



UNIVERSIDAD DE VALLADOLID

E.T.S. DE INGENIERÍA INFORMÁTICA

Ingeniería Técnica en Informática de Gestión

---

**APLICACIÓN COMPLEMENTARIA A LA  
INICIATIVA DE LA AECC DIARIO DE UN  
SUPERVIVIENTE**

---

*Autores:*

Fernando Santa Olaya Rodríguez  
Rubén Toquero González

*Tutor:*

Fernando de Prada Moraga



---

## Agradecimientos

---

*A mis padres Fernando y Filomena que me dieron a mí y a mis hermanos todo lo que ellos no tuvieron y por lo que se desvivieron trabajando, sin ellos no sería quien soy. Y a mis hermanos y hermana, David, Marta y José Ignacio, porque sin ellos mi vida hubiera sido muchísimo más aburrida y tampoco sería el hombre que hoy soy si ellos no hubiesen estado a mi lado.*

*También quiero agradecer a nuestro tutor Fernando de Prada Moraga por toda la paciencia que ha tenido con nosotros, la comprensión y la ayuda que nos ha brindado tantas veces con este y otros asuntos durante la etapa estudiantil.*

*A la familia más extensa, tíos, tías, primos etc por apoyarme y animarme cada vez que nos juntábamos.*

*A mis amigos y compañeros de clase, la familia que eliges, por todos los sin sabores que me han acompañado a transitar en estos años y no mandarme muy lejos casi nunca.*

*A Rubén por ser mi motor en esta última etapa y forzarme a empujar un poco más cada día y robar tiempo de donde no lo había.*

*Por último a todos aquellos profesores que me ayudaron a llegar aquí, desde EGB hasta este último proyecto, por saber despertar la curiosidad y ansia de saber y por su dedicación a mi formación no sólo como ingeniero sino también como persona.*

*Fernando Santa Olaya Rodríguez*

*Quiero agradecer a mi familia todo el apoyo que me han dado durante todos estos años, a los que están y a los que ya no están, a los que dedico este trabajo por razones obvias.*

*También quiero agradecer al tutor Fernando de Prada Moraga toda la paciencia que ha tenido con nosotros, nos ha comprendido y nos ha ayudado a llevar a cabo esta necesaria tarea.*

*A Nuria, que es la que me ha sufrido la mayor parte del tiempo, todavía me resulta incomprendible que me siga aguantando después de tantos años.*

*A Fer que también me ha aguantado como un jabato y ha tenido paciencia y comprensión.*

*Por último agradezco a los profesores de esta escuela sus enseñanzas (espero que bien aprovechadas por mi parte) y a todos aquellos con los que he compartido mi tiempo en mi paso por esta etapa de la vida que aquí se cierra, o no.*

*Rubén Toquero González*



---

## Resumen

---

El presente proyecto implementa una aplicación móvil enmarcada dentro del programa 'Diario de salud para supervivientes de Cáncer' de la Asociación Española Contra el Cáncer (AECC) [3], esta aplicación funcionará como un micro asistente virtual para personas que padeczan o hayan padecido la lacra moderna que resulta ser la enfermedad del cáncer.

Es decir, esta aplicación proporcionará las herramientas necesarias para llevar al día el control de los cuidados necesarios para el tratamiento de un paciente que ha superado la enfermedad pero necesita llevar un control de sus actividades rutinarias, citas médicas, medicación, pruebas, síntomas y de las personas que participan diariamente de su vida.

Esta aplicación es la evolución lógica del libro que utiliza la AECC para guiar a los 'supervivientes' de la enfermedad a enfocar su día a día y facilitarles la convivencia con las acciones dedicadas o no al tratamiento de su enfermedad y a hacer de este algo más llevadero.



Figura 1: Portada del folleto de la AECC.



---

## **Abstract**

---

This project implements a mobile application framed in the 'Journal of Health Cancer survivors' of the Spanish Association Against Cancer (AECC) program [3], this application will function as a virtual micro assistant for people who is suffering or have suffered from the modern scourge that turns out to be the disease of cancer.

That is to say, this application will provide the necessary tools to keep up to date control of the care needed to treat a patient who has overcome the disease but needs to keep track of their routine activities, medical appointments, medications, tests, symptoms and people involved in their daily life.

This application is the logical evolution of the book used by the AECC to guide the 'survivors' of the disease to focus the day and provide coexistence with the dedicated actions or no actions to treat their condition and make this more bearable [24].



---

## Índice general

---

<b>Índice de figuras</b>	<b>13</b>
<b>Índice de cuadros</b>	<b>15</b>
<b>1. Introducción</b>	<b>17</b>
1.1. Antecedentes y Motivación del proyecto . . . . .	17
1.2. Ámbito de trabajo . . . . .	19
1.3. Objetivos del proyecto . . . . .	19
1.4. Estructura del proyecto . . . . .	20
1.5. Estructura de la memoria . . . . .	20
<b>2. Visión general del Proyecto</b>	<b>21</b>
2.1. Estado del arte de la movilidad . . . . .	21
2.1.1. Smartphones . . . . .	21
2.1.2. Sistemas operativos Móviles . . . . .	22
2.2. ¿Porqué Android? . . . . .	23
2.3. Parte App móvil . . . . .	24
2.4. Parte Servidora . . . . .	25
2.5. Generación de la Documentación . . . . .	26
<b>3. Planificación y Metodología</b>	<b>29</b>
3.1. Plan de desarrollo de software . . . . .	29
3.2. Propósito general de la planificación . . . . .	29
3.3. Metodología . . . . .	29
3.3.1. SCRUM . . . . .	30
3.3.2. Roles y responsabilidades . . . . .	32
3.3.3. Eventos y rituales . . . . .	33
3.4. Planificación completa . . . . .	35
3.4.1. Puntos de Historia . . . . .	35
3.4.2. Historias de Usuario, Historias Épicas . . . . .	35
3.4.3. Tipos de historias . . . . .	37
3.4.4. Velocidad de un equipo . . . . .	38
3.4.5. Planificación de los Sprints . . . . .	38
3.4.6. Diagrama de sprints . . . . .	39
3.5. Recursos necesarios . . . . .	40

<b>4. Análisis</b>	<b>43</b>
4.1. Explicación detallada de las funcionalidades de la aplicación . . . . .	44
4.2. Documento de Análisis . . . . .	46
4.2.1. Descripción de objetivos de manera detallada . . . . .	46
4.2.2. Captura de Requisitos . . . . .	46
4.2.3. Identificación de actores . . . . .	48
4.2.4. Casos de uso . . . . .	49
4.2.5. Descripción de Casos de Uso . . . . .	51
4.2.6. Diagrama de clases de análisis . . . . .	78
4.2.7. Diagrama Entidad-Relación de la base de datos . . . . .	78
4.2.8. Modelo relacional del análisis . . . . .	78
<b>5. Diseño</b>	<b>79</b>
5.1. Parte App móvil . . . . .	79
5.2. Arquitectura . . . . .	79
5.2.1. Ley de Deméter . . . . .	80
5.2.2. Capa de Vista . . . . .	81
5.2.3. Capa de Dominio . . . . .	81
5.2.4. Capa de Datos . . . . .	82
5.2.5. Principios SOLID . . . . .	82
5.3. Diagrama de Clases . . . . .	85
5.3.1. Capa de Datos . . . . .	85
5.3.2. Capa de Dominio . . . . .	86
5.3.3. Capa de Presentación . . . . .	86
5.4. Diagrama de Clases . . . . .	86
5.5. Diagrama E/R de la base de datos . . . . .	86
5.5.1. Diagrama . . . . .	87
5.5.2. Descripción . . . . .	87
5.5.3. Capa de Datos . . . . .	88
5.5.4. Capa de Dominio . . . . .	89
5.5.5. Capa de Presentación . . . . .	89
5.6. Diseño de la base de datos . . . . .	89
5.7. Prototipado . . . . .	89
5.7.1. Material Design . . . . .	90
5.7.2. Pantallas Principales . . . . .	90
5.8. Iconografía y otros cambios . . . . .	101
<b>6. Implementación y Pruebas</b>	<b>105</b>
6.1. Construcción . . . . .	105
6.2. Plan de desarrollo . . . . .	111
6.3. Versionado y Sincronización . . . . .	113
6.4. Pruebas . . . . .	120
6.4.1. Tipos de pruebas . . . . .	120
6.4.2. ATDD y TDD . . . . .	121
6.4.3. BDD . . . . .	121
6.4.4. Pruebas en el dispositivo . . . . .	123
6.5. Puesta en producción . . . . .	124

<b>7. Conclusiones y trabajo futuro</b>	<b>125</b>
7.1. Conclusiones . . . . .	125
7.2. Dificultades encontradas . . . . .	126
7.3. Consecución de Objetivos . . . . .	127
7.4. Conocimientos adquiridos . . . . .	127
7.5. Trabajo futuro . . . . .	127
7.6. Estructura y descripción de los contenidos de CDROM . . . . .	129
<b>8. Bibliografía</b>	<b>131</b>
<b>Bibliografía</b>	<b>133</b>



---

## Índice de figuras

---

1.	Portada del folleto de la AECC . . . . .	5
2.1.	Gráfico sobre el uso actual de SO móviles internacional en Marzo de 2015 . . . . .	23
2.2.	Vista general de TexStudio . . . . .	26
2.3.	Configuración del documento . . . . .	27
2.4.	Capítulos dependientes del documento principal . . . . .	28
3.1.	Resumen de Scrum . . . . .	32
3.2.	Ejemplo de historia de usuario . . . . .	36
3.3.	Diagrama de sprints en el tiempo . . . . .	39
4.1.	CRUD de medicamento (caso de uso) . . . . .	50
5.1.	Arquitectura ' limpia ' propuesta por Robert Martin . . . . .	80
5.2.	Diagrama de clases de la capa de datos . . . . .	85
5.3.	Diagrama de clases de la capa de dominio . . . . .	86
5.4.	Diagrama de clases de la capa de presentación . . . . .	86
5.5.	Diagrama entidad relación de la base de datos local . . . . .	87
5.6.	Diagrama de clases de la capa de datos . . . . .	88
5.7.	Diagrama de clases de la capa de dominio . . . . .	89
5.8.	Diagrama de clases de la capa de presentación . . . . .	89
5.9.	Iniciativa de la AECC . . . . .	90
5.10.	Drawer menú y Pantalla principal . . . . .	91
5.11.	Vista de los días en el folleto . . . . .	92
5.12.	Horario de actividades . . . . .	92
5.13.	Cuadro de actividades a realizar . . . . .	93
5.14.	Pantalla con la interfaz para las rutinas . . . . .	93
5.15.	Página con información a completar en la cita médica . . . . .	94
5.16.	Pantalla con la actividad de las citas . . . . .	94
5.17.	Medicación y posología de los medicamentos . . . . .	95
5.18.	Pantalla con la actividad de la medicación . . . . .	95
5.19.	Pantalla con la actividad de los personajes . . . . .	96
5.20.	Pantalla con la actividad de las pruebas . . . . .	96
5.21.	Cuadro de los síntomas en el libro . . . . .	97
5.22.	Pantalla con la actividad de los síntomas . . . . .	97
5.23.	Ejercicios de meditación . . . . .	98
5.24.	Contactos importantes a tener en cuenta . . . . .	98
5.25.	Pantalla con la actividad de los recursos . . . . .	99
5.26.	Pantalla con la actividad configuración del perfil . . . . .	99
5.27.	Pantalla de presentación de la aplicación . . . . .	100
5.28.	Icono de la aplicación principal . . . . .	101

5.29. Conjunto de iconos de apoyo para las pantallas de las entidades . . . . .	101
5.30. Nuevo diseño de las pantallas generales de la aplicación . . . . .	102
5.31. Nuevo diseño de los recursos . . . . .	103
6.1. Logo de la aplicación Trello . . . . .	106
6.2. Vista general de la columnas de scrum en Trello . . . . .	106
6.3. Detalle de las columnas product backlog y ToDo . . . . .	107
6.4. Detalle de las columnas Doing, Done, Tested y To Ship . . . . .	108
6.5. Detalle de la columna Shipped . . . . .	109
6.6. Implementación de las Historias de usuario en Trello . . . . .	110
6.7. Agrupación de tareas por sprints . . . . .	111
6.8. Integración de Git en Android Studio . . . . .	114
6.9. Funcionamiento de Git en el versionado de archivos . . . . .	115
6.10. PowerShell para Windows . . . . .	116
6.11. PowerShell con los comandos más habituales a la hora de operar. . . . .	117
6.12. GitHub desktop, repositorios de la aplicación y de la memoria . . . . .	118
6.13. Repositorio de GitHub desde la propia página web . . . . .	119
6.14. Ciclo de TDD con breves tips de cada una . . . . .	120
6.15. Ciclos integrados de TDD y BDD . . . . .	122
6.16. Otra imagen de los ciclos integrados de TDD y BDD . . . . .	123
6.17. Consola de Play Store, mostrando gestión de APK y versiones beta y alpha .	124
7.1. Estructura del contenido del CDROM . . . . .	129

---

## Índice de cuadros

---

4.1.	Perfil de usuario - Inserción . . . . .	51
4.2.	Perfil de usuario - Consulta . . . . .	51
4.3.	Perfil de usuario - Edición . . . . .	52
4.4.	Perfil de usuario - Borrado . . . . .	52
4.5.	Cita médica - Inserción . . . . .	53
4.6.	Cita médica - Consulta . . . . .	54
4.7.	Cita médica - Edición . . . . .	55
4.8.	Cita médica - Borrado . . . . .	56
4.9.	Rutina - Inserción . . . . .	57
4.10.	Rutina - Consulta . . . . .	58
4.11.	Rutina - Edición . . . . .	59
4.12.	Rutina - Borrado . . . . .	60
4.13.	Medicamento - Inserción . . . . .	61
4.14.	Medicamento - Consulta . . . . .	62
4.15.	Medicamento - Edición . . . . .	63
4.16.	Medicamento - Borrado . . . . .	63
4.17.	Personaje - Inserción . . . . .	64
4.18.	Personaje - Consulta . . . . .	65
4.19.	Personaje - Edición . . . . .	66
4.20.	Personaje - Borrado . . . . .	66
4.21.	Síntoma - Inserción . . . . .	67
4.22.	Síntoma - Consulta . . . . .	68
4.23.	Síntoma - Edición . . . . .	69
4.24.	Síntoma - Borrado . . . . .	69
4.25.	Prueba - Inserción . . . . .	70
4.26.	Prueba - Consulta . . . . .	71
4.27.	Prueba - Edición . . . . .	72
4.28.	Prueba - Borrado . . . . .	73
4.29.	Ordenación de listas . . . . .	73
4.30.	Consulta de listas . . . . .	74
4.31.	Añadir notificación - Medicamento . . . . .	74
4.32.	Ordenación de listas . . . . .	75
4.33.	Añadir notificación - Rutina . . . . .	75
4.34.	Añadir objetos a la cita . . . . .	76
4.35.	Añadir objetos a la rutina . . . . .	77
4.36.	Configuración de la aplicación . . . . .	77



# CAPÍTULO 1

---

## Introducción

---

### 1.1. Antecedentes y Motivación del proyecto

Actualmente existen en España alrededor de un millón y medio de supervivientes de cáncer, personas que necesitan reincorporarse a su vida cotidiana, afectados que a menudo presentan secuelas físicas temporales o permanentes que pueden dificultar, entre otros, la reinserción laboral y la vuelta a su rutina diaria[11].

El cáncer es una enfermedad provocada por un grupo de células que se multiplican sin control y de manera autónoma, invadiendo localmente y a distancia otros tejidos. El cáncer es una de las principales causas de mortalidad en todo el mundo; en 2012 hubo unos 14 millones de nuevos casos y 8,2 millones de muertes relacionadas con el cáncer.

Se prevé que el número de nuevos casos aumente en aproximadamente un 70 % en los próximos 20 años. En 2012, los cánceres diagnosticados con más frecuencia en el hombre fueron los de pulmón, próstata, colon y recto, estómago e hígado, en la mujer fueron los de mama, colon y recto, pulmón, cuello uterino y estómago.[15]

Aproximadamente un 30 % de las muertes por cáncer son debidas a cinco factores de riesgo conductuales y dietéticos: índice de masa corporal elevado, ingesta reducida de frutas y verduras, falta de actividad física, consumo de tabaco y consumo de alcohol.

«Cáncer» es un término genérico que designa un amplio grupo de enfermedades que pueden afectar a cualquier parte del organismo; también se habla de «tumores malignos» o «neoplasias malignas». Una característica del cáncer es la multiplicación rápida de células anormales que se extienden más allá de sus límites habituales y pueden invadir partes adyacentes del cuerpo o propagarse a otros órganos, proceso conocido como metástasis. Las metástasis son la principal causa de muerte por cáncer.

¿Cuál es la causa del cáncer?

El cáncer comienza en una célula en la que se produce la transformación de una célula normal en tumoral.

El envejecimiento es otro factor fundamental en la aparición del cáncer. La incidencia de esta enfermedad aumenta muchísimo con la edad, muy probablemente porque se van acumulando factores de riesgo de determinados tipos de cáncer. La acumulación general de factores de riesgo se combina con la tendencia que tienen los mecanismos de reparación celular a

perder eficacia con la edad.

Factores de riesgo del cáncer El consumo de tabaco y alcohol, la dieta malsana y la inactividad física son los principales factores de riesgo de cáncer en todo el mundo. Algunas infecciones crónicas también constituyen factores de riesgo, y son más importantes en los países de ingresos medios y bajos.

En resumidas cuentas, el cáncer es una gran lacra en nuestros días [20] y uno de los aspectos importantes en el tratamiento de este grupo de enfermedades, es la buena disposición y estado mental positivo del enfermo. No cura la enfermedad pero ayuda a superar los duros tratamientos a los que se ven sometidos los pacientes.

Por otro lado, de un tiempo a esta parte las aplicaciones móviles o apps forman parte de nuestros smartphones y por tanto de nuestras vidas. Son tan comunes que para cada ámbito o actividad existe una app que apoya, ayuda, informa o al menos lo pretende con mayor o menor éxito.

Dentro de las apps relacionadas con la salud ninguna o muy pocas de estas aplicaciones tratan sobre esta enfermedad o grupo de enfermedades.

Debido a que ambos autores de este proyecto fuimos tocados de cerca por esta enfermedad y que a la Asociación Española contra el Cáncer le surgió la necesidad, decidimos apoyar de la mejor manera que podemos, que es escuchando sus necesidades e implementándolas en una app para el sistema operativo móvil Android y aconsejándoles las mejores alternativas en ese sentido.

Mediante el desarrollo de esta aplicación pretendemos hacer más llevaderos los procesos que ocurren a continuación de pasar por el traumático post tratamiento y cicatrices visibles o no que deja esta enfermedad y devolver la normalidad a esos pacientes que quieren continuar de una manera convencional con sus vidas. Haciéndoles más llevaderas algunas de las tareas de obligado cumplimiento 'posterior' que deben realizar estos verdaderos luchadores y supervivientes.

Sin ánimo de ser victimistas, nosotros mismos conocemos de primera mano todo el sufrimiento que se genera, la angustia y el dolor que causa esta enfermedad y que es una de las principales causas de muerte en el mundo actual.

Esta aplicación surge como complemento digital a un dossier/iniciativa de la AECC que pretende servir al enfermo como centro de datos y recolección de información que es el '**Diario de salud para supervivientes de Cáncer**', en ese sentido esta app pretende ofrecer todo lo que permite el dossier adaptado al mundo digital y aportar funcionalidades extra y de valor, como pudieran ser filtros sobre los datos, alarmas y notificaciones en las citas, y como modo más experimental el contacto directo con el agente del la propia AECC como si de una app de mensajería instantánea se tratase (actualmente en proceso).

## 1.2. Ámbito de trabajo

En el desarrollo de este proyecto hay varios ámbitos de trabajo, algunos cercanos pero ligeramente diferenciados, y otros más alejados del de desarrollo.

El primero de ellos es el de una app de lo que podríamos llamar usuario final, para el uso que el propio superviviente crea conveniente dentro de las capacidades de la misma. Es por eso que la usabilidad, el diseño, los tutoriales, ayudas, etc, deben ir en linea, con lo cual el espectro de usuarios potenciales es muy amplio, y estos mismos ademas están atravesando un trance complicado.

El segundo ámbito de trabajo se enmarca en la parte servidora, por usuarios de la misma, aunque en este caso podemos presuponer cierto entrenamiento y hábito en el uso de herramientas web, además de que desde la propia AECC se formará a quienes estén manejando esta web y que hemos denominado agentes (actualmente en proceso).

El otro es probar un desarrollo usando las metodologías ágiles, sus prácticas, rituales, etc tratando de adaptarlas al desarrollo vivo de un producto en el que estamos involucrados pero que no es nuestra decisión el por donde va a acontecer el desarrollo futuro del mismo.

En resumen el uso y ámbito de trabajo donde general es el de uso publico por usuarios no categorizados ni expertos por un lado, y usuario final con entrenamiento y conocimiento por otro.

## 1.3. Objetivos del proyecto

El objetivo principal del proyecto es servir de apoyo a las acciones emprendidas por la AECC en el marco de la nueva estrategia de identidad digital mediante aplicaciones útiles para cada una de sus iniciativas y las de apoyo general y las acciones de la asociación, más concreto en este primer paso con la iniciativa de Diario de un Superviviente.

Por tanto uno de los objetivos del proyecto será el desarrollo de una app móvil denominada "Diario de un Superviviente" que sirva a los intereses de la AECC en la línea de apps en los stores y presencia corporativa e imagen digital. Es por esto que no solamente será el objetivo de este proyecto la realización de la app, sino que la ayuda a la campaña en la web mediante banners, y desarrollo de la propia sección en la Play store, quedando para futuras evoluciones el desarrollo y la inclusión en la tienda del sistema operativo iOS.

En este sentido el otro gran objetivo es desarrollar una parte servidora, en forma de API REST, para realizar sincronizaciones entre dispositivos, recogida de estadísticas, seguimiento por parte del agente asignado, estos últimos puntos ademas presentados en una web para dicho agente aprovechando la API antes mencionada, quedando también para futuras líneas la implementación de apps que sirvan a los agentes en las tareas que ahora realizan con la web.

Por último el otro objetivo es ver si podemos adaptar una metodología de desarrollo nueva para ver que pros y contras pudiera tener la misma en el desarrollo de aplicaciones en el mundo real de las empresas de software.

## 1.4. Estructura del proyecto

El proyecto se estructurará en tres partes bien diferenciadas la app móvil, la parte servidora y por último toda la documentación, tanto de esta memoria como los manuales de instalación y usuario del resto de partes.

## 1.5. Estructura de la memoria

Explicación breve del contenido de cada capítulo:

**Capítulo 1** contexto general en el que se desenvuelve la aplicación, explicando el problema existente y la solución propuesta para dicho problema. Termina mostrando la estructura del presente documento.

**Capítulo 2** breve explicación de las tecnologías seleccionadas resumen de la plataforma elegida, presentación general del proyecto y justificación del mismo.

**Capítulo 3** metodología utilizada para el desarrollo de la aplicación, planificación, plazos, técnicas utilizadas para llevar a cabo las funcionalidades de la aplicación.

**Capítulo 4** fase de análisis donde podrá encontrar aspectos clave para el buen desarrollo del producto como una introducción a la aplicación y los estudios sobre la arquitectura utilizada, historias de usuario, casos de uso, diagrama de clases, entidad-relación.

**Capítulo 5** diseño, determina el comportamiento esperado de la aplicación mediante una serie de diagramas de secuencia y un prototipado de bajo nivel.

**Capítulo 6** construcción de la aplicación, como se ha conseguido llevar a cabo la implementación de las funcionalidades de la aplicación. Además incluye las pruebas realizadas para determinar el buen funcionamiento de la misma.

**Capítulo 7** presenta las conclusiones obtenidas una vez finalizado el desarrollo del proyecto, y una serie de trabajos futuros que podrían desarrollarse como mejoras de la aplicación.

**Capítulo 8** bibliografía y referencias web, que incluyen toda documentación consultada para la elaboración del proyecto.

**Glosario** recoge los términos que necesiten una breve explicación para aclarar su significado y facilitar al lector la comprensión de la memoria.

**Anexo I** contiene el manual de usuario, así como el manual de instalación para aquellos usuarios que no estén tan familiarizados con la plataforma Android.

# CAPÍTULO 2

---

## Visión general del Proyecto

---

El concepto de aplicación es un asistente-diario para un enfermo de cáncer en la parte de la app móvil y una parte servidora que se encarga de sincronizar datos entre diferentes dispositivo, una base datos para dar apoyo a esto y que ademas reciba datos estadísticos de la app, por si alguna vez pueden extraer conclusiones tras un tratamiento estadístico de los mismos. Además se incluye un servicio de comunicación con los usuarios mediante notificaciones push a los dispositivos

### 2.1. Estado del arte de la movilidad

Dentro del concepto Movilidad se agrupan, tanto el hardware como el software de pequeños dispositivos portables o dispositivos móviles. Podríamos definir a estos últimos como “aparatos de pequeño tamaño, con algunas capacidades de procesamiento, con conexión permanente o intermitente a una red, con memoria limitada, diseñados específicamente para una función, pero que pueden llevar a cabo otras funciones más generales.”

De entre los elementos que pueden llegar a ser un dispositivo móvil (una PDA, un teléfono móvil, un lector de libros electrónicos o un ordenador portátil) destaca, sin lugar a dudas, el teléfono móvil como el dispositivo más utilizado de entre todos. Y entre los teléfonos móviles destacan los denominados “SmartPhones y Tablets” por ser tener unas capacidades y potencia de ejecución de aplicaciones complejas de consumo de lo más variadas, desde reproductores de todo tipo de media, hasta mensajería instantáneas, llamadas apps. Es el desarrollo de estas apps y su comercialización por parte de grandes compañías tecnológicas como Google, Apple y Microsoft lo que ha devenido en todo un nuevo mercado de desarrollo de software de todo tipo, no solo para grandes compañías sino también para pequeños grupos de desarrollo o incluso de desarrolladores indies.

#### 2.1.1. Smartphones

Un Smartphone (cuya traducción sería “teléfono inteligente”) “es una evolución del teléfono móvil tradicional que cuenta con ciertas características y prestaciones que lo acercan más a un ordenador personal que a un teléfono tradicional.” Entre dichas prestaciones y características, se encuentran una amplia mejora del almacenamiento de datos, conexión a Internet mediante una tarifa contratada o haciendo uso de redes WIFI, acelerómetro, pantalla táctil, teclado QWERTY ...y un sinfín de aplicaciones de usuario, además de la posibilidad

de descarga de nuevas aplicaciones.

Todas estas prestaciones y características de los Smartphone estarían desaprovechadas sin software que las saque partido. Por ello los Smartphone llevan un SO (Sistema Operativo) que les permite realizar todas estas tareas de una forma rápida y sencilla, además de optimizando el consumo energético ya que dependen de una batería para ello la mayoría de las veces. En el siguiente apartado, realizaremos una breve descripción de los SO para dispositivos móviles y Smartphone más importantes que se encuentran en el mercado a día de hoy.

### 2.1.2. Sistemas operativos Móviles

Partiendo de una definición de sistema operativo: “Capa compleja entre el hardware y el usuario, concebible también como una máquina virtual, que facilita al usuario o al programador las herramientas e interfaces adecuadas para realizar sus tareas informáticas, abs-trayéndole de los complicados procesos necesarios para llevarlas a cabo.”

Por ende, un sistema operativo móvil es un sistema operativo que controla un dispositivo móvil. Sin embargo, estos son mucho muy diferentes a los tradicionales SO de equipos de escritorio, para empezar hay mucha mas variedad, pero es que la interacción con el usuario, la gestión de la memoria, las tareas de llamadas que deben ser prioritarias, etc

El sistema operativo destinado a controlar un dispositivo móvil necesita ser fiable y tener una gran estabilidad, ya que incidencias habituales y toleradas en ordenadores personales como reinicios o caídas no tienen cabida en un dispositivo de estas características. Además, ha de adaptarse adecuadamente a las consabidas limitaciones de memoria y procesamiento de datos, proporcionando una ejecución exacta y excepcionalmente rápida de cara al usuario.

Estos sistemas han de estar perfectamente testados y libres de errores antes de incorporarse definitivamente a la línea de producción. Las posibilidades que existen en un ordenador estándar de realizar actualizaciones e incluso reinstalar mejores versiones del sistema para cubrir fallos o deficiencias son más limitadas en un dispositivo móvil.

Es posible incluso que un aparato de esta naturaleza deba estar funcionando ininterrumpidamente durante semanas e incluso meses antes de ser apagado y reiniciado, a diferencia de lo que ocurre con un ordenador personal. El consumo de energía es otro tema muy delicado: es importante que el sistema operativo haga un uso lo más racional y provechoso posible de la batería, ya que esta es limitada y el usuario siempre exige una mayor autonomía.

En la actualidad, existen varios sistemas operativos para toda la gama de dispositivos móviles. Más adelante veremos por qué se ha elegido Android para la realización de este proyecto fin de carrera, pero antes veamos las características más importantes de los principales sistemas operativos móviles.

En 2015 en España y el mundo dentro de este mercado de SO de la movilidad se pueden encontrar diferentes actores que proveen de lo que denominamos sistemas operativos móviles, existiendo desde los provistos por las antes mencionadas grandes compañías tecnológicas hasta aquellos provistos por fundaciones, empresas que han perdido gran parte del negocio o empresas compradas por alguna de las grandes compañías, un repaso somero de los distintos SO móviles que podemos encontrar hoy día nos arroja nombres como Android, iOS, Windows Phone, Balckberry, Symbian, Firefox OS, Ubuntu Touch, etc

## 2.2. ¿Porqué Android?

Dentro del mercado de SO móviles solamente Android e iOs pueden considerarse grandes actores propiamente dichos pues entre ambos copan cerca del 90 % del mercado de Smartphones en España y más del 91 % mundial[37]. El resto algunas tuvieron gran peso pero hoy están en franca retirada (Blackberry), no terminan de funcionar todo lo bien que se desea entre el gran público (Windows Phone), son de dispositivos de otra era no están pensados para la movilidad moderna (Symbian), o no terminan de salir del halo de prototipo y aun deben pasar ciclos en fases beta para poder ser pasadas al gran público (Ubuntu Touch, Firefox OS).

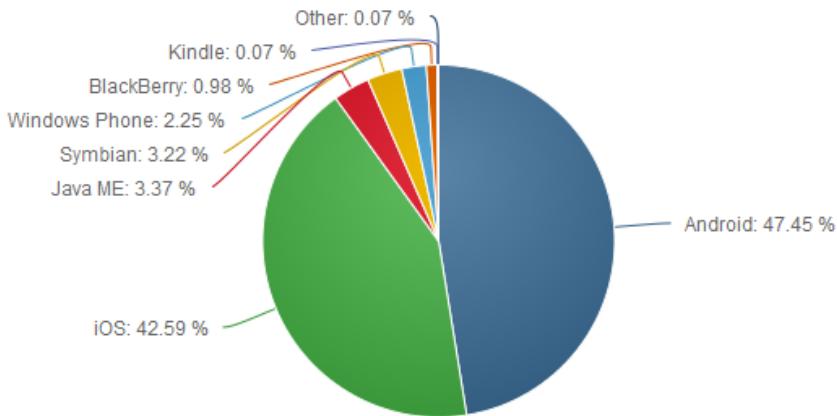


Figura 2.1: Gráfico sobre el uso actual de SO móviles internacional en Marzo de 2015.

Así pues la elección se encontraba entre Android e iOs de cara a afrontar este proyecto. Si tenemos en cuenta la cuota de mercado de cada uno de ellos, Android sin duda alguna es el gran vencedor, pero es que además la gama de dispositivos que ofrece dicho SO es bastante más amplia y con mayor variabilidad en el precio de los terminales, permitiendo que más y más personas accediesen a la app.

A esto hay que añadir los costes de desarrollo, ya que para publicar aplicaciones en las tiendas de las mismas de los SO móviles se necesita de un proceso de registro como tal desarrollador en las tiendas, y este proceso no es gratuito. Para ser claros diremos que el coste en iOs es de 99 dólares anuales, además de disponer de un hardware específico para realizar el desarrollo y compilación de la app. Mientras que en Android el coste de 25 dólares en un pago único y el desarrollo y compilación puede llevarse a cabo con casi cualquier infraestructura de SO, hardware etc.

Es por eso que para este piloto de la iniciativa de la Asociación Española contra el Cáncer se haya optado por un desarrollo más económico, y tras analizar los resultados si evalúa la iniciativa como positiva y avanza del estadio de piloto, se planteará por parte de la AECC futuros desarrollos en ese sentido, pero eso queda fuera del alcance y ámbito de este proyecto.

## 2.3. Parte App móvil

En la parte de la app móvil hay 5 funcionalidades, aparte de los ajustes y preferencias, Personas Implicadas, Calendario de Citas y Posología, Seguimiento de Análisis, Rutina Diaria y Hablar con un agente, además de notificaciones locales preguntando por diferentes cosas, para mantener al usuario tanto en la app como con la moral lo más alta posible.

**Personas Implicadas** es una mini agenda personal, con el numero de teléfono o correo electrónico u otras formas de contacto de las personas que estén mas implicadas para el enfermo, se podrá especificar la relación que le une con el paciente, ya sea personal o derivada de su dolencia, como pueden ser su oncólogo, el agente de la AECC, un psicólogo propio, familiares o amigos de confianza con los que el agente de la AECC pueda contactar, previo consentimiento de estos y del paciente. En fin personas con interés en apoyar al enfermo en su duro trance de la enfermedad.

**Calendario de Citas y Posología** tiene como fin registrar las citas importantes relacionadas con el proceso del enfermo, como pueda ser revisiones, sesiones de quimio, entrega de análisis, ingresos, operaciones, etc Para que el enfermo disponga en un lugar de toda la información referente a su caso. Si toma algún medicamento, analgésico etc también se reflejar aquí como parte importante del proceso puramente médico en sí, esta parte constituye la funcionalidad principal y 'bebé' de la información y datos obtenidos de las otras secciones de la aplicación.

**Seguimiento de análisis** permite tener un pequeño histórico para el usuario de los análisis que le hayan sido realizados, permitiendo fotografiar los mismos para poder consultarlo siempre y ademas introducir que parámetros quiere obtener un especial seguimiento y gráfico de los mismos, es decir para ir comprobando su evolución

**Sintomatología** permite apuntar los síntomas que acontecen al paciente con fecha y hora de manera que estos puedan ser añadidos a la cita médica y sirvan de ayuda a la hora de hacer un diagnóstico más preciso de la situación de ese paciente, mejorando de manera directa su calidad de vida y el control de la enfermedad por parte de los facultativos al cargo del mismo.

**Rutina Diaria** tiene como finalidad la de ser un horario de actividades, tanto de dentro de la AECC como de fuera con la esperanza de que mediante la rutina y la realización de actividades el enfermo se encuentre mejor psicológicamente, además de físicamente en el caso de actividades físicas, y que mediante la rutina y realización activa de actividades consiga apartar de la mente la enfermedad, además el paciente podrá valorar de forma explícita el grado de satisfacción que le produce la realización de esta actividad.

**Hablar con un agente** permite durante ciertas horas del día que el enfermo pueda consultar o incluso simplemente charlar con el agente asignado de la AECC, para que sienta que siempre dispone de alguien que lo apoye desde el lado de la asociación.

