PHOTO DICOM

TRABAJO FINAL DE MASTER
MASTER EN INGENIERÍA INFORMÁTICA

www.uoc.edu

UNIVERSITAT OBERTA DE CATALUNYA

SERGIO GARCÍA PRADO
ENERO 2016

CONSULTOR: IGNASI LORENTE PUCHADES

I. INDICE GENERAL

II. LISTA DE FIGURAS

1. INTRODUCCIÓN

1.1 CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO

La imagen médica se lleva digitalizando y almacenando en sistemas PACS[1] desde hace más de dos décadas de forma generalizada en los países desarrollados. Para poner de acuerdo a fabricantes e instituciones sanitarias se desarrolló el protocolo DICOM[2] por la propia industria, que terminó de definirse en su versión 3.0 en 1993. Dicho protocolo posibilita la comunicación de imágenes generadoras de imagen, sistemas de almacenamiento PACS, clientes visores de imágenes y cualquier otro actor relacionado con la imagen médica.

El protocolo DICOM, lejos de ser óptimo en cuanto a su estructura, optimización de las transmisiones (es un protocolo "muy pesado" y poco ágil) y con importantes limitaciones, se adoptó como estándar de facto por la industria y especialmente por la industria radiológica.

A día de hoy prácticamente todos los Hospitales del mundo desarrollado disponen de uno o varios sistemas PACS donde almacenan las imágenes procedentes de las distintas modalidades diagnósticas a través del protocolo DICOM 3.0 en sistemas de almacenamiento seguros y replicados.

Así como dicho protocolo ha sido adquirido como estándar en el ámbito de la radiología, radioterapia, medicina nuclear y servicios afines, su adopción en otros ámbitos de la imagen médica no ha sido tan rápida ni tan generalizada.

Una vez solucionado el almacenaje y distribución de las imágenes de los servicios anteriormente citados, (y a su vez obtenido un retorno de rentabilidad al lograr hacer desaparecer la película radiográfica – de importante costo), los diferentes organismos sanitarios se están planteando reutilizar las infraestructuras ya existentes para dar una solución similar al almacenaje (seguro) y distribución de la imagen no radiológica de los servicios generadores de imagen médica que no han adoptado (aún) DICOM como base de funcionamiento, como por ejemplo la imagen procedente de oftalmología, dermatología, cirugía plástica, las gráficas procedentes de los electrocardiógrafos, digitalización de las muestras de anatomía patológica, etc.

[1] PACS: Picture Archiving and Communication System. Más información: https://en.wikipedia.org/wiki/Picture_archiving_and_communication_system

[2] DICOM: Digital Imaging and Communications in Medicine. Más información: https://en.wikipedia.org/wiki/DICOM

Parte de esos servicios utilizan máquinas específicas (como por ejemplo en anatomía patológica) que son las que deben implementar el protocolo DICOM para sumarse a los sistemas pre-existentes en los Hospitales. Otra serie de servicios generan imágenes médicas vía cámaras de fotos más o menos estándar y por tanto con formatos de imagen no médicos (jpeg y similar).

En la actualidad las imágenes obtenidas con cámaras fotográficas se están almacenando, de forma general, sin orden ni concierto en discos duros de computadoras locales a los servicios, sin ninguna seguridad (ante fallo por ejemplo de un disco duro), ni con posibilidad de distribuir dichas imágenes a otros servicios que pudiesen necesitar su consulta.

En el mejor de los casos, una vez descargadas las fotos desde la cámara a un PC, un usuario se dedica a la conversión de dichas imágenes a formato DICOM, casación de cada grupo de imágenes con el correspondiente paciente y finalmente envío al sistema PACS del Hospital, siendo éste un proceso lento y tedioso.

El presente proyecto pretende dar solución a los servicios que utilizan cámaras fotográficas para automatizar el proceso de conversión a formato DICOM y transmisión vía el protocolo del mismo nombre.

Para ello se sugiere adoptar cámaras fotográficas con sistema operativo Android que permitan la instalación de aplicaciones que faciliten las anteriormente citadas tareas.

Se pretende así mismo que la solución sea portable o como también se puede definir *Vendor Neutral*, para que el *pack* propuesto de cámara Android, app específica para Android y servicios de Servidor, pueda ser útil para cualquier centro sanitario que disponga de un sistema PACS, sea éste del proveedor que sea. Esto lo logramos gracias a la utilización del protocolo DICOM 3.0, estándar de facto de la industria de la imagen médica.

1.2 OBJETIVOS DEL TRABAJO

Los objetivos de este proyecto son los siguientes:

- Desarrollo de aplicación para la plataforma Android que permita las siguientes acciones:
  - Autenticación en la aplicación.
  - Recepción de listado de pacientes pendientes (*agenda del médico*).
  - Captura de imágenes (fotografías)
  - Asignación de las imágenes capturadas a un paciente concreto
  - Transmisión de dichas imágenes hacia un servidor externo (para su almacenaje final en un sistema PACS).
  - Consulta de la actividad realizada con el dispositivo.

Para que la *app* pueda realizar las acciones anteriores está previsto que se apoye en un componente externo al dispositivo Android que facilite ciertas tareas y transmisiones de datos. Por tanto el proyecto conlleva también:

- Desarrollo de un Servicio Web alojado en un servidor de la red que permite a su vez:
  - Recepción de peticiones desde la aplicación Android (cámara), interrogación al sistema del Hospital (PACS) vía protocolo DICOM (*DICOM Worklist*) y devolución de resultados (listado de pacientes) a la aplicación Android.
  - Recepción de imágenes desde la aplicación Android.
  - Conversión de las imágenes recibidas a formato DICOM.
  - Transmisión de las imágenes convertidas a formato DICOM a un sistema PACS de la red vía protocolo DICOM.

Finalmente, existe un objetivo secundario dentro del proyecto que es garantizar la seguridad y confidencialidad de las comunicaciones, ya que al tratarse de información de pacientes y datos médicos, ésta es información de alto nivel según marca la LOPD.

Como no se puede garantizar la existencia de un entorno de producción idóneo (por ejemplo funcionando sobre HTTPS), la seguridad será parte intrínseca de los elementos que componen la solución y al margen del entorno de producción final.

Tal como se ha indicado anteriormente la plataforma para las que se desarrolla la solución es cualquier dispositivo Android capaz de realizar fotografías. El dispositivo tipo ideal para su uso y comercialización son cámaras fotográficas con sistema Operativo Android, como por ejemplo la Samsung Galaxy Camera 2, Nikon Coolpix S810c, Samsung Galaxy Camera NX, Polaroid SC1630, Polaroid iM1836, Panasonic DMC-CM1 o cualquier otra cámara que pueda aparecer en el mercado en el futuro cuyo sistema operativo sea Android, lo cual no prohíbe a cualquier otro dispositivo Android (smartphones, tablets, etc.) que dispongan de cámara de fotos integrada de utilizar esta aplicación para los mismos fines (con mayor o menor calidad de imagen según los sensores fotográficos de los que dispongan).

Para la parte servidora se evitará tener dependencias con gestores de bases de datos, que dificultarían el despliegue y encarecerían el producto, y bastará con contar con un servidor Web. Para este proyecto por familiaridad con el entorno y el lenguaje se desarrollará el Servicio Web en lenguaje .NET y por tanto se requerirá para su funcionamiento de un servidor (o PC convencional) con sistema operativo Windows que cuente con un servidor Web IIS (*Internet Information Services*) de Microsoft.

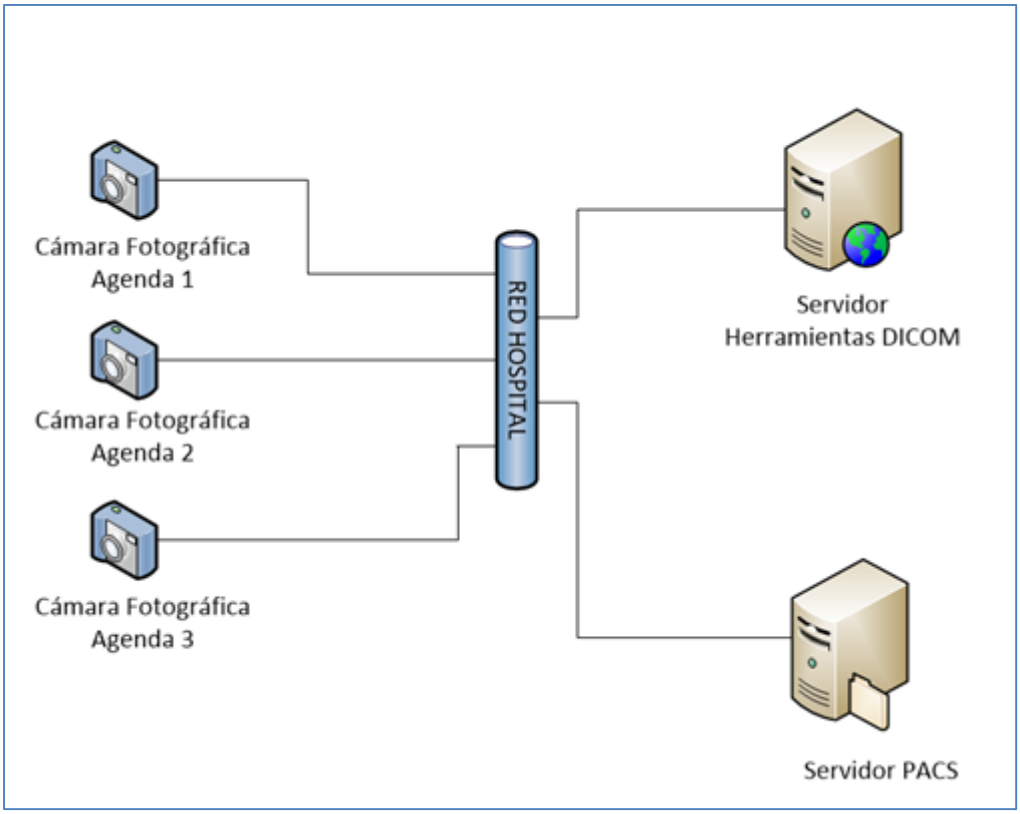


Ilustración 1 - Esquema de red

1.3 ENFOQUE Y MÉTODO SEGUIDO

Existen varias alternativas para el desarrollo de una solución que permita resolver el problema planteado en el punto 1.1 de este documento. Entre ellas podemos citar:

- Desarrollo de solución basada exclusivamente en Web que permita cubrir los objetivos planteados: consulta de lista de trabajo y subida de fotos realizadas para su posterior conversión a DICOM y envío a PACS.
- Otra alternativa podría plantearse sobre sistemas de mensajería y, en concreto, sobre mensajería HL7[3] mediante encapsulación de las imágenes dentro de la propia mensajería, por ejemplo mediante mensajes CDA (*Clinical Document Architecture*, HL7 versión 3) en conjunción con algún tipo de conversor DICOM.
- La planteada en los puntos 1.1 y 1.2 de este documento.

Asimismo se podrían plantear otro tipo de repositorios para el almacenaje y distribución de las imágenes médicas adquiridas mediante cámara de fotos. El problema de este tipo de

[3] HL7: Health Level Seven. Más información: http://www.hl7.org

soluciones es que serían muy específicas y poco interoperables con otros actores de la red sanitaria y requerirían además de nuevo *Hardware* de almacenamiento, entre otros elementos.

Con la solución planteada en los anteriores puntos obtenemos:

- La reutilización de las infraestructuras ya existentes en el Hospital a distintos niveles:
  - A nivel Software: PACS, Distribuidores de imagen vía Web, Visores con calidad diagnóstica, etc.
  - A nivel físico: almacenamiento, sistemas de copia de seguridad, servidores, redes de comunicaciones, estaciones de trabajo diagnósticas, etc.
- Homogeneidad con el resto de imagen médica generada y distribuida por toda la red sanitaria.
- Se logra también incluir la imagen médica digitalizada con la solución planteada en los flujos de trabajo requeridos por la *Historia Clínica Electrónica* de los centros sanitarios.

Con el enfoque adoptado para solucionar el enfoque planteado obtenemos otra serie de ventajas adicionales:

- Reaprovechamiento de librerías DICOM y abstracción del protocolo y conversiones necesarias de la aplicación Android, lo que simplifica el desarrollo.
- Se plantea una aplicación Android sencilla y de fácil uso.
- Familiaridad, por los conocimientos previos del autor, de las tecnologías a utilizar.
- Nivel de recursos necesarios: bajo, un único servidor Windows para dar servicio a múltiples dispositivos Android.

Por tanto la estrategia elegida supone una reutilización parcial de los recursos existentes y de la tecnología ya existente (el autor trabaja en este ámbito y dispone de recursos y conocimientos previos).

Tecnologías Utilizadas

Como ya se ha citado el actual proyecto plantea el desarrollo de dos componentes distintos y con tecnología distinta:

- Aplicación Android, con las herramientas de desarrollo proporcionadas por el SDK de Android.
- Web Service, a desarrollar con tecnología Microsoft© y en particular con tecnología .NET así como con los estándares intrínsecos de los Servicios Web:
  - XML,
  - SOAP,
  - WSDL,

- UDDI, …

Para simplificar el desarrollo y reducir costos de implementación (y a su vez en una posterior venta de la solución), se ha tratado de evitar la dependencia de un gestor de base de datos, que supondría un importante costo de licenciamiento (según gestor de BD elegido) y dificultaría la implantación final en cada cliente.

Capa de seguridad

Partimos del hecho de que las redes sanitarias son redes privadas y de alta seguridad. No obstante este hecho no es suficiente para garantizar la confidencialidad de los datos manejados por la aplicación, catalogados por la LOPD como del más alto nivel.

Se podría plantear que las comunicaciones entre la aplicación Android y el Web Service .NET se realizasen mediante protocolo seguro (HTTPS), pero eso requeriría una inversión (económica) en los correspondientes certificados de servidor y añadiría una complejidad de implantación y puesta en marcha, que se trata de evitar.

La solución adoptada para garantizar un nivel suficiente de seguridad es realizar las comunicaciones de forma normal sobre HTTP pero con encriptación de la información. Por tanto los *mensajes* que se intercambiarán por la red serán los XMLs devueltos por el Web Service, pero con una encriptación a nivel de dato. Para esta labor se plantea inicialmente la utilización de un algoritmo *AES*[4] de cifrado simétrico, mediante la compartición de las mismas claves de seguridad y vector de inicialización entre ambas aplicaciones.

El posterior envío al PACS se podría realizar de forma segura mediante la utilización de DICOM-TLS (*Transport Layer Security*) pero esta parte queda fuera del ámbito del proyecto. Mencionar para el lector de este documento que el DICOM-TLS está escasamente implementado en los Centros Sanitarios y la comunicación con los PACS se suele realizar directamente (vía protocolo DICOM estándar).

1.4 PLANIFICACIÓN DEL TRABAJO

El autor de este proyecto, *Project Manager Professional* por el PMI[5], plantea el proyecto siguiendo las buenas prácticas auspiciadas por dicho instituto. Para ello fijamos un ciclo de vida del proyecto predictivo o *en cascada*, tal como se indica en el siguiente gráfico:

[4] Más información: https://es.wikipedia.org/wiki/Advanced_Encryption_Standard
[5] PMI: Project Management Institute. Más información: http://www.pmi.org/

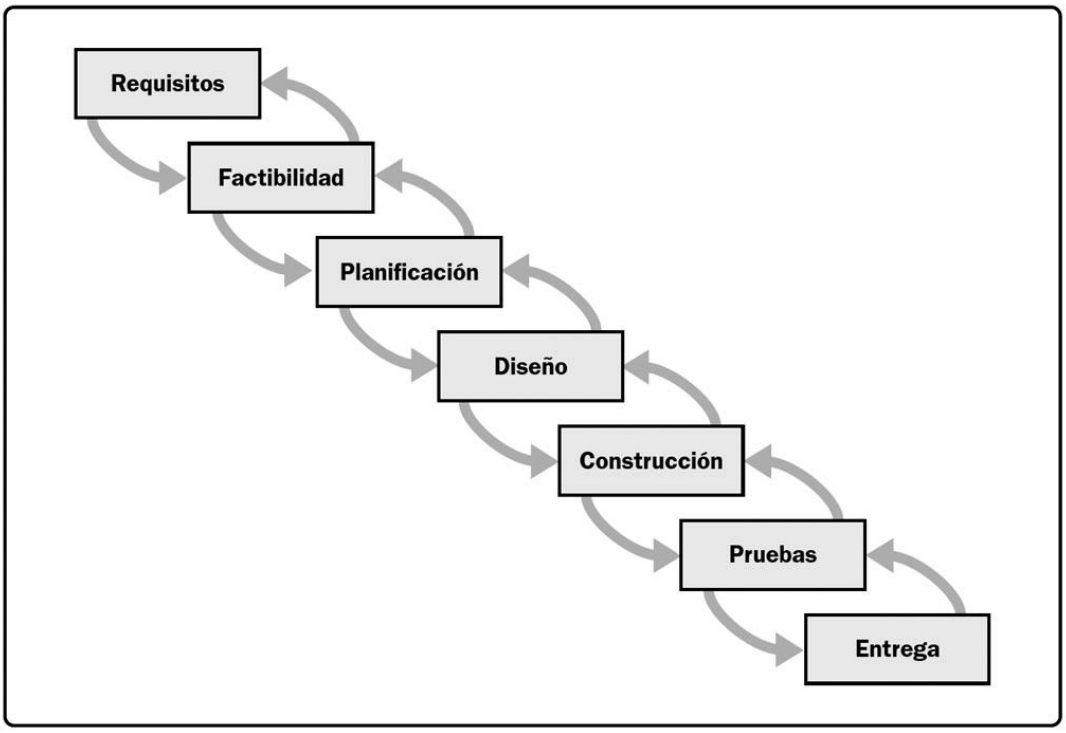


Ilustración 2 - Ciclo de vida predictivo

Esto lo podemos lograr gracias a que conocemos de antemano el alcance, el tiempo y en cierta medida el costo requerido para lograr el alcance. Cada tarea secuencial se corresponde con una o varias fases del TFM tal como se indica a continuación:

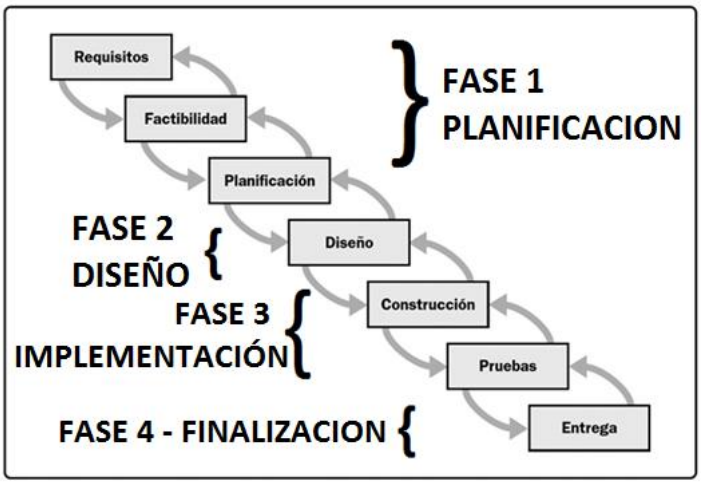


Ilustración 3 - Concordancia ciclo de vida / Fases del Proyecto

Por tanto las tareas se corresponden de la siguiente forma:

- FASE 1: Requisitos funcionales, Factibilidad y Planificación.
- FASE 2: Análisis, Diseño y Prototipado.
- FASE 3: Construcción (implementación) de la solución y Pruebas.

- FASE 4: Tareas de cierre de proyecto.

A continuación nos centraremos en el alcance del proyecto así como las funcionalidades buscadas en el desarrollo de la solución. La Factibilidad se ha analizado conjuntamente con el consultor estimándose por ambas partes la accesible realización del presente proyecto.

Durante el desarrollo de la FASE 2 se realiza un análisis más exhaustivo, así como un diseño inicial de la solución dando lugar al prototipado de la aplicación Android.

Durante la FASE 3 procedemos a la implementación del prototipo generado durante la fase anterior junto con los servicios anexos de servidor. Se aborda también una fase de pruebas para dar validez a la solución resultante.

Finalmente en la FASE 4, procedo a maquetar y empaquetar todo el resultado del proyecto y realizamos las labores de finalización del mismo.

1.4.1. DESGLOSE DE ACTIVIDADES

FASE 1. PLANIFICACION

Durante esta fase realizo las siguientes actividades:

- Se localiza la necesidad a cubrir y plantea las distintas opciones que cubran dicha necesidad (requisitos funcionales de alto nivel).
- Se establece la factibilidad o no del proyecto planteado junto con el Consultor.
- Se realiza la planificación.

