticas de información. La principal alternativa a JSON es XML, aunque en muchos casos es posible utilizar ambos formatos a la vez en el mismo programa para extraer las mejores características de cada uno de ellos. En la Figura 8 se muestra un ejemplo de código JSON. En dicha figura se puede apreciar la arquitectura básica de cualquier código JSON, donde se define tres objetos llamados “msg”, “ok” y “patients” con sus respectivos valores. El objeto “patients” es de especial interés pues su contenido no es una simple cadena de texto, sino un array de datos que a su vez contiene más objetos.

```
{ ⊟
   "msg":"",
   "ok":"true",
   "patients":[ ⊟
      { ⊟
         "name":"Paciente Gib Gib",
         "id":"5"
      },
      { ⊟
         "name":"Paciente Gib Dos",
         "id":"6"
      },
      { ⊟
         "name":"ss ss s",
         "id":"7"
      }
   ]
}
```


Figura 8. Fragmento de código JSON.

Por último, sólo se podrá leer los contenidos ofrecidos por el servicio web y no más allá. Esta característica supone otro requisito adicional: los métodos del servidor. Desarrollados por el Grupo de Ingeniería Biomédica y junto con la colaboración del autor del proyecto, estos métodos deben ser implementados por el código cliente de la manera más conveniente posible.

2.2 Materiales empleados en el desarrollo técnico del proyecto

Para el desarrollo de la aplicación ha sido necesario el uso de diversas herramientas que a continuación se listan:

- Java Development Kit 8 (JDK 8) [5]. Software desarrollado por la compañía Oracle™, el cual provee herramientas de desarrollo para la creación de programas en Java. JDK incluye Java Runtime Environment, el compilador de Java y las API de Java.

- Android Studio [6]. Herramienta principal para el desarrollo de programas Android. No es más que una interfaz de desarrollo de las muchas que existen actualmente en el mercado, por lo que podría haberse escogido otro entorno como Eclipse o NetBeans. No obstante, es un hecho que Android Studio -IDE desarrollado por la misma compañía propietaria del sistema operativo Android-, aporta numerosas ventajas adicionales a la hora de escribir aplicaciones para su plataforma, como puede ser su avanzado sistema de refactoring o la posibilidad de usar dos monitores duales (por ejemplo, uno para el diseño gráfico y otro para la escritura de código).

- Android SDK [6]. Paquete de desarrollo que incluye las librerías API y las herramientas de desarrollo necesarias para construir aplicaciones compatibles con el sistema Android. Dentro de estas herramientas es posible encontrar un depurador de código (DDMS), una biblioteca, un simulador de teléfonos basado en QEMU, documentación, ejemplos de código y tutoriales.

- DDMS [7] (o Dalvik Debug Monitoring Server). Aunque esta herramienta forma parte del Android SDK, merece un apartado adicional debido a su importancia a la hora de desarrollar correctamente la aplicación. Como más adelante se comprobará, una parte fundamental en el desarrollo de cualquier aplicación es la depuración de código. Esto es posible gracias a la herramienta presentada en este punto,

que unida a un emulador Android forman el tándem perfecto para la detección y corrección de errores en el código.

Estas son las cuatro herramientas mínimas necesarias para construir una aplicación para Android. Información adicional sobre cómo enlazar adecuadamente estas tres herramientas puede consultarse en el e-book desarrollado por el autor de este mismo trabajo en la sección bibliográfica [8].

Además se ha hecho uso de otras herramientas, opcionales pero necesarias para tener éxito durante la fase de desarrollo/publicación:

- Dispositivos Android virtuales usando la tecnología de Genymotion [9]. Un dispositivo virtual permite emular en el propio ordenador diferentes tipos de dispositivos basados en Android. De esta forma, es posible probar las aplicaciones que se estén desarrollando en una gran variedad de dispositivos, sin la necesidad de disponer físicamente de ellos. La tecnología de Genymotion destaca por su alta velocidad a la hora de iniciar dispositivos virtuales en la estación de desarrollo, disminuyendo dramáticamente los tiempos de espera. La versión gratuita de su emulador es más que suficiente para hacer las pruebas necesarias y comprobar que las interfaces gráficas son compatibles con una variedad de dispositivos. En particular, se ha decidido hacer las pruebas usando estos cuatro dispositivos virtuales:
    - Google Nexus S – 2.3.7 – 480x800
    - Google Nexus 7 – 4.1.1 – 800x1280
    - Samsung Galaxy S5 – 4.4.4 – 1080x1920
    - Google Nexus 10 – 5.0.0 – 2560x1600

- Dispositivo Android externo. Además se ha considerado necesario hacer pruebas sobre un dispositivo externo real, eliminando así posibles fallos de los emuladores. El dispositivo externo utilizado es:
    - Huawei Ascend P7 – 4.4.2 – 720x1280

- A lo largo de la aplicación se han utilizado imágenes e iconos en la mayoría de los menús para facilitar la comprensión por parte del usuario. Esto, junto con la paleta de colores usada, facilita en gran medida la navegación a través de la app. Las imágenes e iconos utilizados se han descargado del repositorio público iconarchive [10]. Los iconos utilizados son gratuitos bajo la condición de uso no comercial.

- Photoshop CS6 [11]. Se aprovechó la versión trial (o versión de prueba) de este producto para convertir los diseños en papel a diseños digitales de los diferentes menús de la aplicación. Existen otras alternativas que se podrían considerar más sofisticadas, como Balsamiq Mockups o Pixate, esta última recién adquirida por Google.

- Herramienta draw.io [12] en la nube para realizar los diagramas de flujo necesarios para la correcta comprensión del alcance de la aplicación, así como su integración con la parte del servidor.

- Aplicación Evernote [13]. Se utiliza conjuntamente con una tableta Samsung Galaxy Note 4 de 10.1 pulgadas, ofreciendo una muy buena herramienta para realizar bocetos de algunas partes de la interfaz gráfica de la aplicación.

- Servicio de almacenamiento en la nube gratuito Dropbox [14]. Se utiliza como medio de intercambio de archivos y como soporte de backup ante posibles pérdidas de información.

- Desde el punto de vista de la programación, se ha decidido hacer uso de la librería open-source Retrofit [15], desarrollada por la compañía Square [16]. Retrofit facilita y, sobre todo, optimiza las labores de conexión y petición al servidor. Para que dicha librería funcione de la manera más óptima posible, es necesario además implementar otras dos librerías desarrolladas por la misma compañía: Okhttp [17] y Okio [18]. Aunque el empleo de estas dos últimas no es estrictamente necesario, cabe destacar que Retrofit delega la mayor parte de sus tareas de optimización en ellas. Como alternativa a Retrofit podría destacarse la librería open-source Volley [19] desarrollada por Google. Sin embargo, la escasez de documentación y ejemplos ofrecidos por el gigante americano ha sido más que suficiente para desechar su uso en este proyecto. Además, un cuidadoso testeo llevado a cabo por un particular [20] muestra la notable rapidez de Retrofit con respecto a su análoga Volley o incluso a una implementación manual usando la clase AsyncTask (ver Tabla 1)

Tabla 1. Comparativa de la velocidad para diferentes métodos

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

2.3 Metodología a seguir para el desarrollo del sistema

Cualquier proyecto necesita de un plan de ejecución. Este proyecto no es una excepción y a continuación se va a intentar explicar de manera sucinta cuál es la metodología a seguir para desarrollar la aplicación por completo.

Al inicio del proceso, lo único con lo que se cuenta es con un conjunto de requerimientos y necesidades que la aplicación debe satisfacer.

Así que en primer lugar parece lógico que deban estudiarse dichos requerimientos y necesidades para tenerlas en cuenta a lo largo del desarrollo de la aplicación.

Una vez estudiado el primer paso, se debe hacer una recopilación inicial de todas las herramientas que nos harán falta durante el proceso. Dichas herramientas han sido descritas anteriormente en este mismo capítulo.

Una vez se tienen las herramientas con las que satisfacer dichos requerimientos, debe hacerse una versión inicial a mano alzada de cuál será el diseño final de nuestra aplicación. Hay que fragmentar la aplicación en distintos módulos, cuyos diseños finales (los resultantes del proceso iterativo de prediseño) deben representarse profesionalmente utilizando una de las muchas herramientas disponibles en Internet. En este proyecto se utilizará Photoshop para el diseño final de cada menú, y a partir de ellos se dará lugar al proceso de creación de archivos gráficos XML.

Representados todos los menús, ahora hay que pensar cuál es la manera más sencilla y amigable posible para navegar a través de ellos. A esta fase se la denominó Diseño de la estructura lógica de navegación en el capítulo 1.

Una vez se tiene todos los materiales sobre los que sustentarse en el desarrollo de la aplicación, es necesario pensar qué liberarías va a utilizarse a la hora de realizar las peticiones al servidor web. La elección de una librería u otra afectará de manera radical al desarrollo de la aplicación. La librería que se utilizará para este fin es Retrofit, dada su rapidez, documentación, y gran comunidad de usuarios.

A continuación, deberá hacerse un diseño de la arquitectura MVC para cada uno de los menús:

- Por una parte, la capa Modelo. En todos los casos, la capa modelo se encuentra dentro de la misma clase java que forma el Activity de cada menú. En Android, un Activity no es más que la pantalla de una aplicación. Hay algunas variaciones a esta definición tan rudimentaria, pero ninguna de ellas aplica en este proyecto. Por lo tanto, si la aplicación consta de doce menús o pantallas, se dice que está compuesta por doce actividades o activities. Las clases personalizadas que se desarrollen a partir de la librería Retrofit también formarán parte de la capa Modelo. Por ejemplo, para la descarga de pacientes podría crearse una clase llamada DownloadPatients, o una clase llamada DownloadIncidences para la descarga de incidencias.

