PROYECTO DE SISTEMAS INFORMÁTICOS
CURSO 2010/2011

Arquitectura genérica para telemedicina basada en servicios web y Google Health. Aplicación práctica en diabetes.

Autores:

Rafael Hernández de Diego

Francisco Martínez Jiménez

Javier Rocamora González

Director:

José Luís Risco Martín

Autorizaciones

Autorizamos a la Universidad Complutense de Madrid a utilizar y/o difundir con fines académicos y no comerciales, siempre mencionando expresamente a sus autores, tanto la propia memoria como el código, la documentación y/o el prototipo desarrollado.

Rafael Hernández de Diego

Francisco Martínez Jiménez

Javier Rocamora González

Agradecimientos

Queremos agradecer de manera especial a nuestro director José Luís Risco Martín por su apoyo y ayuda a lo largo del último año. Agradecer también al profesor José Ignacio Hidalgo Pérez y a la doctora Esther Maqueda por su por su valiosa colaboración y su disposición permanente para aclarar nuestras dudas durante el desarrollo del prototipo. Además agradecer al Hospital de Toledo por cedernos los aparatos de glucemias. Finalmente, agradecer a nuestros familiares y amigos por su comprensión y ayuda a lo largo de este año y, en concreto, a Markel Arizaga por su fundamental ayuda en el campo de la programación móvil para Android y ayudarnos a realizar las pruebas oportunas.

Palabras clave

Google Health, CCR, PHR, Telemedicina, Android, Java, API, Diabetes mellitus, Glucemia, Hipoglucemia, Hiperglucemia, Insulina, Cuerpos cetónicos.

Resumen

La telemedicina se está erigiendo como una de las vías de mejora más importantes del mundo médico. En concreto, la telemedicina aplicada a la diabetes está permitiendo mejorar la calidad de vida del paciente diabético. La diabetes necesita un control exhaustivo, por ello, es importante disponer de medios que faciliten ese control y su seguimiento por parte del médico.

En nuestro caso hemos aportado una aplicación que permite mejorar la calidad de vida del paciente y facilitar el seguimiento por parte de su endocrino. Por esto, debido a la importancia de este sector en la informática, las compañías más poderosas se están preocupando por encontrar un hueco en este campo de la medicina. Tanto Google, como Microsoft han desarrollado plataformas para almacenamiento de perfiles de pacientes. Nosotros hemos optado por Google Health como operador de servicio de almacenamiento. Además, hemos desarrollado una interfaz de programación de aplicaciones o API (del inglés *Application Programming Interface*) para Java que permite interactuar con Google Health de forma sencilla. Por lo tanto, se ha creado una API para la conexión con Google Health y se ha elaborado una aplicación práctica de este API mediante un programa de gestión de la diabetes denominado *GluControl*.

Telemedicine has the potential to provide great advances in the medical field. More concretely, telemedicine applied to diabetes has improved diabetic patient’s quality of life. Diabetes needs a thorough control and thus, it is important to have the means in which doctors may monitorize and control their diabetic patients easier.

In this regard, we have developed an application that tries to improve patients' quality of life, as well as tries to improve monitoring patients under a cloud-based telecare system. Due to the importance of this research field, most of the powerful companies are creating databases in the cloud. Both Google and Microsoft have developed platforms in order to store patient profile data. We have selected Google Health as a personal health information centralization service. In addition, we have developed a Java application, which easily allows us to interact with Google. To this end, we have created an Application Programming Interface (API), connecting Google Health with both patient and doctor user interfaces in an easy manner. Finally, with both API and user interfaces we have developed Glucontrol, a diabetenes telecare management system.

Índice

1 Introducción y trabajo relacionado

1.1 Introducción

La diabetes mellitus es una de las enfermedades más significativas en el mundo sanitario, no solo por su prevalencia, si no por ser un importante factor de riesgo para desencadenar una patología cardiovascular si no se controla de manera adecuada. La diabetes mellitus está caracterizada por la falta o déficit de secreción de insulina por las células del páncreas. La diabetes mellitus tipo I suele diagnosticarse antes de los 30 años de edad y afecta a cerca de 5 millones de personas en todo el mundo. El correcto tratamiento de este tipo de diabetes depende de tres pilares fundamentales: dieta, ejercicio físico y aplicación de dosis de insulina. Para lograrlo, es fundamental mantener un control diario y exhaustivo de los niveles de glucosa, rutina que, sin embargo, no todos los diabéticos llevan a cabo. Por ejemplo, no llevar un registro adecuado de las glucemias, no se inyectarse insulina el número de veces idóneo, no controlar la dieta o no realizar ejercicio físico suficiente, aumentan considerablemente las posibilidades de desarrollar complicaciones de la diabetes, tales como problemas cardiovasculares, renales o de visión. Por tanto, es imprescindible que el paciente disponga de una herramienta que facilite el seguimiento de estos datos, así como la comunicación con su endocrinólogo.

La irrupción generalizada de Internet en los hogares, el incremento de las capacidades de la telefonía móvil y la televisión digital terrestre, han abierto nuevas vías de comunicación entre el paciente y el médico, dando lugar a una nueva forma de medicina: la *telemedicina*.

La telemedicina ofrece la posibilidad de prestar servicios médicos a distancia, que en el caso de enfermedades como la diabetes, facilita la comunicación médico-paciente, reduciendo considerablemente la necesidad de recurrir a la consulta presencial. Esto supone, sin duda, una gran mejora en la calidad de vida para pacientes crónicos o de avanzada edad. Por razones como ésta, la telemedicina se está posicionando como una de las soluciones para los pacientes diabéticos. Muestra de ello es el programa pionero desarrollado en EEUU durante los años 1996 a 1998, *DEMS (*Sistema Electrónico del Manejo del Diabético), que obtuvo buenos resultados o el proyecto *M2DM***,** financiado y promovido por la Unión Europea, cuyo objetivo es mejorar la relación y la comunicación que existe entre médicos y pacientes mediante la utilización de servicios de telemedicina de bajo costes y sencillos de utilizar.

En España también se están llevando a cabo numerosos proyectos de Telemedicina. Entre ellos, destaca el *Proyecto de* p*revención de la retinopatía*

diabética por teleconsulta llevado a cabo por el Hospital Navarra o el propio *M2DM* desarrollado por la Universidad Politécnica de Madrid.

La repercusión de la telemedicina en la sociedad es tal, que incluso los dos gigantes de la informática, Microsoft y Google, ya están tomando posiciones, con Microsoft HealthVault y Google Health, respectivamente. Ambos casos son servicios gratuitos que permiten a sus usuarios albergar el historial clínico en la red y poder compartirlo con distintas organizaciones del área de la salud (empresas, hospitales, aseguradoras e incluso médicos privados).

Google Health es un sistema que se encuentra en fase Beta y del que aún no se ha explotado todo su potencial. Por ello, hemos desarrollado una interfaz de programación de aplicaciones o API (del inglés *Application Programming Interface*) para Java que facilita la conexión con Google Health para almacenar y monitorizar de datos médicos del paciente. Dicha API puede ser utilizada desde cualquier aplicación y proporciona un acceso a Google Health de forma sencilla. De hecho, la modularidad con la que ha sido desarrollada permitiría migrar a otro proveedor de servicios de salud (*Microsoft Health Vault*, *Dossia*, etc.), siempre respetando el formato *Continuity of Care Record* (CCR), sin realizar grandes cambios en el código.

Para demostrar la utilidad de este API, hemos desarrollado una herramienta de telemedicina para el control de la diabetes. Nuestra herramienta, GluControl, trata de facilitar la transferencia de información entre paciente y médico, es decir, mejorar la comunicación. El paciente podrá introducir sus medidas diarias de glucosa, insulina, dietas, etc.; medidas que serán monitorizadas por su médico de forma inmediata. Estos datos se podrán visualizar mediante gráficas. Por ejemplo, un médico podrá observar una gráfica comparativa entre las glucemias de los dos últimos meses, o podrá obtener un gráfico de la insulina inyectada la última semana. Además el médico tendrá acceso a otra información clínica relevante como, por ejemplo, el historial médico del paciente. Todo ello, facilita el seguimiento de la diabetes y disminuye el riesgo de desarrollar complicaciones típicas de esta enfermedad.

Tratándose la diabetes de una enfermedad crónica, facilitar la convivencia con dicha enfermedad supone un reto para la telemedicina. Debido a ello, en nuestra herramienta hemos dedicado un gran esfuerzo a facilitar la anotación de las medidas diarias del paciente. Con este objetivo, hemos adaptado la aplicación para que pueda ser utilizada desde Android. Basta con disponer de un teléfono móvil con sistema operativo Android y una conexión a la red, para que el paciente pueda introducir sus medidas de glucosa, insulina o dieta; en cualquier momento y lugar, sin necesidad de tener acceso a su ordenador personal.

Además, también se ha tratado de adaptar GluControl a los nuevos dispositivos de medida de glucosa que, por medio de conexiones USB, permiten descargar las medidas en el ordenador. Así, un paciente podrá

registrar en la aplicación las glucemias que previamente haya descargado de su glucómetro en un fichero XML.

La Figura 1.1 ilustra todas las vías posibles con las que el paciente podrá introducir nuevas medidas en su perfil clínico albergado en Google Health.

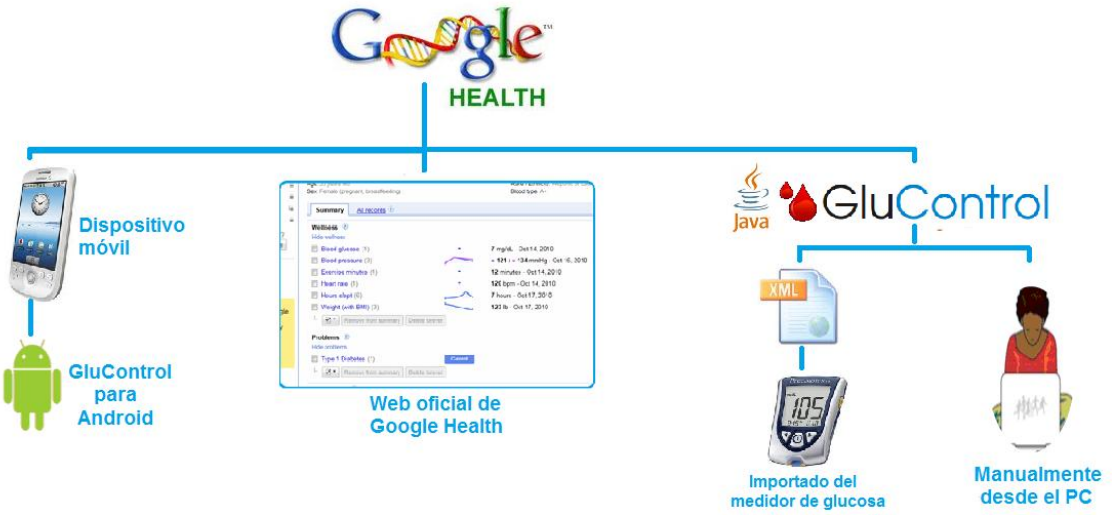


Figura 1.1 Vías posibles para registrar nuevas medidas

1.2 Trabajo relacionado

Debido a la importancia de la diabetes en la sociedad actual, se están llevando a cabo multitud de proyectos. Pero no todo el esfuerzo por mejorar la telemedicina se hace a gran escala, son numerosas las aplicaciones desarrolladas para mejorar el control de la diabetes. Sin embargo, muchas de estas no se atienen a las necesidades de los pacientes, ya que la mayoría de ellos son pacientes de avanzada edad o pacientes crónicos. Teniendo en cuenta que gran parte de los diabéticos no son expertos en informática nuestra labor es la de simplificar al máximo la introducción de glucemias para su posterior seguimiento y monitorización por parte del médico.

Por otra parte, la mayoría de estas aplicaciones están centradas en la interfaz del paciente; no facilitando la comunicación con su médico. Ofrecen numerosas herramientas para el paciente, sin embargo no ofrecen una comunicación directa con su médico; resultando tedioso y complejo el hecho de presentar los resultados analizados/obtenidos a su médico. En este contexto se encuentran muchas aplicaciones desarrolladas por laboratorios propietarios de los glucómetros y, que por tanto, su aplicación está restringida a los datos descargados de los glucómetros (sus glucómetros). Este es el caso de aplicaciones como Menadiab del laboratorio Menarini Industrie Farmaceutiche Reunite.

Esta aplicación ofrece una amplia gama de herramientas de visualización/monitorización de medidas para el paciente, pero no es ofrece la comunicación directa con el médico. Además las herramientas que ofrece no

son sencillas de utilizar para una persona con conocimientos bajos en informática. En la Figura 1 2 podemos ver el aspecto visual de Menadiab.

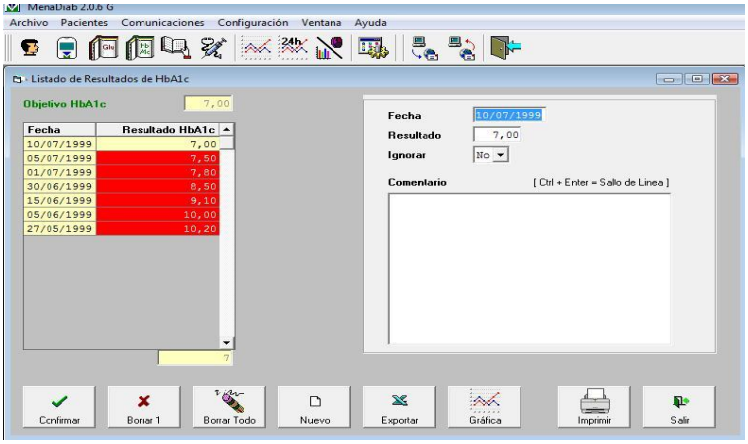


Figura 1 2 Aspecto visual del programa Menadiab

Otro ejemplo es la aplicación CoPilot de laboratorios Abbot. Dicha aplicación ofrece multitud de posibilidades para el paciente, como puede observarse en la Figura 1-3.

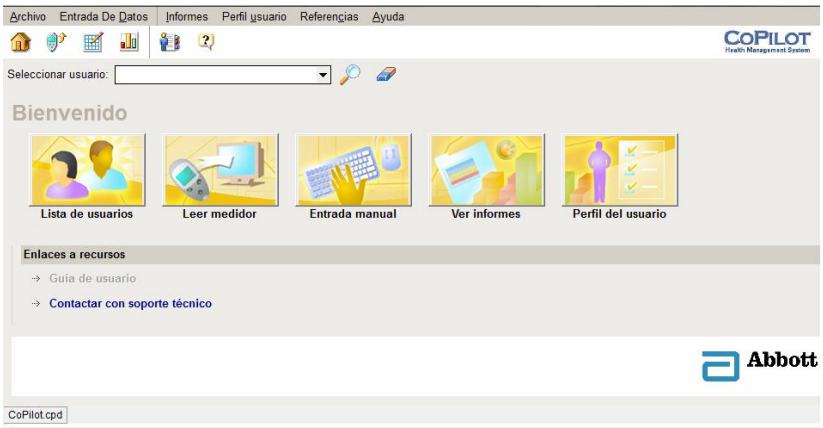


Figura 1-3 Vista de la pantalla principal de CoPilot Management System

Además ofrece la posibilidad de enviar vía correo electrónico las medidas almacenadas por el paciente. Sin embargo no existe una comunicación de forma inmediata si no que ha ser el paciente el que envíe los datos a su médico.

En *GluControl*, sin embargo, todos los datos introducidos por el paciente son actualizados de forma inmediata en Google Health, por lo que el médico verá siempre las medidas del paciente actualizadas. Además el médico tendrá acceso a todo el historial médico del paciente así como a sus análisis actuales, cosa que en estas aplicaciones no es posible llevar a cabo.

Mientras que las aplicaciones desarrolladas por los laboratorios son utilizadas para sus propios glucómetros, *GluControl* permite cargar medidas a partir de los ficheros XML descargados de cualquier glucómetro.

Además nuestra herramienta es compatible con Android por lo que ofrece muchas facilidades a la hora de introducir medidas en cualquier momento y lugar.

Existen otras aplicaciones que sí ofrecen una comunicación paciente-médico además de permitir la extracción de medidas desde distintos glucómetros. Estas aplicaciones disponen además de herramientas para monitorizar y visualizar en gráficas las distintas medidas. Es el caso de MyCareTeam. Esta aplicación reúne las características anteriormente descritas y facilita la comunicación paciente-médico a través de Google Health. Es una aplicación Web con bastante impacto en el mundo del control de la diabetes. No obstante, se trata de una aplicación de pago por lo que no todo el mundo puede acceder a ella.

En nuestro caso hemos desarrollado una herramienta similar en muchos aspectos a *MyCareTeam,* pero con la diferencia de ser *open source* y totalmente gratuita. Además, el API para la conexión con Google Health puede ser reutilizado como base para multitud de aplicaciones libres no obligatoriamente centradas en la diabetes.

A continuación, expondremos las principales características de las aplicaciones anteriormente descritas (Tabla 1-1).

Aplicación	Seguimiento y Monitorización Medidas	Inserción medidas desde Glucómetros	Acceso al historial completo del paciente	Comunicación paciente-médico	Compatibilidad con dispositivos móviles
MenaDiab	Si, completo	Glucómetros Arkray	No	Vía e-mail	No
Copilot	Si, completo	Glucómetros Abbot	No	Vía e-mail	No
MyCareTeam	Si, completo	Si, amplia gama glucómetros	No completo	Vía Google Health	No
GluControl	Si, completo	No directamente. Sí desde .xml de medidas	Si, completo	Vía Google Health	Sí (Android)

Tabla 1-1 Comparativa de aplicaciones para monitorización de la diabetes

En cuanto al módulo de conexión con GH desarrollado. Existe un API desarrollada por Google a partir de la cual ha sido desarrollado nuestro módulo. Sin embargo, el API de Google es más complicada y difícil de utilizar que nuestro módulo.

2 Contribuciones del proyecto

Como hemos explicado en la sección 1.2 nuestro trabajo ha consistido en realizar una API para Java que permite interactuar con Google Health de forma sencilla. Este API se ha utilizado para realizar un programa de gestión de la diabetes denominado GluControl y de esta manera mejorar la vida de los pacientes diabéticos.

En la sociedad actual, la diabetes afecta a cerca de 5 millones de personas del todo el mundo. La causa exacta se desconoce, pero lo más probable es que haya un desencadenante viral o ambiental en personas genéticamente susceptibles que causa una reacción inmunitaria. Esta enfermedad necesita llevar un control exhaustivo de las glucemias, ejercicio, físico, dietas, etc. Nuestra aplicación permite que el paciente introduzca todas las necesidades básicas para la mejora en el control de la diabetes y de esta manera el endocrino pueda observar cualquier anomalía o mejoras en el paciente.

Debido a la gravedad del problema y al avance la de telemedicina, se han desarrollado numerosas aplicaciones que tratan de facilitar la vida del paciente diabético. Sin embargo, muchas de estas aplicaciones son demasiado complejas y tediosas de utilizar como para llevar un control diario de la enfermedad. Por ello, nuestro objetivo ha sido, en todo momento, elaborar una aplicación que utilice nuestra API y que cualquier paciente diabético pueda usar: desde una persona con escasos conocimientos médicos e informáticos, a una persona que disponga de un móvil y quiera introducir sus medidas a través del mismo.

Con el API que proporcionamos se podrían desarrollar multitud de aplicaciones que utilizaran Google Health como proveedor de almacenamiento de perfiles. Tan sólo tendrían que desarrollar su aplicación sin preocuparse por la comunicación con Google Health. Esto permite dedicar más tiempo al desarrollo de la aplicación y por lo tanto elaborar mejores aplicaciones. Es más, hemos redactado un manual en castellano para realizar operaciones básicas sobre Google Health. Operaciones como insertar, borrar, actualizar un elemento. Este manual ha sido desarrollado porque la documentación que proporciona Google sobre su API para interactuar con Google Health está demasiado resumida, es difícil de encontrar y sólo está disponible en inglés. Bien es cierto que Google Health está en fase beta por lo que es lógico que aún no se haya proporcionado una documentación tan extensa como poseen otros servicios de Google. Tras muchas preguntas en el foro de *Google Developers*, muchas pruebas propias, y muchas lecturas no muy relacionadas con Google Health conseguimos desarrollar el API para la conexión con Google Health. Por todo esto, pensamos que si cualquier persona decide trabajar con Google Health le sería más fácil disponer de una buena documentación.

Nuestro manual ha sido enviado a Google España y esperamos que sea de utilidad para la comunidad.

Debido al hecho de que Google Health está en fase beta de desarrollo hemos encontrado diversos *bugs* que hemos reportado a Google (ver APÉNDICE II Bugs de Google Health). A su vez, diversas funcionalidades importantes aún no han sido implementadas por Google. Funcionalidades como registrar nuevos perfiles desde Java, sólo, agrupar medidas del mismo tipo, etc. por lo que nos hemos visto obligados a buscar alternativas frente a estas ausencias. Un ejemplo de ello es el proceso de alta de pacientes desde la aplicación, que hemos solucionado redirigiendo directamente a la página de Google Health encargada de ello y mostrando un pequeño tutorial con el que guiar al usuario.

En resumen, hemos aportado una potente envoltura para el API de Google Health, que facilita con creces el trabajar con esta aplicación y hemos desarrollado una potente aplicación telemática, basada en este API, que permite llevar un control de la diabetes tanto por parte del médico como del diabético.

3 La diabetes

3.1 La diabetes en España

La diabetes mellitus (DM) es una de las enfermedades con mayor prevalencia y repercusión sociosanitaria, no solo por su elevada frecuencia, sino también por el impacto de las complicaciones crónicas de la enfermedad o el papel que desempeña como factor de riesgo de la patología cardiovascular.

En España la prevalencia de la DM se estima en un 6,2% para los grupos de edad entre 30-65 años y del 10% para 30-89 años. Por otra parte, la incidencia de DM tipo 2 se estima en 8/1000 habitantes/año y la de tipo 1 en 11-12 casos por 1000 habitantes/año.

La prevalencia de las distintas complicaciones crónicas varía en función del tipo de DM, tiempo de evolución y grado de control metabólico. Las estimaciones globales son las siguientes: neuropatía, un 25%; retinopatía, un 32% y nefropatía un 23%. La DM es una de las principales causas de mortalidad en España, ocupando el tercer lugar en mujeres y el séptimo en varones.

Actualmente en España, la diabetes tipo 2 está apareciendo más en los inmigrantes. Se calcula que residen en nuestro país entre 300.000 y 500.000 inmigrantes que sufren esta enfermedad. Los médicos están preocupados por los hábitos alimenticios y las costumbres que tienen los inmigrantes. La adquisición de hábitos occidentales poco saludables unida a una mayor predisposición genética en ciertas etnias contribuye a que la diabetes mellitus tipo 2 se dé entre estas personas. Los diabéticos inmigrantes tipo 2 son más jóvenes entre los 30 y 40 años, y según explican expertos en la materia, los inmigrantes que han sufrido mucha hambre desarrollan resistencia a la insulina. Cuando llegan a un país donde la comida es abundante y el ejercicio físico disminuye, lo que parece una ventaja, se convierte en una desventaja.