Ademas de esto la app dispone de preferencias tanto de sonidos y notificaciones como aspecto, así como ajustes de usuario borrar cuenta, etc

Por ultimo la app funcionará mucho en base a la información recolectada en esta parte para lanzar notificaciones locales preguntando por diferentes cosas al enfermo, desde como te encuentras esta mañana? a la hora aproximada que en Rutina Diaria el usuario haya esta-

blecido como hora de despertar a has ido hoy a bailes de salón? el día que tenga marcado que tiene que ir a bailes pasando que animo tienes? tras haber salido de una sesión de quimioterapia. Todos estos datos desagregados del usuario se envían a la parte servidora para tratarlos con fines estadísticos y ponerlos a disposición de investigadores en el campo de la enfermedad.

## 2.4. Parte Servidora

La parte servidor lo primero que proporciona es una API REST(possible referencia) para interactuar con los recursos que ofrecerá.

Por una parte permite la sincronización de la información entre los diferentes dispositivos que un mismo enfermo pueda disponer. Esto lo hace de manera silenciosa enviando cada cambio al servidor, y este en cada conexión de un dispositivo pregunta por su estado de sincronización.

Por otro ofrece servicios para que cada dispositivo envíe la información estadística pertinente, ademas la almacenará en una base de datos, haciendo anónimos esos datos en el proceso por confidencialidad hacia el usuario, y permitiendo después su consulta mediante servicio web o en la web donde este alojada la parte servidora, para la monitorización de los mismos.

Dispondrá de control de usuarios, esto es que existirá un usuario encargado de ir asignando a los distintos agentes, generalmente por proximidad geográfica, para que este sea el encargado de monitorizar la actividad de sus enfermos

Se le ofrecerá al agente los contactos de cada enfermo que tenga asignado para en el caso de que necesitase conversare de alguna manera con alguno de ellos, bien sean familiares o el oncólogo si por ejemplo se encuentra en algún ensayo poder contrastar información

En relación a esto ultimo la parte servidora dispondrá de un chat que permita comunicar directamente al agente con el enfermo, bajo ciertas premisas.

## 2.5. Generación de la Documentación

Para la realización de la memoria, hemos elegido LaTeX y el editor TexStudio.

LaTeX es un sistema de composición de textos de alta calidad; que incluye características diseñadas para la producción de documentación técnica y científica.

LaTeX es el estándar de facto para la comunicación y publicación de documentos científicos y está disponible como software libre.

TexStudio es un entorno de escritura integrado para la creación de documentos LaTeX.

TexStudio hace la escritura LaTeX fácil y cómoda, tiene numerosas características como el resaltado de sintaxis, visor integrado, verificación de referencias y varios asistentes que hacen que introducir imágenes, tablas y diagramas sea sencillo.

TexStudio es de código abierto y está disponible para los principales sistemas operativos.

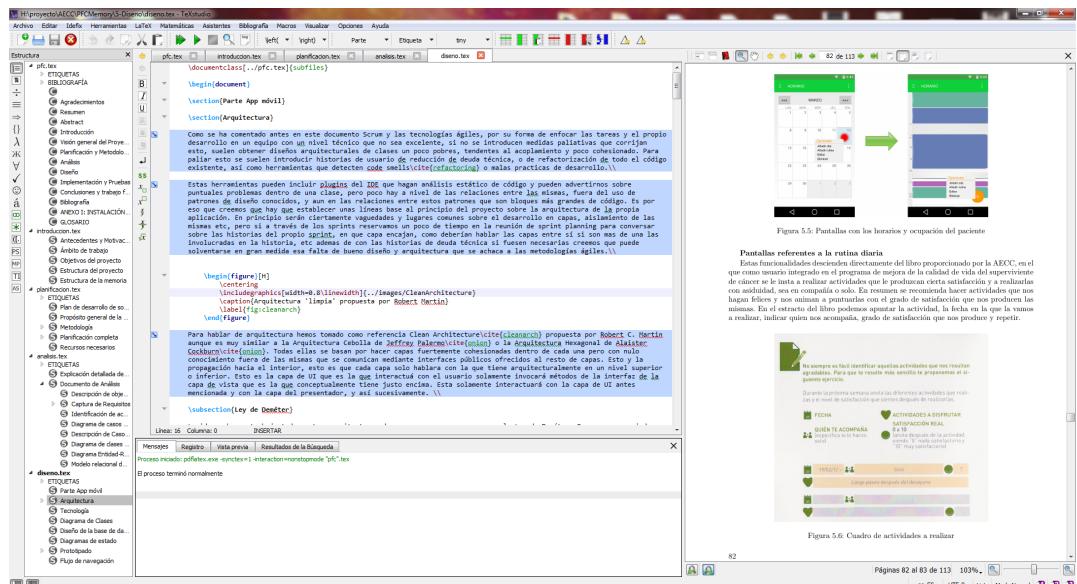


Figura 2.2: Vista general de TexStudio

TexStudio y LaTeX en general requieren de un aprendizaje, no es tan inmediata la generación de documentos con los formatos deseados como podría serlo Word, o incluso editores de texto online, des estilo a googleDocs, sin embargo, una vez que tienes el documento creado con las características deseadas, no es tan difícil de operar.



Figura 5.5: Pantallas con los horarios y ocupación del paciente

Figura 5.6: Cuadro de actividades a realizar

Puntualizar referencias a la rutina diaria  
Estos finalizan las descripciones directamente del libro proporcionado por la AECG, en el que como usuario integrado en el programa de mejora de la calidad se vela del superviviente que cumple su función de realizar actividades que le produzcan cierta satisfacción y a resultados concretos, siendo estos más eficientes. Es recomendable que las actividades que nos hagan felices y nos animan a puntualizar con el grado de satisfacción que nos producen las mismas. En el extracto del libro podemos apuntar la actividad, la fecha en la que la vamos a realizar, indicar si nos acompaña, grado de satisfacción que nos produce y repetir.

TexStudio, utiliza por debajo de todo esto miktex que es una aplicación que se encarga de mantener actualizados todos estos paquetes a la última versión y proporciona las funcionalidades que permiten que TexStudio genere los documentos.

Aquí podemos ver la configuración del documento, paquetes necesarios, márgenes, librerías de idioma,...

```
\documentclass[b5paper,10pt,twoside]{book}
\usepackage[inner=1.5cm,outer=2cm,tmargin=1.5cm,bmargin=2cm]{geometry}

\usepackage[utf8]{inputenc}
\usepackage[spanish]{babel}
\usepackage{subfiles}
\usepackage{fancyhdr}
\usepackage[nottoc]{tocbibind}
\usepackage[square,numbers]{natbib}
\usepackage{titling}
\usepackage[export]{adjustbox}
\usepackage{titlesec}
\usepackage{url}
\usepackage{lipsum}
\usepackage{graphicx}
\graphicspath{ {images/} }
\bibliographystyle{abbrvnat}
\usepackage{float}

\setcounter{secnumdepth}{3}
\pagestyle{fancy}
\fancyhf{}
\fancyhead[LE,RO]{Diario de un superviviente}

\fancyfoot[LE,RO]{\thepage}

\title{Proyecto Fin carrera}
\author{Fernando Santa Olaya Rodríguez \\
\and
Rubén Toquero González}
```

Figura 2.3: Configuración del documento

Una de las ventajas que nos han proporcionado LaTeX y TexStudio sobre el resto de los procesadores de texto ha sido el poder añadir sus archivos al repositorio, y poder tratar estos archivos de la misma manera que trataríamos cualquier archivo con una clase java o de otro tipo similar.

En principio el documento constaba de un .tex donde se realizaban todos los cambios para la generación del documento, sin embargo a medida que surgían los conflictos entre las versiones que subíamos al repositorio Git, decidimos particionar el .tex original en otros archivos del mismo tipo que fuesen llamados por el primero y que mantuviesen el formato y la configuración inicial.

Cada capítulo del libro pasó a ser un .tex independiente que se pudiese editar en TexStudio por separado y así minimizar los conflictos que iban surgiendo por la edición de la misma línea.

```
\chapter{Introducción}
\subfile{./1-Introduccion/introduccion}
\chapter{Visión general del Proyecto}
\subfile{./2-Vision/vision}
\chapter{Planificación y Metodología}
\subfile{./3-Planificacion/planificacion}
\chapter{Análisis}
\subfile{./4-Analisis/analisis}
\chapter{Diseño}
\subfile{./5-Diseno/diseno}
\chapter{Implementación y Pruebas}
\subfile{./6-Pruebas/pruebas}
\chapter{Conclusiones y trabajo futuro}
\subfile{./7-Conclusiones/conclusiones}
\chapter{Bibliografía}
\nocite{*}
\bibliography{bibliography}
\chapter*[ANEXO I: INSTALACIÓN Y MANUAL DE USUARIO ]
\subfile{./8-Anexoi/anexoi}
\chapter*[GLOSSARIO]
\subfile{./Glosario/glosario}
```

Figura 2.4: Capítulos dependientes del documento principal

El documento inicial con las configuraciones y paquetes necesarios para la generación del documento junto a los capítulos del libro, Bibliografía y Glosario, forman el total del documento que es la memoria del proyecto.

# CAPÍTULO 3

---

## Planificación y Metodología

---

### 3.1. Plan de desarrollo de software

Como ya se ha comentado anteriormente en esta memoria para acometer el desarrollo del producto software fin de este proyecto, se van a utilizar las denominadas metodologías ágiles cuyos principios generales vienen reflejados el Agile Manifesto [31], y de entre todas las metodologías que lo implementan utilizaremos Scrum, ya que nos permite desarrollar siempre sobre algo ejecutable y tiene una buena "pelea contra el tiempo."<sup>o</sup> "timeboxing" que al final de cada sprint hay que hacer una retrospectiva sobre lo que ya se ha construido y entregado.

### 3.2. Propósito general de la planificación

La planificación nos debe dar una cifra orientativa del esfuerzo a comprometer para acometer un desarrollo de un proyecto software. Pero debido a lo mencionado anteriormente sobre el manifiesto ágil, creemos que dar una cifra estimativa en tiempo es venturoso, más si queremos ceñirnos a él y más aún cuanto más a largo plazo sea la estimación. Por eso las metodologías ágiles suelen ocultar la referencia temporal de los desarrollos y estiman la complejidad de las tareas, sabiendo por el histórico, ya que los equipos deben ser fijos en el tiempo, la complejidad aproximada que un equipo dedicado a un proyecto puede acometer.

### 3.3. Metodología

Tradicionalmente se denomina metodología de desarrollo de software a un marco de referencia para estructurar, planificar, controlar y medir el proceso de desarrollo de software.[21]

En los últimos años se han popularizado diversas metodologías que rompen.<sup>en</sup> parte con los paradigmas anteriores a cambiar el punto sobre el que ponen el foco. Es en estas nuevas metodologías, que se han venido a denominar ágiles por su capacidad rápida de respuesta al cambio, donde vamos a encontrar la metodología que vamos a utilizar para el desarrollo del proyecto.

Como generalidades sobre las metodologías ágiles decir que se basan todas ellas en la formación de grupos auto organizados y multidisciplinares que colaboran para llevar a cabo

el proyecto. De hecho es esta colaboración y esta auto organización las piezas clave para que el equipo vea que tiene poder de decisión durante la vida del proyecto, y que de esta manera se implique mucho más en el éxito del mismo.

Para la realización del proyecto buscaremos como ya hemos comentado dentro las diferentes metodologías ágiles, hemos elegido SCRUM, por ser la que mejor se adapta a la continua pelea contra el tiempo que el equipo debe mantener.

Lo primero que debemos decir es que SCRUM es una metodología iterativa e incremental que promueve la auto organización de los equipos de desarrollo y un esquema de colaboración con el cliente, haciéndose responsable de que las prioridades, los requisitos, etc pueden cambiar por parte de este, y que es responsabilidad del propio equipo el asumir y responder a estos cambios.

### 3.3.1. SCRUM

SCRUM como metodología fue propuesto por Ikujiro Nonaka e Hirotaka Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica: Fuji-Xerox, Canon, Honda, Nec, Epson, Brother, 3M y Hewlett-Packard (Nonaka & Takeuchi, *The New New Product Development Game*, 1986).

Aunque esta forma de trabajo surgió en empresas de productos tecnológicos, es apropiada para proyectos con requisitos inestables y para los que requieren rapidez y flexibilidad, situaciones frecuentes en el desarrollo de determinados sistemas de software.

SCRUM define un conjunto de prácticas y roles, que pueden tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Abordar un proyecto mediante SCRUM lleva a la repetición o iteración de la unidad básica del proyecto denominada Sprint. Durante cada sprint, un periodo entre una y cuatro semanas (la magnitud es definida por el equipo y debe ser lo mas corta posible), el equipo crea un incremento de software potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar a menudo denominadas Historias de usuario. Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de Sprint Planning. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo conversa con el Product Owner buscando claridad y magnitud adecuadas para luego determinar la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint. Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos del sprint están congelados durante el propio sprint, pero podrían modificarse aquellos que aun estuviesen en el Product Backlog.

SCRUM permite la creación de equipos auto organizados impulsando la co-localización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto, ya que una de las bases de SCRUM en la colaboración y comunicación entre todos los roles de un proyecto.

Un principio clave de SCRUM es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan, y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Por lo tanto, SCRUM adopta una aproximación pragmática, aceptando que el problema no puede ser

completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes.

Las características más marcadas que se logran notar en SCRUM serían: gestión regular de las expectativas del cliente, resultados anticipados, flexibilidad y adaptación, retorno de inversión, mitigación de riesgos, productividad y calidad, alineamiento entre cliente y equipo, por último equipo motivado. Cada uno de estos puntos mencionados hacen que el SCRUM sea utilizado de manera regular en un conjunto de buenas prácticas para el trabajo en equipo y de esa manera obtener resultados posibles.

Los beneficios para las empresas más notables al utilizar SCRUM son

- Flexibilidad a cambios. Gran capacidad de reacción ante los cambiantes requerimientos generados por las necesidades del cliente o la evolución del mercado. El marco de trabajo está diseñado para adecuarse a las nuevas exigencias que implican proyectos complejos. Reducción del Time to Market. El cliente puede empezar a utilizar las características más importantes del proyecto antes de que esté completamente terminado.
- Mayor calidad del software. El trabajo metódico y la necesidad de obtener una versión de trabajo funcional después de cada iteración, ayuda a la obtención de un software de alta calidad. Mayor productividad. Se logra, entre otras razones, debido a la eliminación de la burocracia y la motivación del equipo proporcionado por el hecho de que pueden estructurarse de manera autónoma.
- Maximiza el retorno de la inversión (ROI). Creación de software solamente con las prestaciones que contribuyen a un mayor valor de negocio gracias a la priorización por retorno de inversión.
- Predicciones de tiempos. A través de este marco de trabajo se conoce la velocidad media del equipo por sprint, con lo que es posible estimar de manera fácil cuando se podrá hacer uso de una determinada funcionalidad que todavía está en el Backlog.
- Reducción de riesgos. El hecho de llevar a cabo las funcionalidades de mayor valor en primer lugar y de saber la velocidad a la que el equipo avanza en el proyecto, permite despejar riesgos efectivamente de manera anticipada.

Como puntos flacos tanto de SCRUM como de muchas de las metodologías ágiles podemos observar:

- Fuerte dependencia de las personas y en sus habilidades comunicativas y de auto organización.
- Participación fuerte e intensa de todos los implicados en el proceso, con los riesgos por parte de un cliente poco participativo.
- Falta de documentación o mucha laxitud en la que existe.

Aunque más adelante en este documento se profundiza en la metodología la figura 3.1 muestra una imagen que puede servir como resumen de todos los conceptos roles y rituales de SCRUM. Cabe destacar que poco a poco iremos desgranando los puntos más importantes de la misma, mientras proponemos nuestra práctica o solución encontrada a la misma.

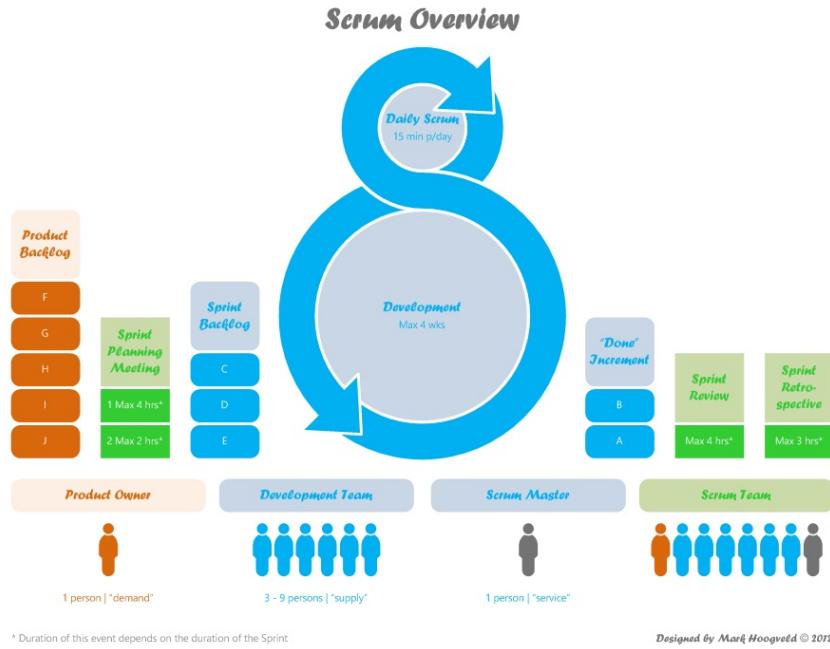


Figura 3.1: Resumen de Scrum.

### 3.3.2. Roles y responsabilidades

Los roles implicados en el proceso de desarrollo y sus responsabilidades de la app son:

#### Product Owner

El Product Owner representa la voz del cliente, que puede ser el cliente mismo o alguien en su nombre. Se asegura de que el equipo trabaje de forma adecuada desde la perspectiva del negocio. El Product Owner escribe historias de usuario, las prioriza, y las coloca en el Product Backlog, de hecho se suele decir que el Product Owner es el dueño del Product Backlog. En nuestro caso este rol lo llevaban a cabo varias personas con las que hemos tenido contacto en la AECC. Lo ideal hubiera sido que fuese solamente una persona y que fuese esa misma persona quien se entrevistase con el resto de la organización, para poder desarrollar un lenguaje común y una relación con mas confianza entre esta y el equipo.

#### Scrum Master (o Facilitador)

El Scrum es facilitado por un Scrum Master, cuyo trabajo primario es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint. El Scrum Master no es el líder del equipo (porque ellos se auto-organizan), sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. El Scrum Master se asegura de que el proceso Scrum se utilice como es debido. El Scrum Master es el que hace que las reglas se cumplan. Quizá este es el puesto clave de Scrum ya que se encarga de que el equipo se centre en producir aquello que el Product Owner ha marcado. En nuestro caso este rol ha sido una surte de Frankenstein ya que para algunas tareas ha sido el personal de la AECC, para otras ha sido el tutor y en algún caso nosotros mismos saliendo del rol de equipo. También lo ideal es que

fuese la misma persona siempre ya que es un puesto en el que no la experiencia puede aportar soluciones o anticiparse mucho a los problemas.

### **Equipo de desarrollo**

El equipo tiene la responsabilidad de entregar el producto. Es recomendable un pequeño equipo de 3 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc).

A veces se definen más roles que pueden complementar en alguna fase a estos, o ayudar en tareas administrativas, pero formalmente quedan un poco fuera de la metodología por mas que sean muy necesarios en las empresas hoy día. Pero en nuestro caso al colaborar de manera altruista y al ser además un proyecto académico creemos que el resto de roles en la propia metodología están de más.

### **3.3.3. Eventos y rituales**

Dentro de la metodología de SCRUM existen varias reuniones o rituales, tras las que se esconde alguno de los principios o prácticas de las metodologías ágiles[31], dentro de que cada equipo debe adaptar la metodología a su idiosincrasia y condiciones particulares, se recomienda en la medida de lo posible ajustarse a las mismas ya que la ganancia que reportan está comprobado que refuerza todos los esfuerzos de la metodología.

#### **Sprint**

No es en sí un evento o un ritual, simplemente es la medida de tiempo en los que se dividen los proyectos a acometer por el equipo, así el Sprint es el período en el cual se lleva a cabo el trabajo en sí. Se recomienda que la duración de los sprints sea constante y definida por el equipo con base en su propia experiencia. Se puede comenzar con una duración de sprint en particular (2 o 3 semanas) e ir ajustándolo con base en el ritmo del equipo, aunque sin relajarlo demasiado. Al final de cada sprint, el equipo deberá presentar los avances logrados, y el resultado obtenido es un producto potencialmente entregable al cliente. Asimismo, se recomienda no agregar objetivos al sprint o sprint backlog a menos que la falta de estos objetivos amenace al éxito del proyecto. La constancia permite la concentración y mejora la productividad del equipo de trabajo. Para nuestro caso nos definimos un sprint de dos semanas pero en el que no contabilizariamos más que unas 15 horas de trabajo por miembro del equipo en él.

#### **Daily Scrum o Stand-up meeting**

Cada día de un sprint, se realiza la reunión sobre el estado de un proyecto. Esto se llama daily standup o Stand-up meeting. El scrum tiene unas guías específicas:

- La reunión comienza puntualmente a su hora.
- Todos son bienvenidos, pero sólo los involucrados en el proyecto pueden hablar. (Esto es el equipo y el scrum master).
- La reunión tiene una duración fija de 15 minutos, de forma independiente del tamaño del equipo.
- La reunión debe ocurrir en la misma ubicación y a la misma hora todos los días.
- Durante la reunión, cada miembro del equipo contesta a tres preguntas:

1. ¿Qué has hecho desde ayer?
2. ¿Qué es lo que harás para mañana?
3. ¿Has tenido algún problema que te haya impedido alcanzar tu objetivo? (Es el papel del Scrum Master recordar y anotar estos impedimentos para su posterior análisis y subsanación si procede).

Para nuestro caso y debido a las particularidades del proyecto definimos "dailies" los martes, miércoles y jueves de la semana. Siendo estos la mayor parte de las veces una videoconferencia vía hangouts o si la red no estaba operativa un chat en la misma aplicación.

### **Reunión de Planificación del Sprint**

Al inicio de cada ciclo de Sprint, se lleva a cabo una reunión de planificación del Sprint. En esta reunión debe estar el equipo al completo, el scrum master y el product owner. Se pretende:

- Seleccionar qué trabajo se llevará a cabo por el equipo durante el sprint
- Preparar, con el equipo completo, el Sprint Backlog o sea las tareas del sprint clarificando el Product Owner las posibles dudas o inconsistencias que el equipo detecte en la redacción de las tareas, este se debe extraer de las tareas más prioritarias del Product Backlog, de acuerdo con el Product Owner.
- Identificar y comunicar cuánto del trabajo es probable que se realice durante el actual Sprint.

Realizarse esta planificación en ocho horas como tiempo límite. Debido a que nuestros sprints eran mas cortos que de lo normal la reunión duraba una hora o dos como mucho. Y tenía lugar lunes alternos en la sede de la AECC en Madrid con uno de los integrantes del equipo, y el sábado de esa semana se completaba con ambos miembros del equipo en Valladolid redactando e introduciendo las historias en la herramienta Trello, de la que hablaremos más adelante, hasta que el propio personal de la AECC se vio con la habilidad para llevar a cabo esta tarea.

### **Reunión de Revisión del Sprint**

Al finalizar cada sprint se organiza esta reunión en la que el equipo presenta el trabajo desarrollado durante el sprint en lo que se denomina Incremento de Producto Potencialmente Entregable. Durante esta reunión tiene lugar la presentación del ejecutable en lo que se denomina Demo. Así pues las principales actividades de esta reunión serían:

- Revisar el trabajo que fue completado y no completado
- Presentar el trabajo completado a los interesados, entre ellos el product owner y que estos de su aprobación al mismo o presenten sus quejas o matices (alias "demo"), pero teniendo en cuenta que los estados del trabajo son binarios, está acabado o no está acabado, no hay porcentajes intermedios.
- El trabajo incompleto no puede ser demostrado

Para esta reunión y dependiendo del tamaño del sprint se dispondrá de cuatro horas como límite. Para esta actividad las semanas que había nuevo sprint se hacían ambas reuniones seguidas, esta era una validación relativamente informal de la misma por el interlocutor de la AECC aunque más tarde distribuían el software al resto de implicados para validar más exhaustivamente la funcionalidad entregada por el equipo.

### **Retrospectiva del Sprint**

Después de cada sprint, se lleva a cabo una retrospectiva del sprint, en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado. El propósito de la retrospectiva es realizar una mejora continua del proceso. Esta reunión tiene un tiempo fijo de cuatro horas. Los sábados antes de comenzar a puntualizar el nuevo sprint por parte de un miembro del equipo al otro se realizaba una pequeña valoración del último sprint en términos de velocidad, calidad entregada, satisfacción del personal de la AECC, para tratar de mejorar en futuros sprints.

## **3.4. Planificación completa**

Para realizar la planificación completa de la app debemos antes introducir algunos conceptos a la estimación en el contexto de las metodologías ágiles, como son velocidad del equipo, historias de usuario, puntos de historia, etc.

### **3.4.1. Puntos de Historia**

Los puntos de historia son una medida de la complejidad de la tarea a abordar. Se utilizan para evitar determinados efectos asociados a estimar en tiempo y que repercuten en baja calidad de código y tensiones innecesarias en el equipo. De esta manera el equipo fija una historia de usuario tipo y conocida con una medida arbitraria de puntos de historia, y el resto de historias se evalúan en torno a esa misma, de tal manera que si por ejemplo a nuestra tarea de usuario modelo le fijamos una media de puntos de historia de 3, si después tenemos una tarea que tiene el doble de complejidad, deberíamos estimarla en 6.

A este respecto decir que para estimar la complejidad de las tareas existen varias maneras, casi todas basadas en variantes del planning poker, consistente en que todos los miembros del equipo establecen en secreto un valor de puntos de historia después de que la tarea a estimar sea leída, en su caso aclarada y comprendida por todos los miembros del equipo. Estos valores no son libres y suelen estar relacionados con valores similares a la serie de Fibonacci (1, 2, 3, 5, 8, 13, 20) a los que se elimina algún valor 1, 3, 5, 8, 13, 20 suelen ser los valores mas usados por ser 1 una tarea quasi trivial, 3 se suele asignar a la tarea base de estimación y 20 se suele usar para indicar al product owner que debe partir la tarea por ser esta demasiado grande. Al realizar la elección de la estimación de cada integrante del equipo en secreto se trata de eliminar el efecto lidera, si existe discrepancia entre miembros del equipo el scrum master puede comunicarles a que expliquen los motivos por los que asignan dicho valor, tras lo cual se repetirá la votación. Esto se hace así para poner de relieve complejidad oculta que nadie ha detectado o viceversa. En ocasiones si la discrepancia es un paso en la serie, el scrum master puede en secreto otorgar peso a las opiniones de los integrantes, si considera su nivel muy desigual, lo que atenuaría esa discrepancia en las estimaciones. Para la medida del proyecto tomamos como medida que una historia en la que hay una pantalla y se interacciona con la misma y se graba un dato en BBDD de manera directa tenía el valor de 3 PH.

### **3.4.2. Historias de Usuario, Historias Épicas**

Los requisitos en la mayoría de las metodologías ágiles se condensan en lo que se viene a llamar historias de usuario. Formalmente hablando una historia de usuario es una represen-

tación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario o del negocio del usuario, esto es esencialmente NO técnico. Describe una funcionalidad que, por sí misma, aporta valor al usuario.

El formato físico o registral de estas historias de usuario suelen ser tarjetas de papel/cartón con espacio para colocar post its de tareas asociadas a la historia, y estas pueden ser tan complejas o simples como el equipo, scrum master y product owner decidan entre ellos, pero como mínimo deberían tener una descripción del tipo como XXX quiero YYY para ZZZ y una estimación en puntos de historia. Pero no hay problema si se añaden un criterio de aceptación de la historia, una identificación de la misma, un título, etc

Idealmente las historias de usuario deberían escribirse por el product owner, pero más idealmente aún es si se escriben en conjunto y conversando sobre el aporte de la misma al producto final. En el sprint planning se decide cuales de las historias más arriba en la pila del product backlog entraran a realizarse en este sprint, clarificando el product owner las dudas que sobre las mismas puedan surgir al equipo.

La figura 3.2 muestra lo que podría ser una típica tarjeta de historia de usuario, con los datos más importantes y alguno que en esa organización tenga sentido, pues recordamos que Scrum son guías y que cada equipo debe encontrar la manera de adaptarse lo mejor posible a la metodología pero también de adaptar esta a unas formas en las que se sientan cómodos.

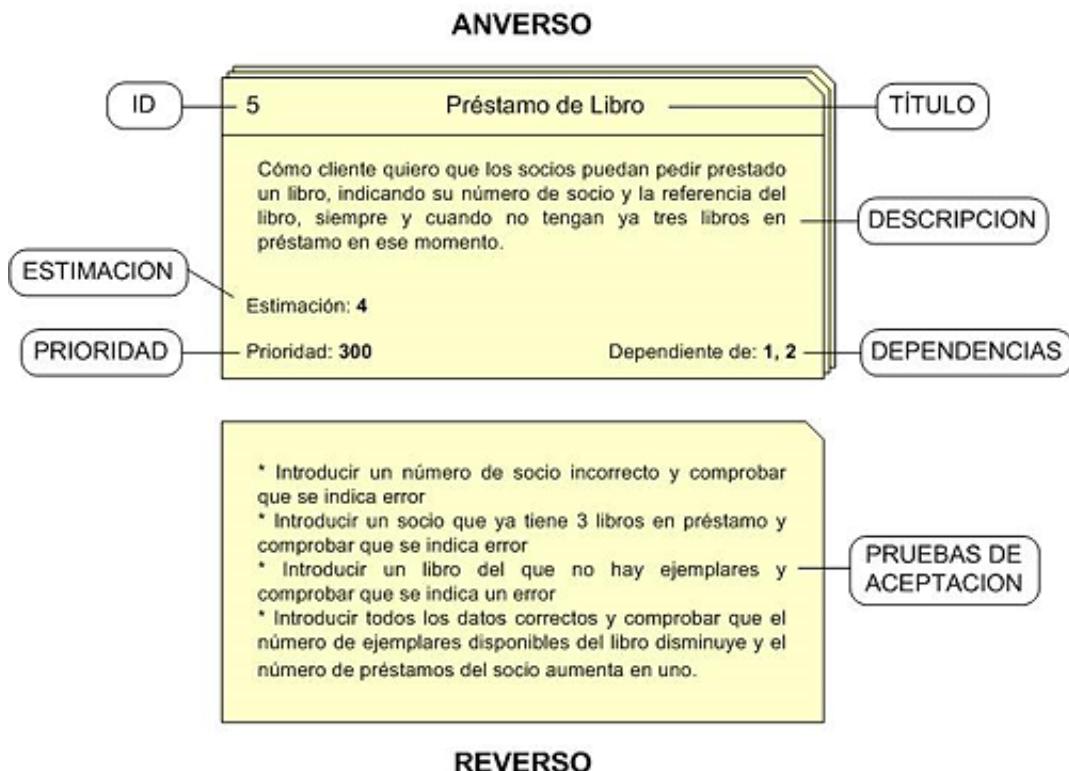


Figura 3.2: Ejemplo de historia de usuario

Al inicio de un proyecto y para mitigar el temor del folio en blanco suele dividirse el proyecto en lo que se denominan Historias Épicas, que no son más que historias de usuario

muy grandes y por tanto tendrán una estimación en puntos de historia muy grande y que deben ser divididas en historias más manejables. Esto sirve para que ninguna funcionalidad importante se queda sin estimar o tener en cuenta durante las reuniones de preparación. Estas historias épicas una vez identificadas pueden quedarse en el estado de épicas en el backlog hasta que el product owner decida que deben ser acometidas momento en el que deberían partirse en historias más manejables y estimarse estas cada una en sus puntos de historia, quedando la suma de estas aproximadamente en el valor que tenía asignado la historia épica.

Debido a la naturaleza del proyecto el equipo estimó que no era necesario comenzar con historias épicas de grandes funcionalidades, sino que con dos historias por funcionalidad en principio era el tamaño justo para poder acometerlas. El formato de historia que optamos fué el de describir un título, un texto en el modo COMO x QUIERO Y PARA Z, la estimación. Además contar con un formato extendido en el que se mostraba un caso de aceptación simple para la funcionalidad.

### 3.4.3. Tipos de historias

Además de las historias de usuario en ocasiones surge la necesidad de realizar tareas técnicas, o para mejorar el software a futuro, incluso de reparación de errores por eso no todas las historias o puntos de historia de un sprint a veces son de funcionalidades visibles para el product owner pese a que estas deberían ser las mínimas y abordarse en el tiempo que pudiese sobrar en un sprint, ya que estas a veces no aportan nada al product owner.

Para ello vamos a introducir el término de Deuda Técnica. Esta es un concepto relacionado con la calidad de código y el esfuerzo y retorno de la inversión. En principio las metodologías ágiles buscan la calidad máxima que un equipo puede aportar, pero a veces esta llegar a esta calidad está reñida con el plazo o el esfuerzo que el equipo debe aportar para llegar a la misma y el tiempo que el product owner decide comprometer en el proyecto. Esta deuda técnica es consciente pero ocurre a veces que una tarea es acometida por algún miembro reciente del equipo que sabe que no está al máximo de calidad pero que ese miembro no puede elevarlo, por desconocimiento del entorno técnico, del contexto del proyecto, etc esa deuda suele ser inconsciente.

Por todo esto a veces se introducen historias de reducción de deuda técnica, en algunos equipos si la deuda es consciente al terminar una tarea generan otra en la sombra para poder seguir el aumento de la deuda. Estas tareas suelen ser eminentemente técnicas o como mucho de mejora de UI/UX pero hacen que la calidad del producto crezca y que a largo plazo se introduzca cada vez menos deuda. Estas historias si bien no constituyen un avance a corto plazo del proyecto pueden suponer un ahorro en el largo plazo o que las historias restantes sean más fáciles de acometer. Por eso aunque no aporten valor per sé al proyecto deberían estimarse y al menos contabilizar la mitad de sus puntos de historia en el sprint de proyecto y quizás todos ellos de cara al cálculo de la velocidad del equipo.

También puede suceder que una historia validada por el equipo y el equipo de QA resulte errónea o con defecto en cierta situación no común, en ese caso se introduciría una historia de resolución de defecto. Estas historias deben abordarse en el sprint siguiente al que se detectan y se debe consensuar con el product owner si deben sacarse historias del sprint para no sobrecargar al equipo o no. De todos modos lo que nunca deben hacer es contabilizarse sus puntos de cara a la revisión de sprint ni al cálculo de la velocidad del equipo, pues aunque están resolviendo complejidad, se supone que ya antes la resolvieron y contabilizaron unos

puntos de historia que no estaban conformes y por tanto no aportaban al proyecto.

En principio para nosotros las historias normales serían blancas, mientras que reservábamos el amarillo para las historias que suponían un defecto y el azul para aquellas que eran de deuda técnica conocida cuando se introducía.

#### **3.4.4. Velocidad de un equipo**

El concepto de velocidad del equipo es la cantidad de puntos de historia que se acometen en un sprint. Es un valor que depende de la experiencia pasada del equipo y que suele calcularse manteniendo un registro histórico de los puntos resueltos por el equipo en los anteriores sprints. Es una medida que depende de que el equipo sea una unidad más o menos constante a lo largo del tiempo.

