Proyecto Fin de Carrera

Escuela Técnica Superior

De Ingeniería Informática

Departamento de Física Aplicada

Ejecución de sistemas de ayuda a la decisión en servicios web

Autor: **Alfonso Agud Climent**

Director: **Juan Miguel García Gómez**

Co-Director: **Carlos Sáez Silvestre**

Valencia, Septiembre 2010

Agradecimientos

Me gustaría agradecer a mi tutor Juan Miguel García Gómez la dirección, disponibilidad y rápidas contestaciones las cuales han permitido que el proyecto pudiera llevarse a cabo y finalizarse.

Al grupo Ibime por las facilidades prestadas en el desarrollo del proyecto destacando a Carlos Sáez Silvestre por su gran aportación.

Finalmente a mi familia por haber estado ahí.

Índice

Capítulo 1: Introducción

1. Motivación del proyecto

Actualmente el diagnóstico médico es un procedimiento que exige precisión, rapidez y en el que intervienen un gran número de factores. Es por eso que el desarrollo de herramientas que ayuden a perfilar una decisión clínica puede resultar de gran ayuda a los profesionales médicos, sobre todo a aquellos que poseen menos experiencia. Estas aplicaciones son creadas por grupos de trabajo e implantadas en hospitales donde su objetivo es dar el soporte necesario para que los profesionales puedan tomar una decisión lo más definida posible.

Por otra parte, hoy en día, en el ámbito médico se están abriendo nuevos frentes como la telemedicina donde las aplicaciones nacen con el objetivo de orientarse a las necesidades del paciente y es él quien interactúa directamente con ellas a fin de obtener un beneficio saludable propio. A este ámbito ayudan grandes multinacionales de la información con sus iniciativas como Google Health. Las cuales promueven las aplicaciones médicas online y es por tal motivo que se ha creído necesario aprovechar el potencial de grandes y sofisticadas aplicaciones médicas y facilitar su interacción vía web para que el usuario pueda tener una visión general de su estado de salud, y mismo tiempo, disponga de una serie de facilidades que junto a la supervisión del médico puedan aportarle un amplio conocimiento de su salud, potenciando su calidad de vida.

2. Introducción a los CDSS

Los Sistemas de ayuda a la decisión clínicos[1] (CDSS) son aplicaciones interactivas que ayudan a los profesionales de la salud en su tarea de toma de decisiones. A menudo también son conocidos como DDSS o sistemas de soporte de ayuda al diagnóstico. Se diferencian de los DSS principalmente porque resuelven el caso de un paciente en función de las características de una población de pacientes. Su objetivo principal es ayudar en la toma de decisiones resolviendo la probabilidad de cada alternativa o mostrando un conjunto de sugerencias y con esta información el especialista elige la óptima desechando los resultados superfluos o erróneos.

Estos programas se clasifican según el momento de su uso, siendo los pre-diagnósticos utilizados para ayudar a elaborar un diagnóstico, los diagnósticos para ayudar a revisar y mejorar una valoración preliminar del médico y los post-diagnósticos utilizados para establecer relaciones entre pacientes y predecir eventos futuros.

A pesar de los esfuerzos de las instituciones por incorporar estos sistemas en los centros hospitalarios, todavía hay una serie de impedimentos que afectan a estas aplicaciones. Principalmente el mayor inconveniente es conseguir adaptarse al flujo de trabajo. Igualmente el aprendizaje que requieren por parte del personal y el hecho de que en muchas ocasiones no puedan importar información interna del propio centro automáticamente hace que su integración sea un proceso laborioso.

Respecto a su tecnología están los basados en conocimiento que son aquellos que poseen una base de reglas de producción, un motor de inferencia que relaciona las reglas con la información del paciente, y un mecanismo para comunicarse con el usuario.

En el otro extremo se encuentran los que utilizan técnicas de aprendizaje a partir de experiencias pasadas y de reconocimiento de patrones en los datos clínicos. Este tipo de software utiliza algoritmos genéticos o basados en redes neuronales.

3. Introducción a los Servicios Web, Web 2.0 y Web 3.0

Desde que apareció el *World Wide Web*, internet ha experimentado una gran evolución. A este progreso nos referimos cuando citamos la red con un neologismo numérico. En un principio internet estaba formado por páginas estáticas escritas en HTML. Éstas no eran actualizadas muy a menudo y los usuarios no tenían posibilidad de participar en la creación de los contenidos que se mostraban por la red. A este tipo de servicios nos referimos cuando hablamos de Web 1.0.

El diseño de este tipo de sitios estáticos, se limitaba a la edición de etiquetas, y predominaban las imágenes GIF, los formularios vía email… y en su conjunto componían sitios web que a menudo eran páginas personales y otros servicios de contenidos en los que no solía interactuar directamente el usuario sino un administrador que iba creando sus contenidos mediante la edición manual de código HTML.

Progresivamente aparecieron las páginas web dinámicas en las que una simple consulta o inserción en la base de datos le permitían al administrador modificar fácilmente el contenido de sus páginas. Éstas han sido conocidas menos popularmente como Web 1.5.

Pero la verdadera revolución de internet llegó con la creación de espacios dinámicos en los que el usuario era el protagonista de la información, permitiéndole la posibilidad de crear, modificar y borrar contenidos. Éste es el principio en el que se sustenta la Web 2.0[1] tal y como la presentó por primera vez Tim O'Reilly en una conferencia en el año 2004. Esta actualización está relacionada con un fenómeno social en el que el predomina el aporte del usuario, nutriéndose así de la inteligencia colectiva.

El creador de este concepto incluso llega a comparar la Web 1.0 con la 2.0 de forma análoga a como compararíamos las webs personales con los blogs, la publicación con la participación, los sistemas de gestión de contenidos con las wikis o la Enciclopedia Británica con la Wikipedia.

Actualmente existen multitud de aplicaciones con estas características como pueden ser wikis, blogs y demás servicios de red donde compartir todo tipo de información multimedia. Igualmente existen muchos ámbitos profesionales en los que esta evolución puede incorporarse en el día a día, como pueden ser las aplicaciones educativas con una intranet donde el profesorado y el alumnado comparten material didáctico y conocimientos, o

médicas, donde aplicaciones como Google Health permiten organizar información médica con la que interactúa el propio usuario, aunque esta última con la privacidad necesaria.

Siguiendo el desarrollo que está experimentando constantemente la red, podemos observar características que van más allá de la simple interacción del usuario con las aplicaciones web. Nuevos frentes se están abriendo en los que la interoperabilidad de la red con procesadores o agentes inteligentes permiten procesar de forma eficiente multitud de información.

Siguiendo la dinámica del proceso evolutivo de internet surge la Web 3.0, en la cual se extiende el concepto de web semántica, en el que se le dota de significado a los elementos de cualquier web para que sea más fácil y directa su localización, también se apuesta por evolucionar en el campo de las 3D y en el de la inteligencia artificial. Grandes compañías como Google están aportando tecnología que permite directamente trabajar en este aspecto, permitiendo la creación de modelos predictivos que utilicen técnicas de inteligencia artificial vía web.

4. Servicios médicos online

Gracias a esta plataforma web tan desarrollada actualmente existen multitud de aplicaciones destinadas a ofrecer un servicio cada vez más inteligente al usuario, donde pueda obtener detalladamente información específica a través de sofisticados algoritmos. Los servicios pueden ser de cualquier ámbito profesional siempre que traten de cubrir las necesidades informativas de todo tipo de personal.

En el ámbito médico, hay multitud de información que necesita una interacción vía web. Entre ellas destaca la Historia Clínica Electrónica por ser un conjunto de información que cada paciente o al menos cada centro de salud debería poder acceder vía web y actualizar al momento para poder ser contrastada en los sucesivos diagnósticos que se le realicen al paciente. La integración a nivel nacional e incluso internacional de dicha información no es tarea fácil y aunque muchos son los esfuerzos y las iniciativas por aunarla todavía queda mucha información deslocalizada e incluso en formato no digitalizado. Ante este vacío informativo y siguiendo con su misión empresarial de *organizar la información mundial y hacerla universalmente accesible y útil*[2], Google ha tomado la iniciativa al desarrollar un sistema que permita a los

usuarios obtener acceso a su información sanitaria y a mantener así un mayor conocimiento respecto a las decisiones que se tomen acerca de la salud. Para ello ha creado Google Health, un servicio de información personal centralizado enfocado a la sanidad y en el que interactúa directamente el usuario permitiéndole escribir voluntariamente su historial clínico.

Esta plataforma contiene muchas otras funcionalidades como avisos de los servicios a los que se haya vinculado el usuario o ante interacciones de medicamentos, posibilidad de compartir el perfil con el médico, de importar la cuenta a empresas con la finalidad de que vean el historial y aconsejen al usuario o le den un tratamiento, otras pasan el historial clínico del papel al perfil, también dispone de servicio de alertas personalizadas, posibilidad de buscar un medico en EEUU y muchas opciones más.

A día de hoy acaban de lanzar una nueva versión más evolucionada, de uso sencillo y con un aspecto más profesional tal y como vemos en la ilustración 1.1. Con la posibilidad de crear rastreadores que mediante gráficos le permiten al usuario comprobar la evolución en cualquier tema que desee tratar manteniendo un diario de progreso. Además se vinculan nuevas aplicaciones que tratan temas de salud, de estado físico y de recopilación de datos desde dispositivos como básculas electrónicas o iPhones[3].

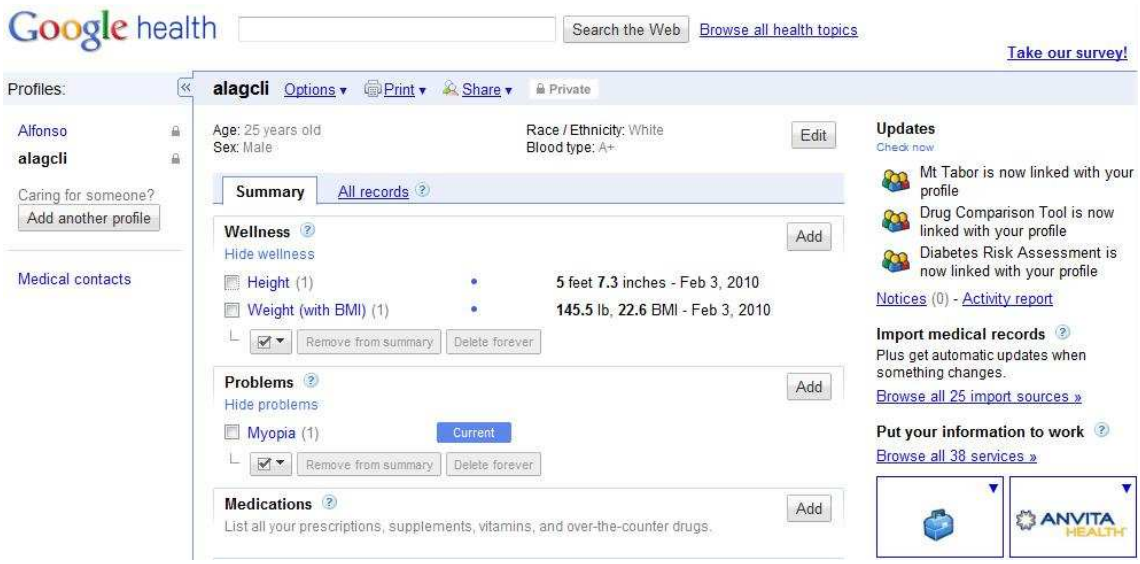

Ilustración 1.1: Versión 2 de Google Health

Pero Google no está solo en la gestión de esta disciplina, otras compañías como Microsoft compiten con aplicaciones como HealthVault[4]. Esta aplicación también permite compartir la historia clínica online, pero a diferencia de Google Health, en HealthVault no se pueden añadir características como el peso o las vacunas directamente, sino que se rellena un formulario con

información básica y el resto de información es importada desde centros médicos que visita el usuario o desde dispositivos móviles.

Ambas aplicaciones interactúan con servicios online de terceros y muchas de sus funcionalidades requieren vincular la cuenta con centros de salud de EEUU.

Otro sistema similar es Dossia[5], una plataforma online que centraliza todos los informes médicos de los empleados de empresas estadounidenses. También existen plataformas de código abierto para historiales y con capacidad de insertar servicios de terceros como IndivoHealth[6].

Por otra parte muchos servicios online que actualmente ofrecen las empresas por internet y que se vinculan a plataformas de historiales clínicos como las anteriormente citadas, son los de evaluación de diversos factores que afectan a la salud como glucosa o colesterol, comparación de precios de medicamentos, oferta de pruebas de laboratorio en línea, posibilidad de encontrar ensayos clínicos de nuevos tratamientos basados en información relativa al usuario…

Actualmente hay muchos otros servicios de telemedicina que aprovechan el potencial de la red. Estos sistemas tratan de centrarse más en los pacientes que en los médicos. Con lo cual, en este campo se está investigando en el desarrollo de tecnologías que permitan adquirir hábitos saludables entre la población, como son programas de ejercicio con un entrenador virtual, carros de compra que informen de la propiedades de los alimentos, dispositivos de monitorización de datos para enfermos crónicos… y una larga lista que va desde los cuestionarios a pacientes hasta de la integración de las TIC en las casas con inteligencia ambiental.

Estos sistemas están destinados a imponerse cada vez más, ya que herramientas como internet permiten un amplio abanico de nuevas posibilidades en este campo. Además esto evita que se colapsen sistemas sanitarios y se previenen diversas enfermedades simplemente prediciendo una posible situación de riesgo y fomentando conductas saludables entre los pacientes. Pero debe de tenerse en cuenta que estas novedades a veces pueden poner en controversia aspectos éticos como son la privacidad de datos, cuando se controla demasiado a un paciente, y pueden ser rechazados sobre todo por personas de tercera edad a quienes no terminen de gustarle los sistemas a priori complicados que se les está ofreciendo. Con lo cual siempre es una buena solución hacer campañas informativas para dar a conocer estas nuevas tecnologías con la finalidad de que vayan imponiéndose entre la población de manera gradual e informando siempre de la calidad de vida que se puede adquirir con ellas[7].

5. Objetivos del proyecto

El presente proyecto tiene como objetivo principal profundizar en la creación de espacios web para la ayuda a la decisión médica utilizando las últimas tecnologías desarrolladas para tal fin. En este caso se ha utilizado como herramienta Google Web Toolkit que combina la robustez del lenguaje orientado a objetos con la posibilidad de crear espacios dinámicos que entrelacen con código diseñado inicialmente para trabajar en aplicaciones de escritorio. De esta forma se le da una mayor diversidad y cobertura al software adaptándolo a nuevas aplicaciones y comprobando su eficiencia a través de nuevos escenarios de utilización como puede ser su uso vía web.

Nos planteamos aprovechar la utilidad de los sistemas de ayuda a la decisión vía web, dándole la oportunidad al usuario experto (o quizá en el futuro también al paciente) a realizar una comprobación de un nuevo caso médico. Aunque en este caso se ha optado por una comprobación de tumores, su diseño permite adaptarle fácilmente otros programas de diagnostico con los que pueda tener una idea general e instantánea del riesgo de diabetes o de padecer un exceso de colesterol entre otras muchas aplicaciones interesantes para el usuario. En este caso se ha optado por desarrollar la herramienta de modo que informe al usuario del riesgo de padecer un tumor de partes blandas a partir de los datos que podamos introducir manualmente.

A fin de promover tal herramienta y ponerla disponible al público se ha investigado la posibilidad de adaptarla en la plataforma online de salud Google Health, desde la cual cualquier usuario podría vincular los datos de su historial y hacerse una evaluación personal a partir de su historial en Google o añadiendo los parámetros necesarios directamente en la aplicación. Desafortunadamente esta compañía todavía no permite vincular aplicaciones de terceros que no estén dentro de EEUU.

6. Plan de trabajo

Para el desarrollo de la aplicación se ha seguido el siguiente proceso de trabajo que, aunque puede no corresponder plenamente con el procedimiento de desarrollo del software, es debido a que gran parte del esfuerzo se centra en el proceso de aprendizaje de las tecnologías empleadas y en el estudio de viabilidad del proyecto.

1. Aprendizaje de la herramienta GWT para el desarrollo de aplicaciones web junto al de librerías compatibles que proporcionen resultados gráficos.

2. Estudio de la API Google Healh, su funcionalidad y viabilidad con el proyecto.

3. Se ha configurado Tomcat con Eclipse para comprobar el funcionamiento de un proyecto de clasificación online.

4. Se ha integrado el módulo de clasificación del anterior proyecto en uno creado en GWT adaptándolo a las peculiaridades de esta herramienta.

5. Se ha procedido a hacer un análisis de los requisitos de la aplicación a través de diagramas de casos de uso, plantillas textuales y diagramas de secuencia.

6. Se ha estudiado XML y la creación de base de datos con esta tecnología junto con su interacción con GWT.

7. Se ha diseñado la interfaz en GWT y demás librerías compartibles para el desarrollo de los requerimientos necesarios.

8. Se ha procedido a documentar el proyecto en esta memoria.

7. Organización de la memoria

A continuación se explica brevemente como se ha estructurado la memoria de este proyecto.

1. En el capítulo 1, se introduce la evolución que están experimentando las aplicaciones web, especialmente las de ámbito médico y se señala la importancia de las CDSS.

2. En el capítulo 2 nos centraremos en las últimas tecnologías que permiten desarrollar aplicaciones médicas de forma online. Entre ellas estudiaremos el aporte de Google tanto en su framework GWT como en su plataforma Google Health junto a su API. También Configuraremos el Entorno Eclipse con el servidor Tomcat.

3. En tercer lugar explicamos el proceso de desarrollo del servicio creado con GWT con las peculiaridades de la herramienta, empezando por el análisis, siguiendo con la BD y todos los componentes y librerías visuales que le hemos tenido que añadir al proyecto para que cumpla con las exigencias marcadas.

4. En el siguiente capítulo estudiaremos la parte del diagnóstico referida a tumores de partes blandas junto a la herramienta Curiam destinada a tal objetivo. También explicaremos detalladamente el proceso de integración del motor de clasificación que aprovecha nuestro proyecto de dicha herramienta.

5. En el capítulo 5 se trata de explicar de forma detallada el proceso de integración de una aplicación médica online en la plataforma Google Health, describiendo igualmente las sugerencias y exigencias que nos proporciona la compañía desde su API destinada tal efecto.

6. En el último capítulo se han expuesto las conclusiones de este trabajo, observaciones tanto positivas como negativas en diferentes aspectos y las vistas de futuro que puede llegar a tener un proyecto de estas características.

7. Finalmente, se ha creado un anexo para explicar la usabilidad de la aplicación y se ha adjuntado la bibliografía correspondiente.

Capítulo 2: Tecnologías Web 2.0/3.0 para servicios médicos online de nueva generación

1. GWT

Actualmente las aplicaciones web de última generación necesitan un alto grado de versatilidad para adaptarse a los requisitos y circunstancias que exige el mercado. Como hemos visto, las aplicaciones Web 2.0 hacían hincapié en la interoperabilidad del usuario y las 3.0 incluso iban más allá perfilando aspectos técnicos como son la capacidad de desarrollar una web en un lenguaje que transmita un mayor significado de los elementos (web semántica), que mejore el aspecto con diseños elaborados en 3D e incluso que tenga capacidad de incorporar algoritmos de inteligencia artificial.

Siguiendo estas tendencias la compañía Google ha desarrollado Google Web Toolkit (GWT), un framework de código abierto que permite escribir aplicaciones online en Java las cuáles traduce a Javascript y HTML evitando además al programador la ardua tarea de ir adaptando un servicio web a las peculiaridades de distintos sistemas[8] como pueden ser la diferente calidad de la resolución, el tipo de navegador e incluso su versión…

Por eso aunque las aplicaciones AJAX son indicadas cuando se requiere una alta interacción, los proyectos Javascript suelen hacerse grandes, su desarrollo acaba siendo lento y su mantenimiento complejo. GWT fue creado con la intención de agilizar esta tarea.

Java posee la ventaja de que los diseños orientados a objetos son más fáciles de entender y de comunicarse entre sí. Además nos proporciona ventajas como el autocompletado del código o la optimización del mismo al ser traducido.

Este plugin se instala en el entorno de desarrollo adecuado, en nuestro caso en el Eclipse y permite crear aplicaciones directamente en Java, que interactúen en un protocolo cliente-servidor. En la parte servidor podemos incorporarle todo el código y librerías Java que necesitemos, pudiendo así llegar a programar aplicaciones en línea dando un amplio margen de posibilidades al programador. Además con esta interacción se libera carga en el servidor que actúa como proveedor de servicios, y esto provoca que su

ejecución sea más fluida. En la parte cliente no importa el navegador que se esté utilizando, ya que GWT produce código capaz de ejecutarse en los principales navegadores, creando una salida optimizada para cada uno de ellos[9].

A la hora de ejecutar una aplicación ésta puede estar en Hosted Mode, en este modo la encontramos cuando está siendo desarrollada, la cual correría como bytecode dentro de la Virtual Machine. Mientras que cuando la aplicación es vista por usuarios finales, se procesa como código Javascript compilado a partir del Java original, este estado del servicio es conocido como Web Mode.

Otra ventaja de este sistema es el conjunto de Widgets que actualmente existen en internet. Éstos son componentes gráficos e incluso pequeñas aplicaciones disponibles a modo de clases que forman parte del lenguaje Java que compila GWT. Con ellas se facilita la programación al usuario no teniendo que recrearlos de cero a partir de Javascript.

Algunas otras características son la gestión del historial web, el soporte para el depurado de Java, la integración de JUnit, que es un framework que evalúa si el funcionamiento de los métodos de una clase es el correcto, la posibilidad de utilizar directamente código Javascript, la API de Google que sirve como soporte, numerosas librerías programadas para GWT que amplían su funcionalidad y muchas ventajas más entre las que destaca el hecho de que, aunque está desarrollado bajo la licencia Apache versión 2.0, es de código abierto y no tiene restricciones sobre las aplicaciones desarrolladas.

2. API Google Health

Ante las nuevas iniciativas que están tomando las grandes empresas por organizar el Historial Clínico Electrónico y habiendo desarrollado una aplicación online de ámbito médico se ha considerado como uno de los objetivos de este proyecto el vincular la aplicación web creada con el apartado de servicios de terceros de la página oficial de Google Health. El intento ha resultado fallido ya que, aunque se encuentre la plataforma de sanidad de Google en fase Beta, de momento solo permiten la vinculación de servicios de terceros que residan e interactúen con datos de pacientes de los EEUU.

Ante esta negativa se ha procedido a documentar como sería el proceso de adaptación en el caso de que se pudiera enlazar o de que se resida en los EEUU.

De entre las posibilidades que ofrece esta API[10] está la de añadir información médica al perfil mediante el envío de mensajes que contienen elementos XHTML y datos CCR (que es un formato definido por la industria médica). Éstos se envían en forma de mensajes y pueden informar al usuario de los resultados de una prueba de laboratorio o recordarle que recoja una preinscripción.

Otra característica es la capacidad de leer el perfil, consultar historiales que coincidan con algún determinado criterio, modificarlo, insertar entradas o borrarlas siempre que dispongamos de los permisos adecuados y nuestra aplicación trate con pacientes de EEUU.

Esta interfaz de biblioteca utiliza el mismo formato de datos que utilizan otras API de Google con lo cual se le facilita la tarea de utilizarla a quien ya domine otras de la misma compañía.

3. Entornos de desarrollo: Eclipse para desarrollo web

Un entorno de desarrollo es una plataforma provista de herramientas que permiten la creación de programas. Pueden ser multiplataforma o especializarse en un tipo de código.

Un IDE por su parte, es un entorno que ha sido empaquetado dentro de un programa de aplicación, el cual se compone normalmente de un editor, un compilador, un depurador y un constructor de interfaz gráfica.

