UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Meer: Generación automática de aplicaciones para terminales móviles inteligentes a partir de informes DICOM SR.

Proyecto Final de Carrera

Ingeniería Informática

Autor: Mayte Giménez Fayos
Director: Ignacio Blanquer
Co Director: Damià Segrelles
11 de septiembre de 2013

Resumen

El objetivo del siguiente Proyecto Fin de Carrera consiste en el desarrollo de una aplicación que a partir de un esquema de un informe médico, codificado en el estándar DICOM-SR, genere de manera automática una aplicación para dispositivos móviles de tipo Android de modo que facilite la adquisición de datos de pacientes e informes válidos.

El estándar Digital Imaging and Communication in Medicine (DICOM) define como tratar imágenes médicas desde su procesado hasta su comunicación. DICOM también especifica una extensión para soportar informes estructurados médicos (DICOM-SR), lo que ha permitido el desarrollo de distintas líneas de investigación basadas en este estándar.

En el ámbito médico, es fundamental que los datos adquiridos en los informes sean exhaustivos y fidedignos, sin embargo la gran versatilidad el estándar DICOM-SR hace que la propia adquisición de los datos sea un reto, en la que la aparición y expansión de terminales móviles inteligentes pueden facilitar la interacción entre el usuario y la máquina. Estas tecnologías nos ofrecen la posibilidad de obtener datos de informes clínicos sin incrementar significativamente la carga de trabajo del personal médico. Por ello, son estas cualidades las que pretendemos explorar y explotar en este PFC, simplificando el proceso de adquisición de información para la obtención masiva de datos y de una mayor calidad.

Para ello se plantea como objetivo del PFC, una aplicación para terminales Android, que permita al usuario introducir los datos de los informes estructurados basados en DICOM-SR, de un modo productivo, eficiente e intuitivo. La aplicación propuesta debe generar automáticamente a partir de un fichero plantilla donde se defina la estructura de informe DICOM-SR, el código de la aplicación Android que proporcione la funcionalidad de inserción de informes.

Palabras clave: DicomSR, Android, Python, Generación automática de código, imagen médica.

Índice general

Índice de figuras

Índice de código fuente

Índice de tablas

1 | Introducción

1.1. | Descripción del problema

Antes de profundizar en la descripción del problema que abordaremos en este proyecto final de carrera (PFC), hablaremos acerca del contexto en el que se inscribe.

La introducción de la imagen médica digitalizada ha transformado la práctica clínica. En la década de los 60 [1] comienzan los esfuerzos por desarrollar líneas de investigación para el diagnóstico asistido por ordenador (CAD), pero no es hasta la década de los 80 cuando la tecnología permite que esta disciplina despegue. Solo como apunte para mostrar el impacto que tiene el desarrollo de la imagen médica: la detección de cáncer de mama ha incrementado aproximadamente en un 10 % [2], aunque podemos encontrar muchos otros ejemplos [3, 4, 5, 6].

El formato de estas imágenes es crucial para el desarrollo de la investigación. Los Sistemas de Adquisición y Procesado de Imagen médica (PACs) [7] contienen tanto el software como el hardware necesario para el tratamiento de la imagen médica. Los principales elementos que forman los PACs son [8]:

- *Dispositivos de adquisición*: permiten la captura de la imagen médica. Estos dispositivos no tiene que compartir la misma tecnología de captura. Por lo que encontramos dispositivos de adquisición de imágenes como:
  - Ultrasonidos.
  - Resonancia magnética.
  - Tomografía Axial Computarizada (TAC).
  - Rayos X.
  - ...
- *Archivos de la imagen médica*: servidores donde almacenar las imágenes.
- *Estaciones de trabajo*: estaciones dónde los profesionales leen y trabajan con estas imágenes.

En la figura 1.1 vemos un esquema de los sistemas de adquisición y procesamiento de la imagen médica que acabamos de describir.

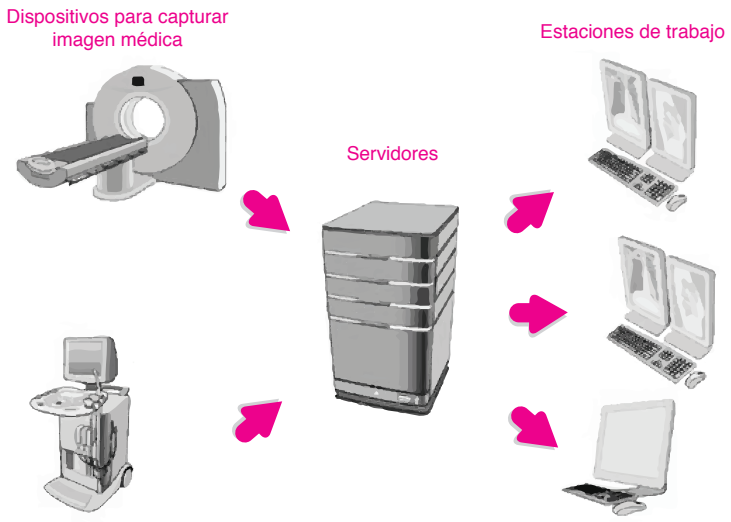


Figura 1.1: Componentes principales de los PACs

Y aunque en los inicios de estos sistemas cada fabricante comenzó desarrollando su propio formato para el majejo de las imagenes [9, 10], no tardaron en aflorar problemas derivados de la falta de un estándar para compartir y estudiar estas. Por este motivo surge en 1983 el estándar Digital Imaging and Communication in Medicine (DICOM).

DICOM es un estándar [11] desarrollado por el colegio Estadounidense de Radiología (ACR) y la asociación Nacional de Fabricantes eléctricos (NEMA), especifica no solo el formato de las imágenes sino también como deben ser almacenadas, los protocolos para el intercambio de las mismas.

En los veinte años de vida el estándar no ha dejado de crecer. Se han desarrollado más de 160 suplementos al mismo que completan las carencias detectadas en su uso y se adaptan a las nuevas tecnologías. Entre el 2006 y el 2013, los temas en los que se ha centrado el foco de atención son la mejora en las tecnologías para la adquisición de las imágenes y los informes estructurados asociados a estas.[12]. Precisamente de estos dos temas dan pie a dos líneas de investigación bien diferenciados en la literatura [13]:

- *Imagen médica*: cómo almacenar y procesar la información de las imágenes médicas.
- *Informes médicos*: informes estructurados con información de las imágenes.

Es precisamente en esta segunda rama de la investigación en la que nos centraremos en este PFC. Concretamente dentro del estándar DICOM, se especifica una extensión para tratar con los informes médicos estructurados, se trata del estándar DICOM-SR [14], del cual relegamos una definición más en profundidad en el aparatado 5.1.

La correcta estructuración de los informes permite organizar el conocimiento sobre las imágenes, permitiendo comparar imágenes de manera precisa mediante métodos cuantificables, y con esta información mejorar el diagnóstico. Estos informes estructurados, (así como la imagen médica) ofrece a los profesionales sanitarios herramientas de apoyo a la medicina basada en evidencias [15, 16, 17], y con los datos necesarios para entrenar a los sistemas de diagnostico asistidos por ordenador.

En la literatura podemos encontrar diferentes esfuerzos por mejorar el tratamiento del conocimiento contenido en los informes médicos[18, 19, 20] y de la utilidad del uso de los informes DICOM-SR.

Pero lo que es evidente es que para que los estudios derivados de los informes médicos estructurados (DICOM-SR) sean eficaces necesitamos disponer de un conjunto amplio y preciso de estos. Aunque el uso de métodos estadísticos nos ayuda a paliar en este problema, los modelos que se generen a partir de los datos y por extensión las conclusiones a las que pudiéramos llegar, serían en el mejor de los casos imprecisos.

Paradójicamente, nos encontramos con un cuello de botella en la adquisición de los datos en los informes DICOM-SR a partir de la imagen médica, ya que este proceso debe hacerlo un profesional. Rellenar formularios electrónicos pude ser una tarea tediosa, por lo que en la actualidad se emplea el reconocimiento del habla para transcribir los informes de los profesionales como texto plano. Pero el lenguaje natural es impreciso por lo que se necesitan herramientas para aplicar la estructura de un informe DICOM a este texto plano [21, 22, 23].

Y es en este punto dónde encontramos el problema que queremos abordar. Necesitamos informes médicos estructurados, pero su adquisición es demasiado costosa y poco productiva de la manera tradicional, ya que la interacción requerida para que el usuario introduzca los datos es poco intuitiva y monótona, y las herramientas para aplicar la estructura del informe médico DICOM-SR al texto plano se ciñen a tipos de imagen concretos (Radiografías, radiologías,. . .) debido a la complejidad de interpretar correctamente el lenguaje natural.

El otro elemento que nos hace falta para terminar de definir el problema, nos vincula con la solución que propondremos en el apartado 1.2, y es que afortunadamente las nuevas tecnologías nos ofrecen nuevas posibilidades. La interacción con los terminales móviles inteligentes han demostrado ser mucho más intuitiva y amigable para todo tipo de usuarios.

En los últimos años podemos encontrar trabajos que han comenzado a introducir terminales móviles inteligentes en el campo médico con resultados satisfactorios. [24, 25, 26]

Recapitulando, la existencia de estándares en la imagen médica ha facilitado el desarrollo de distintas líneas de investigación que han revolucionado la práctica clínica.

Los informes estructurados son una valiosa fuente de información médica, pero su adquisición requiere aumentar la carga de trabajo de los profesionales.

Los terminales móviles ofrecen nuevas formas de interactuar con los usuarios de manera más intuitiva, y su uso en el ámbito médico se ha probado productivo.

1.2. | Solución propuesta

La solución que se propone para el problema que hemos descrito en el apartado anterior, explota las ventajas que nos ofrecen los terminales móviles. Lo que se propone es desarrollar una aplicación que permita generar automáticamente una aplicación para terminales móviles inteligentes con el sistema operativo Android a partir de plantillas de los informes médicos estructurados (DICOM-SR).

Partimos de ficheros que sigan el estándar para informes médico DICOM-SR con la definición de la plantilla del tipo de informe en el que estamos interesados. Estos ficheros contienen información acerca del tipo de informe, así como de todos los campos que deben estar presentes en el mismo; también contiene para cada campo información acerca del tipo, de las propiedades de cardinalidad, verificación de los mismos y posibles valores a asignar.

A partir de estos ficheros, extraeremos la información de los mismos y generamos el código necesario para crear un aplicación para terminales móviles inteligentes Android. Una vez hemos generado todos los ficheros necesarios (interfaz, modelo de clases, interacciones entre la interfaz de usuario, ...), los integraremos en el esqueleto de una aplicación Android estándar para este propósito, y de este modo tendremos el informe estructurado, convertido en una aplicación para terminales inteligentes.

En la figura 1.2, podemos ver un esquema muy básico de la solución propuesta: la entrada del sistema es el fichero con la plantilla DICOM-SR, que se pasa a la aplicación en python que hemos desarrollado y esta genera una aplicación en Android con los datos del informe estructurado DICOM-SR. Entraremos en más detalles acerca de la arquitectura y la justificación de los detalles técnicos de la solución en los siguientes apartados.

Figura 1.2: Esquema básico

El objetivo que pretendemos lograr con esta solución es claro: hacer uso de una interfaz de usuario muy intuitiva característica de los terminales móviles inteligentes, para interactuar con el usuario. Cuanto más podamos simplificar el trabajo del usuario a la hora de codificar los informes, más probablemente el profesional introduzca un número mayor de informes y de una forma más productiva.

La solución propuesta tiene varios puntos fuertes, que son los siguientes:

- Se trata de una solución *genérica*. Independientemente del tipo de informe médico podemos de manera automática crear una aplicación Android que lo soporte.
- Es *adaptable* a los cambios en la plantilla DICOM-SR. Si añadimos nuevos campos, o modificamos el tipo de los mismos, la aplicación que se generaría cambiaría adaptándose a los cambios realizados en el informe.
- Es *configurarle*, ya el usuario podría decidir la configuración más intuitiva para él/ella.
- La *ubicuidad* de los terminales Android aporta al usuario la posibilidad al personal médico de comenzar la introcción de los datos en una pantalla táctil y continuar en una tableta.

En definitiva, se trata de una solución que busca que los profesionales médicos introduzcan los datos de las imágenes que analicen en los informes médicos estructurados de tipo DICOM-SR de una manera rápida e intuitiva. Los datos específicos de cada paciente introducidos por el profesional interactuando con la aplicación Android, se guardarán de nuevo en el formato estándar DICOM-SR.

2 | Objetivos

En este apartado enumeraremos los objetivos que se pretenden alcanzar en este PFC por parte de la alumna.

Esto nos permitirá acotar el alcance del proyecto y establecer los hitos a alcanzar, permitiendo elaborar una planificación lo más precisa posible.

Los objetivos a lograr son los siguientes:

- Demostrar la habilidad para emprender un proyecto parcialmente autónomo a partir de lo estudiado durante el periodo de formación académica en la Ingeniería Informática.
- Comprender el estándar DICOM-SR y ser capaz de interpretar el contenido de los informes estructurados basados en este.
- Desarrollar una aplicación que realice el análisis sintáctico y semántico de ficheros DICOM-SR.
- Desarrollar una aplicación Android genérica para generar interfaces que permitan la introducción de informes DICOM-SR en base a plantillas.
- Desarrollar una aplicación que a partir de un fichero DICOM-SR genere los ficheros necesarios para completar la aplicación Android básica para el informe concreto de entrada.
- Comprender la generación de código haciendo uso de plantillas.
- Realizar las pruebas pertinentes a la aplicación para un informe concreto. Estas pruebas las aplicaremos exhaustivamente a 4 plantillas de DICOM-SR relacionados con:
  - Exploración de mama.
  - Mamografía.
  - Ultrasonidos
  - Resonancia magnética.

  Y consistirán en realizar las siguientes tareas:

- Generar el código XML para soportar la internacionalización
- Generar el código XML que desarrolle la interfaz de usuario.
- Generar el código XML con las propiedades de la aplicación Android.
- Generar el código java que codifique los modelos de clase del informe.
- Generar el código java que codifique las interacciones entre las pantallas de la aplicación Android.
- Integrar todo el código generado dentro de la aplicación base de Android.
- Comprobar su correcto funcionamiento en terminales físicos.

3 | Planificación del PFC

Aunque no sea estrictamente necesario, hemos creado un diagrama de Gantt que podemos ver en la figura 3.1 y 3.2, que nos ayude a planificar el tiempo para lograr los objetivos que acabamos de describir.

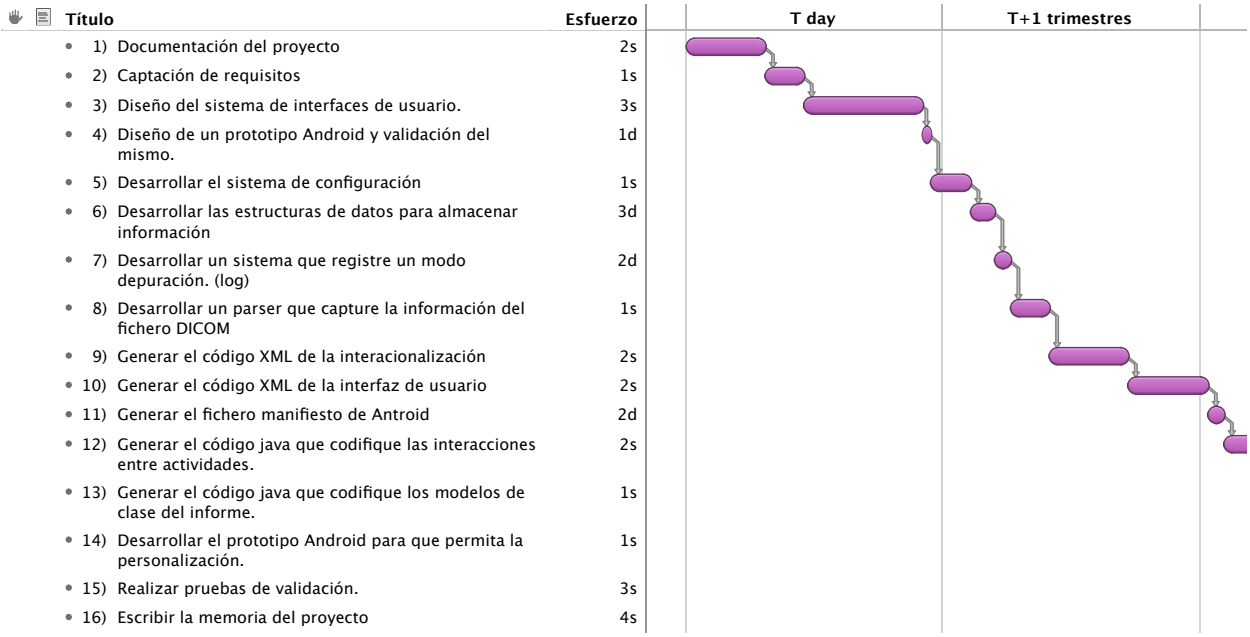


Figura 3.1: Parte 1 del diagrama de Gantt

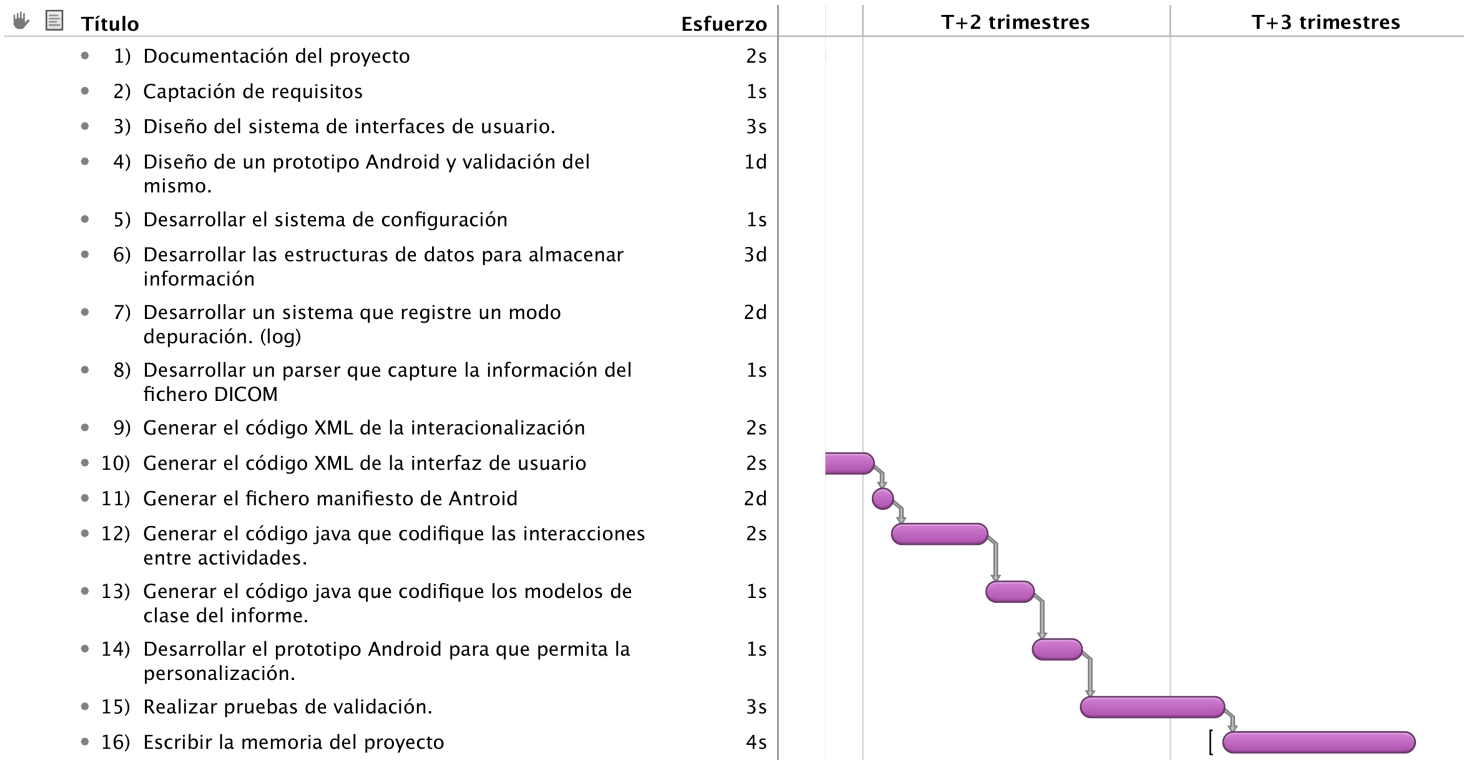


Figura 3.2: Parte 2 del diagrama de Gantt

4 | Estado del arte

Hasta este punto hemos introducido el problema y el contexto en el que ser circunscribe de manera generalista. En los siguientes apartados profundizaremos un poco más en el contexto del problema. Haremos un resumen del estado del arte de las áreas de conocimiento en las que vamos a trabajar.

4.1. | Imagen médica

Situamos la primera imagen médica en 1895 con la imagen de rayos X de Wilhelm Conrad Röntgen, pero no es hasta 1972 cuando encontraremos la primera imagen médica digitalizada: una tomografía computarizada de Godfrey Newbold Hounsfield. Paralelamente al desarrollo de la tecnología de los aparatos para la captación de imágenes hemos visto el desarrollo de las técnicas de la informática médica.

Encontramos en [27], una definición de lo que es la informática médica. Se trata del procesamientos sistemático de datos, información y conocimiento para la toma de decisiones óptima.

La medicina es un ámbito en que se puede aprovechar al máximo las nuevas posibilidades que nos ofrecen las tecnologías de la información. Es por esto que en los últimos 20 años hemos visto como ha florecido distintas líneas de investigación alrededor de la imagen médica.

Cuando hablamos de imagen médica nos referimos a un conjunto de técnicas y procesos para capturar imágenes totales o parciales del cuerpo humano con propósitos cínicos [28].

En la literatura [29, 30] podemos encontrar un repaso de los distintos dispositivos y tecnologías para captar imagen médica. Lo que es innegable es el gran impacto de la imagen médica en la práctica clínica.

Como afirmábamos en el apartado 1.1, para hacer posible esta evolución son necesarios los estándares que permitan manipular las imágenes de manera óptima. Es precisamente por esto que se desarrolló el estándar de imagen médica DICOM. El acrónimo DICOM significa *Imagen Digital y Comunicación en Medicina*. Por lo tanto no se trata únicamente del formato de la imagen sino que se diseñó para cubrir todas las necesidades vinculadas con la imagen médica. Necesidades como pueden ser: la compresión de imágenes, la comunicación, el almacenamiento, ...

Cada una de las necesidades específicas para el desarrollo de investigaciones y dispositivos relacionados con la imagen médica se recogen en el estándar y sus anexos que podemos encontrar en la web de la asociación Nacional de Fabricantes eléctricos (NEMA) [31].

La literatura al respecto de la imagen médica en muy profusa. Por lo tanto, lo visto hasta el momento son únicamente los conceptos más básicos así como sus definiciones. La teoría sobre la imagen médica que hemos descrito forma parte de los cimientos sobre los que se sustenta este PFC.

4.2. | Generación automática de código

Otro de los pilares teóricos en los que se sustenta este proyecto es la generación automática de código.

Cada una de las pruebas de imagen médica tiene asociado un informe DICOM-SR. Si nos planteáramos seguir un paradigma de programación diferente, necesitaríamos crear una aplicación Android para cada informe DICOM-SR. Aunque gran parte de este código fuera reutilizable, estaríamos desplazando el cuello de botella que ahora se encuentra en la captación de datos al desarrollo de aplicaciones.

Es por este motivo que optamos por la generación de código o traducción automática, términos sinónimos en la literatura [32].

La generación automática de código es uno de los paradigmas de programación existentes y consiste en escribir programas que sean capaces de escribir el código fuente de otros programas basándose en modelos ontológicos.

En nuestro caso los modelos ontológicos son las plantillas DICOM-SR. Para conseguir llevar a cabo esta tarea se dispone de un sistema de patrones que se instancian con los datos concretos, siguiendo unas reglas que son generalmente sencillas.

La generación automática de código es una de las líneas de investigación que más interés despierta en el ámbito de la ingeniería de software [33]. Existe un buen número de razones para que esto sea así, a pesar de los desafíos que plantea este paradigma de programación. Entre los beneficios que ofrece la generación automática de código podemos enumerar los siguientes [34]:

- *Calidad*: El código generado mediante plantillas permite que la calidad sea consistente a lo largo de todo el software desarrollado y simplifica el proceso de aplicar parches que solucionen bugs.
- *Consistencia*: El estilo de programación mantiene una consistencia a lo largo de todo el software desarrollado.
- *Más tiempo para el diseño del software y la arquitectura*: el análisis y desarrollo de la API en este paradigma es muy importante, por lo que se dedica más tiempo y recursos a captar los requisitos y crear la arquitectura. Estamos aprovechando el tiempo que de otro modo hubiéramos empleado en escribir manualmente el software.

- *Abstracción*: la generación automática de código es independiente del lenguaje de programación, simplemente modificando el lenguaje de las plantillas podremos generar software en el lenguaje de programación que sea más conveniente.

La investigación actual tiene como gran reto el desarrollar modelos independientes del lenguaje y demostrar que el código que generan es computacionalmente equivalente al código generado por un desarrollador.

Sin embargo este enfoque queda fuera del alcance de nuestro PFC. Lo que haremos es emplear las bases teóricas del paradigma de generación automática en nuestro desarrollo.

Existen diversas técnicas para generar código. Para este proyecto optamos por *generación parcial de clases*. La generación parcial de clases consiste en leer un fichero de texto con la información abstracta de las clases a generar, a continuación lee una serie de plantillas y con la información recogida de estas dos fuentes, generará el código necesario. Este código que hemos generado se integrará con el código escrito por ingenieros para formar la solución final.

En la figura 4.1 vemos como se aplica esta técnica de generación de código en nuestro ejemplo concreto. El sistema tendrá como entrada el informe médico de tipo DICOM-SR y una serie de plantillas, con esto generará el código necesario para que cuando lo integremos en la aplicación Android obtengamos una aplicación Android funcional específica para el informe de entrada.

4.2.1. Desarrollo de software guiado por modelos

El desarrollo de software guiado por modelos forma parte del paradigma de programación automática. De los paradigmas que descienden de la programación automática este es el que más se ajusta al trabajo que estamos presentando.

Consiste en desarrollar uno o varios modelos que representen el sistema y a partir de estos modelos se generará el software. Existen numerosas herramientas para definir modelos.

Según la literatura [35] lo fundamental es que los modelos tengan las siguientes características:

- Tan simples como sea posible (KISS).
- Canónico (DRY).
- Correcto nivel de abstracción.
- Separación de aspectos (SoC).
- Mantener modelos organizados y manejables.
- Independiente de la tecnología.
- Pragmáticos: sólo modelamos aquello que vayamos a emplear.

Podemos modelar todo o parte del sistema. Lo recomendable es que se modele aquello que varíe poco, con lo que el tiempo invertido en el modelado sea eficiente.

En nuestro caso el modelo viene determinado por el estándar soportado por los dispositivos PACs, es decir, utilizaremos el estándar DICOM-SR como modelo para generar la aplicación.

Debido a las características del modelo, este proyecto no se adscribe dentro del desarrollo guiado por modelos tal y como se define en la literatura [36], pero si obviamos las características del modelo que vienen impuestas por los estándares médicos, seguimos la filosofía de este paradigma de programación, ya que a partir del modelo del sistema, el informe médico DICOM-SR, generamos el software con las características que el modelo define.

4.3. | Interacción persona-máquina y usabilidadad

Por último, antes de cerrar este capítulo nos falta hablar del tercer pilar que sustenta este proyecto: la interacción persona-máquina.

Como hemos discutido en la introducción 1.1, uno de los problemas a los que nos enfrentamos a la hora de recoger informes médicos es la de rellenar estos informes médicos, ya que es una tarea pesada para los profesionales clínicos que deben invertir mucho tiempo en la captación de datos. La tecnología per se no soluciona este problema, sino que nos da herramientas para afrontarlo. Lo que es importante es que realicemos un diseño de las interfaces de usuario centrándonos en las necesidades de las personas que utilizarán las aplicaciones.

El término Interacción Persona-Máquina (***H**uman-**C**omputer **I**nteraction* en inglés) no se popularizó hasta la década de los 80, aunque podemos encontrar raíces similares en los estudios de ergonomía de principios del siglo XX [37]. Pero es con el auge de la informática y diseño de sistemas cuando el estudio de la Interacción persona-máquina comienza a desarrollarse plenamente de manera autónoma. Con la aparición de los ordenadores personales todo el mundo se convirtió en un usuario potencial y era necesario encontrar formas de interactuar adecuadamente con este nuevo público.

La Interacción Persona-Máquina se encarga del diseño, implementación y validación de las interfaces de usuario. Es decir, se encarga de modelar cómo los usuarios se relacionan con las máquinas.

Al desarrollar un proyecto, como el que afrontamos en este PFC, corremos el peligro cuando estemos absortos en los detalles técnicos de olvidar la importancia del diseño. Un buen diseño debe centrarse en las personas, y tiene un tremendo impacto en las tareas que realizan a pesar de que el diseño será transparente para los usuarios.

En ámbitos que incluyan sistemas críticos como puede ser la medicina, los fallos en el diseño de las interacciones pueden tener un gran coste económico y personal. A pesar de que nuestro ámbito de aplicación no es crítico tenemos presente que la aceptación del sistema por parte de los usuarios es un punto clave para resolver el problema que nos planteamos en este proyecto.

Los puntos claves para conseguir un buen diseño de interfaces se basan en que todo el sistema sea consistente y en solicitar información al usuario final acerca de sus impresiones y sensaciones con el diseño.

Respecto a la consistencia, esta es una ventaja propia de la generación automática de código ya que mientras seamos consistentes en el diseño de las plantillas la aplicación final tendrá una aspecto coherente.

Para conseguir la información acerca de la idoneidad del diseño, el procedimiento consiste en crear un diseño y evaluar la satisfacción por parte de los usuarios, con las respuestas que obtenemos de nuestro usuarios, que idealmente será un grupo diverso que represente a las personas que utilizarán la aplicación, cambiamos el diseño y solicitamos de nuevo una evaluación del diseño. Idealmente seguiríamos iterando en el diseño de este prototipo hasta llegar a un punto en el los usuarios estén satisfechos con el resultado. En la práctica se llega a una solución de compromiso entre el tiempo empleado en el diseño y la satisfacción de los usuarios.

En nuestro PFC hemos integrado los ciclos de diseño y validación de la interfaces dentro del desarrollo de la aplicación. En la fases iniciales de diseño, creamos las interfaces que se fueron refinando durante el proceso, y empleamos la aplicación Android que nos serviría de esqueleto dónde integrar el código generado como prototipo para poder validar el diseño.

Finalizamos este apartado recalcando la importancia del diseño, muchas veces denostado, en la aceptación por parte de los usuarios de las soluciones software que les proponemos.

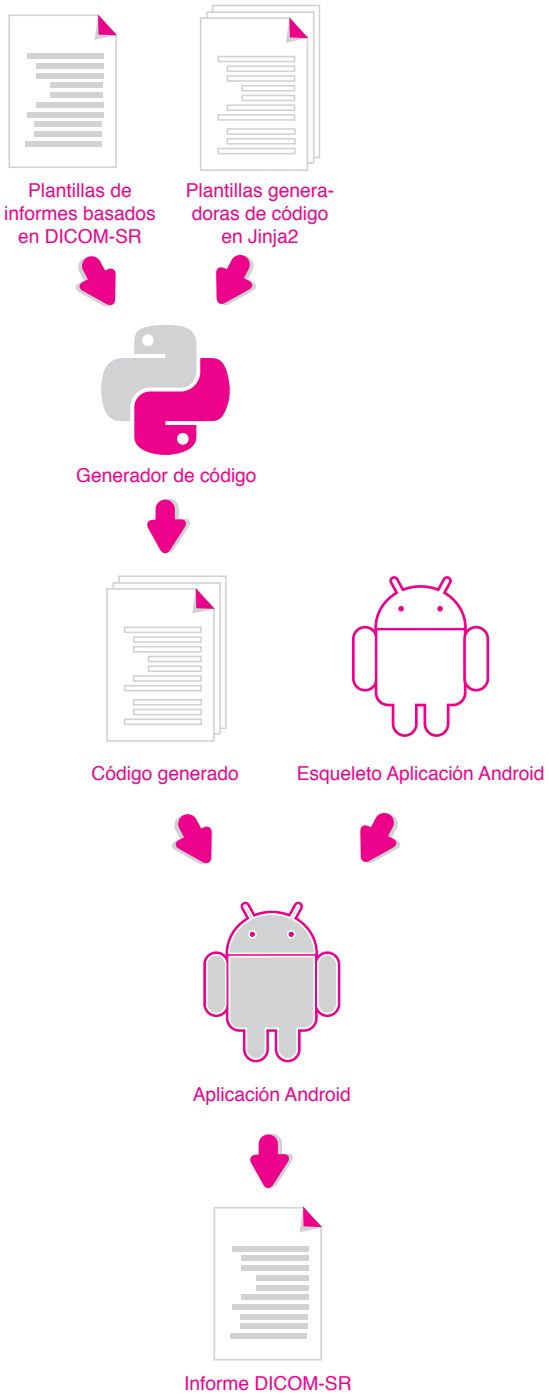


Figura 4.1: Generación parcial de clases

5 | Tecnologías empleadas en el PFC

En el capítulo 1.1 hemos planteado el problema y una posible solución al mismo. El objetivo tanto del capítulo anterior como este es el de sentar las bases teóricas y tecnológicas de la solución que hemos propuesto.

Si en el capítulo 4 repasábamos el estado del arte y sentábamos las bases teóricas del proyecto, en este capítulo describiremos la tecnología de la que hacemos uso para ejecutar de solución que proponemos. Además justificaremos los motivos que nos han llevado a seleccionar esta tecnología y no otra.

5.1. | DICOM-SR

En primer lugar hablaremos de la tecnología que nos impone el proyecto. Los PACs utilizan la extensión del estándar DICOM para informes estructurados(DICOM-SR) y como nuestro objetivo es que la solución se integre en el ecosistema existente en los centros médicos deberemos emplear este estándar.

5.1.1. Definición

DICOM-SR se incluye dentro del suplemento 23 del estándar de imagen médica DICOM. Describe una arquitectura de documento que permite compartir, almacenar y transmitir información de informes médicos estructurados [38].

Se diseñó con la intención de suplir la brecha entre la información contenida en las imágenes y la información que los profesionales médicos introducen en estos. Los ficheros DICOM-SR almacenan de forma no ambigua y jerárquica todos los conceptos que podemos encontrar en un informe médico tradicional y además pueden incluir referencias a:

- Imágenes en formato DICOM.
- Informes de estudios previos.
- Detalles de las imágenes.

El estándar define el modelo de la información y cómo debe gestionarse el documento, lo que permite personalizar muchos aspectos de la implementación final[39].

Sin embargo al tratarse de un estándar bastante complejo, han surgido soluciones bastante dispares, entre las que podemos encontrar:

- Una implementación basada en la orientación objetos integrando el estándar DICOM-SR en la estructura de XML.[40]
- Una implementación extendiendo el modelo de objetos de documento XML (DOM).[41]
- Una implementación en C y C++ del estándar. [42]

Desde la Asociación Nacional de Fabricantes eléctricos (NEMA), se están haciendo esfuerzos por unificar los criterios y seguir avanzando en la definición del estándar.

Para este proyecto se emplea una implementación en la que el informe se estructura en un XML. Expondremos los detalles de los ficheros DICOM-SR con los que trabajaremos en los apartados 5.1.3,5.1.4, 5.1.5 y 5.1.6.

5.1.2. Beneficios del estándar DICOM-SR

A pesar de los problemas que surgen en la definición y en la implementación del estándar, los beneficios que aporta su desarrollo merecen el esfuerzo. Podemos encontrar un resumen exhaustivo de las ventajas de DICOM-SR en el siguiente artículo [43]. Entre las mejoras que aporta a la práctica clínica más relevantes encontramos:

- Mejora la comunicación entre los profesionales, al utilizar un léxico estándar no hay lugar a traducciones o interpretaciones erróneas.
- Los informes son más precisos y concisos. Los profesionales rellenan el informe utilizando los códigos adecuados sin las estructuras gramaticales que serían necesarias al redactar los informes de modo tradicional.
- Se evitan los errores gramaticales y de transcripción.
- La interpretación de los informes es más sencilla y permite una interpretación asistida por ordenador.
- Las imágenes DICOM y los informes DICOM-SR comparten la misma cabecera que contiene información acerca del paciente, lo que mejora el registro de información médica.
- El informe incluye medidas numéricas de las evidencias encontradas, que redundará en la precisión del informe.
- Permite ejecutar acciones automáticas sobre los informes permitiendo la minería de datos.
- Se enfatizan los contenidos. El informe en DICOM-SR no guarda información de cómo debe mostrarse al usuario, separa el contenido de la presentación de los datos.
- Permite la transformación a otros formatos.

- Permite la integración con sistemas que reconozcan el habla.

Como podemos comprobar el desarrollo del estándar tiene un impacto beneficioso directamente sobre la práctica clínica, es por esto que en los últimos 5 años ha crecido el interés de la comunidad científica por explotar el estándar DICOM-SR.

5.1.3. Estructura de un informe médico estructurado

A continuación describiremos las características de un informe médico DICOM-SR representado siguiendo el formato XML.

En un fichero XML que contiene un informe médico DICOM-SR, la información se organiza mediante contenedores *<CONTAINER>*. Los contenedores no almacenan la información, únicamente la estructuran. Los contenedores tienen hijos *<CHILDS>*, y es dentro de los hijos dónde se almacena la información correspondiente a un informe concreto. Cada elemento de información se agrupa con las etiquetas con el tipo del valor que designan el tipo de elemento que se almacena en el campo. En nuestro PFC soportamos las siguientes, aunque existen otros muchos:

- *<DATE>*: almacena fechas.
- *<TEXT>*: almacena texto sin formato.
- *<NUM>*: almacena enteros o boleanos.
- *<CODE>*: almacena campos multievaluados.

Dentro de las secciones definidas por estas etiquetas de tipo valor encontramos los datos del informe médico en pares de elementos clave-valor. Para designar la clave utilizamos la etiqueta *<CONCEPT_NAME>*, mientras que para referirnos al valor de este contenedor utilizamos la etiqueta *<VALUE>*. En el ejemplo de la figura 5.1 podemos ver como se codifica el identificador *M001* de una lesión de tipo masa. Vemos un contenedor que abarca el concepto *masa* y entre los hijos encontramos el identificador de la masa que es de tipo texto.