#### **3.4.5. Planificación de los Sprints**

Teniendo en cuenta las historias de usuario, desglosas y tratadas con más detalle en el capítulo siguiente, y la estimación de estas que podemos ver en el listado posterior, vemos que el total de puntos de historia del proyecto puramente técnico son 245, haciendo un par de sprints iniciales de setup de proyecto y para abordar tareas muy generales y las primeras historias de usuario, obtuvimos que la velocidad del equipo era de unos 12-14 puntos de historia, valor que se modificó posteriormente a 20 al final del proyecto, por tanto estimamos que fueron 16 sprints que al término del proyecto con sprints de deuda técnica y de resolución de algún error se quedó en 272 lo cual no se había alejado en exceso de la estimación inicial.

Decir que en los contratos propios de Scrum, lo que suele hacerse es hacer una estimación del proyecto inicial y contratar al equipo por un numero de sprints suficientes para el desarrollo del mismo. Pudiéndose después alargar estos sprints o acortarse si se consigue el éxito con antelación. En este tipo de contratos lo que suele hacerse es contratar un tiempo determinado y en ese mismo tiempo decidir qué funcionalidades implementar, pudiéndose cortar el proyecto al término de un sprint si el product owner considera que el producto tiene la suficiente madurez, o por contra modificar, ampliar o repensar esas funcionalidades en sprints hasta que se considere que tiene lo que desea.

### 3.4.6. Diagrama de sprints

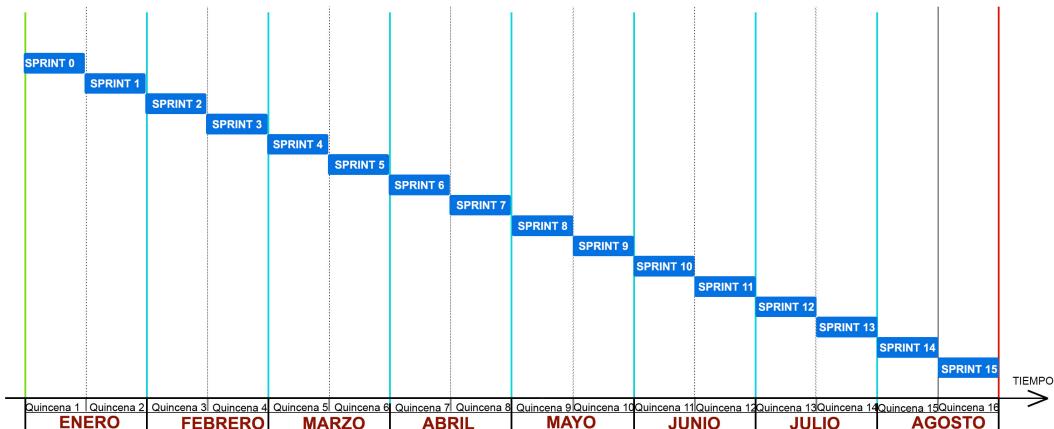


Figura 3.3: Diagrama de sprints en el tiempo

A continuación vamos a explicar de manera breve la composición de los sprints en términos de tiempo y funcionalidades.

Los miembros del equipo en un principio, han dispuesto de unas 6 horas semanales para el desarrollo del proyecto técnico, dado que su situación actual es 'trabajando' disponen de tiempo limitado para su realización. Este tiempo se divide entre 3 días laborables, los otros dos días entre Lunes y viernes, se han dedicado a comunicaciones con la AECC, reuniones y/o viajar. El tiempo de dedicación al final del proyecto ha variado, incluyendo fines de semana en la persecución de la consecución de los objetivos a unas 10 horas semana, manteniéndose las reuniones y viajes pero ampliando el horario de trabajo.

#### Resumen de los sprints

Cada SPRINT exceptuando los primeros empieza con una sprint plan donde se acuerdan las historias de usuario que van a entrar en el mismo y acaba con una reunión con el PO o en su defecto, una comunicación, sea a través del email, teléfono,...

- **SPRINTS 0 y 1** Utilizados para la toma de contacto entre los alumnos y el tutor y entre los alumnos y la AECC para establecer las bases del proyecto.
- **SPRINT 2** Preparación de los entornos de desarrollo, creación de historias,...
- **SPRINT 3** Aplicación base funcional inicial.
- **SPRINT 4** Primeros diseños y propuestas, prototipos, solo aspecto.
- **SPRINT 5** Estructura de la aplicación inicial, prototipo.
- **SPRINT 6** Diseño avanzado de la aplicación.
- **SPRINT 7** Cambio de la arquitectura de la aplicación hacia una arquitectura limpia, no todas las entidades implicadas. Capa de Datos y de dominio.

- **SPRINT 8** Deuda técnica, solución de problemas derivados de los cambios del anterior sprint.
- **SPRINT 9** Propagación de la arquitectura al resto de las entidades, Capa de datos y dominio.
- **SPRINT 10** Base de datos y flujo de la aplicación.
- **SPRINT 11** Capa de presentación.
- **SPRINT 12** Re-diseño de la aplicación de acuerdo al Dossier de la AECC.
- **SPRINT 13** Deuda técnica problemas en la capa de presentación (inclusión del resto de entidades).
- **SPRINT 14** Finalización de la documentación técnica.
- **SPRINT 15** Finalización de la documentación técnica y deuda técnica.

### 3.5. Recursos necesarios

Los recursos designados para la realización completa de este proyecto serán los siguientes:

#### Recursos humanos

- El equipo, en este caso los alumnos que defienden este proyecto.
- El scrum master, como se ha mencionado este papel esta repartido entre el tutor, personas involucradas desde la AECC y en ocasiones alguno de los alumnos.
- El product owner, que ya se ha dicho antes en este documento son varias personas dentro de la AECC.

Determinar el esfuerzo del rol equipo en este proyecto es sencillo, pero el resto de roles es más complejo el computar las horas de dedicación a este proyecto de cara a realizar una planificación, de tal manera que se estimará que el rol scrum master sea un 30 % del tiempo consumido por un miembro equipo y por contra el rol de product owner se estimará en un 25 % de la anterior medida.

#### Recursos Software

Los recursos software para esta iteración serán:

- PowerShell y GitHub desktop para el control y visualización del repositorio de Git.
- TexStudio + ShareLatex herramienta de elaboración de documentos PDF.
- Mktex como herramienta gestora de los paquetes que utiliza TexStudio para formar el documento.
- Adobe Photoshop CS6 para realizar ilustraciones e imágenes.
- StarUML y Enterprise Architect para modelado y diagramas.
- Dropbox y GitHub como herramientas para compartir código y archivos.

- Android Studio como IDE (Integrated Development Environment o entorno integrado de desarrollo) para el desarrollo del código Android necesario.
- NotePad ++ como apoyo en la visualización de documentación.
- Trello como gestión de la metodología SCRUM.
- Atom como herramienta de visualización de código y apoyo a AS.
- Word revisión y preparación de documentos burocráticos.
- SQLite como sistema gestor de bases de datos en el dispositivo móvil

### Recursos Hardware

Los recursos hardware para esta iteración serán:

- Dos ordenadores portátiles personales
- Un ordenador Personal de sobremesa
- Varios móviles con SO Android
- Memoria USB para el intercambio de datos
- Cables USB - Micro USB (Debug)



# CAPÍTULO 4

---

## Análisis

---

Para la fase de análisis dentro de las metodologías ágiles no existen preceptos firmes a cumplir, pero dado que se valora mucho la comunicación y las relaciones interpersonales, se establece que debería haber una serie de reuniones previas al inicio de los sprints para que el equipo pueda obtener contexto del proyecto y del dominio del mismo, así como ir escribiendo las historias de usuario y llenar y priorizar el product backlog. En estas reuniones idealmente asistirían tanto el product owner, el scrum master y el equipo, pero con el product owner, el scrum master y aquel del equipo que mejor pueda ayudar a escribir las historias puede ser suficiente, recordemos que no se está estimando, simplemente se escuchan las necesidades del product owner y estas se traducen en funcionalidades y estas a su vez en requisitos mediante historias de usuario.

Para acometer el trabajo y dividirlo en trozos más manejables en estos casos se suele recurrir a escribir historias épicas, que más tarde ya si pueden particionar el scrum master y el equipo para posteriormente remitirlas al product owner para que este exprese su conformidad o sus matices a las historias de usuario ya redactadas en piezas manejables y que priorice el product backlog inicial, teniendo en cuenta que tras la estimación inicial puede haber algún reordenamiento para aprovechar mejor los tiempos del sprint.

Una vez las historias tienen una forma más o menos fija, recordemos que uno de los fines del agilismo es responder al cambio y más aún al cambio en los requisitos, se hace una reunión de estimación de las mismas. Como ya se ha comentado existen varias maneras de estimar la complejidad que suponen las historias de usuario, casi todas basadas en variaciones del poker planning que se explicó anteriormente.

En el momento de redactar las historias de usuario conviene que de algún modo alguien, típicamente el scrum master o el product owner, se encargue de recordar los grandes objetivos del proyecto. Estos objetivos son los objetivos claros, concisos y lo más sencillos posibles que mueven al cliente a dar comienzo al proyecto. Todas las historias de usuario, así como las acciones que se lleven a cabo, deben alinearse con estos objetivos.

Para llevar a cabo todo este proceso se llevaron a cabo varias reuniones con el personal de la AECC, se cruzaron correos con historias redactadas, se conversó vía Hangout, Skype, Whatsapp, etc se volvió a reunir las veces que hicieron falta, hasta que dicho personal, que en este caso hacia el papel de product owner, se sintió lo suficientemente cómodo con el redactado inicial de las historias de usuario y su priorización, sabiendo que después estas podrían modificarse.

## 4.1. Explicación detallada de las funcionalidades de la aplicación

Este sería el proceso donde algún miembro del equipo ganaría contexto por parte del product owner, al este describirle lo que quiere, vamos a pasar al redactado final de las necesidades que se expresaron, pues debido a la inexperiencia de los miembros de la AECC en el mundo de la movilidad el punto de partida del proyecto estaba mucho más alejado de aquí. Esto en el argot del agilismo se llama educar al cliente, y como siempre se basa en la comunicación, las relaciones interpersonales y en la confianza mutua.

### USO SIN REGISTRO

En todo momento se debe poder usar la app sin hacer ningún tipo de registro previo. Solamente algunas funcionalidades específicas necesitarán del mismo. Para ello los datos serán almacenados en el dispositivo, sin perjuicio de que puedan una vez registrado ser compartidos en 'la nube' y se carguen en varios dispositivos.

### PRINCIPAL

Este será el punto de entrada habitual a la aplicación. salvo la primera vez que se ofrecerá la posibilidad de realizar un pequeño tutorial en la misma. Al entrar en la misma se obtiene un pequeño resumen con los eventos de todo tipo próximos en el día de hoy o un texto que nos informe de la ausencia de algún tipo de evento para el día de hoy.

### PERSONAJES

Una pequeña lista de contactos importantes relacionados con la enfermedad, como puedan ser el oncólogo, el psicólogo, familiares de especial relevancia, etc.

Como toda lista de contactos podrán añadirse, modificarse, borrarse y consultarse cada uno de los ítems que la componen de una manera sencilla, siguiendo los lineamientos del SO en cuestión.

Por cuestiones de practicidad y simplicidad, si se añade un contacto o se modifica en esta parte, lo hará también en la app general de contactos del SO. Esto es así para que en cualquiera de las listas el usuario pueda encontrar al contacto que busque y no se sienta desorientado manejando varias listas de contactos en el dispositivo móvil.

### CITAS

Es una funcionalidad para introducir citas médicas en un calendario. En las mismas se podrá añadir al profesional que nos atiende si es que se conoce. Poco después de una cita se generará una notificación preguntando por la misma, en la que se preguntará si la cita ha generado nuevas citas, momento en que se pasará a la pantalla para crear una nueva cita, si ha habido cambios en la medicación, yendo a la lista de medicamentos para añadir o cambiar la dosis de los mismos. Poco antes de la cita y siguiendo las pautas marcadas en el folleto se guiará al usuario en las actividades o como afrontar la cita. Lista de citas Nombre de la cita, fecha y hora.

Desde el propio ítem de la lista se podrán añadir medicamentos, personas, síntomas y/o pruebas, al pulsar sobre la opción correspondiente nos llevará al listado de estas opciones añadiéndose así.

## AVATAR

En esta funcionalidad se permite crear un perfil de registro al usuario de la app para acceder a alguna funcionalidad mas, como compartir los datos entre dispositivos, mediante el uso de un servicio de back end que almacene los nuevos datos cuando se disponga de conectividad. El acceso a tratar con un agente como parte del piloto del proyecto, pudiéndole dejar alguna pregunta que desde la parte servidora se encargara de responder.

## RUTINA

La rutina es una vista semanal de actividades programadas y de ritmo de vida. Se pueden introducir actividades reiterativas y en que consisten, si se hacen solo o en grupo y la satisfacción que el usuario estima que producen. De vez en cuando aparecerán notificaciones que pregunten si se realizo o no una actividad determinada, si esta causo satisfacción etc. Puede marcarse en este horario la hora en la que el usuario se levanta y acuesta para excluir notificaciones en ese horario de descanso.

Revisar esta página <https://code.google.com/p/yadvie/>

## MEDICACIÓN

Es un gestor de aquellos medicamentos que debe tomar el enfermo y en que cantidades, así como histórico de las tomas de los mismo o de alguno que no tengan prescrito como pudieran ser analgésicos, antidepresivos, etc

La app debe avisar con una notificación de la necesidad de toma de aquellos medicamentos que se marquen como prescritos por el facultativo, a partir de la posología indicada para el mismo.

## PRUEBAS

Es una funcionalidad para llevar un registro de las pruebas que se han realizado al paciente clasificadas por tipos y llegado el caso hacer seguimiento de los valores clave en cada prueba diagnostica o análisis Se podrán adjuntar fotografías de los resultados en papel si es que se dispone de ellos para disponer de un pequeño registro documental”.

## SÍNTOMAS y ANOTACIONES

Esta funcionalidad cubre anotaciones de síntomas, sensaciones y cualquier cosa que el paciente crea conveniente anotar. Antes de una cita le recordará que tiene anotaciones por si quisiese comentárselas al profesional y también permitirá pasar una anotación para la siguiente cita.

## INFORMACIÓN ÚTIL

Una sección con los consejos de vida sana, meditación, relajación etc que también disponen en el folleto pero con un enfoque más accesible y con algún tema relajante para facilitar la relajación del usuario.

## PARTE SERVIDORA

El back end almacenara los datos que los usuarios registrados envíen desde sus dispositivos y los devolverá hacia aquellos que no se encuentren sincronizados. Ademas permitirá a un rol supervisor provincia asociar a un usuario registrado a un agente que desde ese momento velara por el. Dicho agente podrá comunicarse con el usuario de manera directa y también respondiendo a sus preguntas.

## 4.2. Documento de Análisis

Como se ha comentado anteriormente es difícil capturar todos los requisitos y el contexto de la aplicación a partir de una reunión, en nuestro caso mantuvimos al menos cuatro reuniones presenciales y al menos casi otras tantas por medios telemáticos al residir en provincias diferentes y además contamos con la ayuda del folleto que querían "digitalizarMis Cuidados, diario de salud para supervivientes de cáncer".

Una de las ventajas de utilizar este primer tipo de metodología, es el de poder adaptar de una manera mucho más eficiente los recursos a las fechas y a los requisitos, y poder modificar estos requisitos de manera 'ágil' debido al panorama cambiante que podemos tener de cara al cliente.

### 4.2.1. Descripción de objetivos de manera detallada

**OBJ-001** La aplicación deberá poder usarse en un dispositivo sin registro y sin conexión.

**OBJ-002** La aplicación debe ser fácil de usar para usuarios no avanzados y personas mayores.

**OBJ-003** Cualquier opción que se pueda realizar en el folleto debe estar cubierta por alguna funcionalidad de la app.

**OBJ-004** Se hará énfasis en que el registro proporciona grandes ventajas, a pesar de poder usarse sin él.

**OBJ-005** El estilo gráfico de la app sera coherente con el marcado por el folleto de la AECC.

**OBJ-006** En definitiva la app será algo útil por su funcionalidad pero también por su usabilidad y facilidad de manejo.

### 4.2.2. Captura de Requisitos

Como consecuencia de las diferentes reuniones realizadas con el cliente se elabora un documento de especificación de requisitos con el propósito de describir las funcionalidades necesarias para poder validar el producto final.

Estos requisitos se escriben aquí como historias de usuario con un id, un título y una descripción para ser implementadas.

#### 4.2.2.1. Historias de usuario

##### **Historia 1: Navegación**

Como usuario quiero poder acceder al menú de navegación desde cualquier punto del primer nivel de la misma, mediante la pulsación en el ícono de tres líneas en la izquierda de la pantalla, para poder navegar entre las distintas opciones de la app. Dejando en el resto de niveles de navegación una flecha para volver al nivel 1.

## **Historia 2: Registro**

Como usuario quiero poder registrarme, acceder y modificar mas tarde estos datos para poder acceder a las funcionalidades de sincronización y consulta con agente.

## **Historia 3: Personas**

Como usuario quiero poder añadir, consultar, modificar y borrar uno de los contactos de la lista de personas relacionadas desde la app y que estos cambios se reflejen también en la app de contactos para poder tener un listado de personas importantes.

## **Historia 4: Citas**

Como usuario quiero poder añadir, consultar modificar y borrar citas medicas con los datos de lugar, fecha, hora, especialista para poder tener un control de mis citas y que estas pasen a la app calendario del dispositivo.

## **Historia 5: Notificaciones citas**

Como usuario quiero que la app me avise de las citas próximas en la manera en la que el folleto lo indica para poder preparar la cita de cara a sacarle el máximo partido.

## **Historia 6: Principal**

Como usuario al entrar en la aplicación quiero que me muestre el siguiente evento hoy si lo hubiese de cada una de las categorías: citas, medicación, horarios, etc para poder tener una idea aproximada del día.

## **Historia 7: Rutina**

Como usuario quiero poder añadir, consultar, modificar y borrar elementos en el calendario semanal de rutinas para poder planificar mis actividades semanales.

## **Historia 8: Notificaciones Rutinas**

Como usuario quiero que me la app me avise del comienzo de alguna actividad del horario semanal, así como que me pregunte por algunas de ellas para recabar información de mi estado de ánimo y como me favorece la rutina.

## **Historia 9: Medicación**

Como usuario quiero poder introducir, modificar, consultar y borrar información sobre medicamentos que el profesional me prescriba, así como de otros tomados de forma puntual pero de los que tengo registro de que son para poder llevar un histórico de lo que he tomado o poder comentárselo al profesional en las siguientes citas médicas.

## **Historia 10: Notificaciones Medicación**

Como usuario quiero que la aplicación me notifique de que debo tomar una determinada medicación para no saltarme ninguna de las prescripciones del profesional. Además si tomo alguna medicación no prescrita quiero que me avise de la misma poco antes de la próxima cita en el tiempo y que me permita pasarla de cita.

## **Historia 11: Pruebas**

Como usuario quiero que la aplicación me permita añadir, modificar, consultar y borrar la información referente a una prueba diagnostica o de análisis para poder disponer de un histórico de los valores a tener en cuenta.

## **Historia 12: Síntomas y anotaciones**

Como usuario quiero que la aplicación me permita añadir, modificar, consultar y borrar anotaciones libres para consultar en las citas con el profesional para poder comentar con el mismo la gravedad o no de los mismos.

#### **Historia 13: Notificaciones Síntomas**

Como usuario de la aplicación quiero que me notifique poco antes de una cita que existen síntomas o comentarios que pueden tener que comentarse con el profesional o poder moverlos a la siguiente cita con otro profesional para poder poner en común con el mismo toda la información que pudiera ser relevante.

#### **Historia 14: Información Útil**

Como usuario quiero poder consultar toda la información recopilada en el folleto de la AECC para que esta se pueda consultar de una manera sencilla y aprovechar el conocimiento de la misma.

#### **Historia 15: Servicios Web**

Como usuario registrado quiero disponer de unos servicios web para sincronizar la información entre varios dispositivos.

#### **Historia 16: Estadísticas**

Como personal de la AECC quiero obtener estadísticas eliminando los datos personales y de identificación para mejorar el conocimiento de la enfermedad y su relación con las actividades que esta aplicación almacena.

#### **Historia 17: Asignar Usuario**

Como superusuario de la parte servidora quiero poder asignar un usuario registrado de una provincia a uno de los agentes disponibles en la misma para que este sea su agente de referencia.

#### **Historia 18: Vision Agente**

Como agente de la AECC en la parte servidora quiero poder consultar mis supervivientes, consultar sus eventos adversos así como lanzares comunicaciones y que estas les lleguen por notificación push a sus dispositivos

### **4.2.3. Identificación de actores**

Cualquier aplicación necesita interactuar con algún actor, ya sea humano o de software.

Los actores humanos se van a identificar en nuestra aplicación como los distintos tipos de usuario y el propio agente de la AECC que se encarga de la parte servidora y que será el que se encargue de recibir parte de la información que transmita el tipo de usuario y realizar con ella las acciones oportunas.

A continuación se pasará a describir cada uno de los actores identificados en nuestra aplicación:

**USUARIO NO REGISTRADO** Este actor representa a los usuarios que todavía no se han registrado en la aplicación o aquellos que no quieren registrarse e interactúan con la misma. Incluimos también a los usuarios que acceden por primera vez y no han completado el registro.

**USUARIO REGISTRADO** Este actor representa una generalización de todos los usuarios que pueden interactuar con la aplicación, representa a los usuarios que pueden identificarse en la aplicación después de haberse registrado en la misma.

Los usuarios registrados van a tener una serie de ventajas sobre los que no lo están con forma de atención personalizada por parte de un agente de la AECC que hará de "ángel de la guardia" vigilará de manera sistemática el estado de este usuario.

**AGENTE DE LA AECC** Este actor representa a los agentes de la AECC destinados a hacer un seguimiento de los usuarios registrados en la aplicación, tendrá acceso a la información facilitada por estos últimos y podrá dar una atención personalizada a aquellos a los que tutele.

#### 4.2.4. Casos de uso

Un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso.

Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la especialización y la generalización son relaciones. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.

Los más comunes para la captura de requisitos funcionales, especialmente con el desarrollo del paradigma de la programación orientada a objetos, donde se originaron, si bien puede utilizarse con resultados igualmente satisfactorios con otros paradigmas de programación.

Pudiera parecer que las metodologías ágiles no valoran los casos de uso, nada más lejos de la realidad, los casos de uso son el guión de lo que una o varias funcionalidades deben realizar. Es por tanto el nexo de unión entre varias historias, es el guión de test automáticos de validación, es quien atesora las reglas de negocio que deben gobernar en nuestra implementación, y es quien conoce las operaciones y entidades que son necesarias en la aplicación. Por eso para los procesos de pruebas, documentación, elaboración de historias, e implementación son una pieza básica que no debemos perder de vista. Tanto es así que es la piedra sobre la que se erige la aplicación para lograr una arquitectura desacoplada y de calidad.

#### WIKIPEDIA

Dado el número de casos de uso existente y que estos quedan descritos por el siguiente apartado, ponemos un ejemplo de uno de ellos, referente a la parte de la medicación.

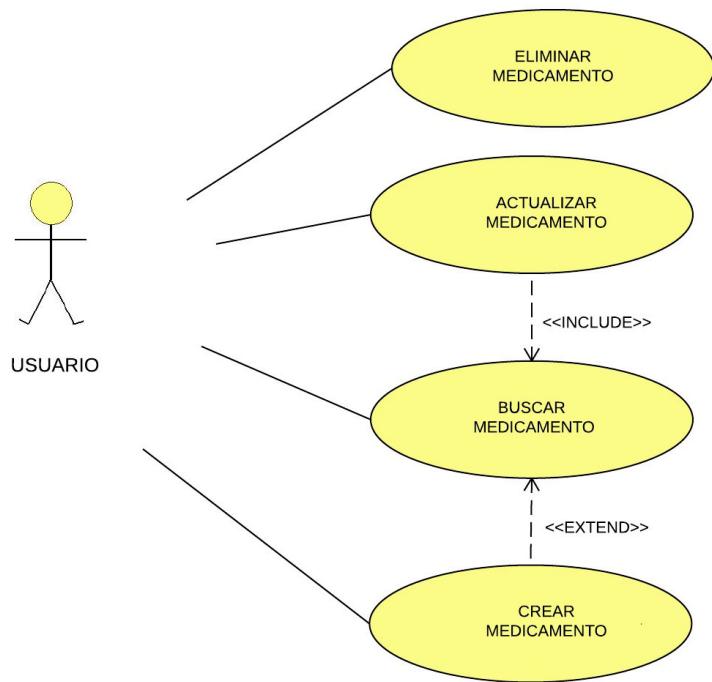


Figura 4.1: CRUD de medicamento (caso de uso)

#### 4.2.5. Descripción de Casos de Uso

##### CASOS DE USO

###### CU-001 PERFIL DE USUARIO - Inserción

CU-001	PERFIL DE USUARIO- INSERCIÓN
Versión	1.0
Fecha	23/07/2015
Actores	Usuario
Objetivo	Incluir en la aplicación los datos personales del usuario
Precondición	Los datos del usuario están sin completar
Flujo normal	1.- Desplegamos el menú drawer 2.- Pulsamos el círculo que está al principio del menú drawer 3.- Completamos los datos personales del usuario
Flujo alternativo	1.- Desplegamos el menú drawer 2.- Pulsamos la opción AJUSTES 3.- Pulsamos la opción en pantalla 'Perfil' 3.- Completamos los datos personales del usuario
Postcondición	El usuario posee sus datos personales en la aplicación
Comentarios	A la hora de introducir la fotografía dentro de los datos personales, podrá elegir entre las que tenga en la galería del propio terminal o acceder a la aplicación de cámara para hacer una nueva fotografía

Cuadro 4.1: Perfil de usuario - Inserción

###### CU-002 PERFIL DE USUARIO - Consulta

CU-002	PERFIL DE USUARIO - CONSULTA
Versión	1.0
Fecha	24/07/2015
Actores	Usuario
Objetivo	Consultar en la aplicación los datos personales del usuario
Precondición	Los datos del usuario existen en la aplicación
Flujo normal	1.- Desplegamos el menú drawer 2.- Pulsamos el círculo que está al principio del menú drawer 3.- Consultamos los datos personales del usuario
Flujo alternativo	1.- Desplegamos el menú drawer 2.- Pulsamos la opción AJUSTES 3.- Pulsamos la opción en pantalla 'Perfil' 3.- Consultamos los datos personales del usuario
Postcondición	El usuario ha consultado sus datos personales
Comentarios	Se pueden consultar todos los datos referentes al perfil del usuario

Cuadro 4.2: Perfil de usuario - Consulta

**CU-003 PERFIL DE USUARIO - Edición**

<b>CU-003</b>	<b>PERFIL DE USUARIO - EDICIÓN</b>
<b>Versión</b>	1.0
<b>Fecha</b>	24/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Modificar en la aplicación los datos personales del usuario
<b>Precondición</b>	Los datos del usuario existen en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos el círculo que está al principio del menú drawer</p> <p>3.- Completamos los datos personales del usuario que queramos cambiar</p>
<b>Flujo alternativo</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción ajustes</p> <p>3.- Pulsamos la opción en pantalla 'Perfil'</p> <p>3.- Completamos los datos personales del usuario que queramos cambiar</p>
<b>Postcondición</b>	El usuario posee sus datos personales modificados
<b>Comentarios</b>	El usuario puede a su elección dejar los datos tal y como estuviesen en un principio

Cuadro 4.3: Perfil de usuario - Edición

**CU-004 PERFIL DE USUARIO - Borrado**

<b>CU-004</b>	<b>PERFIL DE USUARIO - BORRADO</b>
<b>Versión</b>	1.0
<b>Fecha</b>	24/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Borrar de la aplicación los datos personales del usuario
<b>Precondición</b>	Los datos del usuario existen en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos el círculo que está al principio del menú drawer</p> <p>3.- Eliminamos los datos personales del usuario</p>
<b>Flujo alternativo</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción ajustes</p> <p>3.- Pulsamos la opción en pantalla 'Perfil'</p> <p>3.- Eliminamos dos datos del usuario</p>
<b>Postcondición</b>	El usuario ha eliminado sus datos personales
<b>Comentarios</b>	El usuario puede a su elección dejar los datos tal y como estuviesen en un principio

Cuadro 4.4: Perfil de usuario - Borrado

**CITA MÉDICA****CU-005 CITA MÉDICA - Inserción**

<b>CU-005</b>	<b>CITA MÉDICA - INSERCIÓN</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Insertar en la aplicación una cita médica que tiene el usuario
<b>Precondición</b>	La cita médica no existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción CITA MÉDICA</p> <p>3.- Pulsamos el ícono +</p> <p>4.- Completamos los datos pertinentes de la cita médica</p>
<b>Flujo alternativo</b>	<p><b>Flujo A</b></p> <p>1a.- Desplegamos el menú drawer</p> <p>2a.- Pulsamos la opción PRINCIPAL</p> <p>3a.- Elegimos la opción 'añadir cita' al pulsar el ícono +</p> <p>4a.- Completamos los datos de la cita</p> <p><b>Flujo B</b></p> <p>1b.- Desplegamos el menú drawer</p> <p>2b.- Pulsamos la opción HORARIO</p> <p>3b.- Elegimos la opción 'AÑADIR CITA' al pulsar el ícono +</p> <p>4b.- Completamos los datos de la cita</p> <p><b>Flujo C</b></p> <p>1c.- Desplegamos el menú drawer</p> <p>2c.- Pulsamos la opción HORARIO</p> <p>3c.- Cambiamos a la vista mensual</p> <p>4c.- Elegimos la opción 'AÑADIR CITA' al mantener pulsado sobre un día</p> <p>5c.- Completamos los datos de la cita</p>
<b>Postcondición</b>	El usuario ya posee los datos de la cita en la aplicación
<b>Comentarios</b>	Los valores de la fecha de la cita y el nombre de la misma son obligatorios, en caso de no ser rellenados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma. En la sección de medicamentos de los datos de la cita médica, al pulsar el ícono + podemos crear un nuevo medicamento o seleccionar uno de los ya existentes. Esto es similar para las secciones de personajes, síntomas y pruebas.

Cuadro 4.5: Cita médica - Inserción

**CU-006 CITA MÉDICA - Consulta**

<b>CU-006</b>	<b>CITA MÉDICA - Consulta</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Consultar una cita médica que ya posee el usuario
<b>Precondición</b>	La cita médica existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción CITA MÉDICA</p> <p>3.- Pulsamos sobre la cita médica que queremos consultar</p>
<b>Flujo alternativo</b>	<p><b>Flujo A</b></p> <p>1a.- Desplegamos el menú drawer</p> <p>2a.- Pulsamos la opción PRINCIPAL si no estamos en esa opción</p> <p>3a.- Pulsamos sobre la cita medica que queramos consultar</p> <p><b>Flujo B</b></p> <p>1b.- Desplegamos el menú drawer</p> <p>2b.- Pulsamos la opción HORARIO</p> <p>3b.- Pulsamos sobre la cita que queramos consultar</p> <p><b>Flujo C</b></p> <p>1c.- Desplegamos el menú drawer</p> <p>2c.- Pulsamos la opción HORARIO</p> <p>3c.- Cambiamos a la vista mensual</p> <p>4c.- Pulsamos sobre el día que contenga una cita que queramos consultar</p> <p>5c.- Pulsamos sobre la cita</p>
<b>Postcondición</b>	El usuario ha consultado la cita a la que quería acceder
<b>Comentarios</b>	

Cuadro 4.6: Cita médica - Consulta

**CU-007 CITA MÉDICA - Edición**

<b>CU-007</b>	<b>CITA MÉDICA - Edición</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Editar una cita médica que ya posee el usuario
<b>Precondición</b>	La cita médica existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción CITA MÉDICA</p> <p>3.- Pulsamos sobre los tres puntos a la derecha de la cita médica que queremos editar</p> <p>4.- Seleccionamos la opción 'editar'</p> <p>5.- Modificamos los datos pertinentes de la cita médica que queremos cambiar</p>
<b>Flujo alternativo</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción Horario</p> <p>3.- Pulsamos sobre el ícono de los tres puntos de cita que queremos modificar</p> <p>4.- Elegimos la opción 'editar'</p> <p>5.- Modificamos los datos de la cita</p>
<b>Postcondición</b>	El usuario ya posee los nuevos datos modificados de la cita en la aplicación
<b>Comentarios</b>	Los valores de la fecha de la cita y el nombre de la misma son obligatorios, en caso de no ser completados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma. En la sección de medicamentos de los datos de la cita médica, al pulsar el ícono + podemos crear un nuevo medicamento o seleccionar uno de los ya existentes. Esto es similar para las secciones de personajes, síntomas y pruebas.

Cuadro 4.7: Cita médica - Edición

**CU-008 CITA MÉDICA - Borrado**

<b>CU-008</b>	<b>CITA MÉDICA - Borrado</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Borrar una cita médica que ya posee el usuario
<b>Precondición</b>	La cita médica existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción CITA MÉDICA</p> <p>3.- Pulsamos sobre los tres puntos a la derecha de la cita médica que queremos eliminar dentro del listado</p> <p>4.- Seleccionamos la opción 'eliminar'</p>
<b>Flujo alternativo</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción Horario</p> <p>3.- Pulsamos sobre el ícono de los tres puntos de cita que queremos eliminar</p> <p>4.- Elegimos la opción 'Eliminar'</p>
<b>Postcondición</b>	La cita médica ya no existe en la aplicación
<b>Comentarios</b>	

Cuadro 4.8: Cita médica - Borrado

**RUTINA DIARIA****CU-009 RUTINA - Inserción**

CU-009	RUTINA - INSERCIÓN
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Insertar en la aplicación una rutina que tiene el usuario
<b>Precondición</b>	La rutina no existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción RUTINA</p> <p>3.- Pulsamos el icono +</p> <p>4.- Completamos los datos pertinentes de la rutina</p>
<b>Flujo alternativo</b>	<p><b>Flujo A</b></p> <p>1a.- Desplegamos el menú drawer</p> <p>2a.- Pulsamos la opción PRINCIPAL</p> <p>3a.- Elegimos la opción 'añadir rutina' al pulsar el icono +</p> <p>4a.- Completamos los datos de la rutina</p> <p><b>Flujo B</b></p> <p>1b.- Desplegamos el menú drawer</p> <p>2b.- Pulsamos la opción Horario</p> <p>3b.- Elegimos la opción 'añadir rutina' al pulsar el icono +</p> <p>4b.- Completamos los datos de la rutina</p> <p><b>Flujo C</b></p> <p>1c.- Desplegamos el menú drawer</p> <p>2c.- Pulsamos la opción HORARIO</p> <p>3c.- Cambiamos a la vista mensual</p> <p>4c.- Elegimos la opción 'AÑADIR RUTINA' al pulsar sobre un día</p> <p>5c.- Completamos los datos de la rutina</p>
<b>Postcondición</b>	El usuario ya posee la nueva rutina en la aplicación
<b>Comentarios</b>	Los valores de la fecha de la rutina y el nombre de la misma son obligatorios, en caso de no ser rellenados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma. En la sección de personajes de los datos de la rutina, al pulsar el icono + podemos crear un nuevo personaje o seleccionar uno de los ya existentes.