Eclipse es un IDE multiplataforma basado en Java de código abierto[1]. La herramienta fue desarrollada inicialmente por IBM y actualmente mantenida por una organización independiente sin ánimo de lucro conocida como Fundación Eclipse. Dada la versatilidad y la cantidad de trabajo invertido en su desarrollo, ya que actualmente lo componen más de dos millones de líneas de código, en esta plataforma es posible el desarrollo de múltiples lenguajes de programación como Java, C/ C++/Cobol… e incluso se le pueden añadir soportes de lenguajes adicionales especializados en entornos web como PHP, Javascript o GWT, el cual estudiaremos en profundidad. Por lo tanto Eclipse nos permite la posibilidad de desarrollar aplicaciones web de última generación en Java al añadirle el plugin de GWT.

Algunas de las ventajas de Eclipse son que dispone de un editor de texto que resalta la sintaxis corrigiéndola en tiempo real y ofreciendo sugerencias de autocompletado en la elaboración del código.

Para trabajar en Eclipse simplemente lo descomprimimos y al ejecutarlo nos pide la ruta del espacio de trabajo. Creamos el directorio, le marcamos la ruta y accedemos a la pantalla de bienvenida del entorno. En esta pantalla pulsamos la flecha de la izquierda para empezar a utilizar el entorno.

Al entrar vamos a configurar Eclipse para su correcto funcionamiento. Seleccionamos la función ventana / preferencias para abrir las preferencias del escritorio de trabajo. Confirmamos que hay un JRE detectado y marcado. Este soporte es un conjunto de utilidades que permite la ejecución de programas Java. En caso de no haber uno marcado lo agregamos desde el botón añadir o lo buscamos desde el botón de búsqueda, lo agregamos, seleccionamos y aplicamos los cambios.

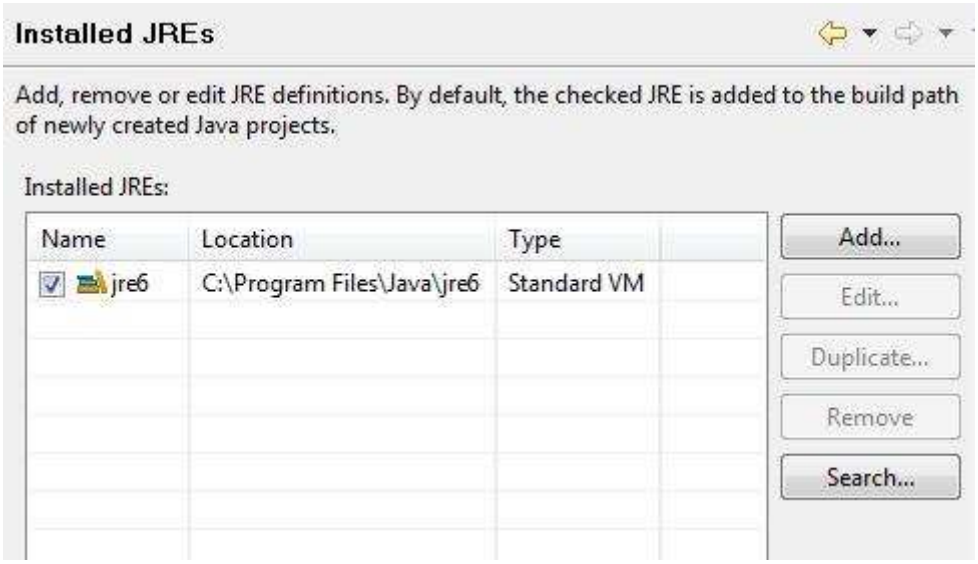

Ilustración 2.1: JREs instalados

Dentro aún del panel de preferencias, seleccionamos general / Workspace y confirmamos que tenemos marcada la opción de construir automáticamente.

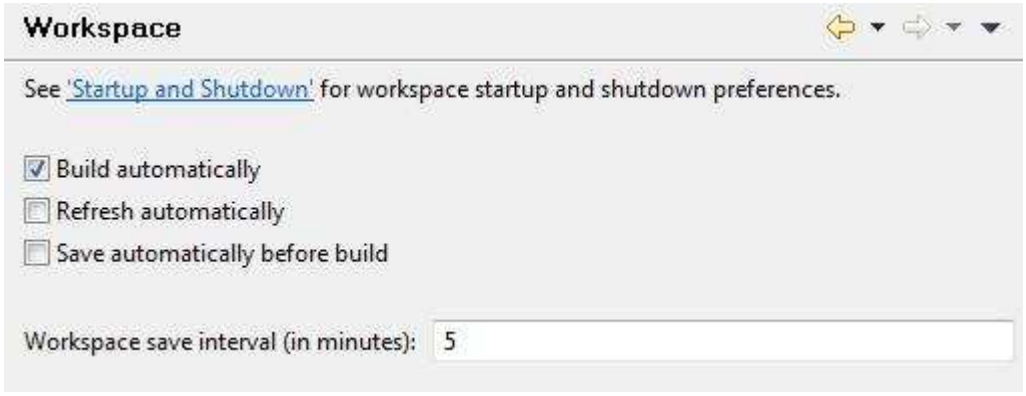

Ilustración 2.2: Generación automática del Workspace

Vamos a Java / Editor en panel de preferencias y comprobamos que el proyecto reporta problemas a medida que escribe.

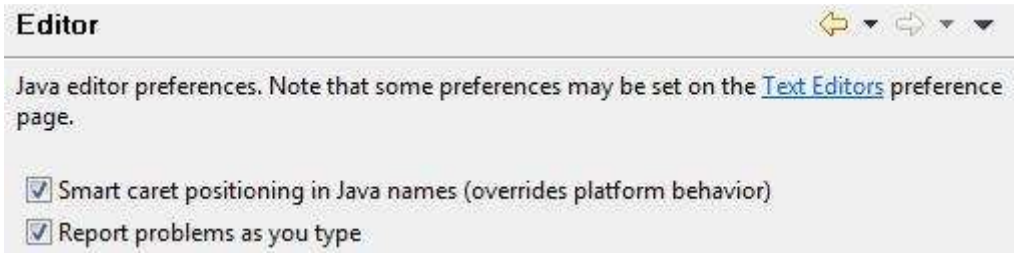

Ilustración 2.3: Generación de errores instantánea

Seleccionamos Java / Compiler y comprobamos que esté seleccionada la versión 1.6 de Java que es la que GWT utiliza para traducir a Javascript. Aplicamos cambios y salimos.

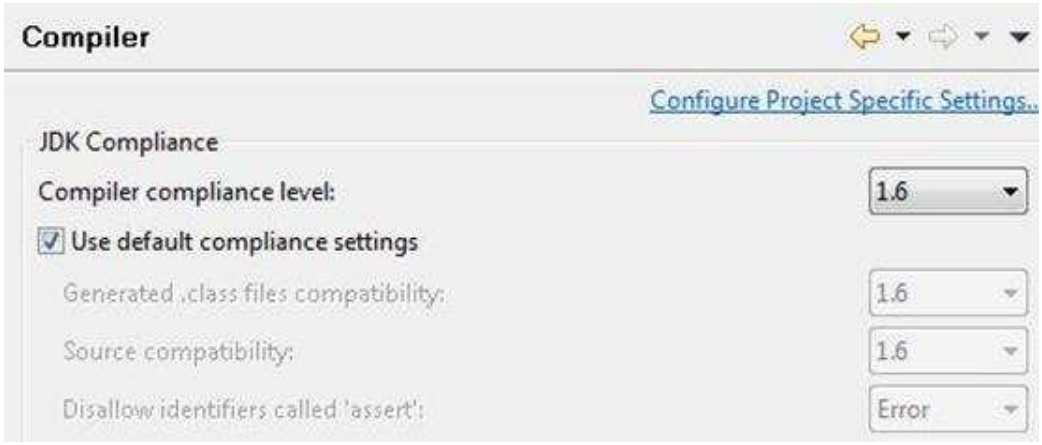

Ilustración 2.4: Selección del compilador

4. Tomcat

Tomcat es un contenedor de servlets, es decir, un módulo que sirve para que éstos se ejecuten junto a archivos JSP en las aplicaciones. Es un servidor que da este tipo de soporte a los servicios web pero no a las aplicaciones y aunque al inicio su rendimiento no era el ideal para aplicaciones con abundante tráfico simultáneo, en la actualidad esta tendencia está desapareciendo en sus últimas versiones. Este software está desarrollado por la fundación Apache y por aficionados que tienen acceso bajo la licencia de código libre de Apache.

En nuestro proyecto hemos tenido que adaptar esta herramienta a Eclipse para poder comprobar el funcionamiento de los módulos de clasificación.

A continuación se procederá a hacer una descripción del proceso de instalación y configuración en nuestro entorno.

Descargamos la última versión disponible de Apache Tomcat desde su web oficial, en nuestro caso hemos bajado la versión 6.

A continuación, teniendo en cuenta que disponemos del Eclipse 3.5 (Galileo) debemos de añadirle a éste la Plataforma web para Eclipse ya que no la trae incorporada por defecto. Con lo cual ejecutamos Eclipse y vamos a Help / Instal new software y en el cuadro de instalación seleccionamos añadir e indicamos la ruta de Web Tools Plataform (WTP):

http://download.eclipse.org/webtools/updates

Aceptamos y la seleccionamos de la lista con lo que nos aparecerán una lista de herramientas disponibles para su instalación. Aunque las esenciales son la Web Tools SDK y Project provided component, es recomendable instalar una versión de cada herramienta junto a los parches de la lista para futuros imprevistos.

Llegados a este punto Eclipse dispone de las funcionalidades necesarias para incorporarle el Tomcat, no obstante para comodidad del usuario hay disponible un plugin llamado tomcatPluginvxx.zip que proporciona una fácil interacción con el servidor mediante funcionalidad que es agregada a la interfaz del entorno. Es recomendable descargarlo y descomprimirlo dentro de la carpeta plugins.

A continuación importamos el proyecto que se nos ha proporcionado de interacción con el módulo de clasificación y le agregamos las librerías BMGClassificationEngine.jar y jdom.jar. Junto a éstas le agregamos también la

servlet-api.jar ubicada en el directorio bin/ de la capeta donde hayamos descomprimido el servidor Apache. Este proceso se hace con botón derecho en el proyecto, seleccionando propieties / Java build path, pestaña libraries, add external JARS.

Seguidamente configuramos el servidor Tomcat yendo a Windows / preferences / Tomcat, marcando la versión instalada, la 6.x en nuestro caso e indicando en Tomcat Home el directorio donde se ha descomprimido el Tomcat. Ahora falta el entorno de ejecución, con lo cual volvemos a preferences y seleccionamos Servers / Runtime Environment pulsamos añadir y seleccionamos la versión de Apache Tomcat que tenemos instalada.

Por último vamos a Run configurations, hacemos doble clic en Apache Tomcat y en la pestaña Classpath le añadimos como entrada de usuario el archivo bootstrap.jar ubicado en el directorio bin de la carpeta donde descomprimimos el Apache.

Llegados a este punto ya tenemos el contenedor de servlets instalado y configurado. Para obtener un mayor control sobre el mismo le añadimos la vista servers desde Windows / Show views / Others.

Ahora desde los botones proporcionados por el plugin podemos inicializar, finalizar y reinicializar el servidor o directamente ejecutarlo desde la configuración de ejecución que le hemos creado con lo que nos saldrá una pequeño navegador incorporado en el Eclipse con la URL ubicada dentro del localhost. Copiándola en cualquier navegador podemos hacer las pruebas necesarias.

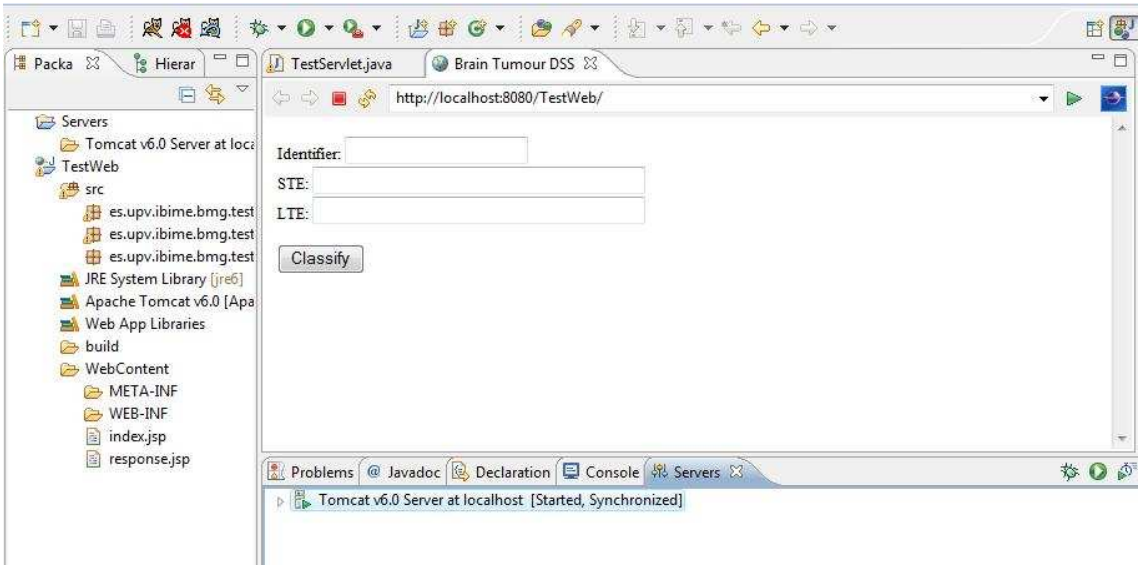

Ilustración 2.5: Proyecto Tomcat configurado.

Capítulo 3: Desarrollo de servicios web con Google Web Toolkit

1. Introducción

Como ya hemos introducido en el anterior capítulo, GWT es un framework diseñado para el desarrollo de aplicaciones web de forma rápida y poniendo al alcance de los servicios online el potencial de Java. Por esta razón hemos elegido esta herramienta para la creación del servicio web.

El plugin está disponible en su página oficial. Como el entorno con el que se compilará el lenguaje Java ha sido en Eclipse vamos al apartado *complemento de Google para Eclipse*. En la guía de usuario de dicha página aparece detalladamente todo el proceso de instalación que aquí resumiremos brevemente.

En primer lugar teniendo descomprimido el Eclipse, se procede a instalarle un nuevo software en las opciones de Ayuda /Instalar nuevo software, donde le añadimos la URL que hace referencia al mismo según la versión de Eclipse que tengamos instalada, en nuestro caso la 3.5 (Galileo), introduciéndole la dirección:

http://dl.google.com/eclipse/plugin/3.5

A continuación marcamos Plugin y SDKs y pulsamos Siguiente, examinamos las características que se van a instalar y volvemos a darle a Siguiente. Por último leemos y aceptamos los términos de acuerdo de licencia, le damos a Finalizar y reiniciamos el programa.

A partir de este momento ya tenemos instalado el plugin y las nuevas extensiones del mismo formarán parte de la interfaz de Eclipse.

Ahora es cuando debemos de crear un nuevo proyecto pulsando el nuevo botón azul con una con la inscripción (g^{+}). Nos pedirá el nombre de proyecto y de paquete y en nuestro caso le insertamos Curiam y com.curiam respectivamente. Al finalizar se nos crea un árbol de directorios. Dentro del directorio raíz tenemos la carpeta src/ la cual contiene los paquetes Java que vamos a ir programando, estos están divididos en un paquete que contiene un

archivo de configuración XML y dos paquetes cliente y servidor que ubicarán las partes de código que se ejecutarán en el procesador del usuario y en el del host donde se albergue la web respectivamente.

Desde el directorio raíz también se indexa a la carpeta war/ la cual contiene las clases compiladas, librerías disponibles en tiempo de ejecución, contenido estático y archivos de configuración.

Al crear un nuevo proyecto por defecto crea una pequeña aplicación que consiste en una sencilla interfaz con un botón que al pulsarlo hace una llamada a un procedimiento remoto que devuelve la hora actual en el servidor. Para poder visualizar esta interfaz basta con ir a la opción de ejecutar / configuraciones de ejecución… y una vez dentro, dar doble clic a aplicación web, ponerle un nombre, elegir nuestro proyecto y la clase principal del mismo. En la pestaña Servidor es recomendable marcar la casilla que habilite un puerto diferente para cada ejecución. Así evitamos que Eclipse interprete como error el hecho de que haya varias aplicaciones corriendo en el mismo puerto al ejecutar varias veces la misma aplicación. En este momento posiblemente en la misma pantalla Eclipse nos muestre a modo de advertencia los argumentos de arranque que necesitamos insertarle en la pestaña parámetros para que el framework GWT funcione correctamente. Hay que incluir el parámetro –javaagent con la ruta del agente. Éste archivo básicamente es un interceptor del proceso de carga de clases que permite monitorizar dicho proceso y modificar el bytecode que se va a ejecutar. También se puede optar por poner el parámetro –Xmx512m que sirve para extender la memoria virtual que usará el programa y así evitar que, transcurrido un tiempo, el Eclipse se bloquee por falta de memoria virtual. Estos comandos nos lo muestra el propio entorno y solo debemos de copiarlos y pegarlos en el apartados parámetros VM de la pestaña parámetros, después de aplicarle los cambios nos aparecerá el programa de ejemplo.

Una vez ya integrados los plugins necesarios en nuestro entorno de desarrollo, podemos empezar la tarea de desarrollar la aplicación. Para ello primeramente necesitamos conocer el lenguaje Java en profundidad y familiarizarnos con las peculiaridades propias del framework instalado. Algunas de éstas son que las aplicaciones GWT necesitan un punto de acceso a la aplicación, éste se establece implementando el interface EntryPoint. También es imprescindible el método OnModuleLoad, el cual es ejecutado cuando la página se descarga del servidor y viene a ser como la clase main de la aplicación.

2. Análisis del sistema

En este apartado se procede a hacer un análisis detallado del sistema que vamos a desarrollar. En primer lugar identificamos los diferentes actores en un diagrama de contexto. Éstos nos servirán para ir extrayendo los casos de uso que vamos a implementar en el proyecto.

A continuación se hace una descripción elemental de los casos de uso mostrando la interacción usuario – sistema mediante plantillas de texto.

Por último se realizará un estudio de las interacciones cliente-servidor junto con el pase de mensajes entre los distintos elementos que forman el sistema. Éste se analizará mediante un diagrama de secuencia.

2.1. Diagrama de Contexto

Para hacer un estudio del sistema se diseña su diagrama de contexto. En este caso tenemos como actores el usuario y el clasificador. No se han considerado como usuarios ni el cliente ni el servidor ya que forman el propio sistema y no un rol que interactúa con éste.

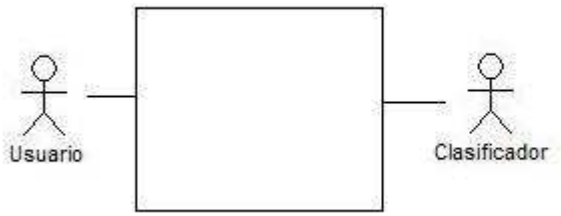

Ilustración 3.1: Diagrama de contexto.

2.2. Casos de uso

A continuación se analizan los casos de uso en los que extraemos los requisitos del proyecto.

Casos de uso relativos al proceso de entrada:

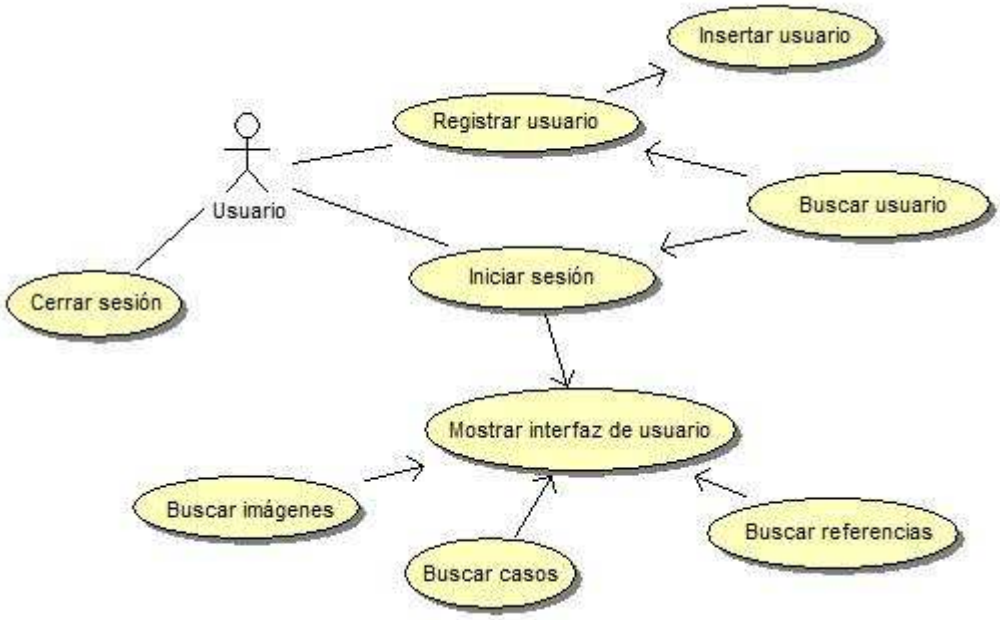

Ilustración 3.2: casos de uso para la entrada al sistema

Casos de uso del análisis médico:

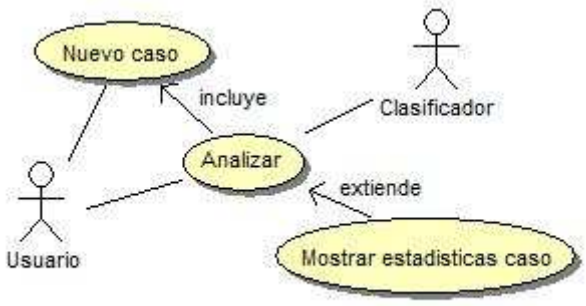

Ilustración 3.3: Casos de uso para el análisis médico.

Casos de uso de la gestión de imágenes:

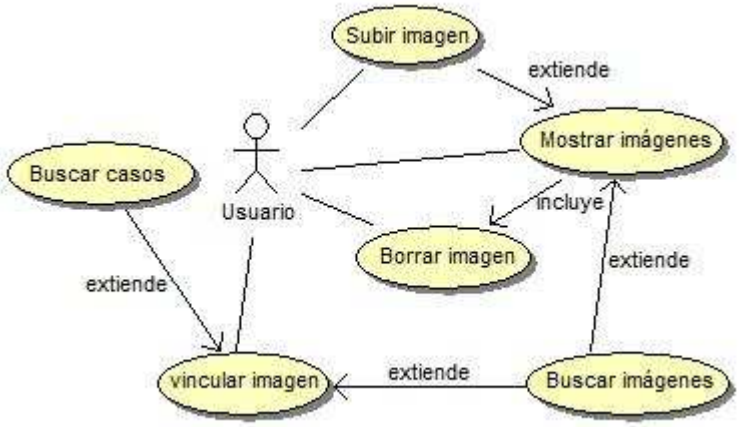

Ilustración 3.4: Casos de uso para la gestión de imágenes.

Casos de uso relativos a la eliminación de la información:

Ilustración 3.5: Casos de para la eliminación de información.

2.3. Plantillas textuales

Se ha realizado una descripción textual de las plantillas que describen los casos de uso así como del flujo de interacción entre las intenciones del usuario y las obligaciones del sistema.

Caso de uso	Iniciar sesión
Actores	Usuario (iniciador)
Precondiciones	El usuario se ha registrado en el sistema.
Postcondiciones	Se accede al panel de casos médicos.
Descripción	El usuario se identifica en el sistema y se le concede o deniega el acceso.

