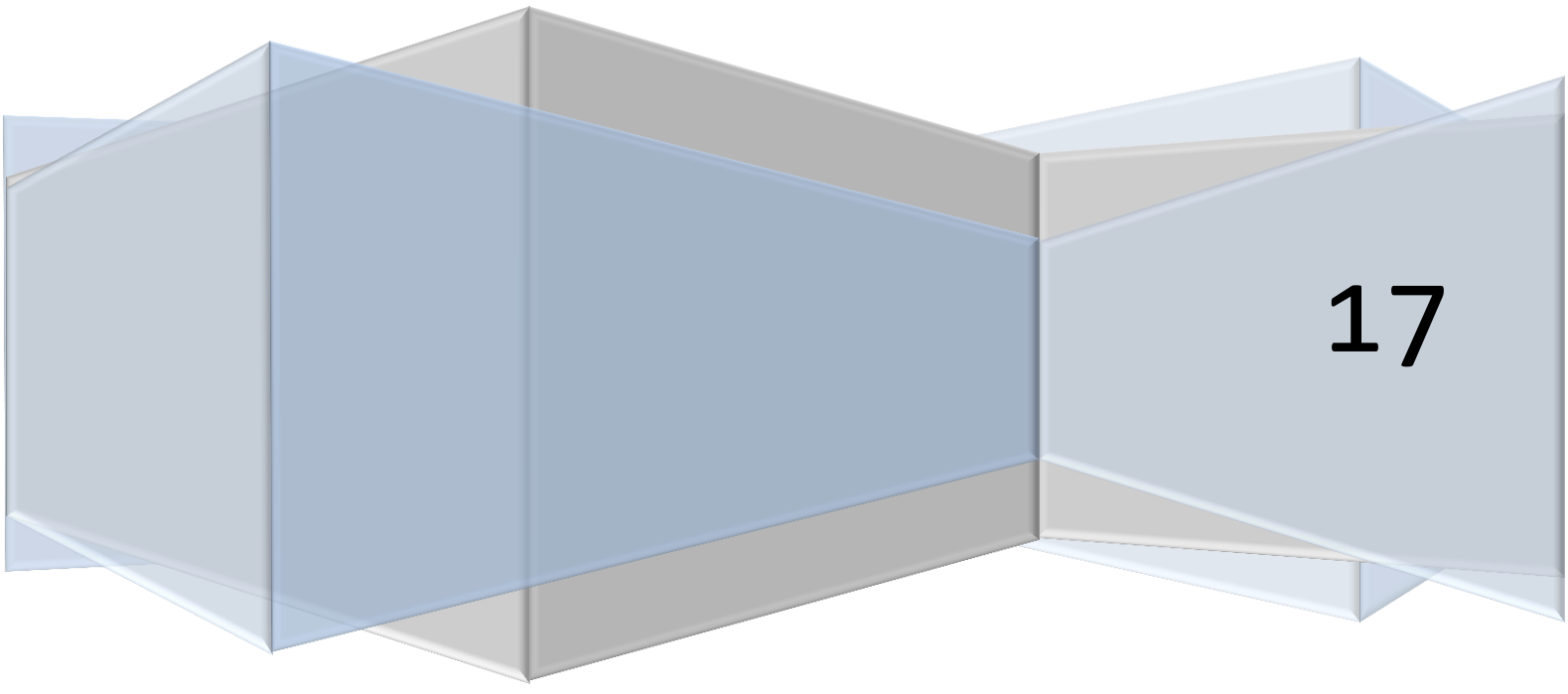
UOC

Memoria App Pink Card

Proyecto de grado

Francesc Cantallops Cifre



Abstracto

Proyecto de grado en el que se pretende crear una app para uso sanitario en el que el usuario o paciente podrá ver las citas que tiene programadas, cancelar y programar nuevas citas con el objetivo de ahorrar tiempo y recursos a la sanidad.

La aplicación será capaz de mostrar al paciente las citas que tiene programadas y dará avisos en diferentes intervalos con el objetivo de recordar las citas y que el paciente pueda, en caso de no poder asistir, cancelar y reprogramar sus citas.

Project of degree in which it is tried to create an app for sanitary use in which the user or patient will be able to see the appointments that have programmed, to cancel and to schedule new appointments with the objective of saving time and resources to the sanity.

The application will be able to show the patient the appointments that are scheduled and will give warnings at different intervals in order to remember the appointments and that the patient can, in case of not being able to attend, cancel and reschedule their appointments.

Resumen

En tiempos de constante transformación dónde las nuevas tecnologías están revolucionando el sistema en el que vivimos y facilitan la vida de cada uno de los que hacen uso de ella; vemos cómo la tecnología incursiona en campos antes insospechados llegando incluso al campo sanitario.

La sanidad se beneficia de este fenómeno de las apps desde hace varios años, pues existen aplicaciones de todo tipo que están en uso, en entidades públicas y privadas del mundo entero facilitando la atención de las entidades y mejorando la satisfacción de los usuarios.

Es por ello que el objeto de este proyecto es brindar una solución práctica y económica tanto para la sanidad como para el paciente; para ello se propone la creación de una aplicación para dispositivos móviles que les recuerde a los usuarios sus citas, analíticas y demás pruebas, permitiéndoles gestionar la confirmación o cancelación de las citas, así como la reprogramación de las mismas.

Vemos que un porcentaje importante de la población tiene acceso a dispositivos móviles, según datos de 2014 más de 15 millones de personas en España utilizan un **teléfono inteligente** y la conexión a Internet a través de estos dispositivos es cada vez mayor que desde los ordenadores, por lo que llegaríamos a brindar una mayor cobertura. IAB Spain señala que el teléfono inteligente ya es el principal dispositivo para **acceder a internet** en un 85%, dejando a los ordenadores en segundo lugar, con un 67%, y a las tabletas en tercero, con un 45%.

Para la creación de la app decidí optar por una aplicación nativa, en específico, Android ya que los españoles prefieren este sistema operativo en un 84%, frente a iOs, el sistema operativo de Apple, con un 12%.

Me propongo desarrollar una aplicación capaz de visualizar todas las citas de un paciente así como de enviar avisos de recordatorio y la posibilidad de confirmar o cancelar la cita. La idea es enviar avisos en diferentes intervalos, siendo el más interesante el de unas 72 horas antes de la cita para que el paciente en caso de cancelar la cita, esta pueda ser reutilizada por otro paciente y así ahorrar tiempo y costos.

Otro caso de estudio es la creación de colas de espera entre los usuarios registrados de la aplicación para que ellos puedan aprovechar las citas que son canceladas. Ofreciendo la cita a los primeros 4 de la bolsa de espera para que quién primero confirme pueda aprovecharla. De esta forma ayudamos a reducir las listas de espera y reducir costos a la administración.

También se da especial atención al apartado de la confidencialidad de los datos de paciente y en conformidad a la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal; especialmente a los citados en los artículos Artículo 7. (Datos especialmente protegidos) Artículo 8. (Datos relativos a la salud), Artículo 10 (Deber de secreto) y 11 (Comunicación de datos).

La Ley Orgánica de Protección de Datos establece una serie de principios en los que se basan las obligaciones que los responsables de los ficheros deben cumplir. Con carácter general, será el responsable del fichero el titular de la clínica, ya sea una persona física o jurídica. Los responsables, para cumplir con la Ley, deben observar estos principios y cumplir las siguientes obligaciones.

- PRINCIPIO DE CALIDAD DE LOS DATOS
- PRINCIPIO DE INFORMACIÓN Y CONSENTIMIENTO.
- PRINCIPIO DE SEGURIDAD.
- PRINCIPIO DE CONFIDENCIALIDAD.
- PRINCIPIO DE COMUNICACIÓN DE DATOS
- FACILITAR LOS DERECHOS DE ACCESO, RECTIFICACIÓN, CANCELACIÓN Y OPOSICIÓN.
- INSCRIPCIÓN DE LOS FICHEROS EN LA AGENCIA ESPAÑOLA DE PROTECCIÓN DE DATOS.

Para llevar este proyecto a cabo usaré una metodología Agile. En concreto Scrum. Scrum (Info 4 Media SL.-Alexander Menzinsky) es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad**,** la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado**,** los costes se disparan **o** la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto**.**

En conclusión se pretende dar un mejor servicio al usuario ayudándole en la compleja labor de gestionar sus citas, cancelaciones y reprogramaciones, al mismo tiempo pretendemos ahorrar costos a la administración, optimizando los recursos que como indicábamos al inicio son cada vez más ajustados. Para lograr estos objetivos se innova dando uso a las nuevas tecnologías, todo ello en un marco de seguridad establecido por la Ley Orgánica de Protección de Datos.

Agradecimientos

Este proyecto de grado ha sido todo un reto lleno de altos y bajos en el proceso de investigación y creativo; de ahí que sea muy importante agradecer a todos aquellos que me ha ayudado a obtener el conocimiento y las herramientas para poder realizar tal empresa.

Agradecer a las personas desconocidas que han respondido a mis consultas en foros de temáticas relacionadas con el proyecto. A mi tutor por guiarme en este arduo trabajo. A los colegas de trabajo que me han ayudado a enfocar el proyecto. A mi novia que ha sido la persona que ha estado a mi lado durante todo este proyecto por su sapiencia y paciencia ;). Y por su puesto a mis padres por darme la oportunidad de llegar hasta aquí.

Índice

Índice de figuras

Índice de tablas

Capítulo 1- Introducción

En la pasada feria de tecnología de Texas se presentó todo tipo de dispositivos y droides que pretenden revolucionar lo que ahora se conoce en como tecnología y mostrarnos un nuevo camino a seguir. El objetivo en común de todos es “facilitarnos la vida”, hacer de la vida de los usuarios toda una experiencia gratificante.

Es por ello que consideré que mi mayor aportación con este proyecto de grado a la sociedad y a mi lugar de trabajo era crear una aplicación que brindara una solución práctica y económica para la sanidad.