FASE 2. DISEÑO

Durante esta fase realizo las siguientes tareas:

- Recogida de requisitos (Análisis de Requisitos).
- Análisis funcional de la aplicación.
- Diseño Técnico de la solución que engloba:
    - Diseño Arquitectónico
    - Diseño de Procesos.
- Prototipado inicial de las aplicaciones (*app* y *Web Service*).

FASE 3. IMPLEMENTACION

En esta fase abordo las siguientes actividades:

- Implementación de la solución. Que conlleva, prácticamente en paralelo:
  - Desarrollo de la aplicación Android
  - Desarrollo del Servicio Web.
- Realización de pruebas. Que a su vez se compone de:
  - Pruebas unitarias.
  - Pruebas de integración.

FASE 4. ENTREGA FINAL

En esta última fase desarrollo las siguientes tareas:

- Maquetación final de esta memoria.
- Realización del vídeo de presentación de la solución.
- Realización del material de apoyo correspondiente (diapositivas).
- Empaquetado final y tareas de finalización del proyecto.

1.4.2. CALENDARIO TEMPORAL

Estas actividades que acabo de exponer tienen el siguiente calendario temporal inicial, teniendo en cuenta el siguiente calendario laboral:

- Trabajo de Lunes a Viernes 18.00 a 22.00 Horas (4 horas diarias).
- Trabajo los Sábados de 18.00 a 20.00 Horas (2 Horas).
- No se trabaja los Domingos.
- No se contemplan excepcionalidades por ser festivo.

Con estas premisas tenemos el siguiente calendario temporal:

Nombre de tarea	Duración
FASE 1 - PEC 1	**36 horas**
Requisitos Funcionales (Alto Nivel)	6 horas
Factibilidad del Proyecto	2 horas
Planificación del proyecto	12 horas
Realización PEC1	8 horas
FASE 2 - PEC 2	**86 horas**
Análisis de Requisitos (Detalle)	18 horas
Análisis Funcional	14 horas
Diseño Técnico de la Solución	**36 horas**
Diseño Arquitectónico	16 horas
Diseño de Procesos	16 horas
Prototipado	10 horas
Realización PEC2	6 horas
FASE 3 - PEC 3	**150 horas**
Implementación (Construcción)	**112 horas**
Desarrollo de Servicio WEB	50 horas
Desarrollo de Aplicación Android	60 horas
Pruebas	**24 horas**
Pruebas Unitarias	12 horas
Pruebas de Integración	10 horas
Realización PEC 3	8 horas
FASE 4 - ENTREGA FINAL	**64 horas**
Maquetación Final Memoria	12 horas
Realización Video Presentación	8 horas
Realización Material Apoyo	6 horas
Empaquetado Entrega Final	6 horas

Ilustración 4 - Calendario Temporal del Proyecto

1.4.3. DIAGRAMA DE GANTT

Con la duración de las tareas fijadas (ver anterior punto) obtenemos el siguiente diagrama de Gantt:

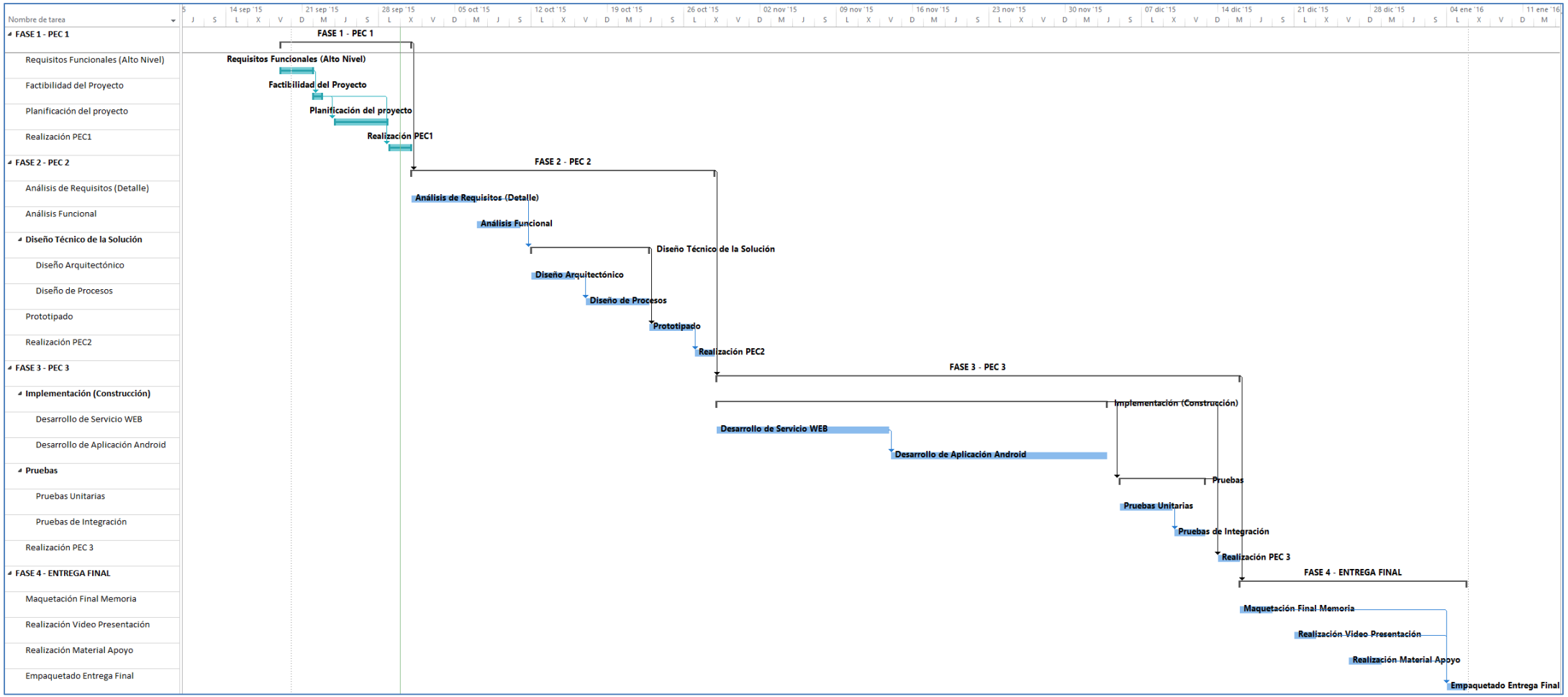


Ilustración 5 - Diagrama de Gantt del Proyecto

1.5 BREVE SUMARIO DE PRODUCTOS OBTENIDOS

Durante la realización del proyecto se generarán los siguientes entregables:

- **Software**
  - Aplicación Android "PHOTO DICOM", para ser instalada en cámaras fotográficas con Sistema Operativo Android, a partir de la versión 4.0 (u en otros dispositivos Android).
  - Web Service .NET "DICOMWebService", para ser instalada en equipo Windows (preferiblemente Windows Server) que disponga de las librerías Microsoft Framework 4.0

- **Documentación**
  - Memoria del proyecto (el presente documento).
  - Vídeo explicativo de funcionamiento de la solución.
  - Material de apoyo (diapositivas) para la presentación del proyecto.

1.6 SOBRE EL CONTENIDO DESCRITO EN LA PRESENTE MEMORIA

En el presente documento, memoria del Trabajo Final de Master, se recogen todos los ítems relacionados con el desarrollo del proyecto *PHOTO DICOM* ya explicado en puntos anteriores.

El apartado 1 que ahora finaliza aborda la motivación del desarrollo de esta aplicación, su justificación y las primeras decisiones adoptadas al respecto, así como un horizonte temporal previo (que pudiese ser modificado según evolucionen el resto de actividades).

En los siguientes apartados se documentan las tareas indicadas en el punto 1.4 de este documento.

2. ANALISIS DE REQUISITOS

2.1. INTRODUCCION

El presente proyecto tiene como objetivo el desarrollo de una aplicación sanitaria y por tanto con unos usuarios y contextos de uso concretos y bien definidos que veremos en los siguientes puntos.

Nota del Autor: Aunque acabo de citar que la aplicación está orientada al ámbito sanitario no se tendrán en cuenta en el entorno del presente proyecto la normativa sanitaria vigente o futura que pueda haber en cada momento por exceder al objetivo del presente proyecto que es el desarrollo de la aplicación en sí y no su puesta en el mercado con todos los beneplácitos de las Autoridades Sanitarias Competentes.

Una vez que hemos definido en la fase anterior el objetivo del proyecto, las metas a alcanzar, enfoque, método y planificación del trabajo, acometemos a continuación una investigación de campo que nos lleve a identificar los posibles usuarios del producto resultante de este proyecto, sus necesidades reales (o lo más próximo posible a las mismas), sus inquietudes y anhelos y cualquier característica o requisito que de dicha investigación se derive para lograr desarrollar una aplicación conforme a las exigencias ciertas del mercado.

Para la realización de citada investigación, procedemos a involucrarnos entre el personal sanitario destino inicial (y final) del producto a desarrollar.

2.2. ANALISIS, OBSERVACION E INVESTIGACION

Empezamos el proceso por la recogida de requisitos de usuario, con el objetivo de conocer la tipología de los mismos y los flujos de trabajo llevados por éstos. Iniciamos el trabajo de campo realizando una *observación e investigación contextual*. Para ello y previa obtención de las pertinentes autorizaciones nos desplazamos a un par de Hospitales Públicos para ver cómo trabajan los potenciales “clientes” de la aplicación: observamos el flujo de trabajo de un departamento de dermatología y de otro de cirugía estética.

El principal motivo que nos lleva a elegir este tipo de técnica de análisis es que aun siendo hasta cierto punto intrusiva en las acciones cotidianas de los servicios médicos, es la menos intrusiva y que menos obstaculiza la labor asistencial de los facultativos.

Dado que la parte puramente médica no nos es de nuestro interés, nos centramos en la fase de recogida y procesamiento de muestras de estos departamentos y, dichas muestras, no son más que la realización de fotos de diversas lesiones y patologías cutáneas.

Los objetivos perseguidos por la toma de información efectuada son los siguientes:

- Identificación del personal involucrado en la toma de fotografías y su posterior tratamiento.
- Flujos de trabajo diarios en los que se involucra al personal sanitario y pacientes.
- Equipos (tecnológicos) utilizados durante todas las fases.
- Tratamiento posterior y custodia de las muestras (fotografías).
- Procesos, pautas y contextos de uso.
- Necesidades detectadas y posibles requerimientos.

Para alcanzar estos objetivos nos involucramos con el personal sanitario (en la medida en que nos dejan estar presentes en ciertos actos médicos) y observamos el día a día de facultativos, enfermería y otro personal auxiliar.

La observación realizada ha incluido los siguientes hitos:

- Identificación del flujo de pacientes que deben ser atendidos por ambos departamentos:
  - Se inicia el proceso en el momento de citación (presencial o vía carta).
  - El paciente acude a su cita el día y hora indicados.
  - Recepción del paciente, recogida del volante médico y paso a sala de espera.
  - Prestación del servicio (se atiende al paciente). Incluye la recogida de muestras si da lugar.
  - Espera del paciente fuera de la consulta.
  - Entrega del informe médico y posible citación para nueva consulta o para pruebas complementarias de otros servicios médicos.
- Identificación del personal involucrado en cada fase del ciclo de flujo de pacientes.
- Identificación de las herramientas utilizadas en cada caso y para cada acción.
- Observación y registro de posibles mejoras del flujo de trabajo en base a la introducción de nuevas herramientas tecnológicas y, en particular, del posible uso de una aplicación como la planteada en este proyecto.
- Estudio de contextos de uso y casos de uso de la solución planteada.

2.2.1. RESULTADOS Y CONCLUSIONES

Se aprecia en ambos casos que la forma de utilización de la tecnología en estos servicios es aún muy primitiva (al margen de la calidad de las cámaras fotográficas utilizadas para la obtención de las evidencias clínicas). Identificamos los siguientes problemas:

- Aunque el flujo de trabajo con los pacientes está más o menos normalizado (tal como se indicaba en los puntos anteriores), la mecánica o sistemática para la toma de evidencias clínicas o diagnósticas (fotografías) no está estandarizada, cada facultativo (junto con su equipo) realizan las distintas acciones de forma no uniforme.
- Los itinerarios utilizados para la toma de las evidencias posibilitan, con un relativamente alto grado de riesgo, que pueda llegar a haber una confusión entre evidencias y pacientes (datos demográficos).
- La custodia de las evidencias (fotografías) es muy precaria.
- No existe un método claro y definido de compartir la información con otros servicios que puedan requerir de las evidencias ya capturadas, por lo que en muchos casos se repite la toma de muestras.
- No se aprovecha la tecnología existente en los quirófanos (especialmente hablando del departamento de cirugía estética) para llevar la imagen diagnóstica hasta la mesa de operaciones.
- Las evidencias capturadas no se incorporan a la *Historia Clínica Electrónica* (HCE) del paciente.

Cabe destacar que aunque se han observado dos servicios distintos (Dermatología y Cirugía Estética), con particularidades y diferencias importantes en algunos aspectos, a la hora de desarrollar las actividades relacionadas con el presente proyecto (toma de evidencias clínicas/diagnósticas) su funcionamiento es muy similar, apreciándose pocas diferencias en dichas actividades.

2.2.2. PERFILES IDENTIFICADOS

Gracias a la observación y análisis efectuados podemos identificar los perfiles de posibles usuarios de la aplicación, como podremos apreciar a continuación, las agrupaciones resultantes vienen dadas tanto por la categoría profesional del usuario como por aspectos comunes de tareas realizadas por los mismos.

2.2.2.1. PERFIL FACULTATIVO

El perfil facultativo se corresponde con los médicos de los Departamentos (dermatología, cirugía estética, etc.) que pasan consulta y exploran a los pacientes.

PERFIL	FACULTATIVO
CARACTERISTICAS DEL PERFIL	
DATOS DEMOGRÁFICOS	Personal adulto de todas las edades: entre 25 y 65 años.
INTERESES Y MOTIVACIONES	La toma de muestras (fotografías) es una fase más de su actividad asistencial. Quieren perder el menor tiempo posible en estas tareas.
HABILIDAD TECNOLÓGICA	Excepto el personal más joven, generalmente es personal con poca experiencia en el uso de ordenadores y *gadgets* tecnológicos.
CONTEXTOS DE USO (Ver descripción de contextos en siguiente apartado)	
CONTEXTO 1	Consulta Médica – Exploración del Paciente.
CONTEXTO 2	Consulta Médica – Post-Exploración del Paciente.
ANALISIS DE TAREAS	
TAREA 1	Acceso al sistema e identificación del facultativo.
TAREA 2	Consulta de pacientes pendientes (consulta de agenda).
TAREA 3	Realización de la toma de muestras (fotografías).
TAREA 4	Asociación de paciente y fotografías realizadas.
TAREA 5	Envío de fotografías al servidor para su archivo.
TAREA 6	Consulta de histórico reciente.
CARACTERISTICAS Y REQUISITOS DESEABLES	
REQUISITO 1	Facilidad de uso, usuarios con poca habilidad tecnológica
REQUISITO 2	Interfaz sencilla. Pasos (tareas) claros.
REQUISITO 3	Identificación del facultativo que está logado y ha realizado la actividad médica.

2.2.2.2. PERFIL EFERMERIA

El perfil enfermería se corresponde con el personal de enfermería de los Departamentos (dermatología, cirugía estética, etc.) que apoyan a los facultativos en las tareas diarias. Se prevé un acceso a la aplicación más esporádica y siempre asociado al facultativo correspondiente.

PERFIL	ENFERMERIA
CARACTERISTICAS DEL PERFIL	
DATOS DEMOGRÁFICOS	Personal adulto de todas las edades: entre 25 y 65 años.
INTERESES Y MOTIVACIONES	Consulta de agenda y pacientes realizados (histórico).
HABILIDAD TECNOLÓGICA	Baja habilidad tecnológica por regla general.
CONTEXTOS DE USO (Ver descripción de contextos en siguiente apartado)	
CONTEXTO 1	Consulta Médica – Exploración del Paciente.
CONTEXTO 2	Consulta Médica – Post-Exploración del Paciente.
CONTEXTO 3	Consulta Médica – Actividades fuera del horario de atención a pacientes.
ANALISIS DE TAREAS	
TAREA 1	Acceso al sistema e identificación del facultativo.

TAREA 2	Consulta de pacientes pendientes (consulta de agenda).
TAREA 3	Consulta de histórico reciente.
TAREA 4	Excepcionalmente resto de tareas del perfil facultativo
CARACTERISTICAS Y REQUISITOS DESEABLES	
REQUISITO 1	Facilidad de uso, usuarios con poca habilidad tecnológica
REQUISITO 2	Interfaz sencilla. Pasos (tareas) claros.
REQUISITO 3	Consulta de pacientes realizados en el día y fotos asignadas a los mismos.

2.2.2.3. PERFIL PERSONAL AUXILIAR

El perfil de personal auxiliar se corresponde con el resto de personal relacionado con los Departamentos (dermatología, cirugía estética, etc.) que sirven de apoyo para ciertas funciones (secretaría, auxiliares, celadores, etc.). Se prevé un acceso muy poco frecuente a la aplicación y siempre asociado a un facultativo concreto facultativo.

PERFIL	PERSONAL AUXILIAR
CARACTERISTICAS DEL PERFIL	
DATOS DEMOGRÁFICOS	Personal adulto de todas las edades: entre 25 y 65 años.
INTERESES Y MOTIVACIONES	Consulta de agenda y pacientes realizados (histórico) de forma muy esporádica.
HABILIDAD TECNOLÓGICA	Baja habilidad tecnológica por regla general.
CONTEXTOS DE USO (Ver descripción de contextos en siguiente apartado)	
CONTEXTO 1	Consulta Médica – Actividades fuera del horario de atención a pacientes.
ANALISIS DE TAREAS	
TAREA 1	Acceso al sistema e identificación del facultativo, de forma poco frecuente.
TAREA 2	Consulta de pacientes pendientes (consulta de agenda), de forma poco frecuente.
TAREA 3	Consulta de histórico reciente, de forma poco frecuente.
CARACTERISTICAS Y REQUISITOS DESEABLES	
REQUISITO 1	Facilidad de uso, usuarios con poca habilidad tecnológica
REQUISITO 2	Interfaz sencilla. Pasos (tareas) claros.
REQUISITO 3	Consulta de pacientes realizados en el día y pendientes. No necesitan ver la evidencia clínica (fotografía)

2.2.3. REQUISITOS IDENTIFICADOS DEL ANALISIS DE PERFILES

Una vez finalizada la fase de indagación, investigación y análisis identificamos los siguientes requisitos para la aplicación a desarrollar:

- REQ01 – Sistema de autenticación con usuario y contraseña / Inicio de sesión.
- REQ02 – Selección de agenda y del facultativo "responsable de sala" (*facultativo que se hará responsable de las evidencias capturadas y transmitidas*).
- REQ03 – Consulta *online* de agenda (pacientes pendientes).
- REQ04 – Asociación a paciente de fotografías realizadas (*fotografías realizadas con las utilidades nativas del dispositivo*).
- REQ05 – Transmisión de fotografías a almacenamiento, conversión a formato adecuado (*DICOM*) y custodia.
- REQ06 – Consulta de histórico realizado con el dispositivo en el que está instalada la aplicación. (*Histórico de la cámara fotográfica*).
- REQ07 – Seguridad de las comunicaciones: transmisión de datos de alto nivel LOPD.
- REQ08 – Cierre de sesión.
- REQ09 – Cambio de agenda en curso y/o facultativo responsable.

No se incluyen en esta relación los requisitos que debe satisfacer el servicio Web sobre el que se apoyará la aplicación ya que los anteriormente citados son requisitos de usuario y el Servicio Web queda al margen del conocimiento de dichos usuarios.

Se concretarán las funcionalidades de dicho Servicio Web posteriormente.