Desde hace 10 años la diabetes ha aumentado considerablemente en la sociedad española, unas posibles causas de este aumento son debidas a:

1. Cambio de criterios diagnósticos: en el año 1999 se cambió el valor de glucemia en ayunas y se pasó a considerar la presencia de diabetes de 140mg/dl a 126 mg/dl; esta reducción de las cifras podría haber producido un incremento entre el 1,4% y el 3,5%.
2. Envejecimiento de la población: el aumento de la esperanza de vida conduce a que haya más personas que puedan desarrollar diabetes, y que aquellas que ya lo han hecho vivan más años, lo cual, naturalmente, aumenta el número de individuos con diabetes.
3. Descenso de la mortalidad: la implementación de terapias para tratar la Diabetes Mellitus 2 y sus factores de riesgo puede haber comportado un descenso de la mortalidad y una mayor esperanza de vida. Estos datos son

claros en otros países, aunque en España no los hay para poder confirmarlos.

4. Aumento en la incidencia: otro posible factor implicado sería un verdadero aumento en nuevos casos diabetes. En España hay escasos datos que puedan avalar esta tendencia, aunque en otros países sí se ha demostrado.
5. Sean cuales sean los factores implicados, lo cierto es que el número de personas con diabetes aumenta día a día, hecho que está directamente relacionado con el aumento de la obesidad, el sedentarismo y el actual estilo de vida.

3.2 ¿Qué es la diabetes?

La diabetes mellitus está caracterizada por una secreción anormal y deficitaria de insulina por las células del páncreas, cuya consecuencia inmediata es la tendencia a mantener los niveles de glucosa en sangre inapropiadamente elevados. El páncreas es una glándula mixta que tiene dos funciones: endocrina y exocrina.

La función endocrina es la encargada de producir y segregar dos hormonas importantes, la insulina y el glucagón, a partir de unas estructuras llamadas islotes de Langerhans. En ellas, las células alfa producen glucagón, las células beta producen insulina y las células delta producen somatostatina.

La función exocrina consiste en la producción del jugo pancreático. Este está formado por agua, bicarbonato, y numerosas enzimas digestivas, como la tripsina y quimotripsina (digieren proteínas), amilasa (digiere polisacáridos), lipasa (digiere triglicéridos o lípidos), ribonucleasa (digiere ARN) y desoxirribonucleasa (digiere ADN).

El glucagón

Es una hormona que se produce en las células alfa. Por un lado interviene regulando las hipoglucemias: cuando detecta una disminución de glucosa en nuestro organismo, el glucagón actúa liberando la glucosa que estaba almacenada en el hígado. Esto suele ocurrir mayoritariamente durante la noche de manera espontánea. Por otro lado, también se encarga de estimular la liberación de insulina, favoreciendo así la metabolización del aumento de glucosa que ha desencadenado.

La insulina

Es una hormona que se produce en las células betas del páncreas y que se libera a la sangre de forma continuada y en mayor o menos cantidad en función sean sus requerimientos por parte del organismo. Principalmente por el mayor o menor ingreso de hidratos de carbono y, de manera secundaria, por los ácidos grasos, cuerpos cetónicos y aminoácidos en nuestro cuerpo.

La secuencia metabólica que de ordinario se produce en nuestro organismo es la siguiente: ingestión de alimentos ricos en hidratos de carbono,

aumenta la tasa de glucemia, liberación de insulina, ingreso y utilización de glucosa, ácidos grasos y aminoácidos por las células, almacenamiento de la glucosa, aminoácidos y ácidos grasos.

La carencia o disminución de la insulina es la que determina la falta de control en la producción de glucosa, por parte del hígado. En efecto, si la insulina de que dispone el organismo resulta insuficiente, la glucosa no puede ingresar dentro de las células y en consecuencia, aquellas se quedan sin el necesario aporte energético que está previsto que les llegara a través de ella. Entonces el organismo liberara grasa y proteínas de donde están almacenadas para tratar a las células la energía que necesitan. En el momento que hubiese insulina disponible, la llegada a la sangre de esta porción de glucosa restablecería el equilibrio perdido. Pero si la insulina no está disponible lo único que hace es complicar todavía más el cuadro, pues aumenta su concentración en sangre (hiperglucemia).

Somatostatina

Las células delta producen somatostatina, hormona que se cree que regularía la producción y liberación de la insulina por las células beta y la producción y liberación de glucagón por las células alfa.

3.2.1 Origen

La diabetes mellitus tipo I suele diagnosticarse antes de los 30 años de edad. Afecta a cerca de 5 millones de personas del todo el mundo. La causa exacta se desconoce, pero lo más probable es que haya un desencadenante viral o ambiental en personas genéticamente susceptibles que causa una reacción inmunitaria. Los glóbulos blancos del cuerpo atacan por error a las células beta pancreáticas productoras de insulina. De esta manera el cuerpo deja de generar insulina y tiene que ser pinchada. Se vincula esta reacción inmunitaria a una modificación en el cromosoma 6.

En la diabetes mellitus tipo II suele afectar a personas con antecedentes familiares. Aunque existe un componente genético, el desarrollo de la diabetes está relacionado con los hábitos de vida, el sedentarismo, la mala alimentación, obesidad, hipertensión….

En el caso de la diabetes mellitus gestacional se cree que podría deberse al aumento en la producción de hormonas de la placenta durante el embarazo. Estas hormonas pueden afectar o bloquear la producción de insulina en el cuerpo de la madre. Esto hace difícil que el cuerpo de la madre utilice la insulina para procesar la glucosa necesaria lo que provoca el incremento de los niveles de azucares en sangre, dado lugar a una hiperglucemia.

3.2.2 Tipos

Diabetes mellitus Tipo1

Este tipo de diabetes corresponde a la llamada antiguamente Diabetes insulinodependiente o Diabetes de comienzo juvenil. Se presenta mayoritariamente en individuos jóvenes(menores de 25 años), aunque puede aparecer en cualquier etapa de la vida, y se caracteriza por la nula producción de insulina debida a la destrucción autoinmune de las células β de los Islotes de Langerhans del páncreas. La diabetes tipo I se clasifica en dos subtipos, la **1a** (diabetes juvenil) y la **1b** (diabetes asociada a otras numerosas y variadas alteraciones endocrinas).

El paciente cuya diabetes está más vinculada a factores etiológicos de tipo genético se caracteriza por tener autoanticuerpos séricos persistente frente a los antígenos de las células de los islotes, siendo más frecuente encontrar en ellos y en sus familiares otros trastornos endocrinos de tipo autoinmunitario. Entre dos gemelos idénticos se ha encontrado que tiene un alto grado de concordancia respecto del padecimiento de la diabetes. Hay un 50 % de posibilidades de que uno de los gemelos padezca también diabetes tipo I, si el otro hermano la padece.

Diabetes mellitus Tipo2

El rasgo principal de la diabetes tipo 2 es el déficit relativo de la producción de insulina, y una deficiente utilización periférica por los tejidos de glucosa. Esto quiere decir que el receptor de la insulina de las células que se encargan de facilitar la entrada de la glucosa a la propia célula está dañado.

A menudo se desarrolla en etapas adultas, a partir de los 45 años. En menores de 45 años suele surgir debido a obesidad, antecedentes familiares, hipertensión arterial, colesterol o antecedentes de la glucosa en sangre. La diabetes tipo 2 en gente joven no es frecuente.

Anteriormente se denominaba diabetes del adulto o diabetes relacionada con la obesidad. A este tipo de diabetes pertenece entre el 80-90 % de todos los pacientes diabéticos. Las personas con diabetes tipo II también de vez en cuando sufren de hipoglucemias o hiperglucemias.

Esta enfermedad puede llevar consigo diferentes complicaciones en órganos internos y afectar al desempeño de la actividad diaria. Los problemas más frecuentes son los relacionados con la vista, pies, riñones, dientes, sistema cardiovascular y sistema nervioso.

Diabetes mellitus gestacional

También llamada la diabetes del embarazo, aparece durante la gestación entre el 1% y 14 % de las pacientes, casi siempre debutan entre las semanas 24 y 28 de embarazo. En algunas ocasiones puede persistir después del parto.

La razón por la cual existe la posibilidad de tener diabetes durante el embarazo es debido al esfuerzo mayúsculo que se produce a lo largo del mismo. El feto utiliza los órganos de la madre para obtener alimento, energía, eliminar desechos, etc. Por estas razones puede presentar una deficiencia de la hormona de la insulina, haciendo que aparezca este problema. Si aparece la diabetes en un embarazo, la posibilidad que surja en los siguientes es grande. Respecto al feto, en principio no debería tener ningún problema, pero puede contribuir al desarrollo de un feto con macrosomía, es decir, un recién nacido con peso por encima de lo normal. Esta diabetes puede producir una diabetes de tipo II en el recién nacido.

Otros tipos de diabetes mellitus menores **(< 5% de todos los casos diagnosticados)**

Tipo 3A: defecto genético en las células beta.

Tipo 3B: resistencia a la insulina determinada genéticamente.

Tipo 3C: enfermedades del páncreas.

Tipo 3D: causada por defectos hormonales.

Tipo 3E: causada por compuestos químicos o fármacos.

3.2.3 Manifestaciones clínicas

Algunas personas no tendrán ningún síntoma antes de que se les diagnostique la diabetes. Otras pueden notar los siguientes síntomas como los primeros signos de diabetes tipo 1 o cuando la glucemia está alta:

Sentirse cansado o fatigado

Sentirse hambriento

Estar muy sediento

Orinar con mayor frecuencia

Perder peso sin proponérselo

Tener visión borrosa

Perder la sensibilidad o sentir hormigueo en los pies

Para otras personas, los síntomas de advertencia de que se están poniendo muy enfermos pueden ser los primeros signos de diabetes tipo 1, o pueden suceder cuando la glucemia está muy alta:

Respiración profunda y rápida

Boca y piel seca

Cara enrojecida

Aliento con olor a fruta

Náuseas o vómitos, incapacidad para retener los líquidos

Dolor de estómago

La glucemia baja (hipoglucemia) se puede desarrollar rápidamente en personas con diabetes que estén tomando insulina. Los síntomas aparecen típicamente cuando el nivel de glucemia cae por debajo de 70 mg/dl. Tenga cuidado con:

Dolor de cabeza

Hambre

Nerviosismo

Latidos cardíacos rápidos (palpitaciones)

Temblores

Sudoración

3.3 Prevención de la diabetes

Para la diabetes tipo 1 no existe ningún método eficaz por el momento. En cambio, está comprobado que la de tipo 2, que es la que aparece con más frecuencia, al estar relacionada con la obesidad se puede tratar de evitar en gran medida adoptando unos hábitos de vida saludables:

Evitando el sobrepeso y la obesidad.

Realizando ejercicio físico de forma regular.

Abandonando el tabaco y las bebidas alcohólicas.

Siguiendo una dieta alimentaria sana.

Para prevenir las hipoglucemias, los diabéticos deben tener en cuenta lo siguiente:

Ajustar las dosis de los medicamentos a sus necesidades reales.

Mantener un horario de comidas regular en la medida de lo posible;

Tomar cantidades moderadas de hidratos de carbono antes de realizar ejercicios extraordinarios;

Llevar siempre azúcar consigo. En cuanto aparezcan los primeros signos de hipoglucemia, hay que tomar azúcar (2 o 3 terrones), galletas (de 3 a 5 unidades) o beber un vaso (150 ml) de alguna bebida que contenga hidratos de carbono de absorción rápida (zumos de frutas, cola, etc.). Los síntomas suelen pasar en 5 o 10 minutos. Si la hipoglucemia es grave o la persona pierde la conciencia, es necesario inyectarle una ampolla de glucagón por vía subcutánea (igual que la insulina) o intramuscular (en la nalga). El glucagón moviliza las reservas de glucosa del organismo y hace efecto en unos 10 minutos. Si no hay recuperación, el afectado debe recibir asistencia médica inmediata.

3.4 Diagnóstico de la diabetes

La diabetes se diagnostica con los siguientes exámenes de sangre:

Nivel de glucemia en ayunas: la diabetes se diagnostica si es superior a 126 mg/dl en dos ocasiones.

Nivel de glucemia aleatoria: la diabetes se sospecha si es superior a 200 mg/dl y el paciente tiene síntomas como aumento de la sed, de la micción y fatiga (esto se debe confirmar con examen en ayunas).

Prueba de tolerancia a la glucosa oral: la diabetes se diagnostica si el nivel de glucosa es superior a 200 mg/dl después de dos horas.

Examen de hemoglobina A1c: este examen se ha usado en el pasado para ayudarles a los pacientes a vigilar qué tan bien están controlando sus niveles de glucosa en la sangre. En el 2010, Asociación Estadounidense para la Diabetes recomendó que el examen se use como otra opción para diagnosticar la diabetes e identificar la prediabetes. Los niveles indican:

- Normal: Menos de 5.7%
- Prediabetes: Entre 5.7% y 6.4%
- Diabetes: 6.5% o superior

El examen de cetonas también se utiliza en la diabetes tipo 1. Las cetonas son producidas por la descomposición de la grasa y el músculo, y son dañinas en niveles altos. El examen de cetonas se hace empleando una muestra de orina, por lo general se realiza en los siguientes momentos:

Cuando la glucemia es superior a 240 mg/dl.

Durante una enfermedad como neumonía, ataque cardíaco o accidente cerebrovascular.

Cuando se presentan náuseas o vómitos.

Durante el embarazo.

Los siguientes exámenes ayudarán a que el paciente y su médico vigilen su diabetes y prevengan complicaciones:

Inspecciona de la piel, los pies y las piernas.

Verificación de la sensibilidad en los pies.

Revisión de la presión arterial al menos cada año (la presión arterial ideal debe ser de 130/80 mm/Hg o más baja).

Revisión de la hemoglobina glucosilada (HbA1c) cada 6 meses si la diabetes está bien controlada; de lo contrario, cada 3 meses.

Revisión de los niveles de colesterol y triglicéridos anualmente (procure lograr niveles de colesterol por debajo de 70-100 mg/dl).

Exámenes anuales para verificar que los riñones estén trabajando bien (microalbuminuria y creatinina en suero).

Revisión por el oftalmólogo al menos una vez al año o con mayor frecuencia si tiene signos de retinopatía diabética.

Revisión por el odontólogo cada 6 meses para una limpieza y examen dental completos, asegurándose de que el odontólogo y el higienista sepan que usted padece diabetes.

3.5 Tratamiento de la diabetes

El tratamiento de la diabetes mellitus se basa en tres pilares: dieta, ejercicio físico y medicación. Tiene como objetivo mantener los niveles de glucosa en sangre dentro de la normalidad para minimizar el riesgo de complicaciones asociadas a la enfermedad. En muchos pacientes con diabetes tipo II no sería necesaria la medicación si se controlase el exceso de peso y se llevase a cabo un programa de ejercicio físico regularmente. Sin embargo, es necesaria con frecuencia una terapia sustitutiva con insulina o la toma de fármacos hipoglucemiantes por vía oral.

Fármacos hipoglucemiantes orales. Se prescriben a personas con diabetes tipo II que no consiguen descender la concentración de azúcar en sangre a través de la dieta y la actividad física, pero no son eficaces en personas con diabetes tipo I.

Tratamiento con insulina. En pacientes con diabetes tipo I es necesario la administración exógena de insulina ya que el páncreas es incapaz de producir esta hormona. También es requerida en diabetes tipo II si la dieta, el ejercicio y la medicación oral no consiguen controlar los niveles de glucosa en sangre. La insulina se administra a través de inyecciones en la grasa existente debajo de la piel del brazo, ya que si se tomase por vía oral sería destruida en aparato digestivo antes de pasar al flujo sanguíneo. Las necesidades de insulina varían en función de los alimentos que se ingieren y de la actividad física que se realiza. Las personas que siguen una dieta estable y una actividad física regular varían poco sus dosis de insulina. Sin embargo, cualquier cambio en la dieta habitual o la realización de algún deporte exigen modificaciones de las pautas de insulina. La insulina puede inyectarse a través de distintos dispositivos:

Jeringuillas tradicionales, de un solo uso, graduadas en unidades internacionales (de 0 a 40).

Plumas para inyección de insulina. Son aparatos con forma de pluma que tienen en su interior un cartucho que contiene la insulina. El cartucho se cambia cuando la insulina se acaba, pero la pluma se sigue utilizando.

Jeringas precargadas. Son dispositivos similares a las plumas, pero previamente cargados de insulina. Una vez que se acaba la insulina se tira toda la jeringa. El nivel de glucosa en sangre depende de la zona del cuerpo en que se inyecta la insulina. Es aconsejable que se introduzca a través del abdomen, los brazos o muslos. Penetra más rápidamente si se inyecta en el abdomen. Se recomienda inyectar siempre en la misma zona, aunque desplazando unos dos centímetros el punto de inyección

de una vez a otra. Hay que evitar las inyecciones en los pliegues de la piel, la línea media del abdomen y el área de la ingle y el ombligo.

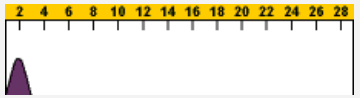
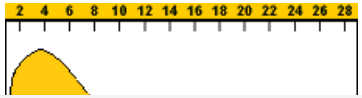
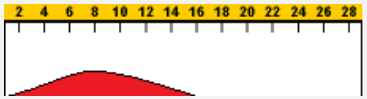
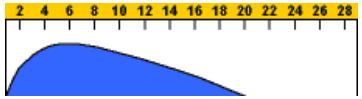
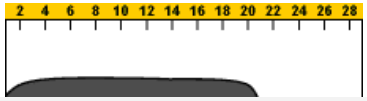
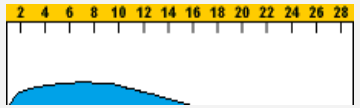
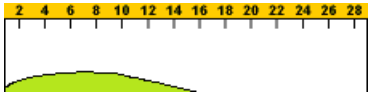
Cada tipo de insulina toma cierto tiempo para empezar a surtir efecto, para lograr su máximo efecto y para llegar al fin de su duración:

El inicio es el momento en el que la insulina empieza a bajar su nivel de azúcar en la sangre.

El pico es el momento en el que la insulina logra su máximo efecto para bajar su nivel de azúcar en la sangre.

La duración es el tiempo que dura la insulina, el período durante el que sigue disminuyendo su nivel de azúcar en la sangre.

En la Tabla 3-1 podemos observar el comportamiento temporal aproximado de los tipos de insulina más comunes.

Tipo de insulina	Marca	Nombre genérico	Inicio (minutos)	Pico (minutos)	Duración (horas)	Curva
De acción rápida	NovoLog	Insulina asparto	15.	30 a 90	3 a 5	
	Apidra	Insulina glulisina	15.	30 a 90	3 a 5	
	Humalog	Insulin lispro	15.	30 a 90	3 a 5	
De acción corta	Humulin R Novolin R	Regular (R)	30 a 60.	120 a 240	5 a 8	
De acción intermedia	Humulin N Novolin N	NPH (N)	60 a 180	8 horas	12 a 16	
De acción prolongada	Levemir	Insulina detemir	60	Sin pico	20 a 26	
	Lantus	Insulina glargina				
NPH premezclada (de acción intermedia) y regular (de acción corta)	Humulin 70/30 Novolin 70/30	70% NPH y 30% regular	30 a 60.	Varía	10 a 16	
	Humulin 50/50	50% NPH and 50% regular	30 a 60.	Varía	10 a 16	
Suspensión premezclada de insulina lispro con protamina (acción intermedia) e insulina	Humalog Mix 75/25	75% insulina lispro con protamina y 25% insulina lispro	10 a 15.	Varía	10 a 16	

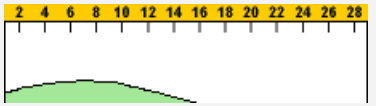
Tipo de insulina	Marca	Nombre genérico	Inicio (minutos)	Pico (minutos)	Duración (horas)	Curva
lispro (acción rápida)	Humalog Mix 50/50	50% insulina lispro con protamina y 50% insulina lispro	10 a 15.	Varía	10 a 16	
Suspensión premezclada de insulina asparto con protamina (acción intermedia) e insulina asparto (acción rápida)	NovoLog Mix 70/30	70% insulina asparto con protamina y 30% insulina asparto	5 a 15.	Varía	10 a 16	

Tabla 3-1 Períodos de tiempo aproximados de los tipos de insulina más comunes

La bomba de insulina

La bomba de insulina tiene un depósito (similar a una jeringa normal de insulina, pero un tamaño mayor), lleno de insulina rápida (humalog o novorapid), el tamaño es pequeño, funciona con pilas; y tiene un chip de computador, que permite al usuario controlar exactamente la cantidad de insulina suministrada por la bomba; todo lo cual está contenido dentro de un estuche de plástico. El depósito de la bomba suministra insulina al cuerpo del usuario mediante un tubo de plástico de distintos tamaños de largo y delgado, llamado "equipo de infusión" en la punta tiene un aguja o cánula blanda, por la cual pasa la insulina. La aguja se introduce por debajo de la piel, por lo general, en el abdomen. El proceso de colocación del equipo de infusión se denomina "inserción" y es muy semejante a la administración de una inyección de insulina estándar. El equipo de infusión se cambia habitualmente cada 3 días. Las bombas están diseñadas para ser usadas en forma continua y suministrar insulina las 24 horas del día, de acuerdo con un plan programado, adaptado a las necesidades de cada usuario. Una pequeña cantidad de insulina "dosis basal" suministrada en forma constante, mantiene el nivel de glucosa en la sangre entre comidas y durante la noche, dentro de los límites deseados. Al ingerir alimentos, el usuario programa la bomba para que suministre una dosis "bolo" de insulina, de acuerdo con la cantidad de alimento que va a ingerir. La bomba no es automática. El usuario debe decidir cuánta insulina necesita administrarse. Sin embargo, este dispositivo médico constituye el sistema de suministro de insulina más exacto, preciso y flexible, disponible en la actualidad. El usuario deberá realizar controles glucémicos para lograr un excelente control metabólico, llevando, al mismo tiempo, un estilo de vida normal, libre de las estrictas exigencias de horarios que imponen los regímenes de insulina convencionales.

Ventajas

Mejora el control de las glucemias, como recomiendan en el estudio DCCT.

Previene o evita las complicaciones.

Muy pocas reacciones a la insulina.

Reducción de las fluctuaciones de glucosa en sangre.

Corrige el Fenómeno del alba.

Realizar ejercicio físico sin miedo a las hipoglucemias.

Flexibilidad en cuanto a los horarios y las comidas.

Mejora el control cuando viajas o con un horario complejo de trabajo.

Riguroso control durante el crecimiento en los adolescentes.

Mayor libertad en su estilo de vida.

Reducción de las hospitalizaciones

Requisitos Previos

El paciente tiene que realizar 4 glucemias a día.