En el campo valor, *<VALUE>*, podremos encontrar entre otros:

- Texto plano.
- Valores numéricos con sus unidades.
- Fechas.
- Referencias a imágenes DICOM.
- ...

Un punto importante para alcanzar los objetivos descritos en 5.1.2, es trabajar con un léxico limitado y bien conocido, por lo que se utilizan diccionarios de conceptos médicos para identificar cada concepto de los que puede aparecer en un informe.Cada concepto del informe se identifica por su *CONCEPT_NAME* que está formado por tres elementos:

- *CODE_SCHEMA*: identifica el diccionario dónde se encuentra el término que estamos definiendo.
- *CODE_VALUE*: es el código al término médico definido.
- *CODE_MEANING*: contiene una descripción en texto plano del término para que sea legible por el usuario.

La combinación de *CODE_SCHEMA* y *CODE_VALUE* identifica de modo único un concepto dentro del informe.

De nuevo empleando el ejemplo 5.1, vemos que el concepto *masa* se encuentra en el diccionario *RADLEX* con el código *RID3874*.

```
...
    <CONTAINER>
      <CONCEPT_NAME>
        <CODE_VALUE>RID3874</CODE_VALUE>
        <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
        <CODE_MEANING>Mass</CODE_MEANING>
      </CONCEPT_NAME>
      <CHILDS>
        <TEXT>
          <CONCEPT_NAME>
            <CODE_VALUE>118522005</CODE_VALUE>
            <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
            <CODE_MEANING>Identifier</CODE_MEANING>
          </CONCEPT_NAME>
          <VALUE> M001 </VALUE>
        </TEXT>
        ...
```

Código 5.1: Fragmento de un informe estructurado de una exploración de mama

La información DICOM-SR del informe rellena el árbol XML. Existen implementaciones del estándar que hacen uso de etiquetas específicas para indicar las relaciones entre los contenedores DICOM, pero debido a que los ficheros XML almacenan la información per se en forma de árbol estas etiquetas son redundantes.

En la figura 5.1 se puede ver un esquema del árbol de conceptos de un informe siguiendo el formato DICOM-SR. Se trata de un exploración de mama, en la que se ha encontrado una masa en la mama derecha y dos asimetrías en la mama izquierda. Los campos de tipo valor, es decir los atributos del contenedor padre los representamos utilizando los cuadrados grises. Mientras que los contenedores, que no almacenan información pero estructuran el informe, los representamos con los cuadrados magentas.

Podemos encontrar el fichero DICOM-SR completo del informe que representa esta figura en el apéndice A.

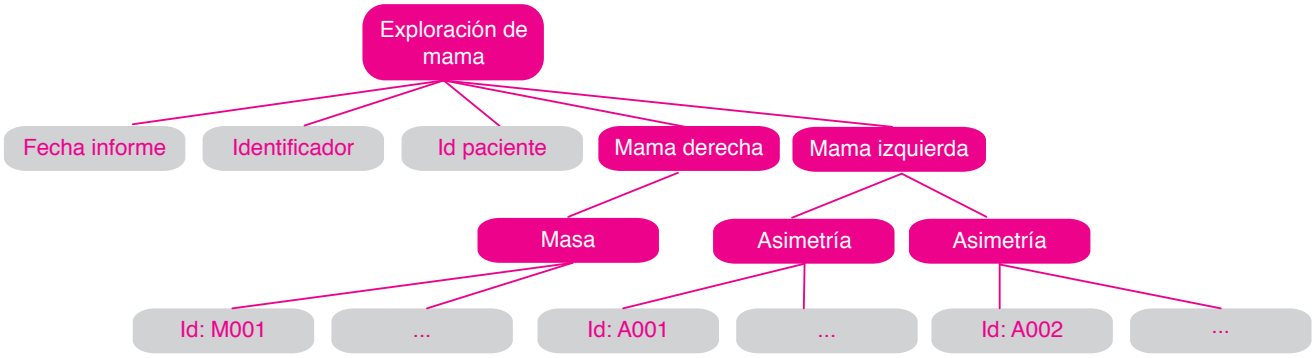


Figura 5.1: Representación del árbol XML de un informe en formato DICOM-SR

5.1.4. Estructura de una plantilla para un informe médico estructurado

El estándar DICOM-SR, permite gran flexibilidad a la hora de escribir los informes estructurados. Esta característica puede complicarnos mucho el trabajo si cada especialista médico escribiera los informes siguiendo criterios correctos según el estándar pero aleatorios. Para solucionar esto se crean las plantillas. Las plantillas describen cómo debe ser un informe de tipo DICOM-SR, incluye los conceptos que debe tener así como las relaciones de jerarquía entre ellos y las propiedades que deben cumplir: si los conceptos son obligatorios o no, si pueden repetirse a lo largo del informe, ...

Únicamente se guardan los conceptos en las plantillas, los campos correspondientes a los valores *<VALUE>* se incluirán después cuando el profesional rellene los datos para un paciente concreto.

El formato de las plantillas DICOM-SR es muy similar al de los informes DICOM. Carecen de las etiquetas *<VALUE>* pero incluyen etiquetas *<PROPERTIES>* para especificar las propiedades que deben tener los campos.

En el ejemplo de código 5.2 vemos como se almacenaría un concepto de tipo masa en una plantilla para la exploración de mama. Este contenedor podrá aparecer 0 o más veces en el informe final (*<CARDINALITY max="-1" min="0"/>*) y lo introduce el usuario (*<CONDITION_TYPE type="U"/>*).

```
1       ...
2             <CONCEPT_NAME>
3               <CODE_VALUE>RID3874</CODE_VALUE>
4               <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
5               <CODE_MEANING>Mass</CODE_MEANING>
```

```
            </CONCEPT_NAME>
            <PROPERTIES>
              <CARDINALITY max="-1" min="0"/>
              <CONDITION_TYPE type="U"/>
              <EXPRESION_CONDITION xquery=""/>
            </PROPERTIES>
          ...
```

Código 5.2: Fragmento de un plantilla informe estructurado: codificar una anomalía de tipo masa en una exploración de mama.

Además, en las plantillas de los contenedores se pueden incluir etiquetas para indicar la unidad de medida de los mismos, dentro de las etiquetas *<UNIT_MEASUREMENT>*. Estas etiquetas también servirán como modificadores como es el caso del ejemplo 5.3, donde la etiqueta para el tipo del valor número, *<NUM>*, se modifica de modo que el valor que introduzca el usuario será de tipo boleano, es decir sólo se permiten los valores 0 y 1. Así el concepto "Cuadrante superior exterior de la mama derecha" tendrá el valor verdadero o falso si la anomalía se sitúa en esa posición de la mama.

```
    ...
            <NUM>
              <CONCEPT_NAME>
                <CODE_VALUE>RID29929</CODE_VALUE>
                <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
                <CODE_MEANING>Upper Outer Quadrant of Right Female
                    Breast</CODE_MEANING>
              </CONCEPT_NAME>
              <PROPERTIES>
                <CARDINALITY max="1" min="1"/>
                <CONDITION_TYPE type="M"/>
                <EXPRESION_CONDITION xquery=""/>
                <DEFAULT_VALUE value="0"/>
                <UNIT_MEASUREMENT>
                  <CONCEPT_NAME>
                    <CODE_VALUE>000000001</CODE_VALUE>
                    <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
                    <CODE_MEANING>Boolean Units</CODE_MEANING>
                  </CONCEPT_NAME>
                </UNIT_MEASUREMENT>
              </PROPERTIES>
            </NUM>
            ...
```

Código 5.3: Fragmento de una plantilla de un informe estructurado: codificar un atributo de tipo booleano.

Las plantillas deben sintetizar el conocimiento de los especialistas técnicos que conozcan el estándar DICOM-SR y los profesionales médicos que conocen las características que deben tener los informes de las pruebas clínicas, por lo que será imprescindible la colaboración entre los profesionales de ambos ámbitos para crear las plantillas DICOM-SR.

Utilizamos estas plantillas como entrada para nuestro sistema. Tenemos una plantilla por ontología de exploración médica. Para este proyecto estamos trabajando con 4 ontologías diferentes.

- Exploración de mama.
- Mamografía.
- Escáner de ultrasonidos.
- Resonancia Magnética

En el caso de las plantillas la información también se estructura en forma de árbol. En el apéndice B, hemos incluido una plantilla truncada para el informe de una exploración de mama. En este apéndice se describe una plantilla que permite introducir lesiones de tipo masa en la mama derecha y asimetrías en la mama izquierda. El esquema 5.2 muestra esta plantilla con su forma arbórea.

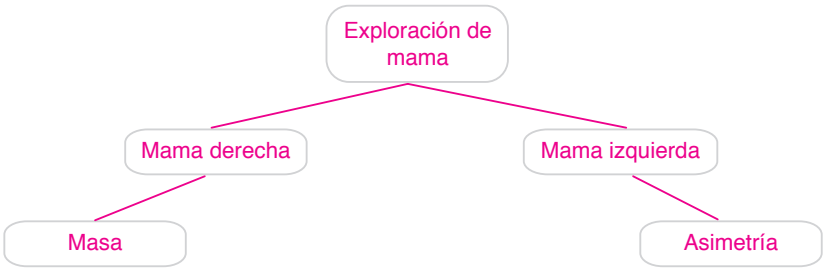


Figura 5.2: Representación del árbol XML de una plantilla para un informe en formato DICOM-SR

5.1.5. Vocabulario

Un punto en el que hemos hecho hincapié es en la ventaja que supone que los informes hagan uso de un vocabulario estandarizado. Esto permite hacer referencia a conceptos concretos de manera precisa. Evitando los problemas que supone la barrera lingüística y la falta precisión del lenguaje natural.

También en lo que respecta al uso de vocabularios médicos DICOM-SR permite gran flexibilidad, permitiendo a los usuarios seleccionar el léxico que más les convenga en cada momento. En un mismo informe estructurado DICOM-SR pueden aparecer varios vocabularios.

Los vocabularios más habituales que podemos encontrar son:

- *RadLex*: léxico desarrollado para cubrir las necesidades de la imagen radiológica. Contiene más de 30.000 términos [44].
- *ICD-10*: léxico generalista que contiene un índice internacional de enfermedades y problemas relacionados con la salud. Está traducido en 42 idiomas [45].
- *SNOMED-CT*: se considera la terminología clínica que más términos abarca y que soporta más lenguajes [46]. Se está convirtiendo en un estándar de facto al estar apoyado por numerosos gobiernos internacionales [47].

Sin embargo estos léxicos también presentan algunos inconvenientes: los conceptos pueden aparecer en varios léxicos, existen léxicos muy específicos para un país o idioma...

Para este proyecto utilizaremos los siguientes léxicos:

- Estándares internacionales:
  - *RadLex*.
  - *SNOMED-CT*.
- Vocabularios definidos por nostros:
  - *TRENCADIS_MAMO*.
  - *UNIT_MEASUREMENT*.

5.1.6. Internacionalización

Las máquinas pueden trabajar con códigos como los que se almacenan en los léxicos médicos, incluso para algunas tareas puede ser incluso más beneficioso que tener que lidiar con secuencias de caracteres, especialmente si estos incluyen caracteres especiales. Pero las personas necesitamos el vocabulario para comprender los conceptos y poder trabajar con ellos.

La internacionalización y localización consiste precisamente en adaptar el software a los distintos idiomas y a las diferencias regionales como puedan ser los formatos de las divisas. No se trata únicamente de traducir las cadenas de texto.

La internacionalización es la primera parte del proceso, consiste en diseñar y preparar el software que vamos a desarrollar y la localización es el proceso de adaptar el software creado a los distintos idiomas que se soporten [48].

Para presentar el informe a los profesionales médicos hacemos uso del texto codificado en las etiquetas *<CODE_MEANING>*. Sabiendo que algunos de los léxicos que se pueden incluir dentro un informe estructurado DICOM-SR están traducidos a múltiples idiomas, necesitamos encontrar una manera de que el significado del concepto sea diferente para cada lenguaje.

El estándar DICOM-SR especifica que las cadenas que se introduzcan en el campo *<CODE_MEANING>* pertenezcan al léxico con el que se esté trabajando, lo que no implica que estos textos deban estar en inglés, pero restringe la posibilidad de implementar soluciones en las que las traducciones no provengan de fuentes oficiales, porque esto nos conduciría de nuevo a la confusión semántica que pretendíamos evitar con el uso de vocabularios.

La solución que se propone en la literatura [14], es hacer uso de la codificación del fichero para deducir el lenguaje del mismo, pero aunque se trata de una solución sencilla que no implica tener que trasmitir ni almacenar más información, si varios idiomas pueden utilizar la misma codificación no hay ninguna manera de diferenciarlos siguiendo este método. La otra solución que se propone es utilizar una etiqueta para identificar el idioma. Los ficheros XML con los que trabajamos optan por la segunda opción. Utilizan la etiqueta *<CODE_MEANING>* para un primer idioma y *<CODE_MEANING2>* para un segundo como podemos ver en el ejemplo 5.4.

```
1      ...
2   <CONTAINER>
3     <CONCEPT_NAME>
4       <CODE_VALUE>RID10312</CODE_VALUE>
5       <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
6       <CODE_MEANING>Resonancia Magnética</CODE_MEANING>
7       <CODE_MEANING2>Magnetic Resonance Imaging</CODE_MEANING2>
8     </CONCEPT_NAME>
9               ...
```

Código 5.4: Internacionación y localización en ficheros DICOM-SR.

En nuestro PFC trabajaremos con plantillas localizadas para castellano e inglés como las que hemos visto en el ejemplo 5.4, así como con plantillas únicamente en inglés sin localizar como los fragmentos vistos en los ejemplos 5.1, 5.2 y 5.3

5.2. | Android

Los teléfonos móviles inteligentes y las tabletas son tecnologías que ya llevan bastante tiempo entre nosotros, pero es en los últimos cinco años cuando la tecnología ha crecido exponencialmente. El índice de penetración en el mercado supera el 50 % en el caso de China y Estados Unidos, mientras que en Europa encontramos porcentajes inferiores, alrededor del 25 % pero las previsiones estiman que estas tasas crezcan rápidamente [49].

La integración de dispositivos, la posibilidad de acceder a Internet y sobretodo la simplicidad de su uso fomentada por la posibilidad de interactuar con el dispositivo a través de la pantalla táctil, mucho más intuitivo que el teclado y el ratón, junto con un muy buen diseño de interfaces han facilitado la gran adopción de esta tecnología ya que han conseguido ampliar el espectro de usuarios potenciales

Un punto fundamental en el éxito de estas tecnologías es sin duda el nacimiento de Android.

Android es un sistema operativo basado en Linux, diseñado para trabajar con dispositivos móviles con pantallas táctiles. Fue desarrollado inicialmente por Android Inc., que fue comprada por Google en 2005. En el 2007 se forma la Open Handset Alliance: un conglomerado de empresas de hardware, software y telecomunicaciones cuyo objetivo es el de crear un entorno estandarizado para el desarrollo de dispositivos móviles. La OHA disponía de todos los agentes necesarios para revolucionar el mercado de las comunicaciones móviles y es en Octubre de 2008 cuando aparece el primer teléfono inteligente el HTC Dream que corría un Android 1.0. [50].

A partir de este momento el desarrollo de Android no ha dejado de crecer, convirtiéndose en una tecnología clave.

Como ya se ha comentado en el apartado 1.1, el uso de terminales móviles inteligentes en la práctica clínica para trabajar con imagen médica ha dado resultados muy satisfactorios. Por lo tanto, es obvio pensar que una aplicación para recopilar informes médicos sobre Android tendrá mejor acogida por parte del personal médico, ya que es más sencillo e intuitivo introducir los datos de los informes DICOM-SR. Y al facilitar esta tarea al personal médico, es más factible que podamos adquirir más informes DICOM-SR con los que realizar estudios posteriores.

5.2.1. Arquitectura

En este apartado hablaremos brevemente de la arquitectura de Android.

Android es un sistema operativo abierto basado en Linux, cuyo núcleo está escrito en C y C++, y que fundamentalmente tiene soporte para Java. Google libera periódicamente el código del sistema operativo bajo la licencia Apache.

En la figura 5.3 vemos un esquema de la arquitectura Android, compuesta por:

- *Núcleo Linux*: el núcleo es un fork de la versión 2.6 de Linux. Sirve de abstracción con el hardware y gestiona los recursos del sistema.

- *Librerías*: librerías nativas del sistema las utilizará el framework de Android para comunicarse con el hardware. Están escritas en C o C++.

- *Android Runtime*: se trata del entorno de ejecución de Android. Se integra con las librerías. Aquí tenemos el conjunto de librerías habituales de java, así como librerías Java específicas para Android. La Dalvik VM, es la máquina virtual de Java que ejecutará la aplicaciones.

- *Framework para aplicaciones*: Capa formada por las clases y servicios que utilizan directamente las aplicaciones de usuario para interactuar con Android.

- *Aplicaciones*: esta última capa incluye todas las aplicaciones del sistema y las instaladas por el usuario, tanto las nativas como las escritas en java.

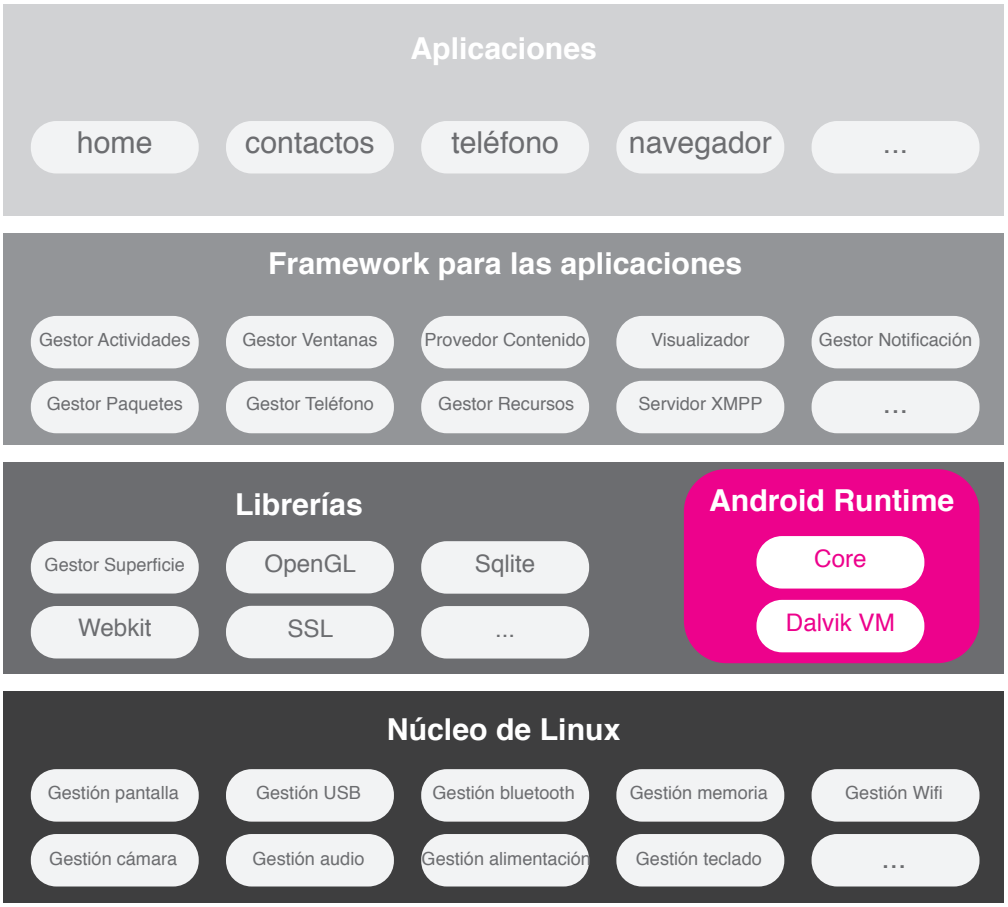


Figura 5.3: Arquitectura Android

5.2.2. Segmentación

El hecho de que exista una amplia gama de dispositivos que utilizan Android como sistema operativo también tiene una parte negativa y es que los terminales más antiguos ejecutan versiones obsoletas de Android y no es posible actualizarlos. Lo que lleva a que en el mercado convivan distintas versiones de Android.

La segmentación es uno de los problemas con los que tiene que lidiar Android. En la actualidad conviven 7 distribuciones de Android.

Desde la web de Android [51] mantienen una tabla con el porcentaje de dispositivos que usan cada versión (tabla 5.1).

Lo que recomienda Google a la hora de seleccionar una versión de Android es que lleguemos a una solución de compromiso entre las características que necesitamos utilizar de la API y el porcentaje de dispositivos en los que nuestra aplicación se podrá instalar.

Versión	Nombre	Nivel API	Distribución
1.6	Donut	4	0.1 %
2.1	Eclair	7	1.2 %
2.2	Froyo	8	2.5 %
2.3-2.3.2	Gingerbread	9	0.1 %
2.3.3-2.3.7	Gingerbread	10	33.0 %
3.2	Honeycomb	13	0.1 %
3.2	Ice Cream Sandwich	15	22.5 %
4.1	JellyBean	16	34.0 %
4.2	JellyBean	17	6.5 %

Tabla 5.1: Distribución de las versiones de la plataforma Android

5.2.3. Tabletas e interfaces táctiles de grandes dimensiones

Las pantallas táctiles de los móviles facilitan la interacción con los usuarios, pero cuanto más grande son las pantallas más sencillo es el proceso. Por lo tanto planteamos este proyecto centrándonos en tabletas de al menos 10 pulgadas, y teniendo en mente dispositivos de mayores dimensiones.

A partir de la versión 3.2 *Honeycomb*, Google hace un profundo rediseño de su plataforma para adaptarse a las tabletas y el resto de dispositivos con altas resoluciones. A partir de esta versión introducen mecanismos específicos para trabajar con pantallas más grandes.

Android permite establecer la versión mínima necesaria para ejecutar la aplicación (*android:minSdkVersion*), en nuestro caso escogeremos *Honeycomb*, pues se trata de la primera versión que soporta directamente tabletas. Sin embargo para de seleccionar la versión para la que diseñaremos la aplicación tendremos en cuenta las características que nos ofrecen *Honeycomb, Ice cream y JellyBean* y el porcentaje de dispositivos que las soportan.

Vemos el gráfico con la distribución de versiones de la SDK entre los dispositivos Android en la figura 5.4.

- *Honeycomb*:
  - Presenta una nueva interfaz de usuario.
  - Gráficos acelerados por hardware.
  - Multiselección
  - Soportada por el 63.1 % de los dispositivos.
- *Ice cream Sandwich*:
  - Unificación de la interfaz de usuario.

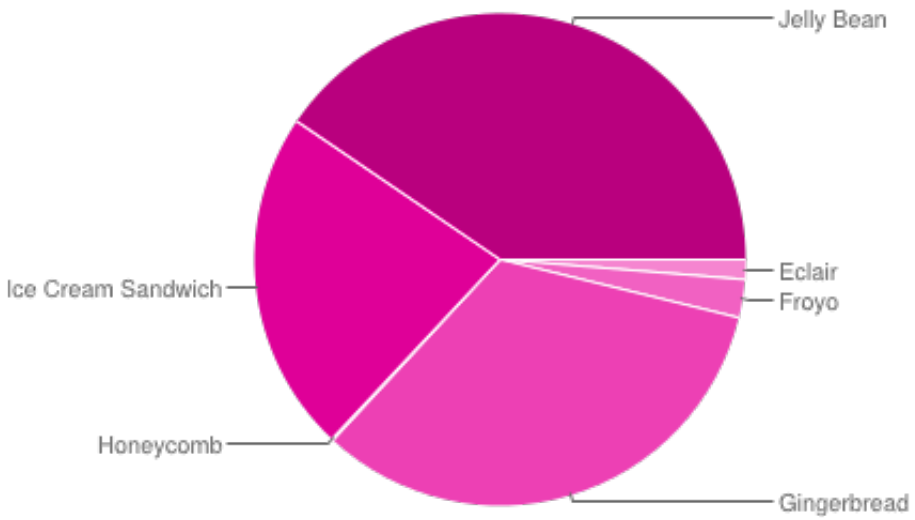


Figura 5.4: Gráfico con la distribución de las versiones de la plataforma Android

- Mejora la aceleración gráfica.
- Mejora el soporte a múltiples pantallas.
- Soportada por el 63 % de los dispositivos.

- *JellyBean*:
  - Simplificada la navegación.
  - Mejora el comportamiento de los dispositivos.
  - Mejora las notificaciones
  - Soportada por el 40.5 % de los dispositivos.

A la vista de esto, optamos por que la aplicación que se genera a partir de una plantilla de informe DICOM-SR fuera dirigida a Ice Cream Sandwich, puesto que ya tiene la mayor parte de las mejoras significativas de Android para tabletas y es soportada por un importante número de dispositivos.

Si en el futuro aparecieran mejoras que creyereamos indispensables siempre podríamos modificar las plantillas con el código que genera la aplicación para utilizar otra versión de Android.

5.3. | Python

La elección óptima de un lenguaje de programación para un proyecto determinado depende de varios factores: el equipo de desarrollo, las características del propio proyecto, las características del lenguaje...

En este proyecto lo fundamental es la generación de código y uno de los puntos fuertes de este paradigma es que el lenguaje de programación que utilizamos para

generar la aplicación y el lenguaje de la propia aplicación no tienen porque coincidir. En nuestro caso el lenguaje de la aplicación viene fijado por el problema que queremos resolver. Sin embargo podremos elegir el lenguaje para generar el código que más se adapte a nuestras necesidades.

Las características que debe tener un lenguaje de programación para ser apropiado para la generación de código son las siguientes [34]:

- Deben ser muy eficientes en la lectura, análisis gramatical y escritura de texto.
- Deben disponer de mecanismos para crear plantillas de texto fácilmente.
- Si soportan de manera nativa XML es útil aunque siempre se pueden incluir librerías externas para leer y manipular XML.
- Deben ser multiplataforma.
- La velocidad de ejecución del lenguaje no es un punto clave porque se ejecutará en producción. Es la aplicación que genere la que debe tener un buen rendimiento.
- Su desarrollo y mantenimiento debe ser sencillo.

Pensando en todos estos puntos decidimos utilizar Python como lenguaje para generar el código de la aplicación Android.

Python es un lenguaje de alto nivel multipropósito y multiplataforma diseñado por Guido van Rossum en 1991. Es un lenguaje que fomenta la escritura de código legible con una sintaxis sencilla. Por su simplicidad muchas veces se utiliza como un lenguaje de scripting pero es un lenguaje suficientemente potente para utilizarlo en otros muchos contextos.

La implementación más común de Python esta escrita en C, se trata de CPython. Es una implementación de código abierto y gratuita, gestionada por la asociación sin ánimo de lucro de Python (*Python Software Foundation*) [52].

Uno de los puntos fuertes de este lenguaje es su amplia librería estándar que nos permite hacer análisis sintáctico, acceder a la API de XML, utilizar expresiones regulares, hacer uso de plantillas,... de manera simple.

Es especialmente apropiado para este problema porque cumple todas las premisas que debe tener un buen lenguaje para la generación de código.

En 2009 Python lanzó su versión 3.0, la primera versión no compatible con las versiones anteriores. A pesar de que introduce un buen número de mejoras, algunas librerías externas todavía no son compatibles con esta versión. Por esto para este proyecto utilizaremos la versión 2.7 de Python. Aunque existen mecanismos [53] para portar automáticamente una aplicación de Python 2.x a Python 3.x.

En este proyecto haremos uso de un paradigma de programación orientado a objetos, así como de las librerías estándar para gestionar la configuración, el análisis sintáctico y la parte más sencilla de las plantillas. La única librería externa que utilizaremos será Jinja, para la gestión de plantillas más complejas. Hablaremos en detalle de esto en los apartados 6.1.3, 8.5.1 y 8.5.2.

6 | Análisis

6.1. | Análisis de requisitos

Afrontamos el desarrollo de la aplicación siguiendo las pautas del esquema tradicional de desarrollo de software en cascada. Por la naturaleza del problema algunas etapas se han reducido para poder dedicar más tiempo al desarrollo de la solución.

En el siguiente apartado describiremos los requisitos de la solución que hemos propuesto. Durante esta fase del proyecto analizaremos las necesidades que tienen los usuarios de nuestro sistema.

Muchos de los requisitos vienen definidos por el propio problema y el resto por la solución que proponemos y que hemos descrito a alto nivel en el apartado 1.2. La complejidad de este proyecto estriba en la propia generación de código a partir de un fichero plantilla basada en DICOM-SR. Los casos de uso y los actores se reducen a la mínima expresión.

Recopilar y analizar los requisitos sienta las bases de nuestro proyecto.

6.1.1. Requisitos funcionales

- A partir de un fichero con la plantilla de un informe médico en formato DICOM-SR, como el fichero incluido en el apéndice B, que pertenezca a una ontología, se generarán los ficheros necesarios para instanciar una aplicación Andoid para esta plantilla que tendrá soporte para 1 o 2 idiomas en función de los idiomas que soporte el fichero DICOM.

- Con los ficheros generados el usuario podrá introducirlos en el esqueleto de una aplicación Android. Obteniendo una aplicación Android funcional.

- El entorno para la ejecución del script que genera el código y el de la aplicación esqueleto de Android deben ser fácilmente replicables, para que el usuario pueda genererar sus propias aplicaciones en distintas plataformas.

6.1.2. Requisitos no funcionales

- La aplicación que genera el código debe ser lo más genérica posible para así abarcar la mayor parte de informes DICOM-SR posibles.

- La interfaz de la aplicación se construirá mediante piezas de código intercambiables, lo que permitirá al usuario modificar la interfaz para que se acople a su metodología de trabajo.
- La interfaz debe ser simple e intuitiva. La interfaz se construirá siguiendo un sistema de cuadrícula dónde se dará especial importancia a la tipografía y al esquema de color, para que con los mínimos elementos consigamos una interfaz de usuario amigable.
- El usuario podrá configurar ciertos aspectos del script que genera al código de la aplicación Android. Como por ejemplo el directorio donde estos ficheros se almacenarán, cual es el lenguaje principal, ...
- La aplicación deberá facilitar la introducción de datos

6.1.3. Requisitos de implementación

- La aplicación que genere el código se desarrollará en Python 2.7.x y seguirá las guías de estilo PEP-8 para la escritura de código y PEP-257 para la documentación.
- La aplicación final Android se desarrollará para la versión Ice Cream Sandwich (3.2) o superior. Deberá soportar tamaños de pantalla de 10 pulgadas o más y alta resoluciones.
- La aplicación Android se ajustará a las recomendaciones de diseño que podemos encontrar en las guías para diseñadores y desarrolladores de la web de Android Developers.
- Se encapsularán las librerías de las que dependa el proyecto haciendo uso de *virtualenv*, que permite crear entornos de desarrollo para python.

6.2. | Actores

En la tabla 6.1, podemos ver la definición formal de los usuarios que emplearán el sistema que vamos a desarrollar.

Únicamente tenemos un actor y un caso de uso, que veremos en el apartado 6.3, porque la generación de código debe necesitar la mínima interacción de los usuarios. Sin embargo por coherencia con la metodología de desarrollo realizamos el modelado de los actores y de los casos de uso.

Actor	Usuario	Identificador	USER1
Descripción	Usuario que generará automáticamente aplicaciones Android.		
Características	El usuario dispondrá del entorno python y de las aplicación esqueleto de Android, así como de un fichero con la plantilla de un informe DICOM-SR.		
Relaciones			
Referencias	CONF,GENCOD,INSAND		
Autor	Mayte Giménez	Fecha	16/08/2013
Version	v1.0		

Atributos		
Nombre	Descripción	Tipo
Vacío	Vacío	Vacío

Comentarios
Aunque para ejecutar la aplicación el usuario no requiera ningún tipo de conocimiento del sistema generador de código o de los informes médicos, el usuario debe disponer de un fichero plantilla basado en DICOM-SR correctamente elaborado. Además debe de tener la aplicación esqueleto Android para instanciar la aplicación final. Además debe disponer de un dispositivo Android con la versión 3.2 o superior.

Tabla 6.1: Actores del sistema

6.3. | Casos de uso

Caso de uso	Configuración del generador de código	Identificador	CONF
Actores	Usuario		
Tipo	Primario		
Referencias			
Precondición	El usuario dispone de un fichero para generar la aplicación que sigue el estándar DICOM-SR. Y tiene el entorno de desarrollo para la aplicación generadora de código preparado.		
Postcondición	Los ficheros de configuración del generador quedarán personalizados según las necesidades del usuario.		
Autor	Mayte Giménez	Fecha	16/08/2013
Version	v1.0		
Propósito	Configurar las opciones parametizables del generador de código de la aplicación Android.		
Resumen			

Existen un número de opciones que el usuario puede modificar como son: las cadenas de texto que aparecen en los idiomas soportados, la ruta dónde se creará el código generado automáticamente y las ontologías soportadas. Los ficheros de configuración permiten que el usuario edite estas opciones para que el código generado se adapte a sus necesidades.			
Curso normal			
1	El usuario modifica alguno o varios de los parámetros de configuración		
		2	El sistema lee la configuración y cuando el usuario ejecute el script de generación de código aplicará las modificaciones.
Curso alterno			
2b	Si alguno de los parámetros de configuración no es válido, este volverá al valor por defecto.		

Tabla 6.2: Casos de uso: Configuración del generador de código

Caso de uso	Generación del código de la aplicación Android		Identificador		GENCOD
Actores	Usuario				
Tipo	Primario				
Referencias					
Precondición	El fichero para generar la aplicación debe seguir el estándar DICOM-SR y pertenecer a las ontologías soportadas en la configuración.				
Postcondición	Se generarán los ficheros necesarios para instanciar la aplicación Android esqueleto.				
Autor	Mayte Giménez	Fecha	16/08/2013	Version	v1.0
Propósito	Generar parte del código necesario de una aplicación Android.				
Resumen					
El usuario ejecutará el script pasándole el fichero plantilla basado en DICOM-SR y el método de internacionalización de este fichero (i18n o default) y el sistema generará los ficheros de código necesarios para instanciar la aplicación Android.					
Curso normal					

1	El usuario ejecuta el script generador de código. Los argumentos serán el fichero plantilla basado en DICOM-SR y el método de internacionalización		
		2	El sistema comprueba que el fichero existe y que el método de internacionalización es correcto
		3	El sistema realiza el análisis sintáctico del fichero DICOM-SR
		4	El sistema carga en una estructura de árbol la información del fichero DICOM-SR
		5	El sistema genera las cadenas de internacionalización necesarias para la aplicación Android
		6	El sistema genera los ficheros XML para la interfaz de usuario.
		7	El sistema genera las clases Java que modelan el informe médico
		8	El sistema genera las clases Java que controlan la aplicación Android.
		9	El sistema genera el manifiesto de la aplicación Android.
Curso alterno			
2b	Si el sistema detecta que el método de internacionalización no es correcto usará el método por defecto, descartando la información del segundo idioma si existiera.		
5b	Si los ficheros XML de las cadenas de internacionalización ya existen en la ruta especificada no se vuelven a generar.		
6b	Si los ficheros XML de la interfaz de usuario ya existen en la ruta especificada no se vuelven a generar.		
7b	Si los ficheros Java con el modelo ya existen en la ruta especificada no se vuelven a generar.		
8b	Si los ficheros Java que controlan la aplicación Android ya existen en la ruta especificada no se vuelven a generar.		
9b	Si el manifiesto Android ya existe en la ruta especificada no se vuelven a generar.		

Tabla 6.3: Casos de uso: Generación de código

Caso de uso	Instanciación de la aplicación Android		Identificador		INSAND
Actores	Usuario				
Tipo	Primario				
Referencias					
Precondición	El usuario dispone de los ficheros generados a partir de un fichero plantilla basado DICOM-SR y la estructura de la aplicación Android que los contendrá.				
Postcondición	Se crea una aplicación Android para el informe médico DICOM-SR a partir del cual se habían generado los ficheros.				
Autor	Mayte Giménez	Fecha	16/08/2013	Version	v1.0
Propósito	Disponer de una aplicación Android que permita al usuario introducir los datos de informes DICOM-SR de la ontología instanciada.				
Resumen					
Con el esqueleto de la aplicación Android para este proposito y todos los ficheros necesarios para personalizarla, el sistema instancia la aplicación Android para esta plantilla específica.					
Curso normal					
1	El usuario llama al script con la ruta raíz dónde se encuentran los ficheros generados y la ruta raíz de la aplicación Android.			El sistema comprueba que los ficheros generados y la aplicación Android esqueleto existan.	
			3	El sistema copia los ficheros en la ruta que les corresponde dentro de la aplicación Android.	
Curso alterno					
2b	Si alguna de las rutas no existe o falta algún fichero necesario la ejecución se aborta.				

Tabla 6.4: Casos de uso: Instanciación de la aplicación Android

7 | Diseño

7.1. | Arquitectura de la solución

En este apartado continuaremos profundizando en los detalles de la solución propuesta en el apartado 1.2. Describiremos tanto la arquitectura de la aplicación generadora del código como de la aplicación esqueleto de Android que instanciaremos con la plantilla de un informe DICOM-SR.

7.1.1. Aplicación Android

Nuestro objetivo último es disponer de una aplicación Android para una determinada ontología de informe médico, que facilite al personal clínico la tarea de rellenar los informes de los pacientes siguiendo el estándar DICOM-SR. Por lo tanto el código que generaremos deberá contener todos los componentes de una aplicación Android.

Justificación para desarrollar un aplicación nativa para Android

En este proyecto generamos una aplicación nativa para Android. Antes de entrar en profundidad a describir la anatomía de una aplicación Android, dedicaremos unas líneas a justificar la decisión de escribir una aplicación nativa para Android en lugar de desarrollar una aplicación genérica en HTML5, o utilizando frameworks como Phonegap [54], Titanium [55] o Corona [56] para crear una aplicación híbrida o ya que nuestro generador de código está escrito en Python porque no utilizar Python también en Android mediante el framework Kivy[57].

Comenzaremos por argumentar esta última opción ya que es también la más sencilla de rebatir. Que escribamos el generador de código en un lenguaje no implica que el software generado deba estar escrito en ese mismo lenguaje, de hecho esta disociación es un punto fuerte de la generación de código, como ya explicamos en el apartado 4.2.