- Por otra parte, la capa Vista. No es más que cada uno de los archivos XML diseñados y que conforma la parte visual de cada menú. Todo lo que se pueda ver e interactuar forma parte de la capa Vista. Un ejemplo de código XML se muestra en el siguiente fragmento:

```
<FrameLayout     xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragmentContainer_MMensActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

- Por último, la capa *Controlador*. Todos los archivos que contribuyen a la lógica del programa se enlazan con la capa visual a través de lo que se conoce como un controlador. No es más que un conjunto de clases y métodos proporcionados por la librería *SDK* de *Android* como son los *Activities* o los *Fragments*.

Habiendo definido cómo se compondrá la arquitectura MVC, ya es posible dar el paso de empezar a escribir el código de la aplicación. Se empezará diseñando cada uno de los menús XML sin implementar aún la lógica que conduce de un menú a otro. El diseño de XML debe ser un fiel reflejo de lo bosquejado anteriormente. Aquí lo importante es simplemente el diseño, y no el funcionamiento de la aplicación, pues una vez se tenga el diseño ya se podrá empezar a enlazar menús desde *java.*

Ya con todos los menús diseñados usando código XML, sólo faltaría escribir el código de fondo que mueva la aplicación. Este es el motor de la aplicación, y su calidad de diseño influirá en la experiencia final del usuario. Se deberá, entre otras cosas:

- Conseguir la máxima fluidez posible al ir de un menú a otro.
- Conseguir los mínimos tiempos posibles de descarga, y que dichos tiempos no afecten al *renderizado* de los menús. Esto se consigue haciendo un uso adecuado de los *Threads* o *hilos* de la aplicación, los cuales a su vez se aprovechan de la capacidad multi-núcleo de los *smartphones* actuales.
- Hacer un extensivo chequeo de los posibles errores que puedan ocurrir durante la ejecución de la aplicación y manejarlos adecuadamente. Si durante la ejecución de la aplicación ocurriera un error que no se hubiera contemplado en el desarrollo de la misma, ésta se cerrará abruptamente mostrando al usuario un mensaje bastante indeseado.
- Evitar molestar al médico en todo lo posible y sólo informar cuando se produzca un error.

Por supuesto, la programación de la aplicación no será un proceso lineal sino más bien un proceso iterativo: se escribe un fragmento de código y se comprueba si su funcionamiento es el esperado. Si el trozo de código no funciona según lo previsto, es necesario volver atrás, modificarlo, e intentarlo de nuevo. Llegados a este punto cabe resaltar la importancia del *DDMS* o *Dalvik Debug Monitor Server.* Esta herramienta, la cual se encuentra disponible con el paquete SDK de Android, será muy útil para depurar código y hacer este proceso iterativo mucho más sencillo.

Una vez completada la aplicación, deberá procederse a una fase de testeo y búsqueda de errores. Se someterá a la aplicación a usos totalmente fuera de lo previsto y se comprobará cuál es su comportamiento. Se comprobará que la app informa adecuadamente bajo cualquier tipo de error, los cuales se deberán simular a propósito (por ejemplo: caso de estar sin conexión, caso de servidor caído, caso de conexión lenta, caso de ausencia de

información en la respuesta ofrecida por el servidor web, etc.)

Lo más común es que tras la fase anterior se encuentren errores adicionales por el camino. Estos errores deberán ser corregidos y comprobar que no vuelvan a aparecer.

En caso de disponerse de tiempo adicional tras la última iteración de búsqueda y eliminación de errores, podría contemplarse la posibilidad de añadir funciones más avanzadas a la aplicación. Algunos ejemplos de *funciones extra* se sugerirán más adelante en el documento. Un ejemplo de función extra sería la opción de notificar al usuario médico de una nueva incidencia a través de la barra de notificaciones.

Por último, si el tiempo y la profundidad de menús de la aplicación lo permiten, se podría desarrollar un manual de usuario para mostrar cómo se utiliza la aplicación. Sin embargo, si la aplicación resulta ser demasiado sencilla de entender y/o falta tiempo para completar este punto, podría dejarse el desarrollo del manual como una mejora futura del proyecto.

3 RESULTADOS

3.1 Introducción a la aplicación

La aplicación fruto de este proyecto pretende facilitar el seguimiento por parte del nefrólogo de cómo sus pacientes ejecutan el plan de cuidados prescrito en el hogar. Ofrece al médico la posibilidad de consultar la información almacenada sobre sus pacientes en cualquier momento y lugar, donde sólo se necesite un *smartphone* y conexión a internet. Se considera bastante conveniente el hecho de que el médico esté informado de todas las incidencias y registros de sus pacientes, y se le permita actuar con rapidez y de la manera que mejor considere en cada momento.

Desde la aplicación, el médico podrá navegar entre diferentes menús de una manera fluida para encontrar en cualquier momento la información que más le interese: citas, mensajes de chat, incidencias, registro de variables, plan de cuidados,… De hecho, a continuación se presenta un esquema donde se muestra lo que el médico puede solicitar desde la aplicación al servidor web:

- *Lista de pacientes*. Tras solicitarla, el médico puede pulsar sobre un paciente en particular y obtener:
    - *Registro de variables.* Los registros de variables se categorizaron en Variables, Medicamentos, Imágenes, por lo que el médico podrá seleccionar cualquiera de esta información:
        - *Variables.*
        - *Medicamentos.*
        - *Imágenes.*
    - *Incidencias.* Desde este menú se consultan todas las incidencias de un paciente en concreto.
    - *Plan de cuidados*. Los planes de cuidado se dividieron en cuatro categorías: Variables, Medicamentos, Imágenes y Citas, por lo que el médico puede consultar cualquiera de ellas:
        - *Variables.*
        - *Medicamentos.*
        - *Imágenes.*
        - *Citas.*
- *Lista de citas.* En este menú se muestran todas las citas de un médico con todos sus pacientes.
- *Lista de incidencias.* Aquí se muestran las últimas incidencias de todos los pacientes.
- *Lista de mensajes.* En este menú es posible consultar todos los mensajes intercambiados con los pacientes. Recordar que, al ser la aplicación de sólo lectura, no es posible enviar mensajes desde la misma: sólo podrá consultarse la información intercambiada.
- *Menú de configuración.* Desde aquí se podría configurar la aplicación para que se ajuste mejor a las necesidades de cada nefrólogo.
- *Botón de salir.* Botón que facilita al usuario abandonar la aplicación.

Para que la navegación entre las distintas pantallas de la aplicación sea lo más efectiva posible, es necesario primero introducir el servidor web, su funcionamiento e interacción con el código cliente.

El sistema eNefro, como ya se ha explicado anteriormente, consta principalmente de un servidor web y una base de datos con la información de todos los médicos adheridos y sus respectivos pacientes (plan de cuidados de cada uno, registros de información, incidencias, etc.). El mecanismo de interacción cliente-servidor es el siguiente:

1. El médico solicita a la aplicación móvil cierta información, por ejemplo su lista de pacientes. A su vez, la aplicación se conecta al servidor web a través de una conexión de red (que suele ser una tarifa de datos o una conexión *WIFI*).
2. El servicio web recibe la petición por parte de la aplicación. Si dicha petición cumple unos ciertos requisitos (por ejemplo, que el usuario y clave sean correctos o que el ID de usuario no haya expirado), el servicio hace una consulta a la base de datos del servidor.

3. Si la consulta se ha realizado de la manera correcta, la base de datos nutre al servidor web con la información deseada. Puede ocurrir que la base de datos no sea capaz de encontrar dicha información, en cuyo caso devuelve un objeto vacío que más tarde deberá ser interpretado para evitar errores en la aplicación.
4. Si todo ha ido correctamente, el servicio web se vuelve a poner en contacto con la aplicación. Le proporciona toda la información que se le peticionó en un formato muy específico conocido como *JSON* (consultar el Capítulo 2 para una descripción de *JSON*). Es labor de la aplicación interpretar, fragmentar y por último almacenar toda la información recibida, y que esté lista para ser presentada al usuario médico.
5. Una vez almacenada y, sobre todo, ordenada la información recibida, es posible presentarla al usuario a través de métodos muy específicos que han sido desarrollados en código *Java.* Para presentar dicha información al usuario, primero es fundamental disponer de los menús correctamente diseñados y listos para ser poblados con todos los datos recién recibidos.