Flujo de eventos

Intenciones de usuario	**Obligaciones de sistema**
1. El usuario introduce su pseudónimo y su contraseña	2. El sistema busca el usuario en su almacén de datos
	3. le concede el acceso, busca sus resultados, imágenes y referencias y le muestra el panel de casos médicos
Extensiones síncronas	
	#1. Si en 2 no se encuentra el usuario se muestra un error y se vuelve al estado 1

Caso de uso	Cerrar sesión
Actores	Usuario (iniciador)
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Se vuelve a la página principal exigiéndole autorización.
Descripción	El usuario cierra la sesión actual volviendo a la pantalla de inicio.

Flujo de eventos

Intenciones de usuario	**Obligaciones de sistema**
1. El usuario pulsa cerrar sesión.	2. El sistema lo redirige a la página principal impidiéndole el acceso directo a sus datos.

Caso de uso	Registrar usuario
Actores	Usuario (iniciador)
Precondiciones	------
Postcondiciones	Los datos del usuario quedan almacenados.
Descripción	El usuario introduce sus datos y éstos quedan guardados en el sistema

Flujo de eventos

Intenciones de usuario	**Obligaciones de sistema**
1. El usuario introduce su nombre y su contraseña por duplicado.	2. El sistema busca si el usuario ya se encuentra en la BD.
	3. El sistema guarda los datos del usuario y lo redirecciona a la página principal
Extensiones síncronas	
	#1. Si en 2 el sistema encuentra el nombre o de usuario en la BD muestra un mensaje de error y vuelve al estado 1.
	#2. Si en 2 el nombre o la contraseña es menos de 6 dígitos, contiene caracteres especiales o no coinciden ambas contraseñas se muestra mensaje de error y se vuelve al estado 1.
Extensiones asíncronas	
Se permite cancelar el registro en cualquier momento.	

Caso de uso	Analizar
Actores	Usuario (iniciador)
Precondiciones	Se ha iniciado sesión.
Postcondiciones	Se muestran los resultados del análisis y quedan almacenados.
Descripción	El usuario crea un análisis, el sistema muestra los resultados y los almacena en la sesión del usuario.

Flujo de eventos

Intenciones de usuario	**Obligaciones de sistema**
1. El usuario pulsa el botón de nuevo análisis.	
2. El usuario introduce los parámetros de entrada y pulsa analizar.	3. El sistema muestra los resultados del análisis con gráficos y estadísticas, éstos quedan almacenados.
Extensiones síncronas	
#1. Si el usuario reanaliza un análisis ya hecho modificando algún campo y pulsando analizar el caso empieza en el estado 2.	#2. Si en 3 algún parámetro no es correcto se muestra un mensaje de error y se vuelve al estado 2.

Caso de uso	Subir imagen
Actores	Usuario (iniciador)
Precondiciones	Se ha iniciado sesión.
Postcondiciones	La imagen queda guardada en el servidor y se muestra en el panel de imágenes.
Descripción	El usuario sube una imagen al servidor y esta queda reflejada en el panel de imágenes.

Flujo de eventos

Intenciones de usuario	**Obligaciones de sistema**
1. El usuario pulsa el botón de elegir imagen y la elige.	2. El sistema habilita el botón de enviar.
3. El usuario pulsa el botón de envío.	4. Se envía la imagen al servidor y se agrega al panel imágenes.
Extensiones síncronas	
	#2. SI en 2 o en 4 no se puede subir se vuelve al estado 1 mostrando un mensaje de error.

Caso de uso	Eliminar casos
Actores	Usuario (iniciador)
Precondiciones	El usuario ha iniciado sesión.
Postcondiciones	Quedan borrados los casos del usuario.
Descripción	El usuario borra todos los casos médicos dejando el panel únicamente con un caso nuevo sin datos.

Flujo de eventos

Intenciones de usuario	**Obligaciones de sistema**
1. El usuario pulsa el botón de eliminar casos.	2. El sistema muestra un cuadro de diálogo cuestionando su decisión.
3. El usuario pulsa aceptar.	4. El sistema borra los datos de casos y de referencias de imágenes con casos.
	5. El sistema borra todos los casos médicos del panel y deja un panel de nuevo análisis.
Extensiones síncronas	
#1. En 3 el usuario puede cancelar su petición y se vuelve al estado 1	

Caso de uso	Vincular imagen
Actores	Usuario (iniciador)
Precondiciones	El usuario ha cargado imágenes en el servidor desde su cuenta y ha realizado un caso médico.
Postcondiciones	La imagen queda vinculada los casos médicos seleccionados.
Descripción	El usuario selecciona una imagen y al menos un caso médico, al vincularlos la imagen se muestra en todos los casos vinculados.

Flujo de eventos

Intenciones de usuario	**Obligaciones de sistema**
1. El usuario selecciona una imagen y varios casos de uso.	
2. El usuario pulsa el botón de vincular.	3. El sistema vincula la imagen a los casos seleccionados.
	4. El sistema muestra las imágenes vinculadas a los casos correspondientes en el menú de casos médicos.
Extensiones síncronas	
#1. Si en 2 el usuario no ha seleccionado una imagen y al menos un caso médico se muestra mensaje de error volviendo al estado 1.	

Caso de uso	Borrar imagen
Actores	Usuario (iniciador)
Precondiciones	Hay imágenes subidas en la cuenta del usuario.
Postcondiciones	La imagen queda borrada del servidor y eliminada de los paneles que la mostraban.
Descripción	El usuario borra la imagen y esta se elimina del servidor, del panel de imágenes y del panel de casos médicos si estába vinculada a alguno.

Flujo de eventos

Intenciones de usuario	**Obligaciones de sistema**
1. El usuario pulsa el botón ‘x’ asociado a una imagen.	2. El sistema busca la imagen y la borra del servidor. Ésta desaparece del panel imágenes y del panel de casos médicos si estaba vinculada a alguno.

Caso de uso	Eliminar cuenta
Actores	Usuario (iniciador)
Precondiciones	El usuario ha iniciado sesión.
Postcondiciones	Queda borrada la cuenta del usuario.
Descripción	El usuario borra su cuenta en el sistema con todos los datos e imágenes asociadas.

Flujo de eventos

Intenciones de usuario	**Obligaciones de sistema**
1. El usuario pulsa eliminar cuenta.	2. El sistema muestra un cuadro de diálogo.
3. El usuario introduce su contraseña y pulsa aceptar.	4. En sistema hace CALL a Eliminar casos.
	5. El sistema busca el registro con datos de usuario y lo borra.
	6. El sistema se redirige hacia la página inicial de la web.
Extensiones síncronas	
#1. En 3 el usuario puede cancelar su decisión pasando al estado 1.	#2. si en 4 se verifica que la contraseña es incorrecta, se muestra un mensaje de error y se vuelve al estado 3.

2.4. Diagramas de secuencia

Se analizan las interacciones cliente-servidor junto con otros elementos.

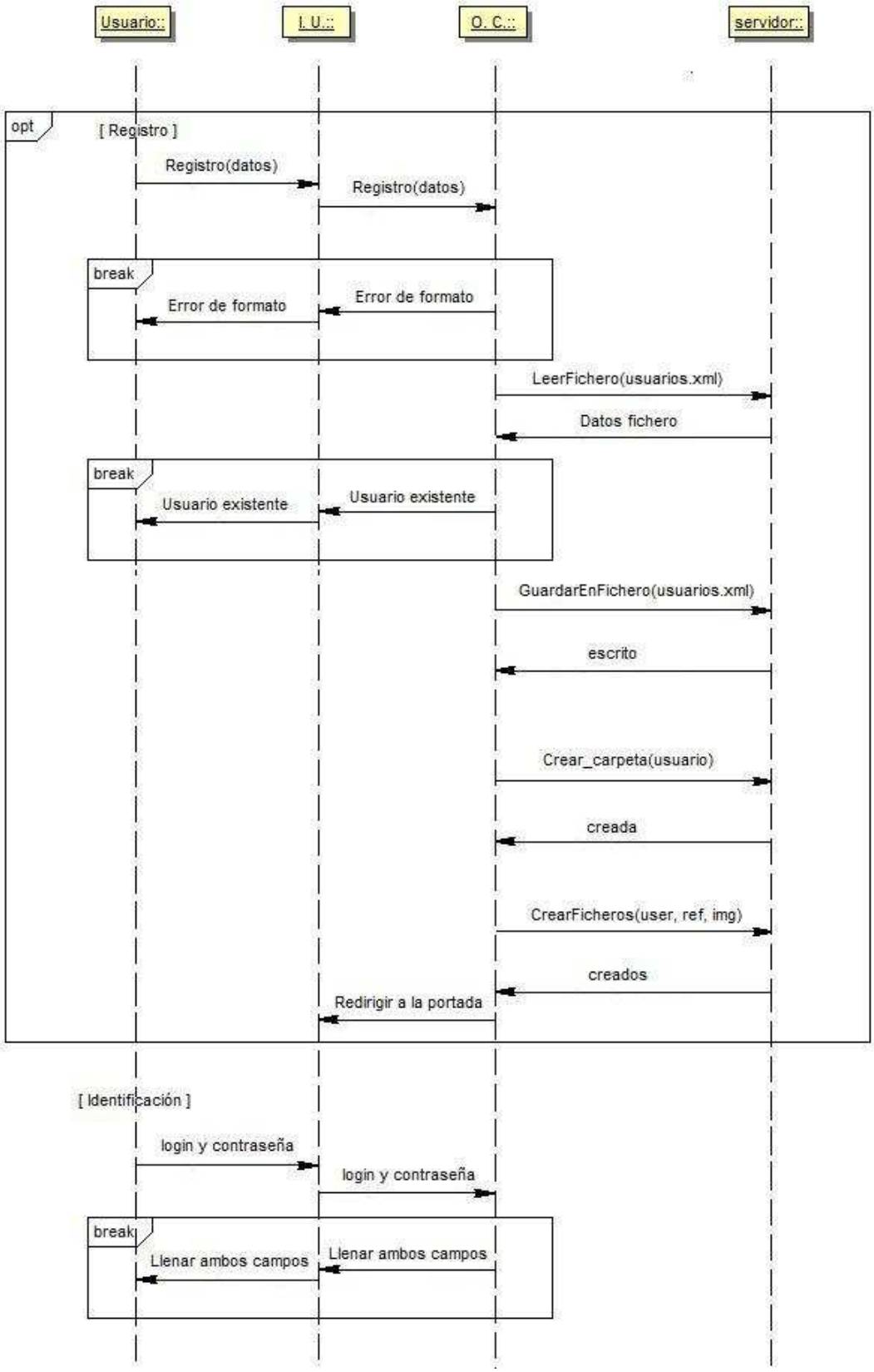

Ilustración 3.6: Diagrama de secuencia.

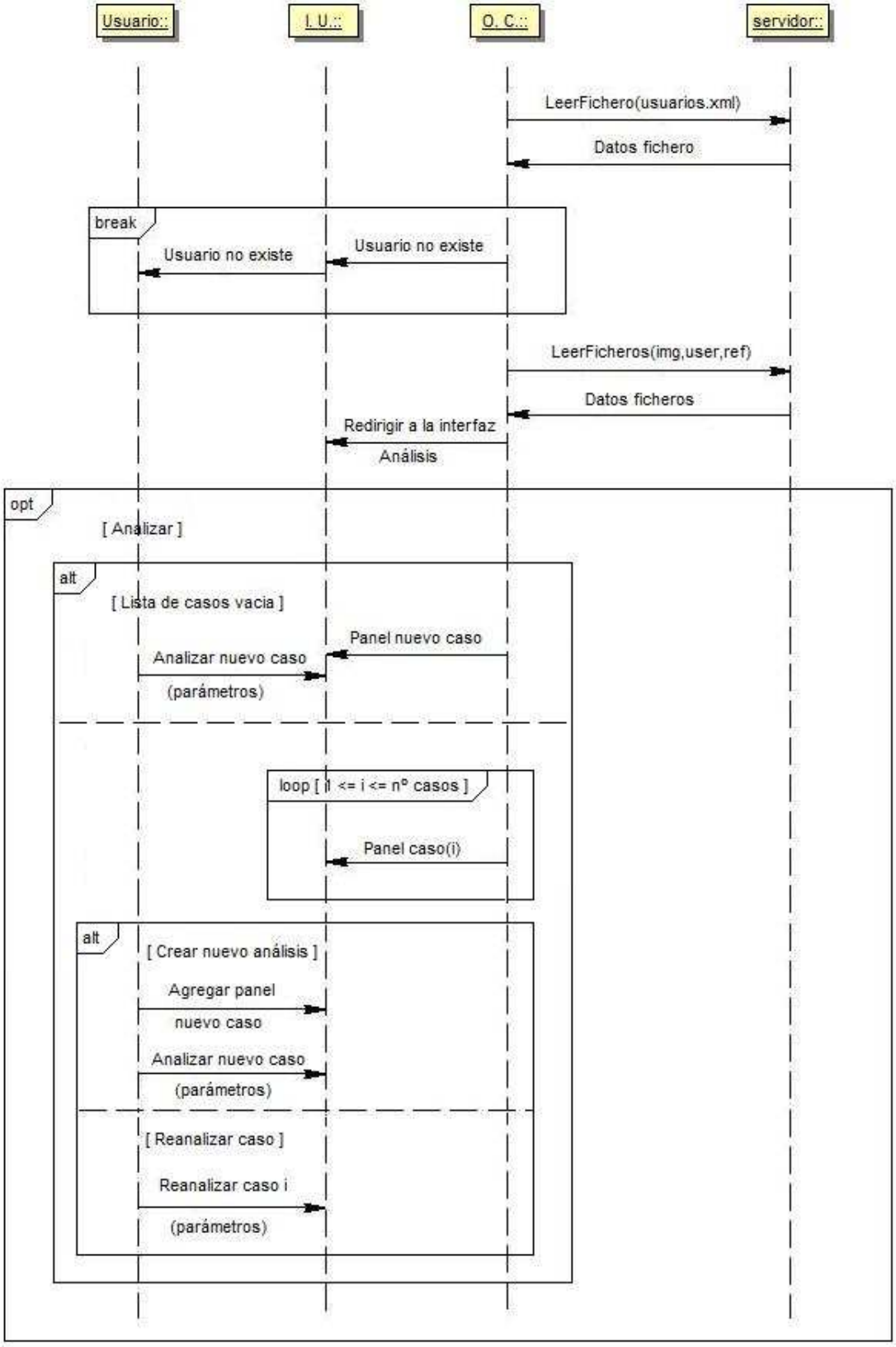

Ilustración 3.7: Diagrama de secuencia.

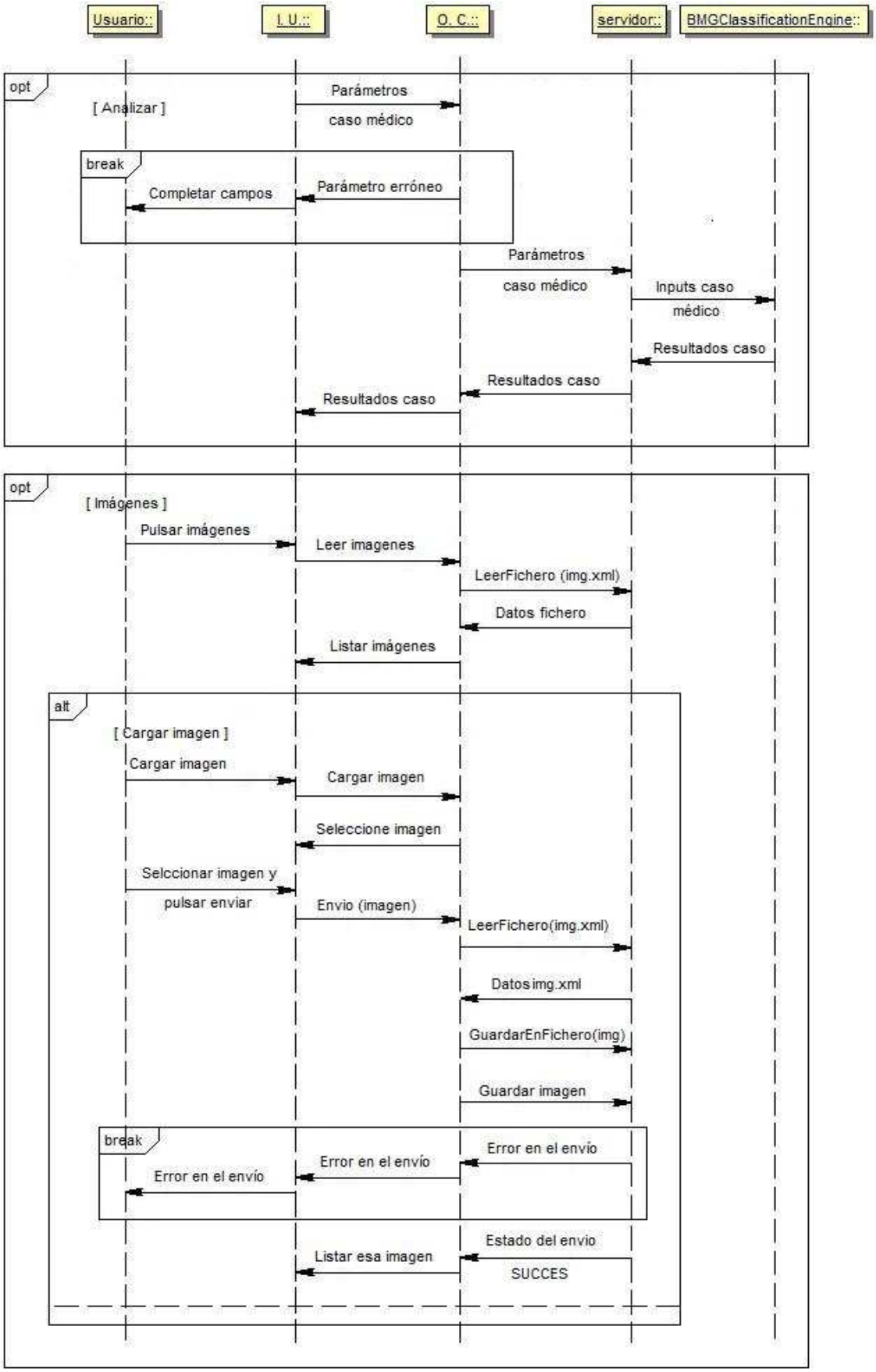

Ilustración 3.8: Diagrama de secuencia.

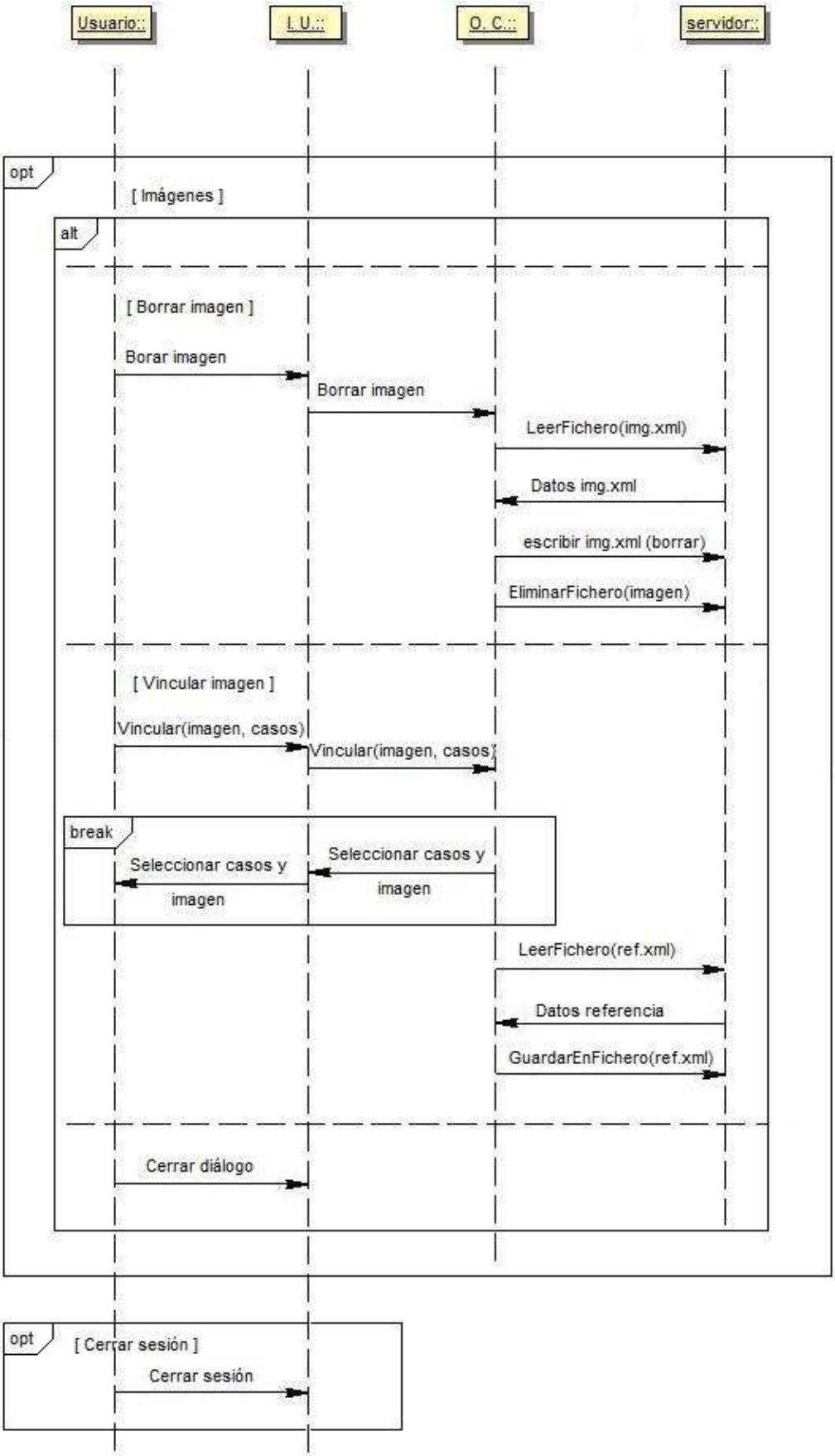

Ilustración 3.9: Diagrama de secuencia.