Cuadro 4.9: Rutina - Inserción

**CU-010 RUTINA - Consulta**

<b>CU-010</b>	<b>RUTINA - Consulta</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Consultar una rutina que ya posee el usuario
<b>Precondición</b>	Se quiere acceder a una rutina existente en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción RUTINA</p> <p>3.- Pulsamos sobre la rutina que queremos consultar</p>
<b>Flujo alternativo</b>	<p><b>Flujo A</b></p> <p>1a.- Desplegamos el menú drawer</p> <p>2a.- Pulsamos la opción PRINCIPAL si no estamos en esa opción</p> <p>3a.- Pulsamos sobre la rutina que queremos consultar</p> <p><b>Flujo B</b></p> <p>1b.- Desplegamos el menú drawer</p> <p>2b.- Pulsamos la opción HORARIO</p> <p>3b.- Pulsamos sobre la rutina que queremos consultar</p> <p><b>Flujo C</b></p> <p>1c.- Desplegamos el menú drawer</p> <p>2c.- Pulsamos la opción HORARIO</p> <p>3c.- Cambiamos a la vista mensual</p> <p>4c.- Pulsamos sobre el día que contenga una rutina que queremos consultar</p> <p>5c.- Pulsamos sobre la rutina</p>
<b>Postcondición</b>	El usuario ha consultado la rutina a la que quería acceder
<b>Comentarios</b>	

Cuadro 4.10: Rutina - Consulta

**CU-011 RUTINA - Edición**

<b>CU-011</b>	<b>RUTINA - Edición</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Editar una rutina que ya posee el usuario
<b>Precondición</b>	La rutina existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción RUTINA</p> <p>3.- Pulsamos sobre los tres puntos a la derecha de la rutina que queremos editar</p> <p>4.- Seleccionamos la opción 'editar'</p> <p>5.- Modificamos los datos pertinentes de la rutina que queremos cambiar</p>
<b>Flujo alternativo</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción Horario</p> <p>3.- Pulsamos sobre el ícono de los tres puntos de rutina que queremos modificar</p> <p>4.- Elegimos la opción 'editar'</p> <p>5.- Modificamos los datos de la rutina</p>
<b>Postcondición</b>	El usuario ya posee los nuevos datos modificados de la rutina en la aplicación
<b>Comentarios</b>	<p>Los valores de la fecha de la rutina y el nombre de la misma son obligatorios, en caso de no ser completados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma.</p> <p>En la sección de personajes de los datos de la rutina, al pulsar el ícono + podemos crear un nuevo personaje o seleccionar uno de los ya existentes.</p>

Cuadro 4.11: Rutina - Edición

**CU-012 RUTINA - Borrado**

<b>CU-012</b>	<b>RUTINA - Borrado</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Borrar una rutina que ya posee el usuario
<b>Precondición</b>	La rutina existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción RUTINA</p> <p>3.- Pulsamos sobre los tres puntos a la derecha de la rutina que queremos eliminar dentro del listado</p> <p>4.- Seleccionamos la opción 'eliminar'</p>
<b>Flujo alternativo</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción Horario</p> <p>3.- Pulsamos sobre el ícono de los tres puntos de rutina que queremos eliminar</p> <p>4.- Elegimos la opción 'Eliminar'</p>
<b>Postcondición</b>	La rutina que se quería eliminar de la aplicación ya no existe
<b>Comentarios</b>	

Cuadro 4.12: Rutina - Borrado

## MEDICAMENTO

### CU-013 MEDICAMENTO - Inserción

CU-013	MEDICAMENTO - INSERCIÓN
Versión	1.0
Fecha	27/07/2015
Actores	Usuario
Objetivo	Insertar en la aplicación un medicamento que tiene el usuario
Precondición	El medicamento no existe en la aplicación
Flujo normal	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción MEDICAMENTO</p> <p>3.- Pulsamos el ícono +</p> <p>4.- Completamos los datos pertinentes al medicamento</p>
Flujo alternativo	<p><b>Partiendo de CU-005</b> en Inserción de datos de la cita</p> <p>1.- Pulsamos el ícono + de la sección de medicamentos</p> <p>2.- Seleccionamos la opción 'Nuevo medicamento'</p> <p>3.- Completamos los datos pertinentes al medicamento</p> <p><b>Partiendo de CU-007</b> en Edición de datos de la cita</p> <p>1.- Pulsamos el ícono + de la sección de medicamentos</p> <p>2.- Seleccionamos la opción 'Nuevo medicamento'</p> <p>3.- Completamos los datos pertinentes al medicamento</p>
Postcondición	El usuario ya posee el medicamento en la aplicación
Comentarios	Los valores de la posología del medicamento y el nombre de la misma son obligatorios, en caso de no ser completados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma.

Cuadro 4.13: Medicamento - Inserción

**CU-014 MEDICAMENTO - Consulta**

<b>CU-014</b>	<b>MEDICAMENTO - Consulta</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Consultar un medicamento que ya posee el usuario
<b>Precondición</b>	El medicamento existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción MEDICAMENTO</p> <p>3.- Pulsamos sobre el medicamento que queremos consultar</p>
<b>Flujo alternativo</b>	<p><b>Partiendo de CU-005</b> en Inserción de datos de la cita</p> <p>1.- Pulsamos el ícono del medicamento ya asociado</p> <p>2.- Seleccionamos la opción 'Ver medicamento'</p> <p><b>Partiendo de CU-006</b> en Consulta de datos de la cita</p> <p>1.- Pulsamos el ícono del medicamentos asociado a la cita</p> <p><b>Partiendo de CU-007</b> en Edición de datos de la cita</p> <p>1.- Pulsamos el ícono del medicamento ya asociado</p> <p>2.- Seleccionamos la opción 'Ver medicamento'</p>
<b>Postcondición</b>	El usuario ya ha accedido a la información sobre ese medicamento
<b>Comentarios</b>	El flujo alternativo requiere que el medicamento este asociado previamente a la cita médica

Cuadro 4.14: Medicamento - Consulta

**CU-015 MEDICAMENTO - Edición**

<b>CU-015</b>	<b>MEDICAMENTO - Edición</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Editar un medicamento que ya posee el usuario
<b>Precondición</b>	Se quiere modificar la información de un medicamento existente en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción MEDICAMENTO</p> <p>3.- Pulsamos sobre los tres puntos a la derecha del medicamento que queremos editar</p> <p>4.- Seleccionamos la opción 'editar'</p> <p>5.- Modificamos los datos pertinentes del medicamento que queremos cambiar</p>
<b>Flujo alternativo</b>	No se contempla en esta versión de la aplicación
<b>Postcondición</b>	La información de un medicamento existente en la aplicación ha sido modificada
<b>Comentarios</b>	Los valores de la posología del medicamento y el nombre de la misma son obligatorios, en caso de no ser completados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma.

Cuadro 4.15: Medicamento - Edición

**CU-016 MEDICAMENTO - Borrado**

<b>CU-016</b>	<b>MEDICAMENTO - Borrado</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Borrar un medicamento que ya posee el usuario
<b>Precondición</b>	El medicamento existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción MEDICAMENTO</p> <p>3.- Pulsamos sobre los tres puntos a la derecha del medicamento que queremos eliminar dentro del listado</p> <p>4.- Seleccionamos la opción 'eliminar'</p>
<b>Flujo alternativo</b>	No se contempla en esta versión de la aplicación
<b>Postcondición</b>	El medicamento que se quería eliminar ya no existe en la aplicación
<b>Comentarios</b>	El medicamento que se ha eliminado desaparece de todas las citas médicas, así como sus notificaciones

Cuadro 4.16: Medicamento - Borrado

**PERSONAJE****CU-017 PERSONAJE - Inserción**

<b>CU-017</b>	<b>PERSONAJE - INSERCIÓN</b>
<b>Versión</b>	1.0
<b>Fecha</b>	28/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Insertar en la aplicación un personaje que conoce el usuario
<b>Precondición</b>	El personaje no existe en la aplicación
<b>Flujo normal</b>	<p><b>Partiendo de CU-005</b> en Inserción de datos de la cita</p> <p>1.- Desplegamos el menú drawer      2.- Pulsamos la opción PERSONAJE      3.- Pulsamos el ícono +      4.- Completamos los datos pertinentes del personaje</p>
<b>Flujo alternativo</b>	<p><b>Partiendo de CU-007</b> en Edición de datos de la cita</p> <p>1.- Pulsamos el ícono + de la sección de personajes      2.- Seleccionamos la opción 'Nuevo personaje'      3.- Completamos los datos pertinentes al personaje</p> <p><b>Partiendo de CU-009</b> en Inserción de datos de la rutina</p> <p>1.- Pulsamos el ícono + de la sección de personajes      2.- Seleccionamos la opción 'Nuevo personaje'      3.- Completamos los datos pertinentes al personaje</p> <p><b>Partiendo de CU-011</b> en Edición de datos de la rutina</p> <p>1.- Pulsamos el ícono + de la sección de personajes      2.- Seleccionamos la opción 'Nuevo personaje'      3.- Completamos los datos pertinentes al personaje</p>
<b>Postcondición</b>	El usuario ya posee los datos del personaje en la aplicación
<b>Comentarios</b>	Los valores del nombre y la relación son obligatorios, en caso de no ser rellenados, la aplicación avisará con un mensaje de error y no se podrá guardar el mismo.

Cuadro 4.17: Personaje - Inserción

**CU-018 PERSONAJE - Consulta**

<b>CU-018</b>	<b>PERSONAJE - Consulta</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Consultar un personaje que ya posee el usuario
<b>Precondición</b>	El personaje existe en la aplicación
<b>Flujo alternativo</b>	<p><b>Partiendo de CU-005</b> en Inserción de datos de la cita            1.- Pulsamos el ícono del síntoma ya asociado            2.- Seleccionamos la opción 'Ver personaje'</p> <p><b>Partiendo de CU-006</b> en Consulta de datos de la cita            1.- Pulsamos el ícono del personaje asociado a la cita</p> <p><b>Partiendo de CU-007</b> en Edición de datos de la cita            1.- Pulsamos el ícono del personaje ya asociado            2.- Seleccionamos la opción 'Ver personaje'</p> <p><b>Partiendo de CU-009</b> en Inserción de datos de la rutina            1.- Pulsamos el ícono del personaje ya asociado            2.- Seleccionamos la opción 'Ver personaje'</p> <p><b>Partiendo de CU-010</b> en Consulta de datos de la rutina            1.- Pulsamos el ícono del personaje asociado a la rutina</p> <p><b>Partiendo de CU-011</b> en Edición de datos de la rutina            1.- Pulsamos el ícono del personaje ya asociado            2.- Seleccionamos la opción 'Ver personaje'</p>
<b>Postcondición</b>	El usuario ya ha consultado los datos del personaje en la aplicación
<b>Comentarios</b>	El flujo alternativo requiere que el personaje este asociado previamente a una cita o a una rutina dependiendo del flujo

Cuadro 4.18: Personaje - Consulta

**CU-019 PERSONAJE - Edición**

<b>CU-019</b>	<b>PERSONAJE - Edición</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Editar un personaje que ya posee el usuario
<b>Precondición</b>	El personaje existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción PERSONAJE</p> <p>3.- Pulsamos sobre los tres puntos a la derecha del personaje que queremos editar</p> <p>4.- Seleccionamos la opción 'editar'</p> <p>5.- Modificamos los datos pertinentes del personaje que queremos cambiar</p>
<b>Flujo alternativo</b>	No se contempla en esta versión de la aplicación
<b>Postcondición</b>	El usuario ya posee los nuevos datos modificados del personaje en la aplicación
<b>Comentarios</b>	Los valores del nombre y la relación son obligatorios, en caso de no ser llenados, la aplicación avisará con un mensaje de error y no se podrá guardar el mismo.

Cuadro 4.19: Personaje - Edición

**CU-020 PERSONAJE - Borrado**

<b>CU-020</b>	<b>PERSONAJE - Borrado</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Borrar un personaje que ya posee el usuario
<b>Precondición</b>	El personaje existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción PERSONAJE</p> <p>3.- Pulsamos sobre los tres puntos a la derecha del personaje que queremos eliminar dentro del listado</p> <p>4.- Seleccionamos la opción 'eliminar'</p>
<b>Flujo alternativo</b>	No se contempla en esta versión de la aplicación
<b>Postcondición</b>	El personaje que se quería borrar ya no existe en la aplicación
<b>Comentarios</b>	El Personaje borrado desaparece de todas las citas médicas o rutinas en las que pudiera aparecer

Cuadro 4.20: Personaje - Borrado

## SÍNTOMA

### CU-021 SÍNTOMA - Inserción

CU-021	SÍNTOMA - INSERCIÓN
Versión	1.0
Fecha	27/07/2015
Actores	Usuario
Objetivo	Insertar en la aplicación una síntoma que tiene el usuario
Precondición	El síntoma no existe en la aplicación
Flujo normal	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción SÍNTOMA</p> <p>3.- Pulsamos el ícono +</p> <p>4.- Completamos los datos pertinentes del síntoma</p>
Flujo alternativo	<p><b>Partiendo de CU-005</b> en Inserción de datos de la cita</p> <p>1.- Pulsamos el ícono + de la sección de síntomas</p> <p>2.- Seleccionamos la opción 'Nuevo síntoma'</p> <p>3.- Completamos los datos pertinentes al síntoma</p> <p><b>Partiendo de CU-007</b> en Edición de datos de la cita</p> <p>1.- Pulsamos el ícono + de la sección de síntomas</p> <p>2.- Seleccionamos la opción 'Nuevo síntoma'</p> <p>3.- Completamos los datos pertinentes al síntoma</p>
Postcondición	El usuario ya posee los datos del síntoma en la aplicación
Comentarios	Los valores del nombre del síntoma y la descripción son obligatorios, en caso de no ser rellenados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma.

Cuadro 4.21: Síntoma - Inserción

**CU-022 SÍNTOMA - Consulta**

<b>CU-022</b>	<b>SÍNTOMA - Consulta</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Consultar un síntoma que ya posee el usuario
<b>Precondición</b>	El síntoma existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción SÍNTOMA</p> <p>3.- Pulsamos sobre el síntoma que queremos consultar</p>
<b>Flujo alternativo</b>	<p><b>Partiendo de CU-005</b> en Inserción de datos de la cita</p> <p>1.- Pulsamos el ícono del síntoma ya asociado</p> <p>2.- Seleccionamos la opción 'Ver síntoma'</p> <p><b>Partiendo de CU-006</b> en Consulta de datos de la cita</p> <p>1.- Pulsamos el ícono del síntoma asociado a la cita</p> <p><b>Partiendo de CU-007</b> en Edición de datos de la cita</p> <p>1.- Pulsamos el ícono del síntoma ya asociado</p> <p>2.- Seleccionamos la opción 'Ver síntoma'</p>
<b>Postcondición</b>	El usuario ya ha accedido a la información sobre ese síntoma
<b>Comentarios</b>	El flujo alternativo requiere que el síntoma este asociado previamente a la cita médica

Cuadro 4.22: Síntoma - Consulta

**CU-023 SÍNTOMA - Edición**

<b>CU-023</b>	<b>SÍNTOMA - Edición</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Editar un síntoma que ya posee el usuario
<b>Precondición</b>	El síntoma existe en la aplicación
<b>Flujo normal</b>	1.- Desplegamos el menú drawer 2.- Pulsamos la opción SÍNTOMA 3.- Pulsamos sobre los tres puntos a la derecha del síntoma que queremos editar 4.- Seleccionamos la opción 'editar' 5.- Modificamos los datos pertinentes del síntoma que queremos cambiar
<b>Flujo alternativo</b>	No se contempla en esta versión de la aplicación
<b>Postcondición</b>	Se han modificado los datos del síntoma existente
<b>Comentarios</b>	Los valores del nombre del síntoma y la descripción son obligatorios, en caso de no ser completados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma.

Cuadro 4.23: Síntoma - Edición

**CU-024 SÍNTOMA - Borrado**

<b>CU-024</b>	<b>SÍNTOMA - Borrado</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Borrar un síntoma que ya posee el usuario
<b>Precondición</b>	El síntoma existe en la aplicación
<b>Flujo normal</b>	1.- Desplegamos el menú drawer 2.- Pulsamos la opción SÍNTOMA 3.- Pulsamos sobre los tres puntos a la derecha de la síntoma que queremos eliminar dentro del listado 4.- Seleccionamos la opción 'eliminar'
<b>Flujo alternativo</b>	No se contempla en esta versión de la aplicación
<b>Postcondición</b>	El síntoma que se quería eliminar ya no existe en la aplicación
<b>Comentarios</b>	El síntoma que se ha eliminado desaparece de todas las citas médicas

Cuadro 4.24: Síntoma - Borrado

**PRUEBA****CU-025 PRUEBA - Inserción**

<b>CU-025</b>	<b>PRUEBA - INSERCIÓN</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Insertar en la aplicación una prueba que tiene el usuario
<b>Precondición</b>	La prueba no existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción PRUEBA</p> <p>3.- Pulsamos el ícono +</p> <p>4.- Completamos los datos pertinentes de la prueba</p>
<b>Flujo alternativo</b>	<p><b>Partiendo de CU-005</b> en Inserción de datos de la cita</p> <p>1.- Pulsamos el ícono + de la sección de pruebas</p> <p>2.- Seleccionamos la opción 'Nueva prueba'</p> <p>3.- Completamos los datos pertinentes a la prueba</p> <p><b>Partiendo de CU-007</b> en Edición de datos de la cita</p> <p>1.- Pulsamos el ícono + de la sección de pruebas</p> <p>2.- Seleccionamos la opción 'Nueva prueba'</p> <p>3.- Completamos los datos pertinentes a la prueba</p>
<b>Postcondición</b>	El usuario ya posee los datos de la prueba en la aplicación
<b>Comentarios</b>	Los valores de la fecha de la prueba y el nombre de la misma son obligatorios, en caso de no ser rellenados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma.

Cuadro 4.25: Prueba - Inserción

**CU-026 PRUEBA - Consulta**

<b>CU-026</b>	<b>PRUEBA - Consulta</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Consultar una prueba que ya posee el usuario
<b>Precondición</b>	La prueba existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción PRUEBA</p> <p>3.- Pulsamos sobre la prueba que queremos consultar</p>
<b>Flujo alternativo</b>	<p><b>Partiendo de CU-005</b> en Inserción de datos de la cita</p> <p>1.- Pulsamos el ícono de la prueba ya asociada</p> <p>2.- Seleccionamos la opción 'Ver prueba'</p> <p><b>Partiendo de CU-006</b> en Consulta de datos de la cita</p> <p>1.- Pulsamos el ícono de la prueba asociada a la cita</p> <p><b>Partiendo de CU-007</b> en Edición de datos de la cita</p> <p>1.- Pulsamos el ícono de la prueba ya asociada</p> <p>2.- Seleccionamos la opción 'Ver prueba'</p>
<b>Postcondición</b>	El usuario ha accedido a la información de la prueba en la aplicación
<b>Comentarios</b>	El flujo alternativo requiere que la prueba esté asociada previamente a la cita médica

Cuadro 4.26: Prueba - Consulta

**CU-027 PRUEBA - Edición**

<b>CU-027</b>	<b>PRUEBA - Edición</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Editar una prueba que ya posee el usuario
<b>Precondición</b>	La prueba existe en la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción PRUEBA</p> <p>3.- Pulsamos sobre los tres puntos a la derecha de la prueba que queremos editar</p> <p>4.- Seleccionamos la opción 'editar'</p> <p>5.- Modificamos los datos pertinentes de la prueba que queremos cambiar</p>
<b>Flujo alternativo</b>	No se contempla en esta versión de la aplicación
<b>Postcondición</b>	El usuario ya posee los nuevos datos modificados de la prueba en la aplicación
<b>Comentarios</b>	Los valores de la fecha de la prueba y el nombre de la misma son obligatorios, en caso de no ser completados, la aplicación avisará con un mensaje de error y no se podrá guardar la misma.

Cuadro 4.27: Prueba - Edición

**CU-028 PRUEBA - Borrado**

<b>CU-028</b>	<b>PRUEBA - Borrado</b>
<b>Versión</b>	1.0
<b>Fecha</b>	27/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Borrar una prueba que ya posee el usuario
<b>Precondición</b>	La prueba existe en la aplicación
<b>Flujo normal</b>	1.- Desplegamos el menú drawer 2.- Pulsamos la opción PRUEBA 3.- Pulsamos sobre los tres puntos a la derecha de la prueba que queremos eliminar dentro del listado 4.- Seleccionamos la opción 'eliminar'
<b>Flujo alternativo</b>	No se contempla en esta versión de la aplicación
<b>Postcondición</b>	La prueba que se quería eliminar ya no existe en la aplicación
<b>Comentarios</b>	

Cuadro 4.28: Prueba - Borrado

**CU-029 ORDENACIÓN DE LISTAS**

<b>CU-029</b>	<b>ORDENACIÓN DE LISTAS</b>
<b>Versión</b>	1.0
<b>Fecha</b>	28/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Ordenar un listado de elementos
<b>Precondición</b>	La lista posee más de dos elementos ordenados de manera distinta a que se desea
<b>Flujo normal</b>	1.- Desplegamos el menú drawer 2.- Pulsamos cualquiera de las opciones CITA MÉDICA, RUTINA, MEDICAMENTO, PERSONAJE, SÍNTOMA o PRUEBA 3.- Pulsamos sobre el ícono del engranaje del listado 4.- Seleccionamos el tipo de ordenación
<b>Flujo alternativo</b>	Para cualquiera de los diferentes tipo de objetos es similar
<b>Postcondición</b>	La lista los elementos ordenados ahora está de la manera deseada
<b>Comentarios</b>	Los modos de ordenación varían de un caso a otro

Cuadro 4.29: Ordenación de listas

**CU-030 CONSULTA DE LISTAS**

<b>CU-030</b>	<b>CONSULTA DE LISTAS</b>
<b>Versión</b>	1.0
<b>Fecha</b>	28/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Consultar un listado de elementos
<b>Precondición</b>	Existe alguna lista de elementos
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos cualquiera de las opciones CITA MÉDICA, RUTINA, MEDICAMENTO, PERSONAJE, SÍNTOMA, PRUEBA</p>
<b>Flujo alternativo</b>	Para cualquiera de los diferentes tipo de objetos es similar
<b>Postcondición</b>	Se ha consultado algún listado
<b>Comentarios</b>	

Cuadro 4.30: Consulta de listas

**CU-031 AÑADIR NOTIFICACIÓN - MEDICAMENTO**

<b>CU-031</b>	<b>AÑADIR NOTIFICACIÓN - MEDICAMENTO</b>
<b>Versión</b>	1.0
<b>Fecha</b>	28/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Añadir alerta a un medicamento
<b>Precondición</b>	El medicamento no posee ninguna alerta
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción MEDICAMENTO</p> <p>3.- Pulsamos sobre los tres puntos a la derecha del que queremos editar</p> <p>4.- Seleccionamos la opción 'editar'</p> <p>5.- Insertamos la fecha de inicio</p> <p>6.- Insertamos la hora de inicio</p> <p>7.- Insertamos la fecha de fin</p> <p>8.- Insertamos la hora de fin</p> <p>8.- Insertamos las horas entre dosis</p>
<b>Flujo alternativo</b>	Partiendo de CU-009 en Inserción de datos de la cita médica Cumplimentamos como en el flujo normal
<b>Postcondición</b>	El medicamento posee una alerta
<b>Comentarios</b>	El tono de la alerta se configurará en la opción AJUSTES

Cuadro 4.31: Añadir notificación - Medicamento

**CU-032 AÑADIR NOTIFICACIÓN - CITA**

<b>CU-032</b>	<b>AÑADIR NOTIFICACIÓN - CITA</b>
<b>Versión</b>	1.0
<b>Fecha</b>	28/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Añadir alerta a una cita
<b>Precondición</b>	La cita no posee ninguna alerta
<b>Flujo normal</b>	1.- Desplegamos el menú drawer 2.- Pulsamos la opción CITA MÉDICA 3.- Pulsamos sobre los tres puntos a la derecha de la cita médica que queremos editar 4.- Seleccionamos la opción 'editar' 5.- Insertamos la fecha de aviso 6.- Insertamos la hora de aviso
<b>Flujo alternativo</b>	<b>Partiendo de CU-005 en Inserción de datos de la cita</b> 1.- Insertamos la fecha de aviso 2.- Insertamos la hora de aviso
<b>Postcondición</b>	La cita posee una alerta
<b>Comentarios</b>	El tono de la alerta se configurará en la opción AJUSTES

Cuadro 4.32: Ordenación de listas

**CU-033 AÑADIR NOTIFICACIÓN - RUTINA**

<b>CU-033</b>	<b>AÑADIR NOTIFICACIÓN - RUTINA</b>
<b>Versión</b>	1.0
<b>Fecha</b>	28/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Añadir alerta a una rutina
<b>Precondición</b>	La rutina no posee ninguna alerta
<b>Flujo normal</b>	1.- Desplegamos el menú drawer 2.- Pulsamos la opción RUTINA 3.- Pulsamos sobre los tres puntos a la derecha de la cita médica que queremos editar 4.- Seleccionamos la opción 'editar' 5.- Insertamos la fecha de aviso 6.- Insertamos la hora de aviso
<b>Flujo alternativo</b>	<b>Partiendo de CU-009 en Inserción de datos de la rutina</b> 1.- Insertamos la fecha de aviso 2.- Insertamos la hora de aviso
<b>Postcondición</b>	La rutina posee una alerta
<b>Comentarios</b>	El tono de la alerta se configurará en la opción AJUSTES

Cuadro 4.33: Añadir notificación - Rutina

**CU-034 AÑADIR OBJETOS A LA CITA**

<b>CU-034</b>	<b>AÑADIR OBJETOS A LA CITA</b>
<b>Versión</b>	1.0
<b>Fecha</b>	28/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Añadir objetos a una cita
<b>Precondición</b>	La cita no posee algún objeto de la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción CITA MÉDICA</p> <p>3.- Pulsamos sobre los tres puntos a la derecha de la cita médica que queremos editar</p> <p>4.- Seleccionamos la opción 'editar'</p> <p>5.- Pulsamos el ícono + en alguna de las secciones de personajes, medicamentos, síntomas o pruebas</p> <p>6.- Pulsamos la opción de menú 'Personaje, 'Medicamento', 'Síntoma' o Prueba' dependiendo de la sección</p> <p>7.- Seleccionamos de la lista el personaje, medicamento, síntoma o prueba dependiendo de la opción anterior seleccionada</p>
<b>Flujo alternativo</b>	<b>Partiendo de CU-005 en Inserción de datos de la cita</b> <p>1.- Pulsamos el ícono + en alguna de las secciones de personajes, medicamentos, síntomas o pruebas</p> <p>2.- Pulsamos la opción de menú 'Personaje, 'Medicamento', 'Síntoma' o Prueba' dependiendo de la sección</p>
<b>Postcondición</b>	La cita posee un objeto de la aplicación que antes no tenía
<b>Comentarios</b>	Los objetos de los que hablamos son objetos persistentes de la aplicación personajes, medicamentos, síntomas y pruebas.

Cuadro 4.34: Añadir objetos a la cita

**CU-035 AÑADIR OBJETOS A LA RUTINA**

<b>CU-035</b>	<b>AÑADIR OBJETOS A LA RUTINA</b>
<b>Versión</b>	1.0
<b>Fecha</b>	28/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Añadir objetos a una rutina
<b>Precondición</b>	La cita no posee algún personaje
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción RUTINA</p> <p>3.- Pulsamos sobre los tres puntos a la derecha de la rutina que queremos editar</p> <p>4.- Seleccionamos la opción 'editar'</p> <p>5.- Pulsamos el icono + en la sección de personajes</p> <p>6.- Pulsamos la opción de menú 'Personaje'</p> <p>7.- Seleccionamos de la lista el personaje que queramos añadir.</p>
<b>Flujo alternativo</b>	<b>Partiendo de CU-009</b> en Inserción de datos de la rutina <p>1.- Pulsamos el icono + en la sección de personajes</p> <p>2.- Pulsamos la opción de menú 'Personaje'</p> <p>3.- Seleccionamos de la lista el personaje que queramos añadir.</p>
<b>Postcondición</b>	La rutina posee un personaje de la aplicación que antes no tenía
<b>Comentarios</b>	Los objetos que acepta la rutina son personajes.

Cuadro 4.35: Añadir objetos a la rutina

**CU-036 CONFIGURACIÓN DE LA APLICACIÓN**

<b>CU-028</b>	<b>CONFIGURACIÓN DE LA APLICACIÓN</b>
<b>Versión</b>	1.0
<b>Fecha</b>	28/07/2015
<b>Actores</b>	Usuario
<b>Objetivo</b>	Personalizar la aplicación al gusto del usuario
<b>Precondición</b>	Ajustes por defecto de la aplicación
<b>Flujo normal</b>	<p>1.- Desplegamos el menú drawer</p> <p>2.- Pulsamos la opción AJUSTES</p> <p>3.- Personalizamos la aplicación en la medida en la que estan nos lo permita</p>
<b>Flujo alternativo</b>	
<b>Postcondición</b>	Ajustes personalizados del usuario
<b>Comentarios</b>	Letra, canción a reproducir, notificaciones.

Cuadro 4.36: Configuración de la aplicación

**4.2.6. Diagrama de clases de análisis**

**4.2.7. Diagrama Entidad-Relación de la base de datos**

**4.2.8. Modelo relacional del análisis**

# CAPÍTULO 5

---

## Diseño

---

### 5.1. Parte App móvil

### 5.2. Arquitectura

Como se ha comentado antes en este documento Scrum y las tecnologías ágiles, por su forma de enfocar las tareas y el propio desarrollo en un equipo con un nivel técnico que no sea excelente, si no se introducen medidas paliativas que corrijan esto, suelen obtener diseños arquitecturales de clases un poco pobres, tendentes al acoplamiento y poco cohesionado. Para paliar esto se suelen introducir historias de usuario de reducción de deuda técnica, o de refactorización de todo el código existente, así como herramientas que detecten code smells[23] o malas prácticas de desarrollo.

Estas herramientas pueden incluir plugins del IDE que hagan análisis estático de código y pueden advertirnos sobre puntuales problemas dentro de una clase, pero poco hay a nivel de las relaciones entre las mismas, fuera del uso de patrones de diseño conocidos, y aun en las relaciones entre estos patrones que son bloques más grandes de código. Es por eso que creemos que hay que establecer unas líneas base al principio del proyecto sobre la arquitectura de la propia aplicación. En principio serán ciertamente vaguedades y lugares comunes sobre el desarrollo en capas, aislamiento de las mismas etc, pero si a través de los sprints reservamos un poco de tiempo en la reunión de sprint planning para conversar sobre las historias del propio sprint, en que capa encajan, como deberían hablar las capas entre sí si son más de una las involucradas en la historia, etc ademas de con las historias de deuda técnica si fuesen necesarias creemos que puede solventarse en gran medida esa falta de bueno diseño y arquitectura que se achaca a las metodologías ágiles.

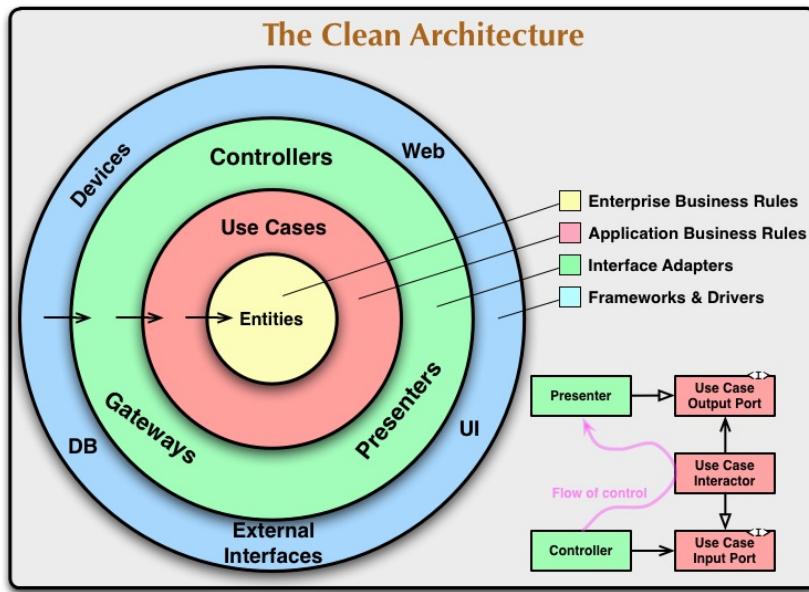


Figura 5.1: Arquitectura 'limpia' propuesta por Robert Martin

Para hablar de arquitectura hemos tomado como referencia Clean Architecture[35] propuesta por Robert C. Martin aunque es muy similar a la Arquitectura Cebolla de Jeffrey Palermo[41] o la Arquitectura Hexagonal de Alaister Cockburn[41]. Todas ellas se basan por hacer capas fuertemente cohesionadas dentro de cada una pero con nulo conocimiento fuera de las mismas que se comunican mediante interfaces públicos ofrecidos al resto de capas. Esto y la propagación hacia el interior, esto es que cada capa solo hablará con la que tiene arquitecturalmente en un nivel superior o inferior. Esto es la capa de UI que es la que interactúa con el usuario solamente invocará métodos de la interfaz de la capa de vista que es la que conceptualmente tiene justo encima. Esta solamente interactuará con la capa de UI antes mencionada y con la capa del presentador, y así sucesivamente.

### 5.2.1. Ley de Deméter

La idea subyacente bajo todas estas arquitecturas de capas que no se conocen, es la Ley de Deméter. Que en su enunciado más simple dice:

- Cada unidad debe tener un limitado conocimiento sobre otras unidades y solo conocer aquellas unidades estrechamente relacionadas a la unidad actual.
- Cada unidad debe hablar solo a sus amigos y no hablar con extraños.
- Solo hablar con sus amigos inmediatos.

La noción fundamental es que dado un objeto, este debería asumir tan poco como sea posible sobre la estructura o propiedades de cualquier otro (incluyendo sus subcomponentes).

Aplicando la ley a la orientación a objetos a la Ley de Deméter puede ser llamada más precisamente la "Ley de Deméter para funciones/Métodos" (LD-F). En este caso, un objeto A puede solicitar un servicio (llamar a un método) de un objeto B, pero el objeto A no debería

”llegar a través” del objeto B poder acceder a otro objeto, C, para solicitar sus servicios. De hacerlo podría significar que el objeto A implícitamente requeriría un mayor conocimiento de la estructura interna del objeto B. En su lugar, la interfaz de B debe ser modificada si es necesario para que pueda servir directamente la petición de A, propagándolo a cualquiera de sus subcomponentes necesarios. Alternativamente, A podría tener una referencia directa al objeto C y hacer la solicitud directamente a él. Si la ley se sigue, B es el único objeto que debería conocer su propia estructura interna.

Más formalmente, la Ley de Deméter para funciones requiere que un método m de un objeto O sólo se puede invocar los métodos de los siguientes tipos de objetos:

- El propio O
- Los parámetros de m
- Cualquier objeto creado / instanciados dentro de m
- Objetos que sean propiedades de O
- Una variable global, accesible por O, en el ámbito de m

En particular, un objeto debe evitar invocar métodos de un objeto miembro devuelto por otro método. Para muchos lenguajes orientados a objetos modernos que usan un punto como identificador de campo, la ley puede resumir simplemente como “usar sólo un punto”. Es decir, el a.b.Method() rompe la ley mientras que a.Method() no lo hace. Como analogía, cuando uno quiere que un perro ande, no se ordena a las patas del perro a caminar directamente, sino que se manda al perro que luego manda sus propias piernas.