En la Figura 9 se muestra un esquema gráfico de todos estos pasos para una mejor comprensión de los mismos. Cabe resaltar que los tiempos de demora más importantes durante estos cinco pasos son los correspondientes a los pasos 1 y 4. Si la conexión del usuario es lo suficientemente pobre, dichos tiempos de espera serán más altos de lo normal.

En la primera versión de la app se ha optado por este tipo de comunicación en cinco pasos. Sin embargo, en una posible ampliación del proyecto podría añadirse una mejora a dicho mecanismo de comunicación, añadiendo por ejemplo un temporizador interno o una opción de refresco manual. Aunque en la Figura 10 se muestra gráficamente el fundamento de esta mejora, ésta se tratará más adelante en el Capítulo 4.

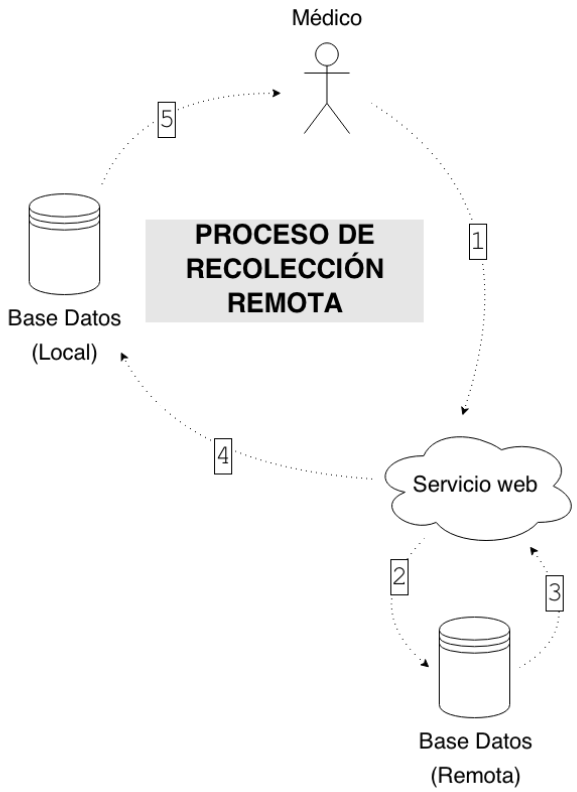


Figura 9. Ilustración del proceso de recolección remota.

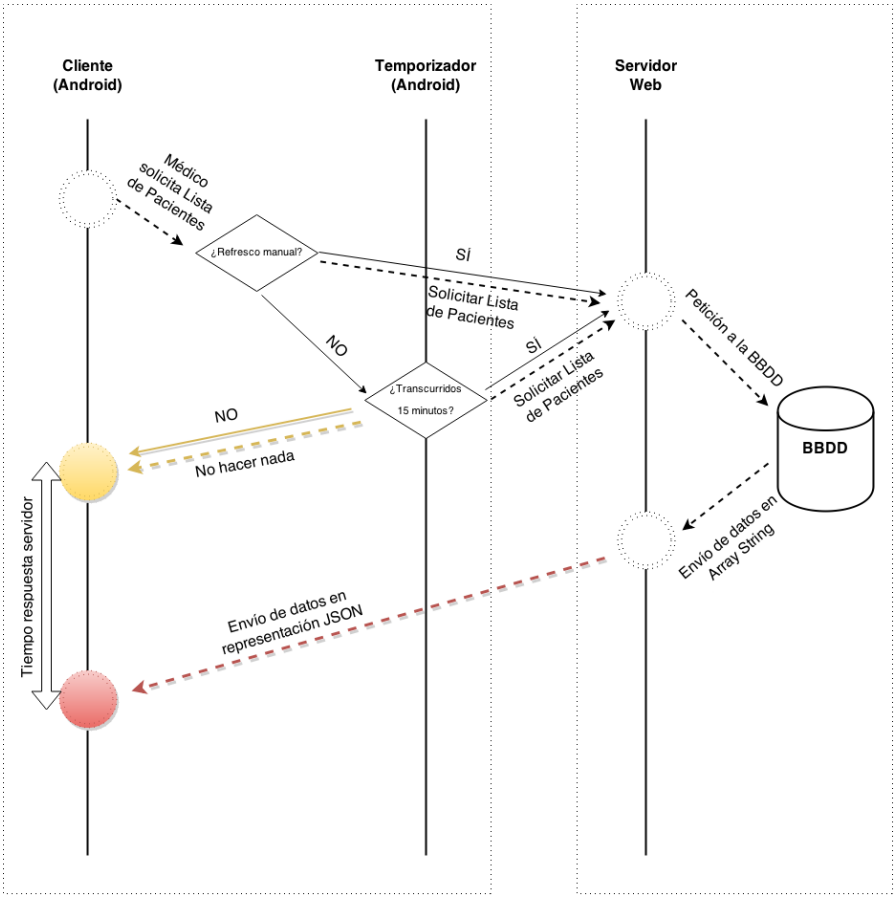


Figura 10. Recolección remota mejorada (Capítulo 4).

3.2 Descripción del servicio web e interacción con la app

Como ya se ha explicado, el servicio web proveerá de los medios necesarios para que el cliente (es decir, la aplicación) pueda obtener la información que necesite en ese momento. Al inicio del proyecto se contempló dos alternativas: el uso de *Servlets/JSP* o el uso de código *PHP*. De hecho, la decisión se tomó por el Grupo de Ingeniería Biomédica antes incluso del nacimiento de este proyecto, y el resultado de la decisión fue usar código *PHP* para la interacción con el cliente.

Esta decisión marcaría desde un primer momento la manera de programar la aplicación. Diversos archivos *PHP*, conocidos también como los métodos del servidor, son los que más adelante proporcionarían los medios de intercambio de información cliente-servidor. Cada uno de estos archivos está especializado en hacer una consulta personalizada a la base de datos. Por ejemplo, el archivo llamado *incidences.php*, como podrá imaginarse, se encarga de listar las últimas incidencias de todos los pacientes asignados a un usuario médico. Cada archivo tiene dos clases de argumentos: los de entrada y los de salida. Los argumentos o parámetros de entrada son absolutamente necesarios para que el método encargado pueda procesar la solicitud. Uno de los parámetros de entrada más utilizados es el que se conoce como *id_sesion*, que no es más que el identificador asignado a un usuario médico desde el mismo momento en que se autentifica como un usuario legítimo. Una vez el método procesa el id_sesion y comprueba que sea válido y no haya expirado, se procede a la solicitud de información y su devolución en los distintos parámetros de salida.

Debe entenderse que el mecanismo de envío de datos entre cliente y servidor sigue el esquema POST. El método POST pertenece al protocolo HTTP y consta, al igual que su análoga GET, de una petición (por ejemplo, petición de obtener las incidencias de un paciente) y una respuesta a dicha petición (las incidencias solicitadas). La elección de este método se llevó a cabo por los encargados del servidor, elección en gran parte correcta pues existe un acuerdo en que POST es más seguro de utilizar que GET.

La devolución de datos desde el servicio web hasta el cliente podría haberse efectuado de diversas maneras. Por

ejemplo, podría haberse enviado toda la información en formato String y permitir que sea la aplicación quien se encargue de convertirla a un formato más manejable como *JSON*, haciendo uso, por ejemplo, de la biblioteca *Gson* creada por *Google*. En este proyecto se ha decidido, no obstante, realizar las tareas de interpretación en el código del servidor por dos motivos:

- Abstracción. Si la tarea de conversión String a *JSON* se realiza en el servidor, dicha tarea se lleva a cabo una sola vez, sirviendo para cualquier aplicación que en el futuro se decidiera desarrollar. Por otro lado, si se hubiera elegido realizar la tarea de conversión en el código cliente, dicha tarea debería repetirse en caso de programar otra aplicación diferente. Puede utilizarse el término acuñado por *Sun Microsystems*: *Write once, run anywhere* (*WORA*). Aunque dicho slogan surgió inicialmente para ilustrar los beneficios del lenguaje Java, su concepto de fondo es aplicable también en este caso.
- Mantenibilidad. Dado que la aplicación está sujeta a futuras mejoras, es más fácil hacerlo cuanto menor sea el código utilizado. Las labores de cambio de formato *String* a *JSON* pueden llegar a ser arduas tras un cambio lo suficientemente grande en las especificaciones de los métodos del servidor.

Podría también haberse utilizado el formato *XML* en lugar de *JSON*, aunque el uso de uno u otro no tiene ningún impacto sobre el rendimiento final de la aplicación.

Cabe resaltar que los métodos, aun habiendo sido inicialmente propuestos por el Grupo de Ingeniería Biomédica, fueron más tarde estudiados por el autor del proyecto. Este proceso no fue lineal sino iterativo: se terminó alcanzando la solución más conveniente para ambos servidor y cliente tras haber superado varias fases iterativas intermedias. De esta manera se alcanza una solución de compromiso que permita trabajar a ambos cómodamente y siempre de la manera más eficiente posible.

La última versión de la tabla de métodos se encuentra en el *Capítulo 5 (Anexo A, Tabla 3),* aunque a continuación se enumera brevemente los distintos métodos utilizados y cuál es su función:

- */login.php* – Este método permite al nefrólogo acceder al resto de información a través de un usuario y clave.
- */patients.php* – Este método recupera la lista de pacientes asociada al usuario médico que lo invoque.
- */patientView.php* – Este método recupera todas las incidencias, registros y planes de cuidados de un paciente en particular.
- */appointments.php* – Este método recupera todas las citas asociadas al usuario profesional.
- */inbox.php* – Este método recupera todos los mensajes intercambiados por el usuario profesional con sus pacientes.
- */incidences.php* – Este método recupera la lista de las todas últimas incidencias asociadas a todos los pacientes en general, en caso de que quiera consultarse todas las incidencias a la vez.

3.3 Alcance de la aplicación y diseño de la interfaz

El alcance de la aplicación se verá influenciado de manera directa por las funcionalidades que se decida implementar. Antes de empezar a desarrollar la aplicación es necesario describir: quién es el usuario final; cuáles son las funcionalidades básicas; y cuáles son las posibles funcionalidades extra.

- Usuario final: el médico. La aplicación debe optimizarse para facilitar su uso a los médicos. Como se mencionó en el apartado de alcance y limitaciones del proyecto, el flujo de información tiene lugar entre el médico y la base de datos, por lo que los pacientes no intervienen en ningún momento en el proceso de desarrollo de la aplicación.
- Funcionalidades básicas:
    - Consulta de información sobre pacientes. Esta información incluye:
        - Incidencias.
        - Variables fisiológicas (temperatura, presión arterial,…).
        - Plan de cuidados.

        - Citas programadas.
        - Mensajes internos.
    - Configuración del comportamiento de la aplicación.
- Funcionalidades extra (ampliable):
    - Posibilidad de escribir notas personales (sólo visibles por el médico) sobre un paciente.
    - Posibilidad de representar gráficamente la evolución de un parámetro fisiológico de un determinado paciente.
    - Alerta sonora a modo de notificación en caso de nuevas incidencias.

3.3.1 Diseño de la estructura lógica de navegación

Se ha desarrollado cómo será la navegación natural a través de los diferentes menús que forman la aplicación, diseñando unos esquemas gráficos que facilitarán su comprensión.

Se ha empleado una notación simple que permita referenciar un menú en particular de la manera más rápida posible. La notación se forma precediendo por la letra "M" (de la palabra Menú) al sustantivo que mejor describa la finalidad del menú en cuestión (abreviando el sustantivo cuando proceda). De esta manera, al menú de acceso inicial se le ha denominado "MAcceso".

Se ha identificado un módulo reusable a lo largo de toda la aplicación: el menú de paciente o "MPac", como se muestra en la Figura 11.

Este módulo es importante debido a que se puede acceder a él desde casi cualquier otro menú de la aplicación (excepto desde el menú de configuración). A partir de ahora, a este módulo se le denominará simplemente "MPac" y facilitará la representación del esquema global de navegación de la aplicación.

El árbol de navegación de la aplicación será:

1. MAcceso
    a. MCentral
        i. MLista
            1. MFiltro
            2. MPac
        ii. MUIncid
            1. MPac
        iii. MConfig
        iv. MMens
            1. MPac
        v. MCitas
            1. MPac

También se ha representado visualmente este árbol de navegación en la Figura 12.

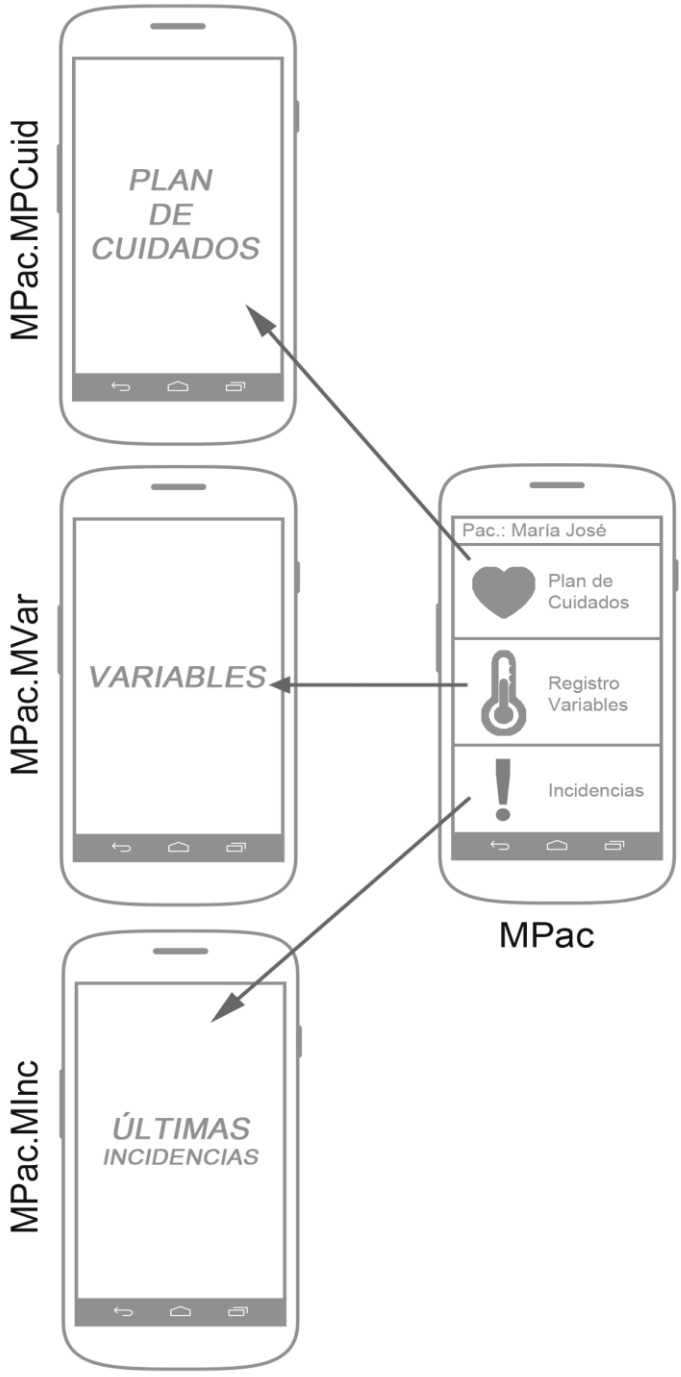


Figura 11. Módulo MPac.

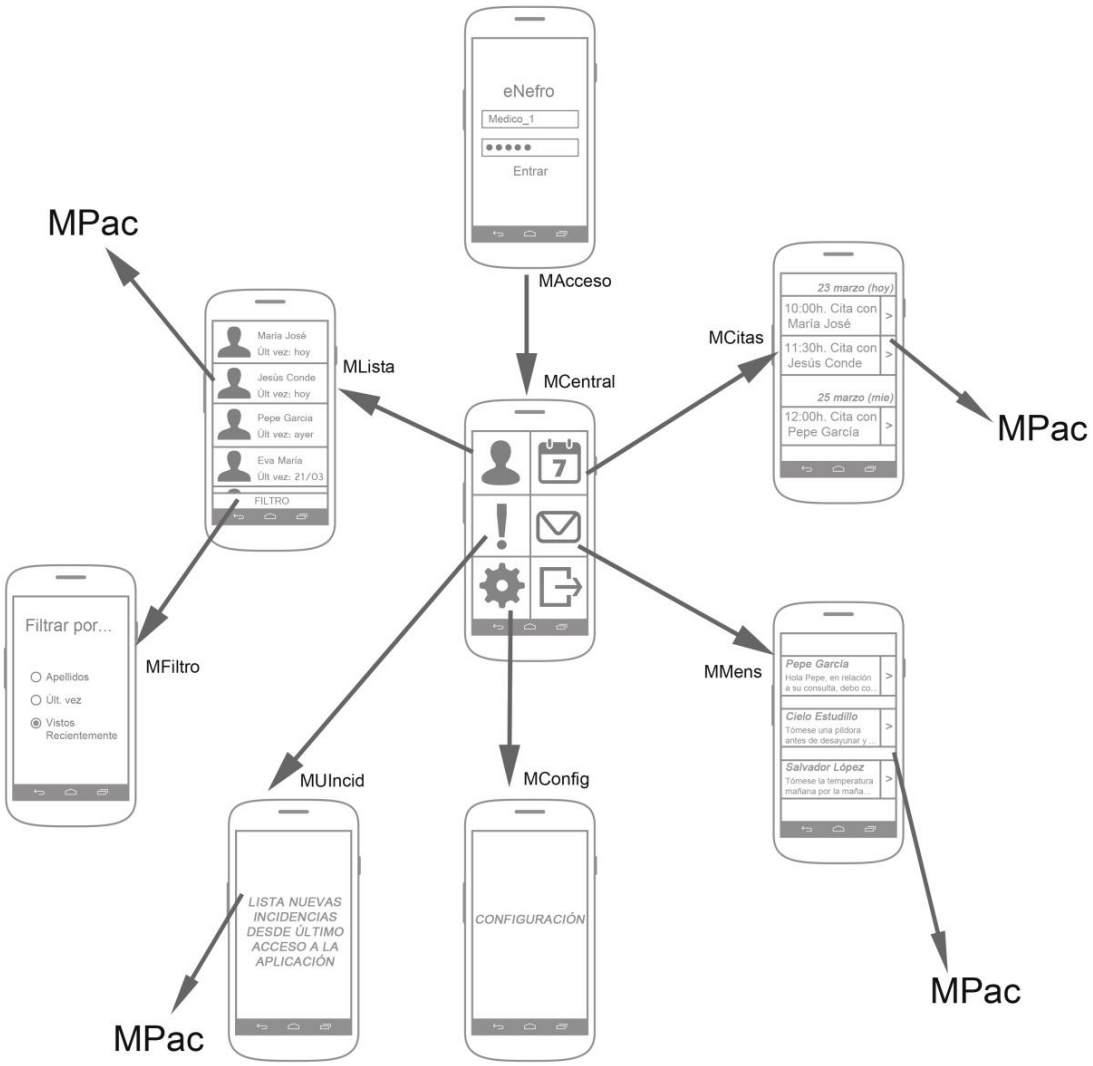


Figura 12. Esquema global de la aplicación.

3.3.2 Diseño de la interfaz