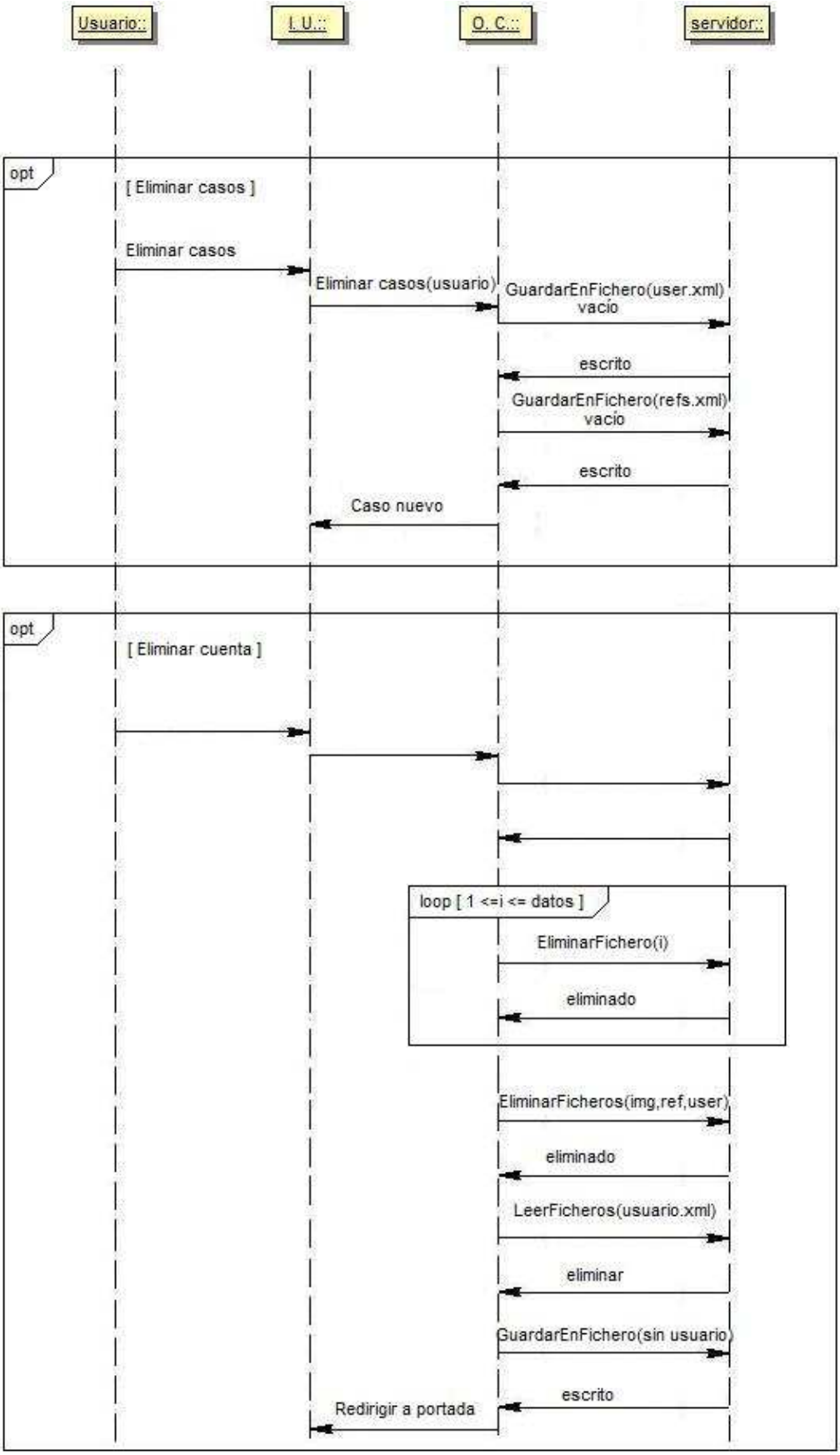

Ilustración 3.10: Diagrama de secuencia.

3. Base de datos

La aplicación se ha creado con una pequeña base de datos que almacena los datos esenciales para el funcionamiento de la misma.

En la siguiente imagen podemos ver su diagrama de clases:

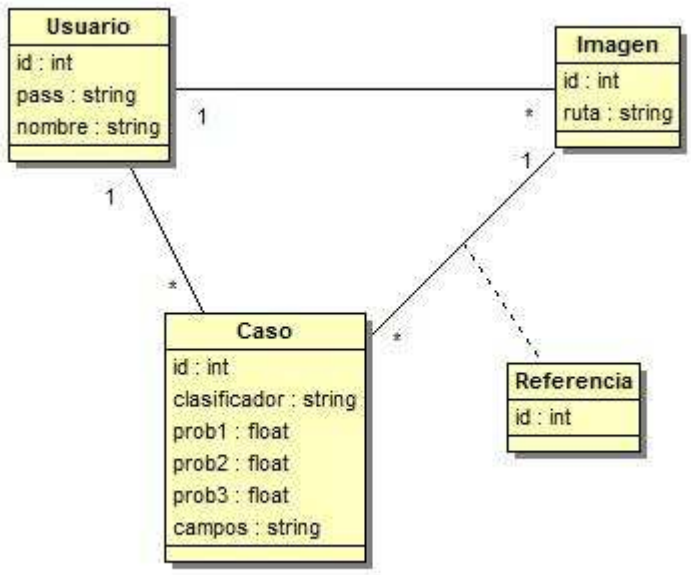

Ilustración 3.11: Diagrama de la BD

La principal es la clase usuario, la cual solo guarda el nombre o pseudónimo del usuario y su contraseña junto a un identificador que se asigna automáticamente. Se hubieran podido recopilar muchos más datos del usuario como país o e-mail, pero se ha considerado innecesario y así el registro se vuelve más cómodo y rápido para el usuario.

De cada caso médico analizado se guarda el nombre del clasificador que lo ha analizado, las probabilidades resultantes y un campo que guarda una lista de las posiciones de los valores seleccionados en los ListBox.

La tabla imagen almacena la ruta y un identificador de la imagen y la tabla referencia guarda tuplas que vinculan una imagen con una lista de identificadores de casos.

La base de datos la componen ficheros XML. Se ha creado uno por tabla para su sencilla manipulación. Estos han sido guardados siguiendo una

pequeña jerarquía de directorios. Dentro del directorio war/config/ se guarda la BD. Ésta almacena el archivo usuarios.xml que representa dicha tabla y que va almacenando los datos de registro de cada usuario. En la misma ubicación se crea un directorio para cada usuario con su nombre, en el que se almacenan 3 ficheros XML correspondientes a las 3 otras tablas, la cuales guardan los casos, las referencias o vínculos, la información de las imágenes y las propias imágenes.

Para poder manipular la base de datos se ha creado una serie de funciones que interactúan con la tecnología cliente – servidor, como veremos más adelante, ya que las clases que manipulan ficheros solamente las puede gestionar el servidor, que a su vez es el que alberga los archivos que componen la base de datos. Estas clases crean y eliminan directorios al igual que leen, escriben y borran ficheros.

A continuación se procederá a explicar el procedimiento de lectura y escritura de ficheros XML.

Para escribir datos se ha creado en el servidor una función llamada g*uardarEnFichero* que recibe una cadena con el nombre del archivo y otra con el texto que almacena. Ésta abre el archivo en modo escritura y le escribe el texto que le pasamos por parámetro devolviendo true al cliente si la operación se realiza correctamente.

Desde el cliente llamamos a la función *escribirXML* dentro de la clase Acceso, la cual contiene los dos mismos parámetros y se encarga de llamar al servidor sacando sus correspondientes mensajes de error si no obtenemos respuesta del mismo o si nos responde con un valor falso de petición realizada correctamente. El parámetro con los datos del fichero que le pasamos al servidor es un string correctamente formateado para ser almacenado.

En el caso de lectura de fichero, en la parte servidor se ha creado una función llamada *leerFichero* que contiene un parámetro con el nombre del mismo y que devuelve un string con los datos leídos.

En la parte cliente, en el fichero Acceso hay varias funciones que llaman al servidor para leer datos. Normalmente, dentro de éstas, se define la llamada al servidor y ésta devuelve el string con los datos del archivo. Según la respuesta, el cliente nos informa con mensajes de error si el servidor no responde o si el string recibido es nulo. Si la respuesta a la petición es correcta, el servidor devuelve un string con los datos. Este string se formatea y se le pasa a un ArrayList que tiene como tipo la tupla requerida según el fichero. Normalmente este proceso de *vectorización* lo realizan las funciones:

ArrayList<tupla> = parsear_[nombre_de_tabla](String datos)

Una vez hecho esto se nos es más fácil manipular todos los datos leídos del fichero.

En la función descrita para la conversión de String – ArrayList<tupla> se define la clase ModelType y mediante sus métodos *setRoot(etiqueta), setRecordName(etiqueta)* y *addField(etiqueta)* se va informando de las etiquetas del fichero XML. Esta clase se le pasa como parámetro para crear la clase XmlReader, la cual con el método *read* va leyendo el string y lo va guardando en un ArrayList.

4. Comunicación cliente – servidor

Como se ha introducido anteriormente, GWT utiliza un sistema de cliente – servidor[11] en sus aplicaciones web. La razón es aligerar la carga en el servidor para que no se colapse. A su vez, este mecanismo sirve para vincularle al servidor los archivos guardados en el sistema y en general para ejecutar todo tipo de código Java independientemente de que sean clases diseñadas expresamente para GWT.

A continuación procederemos a describir los elementos que componen esta interacción.

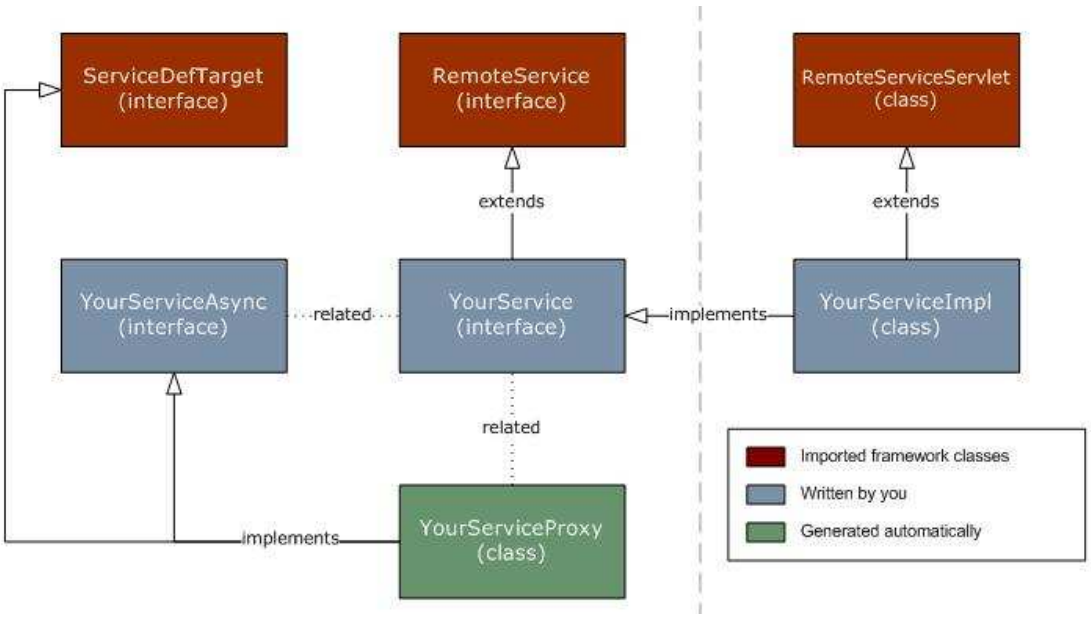

Ilustración 3.12: Componentes de la interacción cliente – servidor.

En la ilustración 3.12 aparece la relación de clases que interactúan. En ella vemos separada por una línea discontinua la parte servidora del cliente. Cada servicio tiene una pequeña familia de interfaces y algunas clases como la del servicio proxy se generan automáticamente.

Las llamadas siempre siguen la misma metodología de clases. Se define la interfaz *RemoteService,* que vemos en la imagen, en la que se listan todos los métodos destinados a hacer peticiones.

Se crea la clase *YouServiceImpl* que extiende a la otra clase *RemoteServiceServlet*. Le implementamos las funciones de la interfaz anterior y por último se crea la interfaz asíncrona en la parte cliente.

Aunque con el Eclipse su creación está bastante automatizada debido a la función de autocompletar código, vamos a explicar paso a paso para poder entenderla y reproducirla en cualquier entorno adaptado.

Empezaríamos creando la parte cliente con la interfaz síncrona que extiende a RemoteService.

```
public interface miservicio extends RemoteService {
  public String leerFichero(String ruta);
}
```

Creamos la clase servidora que extiende a *RemoteServiceServlet* y le integramos el método cuya interfaz ya hemos definido en el cliente.

```
public class miservicioImpl extends RemoteServiceServlet implements
    miservicio {

 public String leerFichero(String ruta){
    String datos;
    //desarrollamos la función
    return datos;
  }
}
```

También es necesario la interfaz asíncrona en la parte cliente para que permita gestionar la petición y el servidor devuelva su respuesta a dicha parte.

```
interface miservicioAsync {
  public void leerFichero (String ruta, AsyncCallback<String>
callback);
}
```

Por último debemos configurar el archivo web.xml del directorio WEB-INF para indicar donde se encuentra servlet y como mapearlo.

```
<servlet>
  <servlet-name>miservicioImpl</servlet-name>
  <servlet-class>
    com.example.foo.server.miservicioImpl
  </servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>miservicioImpl</servlet-name>
  <url-pattern>/com.example.foo.Foo/miservicio</url-pattern>
</servlet-mapping>
```

La etiqueta servlet-name indica el nombre que le daremos al servlet y la servlet-class el nombre de su clase. En servlet-mapping indicamos que para llamar al servlet *miservicioImpl* hay que hacerlo a través del directorio contenido en el url-pattern.

5. Componentes Gráficos

5.1. Widgets

Uno de las ventajas de esta herramienta, es la facilidad de añadirle componentes gráficos predeterminados. Estos componentes los encontramos en varias webs y permiten diseñar el servicio dándole el aspecto claro y limpio que caracterizan las interfaces de Google.

Los componentes visuales más importantes que Google pone a nuestra disposición los podemos observar en la ilustración 3.13.

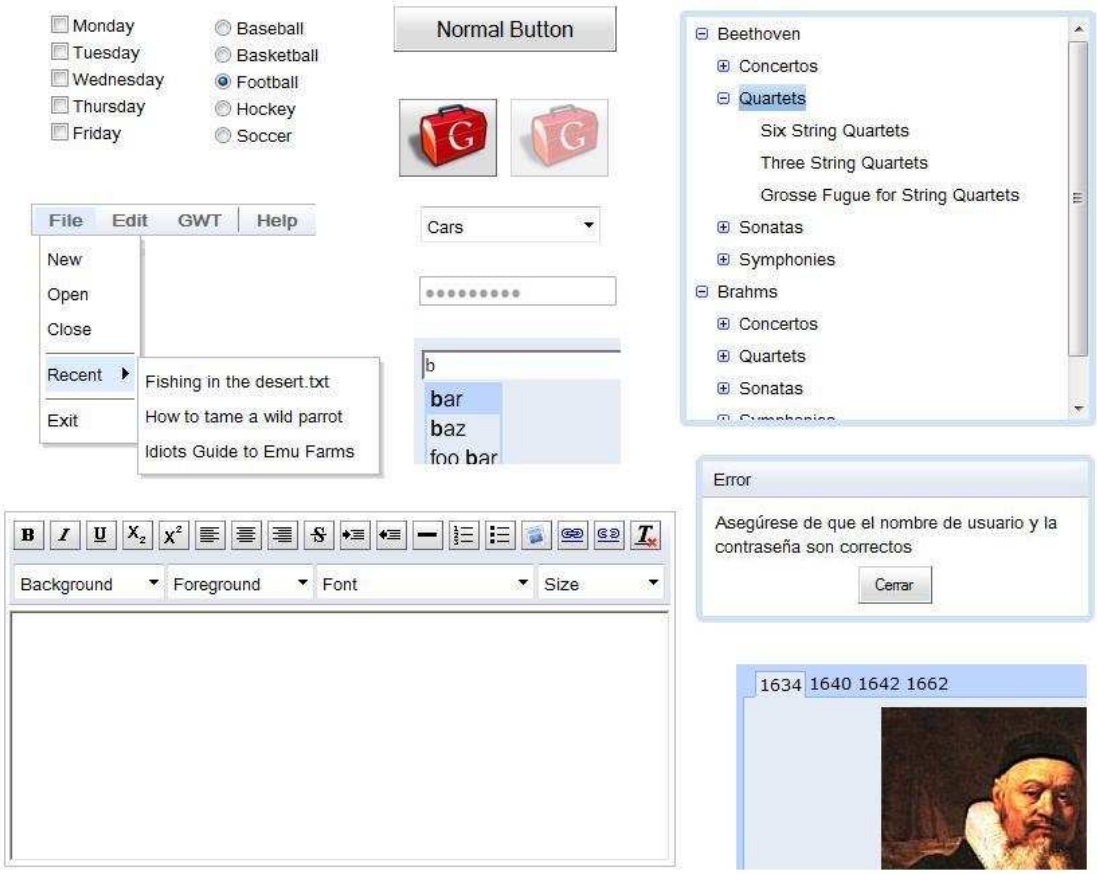

Ilustración 3.13: Widgets de GWT

Estos componentes se incorporan en el código nombrándolos como clases y accediendo a sus atributos a través de funciones que proporcionan los mismos.

En la imagen 3.13 vemos los típicos componentes visuales como son la casilla de verificación, el radiobutton, el botón simple, botones con imágenes e incluso que la cambian según el estado, como los botones de presionar o de alternar, el cuadro de lista, la caja de texto, la habilitada para contraseñas, la que nos proporciona sugerencias al realizar una búsqueda[12]…

También nos proporciona componentes más elaborados como puede ser un pequeño editor donde poder personalizar el texto que introduce el usuario en la web o barras con menús de elementos o de pestañas que nos clasifiquen apartados.

Esta herramienta permite además crear tablas estáticas y dinámicas. Las estáticas las nombra *Grid* y al definirlas se le pasan como parámetros el

número de filas y columnas. Las dinámicas son conocidas como *FlexTable*, no necesitan parámetros al definirlas y las filas y columnas se van añadiendo interactivamente según la necesidad de la aplicación.

Para poder vincular todos los elementos descritos hasta el momento, necesitamos unos soportes que permitan indexarlos de forma ordenada para mostrarlos por pantalla. Éstos son los paneles[13] y algunos de los más importantes podemos observarlos en la ilustración 3.14.

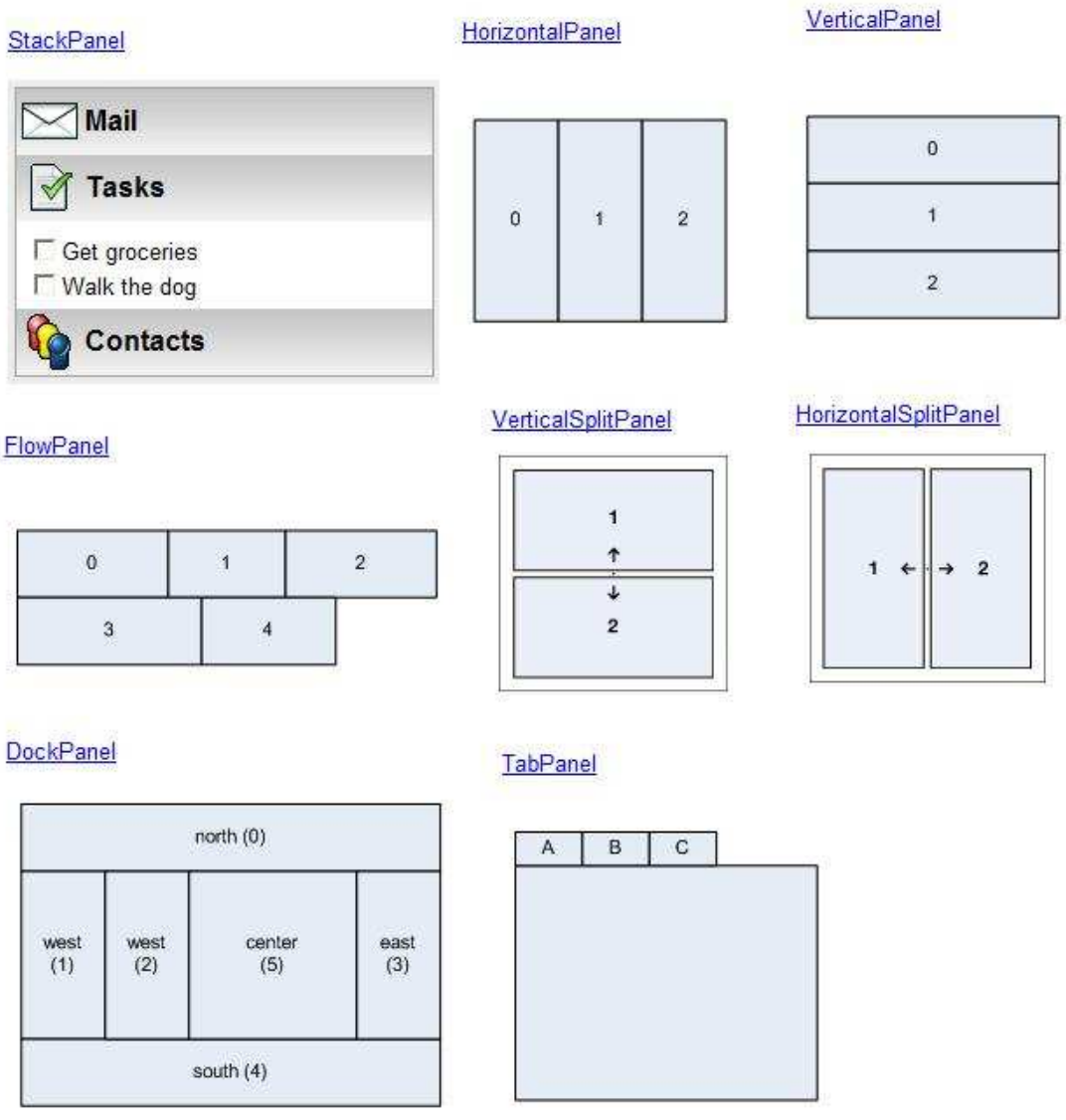

Ilustración 3.14: Paneles de GWT

Éstos se describen a continuación:

- StackPanel: Panel desplegable incorporado en la aplicación en el apartado de casos médicos.
- DecoratedStackPanel: Igual pero cambiando algunas características en el diseño.
- HorizontalPanel, VerticalPanel: Lista visual de elementos representados de la forma descrita en su nombre.
- FlowPanel: indexa el flujo de contenidos de manera natural.
- VerticalSplitPanel, HorizontalSplitPanel: Permiten organizar dos Widgets en una columna vertical u horizontal y posibilitan al usuario el cambiar de forma interactiva la proporción de la altura o la anchura respectivamente reservada a cada uno de los dos sub-paneles.
- TabPanel: Distribuye los paneles por pestañas
- DockPanel: Establece una división general de la web con anidamiento de paneles internos distribuidos según su posición en la pantalla.

En el sitio web hemos utilizado un DockPanel para la estructura de la interfaz externa, un StackPanel en el listado de los casos médicos y todos los paneles horizontales, verticales y de flujo que han sido necesarios.

5.2. Librería EXT GWT

En nuestro caso, debido a la cantidad de datos estadísticos que hemos tenido que manipular, los componentes de Google han sido insuficientes para crear la aplicación. Por eso se ha creído conveniente recurrir a otras librerías visuales compatibles.

Una de las más completas y más perfeccionadas es Ext GWT[14] también conocida como GXT, la cual contiene hasta el momento unos 93 Widgets que conforman tablas, árboles, pestañas, gráficos, componentes de arrastrar y soltar, ventanas, controladores de distribución, cuadros y listas de texto, formularios, enlaces de datos, barras de herramientas, menús, plantillas,

botones… y muchos más elementos con un alto nivel de detalle. Algunos de los elementos se muestran en la imagen 3.15.

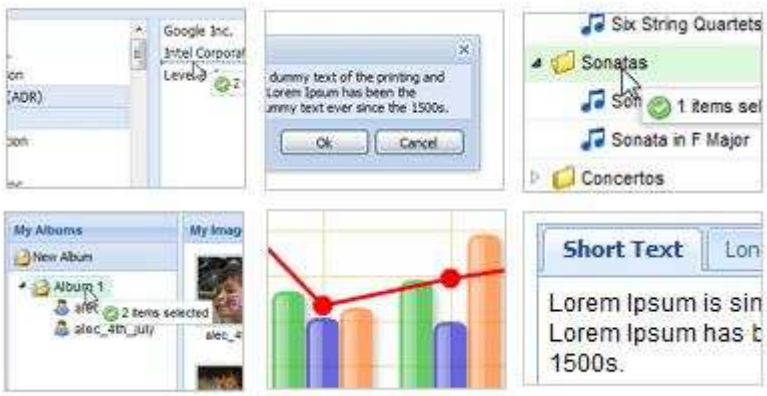

Ilustración 3.15: Widgets de la librería Ext-GWT

Para incorporarle la librería al proyecto,

- La bajamos del sitio oficial[9], dentro de la misma hay un directorio llamado recursos, lo renombramos en nuestro caso como gxt.
- Lo copiamos dentro del directorio war/ de nuestro proyecto.
- Enlazamos la hoja de estilo con la página principal HTML mediante el comando:

```
<link rel="stylesheet" type="text/css" href="gxt/css/gxt-all.css" />
```

- Si le añadimos gráficos, como ha sido nuestro caso, también le vinculamos el siguiente script:

```
<script language='javascript' src='gwt/flash/swfobject.js'></script>
```

- Le agregamos la siguiente línea dentro de archivo Curiam.gwt.xml:

```
<inherits name='com.extjs.gxt.ui.GXT'/>
```

- Comprobamos que el HTML contenga el siguiente doctype:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

- Enlazamos el archivo gxt.jar pulsando el botón derecho sobre el proyecto, propieties / Java build path, seleccionamos la pestaña libraries, add external JAR.