Tiene que estar motivado a llevar a cabo un control glucémico y mantenerlo

Candidatos

Extracto de:

Ministerio de Sanidad y Consumo:

ORDEN SCO/710/2004, de 12 de marzo; B.O.E. Nº 68 de 19 de marzo, Hoja 12217

Recomendaciones para la selección de pacientes susceptibles de la indicación de bombas de insulina:

1. *Pacientes diagnosticados de diabetes tipo 1 en estado de gestación o que se hayan mantenido, al menos seis meses antes de adoptar la bomba de insulina, dentro de un programa de inyecciones múltiples, como mínimo tres diarias, y que hayan requerido autoajustes frecuentes de la dosis de insulina.*
2. *Que hayan completado un programa educativo sobre el cuidado en la diabetes.*
3. *Que acrediten una frecuencia media de cuatro autocontroles diarios de glucemia durante los dos meses previos a la adopción de la bomba.*
4. *Que, manteniéndose en régimen de inyecciones múltiples, experimenten algunas de las siguientes circunstancias:*
    a. *Hemoglobina glicosilada > 7,0%.*
    b. *Historia de hipoglucemia recurrente.*
    c. *Amplias variaciones en la glucemia prepandial.*
    d. *Fenómeno del alba con glucemias que superen los 200 mg/dl.*
    e. *Historia de desviaciones glucémicas severas.*

3.6 Investigaciones en la diabetes

En los últimos años se ha producido un gran avance y modernización de los instrumentos que utiliza el diabético en el control de su glucemia diaria (pinchadores, agujas, jeringas, etc.), así como el tipo de insulina y su almacenamiento. Todo ello ha contribuido a una mejor calidad de vida del diabético. No obstante, el gran reto está por lograrse: recomponer las células de los islotes de Langerhans del páncreas.

Un ejemplo de estas investigaciones, que abre un camino a la esperanza, se encuentra en el Instituto de Bioingeniería de la Universidad Miguel Hernández de Elche. Las investigaciones han logrado una beca internacional para aplicar en seres humanos un método que, de momento, suprime radicalmente la diabetes en ratones enfermos. El protocolo empleado en los ratones se lleva a cabo con la creación desde las células madre, a partir de

células betapancreáticas, que generan la cantidad de insulina suficiente que requieren los diabéticos.

Las células, procedentes de animales sanos, son instaladas en los hígados y consiguen que se regenere el órgano afectado y que se elimine la diabetes. Hasta la fecha, el protocolo utilizado se basa en la selección de clones y diferenciación *in vitro* de las células que luego son trasplantadas a los organismos afectados. Aunque la fórmula ha tenido éxito en ratones ahora es necesario saber si la respuesta del cuerpo humano es la misma que en el organismo de los roedores estudiados, lo que tardará algún tiempo en conocerse debido a la dificultad que supone encontrar donantes.

3.7 Conclusión

La diabetes es una enfermedad que exige, por parte del afectado, una gran dosis de autocontrol ya que afecta a su vida diaria, a sus hábitos y costumbres. Un enfermo de diabetes debe ser riguroso en el control de las comidas y los horarios de inyección, respetando los intervalos en cada comida: programar el menú de cada día, seleccionando los alimentos que se van a consumir y repartiendo según sea el desayuno, comida, merienda, cena o el tentempié de la noche, los alimentos que contienen hidratos de carbono en las cantidades adecuadas. El enfermo de diabetes debe vigilar la evolución de su enfermedad mediante un autoanálisis regular para tener a la vista cómo está siendo el control glucémico.

Este trabajo quiere contribuir a una vida más sana de la persona con una diabetes tipo 1 a través de sus controles diarios de glucemia.

Las nuevas soluciones están muy lejos de que aparezcan ,por lo que en este momento el tratamiento más fiable es el seguimiento de los niveles de glucemia, de ejercicio físico, la dieta, cuerpos cetónicos, etc. Es por ello que este trabajo se enfoca en facilitar la labor tanto del endocrino como la de los pacientes, pudiendo tener un control más exhaustivo de la diabetes sin tener que ir a consulta médica. Para ello el paciente podrá introducir sus glucemias, sus dosis de insulina, dietas, medicamentos, ejercicios físicos, etc. de una manera clara y sencilla y el endocrino visualizar estos datos y guiar al paciente a un control más adecuado.

4 Google Health

4.1 El concepto de Salud 2.0 y los programas PHR

4.1.1 Definición de Salud 2.0

Salud 2.0 es el término elegido para describir una nueva visión del sistema sanitario, resultado de aplicar las nuevas tecnologías informáticas sobre el modelo de sanidad actual. Esta nueva visión supondría un modelo de salud mucho más ágil, sostenible e "interactivo", en el que participen activamente profesionales de la salud, pacientes y empresas del sector. Esta evolución se ve propiciada por los actuales entornos demográficos, sociales y económicos: el envejecimiento de la población, el efecto de la inmigración, la nuevas enfermedades derivadas del estilo de vida, etc. suponen un aumento del gasto sanitario y la saturación de los servicios ofrecidos tanto en hospitales como en los centros de Atención primaria. Por tanto, la aplicación de las tecnologías de la información y la comunicación (TIC) para el desarrollo de nuevas técnicas de telemedicina, parecen una solución aceptable para la situación de congestión e insostenibilidad del sistema sanitario actual. Entre las características que destacan los defensores de este nuevo modelo, cabría destacar las siguientes:

Movilidad, ya que facilita el acceso a la información sanitaria allí donde se precise, a través de tecnologías como Internet, Televisión Digital Terrestre o la telefonía móvil, salvando las barreras de tiempo y distancia.

Personalización, pues el aumento de la información disponible, conlleva el uso de tratamientos mucho más personalizados y afines al estilo de vida del paciente.

Monitorización, pues se aporta al profesional de la salud una información continua de estado de los pacientes, suponiendo una ahorro en tiempo y dinero y, sobretodo, un aumento del grado de independencia y bienestar del paciente, especialmente en enfermos crónicos.

Interoperabilidad. El uso de estándares para representar la información, facilitaría la colaboración y compartición de información relevante para el tratamiento de un paciente. La interoperabilidad permite el verdadero uso de los servicios de movilidad, personalización y monitorización anteriormente descritos.

4.1.2 Definición e historia de los PHR

Un ejemplo de cómo las TIC pueden revolucionar el modelo sanitario actual lo encontramos en el historial clínico electrónico (*Electronic Health Record, EHR*), cuyo objetivo es que la historia del paciente pase a ser un registro unificado y personal, accesible desde cualquier punto donde sea necesario (manteniendo siempre las garantías de consentimiento, confidencialidad, seguridad y demás requisitos). Además, su uso pretende

acabar con los problemas derivados del uso de la historia clínica en papel, tales como el desorden y falta de uniformidad de los documentos, la información ilegible, errores de archivado y la dudosa garantía de confidencialidad, entre otros (Carnicero Giménez, 2004). No obstante, pese a la gran expectación que produce, el historial clínico electrónico (desde ahora *HCE)* está todavía en desarrollo y será necesario mucho trabajo para que pueda llegar a ser una tecnología aplicable:

"[...] En la historia clínica electrónica deberá integrarse toda la información multimedia que se utiliza en la práctica clínica. Almacenar adecuadamente esta información, hacerla amigablemente accesible, difundirla de forma adecuada a los posibles usos y con las garantías debidas (consentimiento, confidencialidad, seguridad y demás requisitos), y recibirla y reutilizarla en la forma más conveniente es un proceso todavía en potencia. Conviene, pues, atemperar las excesivas expectativas acerca de la historia clínica electrónica." (Gérvas, 2001).

Sin embargo, en los últimos años, el desarrollo del *HCE* ha sufrido un gran impulso gracias a la entrada en el sector de dos gigantes, Google y Microsoft. En concreto, ambas empresas han apostado por el desarrollo de sus versiones de *Personal Health Record (PHR*), una modalidad de *HCE* que, a diferencia de ésta, no está únicamente dirigida al profesional de la salud, sino que el paciente toma parte activa en él, pudiendo consultar e introducir información referente a su estilo de vida.

Los *PHR* pueden contener un amplio abanico de datos, siendo los más comunes:

Alergias

Medicamentos (descripción, posología, etc.). Algunos *PHR* incluso controlan las interacciones entre medicamentos.

Enfermedades del paciente (puntuales, crónicas etc.)

Vacunas

Resultados de test (glucemias, análisis de sangre, etc.)

Datos variados que reflejen el estilo de vida (deporte realizado, horas de sueño, etc.)

El origen de los programas PHR debemos buscarlo en el proyecto *Guardian Angel* desarrollado en 1994 por *MIT Lab for Computer Science's Clinical Decision Making Group* (MEDG) en colaboración con *The Children's Hospital Informatics Program* (CHIP). Este proyecto pretendía desarrollar un sistema enfocado a pacientes diabéticos, pacientes con hipertensión y a pacientes tratados con anticoagulantes, que pretendía recopilar y gestionar su información sanitaria, educarles sobre su enfermedad, ayudarles a controlar su tratamiento y permitir a los profesionales de la salud conocer más sobre los efectos de la enfermedad y los tratamientos en el día a día de los pacientes.

Como podemos observar en la Figura 4-1 el proyecto *Guardian Angel* estableció las bases para el desarrollo de otras herramientas PHR tales como *Patient Site* (del *Beth Israel Deaconess Medical Center* de Boston), *MyHealthVet* (del *Department of Veterans Affairs*) y el más relevante de todos, el proyecto *Personal Internetworked Notary and Guardian (*PING), rebautizado en 2006 como *Indivo.* En los últimos años, el modelo de PHR se ha redescubierto, gracias a la inversión de grandes empresas, dando lugar a *Microsoft HealthVault*, *Google Health* y el proyecto open-source *Dossia* (construido sobre *Indivo*).

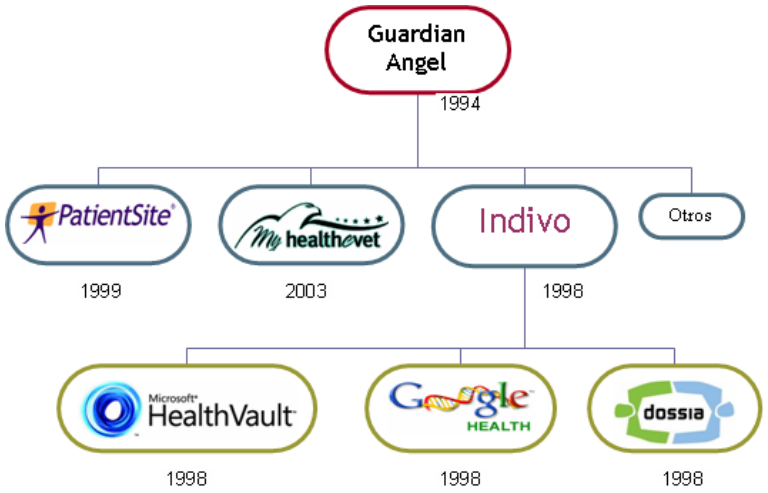


Figura 4-1 Diagrama de evolución de los principales PHR

4.2 Un ejemplo de PHR: Google Health

4.2.1 Breve historia de Google Health

Figura 4-2 Logotipo de Google Health

Figura 4-3 Primeras imágenes del prototipo de Google Health Agosto de 2007

Google Health es la apuesta de la compañía estadounidense para entrar en el mundo de los servicios de PHR. El desarrollo de Google Health comenzó en el año 2006 (en la Figura 4-3 podemos ver una de las primeras imágenes del prototipo) y en 2008 el sistema se puso a prueba con cerca de 1500 pacientes voluntarios de la clínica *Cleveland* (considerada una de las 4 clínicas más prestigiosas de EEUU según el estudio *America's Best Hospitals report* realizado en 2009 por la revista *U.S. News & World Report*). Tras el éxito obtenido en esta prueba, Google Health fue lanzado para el uso de todos los usuarios de Google como un servicio en versión *Beta* en Mayo de 2008. Desde entonces Google Health no ha parado de crecer especialmente gracias a las críticas y peticiones de los usuarios. En abril de 2009, Google introdujo la posibilidad de compartir los datos *PHR* del paciente con otros usuarios de Google, así como la posibilidad de exportarlos a formato *PDF*, imprimirlos y visualizarlos mediante gráficas. En junio de 2009, *CVS Pharmacy* (la segunda farmacéutica más grande de EEUU) se unió al proyecto de Google, permitiendo a *más de 100 millones de personas* (Arora, Maneesh 2009) poder importar su historial de medicamentos de las bases de datos de *CVS* de manera que tanto el médico como el paciente puedan tener acceso a esta información desde Google Health. Además, se incluyó un servicio de almacenamiento de ficheros (de máximo 4 MB) como documentos escaneados, resultados de analíticas etc. con una capacidad de almacenamiento total de 100 MB por cada perfil. La última actualización, hasta el momento, fue en Septiembre de 2010. En esta última actualización, Google cambió radicalmente el diseño de su herramienta, dándole un aspecto más limpio, ordenado e intuitivo (Figura 4-4) y, para mejorar el seguimiento diario de la salud del paciente, incluyó nuevas herramientas para que el usuario pueda crear gráficas personalizadas (para horas de sueño, peso, etc.).

Figura 4-4 Nuevo aspecto de Google Health tras la actualización de Septiembre de 2010

Además, a medida que Google Health ha ido creciendo, son cada vez más las empresas y otras organizaciones de la salud (en su mayoría estadounidenses) que deciden formar parte de este proyecto. Actualmente (febrero de 2011) se cuentan 25 organizaciones (hospitales, clínicas, farmacéuticas, etc.) que permiten al paciente importar los datos médicos pudiera tener con éstas; 38 organizaciones (empresas, asociaciones de médicos, clínicas, hospitales, etc.) que ofrecen servicios sanitarios al paciente en función de los datos almacenados en su perfil (siempre y cuando el paciente haya autorizado a dichas organizaciones para poder acceder a sus datos); 2 empresas que localizan y convierten a formato electrónico los datos médicos del paciente; 16 herramientas para el control enfermedades como la diabetes, la epilepsia o enfermedades cardiacas que utilizan la cuenta de Google Health para almacenar los datos y, para finalizar, 8 servicios *online* que permiten importar los datos de Google Health.

4.2.2 Descripción de los servicios de Google Health

Listado de servicios que ofrece oficialmente.

Por tratarse de un programa PHR, Google Health ofrece la mayoría de los servicios que definen los PHR, es decir, todas aquellas herramientas necesarias para que el paciente pueda llevar un registro de los datos de salud más relevantes (medidas, medicamentos, etc.).

En concreto, Google Health permite registrar datos relativos a las siguientes categorías:

Wellness: esta categoría representará a modo de tabla o mediante gráficas (Figura 4-5), medidas de la salud del paciente tales como el peso, la altura, presión sanguínea, ejercicio realizado, etc. A diferencia de otros programas PHR, Google Health ofrece la posibilidad al paciente de definir sus propias medidas y marcar objetivos personales.

Figura 4-5 Gráfica de presión sanguínea vista con Google Health

Problems: como podemos observar en la Figura 4-6, esta categoría representará las enfermedades y dolencias que el paciente ha padecido o padece. A la hora de insertar una nueva dolencia, Google Health ofrece un amplio listado de enfermedades, aunque también ofrece la posibilidad de introducir una personalizada.

Figura 4-6 Vista de una enfermedad, con sus datos e información relativa a la enfermedad (drcha.)

Medications: lista todos los medicamentos que esté tomando o haya tomado el paciente con información de la posología, etc. (Figura 4-7). Google Health ofrece un enorme listado de medicamentos (ejemplo en la Figura 4-8) y, además, incluye una característica novedosa: avisa de las posibles interacciones que existan entre medicamentos que esté tomando el paciente.

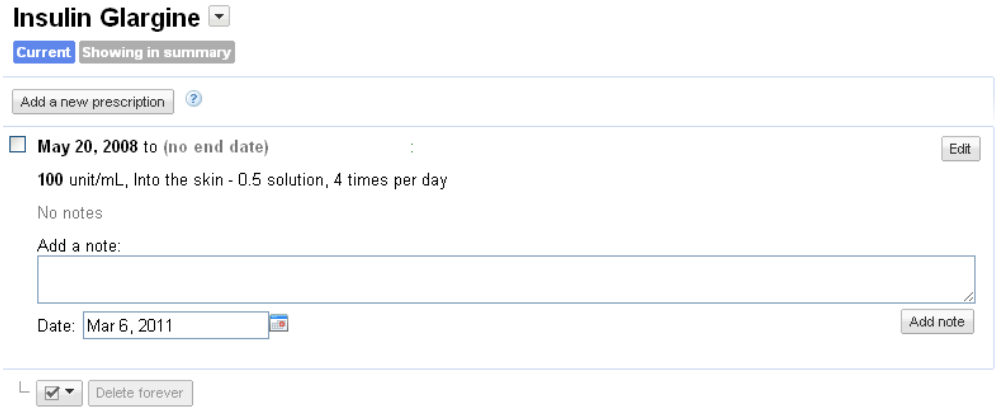


Figura 4-7 Información de un medicamento

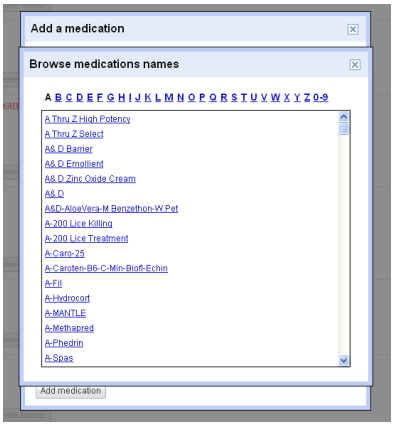


Figura 4-8 Listado con medicamentos ofrecido por Google Health

Allergies: contiene todas las alergias conocidas del paciente. Al igual que en Problems y en Medications, se ofrece un amplio listado de posibles alergias. Además, Google Health controla y, en su caso, avisa si el paciente es alérgico a algún componente de un medicamento que esté tomando.

Test Results: mediante esta categoría, se almacenan los datos de cualquier análisis realizado al paciente. Los resultados pueden verse a modo de tabla o en gráficas (como se aprecia en la Figura 4-9). También se ofrece un listado con los tipos de análisis más comunes.

Figura 4-9 Gráfica de glucemias en sangre

Procedures: permite registrar cualquier intervención (cirugías, etc.) realizada al paciente. Al igual que en las categorías anteriores, Google Health ofrece una largo listado de cirugías y procedimientos médicos.

Immunizations: registro de todas las vacunas administradas al paciente.

Insurance: guarda los datos de cualquier seguro q tenga contratado el paciente (seguro médico, seguro dental, etc.).

Files: permite almacenar ficheros (por ejemplo radiografías) de hasta 4 MB con un total de 100 MB por paciente.

Además del servicio de almacenamiento de información, Google Health ofrece varios servicios que usan esta información almacenada:

Colaboración de empresas de la salud

Importar datos médicos: algunas empresas y centros médicos permiten importar los datos médicos del paciente que tenían almacenados en el perfil de Google Health.

Explorar medicamentos y posibles tratamientos: Google Health permite compartir la información médica almacenada con las empresas que el paciente elija (siempre bajo consentimiento previo del paciente). El objetivo de este servicio es que las distintas empresas, clínicas y hospitales colaboradores ofrezcan al paciente posibles tratamientos y medicamentos para cualquier dolencia que padezca.

Digitalizar historiales en papel: existen 2 empresas que ofrecen al paciente digitalizar toda la información médica importándola en su perfil de Google Health.

Herramientas personalizadas: existen numerosas aplicaciones médicas que utilizan la información almacenada en el perfil de Google Health con distintos objetivos.

Exportar el perfil: numerosas empresas y centro médicos ofrecen la posibilidad de importar los datos del perfil de Google Health dentro de sus datos médicos del paciente.

Compartir el perfil: Otro de los servicios novedosos que ofrece Google Health es el de compartir el perfil del paciente con quien éste desee (su médico, un familiar, etc.). La persona invitada tendrá acceso de tipo sólo lectura para evitar que pueda modificar la información insertada por el paciente.

Formación del paciente: Ya que Google Health es una aplicación especialmente diseñada para el paciente, además del almacenar su información trata de informarle acerca de las enfermedades, medicamentos y demás conceptos que el paciente maneje. Para ello, utiliza la información introducida (de manera anónima) y muestra definiciones y artículos relacionados, buscando que el paciente no se limite sólo a introducir información si no que mejore sus hábitos de salud, comprenda mejor sus dolencias y amplíe sus conocimientos en general sobre la salud.

4.3 Resumen breve de la estructura de Google Health

4.3.1 Arquitectura de Google Health

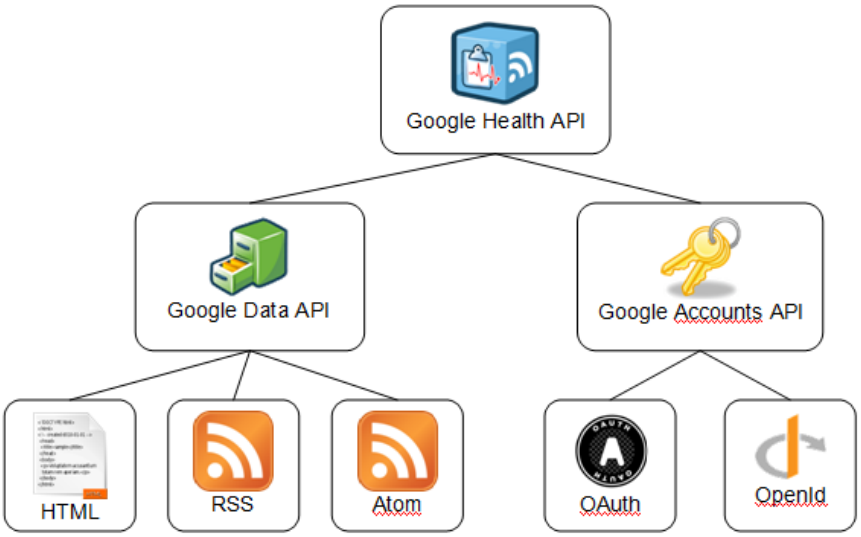


Figura 4-10 Pilares básicos de la estructura de Google Health

Aunque no podemos explicar específicamente la arquitectura de Google Health (ya que el propio Google no aporta esa información), podemos hacer una aproximación de los aspectos más importantes que servicios básicos sobre los que se implementa, esquema que podemos observar en la Figura 4-10.

Google Health, al igual que el resto de las aplicaciones ofrecidas por Google se construye en torno a los servicios ofrecidos por la interfaz de programación de aplicaciones (API en inglés) de Google Data (envío/recepción de datos, etc.).

Google Data API

El API de datos de Google proporciona los servicios básicos para leer e introducir datos en la Web siguiendo el Protocolo de Datos de Google. Por tratarse, en su mayoría, de servicios Web, el API de Google Data depende directamente de los estándares que son parte de la Web: HTTP y XML. En concreto, se basa en *Atom Publishing Protocol* (*AtomPub*), utilizando el formato de sindicación *Atom* estándar para representar datos y HTTP para administrar la comunicación (servicios GET y PUT). Además, amplía los servicios ofrecidos por AtomPub, introduciendo autenticación, consultas e incluso permitiendo elegir el formato de salida (JSON, RSS).

Para gestionar los servicios de autenticación y seguridad, Google Health y el resto de las aplicaciones desarrolladas por Google, hacen uso del API de Google Accounts.

Google Accounts API

La API de Google Accounts permite que las aplicaciones externas tengan un acceso limitado a la cuenta de Google de un usuario para ciertos tipos de actividad. Todas las solicitudes de acceso deben tener la aprobación del titular de la cuenta de Google. Actualmente, las API de cuentas de Google proporcionan autenticación para las siguientes actividades:

Intercambio de datos de usuario entre aplicaciones externas y servicios de Google,

Permiso para que los usuarios puedan acceder a aplicaciones externas mediante su cuenta de Google.