2.3. REQUISITOS FUNCIONALES

Una vez realizado el proceso de identificación de perfiles en base al trabajo de campo efectuado, fijamos cuáles serán los requisitos funcionales incluidos en este proyecto y de los cuales se definirán posteriormente los casos de uso son los siguientes:

- **REQF01 – Iniciar Sesión**: Se permitirá que un usuario se pueda identificar ante el sistema y acceder al mismo. De igual forma un usuario no autorizado no podrá acceder al resto de funcionalidades. El acto de autenticación se hará en base a un usuario y una contraseña. (Caso de Uso CU01)
- **REQF02 – Finalizar Sesión/Cierre**: Finaliza la sesión del usuario actual y cierra la aplicación Android. (CU02)

- **REQF03 – Consulta de Lista de Trabajo**: Para una agenda determinada se interrogará a un servidor de Worklist DICOM la lista de pacientes pendientes y se mostrará en la aplicación en un listado con la información necesaria. (CU03)
- **REQF04 – Visualización de información ampliada de ítem de Lista de Trabajo**: Para un ítem concreto (paciente) de la Lista de Trabajo (obtenida de un servidor Worklist DICOM), se permitirá al usuario visualizar una ficha de información completa sobre el paciente y la exploración a realizar. (CU04)
- **REQF05 – Asociación de fotografías a paciente y envío a PACS**: Para un ítem concreto (paciente) de la Lista de Trabajo (obtenida de un servidor Worklist DICOM), se permitirá al usuario seleccionar una o varias fotografías realizadas con el dispositivo y se enviarán con conversión previa a formato DICOM a un servidor PACS vía protocolo de comunicaciones DICOM. (CU05)
- **REQF06 – Consulta de Actividad Histórica Realizada**: Se permitirá al usuario consultar un listado de exploraciones realizadas con el propio dispositivo. (CU06)
- **REQF07 – Visualización de información ampliada de ítem de lista de Actividad Histórica**: Para un ítem concreto (paciente-exploración) de la lista de Actividad Histórica se mostrará una ficha con información completa con la información existente de dicha exploración. (CU07)
- **REQF08 – Visualización de imágenes asociadas a Paciente/Prestación**: Para un ítem concreto (paciente) de la lista de Actividad Histórica se permitirá visualizar las fotografías asociadas a dicha exploración (en caso de mantenerse en la memoria del dispositivo). (CU08)

2.4. REQUISITOS NO FUNCIONALES

Además de los requisitos funcionales que acabamos de relatar la plataforma PHOTO DICOM tiene también una serie de requisitos no funcionales entre los que podemos destacar:

- REQNF01 – Seguridad: Dado que la transmisión de información relativa a la salud de las personas es de la más alta protección según la LOPD, todas las comunicaciones entre cliente (aplicación Android) y servidor deben ser seguras.
- REQNF02 – Compatibilidad DICOM: El sistema debe ser DICOM 3.0 Standard, y por tanto debe poder comunicarse con cualquier sistema de lista de trabajo (Worklist DICOM) y cualquier sistema de almacenaje PACS que cumpla dicho estándar independientemente del proveedor del mismo.
- REQNF03 – Compatibilidad Android 4.0: Dado que en el momento del desarrollo del proyecto PHOTO DICOM se desconoce con exactitud el dispositivo Android que va a ser

utilizado para la captura de evidencias clínicas (fotografías) se debe garantizar que al menos la compatibilidad será con versión de Android 4.0 o superiores.

- REQNF04 – Facilidad de uso: El usuario objetivo de la aplicación no tiene por qué tener altos conocimientos de tecnología y dispositivos tecnológicos, por lo que la utilización de la plataforma debe ser sencilla para todo usuario.

3. ANALISIS DEL SISTEMA

Una vez identificados los potenciales usuarios de la aplicación, así como los requisitos que fijamos para el presente proyecto, a continuación describiremos las situaciones de uso previsibles del sistema, tratando de establecer los contextos en los que tiene lugar las acciones del personal sanitario y situaciones concretas que se generan a partir de éstos. Estas "situaciones de uso" son lo que se denomina comúnmente escenarios de uso.

3.1. ESCENARIOS DE USO

Un escenario de uso se define como la descripción de un personaje en una situación de uso real del producto a elaborar y, esta descripción incluye el contexto en el que tiene lugar y la secuencia de acciones que se llevan a cabo. [Garreta et al. 2011]

3.1.1. ESCENARIO DE USO 1: CONSULTA MÉDICA – EXPLORACION DEL PACIENTE

Contexto:

- Un paciente después de entregar la documentación (volante) al personal sanitario y haber esperado en la sala de espera es llamado a consulta.
- El paciente entra en la consulta y explica al facultativo su dolencia.
- El facultativo realiza un informe de la prestación realizada.
- Opcionalmente toma evidencias (fotografías) de la dolencia del paciente.
- Una vez realizada la exploración se indica al paciente que espere en sala de espera por su informe y se le dan las instrucciones oportunas (volver a citarse, medicación, o cualquier otra evolución clínica que deba seguir).

Objetivo:

- Diagnóstico del paciente o control de una dolencia previa.

Personal que interviene y tareas efectuadas:

- Facultativo.
  - Generalmente será el que capture las evidencias clínicas (fotografías).
  - Realiza el informe
  - Indica las pautas a seguir al paciente.
- Enfermería.
  - Llama al paciente para que entre desde la sala de espera.

- Excepcionalmente puede ser la encargada de tomar las evidencias clínicas.
- Realiza las tareas clínicas necesarias (vendajes, etc.)

Necesidades de información:

- Enfermería: lista de pacientes pendientes. Información clínica previa si existiese.
- Facultativos: lista de pacientes pendientes. Información clínica previa si existiese.

Funcionalidades necesarias:

- La aplicación debe ser capaz de consultar la lista de pacientes pendientes (servida desde un sistema informático).
- Se debe poder seleccionar un paciente de la lista.
- Se deben poder realizar las capturas de evidencias clínicas (fotografías).
- Se deben poder asociarlas al paciente actual.

3.1.2. ESCENARIO DE USO 2: CONSULTA MÉDICA – POST-EXPLORACION DEL PACIENTE

Contexto:

- El paciente ya ha sido explorado y se le ha indicado que debe esperar por el informe en la sala de espera.
- El facultativo ultima la realización del informe.
- Se aprovecha este momento entre paciente y paciente para archivar documentación y evidencias clínicas (fotografías) en caso de existir.
- Finalmente la enfermera sale a sala de espera y le entrega al informe al paciente.

Objetivo:

- Finalización de la prestación efectuada. Entrega de informe y archivo de documentación (física o digitalmente).

Personal que interviene y tareas efectuadas:

- Facultativo.
  - Termina de realizar el informe y lo envía a la impresora.
  - Puede volver a consultar las evidencias clínicas.
  - Puede ser el encargado de enviar las evidencias clínicas (fotografías) a archivo y custodia.

- Enfermería.
    - Recoge el material utilizado.
    - Puede ser el encargado de enviar las evidencias clínicas (fotografías) a archivo y custodia.
    - Prepara el material para la siguiente consulta.
    - Sale a entregar el informe al paciente.

Necesidades de información:

- Facultativos: Imágenes tomadas del paciente.

Funcionalidades necesarias:

- La aplicación debe ser capaz de consultar las imágenes asociadas al paciente durante la exploración para su consulta.
- La aplicación debe permitir el envío de imágenes a archivo y custodia.
- La aplicación debe permitir "cerrar" el paciente en curso.

3.1.3. ESCENARIO DE USO 3: CONSULTA MÉDICA – ACTIVIDADES FUERA DEL HORARIO DE ATENCION A PACIENTES

Contexto:

- El personal del departamento puede estar:
    - Preparando las próximas consultas con pacientes
    - Cerrando asuntos pendientes de las últimas consultas (exploraciones) realizadas.
- El personal auxiliar se encarga por ejemplo de sacar listados de pacientes pendientes futuros.
- Comprueba también cuáles de los pacientes citados se han realizado.
- Archivan documentación o piden historias clínicas.

Objetivo:

- Preparación de la consulta para atender pacientes.
- Finalización de tareas pendientes de exploraciones previas.

Personal que interviene y tareas efectuadas:

- Personal Auxiliar.
    - Puede necesitar consultar el histórico de pacientes realizados en el día.

    - Puede necesitar consultar la lista de pacientes pendientes a través de la aplicación.
- Enfermería.
    - Puntualmente pueden colaborar en el cierre de actividad o preparación de las siguientes exploraciones realizando las mismas actividades que el personal auxiliar o similares.

Necesidades de información:

- Personal Auxiliar (y puntualmente personal de enfermería): Histórico de pacientes realizado y listado de pacientes pendientes.

Funcionalidades necesarias:

- La aplicación debe ser capaz de consultar los pacientes realizados más recientemente.
- La aplicación debe ser capaz de consultar la lista de pacientes pendientes (servida desde un sistema informático).

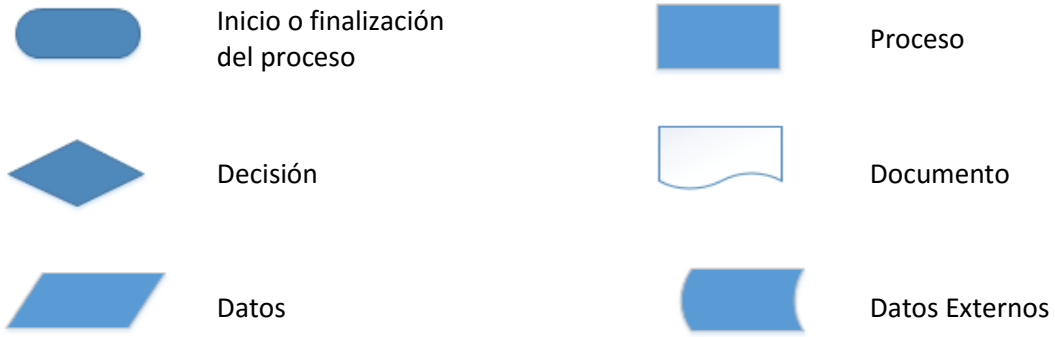
3.2. FLUJOS DE INTERACCIÓN

Una vez que hemos identificado los distintos escenarios de uso de la aplicación, a continuación procedemos a definir los flujos de interacción posibles para dichos escenarios. Para su representación gráfica utilizaremos Diagramas de Flujo.

Un diagrama de flujo es una representación gráfica de un proceso. Cada etapa del proceso se representa con un símbolo gráfico, el cual lleva asociado una breve descripción de dicha etapa dentro del proceso general. Los símbolos gráficos se unen entre sí con flechas que indican la dirección del flujo dentro del proceso.

Estos diagramas ofrecen una descripción visual de las actividades implicadas, definidas en los puntos anteriores, mostrando la relación secuencial entre ellas y facilitando de este modo la compresión de cada actividad y su interrelación con el resto de fases de dicha actividad.

Leyenda:



3.2.1. CONSULTA MÉDICA – EXPLORACIÓN DEL PACIENTE

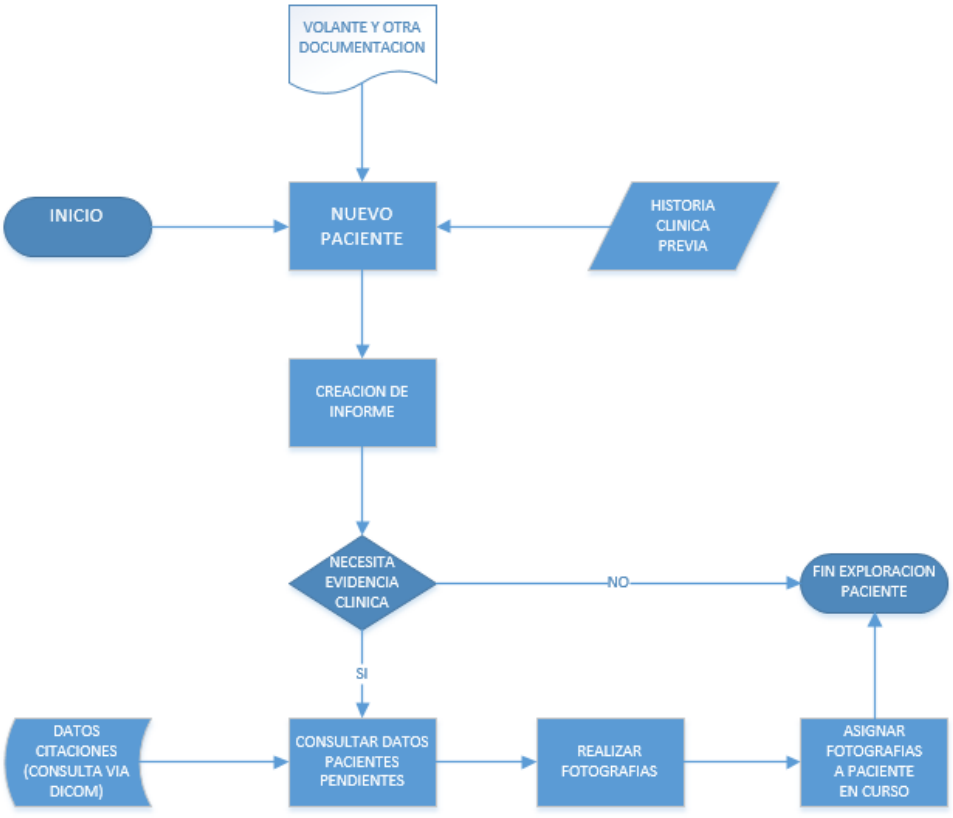


Ilustración 6 - Consulta Médica - Exploración del Paciente

3.2.2. CONSULTA MÉDICA – POST-EXPLORACIÓN DEL PACIENTE

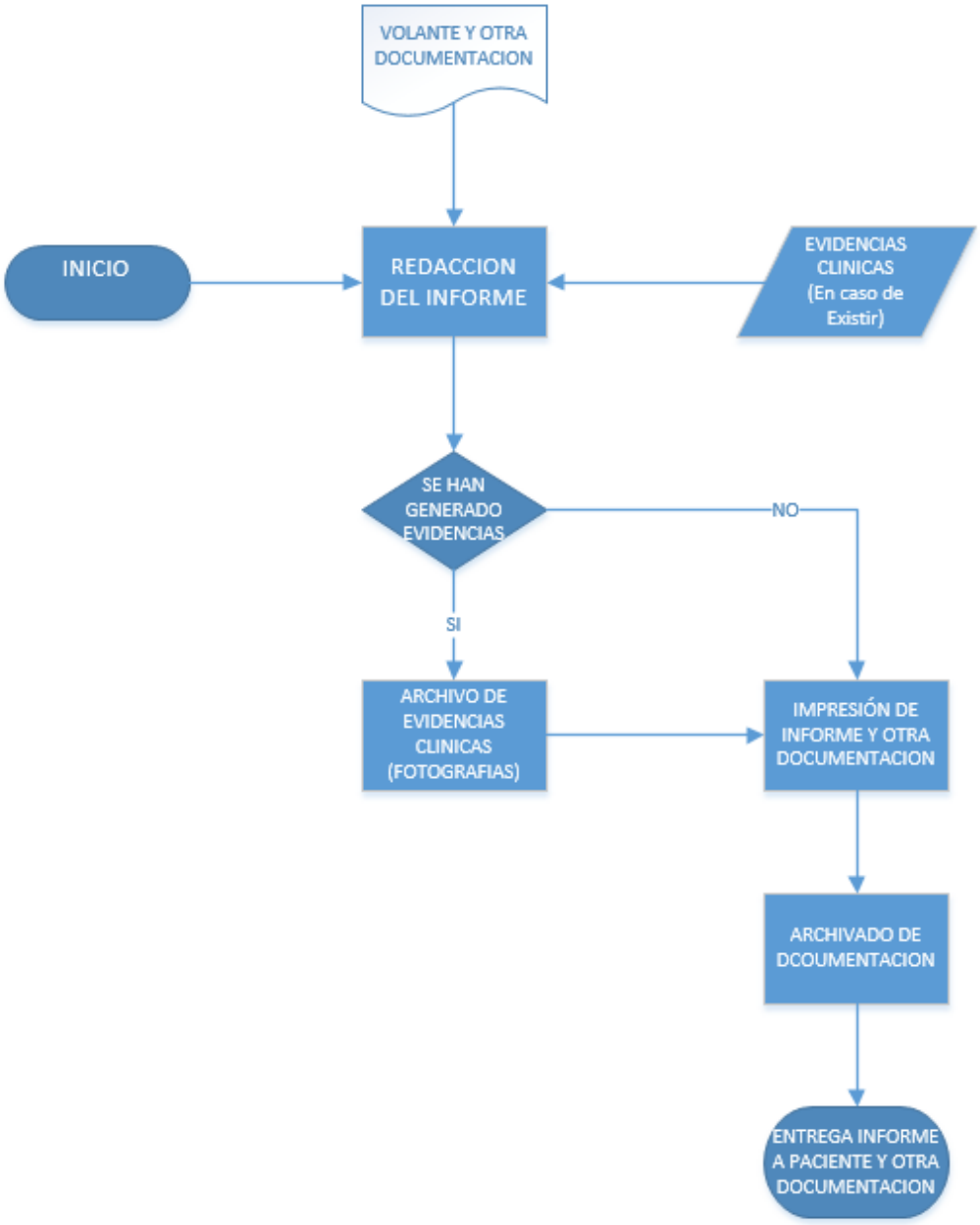


Ilustración 7 - Consulta Médica - Post-exploración del paciente.

3.2.3. CONSULTA MÉDICA – ACTIVIDADES FUERA DEL HORARIO DE ATENCIÓN A PACIENTES (REVISION DOCUMENTACION)

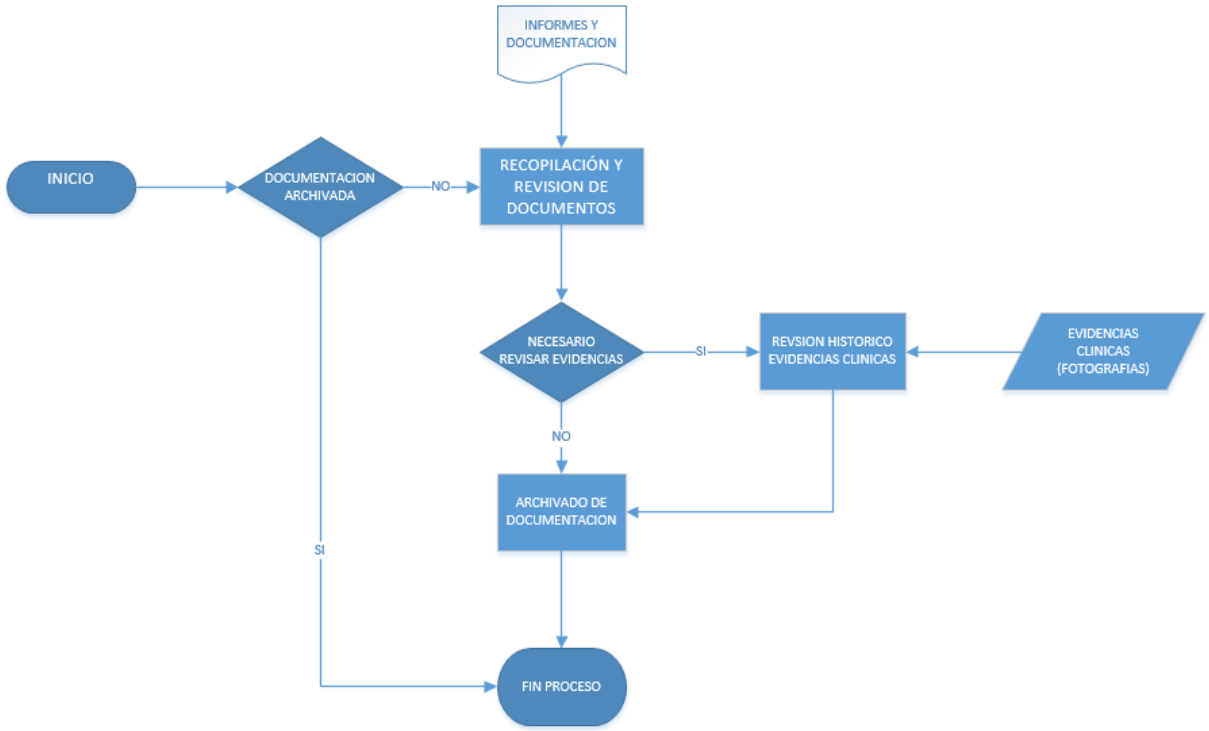


Ilustración 8 - Consulta Médica - Actividades fuera del horario de atención a pacientes (Revisión Documentación)