- Abrimos el diálogo ejecutar, configuraciones de ejecución, seleccionamos la configuración utilizada, la pestaña ruta de clase y añadimos gxt.jar

- Llegados a este punto para añadir un componente visual creamos una clase nueva y le copiamos de la web GXT el código Java referente al widget que queremos agregar.

En nuestro caso de esta librería hemos utilizado el componente *BasicChart*, el cual para que funcione además hay que añadirle en Curiam.gwt.xml la línea:

```
<inherits name="com.extjs.gxt.charts.Chart"/>
```

También se ha integrado de esta librería el componente tabla por su calidad gráfica. El proceso de integración ha sido copiar el código Java de la web oficial Ext GXT y a continuación buscar por internet un par de clases de esta librería que faltaban. Una de ellas es la clase *Stock* que establece los campos o columnas que va a tener la tabla, dándole un nombre a cada columna. La otra es *TestData* cuya funcionalidad consiste en ir insertando filas o clases Stock con información para ir rellenando la tabla.

5.3. Librería Gwtupload

A la hora de intentar subir imágenes GWT y GXT ofrecen sólo el soporte visual, es decir, únicamente el widget que permite seleccionar el archivo y un botón de cargar sin ninguna funcionalidad agregada. Con lo cual indagando por la web y comprobando como éste es un problema en el que GWT no ha aportado demasiado, se ha diseñado y documentado una librería independiente la cual permite solucionar este inconveniente. La librería, conocida como Gwtupload[15], permite al usuario subir archivos al servidor mostrando una barra de progreso.

El componente servidor de la librería consiste en un servlet que recibe ficheros, el cual al preguntarle vía AJAX nos devuelve el estado de subida mientras la parte cliente se encarga de gestionar la cola de imágenes.

Esta librería está alojada en Google Code como un proyecto de software libre, con lo cual si la buscamos en esta plataforma comprobaremos el conjunto de versiones que ha subido el autor. Bajamos la última y la incluimos en el classpath de Eclipse.

A continuación debemos de incluirle otras tres librerías que son log4j.jar, commons-fileupload-1.2.jar y commons-io-1.3.1.jar

log4j.jar es una herramienta para administrar avisos y mensajes de depuración de código y las otras dos, como su propio nombre indica, son librerías para cargar archivos y para manipular la entrada y la salida de información.

En este momento, editamos el módulo Curiam.gwt.xml con el siguiente código:

```
<module>
    <inherits name="gwtupload.GWTUpload"/>
    <stylesheet src="Upload.css"/>
</module>
```

Este código se encarga de cargar la librería GwtUpload y su hoja de estilo asociada.

A continuación en web.xml incorporamos el siguiente código:

```
    <servlet>
        <servlet-name>uploadServlet</servlet-name>
        <servlet-class>com.curiam.server.GettingStartedServlet</servlet-
    class>
    </servlet>
    <servlet-mapping>
        <servlet-name>uploadServlet</servlet-name>
        <url-pattern>*.gupld</url-pattern>
    </servlet-mapping>
```

El cual establece el nombre y la ruta del servlet y el paquete donde en nuestro caso lo hemos guardado.

A continuación copiamos el código de la clase cliente y servidor que por su extensión no vamos a copiar aquí, pero que está disponible en la web Gwtupload[10] de Google Code, apartado wiki / GwtUpload GettingStarted. En el proyecto la parte servidor está desarrollada dentro de la clase *GettingStartedServlet* y el cliente lo forma la clase *GettingStarted*. Cabe puntualizar que dichos archivos han sido modificados para adaptarlos al funcionamiento del sitio web.

En concreto, en el proyecto se ha cambiado el archivo temporal con el que se creaba la imagen en el servidor, por un archivo permanente. Además se ha modificado la parte cliente para que al acceder al *DialogBox* de imágenes, aparte de cargar y mostrarnos las imágenes que se han subido desde la última sesión (que es lo que hace la librería por defecto) se nos listen todas las imágenes que hay ubicadas en el servidor dentro de la cuenta del usuario. También se le han añadido detalles necesarios como que se puedan borrar las imágenes, enlazarlas a los casos médicos y que éstas se actualicen en el StackPanel principal, mostrándose o borrándose en los casos analizados según sea necesario.

5.4. Google Chart API

A la hora de programar el gráfico de tarta se ha escogido el de la librería GXT tal y como hemos comentado antes, debido a que no está disponible entre los Widgets de GWT y, aunque ese gráfico se puede dimensionar, no cabía la posibilidad de empequeñecerlo tanto como para ponerlo en el titulo de las solapas que representan los casos médicos.

Con lo cual, buscando otras alternativas, se ha encontrado una API[16] de gráficos creada por Google la cual permite insertar gráficos de la forma más sencilla vista hasta el momento. En la ilustración 3.16 se pueden apreciar algunos de los gráficos que podemos añadir a nuestras aplicaciones.

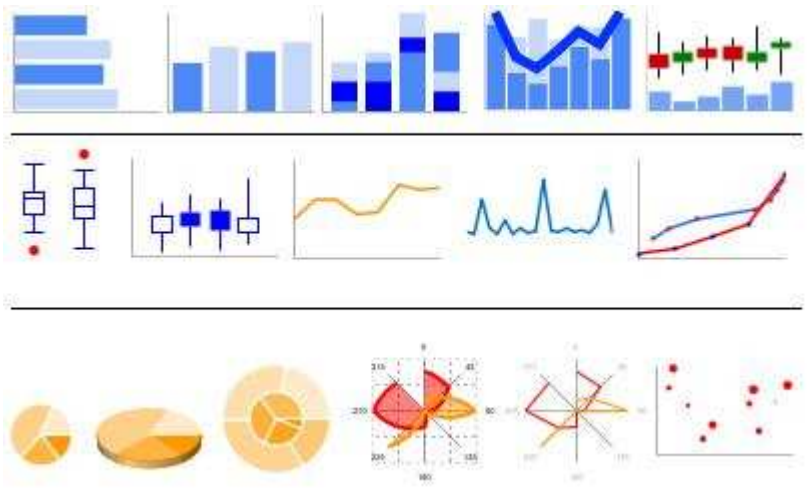

Ilustración 3.16: Gráficos de la API Google Chart

Estos gráficos se generan en el servidor de Google y se añaden en la aplicación del usuario como imágenes cuya URL es el generador automático de gráficos de Google. Éstos permiten su personalización pasándole parámetros de tipo GET a la misma URL relativos al tipo de gráfico, tamaño, porcentages, leyenda, rotación, márgenes, gradientes de color…

La API es de uso gratuito y aunque inicialmente limitaba su uso a 50.000 peticiones por día y url, actualmente han ampliado el rango hasta 250.000, el cual parece un valor bastante aceptable.

6. Desarrollo de la Interfaz Gráfica

Por último vamos a comentar a modo de resumen la forma general en la que hemos programado la interfaz gráfica para que no se nos olvide ningún aspecto relevante en la parte visual.

Como hemos dicho, el código del proyecto, está divido en paquetes que implementan la parte cliente y servidora. El cliente está compuesto además por los paquetes componentes, datos, interfaz y llamada.

El paquete interfaz es el que contiene las clases que conforman la interfaz externa. En él la clase *interfaz_portada* contiene los *Widgets* necesarios para desarrollar la página de presentación. Éstos básicamente son un *DockPanel* que ha servido para crear su estructura general y varias tablas que conforman secciones como el menú lateral y otros detalles estéticos. En su parte izquierda se ha utilizado un *DecoratedPanel* para crear el panel de

acceso. Los enlaces del menú izquierdo son botones decorados con CSS que integran un texto HTML diferente en la tabla de texto principal según el botón que sea pulsado, con lo que nos evitamos tener que crear una clase para cada apartado minimizando la cantidad de código y el tiempo de carga de la web.

Sin embargo el apartado de registro difiere lo suficiente como para crearle su propia clase llamada en este caso, *interfaz_registros* cuyos componentes son bastante parecidos.

En el paquete com.curiam.client tenemos la clase Análisis que representa la interfaz interna cuando entramos en la aplicación con nuestro pseudónimo y contraseña. Esta clase implementa la parte visual con el menú superior de botones en un *HorizontalPanel*, los cuadros de diálogo que estos muestran al pulsarse y la creación del *StackPanel* con los casos de uso.

En el mismo paquete se ha creado la clase *Acceso*, con la intención de albergar la funcionalidad interna de la aplicación formada por el conjunto llamadas al servidor y procesos internos de funcionamiento.

Nuestra aplicación dispone de una base de datos y su manera de funcionar es la de ir almacenando los casos médicos en la cuenta del usuario a medida que éste pulse el botón analizar. Con lo cual al entrar en la interfaz interna identificándonos, se nos lista el *StackPanel* con los casos analizados de la sesión anterior. Para realizar esta tarea a nivel interno se ha creado en la clase *Acceso* la función *solapa,* que nos declara y dibuja todos los componentes de un caso médico compuesto por los parámetros de entrada, el botón de análisis y los resultados de la clasificación. Esta función además llama al servidor para interactuar con la librería de clasificación *BMGClassificationEngine* y así recuperar los datos del análisis e insertarlos en el panel izquierdo mediante gráficas y estadísticas, tal y como explicaremos más detalladamente en el siguiente capítulo. Volviendo al tema de la interfaz, es fácil suponer que en la interfaz interna del usuario contiene la definición del *StackPanel* y un bucle que va añadiéndole los casos médicos recuperados a partir de los datos que lee de los XML.

A esta función *solapa* se le pasa el *StackPanel*, la posición del panel o hoja que le estamos creando, la ruta del fichero en la que se guardarán los resultados de su análisis, un *ArrayList* con los valores de dicho fichero ya leído (aunque la llamada se hubiera podido hacer dentro de la función, se ha procedido a hacerla fuera para que en cada solapa u hoja del panel no se tenga que leer el archivo entero XML, sino que se le pasen los datos leídos sólo una vez a la función solapa y así nos evitamos tener que ir leyendo el archivo en cada caso analizado). También se le pasa un *ArraList* de tipo *String* llamado *refcaso* que alberga el nombre de las imágenes ordenadas por casos, la solapa agregará sólo la perteneciente a la posición pos dentro del vector. Por último hay un parámetro booleano llamado inicio el cual si es true indica que la

pestaña que estamos creando es un nuevo caso, sin aún estadísticas ni parámetros seleccionados, o si es un análisis ya hecho del que se espera toda su información asociada.

La distribución de una hoja del *StackPanel* está formada por tres tablas según se ve en la ilustración 3.17:

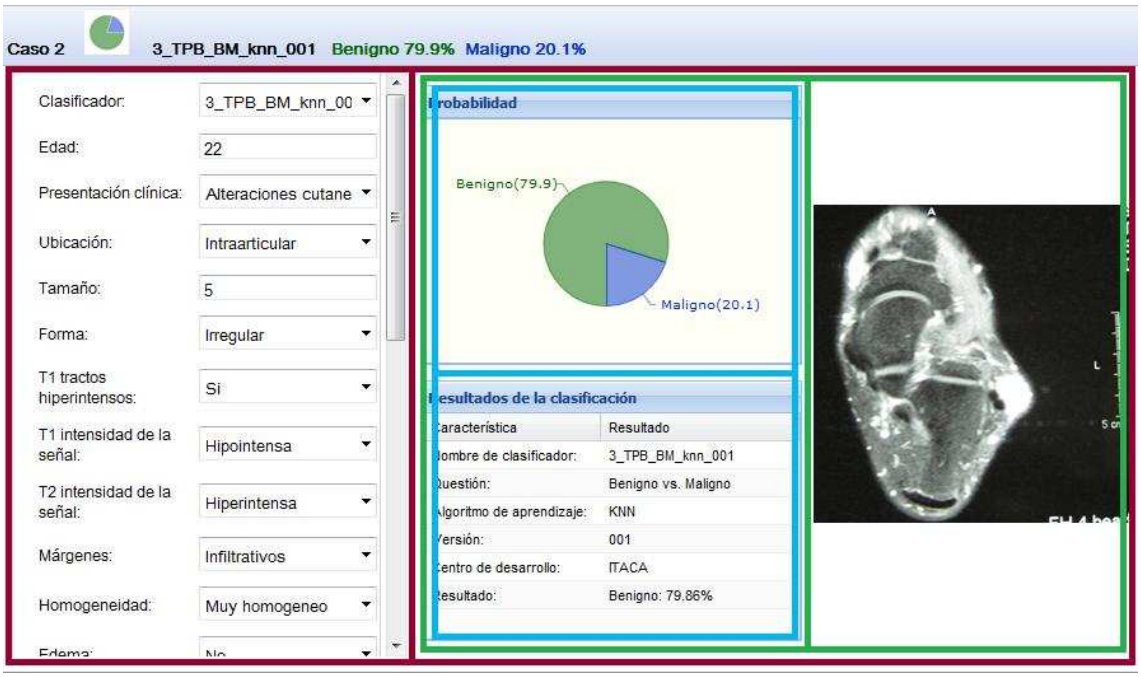

Ilustración 3.17: Panel de caso de estudio con las tablas internas remarcadas.

Esta distribución se ha hecho así para que al modificar la aplicación resulte más fácil perfeccionar la parte de resultados, eliminando simplemente la tabla en verde y dejando la sección de parámetros intacta. Estas 3 tablas en el código se llaman t_ext1, t_ext2 y t_ext3 de fuera a dentro respectivamente.

Siguiendo con este panel, podemos observar que la gráfica y tabla de resultados de clasificación forman parte de los Widgets de la librería GXT. Estas clases están dentro del paquete componentes formadas por las clases *BasicChartExample* y *GridExample* respectivamente. A éstas se les ha creado un constructor en el que se le pasaban los resultados del análisis y éstos se han añadido como parte de los datos que muestran por pantalla. Otros elementos del paquete componentes es un clase llamada *Utiles* la cual alberga un diálogo para los mensaje de error, una función de modificación del aspecto de la letra y una llamada post destinada a interactuar con Google Health.

Otro aspecto visual importante es la gestión de imágenes que se observa en la imagen 3.18:

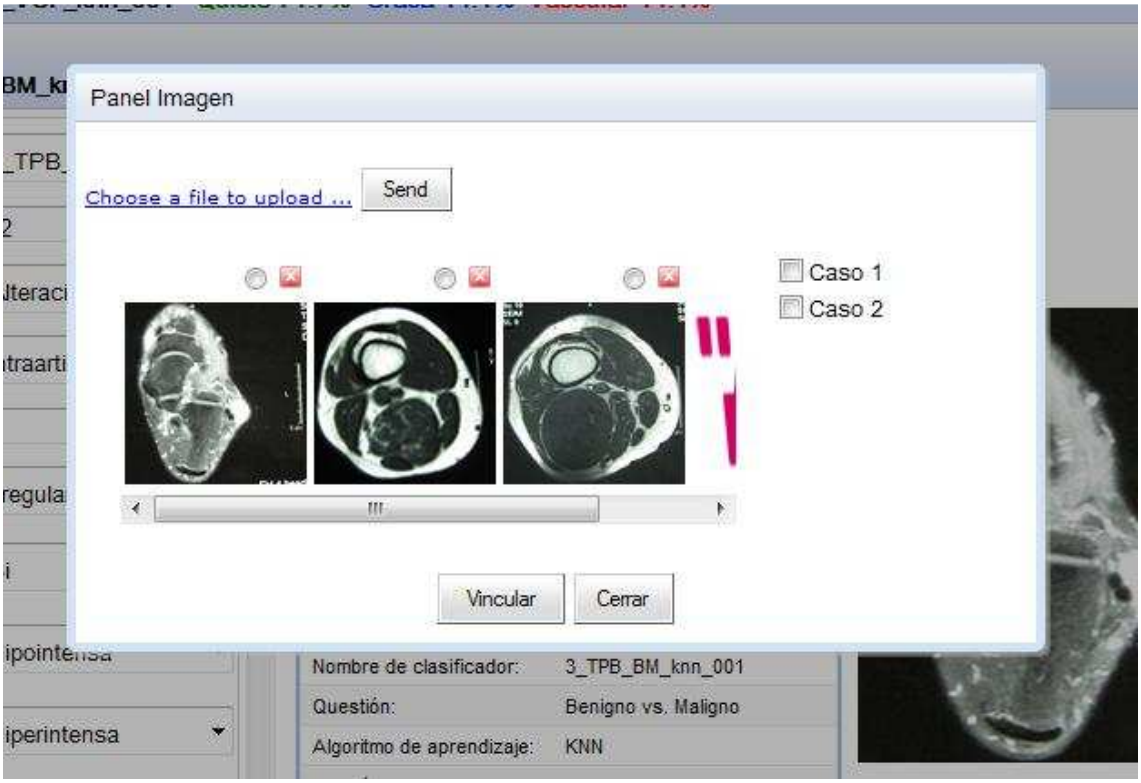

Ilustración 3.18: Panel de gestión de imágenes.

En la cual vemos se ha creado un *DialogBox* que cargan imágenes, las muestra, permite eliminarlas y enlazarlas con los casos médicos.

Para su desarrollo se ha utilizado la clase *GettingStarted* de la librería GwtUpload que forma la parte cliente de la misma y la Clase *GettingStartedServlet* que proporciona la parte servidora.

A la clase cliente se le ha incorporado el *ScrollPanel* de la imagen con la lista de imágenes disponibles en la cuenta del usuario ubicadas en el servidor. En este caso, para cada imagen se crea una tabla externa que contiene la imagen y a otra tabla interna. Esta última a su vez contiene un radiobutton para elegir la imagen a vincular y un botón para eliminar la imagen del servidor junto a sus vínculos y sus visualizaciones en el *StackPanel* como vemos en la parte trasera de la caja de diálogo de la ilustración 3.13.

Para poder acceder a elementos (como la imagen ubicada en el *StackPanel*) desde el *DialogBox* se ha utilizado un anidamiento de métodos *getWidget* los cuáles al hacerle un casting devolvían el elemento integrado con la posibilidad de acceder sus propios métodos. Se ha procedido de la misma

forma para leer los *CheckBox* y *RadioButtons*, que con un anidamiento de métodos *getWidget*, *getElement* y *getChild*, se han leído elementos hasta llegar a las clases finales, las cuáles no podemos acceder de forma directa por estar definidas inicialmente desde un bucle y a las que le hemos leído finalmente la sentencia *checked* para escoger si las vinculamos.

Otro pequeño detalle es que la librería GwtUpload contiene sus propios métodos para saber si ha terminado la transferencia del fichero devolviéndonos el estado del mismo. Con lo cual al subir el fichero cuando éste devuelve *Success* le vinculamos al panel de imágenes la última imagen junto a sus propiedades de vinculado y borrado para no tener que cargar el panel entero nuevamente.

Capítulo 4: Servicio online de diagnóstico de tumores de partes blandas

1. Introducción

Actualmente, con el crecimiento y la expansión de internet, inevitablemente se tiende a desarrollar una tecnología progresivamente más evolucionada y destinada a la creación de las aplicaciones online cada vez más exigentes. Con un alto grado de detalle e incluso con la posibilidad de adaptar todo tipo de módulos basados en cualquier metodología relativa a las ramas de la ingeniería del software y de la inteligencia artificial.

Ante esta evolución de los servicios online, éstos dejan de ser una simple web en la que interactúa el usuario, para convertirse en aplicaciones inteligentes capaces de recrear todo tipo de software. Con esta tendencia es lógico suponer que la red a día de hoy proporciona una plataforma que fomenta la competencia entre empresas y dota de soporte a las instituciones. Ante esta inclinación cada vez más aplicaciones o servicios antiguamente cerrados encuentran nuevas líneas de mercado y desarrollo, aprovechando positivamente las ventajas y posibilidades que a día de hoy ofrece la red.

En este caso un aspecto de ámbito médico tan delicado y novedoso como el diagnóstico automatizado es ya una realidad en multitud de portales que llegan a ser integrados por las grandes compañías de la información. En este proyecto se ha optado por integrar módulos que sirven de soporte a cuestiones relativas a los tumores de partes blandas. Una herramienta así identifica alteraciones que pueden presentar una apariencia engañosa y que al ser poco frecuentes en la población, puede servir de ayuda sobre todo a los radiólogos nóveles.

Al ser online estamos poniendo el potencial de aplicaciones médicas al servicio de expertos en distintas ubicaciones. Siendo ésta una aplicación abierta al mundo con la comodidad de no tener que instalarla y con la finalidad de proporcionar un soporte informativo en temas delicados a quien lo necesite.

2. Curiam. Sistema de ayuda a la decisión médica

Curiam es un sistema genérico de ayuda a la toma de decisiones creado por el grupo de informática biomédica IBIME y destinado a servir de soporte en el diagnóstico en centros hospitalarios. La aplicación está formada por modelos predictivos que exponen una respuesta automatizada a distintas alteraciones de ámbito médico. Tratando problemas como la capacidad invasiva de tumores a través del análisis de las características más influyentes en su predicción.

Este sistema es una plataforma con una arquitectura escalable que integra diversos modelos que permiten el análisis de tareas nuevas o ya existentes.

Dispone de módulos de clasificación entrenados a partir de una base de pacientes y con un alto porcentaje de aciertos en sus predicciones debido a múltiples algoritmos de inteligencia artificial y de reconocimiento de formas. Los cuáles tratan de relacionar la información de una base de pacientes, a través de distintos procesos algorítmicos, para extraer patrones que sirven como base de conocimiento. Esta información se relaciona con un nuevo caso específico que le introducimos al sistema, el cual es clasificado en el subconjunto más afín con él. De este modo se permite un soporte que dote de mayor exactitud a importantes y complejas decisiones médicas.

La aplicación almacena una base con resultados e imágenes de casos médicos. Permite la creación de nuevos análisis de los que extrae la probabilidad estadística a un conjunto de cuestiones médicas relacionadas con el caso que se pretenda analizar. Del mismo modo multitud de gráficos dan un soporte visual y permiten mostrar detalles internos del proceso de clasificación evaluando como es la relación del nuevo caso con la base de pacientes comparada.

Su interfaz permite visualizar múltiples análisis y perfila un diagrama de árbol que deriva de los datos que afectan al paciente. El cual a través de algoritmos se ramifica y muestra la influencia de opciones que ayudan al médico crear su propia valoración.

En la actualidad hay distintas versiones de la aplicación que tratan diversos temas que necesitan de una rápida y precisa evaluación médica de las cuales destacan las siguientes:

- Curiam-BT está diseñado para el diagnostico de tumores cerebrales a través de la espectroscopia de resonancia magnética.
- Curiam-PPD está optimizado para la rápida detección de depresión postparto.
- Curiam-STT es la versión, que al igual que el sistema online desarrollado, analiza distintas características en los tumores de partes blandas y cuyas pantallas vemos en la ilustración 4.1.

Ilustración 4.1. Capturas de pantalla de la aplicación Curiam.

3. Clasificadores TPB

Como ya se han introducido en el desarrollo de diversos puntos, el proyecto es una interfaz que integra un módulo de clasificación que sirve de soporte en la predicción de características en tumores de partes blandas. A este módulo se le pasan archivos XML con una amplia variedad de datos entre los que destaca el nombre de las variables de entrada y salida con la aplicación.