La utilización de Kivy está justificada cuando queremos hacer una aplicación multiplataforma, para dispositivo móviles y de escritorio, sin necesidad de reescribir del código Kivy. Aporta la potencia y simplicidad de Python y está recomendado para el desarrollo de juegos o aplicaciones con gran carga gráfica, ya que en el caso de aplicaciones genéricas la interfaz que crea este framework no es la más apropiada.

En nuestro caso, la aplicación final es un formulario, y una de las premisas que nos hemos marcado es que sea intuitiva y fácil de utilizar, por lo tanto los elementos de la

interfaz no deberán sorprender al usuario y la interacción con el usuario debe ser ágil. Generar código en Python supone el mismo esfuerzo que generarlo en cualquier otro lenguaje. Por ello esta opción no nos ofrece ventajas adicionales así que descartamos esta opción.

La siguiente opción sería desarrollar una aplicación en HTLM5. Las ventajas principales que tiene esta opción es la rapidez en el desarrollo y que no es necesario pasar un proceso de validación de la misma. Por otra parte se resiente la velocidad de ejecución y no se tiene acceso a todas las funcionalidades de los teléfono o tabletas para las que se desarrolla.

No vamos a necesitar funcionalidades complejas pero el hecho de que la velocidad de ejecución se reduzca respecto a las aplicaciones nativas empeora la usabilidad y por tanto el índice de aceptación de los usuarios y este si es un punto relevante para nuestro objetivo, por lo tanto también descartamos esta opción.

Finalmente quedaría la opción de desarrollar una aplicación híbrida. Esta opción puede hacer uso de las funcionalidades de las tabletas y puede funcionar a la misma velocidad que una aplicación nativa, además tiene aporta el beneficio de que las aplicaciones sobre estos frameworks son más sencillas de desarrollar, pero aportan las complicaciones propias de instalar y trabajar con el framework.

Nuestro objetivo es que con un fichero plantilla basado en DICOM-SR se genere automáticamente una aplicación Android. El esqueleto de la aplicación en Android podemos tenerlo preparado e instanciar la aplicación para el fichero concreto, en cambio incluir un framework introduce un nivel más que nos complica la generación automática. Y respecto a la complicación del desarrollo, como el código que generamos lo hacemos mediante generación automática guiada por modelos, desarrollar una aplicación en Android no implica un esfuerzo mucho mayor que hacerlo para algún framework para aplicaciones híbridas.

Por lo tanto la opción que más se adapta a las necesidades concretas de este proyecto es el desarrollo de una aplicación nativa para Android. Ya que la aplicación que obtenemos como resultado está optimizada para dispositivos Android y es capaz de aprovechar todas las funcionalidades que necesitemos. Además crear un framework sobre el que instanciar los informes DICOM-SR es relativamente sencillo y el desarrollo no es más complicado que optando por otras soluciones, gracias a la generación automática de código.

Esquema de una aplicación Android

Las aplicaciones Android siguen normalmente el patrón de desarrollo Modelo-Vista-Controlador(MVC). El modelo y los presentadores o controladores se programan en Java y para definir la interfaz de usuario se utilizan ficheros XML.

Una aplicación que sigue este patrón de diseño se compone por:

- **Vista**: Contiene la interfaz de usuario y las cadenas de texto para internacionalizar dicha interfaz.

- **Modelo**: Es la información del sistema, encapsula el estado del sistema.
- **Controlador**: Responde a los eventos generados por la vista e interactua con el modelo.

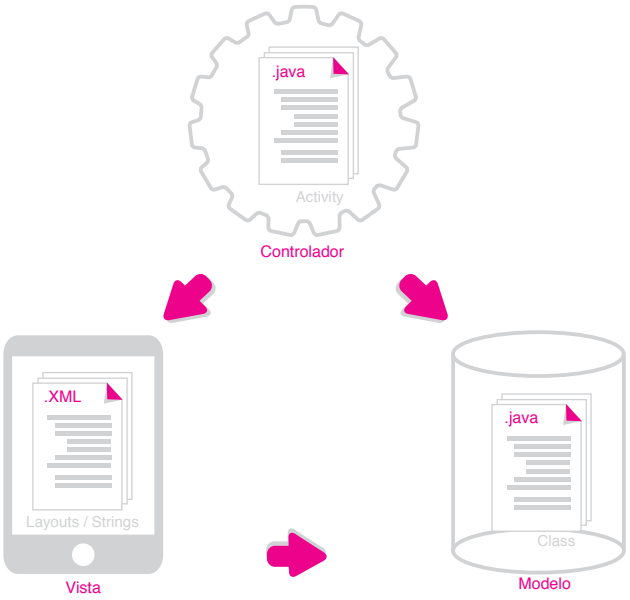


Figura 7.1: Generación parcial de clases

En la figura 7.1, vemos el esquema del patrón de desarrollo MVC aplicado en una aplicación Android.

Las partes de las que se compone una aplicación Android son las siguientes:

- **Vista**: La vista de la aplicación se desarrolla mediante ficheros XML. La vista en Android se compone fundamentalmente por dos elementos: los ficheros XML que definen la interfaz de usuario, que podemos encontrar en $\backslash res \backslash layout$ y por otra parte tenemos los ficheros XML con las cadenas de texto que se utilizarán en la interfaz, correctamente internacionalizadas y almacenadas en $\backslash res \backslash values$. Adicionalmente se puede definir estilos para los elementos de la interfaz gráfica, análogamente a como se hace en CSS, estos ficheros de estilo los podemos encontrar también en $\backslash res \backslash values$. Para soportar distintos tipos de pantalla en Android, las imágenes y los estilos se organizan dentro de carpetas determinadas en la aplicación Android y en función de la resolución del dispositivo se cargarán unas u otras.

- **Modelo**: El modelo en una aplicación Android no difiere del modelo en otro lenguaje. Se trata de una serie de clases con la información del sistema. En nuestro caso serán las clases que contengan la información del informe médico. Encapsularemos el modelo dentro del paquete *model*.
- **Controlador**: Finalmente los controladores en las aplicaciones Android son las Actividades (*Activity*). Las actividades gestionan la interfaz de usuario, interactuan con el usuario y acceden al modelo bien para mostrarlo en la vista o para modificarlo.

Creamos la estructura de una aplicación Android pero sin contenido. Tendremos una serie de directorios, que podemos ver en la figura 7.2, que sigen la organización de vista, modelo y controlador que Android es capaz de interpretar como una aplicación nativa correcta. Esta estructura que hemos creado debemos completarla con el código generado automáticamente a partir de un fichero plantilla basado en DICOM-SR.

7.1.2. Aplicación generadora de código

La aplicación que generará el código a partir de un fichero con una plantilla DICOM-SR está escrita en Python. La aplicación la organizamos siguiendo el esquema de ficheros de una librería de Python, para poder utilizarla en un futuro dentro de en un framework más completo.

El código de la aplicación se estructura de manera lógica en función de las tareas a realizar. Las parte más importantes son las siguientes:

- *Análizador sintáctico*: realiza el análisis sintáctico y almacena la información en una estructura de datos en memoria.
- *Generador de código*: a partir de la plantilla basada en DICOM-SR cargada en memoria genera el código necesario para instanciar la aplicación Android.
- *Gestión de la configuración y los ficheros*: lee la configuración del usuario y la aplica en la generación de código.
- *Modelo del informe*: Estructura de datos que almacena en memoria el informe DICOM-SR.

Estas secciones lógicas las encontraremos en el código de la aplicación como aparecen en la figura 7.3.

Para llamar al generador de código, *fromDicomAndroid.py*, y generar los ficheros necesarios para crear una aplicación Android, lo ejecutaremos pasando como parámetros el fichero con la plantilla DICOM-SR y el método de internacionalización empleado. Vemos un ejemplo de como se lanzará el generador de código en el segmento de código 7.1.

```
1 ./fromDicomtoAndroid.py ../../res/xml/02_Mamography_Unix.xml i18n
```

Código 7.1: Ejecución del generador de código

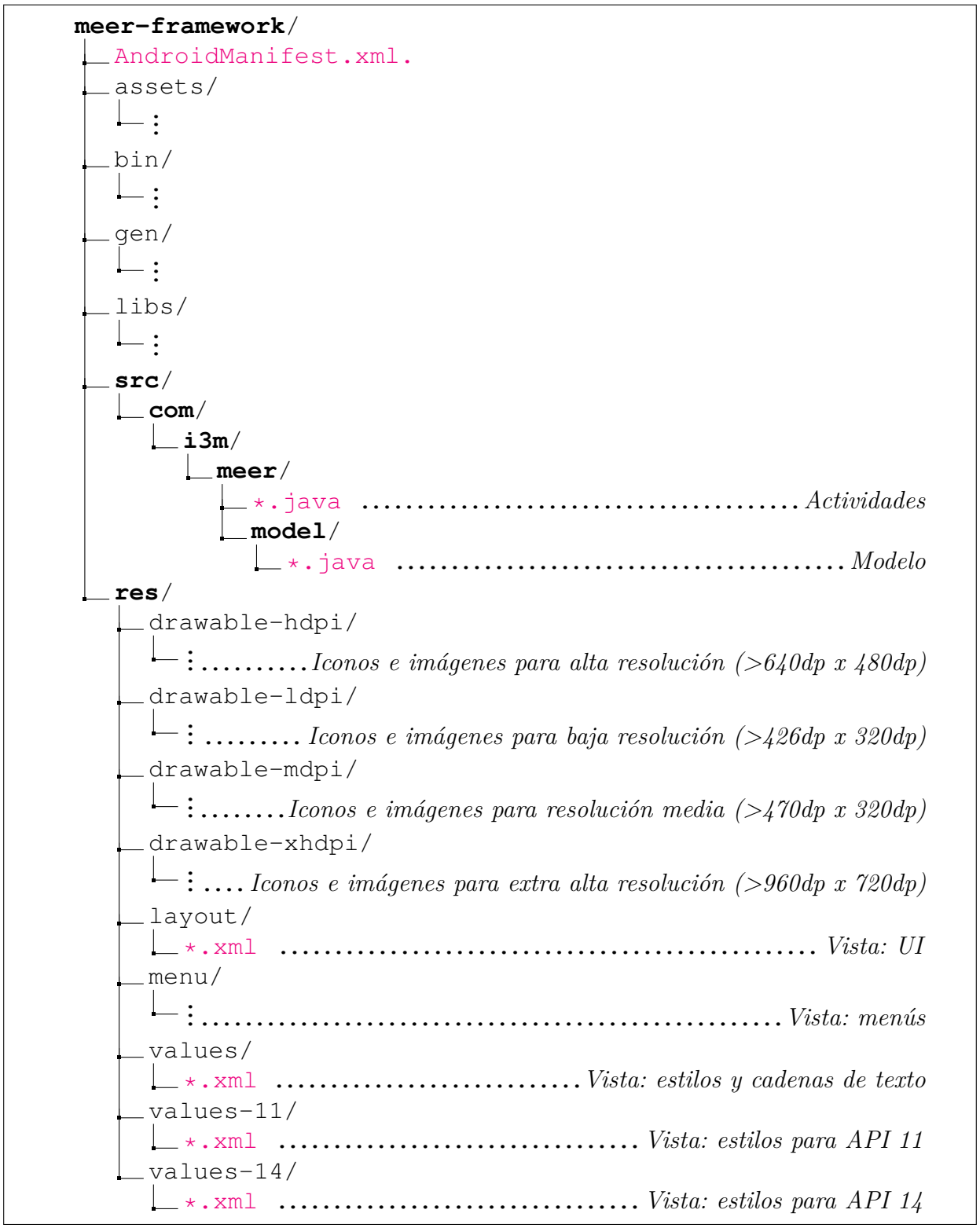


Figura 7.2: Estructura de ficheros de la aplicación framework Android

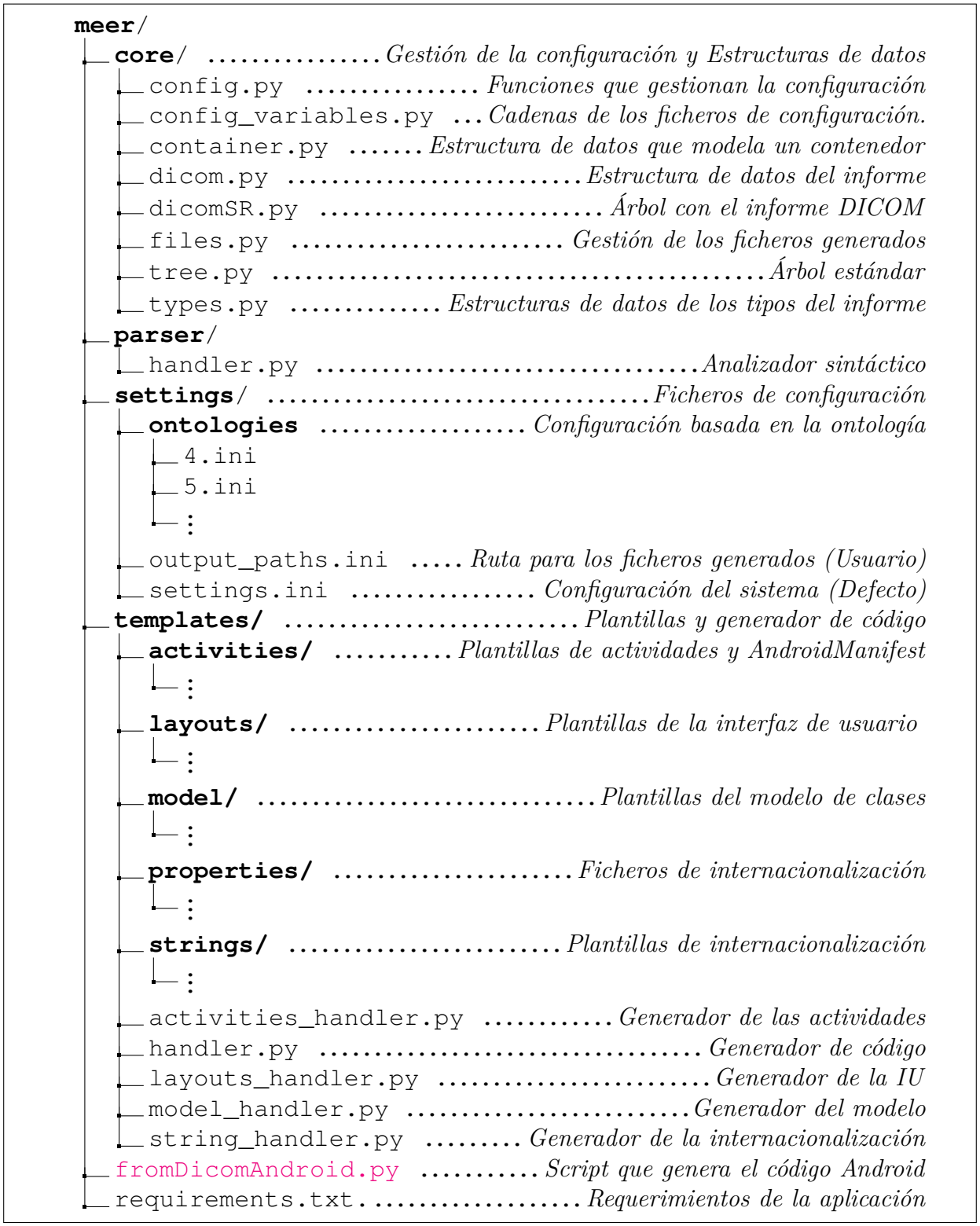


Figura 7.3: Estructura de ficheros de la aplicación Python generadora de código

Diagrama de clases

A continuación veremos los diagramas de clases de las secciones más relevantes del código.

El código que gestiona la configuración y el generador de código agrupan un conjunto de funciones y veremos su implementación en el apartado 8.5.

Comenzaremos por ver en la figura 7.4, la clase creada para realizar el análisis sintáctico, extiende de *xml.sax.handler.ContentHandler*, que es el módulo de la librería estándar de Python que implementa la API simple de XML en Python.

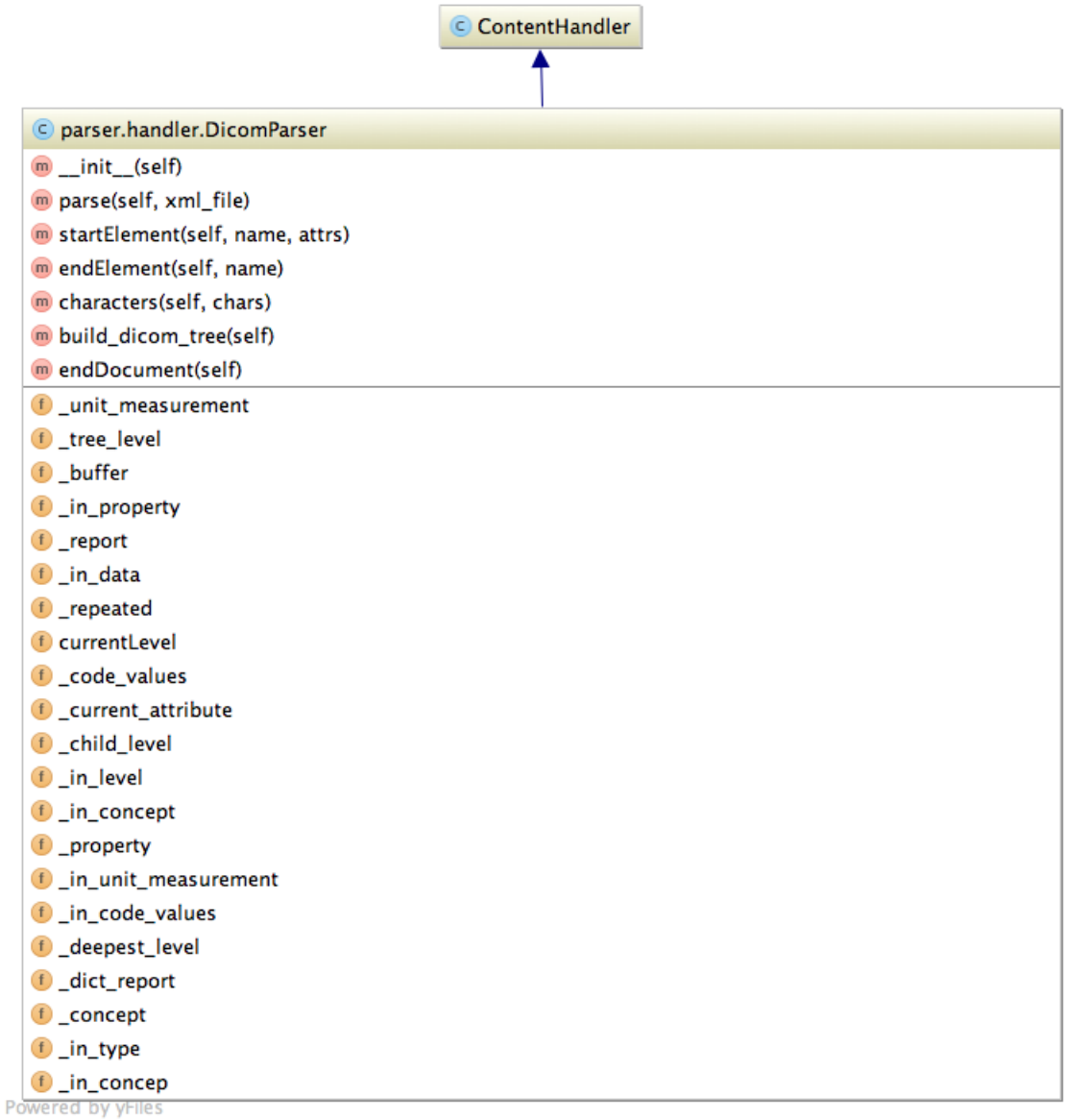


Figura 7.4: Diagrama de clase UML del analizador sintáctico.

Al emplear un analizador de tipo SAX, necesitaremos una estructura de datos que a medida que leemos el fichero DICOM-SR almacene la información del mismo. La figura 7.5 muestra el diagrama de esta clase. Por simplicidad el informe se modela mediante una lista de contenedores con la información del contenedor DICOM. Cada contenedor guarda información acerca de que contenedor es su padre.

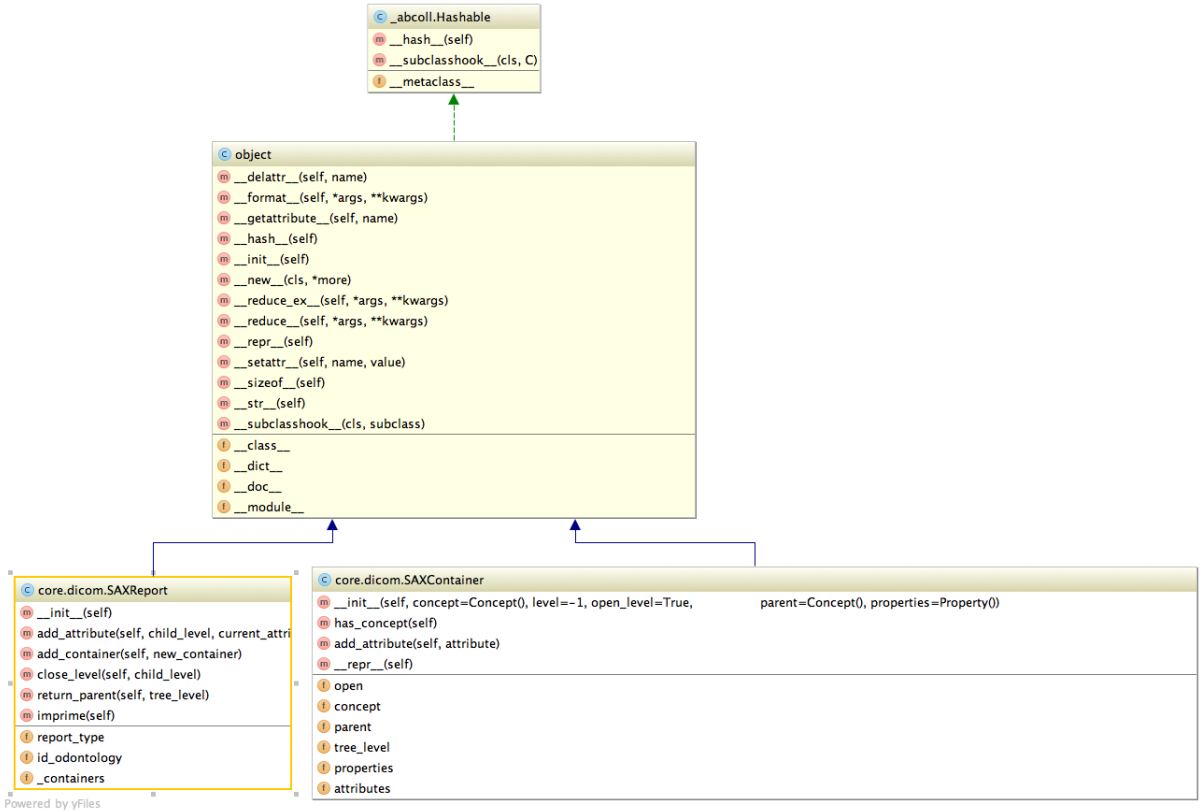


Figura 7.5: Diagrama de clase UML de la estructura de datos del informe DICOM.

Una vez terminado el análisis sintáctico, tener el informe como una lista de contenedores ralentiza el proceso de extraer información para generar el informe, así que guardamos la información dentro de un árbol.

Para esto disponemos de una clase árbol genérica que podemos ver en el diagrama 7.6. Los nodos de este árbol estarán formados por contenedores con la información de un contenedor de tipo DICOM. El diagrama UML que corresponde a estos contenedores es el de la figura 7.7. Todo el informe se encapsula dentro de la clase DicomSR y podemos ver el diagrama de esta clase en la figura 7.8.

Terminamos esta sección con el diagrama UML de los tipos de datos soportados en DICOM-SR. Dentro de los contenedores, almacenamos conceptos DICOM, y son estos conceptos los que se modelan en el diagrama de la figura 7.9.

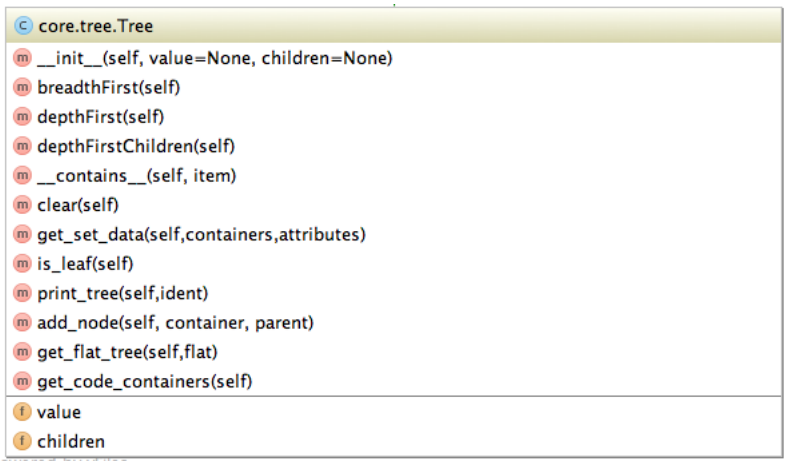


Figura 7.6: Diagrama de clase UML del árbol genérico

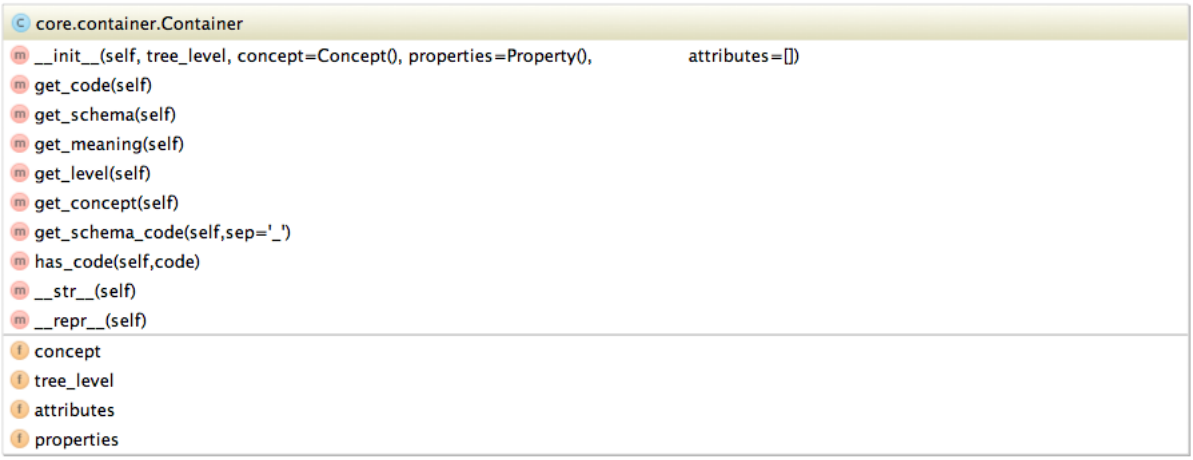


Figura 7.7: Diagrama de clase UML del contenedor DICOM

7.2. | Diseño de la experiencia de usuario

En el apartado 4.3 ya hemos hablado de el impacto que tiene el diseño de las interfaces que realicemos en la aceptación de las aplicaciones por parte de los usuarios.

Para que la aplicación Android que generermos automáticamente sea gráficamente atractiva para los usuarios finales, deberemos diseñar correctamente toda la experiencia de usuario.

En la industria actual se intenta involucrar a todos los responsables de un desarrollo para lograr que la experiencia del usuario sea satisfactoria. Al tratarse de un proyecto personal trataremos, en la medida de lo posible, de asumir todos los roles que entran en juego al diseñar una experiencia de usuario.

En el esquema de la figura 7.10 podemos ver los roles que intervienen en el diseño de una experiencia de usuario satisfactoria.

Para diseñar la experiencia de usuario debemos analizar el problema desde el punto

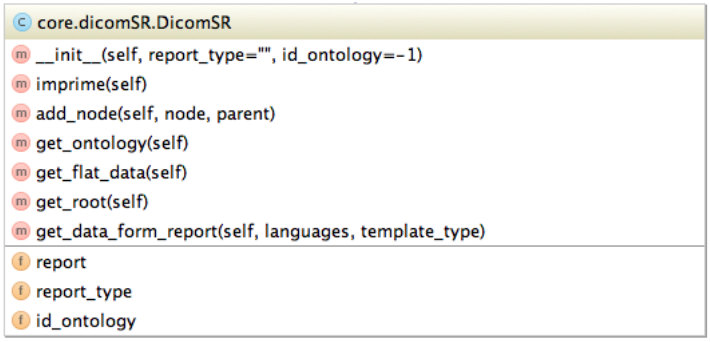


Figura 7.8: Diagrama de clase UML de la estructura de datos del informe DICOM utilizando un árbol.

de vista del usuario, pensar el uso que el usuario hará de la herramienta que vamos a crear y como se integrará en su rutina. Analizamos el trabajo facultativo, así como los datos que debe contener un informe médico estructurado, a partir de estos datos diseñaremos la interacciones que realizará el usuario con la aplicación.

Después de varias iteraciones con los prototipos llegamos a la conclusión de que la forma más coherente y genérica de estructurar la aplicación es que cada pantalla corresponda a los datos de un nivel del informe estructurado. Además en cada una de estas pantallas el usuario podrá acceder a un resumen del informe que esta introduciendo, así como la posibilidad de navegar entre niveles de manera intuitiva y acceder a validar la información en cualquier momento.Podemos ver la primera iteración del diseño de la experiencia en el apartado 7.2.2.

Simultáneamente al diseño de la interacción, diseñaremos la interfaz de usuario. Veremos en el apartado 7.2.1, la guía de estilo que hemos definido para este proyecto.

7.2.1. Guía de estilo

Android es muy versátil en lo que a desarrollo de interfaz de usuario se refiere, lo cual es algo positivo pero que si no se diseña con cuidado la interfaz pude dar lugar a aplicaciones inconsistentes desde el punto de vista del diseño. Para incentivar el diseño correcto de aplicaciones, Android tiene una guía exhaustiva de diseño de interfaces para sus dispositivos [58] que en este proyecto seguiremos.

Debido a la naturaleza de la aplicación y que uno de nuestros objetivos es que la aplicación sea rápida y simple de utilizar, prescindiremos de las imágenes en la medida de lo posible. Todo el peso del diseño recaerá en la tipografía, el esquema de color y la estructura interna que organiza estos elementos, es decir, la cuadrícula.

El sistema de cuadrícula

Aunque el diseño de interfaces utilizando la cuadrícula es un método que nos pueda parecer actual, se trata de un sistema desarrollado después de la segunda guerra mundial y que se integra en el diseño internacional o diseño suizo.

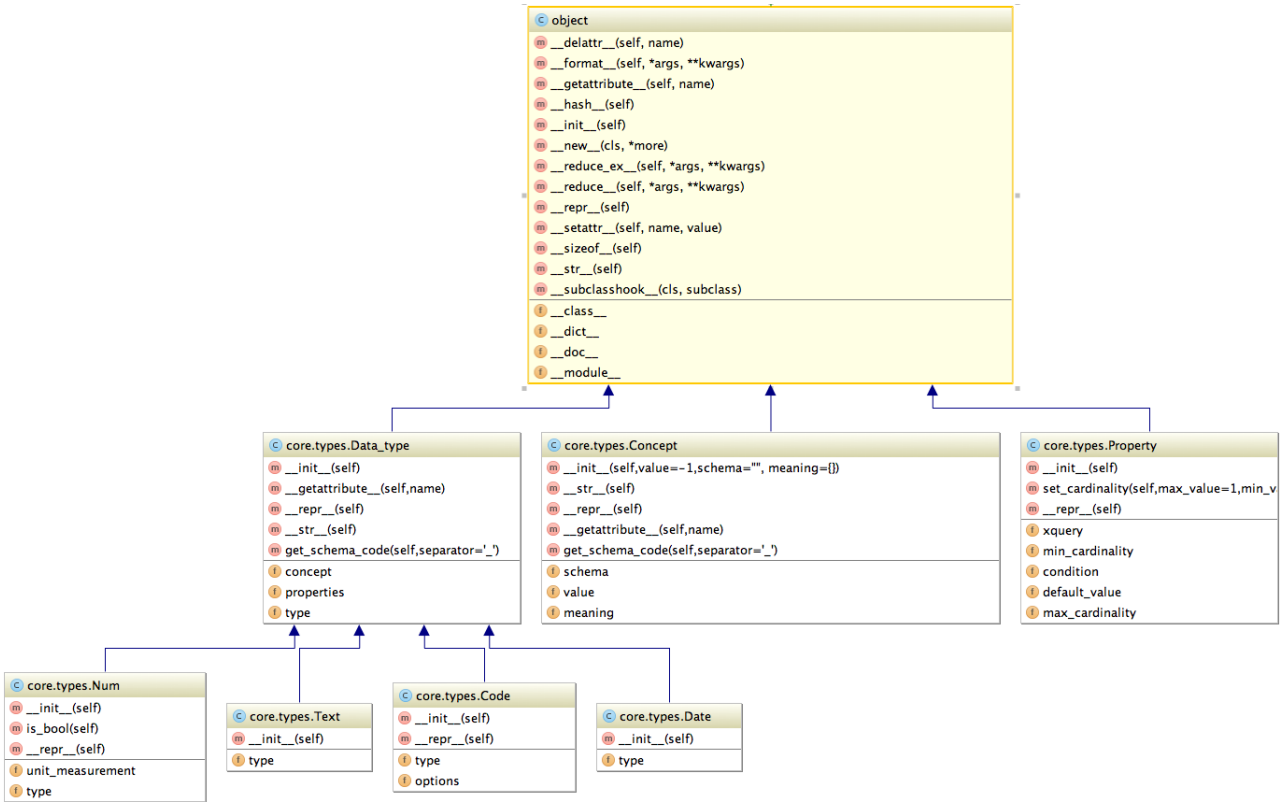


Figura 7.9: Diagrama de clase UML de los conceptos DICOM.

Se trata simplemente de un diseño en el que una cuadrícula organiza los elementos del diseño, en nuestro caso de la interfaz de usuario. Entre los principales puntos fuertes que tiene este sistema es que ayuda a la navegación, capta la atención del usuario facilitando la lectura y escritura y son diseños estables y previsibles lo que hace que sea más sencillo a los usuarios interactuar con estos diseños.

En la figura 7.11, vemos un ejemplo de la cuadrícula superpuesta sobre una de las pantallas de la interfaz del prototipo.

La tipografía

Con la versión 3.2 de Android se introduce *Roboto*, una tipografía de palo seco diseñada específicamente para dar respuesta a los requisitos de los dispositivos con alta resolución de pantalla.

Se trata una tipografía que favorece la lectura en dispositivos digitales por no tener remates, porque la mancha está balanceada y los ojos de los caracteres son amplios.

El esquema de color

Finalmente hablaremos del esquema de color escogido para la aplicación. Aunque lo general para las aplicaciones dentro del ámbito médico es emplear colores fríos en los

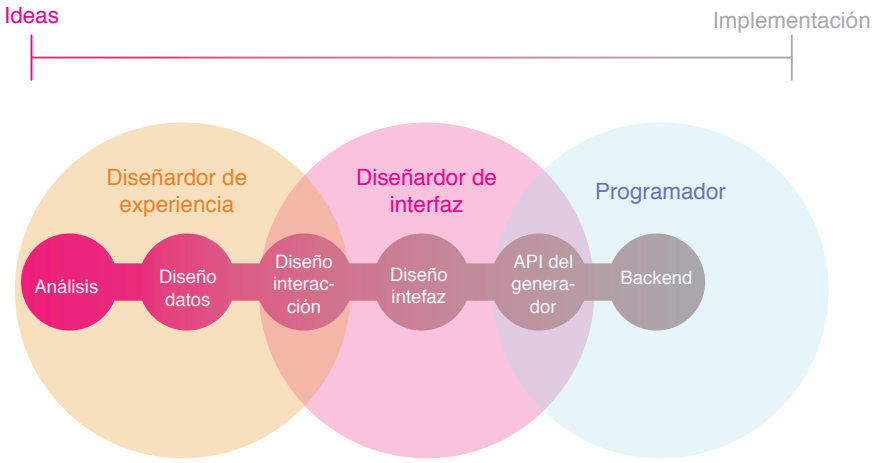


Figura 7.10: Roles que intervienen en el desarrollo de aplicaciones útiles

que predominan los azules y los verdes, porque se trata de colores que sugieren espacios asépticos y tranquilos.

Empleamos turquesas y grises en la paleta básica y utilizamos como color secundario dos tonos naranjas. La paleta tiene una componente cálida que se aleja de las recomendaciones, pero que confiere a la aplicación un carácter más orgánico, lo que suele aumentar el nivel de aceptación de los usuarios.

Se evitan los grandes contrastes, evitando los blancos y los negros puros en la interfaz, lo que reduce la fatiga visual.

Android permite definir paletas de colores como un elemento de los recursos más. La paleta que definimos para la aplicación es la que vemos en el fragmento de código 7.2.

```
1
2 <?xml version="1.0" encoding="utf-8"?>
3 <resources>
4     <color name="dirtyWhite">#F6F8F8</color>
5     <color name="turquoise">#50BAB5</color>
6     <color name="babyBlue">#A2D2D5</color>
7     <color name="orange">#EB6E51</color>
8     <color name="dust">#DDC0B2</color>
9     <color name="darkGrey">#686868</color>
10    <color name="lightGrey">#A7A7A8</color>
11    <color name="black">#000000</color>
12 </resources>
```

Código 7.2: Esquema de color

En la figura 7.12 encontramos un resumen de las características del estilo de la aplicación.