Para conseguirlo he trabajado en la investigación de las necesidades actuales del hospital y los pacientes, las apps que existen en el campo sanitario y las herramientas disponibles para desarrollar una app propia para poder ayudar a resolver los problemas en común entre el hospital y los pacientes en la programación de citas, cancelación y reprogramación entre otros.

A continuación mostraré todo el proceso de investigación y creación de esta aplicación.

Capítulo 2 - Estado del arte

Visión Usuario

El ajetreado ritmo de vida del sistema actual, problemas personales, obligaciones laborales entre otras razones hace que muchos pacientes olviden sus citas programadas con meses de antelación; perderla les supone tener que esperar meses, lo que genera grandes molestias para el usuario, desmejora en su estado de salud y un importante aumento en gastos innecesarios para la sanidad pública.

Visión Hospital

Como consecuencia de un sin número de recortes por la crisis económica en España, la sanidad hoy se enfrenta al reto de administrar recursos muy limitados, poca contratación de personal, la externalización de algunos servicios, entre otros problemas, haciendo que las listas de espera para un médico especialista sean cada vez más largas (pueden estar entre 1 y 6 meses dependiendo de la provincia y/o municipio), lo cual sin duda desmejora la calidad del servicio y termina en aumentar los costos que se pretende reducir.

“El desaprovechamiento de citas médicas es una práctica extendida en España. Buena cuenta de ello dan los datos, que sitúan el porcentaje de incomparecencia injustificada de los pacientes entre el 5 y el 10 % dependiendo de la comunidad autónoma. En las Islas Baleares, la tasa se situó en el 8%.” (Echegaray, 2014)

Este TFG se contextualiza en el hospital de referencia de las Islas Baleares, el Hospital Universitario de Son Espases, pero sería aplicable a cualquier hospital que utilizara el mismo HIS; en concreto nuestro hospital utiliza el HIS de Millenium de Cerner, que es el Sistema de Información de Salud más implantado en el mundo.

La suite de soluciones Millenium es un sistema de gestión de pacientes, Historia Clínica Electrónica (HCE) y soluciones departamentales de vanguardia completamente integradas, capaz de realizar un seguimiento del paciente a través de todo el proceso asistencial. Millenium dispone de herramientas avanzadas que optimizan el flujo de trabajo de especialidades médicas: desde el momento en que un paciente es atendido en Urgencias o Consultas Externas hasta su paso por cualquier otra área del hospital, todos sus datos están siempre disponibles en tiempo real.

El sistema unificado Millenium permite automatizar completamente todo el circuito de medicación, minimizando los errores y aumentando la seguridad del paciente. Por otro lado, su capacidad multi-centro de hace posible que diferentes entes puedan operar de manera independiente, mientras comparten una misma base de datos y contribuyen a la creación de una historia clínica única y completa para los pacientes.

A la suite se le echa en falta aplicaciones para facilitar la vida de los pacientes, entre ellas una que sea capaz de recordar las citas a los pacientes. Para ello Cerner dispone de **MillenniumObjects**.

Visión tecnológica

MillenniumObjects

MillenniumObjects es un conjunto de herramientas basadas en Java, servicios Web que permite la lectura y escritura a Cerner Millennium. La lectura y escritura es segura y protegida en virtud de la seguridad integrada en las API proporcionada por MillenniumObjects.

MillenniumObjects ayuda a los clientes de Cerner crear soluciones personalizadas e integrar soluciones de terceros dentro de su entorno de Cerner Millennium.

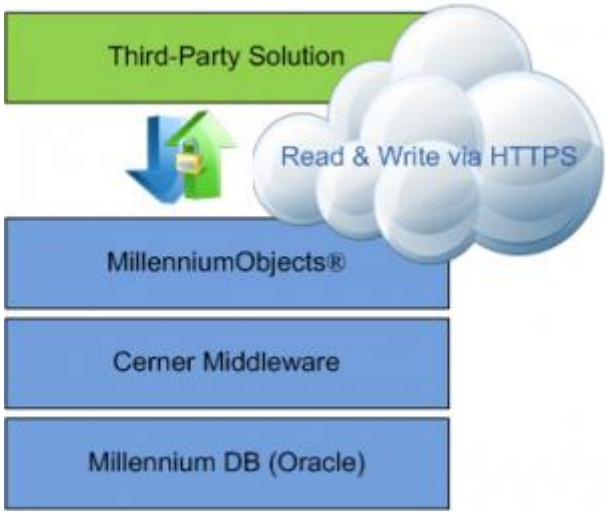


Ilustración 1: Funcionamiento MilleniumObjects (Cerner, mar 22, 2016)

Para profundizar en esta API, nos ayudaremos de la documentación proporcionada en https://www.ucern.com/ en el apartado de MillenniumObjects Reference Pages (Cerner, mar 22, 2016).

Soluciones ya existentes

Para llevar a cabo este proyecto se han analizado los Hospitales **más premiados por HIMSS**, la organización de referencia en Tecnologías de la Información aplicadas a la salud. The **Healthcare Information and Management Systems Society (HIMSS)** es una organización sin fines de lucro dedicada a mejorar la calidad del cuidado de la salud, la seguridad, la rentabilidad y el acceso, a través del mejor uso de la tecnología de la información y sistemas de gestión.

Se han estudiado las soluciones propuestas para la no presencia a las citas médicas entre los hospitales punteros en implantación tecnológica. Entre los que dispone de una App propia para la gestión de citas se encuentran:

Carolinas HealthCare System

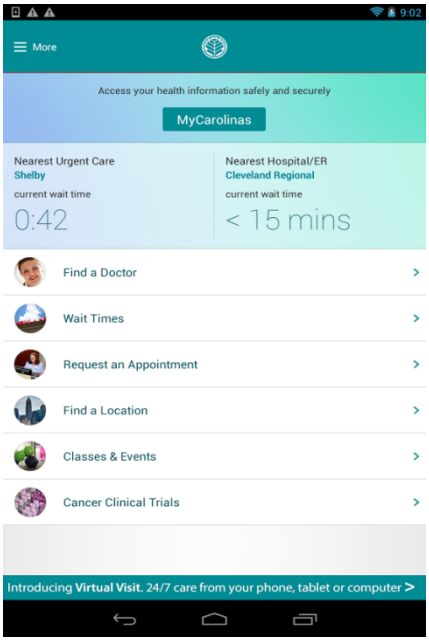


Ilustración 2: Carolinas HealthCare System

Funcionalidades:

- MyCarolinas - Programar las citas, enviar mensajes a su equipo de atención, echa de laboratorio y resultados de pruebas, leer las notas de su médico y más
- Encontrar una ubicación - Mapa hospitales, salas de emergencia (ER) o Urgente Cares más cercana a usted. Obtener direcciones de interior para centros hospitalarios seleccionados
- Tiempos de Espera - Ver ER actual y de atención médica urgente tiempos de espera
- Busque un médico - Búsqueda por nombre, especialidad, médicos de atención primaria o utilizando su posición actual o código postal
- Solicitar una Cita - Envíenos una solicitud de cita para una fecha, hora y lugar que funcione para usted
- Clases y Eventos - Buscar clases y eventos ofrecidos por CHS, y añadirlos a su calendario

- Estudios clínicos de cáncer - Buscar ensayos clínicos de cáncer, y en contacto con el coordinador del estudio para obtener información adicional
- Cómo llegar - Ver mapas e información de estacionamiento
- Direcciones del servicio, direcciones, información de estacionamiento y de click-to-call números de teléfono
- Guardar ubicaciones favoritas para un acceso rápido
- Compartir información médico y la ubicación

MyAdvocate le pone en control

La App MyAdvocate le da la libertad al paciente de administrar su atención médica siempre que lo desee, desde donde quiera que esté.

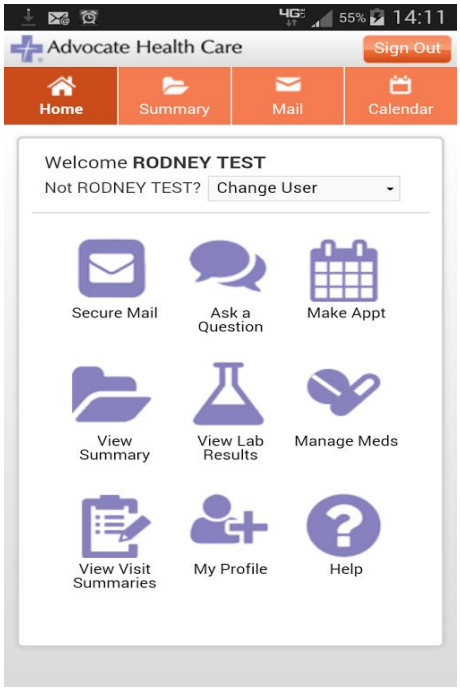


Ilustración 3: MyAdvocate

Le permite al usuario:

• Programar las citas
• Llenar sus medicamentos
• Ver su historial médico
• Compruebe sus pruebas
• Hablar con los médicos
• Sincronización con aplicaciones de bienestar como Fitbit

MyChart

MyChart le da acceso al paciente a los resultados de sus pruebas, información de las citas, medicamentos actuales, historial de vacunación, y más en su dispositivo móvil.