Los archivos XML se encargan de introducirle el conocimiento necesario al motor de clasificación, el cual en función de dicho archivo, clasificará los parámetros que le proporcionará la interfaz gráfica según los datos entrados por el usuario.

Disponemos de 2 tipos de archivos XML llamados *method.xml* ubicados dentro de los directorios 3_TPB_BM_knn_001 y 22_TPB_VCF_knn_001. El primer de ellos se encarga de clasificar un tumor de partes blandas en benigno o maligno y el otro en tipo de tumor vascular, quiste o grasa. La librería devuelve un porcentaje de probabilidad para cada caso utilizando el algoritmo *k-nn* o k vecinos más próximos en estos archivos.

De forma generalizada, este algoritmo consiste en una base de pacientes que relacionamos con un su tipo de tumor. Éstos sirven para entrenar el sistema. Con esta información cuando se analiza un nuevo paciente se calcula la distancia entre los k prototipos más próximos siendo éstos un conjunto de coordenadas formadas por los parámetros de entrada. Al nuevo paciente le asignamos el tipo de tumor que predomina entre sus k pacientes más cercanos.

Con esta forma de proceder evidentemente más evolucionada y con sus propias peculiaridades se ha diseñado la parte del módulo que devuelve un tipo de clasificación para los XML que disponemos. No obstante la librería integra otros algoritmos de clasificación como son ANN o redes neuronales artificiales y SVM conocidos como maquinas de soporte vectorial los cuales son igualmente algoritmos de aprendizaje.

De estos algoritmos sabemos que han sido utilizados para el diagnóstico de diversas cuestiones clínicas como el melanoma, la malignidad de tumores ovarios, macrocalcificaciones en mamografías digitales, tumores de mama y cáncer de cuello uterino entre otros.

Para la realización de los clasificadores y en concreto para el 3_TPB_BM_knn_001 se ha utilizado una base de datos multicéntrica con información de 630 pacientes, de los que el 62% tenían un tumor de naturaleza benigna frente al 38% con tumores malignos. El sistema fue entrenado con 302 casos y otros 128 sirvieron para evaluar su clasificación sacando una exactitud de 88-92% en sus predicciones[17].

Los análisis que podemos realizar en la aplicación nos extraen diversas características que permiten identificar la gravedad y la naturaleza de los TPB. Estos parámetros los tenemos que introducir manualmente a partir de imágenes de resonancia magnética con indicios tumorales. Todos ellos están descritos en el Anexo 1 de la memoria.

4. Integración de clasificadores en servicios web

El módulo de clasificación está compuesto por una librería en Java integrada en nuestra aplicación bajo el nombre de *BMGClassificationEngine*. En un principio se recreó su funcionamiento sobre Tomcat a través de un servlet que llamaba al motor de clasificación para mostrar estadísticas vía web.

La integración de la herramienta Tomcat junto a su configuración es la descrita en el capítulo 2 donde llegábamos a visualizar el navegador que el Eclipse trae incorporado para el desarrollo de aplicaciones web mostrándonos el proyecto de testear webs vía online. El proyecto utiliza una tecnología JSP o Java Server Pages orientada a la creación de páginas web en Java.

Para que funcione el proyecto, además de la instalación y configuración del Tomcat, tenemos que asegurarnos de que hemos incluido las librerías *Jdom* y *BMGClassificationEngine* en el ClassPath. También haberle copiado la carpeta con los clasificadores en la ruta que nos marca Eclipse al ejecutar la aplicación. Éstos están compuestos por directorios con el nombre del clasificador los cuales contienen el archivo *method.xml* con el nombre de parámetros, de variables devueltas y de datos intermedios utilizados para la clasificación. También contienen el archivo *description.xml* que almacena una descripción del análisis.

En este proyecto se crea una ramificación de ficheros parecida a la de GWT en la que encontramos la parte con archivos JSP, que hacen una llamada a un servlet pasándole un formulario de parámetros médicos. Éste lo recibe dicho servlet en Java a través de la función *doPost* cuyos atributos son la petición que le pasamos del formulario y la respuesta que le enviamos a otro archivo JSP. De la petición se leen los parámetros que recibirá el motor de clasificación y se almacenan en clases *Bean*.

Posteriormente se define la clase Input la cual leerá la librería y se le pasa cada parámetro indicándole mediante clases *ClassificationData* el nombre del argumento y el valor del parámetro leído de la clase *bean*. Se define un cliente con la clase *LocalClassificationEngineClient* al que se le pasan los inputs junto al nombre del clasificador asignado. Éste hace el estudio, extrae los resultados en formato XML y seguidamente se pasan a un String.

A continuación con la ayuda de la librería *Jdom*, se define la clase *SAXBuilder* que lee clase *Stringreader* a la que se le pasa el String en formato XML y devuelve una clase *Document* cuyos métodos *getRootElement* y *getChild* van leyendo los valores de las etiquetas según la jerarquía que guardan en el XML.

Una vez leídos se le pasan al parámetro respuesta de la función *doPost* el cual será enviado a un archivo JSP cuyo nombre le pasamos mediante el método *RequestDispatcher* del parámetro respuesta. En la ilustración 4.2 vemos el Eclipse con los parámetros de entrada ya cambiados y adaptados a los clasificadores de TPB.

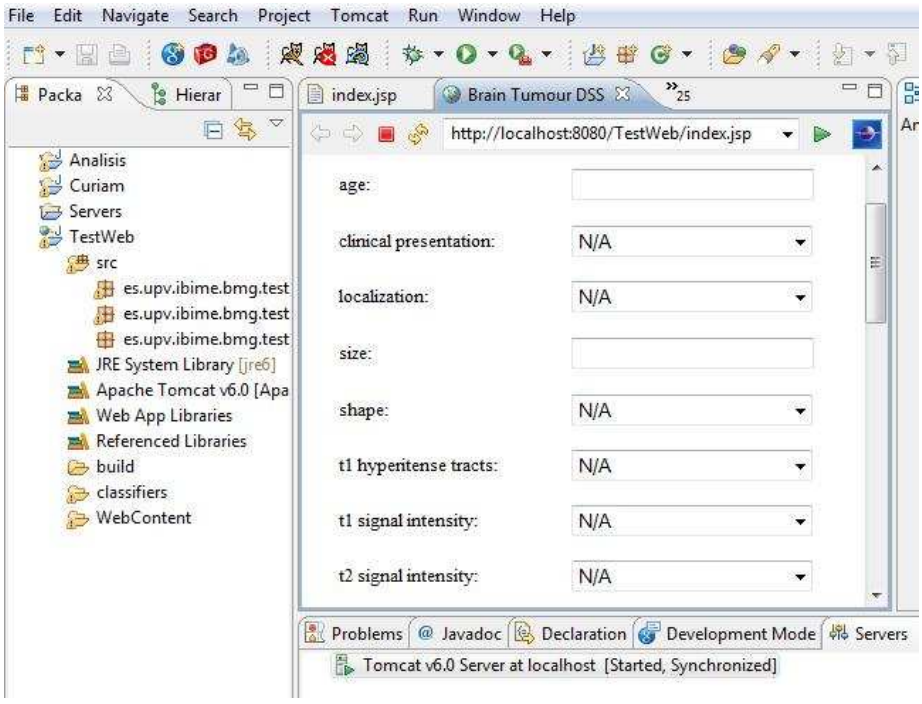

Ilustración 4.2: Eclipse con Tomcat configurado con los clasificadores de TPB y mostrando los parámetros de entrada.

En la ilustración 4.3 se observan los resultados de la clasificación en dicho proyecto.

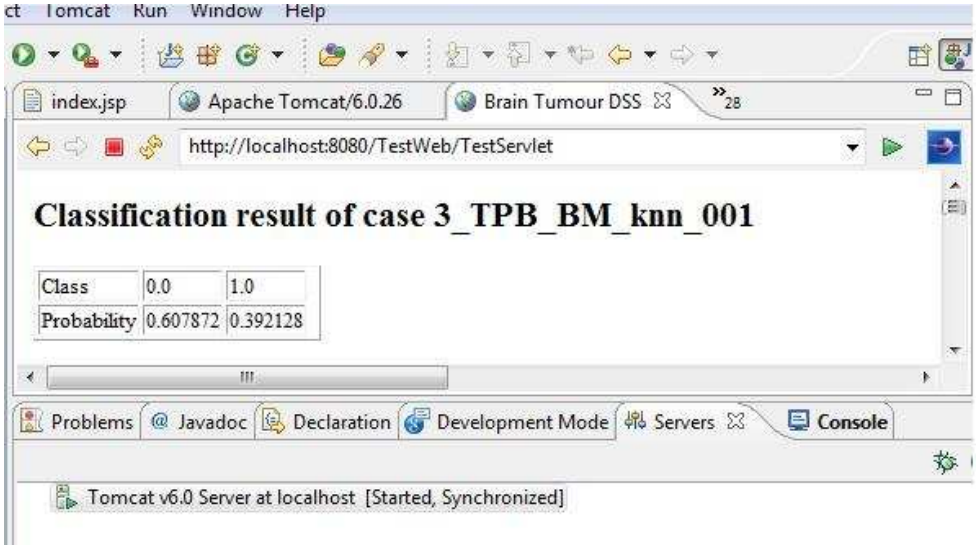

Ilustración 4.3: El proyecto con Tomcat mostrando resultados de clasificación.

Una vez comprobado su correcto funcionamiento se ha procedido a integrar los clasificadores en el proyecto GWT en el que habíamos empezado a crear la interfaz gráfica.

En un principio puesto que GWT sólo admite librerías Java en la parte servidora, le copiamos el interior de la función *doPost* del proyecto con Tomcat dentro de dicha parte y recreamos las partes síncrona y asíncrona en el cliente con la ayuda de la funcionalidad de autocompletado de Eclipse.

Posteriormente, le agregamos las librerías *BMGClassificationEngine* y *Jdom* en el Java build path. Si llegados a este punto ejecutamos el proyecto lo más seguro es que el Eclipse nos informe con los errores *ClassNotFoundException* y *NoClassDefFoundError*. Éstos se solucionan copiando la librería correspondiente dentro del directorio war / WEB-INF en las carpetas clases/ y lib/ respectivamente según el tipo de error.

A continuación debemos de agregarle al proyecto GWT, el fichero classification.propieties. En él se le pasan como parámetros a la librería de clasificación el nombre de los archivos XML necesarios para el análisis y el directorio donde los clasificadores están ubicados.

Llegados a este punto, la librería ya está bien enlazada pero Eclipse no nos permite la ejecución denegándonos el acceso ya que no tenemos permitido modificar un grupo de hilos. Estos hilos de ejecución concurrente, conforman la librería de clasificación y debemos desactivar el *App Engine de Google* ya que, aunque intentemos asignarle los permisos necesarios, GWT no permite trabajar con dichos hilos. Para desactivarlo, pulsamos el botón derecho sobre el proyecto, seleccionamos Google / App Engine Settings… y desmarcamos Use Google App Engine. Con lo cual ya nos muestra los resultados de clasificación si le insertamos valores correctos en los parámetros de entrada tal y como vemos en la ilustración 4.4.

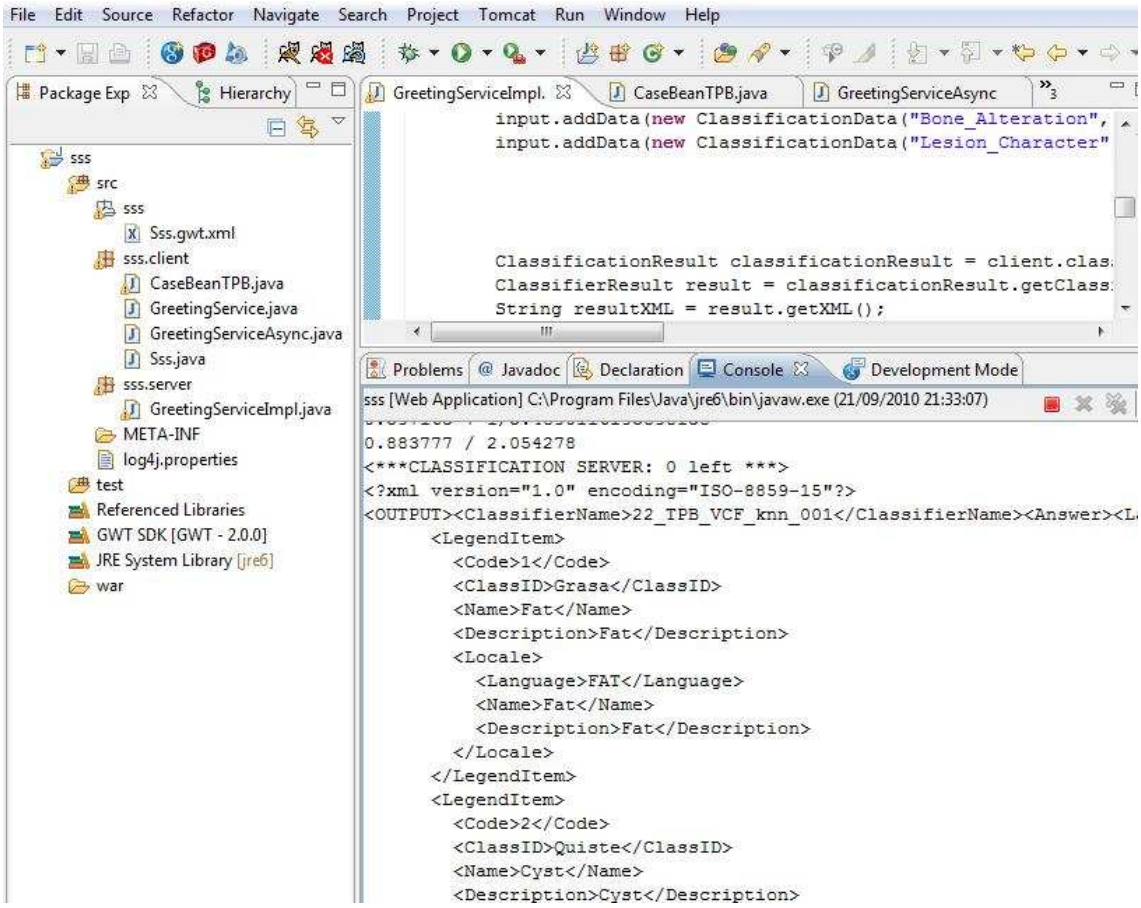

Ilustración 4.4: Proyecto GWT con el módulo integrado y mostrando resultados de clasificación.

Una vez llegados a este punto ya tenemos los clasificadores integrados. Nos faltaría unirlos a la interfaz gráfica para que a través de los Widgets de GWT podamos añadirle los parámetros de entrada y el servicio pueda mostrarnos las gráficas de las librerías asociadas.

Por lo tanto, diseñamos el panel de la interfaz de análisis creando una función (en nuestro caso llamada *solapa* y detalladamente descrita en el apartado 3.6) que se encarga de mostrar todos los elementos de un caso médico.

Los parámetros de entrada los insertamos dentro de *ListBoxs* y cajas de texto que, al pulsarle un botón de análisis, pasarían al servidor. Con estos datos el servidor haría la llamada al clasificador que le acabamos de integrar y éste le devolvería al cliente los datos que se observan en la imagen anterior.

Los resultados en la parte cliente son leídos por las clases *BasicChartExample* y *GridExample*, tal y como vemos en la imagen 4.5, a través de un constructor que le creamos en las mismas donde le pasamos una clase *bean* con los resultados de salida. Dentro de estas clases sólo queda vincular los datos en las salidas oportunas haciendo pequeñas modificaciones con ellos, bien sea dándoles el formato de decimales adecuado o implementándoles el cálculo del resultado de clasificación más alto.

Seguidamente debemos de asegurarnos de que tenemos bien enlazados, tanto el *ScrollPanel* con los *Widgets* de entrada, como los componentes de salida en una página desplegable del *StackPanel* o panel de acordeón. Los resultados del análisis los guardaríamos en ficheros XML dentro de los archivos de la cuenta del usuario para que quedara constancia de ellos aun cuando éste ya ha cerrado sesión.

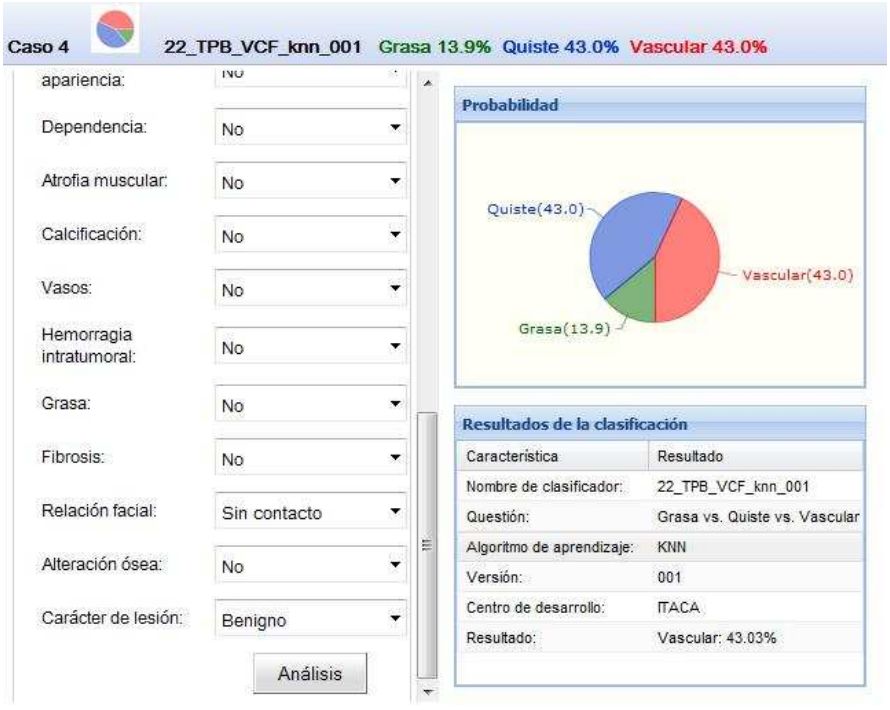

Ilustración 4.5: Proyecto GWT con los Widgets representando el aspecto definivo.

Capítulo 5: API Google Health

1. Introducción

Como ya se ha comentado anteriormente Google Health es la plataforma de la compañía Google que le permite al usuario introducir voluntariamente su Historia Clínica y gestionarla. También se ha comentado la disponibilidad que presta a aplicaciones de terceros al permitirles acceder a los datos clínicos de un paciente para realizar un estudio que proporcione información que beneficie al usuario. Con esta intención se pretendió vincular este proyecto a Google Health no obstante, como de momento sólo prestan estos servicios en EEUU, se anula dicha funcionalidad. Sin embargo aquí describiremos el proceso de su inclusión como parte de los puntos a tratar en el proyecto.

Como en todas las API de Google, esta compañía ofrece una extensa descripción de su servicio en un portal habilitado para servir de ayuda a los desarrolladores que necesiten hacer uso de ella. La API[10] nos informa de que los servicios que se indexen pueden leer e incluso actualizar y borrar datos si los gestionamos desde una aplicación de escritorio.

2. Requerimientos previos

Para empezar debemos de aceptar los términos y condiciones que se requieren al vincular Google Health junto a la aceptación de dos licencias que exigen el correcto uso de los datos prestados, esta información se envía a través de un formulario en que se piden un conjunto de datos personales y de la compañía.

A continuación debemos de enviar otro formulario en el que describamos la aplicación que estamos llevando a cabo, el tipo de datos que manipulamos, número de pacientes y de registros que gestionamos junto a datos personales y de la empresa asociada nuevamente.

Entonces es cuando Google nos contesta nuestra petición dándonos acceso como en otras librerías o denegándonos nuestra solicitud. Google sigue informándonos en la API sobre el uso protegido de los datos que nos proporciona el usuario. No venderlos, crear una política de seguridad que el usuario acepte, si son con fines publicitarios o de investigación obtener el consentimiento y una larga lista de restricciones. También nos pone al día de los principios de la compañía y de directrices sobre las divulgaciones que por ley debemos de establecer en nuestra política. La política debe principalmente exponer que datos recopilamos, como los almacenamos y que datos está compartiendo con terceros.

Otras restricciones en el diseño son poner el nombre de la institución, el logotipo de Google Health, destacar beneficios, disponer de un enlace para volver a Google Health junto a permitir vincular y desvincular perfil.

Si hemos seguido los patrones anteriores de diseño con las características de Google Health, debemos de enviar dos extensos formularios donde, a modo de cuestionario, se deben de contestar y verificar todas las exigencias de la compañía.

A continuación si lo que estamos desarrollando es una aplicación de servicios de terceros, se debe de enviar otro formulario con datos personales del representante del servicio, de la compañía y describiendo la aplicación. Este formulario llamado *Google Health's inquiry form* fue con el que se recibió la negativa por parte de la compañía al tratar de vincular nuestra aplicación informándonos de que el servicio sólo queda disponible para historiales de EEUU y como consecuencia para aplicaciones de centros estadounidenses interesados en los mismos.

Si seguimos la API encontramos otro formulario más en el que a través de múltiples cuestiones sobre nuestra aplicación de terceros se pretende hacer una revisión sobre si estamos siguiendo los requerimientos al pie de la letra. Éstos son bastante útiles para recordarnos peculiaridades como las dimensiones del logotipo de Google Health, el dominio donde tenemos ubicada la web, las exigencias legales… y así evitar que nos saltemos cualquier detalle requerido.

3. Recomendaciones en el desarrollo de aplicaciones

Se proponen una serie de recomendaciones que el desarrollador puede utilizar en la parte de integración de su aplicación. Muchas de ellas apelan al sentido común con lo cual las obviaremos, la cuales básicamente pretenden que en todo momento se mantenga al usuario informado de forma transparente y clara en el proceso de vinculación y desvinculación.

Respecto a su conexión se pretende que pueda desvincularse en cualquier momento, tener acceso a su estado e incluso ver cuando finaliza su sesión.

Al usuario se le proporciona un *token* cuando trata de acceder a la aplicación, el cual sirve para comunicarse con el servicio web vinculado a Google Health. Si el usuario se desvincula, Google recomienda eliminar dicho *token*, ya que sólo permite una señal activa por usuario. No obstante la plataforma de salud reserva un máximo de 5 *tokens* para cada perfil.

Otro aspecto importante es que las actualizaciones del historial del usuario pueden darse de diversas formas, mediante activadores a Google, al hacer clic en un vínculo de nuestra aplicación e incluso cada cierto tiempo. El contenido se requiere codificado en los campos apropiados CCR. Éste protocolo junto a la generación de tokens se explicarán más adelante con más detalle.

Una vez llegados a este punto y habiendo sido aceptados por Google, obtendríamos acceso al *Sandbox h9*. Ésta es una plataforma de aprendizaje en la que podemos simular el enlace con Google Health con menos restricciones. Funciona igual que la *healt* pero las diferencias son que no necesitamos un dominio sino que podemos interactuar directamente desde *localhost*, ni necesitamos que las peticiones estén firmadas. Para cualquier interacción necesaria, se substituye *health* por *h9* en la URL y a partir de ahí ya podemos interaccionar con los registros de los perfiles como explicaremos en los siguientes apartados.

4. Protocolo de enlace e intercomunicación

Aunque en este apartado se explican aspectos básicos del protocolo de enlace para familiarizarnos con su uso, en el siguiente ya se especifica una guía más metódica para integrar nuestra aplicación paso a paso.

Hay tres protocolos de interactuación con Google Health. Dos de ellos, *AuthSub* y *OAuth,* destinados a aplicaciones web y *ClientLogin* a la intercomunicación con aplicaciones de escritorio. Los tres sirven igualmente para acceder al perfil del usuario con lo cual explicaremos *AuthSub* por ser el más utilizado.