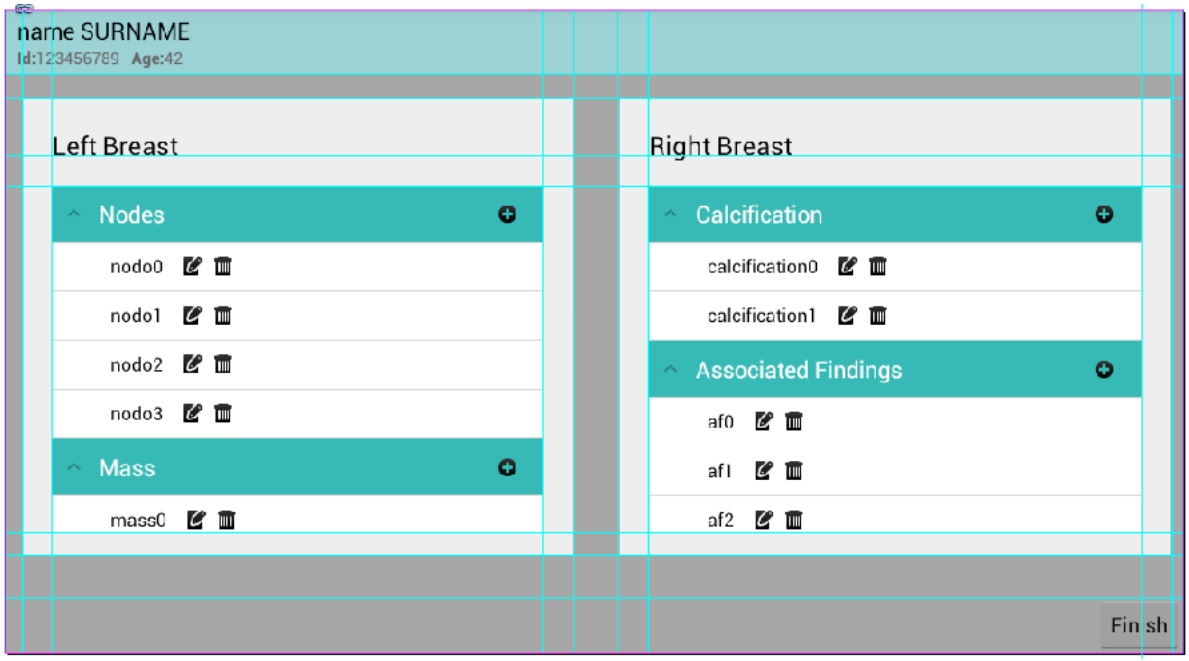


Figura 7.11: Sistema de cuadrícula

7.2.2. Diseño de las interacciones

Para diseñar correctamente las interacciones necesitamos conocer el uso que le dará el usuario.

Si realizáramos el diseño para un informe general no seríamos capaces de captar las necesidades del usuario. Por lo tanto tomamos como ejemplo una mamografía y planteamos las interacciones que el usuario deberá llevar a cabo para introducir el informe de una mamografía.

Los bocetos que veremos de estos diseños no siguen las guías de estilo que definimos en el apartado 7.2.1 porque lo que se pretende validar en este punto es la interacción no la interfaz, de modo que se aísla el diseño de la interfaz de usuario del diseño de las interacciones.

Comenzaremos por la pantalla de inicio que podemos ver en la figura 7.13. En la parte superior, siempre visible tendemos los datos del paciente para que el facultativo en todo momento sepa fácilmente de quien es el informe que está rellenando.

También en la parte superior encontramos un menú que al desplegarlo nos permite ver un resumen del informe que se está codificando, en la imagen 7.14 vemos esta funcionalidad, de este modo el facultativos puede comprobar en todo momento los datos ya introducidos.

En la parte inferior tendremos dos botones para validar y finalizar el informe. Al inicio estará habilitado el botón de *validar*, y cuando el usuario introduzca datos podrá validar el informe y tras la validación se habilitará el botón para finalizar que creará un informe DICOM-SR con los datos del estudio introducidos.

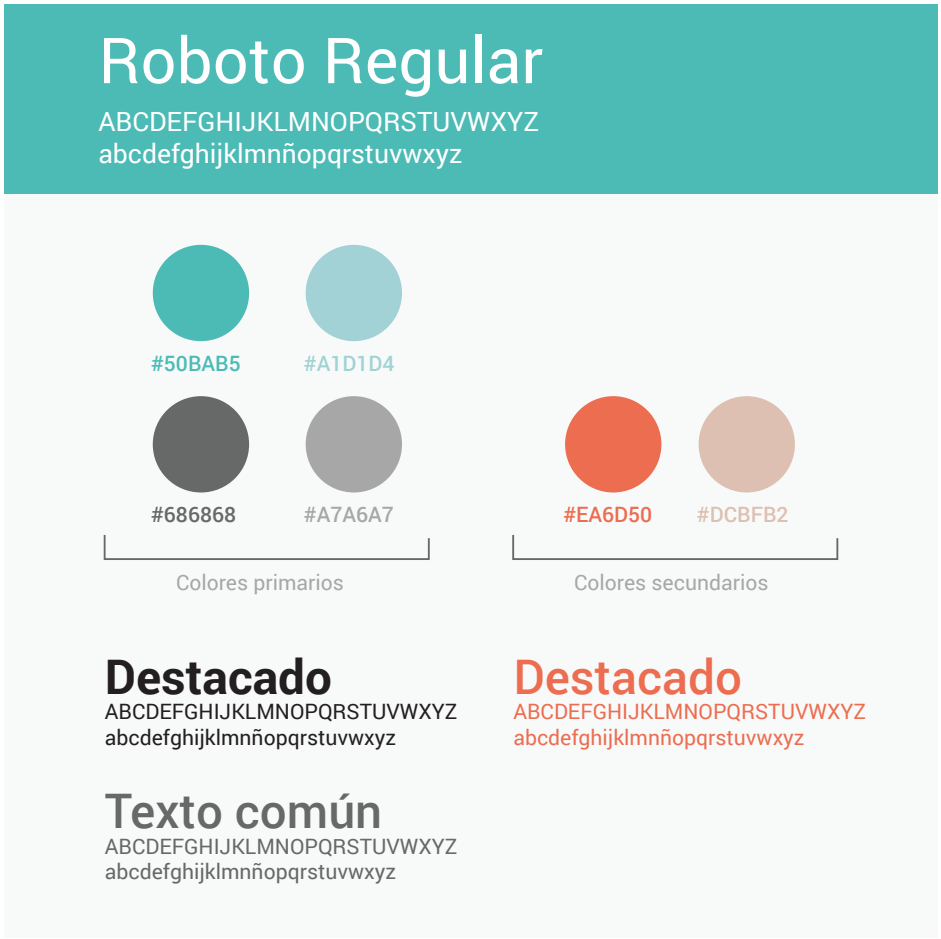


Figura 7.12: Guía de estilo

En los dos fragmentos definidos por la cuadricula estructuraremos la información que será diferente para cada nivel del informe médico.

Un flujo de trabajo normal comenzaría como se muestra en la figura 7.13. A la izquierda vemos los datos que se deben introducir en el nivel superior (identificador del paciente, identificador del estudio,...) y a la derecha veremos un resumen de las lesiones de la mamografía ya codificadas. Como estamos en el inicio y el usuario no ha introducido lesiones, este resumen está vacío.

Para codificar una nueva lesión simplemente deberá pulsar en el botón con el texto *Añadir Lesiones*.

Esta acción llevará al usuario a una pantalla similar a la que vemos en la figura 7.15, donde vemos una lista de las lesiones que puede contener un informe de tipo mamografía. Como hay dos órganos que se estudian en este informe, las anomalías de cada órgano se organizan dentro de una cuadrícula.

Al lado de cada elemento del listado hay un icono para añadir una nueva lesión de este tipo. Cuando el usuario pulse sobre un icono para añadir por ejemplo una

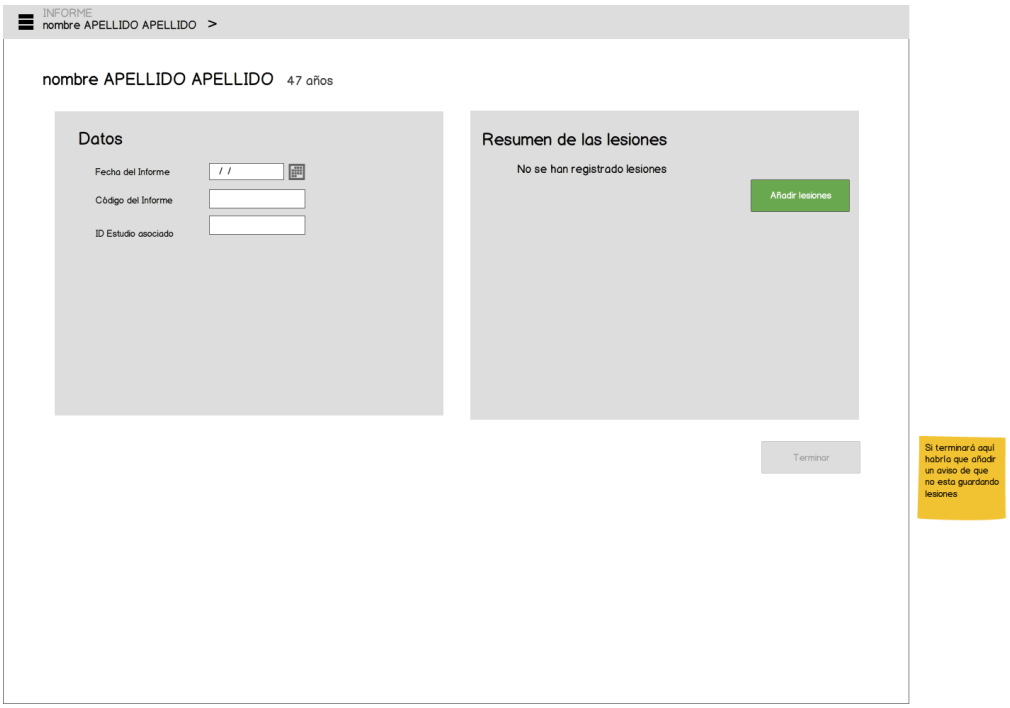


Figura 7.13: Mockup: Inicio de la aplicación

desectructuración en la mama derecha, la aplicación le llevará hasta un pantalla similar a la que podemos ver en la figura 7.16.

En esta figura vemos que las dos cuadrículas albergan los atributos que definen una desestructuración. El personal clínico llenará estos campos con los datos de un informe concreto y cuando haya terminado de introducir los datos podrá guardarlos, validarlos y descartarlos.

Permitimos que la validación se pueda realizar en todos los niveles del informe, facilitando el uso de la aplicación a los usuarios.

Tras introducir unas cuantas lesiones, el usuario las verá en el árbol que contiene las lesiones para cada órgano, agrupadas por el tipo de lesión, como podemos ver en la figura 7.17.

Al lado de cada lesión vemos un pequeño icono que abrirá una lista desplegable con las acciones que podemos realizar sobre esa lesión en concreto. De este modo la lista de lesiones se mantiene lo más compacta posible, y sigue siendo fácil realizar acciones sobre una lesión en concreto.

También podemos ver en esta imagen que estamos comentando que si la cardinalidad de un tipo de lesión está completa desaparece el icono para incluir nuevas lesiones no permitiendo añadir nuevas lesiones de este tipo.

Al pulsar sobre el icono situado a la derecha de la lesión se despliegan las acciones disponibles. Como podemos ver en la figura 7.18, las acciones disponibles en este caso son: editar la lesión y eliminarla. Si el usuario decide borrar una lesión verá un mensaje

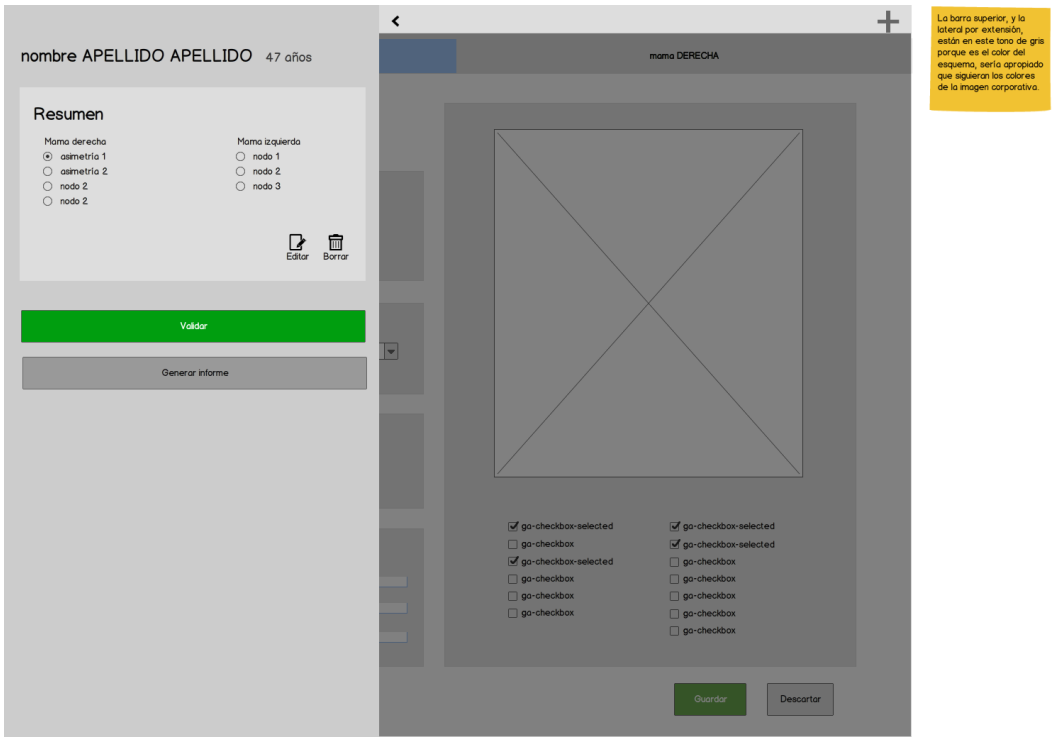


Figura 7.14: Mockup: Barra lateral con el resumen del informe

de error como el que se muestra en la figura 7.19

Pongámonos en el caso ahora de que el usuario quiere editar una lesión, la aplicación le redirigiría a una pantalla como la que muestra la figura 7.20 con los datos de la lesión a modificar.

Si por algun motivo, el usuario deseara volver atrás sin guardar los cambios vería un mensaje advirtiéndolo que los cambios que ha realizado en la lesión, si los hubiera, no se guardarán, como podemos ver en la figura 7.21.

Con todo este proceso creamos un documento con las interacciones. Tras validar las interacciones con los usuarios podremos pasar a la siguiente fase del desarrollo de la aplicación.

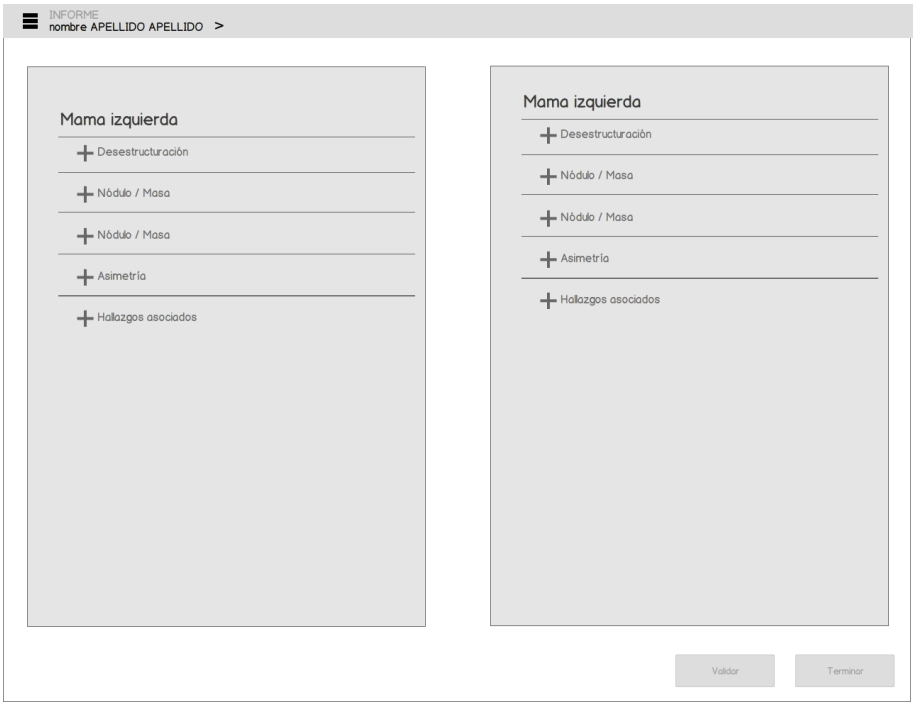


Figura 7.15: Mockup: Lista con las posibles lesiones de una mamografía

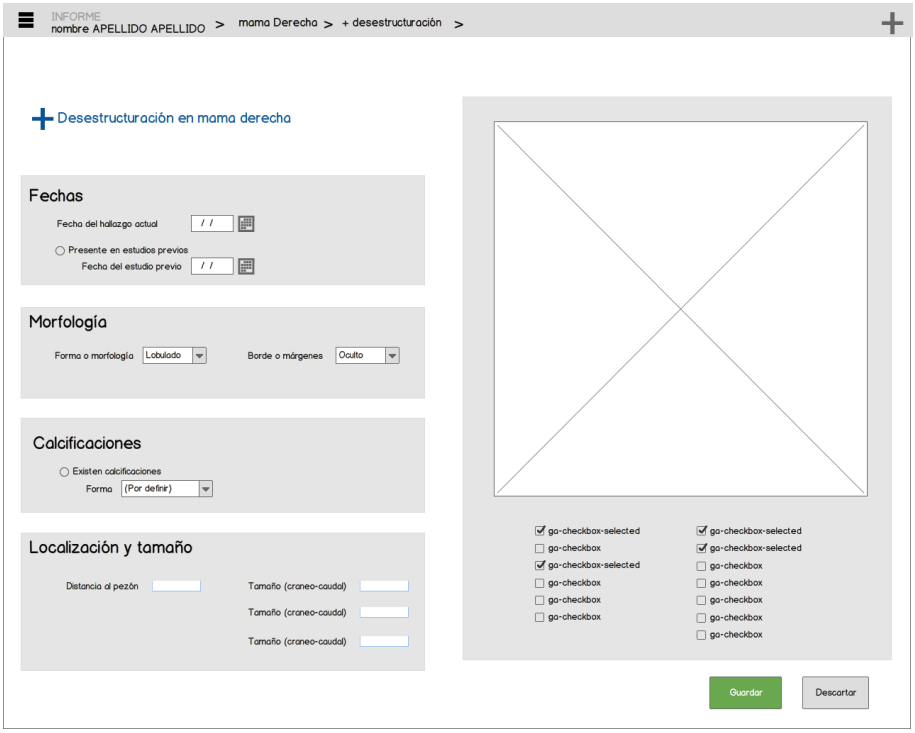


Figura 7.16: Mockup: Añadir una desestructuración en la mama derecha

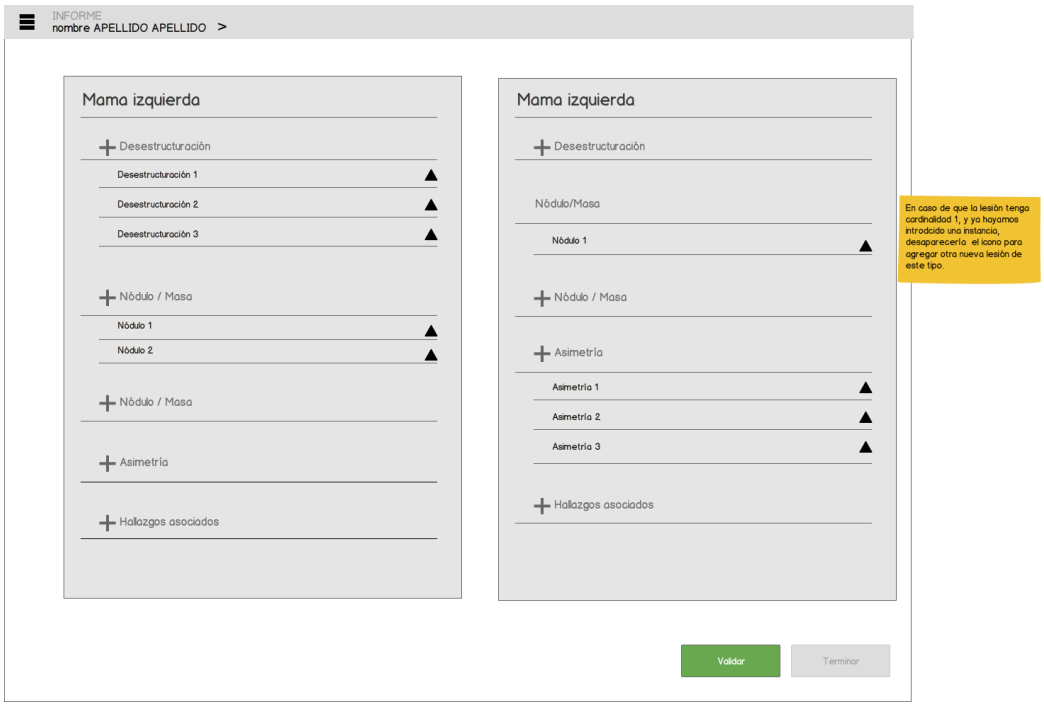


Figura 7.17: Mockup: Lista con las lesiones de una mamografía

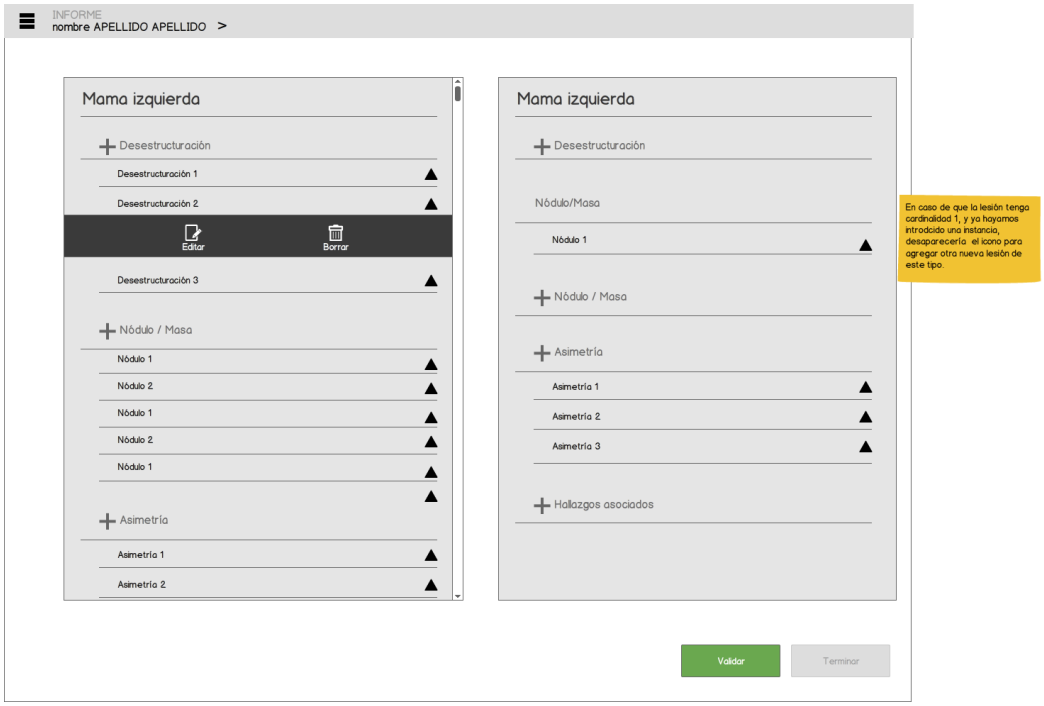


Figura 7.18: Mockup: Acciones que un usuario puede realizar sobre una lesión en concreto.

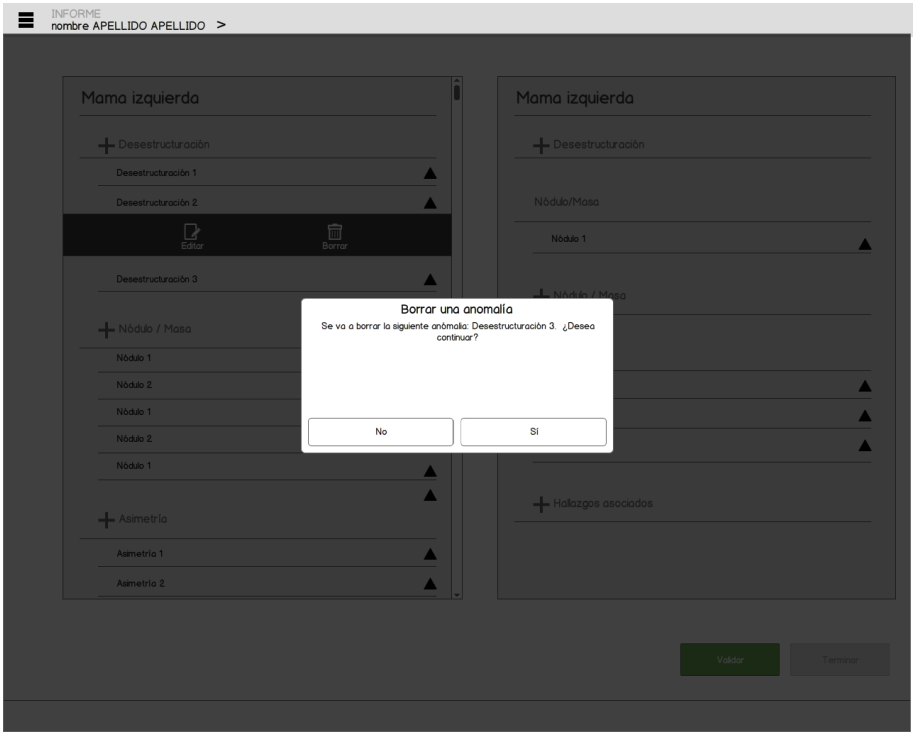


Figura 7.19: Mockup: Aviso de que el usuario va a borrar una lesión.

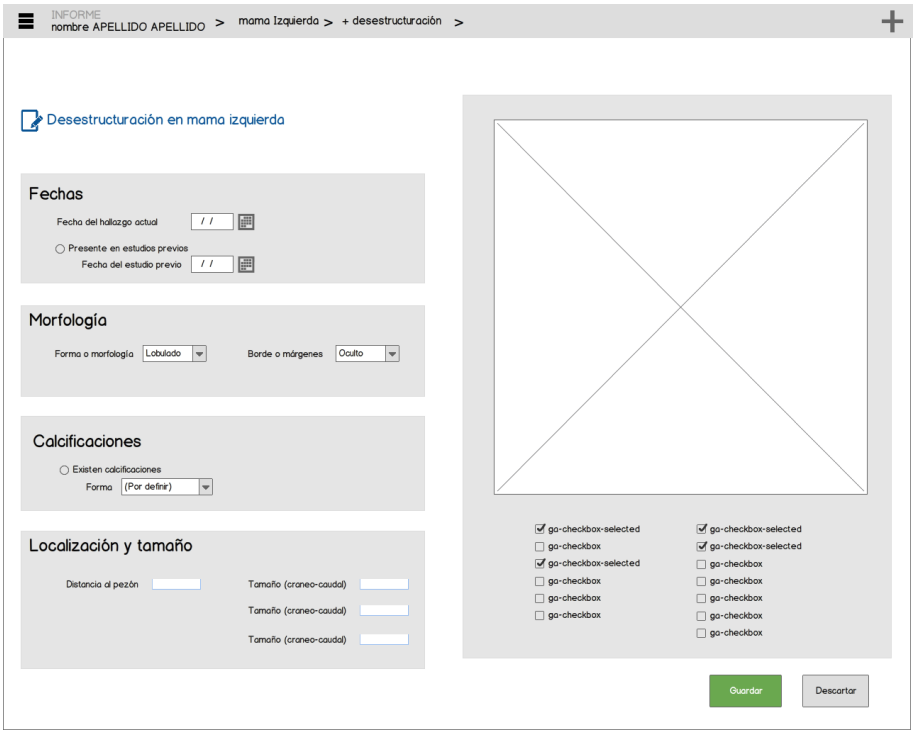


Figura 7.20: Mockup: Editar una lesión

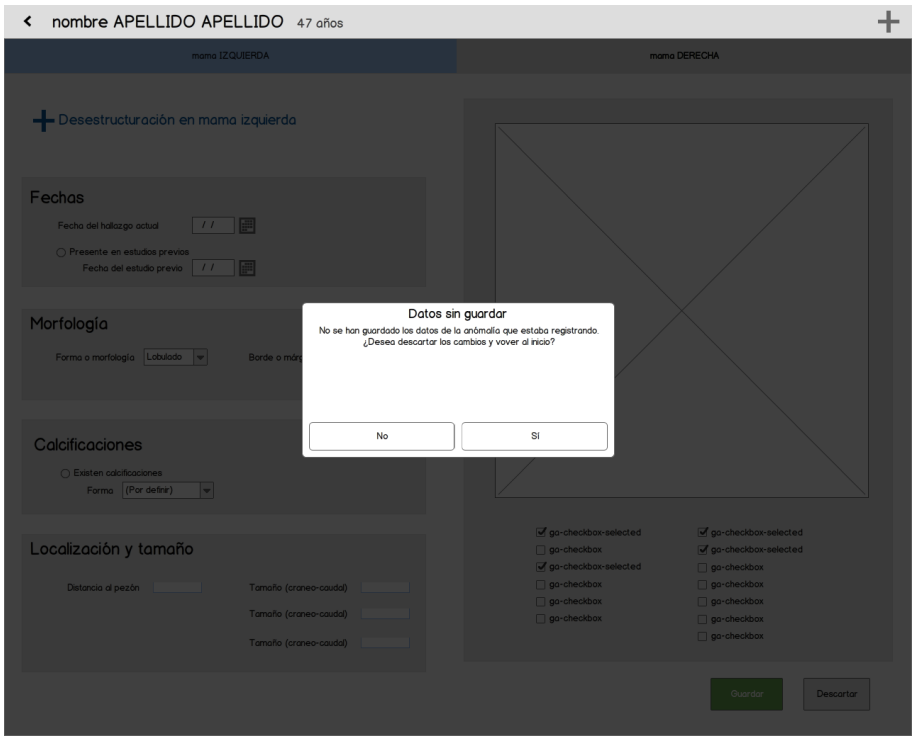
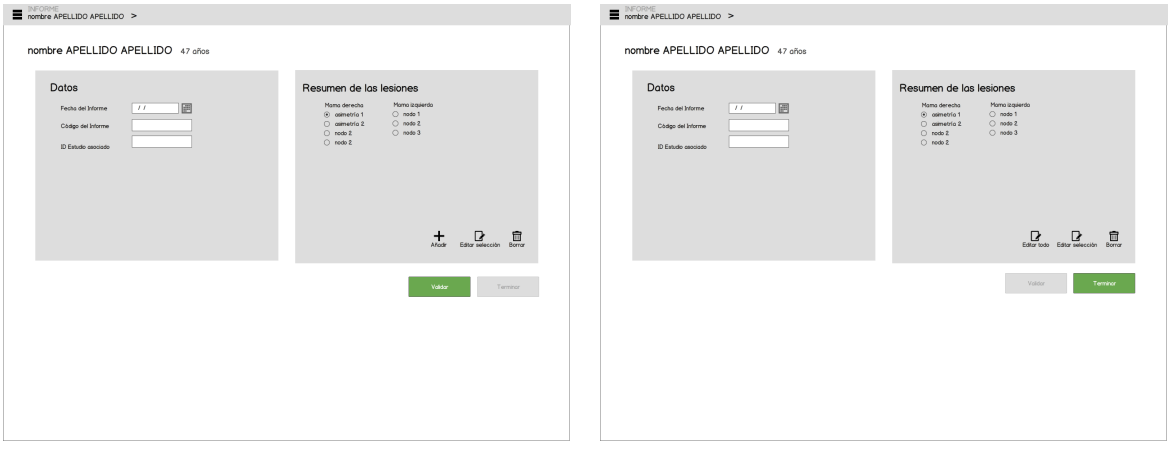


Figura 7.21: Mockup: Aviso de que no se guardan los cambios



(a) Informe pendiente de validar

(b) Informe listo para guardar

Figura 7.22: Informe con datos

7.3. | Prototipo

Antes de adentrarnos en la aplicación generadora de código debemos validar que las interacciones y la interfaz de usuario que hemos diseñado y validado sea correcta dentro del entorno de producción. Para esto, crearemos un prototipo, con parte de la funcionalidad ya implementada de la aplicación Android final.

Este prototipo además nos servirá como entorno de pruebas para generar las plantillas de código.

Hemos desarrollado una aplicación para Android que permitiría introducir datos de un informe de tipo *Exploración de Mama*.

El flujo de trabajo es similar al que hemos descrito en el apartado 7.2.2.

En la figura 7.23 vemos la interfaz de incio de la aplicación.

Cuando el facultativo vaya a añadir una lesión verá la pantalla mostrada en la figura 7.24, donde encontramos todas las posibles lesiones que se pueden asociar a cada órgano.

En este punto el usuario puede seleccionar un tipo de lesión a incluir en el informe y para introducir los datos de esta lesión deberá hacerlo rellenando un formulario como el que vemos en la figura 7.25.

Cuando el informe ya disponga de lesiones podrá ver un listado de las lesiones introducidas agrupadas por tipología como se puede ver en la figura 7.26. En este punto hay un pequeño cambio respecto al diseño de las interacciones donde las acciones disponibles para cada lesión, editar y borrar, se mostraban en un desplegable. Cuando desarrollamos este prototipo nos dimos cuenta que los pictogramas de las accones eran fácilmente identificables y que al ocultar las acciones en un nivel de anidamiento más algunos usuarios no sabían muy bien como editar o borrar una lesión puesto que estas acciones estaban ocultas. De modo que ahora las acciones se sitúan justo a la derecha del identificador de la lesión.

Si el usuario edita una lesión lo haría mediante una pantalla como la que vemos en la figura 7.27.

Al igual que en el diseño de interfaces propuesto, la banda superior permite tener presente en todo momento al paciente al que pertenece el estudio que se está introduciendo y la banda inferior encontramos un botón que permite terminar de introducir el informe y obtenerlo en formato DICOM-SR. En lugar de obligar al usuario a validar para después finalizar, simplificamos el proceso uniendo las dos tareas en una única. Cuando el usuario desee terminar de introducir datos y pulse el botón para finalizar, éste internamente realizará la comprobación antes de exportar el informe.

Este prototipo nos ha servido para validar el diseño que hemos creado. Como hemos descrito a lo largo de esta etapa de desarrollo tanto el diseño como las interacciones con el usuario varían para adaptarse a sus necesidades reales.

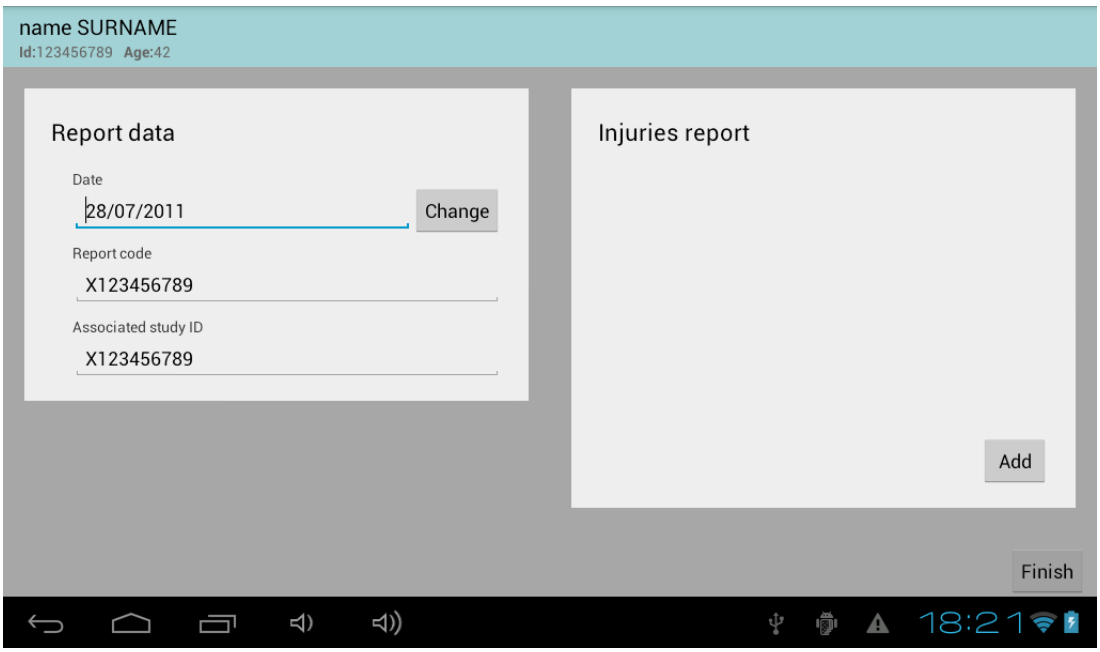


Figura 7.23: Prototipo: Inicio de la aplicación.

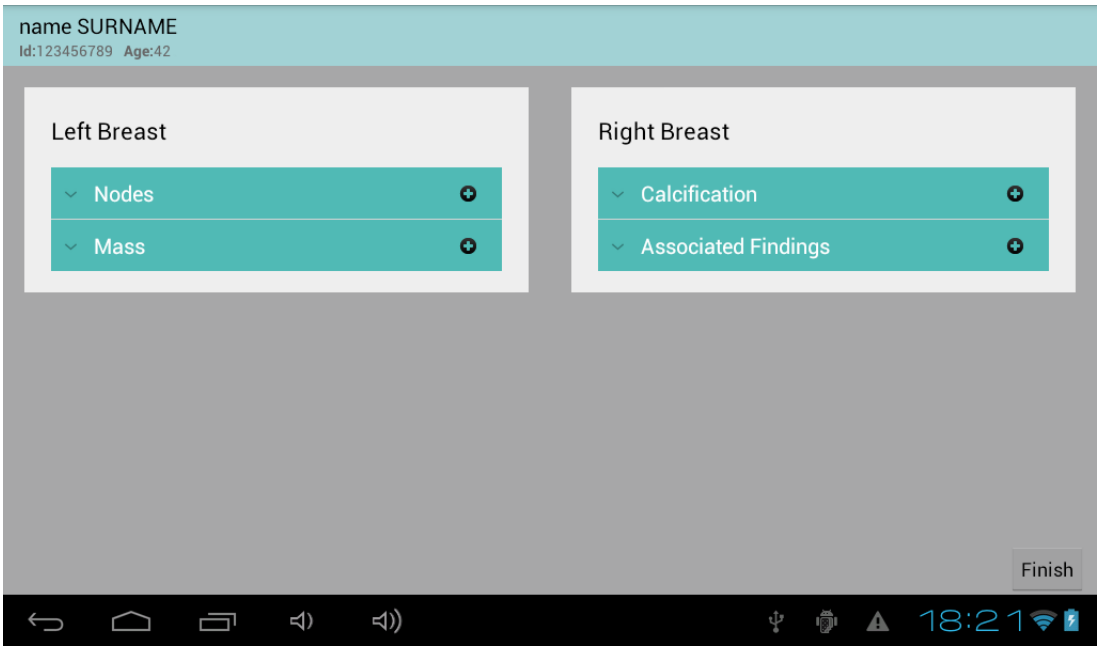


Figura 7.24: Prototipo: Lista con las posibles lesiones de una mamografía.