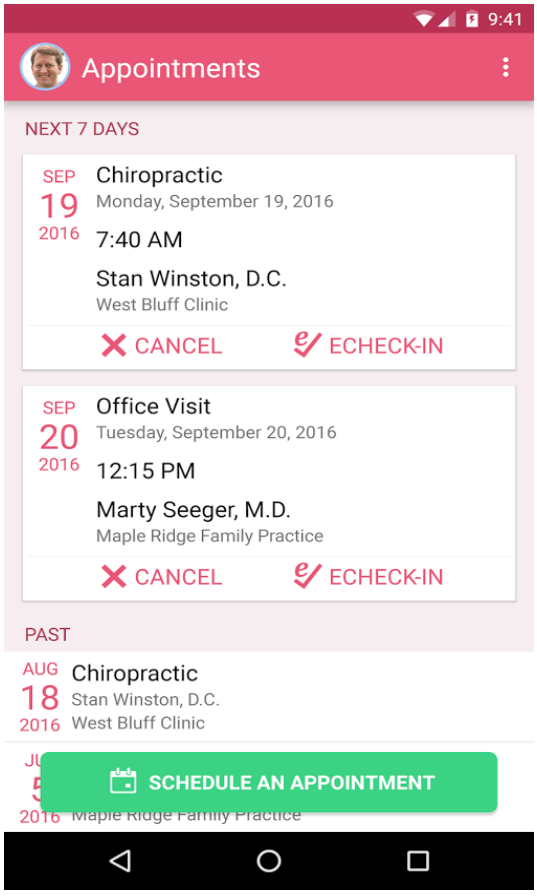


Ilustración 4: MyChart

Con MyChart, usted puede:

- Ver su información médica
- Mantenerse en contacto con su personal médico
- Administrar sus citas
- Acceder a la información médica de su familia

Por otro lado también se ha analizado la solución que nos da Sanitas con Su App

Sanitas App

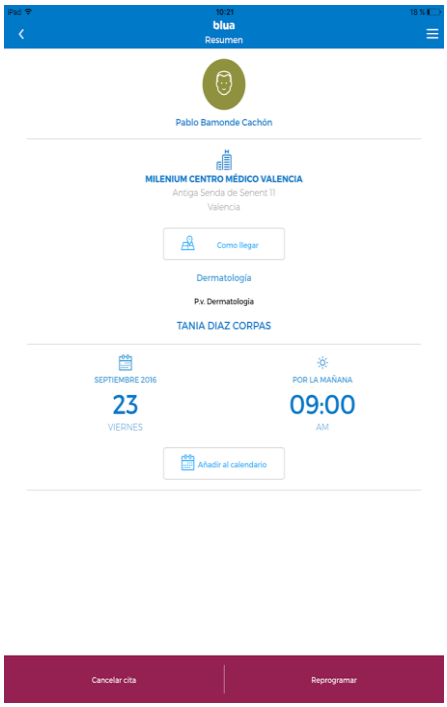


Ilustración 5: Sanitas App

Con la nueva app de Sanitas se puede:

- Pedir citas de forma rápida y sencilla.
- Encontrar información sobre médicos, hospitales, urgencias, clínicas dentales, centros médicos y de bienestar, por especialidad y cercanía
- Disfrutar de la comodidad de la Tarjeta Digital

- Ver tus recibos y copagos en tiempo real
- Gestionar tu póliza
- Consultar, modificar y personalizar el perfil de cada beneficiario
- Cambiar la cuenta bancaria
- Consultar tu histórico de visitas al médico
- Acceder a tu médico personal
- Disfrutar de las ventajas de Blua, con la que tendrás acceso a vídeo consulta, farmacia y análisis a domicilio y mucho más
- Enviar tus preguntas al médico
- Acceder a tus informes médicos
- Consultar la biblioteca de consejos de salud
- Gestionar reembolsos y autorizaciones

Finalmente revisamos la Aplicación más usada en Latinoamérica Doctoralia.

Doctoralia

Esta aplicación se encarga de buscar profesionales y centros médicos cerca de los usuarios, revisa opiniones de otros usuarios, y contacta con ellos por teléfono o mediante cita online desde la propia aplicación.

Ilustración 6: Doctoralia

Funcionalidades:

- Buscar por mapa: Encuentra doctores, profesionales de la salud y centros sanitarios cerca de ti o en cualquier zona, con la opción de filtrar por especialidad, compañía aseguradora, o por disponibilidad de Cita Online Doctoralia.
- Búsqueda de texto: Para encontrar cualquier profesional o centro, o buscar tu médico rápidamente.
- Favoritos: Añade tu doctor o centros y tendrás acceso a sus datos de forma rápida
- Cita Online: La cita online del líder en Internet, con recordatorio y el histórico de citas.

La versión móvil de Doctoralia permite buscar y reservar cita en España, México, Argentina, Chile, Colombia, Perú y 14 países más. Además si viajas a alguno de estos países, podrás encontrar un profesional buscando la especialidad en español.

Herramientas disponibles

Con el objetivo de crear una aplicación propia fácil de usar y disponible para la mayoría de usuarios con smartphones hemos optado por el sistema operativo Android, de ahí que sea necesario ahondar en ella.

Android

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. El sistema permite programar aplicaciones en una variación de Java llamada Dalvik. El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

Breve Historia Android

En Julio de 2005, *Google* adquirió *Android, Inc*, una pequeña *startup* de California. En esos momentos, la compañía se dedicaba a la creación de *software* para teléfonos móviles. Una vez en Google, el equipo desarrolló un S.O. basado en *Linux* para dispositivos móviles. Más adelante, Google adaptó su buscador y sus aplicaciones para su uso en móviles.

En septiembre del 2007, *Google* tenía varias patentes de aplicaciones sobre el área de la telefonía móvil. El 5 de noviembre del mismo año, se anunció la fundación de la *Open Handset Alliance* al mismo tiempo que la creación de la plataforma *Android.* La *Open Handset Alliance* está formada por un consorcio de 34 compañías de hardware, software y telecomunicaciones, entre las cuales

se incluyen *Google, HTC*, *Intel* y *Motorola* entre otras, dedicadas a investigar estándares abiertos para dispositivos móviles.

El primer teléfono en el mercado que posee *Android* es el *T-Mobile G1* (también conocido como *Dream)*, lanzado el día 22 de octubre de 2008 que viene con la versión *Android* 1.0 preinstalada. Este móvil es el resultado conjunto de *T-Mobile, HTC* y *Google.* Por último, desde el 21 de octubre de 2008, *Android* está disponible como código abierto. Gracias a esto, cualquiera puede añadir extensiones, nuevas aplicaciones o reemplazar las existentes por otras dentro del dispositivo móvil.

Versiones

Desde abril de 2009, las versiones de Android han sido desarrolladas bajo un nombre en clave y sus nombres siguen un orden alfabético: Cupcake, Donut, Éclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow y el nuevo Nougat lanzado en agosto de 2016.

Nos interesa saber que versión se utiliza mayormente en los dispositivos Android, para realizar la aplicación lo más compatible posible. En un estudio realizado el pasado mes de Septiembre de 2016 se concluyó que la versión Lollipop es la más usada con un 35% de la cuota total.

(https://developer.android.com/)

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.4%
4.1.x	Jelly Bean	16	5.6%
4.2.x		17	7.7%
4.3		18	2.3%
4.4	KitKat	19	27.7%
5.0	Lollipop	21	13.1%
5.1		22	21.9%
6.0	Marshmallow	23	18.7%

Data collected during a 7-day period ending on September 5, 2016.
Any versions with less than 0.1% distribution are not shown.

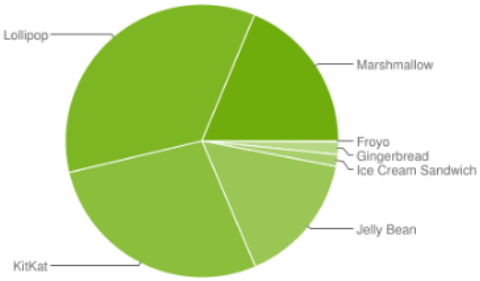


Ilustración 7: Distribución de Android Actualizado para septiembre de 2016 (http://www.droid-life.com)

Arquitectura de Android

Los componentes principales del sistema operativo de Android (cada sección se describe en detalle):

- **Aplicaciones**: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.
- **Marco de trabajo de aplicaciones**: los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.
- **Bibliotecas**: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los

desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.

- **Runtime de Android**: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecutaba hasta la versión 5.0 archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida "dx". Desde la versión 5.0 utiliza el ART, que compila totalmente al momento de instalación de la aplicación.
- **Núcleo Linux**: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

Anatomía de una aplicación de Android (W. Fran Ableson, 2011, Pags. 32 – 55)

Dentro de una aplicación de *Android* hay cuatro componentes principales: *Activities, Listeners, Services y Content Providers*. Todas las aplicaciones de *Android* están formadas por algunos de estos elementos o combinaciones de ellos.

Activity

Las *Activities* (o Actividades) son el elemento constituyente de *Android* más común. Para implementarlas se utiliza una clase por cada Actividad que extiende de la clase base *Activity*. Cada clase mostrará una interfaz de usuario, compuesta por *Views* (o Vistas). Cada vez que se cambie de Vista, se

cambiará de Actividad, como por ejemplo en una aplicación de mensajería que se tiene una Vista que muestra la lista de contactos y otra Vista para escribir los mensajes. Cuando cambiamos de Vista, la anterior queda pausada y puesta dentro de una pila de historial para poder retornar en caso necesario. También se pueden eliminar las Vistas del historial en caso de que no se necesiten más. Para pasar de vista en vista, *Android* utiliza una clase especial llamada *Intent*.

Un ***Intent*** es un objeto mensaje y que, en general, describe qué quiere hacer una aplicación. Las dos partes más importantes de un *Intent* son la acción que se quiere realizar y la información necesaria que se proporciona para poder realizarla, la cual se expresa en formato *URI*. Un ejemplo sería ver la información de contacto de una persona, la cual mediante un *Intent* con la acción *ver* y la URI que representa a esa persona se podría obtener. Relacionado con los *Intents*, hay una clase llamada *IntentFilter* que es una descripción de qué *Intents* puede una gestionar un *Activity*. Mediante los *IntentFilters*, el sistema puede resolver *Intents*, buscando cuáles posee cada actividad y escogiendo aquel que mejor se ajuste a sus necesidades. El proceso de resolver *Intents* se realiza en tiempo real, lo cual ofrece dos beneficios:

- Las actividades pueden reutilizar funcionalidades de otros componentes simplemente haciendo peticiones mediante un *Intent.*
- Las actividades pueden ser remplazadas por nuevas actividades con *IntentFilters* equivalentes.

Listeners

Los *Listeneres* se utilizan para reaccionar a eventos externos (por ejemplo, una llamada). Los *Listeners* no tienen *UI* (*User Interface*), pero pueden utilizar el servicio *NotificationManager* para avisar al usuario. Para lanzar un aviso no

hace falta que la aplicación se esté ejecutando, en caso necesario, *Android* la iniciará si se activa el *Listeners* por algún evento.