A continuación se ha decidido ilustrar de manera sucinta el esqueleto final de la aplicación. Se empieza por el menú de verificación de credenciales, donde el médico debe autenticarse como usuario válido a través de un nombre de usuario y contraseña, tal y como se muestra a modo de ejemplo en la Figura 13 (izquierda). Los campos de usuario y contraseña se comparan a través de una conexión segura con los almacenados en una base de datos alojada en un servidor remoto.

Una vez el médico se ha identificado como válido, podrá acceder al menú principal que se muestra en la Figura 13 (derecha). El diseño de la aplicación ha de ser lo más intuitivo posible, por lo que se ha optado por utilizar imágenes en lugar de texto para cada una de las secciones, que enumeradas de arriba abajo y de izquierda a derecha son: pacientes, citas, incidencias, mensajes, configuración y salir de la aplicación.

Dentro de la sección de pacientes, la aplicación podría presentar algo parecido a lo mostrado en la Figura 14 (izquierda), una lista de pacientes con sus respectivos datos: nombre y apellidos, y última actualización. Sería conveniente que esta lista pudiera ordenarse según sea el criterio de búsqueda del médico, como se muestra en la Figura 14 (centro), funcionalidad que se contemplará como una posible futura mejora de la aplicación.

Finalmente, cuando el médico desee ver el historial de un determinado paciente, no tendrá más que seleccionarlo en la lista y un nuevo menú se abrirá, como indica la Figura 14 (derecha). Este nuevo menú permitirá al médico consultar el Plan de Cuidado, Registro de Variables e Incidencias de un determinado paciente. Dentro del menú de Plan de Cuidados se mostrará el mismo en relación a cada una de las cuatro categorías: Medicamentos, Variables, Imágenes y Citas. Dentro del menú de Registro de Variables se mostrará las variables del paciente en orden cronológico inverso. Finalmente, dentro del menú de Incidencias, se mostrará las últimas incidencias del paciente.

Figura 13. Menú de acceso (izquierda) y menú central (derecha).

Figura 14. Lista de pacientes (izquierda), filtro de búsqueda (centro) y menú de paciente (derecha).

En la sección de citas se mostrará algo parecido a lo que se representa en la Figura 15, es decir, una lista donde se indique las citas ordenadas por orden cronológico. En esta lista se recogerá la fecha y hora de la cita, así como el paciente pertinente y un acceso directo a su historial médico.

La sección de incidencias mostrará las nuevas incidencias extraídas de la base de datos. De esta manera el médico podrá estar informado de cuáles son las últimas incidencias y actuar en consecuencia. Se dispondrán las nuevas incidencias en una lista parecida a la de la Figura 14 (izquierda), por lo que no es necesario mostrar un nuevo esquema. La sección de mensajes mostrará los últimos mensajes intercambiados por el médico y un determinado paciente, como se muestra en la Figura 16.

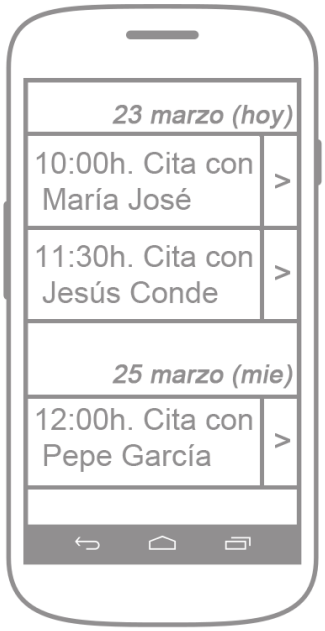


Figura 15. Lista de pacientes.

Finalmente, la sección de configuración permitirá que el médico pueda configurar a su gusto ciertas características de la aplicación, como por ejemplo a partir de qué fecha mostrar el histórico de variables de un determinado paciente.

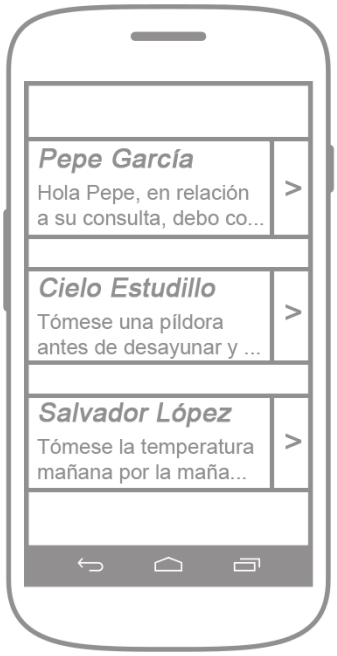


Figura 16. Lista de mensajes.

3.4 Implementación y desarrollo de la aplicación y sus menús

El objetivo de este apartado es presentar cada uno de los menús de la aplicación, complicaciones que surgieron en cada una de las etapas de desarrollo, cómo se afrontaron dichas complicaciones, posibles soluciones alternativas y explicación de la conveniencia de la solución adoptada en cada fase.

Cabe resaltar la importancia de la colaboración entre las partes del cliente y servidor. Ambas partes están sujetas a sus propias restricciones, y es labor de las dos el ponerse de acuerdo en todo lo posible. El Grupo de Ingeniería Biomédica diseñó una muy útil tabla que recoge los métodos de llamada al servidor, así como los parámetros de entrada/salida. Efectivamente, esta tabla (Tabla 3) atravesó múltiples versiones para recoger las necesidades del código del cliente, siempre sujeta a las restricciones inherentes a la propia naturaleza del código servidor.

3.4.1 Menú de acceso

Siguiendo el plan de trabajo y fase de prediseño, se empieza diseñando el menú de autenticación del usuario médico. Debe ser un menú simple e intuitivo, y en una primera iteración se decide ofrecer algo parecido a lo que muestra la Figura 17 (izquierda) Más tarde se rediseñó este menú de acceso, al ser la combinación de colores demasiado llamativa y ofrecer excesivo ruido visual. La segunda versión de este menú se muestra en la Figura 17 (derecha). En la Figura 18 se muestra además una captura de pantalla en mitad del proceso de diseño del menú MAccesoActivity.

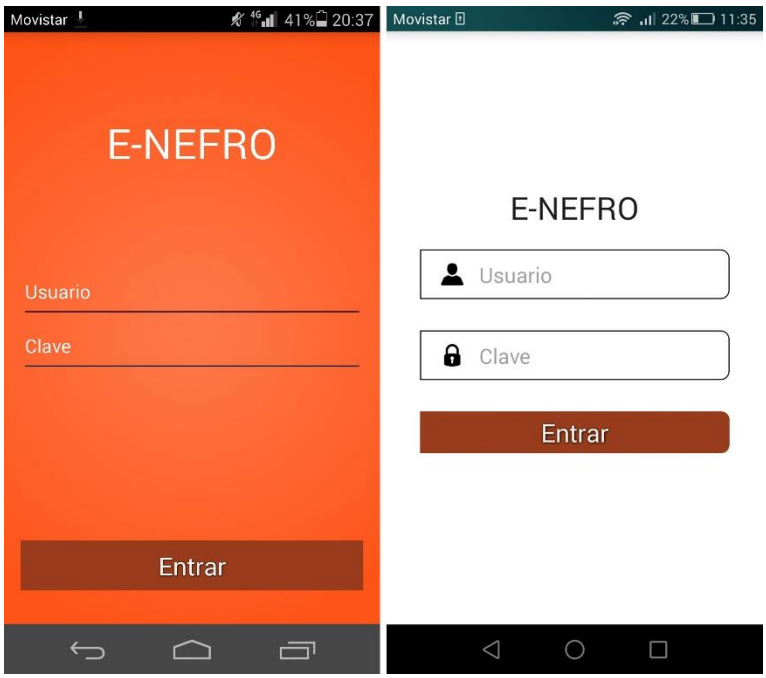


Figura 17. Menú de acceso original (izquierda), menú de acceso rediseñado (derecha).

El funcionamiento de este menú es como sigue.

1. El usuario introduce su identificador y clave en los campos correspondientes, y presiona el botón "Entrar".
2. Antes de realizar una petición al servidor, se hace primero la comprobación de que estos campos son no-nulos. En una versión más avanzada podría incluirse otros sistemas de saneamiento de campos, en un intento de evitar posibles accesos no autorizados o robo de información [21]. Un punto importante de mejora es evitar enviar (y almacenar) la contraseña del usuario sin ser previamente encriptada mediante un algoritmo seguro SHA-2 [22] o similar.
3. Tras las labores de saneamiento y encriptación, se procede a establecer una conexión con el servidor y, al mismo tiempo, solicitar una petición de llamada al método */login*, el primero de la Tabla 3.

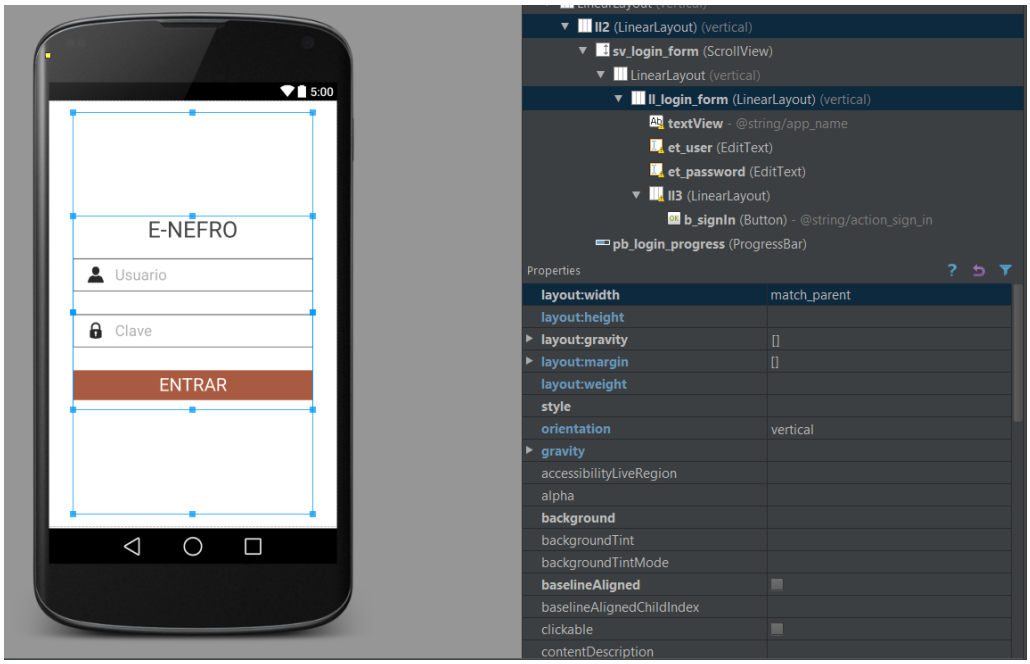


Figura 18. Proceso de diseño del menú de acceso.

Es una buena práctica en Android (y en cualquier otro dispositivo móvil) separar el apartado gráfico del apartado de cálculos. En Android, la creación de gráficos e interacción con el usuario se lleva a cabo en un hilo principal (*UI Thread*) que no debe en ningún momento utilizarse para realizar operaciones de fondo, como en este caso podría ser el realizar peticiones al servidor. Esto originaría una experiencia lenta e incómoda para el usuario. El hecho de que los dispositivos móviles actuales tengan varios núcleos permite crear otro hilo por separado, que se ejecute de manera paralela al hilo principal, ofreciendo una experiencia fluida al usuario al mismo tiempo que se realizan intensas labores de fondo. En la Figura 19 se muestra gráficamente la diferencia entre los dos hilos [23].

El uso de la librería *retrofit* facilita este proceso implementando una clase personalizada de *Callback* el cual devuelve el método *success* en caso de que la petición se haya realizado con éxito (independientemente de si, por ejemplo, el usuario y clave proporcionados por el usuario sean incorrectos o no, lo cual debe manejarse a parte), o el método *failure*. Este último método se devolvería en el caso de que:

1. El servidor no esté funcionando en ese momento.
2. El servidor sí esté funcionando, pero no se haya hecho una petición correcta (por ejemplo, que el servidor no reconozca un parámetro de la llamada a un método determinado).
3. El servidor sí esté funcionando y la petición sea correcta, pero la implementación de las clases *POJO* sea incorrecta (es importante que los nombres de los parámetros devueltos en el JSON por el servidor coincida con el nombre de las variables dentro del *POJO*; si deseara nombrar de manera diferente a dichas variables, habría de hacerse uso de la anotación *@SerializedName*). Una clase *POJO (Plan Old Java Object)* es aquella que no extiende ni implementa otras clases, sino que más bien define sus propias variables y métodos, lo que la hace perfecta para serializar objetos en formato *JSON* (precisamente los objetos *JSON* devueltos por el servidor). Antiguamente el término *POJO* se utilizaba para referirse a las tarjetas perforadas, adoptándose más tarde este término en el campo de la programación *JAVA* para referirse a clases sencillas y que no dependan de otros *frameworks*.
4. Error interno en el funcionamiento de la librería *retrofit*.

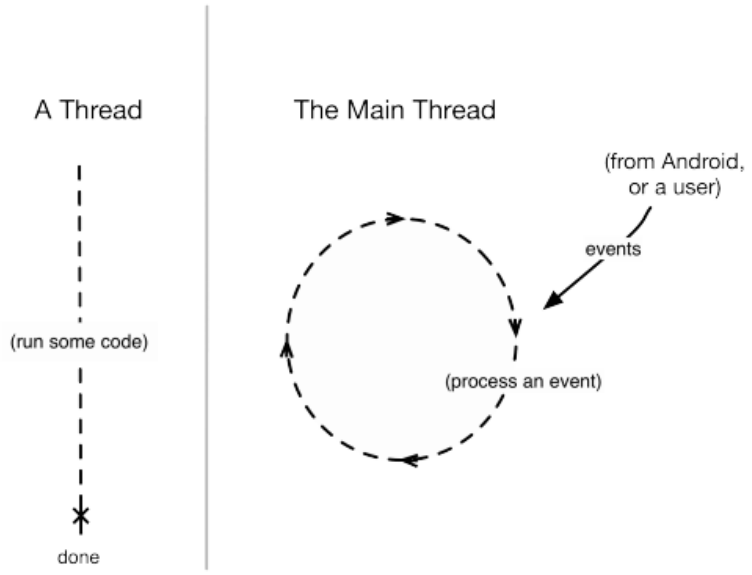


Figura 19. Diferencia entre un hilo secundario secuencial (izquierda) y el hilo principal (derecha).

3.4.2 Menú central (MCentralActivity)

Si la autenticación del usuario médico ha sido correcta, la base de datos le asigna un identificador único. Este *ID* será importante para todas las futuras operaciones que se realizará, por lo que la aplicación se encargará de almacenarlo en una *clase POJO*. La estructura de dicha clase se muestra en la Figura 20. Si el almacenamiento de información ha sido correcto (es decir, si no ha ocurrido ningún problema con la base de datos), se muestra el menú central de la aplicación desde donde se puede acceder a toda la información asignada al usuario médico, Figura 21.