Con esto en mente en mente construiremos tres grandes capas que serán las típicas de vista, dominio y datos. Cada capa después tendrá sus propios patrones o entidades de patrones, y alguna subcapa. Lo importante es que cada capa ofrezca un interfaz a las capas que tienen que interactuar con ella y que estas solamente conocerán dicho interfaz.

Como frontera entre la vista y el dominio pondremos un patrón de MVP o Modelo-Vista-Presentador, este patrón es una modificación del Modelo-Vista-Controlador que sirve para separar de la vista y representación gráfica de nuestros objetos de modelo, de la representación de estos. En este modelo la vista es lo más ”tonta.” simple posible de manera que no merezca la pena testarse. Y los eventos sobre la misma se los comunicará al presentador que será el encargado de lidiar con el modelo de los objetos de negocio de la aplicación.

### **5.2.2. Capa de Vista**

Esta capa contendrá toda la parte de UI, que será lo más tonta posible. Esto es, contendrá los elementos de UI para pintar y actualizar la pantalla del dispositivo, pero cualquier acción realizada por el usuario en una pantalla, será notificada al presentador que será el que siguiendo el caso de uso tome la decisión adecuada.

### **5.2.3. Capa de Dominio**

Esta capa implementará los casos de uso de la app así como las entidades del dominio de la misma. De este modo contendrá los presentadores de los casos de uso y los casos de uso mismos del diagrama anterior que se comunicaran mediante una interfaz. Los presentadores

se comunicarán también, vía interfaz, con las vistas tontas de la capa de UI.

### 5.2.4. Capa de Datos

Esta será la capa de acceso a los datos donde residan las llamadas a servicios web para la obtención o persistencia de los mismos, así como las clases que nos ayuden a interactuar con la base de datos del dispositivo. En esta capa aplicaremos un patrón repositorio sobre las entidades de la capa. Esto hará que independicemos el medio de obtención de los datos que podrá ser una base datos, una cache, un servicio web, etc. Aparte de la capa de modelo nos llegarán operaciones con los datos y será esta capa quien decida si debe actualizar los mismos en base de datos o no y si además debe hacer backup de los mismos para poder mantener los dispositivos sincronizados.

### 5.2.5. Principios SOLID

Todo esto se ha de implementar siguiendo los principios de SOLID[36] que nos muestran en cinco pequeños principios de escritura de software como escribir código reusable, con alta cohesión y bajo acoplamiento, que es por otra parte lo que deseábamos lograr al principio de la sección.

#### 5.2.5.1. S-Responsabilidad única (Single responsibility)

Este principio dice que cada clase debe ocuparse de un solo menester sencillo y concreto. Visto de otro modo, dice que cada clase debería tener un único motivo para ser modificada. En muchas ocasiones estamos tentados a poner un método reutilizable que no tienen nada que ver con la clase simplemente porque lo utiliza y pilla más a mano.

El problema surge cuando tenemos la necesidad de utilizar ese mismo método desde otra clase. Si no se refactoriza en ese momento y se crea una clase destinada para la finalidad del método, nos toparemos a largo plazo con que las clases realizan tareas que no deberían ser de su responsabilidad.

Con la anterior mentalidad nos encontraremos, por ejemplo, con un algoritmo de formateo de números en una clase destinada a leer de la base de datos porque fue el primer sitio donde se empezó a utilizar. Con otro ejemplo si estamos delante de una clase que se podría ver obligada a cambiar ante una modificación en la base de datos y a la vez, ante un cambio en el proceso de negocio, podemos afirmar que dicha clase tiene más de una responsabilidad o más de un motivo para cambiar. Esto conlleva a tener métodos difíciles de detectar y encontrar de manera que el código hay que tenerlo memorizado en la cabeza.

Se aplica tanto a la clase como a cada uno de sus métodos, con lo que cada método también debería tener un solo motivo para cambiar. El efecto que produce este principio son clases con nombres muy descriptivos y por tanto largos, que tienen menos de cinco métodos, cada uno también con nombres que sirven perfectamente de documentación, es decir, de varias palabras: CalcularAreaRectangulo y que no contienen más de 15 líneas de código.

### 5.2.5.2. O-Abierto/Cerrado (Open/Closed)

Principio que habla de crear clases extensibles sin necesidad de entrar al código fuente a modificarlo. Generalizando una entidad software (una clase, módulo o función) debe estar abierta a extensiones pero cerrada a modificaciones. Es decir, el diseño debe ser abierto para poderse extender pero cerrado para poderse modificar. Aunque dicho parece fácil, lo complicado es predecir por como se debe extender y que no tengamos que modificarlo. Para conseguir este principio hay que tener muy claro como va a funcionar la aplicación, por donde se puede extender y como van a interactuar las clases.

El uso más común de extensión es mediante la herencia y la reimplementación de métodos de la clase padre que podría incluso ser abstracta. La otra podría ser inyectando dependencias que cumplen el mismo contrato (que tienen la misma interfaz) pero que implementan diferente funcionamiento. En todos los casos, el comportamiento de la clase cambia sin que hayamos tenido que modificar código interno.

Como la totalidad del código no se puede ni se debe cerrar a cambios, el diseñador debe decidir contra cuáles protegerse mediante este principio. Su aplicación requiere bastante experiencia, no sólo por la dificultad de crear entidades de comportamiento extensible sino por el peligro que conlleva cerrar determinadas entidades o parte de ellas.

Como ya he comentado llega un momento en que las necesidades pueden llegar a ser tan imprevisibles que nos topemos que con los métodos definidos en el interface o en los métodos extensibles, no sean suficientes para cubrir las necesidades. En este caso no habrá más remedio que romper este principio y refactorizar.

Cerrar en exceso obliga a escribir demasiadas líneas de código a la hora de reutilizar la entidad en cuestión.

### 5.2.5.3. L-Sustitucion Liskov (Liskov substitution)

Este principio habla de la importancia de crear todas las clases derivadas para que también puedan ser tratadas como la propia clase base. Cuando creamos clases derivadas debemos asegurarnos de no reimplementar métodos que hagan que los métodos de la clase base no funcionen si se tratasen como un objeto de esa clase base.

Algo más formal, lo que viene diciendo es que si una función recibe un objeto como parámetro, de tipo X y en su lugar le pasamos otro de tipo Y, que hereda de X, dicha función debe proceder correctamente.

Por el propio polimorfismo, los compiladores e intérpretes admiten este paso de parámetros, la cuestión es si la función de verdad está diseñada para hacer lo que debe, aunque quien recibe como parámetro no es exactamente X, sino Y.

El principio de sustitución de Liskov está estrechamente relacionado con el anterior en cuanto a la extensibilidad de las clases cuando ésta se realiza mediante herencia o subtipos. Si una función no cumple el LSP entonces rompe el OCP puesto que para ser capaz de funcionar con subtipos (clases hijas) necesita saber demasiado de la clase padre y por tanto, modificarla. El diseño por contrato (Design by Contract) es otra forma de llamar al LSP.

#### 5.2.5.4. I-Segregacion del interface (Interface segregation)

Este principio trata de algo parecido al primer principio. Cuando se definen interfaces estos deben ser específicos a una finalidad concreta es por ello que defiende que no obliguemos a los clientes a depender de clases o interfaces que no necesitan usar. Por ello, si tenemos que definir una serie de métodos abstractos que debe utilizar una clase a través de interfaces, es preferible tener muchos interfaces que definan pocos métodos que tener un interface con muchos métodos. Ya que de lo contrario seguramente sirve a varios objetos cliente con responsabilidades diferentes, con lo que debería estar dividida en varias entidades.

El objetivo de este principio es principalmente poder reaprovechar los interfaces en otras clases. Si tenemos un interface que compara y clona en el mismo interface, de manera más complicada se podrá utilizar en una clase que solo debe comparar o en otra que solo debe clonar.

En los lenguajes como Java y C# hablamos de interfaces pero en lenguajes interpretados como Python, que no requieren interfaces, hablamos de clases. No sólo es por motivos de robustez del software, sino también por motivos de despliegue. Cuando un cliente depende de una interfaz con funcionalidad que no utiliza, se convierte en dependiente de otro cliente y la posibilidad de catástrofe frente a cambios en la interfaz o clase base se multiplica.

#### 5.2.5.5. D-Inversión de dependencias (Dependency inversion)

El objetivo de este principio conseguir desacoplar las clases. En todo diseño siempre debe existir un acoplamiento pero hay que evitarlo en la medida de lo posible. Un sistema no acoplado no hace nada pero un sistema altamente acoplado es muy difícil de mantener. La inversión de dependencias da origen a la conocida inyección de dependencias, una de las mejores técnicas para lidiar con las colaboraciones entre clases, produciendo un código reutilizable, sobrio y preparado para cambiar sin producir efectos bola de nieve.

El objetivo de este principio es el uso de abstracciones para conseguir que una clase interactúe con otras clases sin que las conozca directamente. Es decir, las clases de nivel superior no deben conocer las clases de nivel inferior. Dicho de otro modo, no debe conocer los detalles. Existen diferentes patrones como la ya mencionada inyección de dependencias o service locator que nos permiten invertir el control.

DIP explica que un módulo concreto A, no debe depender directamente de otro módulo concreto B, sino de una abstracción de B. Tal abstracción es una interfaz o una clase (que podría ser abstracta) que sirve de base para un conjunto de clases hijas.

En el caso de un lenguaje interpretado no necesitamos definir interfaces, ni siquiera jerarquías pero el concepto se aplica igualmente. Veámoslo con un ejemplo sencillo: La clase **Logica** necesita de un colaborador para guardar el dato **Dato** en algún lugar persistente. Disponemos de una clase **MyBD** que es capaz de almacenar **Dato** en una base de datos MySQL y de una clase **FS** que es capaz de almacenar **Dato** en un fichero binario sobre un sistema de ficheros NTFS.

Si en el código de **Logica** escribimos literalmente el nombre de la clase **MyBD** como colaborador para persistir datos, ¿Cómo haremos cuando necesitamos cambiar la base de datos por ficheros binarios en disco?. No quedará otro remedio que modificar el código de

## Logica.

Si las clases **MyDB** y **FS** implementasen una misma interfaz **IPersistor** para guardar **Dato**, podríamos limitarnos a usar **IPersistor** (que es una abstracción) en el código de **Logica**. Cuando los requerimientos exigiesen un cambio de base de datos por ficheros en disco o viceversa, sólo tendríamos que preocuparnos de que el atributo **\_myPersistor** de la clase **Logica**, que es de tipo **IPersistor** contuviese una instancia de **MyDB** o bien de **FS**.

¿Cómo resolvemos esta última parte?. Con la inyección de dependencias, que es una técnica que como ya hemos dicho se usa para obtener la Inversión de Dependencias.

Como se ha comentado la arquitectura de la aplicación divide a esta entre capas que saben lo mínimo unas de otras y que se comunican por interfaces, a modo de contrato para ocultar tras esto la implementación de las clases que cumplirán dicho contrato.

## 5.3. Diagrama de Clases

Como se ha comentado la arquitectura de la aplicación divide a esta entre capas que saben lo mínimo unas de otras y que se comunican por interfaces, a modo de contrato para ocultar tras esto la implementación de las clases que cumplirán dicho contrato.

### 5.3.1. Capa de Datos

La capa de datos se comunica con el dominio implementando las clases Repository de cada entidad, que marcan las operaciones a realizar con las mismas. Tras implementar dicho interfaz, aplica un a su vez un patrón repository para definir las operaciones con las entidades y auxiliares. Mediante una factoría de DataStores decidiremos la estrategia para acceder o guardar las entidades, por ejemplo podríamos establecer una implementación mediante caché, si el elemento esta en caché la factoría devolverá un CacheDataStore y si por contra no se encuentra en caché, devolverá un DatabaseDataStore para traerlo de la base de datos, actualizando la caché. Esta capa maneja las entidades Entity que transforma en entidades de dominio cuando le solicitan operaciones.

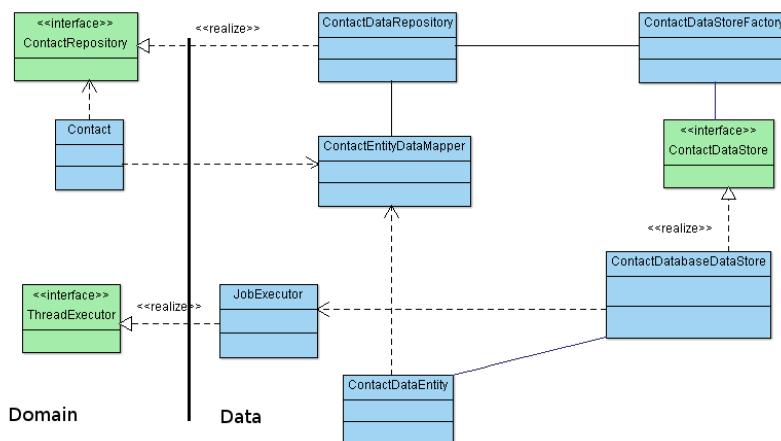


Figura 5.2: Diagrama de clases de la capa de datos

### 5.3.2. Capa de Dominio

La capa de dominio esta basada en los casos de uso de la aplicación. Estos son interfaces que heredan de Interactor, que implementa Runnable. De esta manera las implementaciones de los mismos contienen un ThreadExecutor implementado por JobExecutor en la capa de datos, al que se envian a si mismos para ejecutar la acción del caso de uso. Por el otro lado, son los diferentes presenters de la aplicación, quienes implementan los diferentes casos de uso.

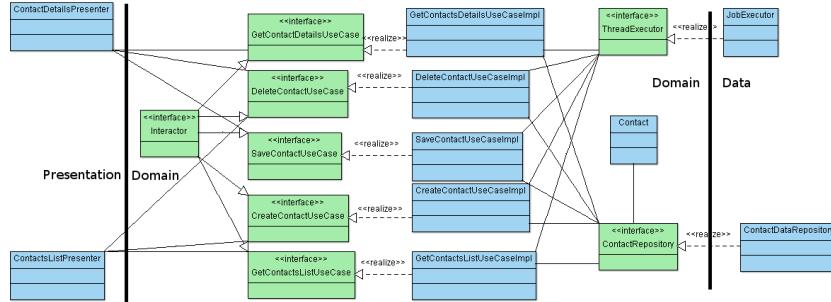


Figura 5.3: Diagrama de clases de la capa de dominio

### 5.3.3. Capa de Presentación

Esta capa se relaciona con el dominio mediante los presenters, que contienen las operaciones de los diferentes casos de uso que se relacionan en su vista. Ademas contienen interfaces sobre las clases de vista para que estas soporten los metodos que los propios presenters necesitan.

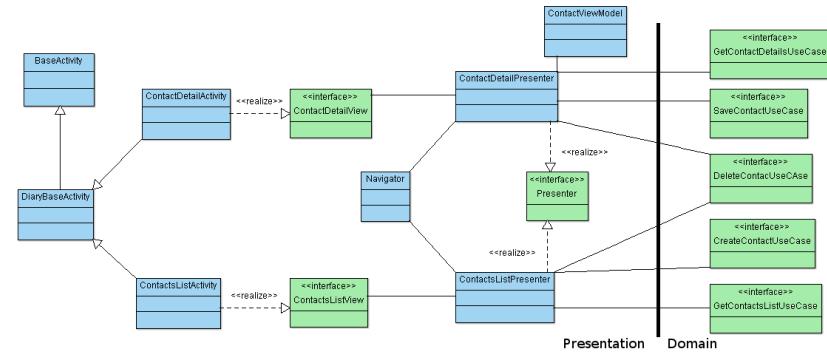


Figura 5.4: Diagrama de clases de la capa de presentación

## 5.4. Diagrama de Clases

### 5.5. Diagrama E/R de la base de datos

Para el diseño de la base de datos incluida dentro de la aplicación y que va a utilizarse para tratar la información introducida por el usuario, hemos recreado un diagrama sencillo de tipo Entidad-Relación en el que se muestran las entidades que serán llevadas a tablas y

sobre las que se realizarán las típicas operaciones de creación, lectura, actualización y borrado (CRUD).

### 5.5.1. Diagrama

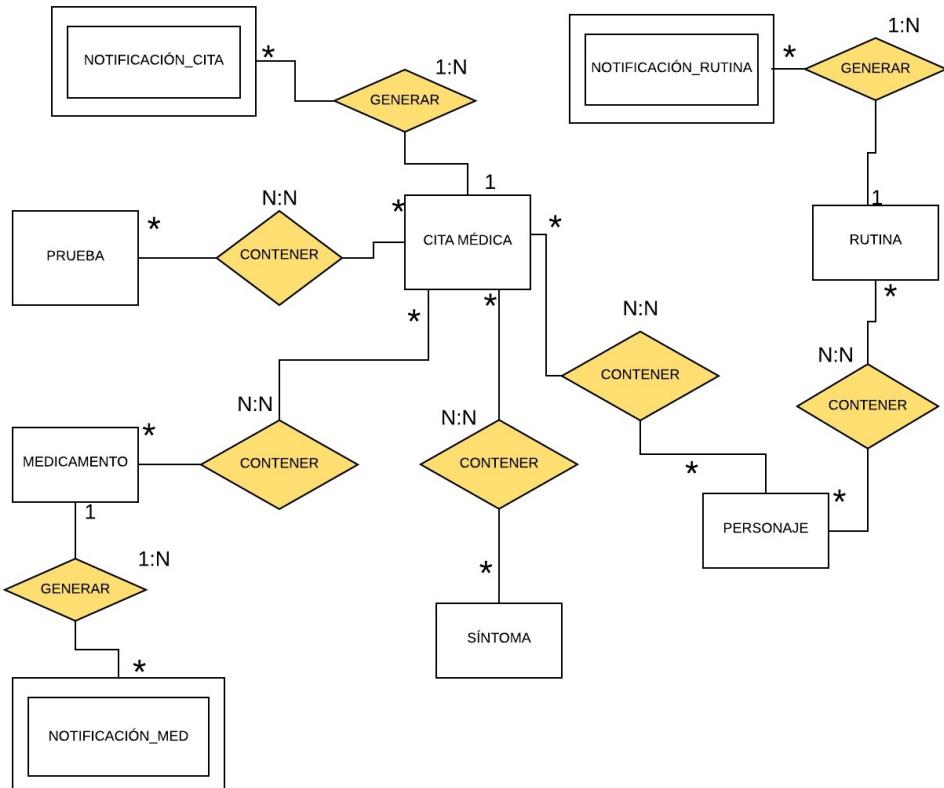


Figura 5.5: Diagrama entidad relación de la base de datos local

### 5.5.2. Descripción

El diagrama de la figura anterior[?] es el diseño en papel de la base de datos local, la que tendrá el usuario en su móvil en la aplicación.

Consta de varias entidades fuertes o formales:

- Cita médica; Sus atributos son id (PK), nombre, lugar, descripción, día de cita, hora de cita, día a notificar , hora a notificar, duración e imagen.
- Rutina; Sus atributos son id (PK), nombre, lugar, descripción, día de rutina, hora de rutina, día a notificar, hora a notificar, duración, satisfacción e imagen.
- Personaje; Sus atributos son id (PK), nombre, apellidos, dirección, teléfono, relación, imagen.
- Medicamento; Sus atributos son id (PK), nombre, descripción, día inicio, hora inicio, día fin, hora fin, intervalo, imagen.

- Síntoma; Sus atributos son id (PK), nombre, descripción, día, hora, imagen.
- Prueba; Sus atributos son id (PK), nombre, descripción, día, hora, imagen.

También encontramos en el mismo varias entidades débiles:

- Notificación medicamento; Sus atributos son id (PK), id del medicamento (FK), día, hora, descripción
- Notificación cita; Sus atributos son id (PK), id de la cita (FK), día, hora, descripción
- Notificación rutina; Sus atributos son id (PK), id de la rutina (FK), día, hora, descripción

Las relaciones que se producen entre las distintas entidades, son las siguientes:

Una cita médica puede contener algún o ningún Medicamento, Prueba, Personaje, Síntoma. Una rutina puede contener algún o ningún Personaje. Síntomas, personajes, pruebas, medicamentos pueden estar contenidos en varias citas o en ninguna. Los personajes pueden estar contenidos en citas o rutinas.

Las citas, rutinas y los medicamentos pueden generar o no notificaciones.

### 5.5.3. Capa de Datos

La capa de datos se comunica con el dominio implementando las clases Repository de cada entidad, que marcan las operaciones a realizar con las mismas. Tras implementar dicho interfaz, aplica una vez un patrón repository para definir las operaciones con las entidades y auxiliares. Mediante una factoría de DataStores decidiremos la estrategia para acceder o guardar las entidades, por ejemplo podríamos establecer una implementación mediante caché, si el elemento esta en caché la factoría devolverá un CacheDataStore y si por contra no se encuentra en caché, devolverá un DatabaseDataStore para traerlo de la base de datos, actualizando la caché. Esta capa maneja las entidades Entity que transforma en entidades de dominio cuando le solicitan operaciones.

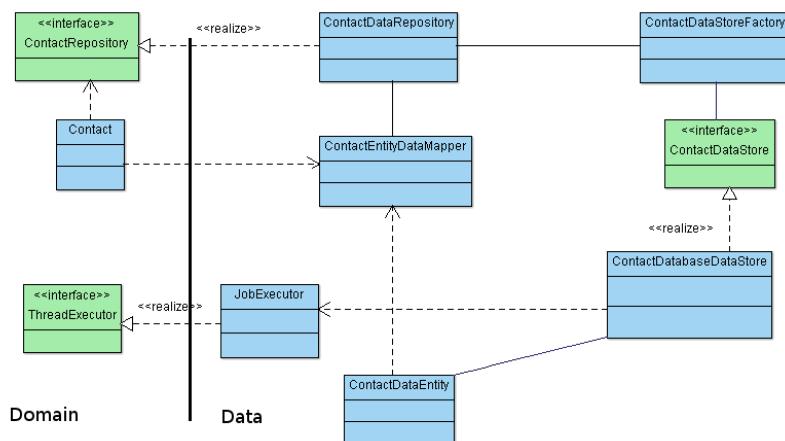


Figura 5.6: Diagrama de clases de la capa de datos

### 5.5.4. Capa de Dominio

La capa de dominio esta basada en los casos de uso de la aplicación. Estos son interfaces que heredan de Interactor, que implementa Runnable. De esta manera las implementaciones de los mismos contienen un ThreadExecutor implementado por JobExecutor en la capa de datos, al que se envian a sí mismos para ejecutar la acción del caso de uso. Por el otro lado, son los diferentes presenters de la aplicación, quienes implementan los diferentes casos de uso.

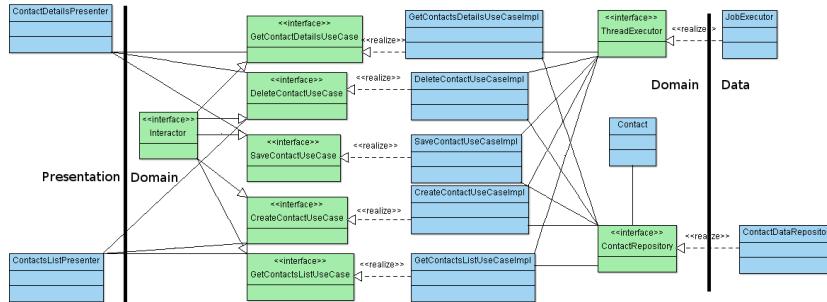


Figura 5.7: Diagrama de clases de la capa de dominio

### 5.5.5. Capa de Presentación

Esta capa se relaciona con el dominio mediante los presenters, que contienen las operaciones de los diferentes casos de uso que se relacionan en su vista. Ademas contienen interfaces sobre las clases de vista para que estas soporten los metodos que los propios presenters necesitan.

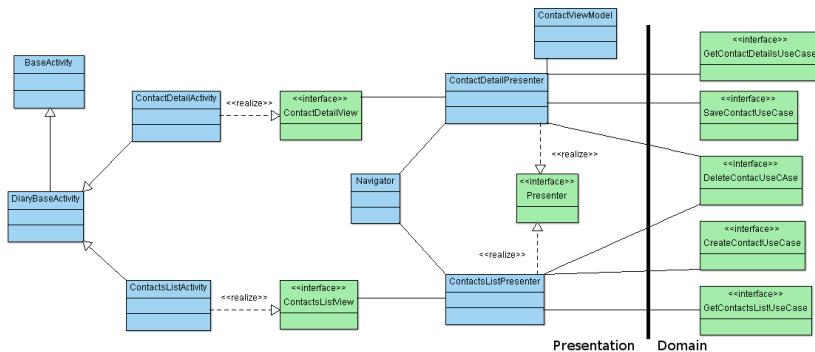


Figura 5.8: Diagrama de clases de la capa de presentación

## 5.6. Diseño de la base de datos

### 5.7. Prototipado

En la fase de diseño, el propósito del prototipo es obtener una primera versión de la apariencia de la interfaz de usuario así como de la funcionalidad incluida (mostrar las ventanas, su navegación, interacción, controles y botones). Con esto se pretende que el cliente tenga

una primera toma de contacto con la futura aplicación antes de su desarrollo final, para así reducir o eliminar todas aquellas disconformidades y cambios en fases futuras.

### 5.7.1. Material Design

Material Design es la nueva metáfora visual de Matias Duarte para el sistema operativo móvil Android, en su versión 5.0 y posteriores, y todo el ecosistema Google en web y supone una pequeña ruptura con la anterior metáfora de Android llamada Holo.

Es un diseño donde la profundidad, las superficies, los bordes, las sombras y los colores juegan un papel principal.

Precisamente este diseño basado en objetos es una manera de intentar aproximarse a la realidad, algo que en un mundo donde todo es táctil y virtual es difícil. Material Design quiere guiarse por las leyes de la física, donde las animaciones sean lógicas, los objetos se superpongan pero no puedan atravesarse el uno al otro y demás.

Material Design es un diseño con una tipografía clara, casillas bien ordenadas, colores e imágenes llamativas para no perder el foco y un sentido del orden y la jerarquía muy marcado. Estas ideas ya se aplican en muchos diseños, pero en Material Design Google ha creado unas normas muy claras de como llevarlo a la práctica.

### 5.7.2. Pantallas Principales

Pantallas iniciales y flujo de la aplicación que de manera inicial hemos planteado para la misma. No es la versión final de la aplicación pero si se aproxima y nos da una idea clara de lo que debería ser o a lo que debería parecerse.

Estos prototipos, sirven para orientarnos a la hora de construir la aplicación y sirven para que el cliente o product owner se haga una idea de hacia donde irá el desarrollo de la misma. Muchas de las pantallas proceden de una adaptación del libro que se utiliza en la AECC 'MIS CUIDADOS, diario de salud para supervivientes de cáncer'.



Figura 5.9: Iniciativa de la AECC

Otras de las pantallas y funcionalidades, se han adaptado para facilitar el uso, adaptarse a los requerimientos del cliente y ampliar el rango de acción de este libro y hacerlo más lógico a nuestro entender.

#### Pantalla Principal y Drawer menú

El usuario, al ejecutar la aplicación accede a una pantalla cuyo contenido es el listado de las actividades próximas a realizar, ya sea una cita médica o una rutina. En ausencia de las mismas, nos mostrará un mensaje alentándonos a realizar alguna rutina.

El drawer nos permite navegar entre actividades, desde la pantalla principal podemos añadir citas y acceder de manera individual a las mismas.

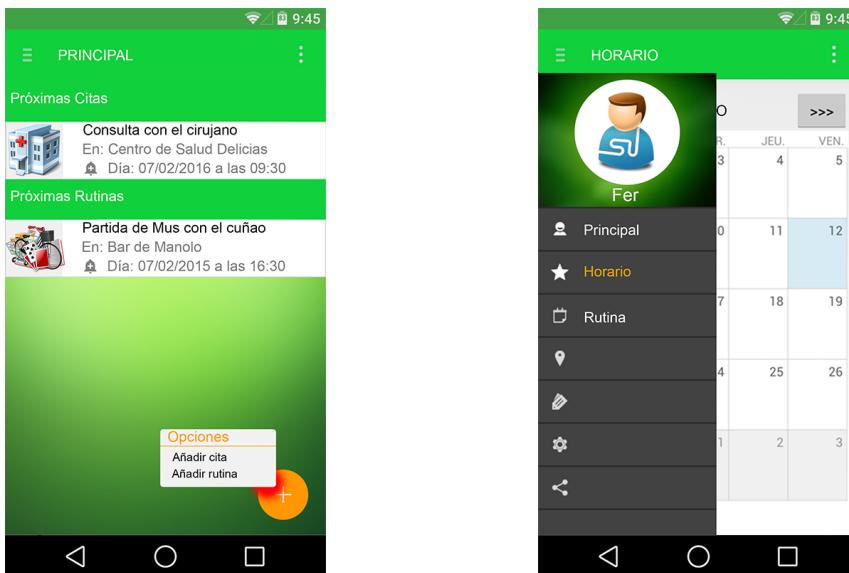


Figura 5.10: Descripción de la pantalla principal y Drawer menú

### Pantallas referentes a la programación de actividades

En la iniciativa de la asociación, se proporciona un cuadro para poder controlar las actividades que realizamos de manera rutinaria a lo largo de los días de la semana, ampliamos esta funcionalidad para que las citas se representen también en el horario.

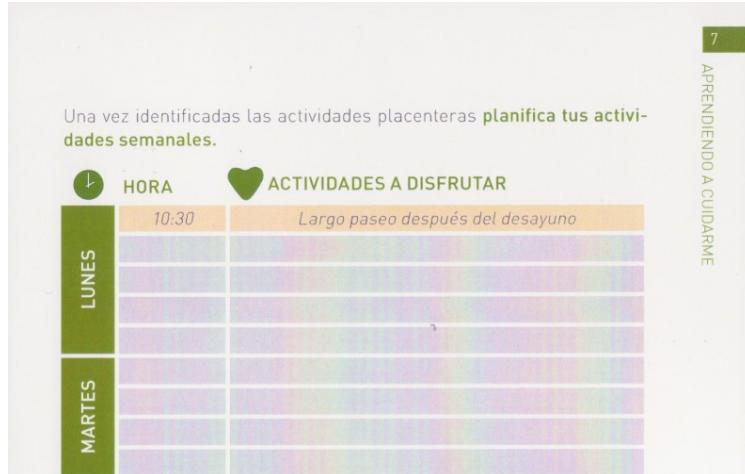


Figura 5.11: Vista de los días en el folleto

Desde estás pantallas tenemos una visión de la ocupación diaria del paciente y de la ocupación mensual, desde la vista mensual al menos se podrán añadir nuevas citas médicas o rutinas para ese paciente.

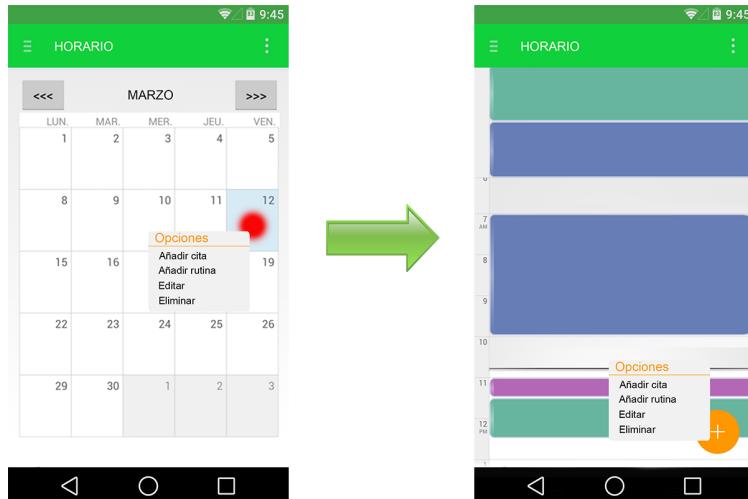


Figura 5.12: Pantallas con los horarios y ocupación del paciente

### Pantallas referentes a la rutina diaria

Estas funcionalidades descienden directamente del libro proporcionado por la AECC, en el que como usuario integrado en el programa de mejora de la calidad de vida del superviviente de cáncer se le insta a realizar actividades que le produzcan cierta satisfacción y a realizarlas con asiduidad, sea en compañía o solo. En resumen se recomienda hacer actividades que nos hagan felices y nos animan a puntuarlas con el grado de satisfacción que nos producen las mismas. En el extracto del libro podemos apuntar la actividad, la fecha en la que la vamos a realizar, indicar quien nos acompaña, grado de satisfacción que nos produce y repetir.

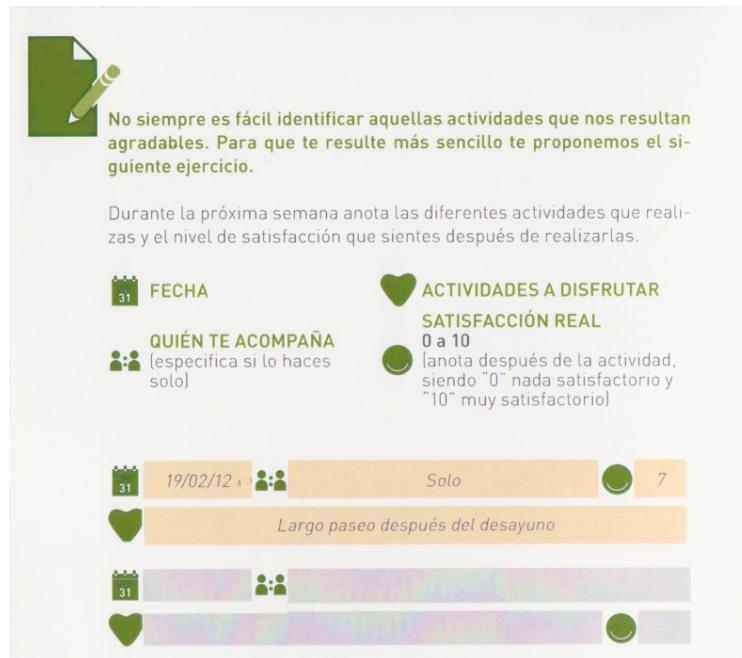


Figura 5.13: Cuadro de actividades a realizar

Para la adaptación y mejora de las especificaciones del libro utilizado en la iniciativa, se ha diseñado un listado de las rutinas diarias que tiene ese paciente y representación de la introducción de una nueva rutina para ese paciente.

Dentro de la introducción de la rutina se podrán añadir nuevos personajes que acompañarán a ese paciente y le harán más fácil recordar con quien quedó y para qué.

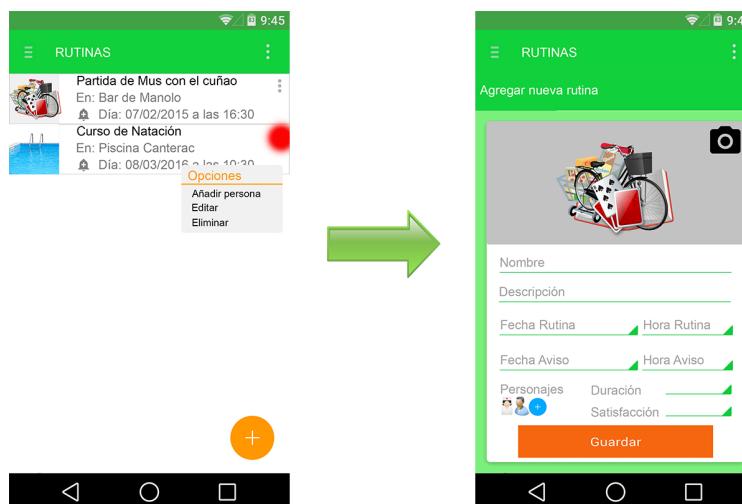


Figura 5.14: Pantalla con la interfaz para las rutinas

## Pantallas referentes a las citas médicas

Las páginas referentes a las citas médicas se basan en una serie de recomendaciones sobre como afrontar la visita al hospital, centro de salud, oncólogo y demás especialistas. Nos aporta una lista de cosas que deberíamos preguntar al especialista para rebajar la ansiedad ante la misma y sacar el máximo partido a la misma.