El API de Google Accounts facilita el proceso de autenticación de aplicaciones externas mediante un mecanismo de solicitud y recepción de la autenticación. Además, es totalmente compatible con la API de datos de Google. Para el acceso a los datos y la autenticación, Google Accounts ofrece los siguientes protocolos:

OAuth: se trata de un protocolo abierto desarrollado con el objetivo de ofrecer una vía de autenticación segura para aplicaciones de escritorio, móviles y Web. El API de Google Accounts ofrece dos versiones de este protocolo_

OAuth 1.0, versión estándar del protocolo y compatible con todas las APIs de Google.

OAuth 2.0 (estado experimental), versión simplificada del protocolo de autenticación compatible con todas las APIs de Google. OAuth 2.0 se basa en el protocolo criptográfico *Secure Sockets Layer* (SSL).

OpenId: estándar de identificación digital descentralizado, con el que un usuario puede identificarse en una página Web a través de una URL y puede ser verificado por cualquier servidor que soporte el protocolo. La versión de OpenId usada en Google Accounts está basada en el

protocolo OpenID 2.0, permite a los usuarios el acceso a tu sitio o aplicación Web mediante su cuenta de Google. Cuando Google autentica la cuenta de un usuario, devuelve una ID de usuario a tu aplicación, lo que te permite recopilar y almacenar la información de usuario.

Protocolo Híbrido: ofrece ambos sistemas de autentificación y autorización para aplicaciones Web, permitiendo a los usuarios conectarse y acceder a sus datos. Este protocolo usa *OpenID* para los servicios de autenticación y OAuth para los de autorización a las APIs de Google.

AuthSub: el API de AuthSub de Google ofrece una alternativa a OAuth con niveles de seguridad distintos. Auque es soportado por la mayoría de las APIs de Google, Google recomienda usar OAuth.

ClientLogin: proporciona a las aplicaciones de dispositivos móviles o de escritorio la capacidad de incorporar un inicio de sesión automático en su interfaz. ClientLogin es la alternativa preferible para enviar las credenciales de inicio de sesión de un usuario con cada solicitud, lo que permite un rendimiento mayor y más seguridad.

Google Health incluye otro estándar Web a los anteriores mencionados: el formato CCR. Éste formato fue desarrollado con el objetivo manejar electrónicamente la información sanitaria relevante de los pacientes. En realidad Google Health maneja solo un subconjunto del formato CCR, suficiente para cubrir los aspectos más significativos de la salud el usuario.

4.3.2 El formato CCR

CCR es la abreviatura de E 2369, especificación para la continuidad de registros de atención médica, la norma de ASTM International que busca avanzar hacia la información de atención médica portátil e interoperable con el resultado final de mejor atención médica para los pacientes y la reducción de los errores médicos (Enright, 2008).

La norma CCR es una norma flexible para crear documentos que contienen la información básica de las historias clínicas con uniformidad. Con este estándar se tendrá la información más relevante y oportuna acerca de un paciente, y permitirá a uno o varios ver el estado de éste médicos de manera electrónica.

El programa CCR busca evitar a médicos y demás profesionales la salud tener que actuar “a ciegas” por no tener acceso fácil a la información relevante del paciente. Ofrecerá la información necesaria para mantener la continuidad del cuidado, con lo que se reducirán los errores médicos, mejorará la eficiencia y se dará un cuidado de mayor calidad (Tessier, 2011).

La norma CCR se desarrolló bajo el control de la Organización de Desarrollo de Estándares (SDO) de ASTM-International. Participaron distintos organismo tales como *American Academy of Family Physicians*, *The*

Massachusetts Medical Society, *The American Academy of Pediatrics* y *The American Osteopathic Association*.

El grupo encargado de desarrollar el estándar CCR buscaba capturar la información más relevante de los pacientes y decidió enfocar el trabajo hacia un estándar XML y desarrollarlo dentro de la normal ASTM E31. Se comenzó entonces a definir el contenido de la norma CCR. Su expresión XML tenía que estar de acuerdo con las normas generales de la industria de la informática y cumpliendo la normativa ANSI. La primera versión del CCR salió a finales de 2005 y en 2009 salió la segunda versión. Actualmente se sigue trabajando para mejorar el estándar CCR.

Un CCR potencialmente se puede crear, leer e interpretar por cualquier HME o aplicación de software *EMR*. El *CCR* también se puede exportar en otros formatos, como *PDF* y *Open Office XML* (formato de Microsoft Word 2007).

En la Figura 4-11 podemos observar la estructura de un documento CCR:

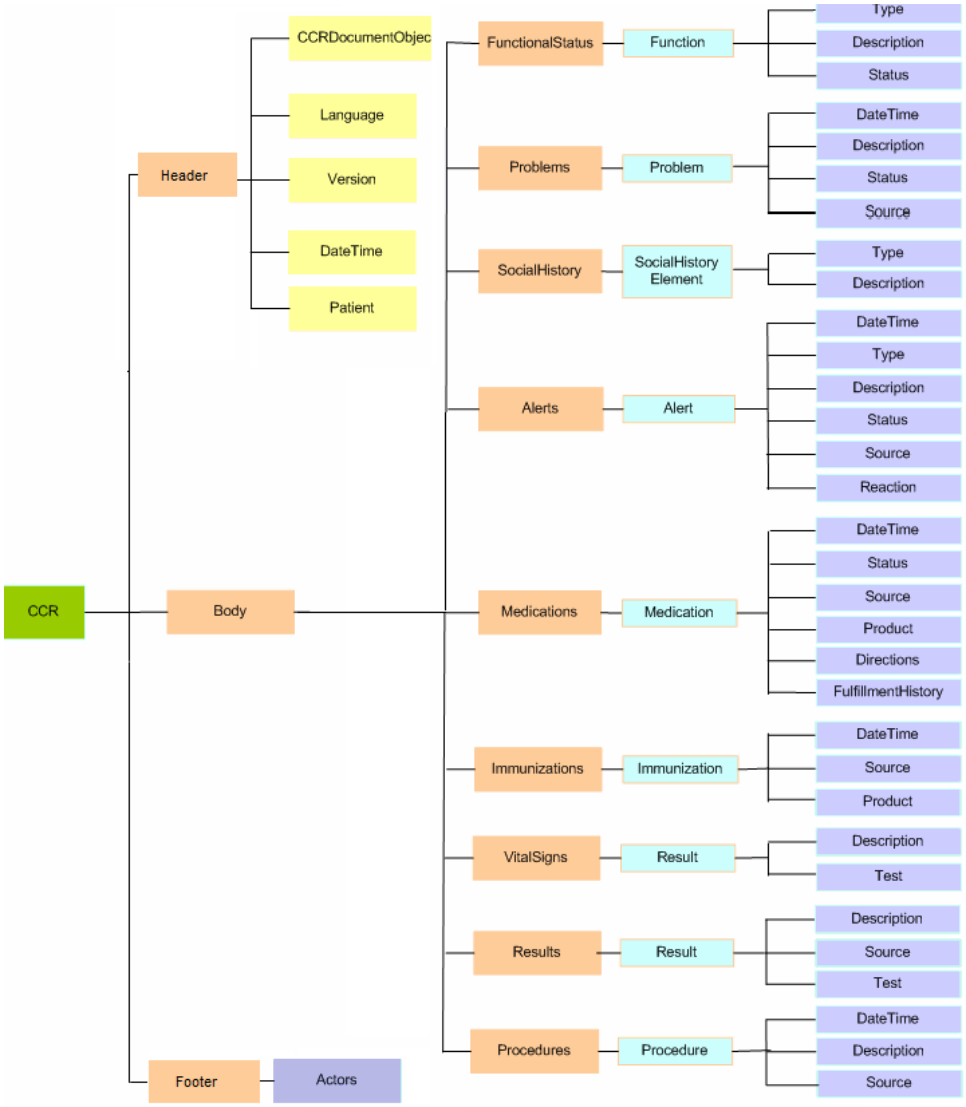


Figura 4-11 Estructura del documento CCR para Google Health

Cabecera: identifica el formato y la fecha de creación del CCR. La cabecera consiste en un documento XML que está constituido por los siguientes elementos:

- CCRDocumentObjectID
- Language
- Version
- DateTime
- Patient.

El componente <Body> es donde se encuentran todos los detalles de la historia médica del paciente. Está formado por los siguientes elementos secundarios:

- *FunctionalStatus* formado por el elemento *Function*.
- *Problems* formado por el elemento *Problem*.
- SocialHistory formado por el elemento SocialHistoryElement.
- *Alerts* formado por el elemento *Alert*.
- Medications formado por Medication.
- *Immunizations* formador por el elemento *Immunization*.
- VitalSigns
- *Results* formados por el elemento *Result*.
- *Procedures* formado por los elemento *Procedure*

El pie de página se compone de información demográfica básica sobre el paciente (nombre, fecha de nacimiento, y género). El pie de página se compone de un único elemento: *Actors*.

En la siguiente imagen observamos el diagrama de apoyo para el desarrollo del CCR. En este diagrama los elementos a la derecha de la rama de una en el árbol son extensiones de los elementos a su izquierda.

El *Continuity Of Care Record* está formado por elementos simples. Estos elementos dan forma a todos los detalles de la historia médica del paciente que nos encontramos en el cuerpo del CCR. Estos elementos son los siguientes:

Type

Sintaxis:

```
<Type>
    <Text>...</Text>
</Type>
```

Descripción:

Type es una cadena de caracteres (String) que puede tomar cualquier valor. Type es un elemento de DateTime, y se le asignar los siguientes valores:

Collection start date, Dispense date, Start date, Stop date, Prescription date, Primary Health Insurance, Supplemental Health Insurance, Prescription Drug Benefit, Dental Insurance, Vision Insurance, Other, Plan code, Group Number, Subscriber Number.

Ejemplo:

```
<Type>
    <Text>Collection start date</Text>
</Type>
```

Description

Sintaxis

```
<Description>
    <Text>...</Text>
    <Code>
        <Value>...</Value>
        <CodingSystem>...</CodingSystem>
    </Code>
</Description>
```

Descripción*:*

Description está formado por 2 atributos: (1) Text que es una cadena de caracteres ,que tomará un valor u otro en función del módulo al que represente, (2) Code es una lista de CodeType, el CodeType está formado por: (1) Value que es cualquier cadena numérica o alfanumérica y (2) CodingSystem que es cualquier cadena de caracteres.

Ejemplo:

```
<Description>
    <Text>Diabetes</Text>
    <Code>
        <Value>250.0</Value>
        <CodingSystem>ICD9</CodingSystem>
    </Code>
</Description>
```

Status

Sintaxis:

```
<Status>
    <Text>...</Text>
</Status>
```

Descripción*:*

Status es cualquier cadena de caracteres. Por ejemplo: activo, pendiente, etc.

Ejemplo*:*

```
<Status>
    <Text>Active</Text>
</Status>
```

DateTime

Sintaxis

```
<DateTime>
    <Type><Text>...</Text></Type>
    <ExactDateTime>...</ExactDateTime>
</DateTime>
```

Descripción

Fecha y hora indica cuándo y produjo el evento. Por ejemplo, cuando el Type = "Start Date" el valor de ExactDateTime indica cuando el paciente comenzó a tomar el medicamento, comenzando con un problema, ejercicio físico…

Ejemplo

```
<DateTime>
    <Type><Text>Start date</Text></Type>
    <ExactDateTime>2007-04-04T07:00:00Z</ExactDateTime>
</DateTime>
```

Source

Sintaxis

```
<Source>
    <Actor>
        <ActorID>...</ActorID>
        <ActorRole>...</ActorRole>
    </Actor>
</Source>
```

Descripción

Source está formado por una lista de actores. A su vez un actor está formado por: (1) ActorID. Es el nombre y apellidos, (2) Lista de Roles. Especifica la relación entre el médico y el paciente. Los valores admitidos son: Ordering clinician, Prescribing clinician, Treating clinician

Ejemplo

```
<Source>
    <Actor>
        <ActorID>Cecil Baker</ActorID>
        <ActorRole><Text>Ordering clinician</Text></ActorRole>
    </Actor>
</Source>
```

Reaction

Sintaxis

```
<Reaction>
    <Severity>...</Severity>
</Reaction>
```

Descripción

En el campo Reaction se va a indicar el nivel de gravedad. Los únicos términos Google mostrará, o aceptar como entrada, son " Mild"(suaves) o “Severe” (grave).

Ejemplo

```
<Reaction>
    <Severity><Text>Severe</Text></Severity>
</Reaction>
```

Product

Sintaxis

```
<Product>
    <ProductName>
        <Text>...</Text>
        <Code>
            <Value>...</Value>
            <CodingSystem>...</CodingSystem>
        </Code>
    </ProductName>
    <Strength>
        <Value>...</Value>
        <Units><Unit>...</Unit></Units>
    </Strength>
    <Form><Text>...</Text></Form>
</Product>
```

Descripción

Este campo está formado por: (1)Product name que es una cadena de caracteres, en la que se introduce el nombre del medicamento, (2) lista de Strength que es una cadena de caracteres ,donde se indican las dosis del medicamento y(3) Lista de Form que es una cadena de caracteres ,en las que se indican las formas del medicamento.

Ejemplo

```
<Product>
    <ProductName>
        <Text>Ibuprofen</Text>
        <Code>
            <Value>198405</Value>
            <CodingSystem>RxNorm</CodingSystem>
        </Code>
    </ProductName>
    <Strength>
        <Value>100</Value>
        <Units><Unit>mg</Unit></Units>
    </Strength>
    <Form><Text>Tablet</Text></Form>
</Product>
```

Directions

Sintaxis

```
<Direction>
    <Dose>...</Dose>
    <Route>...</Route>
    <Frequency>...</Frequency>
</Direction>
```

Descripción

Directions está formada por : (1)Lista de Dosis, una dosis está formada por : (1)Valor: los posibles valores de este elemento son {"25" | "0,5" | "1" | "1,5" | "2" | "3" | ... cualquier número positivo} y (2) Unidades: los posibles valores pueden ser mg, tablet.(2) Lista de Rutas son los distintos valores de la manera de tomar el medicamento, como por ejemplo: "oral", "sobre", etc…(3) Lista de Frecuencia que son los distintos valores de frecuencia de tomar un medicamento. Puede tener los valores de, "un tiempo ", "una vez al día.", "cada 12 horas", etc.

Ejemplo:

```
<Directions>
    <Direction>
        <Dose><Value>1</Value><Units><Unit>tablet</Unit></Units></Dose>
        <Route><Text>Oral</Text></Route>
        <Frequency><Value>1 time per day</Value></Frequency>
    </Direction>
</Directions>
```

FulfillementHistory

Sintaxis

```
<FulfillmentHistory>
    <Fulfillment>
        <Quantity>
            <Value>...</Value>
            <Units>...</Units>
        </Quantity>
        <DateTime>...</DateTime>
    </Fulfillment>
</FulfillmentHistory>
```

Descripción

FulfillmentHistory está formado por: (1) Lista de Cantidades: que a su vez está formado por 2 elementos: (1) Valor los números positivos a partir del 0 y (2) Unidades que son los distintos valores de la manera de tomar el medicamento, como por ejemplo: "oral", "sobre", etc. FulfillmentHistory también está formado por (2) DateTime.

Ejemplo

```
<FulfillmentHistory>
    <Fulfillment>
        <Quantity>
            <Value>30</Value>
            <Units><Unit>Tablet</Unit></Units>
        </Quantity>
        <DateTime>
            <Type><Text>Dispense date</Text></Type>
            <ExactDateTime>2007-05-02T07:00:00Z</ExactDateTime>
        </DateTime>
    </Fulfillment>
</FulfillmentHistory>
```

Test

Sintaxis

```
<Test>
    <DateTime>...</DateTime>
    <Description>...</Description>
    <Source>...</Source>
    <NormalResult>
        <Value>...</Value>
        <Units>...</Units>
    </NormalResult>
    <TestResult>
        <Value>...</Value>
        <Units>...</Units>
    </TestResult>
</Test>
```

Descripción

Un Test está formado por: (1) DateTime, (2) Description (3) Source, (4) NormalResult y (5) TestResult que están formado por Valor y Unidades.

Ejemplo

```
<Test>
    <DateTime>
        <Type><Text>Collection start date</Text></Type>
        <ExactDateTime>2007-02-21</ExactDateTime>
    </DateTime>
    <Description>
        <Text>Spun hematocrit</Text>
        <Code>
            <Value>4545-0</Value>
            <CodingSystem>LOINC</CodingSystem>
        </Code>
    </Description>
    <TestResult>
        <Value>40</Value>
        <Units><Unit>%</Unit></Units>
    </TestResult>
    <NormalResult>
        <Value>40</Value>
        <Units><Unit>%</Unit></Units>
    </NormalResult>
</Test>
```

La cabecera está constituida por los siguientes elementos:

CCRDocumentObjectID

El CCR y todos los objetos de datos contenidos en el CCR deben tener un ObjectIDs. Este CCR de objetos de documento sirve para identificar de forma única cada instancia explícita de un CCR. La singularidad de este ObjectID es que se define cuando se genera en el sistema, que es único y preferiblemente, debe ser único en el universo de todos los CCR.

Sintaxis:

```
<CCRDocumentObjectID>...</CCRDocumentObjectID
```

Language

Sintaxis:

```
<Language>
   <Text>English</Text>
   <Code>
      <Value>en</Value>
      <CodingSystem>ISO-639-1</CodingSystem>
   </Code>
</Language>
```

Version

Sintaxis:

```
<Version>V1.0</Version>
```

DateTime

Cuando una fecha se envía de Google, esta fecha representa la fecha en que el documento fue modificado por última vez.

Sintaxis:

```
<DateTime>
   <ExactDateTime>2007-01-07T13:00:00-0500</ExactDateTime>
</DateTime>
```

Patient

El paciente es un marcador de posición para el identificador de la persona vinculada con el CCR.

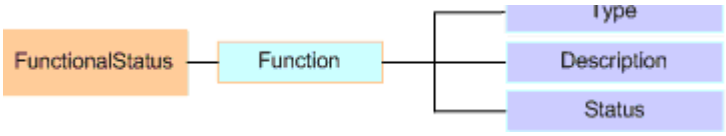
Sintaxis:

```
<Patient>
   <ActorID>Patient</ActorID>
</Patient>
```

El cuerpo del CCR está formado por:

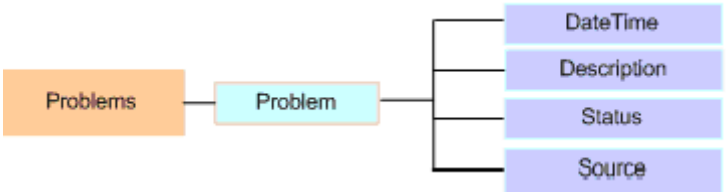
FunctionalStatus

Usado para regsitrar estados transitorios del paciente (embarazo, amamantando, etc.). Formado por elementos de tipo Function, que asu vez están compuesto por un Type, un Description y un Status.



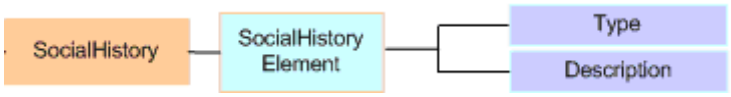
Problems

Esta sección se usará para registrar enfermedades y dolencias del paciente. Estará formado por elementos de tipo **<Problem>** descritos mediante una fecha de inicio de la enferemedad (<**DateTime**>), una descripción <**Description**>, un estado <**Status**> y actores relacionados <**Source**>.



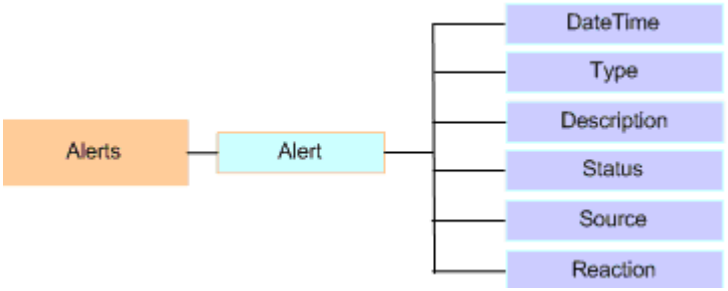
SocialHistory

Esta sección sólo se usará para indicar la raza del paciente. Esta compuesta por elementos de tipo <**SocialHistoryElement**> descritos mediante un <**Type**> y un <**Description**>.



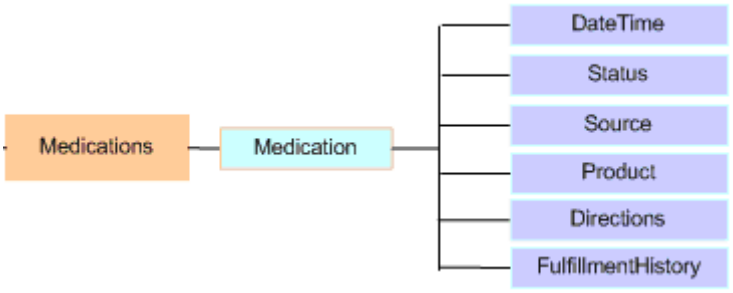
Alerts

Usada para indicar cualquier alergia o evento a tener en cuenta. Cada elemento de tipo <Alert> tendrá una fecha (<**DateTime**>) , un tipo (<**Type**>) , un estado (<**Status**>) , una descripción (<**Description**>), una reacción (<**Reaction**>) y un conjunto de actores relacionados (<**Source**>).



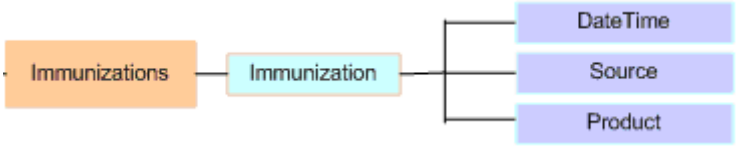
Medications

Listado de los medicamentos que el paciente esté tomando o haya tomado. Cada elemento de tipo **<Medication>** vendrá descrito por un estado (<**Status**>), una fecha de prescripción (<**DateTime**>) , una descripción del producto (<**Product**>) , unas indicaciones de toma (<**Directions**>), un registro de las prescripciones (<**FulfillmentHistory**>) y un conjunto de actores relacionados (<**Source**>).



Immunizations

Vacunas y tratamientos recibidos por el paciente. Cada elemento <**Inmunization**> vendrá descrito por una fecha (<**DateTime**>) , una descripción del producto (<**Product**>) y un conjunto de actores relacionados (<**Source**>).

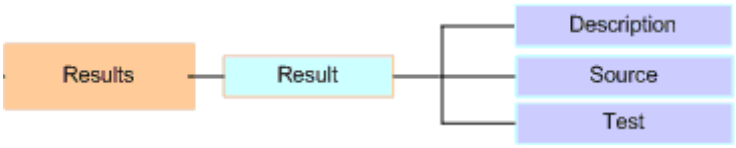


VitalSigns

Se usa para registrar medidas tales como la altura, el peso etc. Estará formado por uno o varios elementos de tipo <**Result**> formados a su vez por elementos de tipo <**Test**>. El objetivo es agrupar dentro de cada <**Result**> resultados de test distintos para una misma prueba (ej. todas las medidas de peso estarían juntas dentro de un <**Result**>, las de altura en otro <**Result**>,...).

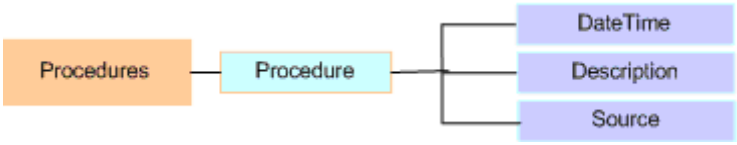
Results

Igual que <**VitalSigns**> pero para registrar otro tipo de pruebas (analíticas, glucemias, etc.).