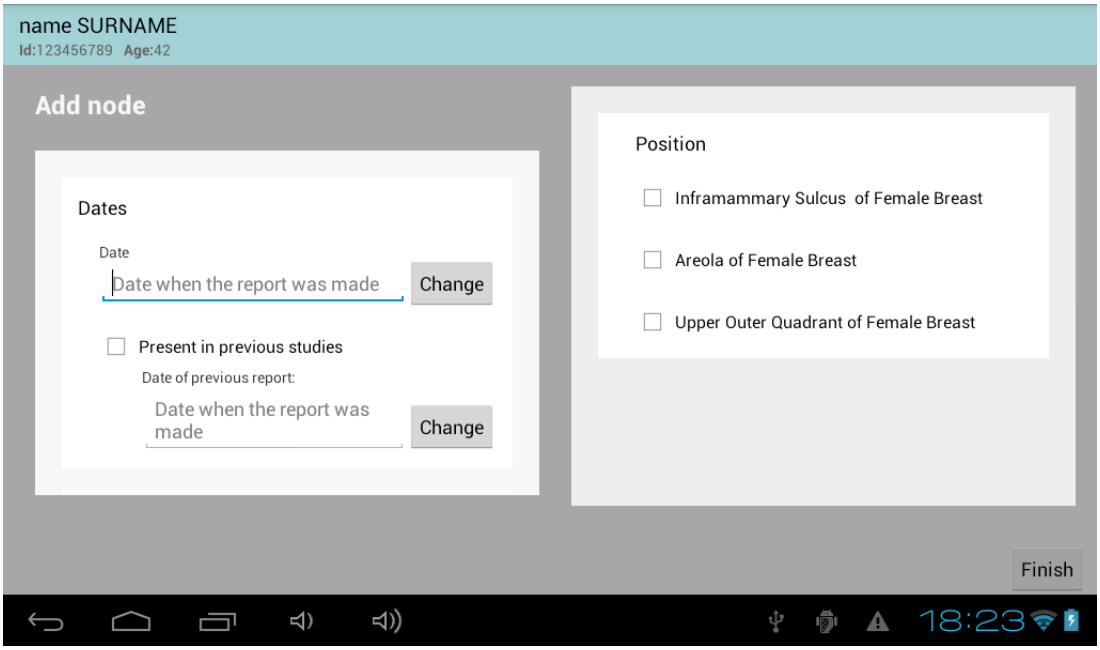


Figura 7.25: Prototipo:Añadir un nódulo a la mama izquierda.

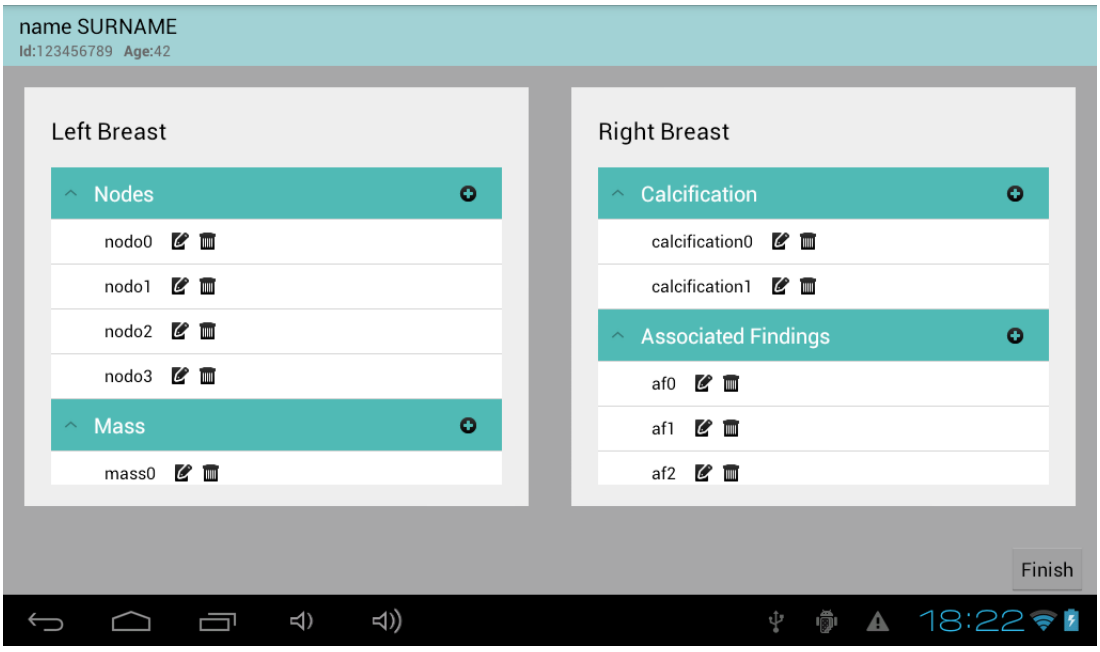


Figura 7.26: Prototipo: Lista desplegada con las lesiones de una mamografía.

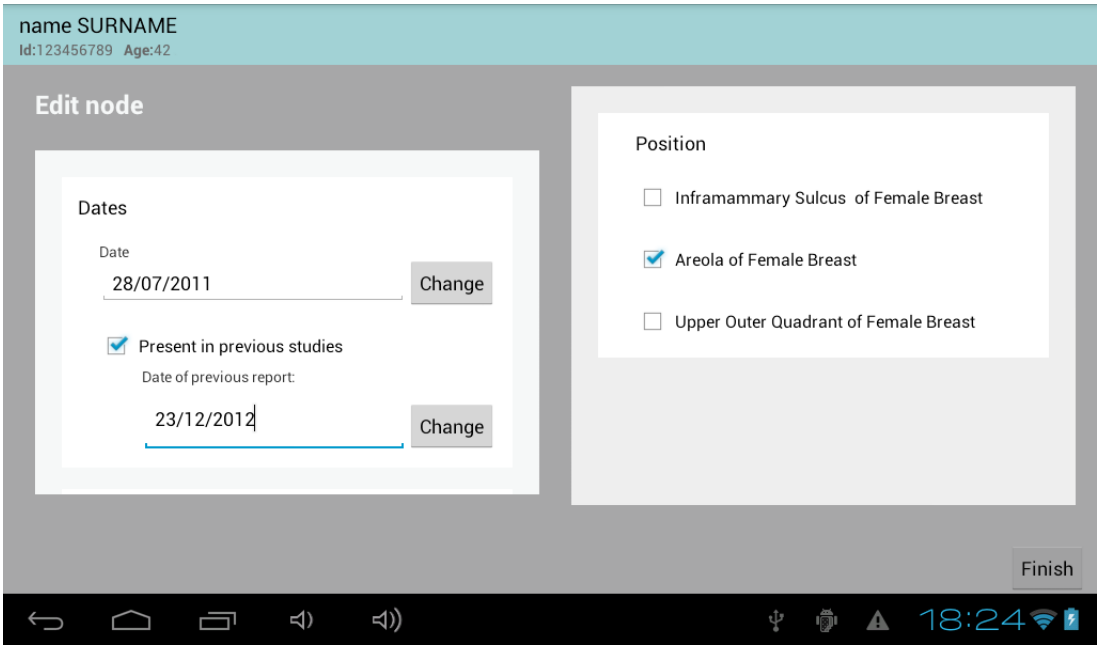


Figura 7.27: Prototipo: Editar un nódulo a la mama izquierda.

8 | Implementación

8.1. | Entorno de trabajo

A continuación describiremos brevemente el entorno de trabajo que hemos empleado para desarrollar el proyecto y que debemos hacer para reproducir este entorno de desarrollo.

Hemos trabajado indistintamente sobre sistemas operativos basados en Unix, utilizando Mac OS X y Linux; pero el entorno podría reproducirse para trabajar también sobre Windows.

Aunque se trata de un proyecto individual, el código se mantenía bajo un sistema de control de versiones, lo que permite mantener un registro histórico del desarrollo. Mantendremos dos repositorios en Github: uno en el que mantendremos el diseño del framework de Android [59] y otro para desarrollar el generador de código [60].

8.1.1. Python

La aplicación que genera el código está escrita en Python, así que obviamente necesitaremos el intérprete de Python para poder ejecutarla.

Emplearemos la rama 2.x de Python por los motivos que justificamos en el aparado 5.3. En las plataformas en las que hemos estado desarrollando podemos encontrar Python en la versión 2.7, la última versión estable del lenguaje, y es sobre esta versión sobre la que trabajaremos.

Pip y VirtualEnv

Uno de los problemas más comunes en el desarrollo son los conflictos entre librerías y la necesidad de replicar entornos de trabajo.

En Python existen dos herramientas que permiten atajar este problema: pip y virtualenv.

Pip es una herramienta para instalar y gestionar paquetes de Python automáticamente. Instalar un paquete con pip es tan sencillo como indicar el nombre del paquete y la versión, como vemos en el ejemplo de código 8.1.

```
1 $ pip install SomePackage==1.0
```

Código 8.1: Instalación de un paquete utilizando pip

También acepta como parámetro para instalar paquetes un fichero las librerías del proyecto, como podemos ver en el ejemplo de código 8.2.

```
$ pip install requirements.txt
```

Código 8.2: Instalación de los requisitos de un proyecto utilizando pip

Para conocer las dependencias de un proyecto únicamente tendremos que ejecutar pip con el parámetro *freeze* y nos devolverá un listado con las librerías que necesita este proyecto, que guardaremos en un fichero requirements.txt, lo que nos permitirá replicar el entorno fácilmente.

En esta aplicación únicamente hacemos uso de la librería externa Jinja2, y esta a su vez necesita MarkupSafe y wsgiref. Así que el fichero con los requisitos será como el que vemos en el código de 8.3.

```
Jinja2==2.7
MarkupSafe==0.18
wsgiref==0.1.2
```

Código 8.3: Requisitos del proyecto

La otra herramienta de la que haremos uso es virtualenv que crea entornos virtuales de Python aislados evitando los conflictos de dependencias entre las librerías de un proyecto y permite replicar el entorno fácilmente.

Cada entorno virtual contiene su propia versión de Pip encapsuladas. Lo que permite instalar librerías dentro del entorno que hayamos creado.

La combinación de estos dos paquetes nos permite tener entornos de trabajo con todas las dependencias del proyecto que pueden ser replicados con facilidad.

8.1.2. Android

Tanto para desarrollar el prototipo como para crear el framework de Android que instanciará los informes DICOM-SR, necesitamos trabajar dentro del entorno de desarrollo de Android.

Para desarrollar una aplicación haremos uso de las siguientes herramientas:

- El kit de desarrollo de Java (JDK).
- La kit de desarrollo de Android (Android SDK). En nuestro caso necesitaremos la SDK de la versión 3.2.
- El IDE de desarrollo Eclipse.
- El plugin para Eclipse para poder trabajar con Android. (ADT)

Concluyendo, tenemos un entorno de trabajo para el generador de código y otro para el desarrollo de la aplicación framework de Android. Esto nos permite desarrollar de modo independiente cada una de las partes que componen la arquitectura de la solución.

8.2. | Aplicación framework de Android

El desarrollo de la solución final sigue la metodología en cascada con prototipado. Por ello, necesitamos desarrollar simultáneamente una aplicación Android que nos servirá como esqueleto (o framework) para insertar en ella el código que generemos a partir de un informe DICOM-SR concreto.

Para esta aplicación Android necesitamos que el usuario valide las decisiones de diseño e interacción que hemos tomado, por lo tanto ya durante la etapa de diseño elaboramos un prototipo como ya hemos explicado en el apartado 7.3.

Este mismo prototipo lo haremos evolucionar para que nos sirva como esqueleto para las aplicaciones generadas. Simplemente eliminaremos del prototipo todas las partes que correspondan a un informe concreto, y las sustituiremos por el código generado automáticamente. Y añadiremos las funcionalidades que sean genéricas para todas las aplicaciones que se generan a partir de informes DICOM-SR.

Durante el desarrollo de este prototipo nos dimos cuenta que algunas de las pantallas diseñadas en la etapa anterior no podían generalizarse fácilmente. Por ejemplo, en las figuras 7.15 y 7.17 veíamos todas las posibles lesiones de todos los órganos de este informe, como no siempre habrán dos órnagos involucrados en un informe, esta pantalla no podía generalizarse.

Decidimos entoces ordenar las pantallas siguiendo la estructura propia del informe. Si bien la solución propuesta en el apartado 7.2.2 agrupa el conocimiento de una forma algo más simple para el usuario final que únicamente sirva para un subconjunto de informes médicos hace que la descartemos. Optando por una solución genérica que permita transformar en una aplicación Android cualquier plantilla de informe médico.

Para no duplicar información, podremos ver las modificaciones realizadas en la aplicación esqueleto en el apartado 9.3.

Resumiendo hemos evolucionado el prototipo desarrollado en la etapa anterior para poder instanciar todo tipo de infores médicos. De esta aplicación Android, eliminamos los fragmentos que pertenecen a un informe concreto quedándonos el esqueleto que después completaremos con el código generado automáticamente.

8.3. | Analizador sintáctico

El análisis sintáctico (parsing) se encarga de leer el fichero de tipo DICOM-SR y crear una estructura de datos en memoria con la información del fichero. La estructura de datos más comúnmente empleada para esta tarea es un árbol ya que encapsula la información tanto de los nodos como de la jerarquía.

Los analizadores sintácticos de XML se dividen en dos categorías fundamentalmente [61]: analizadores DOM (Document Object Model) y SAX (Simple API for XML).

DOM emplea es una interfaz de aplicación (API) basada en árbol que modela todo el documento XML como un árbol. Es adecuado cuando queremos realizar operaciones sobre ficheros complejos y sobre los que realizan accesos aleatorios, porque en el árbol

generado tenemos toda la información del informe. Pero este árbol que genera puede llegar a ser hasta 10 veces más grande que el fichero XML lo hace poco apropiado para tratar con ficheros grandes.

SAX, por el contrario es una interfaz de aplicación basada en eventos. Lee el fichero secuencialmente, cada nodo leído provocará que se lance un evento que es tratado por el analizador de tipo SAX. SAX es apropiado para ficheros grandes, porque según va leyendo el fichero lanzará los eventos asociados y descarta la información de lo leído hasta este punto. Su complejidad espacial es menor porque no necesita almacenar mucha información en memoria. Precisamente por este motivo no permite realizar operaciones que requieran de información global.

En nuestro caso, los informes DICOM-SR con los que trabajamos son potencialmente muy largos, por lo que un análisis tipo DOM queda descartado. Sin embargo, para generar el código necesitaremos un árbol, lo que haremos tras el análisis es generar una estructura arbórea eficiente. Este proceso se describe en el apartado 8.4.

En la librería estándar de Python podemos encontrar el paquete *xml.sax*, que implementa la interfaz de SAX.

Como en nuestro caso queremos extender el comportamiento del analizador sintáctico XML para que sea capaz de interpretar ficheros con plantillas de informes médicos DICOM-SR, por lo tanto crearemos una clase que extienda de la clase *xml.sax.ContentHandler* que es precisamente la clase que trata los eventos lanzados por el analizador sintáctico.

En la figura 7.4 hemos visto el diagrama UML que modela esta clase.

Los atributos internos nos permiten almacenar información acerca de lo leído en el fichero y al terminar el análisis generar un árbol con los datos del informe DICOM-SR.

En el segmento de código 8.4 vemos como la clase *DicomPasrser* extiende del analizador sintáctico SAX estándar. Recibe un informe DICOM-SR que sigue el formato XML y en la variable interna *self._dict_report* devolverá la estructura arbórea eficiente con la información del fichero DICOM-SR.

```
1  class DicomParser(xml.sax.handler.ContentHandler):
2      ...
3
4      def parse(self, xml_file):
5          """ Parse the file using this handler.
6          Returns the report using a DictReporT.
7
8          Keyword Argument:
9          xml_file  -- XML file in DICOM format to parse
10
11         """
12         xml.sax.parse(xml_file, self)
13         return self._dict_report
14     ...
```

Código 8.4: Clase que analiza sintácticamente un fichero DICOM-SR

Los métodos que hemos sobrecargado para recoger información del informe DICOM-SR son los siguientes:

- *startElement*: es llamado cada vez que un nueva etiqueta de apertura de un nodo. En nuestro caso lo empleamos para iniciar variables y preparar las variables que nos indican en que punto del análisis sintáctico nos encontramos.
- *endElement*: se llama cuando se encuentra un final de etiqueta. Si se trata del final de un atributo, almacenamos la información en un contenedor temporal de tipo *SAXContainer* y si se trata del final del contenedor en si, lo añadimos a la variable interna de tipo *SAXReport* que guarda una lista con todos los contenedores DICOM-SR encontrados.
- *characters*: se llama cada vez que se lee un carácter y este es almacenado en un buffer.
- *endDocument*: finalmente cuando termina el documento se llama este método. Como en este punto ya tenemos toda la información del fichero, la reorganizaremos en forma de árbol. Este proceso lo explicaremos en el apartado 8.4.

En los informes DICOM-SR, cada contenedor sigue un formato estándar lo que le da gran versatilidad. En un contenedor DICOM-SR sus atributos y los contenedores que cuelgan de este contenedor se agrupan bajo la etiqueta de los hijos.Por simplicidad para generar el código, hemos agrupado los atributos dentro del contenedor y sus hijos serán únicamente otros contenedores.

En la figura 8.1 vemos un esquema de cómo se estructura la información en el analizador sintáctico, al compararla figura 5.1 encontramos las diferencias que hemos descrito respecto a un fichero DICOM-SR.

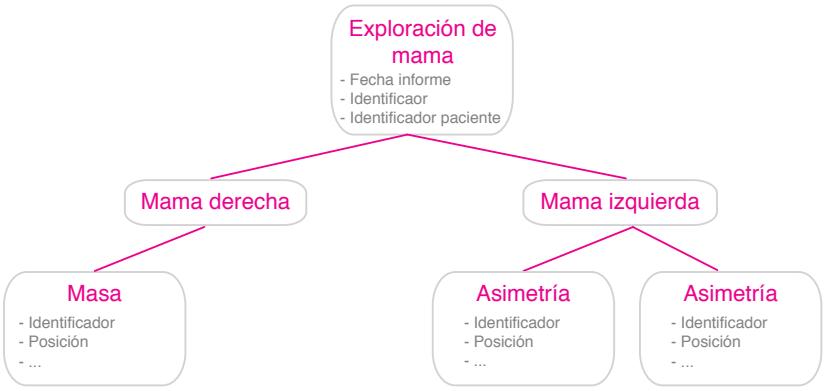


Figura 8.1: Esquqema del árbol DICOM-SR para el analizador sintáctico

Para almacenar la información a medida que vamos leyendo creamos dos clases: *SAXContainer* y *SAXReport*.

En el segmento de código 8.5 vemos la interfaz de la clase desarrollada para almacenar y gestionar los contenedores DICOM-SR mientras se está leyendo el fichero. Como ya hemos explicado los atributos DICOM-SR se integran dentro del contenedor. Además hemos definido atributos como *open*, *parent* y *tree_level* para almacenar elementos de la estructura del informe que nos permitirán después construir el árbol.

```
1  class SAXContainer(object):
2      """This class stores a container tag from xml"""
3      def __init__(self, concept=Concept(), level=-1,
4                   open_level=True, parent=Concept(),
5                   properties=Property()):
6          self.concept = Concept(concept.value, concept.schema,
               concept.meaning)
7          #A list of attributes/nodes (date, num, text)
8          self.attributes = []
9          self.tree_level = level
10         self.open = open_level
11         self.parent = parent
12         self.properties = properties
13
14     def has_concept(self)
15     def add_attribute(self, attribute)
16     def __repr__(self)
```

Código 8.5: Clase que almacena información durante el análisis de un contenedor

Dentro de cada contenedor se almacena un concepto DICOM-SR. Son los conceptos los que mantiene la información en si del sistema. Tenemos la clase *Concept* y *Data type* con sus hijos para cada tipo de dato que pueda aparecer en el informe *Num* (números), *Text* (cadenas de texto), *Code* (cadenas de texto de un conjunto de posibilidades) y *Date* (fechas). Hemos visto el diagrama UML en 7.9.

En el ejemplo de código 8.6 vemos la interfaz de la clase que almacenará el informe durante el análisis sintáctico del informe. Mantiene informcación de la ontología a la que pertenece el informe médico y una lista con los contenedores DICOM que vamos leyendo.

```
1  class SAXReport(object):
2      """This class manages the report while we are reading it from
           xml"""
3      def __init__(self):
4          self.report_type = ""
5          self.id_odontology = -1
6          self._containers = []
7
8      def add_attribute(self, child_level, current_attribute)
9      def add_container(self, new_container)
10     def close_level(self, child_level)
```

```
11      def return_parent(self, tree_level)
12      def __repr__(self)
```

Código 8.6: Clase que almacena información durante el análisis de un informe

Al finalizar en análisis sintáctico tendremos una lista con todos los nodos del informe.

Aunque algunos de los traductores orientados a la sintaxis tras el análisis generan directamente el código, en nuestro caso antes construiremos un árbol con la información leída. La complejidad que añadimos en este paso adicional después simplificará la generación de código, que es el punto más importante de esta aplicación.

8.4. | Estructura de datos para almacenar un informe DICOM-SR

Antes de adentrarnos en la estructura árborea que hemos diseñado justificaremos su idoneidad para este proyecto.

Con este objetivo desarrollamos un sistema de pruebas que generaba las aplicaciones de un conjunto de informes DICOM-SR utilizando el informe almacenado dentro de una lista y con el árbol. Hemos calculado la media del tiempo de usuario empleado para generar un informe. Podemos ver los resultados de generar únicamente la interfaz de usuario en la tabla 8.1 y en la tabla 8.2 vemos el tiempo empleado en generar completamente la aplicación Android.

EDA / Nº informes	Lista	Árbol
10	0.476 s.	0.51 s
100	0.48 s.	0.545 s.
1000	0.464 s.	0.521 s.

Tabla 8.1: Tiempos de ejecución para la generación de la interfaz de usuario

EDA / Nº informes	Lista	Árbol
10	0.92 s.	0.812 s.
100	0.97 s.	0.801 s.
1000	0.956 s.	0.795 s.

Tabla 8.2: Tiempos de ejecución para la aplicación Android

Lo que observamos a partir de estas tablas es que cuando generamos la interfaz de usuario o siendo más generalistas secciones del código que necesitan recorrer todo el informe y que no tienen en cuenta la jerarquía de los nodos, el código que utiliza una lista es más rápido porque no hemos necesitado emplear recursos en crear un árbol con el informe. En cambio si tenemos que generar toda la aplicación, lo que implica segmentos de código que necesitan información jerárquica, el tiempo utilizado en crear el árbol se compensa al generar el código.

Los informes que hemos empleado en el script para calcular los tiempos eran de tamaño medio y pequeño. Si utilizamos informes más grandes los resultados serían más abultados, pero en definitiva el uso de un árbol para generar el código está justificado.

En la librería estándar de Python no exista una estructura de árbol, así que nos declaramos una clase para árboles que pueda almacenar cualquier tipo de valor. La interfaz de esta clase podemos verla en el ejemplo de código 8.7.

Es un árbol sobre el que se implementan métodos típicos como: recorridos en profundidad y anchura, búsqueda, vaciado, añadir nuevos elementos y la impresión en forma de árbol. Y otros métodos orientados a la tarea para la que vamos a utilizar este árbol como: *get_set_data* que devuelve un conjunto de nodos sin elementos repetidos, *get_flat_tree* que devuelve el árbol aplanado dentro de una tabla hash o *get_code_containers* que nos devuelve una lista con los identificadores de los elementos de árbol de tipo "CODE".

```
1  class Tree():
2      """Tree Object, contains value and child references"""
3      def __init__(self, value=None, children=None):
4          self.value = value
5          if children is None:
6              self.children = []
7          else:
8              self.children = children
9
10     def breadthFirst(self)
11     def depthFirst(self)
12     def __contains__(self, item)
13     def clear(self)
14     def get_set_data(self,containers,attributes)
15     def is_leaf(self)
16     def print_tree(self,ident)
17     def add_node(self, container, parent)
18     def get_flat_tree(self,flat)
19     def get_code_containers(self)
```

Código 8.7: Clase para árboles genéricos

En este árbol los nodos almacenarán la información de un Contenedor DICOM-SR, para esto hemos definido una clase de tipo *Container*. La interfaz de esta clase la podemos ver en el ejemplo de código 8.8. Se trata de una clase muy similar a la mostrada en el ejemplo de código 8.5, pero como en este punto ya no necesitamos información acerca de la jerarquía del informe porque esta información se conserva en la estructura del árbol, no necesitamos información acerca del estado del análisis sintáctico porque ya lo hemos completado. Por ello descartamos estos atributos para así aligerar la estructura de datos en la medida de lo posible.

```
1  class Container(object):
2      def __init__(self, tree_level, concept=Concept(),
3                   properties=Property(), attributes=[]):
4          self.tree_level = tree_level
5          self.concept = concept
6          self.properties = properties
```

```
7           self.attributes = attributes[:]
8
9       def get_code(self)
10      def get_schema(self)
11      def get_meaning(self)
12      def get_level(self)
13      def get_concept(self)
14      def get_schema_code(self,sep='_')
15      def has_code(self,code)
16      def __str__(self)
17      def __repr__(self)
```

Código 8.8: Clase para contenedores DICOM-SR

Finalmente nos queda ver como se integra el árbol en la clase que modela el informe DICOM-SR.

Como vemos en el ejemplo de código 8.9, la interfaz de esta clase es análoga a la definida en el ejemplo de código previo 8.6, descartando la información que ya no es necesaria y encapsulando los contenedores del informe dentro de un árbol en lugar de en una lista.

```
1  class DicomSR(object):
2      def __init__(self, report_type="", id_ontology=-1):
3          self.report_type = report_type
4          self.id_ontology = id_ontology
5          self.report = Tree()
6
7      def __repr__(self)
8      def add_node(self, node, parent):
9      def get_ontology(self):
10     def get_flat_data(self):
11     def get_root(self):
12     def get_data_form_report(self, languages, template_type):
```

Código 8.9: Clase para contenedores DICOM-SR

La transformación de una estructura de datos en otra se realiza al finalizar en análisis sintáctico.

Cuando el analizador sintácico SAX alcance el final del documento, lanzará un evento *endDocument* y en este punto construiremos el árbol.

En el ejemplo de código 8.10, vemos como hemos construido este árbol. Primero ordenamos los contenedores de la lista y vamos recorriendo esta lista ordenada insertándolos en su posición correcta en el árbol.

```
1       def build_dicom_tree(self):
2           """ Build a DicomSR tree using the information read in
                the file."""
3           self._dict_report = DicomSR(self._report.report_type,
```

```
4                                       self._report.id_odontology)
5           #Sort containers list by its tree level.
6           self._report._containers.sort(key=lambda x: x.tree_level)
7           for container in self._report._containers:
8               self._dict_report.add_node(
9                   Container(container.tree_level,
10                            container.concept,
11                            container.properties,
12                            container.attributes),
13                  container.parent)
```

Código 8.10: Constructor de un árbol DICOM-SR

Aunque generar el árbol y rellenarlo suponga añadir una etapa más a la generación de código. Este paso intermedio simplifica el proceso de generaración automática de código, especialmente de aquellas partes de código que necesiten información jerárquica del informe médico.

8.5. | Generador de la aplicación Android

Tras la etapa anterior tenemos el informe DICOM-SR cargado en memoria en una estructura de árbol. La siguiente etapa será la de generar el código necesario para instanciar la aplicación Android framework para un informe DICOM-SR concreto.

8.5.1. Configuración

Un generador de código no necesita de la intervención del usuario, de hecho cuanto más automatizado esté el proceso más efectivo será.

Sin embargo existen ciertas opciones que permitimos que el usuario configure, como son: la ruta o el nombre de cada nivel jerárquico del informe DICOM-SR en función de la ontología.

Además, ciertas funciones internas como la ruta en la que se encuentran las plantillas para cada elemento, la internacionalización, . . . también se establecen utilizando ficheros de codificación.

La librería estándar de Python nos ofrece un módulo para gestionar la configuración de manera muy simple, se trata de *ConfigParser*.

Este módulo nos permitirá leer tanto la configuración establecida por el usuario como la establecida por el sistema.

Los ficheros que utiliza *ConfigParser* estructuran la información mediante secciones, que se indican ente corchetes [ ]. Cada sección almacena los datos de configuración mediante pares clave-valor.

En el ejemplo de código 8.11 vemos un ejemplo de configuración con la sección que contiene la ruta dónde se almacenará el código generado. La sección es *[Output Directories]* y como podemos ver los ficheros de configuración permiten interpolar valores; por ejemplo: el directorio dónde se guardarán las actividades es el siguiente:

Activities: %(MainDir)s/activities, cuando preguntemos a *ConfigParser* por este valor nos devolverá la siguiente ruta: *./outputs/activities*.

```
1  ...
2
3  [Output Directories]
4  MainDir: ./outputs
5  Strings: %(MainDir)s/strings
6  Activities: %(MainDir)s/activities
7  Model: %(MainDir)s/models
8  Layouts:  %(MainDir)s/layouts
9  Logs: ./logs
10 Parser: %(Logs)s/parser.log
11 Properties: ./templates/properties
12 Manifest: %(MainDir)s/manifest
13
14 ...
```

Código 8.11: Sección del fichero de configuración

La gestión de estos ficheros de configuración la hacemos dentro del módulo *core*. Puesto que necesitaremos acceder a la configuración desde distintos módulos del programa y por lo tanto tiene sentido que este módulo este accesible.

El fichero *config.py* contiene las funciones que tratarán con los ficheros de configuración, así como la gestión de otros aspectos de la configuración como la interacionalización. Las funciones más relevantes para este propósito son las siguientes:

- *set_environment*: establece las variables de entorno de Jinja para la gestión de plantillas. Hablaremos más en profundidad de este tema en el apartado 8.5.2.
- *get_languages*: devuelve una lista con los idiomas soportados por la plantilla.
- *read_config*: crea una instancia de *ConfigParser* y lee el ficheros con la configuración.
- *get_filepath*: devuelve la ruta dónde debemos generar los ficheros de un tipo concreto.
- *get_filename*: instancia el nombre del fichero con los datos concretos del informe
- *get_property*: devuelve una propiedad concreta del fichero de configuración.
- *get_substitution_options*: devuelve todas las opciones de una sección del fichero de configuración.
- *get_language_code*: devuelve el código del idioma asociado a la etiqueta del informe DICOM-SR. Utilizamos esta funcionalidad para generar las cadenas de texto internacionalizadas. En el apartado 8.5.4 explicaremos esto con más detalles.

- *get_ontology_level*: Devuelve el nombre de un nivel del árbol DICOM-SR basándose e la ontología.
- *get_substitution_dictionary*: devuelve un diccionario con las cadenas listas para internacionalizar las plantillas.
- *get_layout_settings*: cada nivel del árbol se puede configurar tiene ciertas opciones configurables en la interfaz como por ejemplo si los datos se distribuirán en 1 o 2 columnas. Esta función nos devuelve las opciones de configuración para un nivel de una ontología.

Todas estas funciones, junto con los ficheros de configuración, permiten parametrizar y adaptar la aplicación que generemos a las necesidades del usuario. Además al centralizar la gestión de la configuración evitamos tener código repetido en cada uno de los módulos de la aplicación.

8.5.2. Plantillas

Una de las piezas claves en la generación de código son las plantillas. Las plantillas modelan las partes de la aplicación final y sus variables se completan utilizando los datos que hemos obtenido tras en análisis sintáctico del informe DICOM-SR.

Gracias a las plantillas el código que generemos es versatil y se adapta a las necesidades concretas de un informe.

Tendremos plantillas para cada parte de la aplicación Android. Como hemos explicado necesitamos generar las actividades, el modelo, la interfaz gráfica y la interacionalización, por lo tanto tendremos un conjunto de plantillas para cada una de estas secciones.

Combinaremos dos sistemas de plantillas dependiento de la complejidad de las propias plantillas y de los datos con las que las completaremos.

String.Template

De nuevo, dentro de la librería estándar encontramos una solución muy sencilla para la creación de plantillas y la sustitución de sus variables con datos concretos.

Empleamos este método para los casos más sencillos, como para instanciar los nombres de los ficheros.

En el ejemplo de código 8.12, vemos una sección de configuración con los nombres de los ficheros para la distribución de la interfaz de usuario de una Mamografía. Cuando se este generando el código de la aplicación Android para un informe concreto, por cada nivel de la jerarquía del árbol, se generará unos ficheros que tomarán el nombre de esta platilla completada con el código del nivel DICOM-SR.

```
1 ...
2
```

```
layout/
├── summary_radlex_rid10357.xml
├── tree_radlex_rid10357_radlex_rid29896.xml
├── tree_radlex_rid10357_radlex_rid29897.xml
├── edit_radlex_rid29897_trencadis_mamo_trmm0011.xml
├── edit_radlex_rid29897_radlex_rid3874.xml
├── edit_radlex_rid29897_radlex_rid34261.xml
├── edit_radlex_rid29896_trencadis_mamo_trmm0011.xml
├── edit_radlex_rid29896_radlex_rid3874.xml
└── edit_radlex_rid29896_radlex_rid34261.xml
```

Figura 8.2: Ficheros que contienen la interfaz de usuario para una Mamografía

```
3 [Layout Filenames]
4 level_1: summary_${CODE}
5 level_2: tree_${PARENT}_${CODE}
6 level_3: edit_${PARENT}_${CODE}
7
8 ...
```

Código 8.12: Sección de la configuración con los nombres de los ficheros

Dentro de la configuración gestionaremos el completado de estas plantillas haciendo uso de String.Template.

Como resultado obtenemos ficheros necesarios para la aplicación dónde las variable *{$CODE}* y *{$PARENT}* se sustituyen por el esquema y el código DICOM del nodo actual y de su padre respectivamente.

En el ejempo de código 8.2 tenemos un ejemplo de la salida que obtendríamos tras emplear las plantillas para generar los nombres de los ficheros.

En definitiva, cuando queremos instanciar plantillas simples, el uso de *String.Template* nos permite dar una solución simple y efectiva.

Jinja

Sin embargo en este proyecto necesitamos un sistema de plantillas más potente para generar el código.

Para la generación de código utilizamos la librería Jinja2 [62].

Se trata de un módulo que incorpora a Python un lenguaje de modelado de plantillas. Entre las características más importantes de este módulo encontramos:

- Soporte para Unicode. Especialmente importante para internacionalizar las plantillas para los distintos idiomas soportados.
- Las plantillas se ejecutan en un entorno controlado de seguridad.

- Herencia entre plantillas.
- Permite precompilar las plantillas.
- La sintaxis de las plantillas es configurable.

En Python 3.x las cadenas siempre se representan usando Unicode, pero Python 2.7.x soporta dos tipos de cadenas de texto y la opción por defecto no es la Unicode. Así que siempre que vayamos a pasar cadenas de texto a las plantillas que hayamos definido, nos aseguraremos siempre de que el formato es Unicode de manera explícita. En 8.13, vemos un ejemplo de como pasar correctamente cadenas de texto a Jinja2.

```
1     write_string = u"{0}".format(concept_name)
```

Código 8.13: Cadena de texto Unicode para Jinja2

Jinja utiliza una clase de entorno *class jinja2.Environment([options])* para gestionar la carga de plantillas, la configuración y las pruebas; y una clase para gestionar propiamente las plantillas *class jinja2.Template*.

En nuestro proyecto el uso de estas clases la realizaremos dentro del módulo de configuración, de modo que todas las funciones relacionadas con el uso de las plantillas se encuentran centralizadas y el resto de módulos puede acceder.

El lenguaje para diseñar plantillas define variables que se sustituyen al ser evaluados por valores concretos.

Para hacer más versátiles estas plantillas, Jinja2 incluye herencia, bucles para controlar los bloques, pruebas, ...

La filosofía seguida para diseñar las plantillas prima maximizar la usabilidad de los bloques. Por lo tanto se definen plantillas unitarias para tratar con elementos básicos como: botones, secciones, variables, ...; plantillas genéricas que servirán como base y plantillas hijas que hereden de estas plantillas base diseñadas haciendo uso de la herencia.

En el ejemplo de código 8.14 vemos una plantilla que crea la interfaz de un campo de control.

```
1
2 <!-- Bool: {{ concept\_name }} -->
3 <CheckBox android:id="+id/cbox_ concept_value
     "android:layout_width="wrap_content.ªndroid:layout_height="wrap_content"and
    /code\_{{ concept\_value }}"
4
5
6     {%- if not first\_attribute %}
7     android:layout\_alignLeft="id/ previous_item
        "android:layout_below="id/{{ previous\_item }}"
8
9     {%- endif -%}
```

```
10 |     android:layout\_marginBottom="10dp"/>
```

Código 8.14: Plantilla Jinja2

La combinación de estos dos sistemas de plantillas nos proporciona un entorno rápido y sencillo para las tareas que menos complejidad y la potencia suficiente para generar el código de una aplicación Android soportando lenguajes con caracteres Unicode.

8.5.3. Generador del modelo

Ahora que disponemos de un lenguaje para diseñar plantillas y de la información del informe DICOM-SR cargada en memoria dentro de una estructura de árbol, generaremos el código necesario para obtener la aplicación Android.

En primer lugar generaremos el modelo de datos para el informe concreto con el que estamos trabajando.

Deberemos recorrer el árbol y generar las clases pertinentes.