```
import com.google.gson.annotations.SerializedName;

public class POJO_LOGIN {
    @SerializedName("ok")
    public String ok;

    @SerializedName("msg")
    public String msg;

    @SerializedName("data")
    public Dataset_Login data;

    class Dataset_Login {
        @SerializedName("id_sesion")
        String id_sesion;
    }
}
```

Figura 20. Estructura de una clase POJO.

El diseño de dicho menú se realizó con varias premisas establecidas de antemano. La primera es la simplicidad; el menú debe ser lo más simple posible, sin excesiva información redundante. La segunda es la funcionalidad; el menú debe ser lo más funcional posible para un usuario medio. Un menú puede ser simple pero parco en funcionalidad, quizás debido a un ineficiente aprovechamiento de los recursos disponibles. Por último la intuición, que cohesiona ambas simplicidad y funcionalidad mediante una fusión apropiada de colores e iconos

que permita recordar con facilidad cuál es la función de cada botón. El resultado final es el que se muestra en la Figura 21.

Aun habiendo elegido una paleta de colores amigable, la elección de cada color de fondo ha sido totalmente arbitraria. Excepto para el botón de incidencias, donde se le ha asignado un color amarillo representativo de las llamadas de atención o *warning*. Con respecto a los iconos, conviene recordar que su uso es gratuito siempre cuando sea no comercial.

- Para la lista de pacientes, se ha escogido un icono representativo y color de fondo *azul claro (#66CCFF)*.
- Para las citas del médico usuario, se ha escogido un icono en forma de calendario y un color de fondo *verde claro (#00FF80)*.
- Para el botón de incidencias, se ha elegido un icono llamativo en forma de exclamación y un color de fondo *amarillo claro (#F9E559)*.
- Para los mensajes intercambiados con los pacientes, se ha optado por un icono en forma de sobre y un color de fondo *naranja claro (#EF7126)*.
- Para el botón de acceso a la configuración de la app se ha elegido un icono típico en forma de engranaje y un color de fondo *ciruela claro (#C786CF)*.

Por último, para el botón que permite salir de la aplicación se ha elegido un icono en forma de flecha y un color de fondo *gris claro (#D4CBC3)*.

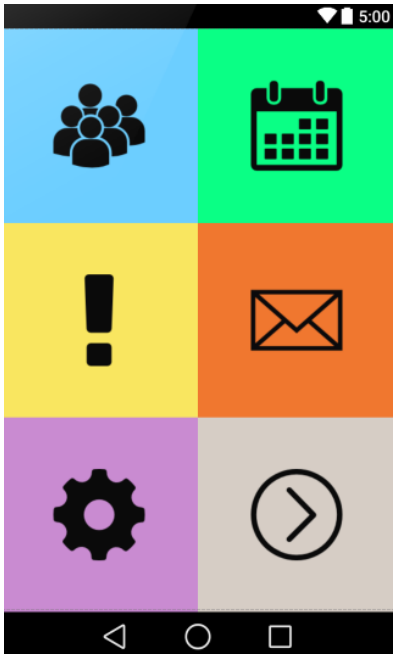


Figura 21. Menú central (MCentral).

Desde el punto de vista de la programación, aquí se tuvo que tener especial cuidado. Para entender por qué, primero hay que saber cuál es el camino recorrido desde que el servidor identifica al médico como usuario válido hasta que éste solicita su lista de pacientes:

1. El médico se autentifica correctamente mediante *usuario* y *clave*. Validación correcta.
2. Automáticamente después de la validación de credenciales, se procede a descargar la lista de pacientes en un segundo *hilo* (o *thread*).
3. Se muestra el menú central, *MCentralActivity*.
4. El médico solicita la lista de pacientes tocando sobre el botón correspondiente del menú central.

Puesto que el punto 2 (descarga de la lista de pacientes) ocurre en segundo plano, ¿qué ocurriría si el médico solicitase su lista de pacientes (punto 4) antes de que dicha información se hubiera descargado? En la Figura 22 se muestra un esquema que recoge la problemática a la que se enfrenta, y más adelante se proponen dos soluciones diferentes a este problema.

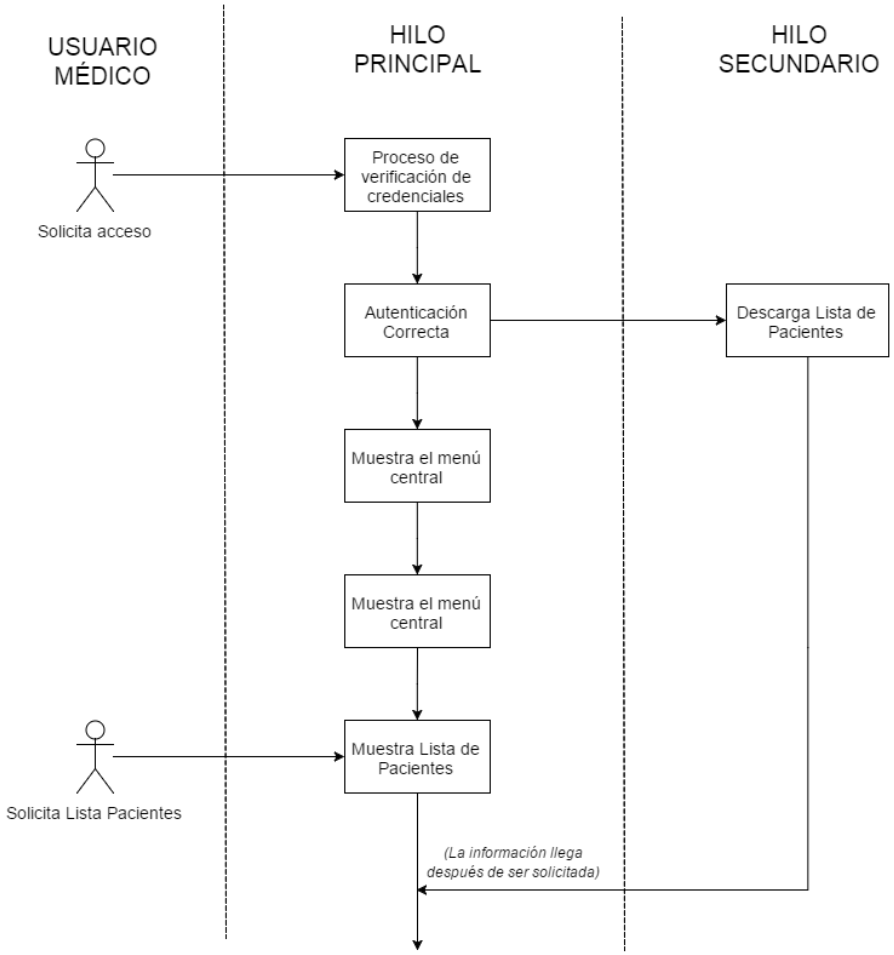


Figura 22. ¿Qué ocurriría si el médico solicita la información antes de que ésta esté disponible?

Aunque se proponen dos soluciones, ambas son casi idénticas y difieren sólo en un matiz psicológico a efectos de tiempo de espera.

1. Mostrar una barra de progreso indeterminada. De esta manera se indicaría al usuario que aún se está realizando un proceso de fondo y que, hasta que no termine, la aplicación estará inaccesible. El tiempo de espera desde que el usuario introduce sus credenciales hasta que su información asignada (pacientes, citas…) es accesible se distribuiría en 2 tiempos de espera diferentes: $t_{espera} = t_1 + t_2$
   a. $t_1 = t_{login}$ ➔ Este tiempo de espera lo experimenta el médico en MAcceso.
   b. $t_2 = t_{descarga}$ ➔ Este tiempo de espera lo experimenta el médico en MCentral. En el menú central sólo se produce la descarga de la lista de pacientes (puesto que el resto de información se descarga a petición del médico). Por lo tanto, t_2 es exactamente el tiempo que se tarda en descargar la lista de pacientes.

2. Realizar la descarga justo antes de mostrar el menú central. De esta manera, se da la sensación al usuario de que el tiempo de descarga de usuarios es instantáneo. A cambio, el tiempo aparente de autenticación es mayor: $t_{espera} = t_1$
   a. $t_1 = t_{login} + t_{descarga}$ ➔ Tiempo de espera experimentado sólo en MAcceso.

Es obvio que los dos tiempos son iguales. Se ha decidido optar, sin embargo, por la segunda alternativa. Esto es así porque el tiempo de espera *aparente* será menor, al tener el usuario que esperar sólo en el menú de acceso y no en ambos menús. Adviértase también que esta solución se alinea a la perfección con el objetivo de la *simplicidad*: si se decidiera usar la primera alternativa, habría de implementarse dos barras de progreso (la primera para indicar que el proceso de autenticación ha terminado, y la segunda para indicar que el proceso de

descarga ha finalizado). Con la segunda alternativa se utiliza sólo una barra de progreso para expresar los dos tiempos a la vez. En la Figura 23 (izquierda) se muestra cuál es la apariencia de MAcceso cuando está en pleno proceso de autenticación más descarga de información. En la Figura 23 (derecha) se muestra el error personalizado que el médico vería en caso de no estar operativo el servidor.

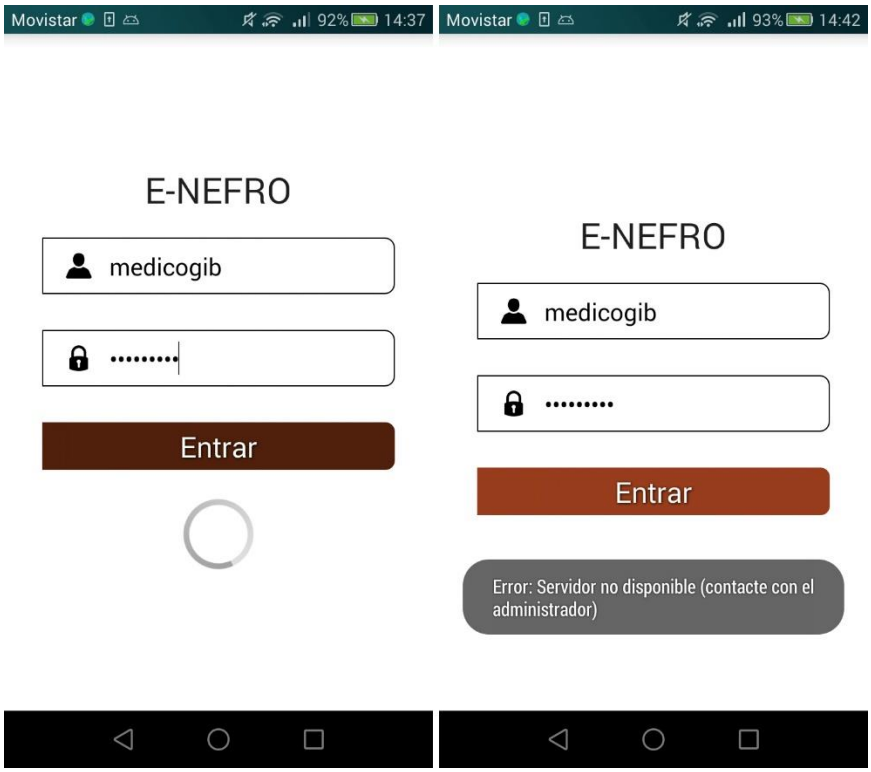


Figura 23. Barra de progreso de autenticación y descarga (izquierda), error personalizado (derecha).

3.4.3 Menú de lista de pacientes (MListaActivity)

Tras presionar el médico usuario el botón de lista de pacientes, se le conduce a un nuevo menú donde se le permite seleccionar un paciente en concreto para poder ver sus registros, planes de cuidados, o incidencias. Para ello simplemente se extrae la información descargada y almacenada con anterioridad por el método */patients*.

El diseño de dicho menú, el cual se muestra en la Figura 24 es lo más minimalista posible: una cabecera y una lista de pacientes. En la lista se muestra cada uno de los nombres de cada paciente en cuestión, aunque en una versión mejorada de la aplicación podría también mostrarse la fecha de su última actualización o incluso una fotografía tipo DNI.

Tras seleccionar un paciente de la lista se da lugar al menú más importante de la aplicación: el Menú Paciente (o MPac).

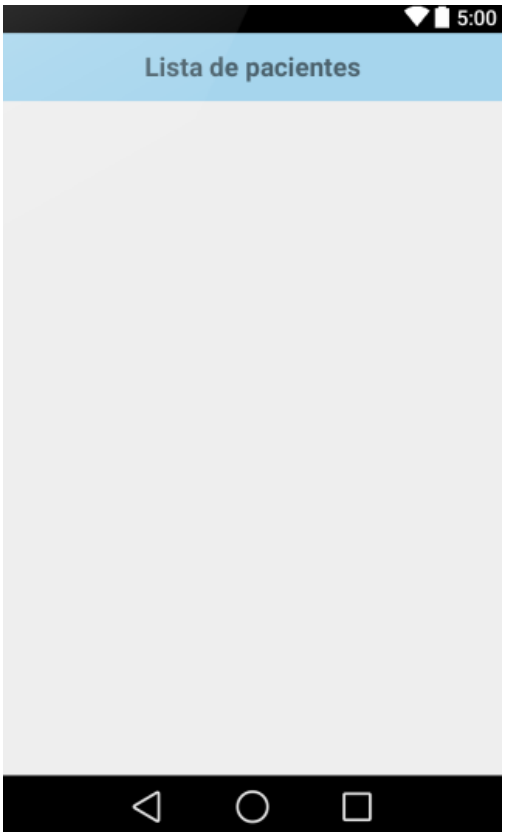


Figura 24. Diseño del menú Lista de Pacientes.

3.4.4 Menú paciente (MPacActivity)

Como ya se comentó anteriormente, este es el menú más importante de la aplicación debido a dos razones principales:

- Desde el menú paciente (en adelante MPac, Figura 25) se accede a todo el historial de un paciente en particular: su registro de variables, su plan de cuidados y sus últimas incidencias.
- MPac es accesible desde cualquier otro menú de la aplicación (Figura 26), como el menú de citas, mensajes, o incidencias, exceptuando el menú de configuración. Por ejemplo, cuando el médico está consultando sus últimas citas le es posible acceder al menú MPac de ese paciente simplemente tocando sobre dicha cita (ver apartado 3.5). Anteriormente en este documento se presentó un esquema que clarifica este escenario, el cual se reproduce nuevamente en este apartado.

Es el momento de introducir un elemento muy utilizado en la programación *Android*: el *fragment*. Un fragment es una porción de la interfaz gráfica que actúa de manera independiente al resto de elementos gráficos dentro de una *actividad* (o *activity*), que puede reutilizarse en otras *actividades,* y cuyo contenido y forma puede configurarse para que se adapte adecuadamente según los diferentes tamaños de pantalla desde los que se pudiera ejecutar la aplicación.

En general, y exceptuando todos los menús tratados hasta ahora, se ha incrustado un *fragment* dentro de cada respectivo menú. Esta decisión, que responde a las necesidades de ergonomía y fluidez, permitiría en una versión mejorada recurrir a las técnicas de *multi-pane layouts* al girar el dispositivo del usuario: mostrar dos menús a la vez en lugar de sólo uno, ahorrando tiempo útil al médico.

El menú MPac se abre inmediatamente después de presionar sobre un paciente y comprobar que la información contenida en la base de datos sobre ese médico no es nula. Llegados a este punto es importante hacer notar los severos controles que se han impuesto a la hora de extraer información de la base de datos: si dicha información no existe o no responde a los estándares acordados cuando se redactó la tabla de métodos del servidor, la aplicación lo hará notar al usuario e impedirá la apertura del menú solicitado.

Figura 25.Menú de paciente (MPac).

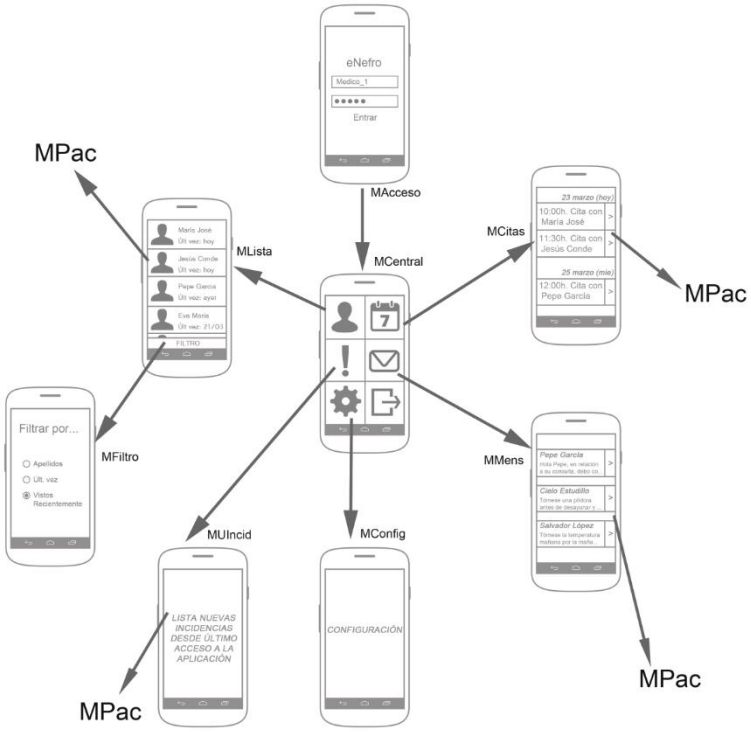


Figura 26. Se puede acceder a MPac desde casi cualquier menú.

La descarga de información de cada paciente se realiza *on demand*, es decir, bajo petición del usuario médico y siempre desde el menú MPac. Siempre será más eficiente, en términos de coste y tiempo, descargar sólo la información relativa a los pacientes deseados que la lista total de pacientes.

La información descargada se almacena en una *clase POJO* cuyas variables deben tener exactamente el mismo nombre que los parámetros del JSON devuelto por el servidor. La ventaja que ofrece este sistema de almacenamiento es su sencillez y la velocidad con la que se acceden los datos almacenados. La gran desventaja que ofrece es su volatilidad: cada vez que se solicita un nuevo paciente, los datos del paciente anterior son borrados. Sólo puede haber un paciente almacenado en el *POJO* a la vez. No obstante, la desventaja puede verse como una ventaja adicional: siempre se tendrá información actualizada del servidor.

En el Capítulo 4 se propondrá dos alternativas al sistema actual de almacenamiento de la información descargada.

Dentro del menú MPac se muestran tres accesos: el registro de pacientes, el plan de variables y las incidencias. Dichos menús se comentan a continuación.

3.4.4.1 Menú de registro de variables (MPacMVarActivity)

Aquí se recogen todos los registros de un paciente, los cuales se distribuyen en tres apartados: Variables, Medicamentos e Imágenes. Toda esta información es útil para el médico, el cual podrá actuar en función de dicha información a través de otro medio, en caso de que la situación lo requiera.

- Desde el menú de Variables podrá consultar cuáles son las últimas variables registradas por el paciente.
- Desde el menú de Medicamentos, el médico podrá saber cuáles fueron los últimos medicamentos tomados por sus pacientes.
- Desde el menú de Imágenes, el médico podrá observar cuál es el registro de imágenes que el paciente ha puesto a su disposición.

En la Figura 27 se muestra la distribución del menú de registro de variables (a partir de ahora MPacMVar). Puede observarse como, al igual que en MPac, en la zona superior se dispone de un espacio reservado al nombre del paciente que el médico seleccionó. De esta forma resultará más sencillo recordar qué paciente se seleccionó por última vez.

Puede observarse que el mecanismo de navegación entre menús es muy sencillo: utilizando el botón *volver* del propio dispositivo, se vuelve al menú anterior, y tocando sobre un botón se abrirá el menú deseado. No se ha involucrado ningún *ActionBar* ni elementos visuales similares que pudieran añadir complejidad extra a una aplicación que requiere de total simplicidad.

Debe recordarse que la navegación entre menús es totalmente fluida, debido a que la descarga se hizo justo después de seleccionar el paciente de la lista: entre menús no existen tiempos de espera.

A continuación, se tuvo que diseñar cada uno de los menús para las *Variables*, los *Medicamentos* y las *Imágenes*. Observando cuidadosamente la tabla de métodos del servidor, se observa que para los tres casos el patrón es el mismo: nombre de paciente, y un conjunto de datos que puede ser fácilmente acomodado en un recuadro personalizado. En la siguiente lista se muestra de un vistazo qué información se proporciona en cada caso:

- Registro de variables:
    - Identificador de la variable en consideración. Cada variable tiene un identificador único para referirse a ella de forma unívoca.
    - Nombre de la variable considerada (ejemplo: Temperatura, Glucemia,…)
    - Medida de dicha variable.
    - Fecha en que el paciente dio parte de dicha variable.
    - Descripción adicional que acompaña a la medida de la variable.
    - Comentario que acompaña al registro de la variable.
- Registro de medicamentos:
    - Identificador del medicamento en consideración. Cada medicamento tiene un identificador único para referirse a él de manera unívoca.

  - Nombre del medicamento.
  - Cantidad y unidad del medicamento registrado.
  - Fecha en que se registró la toma del medicamento.
  - Descripción adicional a la toma del medicamento.
  - Comentario que acompaña al registro del medicamento.
- Registro de imágenes:
  - Fecha en la que se registró la imagen.
  - Imagen en cuestión.
  - Descripción adicional adjunta a la imagen.
  - Comentario que acompaña al registro de la imagen.

Así pues, se procedió al diseño de dichos recuadros. El boceto realizado para el recuadro Variables se muestra en la Figura 28, siendo el diseño del resto de recuadros análogo, cambiando sólo el nombre de los campos y la información mostrada. Por otra parte, en la Figura 29 se muestra el diseño real del menú de registro de variables con los datos de prueba utilizados durante el desarrollo.

Figura 27. Menú de registro de variables (MPacMVar).

Figura 28. Diseño adoptado para acomodar la información de un registro de variable.

Figura 29. Diseño final que muestra varios registros de variables (MPacMVarMVars).

3.4.4.2 Menú de incidencias (MPacMIncActivity)

Las incidencias son importantes debido a que permiten al médico conocer el estado de sus pacientes y estar al tanto de todos los problemas (incidencias) de última hora, pudiendo actuar como mejor convenga por otro medio. Son cinco las incidencias consideradas:

- Dolor abdominal.
- Asfixia/Disnea.
- Fiebre.
- Edemas.
- Calambres.
- Picores.

El funcionamiento y diseño de este menú es muy similar al del MPacMVar anterior, el cual se muestra en la Figura 30. Consiste tan solo en una lista donde se muestran las incidencias del paciente deseado. A continuación se muestra la información que acompaña a una incidencia:

- Descripción de la incidencia observada por el paciente.
- Fecha a la que se tomó nota de la incidencia.
- Comentario que acompaña a la incidencia.

3.4.4.3 Menú de plan de cuidados (MPacMPCuidActivity)

Un plan de cuidados es una protocolización de los pasos a seguir que el médico nefrólogo comunica a su paciente. De esta manera, el médico le pide a su paciente que se mida ciertas variables en momentos determinados del día, que se tome ciertos medicamentos, que se haga ciertas fotografías o que se pase por consulta cada cierto tiempo.

Desarrollado de manera análoga a los dos menús anteriores. Se ofrece el plan de cuidados de un determinado paciente separando el plan de cuidados en cuatro subgrupos: Variables, Medicamentos, Imágenes y Citas. El diseño de dicho menú se muestra en la Figura 31, que como puede observarse es muy similar al menú MPacMVar. De nuevo, todos los submenús están formados por listas, por lo que no hay ninguna novedad en este apartado, más allá de los campos de información obtenidos de la base de datos. Dada la gran cantidad de información devuelta por la base de datos para cada plan, su inclusión en este apartado sería tan incómodo como

redundante: para consultarla, visite el Capítulo 5 (Anexos), donde se muestra la tabla de métodos.

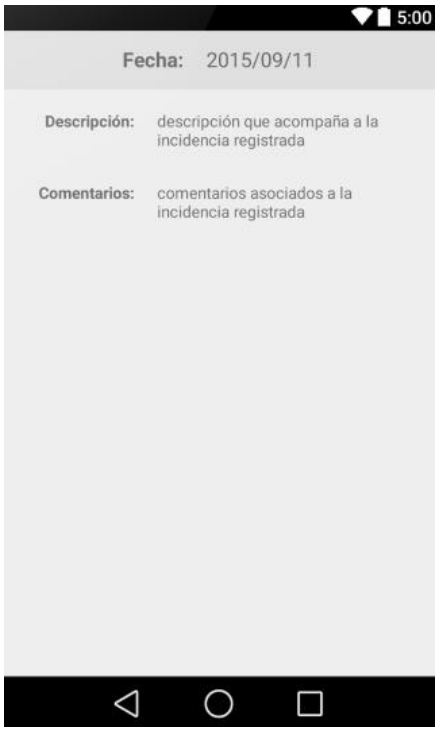


Figura 30. Menú de incidencias para un paciente (MPacMInc).

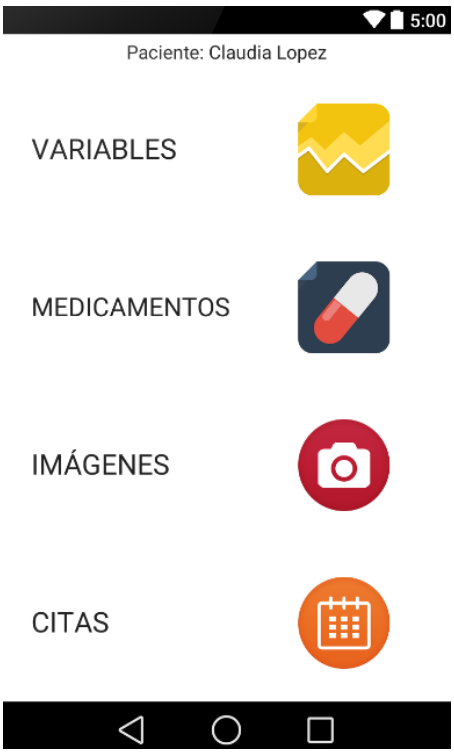


Figura 31. Menú de plan de cuidados (MPacMPCuid).

3.4.5 Menú de consulta de citas (MCitasActivity)

Como ya se explicó previamente, la descarga de la lista de citas se lleva a cabo tras solicitarlo el médico y no antes. Las citas permiten al nefrólogo llevar un control tangible de la mejora del paciente renal, siendo estos controles irremplazables por ninguna aplicación que se pueda diseñar. Es importante recordar que la aplicación

desarrollada no pretende ser un sustitutivo de los controles médicos rutinarios establecidos por el profesional, sino un potenciador de la relación médico-paciente que permita al nefrólogo saber cuál es la situación de sus pacientes en cada momento.

La petición de la lista de citas se hace a través del método */appointments*, el cual devuelve la siguiente información:

- Fecha de la cita.
- Paciente con el que se tiene la cita.
- Localización de la cita.
- Descripción y observaciones correspondientes a la cita.
- Comentarios hechos por el profesional.
- Comentarios hechos por enfermería.

En relación al aspecto gráfico, este menú introduce una novedad. Anteriormente se hizo uso de los elementos-lista *ListView* para mostrar variables, medicamentos e imágenes. Si la información contenida por cada elemento de la lista es muy extensa, no se podrá mostrar todos los elementos en la pantalla del usuario a la vez. Si bien esto no es un gran inconveniente cuando se trata de consultar variables, medicamentos o imágenes, sí que podría suponer un problema a la hora de consultar las citas.

Analizando los elementos visuales ofrecidos por Android, se pudo encontrar el elemento ideal para esta situación: el *ExpandableListView*. Este *widget* permite crear una lista cuyos elementos son plegables/desplegables. Cuando dichos elementos están plegados, sólo se muestra un encabezado con la información suficiente como para hacer saber al usuario en qué fecha es la cita. Desplegando los elementos se muestra toda la información extra. En la Figura 32 se muestra las citas de ejemplo con las que se trabajó durante el desarrollo.

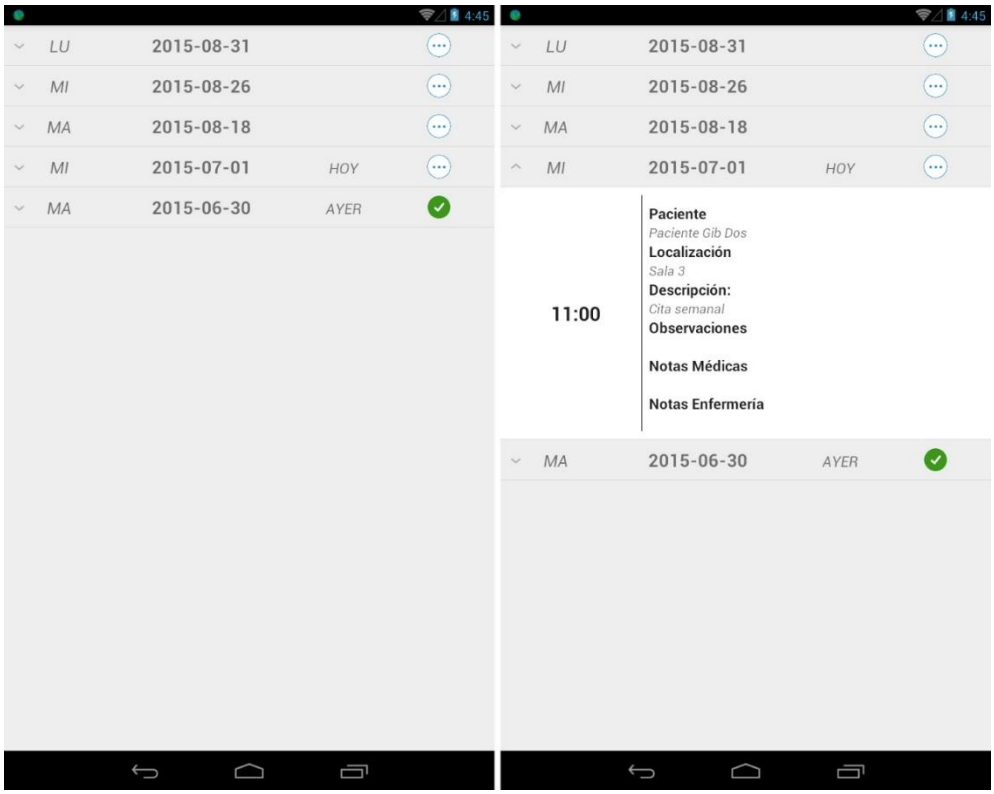


Figura 32. Menú de citas (MCitas).

La proximidad de las citas se puede apreciar a simple vista mediante el uso de tres iconos diferentes y el uso de un texto personalizado. El diseño de los encabezados está formado por:

- Un texto a la izquierda que indica el día de la semana: "LU", "MA", "MI",… "DO".
- Un texto central en negrita que indica la fecha exacta de la cita en formato "*yyyy-mm-dd*"

- Un texto a la derecha que sólo puede tener tres valores: “AYER”, “HOY” o “MAÑANA”. Ayuda a identificar fácilmente la relatividad de fechas de las citas respecto a la fecha del dispositivo del usuario.
- Un icono a la derecha que tomará uno de los siguientes valores:
    - Verde, si la cita ya ha concluido. En la imagen se puede apreciar dicho icono.
    - Azul con tres puntos. Indica que la cita aún no ha tenido lugar.
    - Amarillo. Este icono aparece cuando queda menos de 30 minutos para la cita en cuestión. Como una futura mejora de la aplicación, podría permitirse al usuario elegir en el menú de configuración con cuánto tiempo de antelación desea que sea avisado.

Este menú ofrece una novedad en cuanto a navegación entre menús: permite al médico consultar los registros, variables e incidencias del paciente con el que tiene la cita con tan sólo presionar sobre dicha cita. Sin duda es una funcionalidad bastante útil que se alinea con los objetivos de simplicidad y ergonomía, donde se evita al médico tener que volver hasta la lista de pacientes, buscar al paciente en cuestión y consultar sus historiales.

3.4.6 Menú de consulta de mensajes (MMensActivity)

El médico podrá intercambiar mensajes con sus pacientes a través de otra plataforma. Estos mensajes, que pueden servir por ejemplo para pedir al paciente más información o para dar la bienvenida a un nuevo paciente, pueden consultarse desde la aplicación desarrollada. Es importante recalcar que el médico no escribirá mensajes desde la *app* (recordar que es de sólo lectura), sólo los podrá leer.

Los mensajes se solicitan al servicio web a través del método /*inbox*, el cual devuelve los siguientes campos:

- Fecha del mensaje.
- Remitente.
- Asunto del mensaje.
- Cuerpo del mensaje.
- Indicador de si un mensaje ha sido leído o aún no.

Este menú tiene un comportamiento y diseño similar al menú de citas. Sin embargo, se ha programado un algoritmo especial que permita agrupar todos los mensajes devueltos por el servidor según el remitente. De esta manera, si se tuvieran quince mensajes correspondientes a sólo cinco remitentes (tres mensajes por remitente), en la lista no se mostraría quince encabezados, sino tan sólo cinco. En la Figura 33 se muestran varios mensajes de ejemplo utilizados durante el desarrollo de la aplicación.

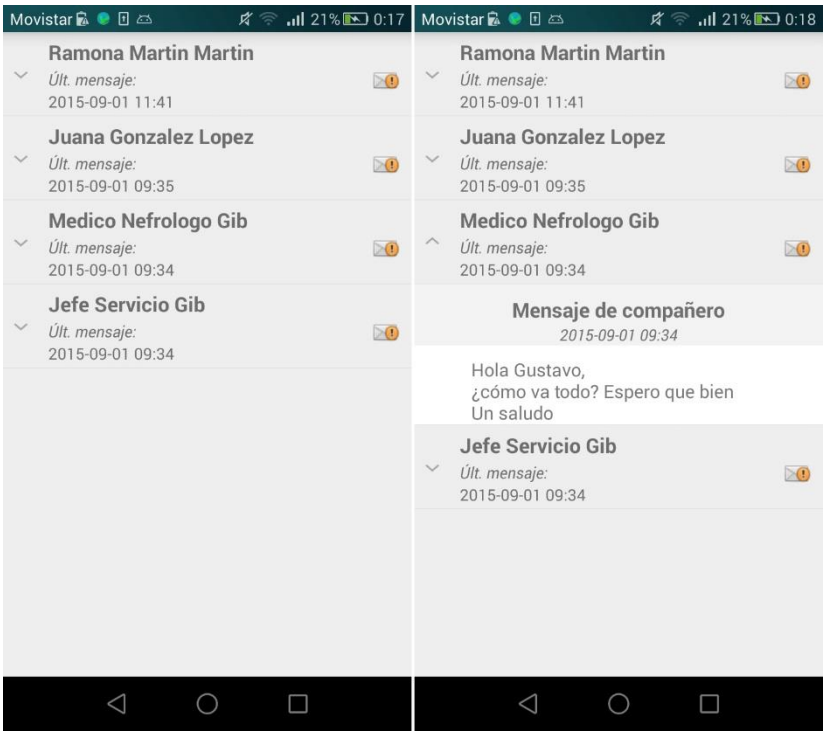


Figura 33. Menú de mensajes (MMens).

Como se puede apreciar en la imagen, se utiliza un icono personalizado para indicar cuándo hay un mensaje no leído en algún grupo de chat. Se mostraría otro en forma de *tick* verde si todos los mensajes de un grupo de chat hubieran sido leídos.

Como funcionalidad adicional y de manera análoga al menú de citas, se ofrece al médico poder consultar el historial del paciente con el que intercambia mensajes con tan sólo presionar sobre el mensaje que desee.

3.4.7 Menú de consulta de incidencias (MUIncidActivity)

Este menú permite al usuario nefrólogo acceder de manera general a todas las incidencias de todos sus pacientes, mientras que el anterior menú de incidencias permitía acceder a las incidencias de un paciente en particular. El menú ordena las incidencias por orden cronológico (se muestran primero las incidencias más recientes). Como futura mejora podría desarrollarse un método que permita mostrar sólo las incidencias no leídas (o bien mostrar las ya leídas con un color degradado de fondo, por si se deseara volver a consultar la incidencia). Para ello sería también necesario modificar la información devuelta por el servidor y añadir un campo adicional. Actualmente, las incidencias se piden al servidor a través del método */incidences*, que devuelve la siguiente información:

- Fecha de la incidencia.
- Paciente que notificó de la incidencia.
- Descripción de la incidencia.
- Comentario que acompaña a la incidencia.

En cuanto al diseño, éste es muy similar a los anteriores menús, y descarga de información como siempre a petición del cliente. La Figura 34 muestra el diseño del menú.

Figura 34. Menú de inciencias globales, de todos los pacientes (MUIncid).

3.4.8 Menú de configuración (MConfigActivity)

Se ha añadido un menú extra en el que se permitiría al médico personalizar la aplicación. Aunque esta opción se contempla como una posible futura mejora de la aplicación, en el Capítulo 4 se proporciona información adicional sobre qué podría añadirse en este menú de configuración.

3.4.9 Tabla de errores

Como ya se ha comentó en el Capítulo 2, la aplicación eNefro incorpora un sistema de detección y comunicación de errores bastante exhaustivo. Tanto es así que se ha diseñado un sistema de enumeración de errores (desde el 000 hasta el 006) además de los errores informativos que no requieren de un código específico, ayudando al médico a comprender la naturaleza del error. Estos errores se muestran en la Tabla 2.

Tabla 2. Posibles errores durante el uso de la aplicación.

Código de error	Explicación del error
000	*Timeout* alcanzado o error en la red. No se pudo conectar o descargar la información. Aumente el *timeout*, compruebe su conexión a la red o contacte con el administrador del servidor.
001	El servidor ha recibido una porción de la petición y ésta es correcta. No se debería obtener este código en ningún caso, excepto posiblemente si se modifican los nombres de los campos de los códigos JSON devueltos por el servidor.
002	Al igual que el error 001, el error 002 no debería obtenerse nunca excepto si se hacen modificaciones a la tabla de métodos. En éste último caso sería necesario cambiar también las clases *POJO* dentro de la aplicación.

003	Error de redirección. El error reside en el servidor y no en el cliente.
004	Error en la petición. Este error es susceptible de aparecer si, por ejemplo, se cambian los nombres de los métodos devueltos por el servidor.
005	Error en el servidor. Puede que se haya producido un error interno en el servidor o que simplemente no esté disponible en ese momento.