Procedures

Registrará mediante elementos de tipo <**Procedure**> aquellas operaciones que haya sufrido el paciente. Cada elemento <**Procedure**> estará formado por una fecha <**DateTime**>, una descripción <**Description**> y un conjunto de actores relacionados (<**Source**>).



El pie de página está formado por:

Actors

Sintaxis:

```
<Actor>
    <ActorObjectID>...</ActorObjectID>
    <Person>
        <DateOfBirth>...</DateOfBirth>
        <Gender>...</Gender>
    </Person>
</Actor>
```

Descripción:

Un Actor está formado por: (1) ActorObjectId este es el id del paciente, (2) Person, una persona está formada por:DateOfBirth, Gender, Name.

Ejemplo:

```
<Actors>
    <Actor>
        <ActorObjectID>Patient</ActorObjectID>
        <Person>
            <DateOfBirth><ExactDateTime>1919-09-21</ExactDateTime></DateOfBirth>
            <Gender>
                <Text>Male</Text>
                <Code>
                    <Value>248153007</Value>
                    <CodingSystem>SNOMED</CodingSystem>
                </Code>
            </Gender>
        </Person>
        <Source><Actor><ActorID>Site</ActorID></Actor></Source>
    </Actor>
</Actors>
```

4.3.3 API de Google Health para Java

Como en la mayoría de sus productos, Google ofrece a los desarrolladores distintas APIs con las que acceder a los datos de Google basadas en sus APIs GData y GAccounts. El API de Google Health permite a las aplicaciones ver y enviar datos de Google Health como *feeds* del API de datos de Google y está implementada para distintos lenguajes de programación: Protocol, .NET, Java, PHP y Python. En esta sección nos centraremos en la versión para Java.

El API de Google Health para Java contiene todas aquellas funcionalidades y tipos de datos necesarios para enviar/recibir información a Google Health.

Autentificación al servicio "Health"

El API de Google Health contiene el código necesario para realizar la conexión con el servicio Google Health usando los protocolos de seguridad anteriormente descritos en el Apartado 4.3.1: AuthSub Authentication (para aplicaciones Web) y ClientLogin para aplicaciones de escritorio.

Ej. Código de autentificación usando ClientLogin

```
HealthService healthService = new HealthService("HealthSample-1.0");
healthService.setUserCredentials("user@domain.com", "secretPassword");
```

Interaccionar con los datos del usuario

Una vez autentificados, podremos comenzar a interaccionar con los datos del usuario de Google. Un usuario puede tener registrados datos médicos de varios sujetos y, por tanto, los datos almacenados en Google Health se archivan según el sujeto al que pertenecen, encapsulados en una entrada

Atom. Cada entrada Atom estará identificada mediante un identificador alfanumérico único y un nombre del sujeto representado.

Ej. Obtención de los nombres e identificadores de todos sujetos registrados en la entrada (*Entry*) de tipo Atom para el usuario autentificado mediante ClientLogin

```
Feed profileListFeed = healthService.getFeed(new
URL("https://www.google.com/health/feeds/profile/list", Feed.class);
List<Entry> entries = profileListFeed.getEntries();
for (Entry profileListEntry : entries) {
    System.out.println("Profile name: " +
profileListEntry.getTitle().getPlainText());
System.out.println("Profile id: " + profileListEntry.getPlainTextContent());
}
```

Además, usando la librería de Java podremos extraer la información médica del paciente que se nos devolverá en un documento CCR.

Ej. Obtención del documento CCR para el primer sujeto registrado para el usuario

```
// Select the user's first profile
String firstProfileID = entries.get(0).getPlainTextContent();
String url = "https://www.google.com/health/feeds/profile/ui/" + firstProfileID;
ProfileFeed profileFeed = healthService.getFeed(new URL(url), ProfileFeed.class);
for (ProfileEntry entry : profileFeed.getEntries()) {
    System.out.println(entry.getContinuityOfCareRecord().getXmlBlob().getBlob());
}
```

El API de Google Health para Java ofrece además otras funcionalidades tales como: *queries* (consultas con parámetros configurables por el desarrollador), enviar nuevas Entries (nueva información médica del paciente en formato CCR y encapsuladas en entradas Atom) y actualizar o borrar datos ya registrados (para más información consultar el APÉNDICE I Manual API de Google Health para Java).

Debido a que Google Health se encuentra actualmente en su fase Beta, la API de Google Health se encuentra todavía en desarrollo y su funcionalidad en algunos casos está limitada o no funciona correctamente como por ejemplo la imposibilidad, hasta la fecha, de crear nuevos sujetos desde la API, limitaciones al actualizar los datos, etc. (para más información consultar el APÉNDICE II Bugs de Google Health).

Por último, comentar la existencia de la biblioteca CCR4J (CCR for Java), proyecto en desarrollo que ofrece una API para facilitar el manejo de la CCR información contenida en documentos XML.

4.4 Problemas de seguridad de Google Health

4.4.1 Problemas propios de los PHR

Obviando los problemas propios de los sistemas informáticos (accesibilidad, almacenamiento, etc.) el principal escollo con que se encuentra el desarrollo y la, futura, aplicación de los programas PHR en los sistemas sanitarios es la privacidad. Para los médicos el deber de secreto de la información relacionada con los pacientes es tan antiguo como su profesión, como muestra el que esa obligación se encuentre presente entre las descritas en el Juramento Hipocrático. En la actualidad el deber de guardar secreto profesional afecta a todo personal, tanto sanitario como no sanitario, que se relaciona con los pacientes o que accede a la información relacionada con ellos. En concreto en España, ese deber se refleja en la Normativa de Protección de Datos, la Normativa Sanitaria e incluso en el código penal. Garantizar esta seguridad y confidencialidad de los datos y, al mismo tiempo, ofrecer acceso a la información clínica desde lugar en el que pueda ser necesario para la debida atención del paciente, son dos objetivos que a priori parecen difíciles de compaginar. Las dificultades encontradas podrían clasificarse en tres categorías distintas:

Técnicas: generadas ante la necesidad de ofrecer sistemas 100% seguros y estables a prueba de cualquier amenaza humana (voluntaria e involuntaria) o catástrofe (como incendios, daños del sistema, etc.). Aunque actualmente existen mecanismos de seguridad muy desarrollados y compactos, ninguno puede considerarse inmune a posibles vulnerabilidades, lo cual conlleva que para una supuesta aplicación de los EHR, sea necesaria una gran inversión en el diseño de un sistema seguro y estable.

Organizativa: dado que el objetivo del EHR es poder ofrecer acceso a la información clínica del paciente a todo aquel personal sanitario que lo necesite, es conveniente que éste disponga de los medios y formación necesaria para manejarlo y además que exista una jerarquía bien definida de roles para limitar los accesos y derechos sobre la información.

Legales: la legislación vigente sobre confidencialidad y seguridad es muy estricta y especialmente crítica cuando se trata de información clínica. Conseguir que el sistema desarrollado garantice la legalidad es imprescindible para que pueda llegar a ser aplicable.

4.4.2 Seguridad de Google Health

Postura oficial

Google, a diferencia de otras aplicaciones PHR, no firma el convenio HIPAA sobre confidencialidad de la información sanitaria redactado por el congreso de los EEUU (ver punto 3). En su lugar, Google ofrece su propia

Política de privacidad que el la propia compañía resume de la siguiente manera (traducción):

"Creemos que su información clínica pertenece a usted, por lo que usted es quien tiene el control. [...] Así es como mantenemos sus datos seguros:

- *Nunca venderemos sus datos.*
- *Su información sanitaria está almacenada en su cuenta segura y no puede ser accedida por otros a través de búsquedas en Google.*
- *Usted elige con quién desea compartir la información.*
- *Si usted da acceso a alguien a su cuenta, podrá revocar este acceso en cualquier momento.*
- *Disponemos de tecnología de última generación y algunos de los mejores expertos en seguridad del mundo asegurando sus datos en Google Health. [...]" (*Google Inc, 2011, traducción*)*

Además Google hace la siguiente afirmación refiriéndose a aquellas terceras personas que pudieran tener acceso a la información clínica (empresas asociadas, servicios online, etc. ver apartado 4.2.2):

"Todos los productos y servicios de Google, incluyendo Google Health, se rigen por nuestra política de privacidad, lo que explica la forma en que tratamos la información personal. Además, hemos creado una política que describe las prácticas de privacidad específicas de Google Health.

[...]Aquellos servicios sanitarios online integrados en Google Health debe cumplir con nuestras políticas de Google para desarrolladores, que establecen normas de privacidad estrictas de cómo recoger, utilizar y compartir la información. Depende de usted si desea compartir su información de salud con alguno de estos servicios." (Google Inc. 2011, traducción)*.*

Por tanto Google afirma disponer de un sistema totalmente seguro y que garantiza la confidencialidad de los datos, siendo el propio usuario el único con capacidad de elegir con quién compartir su información.

El convenio HIPAA y Google Health

A diferencia de otras aplicaciones similares, Google Health no está regulado por la *Health Insurance Portability and Accountability Act* (HIPAA), una ley federal que establece las normas de confidencialidad de los datos de información de salud del paciente aprobada por el Congreso de EEUU en 1996. Pese a las numerosas críticas que esta postura ha acarreado, Google se escuda en que Google Health no almacena datos para profesionales sanitarios, si no para el propio usuario.

Aunque Google Health no está cubierto por la HIPAA, la empresa afirma estar comprometida con la privacidad de los usuarios, contar con estrictas políticas de seguridad de datos y asegurar que sean sólo los usuarios quienes controlen el acceso a su información. Además aseguran ser transparentes sobre cuál es la información que recopila cuando se usa Google Health y de

cómo va a ser usada y se asegura que no se venderá o compartirá la información salvo autorización expresa del usuario.
Además, a diferencia de otras herramientas de Google, no hay publicidad en Google Health. Para completar su defensa, Google aporta documento en el que analiza las distintas prácticas de confidencialidad de Google Health comparándolas con las establecidas en la HIPAA (ver Anexo I Google Health and HIPAA).

Vulnerabilidades de Google

Pese a que asegura en su Política de Privacidad que la información almacenada de la cuenta Google es 100% segura, la realidad ha demostrado que no siempre es así: los servicios Google se ven continuamente atacados en un intento de aprovechar posibles vulnerabilidades y extraer información confidencial de los usuarios. Hasta la fecha, muchos de estos ataques tienen fines meramente comerciales (intereses de los usuarios, crear listas de correos, etc.) aunque otros resultan realmente inquietantes (por ejemplo el ataque sufrido en 2009 contra cuentas de Gmail de disidentes chinos y respaldado por el gobierno chino) (Rizzi, Prados, 2010).

Ante esta situación y consciente de que la información médica del usuario es especialmente delicada, Google ha invertido mucho esfuerzo en desarrollar nuevos protocolos de seguridad y en mejorar lo ya existente, con el fin de reducir al mínimo las posibilidades de sufrir un ataque con éxito.

Por otro lado, existe gran debate acerca de las obligaciones de Google respecto a la información que almacena. Pese a que en su contrato con el usuario y a las condiciones de su Política de Privacidad Google asegura que los datos nunca serán vendidos a terceros, nada le impide cambiar estas condiciones en un futuro. Además, en muchos de sus servicios incorpora publicidad, adaptada al usuario por medio de los datos que previamente se han extraído y procesado. Si esto ya provocaba cierta desconfianza en los usuarios de servicios como *Gmail* o *Google Docs*, con Google Health esta sensación es mayor, pues al no someterse a las condiciones de la *HIPAA*, nada exime a Google de cambiar las condiciones de privacidad de este servicio.

4.5 Conclusiones: ¿Por qué Google Health?

Pese a todo lo explicado en los anteriores apartados acerca de la seguridad y la privacidad de la información almacenada en Google, nuestra aplicación se construye sobre este servicio usándolo como herramienta para el almacenaje de datos. Las razones son varias:

Google Health, al igual que la mayoría de los servicios ofrecidos por Google, es una herramienta muy potente y que, con mucha seguridad, se mantenga como gratuita. No se puede decir lo mismo de las aplicaciones del

principal competidor Microsoft que no suele ofrecer servicios gratuitos de ningún tipo.

Otra de las razones de más peso es el dinero. Desarrollar y mantener un sistema, especialmente en aquellos destinados al mundo de la salud, estable y muy seguro exige una gran inversión. Empresas como Google y Microsoft pueden afrontar esta situación perfectamente, sin embargo, otros proyectos como el caso de Dossia, que dependen directamente de lo que estén dispuestos a donar las empresas colaboradoras no pueden garantizar los recursos mínimos necesarios.

Además, Google Health supera con creces a sus competidores en el número de empresas colaboradoras y servicios opcionales que ofrece, lo cual la convierte en la aplicación más grande y potente del sector.

Desde el punto de vista técnico es importante destacar que Google Health es, junto con Dossia, el único sistema tipo PHR que ofrece un API para el desarrollo de aplicaciones, aunque a diferencia de Dossia, la información almacenada sigue un formato estándar (estándar CCR) aceptado ampliamente para la digitalización de información sanitaria.

5 Implementación de GluControl

5.1 Descripción de la aplicación.

Existen dos tipos de usuarios que van a utilizar GluControl: pacientes y médicos. En función del tipo de usuario que vaya a utilizar la aplicación tendrá acceso a unas determinadas opciones u otras. Es por ello, por lo que podríamos dividir la aplicación en dos "*subaplicaciones*" no disjuntas. Una de las partes es la que utilizará el médico, la otra, la que utilizarán los pacientes.

La aplicación para el médico es más extensa y presenta numerosas funcionalidades que el paciente no necesita. El médico tendrá acceso al historial médico del paciente, podrá añadir medicamentos, enfermedades, analíticas, etc.

Por otra parte, la aplicación para el paciente está orientada a la introducción y seguimiento de medidas diarias de glucosa, comidas, insulina inyectada y fondo de ojo; así como la consulta del estado de sus tratamientos y enfermedades actuales. El objetivo es que el paciente no se vea sobrecargado con información no necesaria y se centre en llevar un buen seguimiento de sus medidas, que al fin y al cabo es lo más importante para la diabetes.

Como puede observarse en la Figura 5-1 la propia aplicación, tanto la dedicada al médico como la del paciente, podría dividirse a su vez en dos módulos independientes. El primero de los módulos se encargará de la conexión con *Google Health* (desde ahora, GH), el otro, se encargará de la lógica de la aplicación.

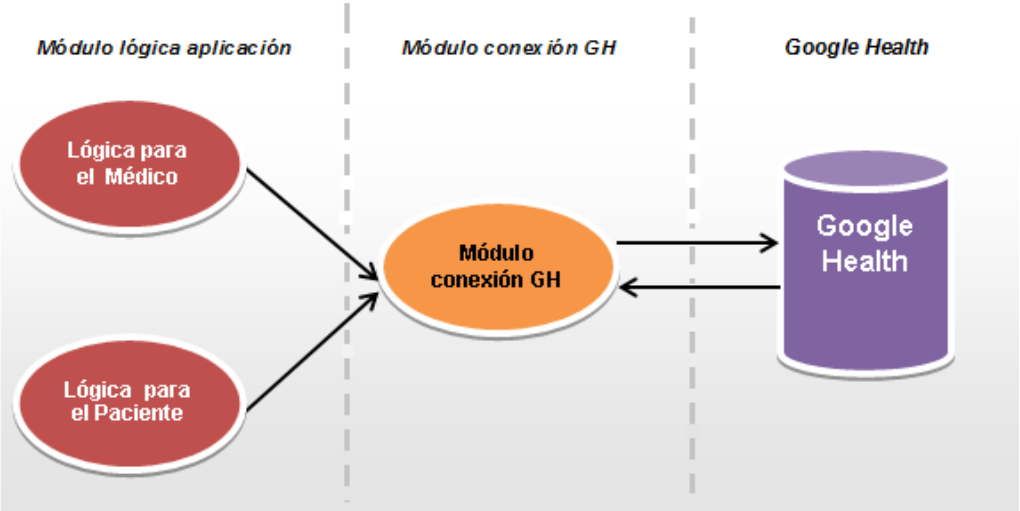


Figura 5-1 Visión global de los módulos que forman GluControl

El objetivo de esta distribución es hacer que la aplicación sea portable de forma sencilla a otros proveedores de servicios de salud como, por ejemplo, *Microsoft HealthVault* Además, de esta manera, cualquiera podría reutilizar el

módulo de conexión con GH pudiendo dedicarse exclusivamente al desarrollo de la lógica de su aplicación. Por lo tanto, no sólo hemos desarrollado un API para la conexión con Google Health, sino también una aplicación para la gestión de la diabetes. Ambos módulos se describirán detalladamente en las secciones 5.2 y 5.3 respectivamente.

El desarrollo de la *Interfaz Gráfica de Usuario* (GUI) para la aplicación intenta simular un entorno Web. El objetivo es que cualquier tipo de persona tenga los conocimientos informáticos que tenga, se sienta cómoda utilizando la aplicación. Con el incremento de acceso a Internet en los hogares, la mayoría de las personas han navegado por Internet. Con la GUI de GluControl se sentirán cómodos y familiarizados con el estilo.

Otro aspecto importante es el proceso de autenticación en la aplicación. En Google Health varios usuarios pueden compartir la información de pacientes. Sin embargo sólo uno puede tener acceso para cambiar e introducir datos. Esto supone un problema ya que nosotros deseábamos que tanto paciente como médico pudieran introducir/modificar datos sobre el paciente. Una de las soluciones podría ser que el médico proporcionara su cuenta y contraseña a todos los pacientes. Esto causaría un grave problema de privacidad ya que cualquier paciente podría tener acceso al historial médico de cualquier paciente del mismo médico. Esta opción por lo tanto, fue desechada. Para solventar este problema hemos diseñado una solución utilizando una base de datos auxiliar. El funcionamiento es el siguiente:

Se conecta en la aplicación utilizando su nombre de paciente y una contraseña proporcionada con anterioridad. La aplicación accede a la base de datos externa y si los datos son correctos obtendrá tanto la cuenta de su médico como su contraseña. Una vez tenga los datos, automáticamente se conectará a GH con los datos de médico y cargará los datos del paciente que se ha autentificado. De esta forma el paciente tendrá las mismas credenciales que el médico pero no tendrá acceso a datos privados de otros pacientes. Este proceso se describe detalladamente en la sección 5.5.

5.2 Módulo de conexión con Google Health

5.2.1 Descripción del módulo

El módulo de conexión con Google Health, con nombre Google Health Manager, es el conjunto de paquetes y clases encargado de gestionar la conexión con Google Health.

Podría definirse como un mediador entre la aplicación y GH. Éste módulo es independiente y podrá ser utilizado por cualquier aplicación Java para intercambiar datos con GH. Para implementar este módulo se ha utilizado la *Interfaz de Programación de Aplicaciones* (API) para Java que el propio GH proporciona. Sin embargo, este API está en fase Beta, y tanto el desarrollo del

código de las funcionalidades del API como la documentación están aún en fase de experimentación. Este hecho ha dificultado la labor del módulo.

La arquitectura del módulo ha sido diseñada teniendo en cuenta que el tipo de datos que se recopilan desde GH siguen el estándar de salud *Continuity Care of Record* CCR (explicado en detalle en la sección 4.3.2). Cada uno de los campos del formato CCR que se utilicen tendrá una representación en nuestra aplicación en forma de objeto con el mismo nombre y atributos similares. Por ejemplo, en el formato CCR existe un campo denominado *DateTime*, este campo es utilizado para almacenar una fecha. En nuestra aplicación existe una clase, en el paquete *Clases Auxiliares, DateTime* que representará un campo *DateTime* del CCR.

Por lo tanto el paquete Clases Auxiliares contiene una serie de clases que servirán para representar campos CCR de los objetos de GH. Por otra parte, para el acceso a GH se ha utilizado el patrón *Data Access Object (*DAO*)*. Este patrón es utilizado para homogeneizar el acceso a la base de datos. Cada uno de los elementos que se almacenen en GH tendrá una clase *objetoDAO* a través de la cual se realizará la petición. De esta forma, la petición a GH se realiza a través del su objetoDAO. Por otra parte se ha utilizado el patrón Data *Transfer Object (*DTO*)* para representar objetos almacenados en GH. Por ejemplo, en GH se almacenan medicamentos, por lo tanto, habrá una clase medicamentoDTO que contendrá como atributos los campos que contiene un medicamento (sirviéndose de las clases existentes en el paquete Clases Auxiliares anteriormente nombrado). En la sección 5.2.2 se hará un análisis más profundo de cada uno de estos paquetes y clases.

Otro componente de este módulo es el Conector. Dicha clase es la encargada de realizar la petición en última instancia a GH. Es decir, es la única con acceso directo a GH. Cualquier petición desde una clase DAO utilizará el conector para intercambiar datos GH. Es el conector el que utiliza el API de Java que proporciona Google para el acceso a los datos de GH.

Sin embargo, el conector utiliza código CCR mientras que los objetos DAO funcionan con objetos Java. Para realizar esta traducción CCR-objeto Java utilizamos el paquete *Interpreters*. Dicho paquete contiene un conjunto de clases *Interpreter* por cada clase DAO existente. De esta manera, la petición al conector desde el DAO tiene un paso previo de traducción de objeto a código CCR. A su vez, cuando se descarga un objeto desde GH y antes de devolverlo a la aplicación hay que traducirlo desde CCR a objeto java utilizable por la aplicación. Esta traducción se hace desde la clase *Interpreter* correspondiente.

5.2.2 Arquitectura del módulo

En la Figura 5-2 observamos la arquitectura global del módulo. Ha sido simplificada para facilitar la visión y su entendimiento. El paquete que realiza la comunicación entre aplicación externa y el módulo es *DAOs*. Este paquete

contiene las clases *DAOs* que implementan la interfaz *DAO* correspondiente (no se muestra en la figura por simplificación). La clase *DAO* utiliza los objetos *DTO* que representan los objetos almacenados en GH en forma de código *CCR*.

Para realizar la traducción entre los objetos *DTO* que utiliza la aplicación y el código CCR proveniente de GH se utiliza el paquete *Interpreters*. Este paquete contiene una clase por cada tipo de objetos que haya que hacer traducción. Por ejemplo, desde la aplicación se realiza la petición a través del objeto *AnalisisDAOGH* de insertar un nuevo *análisisDTO*. Para realizar la conversión entre el objeto DTO y código CCR se utilizará la clase *analisisInterpreter*, la cual, proporcionará el código CCR que representa el objeto. Por último una vez el *objetoDTO* es traducido a código CCR del análisis, se realizará la petición al conector para que realice la inserción pasándole como parámetro el identificador de paciente y el código CCR a insertar.

En el diagrama se observa que las distintas clases *DTOs* contienen objetos de las clases auxiliares. La relación de composición no es de todos a todos (una vez más ha sido mostrada así por simplificar); si no que, cada clase DTO contendrá una serie de atributos que se corresponderán con los campos CCR que contiene ese tipo en GH. Por ejemplo, de nuevo la clase analisisDTO contiene 5 atributos : *idObjeto* de tipo *String*, fechaAnalisis de tipo DateTime, descripción de tipo Description, datosMedico de tipo Source y *resultadosDelAnalisis* que es una lista de ResultadosTestDTO.