Services

Un *Service*, o Servicio, es básicamente un código que se ejecuta durante largo tiempo y sin necesidad de *UI*, como puede ser un gestor de descarga, en el cual se indican los contenidos a descargar y posteriormente el usuario puede acceder a una nueva vista sin que el gestor se interrumpa. En caso de que haya múltiples servicios a la vez, se les puede indicar diferentes prioridades según las necesidades.

Content Providers

En *Android,* las aplicaciones pueden guardar su información en ficheros, BBDD *SQLite*, etc., pero en caso de que lo que se quiera sea compartir dicha información con otras aplicaciones, lo necesario es un *Content Provider*. Un *Content Provider* es una clase que implementa un conjunto estándar de métodos que permite a otras aplicaciones guardar y obtener la información que maneja dicho *Content Provider.*

Android Manifest

En *Android* existe un archivo *XML* llamado *AndroidManifest* que, aunque no forme parte del código principal de la aplicación, es necesario para su correcto funcionamiento. Este archivo es el fichero de control que le dice al sistema qué tiene que hacer con todos los componentes anteriormente mencionados en este apartado que pertenecen a una aplicación en concreto.

Ciclo de vida de una aplicación de Android (W.Frank Ableson, 2011)

Cada aplicación de *Android* corre en su propio proceso, el cual es creado por la aplicación cuando se ejecuta, y permanece hasta que la aplicación deja de trabajar o el sistema necesita memoria para otras aplicaciones. Una característica fundamental de *Android* es que el ciclo de vida de una aplicación no está controlado por la misma aplicación sino que lo determina el sistema a partir de una combinación de estados como pueden ser qué aplicaciones están funcionando, qué prioridad tienen para el usuario y cuánta memoria queda disponible en el sistema. De esta manera, *Android* sitúa cada proceso en una jerarquía de "importancia" basada en los estados comentados, como se puede ver a continuación.

Un proceso en primer plano es uno que se requiere para lo que el usuario está actualmente haciendo. Se considera en primer plano si:

- Está ejecutándose una Actividad perteneciente en la pantalla con la que el usuario está interactuando.
- Está ejecutando un *BroadcastReceiver*.
- Está ejecutándose un servicio.

Un proceso visible es aquel que contiene una Actividad que es visible al usuario mediante la pantalla, pero no en primer plano (está pausada). Este proceso solo se eliminará en caso de que sea necesario para mantener ejecutándose los procesos en primer plano.

Un proceso de servicio es aquel que contiene un servicio que ha sido inicializado. No son directamente visibles al usuario y el sistema los mantendrá a no ser que no pueda servir los dos anteriores.

Un proceso en *background* es aquel que acoge una actividad que no está actualmente visible al usuario. Mientras que dichos procesos implementen bien su propio ciclo de vida, el sistema puede eliminarlos para dar memoria a cualquiera de los 3 procesos anteriores.

Un proceso vacío es aquel que no contiene ningún componente activo de ninguna aplicación. La única razón para mantener dicho proceso es para mejorar sus inicializaciones posteriores a modo de caché.

Para comprender mejor el ciclo de vida de una aplicación de *Android,* en la siguiente figura se muestra el diagrama de flujo de dicho ciclo, mencionando también los métodos que se llaman durante el transcurso del mismo.

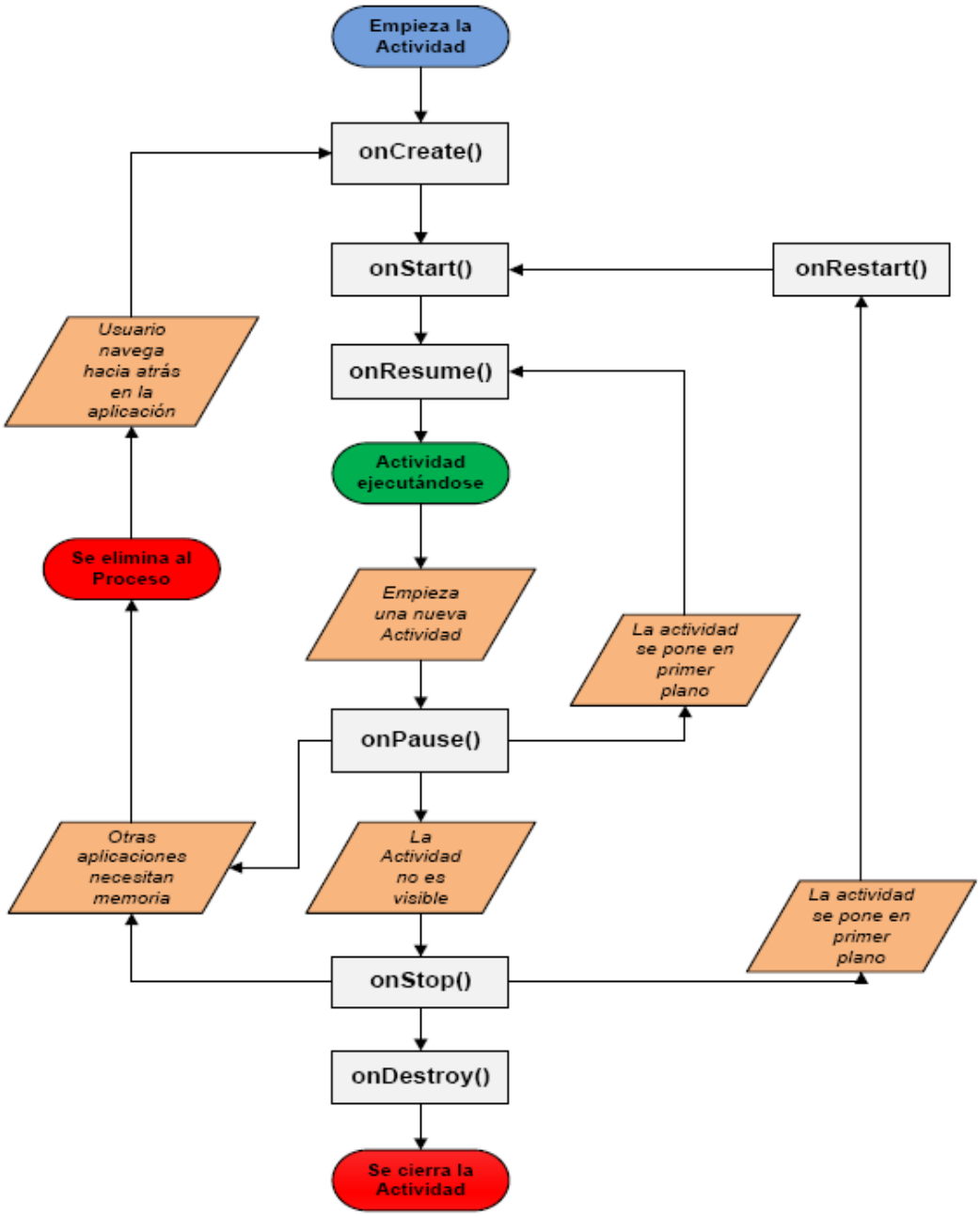


Ilustración 9: Ciclo de Vida Apps Android (W.Frank Ableson, 2011)

Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Fue anunciado el 16 de mayo 2013 en la conferencia de Google I/, y reemplazó a Eclipse el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android, como las siguientes:

- Sistema de compilación flexible basado en Gradle.
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Instant Run, para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK.
- Integración de plantillas de código y GitHub, para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código.
- Gran cantidad de herramientas y frameworks de prueba.
- Herramientas Lint para detectar problemas de rendimiento, uso, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK
- Soporte integrado para Google Cloud Platform, que facilita la integración de Google Cloud Messaging y App Engine.

JSON

JSON se corresponde con las siglas *JavaScript Object Notation* y es un formato ligero para el intercambio de datos entre distintas partes. Es un lenguaje muy sencillo de leer y escribir por las personas, y fácil de ser analizado sintácticamente y ser generado por máquinas. Aunque es un subconjunto de la notación de JavaScript, JSON es un formato de texto completamente independiente de JavaScript o cualquier otro lenguaje de programación.

Una de las ventajas de JSON sobre XML como formato de intercambio de datos es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función `eval()`, lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

En la práctica, los argumentos a favor de la facilidad de desarrollo de analizadores o del rendimiento de los mismos son poco relevantes, debido a las cuestiones de seguridad que plantea el uso de `eval()` y el auge del procesamiento nativo de XML incorporado en los navegadores modernos. Por esa razón, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia.

REST WEB SERVICES (W.Frank Ableson, 2011)

La Transferencia de Estado Representacional (Representational State Transfer) o *REST* es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Si bien el término *REST* se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz web simple que utiliza *XML* y *HTTP*, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web *SOAP*. Es posible diseñar sistemas de servicios web de acuerdo con el estilo arquitectural REST de Fielding y también es posible diseñar interfaces *XMLHTTP* de acuerdo con el estilo de llamada a procedimiento remoto pero sin usar SOAP. Estos dos usos diferentes del término *REST* causan cierta confusión en las discusiones técnicas, aunque RPC no es un ejemplo de REST.

Los sistemas que siguen los principios REST se llaman con frecuencia *RESTful*; los defensores más acérrimos de REST se llaman a sí mismos *RESTafaris*. REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

Un protocolo cliente/servidor sin estado: cada mensaje *HTTP* contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en *HTTP* utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de *URLs*, no son permitidas por REST).

Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: *HTTP* en sí define un conjunto pequeño de operaciones, las más importantes son *POST, GET, PUT* y *DELETE*. Con frecuencia estas operaciones se equiparan a las operaciones CRUD que se requieren para la persistencia de datos, aunque *POST* no encaja exactamente en este esquema.

Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su *URI*.

El uso de hipermedias, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente *HTML* o *XML*. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

Justificación Tecnológica

1) Android

Alternativas: iOS, Windows Phone, Symbian, Bada.