Para el caso general recorreremos cada nodo completando una plantilla Jinja2 como la que podemos ver en el ejemplo de código 8.15.

```
1  package {{ package }};

3  {% for import in imports -%}
       {{import}}
5  {% endfor  %}
   public class {{ class\_name }} {
7
       {% for attribute in attributes -%}
9          {{attribute}}
       {% endfor  %}
11
       {% for method in methods -%}
13         {{method}}
       {% endfor  %}
15 }
```

Código 8.15: Plantilla para una clase genérica

Por cada contenedor DICOM-SR tendremos una clase. Los atributos son todos los hijos del contenedor. En caso de que alguno de los atributos sea multivaluado, es decir que la cardinalidad máxima sea -1, el tipo del atributo será un *ArrayList* de este mismo tipo. Por ejemplo, las lesiones que se pueden encontrar en la mama derecha en una mamografía (*Radlex_rid10357_radlex_rid29896*) son:

- Desestructuraciones (*ArrayList<Radlex_rid29896_radlex_rid34261>*).
- Asimetrías (*ArrayList<Radlex_rid29896_radlex_rid34265>*).

- Nódulos (*ArrayList<Radlex_rid29896_radlex_rid3874>*).
- Calcificaciones (*ArrayList<Radlex_rid29896_radlex_rid34265>*).
- Un hallazgos asociado (*Radlex_rid29896_trencadis_mamo_trmm0011*).

Parte del código que hemos obtenido con el ejemplo anterior lo vemos en ejemplo de código 8.16. Además de los atributos también generamos métodos para acceder a estos y modificarlos.

```
public class Radlex_rid10357_radlex_rid29896 {
    ArrayList<Radlex_rid29896_radlex_rid34261>
       radlex_rid34261;
    ArrayList<Radlex_rid29896_radlex_rid34265>
       radlex_rid34265;
    ArrayList<Radlex_rid29896_radlex_rid3874>
       radlex_rid3874;
    ArrayList<Radlex_rid29896_radlex_rid5196>
       radlex_rid5196;
    Radlex_rid29896_trencadis_mamo_trmm0011
       trencadis_mamo_trmm0011;
    ...
}
```

Código 8.16: Clase generada automáticamente

Debemos generar dos clases adicionales, en el caso especial de que la interterfaz gráfica asociada al nivel tenga que mostrar una lista desplegable. En este caso, además de los datos del modelo generaremos adicionalmente una clase para gestionar el grupo y una interfaz para los hijos de esta clase.

Siguiendo con el ejemplo anterior, la clase que modela la mama derecha contiene un desplegable para las posibles lesiones, y por lo tanto hemos generado la clase que vemos en el ejemplo de código 8.17 que gestiona el grupo que se visualizará dentro de la lista desplegable y en el ejemplo de código 8.18 tenemos la interfaz de la que extenderán las clases que modelen las desesctructuraciones, las asimetrías, los nodos, las calcificiaciones y el hallazgo asociado.

```
package com.i3m.meer.model;

public class Radlex_rid10357_radlex_rid29896_Group {
    String type;
    ArrayList<Radlex_rid10357_radlex_rid29896>
       radlex_rid10357_radlex_rid29896;

    //Getters and Setters
    ...
```

```
}
```

Código 8.17: Clase generada automáticamente para tratar con el grupo de una lista desplegable

```
1 package com.i3m.meer.model;

3 public interface Radlex_rid10357_radlex_rid29896_Children {
      public String get_Snomed_ct_118522005();
5 }
```

Código 8.18: Interfaz generada automáticamente para gestionar lista desplegable

Al finalizar esta etapa tendremos el modelo de datos del informe DICOM-SR como clases Java con las que puede tratar nuestra aplicación Android.

8.5.4. Generador de la vista

La interfaz de las aplicaciones Android se puede crear de dos manera: mediante ficheros XML que definen la interfaz o mediante código desde la clase que controla la ventana.

Si optamos por la segunda opción no estaremos cumpliendo con el principio de diseño de software MVC, que nos insta a separa la presentación del controlador. Además si definimos la interfaz de usuario en un XML podremos definir tantas distribuciones de la interfaz de usuario como queramos (para adaptar la aplicación a distintas resoluciones, tamaños de pantalla, idiomas, ...) y cargar un fichero XML u otro en tiempo de ejecución dependiendo de las características del dispositivo Android o de las preferencias del usuario. Por lo tanto, nosotros optaremos por la primera opción.

Por una parte la interfaz de usuario la forman los ficheros XML que sitúan cada elemento de la interfaz gráfica y para simplifcar el proceso de internacionalización tenemos otros ficheros con las cadenas de texto que se mostrarán en la interfaz gráfica.

Generador de la interfaz de usuario

Cada pantalla de la aplicación Android corresponde directamente con un nivel del informe DICOM-SR. Un nivel se compone de un contendor que identifica el nivel y puede contener atributos, contenedores hijos o ambos elementos.

Para definir la interfaz de usuario debemos contemplar estas posibilidades y diseñar las plantillas necesarias para que se puedan generar los ficheros XML.

En la figura 8.3 vemos un caso en que el nivel informe médico tiene atributos y contenedores hijos, en este caso sólo hay una opción de distribuir los elementos: una columna para los atributos y la otra para los hijos. Otra posibilidad, que vemos en la figura 8.4 es que el nodo que estamos representando sea una hoja del árbol y sólo tenga atributos, en este caso podríamos usar una o dos columnas para mostrar los atributos, dependiendo de lo que haya indicado el usuario en la configuración. Finalmente en

la figura 8.5 vemos como representar un nodo que solo tiene hijos, se trata de una distribución análoga a la vista para nodos sólo con atributos.

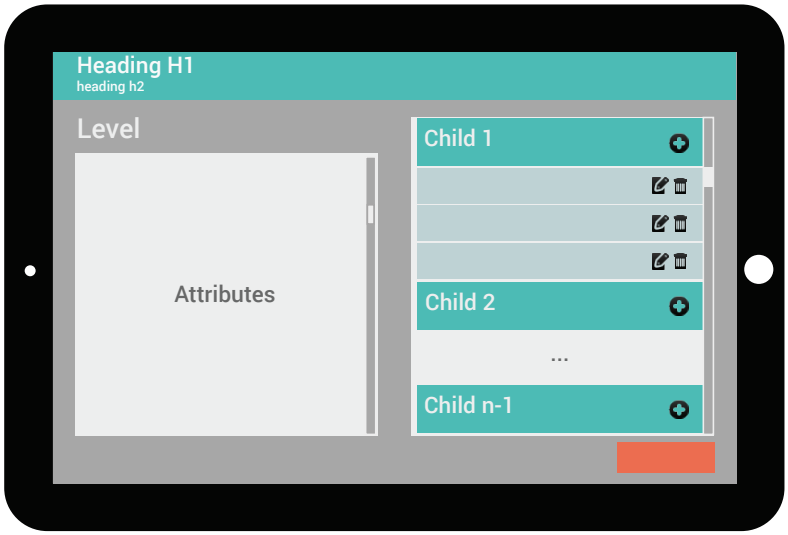


Figura 8.3: Distribución de la interfaz de usuario con atributos e hijos

Figura 8.4: Distribución de la interfaz de usuario con atributos

Diseñamos estas plantillas empleando el lenguaje de definición de plantillas de Jinja. Creamos una plantilla general (*main.xml*), con los bloques de la interfaz de usuario que no varían (franja superior e inferior, segmento central, ...). Del mismo modo crearemos dos plantillas más, que extenderán el comportamiento de la plantilla general para codificar la interfaz de pantalla con una columna (*one_column.xml*) o con dos columnas (*two_columns.xml*).

El resto de los elementos de la interfaz de usuario (botones, campos de validación, campos de texto, ...) se diseñarán también utilizando Jinja. Todas estas plantillas son elementos modulares que se integrarán en la interfaz de usuario.

Figura 8.5: Distribución de la interfaz de usuario con hijos

El programa generador de código se encarga de recorrer en anchura el árbol con el informe DICOM-SR y para cada nivel generar el fichero XML con la interfaz que le corresponda completando las plantillas Jinja con los datos concretos de este nivel.

Este fichero se almacenará en la ruta indicada por el usuario en la configuración.

Generador de las cadenas de texto

Los elementos de la interfaz de usuario que acabamos de crear no muestran ningún texto. En el campo destinado a almacenar una cadena de texto hemos escrito el identificador de un recurso. En este apartado crearemos los ficheros con las cadenas de texto que se mostrarán en el informe.

De nuevo tenemos una serie de plantillas Jinja para definir los recursos de texto que tendrá disponibles la aplicación Android.

Rellenar estas plantillas es mucho más sencillo que en el apartado anterior porque no necesitamos información de la estructura del árbol, únicamente necesitamos un conjunto (sin repetidos) de todos los *CODE_MEANING* que aparecen en el informe DICOM-SR. Extraemos este conjunto del árbol y rellenamos las plantillas.

La única complicación con la que debemos lidiar es la de identificar correctamente en que idioma está escrito el significado (*CODE_MEANING*) de una etiqueta del informe DICOM-SR.

Como en el propio informe no aparece la codificación del idioma,el usuario puede introducir el código en la configuración siguiendo el estándar ISO 3166-1-alpha-2. Este es el estándar que se utiliza en Android y consiste en 2 letras que identifican el idioma, un guión bajo "_" y 2 letras más que identifican el país.

Si el usuario no establece el idioma, el sistema asume que el texto está escrito en inglés si el informe no está internacionalizado y si está internacionalizado le adjudica el inglés al texto dentro de la etiqueta *CODE_MEANING2* y castellano para la etiqueta *CODE_MEANING*.

8.5.5. Generador del controlador

El último elemento que nos falta generar para completar la aplicación Android es el controlador.

En Android la visualización de la interfaz de usuario y la actualización del modelo de datos la realizan las actividades (*Activity*). Una actividad es un componente de una aplicación Android que muestra una pantalla con la que el usuario puede interacturar. Cada actividad es responsable de cargar la interfaz de usuario definida mediante ficheros XML, crear los métodos que escuchen los eventos creados por la interacción con el usuario, interactuar con el modelo y por supuesto gestionar el ciclo de vida de la propia actividad.

El caso general para generar las actividades es análogo al descrito en los apartados 8.5.3 y 8.5.4. En este caso la información de la jerarquía del informe estructurado vuelve a ser relevante, por lo tanto recorreremos el árbol en anchura generando una clase que extiende el funcionamiento de la clase *Activity*.

Al igual que en los casos anteriores hemos diseñado un conjunto de plantillas Jinja2 para modelar estas clases, y estas plantillas las completamos con la interfaz de usuario y los controladores correspondientes a los elementos del contenedor DICOM-SR.

Para que las listas desplegables se comporten de manera adecuada a la funcionalidad que espera el usuario, deberemos crear adaptadores que extiendan el comportamiento básico de estas listas.

De nuevo, tendremos una plantilla Jinja2 para que los contenedores que precisen de listas desplegables las instancien.

Lo último que nos queda generar es el fichero *AndroidManifest.txt*. Toda aplicación Android debe tener este fichero en el directorio raíz donde se encuentra el código.

Este fichero recoge toda la información fundamental de la aplicación al sistema. En este fichero debe figurar los siguientes elementos:

- El nombre del paquete de Java de la aplicación.
- Los componentes de la aplicación Androdid: actividades, indicando cual es la actividad de inicio; servicios; proveedores de contenido; ...
- Los permisos que necesita la aplicación para ejecutarse.
- La API mínima de Android que necesita la aplicación para ejecutarse.

Definimos una plantilla Jinja2 con los elementos del manifiesto Android que instanciamos con los datos de la aplicación Android que hemos generado.

Todos estos ficheros que hemos ido generando se integran dentro de la aplicación Android esqueleto componiendo una aplicación Android específica para el informe estructurado DICOM-SR de entrada.

Hemos conseguido una aplicación Android funcional a partir de un informe estructurado DICOM-SR.

9 | Verificación: un caso práctico

Para verificar el correcto funcionamiento del sistema que hemos creado, hemos llevado a cabo pruebas con distintas plantillas de informe basadas en DICOM-SR.

A modo de ejemplo, en este capítulo veremos cómo funciona el sistema, utilizaremos el fichero XML que contine una plantilla basada en DICOM-SR que podemos encontrar en el apéndice B, y en los siguientes apartados comprobaremos como a partir de este fichero obtenemos una aplicación Android.

9.1. | Análisis sintáctico de un informe plantilla basado en DICOM-SR

En primer lugar realizamos el análisis sintáctico del fichero XML que contiene la plantilla del informe médico estructurado basado en DICOM-SR.

Tras este proceso, tendremos cargada en memoria los datos relativos a la plantilla DICOM-SR en una estructura de datos arbórea. Al imprimir esta estructura de datos por consola (ver en el ejemplo de código 9.1), comprobamos que la información contenida en el árbol es la misma que la que contenía el fichero XML de entrada.

```
1
2   -------  EXPORACIÓN DE MAMA ----------
3
4  - [SNOMED_CT_172117008] EXPLORACIÓN  DE LA MAMA
5  |    - date: Fecha Informe
6  |    - text: Identificador
7  |    - text: Identificador del Paciente
8  ----- [RADLEX_RID29896] MAMA DERECHA
9  --------- [TRENCADIS_MAMO_TRMM0009] MASA
10 |           - text: Identificador
11 |           - num: Cuadrante Supero-Externo (CSE) de la Mama Derecha
12 |           - num: Cuadrante Infero-Externo (CIE) de la Mama Derecha
13 |           - num: Cuadrante Supero-Interno (CSI) de la Mama Derecha
14 |           - num: Cuadrante Infero-Interno (CII) de la Mama Derecha
15 |           - num: Línea Intercudrántica Externa (LIE)
16 |           - num: Línea Intercuadrántica Superior (LIS)
17 |           - num: Línea Intercudrántica Inferior (LIInf)
18 |           - num: Línea Intercudrántica Interna (LIInt)
19 |           - num: Retroareolar de la Mama Derecha
20 |           - num: Pezón de la Mama Derecha
21 |           - num: Areola de la Mama Derecha
```

```
22 |           - num: Prolongación Axilar de la Mama Derecha
23 |           - num: Axila de la Mama Derecha
24 |           - num: Surco Inframamario de la Mama Derecha
25 |           - num: Surco Intermamario de la Mama Derecha
26 ----- [RADLEX_RID29897] MAMA IZQUIERDA
27 --------- [TRENCADIS_MAMO_TRMM0010] ASIMETRIA
28 |           - text: Identificador
29 |           - num: Cuadrante Supero-Externo (CSE) de la Mama Izquierda
30 |           - num: Cuadrante Infero-Externo (CIE) de la Mama Izquierda
31 |           - num: Cuadrante Supero-Interno (CSI) de la Mama Izquierda
32 |           - num: Cuadrante Infero-Interno (CII) de la Mama Izquierda
33 |           - num: Línea Intercudrántica Externa (LIE)
34 |           - num: Línea Intercuadrántica Superior (LIS)
35 |           - num: Línea Intercudrántica Inferior (LIInf)
36 |           - num: Línea Intercudrántica Interna (LIInt)
37 |           - num: Retroareolar de la Mama Izquierda
38 |           - num: Pezón de la Mama Izquierda
39 |           - num: Areola de la Mama Izquierda
40 |           - num: Prolongación Axilar de la Mama Izquierda
41 |           - num: Axila de la Mama Izquierda
42 |           - num: Surco Inframamario de la Mama Izquierda
43 |           - num: Surco Intermamario de la Mama Izquierda
```

Código 9.1: Registro del informe DICOM-SR cargado en memoria

Con el informe cargado en memoria, es el momento de pasar a la generación de los ficheros necesarios para crear una aplicación Android.

9.2. | Generación automática de una aplicación Android

Disponemos de una aplicación Android, cuya estructura hemos visto en 7.2. Esta aplicación carece del contenido necesario para un informe concreto, para completarla necesitamos el código que generamos mediante la aplicación Python que hemos desarrollado.

Como hemos explicado en apartados anteriores generaremos los ficheros necesarios para crear la vista, el modelo y el controlador de una aplicación Android. Para lograrlo recorremos el árbol que contiene la información del informe estructurado de entrada y vamos generando el código de la aplicación. Podemos ver los ficheros que hemos creado en la figura 9.1.

9.3. | Aplicación Android para el informe de entrada

Todos los ficheros de código que hemos generado se integran mediante un sencillo script dentro de la aplicación esqueleto de Android, obteniendo una aplicación funcional en Android.

En las figura 9.2 vemos la pantalla de inicio, en esta pantalla el usuario podrá introducir los datos del informe a la izquierda y a la derecha vemos los órganos que son estudiados en este informe (mama derecha y mama izquierda).

```
outputs/
├── activities/
│   ├── Summary_snomed_ct_172117008.java
│   ├── Tree_snomed_ct_172117008_radlex_rid29896.java
│   ├── Tree_snomed-ct_172117008_radlex_rid29897.java
│   ├── Edit_radlex_rid29897_trencadis_mamo_trmm0010.java
│   ├── Edit_radlex_rid29896_trencadis_mamo_trmm0009.java
│   └── Snomed_ct_172117008_ListAdapter.java
├── layouts
│   ├── tree_snomed_ct_172117008_radlex_rid29897.xml
│   ├── tree_snomed_ct_172117008_radlex_rid29896.xml
│   ├── summary_snomed_ct_172117008.xml
│   ├── edit_radlex_rid29897_trencadis_mamo_trmm0010.xml
│   └── edit_radlex_rid29896_trencadis_mamo_trmm0009.xml
├── manifest
│   └── AndroidManifest.xml
├── model/
│   ├── Snomed_ct_172117008_radlex_rid29897.java
│   ├── Snomed_ct_172117008_radlex_rid29897_Children.java
│   ├── Snomed_ct_172117008_radlex_rid29897_Group.java
│   ├── Snomed_ct_172117008_radlex_rid29896.java
│   ├── Snomed_ct_172117008_radlex_rid29896_Children.java
│   ├── Snomed_ct_172117008_radlex_rid29896_Group.java
│   ├── Snomed_ct_172117008.java
│   ├── Radlex_rid29897_trencadis_mamo_trmm0010.java
│   └── Radlex_rid29896_trencadis_mamo_trmm0009.java
└── strings/
    ├── strings_es_ES.xml
    └── strings.xml
```

Figura 9.1: Estructura de ficheros generados

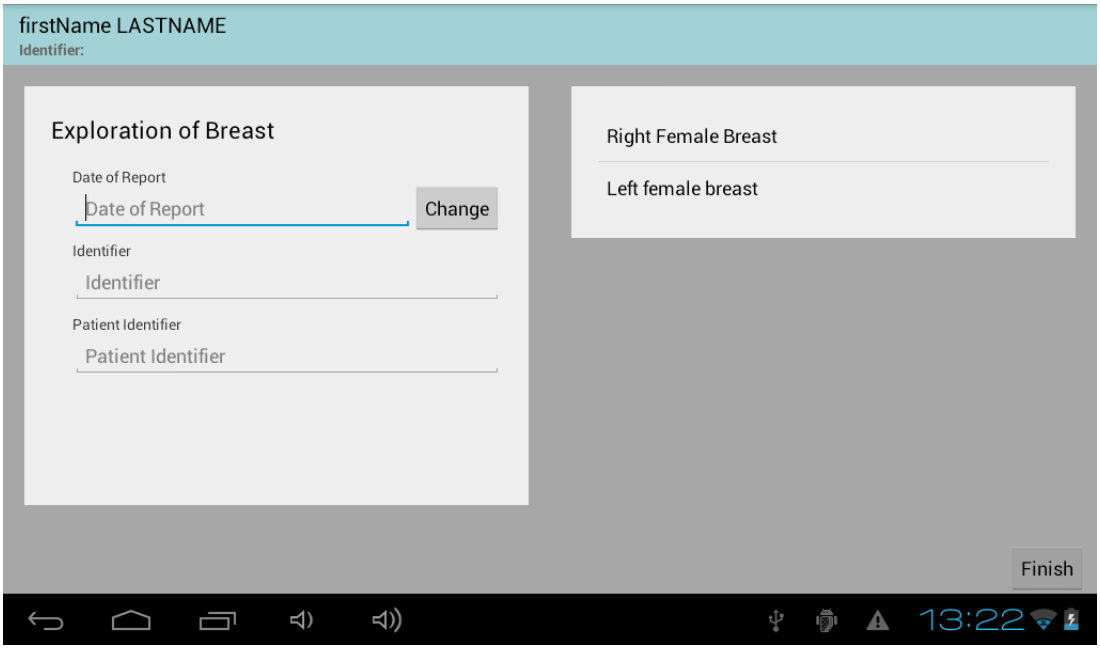


Figura 9.2: Pantalla de inicio de la aplicación Android para una exploración de Mama

Si el usuario decide añadir una lesión en la mama derecha, el sistema le llevará a la pantalla que podemos ver en la figura 9.3. Como en el fichero XML se define que la mama derecha sólo puede contener lesiones de tipo *Masa*, vemos únicamente esta en el listado de lesiones de la mama derecha. El caso de la mama izquierda es análogo, sólo que en este caso en el infome hemos definido que la mama izquierda solamente puede contener lesiones de tipo *Asimetrías*, vemos este ejemplo en la figura 9.4.

Por último veremos las figuras que permiten añadir y editar lesiones del informe. En la figura 9.5 vemos el ejemplo que permite añadir y editar lesiones de tipo *masa*, mientras que en la figura 9.6 podemos ver la pantalla que nos permite añadir y editar lesiones de tipo *asimetría*.

Hemos seleccionado un informe estructurado médico bastante reducido, de modo que el tamaño de los documentos fuera razonable para incluirlos en esta memoria y que contuviera todos los conceptos básicos para explicar el funcionamiento del sistema que hemos desarrollado.

Al tratarse de un ejemplo sencillo, la aplicación resultante también es sencilla. El proceso que hemos descrito puede generalizarse y aplicarse a cualquier fichero XML plantilla basado en XML.

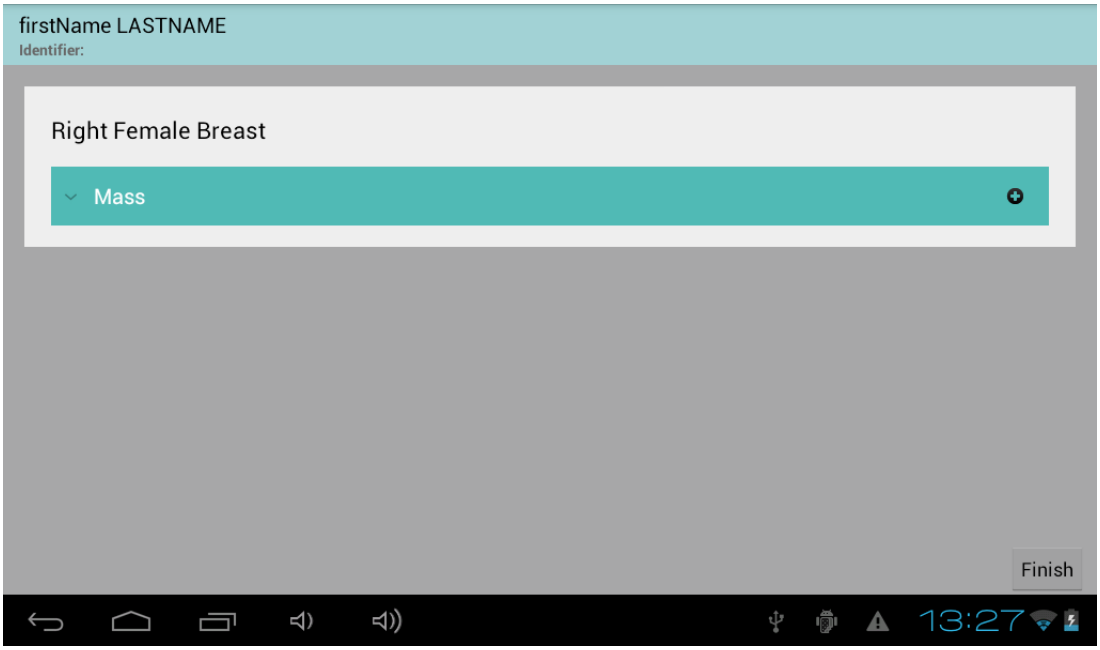


Figura 9.3: Lesiones de la mama derecha

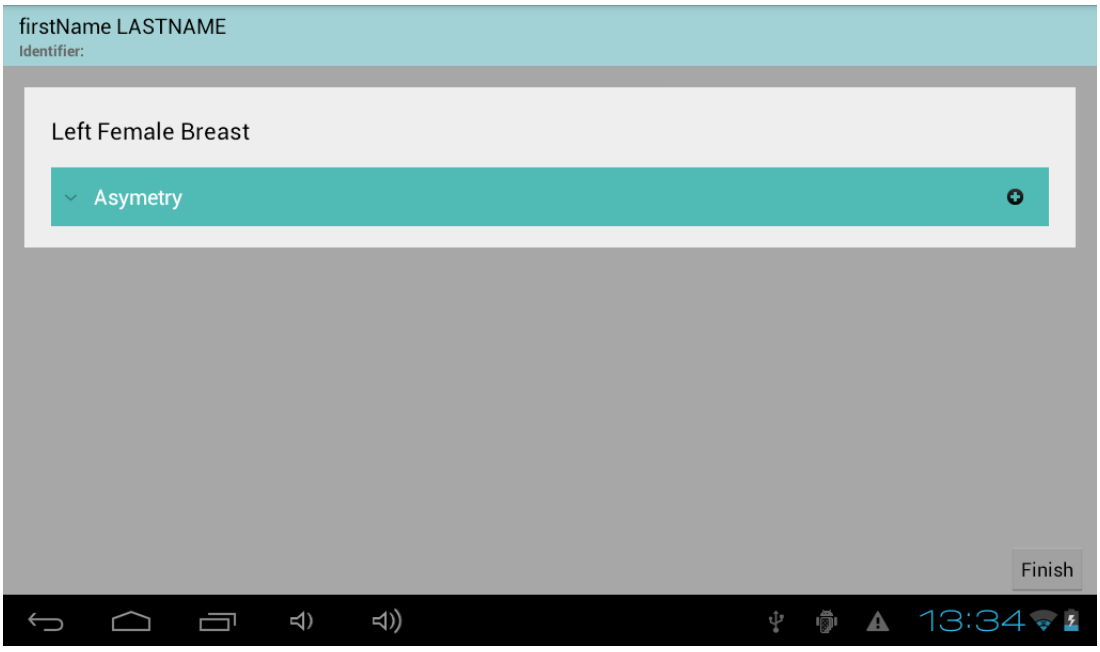


Figura 9.4: Lesiones de la mama izquierda

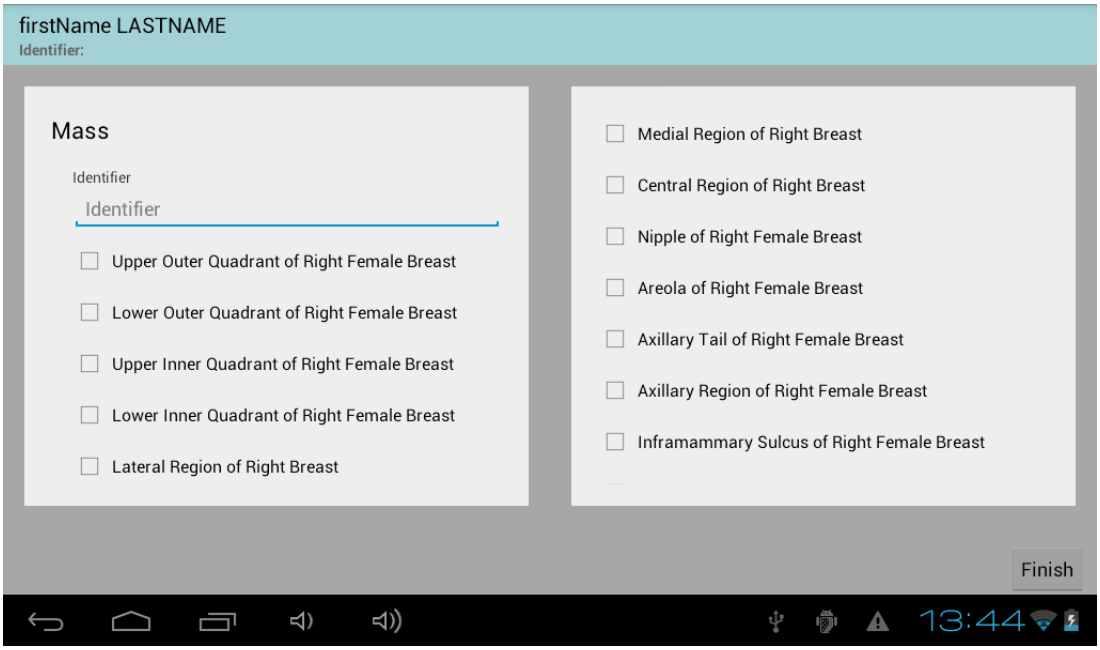


Figura 9.5: Añadir y editar una lesión de tipo masa

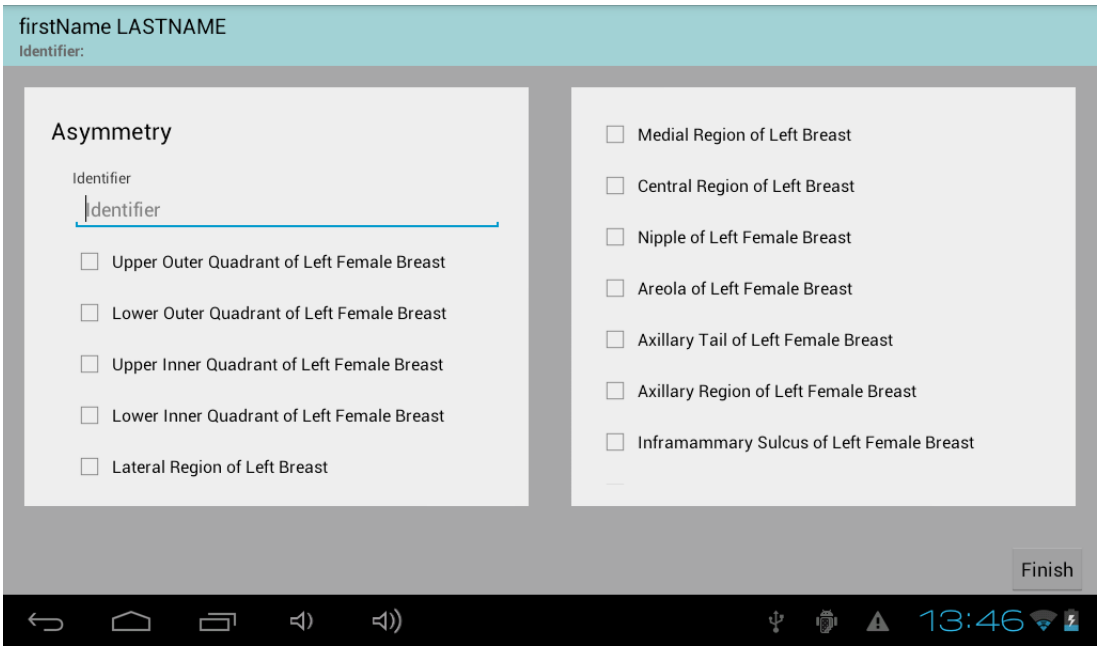


Figura 9.6: Añadir y editar una lesión de tipo asimetría

10 | Conclusiones

A lo largo de esta memoria, hemos descrito el desarrollo de una aplicación que permite a partir de un fichero XML que contiene una plantilla basada en DICOM-SR generar una aplicación Android para la creación de informes.

En este apartado resumiremos el estado del proyecto y las conclusiones a las que hemos llegado.

Nuestro objetivo es que la aplicación final simplificara el proceso de introducir informes médicos estructurados. Para conseguirlo aunamos el desarrollo software con el diseño de una interfaz de usuario comprensible para el personal clínico que hará uso de la misma. Creamos una interfaz de usuario limpia y coherente siguiendo esquemas de diseño estructurado basados en el estilo internacional.

Un diseño correcto de la interfaz de usuario, aprovechando las posibilidades que nos ofrecen las tabletas Android permite que el usuario sea más propenso a utilizar la aplicación, y si a esto le unimos una arquitectura eficiente hemos conseguido crear una experiencia de usuario satisfactoria.

Durante el tiempo de trabajo del PFC hemos conseguido desarrollar: el esqueleto de una aplicación Android y una aplicación generadora de código automática que permite instanciar aplicaciones Android con los datos de un informe estructurado.

Aunque nos quedan cosas que mejorar y que ampliar, que detallaremos en el aparatado 11, el proyecto está en un punto estable y cumple con los requerimientos que nos habíamos marcado.

Una de las partes más interesantes de este proyecto ha sido el de construirlo desde los cimientos. Lo que nos ha permitido aplicar conceptos de la ingeniería del software para planificar el desarrollo, de teoría de lenguajes para crear el analizador sintáctico, de algorítmica y programación orientada a objetos para desarrollar el software generador de código,... En definitiva nos ha dado la posibilidad de poner en práctica múltiples conceptos teóricos vistos a lo largo de la carrera.

Por lo tanto hemos cumplido los objetivos en tanto que hemos sido capaces de desarrollar una aplicación generadora de código automático para Android que simplificará la obtención de informes médicos estructurados poniendo en práctica el conocimiento adquirido en la Ingeniería Informática.

11 | Futuras mejoras

Como en todo proyecto siempre hay partes que podrían mejorar y evolucionar. Detectar estas partes del proyecto y proporcionar una solución es una de nuestras tareas, que permitirá al proyecto seguir creciendo y evolucioralo.

11.1. | Soportar más elementos del estándar DICOM-SR

La mejora más obvia es esta: soportar más elementos del estándar DICOM-SR.

Para hacer viable este proyecto en el tiempo de desarrollo previsto soportamos un conjunto limitado de tipos de datos y de atributos del estándar DICOM-SR. Sin embargo, este estándar es muy amplio e incluye muchos más elementos de los que nosotros soportamos, de modo que la primera mejora fundamental será ampliar el conjunto de elementos soportados.

Deberemos crear nuevas plantillas para acomodar estos elementos dentro de la aplicación Android.

Por otra parte, un contenedor puede tener definidas una serie de propiedades que el dato introducido por usuario en este contenedor debe cumplir para que sea un dato válido. Deberíamos crear métodos para comprobar que se cumplan estos requisitos adicionales.

11.2. | Ampliar las posibilidades de personalización de la interfaz de usuario

Siguiendo con la idea anterior, otro punto interesante para ampliar el PFC será el de crear nuevas posibilidades de personalización de la interfaz de usuario.

Hasta ahora, cada elemento (número, texto,...) tiene una plantilla asociada con la distribución que tendrá este elemento en la interfaz de usuario. Pero dependiendo del tipo de informe o de la posición que ocupe el campo dentro del mismo esta distribución podrá ser inapropiada, por lo que se propone crear para cada elemento un conjunto de plantillas que lo presenten y que el sistema y/o el usuario pueda decidir cual es la plantilla más apropiada para el caso concreto que se esté tratando.

El otro punto de la interfaz gráfica que podemos mejorar es permitir personalizar la distribución de los elementos dentro de la cuadrícula. Ahora el sistema muestra la mitad

de los elementos en una parte y la otra mitad en la otra. Cuando se trata de atributos e hijos le asigna la mitad derecha a los hijos y la izquierda a los atributos, pero si sólo tenemos hijos o atributos divide los elementos según el orden en el que aparecen en el informe.

Permitir que el usuario determine que elementos deben aparecer en cada parte permitirá crear una interfaz de usuario coherente conceptualmente.

11.3. | Paralelizar el proceso de generación de código

Esta mejora es simple, puesto que el sistema ya está preparado y permitirá acelerar el proceso de generación de código.

Una vez tenemos el árbol cargado en memoria podemos lanzar un hilo para cada elemento de la aplicación Android que tenemos que generar (modelo, vista y controlador). Cada generador esta programado de forma independientes y como no se modifica el árbol que contiene el informe y cada elemento que debe ser generado se escribe sobre un fichero diferente, no existen problemas de dependencias que compliquen la implementación de esta mejora.

11.4. | Incrustar el sistema dentro de una aplicación web

Las aplicaciones de consola son muy útiles pero poco intuitivas para el usuario final. Así que lo ideal sería disponer de una aplicación web en la que el personal médico pueda introducir en fichero XML con la plantilla basada en DICOM-SR, personalizar la aplicación Android para que se adapte a sus necesidades concretas y obtener como resultado la aplicación Android.

Existen distintos frameworks en Python para programar aplicaciones web. Podríamos por ejemplo crear una aplicación Django e incluir en el controlador de esta el generador de código que hemos programado en Python. Con lo que simplemente incluyendo una capa de abstracción a nuestro sistema dispondremos de una aplicación web.

El usuario podría seleccionar gráficamente la plantilla que desea para cada elemento del informe DICOM-SR así como su distribución dentro de la cuadrícula, también podría personalizar aspectos gráficos como el esquema de color. De este modo el usuario podrá adaptar la aplicación final al informe de entrada y a sus preferencias personales.

12 | Bibliografía

[1] Kunio Doi. Computer-aided diagnosis in medical imaging: Historical review, current status and future potential. *Comp. Med. Imag. and Graph.*, 31(4-5):198–211, 2007. URL http://dblp.uni-trier.de/db/journals/cmig/cmig31.html#Doi07.

[2] Robert M. Nishikawa. Current status and future directions of computer-aided diagnosis in mammography. *Comp. Med. Imag. and Graph.*, 31(4-5):224–235, 2007. URL http://dblp.uni-trier.de/db/journals/cmig/cmig31.html#Nishikawa07.

[3] Mary E Johnston, Karl B Langton, R Brian Haynes, and Alix Mathieu. Effects of computer-based clinical decision support systems on clinician performance and patient outcome: a critical appraisal of research. *Annals of internal medicine*, 120 (2):135–142, 1994.

[4] Maurice R. M. Samulski, Peter R. Snoeren, Bram Platel, Bram van Ginneken, Laurens Hogeweg, Cornelia Schaefer-Prokop, and Nico Karssemeijer. Computer-aided detection as a decision assistant in chest radiography, 2011. URL http://dx.doi.org/10.1117/12.877968.

[5] Harold L Kundel and Calvin F Nodine. Interpreting chest radiographs without visual search. *Radiology*, 116(3):527–532, 1975.

[6] Hiroshi Fujita, Yoshikazu Uchiyama, Toshiaki Nakagawa, Daisuke Fukuoka, Yuji Hatanaka, Takeshi Hara, Gobert N. Lee, Yoshinori Hayashi, Yuji Ikedo, Xin Gao, and Xiangrong Zhou. Computer-aided diagnosis: The emerging of three cad systems induced by japanese health care needs. *Comput. Methods Prog. Biomed.*, 92 (3):238–248, 2008. URL http://dx.doi.org/10.1016/j.cmpb.2008.04.003.