Para integrar dicho protocolo debemos saber que una entrada de registro *feed* representa una noticia o aviso que se puede recordar al usuario acerca de resultados de laboratorio o de recetas médicas entre otros. Sin embargo un perfil contiene todos los datos de un usuario. A éstos se puede acceder con una petición HTTP GET a las siguientes URL respectivamente.

```
https://www.google.com/health/feeds/register/default

https://www.google.com/health/feeds/profile/default
```

Esto es posible siempre que se disponga de los permisos de acceso según la plataforma *health* o *h9*.

En la misma API hay una aplicación de terceros (una versión funcionando en *h9* y otra en *health*) llamada *Monte Tabor* donde podemos crear un *token*, darle permisos de lectura y opcionalmente de escritura. El servicio también nos redirige a la pantalla de autorización de Google Health y nos vincula el perfil al confirmarla. Ya dentro de la aplicación nos permite publicar avisos mostrándonos su formato en XML a partir de avisos escritos en modo texto (véase la ilustración 5.1) y recuperar un perfil mostrándonos su código XML entre otras funcionalidades útiles.

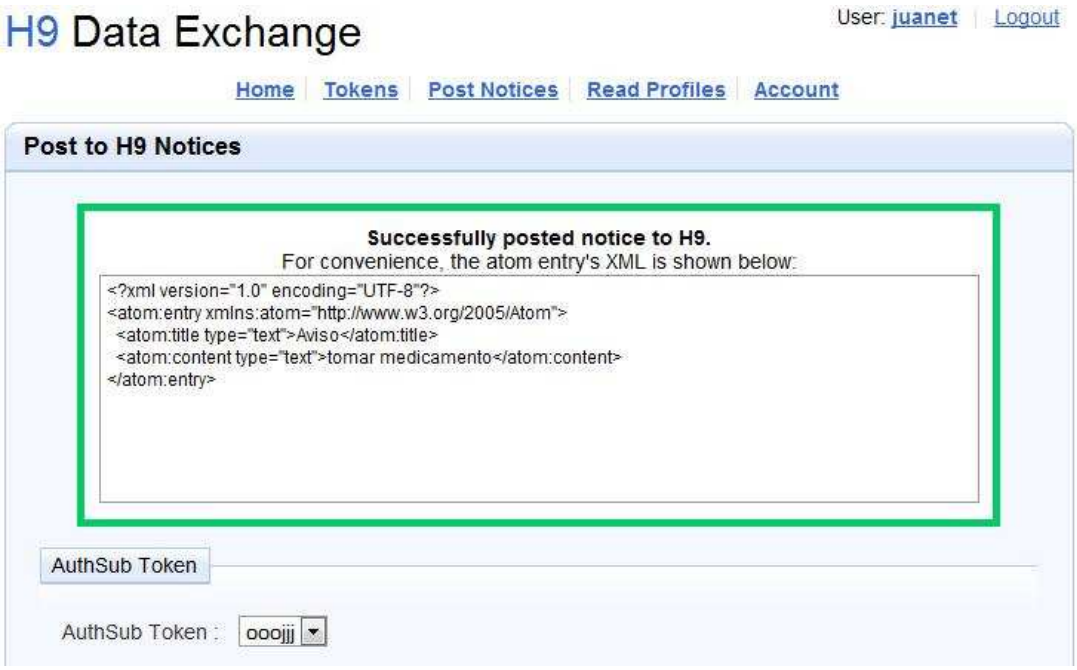

Ilustración 5.1: Aplicación de prueba enlazada en Google Health con funcionalidades útiles para desarrolladores.

Algunas de las posibles llamadas que podemos hacer a Google Health es comprobar si un *token* sigue siendo válido para la aplicación al hacer una petición GET a *AuthSubTokenInfo.*

```
GET /accounts/AuthSubTokenInfo HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: AuthSub token="yourAuthToken"
    data="GET https://www.google.com/accounts/AuthSubTokenInfo
1148503696 15948652339726849410" sig="MCwCFrV93K4agg=="
sigalg="rsa-sha1"
Host: https://www.google.com
```

Y una respuesta positiva desde la misma sería:

```
HTTP/1.1 200 OK

Target=http://www.yourwebapp.com
Scope=https://www.google.com/health/feeds/
Secure=true
```

Por otra parte como desde Google Health se pueden crear varios perfiles y en cada uno asociar las aplicaciones de terceros de forma independiente, un mismo usuario puede requerir diversos tokens para acceder a la misma

aplicación desde distintos perfiles. Con lo cual, para diferenciarlos se puede hacer la siguiente petición que asignaría una descripción al token:

```
POST /health/AuthenticationTokenSettings HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: AuthSub token="yourAuthToken"
    data="POST
https://www.google.com/health/AuthenticationTokenSettings
1148503696 15948652339726849410" sig="MCwCFrV93K4agg=="
sigalg="rsa-sha1"
Host: https://www.google.com

token_desc=Alfonso’s Count
```

Devolviéndonos la siguiente respuesta:

```
HTTP/1.1 200 OK

OK Alfonso’s Count
```

Además Google Health admite los siguientes parámetros en sus peticiones URL.

- **q:** alberga texto en forma de cadena para una consulta.
- **start-index y max-result:** acotan el índice de resultados.
- **published-min, published-max:** establecen los límites de fecha de publicación.
- **updated-min, updated-max:** acotan la fecha de actualización de entradas.
- **category:** filtra la categoría de entradas.
- **digest=true**, devuelve todos los datos en un mismo documento.
- **grouped=true**, hace un recuento de las entradas por grupo (las siguientes instrucciones requieren este parámetro a true).
- **max-results-group:** indica el número máximo de grupos que se recuperarán.
- **max-results-in-group:** especifica el número máximo de registros que se recuperarán en cada grupo.

- **start-index-group:** devuelve los elementos de grupos superiores al indicado.

- **start-in-index-group:** devuelve los elementos de cada grupo superiores al indicado.

*E*n la ilustración 5.2 se muestran los campos CCR que forman un perfil del usuario:

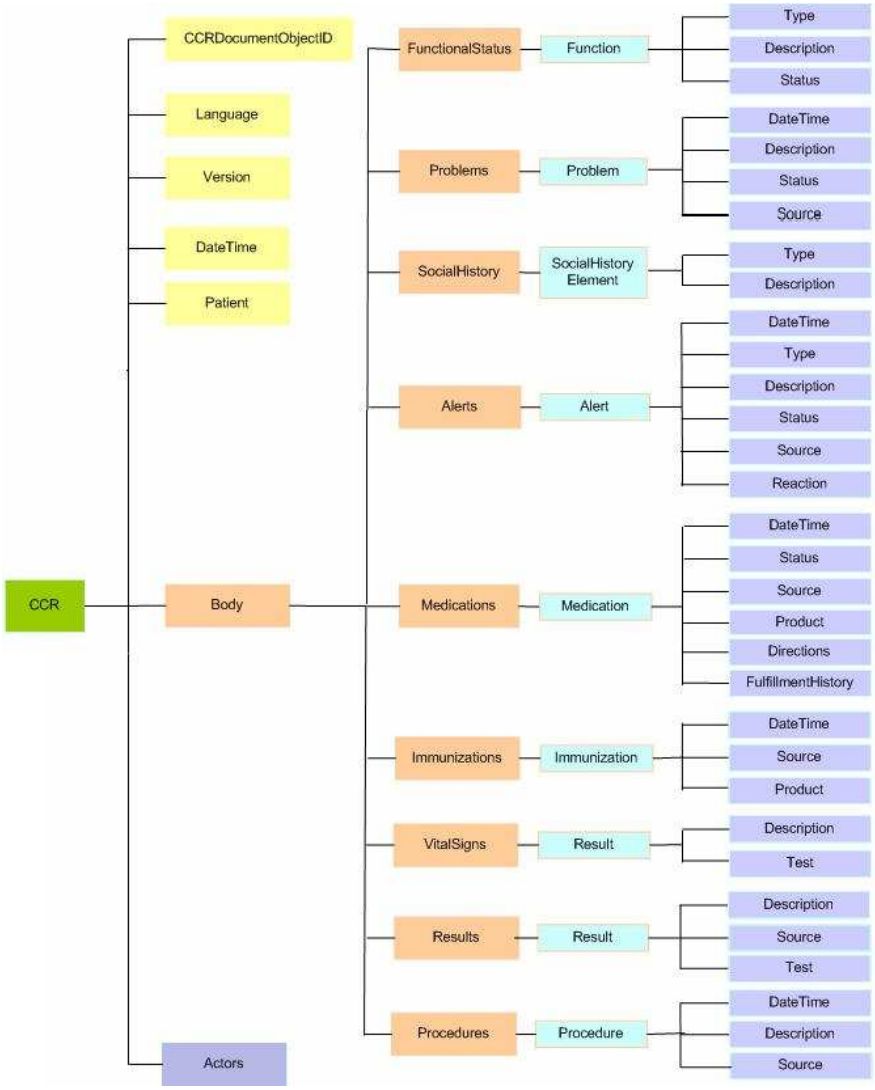

Ilustración 3.2: Campos CCR con el perfil del usuario

Basándonos en esos datos podemos hacer consultas como las que se muestran a continuación:

- Todos los datos de medicamentos del usuario:

```
GET /-/{http://schemas.google.com/health/ccr}medication
```

- Las 10 primeras entradas para el medicamento Ibuprofeno:

```
GET /-
/medication/{http://schemas.google.com/health/item}Ibuprofeno?max-
results=10
```

- Todas las condiciones que son dolor de pantorrilla o tos:

```
GET /-/condition/{http://schemas.google.com/health/item}Calf+Pain |
{http://schemas.google.com/health/item}Cough
```

Todas las peticiones tienen que tener su URL codificado y en los mensajes se puede especificar código HTML mediante *type="html"* en las etiquetas *<entry>* y *<content>.* Aunque las siguientes etiquetas no son compatibles

<html>, <head>, <body>, <script>, <style>, <frame> y <iframe>.

Ahora siguiendo con las llamadas a los protocolos y teniendo en cuenta que el AuthSub sólo admite insertar datos en el perfil del cliente y leerlos, procedemos a su lectura.

Si hay un *feed* llamado */myfeed* y hacemos la siguiente petición HTTP GET:

```
GET / myFeed
```

El servidor nos responde con una respuesta XML con metadatos del título y nombre de autor entre otros:

```
200 OK

<?xml version='1.0' encoding='utf-8'?>
<feed xmlns='http://www.w3.org/2005/Atom'
    xmlns:gd='http://schemas.google.com/g/2005'
    gd:etag='W/"C0QBRXcycSp7ImA9WxRVFUk."'>
  <title>Foo</title>
  <updated>2006-01-23T16:25:00-08:00</updated>
  <id>http://www.example.com/myFeed</id>
  <author>
    <name>Jo March</name>
  </author>
  <link href='/myFeed' rel='self'/>
</feed>
```

Por otra parte para hacer una inserción se envía un POST con la representación XML de la nueva entrada:

```
POST /myFeed

<?xml version='1.0' encoding='utf-8'?>
<entry xmlns='http://www.w3.org/2005/Atom'>
  <author>
    <name>Elizabeth Bennet</name>
    <email>liz@gmail.com</email>
  </author>
  <title type='text'>Entry 1</title>
  <content type='text'>This is my entry</content>
</entry>
```

Y el servidor nos devuelve un XML con los datos insertados:

```
201 CREATED

<?xml version='1.0' encoding='utf-8'?>
<entry xmlns='http://www.w3.org/2005/Atom'
    xmlns:gd='http://schemas.google.com/g/2005'
    gd:etag='"CUUEQX47eCp7ImA9WxRVEkQ."'>
  <id>http://www.example.com/id/1</id>
  <link rel='edit' href='http://example.com/myFeed/1/1/'/>
  <updated>2006-01-23T16:26:03-08:00</updated>
  <author>
    <name>Elizabeth Bennet</name>
    <email>liz@gmail.com</email>
  </author>
  <title type='text'>Entry 1</title>
  <content type='text'>This is my entry</content>
</entry>
```

Procediendo del mismo modo el servidor nos devolvería el XML para las actualizaciones y el borrado de campos pero éstas en el caso de que utilizáramos el protocolo *ClientLogin* desde una aplicación de escritorio. Todo el conjunto de peticiones están en librerías disponibles en la API donde ya las tenemos programadas en diferentes lenguajes, entre ellos Java.

5. Integración del servicio web

Dejando ya aparte la plataforma *h9*, si utilizamos la *health* sí que debemos de pedir autorización a los usuarios para poder acceder a sus datos. La URL donde redirigimos al usuario para que Google nos lo vincule y podamos obtener su *token* es la siguiente:

```
https://www.google.com/health/authsub?scope=https%3A%2F%2Fwww.go
ogle.com%2Fhealth%2Ffeeds%2F&session=1&secure=1&permission=1&nex
t=http%3A%2F%2Fwww.example.com%2Fwelcome
```

Donde la URL llevaría el usuario a la página de confirmación donde se le pregunta por el perfil que quiere enlazar con nuestra aplicación:

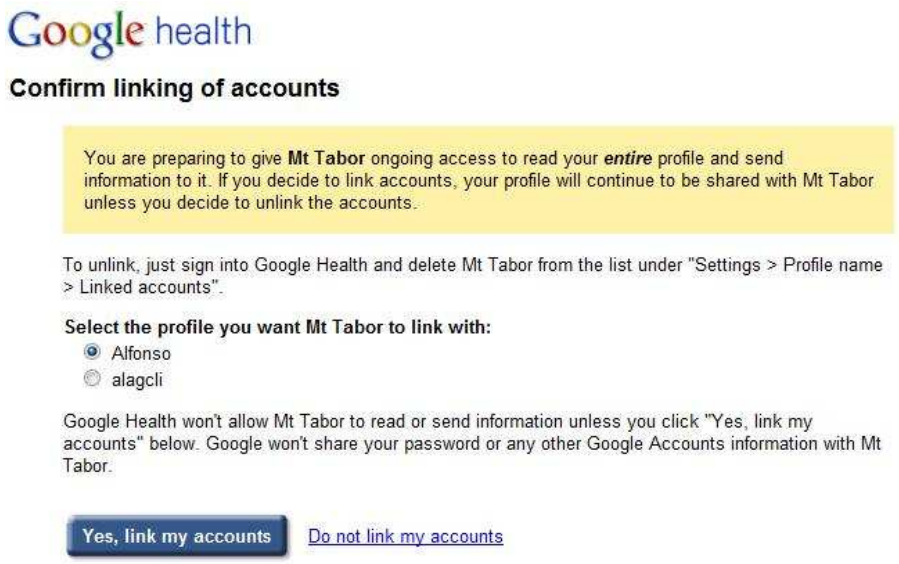

Ilustración 5.3: Interfaz de Google que vincula el perfil del paciente a una aplicación de terceros

A nivel interno, nuestra aplicación le pediría el *token* al centro de autorización con la llamada *AuthSub* que hemos visto, ésta le mostraría la anterior página al usuario, donde si no acepta el vínculo vuelve a Google y si lo acepta, entra dentro de nuestro servicio proporcionándonos el *token* con el cual podemos acceder a sus datos del perfil.

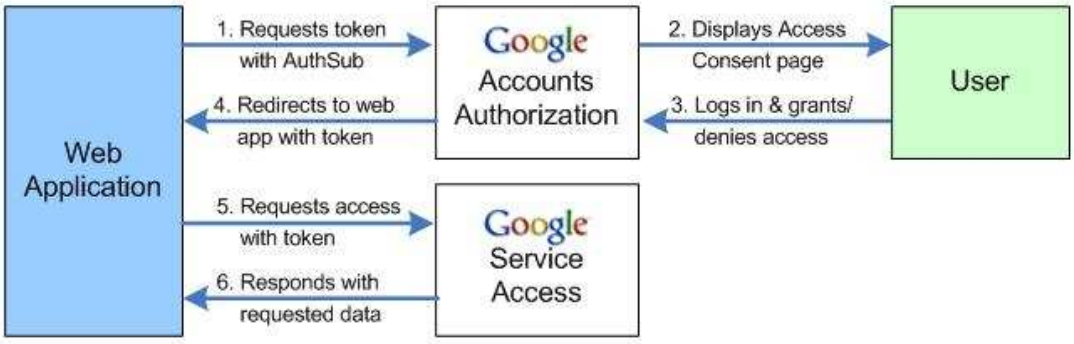

Ilustración 5.4: Esquema del proceso de vinculación del perfil de usuario junto al acceso de sus datos.

Con lo cual accedería en el punto 4 a nuestra aplicación redireccionándose a la web que le hemos dado en el campo *next* de la llamada del punto 1.

```
http://www.example.com/welcome?token=yourAuthToken
```

Como tenemos aceptada la conexión de forma segura con *segure=1* en el punto 1, ya podemos cambiar nuestro *token* por uno de sesión llamando a *AuthSubSessionToken*

```
GET /accounts/AuthSubSessionToken HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: AuthSub token="yourAuthToken"
    data="GET
https://www.google.com/accounts/AuthSubSessionToken 1148503696
15948652339726849410" sig="MCwCFrV93K4agg==" sigalg="rsa-sha1"
User-Agent: Java/1.5.0_06
Host: https://www.google.com
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
```

AutSubSessionToken respondería:

```
Token=yourSessionToken
Expiration=20061004T123456Z
```

Aquí debemos de ignorar la expiración ya que estos *tokens* no caducan si no se indica explícitamente. El símbolo del *token* de sesión lo debemos usar en las siguientes llamadas.

```
GET /health/feeds/profile/default  HTTP/1.1
Host: https://www.google.com
Content-Type: application/atom+xml
Authorization: AuthSub token="yourSessionToken"
```

```
    data="GET
https://www.google.com/health/feeds/profile/default 1148503696
159486523397268494l0" sig="MCwCFrV93K4agg==" sigalg="rsa-sha1"
Connection: keep-alive
```

Hasta aquí se explica en protocolo *AuthSub*, *OAuth* hace lo mismo de forma parecida y *ClientLogin* también pero para programas de escritorio con lo cual ignoramos sus explicaciones ya que no nos aportan nada extra.

Un nuevo detalle es que como acaban de cambiar de interfaz en Google Health ahora hay que especificar que las llamadas son a la versión 2 de la misma con `GData-Version: 2` en la cabecera HTTP o `v=2` si la llamadas es desde parámetros GET.

Se puede enviar una petición GET para recuperar cada dato del perfil por separado, si además le añadimos el parámetro `digest=true` recuperaremos todos los datos del perfil en un mismo documento CCR.

```
https://www.google.com/health/feeds/profile/default?digest=true
```

La respuesta nos devolvería un extenso documento CCR con los datos del perfil del usuario.

Si lo que se desea es recuperar un perfil que ya se ha recuperado antes, podemos mejorar la eficiencia preguntándole a Google Health que nos devuelva el perfil si ha cambiado desde la última vez que lo hemos leído procediendo con la siguiente instrucción:

```
If-None-Match: W/"CEYFSXg4fSp7ImA9WxRVE00."
```

Se compara el *eTag* con el del perfil, si no ha cambiado nos devuelve *HTTP 304 Not Modified* y si lo ha hecho nos responde con el nuevo perfil del usuario.

Si lo que queremos es consultar datos específicos que coinciden con un determinado criterio, podemos agregarle parámetros como *updated-min* y *updated-max* para ver los medicamentos actualizados entre cierta fecha.

Por ejemplo para ver los modificados entre el 16 y el 24 de marzo del 2010 tendríamos la siguiente consulta:

```
GET /health/feeds/profile/default/-/medication/?updated-
min=2006-03-16T00:00:00&updated-max=2006-03-
24T23:59:59&digest=true HTTP/1.1
Content-Type: application/atom+xml
Authorization: AuthSub token="yourSessionToken"
    data="GET
```

```
https://www.google.com/health/feeds/profile/default/-
/medication?updated-min=2010-03-16T00:00:00&updated-max=2010-03-
24T23:59:59&digest=true 1148503696 15948652339726849410"
sig="MCwCFrV93K4agg==" sigalg="rsa-sha1"
Host: www.google.com
```

Como en *AuthSub* sólo podemos leer y crear nuevas entradas, para enviar un nuevo aviso. Enviaríamos un POST con la nueva entrada de la siguiente forma:

```
POST /health/feeds/register/default HTTP/1.1
Content-Type: application/atom+xml
Authorization: AuthSub token="GD32CMCL25aZ-v____8B"
    data="POST
https://www.google.com/health/feeds/register/default 1148503696
15948652339726849410" sig="MCwCFrV93K4agg==" sigalg="rsa-sha1"
Host: www.google.com

<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <title type="text">THE NOTICE SUBJECT LINE</title>
  <content type="text">THE NOTICE MESSAGE BODY</content>

  <ContinuityOfCareRecord xmlns="urn:astm-org:CCR">
    ... optional CCR data ...
  </ContinuityOfCareRecord>
</entry>
```

Capítulo 6: Conclusiones

1. Resumen del trabajo realizado

El grupo de Informática Biomédica Ibime ha desarrollado una aplicación llamada Curiam de ayuda al diagnóstico médico. Ésta en su interior contiene una librería de clasificación que en función del tipo de análisis elegido clasifica un caso médico en distintas alternativas. Con este módulo en su interior y con archivos de clasificación relativos a tumores de partes blandas, se ha realizado un servicio web de ayuda al diagnóstico.

En un principio, se ha procedido a comprobar el funcionamiento de un proyecto de prueba encargado de interactuar con la librería de clasificación y los clasificadores de TPB. Para ello se ha configurado el entorno Eclipse con Tomcat y se le ha integrado dicho proyecto.

Una vez enlazado, se ha procedido a desarrollar el proyecto con GWT instalado sobre Eclipse. En él se ha adaptado la librería con los clasificadores y se han configurado todas las peculiaridades necesarias para su funcionamiento.

Al comprobar la viabilidad del desarrollo se ha procedido a hacer un análisis de sus casos de uso, plantillas textuales y diagramas necesarios para concretar los requisitos y la interacción de sus componentes. Se ha creado una interfaz externa explicativa del servicio el cual permite identificar al usuario que se ha registrado. Una vez dentro, se ha creado un escenario en línea donde el usuario puede realizar análisis introduciendo parámetros que extrae de imágenes. Las imágines y los casos estudiados se guardan en la cuenta del usuario permitiéndole gestionarlos. Para el diseño de la parte visual se han incorporado Widgets de GWT y de la librería GXT. También se ha incorporado la librería GwtUpload para cargar imágenes en el servidor.

En su desarrollo se ha creado una BD formada por archivos XML almacenados en una jerarquía de directorios, se ha programado ateniendo a la interacción cliente - servidor propia de los servicios web.

Por último se ha intentado integrar el proyecto en Google Health, aunque no se ha podido por restricciones geográficas establecidas por Google, sin

embargo, se ha documentado el procedimiento de integración junto a las exigencias y sugerencias de la compañía.

2. Futuras líneas de desarrollo

Analizando la progresión del conjunto global de internet, desde las grandes compañías que compiten por hacerse con el control de la información médica, hasta los centros más modestos que tratan por hacerse un hueco en la red, el proyecto realizado puede ser la base de un proyecto online más sólido, evolucionado y adaptado a las necesidades de los usuarios preocupados por su salud.

La línea de desarrollo que más salta a la vista, es la posibilidad de enlazarla en Google Health, Microsoft HealthVault, IndivoHealth o cualquier otra plataforma destinada a la interacción de sistemas sanitarios con historiales clínicos siempre que estas plataformas hayan evolucionado para dar soporte a servicios de terceros en España.