006	Este error no debería ocurrir nunca. Actualmente los códigos de respuesta HTTP van enumerados del 1xx al 5xx y en ningún caso existe un código más allá del 5xx. Por lo tanto, a excepción de que se modifique el esquema HTTP en un futuro, este error no debería aparecer.
"Error al (…)"	Los errores que siguen este esquema no tienen asignado un código, facilitando así una mejor comprensión por parte del usuario médico. Ejemplos de este error: "Error al obtener la lista de pacientes", "Error al obtener la lista de incidencias". Generalmente ocurren tras obtener un error muy conocido, llamado *NullPointerException*. Significa que la base de datos no tiene la información solicitada (Por ejemplo, se produciría en caso de solicitar las incidencias de un paciente pero en la base de datos no figura ninguna).
"Este usuario no tiene pacientes asignados"	Aparece cuando un médico se autentifica correctamente pero no tiene ningún paciente asignado en la base de datos. La aplicación se cierra para evitar uso indebido.
"Error: debe introducir usuario y clave"	Error al intentar autentificarse uno o más campos en blanco.
"Error: (…)"	La petición es correcta pero los campos *msg* devueltos por el servidor (ver Tabla de Métodos) no están vacíos. Esto quiere decir que, por ejemplo, la sesión del usuario ha expirado, o que la base de datos no puede proporcionar dichos datos. La cadena de texto *msg* es representativa del error y debe diseñarse desde el código del servidor.

4 CONCLUSIONES Y LÍNEAS FUTURAS

4.1 Conclusiones

En este Trabajo Fin de Grado se ha desarrollado una aplicación móvil para Android que permite al usuario médico consultar toda la información relativa a sus pacientes renales en cualquier momento y lugar. Con esta *app* se consigue proporcionar seguridad al paciente, pues sabe que su médico puede consultar instantáneamente cualquier información registrada, como posibles incidencias, registros de variables, registros de medicamentos, mensajes internos, citas… Aun siendo este uno de los principales objetivos perseguidos, también se consigue satisfacer otras necesidades, como la de disminuir el número de consultas entre médicos y pacientes, algo muy importante en casos de pacientes con movilidad limitada.

Como con cualquier proyecto de esta categoría, sólo ha sido posible extraer habilidades positivas. El autor de este TFG, proveniente del Grado en Ingeniería Aeroespacial y voluntario para realizar este proyecto, ha aprendido la importancia de la tecnología en el campo de la sanidad y, en especial, en el de la nefrología. Ha aprendido las habilidades básicas necesarias y metodología propia para realizar cualquier proyecto relacionado con las aplicaciones *Android*. Y en general, ha aprendido cómo se trabaja en equipo junto con otras partes del proyecto, buscando las soluciones más convenientes en cada momento para las dos (o más) partes implicadas.

Una de las conclusiones que se desprende tras la realización de este proyecto es la facilidad con la que *Google* permite desarrollar aplicaciones *Android*, a través de una librería de clases (*SDK*) bien documentada, con una curva de aprendizaje relativamente suave. Cualquier persona con experiencia previa en la programación *Java* podrá aprender en un tiempo razonable a realizar aplicaciones Android, lo que no significa que dichas aplicaciones vayan a ser óptimas; esto conduce a la segunda conclusión: hay muchas formas diferentes para abordar un mismo problema. Dado que existen diversas formas de plantear y resolver un problema, antes de comenzar a trabajar en una solución es importante estudiar qué otras alternativas existen, cuán convenientes son, y qué ventajas/desventajas ofrecen con respecto a las demás. Sólo de esta forma puede garantizarse que el camino recorrido hasta la resolución del problema ha sido el óptimo.

Merece la pena llegados a este punto volver al interrogante sobre la conveniencia de la elección de Android como el sistema operativo sobre el que desarrollar la aplicación. ¿Por qué no se escogió el sistema *iOS*, de *Apple*, o el sistema *Windows Phone* de *Microsoft*? La respuesta a esta pregunta es la siguiente: este proyecto se desarrolla en España, y su uso va dirigido a usuarios españoles. Según una reciente estadística [24] llevada a cabo por *The App Date*, de cada 1000 usuarios con *smartphone,* 890 utiliza Android mientras que sólo 76 utiliza iOS (el resto utiliza Windows o cualquier otro sistema operativo, ver Figura 35). Esto quiere decir que de por cada 12 personas que usan Android, existe otra que utiliza iOS, una estadística más que favorable y con la que se responde al *por qué* de la elección de este sistema operativo.

Aunque anteriormente se afirmó que la curva de aprendizaje necesaria para desarrollar una aplicación Android es relativamente suave, el tiempo que requiere aprender a usar las clases de su librería *SDK* no es nulo. Uno de los más grandes impedimentos con los que se ha encontrado durante la ejecución del proyecto fue el limitado conocimiento que se tenía sobre las clases de la librería *SDK* de *Android*. Dicho conocimiento era lo suficientemente limitado como para tener que repasar toda la documentación de Android (o, al menos, las clases más necesarias para poder completar la aplicación con éxito). Mostrada la principal limitación, merece la pena mencionar también una de las principales facilidades del proyecto: las nociones de programación que se poseían. El haber aprendido a programar desde una muy temprana edad ha influido positivamente en el desarrollo de las habilidades necesarias para completar el proyecto.

El resultado de la realización del proyecto ha sido una aplicación móvil que permite a los nefrólogos consultar información sanitaria de sus pacientes renales. Algunos resultados del desarrollo de dicha aplicación podrían ser:

- Diseño y desarrollo de un protocolo de comunicación con el servicio web, el cual alimenta a la aplicación con los datos que el usuario nefrólogo requiera. Los datos proporcionados por el servicio web vienen bajo el formato *JSON*, por lo que también ha sido necesario diseñar las clases necesarias que almacenasen dicha información.

- Para la descarga de información se ha diseñado numerosas clases intermedias que establecen una conexión con el servidor y finalmente realizan una petición *POST,* para posteriormente almacenar dicha información en las clases *POJO* correspondientes.
- Se ha diseñado un árbol de navegación entre menús bastante sencillo y funcional, por lo que a priori los usuarios nefrólogos deberían encontrar en esta aplicación una solución bastante intuitiva a sus necesidades. Más adelante debe comprobarse que efectivamente esto es así seleccionando médicos voluntarios y entregándoles unas encuestas de satisfacción personalizadas.

Figura 35. Estadística [24] sobre uso de diferentes sistemas operativos en España.

4.2 Futuras líneas de ampliación y mejora

La aplicación fruto de este proyecto es un primer prototipo de lo que sería la aplicación final. Falta validar la efectividad de la solución propuesta para los diferentes menús de la aplicación, quizás a través de unas encuestas de satisfacción que rellenarían diferentes usuarios médicos de manera voluntaria. Tras los resultados de las encuenstas de satisfacción, es probable que se necesite modificar el programa de manera que recoja las posibles sugerencias y/o incomodidades detectadas por los usuarios voluntarios.

A continuación se describe una lista que recoge las posibles mejoras a acometer en una posible futura versión mejorada del mismo. Se recogen a continuación, aunque el orden de presentación ha sido ciertamente aleatorio:

- El menú de configuración. Las opciones a añadir en el menú de configuración son muchas, tantas como usuarios de la aplicación haya. De manera general se pasa a comentar algunos puntos a recoger en el menú de configuración en una futura versión de la aplicación:
    - Permitir al usuario elegir el tamaño de letra utilizado. Aunque en el diseño de los menús se ha utilizado un tamaño de letra rondando los 16-20sp, habrá usuarios que prefieran otro tamaño diferente.
    - Permitir al usuario elegir el tiempo máximo de intento de conexión y descarga. Actualmente se ha diseñado la aplicación de manera que este tiempo, también conocido como *timeout*, sean veinte segundos exactos. Si este tiempo se alcanza antes de establecer la conexión, se devuelve

un error sugiriendo que se compruebe la conectividad del dispositivo. Puede ocurrir, por ejemplo, que la tarifa de datos del usuario sea muy lenta y tome más de veinte segundos en descargar. En dicho caso interesaría aumentar el timeout en unos segundos más, lo que podría hacerse a través del menú de configuración.
    - Permitir al usuario elegir si desea recibir notificaciones de nuevas incidencias o mensajes a través de la barra de notificaciones. O incluso enviar dicha notificación por *SMS* y/o *e-mail* al médico.
    - Permitir al usuario mantener la sesión abierta tras salir de la aplicación. Actualmente la sesión se cierra tras salir de la aplicación para mayor seguridad, teniendo que introducir usuario y clave de nuevo.
- Implementar un sistema mejorado de descarga. En este sistema se haría uso de los sistemas de almacenamiento por memoria caché, permitiendo al usuario ahorrar tiempo y dinero, en caso de estar usando la tarifa de datos. Se proponen dos alternativas:
    - Utilizar un sistema de *almacenamiento con memoria caché*. Su funcionamiento es fundamentalmente el mismo que el usado actualmente, pero la memoria caché permitiría recordar una cierta cantidad de pacientes establecida de antemano. Por ejemplo, recordar los últimos cinco pacientes solicitados, de manera que si el médico quiere volver atrás y revisar su información no tenga que volver a descargar toda la información del servidor.
    - Utilizar un sistema de *almacenamiento con temporizador interno*. Esta solución equivale a la misma que la anterior, pero con un tamaño de memoria caché suficiente como para poder almacenar todos los pacientes deseados durante un tiempo establecido. En la Figura 36 se muestra cuál sería el diagrama para descargar/solicitar la información con un temporizador interno.

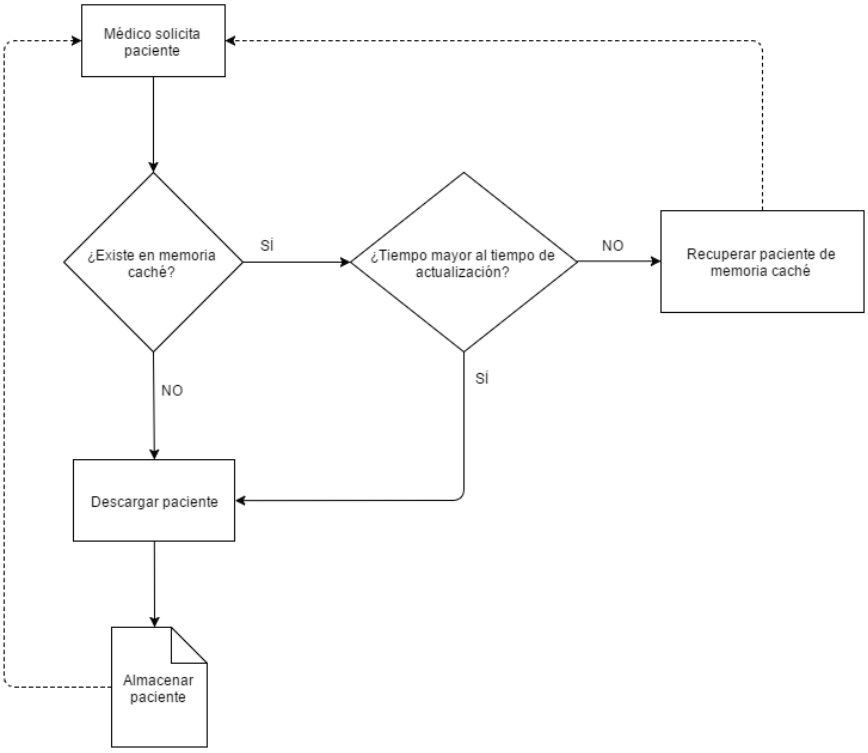


Figura 36. Método de descarga con almacenamiento caché y temporizador interno.

- Mostrar una lista de pacientes con mayor información. Actualmente sólo se muestra el nombre del paciente. En una versión futura podría mostrarse más información, como la fecha de la última actualización o incluso una fotografía del paciente tipo DNI (ver Figura 37).

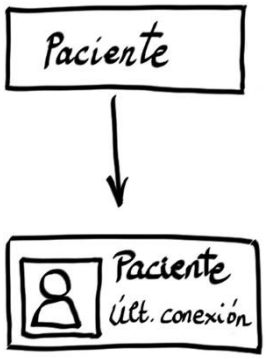


Figura 37. Posible mejora en la manera de mostrar los pacientes.

- Hacer uso del concepto de *multi-pane layouts* en Android. Si el tamaño del dispositivo es muy grande, quizás convenga mostrar dos menús a la vez en lugar de sólo uno, pues en este último caso se estaría derrochando espacio libre. Por ejemplo, en lugar de mostrar primero la lista de pacientes y luego el menú MPac asociado a dicho paciente, podría dividirse el tamaño de la pantalla en dos: a la izquierda la lista de pacientes y a la derecha el menú MPac. Puede leerse más información sobre esta técnica en la página web oficial de desarrolladores Android [25]. Precisamente en la Figura 38 se muestra un ejemplo ilustrando esta técnica.

Figura 38. Técnica multi-pane layout de Android.

- Implementar el filtro de búsqueda ilustrado en la fase de prediseño. Para ello también sería necesario añadir nuevos métodos al repertorio de métodos del servidor.
- Permitir al nefrólogo usuario añadir notas personales acerca de un paciente determinado a nivel local (es decir, sin tener que modificar la base de datos). Estas notas serían visibles únicamente al médico usuario y se le permitiría escribir cualquier cosa sobre un paciente, como recordatorios o información sensible.
- Escribir un algoritmo que permita representar gráficamente la evolución de uno o más parámetros fisiológicos de cualquier paciente.
- Añadir una alerta sonora en caso de que una nueva incidencia haya sido registrada. Para que esto sea posible, es necesario acudir a un nuevo concepto de Android [26] que no se ha utilizado en el proyecto, conocido como *Servicio* (*Service*). Un servicio permite crear una interfaz completamente invisible a ojos del usuario que sirve únicamente para realizar trabajo de fondo incluso aunque el propio usuario