3.2.4. CONSULTA MÉDICA – ACTIVIDADES FUERA DEL HORARIO DE ATENCIÓN A PACIENTES (PREPARACION CONSULTA)

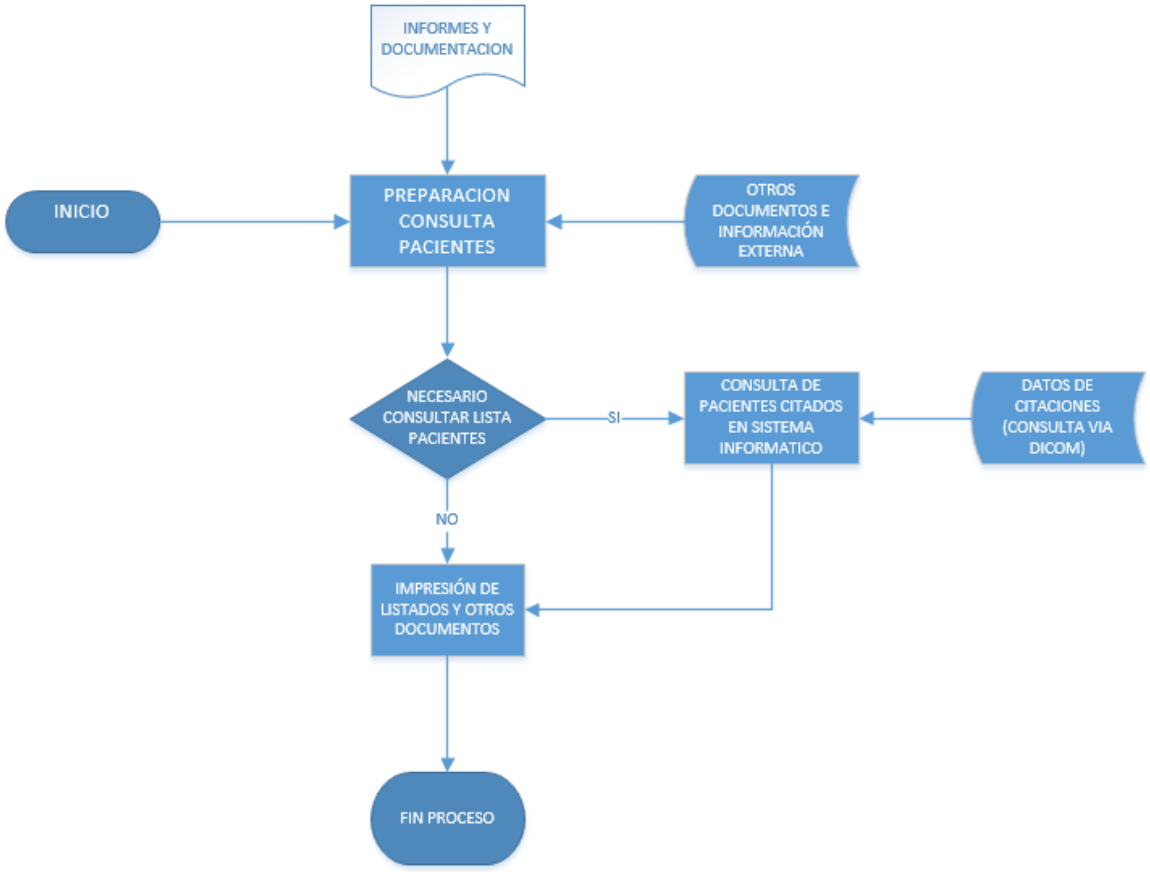


Ilustración 9 - Consulta Médica - Actividades Fuera del Horario de Atención a Pacientes (Preparación Consulta)

4. DISEÑO ARQUITECTÓNICO

A continuación abordaremos el diseño de la arquitectura de la plataforma PHOTO DICOM analizando tres aspectos fundamentales:

- Persistencia de la aplicación
- Entidades y clases
- Arquitectura del sistema

4.1. PERSISTENCIA EN LA PLATAFORMA PHOTO DICOM

En aras de una mayor portabilidad y para lograr una reducción de costos la plataforma se ha diseñado sin conexión directa a BD. Por tanto contamos con las siguientes características:

- Web Service *PHOTODICOM*:
  - Es el encargado de los siguientes procesos típicamente relacionados con información almacenada en base de datos:
    - Autenticación de usuarios.
    - Entrega de listado de agendas a la aplicación Android.
    - Entrega de listado de facultativos a la aplicación Android.
    - Almacena la información de conexión al PACS del centro sanitario.
  - Para evitar el tener que contar con un Gestor de Base de Datos y complicar el escenario, dado que la información que necesita la plataforma en sí a través del WebService es limitada, se opta por almacenar dicha información en fichero, en listados de datos de texto.

- Aplicación Android *PHOTO DICOM:*
  - Los listados de agendas, facultativos y el acto de autenticación de usuarios se realizan contra el WebService (que como acabamos de indicar no almacena en un SGBD).
  - La aplicación Android sí almacena la información de pruebas realizadas con el dispositivo en el que se encuentra instalada. En este caso dicha persistencia se realiza en una base de datos local al dispositivo, SQLite, con operaciones SQL estándar.

4.2. ENTIDADES DEL DOMINIO Y CLASES

El modelo de dominio de la plataforma es realmente simple y se representa en el siguiente esquema:

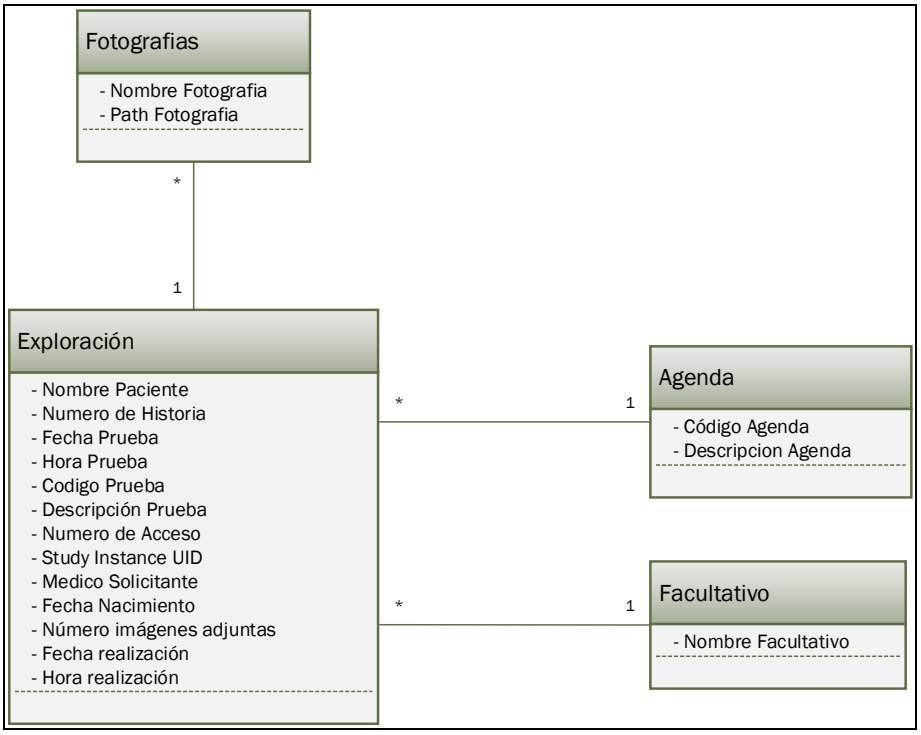


Ilustración 10 - Modelo del Dominio

Las entidades que figuran en el dominio son:

- *Exploración*: representa un ítem de la lista de trabajo. Contiene toda la información sobre el paciente a explorar, así como los datos de la prestación citada o realizada.

- *Agenda*: una agenda tiene citadas 1 o más exploraciones. 1 Exploración sólo pertenece a una agenda concreta.

- *Facultativo* (responsable de la exploración): 1 Facultativo realiza 1 o más exploraciones. 1 Exploración es realizada por un único facultativo.

4.3. ARQUITECTURA DEL SISTEMA

La solución se ha planteado siguiendo el paradigma SOA (*Service Oriented Architecture* ó *Arquitectura Orientada a Servicios*). De esta forma nos planteamos el diseño del sistema como un conjunto de *piezas* más o menos especializadas con las cuales construir la solución.

Estas *"piezas"* (operaciones) del servicio pueden ser combinadas con otras operaciones de otro servicio de una manera estándar. De esta forma logramos:

- Granularidad, siendo posible reutilizar unidades de distintos tamaños.
- Bajo acoplamiento: y por tanto baja dependencia entre operaciones.
- Permite diferentes implementaciones.

Una plataforma bajo el paradigma SOA con cierta madurez nos permite tener plataformas de ejecución distintas (Aplicación Android <> Web Service <> PACS), con implementaciones en tecnologías distintas (Java, .NET, Delphi), logrando hacer SOA no solo a nivel de diseño sino también a nivel marco de trabajo.

Como conector hemos elegido Web Services ya que implementan un conjunto de protocolos y estándares ampliamente conocidos y distribuidos, logrando altos grados de interoperabilidad. Además al funcionar sobre HTTP, logramos evitar problemas de seguridad y limitaciones varias de las redes seguras de los centros de salud.

4.4. SOLUCION ESCALABLE

Aunque no se prevé que en los escenarios de uso la carga de trabajo sea muy elevada – no habrá un elevado número de cámaras fotográficas Android con PHOTO DICOM –, el sistema se ha diseñado para poder ser escalable. Para ello no haría falta más que incluir nuevos servidores Web que atiendan las peticiones de los distintos clientes (y opcionalmente un balanceador de carga por delante de ellos).

4.5. DIAGRAMA DE ARQUITECTURA DE LA PLATAFORMA

A continuación mostramos la estructura física de la solución en un escenario típico y previsible:

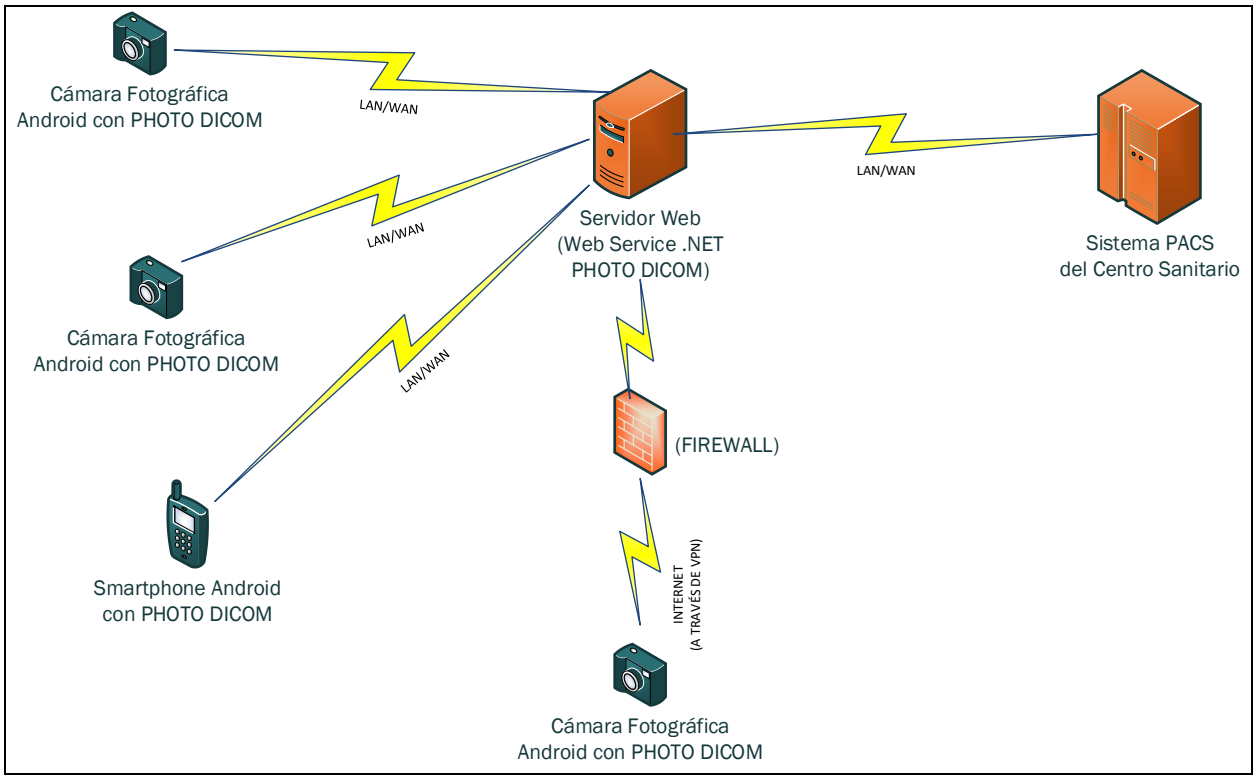


Ilustración 11 - Esquema físico general

Como se puede observar el funcionamiento general es el siguiente:

- Los clientes Android con PHOTO DICOM, se comunican por HTTP (Web Service) con el servidor Web de la plataforma.
- El servidor Web de PHOTO DICOM se comunica vía protocolo DICOM con el PACS central tanto para interrogar al sistema de Worklist como para el envío de imágenes por DICOM.

Excepcionalmente podría haber clientes fuera de la red corporativa sanitaria, que se conectasen vía VPN a dicha red.

4.6. VISTA DE COMPONENTES

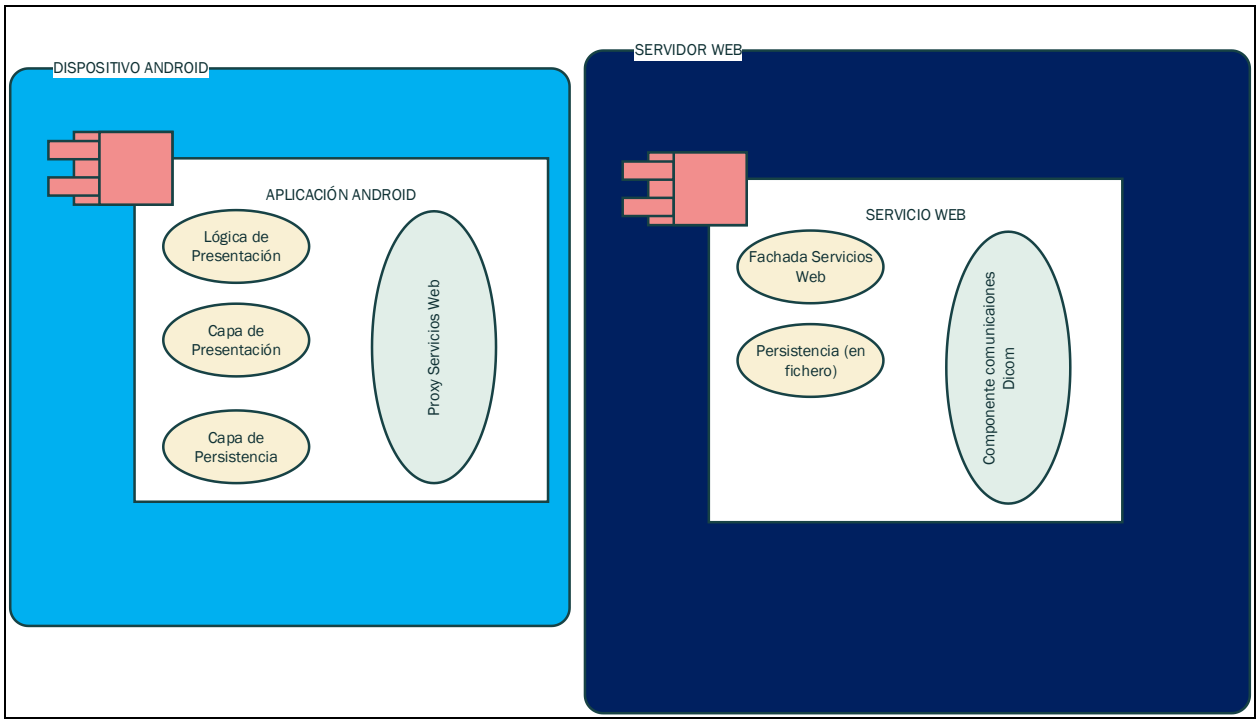


Ilustración 12 - Vista de componentes del sistema

4.7. DISEÑO DE LA PERSISTENCIA

4.7.1. PERSISTENCIA EN PARTE SERVIDOR (WS)

Como ya se ha mencionado a lo largo de este documento se ha preferido evitar el tener que instalar ningún Sistema de Gestión de Base de Datos (SGDB) para simplificar la implementación y evitar el posible coste de la licencia de dicho SGDB (que pudiera ser posiblemente de pago).

La información almacenada en la parte servidor se compone de:

- Información relativa a los Usuarios autorizados para el uso de la aplicación.
- Lista de médicos facultativos (responsables de las pruebas efectuadas).
- Lista de agendas-salas de exploración.

A su vez también se guarda la configuración del/de los distintos PACS a los que se pueda conectar la aplicación (dependiendo de la Agenda seleccionada) tanto a nivel de consulta de Worklist DICOM como para el envío por DICOM Store.

También se guarda un fichero de log para depuración y debug.

Toda esta información que acabamos de exponer se localiza directamente en el fichero Web.config del Web Service .NET, y por tanto está almacenada en fichero.

En caso necesario (por ejemplo listas excesivamente extensas de usuarios), se podría traspasar a un SGDB (por ejemplo Microsoft© SQL Server) con mucha facilidad y sin afectar a la aplicación Android que seguiría recibiendo los listados indicados de igual forma.

4.7.2. PERSISTENCIA EN LA APLICACIÓN ANDROID

Toda prueba realizada (y completada con éxito) en la aplicación es automáticamente eliminada de la *Worklist DICOM* dado que ya ha sido efectuada.

Para tener una trazabilidad de acciones efectuadas por cada usuario y con cada dispositivo caben dos posibilidades:

- Generar persistencia a nivel de Servidor: por cada prueba realizar almacenar dicha información en el servidor y que ésta pueda ser mostrada en la aplicación Android.
- Generar persistencia de forma local en el dispositivo Android. La información almacenada por tanto es relativa al uso realizado con cada dispositivo.

Para evitar nuevamente tener que contar con un SGDB a nivel servidor, decidimos desarrollar la segunda opción a través de la cual la persistencia se almacena a una base de datos relacional local al dispositivo Android.

Dicha base de datos es SQLite y está presente en todos los dispositivos Android.

La información almacenada es la siguiente:

- Información relativa a las pruebas/pacientes realizados (*maestro*)
- Información relativa a las imágenes de cada prueba/paciente realizado (*detalle*).

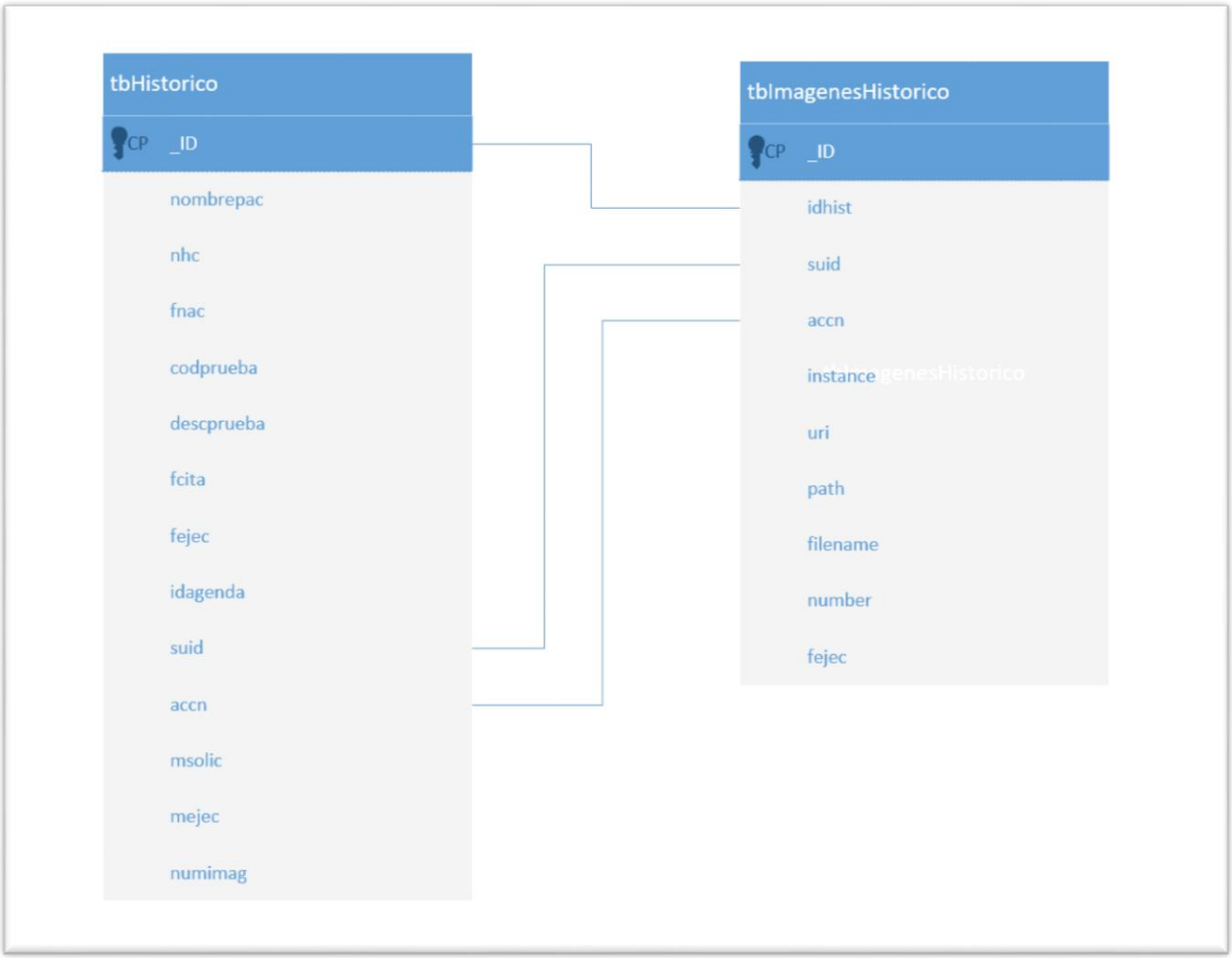


Ilustración 13 - Esquema de Base de Datos app Anroid local (SQLite)

Como podemos ver almacenamos la siguiente información:

- **Maestro: tbHistorico:**
- Id (clave primaria, autonumérico)
- Nombre del Paciente
- Número de Historia Clínica
- Fecha de Nacimiento
- Código de la prueba o exploración
- Descripción de la prueba o exploración
- Fecha de la cita
- Fecha de ejecución (realización) de la exploración
- Id Agenda

- Study Instance UID
- Accession Number
- Médico Solicitante
- Médico Ejecutor (responsable)
- Número de imágenes capturadas

A nivel de detalle tenemos:

➢ **Detalle: tbImagenesHistorico:**

- Id (clave primaria, autonumérico)
- Idhist (Id del maestro correspondiente)
- Study Instance UID*
- Accession Number*
- SOP Instance UID
- URI de la imagen (en formato String)
- Path de la imagen
- Nombre del archivo
- Número de imagen
- Fecha de ejecución (subida al servidor) de la imagen

* Como se puede apreciar estos campos son supérfluos al contar a nivel de detalle del id del maestro. Sin embargo para agilizar las consultas duplicamos esta información para no tener que cruzar ambas tablas. Se "*rompe*" el modelo relacional puro, pero se gana en agilidad.