Por otra parte observando las aplicaciones vinculadas a estas plataformas, apreciamos que muchas de ellas se encargan de prestar servicios que informen del riesgo de tener diabetes, de sufrir ataques o enfermedades cardiacas y de realizar todo tipo de diagnósticos o tests automatizados al leer los datos del perfil en Google Health. Con lo cual además de poder vincular la web con el clasificador de tumores de partes blandas, interesante por el análisis innovador que ofrece, ya que todavía no se encuentran de momento aplicaciones de este tipo al menos en Google Health, se pueden fácilmente integrar muchos otros tipos de clasificadores al igual que se puede perfeccionar la interfaz para permitir múltiples chequeos simultáneos, aportar sugerencias o consejos para la reducción del tumor e incluso realizar un seguimiento del mismo donde el usuario pueda comprobar su evolución junto a las tendencias que éste presenta. Este estudio se puede realizar en diversos aspectos ya que en la interfaz pueden integrarse nuevos clasificadores.

Con lo cual se estaría dando la posibilidad al usuario medio a realizarse una especie de chequeo en diferentes aspectos clave de su salud. Igualmente a hacerse un seguimiento y a obtener un mayor control e información sobre la misma, accediendo desde conocidas plataformas médicas en constante evolución como Google Health. Al mismo tiempo se permite a radiólogos e interesados a disponer de una herramienta alternativa en el diagnóstico vía online.

3. Observaciones finales

A continuación expondremos una serie de observaciones respecto al proyecto creado:

- Desde el punto de vista de los conocimientos obtenidos durante el desarrollo del proyecto me parece una gran aportación haber aprendido a realizar servicios web, los cuales no se tocan prácticamente durante la carrera, con uno de los lenguajes más estudiados en la misma. Más aun después de haber comprobado personalmente el gran esfuerzo que exige la adaptación de estos servicios a las peculiaridades de los sistemas al programarlos con lenguajes web convencionales.

- Posiblemente un aspecto negativo en GWT sería que al ser una herramienta relativamente nueva, no exista demasiada documentación con lo cual dificulta en ocasiones el avance. No obstante la facilidad pasmosa con la que se programa en Java junto al potencial del mismo seguro que propiciará su popularización.

- Respecto a las librerías visuales de internet y la capacidad de incluir gráficos, resulta curioso ver la cantidad de APIs que están apareciendo en internet tanto las aquí mencionadas como diversas otras APIs[18] y librerías cuya evolución ha sido constante durante el desarrollo de este proyecto abarcando gran cantidad de modelos gráficos.

- Otra observación relativa a la aplicación es que estamos poniendo al alcance de todo el mundo una herramienta de diagnóstico. Con lo cual estamos aportando un servicio alternativo que pueden contrastar profesionales que se enfrenten a este tipo de decisiones, al mismo tiempo que ofrecemos a un público más amplio como aprendices, e incluso en aplicaciones de salud y bienestar podría ofrecerse a pacientes afectados, familiares y cuidadores interesados y demás, la posibilidad de disponer de dicha herramienta para lo que necesite. Bien sea para el cuidado de su salud o como soporte informativo.

- De la API Google Health una observación importante es la abundancia de información que hay y su forma de esquematización. Aunque su extensión da muchos conocimientos, se echa en falta una guía rápida para vincular aplicaciones, bien esquematizada en la que siguiendo una lista de pasos podamos desarrollar nuestro objetivo sin perder gran cantidad de tiempo leyendo información superflua y pudiendo seguir

pasos correlativos como en el tutorial de GWT sin perdernos en multitud de contenidos que enlazan a otros tantos y así sucesivamente.

- La última observación importante es relativa a Google Health, este tipo de plataformas está evolucionando a un ritmo inimaginable, prueba de ello es que la semana de redacción de esta memoria han lanzado una nueva versión con multitud de detalles, estudios, gráficos y estadísticas… desde mi punto de vista creo que éste es el futuro de las aplicaciones médicas y no es difícil pensar que pronto se pueda enlazar cualquier aplicación.

Anexo 1: Guía del usuario

Curiam online es la modalidad en línea de la versión Curiam-STT, esta última es una aplicación desarrollada por el grupo de informática biomédica IBIME destinada a servir como soporte de ayuda al diagnóstico de tumores de partes blandas en centros clínicos.

La versión aquí expuesta pretende ofrecer igualmente un servicio de diagnóstico o en su caso informativo a un conjunto de usuarios más extenso, bien sean expertos que gusten de utilizar la aplicación, curiosos, interesados, aprendices, pacientes afectados y todo aquel que necesite hacer uso ella con una finalidad informativa al ofrecerle un chequeo alternativo en aspectos relacionados con tumores de partes blandas. Con esta intención se pretendía incorporar en la plataforma de salud Google Health como aplicación en el apartado de servicios de terceros. Ante la falta de cobertura internacional, se imposibilita de momento está vinculación quedando la interfaz hecha para tal propósito junto a la de proporcionar una experiencia positiva al usuario.

1. Interfaz externa

La interfaz externa sirve como presentación de la propia aplicación en la que se ha procurado darle el aspecto de servicio web médico. Dicho servicio contiene diversos apartados explicando su funcionalidad y descripción.

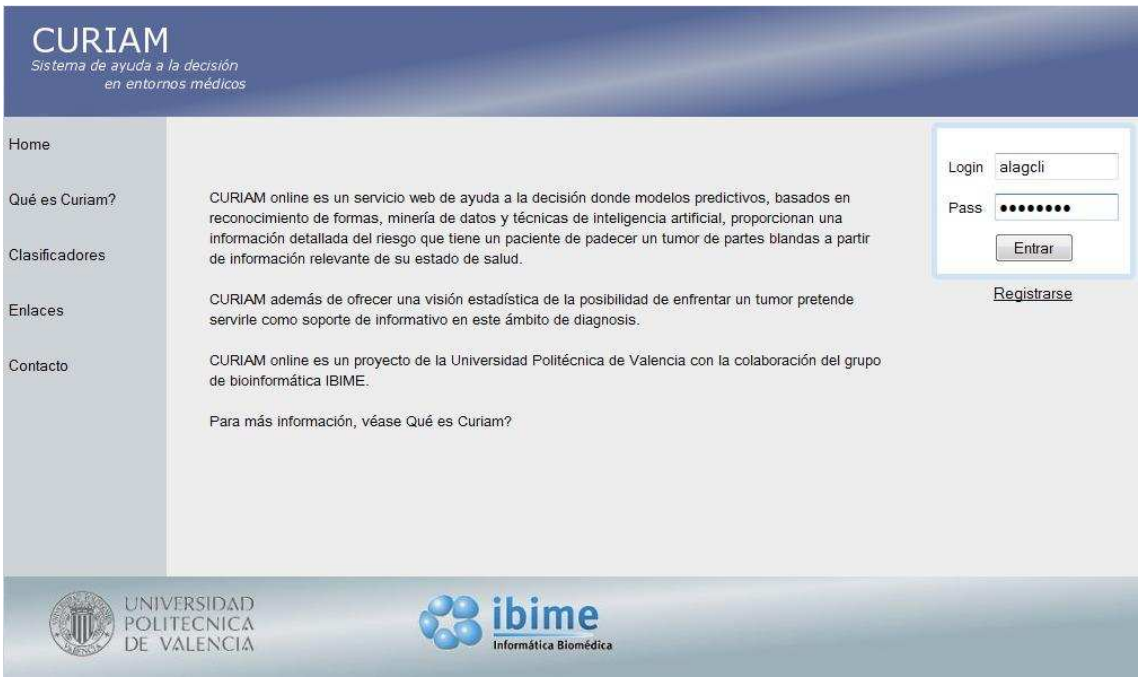

Ilustración 7.1: Interfaz principal del servicio web.

No obstante el principal motivo es la identificación del usuario si éste no ingresa desde una plataforma sanitaria como Google Health, ya que si lo hiciera se leerían sus datos directamente mostrándole la interfaz interna de análisis.

2. Gestión del perfil de usuario

Para acceder al interior de la aplicación simplemente necesitamos un pseudónimo que nos identifique, introducir una contraseña y confirmarla. Ni el *login* ni la contraseña pueden contener el carácter especial _ ni menos de 6 dígitos. En caso contrario se nos avisa del error mediante un cuadro de diálogo, al igual que hace otras comprobaciones como usuario existente, cuadro de texto vacío o contraseñas no idénticas.

Ilustración 7.2: Interfaz de registro del sistema.

3. Clasificadores y parámetros

Una vez dentro de la aplicación accedemos a un panel de análisis donde nos aparece un menú titulado nuevo caso con una serie de parámetros de entrada que debemos de indicarle para realizar dicho análisis. Estos parámetros debemos de extraerlos de imágenes de resonancia magnética que podemos cargar en la web y cuyo significado es el siguiente[12]:

- **Tipo de análisis:** Estudio que queremos realizar al elegir el clasificador adecuado el cual discierne entre benigno y maligno o entre vascular, quiste y grasa.

- **Edad:** Los años del paciente.

- **Presentación clínica:** Motivo de la consulta, ésta puede ser debido al peso, al dolor, al crecimiento, a síntomas neurológicos, a alteraciones de la piel o puede haber ausencia de síntomas.

- **Ubicación:** Compartimento anatómico donde se ubica el tumor. Las opciones son subcutáneo, intramuscular, intermuscular o intraarticular. Si se observan lesiones en varias secciones indicamos el compartimento donde se encuentra la parte más grande del tumor.

- **Tamaño:** El diámetro máximo redondeado en centímetros del tumor.

- **Forma:** La apariencia que perfila la alteración, ésta puede ser redonda, ovalada, lobulada, serpiginosa, fusiforme, cordón o irregular. Las lesiones serpiginosas son aquellas con estructuras tubulares en su interior y con una superficie externa polilobular (Ilustración 7.3).

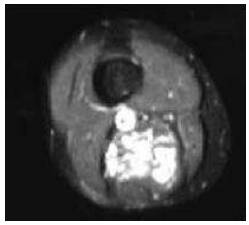

Ilustración 7.3: lesión serpinginosa.

- **Extensiones hiperintensas en T1:** zonas reticulares o rectilíneas con una alta intensidad de señal en el imágenes ponderadas en T1 sin supresión de la grasa (no, sí).

- **Señal de intensidad:** La intensidad de señal de los tumores representados en imágenes obtenidos con distintas ponderaciones (T1 y T2/STIR). El músculo se toma como intensidad de señal de referencia intermedia-baja en la mayoría secuencias (eliminando la grasa en las imágenes T1, T2, STIR). En T1 se dispone de cuatro opciones

(hipointensa, isointensa, hiperintensa y muy hiperintensa) y tres en las imágenes potenciadas en T2 (isointensa, ligeramente hiperintensas, e altamente hiperintensa).

- **Márgenes:** Son los perfiles del tumor, los cuales contienen la opción infiltrante, cuando la mayoría de los márgenes son borrosos o el tumor está acrecentado en cualquier momento al tejido periférico; las lesiones son parcialmente bien definidas, cuando perfilan un sector del margen con fronteras poco claras y no infiltrativa, cuando los perfiles se observan definidos sin ningún tipo de infiltración periférica.

- **Homogeneidad:** Diferentes áreas en las que varía la intensidad de señal en el tumor. Considerando la proporción y las diferencias de intensidad, hay cuatro categorías: muy homogénea, con sólo un componente de intensidad de la señal; homogéneo, con un área menor o igual al 25% del tumor con un ligero cambio en su intensidad de señal en comparación con el resto de la lesión; heterogéneos, en los que hay un área de entre 25% y 75% con diferentes intensidades de señales y muy heterogéneos, donde lesiones con más de un 75% de ellos con distintas componentes muestran grandes diferencias en su intensidad de señal.

- **Edema:** Zona circundante mal definida o halo, hipointensas en las imágenes potenciadas en T1 e hiperintensas en las imágenes ponderadas en STIR/T2 con una anchura de más de 5 mm (no, sí).

- **Multiplicidad:** Haberse dado en el paciente casos similares de TPB (no, sí).

- **Objetivo de la apariencia:** Si en la alteración del tejido había un centro interno bien definido y anillos concéntricos circundantes con diferentes intensidades de señal (no, sí).

- **Dependencia:** Si una estructura anatómica se identificó con una relación muy estrecha con la lesión y la lesión parece provenir de una estructura anatómica. Se tuvo especial cuidado de no incluir como dependencia de una relación de desplazamiento (ninguno, nervios, tendones y vasos).

- **Atrofia muscular:** Una reducción de la masa muscular relacionada con el tumor, con la agrandamiento de los planos grasos entre los fascículos musculares, y sobre todo si la extremidad contralateral estuvo presente para la comparación (no, sí).

- **Calcificación:** Presencia de áreas muy hipointensas en la imagen obtenida después de excluir vasos y hemosiderina. Sólo se consideran Flebolitos si una radiografía simple o un examen de TAC muestra una calcificación redondeada con un centro radiolúcido interior. Las opciones que se aportan son no, si y flebolitos.

- **Vasos:** Si vasos grandes forman parte del tumor (no, sí).

- **Hemorragia intratumoral:** Si en las imágenes se perciben zonas ponderadas heterogéneas, hiperintensa e hipointensa en T1 y STIR/T2 (no, sí).

- **Grasa intratumoral:** Si se observa en el tumor una zona de intensidad de señal igual a la de la vía subcutánea grasa en todas las secuencias. Las lesiones de grasa fueron clasificadas con respecto a la presencia de zonas hiperintensas en las imágenes ponderadas STIR/T2 con las alternativas: no, grasa sin zonas hiperintensas en T2-STIR y grasa con zonas hiperintensa en T2-STIR.

- **Fibrosis:** Presencia de áreas de muy baja intensidad de la señal en el tumor, sobre todo si son de forma irregular o de anillo (no, sí)

- **Relación facial:** Si las lesiones subcutáneas tienen relación facial. Las alternativas son sin contacto, contacto leve, contacto formando ángulos agudos, contacto con ángulos obtusos, inclusión facial o el origen facial del tumor.

- **Alteraciones del hueso:** Si algún hueso subyacente presenta una alteración identificada como remodelación ósea con reacción perióstica o destrucción del hueso, incluyendo cortical permeación ósea. Las opciones son: no, si con erosión de invasión, sí, con reformación o reacción.

- **Carácter de la lesión:** Capacidad de invasión o infiltración a lugares distantes del tumor primario que percibimos en la imagen. Con las opciones benigno y maligno.

Una vez indicados los parámetros realizamos el análisis. Al pulsar analizar nos aparece una gráfica con la probabilidad asociada a cada tipo de resultado según el estudio que estemos realizando. Podemos analizar la capacidad invasiva del tumor o si este se presenta de forma vascular, de quiste o de grasa. Al realizar un análisis los resultados quedan almacenados en nuestra cuenta.

Una vez realizado el primer análisis podemos analizar un nuevo caso al darle a dicho botón donde se nos vincularía una pestaña nueva con información vacía en el panel. También tenemos la opción de reanalizar el mismo caso cambiándole algún parámetro, lo cual nos puede servir si queremos probar distintas opciones en un mismo caso médico sin tener que introducir todos los datos de nuevo, ya que la selección de los parámetros también queda guardada.

Además de la gráfica asociada, el estudio nos muestra una tabla con datos del análisis realizado, del los cuales se extrae el nombre del clasificador utilizado, la cuestión que estamos planteándole al sistema, el tipo de algoritmo que se ha utilizado en la búsqueda de la solución óptima, la versión del módulo del archivo de clasificación, el centro que lo ha desarrollado y el resultado predominante del estudio tal y como vemos en la imagen.

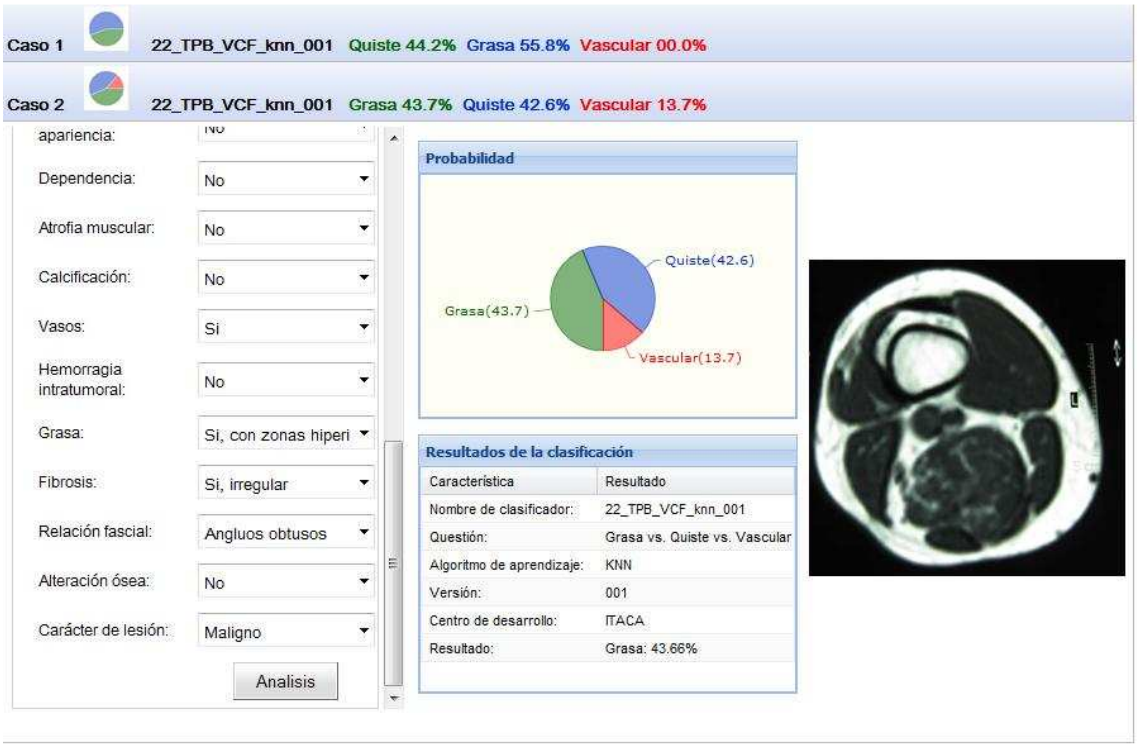

Ilustración 7.4: Análisis de un caso de TPB en la aplicación.

Como vemos en la ilustración 7.4, los casos van guardándose a modo de paneles conjuntos en cuya cabecera se describe el número de caso, una pequeña gráfica de resultados, el clasificador asociado y las distintas probabilidades de cada alternativa de resultados. Los paneles se van guardando a modo de acordeón permitiéndonos desplegar el que queramos al pulsarle la cabecera. Éste hecho junto a la posibilidad de reanalizar los casos tratan de proporcionarle flexibilidad al usuario en el manejo de la interfaz.

Si pulsamos en el panel superior el botón titulado *imágenes*, nos mostrará un cuadro de diálogo para la manipulación de las mismas. El cual permite subirlas al servidor a la vez que inmediatamente nos mostraría una miniatura de la misma en el momento que está ha subido al servidor. Las imágenes reducidas contienen asociado un botón x el cual elimina la imagen correspondiente del servidor, del panel de miniaturas y de casos analizados que se aprecia en la imagen 7.5.

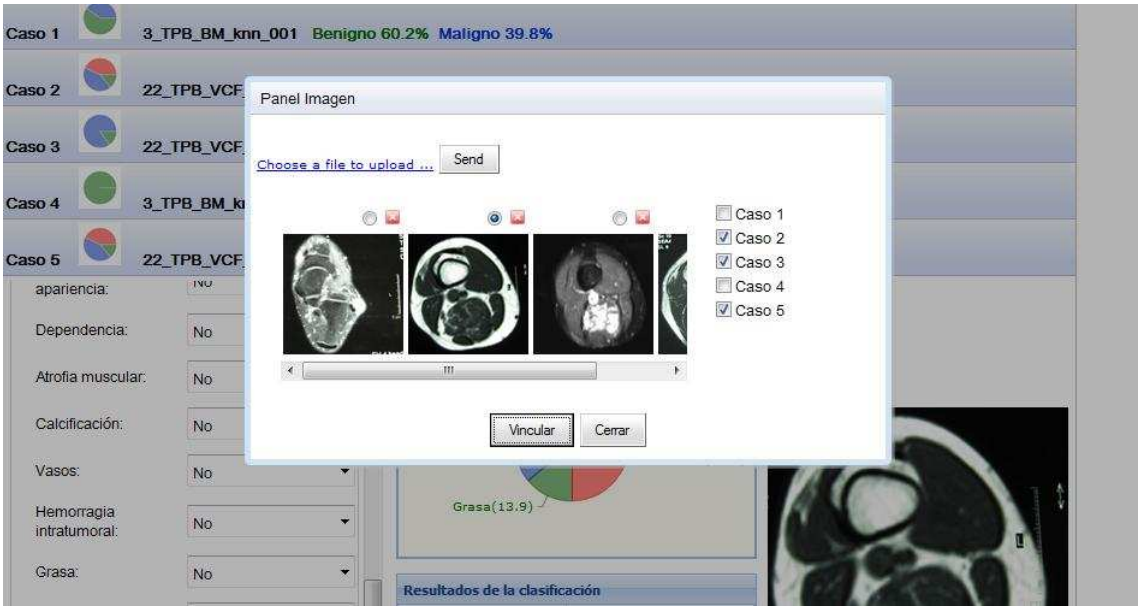

Ilustración 7.5: Panel de gestión de imágenes de la aplicación.

Las imágenes también salen asociadas a un *radiobutton* que permite seleccionar una imagen y vincularla a diversos casos de uso seleccionándolos y pulsando vincular. Tras esta acción se mostraría la imagen seleccionada en el panel principal en cada caso de uso que le hayamos relacionado.

Los siguientes botones del panel superior son *eliminar casos* que tras un mensaje de confirmación elimina todos los casos analizados, *cerrar sesión* que nos devuelve a la página principal y *eliminar cuenta*, que tras pedirnos que introduzcamos la contraseña a modo de confirmación, borraría nuestra cuenta con toda la información de registro, los casos analizados y las fotos cargadas.

Referencias bibliográficas

[1] Wikipedia (2010)

http://www.wikipedia.org

[2] Información corporativa – Visión general de la compañía

http://www.google.com/intl/es/corporate/

[3] Google Health beta (2010)

http://www.google.com/health

[4] Personal Home - HealthVault

http://www.healthvault.com/personal/index.aspx

[5] Dossia Personal Health Plataform

http://www.dossia.org/

[6] The Indivo Personally Controlled Health Record

http://www.indivohealth.org/

[7] Presentación de telemedicina (2010) Informática médica, upv

[8] Google Web Toolkit

http://code.google.com/intl/es-ES/webtoolkit/

[9] Aplicaciones Web con Google Web Toolkit (GWT)

http://www.juglar.org/uploads/presentaciones/gwt-juglar.pdf

[10] Api de datos de Google Health (2010)

http://code.google.com/intl/es/apis/health/

[11] Communicating with a server (2010)

http://code.google.com/intl/es-ES/webtoolkit/doc/1.6/DevGuideServerCommunication.html#DevGuideServerS

[12] Google Web Toolkit Showcase of Features (2010)

http://gwt.google.com/samples/Showcase/Showcase.html

[13] Widget Gallery (2008)

http://code.google.com/intl/es/docreader/#p=google-web-toolkit-doc-1-5&s=google-web-toolkit-doc-1-5&t=DevGuideWidgetGallery

[14] Ext GWT Rich Internet Application Framework for Google Web Toolkit (2006-2010)

http://www.extjs.com/products/gxt/

[15] Manolo Carrasco Moniño, GwtUpload & JspUpload: File Upload Progress with pure Javascript (2009)

http://code.google.com/p/gwtupload/

[16] Google Chart Tools / Image Charts (aka Visualization API) (2010)

http://code.google.com/intl/es/apis/chart/docs/gallery/chart_gall.html

[17] García Gómez JM, Benign / Malignant Classifier of Soft Tissue Tumors Using MRI pp. 25-40

[18] Google Chart Tools / Image Charts (aka Visualization API) (2010)

http://code.google.com/intl/es/apis/visualization/documentation/gallery.html