[7] HK Huang. *PACS and imaging informatics: basic principles and applications*. Wiley. com, 2010.

[8] Oleg S Pianykh. *Digital imaging and communications in medicine (DICOM): a practical introduction and survival guide*. Springer, 2012.

[9] HK Huang. Short history of pacs. part i: Usa. *European journal of radiology*, 78 (2):163–176, 2011.

[10] Heinz U Lemke. Short history of pacs (part ii: Europe). *European journal of radiology*, 78(2):177–183, 2011.

[11] W Dean Bidgood and Steven C Horii. Introduction to the acr-nema dicom standard. *Radiographics*, 12(2):345–355, 1992.

[12] Cor Loef. Trends and future in dicom standardization. DICOM 2013 International Conference and Seminar Program, 2013.

[13] José Salavert Torres, J Damià Segrelles Quilis, Ignacio Blanquer Espert, and Vicente Hernandez García. Improving knowledge management through the support of image examination and data annotation using dicom structured reporting. *Journal of Biomedical Informatics*, 2012.

[14] D.A. Clunie. *DICOM Structured Reporting*. PixelMed, 2000. URL `http://books.google.com.mx/books?id=EVjOolUJNGUC`.

[15] David L. Sackett. Evidence-based medicine. *Seminars in Perinatology*, 21(1): 3 – 5, 1997. URL `http://www.sciencedirect.com/science/article/pii/S0146000597800134`. <ce:title>Fatal and Neonatal Hematology for the 21st Century</ce:title>.

[16] Razvigor Borislavov Darlenski, Neyko Valentinov Neykov, Vitan Dakov Vlahov, and Nikolaï Konstantinov Tsankov. Evidence-based medicine: Facts and controversies. *Clinics in Dermatology*, 28(5):553 – 557, 2010. URL `http://www.sciencedirect.com/science/article/pii/S0738081X10000398`. Controversies in Dermatology: Part III.

[17] Heather E. Elphick and Rosalind L. Smyth. Research: the principles of evidence-based medicine. *Current Paediatrics*, 14(6):525 – 531, 2004. URL `http://www.sciencedirect.com/science/article/pii/S0957583904000880`. <ce:title>Research in Paediatrics</ce:title>.

[18] Ignacio Blanquer Espert, Vicente Hernández García, Fco. Javier Meseguer Anastásio, and J. Damià Segrelles Quilis. Content-based organisation of virtual repositories of dicom objects. *Future Gener. Comput. Syst.*, 25(6):627–637, 2009. URL `http://dx.doi.org/10.1016/j.future.2008.12.004`.

[19] Jose Salavert Torres, J. Damiàn Segrelles Quilis, Ignacio Blanquer Espert, and Vicente Hernández. Improving knowledge management through the support of image examination and data annotation using dicom structured reporting. *Journal of Biomedical Informatics*, 45(6):1066–1074, 2012. URL `http://dblp.uni-trier.de/db/journals/jbi/jbi45.html#TorresQEH12`.

[20] Alfredo Tirado-Ramos, Jingkun Hu, and Kwok Pun Lee. Model formulation: Information object definition-based unified modeling language representation of dicom structured reporting: A case study of transcoding dicom to xml. *JAMIA*, 9(1): 63–72, 2002. URL http://dblp.uni-trier.de/db/journals/jamia/jamia9.html#Tirado-RamosHL02.

[21] M. Sevenster, R. van Ommering, and Y. Qian. Automatically correlating clinical findings and body locations in radiology reports using medlee. *J Digit Imaging*, 25 (2):240–9, 2012.

[22] Informatics in radiology: Automated structured reporting of imaging findings using the aim standard and xml. *Radiographics*, 2011. http://dx.doi.org/10.1148/rg.313105195.

[23] Extensible markup language (XML) in health care: integration of structured reporting and decision support. *Proc AMIA Symp*, pages 725–729, 1998. URL http://view.ncbi.nlm.nih.gov/pubmed/9929314.

[24] Asim F. Choudhri and Martin G. Radvany. Initial experience with a handheld device digital imaging and communications in medicine viewer: Osirix mobile on the iphone. *J. Digital Imaging*, 24(2):184–189, 2011. URL http://dblp.uni-trier.de/db/journals/jdi/jdi24.html#ChoudhriR11.

[25] Tomonori Yamauchi, Masashi Yamazaki, Akihiko Okawa, Takeo Furuya, Koichi Hayashi, Tsuyoshi Sakuma, Hiroshi Takahashi, Noriyuki Yanagawa, and Masao Koda. Efficacy and reliability of highly functional open source dicom software (osirix) in spine surgery. *J Clin Neurosci*, 17(6):756–9, 2010. URL http://www.biomedsearch.com/nih/Efficacy-reliability-highly-functional-open/20359897.html.

[26] Fuk hay Tang, Maria Y. Y. Law, Ares C. H. Lee, and Lawrence Wing-Chi Chan. A mobile phone integrated health care delivery system of medical images. *J. Digital Imaging*, 17(3):217–225, 2004. URL http://dblp.uni-trier.de/db/journals/jdi/jdi17.html#TangLLC04.

[27] John Mantas, Elske Ammenwerth, George Demiris, Arie Hasman, Reinhold Haux, William Hersh, Evelyn Hovenga, KC Lun, Heimar Marin, Fernando Martin-Sanchez, et al. Recommendations of the international medical informatics association (imia) on education in biomedical and health informatics–1st revision. *Methods of information in medicine*, 49:105–120, 2010.

[28] Wikipedia. Imagen médica — Wikipedia, the free encyclopedia, 2013. URL http://es.wikipedia.org/wiki/Imagen_m%C3%A9dica.

[29] Henning Müller, Nicolas Michoux, David Bandon, and Antoine Geissbuhler. A review of content-based image retrieval systems in medical applications—clinical

benefits and future directions. *International Journal of Medical Informatics*, 73(1): 1 – 23, 2004. URL http://www.sciencedirect.com/science/article/pii/S1386505603002119.

[30] J.B.Antoine Maintz and Max A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1 – 36, 1998. URL http://www.sciencedirect.com/science/article/pii/S1361841501800268.

[31] Asociación Nacional de Fabricantes eléctricos (NEMA). Digital imaging and communications in medicine (dicom), 2013. URL http://medical.nema.org/.

[32] M.W. Whalen and M.P.E. Heimdahl. An approach to automatic code generation for safety-critical systems. In *Automated Software Engineering, 1999. 14th IEEE International Conference on.*, pages 315–318, 1999.

[33] Michael G Hinchey, James L Rash, and Christopher A Rouff. Requirements to design to code: Towards a fully formal approach to automatic code generation. *NASA tech. monograph TM-2005-212774, NASA Goddard Space Flight Center*, 2005.

[34] Jack Herrington. *Code generation in action*. Manning Publications Co., 2003.

[35] Pedro J. Molina. An introduction to crossbow hunting. SlideShare, 2009. URL http://www.slideshare.net/genexus/026-estado-del-arte-de-mdd-model-driven-development.

[36] Oscar Pastor, Sergio España, José Ignacio Panach, and Nathalie Aquino. Model-driven development. *Informatik-Spektrum*, 31(5):394–407, 2008. ISSN 0170-6012. doi: 10.1007/s00287-008-0275-8. URL http://dx.doi.org/10.1007/s00287-008-0275-8.

[37] Alan Dix. *Human computer interaction*. Pearson Education, 2004.

[38] Rada Hussein, Uwe Engelmann, Andre Schroeter, and Hans-Peter Meinzer. Dicom structured reporting part 1. overview and characteristics1. *Radiographics*, 24(3): 891–896, 2004.

[39] Rada Hussein, Uwe Engelmann, Andre Schroeter, and Hans-Peter Meinzer. Dicom structured reporting part 2. problems and challenges in implementation for pacs workstations1. *Radiographics*, 24(3):897–909, 2004.

[40] Alfredo Tirado-Ramos, Jingkun Hu, and Kwok Pun Lee. Information object definition–based unified modeling language representation of dicom structured reporting a case study of transcoding dicom to xml. *Journal of the American Medical Informatics Association*, 9(1):63–72, 2002.

[41] David A. Clunie. Dicom structured reporting: an object model as an implementation boundary, 2001. URL http://dx.doi.org/10.1117/12.435478.

[42] Dicom structured reporting—a prototype implementation. *International Congress Series*, 1230(0):795 – 800, 2001. ISSN 0531-5131. doi: http://dx.doi.org/10.1016/S0531-5131(01)00137-6.

[43] Rita Noumeir. Benefits of the dicom structured report. *Journal of digital imaging*, 19(4):295–306, 2006.

[44] Curtis P Langlotz. Radlex: A new method for indexing online educational materials1. *Radiographics*, 26(6):1595–1597, 2006.

[45] World Health Organization. *ICD-10: International statistical classification of diseases and related health problems*. World Health Organization, 2004.

[46] Michael Q Stearns, Colin Price, Kent A Spackman, and Amy Y Wang. Snomed clinical terms: overview of the development process and project status. In *Proceedings of the AMIA Symposium*, page 662. American Medical Informatics Association, 2001.

[47] International Health Terminology Standards Development Organisation (IHTSDO). Members of ihtsdo, 2013. URL `http://www.ihtsdo.org/members/`.

[48] Emmanuel Uren, Robert Howard, and Tiziana Perinotti. *Introduction to Software Internationalization and Localization*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1993. ISBN 0442014988.

[49] Wikipedia. Smartphones — Wikipedia, the free encyclopedia, 2013. URL `http://en.wikipedia.org/wiki/Smartphone`.

[50] Wikipedia. Android — Wikipedia, the free encyclopedia, 2013. URL `http://en.wikipedia.org/wiki/Android_(operating_system)`.

[51] Google. Android developers, 2013. URL `http://developer.android.com/about/dashboards/index.html`.

[52] Python Software Foundation. Python, 2013. URL `http://www.python.org/`.

[53] Python Software Foundation. 2to3 - automated python 2 to 3 code translation, 2013. URL `http://docs.python.org/3.1/library/2to3.html`.

[54] Adobe Systems Inc. Phonegap, 2013. URL `http://phonegap.com/`.

[55] Appcelerator Inc. Titanium, 2013. URL `http://www.appcelerator.com/platform/titanium-sdk/`.

[56] Corona Labs Inc. Corona, 2013. URL `http://www.coronalabs.com/products/corona-sdk/`.

[57] Kivy Developers. Kivy, 2013. URL `http://kivy.org/`.

[58] Google. Android design guides, 2013. URL http://developer.android.com/design/index.html.

[59] Mayte Giménez Fayos. Meer: Android framework, 2013. URL https://github.com/mshopper/meer-android.

[60] Mayte Giménez Fayos. Meer, 2013. URL https://github.com/mshopper/meer-xml.

[61] Chengkai Li. Xml parsing, sax/dom., 2009.

[62] Armin Ronacher. Jinja2, 2013. URL http://jinja.pocoo.org/.

A | Informe DICOM-SR

```
1
2  <?xml version="1.0" encoding="UTF-8"?>
3
4  <!-- DICOM HEADER -->
5  <DICOM_SR Description="Exploration of breast" IDOntology="4">
6    <CONTAINER>
7      <CONCEPT_NAME>
8        <CODE_VALUE>172117008</CODE_VALUE>
9        <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
10       <CODE_MEANING>Exploración  de la Mama</CODE_MEANING>
11       <CODE_MEANING2>Exploration of Breast</CODE_MEANING2>
12     </CONCEPT_NAME>
13     <PROPERTIES>
14       <CARDINALITY max="1" min="1"/>
15       <CONDITION_TYPE type="M"/>
16       <EXPRESION_CONDITION xquery=""/>
17     </PROPERTIES>
18     <CHILDS>
19       <DATE>
20         <CONCEPT_NAME>
21           <CODE_VALUE>399651003</CODE_VALUE>
22           <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
23           <CODE_MEANING>Fecha Informe</CODE_MEANING>
24           <CODE_MEANING2>Date of Report</CODE_MEANING2>
25         </CONCEPT_NAME>
26         <PROPERTIES>
27           <CARDINALITY max="1" min="1"/>
28           <CONDITION_TYPE type="M"/>
29           <EXPRESION_CONDITION xquery=""/>
30           <DEFAULT_VALUE value=""/>
31         </PROPERTIES>
32       </DATE>
33       <TEXT>
34         <CONCEPT_NAME>
35           <CODE_VALUE>118522005</CODE_VALUE>
36           <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
37           <CODE_MEANING>Identificador</CODE_MEANING>
38           <CODE_MEANING2>Identifier</CODE_MEANING2>
39         </CONCEPT_NAME>
40         <PROPERTIES>
41           <CARDINALITY max="1" min="1"/>
```

```
42            <CONDITION_TYPE type="M"/>
43            <EXPRESION_CONDITION xquery=""/>
44            <DEFAULT_VALUE value=""/>
45          </PROPERTIES>
46        </TEXT>
47        <TEXT>
48          <CONCEPT_NAME>
49            <CODE_VALUE>RID13159</CODE_VALUE>
50            <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
51            <CODE_MEANING>Identificador del Paciente</CODE_MEANING>
52            <CODE_MEANING2>Patient Identifier</CODE_MEANING2>
53          </CONCEPT_NAME>
54          <PROPERTIES>
55            <CARDINALITY max="1" min="1"/>
56            <CONDITION_TYPE type="M"/>
57            <EXPRESION_CONDITION xquery=""/>
58            <DEFAULT_VALUE value=""/>
59          </PROPERTIES>
60        </TEXT>
61
62        <!-- RIGHT FEMALE BREAST -->
63        <CONTAINER>
64          <CONCEPT_NAME>
65            <CODE_VALUE>RID29896</CODE_VALUE>
66            <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
67            <CODE_MEANING>Mama Derecha</CODE_MEANING>
68            <CODE_MEANING2>Right Female Breast</CODE_MEANING2>
69          </CONCEPT_NAME>
70          <PROPERTIES>
71            <CARDINALITY max="1" min="1"/>
72            <CONDITION_TYPE type="M"/>
73            <EXPRESION_CONDITION xquery=""/>
74          </PROPERTIES>
75
76          <!-- RIGHT FEMALE BREAST INJURIES -->
77          <CHILDS>
78            <CONTAINER>
79              <CONCEPT_NAME>
80                <CODE_VALUE>TRMM0009</CODE_VALUE>
81                <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
82                <CODE_MEANING>Masa</CODE_MEANING>
83                <CODE_MEANING2>Mass</CODE_MEANING2>
84              </CONCEPT_NAME>
85              <PROPERTIES>
86                <CARDINALITY max="-1" min="0"/>
87                <CONDITION_TYPE type="U"/>
88                <EXPRESION_CONDITION xquery=""/>
89              </PROPERTIES>
90              <CHILDS>
91                <TEXT>
92                  <CONCEPT_NAME>
93                    <CODE_VALUE>118522005</CODE_VALUE>
```

```
94                    <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
95                    <CODE_MEANING>Identificador</CODE_MEANING>
96                    <CODE_MEANING2>Identifier</CODE_MEANING2>
97                  </CONCEPT_NAME>
98                  <PROPERTIES>
99                    <CARDINALITY max="1" min="1"/>
100                   <CONDITION_TYPE type="M"/>
101                   <EXPRESION_CONDITION xquery=""/>
102                   <DEFAULT_VALUE value=""/>
103                 </PROPERTIES>
104               </TEXT>
105               <NUM>
106                 <CONCEPT_NAME>
107                   <CODE_VALUE>RID29929</CODE_VALUE>
108                   <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
109                   <CODE_MEANING>Cuadrante Supero-Externo (CSE) de la Mama
                          Derecha</CODE_MEANING>
110                   <CODE_MEANING2>Upper Outer Quadrant of Right Female
                         Breast</CODE_MEANING2>
111                 </CONCEPT_NAME>
112                 <PROPERTIES>
113                   <CARDINALITY max="1" min="1"/>
114                   <CONDITION_TYPE type="M"/>
115                   <EXPRESION_CONDITION xquery=""/>
116                   <DEFAULT_VALUE value="0"/>
117                   <UNIT_MEASUREMENT>
118                     <CONCEPT_NAME>
119                       <CODE_VALUE>000000001</CODE_VALUE>
120                       <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
121                       <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
122                       <CODE_MEANING2>Boolean Units</CODE_MEANING2>
123                     </CONCEPT_NAME>
124                   </UNIT_MEASUREMENT>
125                 </PROPERTIES>
126               </NUM>
127               <NUM>
128                 <CONCEPT_NAME>
129                   <CODE_VALUE>RID29935</CODE_VALUE>
130                   <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
131                   <CODE_MEANING>Cuadrante Infero-Externo (CIE) de la Mama
                          Derecha</CODE_MEANING>
132                   <CODE_MEANING2>Lower Outer Quadrant of Right Female
                         Breast</CODE_MEANING2>
133                 </CONCEPT_NAME>
134                 <PROPERTIES>
135                   <CARDINALITY max="1" min="1"/>
136                   <CONDITION_TYPE type="M"/>
137                   <EXPRESION_CONDITION xquery=""/>
138                   <DEFAULT_VALUE value="0"/>
139                   <UNIT_MEASUREMENT>
140                     <CONCEPT_NAME>
141                       <CODE_VALUE>000000001</CODE_VALUE>
```

```
142                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
143                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
144                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
145                       </CONCEPT_NAME>
146                     </UNIT_MEASUREMENT>
147                   </PROPERTIES>
148                 </NUM>
149                 <NUM>
150                   <CONCEPT_NAME>
151                     <CODE_VALUE>RID29932</CODE_VALUE>
152                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
153                     <CODE_MEANING>Cuadrante Supero-Interno (CSI) de la Mama
                          Derecha</CODE_MEANING>
154                     <CODE_MEANING2>Upper Inner Quadrant of Right Female
                        Breast</CODE_MEANING2>
155                   </CONCEPT_NAME>
156                   <PROPERTIES>
157                     <CARDINALITY max="1" min="1"/>
158                     <CONDITION_TYPE type="M"/>
159                     <EXPRESION_CONDITION xquery=""/>
160                     <DEFAULT_VALUE value="0"/>
161                     <UNIT_MEASUREMENT>
162                       <CONCEPT_NAME>
163                         <CODE_VALUE>000000001</CODE_VALUE>
164                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
165                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
166                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
167                       </CONCEPT_NAME>
168                     </UNIT_MEASUREMENT>
169                   </PROPERTIES>
170                 </NUM>
171                 <NUM>
172                   <CONCEPT_NAME>
173                     <CODE_VALUE>RID29938</CODE_VALUE>
174                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
175                     <CODE_MEANING>Cuadrante Infero-Interno (CII) de la Mama
                          Derecha</CODE_MEANING>
176                     <CODE_MEANING2>Lower Inner Quadrant of Right Female
                        Breast</CODE_MEANING2>
177                   </CONCEPT_NAME>
178                   <PROPERTIES>
179                     <CARDINALITY max="1" min="1"/>
180                     <CONDITION_TYPE type="M"/>
181                     <EXPRESION_CONDITION xquery=""/>
182                     <DEFAULT_VALUE value="0"/>
183                     <UNIT_MEASUREMENT>
184                       <CONCEPT_NAME>
185                         <CODE_VALUE>000000001</CODE_VALUE>
186                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
187                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
188                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
189                       </CONCEPT_NAME>
```

```
190                     </UNIT_MEASUREMENT>
191                   </PROPERTIES>
192                 </NUM>
193                 <NUM>
194                   <CONCEPT_NAME>
195                     <CODE_VALUE>RID29947</CODE_VALUE>
196                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
197                     <CODE_MEANING>Línea Intercudrántica Externa (LIE) de la
                            Mama Derecha</CODE_MEANING>
198                     <CODE_MEANING2>Lateral Region of Right Breast</
                          CODE_MEANING2>
199                   </CONCEPT_NAME>
200                   <PROPERTIES>
201                     <CARDINALITY max="1" min="1"/>
202                     <CONDITION_TYPE type="M"/>
203                     <EXPRESION_CONDITION xquery=""/>
204                     <DEFAULT_VALUE value="0"/>
205                     <UNIT_MEASUREMENT>
206                       <CONCEPT_NAME>
207                         <CODE_VALUE>000000001</CODE_VALUE>
208                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
209                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
210                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
211                       </CONCEPT_NAME>
212                     </UNIT_MEASUREMENT>
213                   </PROPERTIES>
214                 </NUM>
215                 <NUM>
216                   <CONCEPT_NAME>
217                     <CODE_VALUE>RID29941</CODE_VALUE>
218                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
219                     <CODE_MEANING>Línea Intercuadrántica Superior (LIS) de
                          la Mama Derecha</CODE_MEANING>
220                     <CODE_MEANING2>Superior Region of Right Breast</
                          CODE_MEANING2>
221                   </CONCEPT_NAME>
222                   <PROPERTIES>
223                     <CARDINALITY max="1" min="1"/>
224                     <CONDITION_TYPE type="M"/>
225                     <EXPRESION_CONDITION xquery=""/>
226                     <DEFAULT_VALUE value="0"/>
227                     <UNIT_MEASUREMENT>
228                       <CONCEPT_NAME>
229                         <CODE_VALUE>000000001</CODE_VALUE>
230                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
231                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
232                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
233                       </CONCEPT_NAME>
234                     </UNIT_MEASUREMENT>
235                   </PROPERTIES>
236                 </NUM>
237                 <NUM>
```

```
238                   <CONCEPT_NAME>
239                     <CODE_VALUE>RID29953</CODE_VALUE>
240                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
241                     <CODE_MEANING>Línea Intercudrántica Inferior (LIInf) de
                            la Mama Derecha</CODE_MEANING>
242                     <CODE_MEANING2>Subareolar Region of Right Breast</
                          CODE_MEANING2>
243                   </CONCEPT_NAME>
244                   <PROPERTIES>
245                     <CARDINALITY max="1" min="1"/>
246                     <CONDITION_TYPE type="M"/>
247                     <EXPRESION_CONDITION xquery=""/>
248                     <DEFAULT_VALUE value="0"/>
249                     <UNIT_MEASUREMENT>
250                       <CONCEPT_NAME>
251                         <CODE_VALUE>000000001</CODE_VALUE>
252                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
253                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
254                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
255                       </CONCEPT_NAME>
256                     </UNIT_MEASUREMENT>
257                   </PROPERTIES>
258                 </NUM>
259                 <NUM>
260                   <CONCEPT_NAME>
261                     <CODE_VALUE>RID29944</CODE_VALUE>
262                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
263                     <CODE_MEANING>Línea Intercudrántica Interna (LIInt) de
                          la Mama Derecha</CODE_MEANING>
264                     <CODE_MEANING2>Medial Region of Right Breast</
                          CODE_MEANING2>
265                   </CONCEPT_NAME>
266                   <PROPERTIES>
267                     <CARDINALITY max="1" min="1"/>
268                     <CONDITION_TYPE type="M"/>
269                     <EXPRESION_CONDITION xquery=""/>
270                     <DEFAULT_VALUE value="0"/>
271                     <UNIT_MEASUREMENT>
272                       <CONCEPT_NAME>
273                         <CODE_VALUE>000000001</CODE_VALUE>
274                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
275                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
276                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
277                       </CONCEPT_NAME>
278                     </UNIT_MEASUREMENT>
279                   </PROPERTIES>
280                 </NUM>
281                 <NUM>
282                   <CONCEPT_NAME>
283                     <CODE_VALUE>RID29950</CODE_VALUE>
284                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
285                     <CODE_MEANING>Retroareolar de la Mama Derecha</
```

```
                    CODE_MEANING>
286                 <CODE_MEANING2>Central Region of Right Breast</
                    CODE_MEANING2>
287               </CONCEPT_NAME>
288               <PROPERTIES>
289                 <CARDINALITY max="1" min="1"/>
290                 <CONDITION_TYPE type="M"/>
291                 <EXPRESION_CONDITION xquery=""/>
292                 <DEFAULT_VALUE value="0"/>
293                 <UNIT_MEASUREMENT>
294                   <CONCEPT_NAME>
295                     <CODE_VALUE>000000001</CODE_VALUE>
296                     <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
297                     <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
298                     <CODE_MEANING2>Boolean Units</CODE_MEANING2>
299                   </CONCEPT_NAME>
300                 </UNIT_MEASUREMENT>
301               </PROPERTIES>
302             </NUM>
303             <NUM>
304               <CONCEPT_NAME>
305                 <CODE_VALUE>RID29907</CODE_VALUE>
306                 <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
307                 <CODE_MEANING>Pezón de la Mama Derecha</CODE_MEANING>
308                 <CODE_MEANING2>Nipple of Right Female Breast</
                    CODE_MEANING2>
309               </CONCEPT_NAME>
310               <PROPERTIES>
311                 <CARDINALITY max="1" min="1"/>
312                 <CONDITION_TYPE type="M"/>
313                 <EXPRESION_CONDITION xquery=""/>
314                 <DEFAULT_VALUE value="0"/>
315                 <UNIT_MEASUREMENT>
316                   <CONCEPT_NAME>
317                     <CODE_VALUE>000000001</CODE_VALUE>
318                     <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
319                     <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
320                     <CODE_MEANING2>Boolean Units</CODE_MEANING2>
321                   </CONCEPT_NAME>
322                 </UNIT_MEASUREMENT>
323               </PROPERTIES>
324             </NUM>
325             <NUM>
326               <CONCEPT_NAME>
327                 <CODE_VALUE>RID29918</CODE_VALUE>
328                 <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
329                 <CODE_MEANING>Areola de la Mama Derecha</CODE_MEANING>
330                 <CODE_MEANING2>Areola of Right Female Breast</
                    CODE_MEANING2>
331               </CONCEPT_NAME>
332               <PROPERTIES>
333                 <CARDINALITY max="1" min="1"/>
```

```
334                     <CONDITION_TYPE type="M"/>
335                     <EXPRESION_CONDITION xquery=""/>
336                     <DEFAULT_VALUE value="0"/>
337                     <UNIT_MEASUREMENT>
338                       <CONCEPT_NAME>
339                         <CODE_VALUE>000000001</CODE_VALUE>
340                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
341                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
342                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
343                       </CONCEPT_NAME>
344                     </UNIT_MEASUREMENT>
345                   </PROPERTIES>
346                 </NUM>
347                 <NUM>
348                   <CONCEPT_NAME>
349                     <CODE_VALUE>TRMM0001</CODE_VALUE>
350                     <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
351                     <CODE_MEANING>Prolongación Axilar de la Mama Derecha</
                          CODE_MEANING>
352                     <CODE_MEANING2>Axillary Tail of Right Female Breast</
                          CODE_MEANING2>
353                   </CONCEPT_NAME>
354                   <PROPERTIES>
355                     <CARDINALITY max="1" min="1"/>
356                     <CONDITION_TYPE type="M"/>
357                     <EXPRESION_CONDITION xquery=""/>
358                     <DEFAULT_VALUE value="0"/>
359                     <UNIT_MEASUREMENT>
360                       <CONCEPT_NAME>
361                         <CODE_VALUE>000000001</CODE_VALUE>
362                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
363                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
364                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
365                       </CONCEPT_NAME>
366                     </UNIT_MEASUREMENT>
367                   </PROPERTIES>
368                 </NUM>
369                 <NUM>
370                   <CONCEPT_NAME>
371                     <CODE_VALUE>TRMM0003</CODE_VALUE>
372                     <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
373                     <CODE_MEANING>Axila de la Mama Derecha</CODE_MEANING>
374                     <CODE_MEANING2>Axillary Region of Right Female Breast</
                          CODE_MEANING2>
375                   </CONCEPT_NAME>
376                   <PROPERTIES>
377                     <CARDINALITY max="1" min="1"/>
378                     <CONDITION_TYPE type="M"/>
379                     <EXPRESION_CONDITION xquery=""/>
380                     <DEFAULT_VALUE value="0"/>
381                     <UNIT_MEASUREMENT>
382                       <CONCEPT_NAME>
```

```
                        <CODE_VALUE>000000001</CODE_VALUE>
                        <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
                        <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
                        <CODE_MEANING2>Boolean Units</CODE_MEANING2>
                      </CONCEPT_NAME>
                    </UNIT_MEASUREMENT>
                  </PROPERTIES>
                </NUM>
                <NUM>
                  <CONCEPT_NAME>
                    <CODE_VALUE>TRMM0005</CODE_VALUE>
                    <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
                    <CODE_MEANING>Surco Inframamario de la Mama Derecha</
                       CODE_MEANING>
                    <CODE_MEANING2>Inframammary Sulcus of Right Female
                       Breast</CODE_MEANING2>
                  </CONCEPT_NAME>
                  <PROPERTIES>
                    <CARDINALITY max="1" min="1"/>
                    <CONDITION_TYPE type="M"/>
                    <EXPRESION_CONDITION xquery=""/>
                    <DEFAULT_VALUE value="0"/>
                    <UNIT_MEASUREMENT>
                      <CONCEPT_NAME>
                        <CODE_VALUE>000000001</CODE_VALUE>
                        <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
                        <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
                        <CODE_MEANING2>Boolean Units</CODE_MEANING2>
                      </CONCEPT_NAME>
                    </UNIT_MEASUREMENT>
                  </PROPERTIES>
                </NUM>
                <NUM>
                  <CONCEPT_NAME>
                    <CODE_VALUE>TRMM0007</CODE_VALUE>
                    <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
                    <CODE_MEANING>Surco Intermamario de la Mama Derecha</
                       CODE_MEANING>
                    <CODE_MEANING2>Intermammary Sulcus of Right Female
                       Breast </CODE_MEANING2>
                  </CONCEPT_NAME>
                  <PROPERTIES>
                    <CARDINALITY max="1" min="1"/>
                    <CONDITION_TYPE type="M"/>
                    <EXPRESION_CONDITION xquery=""/>
                    <DEFAULT_VALUE value="0"/>
                    <UNIT_MEASUREMENT>
                      <CONCEPT_NAME>
                        <CODE_VALUE>000000001</CODE_VALUE>
                        <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
                        <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
                        <CODE_MEANING2>Boolean Units</CODE_MEANING2>
```

```
                    </CONCEPT_NAME>
                  </UNIT_MEASUREMENT>
                </PROPERTIES>
              </NUM>
            </CHILDS>
          </CONTAINER>
        </CHILDS>
      </CONTAINER>

       <!-- LEFT FEMALE BREAST -->
      <CONTAINER>
        <CONCEPT_NAME>
          <CODE_VALUE>RID29897</CODE_VALUE>
          <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
          <CODE_MEANING>Mama Izquierda</CODE_MEANING>
          <CODE_MEANING2>Left female breast</CODE_MEANING2>
        </CONCEPT_NAME>
        <PROPERTIES>
          <CARDINALITY max="1" min="1"/>
          <CONDITION_TYPE type="M"/>
          <EXPRESION_CONDITION xquery=""/>
        </PROPERTIES>

        <!-- LEFT FEMALE BREAST INJURIES -->
        <CHILDS>
          <CONTAINER>
            <CONCEPT_NAME>
              <CODE_VALUE>TRMM0010</CODE_VALUE>
              <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
              <CODE_MEANING>Asimetria</CODE_MEANING>
              <CODE_MEANING2>Asymmetry</CODE_MEANING2>
            </CONCEPT_NAME>
            <PROPERTIES>
              <CARDINALITY max="-1" min="0"/>
              <CONDITION_TYPE type="U"/>
              <EXPRESION_CONDITION xquery=""/>
            </PROPERTIES>
            <CHILDS>
              <TEXT>
                <CONCEPT_NAME>
                  <CODE_VALUE>118522005</CODE_VALUE>
                  <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
                  <CODE_MEANING>Identificador</CODE_MEANING>
                  <CODE_MEANING2>Identifier</CODE_MEANING2>
                </CONCEPT_NAME>
                <PROPERTIES>
                  <CARDINALITY max="1" min="1"/>
                  <CONDITION_TYPE type="M"/>
                  <EXPRESION_CONDITION xquery=""/>
                  <DEFAULT_VALUE value=""/>
                </PROPERTIES>
              </TEXT>
```

```
483                 <NUM>
484                   <CONCEPT_NAME>
485                     <CODE_VALUE>RID29930</CODE_VALUE>
486                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
487                     <CODE_MEANING>Cuadrante Supero-Externo (CSE) de la Mama
                            Izquierda</CODE_MEANING>
488                     <CODE_MEANING2>Upper Outer Quadrant of Left Female
                           Breast</CODE_MEANING2>
489                   </CONCEPT_NAME>
490                   <PROPERTIES>
491                     <CARDINALITY max="1" min="1"/>
492                     <CONDITION_TYPE type="M"/>
493                     <EXPRESION_CONDITION xquery=""/>
494                     <DEFAULT_VALUE value="0"/>
495                     <UNIT_MEASUREMENT>
496                       <CONCEPT_NAME>
497                         <CODE_VALUE>000000001</CODE_VALUE>
498                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
499                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
500                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
501                       </CONCEPT_NAME>
502                     </UNIT_MEASUREMENT>
503                   </PROPERTIES>
504                 </NUM>
505                 <NUM>
506                   <CONCEPT_NAME>
507                     <CODE_VALUE>RID29936</CODE_VALUE>
508                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
509                     <CODE_MEANING>Cuadrante Infero-Externo (CIE) de la Mama
                            Izquierda</CODE_MEANING>
510                     <CODE_MEANING2>Lower Outer Quadrant of Left Female
                           Breast</CODE_MEANING2>
511                   </CONCEPT_NAME>
512                   <PROPERTIES>
513                     <CARDINALITY max="1" min="1"/>
514                     <CONDITION_TYPE type="M"/>
515                     <EXPRESION_CONDITION xquery=""/>
516                     <DEFAULT_VALUE value="0"/>
517                     <UNIT_MEASUREMENT>
518                       <CONCEPT_NAME>
519                         <CODE_VALUE>000000001</CODE_VALUE>
520                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
521                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
522                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
523                       </CONCEPT_NAME>
524                     </UNIT_MEASUREMENT>
525                   </PROPERTIES>
526                 </NUM>
527                 <NUM>
528                   <CONCEPT_NAME>
529                     <CODE_VALUE>RID29933</CODE_VALUE>
530                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
```

```
531                     <CODE_MEANING>Cuadrante Supero-Interno (CSI) de la Mama
                            Izquierda</CODE_MEANING>
532                     <CODE_MEANING2>Upper Inner Quadrant of Left Female
                           Breast</CODE_MEANING2>
533                   </CONCEPT_NAME>
534                   <PROPERTIES>
535                     <CARDINALITY max="1" min="1"/>
536                     <CONDITION_TYPE type="M"/>
537                     <EXPRESION_CONDITION xquery=""/>
538                     <DEFAULT_VALUE value="0"/>
539                     <UNIT_MEASUREMENT>
540                       <CONCEPT_NAME>
541                         <CODE_VALUE>000000001</CODE_VALUE>
542                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
543                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
544                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
545                       </CONCEPT_NAME>
546                     </UNIT_MEASUREMENT>
547                   </PROPERTIES>
548                 </NUM>
549                 <NUM>
550                   <CONCEPT_NAME>
551                     <CODE_VALUE>RID29939</CODE_VALUE>
552                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
553                     <CODE_MEANING>Cuadrante Infero-Interno (CII) de la Mama
                            Izquierda</CODE_MEANING>
554                     <CODE_MEANING2>Lower Inner Quadrant of Left Female
                           Breast</CODE_MEANING2>
555                   </CONCEPT_NAME>
556                   <PROPERTIES>
557                     <CARDINALITY max="1" min="1"/>
558                     <CONDITION_TYPE type="M"/>
559                     <EXPRESION_CONDITION xquery=""/>
560                     <DEFAULT_VALUE value="0"/>
561                     <UNIT_MEASUREMENT>
562                       <CONCEPT_NAME>
563                         <CODE_VALUE>000000001</CODE_VALUE>
564                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
565                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
566                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
567                       </CONCEPT_NAME>
568                     </UNIT_MEASUREMENT>
569                   </PROPERTIES>
570                 </NUM>
571                 <NUM>
572                   <CONCEPT_NAME>
573                     <CODE_VALUE>RID29948</CODE_VALUE>
574                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
575                     <CODE_MEANING>Línea Intercudrántica Externa (LIE) de la
                            Mama Izquierda</CODE_MEANING>
576                     <CODE_MEANING2>Lateral Region of Left Breast</
                           CODE_MEANING2>
```

```
577                   </CONCEPT_NAME>
578                   <PROPERTIES>
579                     <CARDINALITY max="1" min="1"/>
580                     <CONDITION_TYPE type="M"/>
581                     <EXPRESION_CONDITION xquery=""/>
582                     <DEFAULT_VALUE value="0"/>
583                     <UNIT_MEASUREMENT>
584                       <CONCEPT_NAME>
585                         <CODE_VALUE>000000001</CODE_VALUE>
586                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
587                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
588                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
589                       </CONCEPT_NAME>
590                     </UNIT_MEASUREMENT>
591                   </PROPERTIES>
592                 </NUM>
593                 <NUM>
594                   <CONCEPT_NAME>
595                     <CODE_VALUE>RID29942</CODE_VALUE>
596                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
597                     <CODE_MEANING>Línea Intercuadrántica Superior (LIS) de
                          la Mama Izquierda</CODE_MEANING>
598                     <CODE_MEANING2>Superior Region of Left Breast</
                          CODE_MEANING2>
599                   </CONCEPT_NAME>
600                   <PROPERTIES>
601                     <CARDINALITY max="1" min="1"/>
602                     <CONDITION_TYPE type="M"/>
603                     <EXPRESION_CONDITION xquery=""/>
604                     <DEFAULT_VALUE value="0"/>
605                     <UNIT_MEASUREMENT>
606                       <CONCEPT_NAME>
607                         <CODE_VALUE>000000001</CODE_VALUE>
608                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
609                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
610                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
611                       </CONCEPT_NAME>
612                     </UNIT_MEASUREMENT>
613                   </PROPERTIES>
614                 </NUM>
615                 <NUM>
616                   <CONCEPT_NAME>
617                     <CODE_VALUE>RID29954</CODE_VALUE>
618                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
619                     <CODE_MEANING>Línea Intercudrántica Inferior (LIInf) de
                          la Mama Izquierda</CODE_MEANING>
620                     <CODE_MEANING2>Subareolar Region of Left Breast</
                          CODE_MEANING2>
621                   </CONCEPT_NAME>
622                   <PROPERTIES>
623                     <CARDINALITY max="1" min="1"/>
624                     <CONDITION_TYPE type="M"/>
```

```
625                     <EXPRESION_CONDITION xquery=""/>
626                     <DEFAULT_VALUE value="0"/>
627                     <UNIT_MEASUREMENT>
628                       <CONCEPT_NAME>
629                         <CODE_VALUE>000000001</CODE_VALUE>
630                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
631                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
632                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
633                       </CONCEPT_NAME>
634                     </UNIT_MEASUREMENT>
635                   </PROPERTIES>
636                 </NUM>
637                 <NUM>
638                   <CONCEPT_NAME>
639                     <CODE_VALUE>RID29945</CODE_VALUE>
640                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
641                     <CODE_MEANING>Línea Intercudrántica Interna (LIInt) de
                          la Mama Izquierda</CODE_MEANING>
642                     <CODE_MEANING2>Medial Region of Left Breast</
                          CODE_MEANING2>
643                   </CONCEPT_NAME>
644                   <PROPERTIES>
645                     <CARDINALITY max="1" min="1"/>
646                     <CONDITION_TYPE type="M"/>
647                     <EXPRESION_CONDITION xquery=""/>
648                     <DEFAULT_VALUE value="0"/>
649                     <UNIT_MEASUREMENT>
650                       <CONCEPT_NAME>
651                         <CODE_VALUE>000000001</CODE_VALUE>
652                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
653                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
654                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
655                       </CONCEPT_NAME>
656                     </UNIT_MEASUREMENT>
657                   </PROPERTIES>
658                 </NUM>
659                 <NUM>
660                   <CONCEPT_NAME>
661                     <CODE_VALUE>RID29951</CODE_VALUE>
662                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
663                     <CODE_MEANING>Retroareolar de la Mama Izquierda</
                          CODE_MEANING>
664                     <CODE_MEANING2>Central Region of Left Breast</
                          CODE_MEANING2>
665                   </CONCEPT_NAME>
666                   <PROPERTIES>
667                     <CARDINALITY max="1" min="1"/>
668                     <CONDITION_TYPE type="M"/>
669                     <EXPRESION_CONDITION xquery=""/>
670                     <DEFAULT_VALUE value="0"/>
671                     <UNIT_MEASUREMENT>
672                       <CONCEPT_NAME>
```

```
673                         <CODE_VALUE>000000001</CODE_VALUE>
674                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
675                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
676                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
677                       </CONCEPT_NAME>
678                     </UNIT_MEASUREMENT>
679                   </PROPERTIES>
680                 </NUM>
681                 <NUM>
682                   <CONCEPT_NAME>
683                     <CODE_VALUE>RID29908</CODE_VALUE>
684                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
685                     <CODE_MEANING>Pezón de la Mama Izquierda</CODE_MEANING>
686                     <CODE_MEANING2>Nipple of Left Female Breast</
                          CODE_MEANING2>
687                   </CONCEPT_NAME>
688                   <PROPERTIES>
689                     <CARDINALITY max="1" min="1"/>
690                     <CONDITION_TYPE type="M"/>
691                     <EXPRESION_CONDITION xquery=""/>
692                     <DEFAULT_VALUE value="0"/>
693                     <UNIT_MEASUREMENT>
694                       <CONCEPT_NAME>
695                         <CODE_VALUE>000000001</CODE_VALUE>
696                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
697                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
698                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
699                       </CONCEPT_NAME>
700                     </UNIT_MEASUREMENT>
701                   </PROPERTIES>
702                 </NUM>
703                 <NUM>
704                   <CONCEPT_NAME>
705                     <CODE_VALUE>RID29919</CODE_VALUE>
706                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
707                     <CODE_MEANING>Areola de la Mama Izquierda</CODE_MEANING
                          >
708                     <CODE_MEANING2>Areola of Left Female Breast</
                          CODE_MEANING2>
709                   </CONCEPT_NAME>
710                   <PROPERTIES>
711                     <CARDINALITY max="1" min="1"/>
712                     <CONDITION_TYPE type="M"/>
713                     <EXPRESION_CONDITION xquery=""/>
714                     <DEFAULT_VALUE value="0"/>
715                     <UNIT_MEASUREMENT>
716                       <CONCEPT_NAME>
717                         <CODE_VALUE>000000001</CODE_VALUE>
718                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
719                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
720                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
721                       </CONCEPT_NAME>
```

```
722                     </UNIT_MEASUREMENT>
723                   </PROPERTIES>
724                 </NUM>
725                 <NUM>
726                   <CONCEPT_NAME>
727                     <CODE_VALUE>TRMM0002</CODE_VALUE>
728                     <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
729                     <CODE_MEANING>Prolongación Axilar de la Mama Izquierda<
                          /CODE_MEANING>
730                     <CODE_MEANING2>Axillary Tail of Left Female Breast</
                          CODE_MEANING2>
731                   </CONCEPT_NAME>
732                   <PROPERTIES>
733                     <CARDINALITY max="1" min="1"/>
734                     <CONDITION_TYPE type="M"/>
735                     <EXPRESION_CONDITION xquery=""/>
736                     <DEFAULT_VALUE value="0"/>
737                     <UNIT_MEASUREMENT>
738                       <CONCEPT_NAME>
739                         <CODE_VALUE>000000001</CODE_VALUE>
740                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
741                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
742                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
743                       </CONCEPT_NAME>
744                     </UNIT_MEASUREMENT>
745                   </PROPERTIES>
746                 </NUM>
747                 <NUM>
748                   <CONCEPT_NAME>
749                     <CODE_VALUE>TRMM0004</CODE_VALUE>
750                     <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
751                     <CODE_MEANING>Axila de la Mama Izquierda</CODE_MEANING>
752                     <CODE_MEANING2>Axillary Region of Left Female Breast</
                          CODE_MEANING2>
753                   </CONCEPT_NAME>
754                   <PROPERTIES>
755                     <CARDINALITY max="1" min="1"/>
756                     <CONDITION_TYPE type="M"/>
757                     <EXPRESION_CONDITION xquery=""/>
758                     <DEFAULT_VALUE value="0"/>
759                     <UNIT_MEASUREMENT>
760                       <CONCEPT_NAME>
761                         <CODE_VALUE>000000001</CODE_VALUE>
762                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
763                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
764                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
765                       </CONCEPT_NAME>
766                     </UNIT_MEASUREMENT>
767                   </PROPERTIES>
768                 </NUM>
769                 <NUM>
770                   <CONCEPT_NAME>
```

```
771                     <CODE_VALUE>TRMM0006</CODE_VALUE>
772                     <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
773                     <CODE_MEANING>Surco Inframamario de la Mama Izquierda</
                            CODE_MEANING>
774                     <CODE_MEANING2>Inframammary Sulcus  of Left Female
                            Breast</CODE_MEANING2>
775                   </CONCEPT_NAME>
776                   <PROPERTIES>
777                     <CARDINALITY max="1" min="1"/>
778                     <CONDITION_TYPE type="M"/>
779                     <EXPRESION_CONDITION xquery=""/>
780                     <DEFAULT_VALUE value="0"/>
781                     <UNIT_MEASUREMENT>
782                       <CONCEPT_NAME>
783                         <CODE_VALUE>000000001</CODE_VALUE>
784                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
785                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
786                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
787                       </CONCEPT_NAME>
788                     </UNIT_MEASUREMENT>
789                   </PROPERTIES>
790                 </NUM>
791                 <NUM>
792                   <CONCEPT_NAME>
793                     <CODE_VALUE>TRMM0008</CODE_VALUE>
794                     <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
795                     <CODE_MEANING>Surco Intermamario de la Mama Izquierda</
                            CODE_MEANING>
796                     <CODE_MEANING2>Intermammary Sulcus  of Left Female
                            Breast</CODE_MEANING2>
797                   </CONCEPT_NAME>
798                   <PROPERTIES>
799                     <CARDINALITY max="1" min="1"/>
800                     <CONDITION_TYPE type="M"/>
801                     <EXPRESION_CONDITION xquery=""/>
802                     <DEFAULT_VALUE value="0"/>
803                     <UNIT_MEASUREMENT>
804                       <CONCEPT_NAME>
805                         <CODE_VALUE>000000001</CODE_VALUE>
806                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
807                         <CODE_MEANING>Unidades Boleanas</CODE_MEANING>
808                         <CODE_MEANING2>Boolean Units</CODE_MEANING2>
809                       </CONCEPT_NAME>
810                     </UNIT_MEASUREMENT>
811                   </PROPERTIES>
812                 </NUM>
813               </CHILDS>
814             </CONTAINER>
815           </CHILDS>
816         </CONTAINER>
817       </CHILDS>
818     </CONTAINER>
```

```
819 </DICOM_SR>
```