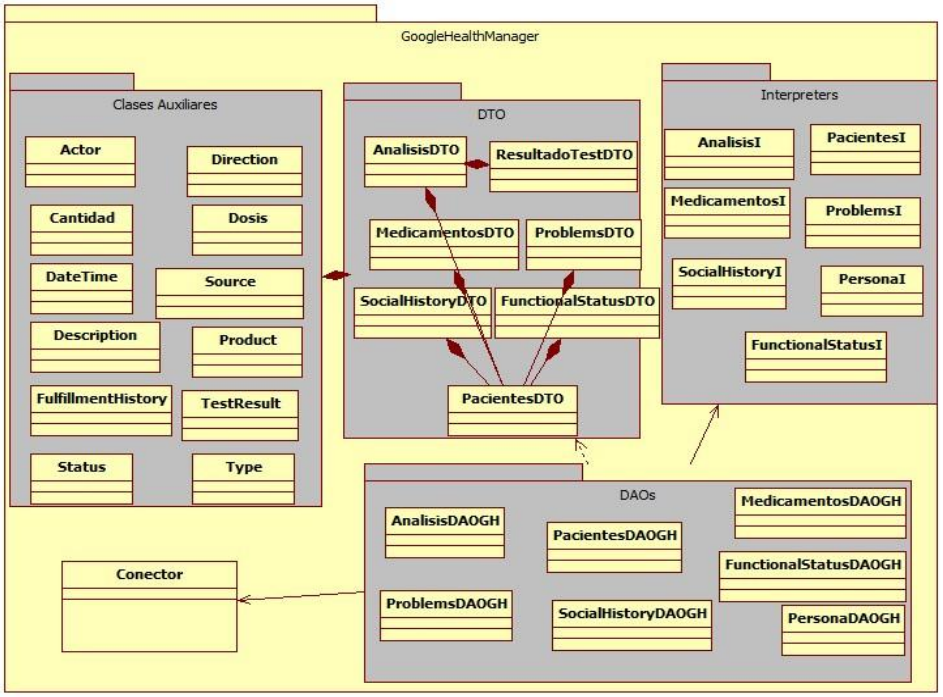


Figura 5-2 Visión global de la arquitectura del módulo de conexión con Google Health

Clase Conector

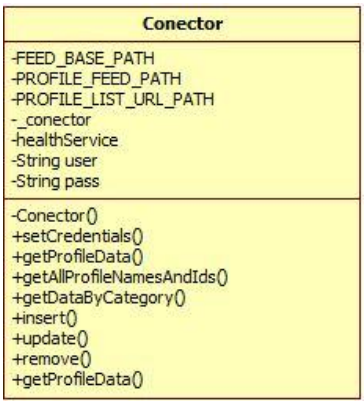


Figura 5-3 Visión de la clase Conector

Como podemos observar en la Figura 5-3 la clase conector está formado por cinco atributos. *FEED_BASE_PATH* es de tipo String y es la ruta base del *feed*, también tendremos la ruta *PROFILE_FEED_PATH* que es la URL de los perfiles y por último tenemos la *PROFILE_LIST_URL_PATH* que es la URL de la lista de todos los perfiles. Por otra parte tenemos el atributo *healthService* es de tipo HealthService tipo que proporciona la API de GH para Java y que es necesario para crear una conexión desde java a Google Health. El atributo _conector es utilizado ya que esta clase está diseñada utilizando el patrón de diseño Singleton. La utilización de este patrón viene motivada para asegurar un sólo ejemplar de la clase Conector y proporcionaremos un punto de acceso global a éste. De esta manera, en vez de tener una variable global para acceder desde toda la aplicación a este ejemplar, la clase se encargar de proporcionar un método de acceso. Para asegurar que solo hay un ejemplar es de la siguiente manera:

```
public static Conector getConector() {
    if (_conector == null) {
        _conector = new Conector();
    }
    return _conector;
}
```

Por otra parte, la clase Conector contiene todas las funcionalidades necesarias para realizar la conexión directa con Google Health. Esta clase llevará a cabo las funciones de listar los perfiles registrados en Google Health con los identificadores internos, obtener la información clínica de un perfil, insertar nuevos datos, actualizar datos y eliminar datos de un perfil de Google Health.

Un usuario puede tener registrados datos médicos de varios perfiles y, por tanto, los datos almacenados en Google Health se archivan según el sujeto al que pertenecen, encapsulados en una entrada *Atom*. Cada entrada *Atom* estará identificada mediante un identificador alfanumérico único y un nombre del perfil representado. Para poder consultar los datos de un perfil necesitamos

conocer el identificador interno que Google Health le asignó en el momento de su creación. Por lo tanto, necesitamos obtener primero el identificador del perfil a consultar. Para listar los perfiles registrados y sus respectivos identificadores usaremos:

```
String PROFILE_LIST_URL = "http://www.google.com/health/feeds/profile/list/";
Feed profileListFeed = healthService.getFeed(new URL PROFILE_LIST_URL,
Feed.class);
List<Entry> entries = profileListFeed.getEntries();
for(Entry profileListEntry : entries){
    System.out.println("Profile name: " +
profileListEntry.getTitle().getPlainText());
```

Una vez encontrado el identificador del perfil que queremos consultar, podemos obtener su información clínica en formato CCR. Para ello:

```
String profileID = “...”;
String PROFILE_FEED_PATH = "https://www.google.com/health/feeds/profile/ui/";
String url = PROFILE_FEED_PATH + profileID;
ProfileFeed profileFeed = healthService.getFeed(new URL(url),
ProfileFeed.class);
for(ProfileEntry entry : profileFeed.getEntries()) {
 System.out.println(entry.getContinuityOfCareRecord().getXmlBlob().getBlob());
```

Si queremos introducir nuevos datos para un perfil de Google Health, debemos empezar generando el código XML con los nuevos datos siguiendo el formato CCR. Una vez generado el código XML, podemos proceder a introducirlo en el sistema. Para ello, recuperaremos el ProfileFeed para el perfil y a través de su función createEntry(), generaremos la estructura para una la Entry. A continuación inicializaremos el campo ContinuityCareOfRecord de la Entry y configuraremos su código XML como el código generado anteriormente. Para finalizar, insertaremos la nueva Entry por medio de la función insert(…) del HealthService que nos devolverá la nueva Entry creada con su respectivo nuevo identificador asociado:

```
String datos = “<urn:Body xmlns:urn="urn:...”;
String profileID = "...";
String PROFILE_FEED_PATH = "https://www.google.com/health/feeds/profile/ui/";
String url = PROFILE_FEED_PATH + profileID;
//Obtenemos el ProfileFeed
ProfileFeed profileFeed = healthService.getFeed(new URL(url),
ProfileFeed.class);
//Creamos una nueva Entry
ProfileEntry entry2 = profileFeed.createEntry();
//Inicializamos y configuramos los datos de la Entry
entry2.setContinuityOfCareRecord(new ContinuityOfCareRecord());
entry2.getContinuityOfCareRecord().setXmlBlob(new XmlBlob());
entry2.getContinuityOfCareRecord().getXmlBlob().setBlob(datos);
//Introducimos los datos
entry2 = healthService.insert(new URL(url), entry2);
```

El API de Google Health también ofrece la posibilidad de actualizar los datos de cierto elemento introducido en un perfil. Para ello necesitaremos el

código original del elemento, es decir, el código introducido previamente obtenido al recuperar el elemento de Google Health. Sobre este código XML que contiene la información del elemento, podremos realizar algunos cambios en los valores encerrados entre la etiquetas XML.

```
String datos = “<urn:Body xmlns:urn="urn:...”;
String profileID = "...";
String PROFILE_FEED_PATH = "https://www.google.com/health/feeds/profile/ui/";
String url = PROFILE_FEED_PATH + profileID;
//Obtenemos el ProfileFeed
ProfileFeed profileFeed = healthService.getFeed(new URL(url),
ProfileFeed.class);

ProfileEntry entry = profileFeed.createEntry();
entry.setContinuityOfCareRecord(new
com.google.gdata.data.health.ContinuityOfCareRecord());
entry.getContinuityOfCareRecord().setXmlBlob(new XmlBlob());
entry.getContinuityOfCareRecord().getXmlBlob().setBlob(xmlText);
String url = PROFILE_FEED_PATH + identificador + "/" + idInterno;
healthService.update(new URL(url), entry);
```

El API de Google Health también nos ofrece la posibilidad de borrar la mayoría de los elementos introducidos en un perfil. Para ello, únicamente necesitaremos el identificador interno asignado por Google Health y que en el código XML entre las etiquetas <CCRDataObjectID> y el identificador del perfil al que pertenece el elemento. Al eliminar un elemento se eliminarán también todos aquellos subelementos que lo compongan. El código para borrar un elemento dado su identificador interno es el siguiente:

```
String PROFILE_FEED_PATH = "http://www.google.com/health/feeds/profile/ui/";
healthService.delete(new URL(PROFILE_FEED_PATH + idPaciente + "/" + idObjeto));
```

Paquete DAOs

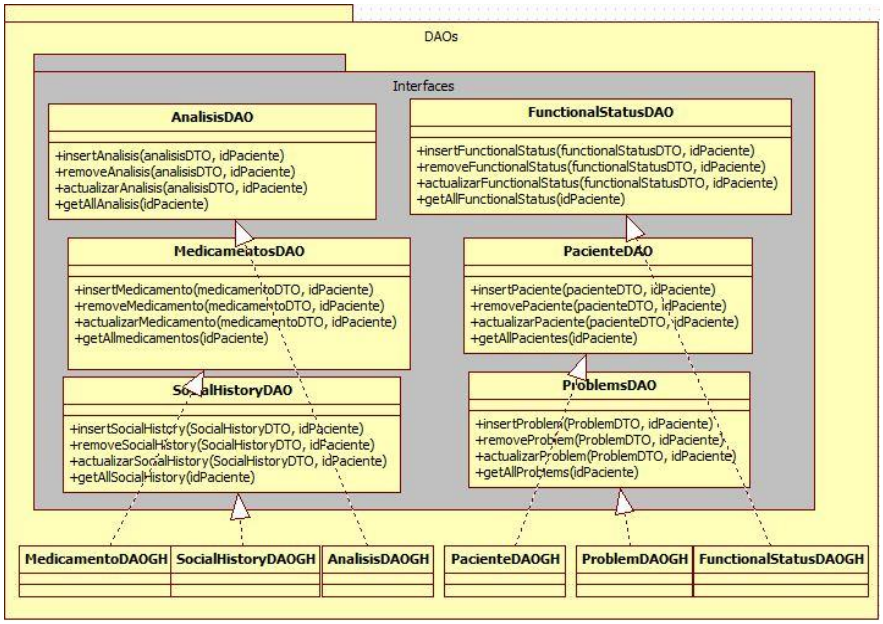


Figura 5-4 Arquitectura del paquete DAOs

El paquete DAOs contiene a su vez otro paquete de interfaces que son:

AnalisisDAO,

FuncionalStatusDAO,

PersonaDAO,

ProblemsDAO,

ResultadoTestDAO,

SocialHistoryDAO y

PacienteDAO.

Estas interfaces tienen 4 funciones: insertar, eliminar, actualizar y obtener el DTO de cada uno de los objetos que representan. Estas interfaces son implementadas por los AnalisisDAOGH, FuncionalStatusDAOGH, PersonaDAOGH, ProblemsDAOGH, ResultadoTestDAOGH, SocialHistoryDAOGH y PacienteDAOGH que son las encargadas de realizar la gestión de acceso a GH.

A continuación se va a explicar esquemáticamente cada una de las funciones que implementan estas clases. La ejemplificación se va a llevar a cabo mediante el tipo ProblemDAOGH, el resto de las clases funcionan de forma similar.

Función de inserción:

El objeto Problem de tipo problemDTO se traduce a código CCR y se almacena en un String. Posteriormente se realiza la inserción a través del conector pasándole el código CCR y el identificador de paciente en GH.

```
//El objeto problem es de tipo problemDTO y proviene de la aplicación
//Traducimos de un DTO al estandar CCR,devolviendo un XML
String XMLCode = new ProblemsInterpreter().translateDTOtoCCR(problem);
//Insertamos el XML para un paciente con un identificador.
Conector.getConector().insert(idPaciente, XMLCode);
```

Función actualización:

El objeto Problem de tipo problemDTO se traduce a código CCR y se almacena en un String. Posteriormente se realiza la actualización a través del conector pasándole el código CCR, el identificador de paciente en GH y el identificador interno del objeto en GH.

```
//El objeto problem es de tipo problemDTO y proviene de la aplicación
//Traducimos de un DTO a un CCR devolviendo un XML
String XMLCode = new ProblemsInterpreter().translateDTOtoCCR(problems);
//Actualizamos el XML para un paciente
Conector.getConector().update(idPaciente, XMLCode, problems.getIdInterno());
```

Función eliminación:

Para eliminar un elemento de GH se necesita el identificador del paciente y el identificador interno de GH del objeto a eliminar.

```
//El objeto problem es de tipo problemDTO y proviene de la aplicación
//Eliminamos  los problems de un paciente
Conector.getConector().remove(idPaciente, problem.getIdInterno());
```

Para obtener todos los DTO:

En primera instancia obtenemos el código CCR en formato XML del paciente. Posteriormente el intérprete de la clase correspondiente traducirá ese código en los objetos DTO necesarios. En este caso el problemsInterpreter nos proporciona una lista de todos los Problems del paciente.

```
//Obtenemos el XML de un paciente
String XMLData = Conector.getConector().getProfileData(idPaciente);

//Traducimos de un codigo XML en formato CCR a un DTO
ArrayList<ProblemsDTO> lista = new
ProblemsInterpreter().translateCCRtoDTO(XMLData);
```

Paquete Clases Auxiliares

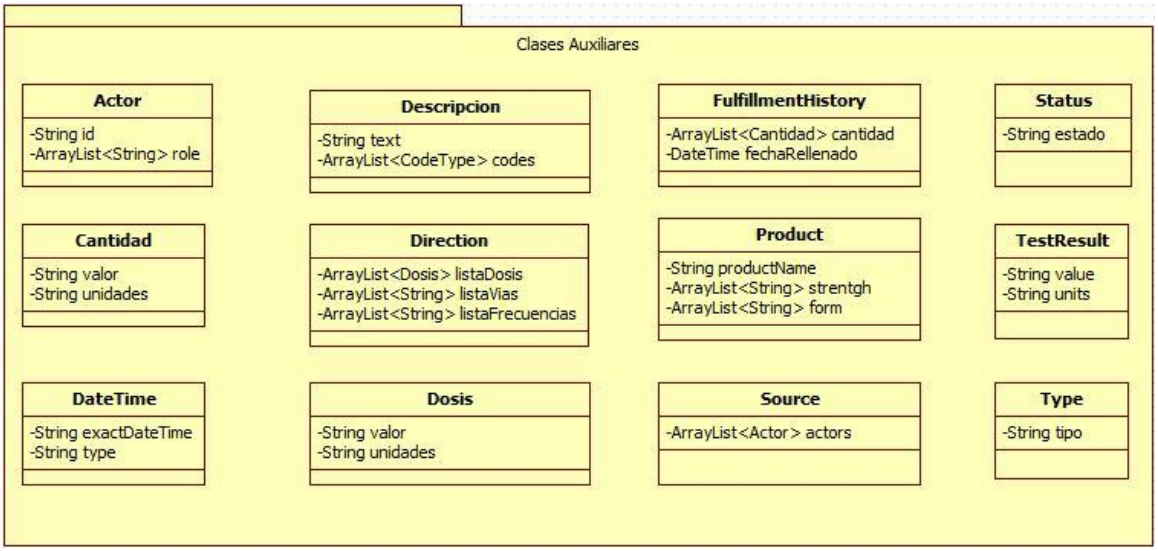


Figura 5-5 Clases contenidas en el paquete Clases Auxiliares

El estándar CCR está formado por elementos básicos. Estos elementos son los que irán dando forma a las clases DTO. Dichos elementos los podemos encontrar en el paquete de Clases Auxiliares y son las clases que podemos observar en la Figura 5-5.

La clase ***Actor*** está formada por: un identificador de tipo String y un role que es un *Arraylist* de String. La clase ***Cantidad*** está formada por un valor y unas unidades que son de tipo String. La clase ***DateTime*** tiene como atributos exactDateTime y type ambos de tipo String.

La ***Description*** como podemos observar en la Figura 5-5 contiene un text que es de tipo String y un array de codeType. La clase ***Direction*** está

construida a partir de un array de dosis, array de vías y array de frecuencias. La ***Dosis*** está formada por una cantidad y un valor que son de tipo String.

La ***FulfillmentHistory*** es un Arraylist de cantidades y una fecha de rellenado que es de tipo DateTime. La clase ***Product*** está formado por un nombre del producto que es de tipo String, un Arraylist de tamaños y de formas que son de tipo String. La clase ***Source*** es una lista de actores de tipo Actor.

El ***Status*** es un estado de tipo String. ***TestResult*** está formado por valor y unidades que son de tipo String y por último la clase ***Type*** contiene un atributo llamado tipo que es una cadena de caracteres.

Paquete DTOs

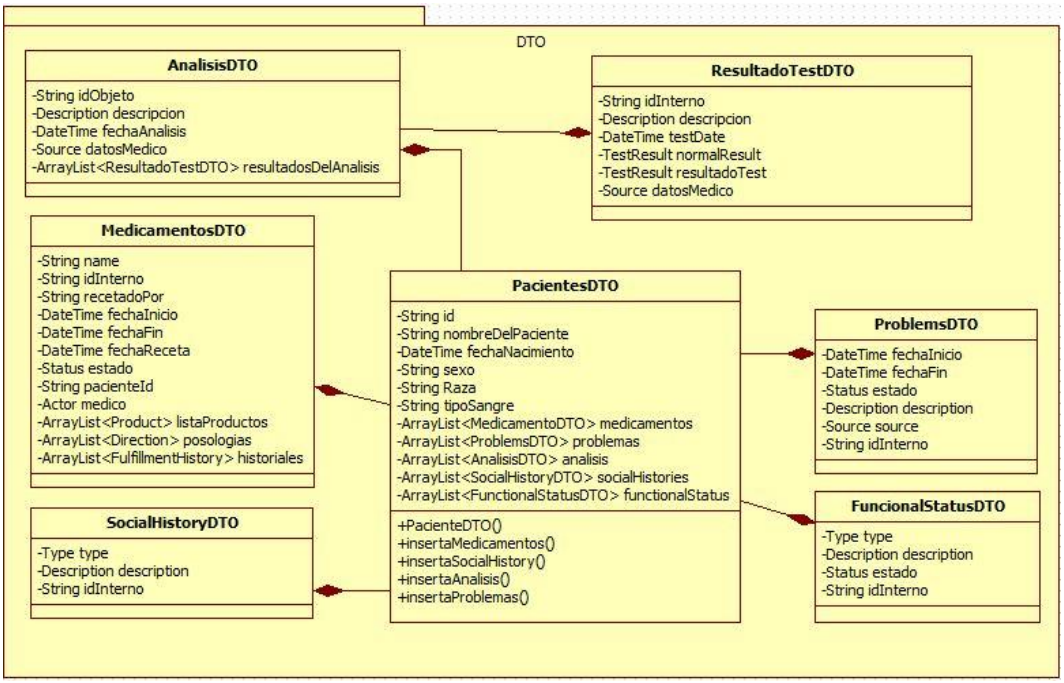


Figura 5-6 Contenido del paquete DTOs

Como podemos observar en la Figura 5-6, el paquete DTOs está formado por:

AnalisisDTO,

FuncionalStatusDTO,

PersonaDTO,

ProblemsDTO,

ResultadoTestDTO,

SocialHistoryDTO,

PacienteDTO.

Estas clases están formadas por los elementos básicos que encontramos en el paquete de Clases auxiliares.

La clase ***FuncionalStatusDTO*** está formada por un Type, una descripción, un estado y un identificador interno. La clase ***ProblemsDTO*** está constituida por una fecha de inicio y fecha de fin de tipo DateTime, un estado de tipo Status, una descripción, Source y un identificador interno. La clase ***SocialHistoryDTO*** tiene tres atributos que son una descripción, un type y un identificador interno.

Como podemos observar en la Figura 5-6 la clase ***MedicamentosDTO*** está formada por un nombre, identificador interno, fecha de inicio, fecha de fin y fecha de receta que son de tipo DateTime, un estado, el identificador del paciente, el médico del paciente que es de tipo Actor, una lista de productos, una lista de posologías y una lista de historiales.

Por otra parte el ***ResultadoTestDTO*** está constituido por una descripción, un testDate de tipo DateTime, un normalResult y un resultadoTest de tipo TestResult y los datos del médico que son de tipo Source. La clase ***AnalisisDTO*** está formado por una descripción, una fecha, los datos del médico de tipo Source y una lista de ResultadoTestDTO.

De esta manera, como observamos en la Figura 5-6 **l**a clase ***pacientesDTO*** está construida por una lista de ProblemsDTO, FuncionaStatusDTO, SocialHistoryDTO, MedicamentosDTO y AnalisisDTO, además de tener otros atributos como, el nombre del paciente, la fecha de nacimiento, sexo, raza y tipo de sangre.

Paquete Interpreters

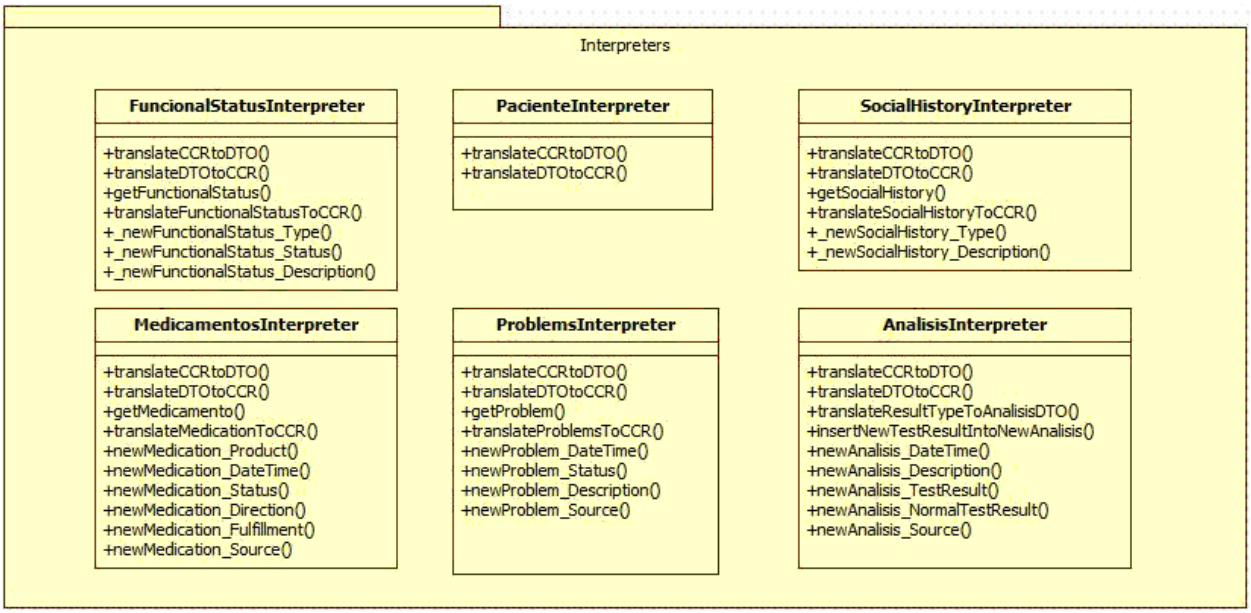


Figura 5-7 Contenido del paquete Interpreters

Como puede observarse en la Figura 5-7, el paquete Interpreters está formado por las clases:

AnalisisInterpreter,

FuncionalStatusInterpreter,

ProblemsInterpreter,

ResultadoTestInterpreter,

SocialHistoryInterpreter,

PacienteInterpreter.