Se ha optado por utilizar Android debido a la gran aceptación de este sistema en los smartphones de todo el mundo, siendo a día de hoy, el sistema operativo para móviles de mayor crecimiento, además de su cada vez mayor utilización en el mundo de los tablets y otros dispositivos.

Además, otro motivo suficientemente importante como para emplear Android ha sido el conocimiento de desarrollo de aplicaciones en Android y no en otras plataformas como iOS.

2) Android Studio

Alternativas: Eclipse

Se optado por entorno de desarrollo integrado (IDE) por ser el oficial para el desarrollo de aplicaciones para Android y por tener conocimientos sobre su uso.

3) JSON

Alternativas: XML.

Se ha optado por la utilización de JSON debido a que MillenniumObjects utiliza esta tecnología.

Por otro lado, con JSON la información transmitida se reduce considerablemente. Esto es un punto a su favor debido a que se van a utilizar smartphones como plataforma de uso final, y cuanto menor volumen de datos se recuperen desde el servidor, más rápida será la respuesta de la aplicación y más satisfactoria la experiencia para el usuario.

4) REST

Alternativas: SOAP.
Se ha decidido utilizar los servicios Web REST debido a la facilidad de desarrollo y sobre todo, la simplicidad en el acceso a los recursos por este sistema.

Propuesta de Valor

Una vez analizada la problemática que presenta el hospital y los pacientes con la programación, cancelación y reprogramación de citas médicas entre otros servicios, también hemos buscado las apps que existen en la actualidad y hemos revisado las herramientas con las que contamos para la creación de

una nueva aplicación que integre las soluciones necesarias tanto para usuarios como para el hospital.

Mi propuesta es la creación de una app capaz de mostrar al paciente en tiempo real las citas médicas que tiene programadas, que la app envíe avisos a diferentes intervalos recordándole las citas y le permita confirmar o cancelar con anticipación estas, permitiéndole también reprogramar las citas sin tener que desplazarse al hospital o esperar un sin número de minutos al teléfono.

Con esta app se pretende contribuir al sistema sanitario Balear con una herramienta que le permita optimizar el servicio que presta en la programación de citas y bajar los costes que invierte; además del valor agregado que supone el hecho que una app estará disponible las 24 horas al día, los 365 días al año.

Capítulo 3 - Propósito general de la app

Esta aplicación tiene como propósito principal que “El Paciente deberá ser capaz de consultar sus citas en menos de 2 segundos y tener la posibilidad de anularlas”.

El Contexto de AppPinkCard

Se plantea este TFG como la entrega del Primer prototipo viable. Para ello se entregará una App capaz de visualizar datos en formato JSON desde un WebService REST.

La proyección futura:

- Es que la aplicación sea capaz de generar avisos.
- Posibilitará el registro de paciente.
- Los Pacientes podrán modificar sus datos.

Capítulo 4 - Recolectar requerimientos

- Consultar las citas futuras.
- Ver detalle de la cita.
- El login para usuarios.
- Cancelar Cita.

En el estado actual, App PinkTarget solo tiene la entidad **Cita**.
Con las siguientes propiedades:

- Código
- Paciente
- Doctor
- Servicio
- Fecha y Hora

Determinar quién usará la app

Pacientes de Consultas externes del Hospital Son Espases.

Capítulo 5 - Wireframe de pantallas

Screen de Login

Ilustración 8: Screen de Login (Cantallops Cifre; Francesc)

¿Qué puntos de interacción tenemos?

A. Dos campos de texto (no. de documento identidad y contraseña)
B. El botón de iniciar sesión

Ahora, ¿qué reacciones de UI se producen?

A. La edición de texto con mal formato muestra un error
B. Si los datos son correctos y hay conexión, se navega a la screen de citas médicas. De lo contrario se muestra un error.

Screen Citas Médicas

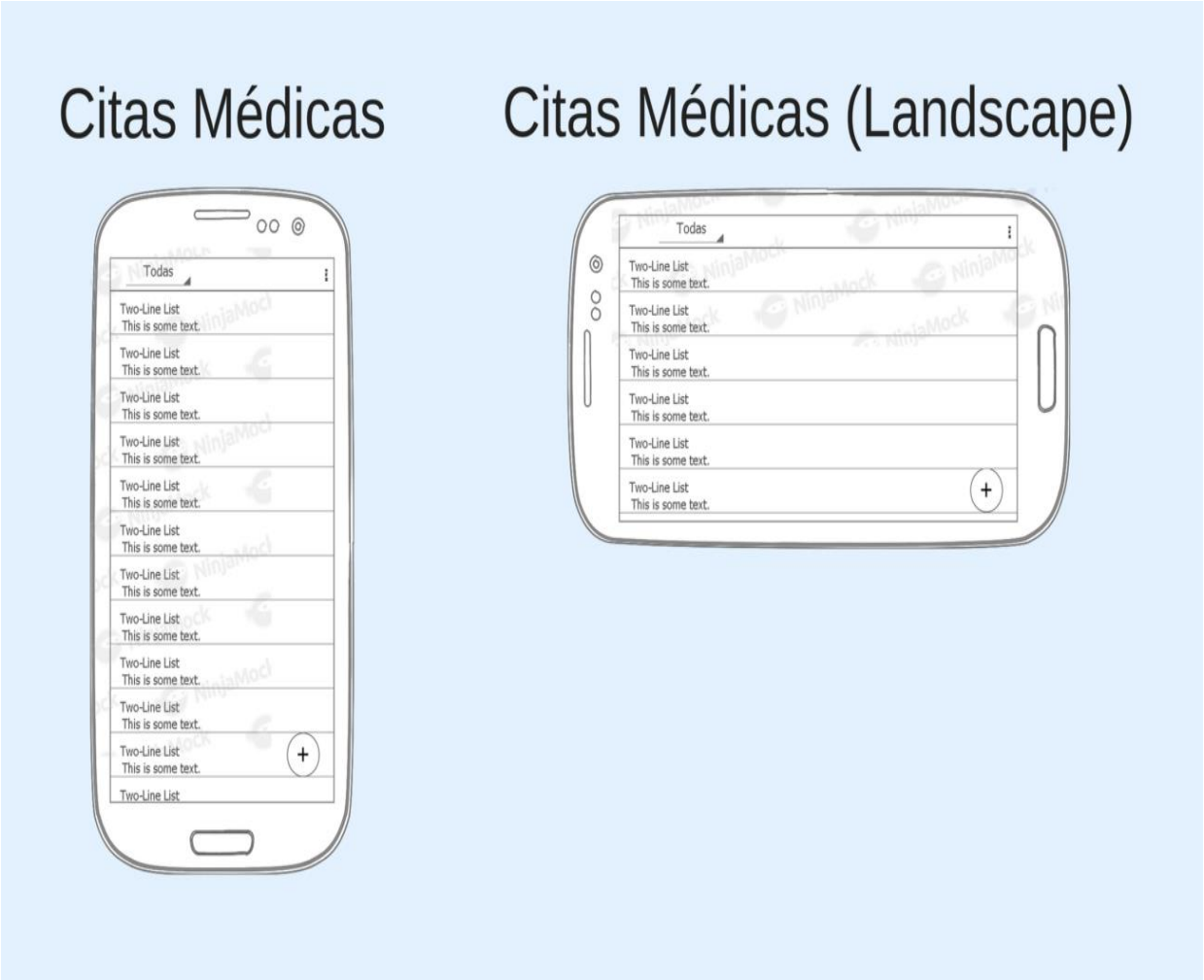


Ilustración 9: Screen Citas Médicas (Cantallops Cifre; Francesc)

De allí tendremos varios puntos de interacción.

A. Spinner en la Toolbar para filtrar
B. Menú con desborde (tres puntos verticales) para mostrar la opción de Radicación de PQRS
C. Lista con las citas médicas del usuario diseñadas en cards
D. Floating Action Button para solicitar citas

El diseño de los ítems de lista vendría así

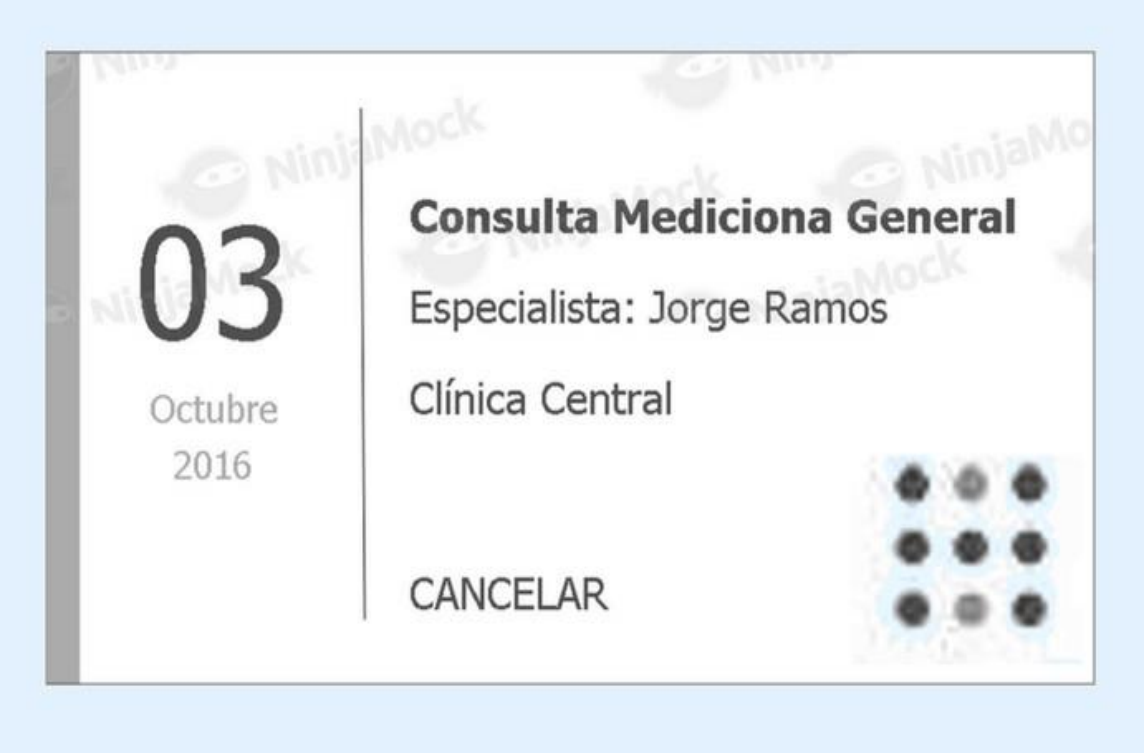


Ilustración 10: Ítems de lista Citas médicas (Cantallops Cifre; Francesc)

Donde un botón de cancelar sobre las citas médicas activas.