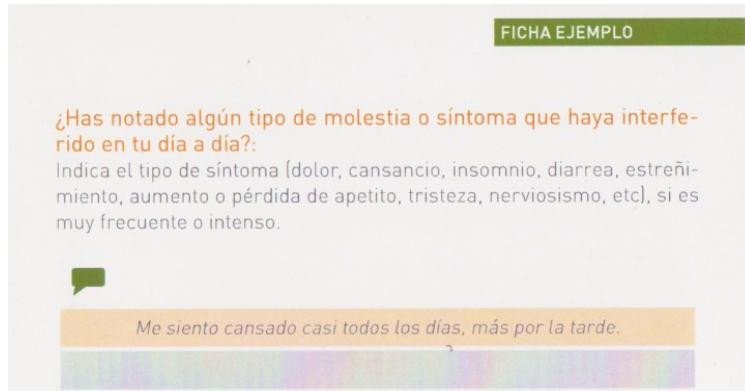


Figura 5.15: Página con información a completar en la cita médica

Listado de las citas medicas de ese paciente e introducción de la cita en si de manera individual.

De funcionamiento similar a las pantallas anteriores pero con diferencias sustanciales en la funcionalidad de la introducción de información, se pueden añadir personajes, síntomas, medicamentos y pruebas, de manera que el paciente cuando acuda a la cita lleve de manera ordenada toda esta información.

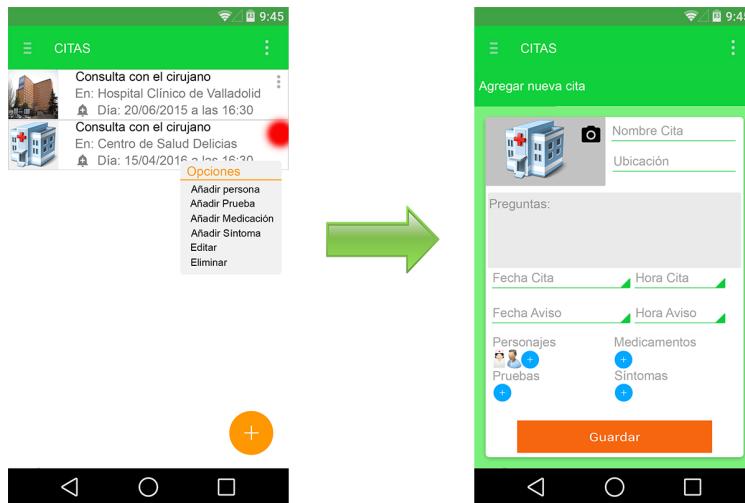


Figura 5.16: Pantalla con la actividad de las citas

### Pantallas referentes a la medicación

El libro nos proporciona un cuadro para el control de los medicamentos, nos permite apuntar el nombre sus indicaciones y parte de la posología del mismo, pero a nuestro modo

de ver es algo insuficiente y se puede completar con notificaciones y más información que le permita ser identificado fácilmente.

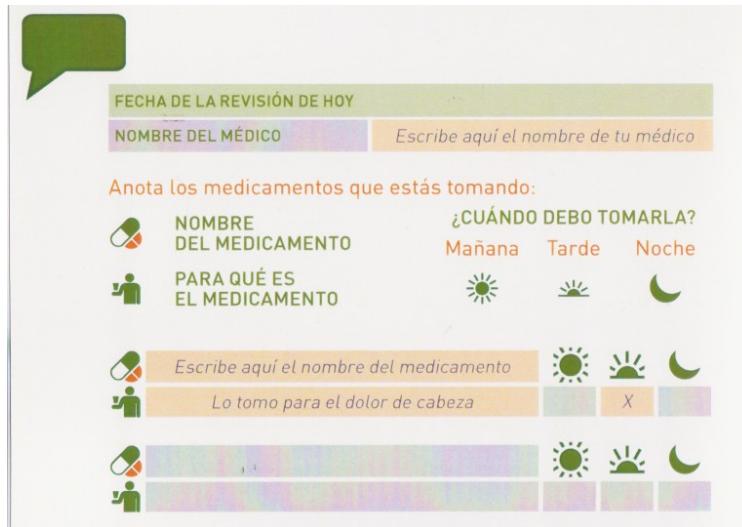


Figura 5.17: Medicación y posología de los medicamentos

El usuario introduce los medicamentos que toma, de manera que puede añadir alertas a los mismos para que se le avise de que ha llegado la hora de la dosis con una notificación. Se podrán introducir otros datos del medicamento como pueden ser los intervalos entre dosis, indicaciones...

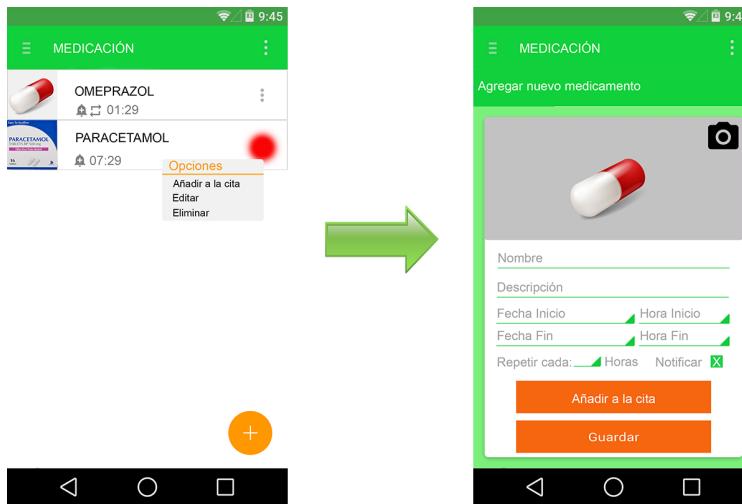


Figura 5.18: Pantalla con la actividad de la medicación

**Pantallas de gestión de personajes** El usuario puede añadir a la aplicación su lista de amigos, personal médico que le atiende, asistentes, etc, con sus datos personales, de manera que a la hora de hacer una rutina (actividad) o acudir a una cita, puedan ser añadidos.

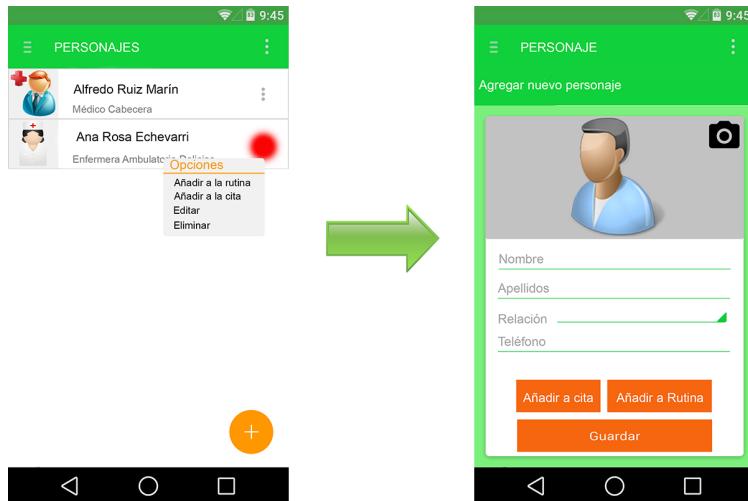


Figura 5.19: Pantalla con la actividad de los personajes

**Pantallas de gestión de pruebas médicas** Se facilita el poder llevar las pruebas médicas en forma de archivo fotográfico para poder consultarse en alguna de las citas médicas a las que acuda, tales como revisiones, visitas al fisio, a la enfermera. Listado del mismo.

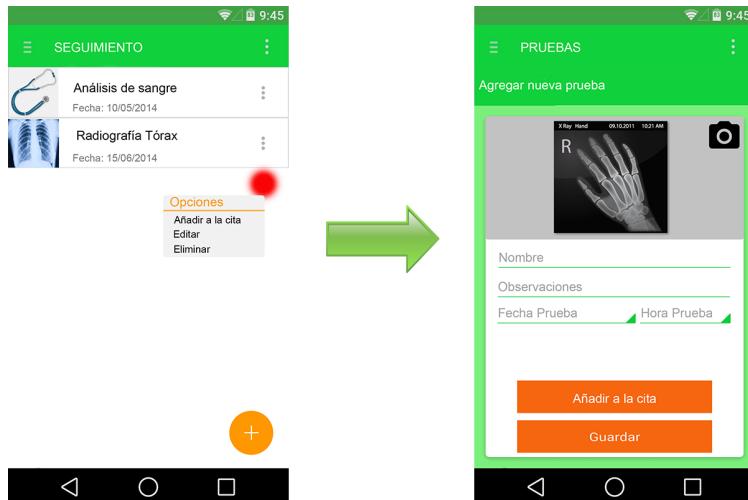


Figura 5.20: Pantalla con la actividad de las pruebas

### Pantallas de gestión de los síntomas

Los síntomas se apuntan en la misma ficha que los datos de la cita médica, con lo cual sería interesante que toda esa información se viese representada en la aplicación, hay que tenerla en cuenta en el flujo.

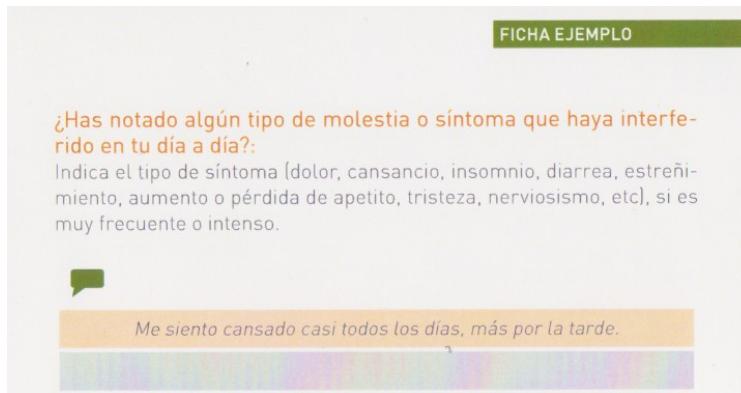


Figura 5.21: Cuadro de los síntomas en el libro

El usuario puede tomar nota de los síntomas, molestias, efectos secundarios negativos o positivos que le acontecen, para poder asignarles después a una cita médica y poder completar de manera más eficiente las mismas, pudiendo aportar información adicional que ayude a su médico a mejorar el diagnóstico.

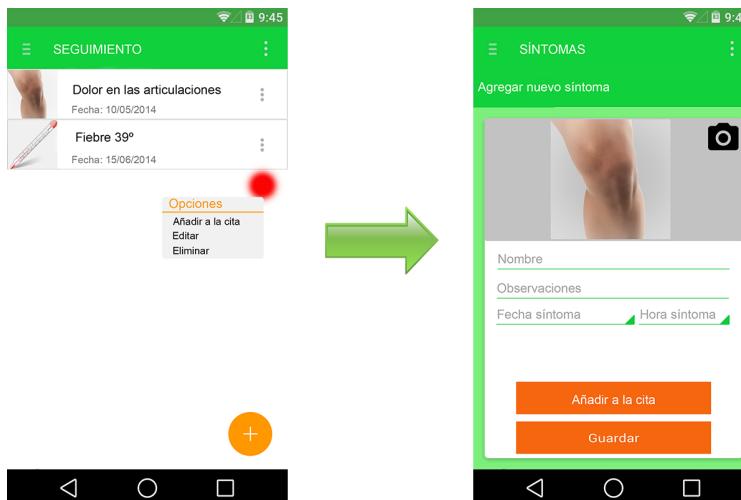


Figura 5.22: Pantalla con la actividad de los síntomas

## Pantallas de Recursos

A parte de todos los consejos que se dan en el libro sobre como afrontar la fase posterior a la enfermedad, se hace especial hincapié en la meditación, necesaria para proporcionar un estado relajado al superviviente, ante citas medicas importantes y que le generen tensión.



Figura 5.23: Ejercicios de meditación

Otra parte importante del libro es un listín telefónico en el que se pueden apuntar los contactos importantes, dirección, y números de teléfono. Toda esta parte se contempla en nuestra aplicación a través de los personajes y de los teléfonos de interés de la sección de recursos.



Figura 5.24: Contactos importantes a tener en cuenta

Estas pantallas están pensadas para que el paciente disponga de ayuda adicional, y una serie de recursos de fácil acceso así como una sección de noticias de la AECC para que puedan ser consultadas por el paciente. Sección de meditación, consejos generales que le puedan ser de ayuda, noticias y teléfonos de interés.

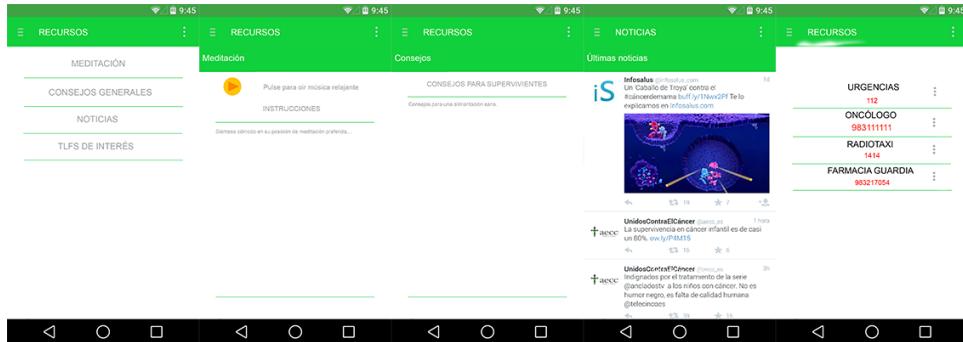


Figura 5.25: Pantalla con la actividad de los recursos

**Pantalla de perfil de usuario** El usuario podrá introducir sus datos personales para que la aplicación pueda tratarle de manera personalizada.

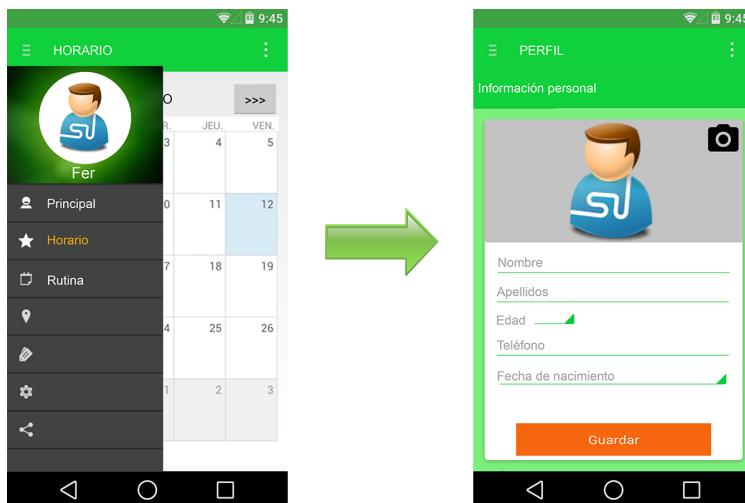


Figura 5.26: Pantalla con la actividad configuración del perfil

**Otros: Pantalla de presentación y carga** Pantalla inicial de carga, se utiliza para aportar más identidad a la aplicación y a modo de pequeña intro.

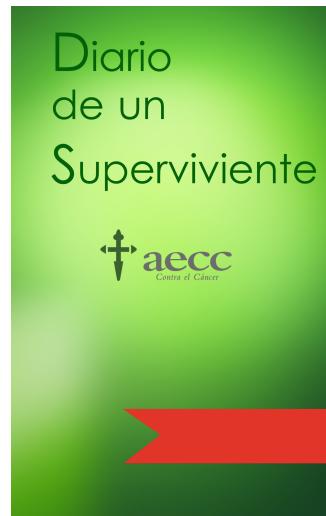


Figura 5.27: Pantalla de presentación de la aplicación

## 5.8. Iconografía y otros cambios

A petición del Product Owner y dado que estamos trabajando con una metodología agil, los cambios y las adaptaciones en el diseño no son una excepción, los prototipos han ido variando en el tiempo con los cambios propuestos por el cliente y se ha adaptado para proporcionar una imagen más 'corporativa' y en consonancia con la iniciativa en la que está situado.

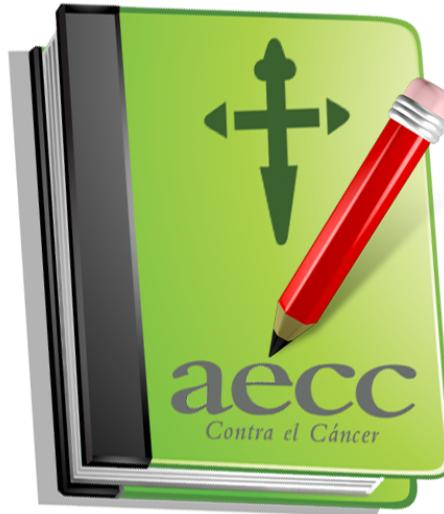


Figura 5.28: Icono de la aplicación principal

La imagen visible de la aplicación, incluso antes de acceder a la misma, va a ser su icono, este nos tiene que dejar entrever parte de la intención de la aplicación y ser fácilmente identificable entre el resto de los iconos de las aplicaciones instaladas.

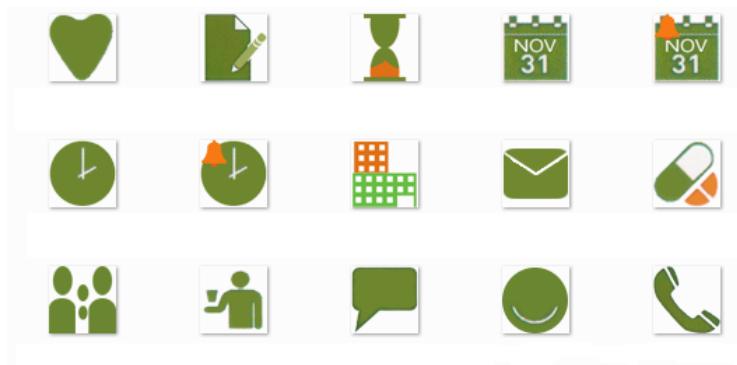


Figura 5.29: Conjunto de iconos de apoyo para las pantallas de las entidades

Se diseñaron gran parte de los iconos que forman parte de las pantallas, con un aspecto que recuerda al del libro y con un estilo acorde al que se utiliza en el mismo folleto, para que el usuario tenga una experiencia de uso que le muestre continuidad.

También se re-diseñaron algunas de las pantallas principales para hacerlas más fácilmente identificables con el libro de la iniciativa de la asociación y mantener la coherencia visual del

proyecto.

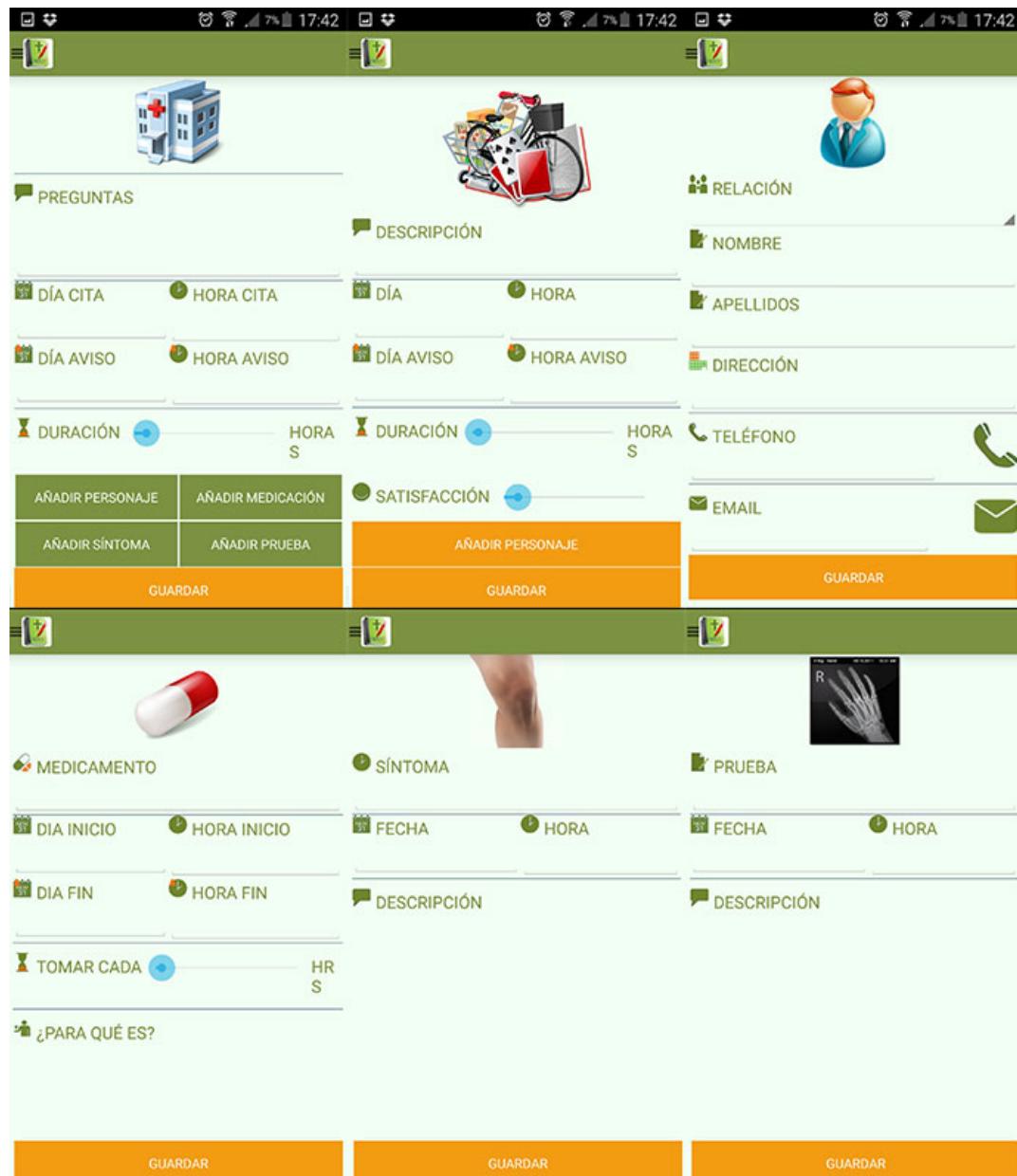


Figura 5.30: Nuevo diseño de las pantallas generales de la aplicación

Por último, la parte de recursos sufrió algunas modificaciones, la parte en la que se insertarían noticias de la AECC se ha suprimido para no perder el foco de la verdadera utilidad de la aplicación y dado que la asociación dispone de otros canales para la propagación de las mismas, se consideró oportuno su supresión en esta versión.

Con lo cual, recursos ha quedado de la siguiente manera, una pestaña de consejos generales, una pestaña de meditación y relajación y una pestaña de asociaciones y páginas de interés.



Figura 5.31: Nuevo diseño de los recursos



# CAPÍTULO 6

---

## Implementación y Pruebas

---

### 6.1. Construcción

Para la implementación en un principio las metodologías ágiles no marcaban ninguna pauta más allá de que todo lo que no está terminado al 100 % no está terminado y por otro lado que siempre hay que entregar al cliente un trabajo con la máxima calidad posible, aunque quizás la más importante fue que la propiedad del código es colectiva, ya no hay más código mio o código tuyo y tenemos responsabilidades y atribuciones sobre nuestras parcelas, el código es del equipo entero y todo miembro puede y debe mejorar el mismo si observa y detecta un error. Seguimos la regla del boy scout de Clean Code[34], que nos dice "Deja el campamento más limpio de como lo encontraste".

Después de un tiempo de maduración de las mismas surgieron técnicas y disciplinas, algunas en el seno de alguna de estas metodologías (XP sobre todo) y algunas se incorporaron rápidamente desde el movimiento Craftmanship software [14] que tiene una filosofía similar aunque más cercana y centrada en el proceso de construcción en sí que en el resto del proceso de desarrollo del proyecto o producto. Estas técnicas son Pair Programming, TDD, Code Reviews y algunas centradas en la mejora del equipo en todos los proyectos como puedan ser Code Retreats, Katas, Koans, etc. todas estas técnicas y disciplinas hoy en día se consideran tan apegadas a las metodologías ágiles que prácticamente se confunden con las mismas cuando hablamos del proceso de implementación.

Lo que si existe es la idea del visual management, o tener algún tipo de respaldo visual del estado de un proyecto. Para esto existen pizarras en las que mediante el uso de tarjetas, imanes, post its, imágenes e incluso avatares de los miembros del equipo, permiten que cualquier persona interesada en el proyecto con solamente dar un vistazo a la pizarra vea el estado del sprint actual, en que trabaja cada miembro del equipo, el estado del backlog, las historias completadas, y demás información que cada equipo adapta a sus necesidades.

Recordemos que la idea es que el equipo y el product owner se sientan cómodos con la metodología y la adapten a sus necesidades o idiosincrasia, teniendo en cuenta que no hay balas de plata que sirvan para todo y arreglen los problemas de la noche a la mañana.

Existen también multitud de utilidades de software que permiten esto mismo para equipos remotos o para aquellos que no les es posible disponer de una pizarra física. Esta ha sido la opción utilizada por nosotros a través de la aplicación web Trello, debido a la composición distribuida tanto del equipo, como de cada uno de los diferentes product owner de parte de

la AEEC



Figura 6.1: Logo de la aplicación Trello

Esta app permite el uso de la pantalla como si de una pizarra electrónica se tratase, no es necesario compartir un espacio físico para ello y de manera sencilla se puede compartir con todos los miembros que forman parte del equipo. Para nuestro propósito la dividimos en diferentes columnas que representarán los estados de una historia de usuario y además algunas zonas para colocar una imagen del burndown, el estado de animo del equipo y del product owner. La explicación de las columnas y como adaptamos nuestra manera de desarrollar y de llevar scrum a Trello se hace más abajo.

A screenshot of a Trello board titled 'AECC ANDROID PFC'. The board has several columns: 'Product Backlog' (54 items), 'ToDo' (14 items), 'Doing' (11 items), 'Done' (35 items), 'Tested' (27 items), 'To Ship' (2 items), and 'Shipped' (72 items). Each column contains various cards with descriptions and due dates. On the right side of the board, there is a sidebar with 'Calendar', 'Move menu', and other board settings. Below the board, there is a note: 'Diseño inicial de una pantalla, con cuatro tonterías, botón, parte de menú, con los colores que utilizamos... para enseñarlos.' and a link to 'Diseño inicial'.

Figura 6.2: Vista general de la columnas de scrum en Trello

Las columnas que definimos como estados para las historias de usuario fueron:



Figura 6.3: Detalle de las columnas product backlog y ToDo

- Product Backlog, es el backlog de la aplicación del que se habló en el Capítulo 3. Debe estar priorizado para que el equipo tenga una visión aproximada de los intereses del product owner en futuros sprints, pero el equipo ha de ser consciente de que esto puede ser modificado por el product owner en cualquier momento. La priorización de las historias y dada la facilidad con la que Trello nos permite mover las tareas en la misma columna y entre columnas se realizó de la siguiente manera, la más prioritaria es la que ocupa el primer lugar en esa columna, y es con la que se empezará una vez se le asigne un sprint.
- Sprint Backlog o ToDo, son las tareas del sprint en curso o por hacer. Pueden estar priorizadas por el product owner (del modo que hemos comentado antes) para que el equipo tenga cierta idea de lo que es más valioso para él, pero el equipo es libre del orden de implementación de las historias en un sprint, en esta columna en un principio se encuentran todas las tareas que forman parte de ese sprint, y será el principal punto de entrada a la hora de asignar tareas al equipo.



Figura 6.4: Detalle de las columnas Doing, Done, Tested y To Ship

- Doing, es la columna de las historias que actualmente se encuentran en proceso de implementación por algún miembro del equipo.
- Done o ToTest, tareas que el equipo ha dado por terminadas con sus test unitarios y prueba de aceptación o definition of done completado, pero no por alguien ajeno al desarrollo de la misma, suele ser el mismo miembro del equipo asignado a la tarea el que la pasa a esta columna.
- Tested, alguien ajeno al desarrollo de la historia de usuario ha testeado el criterio de aceptación de la historia, esto puede ser llevado a cabo por el scrum master o algún miembro del equipo designado para ello, al estar nuestro equipo formado por dos personas solemos cruzar la comprobación de las mismas.
- ToShip, historias que se han probado por alguien ajeno a quien la desarrolló, pero que aun no han sido entregadas o desplegadas para demo o validación por el product owner, hay funcionalidades que se acaban pero que por alguna razón no se quieren llevar a la demo en curso y permanecen en esa columna durante algún tiempo.

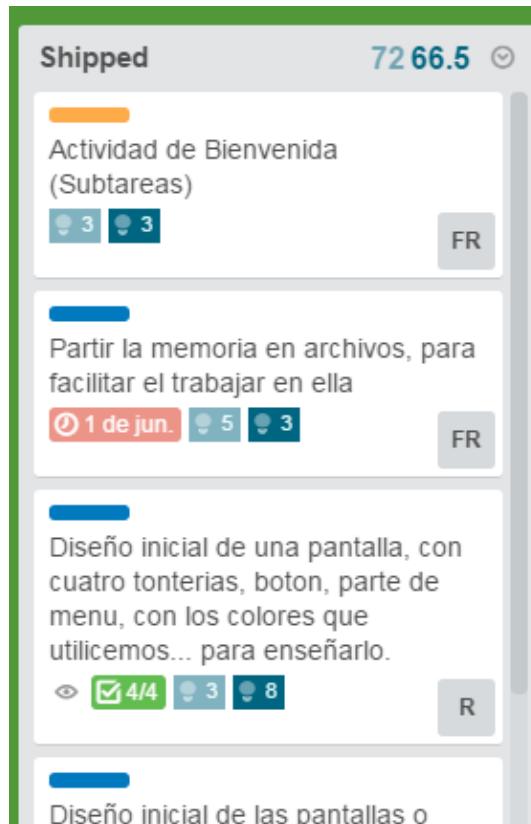


Figura 6.5: Detalle de la columna Shipped

- Shipped, historias que fueron entregadas o desplegadas en otros sprints o en una demo, idealmente no deberían salir de esta columna ya que como se dice han sido validadas por el product owner, pero pudiera ser que se presenten errores en test de regresión al añadir nueva funcionalidad. En ese caso pasarían al Product Backlog para el siguiente sprint como historia de defecto, tal y como esté acordado por el equipo.

La transición de las historias que se escriben en tarjetas físicas a las historias virtuales con las que trabajamos en Trello se ha realizado de una forma sencilla. La tarjeta consta de varias partes como son el título, los puntos de Historia que le asignamos tras previa votación (debajo se encuentran los que ha llevado acabarla una vez este hecha), el miembro del equipo o los miembros que se van a encargar de su elaboración, la descripción, Subtareas que se pueden añadir en forma de checks, comentarios,...

Los puntos de historia estimados y los completados, nos ayudan a la hora de estimar el ¿Cómo vamos? ya que nos va haciendo un recuento sobre la columna de los que quedan y los que se han hecho. También se le puede añadir a la tarea una fecha de vencimiento a modo de "Dead Line", para fijar una fecha de antemano en la que esa tarea debería estar completada y en la columna "ToShip".

Otra de las funcionalidades que nos aporta Trello, es la suscripción, algo muy útil a la hora de recibir un email con su resolución, o cambio de columna y que facilita el flujo de trabajo sobremanera.

El etiquetado lo utilizamos sobre todo para indicar a que sprint pertenece la tarea y a la hora de realizar búsquedas por la misma etiqueta las agrupa, permitiendo revisar las que pertenecen al mismo.

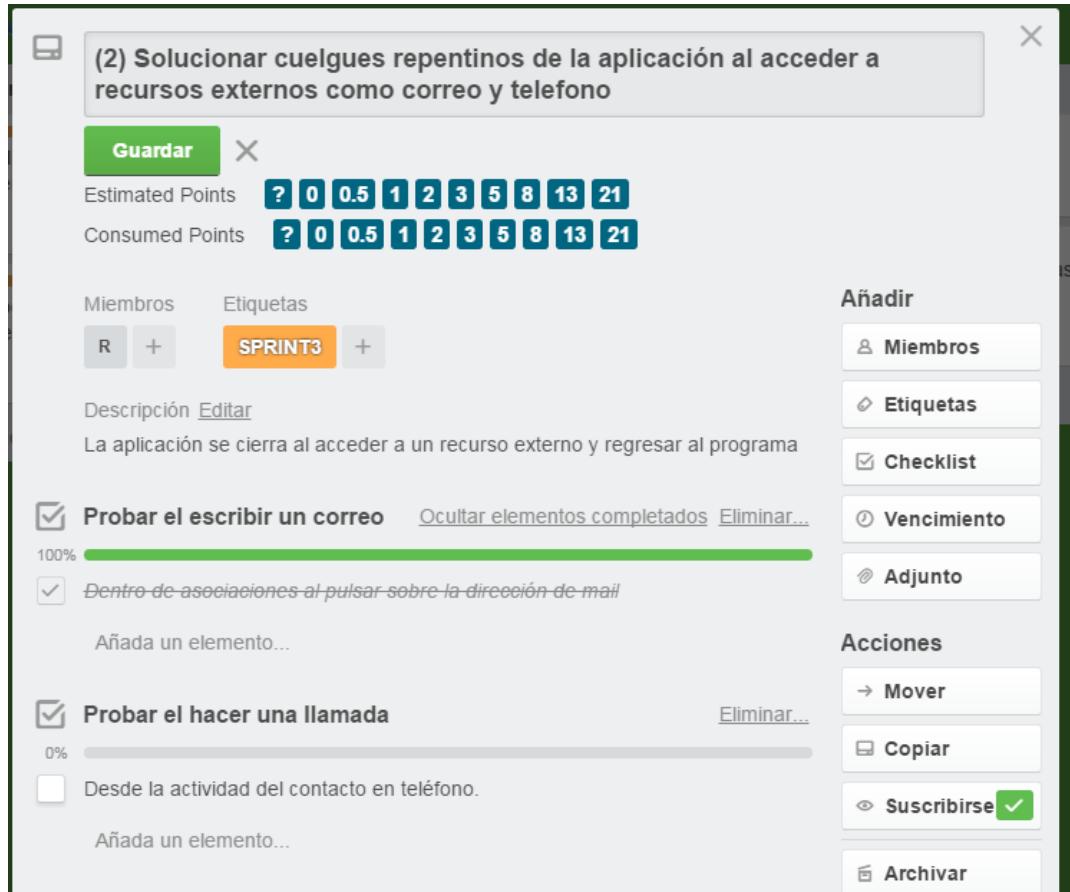


Figura 6.6: Implementación de las Historias de usuario en Trello

Las etiquetas bajo las que señalamos el sprint en el que se encuentra la tarea, nos ayudan a la hora de realizar las búsquedas y son útiles para repasar los puntos de historia que se han completado en ese sprint y poder planificar de manera más eficiente los siguientes. Ya que con la finalización de cada sprint, hay una DEMO y esta debería incluir todas las funcionalidades que en principio se plantearon incluir.