4.8. DECISIONES TECNOLOGICAS

Aunque ya se han citado algunas de las decisiones tomadas durante la planificación del proyecto, hacemos aquí un resumen de las mismas:

APLICACIÓN ANDROID:

- Lenguaje de desarrollo: Java.
- Tipo de aplicación. Nativa.
- Persistencia: SQLite.

WEB SERVICE:

- Lenguaje de desarrollo: VB.NET
- Tipo de aplicación: Web Service SOAP
- Persistencia: ninguna (maestros en fichero web.config).

COMUNICACIONES

- **METODO:** Servicio Web SOAP (XML), WSDL.
- **SEGURIDAD:** Comunicaciones entre aplicación Android y Web Service con comunicaciones encriptadas según algoritmo **AES**, para evitar depender de la existencia de certificados para HTTPS.

4.9. PROTIPADO

La siguiente fase después del análisis, investigación y recogida de requisitos, y del diseño conceptual materializado en la identificación de perfiles, escenarios y la interacción entre ellos (ya presentados), es la creación de un prototipo de la aplicación.

Prototipo: *"Ejemplar original o primer molde en que se fabrica una figura u otra cosa."* [RAE]

Uno de los métodos más sencillos para contar con un modelo de la aplicación a desarrollar con el que poder trabajar su diseño, apariencia, usabilidad, *user experience* es la creación de un prototipo de la misma. Esto nos permitió comprobar de manera fiel cómo será la experiencia del usuario con nuestra aplicación.

En este punto realizamos dos fases de prototipado:

- Un prototipo de bajo nivel, realizado como primera aproximación al problema y generalmente hecho a mano alzada.
- Una segunda iteración sobre el prototipo anterior para lograr un prototipo de alta fidelidad y horizontal. Un prototipo de alta fidelidad trata de construir un modelo tan cercano al producto final como sea posible y, al ser horizontal se busca que éste contenga todas las funcionalidades, espacios y/o menús.

Dado que los prototipos realizados son parte de la fase de diseño, son susceptibles a cambios durante la fase de impementación y fueron entregados en un documento específico durante la fase de diseño DCU (*diseño centrado en el usuario*), incluyo aquí únicamente un par de capturas de la aplicación Android a modo de ejemplo. El Web Service al ser precisamente eso, un servicio, no requiere de un diseño de interfaz de usuario.

4.9.1 PROTOTIPO DE PHOTO DICOM (ANDROID)

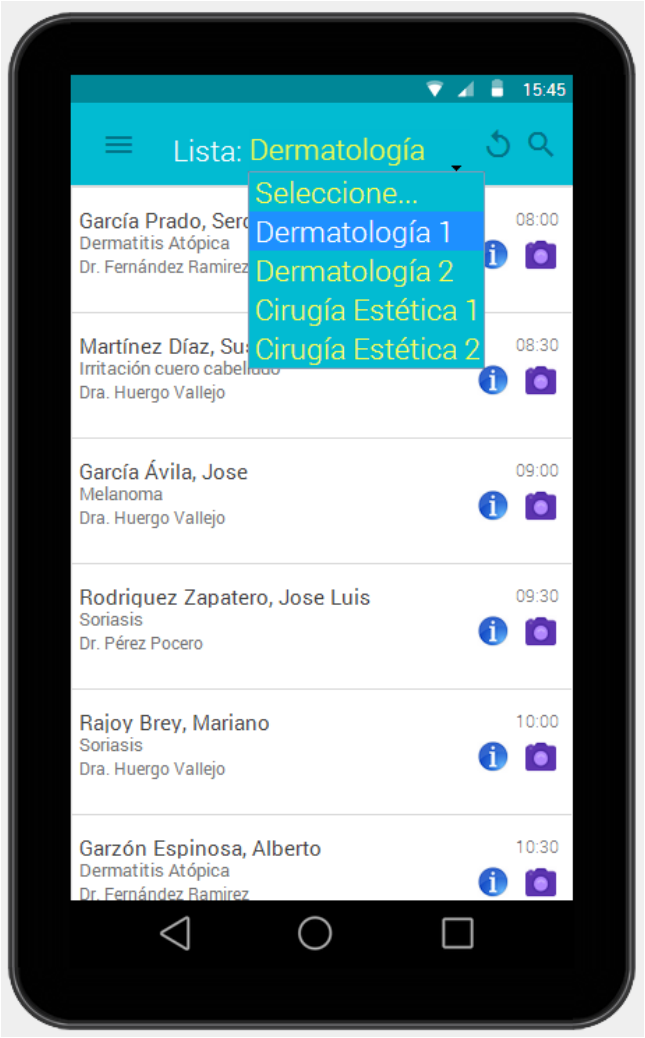


Ilustración 14 - Prototipo Lista de Trabajo (consulta Worklist)

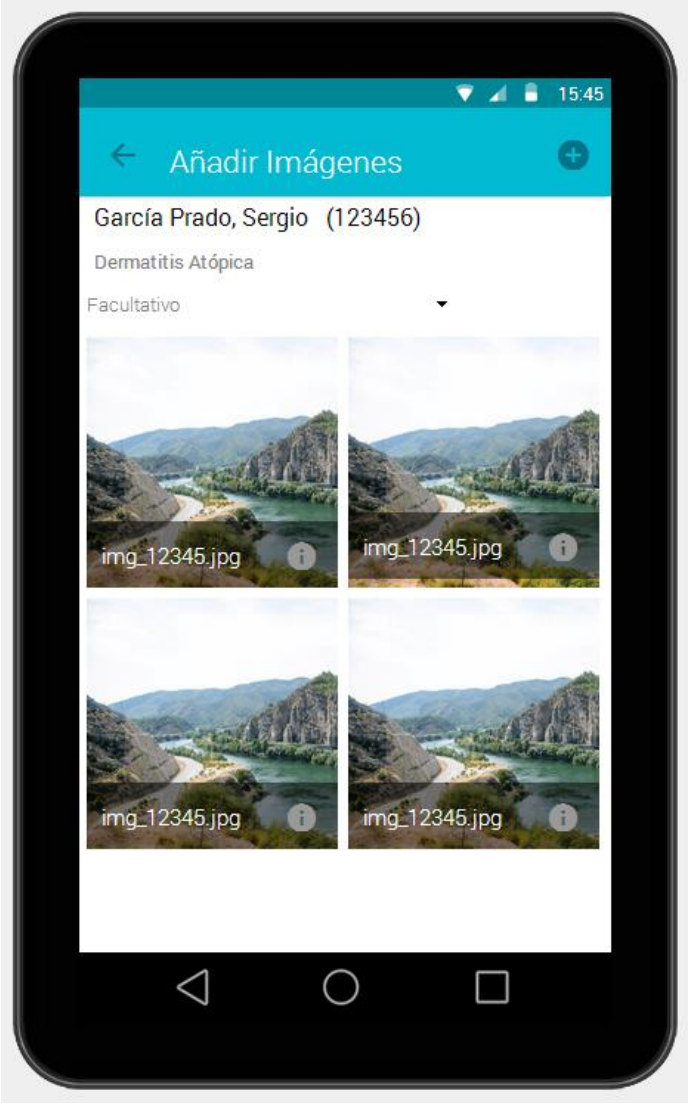


Ilustración 15 - Prototipo captura y selección de imágenes

4.10. ANÁLISIS ECONÓMICO Y VIABILIDAD

El fundamento de este proyecto no es estrictamente comercial. No obstante, no se descarta en un futuro cercano poder llegar a sacar el mercado el producto resultante.

Asimismo, tal como se ha venido comentando, tanto las tecnologías utilizadas como las herramientas de desarrollo no son gratuitas, por lo que existen unos costes de desarrollo.

Igualmente existen unos costes fijos de puesta en el mercado de la solución final.

Procedemos a desglosar estas cifras en los siguientes puntos.

4.10.1. COSTES DE DESARROLLO

Desarrollo Android[6]:

- Entorno de desarrollo: Android Studio – GRATUITO.
- Plantilla de *material design* utilizada – GRATUITO.
- Librerías de conexión con Web Services – GRATUITO.
- Logo de la aplicación: adquirido en repositorio de imágenes en línea- 14 €.

Desarrollo .NET[7]:

- Entorno de desarrollo: Visual Studio 2013 Profesional – 625 Eur (sin suscripción a MSDN).
- Librerías DICOM Objects por Medical Connections* - Existen diversos tipos de licenciamiento, en el caso del licenciamiento para desarrollo nos acogemos a la licencia "*Developer license*"[8] sin coste.

* Se dispone de licencia corporativa en la empresa en la que el autor de este proyecto presta sus servicios.

Nota: no se incluyen los costes derivados de la adquisición de un equipo para las labores de desarrollo y su correspondiente sistema operativo.

4.10.2. COSTES DE PUESTA EN PRODUCCION

- Hardware de servidor: coste variable, se necesita un servidor para instalar el Web Service. Dado que no se necesitan muchos recursos: a partir de 500 €.
- Hardware de captura (dispositivo Android): el dispositivo objetivo ideal de esta aplicación es una cámara semi-profesional con sistema operativo Android. Un ejemplo de esta cámara es la Samsung Galaxy Camara 2, con un coste aproximado de 500 €
- Software de servidor: Sistema Operativo Windows (en versiones 7, 8 o 10). Coste aproximado 200 €. Si la instalación es a mayor escala se necesitará una versión "Server" con un coste mínimo de 800 €.
- Librerías DICOM Objects por Medical Connections* - Existen diversos tipos de licenciamiento, en el caso del licenciamiento para puesta en producción el más

[6] Ver apartado 5: Implementación, para ver qué elementos y qué librerías son utilizados.
[7] Ver apartado 5: Implementación, para ver qué elementos y qué librerías son utilizados.
[8] Modelos de licenciamiento disponibles en el siguiente enlace: https://www.medicalconnections.co.uk/Licensing

adecuado parece ser el de "*Router and modifier applications*" con un coste de 400£ (± 500 €).

* Se dispone de licencia corporativa en la empresa en la que el autor de este proyecto presta sus servicios.

4.10.3. RESUMEN DE COSTES

COSTES DE DESARROLLO	
DESARROLLO .NET - LICENCIA VISUAL STUDIO	625 €
DESARROLLO ANDROID – ENTORNO PROGRAMACION Y LIBRERIAS UTILIZADAS	0 €
EQUIPO DE DESARROLLO, SISTEMA OPERATIVO, DISPOSITIVOS DE PRUEBAS, ETC.	EXCLUIDO

COSTES DE PUESTA EN PRODUCCIÓN (MINIMOS)	
EQUIPO SERVIDOR (HARDWARE)	500 €
EQUIPO SERVIDOR (SISTEMA OPERATIVO – WINDOWS 10)	200 €
LICENCIA DE DICOM OBJECTS PARA SERVIDOR	500 €
DISPOSITIVO DE CAPTURA DE IMÁGENES (CAMARA ANDROID)	500 €

4.10.4. MODELO DE NEGOCIO

El posible modelo de negocio se plantea con distintas variantes:

- Venta de equipo Android + App + SW Servidor + HW de Servidor.
- Venta de equipo Android + App + SW Servidor / Cliente aporta HW Servidor.
- App + SW Servidor / Cliente aporta dispositivos Android + HW Servidor.

Cabe destacar que tanto el Software como el Hardware de servidor en instalaciones en las que se vendan varias cámaras con su correspondiente app, sólo se cobran una vez.

El margen de beneficio estimado para que el proyecto tenga una rentabilidad suficiente es de unos 500 € por cámara + app vendida, conectada al sistema y con la correspondiente formación al usuario.

5. IMPLEMENTACIÓN

5.1. ENTORNO DE TRABAJO UTILIZADO

Para el desarrollo de la aplicación utilizamos las siguientes herramientas de desarrollo:

- *Codificación de App Android:* Android Studio versión 1.5.1

- *Codificación de Web Service*: Visual Studio 2013 Ultimate[9].
    - *Lenguaje de programación*: VB.NET

5.2. HERRAMIENTAS EXTERNAS UTILIZADAS, APIS Y SERVICIOS EXTERNOS

5.2.1. DESARROLLO ANDROID

Para el desarrollo de la aplicación Android hemos reutilizado los siguientes componentes:

- Plantilla *Material Design* de Pedro Olivera.

  URL: http://androidshenanigans.blogspot.pt/2015/03/material-design-template.html

  Download: https://github.com/kanytu/Android-studio-material-template

  Esta plantilla es una alternativa a los menús generados por Android Studio y simplifica en cierta medida la creación del menú flotante típico del paradigma Material Design.

  Licencia: *Apache License*

- Librerías SOAP (Simple Object Access Protocol) "KSOAP2"

  URL: http://simpligility.github.io/ksoap2-android/license-information.html

  Librerías que simplifican la conexión mediante SOAP a Web Services.

  Licencia: MIT License. Las propias librerías dependen de otras librerías licenciadas bajo distintos modelos (Permissive License y Public Domain).

[9] En la sección de costes se ha incluido la versión "Profesional" para poder tener un coste razonable. El autor del proyecto dispone de la licencia "Ultimate" por su actividad profesional, pero dicho nivel de licencia no es necesario para el desarrollo del proyecto.

5.2.2. DESARROLLO .NET

Para el desarrollo del WebService .NET se ha contado con los siguientes componentes:

- *DicomObjects .NET* de la empresa Medical Connections (Reino Unido)

  URL: https://www.medicalconnections.co.uk/Licensing

  Librerias que realizan las operaciones DICOM más habituales, permiten abstraerse de la comunicación DICOM y generar y manipular objetos DICOM.

  Licencia: Copyright. La empresa en la que realizo mi actividad profesional cuenta con licencia legal. Asimismo se ofrece licencia de desarrollador sin coste (con costes sólo por su distribución a clientes finales) y licencia de prueba de 60 días.

- Microsoft Framework .NET 4.0

  URL: https://www.microsoft.com/es-es/download/details.aspx?id=17718

  Framework de desarrollo de .NET, propiedad de Microsoft.

  Licencia: *MIT License / Apache License 2.0*

5.2.3. OTROS COMPONENTES UTILIZADOS (PACS)

Para poder testear el funcionamiento del sistema necesitamos disponer de un PACS. En concreto de dos servicios de éste:

- DICOM Worklist, para la recepción de pacientes pendientes para una agenda concreta (*AE-TITLE*).
- DICOM STORE CLASS PROVIDER, servicio encargado de recibir las imágenes enviadas desde la solución Photo DICOM.

A su vez este PACS utiliza el siguiente gestor de base de datos:

- Microsoft SQL SERVER en sus versiones 2000, 2005, 2008 ó 2012.

  En particular en el entorno de desarrollo utilizado se trabaja con MS SQL SERVER 2012 Enterprise Edition 64Bits.

  Licencia: Copyright (comercial).

Durante todo el desarrollo el autor de este proyecto ha probado y trabajado contra el siguiente PACS:

- *IRE STORE CHANNEL* de la empresa IRE Rayos X, S.A.

  URL: http://www.irerayosx.com

  Este PACS permite testear tanto la parte de consulta de Worklist DICOM como el envío DICOM de imágenes al PACS.

  Licencia: Copyright (comercial). Al autor es empleado de la empresa.

Asimismo se utilizado también un visor diagnóstico (DICOM) capaz de interrogar al PACS y obtener las imágenes almacenadas en éste procedentes de la solución Photo DICOM:

- *IRE RAD VISION PRO* de la empresa IRE Rayos X, S.A.

  URL: http://www.irerayosx.com

  Este visor diagnóstico nos permite comprobar las imágenes enviadas al PACS (tanto visualización con información DICOM insertada durante la creación del objeto).

  Licencia: Copyright (comercial). Al autor es empleado de la empresa.

Ambos productos (PACS y visor) pueden ser reemplazados por soluciones de código abierto dado que la solución PhotoDICOM no es propietaria, gracias a cumplir con el protocolo internacional DICOM 3.0.

Por último indicar el equipo de desarrollo y pruebas:

- Microsoft Surface PRO 3

  Intel i7, 8GB RAM, Disco 256 GB SSD.

5.3. DESARROLLO WEB SERVICE .NET

En este y el siguiente epígrafe desglosaremos parcialmente cómo se han desarrollado tanto el Web Service como la aplicación Android cliente de éste, así como sus peculiaridades y aspectos distintivos.

El Web Service .NET, denominado *DICOMWebService*, se compone de una serie de métodos públicos que son invocados por la aplicación Android.

Los métodos públicos, a excepción de uno, reciben la información cifrada y la devuelven asimismo cifrada a través del método que se explicitará durante el punto 5.5 de este documento.