Además tendremos los siguientes estados de UI:

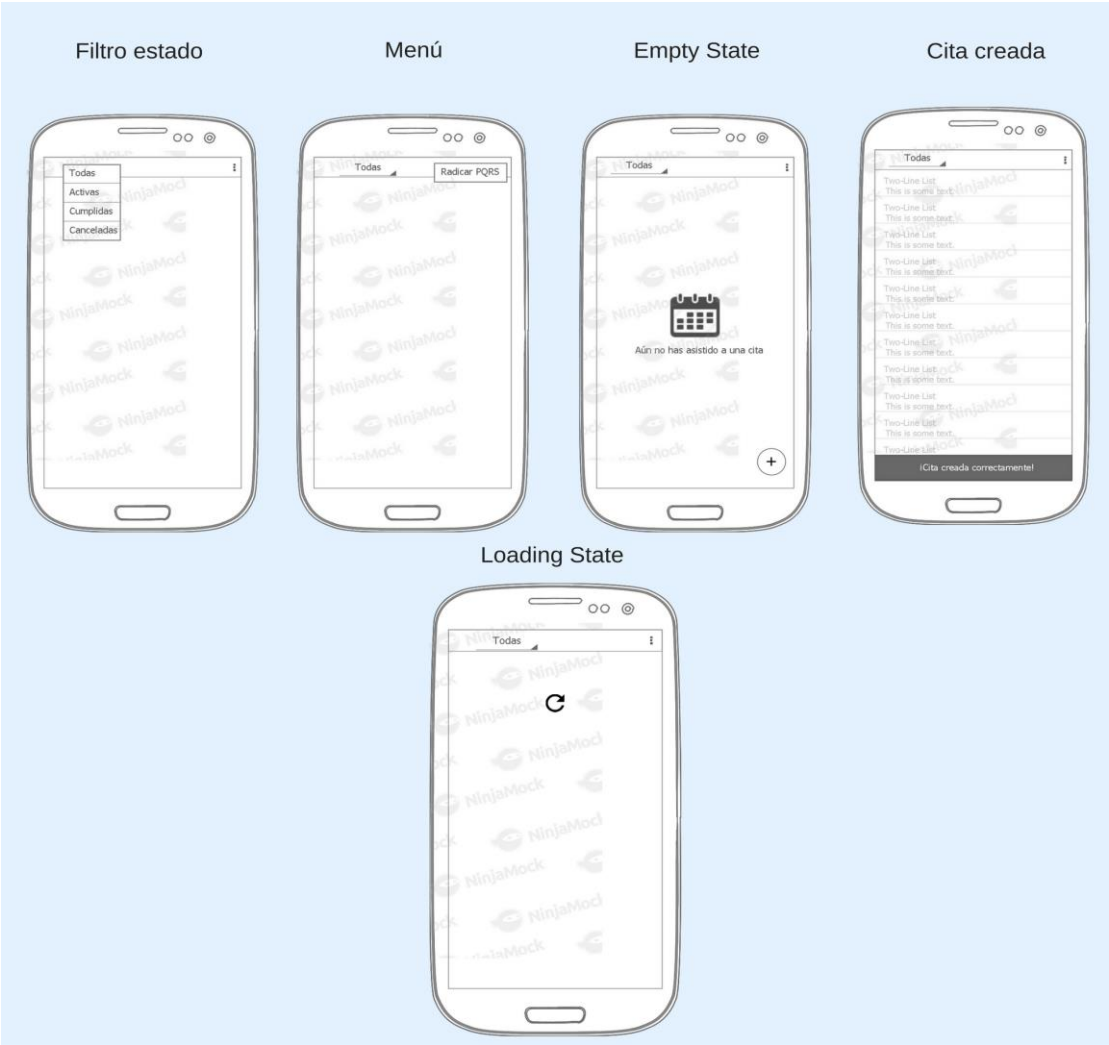


Ilustración 11: estados de UI (Cantallops Cifre; Francesc)

La interacción del spinner en la action bar nos muestra las opciones de filtro. Como te decía, el menú mostrará la opción de los radicados.

Habrá un empty state en caso de no existir citas aún. Además mostraremos un Toast cuando se asigne una nueva cita.

Y el estado de carga de datos.

Capítulo 6 - Determinar fuentes de datos y sincronización

1. En el primer entregable de la app PinkCard tendremos una base de datos remota (servidor) para persistencia. También usaremos la memoria del dispositivo como caché al mostrar elementos.

2. Habrá sincronización de datos

Las entidades se comportarán de esta forma:

- **Citas**: predominan los cambios del servidor, ya que desde el cliente Android no se modificarán.

- Además:
    - El usuario sincronizará manualmente la app con el gesto “Swipe to refresh”

    - Se enviarán modificaciones inmediatamente la persistencia local cambie (futuro)

    - Se notificarán cambios del servidor con notificaciones push (futuro)

    - Si no hay conexión disponible para sincronizar, se programará una actualización a penas se detecte el restablecimiento de la red (futuro).

Capítulo 7 - Herramientas Y Recursos

La siguiente es una lista de los recursos a usar:

- **Hosting**: Localhost. Haremos pruebas locales que nos faciliten la implementación del servicio web.
- **Lenguaje**: PHP para la API y Java para Android.
- **Base de datos remota**: MySQL
- **Formato de intercambio de datos**: JSON
- **Carga asíncrona de imágenes**: Librería Glide
- **Tarjetas y listas**: CardView y RecyclerView
- **Peticiones HTTP**: Retrofit

Habiendo estudiado todo lo anterior, ya tenemos una base clara sobre lo que trabajaremos.

Seleccionar recursos y herramientas

Comencemos por el lado del cliente:

- Manejo de colecciones y precondiciones > Librería Guava
- Carga eficiente y caché en disco de imágenes > Librería Glide
- Base de datos local (futuro) > SQLiteOpenHelper + ContentProvider
- Ejecutor de peticiones HTTP hacia la API: Retrofit

Ahora, el lado del servidor:

- Lenguaje: PHP 5.6
- Gestor de bases de datos: MySQL
- Proveedor del servicio de hosting: WebService de SMART Health
- Formato de intercambio: JSON

Capítulo 8 - Desarrollar versión “cascaron” de la app

Para desarrollar la aplicación usaremos la arquitectura **Clean**. Esta arquitectura nos aporta:

- Permite separar conceptos.
- Aumenta la hermeticidad a la hora de usar tests.
- Reduce la dependencia de entes externos (bases de datos, APIs, sensores, etc)
- La UI se somete solo a mostrar datos (nada de “God Activities”)

Clean architecture se divide en tres capas:

- **Presentación**: Aquí defines todo lo relacionado a la vista y las animaciones. En nuestro caso aplicaremos **Modelo-Vista-Presentador** para cubrir dicha capa. Sin embargo puedes usar MVC o MVVM.

- **Dominio**: Aquí van las reglas del negocio definidas por casos de uso (interactores) y las entidades básicas (objetos planos java).

- **Datos**: Esta capa contiene los datos de la cual se alimentarán la capa de presentación. Existen varios patrones que puedes usar para manejar los datos. En particular, usaremos uno llamado Repositorio para generalizar la toma desde múltiples fuentes de datos (SQLite, Realm, Memoria, Servidor, archivos, etc.).

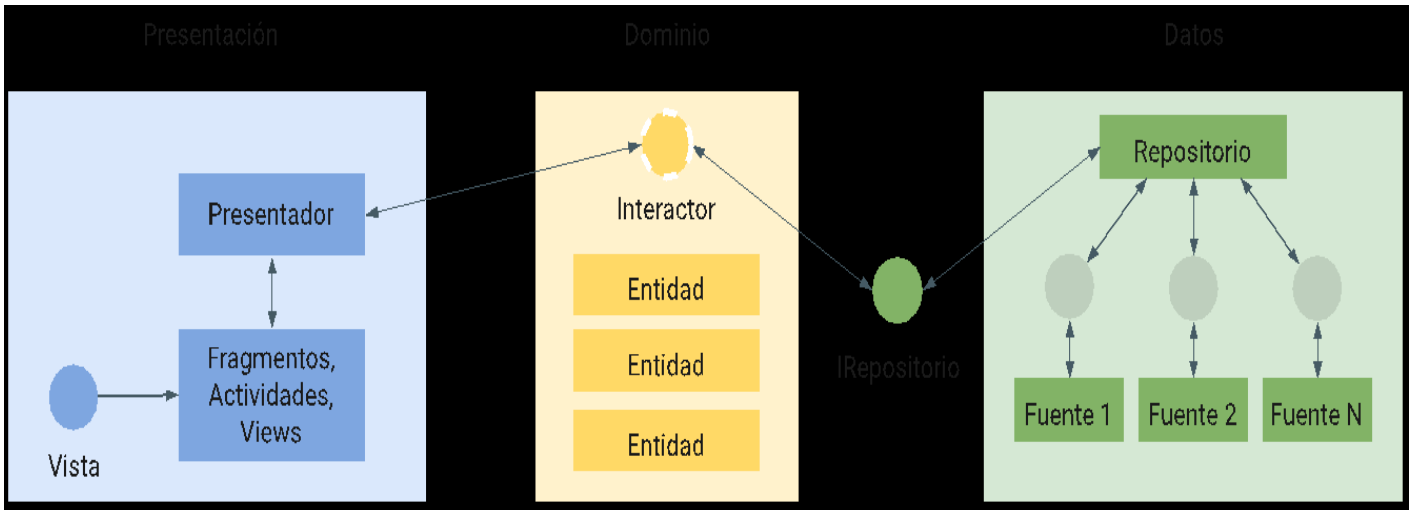


Ilustración 12: Modelo-Vista-Presentador (Cantallops Cifre; Francesc)

Por otro lado utilizaremos la metodología ágil Scrum para la Gestión del Proyecto.