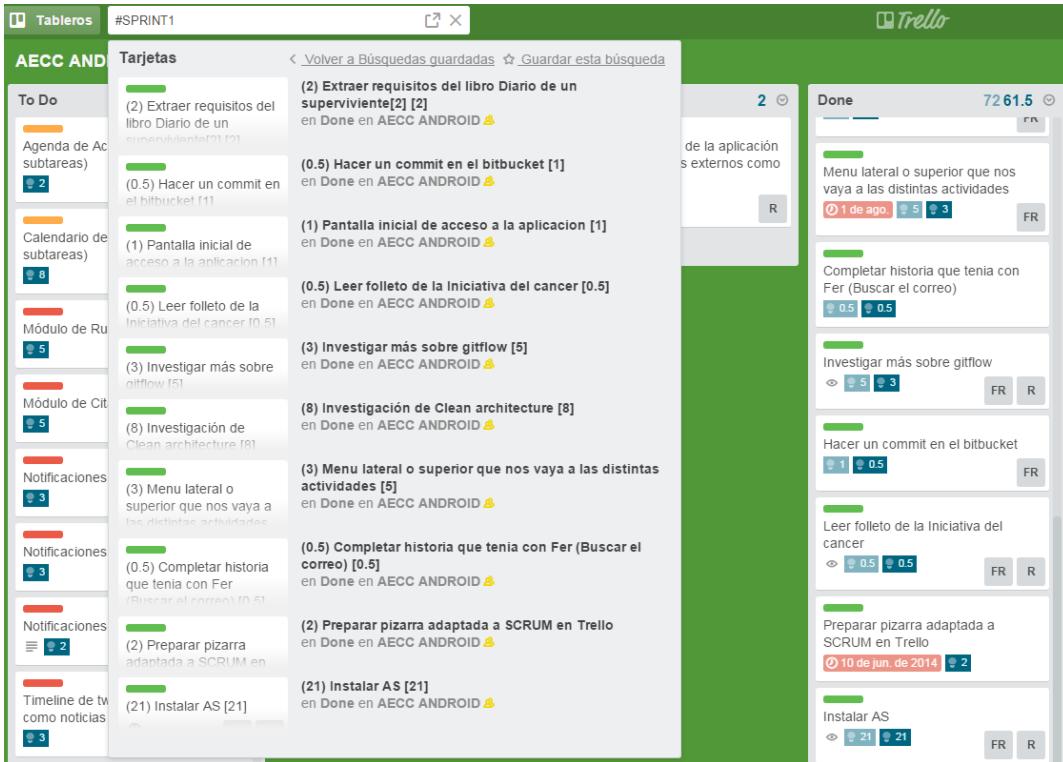


Figura 6.7: Agrupación de tareas por sprints

En todo momento las historias avanzan hacia la derecha de la pantalla, nunca hacia la izquierda. Cada miembro del equipo solo podía estar dedicado a una tarea, ocasionalmente dos miembros del equipo pueden colaborar en la realización de una tarea compleja, o usando la técnica de pair programming.

## 6.2. Plan de desarrollo

Pasos para el desarrollo de la aplicación móvil "Diario de un Superviviente":

### Idea Inicial

Partimos de la 'idea' conformada junto a la AECC de una aplicación Android que permitiese a los enfermos de cáncer llevar de una manera mucho mas sencilla el control de las acciones que acontecen de manera rutinaria en su día a día, tales como la toma de medicamentos, asistencia a citas médicas, control de síntomas y pruebas, rutinas diarias beneficiosas.

Para ello la AECC puso a nuestra disposición un folleto que forma parte de esta iniciativa, pero que dadas sus características físicas se que da algo corto en su planteamiento.

### Captura de requisitos

El proyecto debe estar bien definido, tanto sus objetivos como las funcionalidades que se requieren para que cumpla su cometido. Cuanta mejor definido esté más cerca estaremos de cumplir sus objetivos.

Está tarea, se ha realizado estudiando de manera pormenorizada la documentación que puso a nuestra disposición la AECC dividiendo en partes bien diferenciadas las funcionali-

dades de cada una de las partes de las que se compone esta documentación.

A través del cruce de correos y de alguna que otra reunión presencial se han limado distintas formas de ver algunas partes.

Con la definición del proyecto terminada, es necesario saber el tiempo que nos va a llevar en horas, por lo que hay que valorar el desarrollo. Para ello, será necesario contabilizar y estimar los plazos en horas que nos va a costar cada parte del proyecto. Tanto el plazo como el precio dependerán totalmente de las funcionalidades y del tipo de desarrollo elegido, pues no es lo mismo (ni se obtiene un proyecto de igual calidad) desarrollar apps nativas que híbridas, ni que el proyecto requiera de un complejo Backend orientado a móviles o no requiera siquiera esta parte.

### **Planificación**

Es la primera fase del desarrollo del proyecto. Consiste en tener un programa de trabajo con un desglose de todas las actividades que se van a realizar (desde el diseño hasta las pruebas finales), el plazo estimado de horas que se le va a dedicar cada una de ellas y estableciendo los medios humanos que se van a dedicar para alcanzar los objetivos que se hayan propuesto. En este proceso, que ha de ser continuo se han de reflejar

- Equipos, programas, licencias etc que se vayan a emplear.
- Requerimientos gráficos y fechas límite.
- Necesidades que dependan del cliente (AECC) y fechas para tenerlos disponibles.
- Cambios que puedan ocurrir durante el desarrollo de la app.

Una buena planificación y su actualización es clave para el correcto desarrollo de la aplicación móvil y para su puesta en funcionamiento en la fecha prevista.

### **Diseño UI/UX**

Previo a la implementación es necesario tener totalmente definido el diseño estructural de la app y su comportamiento. Para ello hemos utilizado Photoshop para el diseño inicial, el cual nos mostrará el aspecto y la usabilidad de la aplicación.

El diseño consiste tanto en la confección del aspecto y usabilidad como en la correcta aplicación de las guidelines de diseño de Google de material design[?] materialStructure), además de la correcta adaptación a todas las densidades de pantallas (recordemos que por ejemplo Android tiene MDPI(160 DPI), HDPI(240 DPI), XHDPI(320 DPI), XXHDPI(480 DPI), XXHDPI (640 DPI) y su tratamiento para que sean aptas para la programación.

### **Desarrollo**

Es la programación del proyecto. Esta fase se hará de acuerdo a la tecnología que se haya decidido emplear para cada plataforma de programación y los entornos de desarrollo empleados serán acordes con ello (Android Studio); recordemos que se pueden desarrollar apps nativas o híbridas , y llevará mayor esfuerzo de trabajo en función de lo anterior. A la vista de lo anterior el equipo de desarrollo, de una aplicación, por muy sencilla que sea, puede llegar a estar compuesto por 5 ingenieros informáticos (Android, iOS, Windows Phone, Backend, Frontend) y un diseñador, además del director del proyecto que coordine a todos ellos. De ahí que el coste de una app sea totalmente dependiente de la tecnología que empleemos en el desarrollo y de la complejidad del proyecto en sí.

### **Testing**

Una vez desarrollada la app es necesario hacer un testing profundo de todas las partes del mismo. El testeo se puede dividir en:

- Testeo funcional: para asegurar que la aplicación

trabaja como debería y sigue todos los flujos debidos. -Testeo de rendimiento: para comprobar que el comportamiento de la aplicación bajo ciertas condiciones (múltiples peticiones de acceso simultáneas, poca cobertura, poca batería...) es el correcto. -Comprobaciones de fugas de memoria, cruciales en móviles pues los recursos son mucho más limitados que en programas para ordenadores de sobremesa. Para esta tarea se utilizan habitualmente programas automatizadores de tareas y programas que reportan el código de error, además del testeo manual intensivo.

### Distribución pre-lanzamiento

Previo a la subida a los markets de aplicaciones móviles se pueden hacer distribuciones de las aplicaciones móviles. En Android se puede hacer utilizando el entorno beta de desarrollo Android disponible en la consola de desarrollador.

### Implantación y distribución

A la finalización del desarrollo, el último paso será subirlo a los markets correspondientes. Para este último paso habrá que firmar digitalmente las apps con la cuenta de desarrollador, compilar el paquete y subirlo a Google Play, así como preparar el resto de requisitos necesarios tales como las imágenes, logos, descripciones etc. Requeridos por los markets de apps. A partir de este momento comienza la etapa de mantenimiento de la aplicación, y su escalabilidad, dependiendo de los requerimientos y necesidades futuras de los usuarios o del propio cliente en nuestro caso de la AECC.

Esperamos que aclare las dudas que podáis tener cuando penséis en desarrollar una aplicación móvil.

## 6.3. Versionado y Sincronización

El control de cambios y la gestión de la configuración son piezas claves en el desarrollo de hoy en dia, y los equipos deben poder confiar en un sistema que salvaguarde los cambios del código y que les permita compartir dichos cambios realizados en el ordenador local de cada integrante del equipo.

Es por esto que los sistemas de control de versiones y de gestión de la configuración han sido uno de los puntos claves en las infraestructuras de las empresas dedicadas a la producción de código.

En nuestro caso este pieza toma más importancia si cabe debido a que configuramos un equipo distribuido que se encuentra alejado en el espacio y que no siempre puede coincidir temporalmente para trabajar en el proyecto.

De entre todos los sistemas de control de versiones existentes hoy día, por su capacidad y concepción distribuida desde su inicio nos decidimos por el sistema de control de versiones Git, usado en el desarrollo del kernel de Linux.

Así mismo necesitamos una red de salvamento por si alguno de los integrantes del equipo tenía que volver atrás en la versión de la aplicación que estaba manejando, crear una nueva versión de la aplicación e incluso desestimar todos sus cambios y volver a una versión más estable anterior y empezar desde ahí.

La necesidad de que fuese algo gratuito, que nos ofreciese suficientes garantías, tuviera una gran comunidad detrás y que pudiera integrarse de manera sencilla y compatible con

Android Studio, ademas de que un funcionamiento y rendimiento alto serían deseables así como un acceso web serían características muy deseables en el sistema a utilizar.

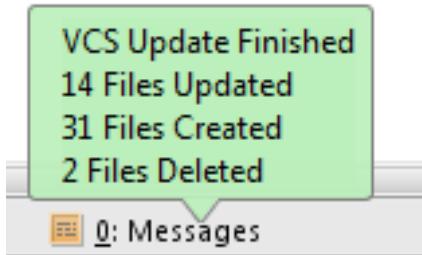


Figura 6.8: Integración de Git en Android Studio

Para ello decidimos utilizar **Git** y **GitHub**. Git es un sistema de control de versiones distribuido, con Git tenemos repositorios de software. GitHub es un servicio para hacer hosting de repositorios de software que se administra con Git. Digamos que en GitHub mantienes una copia de tus repositorios en la nube, que además puedes hacer disponible para otros desarrolladores[2].

Posteriormente decidimos incluir también la construcción de la propia memoria en GitHub, ya que otras soluciones de documentación en la nube nos convencían menos de modo que cada uno de los integrantes del equipo pudiese avanzar en el desarrollo del software o de la documentación asociada.

Al adoptar git, abrazamos este sistema con todas sus virtudes y sus defectos, al principio si ya has trabajado con otros sistemas de control de versiones, debes desechar estos conocimientos, ya que tienen que ver muy poco con el funcionamiento que tiene git a la hora de llevar el control sobre los archivos versionados, debido a su concepción distribuida y a que cada integrante tiene una copia local del repositorio, y solamente es esto lo que comparte, el estado de los repositorios locales, pudiendo compartirlos con uno o varios de lo que se viene a llamar remotos, que no son más que otros repositorios de los que se tiene referencia.

Otros sistemas como podrían ser subversión, (utilizamos este ejemplo por ser bastante conocido) realizan copia de los archivos versionados cada vez que se cambia algo, sin embargo, el sistema de git, hace que sea más eficiente a una fotografía, donde se cambian los archivos nuevos o modificados pero sin embargo mantiene sin modificaciones los que no se han tocado, haciendo su uso bastante más eficiente[10].

Como estrategia de desarrollo hemos utilizado la denominada gitflow, que para desarrollo de productos de software sin fecha de fin determinada es una estrategia de ramas, fusiones, etc muy recomendable. Que consiste en dos líneas infinitas de desarrollo, "Desarrollo" donde se encuentran los siguientes cambios que pasarán a la siguiente versión que se publique. "Producción" que contiene el software que actualmente se encuentra disponible para el uso del público general.

A la vez hay ramas que se van creando según las necesidades del proyecto. Una rama por característica, que nace de la rama de "Desarrollo" cuando la característica se encuentra

implementada, se fusiona con la rama de la que partió.

También se crean ramas temporales de "hotfix.<sup>o</sup> parche para cada error que se deba solucionar en la versión de producción. Esta rama nace con el código existente en la rama "Producción" cuando el error está solucionado se fusiona con esta para subsanar el error y poder publicar una nueva versión, además también se fusiona con la rama "Desarrollo" para subsanar el error en todo el desarrollo del producto.

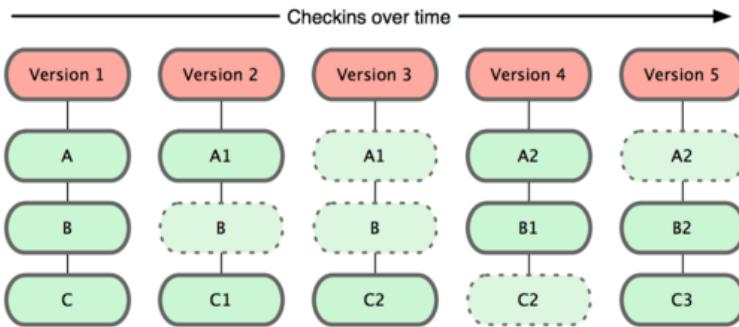


Figura 6.9: Funcionamiento de Git en el versionado de archivos

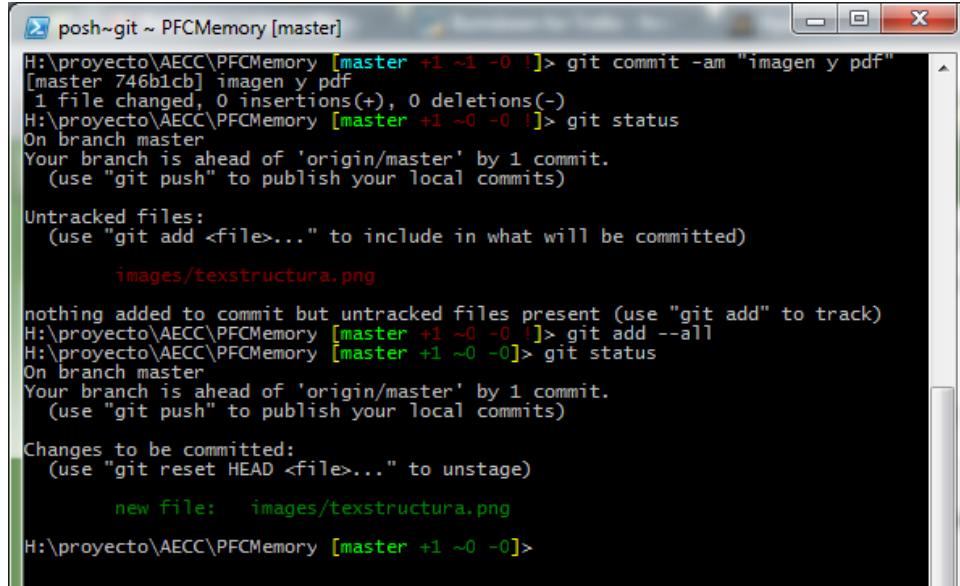
Dado que los dos utilizamos Windows y para tratar de manera más sencilla los archivos que íbamos cambiando y subiendo a la aplicación, nos creamos dos repositorios en GitHub, uno para la memoria y otro para la aplicación.

Sabemos que no es muy habitual que la documentación de un proyecto se realice a través de un sistema distribuido, sin embargo nos facilitó enormemente la tarea de llevar los cambios al día.

Para poder trabajar de una manera más óptima sobre la documentación con Git, partimos el documento inicial de la memoria en capítulos, cada uno se convierte en un .tex diferente, lo cual nos permite trabajar de manera independiente sobre cada capítulo y a la hora de subir los cambios minimizar el número de errores que se nos pudiesen dar al trabajar los miembros del equipo sobre esos mismos archivos.

Las divisiones que se hicieron en el documento, los sitúan en carpetas diferentes, con lo cual tenemos un .tex principal que es el que soporta toda la estructura de carpetas y archivos y es el que los configura (posee la mayor parte de los paquetes utilizados para tratar la estructura y aspecto del documento) y une, de manera que al compilar el archivo principal, se acaban compilando el resto de los archivos .tex que cuelgan de él.

Utilizamos PowerShell para Windows, que es una implementación de un interprete de comandos para Windows enfocado hacia Git, es decir que nos proporciona un entorno para que podamos escribir nuestras ordenes por linea de comandos y tratar de esa manera con nuestro repositorio Git



A screenshot of a Windows PowerShell window titled "posh~git ~ PFCMemory [master]". The window displays a series of Git commands and their outputs:

```
H:\proyecto\AECC\PFCMemory [master +1 ~1 -0 !]> git commit -am "imagen y pdf"
[master 746b1cb] imagen y pdf
 1 file changed, 0 insertions(+), 0 deletions(-)
H:\proyecto\AECC\PFCMemory [master +1 ~0 -0 !]> git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    images/texstructura.png

nothing added to commit but untracked files present (use "git add" to track)
H:\proyecto\AECC\PFCMemory [master +1 ~0 -0 !]> git add --all
H:\proyecto\AECC\PFCMemory [master +1 ~0 -0]> git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   images/texstructura.png

H:\proyecto\AECC\PFCMemory [master +1 ~0 -0]>
```

Figura 6.10: PowerShell para Windows

Windows PowerShell y en general Git, requieren un tiempo de aprendizaje, ya que la linea de comandos y los comandos en general no suelen ser demasiado amistosos con el usuario, y trabajar con ellos acarrea dificultades.

No siempre todo lo que subimos encaja a la perfección y se pueden dar conflictos entre los archivos, de manera que hay que solucionarlos para dejar el repositorio estable. Estos conflictos se pueden resolver de varias maneras posibles, puede ser a través de un merge, aceptando o eliminando los cambios locales, o modificando el resultado de lo que se va a subir, unificando el archivo que se va a quedar en el repositorio, se puede crear una nueva rama del proyecto (branch), o incluso se pueden apartar los cambios para seguir trabajando sobre la rama principal y hacer el commit de los mismos más tarde o desecharlos.

```

posh~git ~ PFCMemory [master]
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Ruhe\Documents\GitHub> cd ..
C:\Users\Ruhe\Documents> h:
H:> cd .\proyecto
H:\proyecto> cd .\AECC
H:\proyecto\AECC> cd .\PFCMemory
H:\proyecto\AECC\PFCMemory [master +1 ~1 -0 !]> git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   pfc.pdf

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        images/texstructura.png

no changes added to commit (use "git add" and/or "git commit -a")
H:\proyecto\AECC\PFCMemory [master +1 ~1 -0 !]> git
usage: git [--version] [--help] [-c <path>] [-c name=value]
         [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
         [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
         [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
         <command> [<args>]

The most commonly used git commands are:
add           Add file contents to the index
bisect        Find by binary search the change that introduced a bug
branch       List, create, or delete branches
checkout     Checkout a branch or paths to the working tree
clone        Clone a repository into a new directory
commit       Record changes to the repository
diff          Show changes between commits, commit and working tree, etc
fetch        Download objects and refs from another repository
grep          Print lines matching a pattern
init          Create an empty Git repository or reinitialize an existing one
log           Show commit logs
merge        Join two or more development histories together
mv            Move or rename a file, a directory, or a symlink
pull          Fetch from and integrate with another repository or a local branch
push          Update remote refs along with associated objects
rebase        Forward-port local commits to the updated upstream head
reset        Reset current HEAD to the specified state
rm             Remove files from the working tree and from the index
show          Show various types of objects
status        Show the working tree status
tag           Create, list, delete or verify a tag object signed with GPG

'git help -a' and 'git help -q' lists available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
H:\proyecto\AECC\PFCMemory [master +1 ~1 -0 !]>

```

Figura 6.11: PowerShell con los comandos más habituales a la hora de operar.

Como complemento a PowerShell, utilizamos como apoyo gráfico GitHub desktop, una aplicación gratuita que nos proporciona GitHub, y nos permite observar desde el propio Windows el estado del repositorio, los cambios que se han realizado, quien los ha realizado, etc.

Por contra, aunque sea muy visual y nos permita de un vistazo ver cuales son los archivos que se han modificado en el último commit, no tiene demasiadas opciones a la hora de interactuar con el repositorio, permite hacer un commit, desechar los cambios, hacer un revert (volver hacia atrás) y poco más.

Es útil porque permite tener varios repositorios asociados y pasar de uno a otro es cosa de un click.

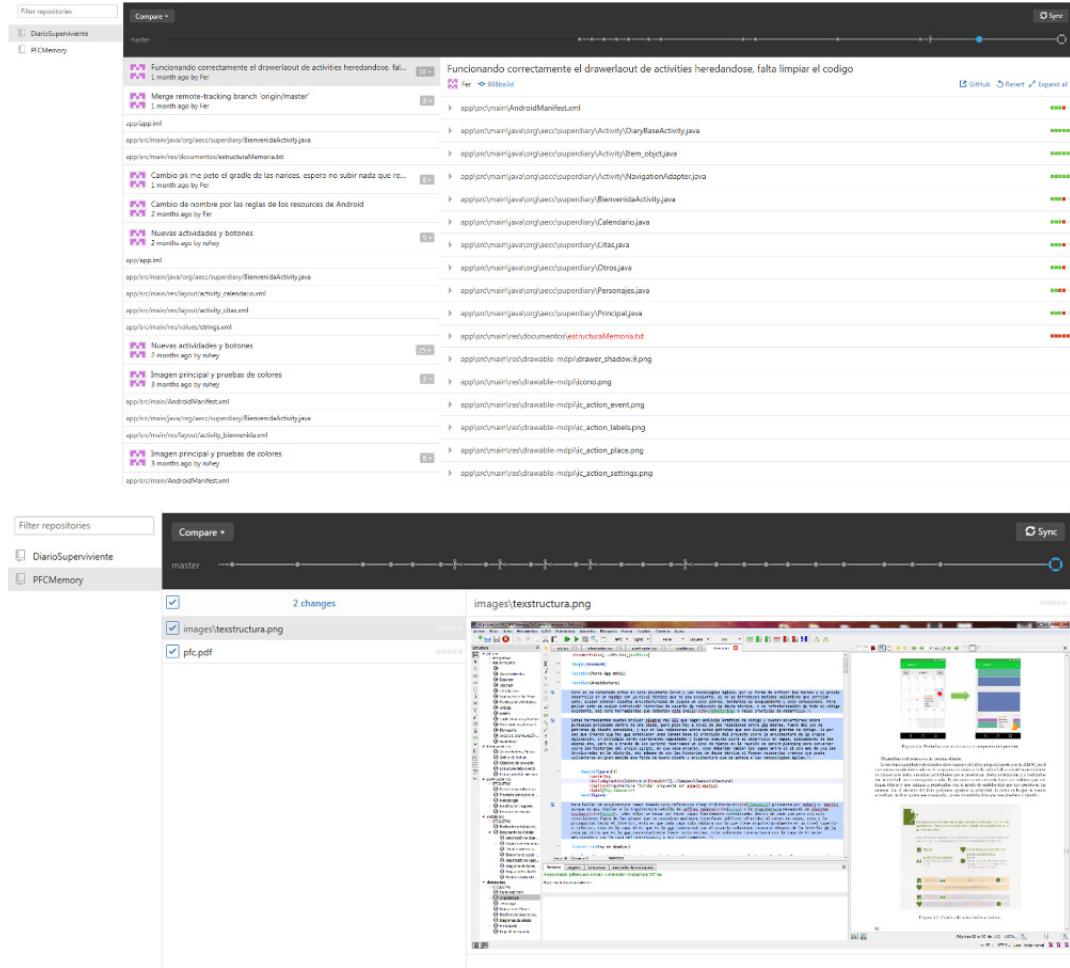


Figura 6.12: GitHub desktop, repositorios de la aplicación y de la memoria

Por ultimo vamos a hablar de las ventajas que nos reporta tener nuestra aplicación y memoria en GitHub,

- Podemos decidir si el código va a ser público o privado.
- Tiene un visor de código bastante avanzado, donde consultar en un instante el contenido de un determinado fichero, con su resultado de sintaxis correspondiente para el lenguaje en el que esté escrito. Además de poder ver las diferentes versiones del mismo que se han subido, e incluso copiar porciones de código, etc.
- Dispone de bastantes herramientas que nos hacen muy fácil el trabajo en equipo, posibilidad de descargar el proyecto como un zip, una wiki, poder meter comentarios dentro del código, Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero, visor de ramas,...
- Herramienta de revisión de código, es decir puedes pedir modificar el código de los demás... o pueden mejorar tu propio código.
- Y por último y no menos importante, es gratis (siempre que sea código open source).

Diario de un superviviente de Cancer — Edit

50 commits 1 branch 0 releases 2 contributors

Branch: master DiarioSuperviviente / +

Strings unificado  
Fer authored 3 days ago latest commit ec87f379f7

app	Strings unificado	3 days ago
gradle/wrapper	Initial commit	3 months ago
.gitignore	ignore iml and .idea files	10 days ago
DiarioSuperviviente.iml	Primera version NO FUNCIONAL de la DI y Clean Architecture	13 days ago
build.gradle	Primera version NO FUNCIONAL de la DI y Clean Architecture	13 days ago
gradle.properties	Entidades Contact y Comment en capas de dominio y data	11 days ago
gradlew	Initial commit	3 months ago
gradlew.bat	Initial commit	3 months ago
settings.gradle	Initial commit	3 months ago

Help people interested in this repository understand your project by adding a README. Add a README

Code Issues Pull requests Wiki Pulse Graphs Settings

HTTPS clone URL <https://github.com/> You can clone with HTTPS, SSH, or Subversion. Clone in Desktop Download ZIP

The screenshot shows a GitHub repository page for 'DiarioSuperviviente'. At the top, it displays '50 commits', '1 branch', '0 releases', and '2 contributors'. The repository name 'DiarioSuperviviente' is followed by a dropdown menu set to 'master' and a '+' button. Below this is a table of commits, each with a file icon, the file name, a brief description, and the time since commit. A note at the bottom encourages adding a README file. On the right side, there's a sidebar with links for 'Code', 'Issues' (0), 'Pull requests' (1), 'Wiki', 'Pulse', 'Graphs', 'Settings', and download options via HTTPS, SSH, or Subversion.

Figura 6.13: Repositorio de GitHub desde la propia página web

## 6.4. Pruebas

Hoy en día no se concibe un desarrollo de software sin disponer de pruebas o tester para poder asegurar la calidad del software y que este realiza las tareas para las que fue concebido correctamente y sin efectos laterales. Todo esto debido a que la complejidad del software desarrollado hoy día es de un orden de magnitud muy grande comparada con la de hace simplemente unos años.

Es por ello que hoy en día la técnica de desarrollo que más adeptos consigue es el TDD (por sus siglas en inglés de Test Driven Development) o Desarrollo Dirigido por Pruebas, en el cual ante una funcionalidad el desarrollador lo primero que ha de hacer es pensar en un test unitario y automático que falle. Después modificar el código todo lo rápido que pueda para hacer que dicho test automático pase a ser aprobado por la aplicación y por último refactorizar el software para mejorar el código antes escrito, eliminar duplicación, etc para volver al punto anterior en un proceso iterativo que acumulará test unitarios en estado de aprobación.

### 6.4.1. Tipos de pruebas

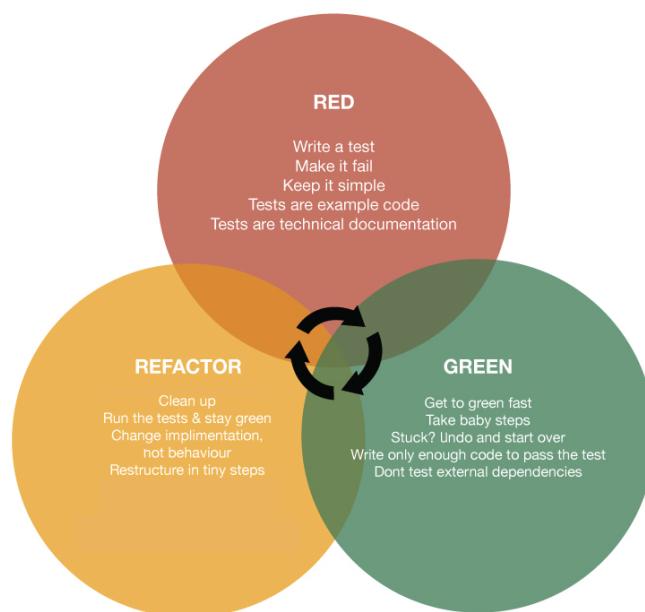


Figura 6.14: Ciclo de TDD con breves tips de cada una

De esta manera podemos obtener una confianza en que nuestro código sigue manteniendo la corrección, mientras vamos añadiendo funcionalidades a nuestra aplicación.

A tener en cuenta sobre esta técnica son los siguientes puntos:

- El test debe ser lo bastante pequeño para ser un test unitario pero a la vez añadir una funcionalidad que testar suficiente como para tener entidad propia.
- Deben ejecutarse de manera rápida para poder ejecutar todos los test unitarios cada vez que se añade uno a la suite de casos de prueba.
- No genera evidencia de corrección en la aplicación por sí mismo.
- Probar de forma independiente la funcionalidad para la que fue diseñado. Casos de test interdependientes o que dependan del orden de ejecución de los mismos a menudo evidencian errores de diseño en la aplicación.

El proceso de desarrollo seguido llevando a cabo TDD da como resultado un software testeable, desacoplado, y generalmente con las dependencias muy acotadas, aunque por contra como se ha mencionado en otras partes de este documento, arquitecturalmente suele desembocar en diseños poco cohesionados, ya que los test unitarios no suelen implicar refactorizar y mejorar el código de grandes bloques de clases, sino más bien de colaboradores directos.

#### 6.4.2. ATDD y TDD

Como hemos hablado TDD es básicamente un proceso de desarrollo de software, de él se deriva una técnica, esta sí de pruebas propiamente dicha denominada ATDD, o Desarrollo Dirigido por Test de Aceptación. Esta técnica nace de la necesidad de obtener una evidencia de pruebas ante el software construido. Es el paso más corto desde el proceso de TDD. Se basa en que el cliente o product owner y el equipo definen en base a una historia de usuario, un test que dará como aceptado el software que ejecute dicho test de forma satisfactoria. Siguiendo la filosofía de TDD se basa en escribir un test junto al cliente a partir de la historia de usuario. Suele usarse el perfil del equipo más experto en pruebas para ayudar a diseñar el test junto al product owner. Después junto al equipo se piensa en como expandir el test, para finalmente pensar por parte de cada uno de los desarrolladores en como abordar partes de este Test de Aceptación, en test unitarios que abordar mediante el proceso de TDD. Cabe mencionar que este test se expresa mayormente en el lenguaje del equipo de desarrollo, por eso es necesario que el test de aceptación inicial se escriba en conjunto entre el product owner y alguien del equipo que domine su lenguaje. Aunque en un principio TDD necesita que los test sean automatizados, el proceso de ATDD no tiene por qué, aunque que este automatizado aumenta la eficiencia y utilidad de estos test así como de los test de regresión efectuados cada ronda de testing.

#### 6.4.3. BDD

BDD es otro proceso de testeo que se denomina así por sus siglas en inglés Behaviour Driven Development, o desarrollo dirigido por comportamiento, es un proceso similar al de ATDD pero con la fundamental diferencia de que el lenguaje empleado en los test es el del dominio de la aplicación, y por lo tanto mucho mas cercano al product owner. Es por eso mucho más natural para él el tener que pensar en escribir un test que valide el comportamiento de una historia de usuario, aunque después este test sea más complicado de trasladar al TDD. El ciclo de BDD sería el mismo que el del TDD solamente que en cada iteración sobre un test de comportamiento implicaría varios ciclos de iteración sobre test de funcionalidad unitaria. Es deseable que estos test también estén automatizados y se lancen en el menor tiempo posible antes de cada compilación del código de la aplicación. Con estos ciclos de iteración a más alto nivel se consigue testar el comportamiento de la aplicación en algo evaluable y valioso para el product owner y forzar el refactorizado de código a más alto nivel para mejorar el diseño arquitectural de grandes bloques en principio menos interrelacionados que los que llevan a

cabo los N test unitarios que subyacen al test de comportamiento.

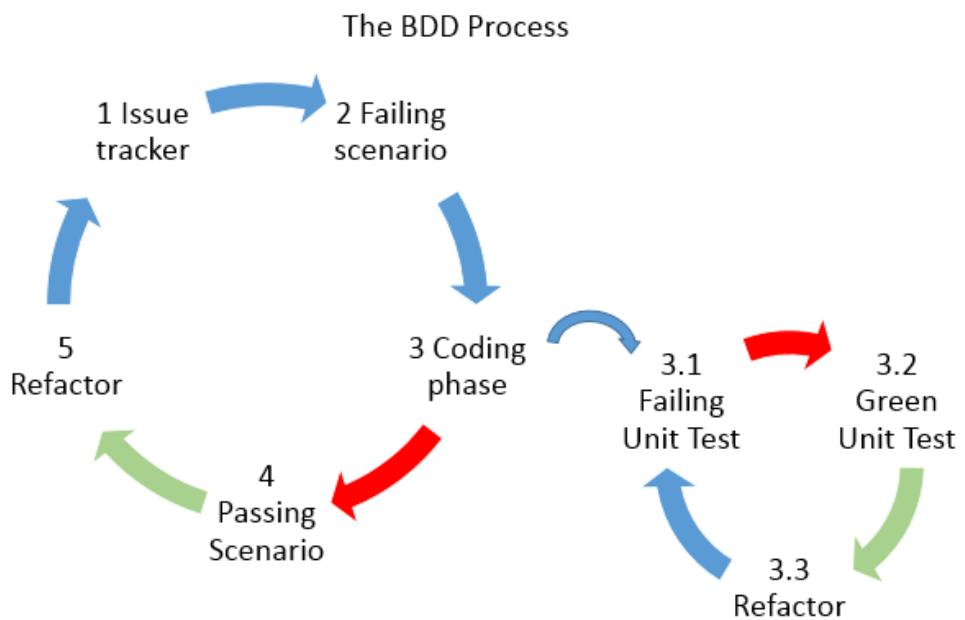


Figura 6.15: Ciclos integrados de TDD y BDD

Esta cercanía al lenguaje de dominio de la aplicación hace que hoy día estos sean unos test muy apreciados por el product owner y que al equipo de desarrollo le obligan a testar no solo unitariamente el software de la aplicación, sino el comportamiento de los casos de uso descritos en la misma.