no tenga la aplicación abierta. Para generar una alerta sonora en caso de nueva incidencia, primero hay que comprobar si hay nuevas incidencias cada cierto tiempo (proceso de conexión a la base de datos más descarga). Si la aplicación está cerrada, éste proceso sólo es posible a través de un servicio.

- Diseñar la aplicación para otras plataformas, como iOS o Windows Phone 8/10. Puede hacerse reescribiendo el código desde cero, aunque si el propósito principal del proyecto hubiera sido la capacidad multi-plataforma hubiera sido más conveniente utilizar la herramienta de Xamarin [27], donde casi todo el código que se escribe se comparte entre todas las plataformas móviles (a excepción de las tareas más específicas, propias de cada plataforma).
- Android Wear. Los relojes inteligentes de Android se están integrando perfectamente en el mercado, por lo que no estaría de más diseñar una versión de esta aplicación para los *smartwatches* (y no sólo para los *smartphones*).

5 REFERENCIAS

[1] «Documento Marco sobre Enfermedad Renal Crónica (ERC) dentro de la Estrategia de Abordaje a la Cronicidad en el SNS,» [En línea]. Available: http://www.msssi.gob.es/organizacion/sns/planCalidadSNS/pdf/Enfermedad_Renal_Cronica_2015.pdf. [Último acceso: 14 Septiembre 2015].

[2] «Documento de consenso sobre la Enfermedad Renal Crónica,» [En línea]. Available: http://www.semfyc.es/es/componentes/ficheros/descarga.php?MTIxMDg%3D. [Último acceso: 14 Septiembre 2015].

[3] «Movilzona,» [En línea]. Available: http://www.movilzona.es/2014/08/08/google-play-supera-por-primera-vez-a-la-app-store-de-apple-en-numero-de-aplicaciones/. [Último acceso: 14 Septiembre 2015].

[4] «Anestesiólogos Colombia,» [En línea]. Available: http://www.anestesiologoscolombia.com/publicaciones/1261/Aplicaciones.pdf. [Último acceso: 14 Septiembre 2015].

[5] «Oracle,» [En línea]. Available: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. [Último acceso: 14 Septiembre 2015].

[6] «Desarrolladores Android (Página oficial),» [En línea]. Available: https://developer.android.com/sdk/index.html. [Último acceso: 14 Septiembre 2015].

[7] «DDMS - Desarrolladores Android,» [En línea]. Available: http://developer.android.com/tools/debugging/ddms.html.

[8] J. López García, « eBook - Android Development (Downloading and Installing the necessary tools),» [En línea]. Available: http://papyruseditor.com/web/27431/Android-Development-1)-Downloading-and-installing-the-necessary-tools. [Último acceso: 14 Septiembre 2015].

[9] «Genymotion homepage,» [En línea]. Available: https://www.genymotion.com. [Último acceso: 14 Septiembre 2015].

[10] «Iconarchive homepage,» [En línea]. Available: http://www.iconarchive.com. [Último acceso: 14 Septiembre 2015].

[11] «Photoshop - Adobe,» [En línea]. Available: http://www.photoshop.com/products/photoshop. [Último acceso: 14 Septiembre 2015].

[12] «Draw.io Flowcharts,» [En línea]. Available: https://www.draw.io. [Último acceso: 14 Septiembre 2015].

[13] «Evernote,» [En línea]. Available: https://evernote.com. [Último acceso: 14 Septiembre 2015].

[14] «Dropbox homepage,» [En línea]. Available: https://www.dropbox.com. [Último acceso: 14 Septiembre 2015].

[15] «Retrofit - GitHub,» [En línea]. Available: https://github.com/square/retrofit. [Último acceso: 14 Septiembre 2015].

[16] «Square homepage,» [En línea]. Available: http://square.github.io. [Último acceso: 14 Septiembre 2015].

[17] «Okhttp - GitHub,» [En línea]. Available: http://square.github.io/okhttp. [Último acceso: 14 Septiembre 2015].

[18] «Okio - GitHub,» [En línea]. Available: https://github.com/square/okio. [Último acceso: 14 Septiembre 2015].

[19] «Librería Volley - Google,» [En línea]. Available: https://developer.android.com/training/volley/simple.html. [Último acceso: 14 Septiembre 2015].

[20] «Instructure Blog,» [En línea]. Available: http://instructure.github.io/blog/2013/12/09/volley-vs-retrofit/. [Último acceso: 14 Septiembre 2015].

[21] «OWASP,» [En línea]. Available: https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet#Defense_Option_3:_Escaping_All_User_Supplied_Input. [Último acceso: 14 Septiembre 2015].

[22] «SHA-2 - Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/SHA-2. [Último acceso: 14 Septiembre 2015].

[23] B. H. a. B. Phillips, de *Android Programming - The Big Nerd Ranch Guide*, Big Nerd Ranch, 2013, p. 421.

[24] «Yeeply - Informe sobre el uso de apps en España (2014),» [En línea]. Available: https://www.yeeply.com/blog/informe-sobre-el-uso-de-apps-en-espana-2014/. [Último acceso: 14 Septiembre 2015].

[25] «Multipane Layots - Página oficial de desarrolladores Android,» [En línea]. Available: http://developer.android.com/training/design-navigation/multiple-sizes.html#multi-pane-layouts. [Último acceso: 14 Septiembre 2015].

[26] «Services - Página oficial de desarrolladores Android,» [En línea]. Available: http://developer.android.com/guide/components/services.html. [Último acceso: 14 Septiembre 2015].

[27] «Xamarin homepage,» [En línea]. Available: http://xamarin.com. [Último acceso: 14 Septiembre 2015].

ANEXO A: TABLA DE MÉTODOS DEL SERVIDOR

Tabla 3. Tabla de métodos del servidor.

Nombre del método	/login
Descripción	Realiza la autenticación con el servicio web
Tipo de método	POST
Parámetros de entrada	usuario (String, obligatorio): nombre del usuario
	password (String, obligatorio): contraseña del usuario
Respuesta (JSONObject)	ok (Boolean, obligatorio): True si todo ha ido correcto, False si ha habido un error
	msg (String, obligatorio): Cadena vacía si todo ha ido correcto y mensaje de error si ha habido algún error
	data (JSONObject, obligatorio): • id_sesion (String, obligatorio): identificador de sesión
Posibles errores	Autenticación incorrecta
	Error en el servidor

Nombre del método	/patients
Descripción	Recupera la lista de pacientes relacionados con el profesional usuario
Tipo de método	POST
Parámetros de entrada	id_sesion (String, obligatorio): identificador de sesión
Respuesta (JSONObject)	ok (Boolean, obligatorio): True si todo ha ido correcto, False si ha habido un error
	msg (String, obligatorio): Cadena vacía si todo ha ido correcto y mensaje de error si ha habido algún error

	patients (ArrayList, obligatorio):
	• (0..*) paciente (JSONObject):
	  ○ id (String, obligatorio): identificador del paciente
	  ○ name(String, obligatorio): nombre del paciente

Posibles errores	Sesión expirada
	Error en el servidor

Nombre del método	/patientview
Descripción	Recupera incidencias, registros y plan de cuidados de un paciente
Tipo de método	POST
Parámetros de entrada	id_sesion (String, obligatorio): identificador de sesión
	id_paciente (String, obligatorio): identificador de paciente
Respuesta (JSONObject)	ok (Boolean, obligatorio): True si todo ha ido correcto, False si ha habido un error
	msg (String, obligatorio): Cadena vacía si todo ha ido correcto y mensaje de error si ha habido algún error
	data (JSONObject, obligatorio): • incidencia (ArrayList, obligatorio) o (0..*) JSONObject: ▪ incidencia_fecha (String, opcional) ▪ incidencia_descripcion (String, opcional) ▪ incidencia_comentario (String, opcional) • registro_medicamento(ArrayList, obligatorio) o (0..*) JSONObject: ▪ registro_med_id (String, opcional) ▪ registro_med_descripcion(String, opcional) ▪ registro_med_medicamento(String, opcional) ▪ registro_med_cantidad (String, opcional) ▪ registro_med_unidad(String, opcional) ▪ registro_med_fecha(String, opcional) ▪ registro_med_comentario(String, opcional) • registro_variable(ArrayList, obligatorio) o (0..*) JSONObject: ▪ registro_var_id (String, opcional) ▪ registro_var_fecha(String, opcional) ▪ registro_var_descripcion(String, opcional) ▪ registro_var_variable (String, opcional) ▪ registro_var_medidad(String, opcional) ▪ registro_var_comentario(String, opcional) • registro_imagenes(ArrayList, obligatorio) o (0..*) JSONObject: ▪ registro_img_fecha (String, opcional) ▪ registro_img_descripcion(String, opcional) ▪ registro_img_comentario(String, opcional) ▪ registro_img_imagen (String, opcional) • plan_imagenes(ArrayList, obligatorio) o (0..*) JSONObject: ▪ plan_img_id (String, opcional) ▪ plan_img_descripcion(String, opcional) ▪ plan_img_frecuencia(String, opcional) ▪ plan_img_xdias (String, opcional) ▪ plan_img_xhoras (String, opcional) ▪ plan_img_relcomidas(String, opcional)

	▪ plan_img_mincomidas(String, opcional) ▪ plan_img_fechainicio (String, opcional) ▪ plan_img_fechafin(String, opcional) ▪ plan_img_localizacion(String, opcional) ▪ plan_img_observaciones (String, opcional) • plan_medicamentos(ArrayList, obligatorio) o (0..*) JSONObject: ▪ plan_med_id (String, opcional) ▪ plan_med_descripcion (String, opcional) ▪ plan_med_medicamento (String, opcional) ▪ plan_med_cantidad(String, opcional) ▪ plan_med_medida(String, opcional) ▪ plan_med_frecuencia(String, opcional) ▪ plan_med_xdias (String, opcional) ▪ plan_med_xhoras (String, opcional) ▪ plan_med_relcomidas(String, opcional) ▪ plan_med_mincomidas(String, opcional) ▪ plan_med_fechainicio (String, opcional) ▪ plan_med_fechafin (String, opcional) ▪ plan_med_localizacion(String, opcional) ▪ plan_med_observaciones (String, opcional) • plan_variables(ArrayList, obligatorio) o (0..*) JSONObject: ▪ plan_var_id (String, opcional) ▪ plan_var_descripcion(String, opcional) ▪ plan_var_variable (String, opcional) ▪ plan_var_frecuencia(String, opcional) ▪ plan_var_xdias (String, opcional) ▪ plan_var_xhoras (String, opcional) ▪ plan_var_relcomidas(String, opcional) ▪ plan_var_mincomidas(String, opcional) ▪ plan_var_fechainicio (String, opcional) ▪ plan_var_fechafin (String, opcional) ▪ plan_var_localizacion(String, opcional) ▪ plan_var_observaciones (String, opcional) • plan_citas(ArrayList, obligatorio) o (0..*) JSONObject: ▪ plan_cita_id (String, opcional) ▪ plan_cita_descripcion(String, opcional) ▪ plan_cita_profesional(String, opcional) ▪ plan_cita_fecha (String, opcional) ▪ plan_cita_localizacion(String, opcional) ▪ plan_cita_observaciones (String, opcional)
Posibles errores	Sesión expirada
	Error en el servidor

Nombre del método	/appointments
Descripción	Recupera las citas programadas del usuario profesional
Tipo de método	POST
Parámetros de entrada	id_sesion (String, obligatorio): identificador de sesión

Respuesta (JSONObject)	ok (Boolean, obligatorio): True si todo ha ido correcto, False si ha habido un error
	msg (String, obligatorio): Cadena vacía si todo ha ido correcto y mensaje de error si ha habido algún error
	appointments(ArrayList, obligatorio) • (0..*) JSONObject: o cita_descripcion(String, opcional) o cita_paciente(String, opcional) o cita_paciente_id(String, opcional) o cita_fecha (String, opcional) o cita_localizacion(String, opcional) o cita_observaciones (String, opcional) o cita_coment_medico(String, opcional) o cita_coment_enfermeria (String, opcional)
Posibles errores	Sesión expirada
	Error en el servidor

Nombre del método	/inbox
Descripción	Recupera los mensajes recibidos del usuario profesional
Tipo de método	POST
Parámetros de entrada	id_sesion (String, obligatorio): identificador de sesión
Respuesta (JSONObject)	ok (Boolean, obligatorio): True si todo ha ido correcto, False si ha habido un error
	msg (String, obligatorio): Cadena vacía si todo ha ido correcto y mensaje de error si ha habido algún error
	messages(ArrayList, obligatorio) • (0..*) JSONObject: o msg_fecha(String, opcional) o msg_remitente(String, opcional) o msg_remitente_id(String, opcional) o msg_asunto (String, opcional) o msg_texto(String, opcional) o msg_leido (String, opcional)
Posibles errores	Sesión expirada
	Error en el servidor

Nombre del método	/incidences
Descripción	Recupera las incidencias de los pacientes
Tipo de método	POST
Parámetros de entrada	id_sesion (String, obligatorio): identificador de sesión
Respuesta (JSONObject)	ok (Boolean, obligatorio): True si todo ha ido correcto, False si ha habido un error
	msg (String, obligatorio): Cadena vacía si todo ha ido correcto y mensaje de error si ha habido algún error
	incidences(ArrayList, obligatorio) • (0..*) JSONObject: o inc_fecha(String, opcional) o inc _paciente(String, opcional) o inc _paciente_id(String, opcional) o inc _descripcion (String, opcional) o inc _comentario (String, opcional)
Posibles errores	Sesión expirada
	Error en el servidor