Fichero A.1: Informe estructurado de una exploración de mama

B | Plantilla de un informe DICOM-SR

```
1
2  <?xml version="1.0" encoding="ISO-8859-1"?>
3
4  <!-- DICOM HEADER -->
5
6  <DICOM_SR Description="Exploration of breast" IDOntology="4">
7    <CONTAINER>
8      <CONCEPT_NAME>
9        <CODE_VALUE>172117008</CODE_VALUE>
10       <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
11       <CODE_MEANING>Exploration of Breast</CODE_MEANING>
12     </CONCEPT_NAME>
13     <PROPERTIES>
14       <CARDINALITY max="1" min="1"/>
15       <CONDITION_TYPE type="M"/>
16       <EXPRESION_CONDITION xquery=""/>
17     </PROPERTIES>
18     <CHILDS>
19       <DATE>
20         <CONCEPT_NAME>
21           <CODE_VALUE>399651003</CODE_VALUE>
22           <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
23           <CODE_MEANING>Date of Report</CODE_MEANING>
24         </CONCEPT_NAME>
25         <PROPERTIES>
26           <CARDINALITY max="1" min="1"/>
27           <CONDITION_TYPE type="M"/>
28           <EXPRESION_CONDITION xquery=""/>
29           <DEFAULT_VALUE value=""/>
30         </PROPERTIES>
31       </DATE>
32       <NUM>
33         <CONCEPT_NAME>
34           <CODE_VALUE>118522005</CODE_VALUE>
35           <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
36           <CODE_MEANING>Identifier</CODE_MEANING>
37         </CONCEPT_NAME>
38         <PROPERTIES>
39           <CARDINALITY max="1" min="1"/>
40           <CONDITION_TYPE type="M"/>
41           <EXPRESION_CONDITION xquery=""/>
```

```
42            <DEFAULT_VALUE value=""/>
43          </PROPERTIES>
44        </NUM>
45        <NUM>
46          <CONCEPT_NAME>
47            <CODE_VALUE>RID13159</CODE_VALUE>
48            <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
49            <CODE_MEANING>Patient Identifier</CODE_MEANING>
50          </CONCEPT_NAME>
51          <PROPERTIES>
52            <CARDINALITY max="1" min="1"/>
53            <CONDITION_TYPE type="M"/>
54            <EXPRESION_CONDITION xquery=""/>
55            <DEFAULT_VALUE value=""/>
56          </PROPERTIES>
57        </NUM>
58
59        <!-- RIGHT FEMALE BREAST -->
60        <CONTAINER>
61          <CONCEPT_NAME>
62           <CODE_VALUE>RID29896</CODE_VALUE>
63           <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
64           <CODE_MEANING>Right Female Breast</CODE_MEANING>
65          </CONCEPT_NAME>
66          <PROPERTIES>
67            <CARDINALITY max="1" min="1"/>
68            <CONDITION_TYPE type="M"/>
69            <EXPRESION_CONDITION xquery=""/>
70            <DEFAULT_VALUE value=""/>
71          </PROPERTIES>
72
73          <!-- RIGHT FEMALE BREAST INJURIES -->
74          <CHILDS>
75            <CONTAINER>
76              <CONCEPT_NAME>
77                <CODE_VALUE>RID3874</CODE_VALUE>
78                <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
79                <CODE_MEANING>Mass</CODE_MEANING>
80              </CONCEPT_NAME>
81              <PROPERTIES>
82                <CARDINALITY max="-1" min="0"/>
83                <CONDITION_TYPE type="U"/>
84                <EXPRESION_CONDITION xquery=""/>
85              </PROPERTIES>
86              <CHILDS>
87                <TEXT>
88                  <CONCEPT_NAME>
89                    <CODE_VALUE>118522005</CODE_VALUE>
90                    <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
91                    <CODE_MEANING>Identifier</CODE_MEANING>
92                  </CONCEPT_NAME>
93                  <PROPERTIES>
```

```
94                    <CARDINALITY max="1" min="1"/>
95                    <CONDITION_TYPE type="M"/>
96                    <EXPRESION_CONDITION xquery=""/>
97                    <DEFAULT_VALUE value=""/>
98                  </PROPERTIES>
99                </TEXT>
100               <NUM>
101                 <CONCEPT_NAME>
102                   <CODE_VALUE>RID29929</CODE_VALUE>
103                   <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
104                   <CODE_MEANING>Upper Outer Quadrant of Right Female
                          Breast</CODE_MEANING>
105                 </CONCEPT_NAME>
106                 <PROPERTIES>
107                   <CARDINALITY max="1" min="1"/>
108                   <CONDITION_TYPE type="M"/>
109                   <EXPRESION_CONDITION xquery=""/>
110                   <DEFAULT_VALUE value="0"/>
111                   <UNIT_MEASUREMENT>
112                     <CONCEPT_NAME>
113                       <CODE_VALUE>000000001</CODE_VALUE>
114                       <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
115                       <CODE_MEANING>Boolean Units</CODE_MEANING>
116                     </CONCEPT_NAME>
117                   </UNIT_MEASUREMENT>
118                 </PROPERTIES>
119               </NUM>
120               <NUM>
121                 <CONCEPT_NAME>
122                   <CODE_VALUE>RID29935</CODE_VALUE>
123                   <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
124                   <CODE_MEANING>Lower Outer Quadrant of Right Female
                          Breast</CODE_MEANING>
125                 </CONCEPT_NAME>
126                 <VALUE> TRUE </VALUE>
127               </NUM>
128               <NUM>
129                 <CONCEPT_NAME>
130                   <CODE_VALUE>RID29934</CODE_VALUE>
131                   <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
132                   <CODE_MEANING>Lower Outer Quadrant of Breast</
                          CODE_MEANING>
133                 </CONCEPT_NAME>
134                 <PROPERTIES>
135                   <CARDINALITY max="1" min="1"/>
136                   <CONDITION_TYPE type="M"/>
137                   <EXPRESION_CONDITION xquery=""/>
138                   <DEFAULT_VALUE value="0"/>
139                   <UNIT_MEASUREMENT>
140                     <CONCEPT_NAME>
141                       <CODE_VALUE>000000001</CODE_VALUE>
142                       <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
```

```
143                         <CODE_MEANING>Boolean Units</CODE_MEANING>
144                       </CONCEPT_NAME>
145                     </UNIT_MEASUREMENT>
146                   </PROPERTIES>
147                 </NUM>
148                 <NUM>
149                   <CONCEPT_NAME>
150                     <CODE_VALUE>RID29932</CODE_VALUE>
151                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
152                     <CODE_MEANING>Upper Inner Quadrant of Right Female
                          Breast</CODE_MEANING>
153                   </CONCEPT_NAME>
154                   <PROPERTIES>
155                     <CARDINALITY max="1" min="1"/>
156                     <CONDITION_TYPE type="M"/>
157                     <EXPRESION_CONDITION xquery=""/>
158                     <DEFAULT_VALUE value="0"/>
159                     <UNIT_MEASUREMENT>
160                       <CONCEPT_NAME>
161                         <CODE_VALUE>000000001</CODE_VALUE>
162                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
163                         <CODE_MEANING>Boolean Units</CODE_MEANING>
164                       </CONCEPT_NAME>
165                     </UNIT_MEASUREMENT>
166                   </PROPERTIES>
167                 </NUM>
168                 <NUM>
169                   <CONCEPT_NAME>
170                     <CODE_VALUE>RID29938</CODE_VALUE>
171                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
172                     <CODE_MEANING>Lower Inner Quadrant of Right Female
                          Breast</CODE_MEANING>
173                   </CONCEPT_NAME>
174                   <PROPERTIES>
175                     <CARDINALITY max="1" min="1"/>
176                     <CONDITION_TYPE type="M"/>
177                     <EXPRESION_CONDITION xquery=""/>
178                     <DEFAULT_VALUE value="0"/>
179                     <UNIT_MEASUREMENT>
180                       <CONCEPT_NAME>
181                         <CODE_VALUE>000000001</CODE_VALUE>
182                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
183                         <CODE_MEANING>Boolean Units</CODE_MEANING>
184                       </CONCEPT_NAME>
185                     </UNIT_MEASUREMENT>
186                   </PROPERTIES>
187                 </NUM>
188                 <NUM>
189                   <CONCEPT_NAME>
190                     <CODE_VALUE>RID29947</CODE_VALUE>
191                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
192                     <CODE_MEANING>Lateral Region of Right Female Breast</
```

```
                    CODE_MEANING>
                </CONCEPT_NAME>
                <PROPERTIES>
                  <CARDINALITY max="1" min="1"/>
                  <CONDITION_TYPE type="M"/>
                  <EXPRESION_CONDITION xquery=""/>
                  <DEFAULT_VALUE value="0"/>
                  <UNIT_MEASUREMENT>
                    <CONCEPT_NAME>
                      <CODE_VALUE>000000001</CODE_VALUE>
                      <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
                      <CODE_MEANING>Boolean Units</CODE_MEANING>
                    </CONCEPT_NAME>
                  </UNIT_MEASUREMENT>
                </PROPERTIES>
              </NUM>
              <NUM>
                <CONCEPT_NAME>
                  <CODE_VALUE>RID29941</CODE_VALUE>
                  <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
                  <CODE_MEANING>Superior Region of Right Female Breast</
                     CODE_MEANING>
                </CONCEPT_NAME>
                <VALUE> TRUE </VALUE>
              </NUM>
              <NUM>
                <CONCEPT_NAME>
                  <CODE_VALUE>RID29953</CODE_VALUE>
                  <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
                  <CODE_MEANING>Subareolar Region of  Right Female Breast
                     </CODE_MEANING>
                </CONCEPT_NAME>
                <PROPERTIES>
                  <CARDINALITY max="1" min="1"/>
                  <CONDITION_TYPE type="M"/>
                  <EXPRESION_CONDITION xquery=""/>
                  <DEFAULT_VALUE value="0"/>
                  <UNIT_MEASUREMENT>
                    <CONCEPT_NAME>
                      <CODE_VALUE>000000001</CODE_VALUE>
                      <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
                      <CODE_MEANING>Boolean Units</CODE_MEANING>
                    </CONCEPT_NAME>
                  </UNIT_MEASUREMENT>
                </PROPERTIES>
              </NUM>
              <NUM>
                <CONCEPT_NAME>
                  <CODE_VALUE>RID29944</CODE_VALUE>
                  <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
                  <CODE_MEANING>Medial Region of Right Breast</
                     CODE_MEANING>
```

```
241                   </CONCEPT_NAME>
242                   <PROPERTIES>
243                     <CARDINALITY max="1" min="1"/>
244                     <CONDITION_TYPE type="M"/>
245                     <EXPRESION_CONDITION xquery=""/>
246                     <DEFAULT_VALUE value="0"/>
247                     <UNIT_MEASUREMENT>
248                       <CONCEPT_NAME>
249                         <CODE_VALUE>000000001</CODE_VALUE>
250                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
251                         <CODE_MEANING>Boolean Units</CODE_MEANING>
252                       </CONCEPT_NAME>
253                     </UNIT_MEASUREMENT>
254                   </PROPERTIES>
255                 </NUM>
256                 <NUM>
257                   <CONCEPT_NAME>
258                     <CODE_VALUE>RID29950</CODE_VALUE>
259                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
260                     <CODE_MEANING>Central Region of Right Breast</
                          CODE_MEANING>
261                   </CONCEPT_NAME>
262                   <PROPERTIES>
263                     <CARDINALITY max="1" min="1"/>
264                     <CONDITION_TYPE type="M"/>
265                     <EXPRESION_CONDITION xquery=""/>
266                     <DEFAULT_VALUE value="0"/>
267                     <UNIT_MEASUREMENT>
268                       <CONCEPT_NAME>
269                         <CODE_VALUE>000000001</CODE_VALUE>
270                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
271                         <CODE_MEANING>Boolean Units</CODE_MEANING>
272                       </CONCEPT_NAME>
273                     </UNIT_MEASUREMENT>
274                   </PROPERTIES>
275                 </NUM>
276                 <NUM>
277                   <CONCEPT_NAME>
278                     <CODE_VALUE>RID29907</CODE_VALUE>
279                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
280                     <CODE_MEANING>Nipple of Right Female Breast</
                          CODE_MEANING>
281                   </CONCEPT_NAME>
282                   <PROPERTIES>
283                     <CARDINALITY max="1" min="1"/>
284                     <CONDITION_TYPE type="M"/>
285                     <EXPRESION_CONDITION xquery=""/>
286                     <DEFAULT_VALUE value="0"/>
287                     <UNIT_MEASUREMENT>
288                       <CONCEPT_NAME>
289                         <CODE_VALUE>000000001</CODE_VALUE>
290                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
```

```
                        <CODE_MEANING>Boolean Units</CODE_MEANING>
                      </CONCEPT_NAME>
                    </UNIT_MEASUREMENT>
                  </PROPERTIES>
                </NUM>
                <NUM>
                  <CONCEPT_NAME>
                    <CODE_VALUE>RID29918</CODE_VALUE>
                    <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
                    <CODE_MEANING>Areola of Right Female Breast</
                       CODE_MEANING>
                  </CONCEPT_NAME>
                  <PROPERTIES>
                    <CARDINALITY max="1" min="1"/>
                    <CONDITION_TYPE type="M"/>
                    <EXPRESION_CONDITION xquery=""/>
                    <DEFAULT_VALUE value="0"/>
                    <UNIT_MEASUREMENT>
                      <CONCEPT_NAME>
                        <CODE_VALUE>000000001</CODE_VALUE>
                        <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
                        <CODE_MEANING>Boolean Units</CODE_MEANING>
                      </CONCEPT_NAME>
                    </UNIT_MEASUREMENT>
                  </PROPERTIES>
                </NUM>
                <NUM>
                  <CONCEPT_NAME>
                    <CODE_VALUE>TRMM0001</CODE_VALUE>
                    <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
                    <CODE_MEANING>Axillary Tail of Right Female Breast</
                       CODE_MEANING>
                  </CONCEPT_NAME>
                  <PROPERTIES>
                    <CARDINALITY max="1" min="1"/>
                    <CONDITION_TYPE type="M"/>
                    <EXPRESION_CONDITION xquery=""/>
                    <DEFAULT_VALUE value="0"/>
                    <UNIT_MEASUREMENT>
                      <CONCEPT_NAME>
                        <CODE_VALUE>000000001</CODE_VALUE>
                        <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
                        <CODE_MEANING>Boolean Units</CODE_MEANING>
                      </CONCEPT_NAME>
                    </UNIT_MEASUREMENT>
                  </PROPERTIES>
                </NUM>
                <NUM>
                  <CONCEPT_NAME>
                    <CODE_VALUE>TRMM0003</CODE_VALUE>
                    <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
                    <CODE_MEANING>Axillary Region of Right Female Breast</
```

```
                    CODE_MEANING>
341               </CONCEPT_NAME>
342               <PROPERTIES>
343                 <CARDINALITY max="1" min="1"/>
344                 <CONDITION_TYPE type="M"/>
345                 <EXPRESION_CONDITION xquery=""/>
346                 <DEFAULT_VALUE value="0"/>
347                 <UNIT_MEASUREMENT>
348                   <CONCEPT_NAME>
349                     <CODE_VALUE>000000001</CODE_VALUE>
350                     <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
351                     <CODE_MEANING>Boolean Units</CODE_MEANING>
352                   </CONCEPT_NAME>
353                 </UNIT_MEASUREMENT>
354               </PROPERTIES>
355             </NUM>
356             <NUM>
357               <CONCEPT_NAME>
358                 <CODE_VALUE>TRMM0005</CODE_VALUE>
359                 <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
360                 <CODE_MEANING>Inframammary Sulcus of Right Female
                      Breast</CODE_MEANING>
361               </CONCEPT_NAME>
362             </NUM>
363             <NUM>
364               <CONCEPT_NAME>
365                 <CODE_VALUE>TRMM0007</CODE_VALUE>
366                 <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
367                 <CODE_MEANING>Intermammary Sulcus of Right Female
                      Breast</CODE_MEANING>
368               </CONCEPT_NAME>
369             </NUM>
370           </CHILDS>
371       </CONTAINER>
372
373       <!-- LEFT FEMALE BREAST -->
374       <CONTAINER>
375         <CONCEPT_NAME>
376          <CODE_VALUE>RID29897</CODE_VALUE>
377          <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
378          <CODE_MEANING>Left Female Breast</CODE_MEANING>
379         </CONCEPT_NAME>
380
381         <!-- LEFT FEMALE BREAST INJURIES -->
382         <CHILDS>
383           <CONTAINER>
384             <CONCEPT_NAME>
385               <CODE_VALUE>RID34265</CODE_VALUE>
386               <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
387               <CODE_MEANING>Asymmetry</CODE_MEANING>
388             </CONCEPT_NAME>
389             <CHILDS>
```

```
390                 <TEXT>
391                   <CONCEPT_NAME>
392                     <CODE_VALUE>118522005</CODE_VALUE>
393                     <CODE_SCHEMA>SNOMED-CT</CODE_SCHEMA>
394                     <CODE_MEANING>Identifier</CODE_MEANING>
395                   </CONCEPT_NAME>
396                   <VALUE> A001 </VALUE>
397                 </TEXT>
398                 <NUM>
399                   <CONCEPT_NAME>
400                     <CODE_VALUE>RID29930</CODE_VALUE>
401                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
402                     <CODE_MEANING>Upper Outer Quadrant of Left Female
                          Breast</CODE_MEANING>
403                   </CONCEPT_NAME>
404                   <PROPERTIES>
405                     <CARDINALITY max="1" min="1"/>
406                     <CONDITION_TYPE type="M"/>
407                     <EXPRESION_CONDITION xquery=""/>
408                     <DEFAULT_VALUE value="0"/>
409                     <UNIT_MEASUREMENT>
410                       <CONCEPT_NAME>
411                         <CODE_VALUE>000000001</CODE_VALUE>
412                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
413                         <CODE_MEANING>Boolean Units</CODE_MEANING>
414                       </CONCEPT_NAME>
415                     </UNIT_MEASUREMENT>
416                   </PROPERTIES>
417                 </NUM>
418                 <NUM>
419                   <CONCEPT_NAME>
420                     <CODE_VALUE>RID29936</CODE_VALUE>
421                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
422                     <CODE_MEANING>Lower Outer Quadrant of Left Female
                          Breast</CODE_MEANING>
423                   <PROPERTIES>
424                     <CARDINALITY max="1" min="1"/>
425                     <CONDITION_TYPE type="M"/>
426                     <EXPRESION_CONDITION xquery=""/>
427                     <DEFAULT_VALUE value="0"/>
428                     <UNIT_MEASUREMENT>
429                       <CONCEPT_NAME>
430                         <CODE_VALUE>000000001</CODE_VALUE>
431                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
432                         <CODE_MEANING>Boolean Units</CODE_MEANING>
433                       </CONCEPT_NAME>
434                     </UNIT_MEASUREMENT>
435                   </PROPERTIES>
436                 </NUM>
437                 <NUM>
438                   <CONCEPT_NAME>
439                     <CODE_VALUE>RID29933</CODE_VALUE>
```

```
440                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
441                     <CODE_MEANING>Upper Inner Quadrant of Left Female
                            Breast</CODE_MEANING>
442                   </CONCEPT_NAME>
443                   <PROPERTIES>
444                     <CARDINALITY max="1" min="1"/>
445                     <CONDITION_TYPE type="M"/>
446                     <EXPRESION_CONDITION xquery=""/>
447                     <DEFAULT_VALUE value="0"/>
448                     <UNIT_MEASUREMENT>
449                       <CONCEPT_NAME>
450                         <CODE_VALUE>000000001</CODE_VALUE>
451                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
452                         <CODE_MEANING>Boolean Units</CODE_MEANING>
453                       </CONCEPT_NAME>
454                     </UNIT_MEASUREMENT>
455                   </PROPERTIES>
456                 </NUM>
457                 <NUM>
458                   <CONCEPT_NAME>
459                     <CODE_VALUE>RID29939</CODE_VALUE>
460                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
461                     <CODE_MEANING>Lower Inner Quadrant of Left Female
                            Breast</CODE_MEANING>
462                   </CONCEPT_NAME>
463                   <PROPERTIES>
464                     <CARDINALITY max="1" min="1"/>
465                     <CONDITION_TYPE type="M"/>
466                     <EXPRESION_CONDITION xquery=""/>
467                     <DEFAULT_VALUE value="0"/>
468                     <UNIT_MEASUREMENT>
469                       <CONCEPT_NAME>
470                         <CODE_VALUE>000000001</CODE_VALUE>
471                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
472                         <CODE_MEANING>Boolean Units</CODE_MEANING>
473                       </CONCEPT_NAME>
474                     </UNIT_MEASUREMENT>
475                   </PROPERTIES>
476                 </NUM>
477                 <NUM>
478                   <CONCEPT_NAME>
479                     <CODE_VALUE>RID29948</CODE_VALUE>
480                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
481                     <CODE_MEANING>Lateral Region of Left Breast</
                            CODE_MEANING>
482                   </CONCEPT_NAME>
483                   <VALUE> TRUE </VALUE>
484                 </NUM>
485                 <NUM>
486                   <CONCEPT_NAME>
487                     <CODE_VALUE>RID29942</CODE_VALUE>
488                     <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
```

```
489                    <CODE_MEANING>Superior Region of Left Breast</
                           CODE_MEANING>
490                  </CONCEPT_NAME>
491                  <VALUE> TRUE </VALUE>
492                </NUM>
493                <NUM>
494                  <CONCEPT_NAME>
495                    <CODE_VALUE>RID29954</CODE_VALUE>
496                    <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
497                    <CODE_MEANING>Subareolar Region of Left Breast</
                           CODE_MEANING>
498                  </CONCEPT_NAME>
499                  <PROPERTIES>
500                    <CARDINALITY max="1" min="1"/>
501                    <CONDITION_TYPE type="M"/>
502                    <EXPRESION_CONDITION xquery=""/>
503                    <DEFAULT_VALUE value="0"/>
504                    <UNIT_MEASUREMENT>
505                      <CONCEPT_NAME>
506                        <CODE_VALUE>000000001</CODE_VALUE>
507                        <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
508                        <CODE_MEANING>Boolean Units</CODE_MEANING>
509                      </CONCEPT_NAME>
510                    </UNIT_MEASUREMENT>
511                  </PROPERTIES>
512                </NUM>
513                <NUM>
514                  <CONCEPT_NAME>
515                    <CODE_VALUE>RID29945</CODE_VALUE>
516                    <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
517                    <CODE_MEANING>Medial Region of Left Breast</
                           CODE_MEANING>
518                  </CONCEPT_NAME>
519                  <PROPERTIES>
520                    <CARDINALITY max="1" min="1"/>
521                    <CONDITION_TYPE type="M"/>
522                    <EXPRESION_CONDITION xquery=""/>
523                    <DEFAULT_VALUE value="0"/>
524                    <UNIT_MEASUREMENT>
525                      <CONCEPT_NAME>
526                        <CODE_VALUE>000000001</CODE_VALUE>
527                        <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
528                        <CODE_MEANING>Boolean Units</CODE_MEANING>
529                      </CONCEPT_NAME>
530                    </UNIT_MEASUREMENT>
531                  </PROPERTIES>
532                </NUM>
533                <NUM>
534                  <CONCEPT_NAME>
535                    <CODE_VALUE>RID29951</CODE_VALUE>
536                    <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
537                    <CODE_MEANING>Central Region of Left Breast</
```

```
                    CODE_MEANING>
538               </CONCEPT_NAME>
539               <PROPERTIES>
540                 <CARDINALITY max="1" min="1"/>
541                 <CONDITION_TYPE type="M"/>
542                 <EXPRESION_CONDITION xquery=""/>
543                 <DEFAULT_VALUE value="0"/>
544                 <UNIT_MEASUREMENT>
545                   <CONCEPT_NAME>
546                     <CODE_VALUE>000000001</CODE_VALUE>
547                     <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
548                     <CODE_MEANING>Boolean Units</CODE_MEANING>
549                   </CONCEPT_NAME>
550                 </UNIT_MEASUREMENT>
551               </PROPERTIES>
552             </NUM>
553             <NUM>
554               <CONCEPT_NAME>
555                 <CODE_VALUE>RID29908</CODE_VALUE>
556                 <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
557                 <CODE_MEANING>Nipple of Left Female Breast</
                      CODE_MEANING>
558               </CONCEPT_NAME>
559               <PROPERTIES>
560                 <CARDINALITY max="1" min="1"/>
561                 <CONDITION_TYPE type="M"/>
562                 <EXPRESION_CONDITION xquery=""/>
563                 <DEFAULT_VALUE value="0"/>
564                 <UNIT_MEASUREMENT>
565                   <CONCEPT_NAME>
566                     <CODE_VALUE>000000001</CODE_VALUE>
567                     <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
568                     <CODE_MEANING>Boolean Units</CODE_MEANING>
569                   </CONCEPT_NAME>
570                 </UNIT_MEASUREMENT>
571               </PROPERTIES>
572             </NUM>
573             <NUM>
574               <CONCEPT_NAME>
575                 <CODE_VALUE>RID29919</CODE_VALUE>
576                 <CODE_SCHEMA>RADLEX</CODE_SCHEMA>
577                 <CODE_MEANING>Areola of Left Female Breast</
                      CODE_MEANING>
578               </CONCEPT_NAME>
579               <PROPERTIES>
580                 <CARDINALITY max="1" min="1"/>
581                 <CONDITION_TYPE type="M"/>
582                 <EXPRESION_CONDITION xquery=""/>
583                 <DEFAULT_VALUE value="0"/>
584                 <UNIT_MEASUREMENT>
585                   <CONCEPT_NAME>
586                     <CODE_VALUE>000000001</CODE_VALUE>
```

```
587                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
588                         <CODE_MEANING>Boolean Units</CODE_MEANING>
589                       </CONCEPT_NAME>
590                     </UNIT_MEASUREMENT>
591                   </PROPERTIES>
592                 </NUM>
593                 <NUM>
594                   <CONCEPT_NAME>
595                     <CODE_VALUE>TRMM0002</CODE_VALUE>
596                     <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
597                     <CODE_MEANING>Axillary Tail of Left Female Breast</
                          CODE_MEANING>
598                   </CONCEPT_NAME>
599                   <PROPERTIES>
600                     <CARDINALITY max="1" min="1"/>
601                     <CONDITION_TYPE type="M"/>
602                     <EXPRESION_CONDITION xquery=""/>
603                     <DEFAULT_VALUE value="0"/>
604                     <UNIT_MEASUREMENT>
605                       <CONCEPT_NAME>
606                         <CODE_VALUE>000000001</CODE_VALUE>
607                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
608                         <CODE_MEANING>Boolean Units</CODE_MEANING>
609                       </CONCEPT_NAME>
610                     </UNIT_MEASUREMENT>
611                   </PROPERTIES>
612                 </NUM>
613                 <NUM>
614                 <CONCEPT_NAME>
615                   <CODE_VALUE>TRMM0004</CODE_VALUE>
616                   <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
617                     <CODE_MEANING>Axillary Region of Left Female Breast</
                          CODE_MEANING>
618                   </CONCEPT_NAME>
619                   <PROPERTIES>
620                     <CARDINALITY max="1" min="1"/>
621                     <CONDITION_TYPE type="M"/>
622                     <EXPRESION_CONDITION xquery=""/>
623                     <DEFAULT_VALUE value="0"/>
624                     <UNIT_MEASUREMENT>
625                       <CONCEPT_NAME>
626                         <CODE_VALUE>000000001</CODE_VALUE>
627                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
628                         <CODE_MEANING>Boolean Units</CODE_MEANING>
629                       </CONCEPT_NAME>
630                     </UNIT_MEASUREMENT>
631                   </PROPERTIES>
632                 </NUM>
633                 <NUM>
634                   <CONCEPT_NAME>
635                     <CODE_VALUE>TRMM0006</CODE_VALUE>
636                     <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
```

```
637                     <CODE_MEANING>Inframammary Sulcus  of Left Female
                           Breast</CODE_MEANING>
638                   </CONCEPT_NAME>
639                   <PROPERTIES>
640                     <CARDINALITY max="1" min="1"/>
641                     <CONDITION_TYPE type="M"/>
642                     <EXPRESION_CONDITION xquery=""/>
643                     <DEFAULT_VALUE value="0"/>
644                     <UNIT_MEASUREMENT>
645                       <CONCEPT_NAME>
646                         <CODE_VALUE>000000001</CODE_VALUE>
647                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
648                         <CODE_MEANING>Boolean Units</CODE_MEANING>
649                       </CONCEPT_NAME>
650                     </UNIT_MEASUREMENT>
651                   </PROPERTIES>
652                 </NUM>
653                 <NUM>
654                   <CONCEPT_NAME>
655                     <CODE_VALUE>TRMM0008</CODE_VALUE>
656                     <CODE_SCHEMA>TRENCADIS_MAMO</CODE_SCHEMA>
657                     <CODE_MEANING>Intermammary Sulcus  of Left Female
                           Breast</CODE_MEANING>
658                   </CONCEPT_NAME>
659                   <PROPERTIES>
660                     <CARDINALITY max="1" min="1"/>
661                     <CONDITION_TYPE type="M"/>
662                     <EXPRESION_CONDITION xquery=""/>
663                     <DEFAULT_VALUE value="0"/>
664                     <UNIT_MEASUREMENT>
665                       <CONCEPT_NAME>
666                         <CODE_VALUE>000000001</CODE_VALUE>
667                         <CODE_SCHEMA>UNIT_MEASUREMENT</CODE_SCHEMA>
668                         <CODE_MEANING>Boolean Units</CODE_MEANING>
669                       </CONCEPT_NAME>
670                     </UNIT_MEASUREMENT>
671                   </PROPERTIES>
672                 </NUM>
673               </CHILDS>
674             </CONTAINER>
675           </CHILDS>
676         </CONTAINER>
677       </CHILDS>
678     </CONTAINER>
679 <DICOM_SR>
```

Fichero B.1: Plantilla de un informe estructurado de una exploración de mama