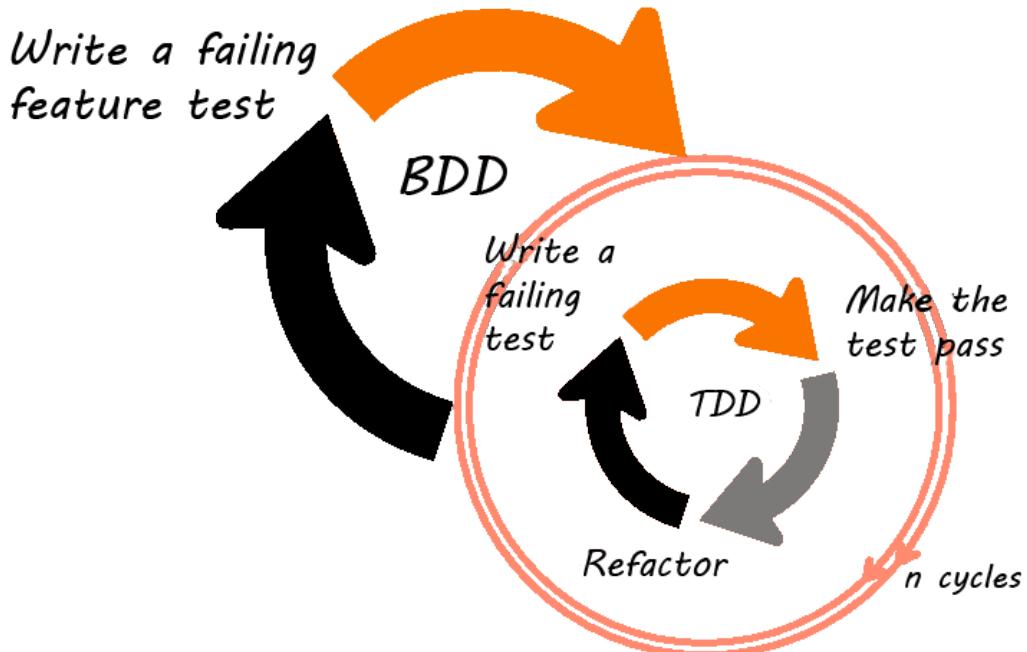


Figura 6.16: Otra imagen de los ciclos integrados de TDD y BDD

#### 6.4.4. Pruebas en el dispositivo

Por ultimo existe un bloque o tipo de test que son los que se realizan en el dispositivo. Estos se encargan sobre todo de testar la UI de la aplicación y que esta responde al comportamiento prefijado en los casos de uso y expresado en los test de comportamiento. Debido a las dependencias circulares a día de hoy no puede testarse en el mismo test el código del comportamiento de la app y el del SDK de Android. Por eso estos test de UI en dispositivo físico se separan del resto de test de la aplicación.

Debido a la gran variedad de dispositivos Android, por su abstracción sobre el hardware, es importante contar con la mayor base de terminales de diferentes características posibles. Además de esto es preciso poder contar con un sistema donde poder categorizar los usuarios de pruebas, así como recoger los ocasionales reportes que puedan enviarnos como feedback.

Como veremos en el siguiente punto esto puede lograrse entre otras maneras a través de la funcionalidad de versiones alpha y beta que la propia consola de desarrollador de la tienda Google Play Store nos ofrece. Con estas funcionalidades logramos disponer de una base de usuarios de test que muy probablemente ya estén familiarizados con la plataforma y no necesiten recibir más formación sobre ella.

## 6.5. Puesta en producción

El fin último de esta aplicación es que el público general se la descargue y use para mejorar su experiencia frente al cáncer. Por ello la misma debe estar en Play Store de Google, que es la tienda de aplicaciones más famosa y conocida del sistema operativo Android. Esta tienda pertenece a Google y para poder publicar en ella se debe hacer un pago único de 25 dólares. Tras esto se deben configurar las típicas pantallas de datos relativos al desarrollador email, web, dirección física, etc. Es recomendable llenar estos datos pues los ofrece confianza a los usuarios, además de proveer a los mismos de toda una gama de formas de contactar con nosotros. Eso sí, si se proporciona un método de contacto ese método debe ser atendido. No hacerlo sería una publicidad y una imagen de marca horrorosa.

En la propia consola podemos crear la página para la o las aplicaciones, que será la que se vea en la web o al acceder a la tienda mediante la app. En ella se introducen datos sobre la propia app, así como vídeos, imágenes y demás media e información sobre la misma, de manera que el usuario pueda hacerse una idea bastante clara de lo que va a obtener al instalar la misma. Ademas estas páginas permiten realizar algo de posicionamiento de cara a ser encontrada más fácilmente, aunque para esto es más importante una buena valoración por cuantos más usuarios mejor.

Este sistema también permite la interacción con los usuarios. Mediante el sistema de valoración de apps podemos responder a los usuarios, si es que estos plantean errores o bugs.



Figura 6.17: Consola de Play Store, mostrando gestión de APK y versiones beta y alpha

En la sección de APK, el binario a distribuir, podemos tener varios binarios en espera de poner en producción, así como activar la característica de utilizar versiones alpha y beta.

# CAPÍTULO 7

---

## Conclusiones y trabajo futuro

---

En este capítulo se presentan las conclusiones obtenidas de la realización de este proyecto, las dificultades encontradas, la consecución de objetivos marcados en un principio, los conocimientos adquiridos y las propuestas para trabajos futuros.

### 7.1. Conclusiones

Después de la realización de este proyecto vemos que se han conseguido los objetivos fijados en un principio.

En este proyecto hemos desarrollado una aplicación Android coherente con la iniciativa de la AECC 'DIARIO DE SALUD PARA SUPERVIVIENTES DE CÁNCER' y que hace más fácil llevar el control de muchas de las actividades relacionadas con la supervisión del tratamiento una vez 'superada' la enfermedad.

La aplicación posibilita el control de las citas médicas, rutinas diarias del paciente, control de la medicación, registro de síntomas y personas relacionadas con el paciente así como la gestión de pruebas médicas y su posterior uso informativo por parte del usuario.

No obstante, defendemos la ampliación de su uso durante el periodo en el que dura la enfermedad, ya que muchas de sus características son compatibles con el control médico y personal del mismo paciente y beneficiosas para el mismo.

La aplicación Android 'Diario de un superviviente' es pionera en el control total del tratamiento de un paciente de cáncer, con lo cual, no existe nada parecido en el mercado Android, y esto nos permite abrir una brecha e iniciar el camino en el desarrollo de aplicaciones destinadas a mejorar y sobrellevar el tratamiento de una de las principales causas de muerte en el mundo actual.

El diseño del proyecto se centró en el objetivo principal que era proporcionar una base de control fiable de las citas médicas de un paciente, facilitando su preparación y recopilación de información por parte del paciente para aprovechar mejor la visita al especialista y hacer más aprovechable la misma, dicho en otras palabras 'Para que no se escape ningún detalle'.

Al usuario además se le anima a realizar actividades diarias rutinarias, pudiendo anotar el grado de satisfacción que es tas le provocan.

Una vez que se consiguió la realización del principal objetivo, se vio necesario demostrar la utilidad de la aplicación en un entorno 'real' con pruebas de un uso diario intensivo por parte de un paciente para así recopilar información acerca de su facilidad de uso y detectar posibles carencias en la misma, esto propició la aplicación de mejoras y el control de errores por nuestra parte.

Por otra parte, la inserción de medicamentos, pruebas, síntomas y personajes, ayudó a que la información fuese más completa y estuviese más integrada en la aplicación, permitiendo añadir funcionalidades a la misma, y abriendo un abanico de posibilidades a la hora de tener que ampliar funcionalidades.

Otra conclusión, esta vez algo más negativa, es la que nos lleva a pensar en el porqué de esta escasez de esfuerzos a la hora de proporcionar herramientas tan útiles que ayuden a mejorar la situación diaria de los pacientes, y que a buen seguro, asociaciones médicas y de enfermos recibirían con los brazos abiertos.

Por otra parte, y no siendo menos importante, se nos da la posibilidad de luchar contra la enfermedad desde otro frente, que es desde el punto de vista del paciente, que es el que sufre la devastación de los tratamientos y las penalidades asociadas al mismo y que dada la cercanía desde la que nos toca el tema nos llena de orgullo.

Como punto final de las conclusiones, indicamos que el desarrollo de la aplicación continuará, ya sea por nosotros o por otros profesionales, para dar continuidad a esta herramienta y mejorarla hasta donde sea posible, alargando la colaboración junto a la AECC el tiempo que sea posible.

## 7.2. Dificultades encontradas

Una de las dificultades encontradas, fue la de idear todo el diseño de la aplicación, ya que al no haber ejemplos existentes no teníamos ninguna referencia, con lo cual, y utilizando muchas de las directrices dadas por Google con Material design pudimos solventar no sin poco esfuerzo.

Otra de las dificultades fue la de realizar todo el desarrollo a través del entorno de desarrollo integrado (IDE) Android Studio. En principio pensamos en utilizar Eclipse junto al ADT bundle de Android, sin embargo y dados los errores recurrentes que llevaban más tiempo de solucionar que el tiempo que estábamos desarrollando nos hizo cambiar de idea.

Una dificultad más que encontramos, ha sido la de realizar la documentación de una manera profesional y con un control total por nuestra parte, para ello desechamos programas propietarios y nos decidimos por LateX[16] junto a TexStudio[17] para la realización de la memoria, que si bien tiene una curva de aprendizaje pronunciada, posteriormente nos ha facilitado bastante las cosas a la hora de incluir figuras y tablas y de dar forma a al documento final.

## 7.3. Consecución de Objetivos

El objetivo principal del proyecto desde el principio ha sido el de servir de apoyo y mejorar la iniciativa de la AECC 'Diario de salud para supervivientes de Cáncer', esto se ha conseguido prestando especial atención a las carencias de este programa, lo cuales pueden ser identificados como la rigidez que ofrece el sistema del libro, el dialogo nulo que tiene con el usuario y obviamente la limitación de espacio que supone un libro de unas 30 páginas respecto a la aplicación.

Las listas, las diferentes ordenaciones, el tratamiento individual y pormenorizado de pruebas, síntomas, notificaciones, el poder añadir archivos gráficos y los recursos que ofrece la aplicación hacen de esta una potente herramienta al servicio del paciente.

## 7.4. Conocimientos adquiridos

El haber realizado este proyecto entre dos personas ha sido interesante en el sentido de la cooperación y coordinación de un equipo pequeño para el desarrollo de un proyecto.

Actualmente, los equipos pequeños de 2-3 personas son tendencia a la hora de utilizar una metodología ágil y las decisiones respecto al diseño, desarrollo, funcionalidades, son más rápidas y se llevan a cabo antes.

Hemos aprendido a diseñar una aplicación desde cero, a manejar tiempos y fechas límite, a 'teletrabajar'.

Así mismo hemos aprendido Android, sql, a documentar a través de LateX, a redactar una memoria, así como bastante Photoshop y versionado a través de Git,

## 7.5. Trabajo futuro

Como posibles mejoras futuras para la aplicación se nos han planteado varias funcionalidades que ya sean por tiempo o por objetivos no se han incluido en esta versión:

Búsquedas dentro de los listados e incluso búsqueda general dentro de la aplicación que facilite encontrar cualquiera de los tipos persistentes de manera rápida una vez su número haya aumentado de manera considerable.

Estadísticas personalizadas y generales, para determinar comportamientos, hábitos,...

Posibilidad de exportar las citas y rutinas a diferentes aplicaciones de sincronización de tareas, tales como facebook, Google, y su posibilidad de recibir correos electrónicos y notificaciones como el resultado de sincronizar con servicios externos para ser visualizadas en otros entornos y por otras personas, haciendo así posible su compartición o utilización por parte de otros servicios.

Parte servidora que permita un dialogo entre la asociación y el usuario más directo, descubriendo un sinfín de posibilidades a nivel de ayuda, cuidados, respuestas y servicios

personalizados.

Posibilidad de interactuar de manera directa con un agente de la asociación a través de la aplicación, asignación de un agente a modo de 'ángel de la guarda' .

Poder variar las notificaciones para que se hagan de manera repetitiva, por ejemplo repetir de manera semanal los jueves a las 8 una actividad, es decir generar una rutina más real.

Poder exportar e importar de Google maps, conectar la ubicación con Google maps para poder ir a la cita medica, ya que estas a menudo no son en la misma ciudad.

Llevar el stock de lo que lleva consumido del medicamento el paciente y generar alertas para cuando se le acabe, duplicar tratamiento, para ahorrar trabajo, opciones de dosis.

Llevarnos ese contacto a la agenda o traernosle de ella (PERSONAJES).

Poder utilizar documentos tipo PDF para no depender de la fotografía solo a la hora de adjuntar pruebas médicas.

Posibilidad de dar feedback.

En la sección de recursos, poder editar los canales RSS, chat entre supervivientes, personalizar la canción utilizada para la meditación.

Poder editar los teléfonos que se ofrecen de interés en la sección, para hacerlo más personalizable.

En los ajustes de la aplicación poder cambiar el tono de aviso, individualmente.

---

## Contenido del CD-ROM

---

### 7.6. Estructura y descripción de los contenidos de CDROM

Según la normativa de entrega de documentación de proyectos, los contenidos alojados en el CDROM que acompaña a la memoria de proyecto 'APLICACIÓN COMPLEMENTARIA A LA INICIATIVA DE LA AECC DIARIO DE UN SUPERVIVIENTE' son los siguientes;

- La versión en PDF del documento impreso completo bajo el nombre 'memoria.pdf'.
- Carpeta con el código fuente de la aplicación llamada 'CodigoFuente'.
- La versión instalable de la aplicación con el nombre 'diario.apk'.
- La versión electrónica del manual de uso dentro de la carpeta 'ManualUsuario'
- Documentación adicional relevante dentro de la carpeta 'OtraDocumentacion'

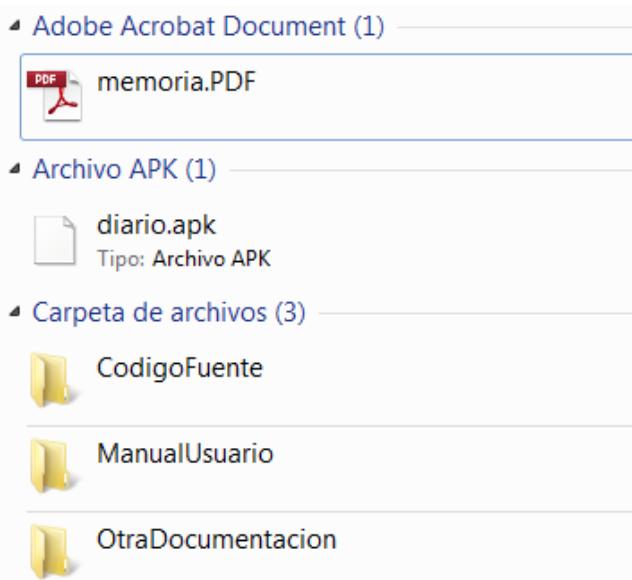


Figura 7.1: Estructura del contenido del CDROM



# CAPÍTULO 8

---

## Bibliografía

---



---

## Bibliografía

---

- [1] X. Albadalejo. Cómo gestionar proyectos con scrum, septiembre 2015. URL <http://www.proyectosagiles.org/>.
- [2] I. Alcázar. Introducción a git y github, agosto 2015. URL <http://www.desarrolloweb.com/articulos/introduccion-git-github.html>.
- [3] V. autores. Asociación española contra el cáncer, mayo 2015. URL <https://www.aecc.es/>.
- [4] V. Autores. Atom editor, agosto 2015. URL <https://atom.io/>.
- [5] V. autores. Conceptos de front-end y back-end, julio 2015. URL <http://www.alegsa.com.ar/Dic/back-end.php>.
- [6] V. autores. Fep niños con cáncer, agosto 2015. URL <http://http://cancerinfantil.org/>.
- [7] V. autores. Code retreat, julio 2015. URL <http://coderetreat.org/>.
- [8] V. autores. Conceptos de diseño del software, julio 2015. URL <http://sistemasumma.com/2011/03/15/conceptos-del-diseno/>.
- [9] V. autores. Historias de supervivientes de cáncer, agosto 2015. URL <http://www.curadosdecancer.com/>.
- [10] V. autores. Git reference, agosto 2015. URL <https://git-scm.com/book/es/v1/Empezando-Fundamentos-de-Git>.
- [11] V. autores. Grupo español de pacientes con cáncer, agosto 2015. URL <http://todosomosupervivientes.com/>.
- [12] V. autores. Referencia y tutorial para el uso de github, julio 2015. URL <https://help.github.com/>.
- [13] V. autores. Katas de programación, mayo 2015. URL <http://katayunos.com/>.
- [14] V. autores. Manifesto for software craftsmanship, agosto 2015. URL <http://manifesto.softwarecraftsmanship.org/#/es>.
- [15] V. autores. Nota descriptiva de prensa sobre el cáncer, agosto 2015. URL <http://www.who.int/mediacentre/factsheets/fs297/es/>.
- [16] V. autores. Editor de documentos latex, julio 2015. URL <https://es.sharelatex.com/>.

- [17] V. autores. Entorno de creacion de documentos latex, junio 2015. URL <http://www.texstudio.org/>.
- [18] cancer.org. Expliación general sobre la metástasis, agosto 2015. URL <http://www.cancer.org/espanol/index>.
- [19] A. Cockburn. Hexagonal architecture, julio 2015. URL <http://alistair.cockburn.us/Hexagonal+architecture>.
- [20] O. M. de la Salud. Cáncer, nota descriptiva 297, septiembre 2015. URL <http://www.who.int/mediacentre/factsheets/fs297/es/>.
- [21] W. en español. Metodologías de desarrollo de software, septiembre 2015. URL [https://es.wikipedia.org/wiki/Metodolog%C3%ADa\\_de\\_desarrollo\\_de\\_software/](https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software).
- [22] R. T. Fernando Santa Olaya. Aplicación diario de un superviviente, mayo 2015. URL <https://github.com/ruhey/DiarioSuperviviente>.
- [23] M. Fowler. *Refactoring: Improving the design of existing code*. Object Technology International, 1999.
- [24] D. frustrated by the restrictions of the traditional LaTeX environment. Editor online de latex, septiembre 2015. URL <https://es.sharelatex.com/>.
- [25] GitHub. Github desktop, agosto 2015. URL <https://desktop.github.com/>.
- [26] Google. Navigation drawer, julio 2015. URL <https://www.google.com/design/spec/patterns/navigation-drawer.html>.
- [27] M. Goossens, F. Mittelbach, and A. Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.
- [28] G. Inc. Android studio, agosto 2015. URL <https://developer.android.com/sdk/index.html>.
- [29] G. Inc. Recursos para desarrolladores android, agosto 2015. URL <http://developer.android.com/index.html>.
- [30] N. C. Institute. *Consejos de alimentación: Antes, durante y después del tratamiento del cáncer*. NIH, USA, 2011.
- [31] R. C. M. e. a. Kent Beck. Agile manifesto, septiembre 2015. URL <http://agilemanifesto.org/>.
- [32] H. Kniberg. *Scrum y XP desde las trincheras*. C4Media, 2007.
- [33] F. d. C. d. l. U. C. Luis M. Molina. *Apuntes de Latex*.
- [34] R. C. Martin. *Código Limpio: Manual de estilo para el desarrollo ágil de software*. ANAYA, 2012.
- [35] R. C. Martin. Clean architecture, agosto 2015. URL <https://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html>.
- [36] R. C. Martin. S.o.l.i.d.: Principles of ood, agosto 2015. URL <http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOOD>.

- 
- [37] N. Montés. Uso mundial de sistemas operátivos móviles, agosto 2015. URL <https://blog.uchceu.es/informatica/ranking-of-operating-systems-and-trends-for-2015/>.
  - [38] M. L. Murphy. The busy coder's guide to android development, 2014.
  - [39] U. o. L. Nikos Drakos, Computer Based Learning Unit. *Manual de LateX*.
  - [40] Oracle. Java sdk, agosto 2015. URL <http://www.oracle.com/technetwork/es/java/javase/downloads/index.html>.
  - [41] J. Palermo. Onion architecture, julio 2015. URL <http://jeffreypalermo.com/blog/the-onion-architecture-part-1/>.
  - [42] F. S. O. Rubén Toquero. Memoria proyecto diario de un superviviente, mayo 2015. URL <https://github.com/fsantaolaya/PFCMemory>.
  - [43] M. B. P. Semblantes. Diferencias de género en el tipo de afrontamiento en pacientes oncológicos. PDF, 2014.
  - [44] X. Struga. 5 pasos de la ingeniería del software, junio 2015. URL <http://proyectosguerrilla.com/blog/2013/02/las-cinco-etapas-en-la-ingenieria-del-software/>.



---

## GLOSARIO

---

**AECC.** La Asociación Española Contra el Cáncer (aecc) es una entidad privada, benéfica y sin ánimo de lucro, declarada de interés público. Fue fundada y aprobada por Orden Ministerial en el año 1953. Los objetivos de la asociación han ido variando desde su fundación hasta el momento actual, intentando siempre dar una respuesta a las necesidades que se han ido planteando en el ámbito de los enfermos de cáncer y sus familias.

### Android

Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tablets; y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró. Como curiosidad: Tanto el nombre Android (androide en español) como Nexus hacen alusión a la novela de Philip K. Dick ¿Sueñan los androides con ovejas eléctricas?, que posteriormente fue adaptada al cine como Blade Runner. Tanto el libro como la película se centran en un grupo de androides llamados replicantes del modelo Nexus-6.

### Android Studio

Android Studio es un entorno de desarrollo integrado (IDE) para la plataforma Android. Fue anunciado por Ellie Powers el 16 de mayo de 2013. Android Studio está disponible para desarrolladores para probarlo gratuitamente. Basado en IntelliJ IDEA de JetBrains, está diseñado específicamente para desarrollar para Android. Esta disponible para descargar para Windows, Mac OS X y Linux.

**App.** En informática, una aplicación es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajos. Esto lo diferencia principalmente de otros tipos de programas, como los sistemas operativos (que hacen funcionar la computadora), los utilitarios (que realizan tareas de mantenimiento o de uso general), y los lenguajes de programación (para crear programas informáticos).

**Arquitectura del software.** Es el diseño de más alto nivel de la estructura de un sistema, la arquitectura del software es la estructura jerárquica a través de la cual se relacionan los módulos del software que reunidos por completo forman nuestra aplicación.

**Backend.** De forma general, back-end hace referencia al estado final de un proceso. Contrasta con front-end, que se refiere al estado inicial de un proceso. La idea general es que el front-end es responsable de recoger entradas de los usuarios, y ser procesadas de tal manera que cumplan las especificaciones para que el back-end pueda usarlas. La conexión entre front-end y el back-end es un tipo de interfaz.

**BBDD.** Una base de datos o banco de datos es un conjunto de datos pertenecientes

a un mismo contexto y almacenados sistemáticamente para su posterior uso. Una base de datos es un “almacén” que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente.

**BDD.** El Desarrollo Guiado por el Comportamiento o BDD es un proceso que amplia las ideas de TDD y las combina con otras ideas de diseño de software y análisis de negocio para proporcionar un proceso ( y una serie de herramientas ) a los desarrolladores, con la intención de mejorar el desarrollo del software, BDD se basa en TDD formalizando las mejores prácticas de TDD, clarificando cuáles son y haciendo énfasis en ellas.

**Cáncer.** El cáncer es el nombre común que recibe un conjunto de enfermedades relacionadas en las que se observa un proceso descontrolado en la división de las células del cuerpo. Puede comenzar de manera localizada y diseminarse a otros tejidos circundantes. En general conduce a la muerte del paciente si este no recibe tratamiento adecuado. Se conocen más de 200 tipos diferentes de cáncer. Los más comunes son: de piel, pulmón, mama y colorrectal.

### **Clean Architecture**

Arquitectura de software cuyo principal objetivo es la separación de requerimientos o 'concerns' en capas, de tal manera que se produzca el mínimo acoplamiento entre ellas.

**Clean Code** Se refiere a los métodos y recomendaciones que hacen que nuestro 'código' sea más limpio. Todas estas buenas prácticas sobre cómo escribir código limpio se deben ejecutar de forma constante para adquirir buenos hábitos y ser capaces de hacerlo de forma natural.

**Code Retreat.** Coderetreat es una práctica intensiva de un día de duración, centrándose en los fundamentos de desarrollo de software y diseño, proporcionando a los desarrolladores la oportunidad de tomar parte en la práctica enfocada lejos de las presiones del "Getting Things Done" haciendo del mismo un método práctico para la mejora de habilidades.

**Code Review.** Code review o revisión de código es el examen sistemático (a menudo conocido como peer review) del código fuente ya sea por nosotros mismos o por otros compañeros. Su objetivo es encontrar y corregir errores pasados por alto en la fase inicial de desarrollo, mejorando tanto la calidad global de software como las habilidades de los desarrolladores.

**Code Smell.** Es cualquier síntoma en el código fuente de un programa que posiblemente indica un problema más profundo. Los code smells usualmente no son un bug de programación (errores), no son técnicamente incorrectos y en realidad no impiden que el programa funcione correctamente. En cambio, indican deficiencias en el diseño que puede ralentizar el desarrollo o aumentan el riesgo de errores o fallos en el futuro.

**Daily meeting.** Es la reunión diaria de sincronización del equipo (Scrum daily meeting). El objetivo de esta reunión es facilitar la transferencia de información y la colaboración entre los miembros del equipo para aumentar su productividad, al poner de manifiesto puntos en que se pueden ayudar unos a otros.

**DAO.** Los Objetos de Acceso a Datos son un Patrón de Diseño y considerados una buena práctica. La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.

**Dead Line.** Es la fecha final que se utiliza como límite para una entrega o DEMO de un producto de cara al cliente, puede ser la de fin de SPRINT o la de fin de proyecto.

**Deuda técnica.** La deuda técnica es un eufemismo tecnológico que hace referencia a las consecuencias de un desarrollo apresurado de software o un despliegue descuidado de hardware, puede entenderse perfectamente como chapuza.

**Diario de salud para supervivientes de Cáncer.** Iniciativa de la Asociación española contra el cáncer con el propósito de ayudar a los supervivientes de cáncer a afrontar la nueva etapa de su enfermedad.

**DPI.** Es una unidad de medida para resoluciones de impresión, concretamente, el número de puntos individuales de tinta que una impresora o tóner puede producir en un espacio lineal de una pulgada. El numero habitual de dpi para la yema de un dedo es 48.

**Drawer menú.** Menú de navegación con desplazamiento lateral que forma parte de la interfaz de usuario y que ayuda a 'navegar' entre las diferentes actividades que forman la aplicación.

**Dropbox.** Se trata de una herramienta de sincronización de archivos a través de un disco duro o directorio virtual. Permite disponer de un directorio de archivos de forma remota y accesible desde cualquier ordenador. Es decir, crea una carpeta en nuestro ordenador y realiza una copia a través de Internet de todos los archivos que depositemos en ella. Se ocupa de mantener la copia de nuestros archivos siempre sincronizada.

**Git.** Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

**GitHub.** Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

**Google Play store.** Es una plataforma de distribución digital de aplicaciones móviles para los dispositivos con sistema operativo Android, así como una tienda en línea desarrollada y operada por Google. Esta plataforma permite a los usuarios navegar y descargar aplicaciones (desarrolladas mediante Android SDK), juegos, música, libros, revistas y películas.

**Historía de usuario.** Una historia de usuario es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos (acompañadas de las discusiones con los usuarios y las pruebas de validación). Cada historia de usuario debe ser limitada, ésta debería poderse escribir sobre una nota adhesiva pequeña. Dentro de la metodología XP las historias de usuario deben ser escritas por los clientes.

**Historía épica.** Una historia épica no es más que un nivel de agrupación por encima de las historias de usuario que permite clasificar las mismas por funcionalidades, módulos, subsistemas, etc... Las epicas se suelen referir a funcionalidades completas que deberán ser divididas en varias historias de usuario.

**IDE (Integrated Development Environment).** Un entorno de desarrollo integrado es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

**iOS.** iOS es un sistema operativo móvil de la multinacional Apple Inc, iOS se deriva de OS X, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo Tipo Unix. Originalmente desarrollado para el iPhone (iPhone OS), después se ha usado en dispositivos como el iPod touch y el iPad.

**Java.** Es un lenguaje de programación orientado a objetos y la primera plataforma informática creada por Sun Microsystems en 1995. Es la tecnología subyacente que permite el uso de programas punteros, como herramientas, juegos y aplicaciones de negocios. Tiene como principal característica ser un lenguaje independiente de la plataforma.

**Kata.** Las katas y dojos son unos ejercicios que se realizan para practicar, son problemas sencillos de los que se conoce la solución pero lo importante no es resolverlos sino aplicar las lecciones aprendidas y mejorar nuestras habilidades de programación que posteriormente usemos en los proyectos que trabajamos.

**Koan.** Koan es un problema que el maestro plantea al alumno para comprobar sus progresos.

**Material design.** Material design es una normativa de diseño enfocada en la visualización del sistema operativo Android, pero también en la web y en cualquier plataforma. Material se trata de un diseño más limpio, en el que predominan animaciones y transiciones de respuesta, el relleno y los efectos de profundidad tales como la iluminación y las sombras.

**Metástasis.** Cuando el cáncer se propaga desde la parte del cuerpo donde comenzó (sitio primario) a otras partes del cuerpo se le llama metástasis. La metástasis puede ocurrir cuando las células se desprenden de un tumor canceroso y se desplazan a otras áreas del cuerpo a través del torrente sanguíneo o los vasos linfáticos. (Los vasos linfáticos se parecen mucho a los vasos sanguíneos con la diferencia que transportan un líquido claro llamado linfa de regreso al corazón). Las células cancerosas que se trasladan a través de los vasos sanguíneos o linfáticos se pueden propagar a otros órganos o tejidos en partes distantes del cuerpo.[18]

**Mockup.** Un mockup, mock-up, o maqueta es un modelo a escala o tamaño real de un diseño o un dispositivo, utilizado para la demostración, evaluación del diseño, promoción, y para otros fines. Un mockup es un prototipo si proporciona al menos una parte de la funcionalidad de un sistema y permite pruebas del diseño. Los mockups son utilizados por los diseñadores principalmente para la adquisición de comentarios por parte de los usuarios. Los mock-ups abordan la idea capturada en la ingeniería popular: 'Usted puede arreglarlo ahora en el dibujo con una goma de borrar o más tarde en la obra con un martillo'.

**Movilidad.** Se entiende por movilidad en el ámbito de nuestro proyecto como cualquier aplicación diseñada para dispositivos móviles.

**MVC.** Modelo Vista Controlador, es un patrón de arquitectura de software que separa

los datos de la aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

**MVP.** Modelo Vista Presentador, es un patrón de arquitectura de software que se parece al MVC pero debido al gran acoplamiento existente en android y que el controlador y la vista puedan acabar fusionados en la Activity es más usado que el anterior (seguir).

**MySQL.** MySQL es un sistema de administración de bases de datos para bases de datos relacionales. No es más que una aplicación que permite gestionar archivos llamados de bases de datos. Utilizado para almacenar todos los datos de interés de la aplicación y datos referentes de las simulaciones realizadas por los usuarios.

**Neoplasia.** Formación anormal en alguna parte del cuerpo de un tejido nuevo de carácter tumoral, benigno o maligno.

**Pair Programming.** Se refiere a Programación en Pareja, esta requiere que dos programadores participen en un esfuerzo combinado de desarrollo en un sitio de trabajo, cada miembro realiza una acción que el otro no está haciendo actualmente: Mientras que uno codifica las pruebas de unidades el otro piensa en la clase que satisfará la prueba.

**Posología.** Parte de la farmacología que trata de las dosis en que deben administrarse los medicamentos.

**Product backlog.** Es la lista de objetivos/requisitos priorizada, representa la visión y expectativas del cliente (o PO) respecto a los objetivos y entregas del producto o proyecto.

**Product Owner.** El product owner o PO, representa al cliente (puede ser interno o externo a la organización) y asume sus responsabilidades de cara al equipo, representa a todas las personas interesadas en los resultados del proyecto (desde agentes de la AECC a usuarios finales de la aplicación).

**Requisito.** En la ingeniería de sistemas, un requisito es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. En la ingeniería clásica, los requisitos se utilizan como datos de entrada en la etapa de diseño del producto, en metodologías ágiles, se utilizan historias de usuario, sin embargo estas no son equivaklesntes.

**Retrospectiva.** Se trata de una reunión que se realiza al finalizar el sprint actual, después de la demo con el cliente y tiene el objetivo de mejorar de manera continua la productividad y la calidad del producto que está desarrollando. El equipo analiza cómo ha sido su manera de trabajar durante la iteración, por qué está consiguiendo o no los objetivos a que se comprometió al inicio de la iteración y por qué el incremento de producto que acaba de demostrar al cliente era lo que él esperaba o no

**ROI retorno de la inversión.** El retorno sobre la inversión (RSI o ROI) es una razón financiera que compara el beneficio o la utilidad obtenida en relación a la inversión realizada, es decir, 'representa una herramienta para analizar el rendimiento que la empresa tiene desde el punto de vista financiero'.

**SCRUM.** Scrum 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. Forma parte de las metodologías ágiles.

**SCRUM MASTER.** También llamado facilitador, Vela por que todos los participantes del proyecto sigan los valores y principios ágiles, las reglas y proceso de Scrum y guía la colaboración intraequipo y con el cliente de manera que las sinergias sean máximas.

**SGBD (Sistema Gestor de Bases de Datos).** En inglés Data Base Management System (DBMS); es un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos y el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

**Sintomatología.** Conjunto de síntomas que son característicos de una enfermedad determinada o que se presentan en un enfermo. También se refiere a la parte de la medicina que estudia los síntomas de las enfermedades.

**SPRINT.** Un sprint (o iteración) es la unidad básica de desarrollo de scrum, se limita a una duración específica. La duración se fija de antemano por cada sprint y normalmente es entre una semana y un mes, a dos semanas.

### SPRINT PLANNING

Con sprint planning, nos referimos a la planificación de las funcionalidades que se deben abordar para esa iteración o sprint, las tareas que entran en un sprint se fijan de acuerdo a la priorización de las mismas.

**StarUML.** Es una herramienta para el modelamiento de software basado en los estándares UML (Unified Modeling Language) y MDA (Model Driven Arquitectura). Permite definir elementos propios para los diagramas, que no necesariamente tienen que pertenecer al estándar de UML.

**SVN.** Es una herramienta de control de versiones open source basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es software libre bajo una licencia de tipo Apache/BSD.

**TDD.** Es una práctica de ingeniería de software que involucra otras dos prácticas: Escribir las pruebas primero (Test First Development) y Refactorización (Refactoring). Para escribir las pruebas generalmente se utilizan las pruebas unitarias (unit test en inglés). En primer lugar, se escribe una prueba y se verifica que las pruebas fallan. A continuación, se implementa el código que hace que la prueba pase satisfactoriamente y seguidamente se refactoriza el código escrito. El propósito del desarrollo guiado por pruebas es lograr un código limpio que funcione. La idea es que los requisitos sean traducidos a pruebas, de este modo, cuando las pruebas pasen se garantizará que el software cumple con los requisitos que se han establecido.

**Testing.** Las pruebas de software es una investigación llevada a cabo para facilitar a los interesados información sobre la calidad del producto o servicio que se está probando. Otra definición: investigación técnica de un producto bajo prueba con el fin de brindar información relativa a la calidad del software, a los diferentes actores involucrados en un proyecto.

**Time to market.** Es el tiempo que se tarda desde que un producto es concebido hasta que está en el mercado.

**UI.** La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar (aunque en el ámbito de la informática es preferible referirse a que suelen ser «amigables e intuitivos» pues es muy complejo y subjetivo decir que algo es «fácil»).

**USB (Universal Serial Bus).** Estándar de comunicaciones serie de alta velocidad.

**UX.** Experiencia de usuario (UX) es un término que mide el nivel de satisfacción total de usuarios cuando utilizan tu producto o sistema.

**XP (extreme programming).** Es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck. Es el más destacado de los procesos ágiles de desarrollo de software, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de la XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.