Se puede acceder la lista completa de métodos a través de la URL:

http:/<ServidorWeb>/DICOMWebService/DICOMWebService.asmx

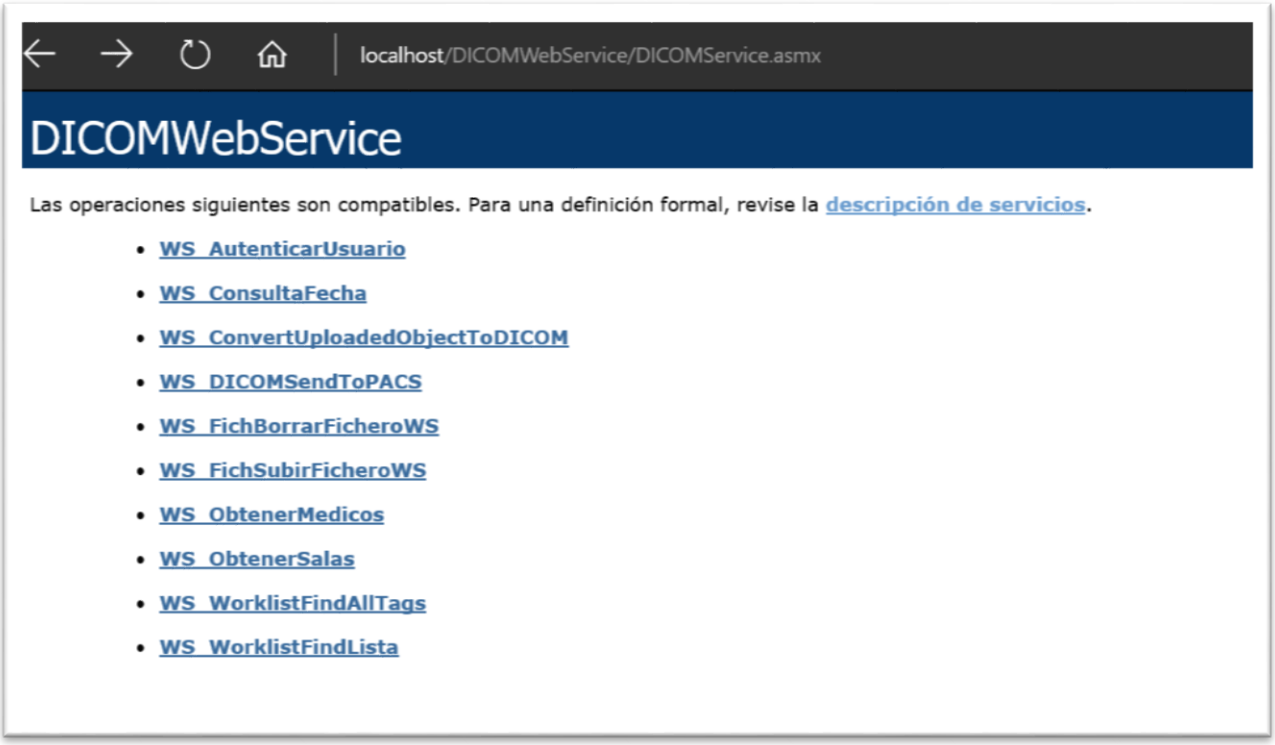


Ilustración 16 - Relación de métodos disponibles en DICOMWebService

La descripción de los métodos (WSDL – Web Service Description Language) se encuentra en:

http://<ServidorWeb>/DICOMWebService/DICOMWebService.asmx?WSDL

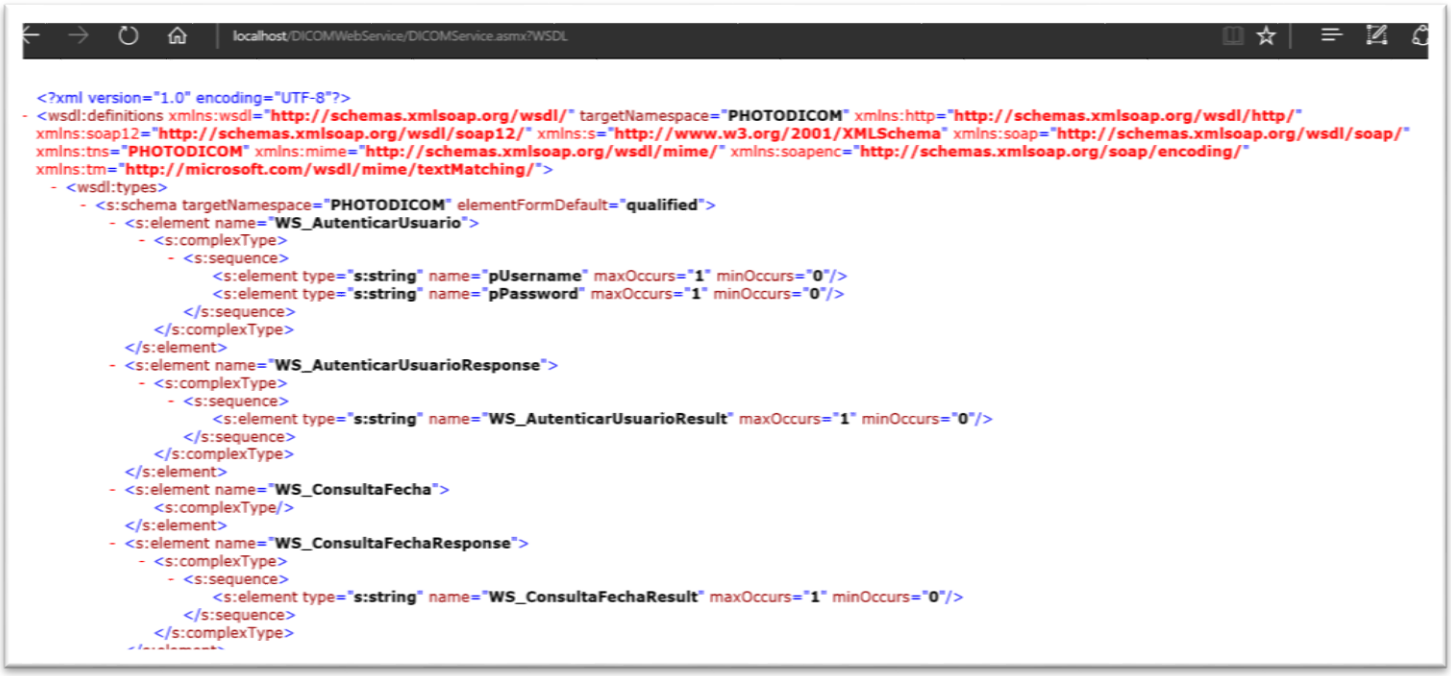


Ilustración 17 - Captura parcial de documento WSDL de DICOMWebService

A continuación explicamos los métodos existentes en el Web Service, sus entradas, salidas y la funcionalidad desarrollada por cada uno de ellos.

5.3.1. METODO WS_CONSULTAFECHA

PARAMETROS DE ENTRADA		
Ninguno		
SALIDA		
String	Fecha del servidor en formato yyyyMMddHHmmss	
DESCRIPCION		
Único método sin encriptar. Es llamado al inicio de la aplicación Android para averiguar la fecha del servidor y poder calcular la diferencia con respecto al dispositivo Android. Esa diferencia se utiliza posteriormente en los métodos de cifrado/descifrado que se relatan en el punto 5.5 de este documento.		

5.3.2. METODO WS_AUTENTICARUSUARIO

PARAMETROS DE ENTRADA		
pUserName	String	Usuario que se conecta a la app. Dato cifrado.
pPassword	String	Password introducida en la app. Dato cifrado.
SALIDA		
String	• OK|NombreUsuario (información cifrada), o bien, • ERROR: <Descripción del error> (información cifrada)	
DESCRIPCION		
Autentica al usuario de la app contra la lista de usuarios autorizados existente en el web.config del Web Service.		

5.3.3. METODO WS_OBTENERSALAS

PARAMETROS DE ENTRADA		
pOrigen	String	Origen de la llamada. (Dato cifrado).
SALIDA		
Array de tipo “ItemSala”	ItemSala es una estructura de datos compuesta por la siguiente información: • MsgError: Posible mensaje de error o “OK” en caso contrario. • NumResults: Número de resultados devueltos. • IdSala: Identificador de la sala/agenda. • Descripcion: Descripción de la sala/agenda. • SrvWkl: Identificador del servidor de Worklist contra el que está configurada esta sala/agenda. • SrvStore: Identificador del servidor de Store contra el que está configurada esta sala/agenda. • SeriePorImagen: Dato boolean que indica si se crea una nueva serie por cada imagen enviada o bien todas las imágenes se incluyen en una única serie. • Activa: Dato boolean que indica si la sala está activa.	
DESCRIPCION		

El parámetro de entrada no se utiliza actualmente. Se reserva por si en el futuro hay que hacer alguna variación según la aplicación que está conectándose al Web Service. Devuelve un array de tipo ItemSala (tal como se ha descrito) con los datos cifrados, que a su vez se recupera de los listados de salas/agendas incluidos en el web.config del Servicio Web.

5.3.4. METODO WS_OBTENERMEDICOS

PARAMETROS DE ENTRADA		
pOrigen	String	Origen de la llamada. (Dato cifrado).
SALIDA		
Array de tipo String	Devuelve el listado de facultativos responsables para poblar los "combos" (spinner) correspondientes de la app Android.	
DESCRIPCION		
El parámetro de entrada no se utiliza actualmente. Se reserva por si en el futuro hay que hacer alguna variación según la aplicación que está conectándose al Web Service. Devuelve un array de tipo String con los nombres de los médicos responsables de las pruebas efectuadas con la aplicación Android. Dicho listado se recupera del web.config del Servicio Web.		

5.3.5. METODO WS_WORKLISTFINDLISTA

PARAMETROS DE ENTRADA		
pIdSala	String	Identificador de la sala/agenda que se consulta (Dato cifrado)
pOrigen	String	Origen de la llamada. (Dato cifrado).
SALIDA		
Array de tipo "ItemLista"	ItemLista es una estructura de datos compuesta por la siguiente información: • MsgError: Posible mensaje de error o "OK" en caso contrario. • NumResults: Número de resultados devueltos. • AccessionNumber: Número de acceso de la prueba. • StudyUID: Study Instance UID de la prueba citada. • IdPaciente: Identificador del paciente (generalmente el Número de Historia Clínica). • NombrePaciente: nombre completo del paciente. • FechaNacPaciente: fecha de nacimiento del paciente. • SexoPaciente: sexo del paciente. • IdExamen:; Identificador de la prueba a realizar. • Examen: Descripción de la prueba a realizar. • FechaExamen: Fecha de la cita. • HoraExamen: Hora de la cita. • Modalidad: Tipo de modalidad DICOM de la prueba a realizar (generalmente "OT" – "Other"). • AETITLE: AE-Title de la sala/agenda (información de protocolo DICOM).	

	• MedicoPet: Médico solicitante de la prueba.
DESCRIPCION	
El parámetro de entrada no se utiliza actualmente. Se reserva por si en el futuro hay que hacer alguna variación según la aplicación que está conectándose al Web Service. Con el parámetro IdSala recuperamos la información necesaria del listado de datos del web.config para dicha sala/agenda relativa a la conexión DICOM a efectuar. Se interroga al sistema PACS, en concreto al servicio de Worklist DICOM a través de la librería DICOMObjects. Se recupera el listado de pacientes para el AE-TITLE asociado a la sala/agenda. Se devuelve dicha información, cifrada, a través de un array de ítems de tipo "*ItemLista*" tal como se ha indicado.	

5.3.6. METODO WS_WORKLISTFINDALLTAGS

PARAMETROS DE ENTRADA		
pIdSala	String	Identificador de la sala/agenda que se consulta (Dato cifrado)
pOrigen	String	Origen de la llamada. (Dato cifrado).
SALIDA		
Array de tipo "DatosWorkListTags"	DatosWorklistTags es una estructura de datos compuesta por la siguiente información: • MsgError: Posible mensaje de error o "OK" en caso contrario. • NumResults: Número de resultados devueltos. • Datos: array de tipo ItemPatientWorklist ItemPatientWorklist es una estrucuta que se compone de: • TagWorklist: ítem de tipo ItemTagWorklist. ItemTagWorklist es una estructura que se compone de: • TagDicom: código del Tag Dicom devuelto. • DescripDicom: descripción del Tag Dicom devuelto. • Value: Valor del Tag Diccom.	
DESCRIPCION		
El parámetro de entrada no se utiliza actualmente. Se reserva por si en el futuro hay que hacer alguna variación según la aplicación que está conectándose al Web Service. Con el parámetro IdSala recuperamos la información necesaria del listado de datos del web.config para dicha sala/agenda relativa a la conexión DICOM a efectuar. Se interroga al sistema PACS, en concreto al servicio de Worklist DICOM a través de la librería DICOMObjects. Se recupera el listado de pacientes para el AE-TITLE asociado a la sala/agenda. Se devuelve dicha información, cifrada, a través de un array de ítems de tipo "*ItemLista*" tal como se ha indicado. A diferencia de WS_WorklistFindLista en este método se devuelven todos los TAGS DICOM devueltos por el servicio de Worklist DICOM del PACS. En WorklistFindLista se devuelven sólo los datos concretos que son de nuestro interés en una estructura concreta. *Este método no es utilizado por la aplicación Android a fecha de hoy.*		

5.3.7. METODO WS_FICHSUBIRFICHEROWS

PARAMETROS DE ENTRADA		
pIdSala	String	Identificador de la sala/agenda a la que pertenece el fichero subido (Dato cifrado)
pNombreFichero	String	Nombre del fichero a subir al Web Service (Dato cifrado)
pDatos	String	Array de bytes convertido a String (Dato cifrado)
pDatosLength	Long	Tamaño del array de bytes subido
pOffset	Long	Desplazamiento dentro del fichero completo.
pOrigen	String	Origen de la llamada. (Dato cifrado).
SALIDA		
String	• "OK|<PathFicheroServidor>" en caso de éxito o en su caso, • "ERROR <Descripción>" en caso de error.	
DESCRIPCION		
Este método se utiliza para subir ficheros al Web Service desde la aplicación Android, en concreto para la subida de las fotografías seleccionadas para el paciente en curso en la app. Dado que los Web Service no implementan un método estándar para la subida de ficheros (no están diseñados para ello) la solución para poder realizar una transmisión de un fichero a un Web Service sin necesidad de implementar algún sistema de FTP paralelo, es hacerlo mediante el "troceado" del fichero original en bloques y la transmisión de éstos en sucesivas llamadas a éste método como un array de bytes transformado a String (y posteriormente cifrado). El método devuelve el Path relativo al servidor del fichero subido.		

5.3.8. METODO WS_FICHBORRARFICHEROWS

PARAMETROS DE ENTRADA		
pFichero	String	Path relativo al servidor del fichero a eliminar (Dato Cifrado).
pOrigen	String	Origen de la llamada. (Dato cifrado).
SALIDA		
Boolean	• True, si ha realizado el borrado con éxito. • False, en caso de error.	
DESCRIPCION		
Este método eliminar del servidor un fichero ya subido a éste (que se recibe como parámetro). *Este método no es utilizado por la aplicación Android a fecha de hoy.*		

5.3.9. METODO WS_CONVERTUPLOADEDOBJECTTODICOM

PARAMETROS DE ENTRADA		
pPathFichero	String	Path relativo al servidor del fichero de imagen a ser convertido a DICOM.
pNombrePaciente	String	Nombre del paciente al que pertenece la imagen a convertir a DICOM.
pIdPaciente	String	Identificador del paciente.

pFechaNacPac	String	Fecha de nacimiento del paciente.
pSexoPac	String	Sexo del paciente
pStudyInstanceUID	String	Study Instance UID. Identificador único del estudio al que pertenece la imagen.
pFechaEstudio	String	Fecha de realización del estudio.
pHoraEstudio	String	Hora de realización del estudio.
pStudyID	String	Identificador de la prueba realizada.
pNombreEstudio	String	Descripción del estudio realizado.
pAccessionNumber	String	Número de acceso relativo al estudio.
pNumeroImagen	Integer	Número secuencial de la imagen dentro del estudio
pIdSala	String	Identificador de la sala/agenda a la que pertenece el estudio.
pMedico	String	Médico responsable de la prueba realizada.
pTipoObjeto	String	Tipo de objeto a convertir a DICOM (BMP, EXIF, GIF, JPG, PNG, TIFF, PDF, AVI, MPEG2, DOC, DOCX, TXT, RTF, HTML, XML)[10]
pVersion	String	Versión de la aplicación cliente del Web Service.
pOrigen	String	Origen de la llamada. (Dato cifrado).
SALIDA		
String	• "OK|<PathFicheroServidor>" en caso de éxito o en su caso, • "ERROR <Descripción>" en caso de error.	
DESCRIPCION		
Este método es invocado posteriormente a la subida de un fichero al servidor. A partir del fichero subido (cuyo path relativo al servidor se incluye como parámetro), se incorporan los datos recibidos como parámetros a la imagen y se genera un objeto DICOM gracias a las librerías DICOMObjects. En caso de éxito se devuelve el path relativo al objeto DICOM generado en el servidor para su posterior envío al PACS. El nombre del fichero es el Instance UID generado para la imagen con extensión .dcm La información se transmite cifrada.		

5.3.10. METODO WS_DICOMSENDTOPACS

PARAMETROS DE ENTRADA		
pInstanceUID	String	Identificador único de la imagen a enviar al PACS.
pPathFicheroDICOMServidor	String	Path al fichero DICOM relativo al servidor.
pTipo	String	Tipo de objeto DICOM (BMP, EXIF, GIF, JPG, PNG, TIFF, PDF, AVI, MPEG2, DOC, DOCX, TXT, RTF, HTML, XML)
pIdSala	String	Identificador de la sala/agenda a la que pertenece el estudio.
pOrigen	String	Origen de la llamada. (Dato cifrado).

[10] Los formatos BMP, EXIF, GIF, JPG, PNG y TIFF son nativos en DICOM y por tanto se convierten a formato DICOM de form directa.
Los formatos AVI y MPEG2 se convierten a objeto multiframe DICOM (vídeo). << Aún no implementado >>
Los formatos DOC, DOCX, TXT, RTF, PDF, HTML y XML se convierten a DICOM a través de un objeto DICOM Encapsulated. Para ello se convierten todos los formatos a PDF para ser posteriormente incrustados en el objeto DICOM.

SALIDA	
String	• "OK" en caso de éxito o en su caso, • "ERROR <Descripción>" en caso de error.
DESCRIPCION	
Este método es invocado posteriormente a la conversión de un fichero ya subido al servidor a formato DICOM. A partir del fichero convertido a DICOM (cuyo path relativo al servidor se incluye como parámetro), y del IdSala que se nos facilita para obtener de los listados de datos del web.config la configuración de envío y conexión a su respectivo PACS para esa agenda/sala, se transmite el fichero vía protocolo DICOM al servidor PACS en cuestión, gracias a las librerías DICOMObjects. La información entre app y WebService se transmite cifrada.	

5.4. DESARROLLO APLICACIÓN ANDROID

En este epígrafe comentaré como se ha desarrollado la aplicación Android, que *actividades* y *fragmentos* la componen y las particularidades de éstos.

Como recordatorio vuelvo a mencionar que el esquema de funcionamiento de la aplicación a nivel de actividades es:

SplashActivity → LoginActivity → MainActivity

(Punto de entrada al sistema) (Autenticación) (Actividad principal)

El resto de elementos que componen la app Photo DICOM son fragmentos que se cargan en el Main Activity.

5.4.1. ACTIVITY: SPLASHACTIVITY

Supone la entrada al sistema.

A nivel de interfaz: carga el Logo de la aplicación que aparece en movimiento desde abajo hasta fijarse en el centro de la pantalla.

No requiere ninguna interacción con el usuario ya que al cabo de 6 segundos se pasa automáticamente al LoginActivity.

A nivel de procesos en *background* esta actividad es la encargada de cargar, solicitándosela al Web Service, las listas de salas/agendas disponibles y médicos responsables que son guardadas en variables "globales" a la app.

Ilustración 18 - Captura de pantalla Splash

Ejemplo de conexión al Web Service para la consulta de Médicos responsables:

```
//LLAMADA A LA TAREA ASINCRONA QUE RECUPERA DEL WS LA LISTA DE MEDICOS.
public class AsyncCallWSMedicos extends AsyncTask<String, Void, Void> {
    @Override
    protected Void doInBackground(String... params) {
        try {
            FuncionesGlobales FG = new FuncionesGlobales(getApplicationContext());
            resultMedicos = FG.WS_ObtenerMedicos("SplashActivity.java", (MiAplicacion) getApplication());
        } catch (Exception e) {
        }

        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        Encriptacion crypt = new Encriptacion();
        int i;
        int NumResults = resultMedicos.length;
         for (i=0; i< NumResults; i++)
        {
            resultMedicos[i] = crypt.Descifrar(resultMedicos[i],(MiAplicacion) getApplication());
        }
        tenemosMedicos = true;
    }

    @Override
    protected void onPreExecute() {
    }
}
```

La mostrada tarea asíncrona se apoya en una clase denominada Funciones Globales que tiene diversos métodos útiles para la conexión con el Web Service. Muestro a continuación el relacionado con la consulta de médicos responsables:

```
//LLAMADA AL METODO PARA OBTENER LOS MEDICOS QUE SE HARAN RESPONSABLES DE LA PRUEBA REALIZADA
//(PARA RELLENAR UN SPINNER).
//DEVUELVE UNA LISTA DE STRINGS.
public String[] WS_ObtenerMedicos(String pOrigen, MiAplicacion pAplicacion)
{
    String[] arrResult;
    SoapObject request = new SoapObject(pAplicacion.getNAMESPACE(), pAplicacion.getMETHOD("Medicos") );
    request.addProperty("pOrigen", pOrigen);
    int indice;
    String literalDesconocido  = "Desconocido.";

    SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
    envelope.dotNet = true;
    envelope.setOutputSoapObject(request);

    HttpTransportSE transporte = new HttpTransportSE(pAplicacion.getURL());

    try
    {
        transporte.call(pAplicacion.getSOAP_ACTION("Medicos"), envelope);
        SoapObject objResultado =(SoapObject)envelope.getResponse();

        if (objResultado == null)
        {
            return  new String[]{literalDesconocido};
        }

        String[] listaMedicos = new String[objResultado.getPropertyCount()];

        for (indice = 0; indice < listaMedicos.length; indice++)
        {
            SoapPrimitive imed = (SoapPrimitive)objResultado.getProperty(indice);
            String itemMedico;
            itemMedico = imed.toString();

            if (itemMedico.equals("anyType{}")) {itemMedico = literalDesconocido;}
            listaMedicos[indice] = itemMedico;
            itemMedico = null;
        }
        return listaMedicos;
    }
    catch (Exception e)  { return  new String[]{literalDesconocido};  }
}
```

SplashActivity también es el encargado de testear la conexión al Web Service (dirección en principio "*hardcodeada*", es decir, está fija en el código ya la implantación se hará cliente a cliente).