Estas clases son las encargadas de traducir entre código CCR y objetos Java y viceversa. Existen tres métodos que toda clase Interpreter ha de implementar: *translateCCRtoDTO, transalateDTOtoCCR, y getObjeto.* A continuación explicaremos detalladamente cada uno de las funciones.

Función *translateCCRtoDTO*

Esta función recibe el texto XML con el código CCR que representa información de un paciente. A partir de esta información y en función del tipo de traductor que sea realizará el proceso de traducción. Para llevar a cabo el proceso de traducción nos hemos servido del paquete *com.google.code.ccr4j* que es independiente de GH y que proporciona una serie de elementos para representar objetos CCR. Para explicar todo esto se va a exponer el código para transformar un código XML a una lista de objetos DTO en este caso sería una lista de ProblemsDTO.

En primer lugar se prepara el código CCR en formato XML para ser parseado:

```
XMLData = formatearCodigoXML(XMLData);
```

En segundo lugar obtenemos la lista de problems en objeto java de tipo ContinuityOfCareRecord.Body.Problems :

```
ContinuityOfCareRecordDocument xmlParseado =
ContinuityOfCareRecordDocument.Factory.parse(XMLData);
ContinuityOfCareRecordDocument.ContinuityOfCareRecord CCR =
xmlParseado.getContinuityOfCareRecord();
ContinuityOfCareRecord.Body.Problems enfermedadesDelPaciente =
CCR.getBody().getProblems();
```

Sin embargo este tipo no es fácilmente utilizable por la aplicación por lo que realizamos un proceso de traducción entre tipos Java. Para ellos utilizamos las clases auxiliares que proporciona *com.google.code.ccr4j :*

```
if (enfermedadesDelPaciente != null) {
    List<ProblemType> listaAnalisis = enfermedadesDelPaciente.getProblemList();
    Iterator<ProblemType> it = listaAnalisis.iterator();
    ArrayList<ProblemsDTO> lista = new ArrayList<ProblemsDTO>();
    while (it.hasNext()) {
    // El tipo Problem es proporcionado por com.google.code.ccr4j  y contiene un objeto
Java con los campos del CCR.
        ProblemType problem = it.next();
    // La acción de traducción se delega en getProblem(problem) quien a partir de problem
de tipo com.google.code.ccr4j.ProblemType  obtendrá un problemDTO utilizable por la
aplicación.
        ProblemsDTO problemas = getProblem(problem);
        problemas.setIdInterno(problem.getCCRDataObjectID().split("-")[0]);
        lista.add((ProblemsDTO) problemas);
    }
    return lista;
```

Función *getObjeto*

Este método se encarga de obtener un objetoDTO a partir de un objeto de tipo *com.google.code.ccr4j. Objeto, para* ello es necesario conocer de que está formado cada objeto y desglosarlo en sus características. Como por ejemplo, continuando con problems, se obtiene un ProblemDTO a partir de un *com.google.code.ccr4j.ProblemType*. Un ProblemDTO estará formado por Fechas, Status, Source, Description. El código sería el siguiente:

```
// Creo el problemDTO vacío
ProblemsDTO p = new ProblemsDTO();
//Obtengo la lista de Fechas
List<DateTimeType> dateTimeList = problem.getDateTimeList();
if (dateTimeList != null) {
    Iterator<DateTimeType> it1 = dateTimeList.iterator();
    while (it1.hasNext()) {
        // DateTimeType es de tipo com.google.code.ccr4j.DateType
        DateTimeType fecha = (DateTimeType) it1.next();
        // fecha1 es de tipo DateTime tipo auxiliar del módulo que representa
        //una fecha. Obtendrá la fecha a través del método
        //getDate(com.google.code.ccr4j.DateType)
        DateTime fecha1 = DateTime.getDate(fecha);
        p.insertaFecha(fecha1);
    }
}
//Obtengo el estado de la enfermedad
CodedDescriptionType status = problem.getStatus();
if (status != null) {
    //estado es de tipo Status que es un tipo auxiliar del módulo . El método
    //getStatus se encarga de transofmar el status de tipo
    //com.google.code.ccr4jCodedDescriptionType en ClasesAuxiliares.Status
    Status estado = Status.getStatus(status);
    p.setEstado(estado);
}
//Obtengo la fuente de la enfermedad
List<SourceType> sourceList = problem.getSourceList();
if (sourceList != null) {
    Iterator<SourceType> it2 = sourceList.iterator();
    while (it2.hasNext()) {
        SourceType fuente = (SourceType) it2.next();
        // fuente1 es de tipo Sourceque es un tipo auxiliar del módulo . El
        // método getSource se encarga de transofmar el tipo de
        //com.google.code.SourceType  en ClasesAuxiliares.Source
        Source fuente1 = Source.getSource(fuente);
        p.insertaFuente(fuente1);
    }
}
//Obtengo la descripcion de la enfermedad
CodedDescriptionType descriptionP = problem.getDescription();
if (descriptionP != null) {
    // descripcion es de tipo Description que es un tipo auxiliar del módulo .
    //El método getDescription  se encarga de transformar el tipo de
    //com.google.code.CodedDescriptionType en ClasesAuxiliares.Description
    Description description = Description.getDescription(descriptionP);
    p.insertaDescription(description);
}
return p;
```

Función *translateDTOtoCCR*

Esta función se encarga de traducir un objetoDTO en su código CCR correspondiente. También se sirve de las clases auxiliares que proporciona *com.google.code.ccr4j.* A continuación se explica detalladamente un ejemplo para problemsDTO:

```
// Creo un nuevo documento CCR
ContinuityOfCareRecordDocument.ContinuityOfCareRecord CCR =
ContinuityOfCareRecordDocument.ContinuityOfCareRecord.Factory.newInstance();
// Le añado un cuerpo y una lista de enfermedades ( problems)
CCR.addNewBody();
Problems addNewProblems = CCR.getBody().addNewProblems();
ProblemType NewProblem = addNewProblems.addNewProblem();
// Delego en el método translateProblemsToCCR a el cual le paso el problemDTO y
//el nuevoProblem vacío
translateProblemsToCCR(NewProblem, problem);
return CCR.xmlText();
```

Este metodo crearia un nuevo DateTime, Status, Description, Source a partir del ProblemsDTO:

```
private void translateProblemsToCCR(ProblemType NewProblem, ProblemsDTO
problema) {
    newProblem_DateTime(problema.getFechaInicio(), problema.getFechaFin(),
NewProblem);
    newProblem_Status(problema.getEstado(), NewProblem);
    newProblem_Description(problema.getDescription(), NewProblem);
    newProblem_Source(problema.getSource(), NewProblem);
}
```

Este método crearía en el caso de que hubiese fecha de inicio o/y fecha de fin, un nuevo código XML insertándolo en el formato CCR necesario para GH. Los métodos de newProblem_Status, newProblem_Description, newProblem_Source, mantiene la misma idea.

```
public void newProblem_DateTime(DateTime fechaInicio, DateTime fechaFin,
com.google.code.ccr4j.ProblemType problem) {
    if (fechaInicio != null) {
        DateTimeType NewDateTime = problem.addNewDateTime();
        CodedDescriptionType NewType = NewDateTime.addNewType();
        NewDateTime.setExactDateTime(fechaInicio.getExactDateTime());
        NewType.setText(fechaInicio.getType());
    }
    if (fechaFin != null) {
        DateTimeType NewDateTime1 = problem.addNewDateTime();
        CodedDescriptionType NewType1 = NewDateTime1.addNewType();
        NewDateTime1.setExactDateTime(fechaFin.getExactDateTime());
        NewType1.setText(fechaFin.getType());
    }
}
```

A continuación se van a mostrar dos diagramas de secuencia, uno de la inserción de un análisis desde el módulo de conexión a GH (Figura 5-8) y otro de la obtención de la lista de análisis presentes en GH (Figura 5-9). De esta forma se pretende explicar los distintos pasos que se llevan a cabo para la inserción/obtención de cualquier elemento en GH. El diagrama se inicia en analisisDAOGH pero la petición provendrá de la aplicación a través de esta clase.

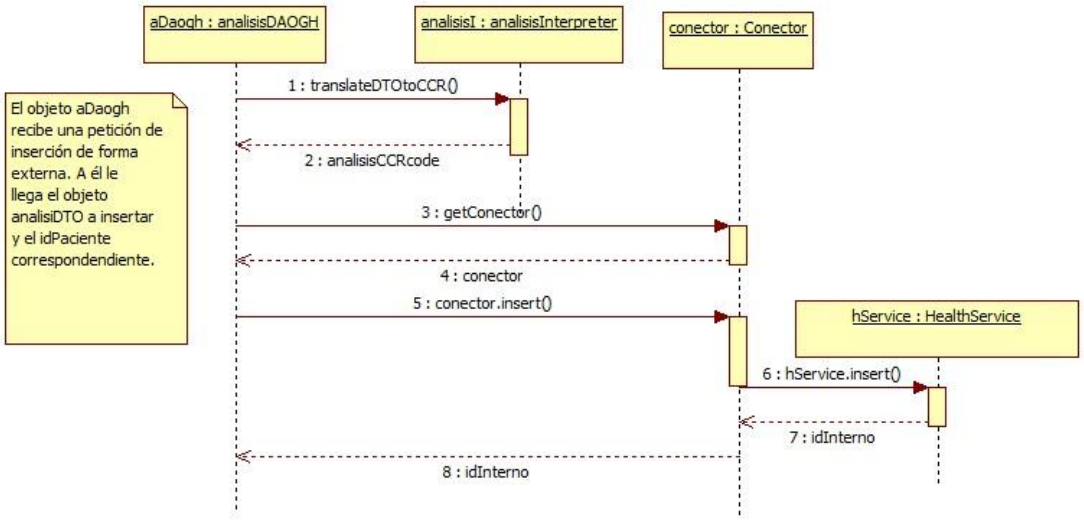


Figura 5-8 Diagrama de secuencia para inserción de un objeto

La petición llega al objeto aDaogh, desde aquí lo primero que se realiza es la traducción de análisis DTO a código CCR mediante el analisisInterpreter. Una vez tenemos el XML con el código CCR necesitamos el conector para realizar la inserción. Al estar implementado mediante Singleton obtendremos la instancia del conector mediante el método *getConector()*. Una vez tenemos la instancia del conector realizaremos la inserción a través del método *insert(XMLCode, IdPaciente)* . Por último en el conector a través del servicio HealthService que ofrece el API para Java de GH se realiza el último paso de la inserción.

Esta petición devuelve un identificador del objeto en Google Health. Identificador muy valioso para posteriores modificaciones.

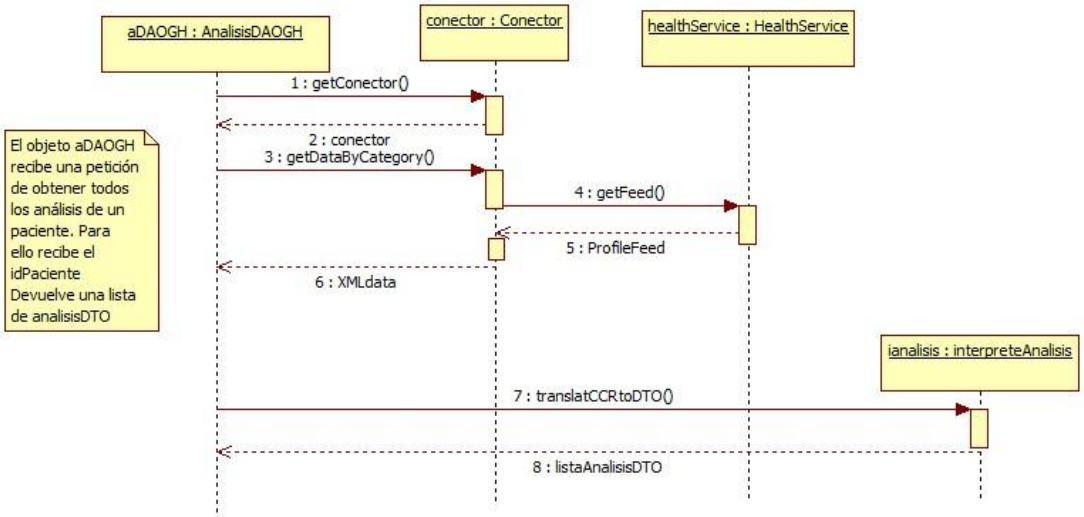


Figura 5-9 Diagrama de secuencia para la obtención de un objeto

Este segundo diagrama representa la obtención del conjunto de análisis presentes en Google Health. El objeto de tipo analisisDAOGH recibe una

petición de la aplicación para obtener la lista de análisis. Para ello es necesario tener el identificador del paciente del que se desea recuperar los datos. Una vez tenemos la instancia del conector realizamos la petición al conector. El conector. Posteriormente, el conector a través de *healthService* el que obtendrá los datos de análisis del paciente. Del conector al objetoDAO que ha realizado la petición se devuelve el código XML que representa la lista de objetos. Por ello se necesitará que el *interpreterAnalisis* realice la traducción y obtenga la lista de objetos DTO.

5.2.3 Utilización del módulo de conexión Google Health

El objetivo que buscamos con la realización de este módulo es facilitar la conexión con Google Health mediante java y construir una API para que mediante sencillos métodos (insert*, update, delete, getAll...)* se pueda tener acceso a los datos almacenados en este servicio de Google. El módulo de conexión a GH es independiente de la aplicación. Bien es cierto, que para que la comunicación sea posible, la aplicación ha de seguir ciertas pautas que se explican a continuación:

Debe existir una traducción entre los objetos con los que trabaje la aplicación y los objetosDTO que recibe las clases DAOGH. Por lo tanto la aplicación que desee utilizar el API deberá conocer el formato de los objetosDTO que se transfieren a GH. Bien es cierto que el formato sigue el mismo que contienen los objetos en GH por lo que en teoría la transformación de objetos de la aplicación a objetosDTO debería ser sencilla e intuitiva.

La comunicación con el módulo de conexión a GH (y en última instancia con GH) se realiza a través de las clases DAOGH por lo que habrá que realizar la instanciación de la clase correspondiente en función del tipo de objeto que estemos tratando.

Se debe almacenar siempre el identificador del paciente con el que se desea interactuar, los objetosDAOGH necesitan de éste identificador para realizar las peticiones al conector (y en última instancia a GH).

Todo objeto que se desee actualizar o borrar debe tener asociado el identificador interno de Google Health. Siempre que se descargan de Google Health el elemento DTO correspondiente lo posee.

Como consecuencia del anterior, cuando se realiza una inserción se devuelve el identificador interno del objeto. Este identificador debe ser almacenado en el objeto que haya sido insertado para posteriores modificaciones. Un ejemplo de cómo llevar a cabo esta labor podría ser:

```
String nuevoId = new AnalisisDAOGH().insertNewAnalisis(new AnalisisDTO(modelo),
idPacienteOwner);
if (nuevoId != null) {
    // insertamos en el modelo de nuestra aplicación el identificador interno
de GH
    modelo.getResultadosTest().get(0).setIdInterno(nuevoId);
}
```

5.3 Módulo Paciente-Médico

5.3.1 Descripción Módulo.

Con el desarrollo de esta aplicación buscábamos dos objetivos fundamentales. Por una parte, utilizar el módulo de conexión con GH que hemos desarrollado y que puede servir como ejemplo de utilización del mismo. Por otra, desarrollar una aplicación que facilite la vida del paciente diabético y gestión de la misma por parte de su médico.

Muchos pacientes diabéticos no tienen altos conocimientos informáticos. Por ello, la idea de nuestra aplicación es la de proporcionar al paciente una aplicación sencilla y que le permita llevar un control diario de sus medidas de glucosa, medidas de insulina, comidas y cuerpos cetónicos. Por otra parte también queríamos servir al médico con una aplicación que le permita llevar un seguimiento exhaustivo del paciente y que le permita llevar actualizado el control de sus medidas. Es por ello que la aplicación puede dividirse en dos partes, la aplicación para el Paciente y la aplicación para el médico. Estas partes no son disjuntas ya que comparten algunas funcionalidades y sobre todo comparten el mismo diseño. Tanto a nivel de arquitectura como a nivel de Interfaz han sido diseñados de forma idéntica.

En relación a la interfaz, ésta ha sido diseñada siguiente un estilo Web. Esto es debido a que la mayoría de pacientes (y médicos) han navegado o utilizado Internet en alguna ocasión. Por ello les resultará más familiar y sencillo utilizar una *Interfaz Gráfica de Usuario* (GUI) que siga este patrón.

La aplicación, por lo tanto, utiliza Google Health como servicio de almacenamiento de información. No sería difícil cambiar el servicio de almacenamiento de información ya que la aplicación es independiente del módulo de conexión con GH (Apartado 5.2)

En cuanto a la arquitectura sigue el patrón modelo-vista-controlador. El objetivo de utilizar este patrón es hacer independiente la gestión de la lógica de la aplicación de la interfaz utilizada para ello. El controlador es el encargado de realizar la comunicación entre vistas y modelos. Además será el encargado de servir las peticiones a el módulo de conexión a GH.

En el paquete de los Modelos existen tantas clases como tipos de datos se desea representar en la aplicación. En concreto existen 7 tipos de modelos Analisis, Medicamentos, Paciente, Problemas, FunctionalStatus,

ResultadosTest y SocialHistory. Estas clases servirán para representar los elementosDTO que provienen del módulo de conexión y serán utilizados por la aplicación. Por otra parte, existen 4 controladores: controladorAnálisis, controladorMedicamentos, controladorProblemas y controladorPaciente. Por último, en vistas existen dos *Frames* principales, el que utiliza la aplicación para el médico y el del paciente.

En la Figura 5-10 vemos un esquema global de la arquitectura.

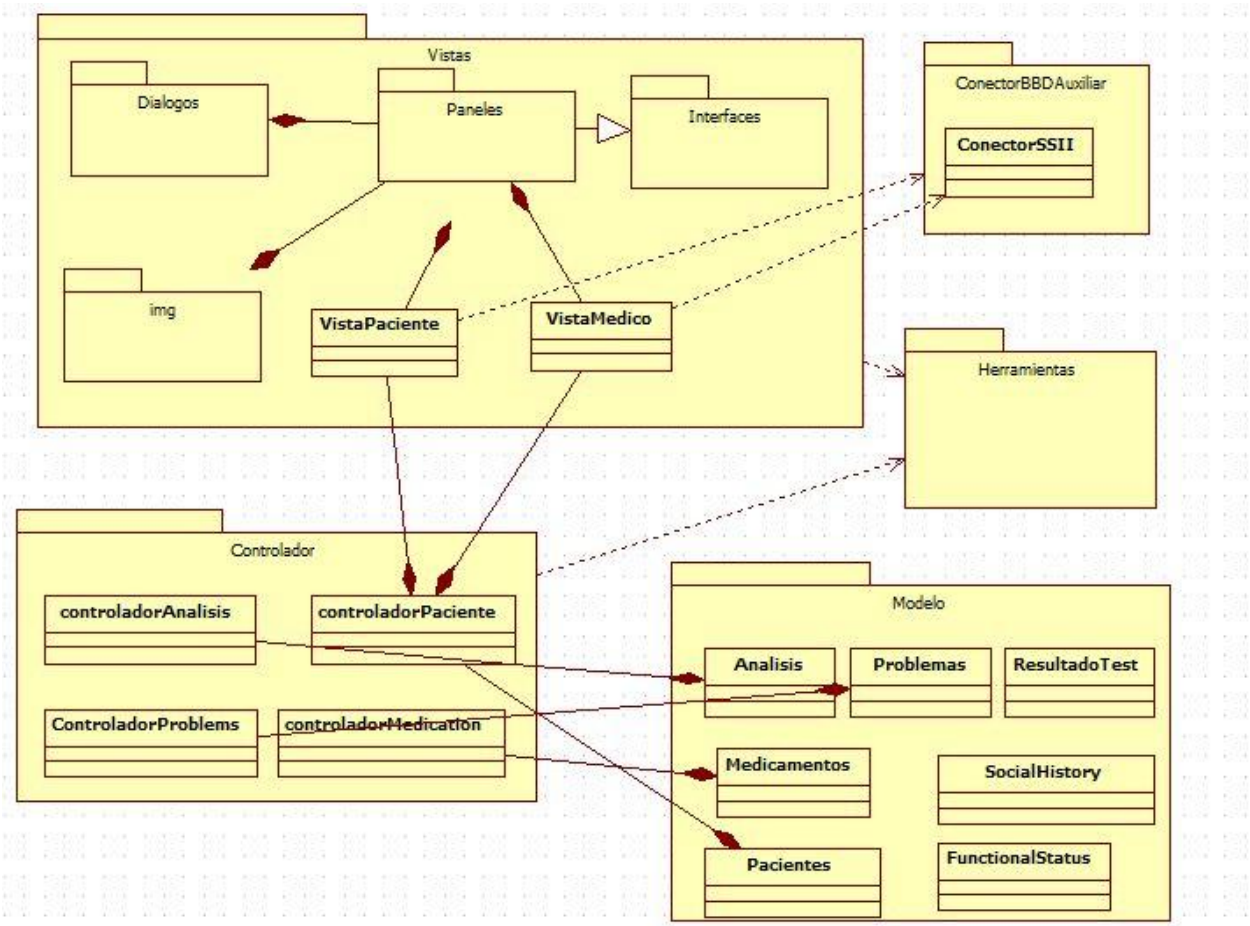


Figura 5-10 Arquitectura global del módulo

Otro aspecto a tener en cuenta es el proceso de autenticación en la aplicación. No ha sido sencillo realizar la tarea de que tanto paciente como médico puedan modificar los perfiles de los pacientes existentes en GH. Google sólo permite al creador del paciente (en nuestro caso el médico) modificar el perfil. Esto era un problema ya que e paciente debía introducir sus medidas diarias. Una de las opciones que se consideraron era que el paciente tuviera la contraseña del médico. Sin embargo, esta medida violaba la privacidad del resto de los pacientes ya que sus perfiles podían ser visitados por el que realice el registro en la aplicación. Por lo tanto optamos por otra solución: utilizar una base de datos externa que serviría para almacenar correspondencia entre pacientes y sus médicos. El funcionamiento es el siguiente: el paciente se inicia sesión en la aplicación con el nombre y

contraseña que su médico le proporcionó al crear su perfil. A continuación se accederá a la base de datos buscando una entrada que posea como nombre de paciente el utilizado para registrarse en la aplicación. De esa entrada se obtendrá el nombre del médico asociado y la contraseña del mismo. Con este nombre y contraseña se realizará el proceso de autenticación en GH de forma que el paciente nunca vea los datos con los que se realiza la conexión. De esta forma el paciente está conectado a GH y podrá modificar y almacenar las medidas que considere necesario. Este proceso de autenticación se explica detalladamente en el más adelante.

5.3.2 Arquitectura Módulo.

Una vez establecidas las funcionalidades básicas que nuestra aplicación debería ofrecer, se procedió a diseñar el módulo para su posterior implementación.