Para llevar a cabo este proyecto nos ayudaremos de Tablas Kahban. En concreto la herramienta web Trello con la que podemos representar tableros y tareas. También nos valdremos de Excels para medir las tareas y su progreso.

Empezaremos por definir un Product Backlog. Que no es más que una pila de las historias clínicas de usuarios ordenadas por preferencia.

Artefactos

Product Backlog.

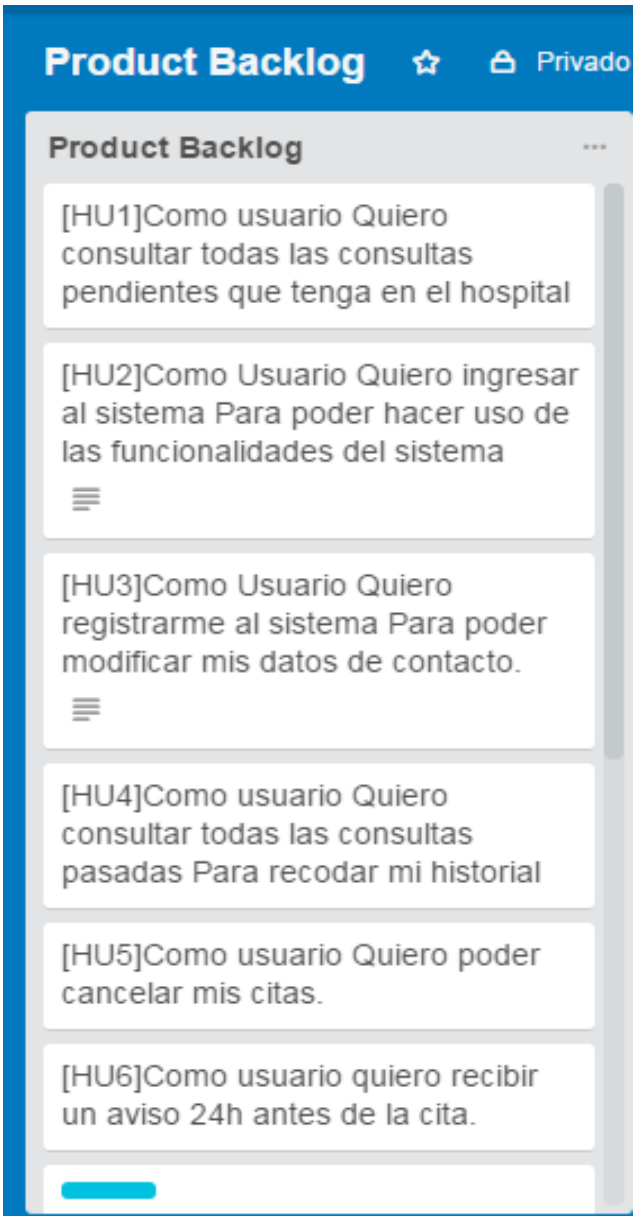


Tabla 1: Product Backlog (Cantallops Cifre; Francesc)

Nos centraremos en la primera Historia de usuario que la realizaremos en un Spring de 40 horas para poder presentar un Producto Mínimo Viable

"[HU1]Como usuario Quiero consultar todas las consultas pendientes que tenga en el hospital"

Tareas [HU1]Consulta citas médicas

1. Crear actividad de citas (clase Java + layout)
2. Crear fragmento de citas (clase Java + layout)
    a. Crear entidad Cita
    b. Crear adaptador de Citas
    c. Crear layout para ítems de Citas
3. Preparar fragmento
4. Crear contrato MVP
5. Implementar “vista” de Citas
6. Implementar “presentador” de Citas
7. Crear repositorio de Citas
8. Crear fuentes de datos
9. Terminar presentador de Citas
10. Establecer dependencias vista-presentador
11. Proveer Endless scroll

Sprint Backlog

El proyecto se ha planificado en un Sprint de 57h distribuïdo en 20 días y llevado acabo por un único desarrollador.

Nuestro Spring Backlog detalla las tareas y el peso de cada una. Ha usado una hoja de Excel para gestinarlo.

Sprint Backlog (Sprint1)

Identificador (ID) de ítem de product backlog	Enunciado del ítem de Product Backlog	Tarea	Dueño / Voluntario	Estatus	Horas estimadas totales
XX-XXXX-XXXX	Como un paciente, necesito consultar la citas pendientes, con la finalidad de poderme acordar de ella.	[Tarea 1]Crear actividad de productos (clase Java + layout)	Xisco		4
		Tarea #2. Crear fragmento de productos	Xisco		4
		[Tarea 2.1]Crear entidad Producto	Xisco		4
		[Tarea 2-2]Crear adaptador de productos	Xisco		4
		[Tarea 3] Preparar fragmento	Xisco		4
		[Tarea 4]Crear contrato MVP	Xisco		4
		[Tarea 5]Implementar "vista" de Citas	Xisco		4
		[Tarea 6]Implementar "presentador" de Citas	Xisco		4
		[Tarea 7]Crear repositorio de Citas	Xisco		4
		[Tarea 8]Crear fuentes de datos	Xisco		6
		[Tarea 9]Terminar presentador de Citas	Xisco		8
		[Tarea 10]Establecer dependencias vista-presentador	Xisco		4
		[Tarea 11]Proveer Endless scroll	Xisco		3
					57

Tabla 2: Sprint Backlog (Cantallops Cifre; Francesc)

Burndown charts

Mediante otra plantilla de Excel podemos generar un gráfico (Burndown chart) donde se representan las tareas quemadas y visualizando el desfase entre tareas realizadas y pendientes.

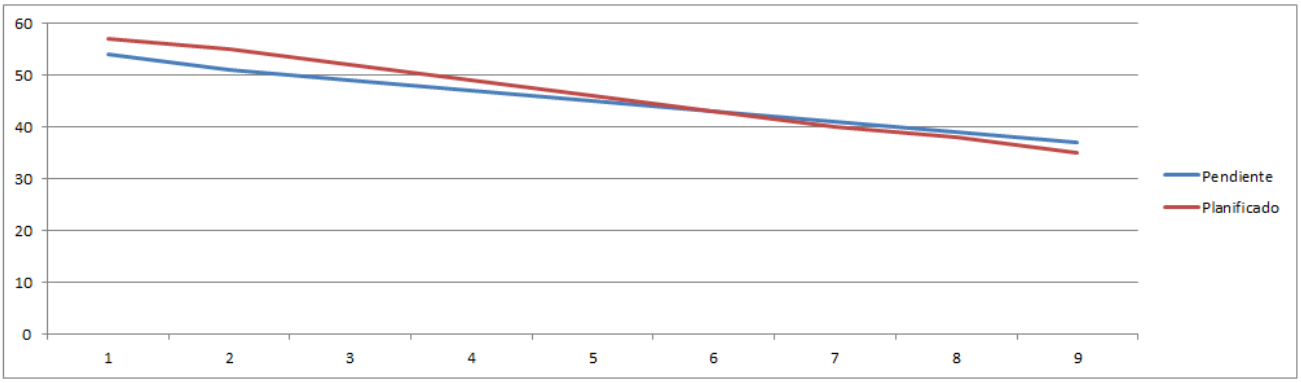


Ilustración 13: Burndown charts (Cantallops Cifre; Francesc)

Tabla Kanban Inicial Spring 1

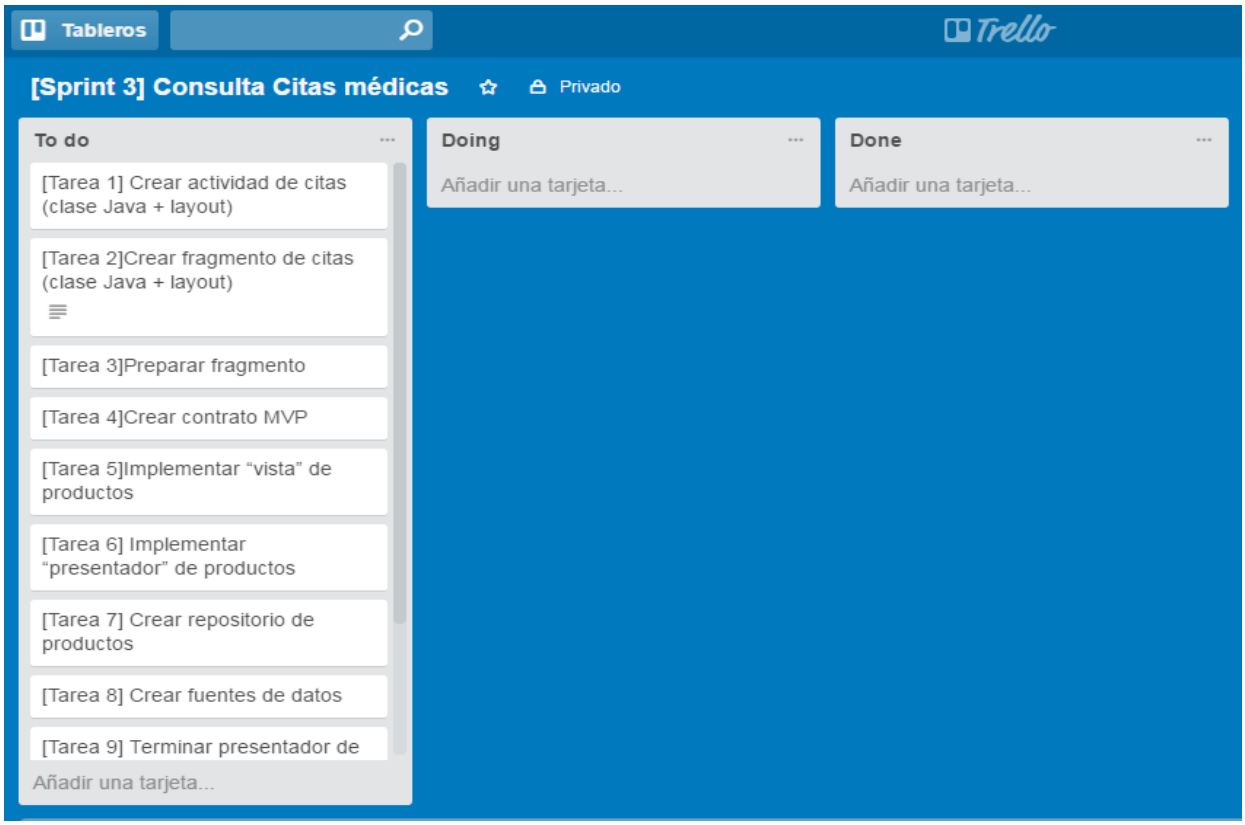


Tabla 3: Tabla Kanban Inicial Spring 1 (Cantallops Cifre; Francesc)