En caso de que no responda el servidor, el mensaje de error se encarga de mostrarlo la siguiente actividad (LoginActivity) ya que dicha comprobación se realiza también con una tarea asíncrona.

5.4.2. ACTIVITY: LOGINACTIVITY

Supone la autenticación del usuario en el sistema. Se carga automáticamente después de que SplashActivity realice las acciones necesarias.

Ilustración 19 - Login Activiy, autenticación en el sistema

La autenticación del usuario se hace nuevamente invocando un método del WS, en concreto el WS_AutenticarUsuario que nos devolverá un "OK" o en mensaje de error correspondiente para ser mostrado en la interfaz.

Si la autenticación es correcta cargamos finalmente el MainActivity, actividad principal de la app.

5.4.3. ACTIVITY: MAINACTIVITY

Esta es la actividad principal.

Se realiza la conexión a la BD o se crea ésta en caso de ser la primera ejecución:

```
dataSource = new HistoricoDatosDataSource(this);

SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
if (settings.getBoolean("primera_ejecucion", true)) {
    //La aplicación se ha lanzado por primera vez. Creo las tablas en la BD.
    Log.v(TAG, "Primera ejecución.");

    //Si es la primera vez que se entra en la aplicación se crean las tablas.
    //Si no, no hace nada, (contiene un if not exist tabla)
    dataSource.CrearTablas();

    //Actualizamos las preferencias.
    settings.edit().putBoolean("primera_ejecucion", false).commit();
}

dataSource.EliminarPorAntiguedad();
//dataSource.EliminacionCompleta();
```

Esta actividad es la que contendrá el resto de “pantallas” de la aplicación, que no serán actividades sino fragmentos que se irán reemplazando o superponiendo. Por tanto cuando desde un *fragment* hacemos referencia a la actividad padre, ésta siempre es la MainActivity.

Asimismo y como se aprecia en la captura de código, también se encarga de realizar la “limpieza” de base de datos, borrando los registros más antiguos (actualmente establecido a 90 días).

Esta actividad al ser la principal es la que implementa el menú de la aplicación, en estilo *Material Design* de Android.

Ilustración 20 - Main Activity - Menú de la aplicación (flotante)

En una primera carga el fragmento cargado (y considerado en cierta forma como "principal") es el de Lista de trabajo que veremos a continuación.

5.4.4. FRAGMENT: WORKLISTFRAGMENT

Se llama al entrar al MainActivity por primera vez y desde la correspondiente opción del menú.

Muestra un desplegable (*spinner*) con la lista de Agendas/Salas (recuperadas desde el WebService durante la carga de la actividad Splash).

Su modificación o hacer click en el botón de refresco provocan la interrogación de Worklist al WebService (y éste a su vez por DICOM al PACS). Los resultados son mostrados en una lista personalizada como podemos ver en la siguiente captura:

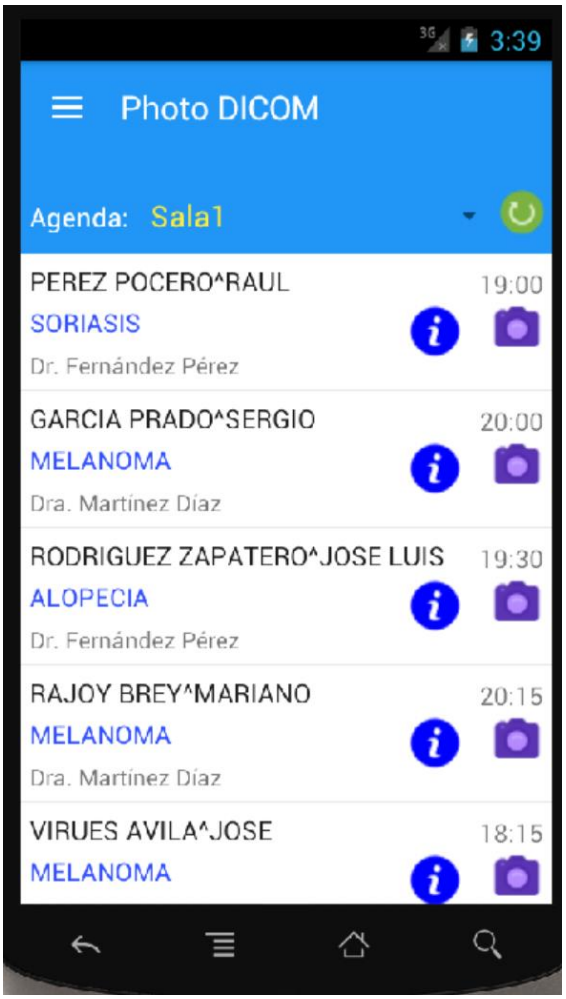


Ilustración 21 - Lista de Trabajo (fragmento).

Para poder plasmar este tipo de lista, se ha tenido que desarrollar un adaptador personalizado (*WorkListAdapter*) que es el encargo de analizar la respuesta del WS, poblar la lista y responder a los eventos (click) en los botones incluidos en cada fila de la lista.

Durante la consulta de la Worklist se guarda en un array toda la información proporcionada por la Worklist DICOM (a través del WS) aunque no se muestre en el *grid*, así al consultar la información extendida de un ítem, no es necesario reconsultar al WS ni a la Worklist DICOM.

5.4.5. FRAGMENT: INFOWLFRAGMENT

Se accede a este fragmento desde el botón “info” de una fila de la lista de trabajo (*WorklistFragment*). Muestra la información extendida de un ítem, que a su vez se ha recuperado del WS (y éste de la Worklist DICOM).

Ilustración 22 - Información extendida de una exploración (captura parcial)

La información está disponible en un Array en *WorklistFragment*. Al acceder a esta pantalla se guarda en "variables globales" que son accedidas luego desde el evento de creación del fragmento.

En *WorkListAdapter:*

```
//BOTON VER INFORMACION EXTENDIDA DE UN ITEM DE LA LISTA (SE CARGA OTRO FRAGMENT).
holder.btnVer.setOnClickListener((v) -> {

        Activity miactivity = (Activity) context;
        ((MiAplicacion) miactivity.getApplication()).setNOMBREPACIENTE(itemworklist.getNombrePaciente());
        ((MiAplicacion) miactivity.getApplication()).setPRUEBA(itemworklist.getPrueba());
        ((MiAplicacion) miactivity.getApplication()).setHORA(itemworklist.getHora());
        ((MiAplicacion) miactivity.getApplication()).setNUMHIST(itemworklist.getNhc());
        ((MiAplicacion) miactivity.getApplication()).setCODPRUEBA(itemworklist.getCodPrueba());
        ((MiAplicacion) miactivity.getApplication()).setFECHANAC(itemworklist.getFechanac());
        ((MiAplicacion) miactivity.getApplication()).setMEDICOSOL(itemworklist.getMedico());
        ((MiAplicacion) miactivity.getApplication()).setACCN(itemworklist.getAccn());
        ((MiAplicacion) miactivity.getApplication()).setSUID(itemworklist.getSUID());
        ((MiAplicacion) miactivity.getApplication()).setSEXO(itemworklist.getSEXO());

        FragmentManager fm = miactivity.getFragmentManager();
        FragmentTransaction ft = fm.beginTransaction();
        Fragment fragmentPrevio;
        Fragment fragmentSiguiente;

        //Por si estuviese abierto el fragment de captura de fotos, lo cerramos:
        fragmentPrevio = fm.findFragmentByTag(CaptureFragment.TAG);
        if (fragmentPrevio != null) { ft.detach(fragmentPrevio); ft.remove(fragmentPrevio);}

        fragmentPrevio = fm.findFragmentByTag(WorklistFragment.TAG);
        fragmentSiguiente = new InfoWLFragment();
        ft.hide(fragmentPrevio);
        ft.add(R.id.container, fragmentSiguiente, InfoWLFragment.TAG);
        ft.commit();

});
```

En *InfoWLFragment:*

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    // Inflate the layout for this fragment
    final View vlView = inflater.inflate(R.layout.fragment_infowl, container, false);

    botonVolver = (Button) vlView.findViewById(R.id.btnVolver);
    botonFotos = (Button) vlView.findViewById(R.id.btnFotos);
    NombrePaciente = (TextView) vlView.findViewById(R.id.txtNombrePaciente);
    Prueba = (TextView) vlView.findViewById(R.id.txtPrueba);
    Hora = (TextView) vlView.findViewById(R.id.txtHora);
    NumHist = (TextView) vlView.findViewById(R.id.txtNumHist);
    CodPrueba = (TextView) vlView.findViewById(R.id.txtCodPrueba);
    FechaNac = (TextView) vlView.findViewById(R.id.txtFechaNac);
    MedicoSol = (TextView) vlView.findViewById(R.id.txtMedicoSol);
    Accn = (TextView) vlView.findViewById(R.id.txtAccn);
    SUID = (TextView) vlView.findViewById(R.id.txtSUID);

    NombrePaciente.setText(((MiAplicacion) this.getActivity().getApplication()).getNOMBREPACIENTE());
    Prueba.setText(((MiAplicacion) this.getActivity().getApplication()).getPRUEBA());
    Hora.setText("Hora de la cita:" + " " + ((MiAplicacion) this.getActivity().getApplication()).getHORA());
    NumHist.setText("Número de Historia:" + " " + ((MiAplicacion) this.getActivity().getApplication()).getNUMHIST());
    CodPrueba.setText("Código Prueba Citada:" + " " + ((MiAplicacion) this.getActivity().getApplication()).getCODPRUEBA());
    FechaNac.setText("Fecha de Nacimiento:" + " " + ((MiAplicacion) this.getActivity().getApplication()).getFECHANAC());
    MedicoSol.setText("Médico Solicitante:" + " " + ((MiAplicacion) this.getActivity().getApplication()).getMEDICOSOL());
    Accn.setText("Accession Number:" + " " + ((MiAplicacion) this.getActivity().getApplication()).getACCN());
    SUID.setText(((MiAplicacion) this.getActivity().getApplication()).getSUID());
```

5.4.6. FRAGMENT: CAPTUREFRAGMENT

Se accede a este fragmento desde el botón “photo” de una fila de la lista de trabajo o desde la info extendida de un ítem (*InfoWLFragment*).

Permite seleccionar imágenes desde la galería o llamar a la aplicación por defecto para realizar una fotografía nueva. El desplegable (*spinner*) de médicos responsables se rellena con la lista recuperada durante la actividad Splash.

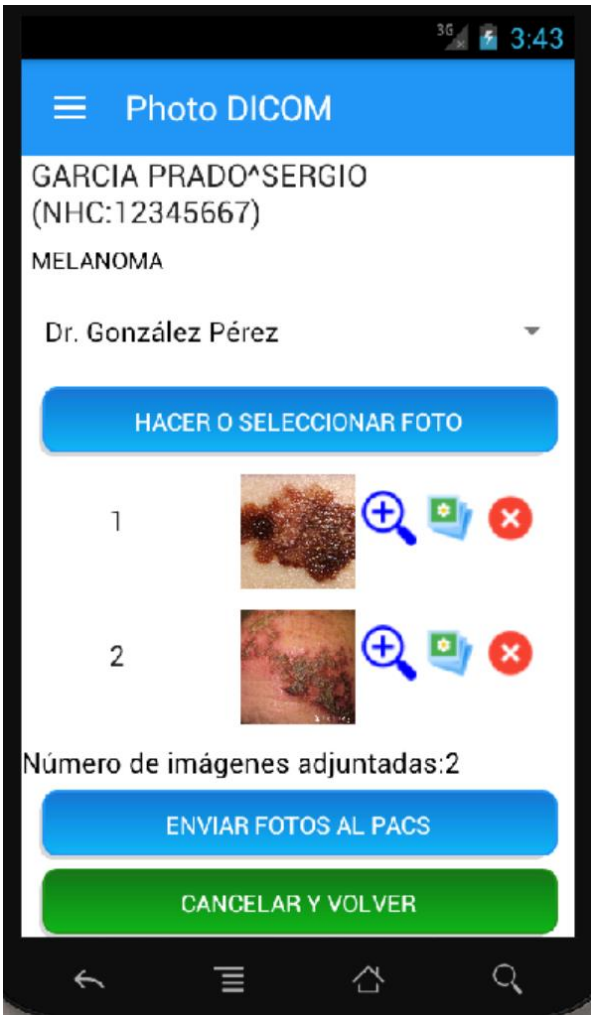


Ilustración 23 - Captura/selección de fotografías, subida al servidor y envío al PACS

Una vez realizadas/seleccionadas todas las fotografías, éstas se suben desde este fragmento al Servidor (WS), se convierten a DICOM en el Servidor (WS) y se envían al PACS configurado para la sala/agenda en curso (también a través del WS).

Ante un proceso exitoso se guarda la información en BD local.

```
// FUNCION PARA GUARDAR LOS DATOS EN BD LOCAL (SQLITE) UNA VEZ UNA TAREA DE SUBIDA,
// CONVERSION A DICOM Y ENVIO A PACS SE HA COMPLETADO CON EXITO. INSERTA O ACTUALIZA.
//
public void GuardarDatos() {
    //GUARDATOS LOS DATOS DE LAS IMAGENES SUBIDAS / PACIENTE REALIZADO A PERSISTENCIA LOCAL (SQLite)
    ItemListaImagenes vlItemActual;
    bd = new HistoricoDatosDataSource(this.getActivity());
    Long vlIdHist = Long.parseLong("-1");
    SimpleDateFormat iso8601Format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String currentDateandTime = iso8601Format.format(new Date());
    SimpleDateFormat iso8601Format_2 = new SimpleDateFormat("yyyy-MM-dd");
    String currentDate = iso8601Format_2.format(new Date());

    String textoMedico = Medicos.getSelectedItem().toString();

    //INSERTAMOS EL MAESTRO (ESTUDIO)
    cursor = bd.ConsultaExisteDatoHistoricoPorSUID(pStudyInstanceUID);
    if (cursor != null) {
        cursor.moveToFirst();                       // Un select count siempre devuelve algo.
        if (cursor.getInt(0) == 0) {                // Resultado 0 significa registro no existe. Insertamos.
            vlIdHist = bd.InsertarHistorico(((MiAplicacion) this.getActivity().getApplication()).getNOMBREPACIENTE(),
                    ((MiAplicacion) this.getActivity().getApplication()).getNUMHIST(),
                    ((MiAplicacion) this.getActivity().getApplication()).getFECHANAC(),
                    ((MiAplicacion) this.getActivity().getApplication()).getCODPRUEBA(),
                    ((MiAplicacion) this.getActivity().getApplication()).getPRUEBA(),
                    currentDate + " " + ((MiAplicacion) this.getActivity().getApplication()).getHORA(),
                    currentDateandTime,
                    ((MiAplicacion) this.getActivity().getApplication()).getIDSALA(),
                    ((MiAplicacion) this.getActivity().getApplication()).getIDSALADESCRIP(),
                    ((MiAplicacion) this.getActivity().getApplication()).getSUID(),
                    ((MiAplicacion) this.getActivity().getApplication()).getACCN(),
                    ((MiAplicacion) this.getActivity().getApplication()).getMEDICOSOL(),
                    textoMedico,
                    listaArray.size());

            cursor.close();

            cursor = bd.ConsultaIdDatoHistoricoPorSUID(pStudyInstanceUID);
            if (cursor != null)
            {
                cursor.moveToFirst();
                vlIdHist = cursor.getLong(0);
            }
        } else {                // Existe el registro, lo actualizamos.
```

Si el registro ya existiese (opción muy improbable) se actualizaría en BD *SQLite.*

En este fragmento, tenemos además la posibilidad de eliminar una foto de la lista, verla en pantalla completa o verla con la aplicación de galería del teléfono móvil.

5.4.7. FRAGMENT: HISTORYFRAGMENT

Este fragmento es el principal del área de datos de persistencia local. Es el equivalente a *WorklistFragment* con la diferencia de que la información no se consulta a un WebService sino que se extrae de la BD local *SQLite*.

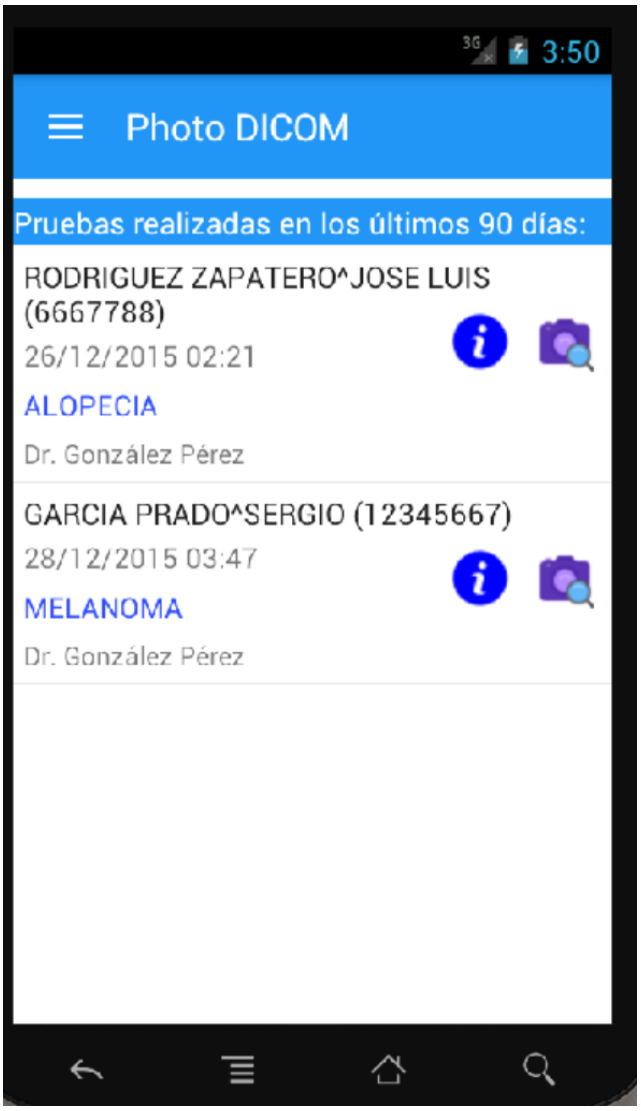


Ilustración 24 - Actividad Histórica realizada con Photo DICOM

Para poblar esta lista, al igual que en el caso de *WorklistFragment*, se ha tenido que crear un adaptador específico que es el que consulta la información de la base de datos local.

```
public void bindView(View pView, Context pContext, Cursor pCursor) {
    TextView idhist = (TextView)pView.findViewById(R.id.his_idhist);
    TextView pac_nombre = (TextView)pView.findViewById(R.id.his_pac_nombre);
    TextView prueba_desc = (TextView)pView.findViewById(R.id.his_prueba_desc);
    TextView fecha_ejec = (TextView)pView.findViewById(R.id.his_fecha_ejec);
    TextView medico_ejec = (TextView)pView.findViewById(R.id.his_medico_ejec);
    TextView imagenes_num = (TextView)pView.findViewById(R.id.his_imagenes_num);

    final HistoryHolder holder = new HistoryHolder();

    LayoutInflater inflater = ((Activity) context).getLayoutInflater();
    holder.strID = pCursor.getString(pCursor.getColumnIndexOrThrow("_id"));
    holder.strNombrePaciente = pCursor.getString(pCursor.getColumnIndexOrThrow("nombrepac"));
    holder.strPrueba = pCursor.getString(pCursor.getColumnIndexOrThrow("descprueba"));
    holder.strCodPrueba = pCursor.getString(pCursor.getColumnIndexOrThrow("codprueba"));
    holder.strFCita = pCursor.getString(pCursor.getColumnIndexOrThrow("fcita"));
    holder.strFEjec = pCursor.getString(pCursor.getColumnIndexOrThrow("fejec"));
    holder.strNHC = pCursor.getString(pCursor.getColumnIndexOrThrow("nhc"));
    holder.strFNac = pCursor.getString(pCursor.getColumnIndexOrThrow("fnac"));
    holder.strMedicoSol = pCursor.getString(pCursor.getColumnIndexOrThrow("msolic"));
    holder.strMedicoEjec = pCursor.getString(pCursor.getColumnIndexOrThrow("mejec"));
    holder.strAccN = pCursor.getString(pCursor.getColumnIndexOrThrow("accn"));
    holder.strSUID = pCursor.getString(pCursor.getColumnIndexOrThrow("suid"));
    holder.strAgenda = pCursor.getString(pCursor.getColumnIndexOrThrow("descagenda"));
    holder.strNumImag = Integer.toString(pCursor.getInt(pCursor.getColumnIndexOrThrow("numimag")));

    idhist.setText(holder.strID);
    pac_nombre.setText(holder.strNombrePaciente + " (" + holder.strNHC + ")");
    prueba_desc.setText(holder.strPrueba);
```

Desde este fragmento se puede acceder de forma análoga a la información ampliada de un ítem histórico y a las fotos realizadas/adjuntadas a dicha exploración.

5.4.8. FRAGMENT: INFOHISFRAGMENT

Este fragmento es similar a *InfoWLFragment* con dos diferencias: muestra información de BD local y exhibe más información que el citado, ya que incorpora información sobre datos de realización de la prueba y no sólo de datos de citación.

Ilustración 25 - Información extendida de exploración histórica (captura parcial)

El método de recuperación de la información es similar al caso de *InfoWLFragment*: el fragmento listado (en este caso *HistoryFragment*) "graba" la información en "variables globlales" y ésta es recuperada en el evento *onCreateView* de este fragmento.

5.4.9. FRAGMENT: VIEWIMAGESFRAGMENT

Por último mostramos este fragmento, que es llamado desde la lista de actividad histórica y desde la información extendida de una exploración histórica. Se encarga de mostrar las imágenes "adjuntadas" a la exploración (en caso de que aún estén en el dispositivo, o un gif predeterminado en caso de haber sido borradas).

Permite asimismo la visualización a pantalla completa y mediante la aplicación galería del teléfono.

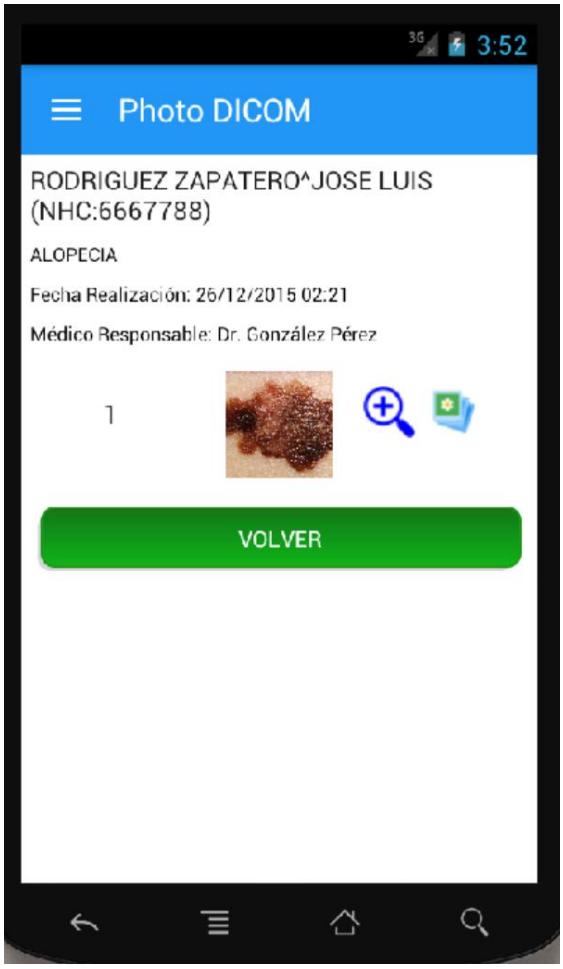


Ilustración 26 - Visualización de imágenes adjuntadas a exploración realizada

5.5. IMPLEMENTACION DE LA SEGURIDAD

Debido a que la solución está orientada a trabajar con datos médicos, es evidente que necesita seguridad en las comunicaciones (la información médica es la información de más alto nivel según establece la LOPD).

Para dotar de seguridad a la plataforma se barajaron varias opciones:

- La más sencilla y más típica es que la seguridad esté en la capa de comunicación, realizando ésta sobre https. Esta solución no comporta ninguna dificultad añadida más allá de la instalación el servidor Web del correspondiente certificado SSL.

  - El problema de esta solución es que obliga a tener un certificado para poder generar el HTTPS sobre éste, y eso tiene un coste económico (no trivial) y una complejidad a la hora de instalar la app y en especial el servidor en un entorno real de trabajo (en un Hospital, por ejemplo), encareciendo el producto.
- Por tanto se pensó en llevar la seguridad al nivel de aplicación: la comunicación puede ser no segura pero entonces es la información es la que es segura y es la solución finalmente implementada.

El mecanismo seguido para cifrar y descifrar la información es el siguiente:

- El cliente (app) recupera del WebService la fecha y hora del servidor.
- A partir de la fecha y hora del servidor calcula la diferencia de la su propia hora con la del servidor en segundos.
- A partir de ahí: Cada parámetro de cada método del WebService va cifrado. El contenido de la cadena cifrada es: yyyyMMddHHmmss|Valor donde la correspondiente fecha ya se ha ajustado sumándole la diferencia en segundos (que puede ser positiva o negativa) para trabajar con "fecha servidor".
- Se cifra dicha información con algoritmo AES/CBC/PK5Padding con clave y vector de inicialización compartido entre cliente y servidor. *(Esto no ha sido ni mucho menos trivial para conseguir encriptar en Java y desencriptar en .NET y viceversa).*
- En cada método dentro del WebService se descifra el contenido de cada parámetro. En el proceso de descifrado se calcula si la fecha recibida varía en x segundos (por ejemplo 60) de la fecha del servidor. Si varía más: Error (un 'hacker' ha *reutilizado* un mensaje antiguo). Si varía menos: continuamos, extrayendo el valor del dato cifrado recibido.
- El WebService efectúa la operación solicitada y devuelve el resultado cifrado de la misma forma, incluyendo su fecha.
- El cliente recibe la respuesta cifrada, la descifra, calcula si su fecha + los segundos de diferencia es aproximadamente igual a la del servidor y si OK, recoge el valor devuelto y opera con él.

Ventajas:

- La comunicación es tan segura como trabajar con HTTPS.
- No necesitamos HTTPS (certificado de seguridad, que tiene un coste económico.
- No se pueden reutilizar los mensajes SOAP después de x segundos por que el contenido caduca.

Inconvenientes:

- Complica el código, hay que cifrar y descifrar en cada operación, aunque con las operaciones bien encapsuladas en diversas funiones o helpers, esto es muy asumible.

- Tiene un coste computacional tanto en servidor como en cliente (dispositivo Android), no obstante este coste es muy bajo y no afecta al rendimiento.

5.6. PRUEBAS

Como en todo proyecto de desarrollo de software que se precie es necesario establecer e implementar una serie de pruebas que garanticen el correcto funcionamiento de la solución.

Las pruebas realizadas se enmarcan en la categoría de pruebas de integración, comprobándose que todos los componentes funcionan correctamente, que actúan en conjunto y que son dependientes de la plataforma o entorno utilizado para su ejecución.

También se han realizado una serie de pruebas funcionales en las que se comprueba que los distintos elementos software cumplen con sus respectivas funciones, observando entradas y salidas y sin profundizar en elementos de diseño o arquitectónicos del software.

En todo caso todas las pruebas se han realizado en un proceso manual iterativo.

No se han realizado pruebas de carga o de estrés por la complejidad que presenta el hacer una prueba de carga sobre el WebService con múltiples clientes interactuando con el citado WebService. Dichas pruebas de carga se postponen a un proceso previo a la comercialización de la solución, pero ya en colaboración con usuarios finales y contando con los dispositivos que se pretendan utilizar ya disponibles.

5.6.1. ENTORNO DE PRUEBAS

En apartados anteriores se ha indicado el entorno de desarrollo existente. Centrándonos más en la cuestión del entorno de pruebas fijamos lo siguiente:

- El desarrollo inicial se hace sobre emulador (*Android Virtual Device*). En concreto se está utilizando un equipo antiguo, que hoy podría considerarse de gama baja, para que los requisitos de la aplicación no sean muy altos y poder abarcar un gran número de posibles dispositivos[11]. Por tanto se utiliza una AVD de Nexus One sobre API 15 (Android 4.0.2.).
- Posteriormente se lleva una segunda fase de pruebas sobre dispositivos físicos. Para ello se cuenta especialmente con un *Smartphone* BQ M5 (con versión de Android 5.0.2 y actualizado posteriormente a 5.1.1.) y con un viejo *Smartphone* Samsung Galaxi SII.

[11] Recordar el que el principal objetivo de este proyecto es el uso desde Cámaras Fotográficas Android. En estos dispositivos no podemos contar con que estén en últimas versiones del Sistema Operativo.

6. CONCLUSIONES

En los Centros Sanitarios se han venido implantando infraestructuras y soluciones tecnológicas orientadas hacia ciertos departamentos como por ejemplo el de Radiología y que no se han generalizado fuera de ellos.

Existe la necesidad de gestionar con criterios de seguridad de la información, interoperabilidad con el resto de ecosistema TI de los Centros Sanitarios y de distribución de la información de otros departamentos generadores de imagen no radiológica.

Se hace patente por tanto la necesidad de dar una solución a las carencias anteriormente citadas y lo más lógico pasa por la reutilización de las infraestructuras existentes (fundamentalmente en cuanto a Hardware y sistemas PACS).

De este planteamiento surge el presente proyecto para proponer una opción válida para el problema concreto de la captura de evidencias clínicas con cámaras fotográficas convencionales.

Dichas cámaras fotográficas permiten pocas opciones o prácticamente ninguna al ser sistemas mono-bloque y sin opción de modificación. Pero gracias a la irrupción del sistema operativo Android en el mundo de la fotografía semi-profesional se abre un abanico de opciones para interactuar con el origen de las imágenes, que es la propia cámara fotográfica.

Asimismo se dispone de herramientas y conocimientos para la transformación de imágenes *convencionales* (por ejemplo *jpeg*) a formato DICOM. Por consiguiente la solución final pasa por casar esos conocimientos y sistemas convencionales de transformación de imágenes con la interacción con las fuentes generadoras de imagen, las cámaras.

La arquitectura SOA surge de la aplicación del paradigma de orientación a servicios y con influencia de diferentes modelos como por ejemplo la orientación a objetos, BMP, la orientación a aspectos. Web Services, etc.

SOA permite casarnos con una tecnología concreta (Java, .NET, etc.), nos permite descomponer la lógica de negocio en pequeñas unidades de funcionalidad y obtener una plataforma transversal de forma que no solucionamos las necesidades cambiantes mediante la creación de nuevas aplicaciones sino mediante la combinación de diversos servicios.

No resulta fácil diseñar una arquitectura orientada a servicios. Aquí entran en juego los servicios Web basados tanto en SOAP (como en REST), como excelentes opciones para crear los servicios que conforman nuestra plataforma y reúnen las condiciones óptimas para poder diseñar servicios que cumplan con los principios de diseño alineados con SOA.

A efecto más práctico hemos visto que se pueden combinar diversas tecnologías (.NET y Java en este proyecto) siempre y cuando se sigan estándares y protocolos comunes, siendo esta la principal lección que podemos obtener de este proyecto.

Cada lenguaje de programación y cada plataforma (dispositivos, sistemas operativos, etc.) tiene sus peculiaridades y sus especificaciones, pero ninguna es mejor que otra y varias tecnologías muy diferentes entre sí pueden interoperar y funcionar conjuntamente como hemos visto.

A nivel personal la parte .NET ha sido la más sencilla, por haber desarrollado a lo largo de mi trayectoria profesional multitud de servicios web con esta tecnología. En esta parte la mayor dificultad residió en aprender a utilizar las librerías que nos abstraen de las acciones propias del protocolo DICOM, las librerías *DICOMObjects*, que nunca había utilizado hasta el momento.

De la parte Android todo ha sido nuevo, ya que se sólo se contaba con la experiencia del pequeño proyecto realizado en una de las asignaturas de este Master. Debido a los plazos tan reducidos, en especial en cuanto a la fase de codificación, el aprendizaje de la plataforma y resolución de problemas concretos ha sido un proceso intensivo y cierta medida estresante. No obstante todas las dificultades y cuestiones que se han ido planteando se han superado satisfactoriamente.

Los objetivos planteados en el proyecto se han logrado con suficiencia. La plataforma es operativa y resuelve el problema planteado, por tanto considero que hemos logrado el objetivo inicial tal como se estableció de un inicio.

La planificación inicial no fue todo lo precisa que debería haber sido. A mitad de la fase de codificación el retraso que acumulábamos era bastante importante y el motivo fue que no se valoró acertadamente lo que supondría superar la curva de aprendizaje del lenguaje Java, de la plataforma Android y de sus peculiaridades.

Para solventar ese retraso y como es bien sabido cabían tres posibilidades: aumentar el tiempo (opción no viable ya que las fechas de entrega son las que son), reducir el alcance (opción posible pero que se quería evitar salvo último recurso) o aumentar el gasto, que en este caso supone aumentar el número de horas dedicadas al proyecto. Así se hizo y se recuperó el retraso al final de la fase de codificación.

En cuanto al producto resultante de este proyecto, Photo DICOM, la solución creo que es suficientemente atractiva como para ser puesta en el mercado con muy pocos retoques. Añadiría varias funcionalidades como por ejemplo una mejor gestión de la configuración (app Android) y una correcta gestión de vídeos (Web Service), pero era algo que se quedaba fuera del alcance de este proyecto.

7. GLOSARIO

TERMINO	DESCRIPCION
AES	Advanced Encryption Standard. Esquema de cifrado por bloques.
DICOM	Digital Imaging and Communications in Medicine.
FTP	File Transfer Protocol, protocolo de transmisión de ficheros.
HL7	Health Level Seven.
HTTP	HyperText Transfer Protocolo, protocolo de transferencia hipertexto, utilizado principalmente para el acceso a contenido web desde navegadores.
IPO	Interacción Persona-Ordenador. Disciplina dedicada a estudiar la relación interactiva entre las personas y la tecnología, y a mejorar esta relación mediante el diseño.
IIS	Internet Information Server, servidor web de Microsoft.
PACS	Picture Archiving and Communication System.
PMI	Project Management Institute.
Prototipo de alta fidelidad	Modelo tan cercano como sea posible al sistema que se diseña y desarrolla. Este tipo de prototipos se utilizan para evaluar de la forma más precisa posible los aspectos funcionales y de usabilidad, tanto para un experto en usabilidad como para realizar test con usuarios. [Millan et al. 2008]
SGBD	Sistema Gestor de Bases de Datos, software especializado en las tareas de almacenamiento y recuperación de información, principalmente según el modelo relacional.
SOA	Service Oriented Architecture (Arquitectura Orientada a Servicios).
SOAP	Simple Object Access Protocol, protocolo simple de acceso a objetos, estándar de la W3C ideado para el intercambio de información entre objetos remotos utilizando XML.
SQL	Structured Query Language, lenguaje estructurado de consulta utilizado para la manipulación de información en el seno de un SGBD.
UML	Unified Model Language, lenguaje unificado de modelado de sistemas de información y componentes.
URL	Uniform Resource Locator, localizalizador uniforme de recursos o dirección única de un elemento en Internet.
WSDL	Web Service Description Language – Formato XML que se utiliza para describir Servicios Web.
XML	Extensible Markup Language, lenguaje extensible de marcas multipropósito, muy utilizado para el intercambio de información entre sistemas heterogéneos.

8. BIBLIOGRAFÍA

- ARLANDY RODRIGUEZ, M. SOA vs SOAP y REST:

http://www.adictosaltrabajo.com/tutoriales/soavs-soap-rest/

- CALVO-FERNANDEZ, S. ORTEGA, A. VALLS, M. ZAPATA 2011. Material didáctico de la UOC "Evaluación de la usabilidad". Universitat Oberta de Catalunya.

- Experiencias TI: Paradigma SOA: http://experiencias-ti.blogspot.com.es/search/label/SOA

- GARRETA DOMINGO, M., MOR PERA, E. (2011). Interacción Persona Ordenador. Diseño centrado en el usuario. UOC.

- Y. HASSAN MONTERO, FRANCISCO J. MARTÍN FERNÁNDEZ 2003. "Método de test con usuarios". (En línea en nosolousabilidad.com).

- K. INSTONE 1996. "Site usability heuristics for the Web". (En línea).

- MILLAN, A., FERMÍN, G., CHACÓN, J. (2008). Cátedra: Instrucción a la Informática. Universidad Nacional Experimental de Guayana. http://www.monografias.com/trabajos59/diagrama-flujo/diagrama-flujo.shtml

- J. NIELSEN 1994. "Enhancing the explanatory power of usability heuristics" en "Proceedings on the ACM CHI'94 Conference". (24-28 de Abril, Págs 152-158). Boston

- D. PIEROTTI 2004. "Heuristic evaluation – A system checklist". (En línea).

- WIKIPEDIA: Arquitectura Orientada a Servicios (SOA):

https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

8.1. REFERENCIAS WEB (RESOLUCION DE PROBLEMAS DE CODIFICACION)

- HARVEY DAVE (Medical Connections), Creating Secondary Capture Images:

https://www.medicalconnections.co.uk/kb/Creating_Secondary_Capture_Images

- Sistema de ayuda de DicomObjects.NET:

https://www.medicalconnections.co.uk/download/Help6/DicomObjects.NET/webframe.html#Intro.html

- KnowledgeBase de Medical Connections:

https://www.medicalconnections.co.uk/list-of-kb-articles

- GOOGLE. Introduction to Material Design:

http://www.google.com/design/spec/material-design/introduction.html

- GOOGLE. Material Design for Android:

https://developer.android.com/design/material/index.html

- GOOGLE. Material icons:

https://design.google.com/icons/

- OLIVEIRA, PEDRO. Material Design Template:

http://androidshenanigans.blogspot.pt/2015/03/material-design-template.html

- STACKOVERFLOW: Send an image from Android to an ASP.NET Web Service:

http://stackoverflow.com/questions/9404067/send-an-image-from-android-to-an-asp-net-web-service

- STACKOVERFLOW: Word Automation with ASP.NET:

http://stackoverflow.com/questions/25289335/word-automation-with-asp-net

- STACKOVERFLOW: Android global variable:

http://stackoverflow.com/questions/1944656/android-global-variable

- GOMEZ OLIVER, SALVADOR. Acceso a Servicios Web SOAP en Android (2/2):

http://www.sgoliver.net/blog/acceso-a-servicios-web-soap-en-android-22/

- KSOAP2-ANDROID PROJECT:

http://simpligility.github.io/ksoap2-android/license-information.html

- GOVENDER, SASHEN. Consuming Web Services with kSOAP:

http://code.tutsplus.com/tutorials/consuming-web-services-with-ksoap--mobile-21242

- MSDN BLOGS. Java and .NET – AES Crypto Interop:

http://blogs.msdn.com/b/dotnetinterop/archive/2005/01/24/java-and-net-aes-crypto-interop.aspx

- SRIVASTAVA, MANISH. Multi Touch List View: Multi Click List View Demo in Android.

http://www.androidhub4you.com/2013/02/muftitouch-listview-multi-click.html

- MARAVITSAS, NIKOS. Android MultiTouch ListView Example:

http://examples.javacodegeeks.com/android/core/ui/listview/android-multitouch-listview-example/

- STACKOVERFLOW. How to create a drop-down list?

http://stackoverflow.com/questions/13377361/how-to-create-a-drop-down-list

- JASANI, TEJAS. Android take photo from camera and gallery – Code Example:

http://www.theappguruz.com/blog/android-take-photo-camera-gallery-code-sample

- Códigos de colores hexadecimales:

http://html-color-codes.info/codigos-de-colores-hexadecimales/

- KANOJIA, R.K. Android Gallery View – Displaying a list of images:

http://www.androidinterview.com/android-gallery-view-example-displaying-a-list-of-images/

- AMATELLANES. Android – Notificaciones en Android (Parte 2 – Dialogs II):

https://amatellanes.wordpress.com/2013/10/06/android-notificaciones-en-android-parte-2-dialogs-ii/

- REVELO, JAMES. Tutorial de bases de datos SQLite en aplicaciones Android:

http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

- JASANI, TEJAS. Android – Image fragment pager view:

http://www.theappguruz.com/blog/android-image-fragment-pager-view

- STACKOVERFLOW. Android soft keyboard covers edittext field:

http://stackoverflow.com/questions/3295672/android-soft-keyboard-covers-edittext-field

- STACKOVERFLOW. How do I preserve the state of a selected spinner/dropdown item on orientation change?

http://stackoverflow.com/questions/20209015/how-do-i-preserve-the-state-of-a-selected-spinner-dropdown-item-on-orientation-c

9. ANEXOS

- TFM_SergioGarcíaPrado - Anexo I - Ficha del TFM.docx
  Ficha resumen del proyecto. (Español / Inglés).

10. SOBRE EL AUTOR DE ESTE TRABAJO FINAL DE MASTER

Sergio García Prado

DNI: **11445532L**

Número de Expediente: **1280420**

Alumno del Máster en Ingeniería Informática.

E-Mail: sergiogarciap@gmail.com / sergiogp@uoc.edu

León, Enero 2016.