Tabla Kanban Final Spring 1

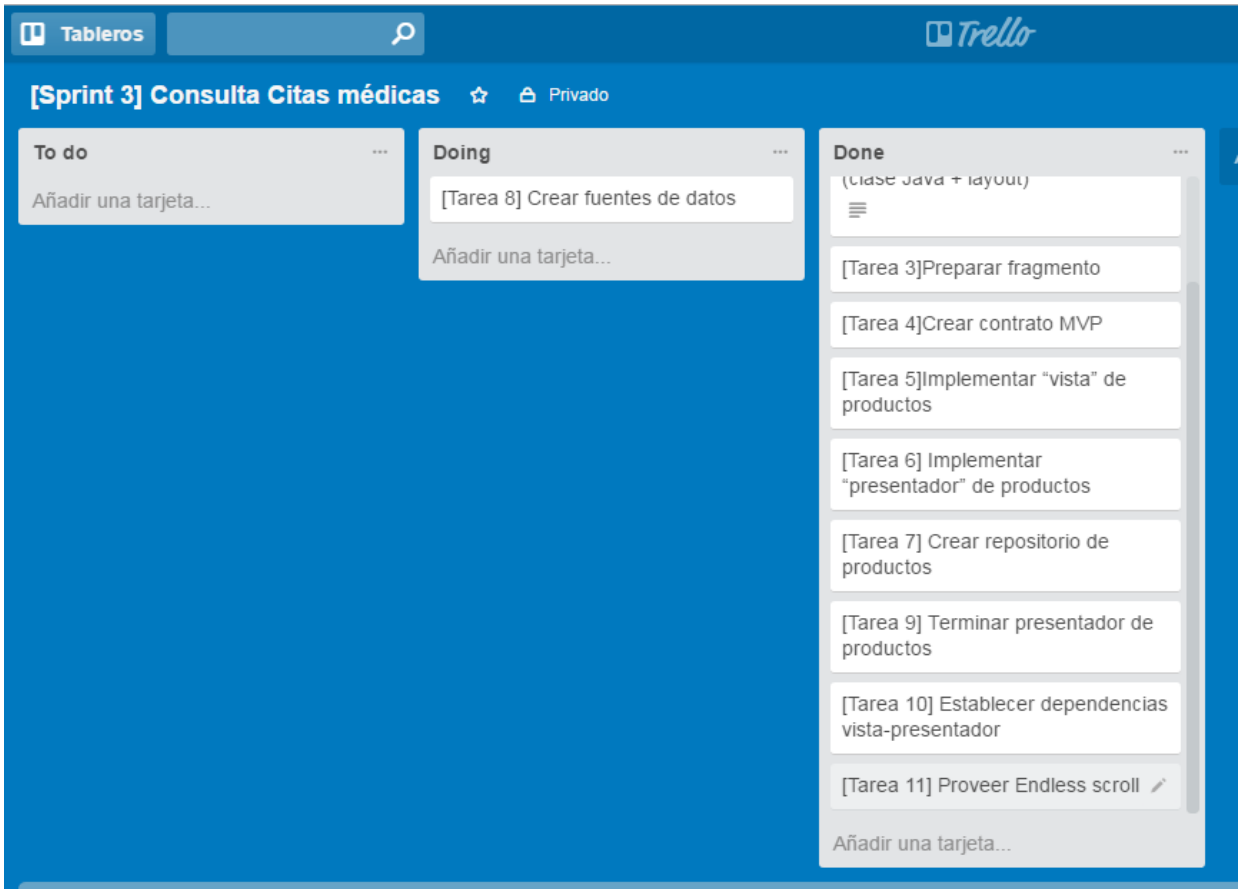


Tabla 4: Tabla Kanban Final Spring 1 (Cantallops Cifre; Francesc)

Como vemos la *"[Tarea8]Crear fuentes de datos"* no se ha podido finalizar. Por eso nuestro primer incremento no leerá los datos de un servidor sino que lo hará de forma local.

Primera Entrega Incremental

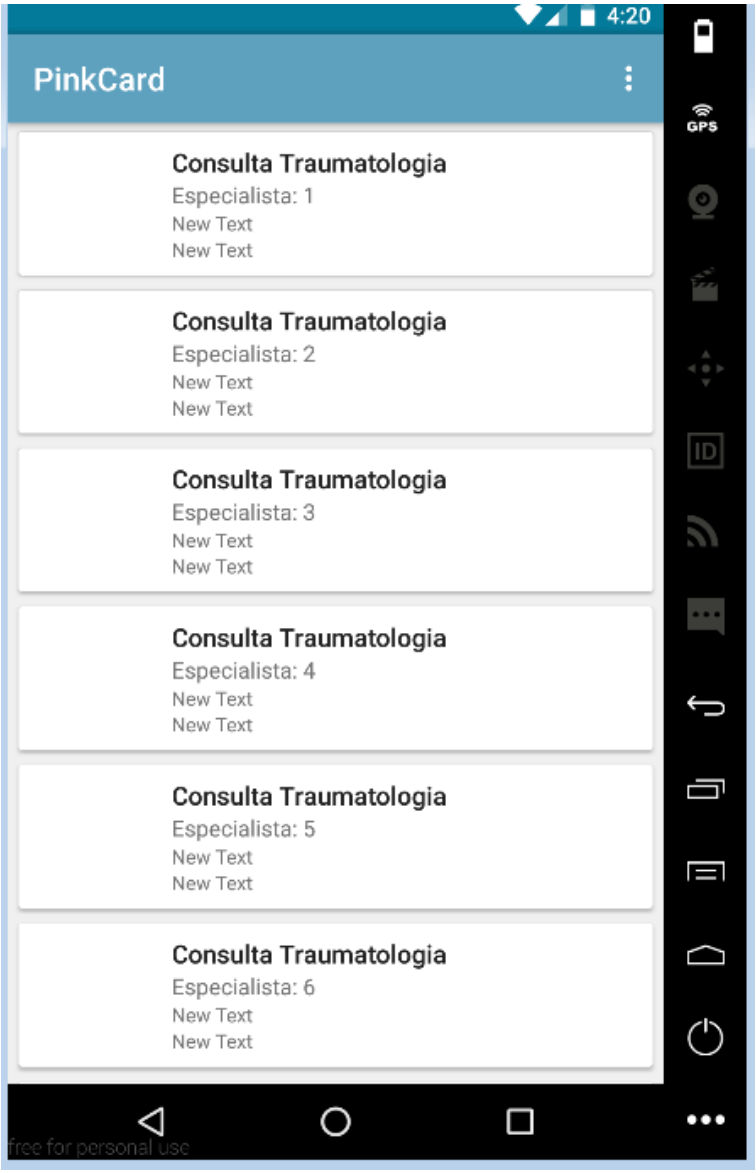


Ilustración 14: Primera Entrega Incremental (Cantallops Cifre; Francesc)

Conclusiones

Si bien el proyecto se había concebido inicialmente como una App completa de Citas Médicas capaz de registrar usuarios, consultar las citas pendientes, recibir avisos recordatorios de cita, entre otras funciones; debido al limitado tiempo con en que contamos para poder desarrollar el producto, únicamente se ha podido implementar la parte más interesante a mi parecer, se ha diseñado e implementado la consulta de citas por parte del usuario.

Sigo trabajando para, en un futuro cercano, implementar el registro/login de usuarios así como el aviso a los usuarios por SMS. La idea es crear una app con todos los detalles necesarios para optimizar el servicio de citas médicas.

Annexos

- ..\PAC3\fcantallops_PAC3\Burnup_Burndown_Sprint1.xlsx
- ..\PAC3\fcantallops_PAC3\Product Backlog.xlsx
- ..\PAC3\fcantallops_PAC3\Sprint Backlog_Sprint1.xlsx

Bibliografia

- 30 ways to reduce patient no-shows. (July 9, 2010). MGMA In Practice Blog.
- Cerner. (mar 22, 2016). MillenniumObjects Reference Pages. https://wiki.ucern.com/display/reference/MillenniumObjects+Reference+Pages.
- DuMontier, C., Kirsten Rindfleisch, M., Jessica Pruszynski, P., & John J. Frey III, M. (OCTOBER 2013). A Multi-Method Intervention to Reduce. FAMILY MEDICINE, VOL. 45, NO. 9,pp. 634-641.
- Echegaray, M. d. (2014). Los 'malos' pacientes salen caros. Revista Médica, n180.
- https://developer.android.com/. (n.d.). https://developer.android.com/. https://developer.android.com/about/dashboards/index.html#Screens.
- Kishor S. Wagh., R. C. (November 2013). Web Service Provisioning on Android Mobile Host. International Journal of Computer Applications (0975 – 8887), Volume 81 – No 14.
- Revelo, J. (Diciembre 2015). Consumir un Servicio Web REST Desde Android. Hermosa Programación.
- Revelo, J. (Octubre 2016). Tutorial Retrofit En Android: Planificación Aplicación Médica. Hermosa Programación.
- W.Frank Ableson, R. S. (2011). Android : Guía desarrolladores. Madrid: Ediciones Anaya Multimedia.