Debido a que todo el trabajo de cargar/descarga de información de las *Bases de Datos* (BBDD) se realiza en un módulo aparte (ver apartado 5.2.3), este módulo simplemente se encarga de representar al información y de ofrecer formularios para nuevas inserciones de información. Tras estudiar cómo debería ser la arquitectura del módulo, se decidió aplicar el patrón *Modelo/Vista/Controlador* (MVC, diagrama UML en Figura 5-11) junto al patrón *Observer* (diagrama UML en Figura 5-12). En concreto, se usó *MVC + Observer* en todas aquellas clases cuyos objetos fueran susceptibles de ser representados/editados/creados a través de la interfaz gráfica.

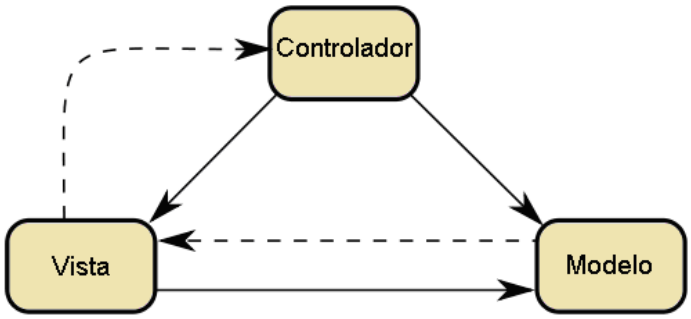


Figura 5-11 Arquitectura del patrón Modelo/Vista/Controlador

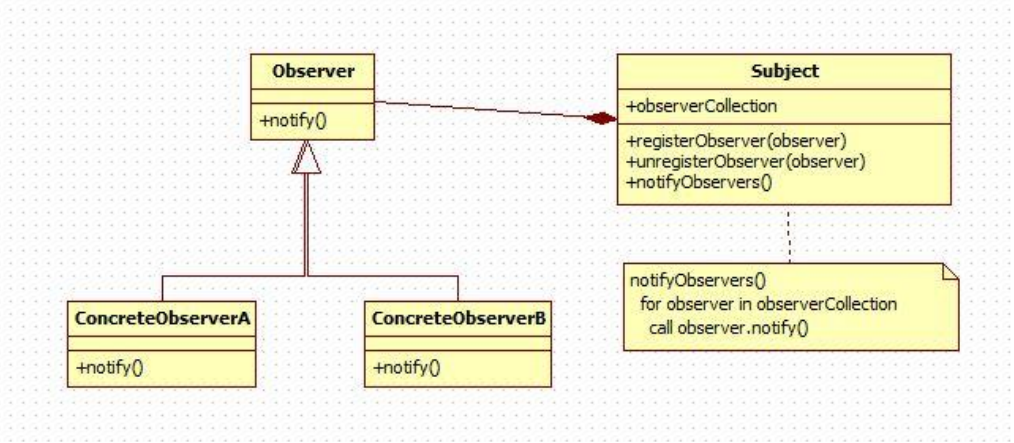


Figura 5-12 Arquitectura del patrón Observer

El papel de cada uno de los elementos de este patrón, podría resumirse de la siguiente manera:

Modelo: es el objeto que está siendo observado por una o varias vistas (implementa la interfaz Observable), contiene toda la lógica necesaria para editar y acceder a sus atributos.

Vista: las vistas se corresponden con toda aquella interfaz gráfica que se utilice para representar un objeto o parte de un objeto. Todas las vistas tienen un Modelo asociado, es decir, el objeto representado, sólo accesible a través del Controlador propio de la vista. La vista no modifica directamente el modelo sino que siempre trabaja a través del controlador. Todas las vistas implementan la interfaz Observer de manera que cuando su objeto observado (el modelo) sufre algún cambio, se les notificará por medio de la función updateView(...)

Controlador: el controlador ofrece toda la lógica necesaria para modificar el modelo asociado por medio de las vistas. Además se encarga de actualizar la Base de Datos con los cambios realizados a nivel local en el modelo usando las herramientas ofrecidas por el módulo de conexión con las BBDD.

Aunque se existen diferentes implementaciones de MVC, podríamos resumir el flujo que sigue el control como el siguiente:

El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón).

El controlador recibe la notificación de la acción solicitada por el usuario y gestiona el evento que llega.

El controlador accede al modelo modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el valor medido de una glucemia).

El patrón Observer provee cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a las vistas asociadas de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los

cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista.

La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

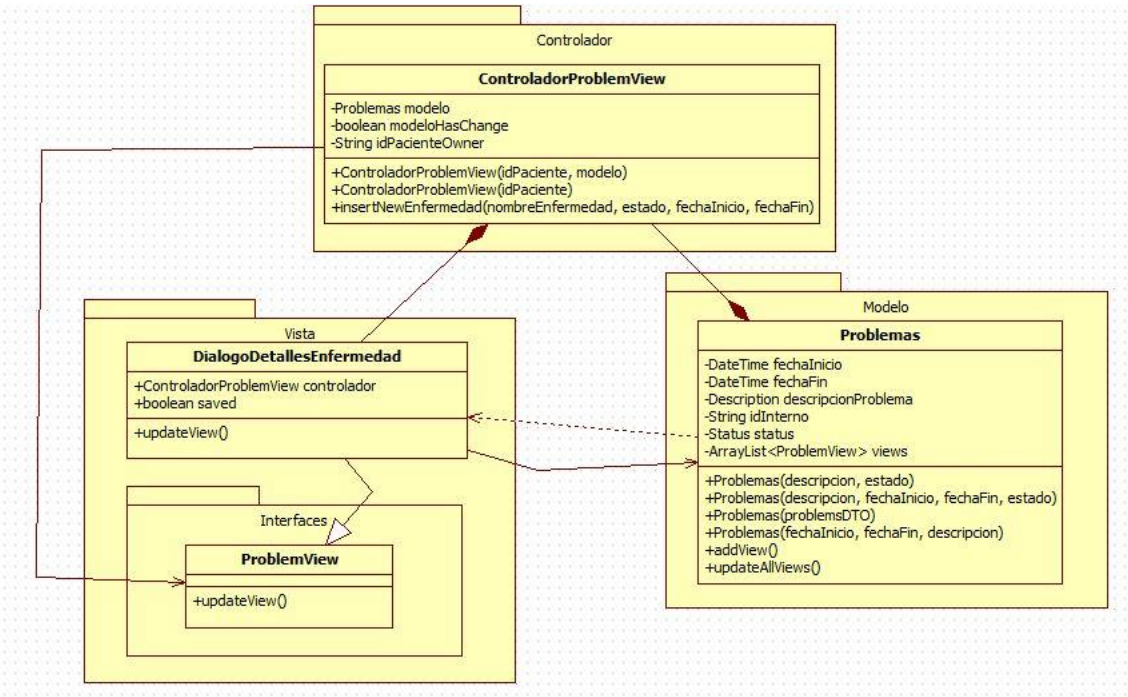


Figura 5-13 Arquitectura resultante tras aplicar MVC + Observer

En la Figura 5-13 se puede observar un ejemplo de la arquitectura resultante tras aplicar los patrones *MVC* y *Observer*. La vista DialogoDetallesEnfermedad implementa la interfaz ProblemView. Esta interfaz tiene un método que es el actualizar vista. Por otra parte el controlador ProblemView está formado por un modelo y dicho modelo tiene una array de vistas. De esta manera el controlador recibirá la notificación de la acción y se accederá al modelo modificándolo de manera adecuada. El patrón Observer provee cierta indirección entre el modelo y la vista, lo cual permite notificar al modelo las vistas que han sido modificadas. La interfaz de usuario esperará nuevas instrucciones del usuario para ir comenzando el ciclo de nuevo.

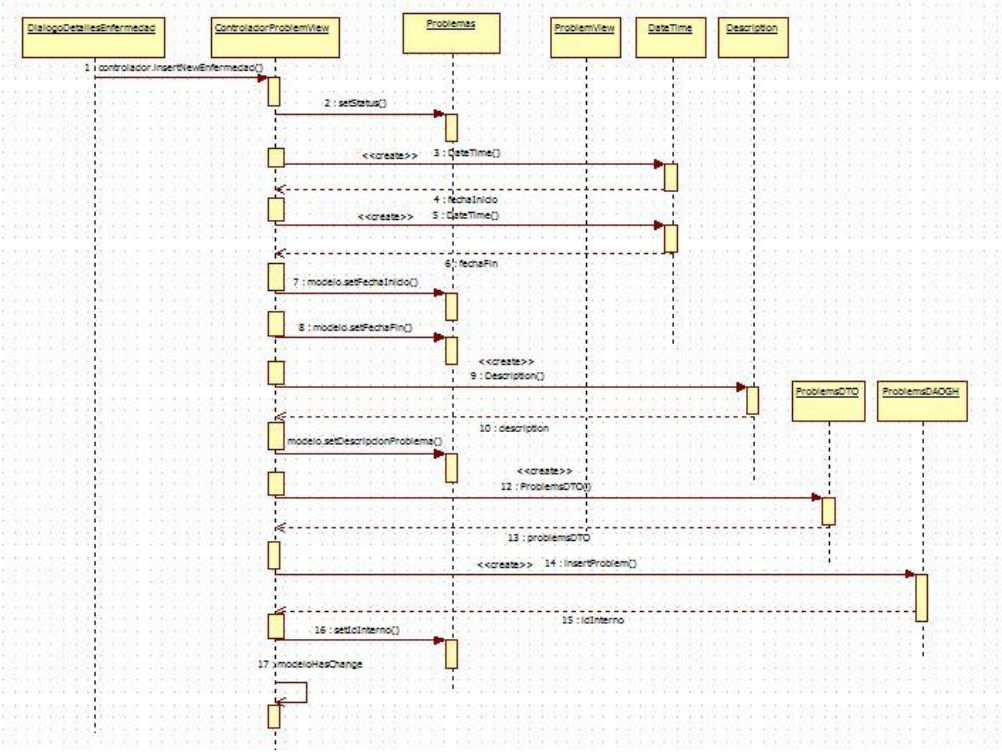


Figura 5-14 Diagrama de flujo para una inserción de un elemento usando MVC

El diagrama de flujo de la Figura 5-14 presenta las diferentes transacciones que se realizan aplicando el patrón MVC + Observer. Como podemos observar inicialmente se invoca a la función *controlador*.*insertNewEnfermedad*. Esta función es la encargada de realizar en el modelo todas las modificaciones necesarias. En ejemplo, se modificarán las fechas de inicio y fin y la descripción de una dolencia.

Luego nos creamos un ProblemDTO a partir del modelo y llamamos a la función insertProblem que se encuentra en la clase ProblemsDAOGH. La clase DAOGH realiza una serie de transformaciones que han sido explicadas en el apartado 5.2 con la diferencia que en este caso es de tipo ProblemsDAOGH y en ese apartado se hizo sobre Análisis, pero el flujo es el mismo. Si la operación se realiza con éxito, retorna el idInterno asignado al nuevo problema. Finalmente editamos el identificador al modelo asignándole con el id nuevo. Con este flujo de datos habríamos insertado una enfermedad con fecha de inicio y fin y una descripción.

A continuación se va a mostrar un diagrama de clases y paquetes de la arquitectura de la aplicación. No han sido representadas todas las clases así como las dependencias entre las mismas para clarificar la visión global de la arquitectura.

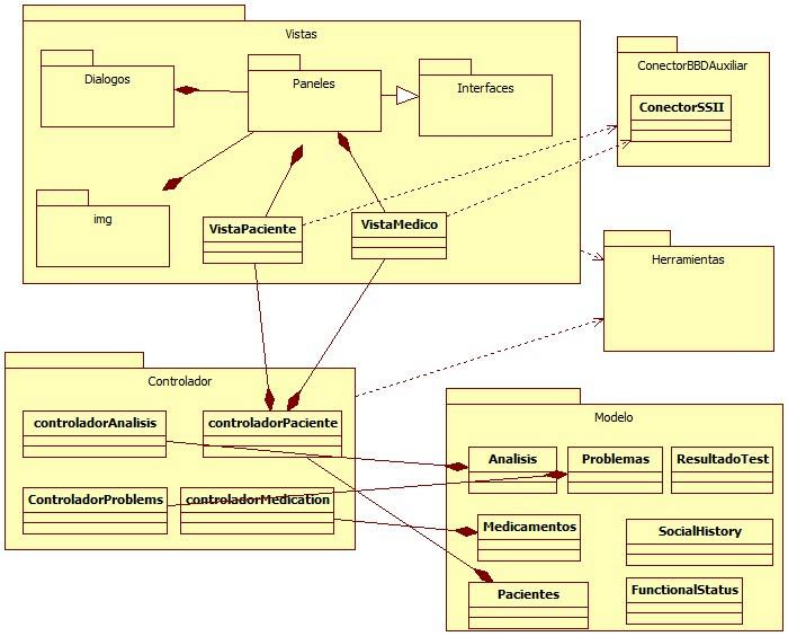


Figura 5-15 Distribución en paquetes de las clases del MVC

Observamos en la Figura 5-15 cómo se distribuye la aplicación respetando el patrón MVC.

Tanto la aplicación paciente como la aplicación para el médico son partes de una misma estructura o arquitectura. La diferencia es el conjunto de paneles y funcionalidades que utiliza una u otra en función de sus necesidades. Ambas vistas se sirven del *controladorPaciente* para realizar la comunicación con los modelos y con otros controladores. En función de las funcionalidades que necesiten cada vista contendrá los paneles y diálogos oportunos. Este hecho se explica detalladamente en la explicación del paquete Vistas.

Por otra parte el paquete Herramientas proporciona una serie de funcionalidades que se explicarán posteriormente y que utilizan tanto las vistas como los controladores. Por último, ConectorBBDD contiene el conector para la base de datos auxiliar y utilizada por los paneles de autenticación.

A continuación se va a exponer una explicación detallada de cada uno de los paquetes.

Paquete ConectorBBDDAuxiliar

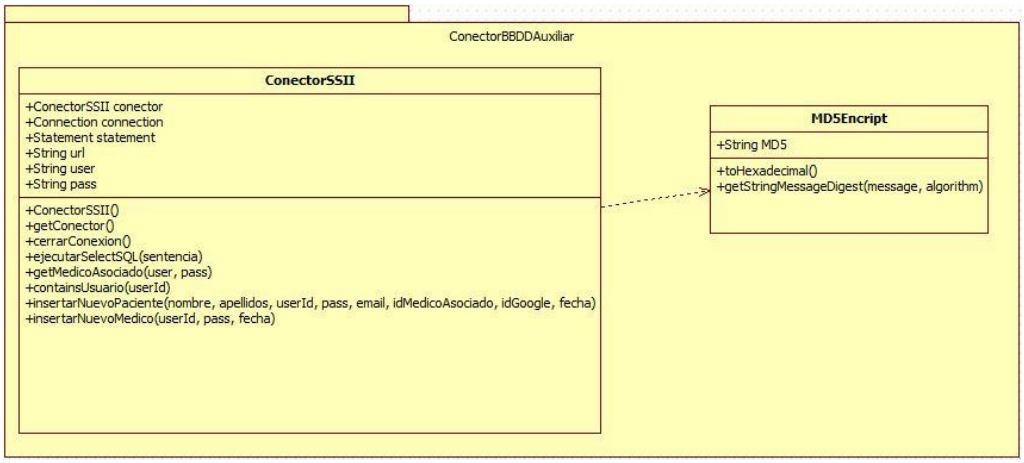


Figura 5-16 Arquitectura del paquete ConectorSSII

Como podemos ver en la Figura 5-16 el paquete está formado por 2 clases. Las clases ConectorSSII y la clase MD5Encript.

La clase *ConectorSSII* está formada por 6 atributos: *url* (ruta a la que vamos a conectar), *user* y *pass* (usuario y contraseña de acceso a la base de datos), *statement* (atributo parar preparar las sentencias SQL a ejecutar), *connection* (realiza la conexión con la base de datos) y el atributo *conector* (clase está diseñada utilizando el patrón de diseño *Singleton*). La utilización de este patrón viene motivada para asegurar un sólo ejemplar de la clase *Conector* y proporcionaremos un punto de acceso global a éste. De esta manera, en vez de tener una variable global para acceder desde toda la aplicación a este ejemplar, la clase se encargar de proporcionar un método de acceso. Para asegurar que solo hay un ejemplar es de la siguiente manera:

```
public static ConectorSSII getConector(){
        if (conector == null)
           conector = new ConectorSSII();
        return conector;
}
```

El método *getMedicoAsociado* se encargar de recuperar los datos del médico asociado al paciente con usuario user y contraseña pass. En caso existir en la BBDD, se devolverán los siguientes datos: id interno de google del paciente, cuenta de google del médico y pass. Para obtener estos campos se hace de la siguiente manera:

```
//Encriptamos la contraseña del paciente CON MD5 (formato usado por
Moodle)
String[] datos = new String[3];
String password_encriptada = MD5Encript.getStringMessageDigest(pass,
"MD5");
//RECUPERAMOS AL PACIENTE IDENTIFICADO COMO user
ResultSet rs = ejecutarSelectSQL("SELECT * FROM usuarios WHERE userid= '" + user + "'");
rs.beforeFirst();
//Si existe alguno registrado bajo ese userId en la BBDD
if (rs.next()) {
        String passRegistrada = rs.getString(4);
        //Si las contraseñas coinciden, el login es correcto
        if (passRegistrada.equals(password_encriptada)) {
            //Guardamos el id interno de google health para el paciente
            datos[0] = rs.getString(7);
            String idMedico = rs.getString(6);
            //Buscamos al medico asociado para recuperar la cuenta Google asociada y su
pass
            if (idMedico != null && !idMedico.isEmpty()) {
                rs = ejecutarSelectSQL("SELECT * FROM usuarios WHERE userid= '" + idMedico
+ "'");
                rs.beforeFirst();
                if (rs.next()) {
                    //Guardamos la cuenta de Google y la pass asociada
                    datos[1] = rs.getString(5);
                    datos[2] = rs.getString(4);
                    return datos;
                }
            }
        } else {
            System.err.println("Select paciente Failed: password
incorrect");
        }
} else {
        System.err.println("Select paciente Failed: user not found");
}
```

La función *containsUsuario* se encarga de comprobar si cierto usuario, ya sea médico o paciente está en la BBDD. La manera de comprobarlo es:

```
ResultSet rs = ejecutarSelectSQL("SELECT * FROM usuarios WHERE userid= '" + userId + "'");
rs.beforeFirst();
if (rs.next()) {
        return true;
}
```

El método insertarNuevoPaciente introduce un nuevo paciente en la base de datos.

```
String password_encriptada = MD5Encript.getStringMessageDigest(pass, "MD5");
String sentenciaUsuarios = "INSERT INTO
usuarios(nombre,apellidos,userid,password,email,medico,googleId,moodleId,altacurso," + "rolcurso) VALUES
" + "('" + nombre + "','" + apellidos + "','" + userId + "','" + password_encriptada + "','" + email +
"','" + idMedicoAsociado + "','" + idGoogle + "',0,0,5)";
String sentenciaEvaluacion = "INSERT INTO evaluacion_usuarios VALUES ('" + userId + "','" + fecha +
"',0,'" + fecha + "','" + fecha + "','" + fecha + "','" + fecha + "','" + fecha + "')";
return ejecutarSentenciaSQL(sentenciaUsuarios) &&
ejecutarSentenciaSQL(sentenciaEvaluacion);
```

El método insertarNuevoMedico introduce una nueva entrada en la tabla de usuarios de la BBDD correspondiente a un médico. Como para nuestra aplicación no nos es imprescindible lo hacemos porque otro grupo depende de nosotros.

```
if (!containsUsuario(userId))
{
        String sentencia = "INSERT INTO usuarios
            (nombre,apellidos,userid,password,email,medico,googleId,moodleId,altacurso," + "rolcurso)
     VALUES " + "('" + userId + "','" + userId + "','" + userId + "','" + pass + "','" + userId + "','"
     + "" + "','" + "" + "',0,0,5)";
        return ejecutarSentenciaSQL(sentencia);
}
```

Paquete Controladores

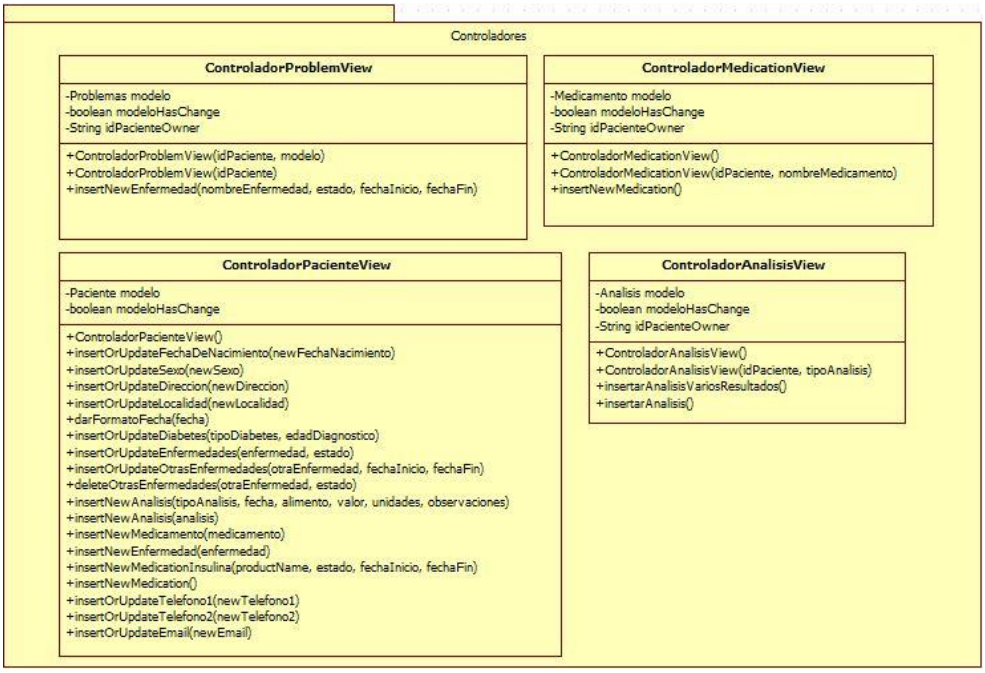


Figura 5-17 Contenido del paquete Controladores

El controlador ofrece toda la lógica necesaria para modificar el modelo asociado por medio de las vistas. Además se encarga de actualizar la Base de Datos con los cambios realizados a nivel local en el modelo usando las herramientas ofrecidas por el módulo de conexión con las BBDD.

Como podemos observar en la Figura 5-17 todos los controladores tienen 3 atributos: un modelo, el identificador del paciente y un atributo de tipo *boolean* que indica si el modelo ha cambiado.

Como podemos observar en la Figura 5-17 las 4 clases tendrán diferentes métodos para insertar o actualizar los datos en la base de datos. Por ejemplo: *insertNewEnfermedad*, *insertOrUpdateFechaDeNacimiento*, *insertOrUpdateSexo*, *insertOrUpdateDireccion*, etc. Para insertar o actualizar una enfermedad se hace de la siguiente manera:

```
if (estado && (modelo.getEnfermedades().get(enfermedad) == null)) {
        Problemas problema = new Problemas(enfermedad, "activo");
        String nuevoId = new ProblemsDAOGH().insertProblem(new ProblemsDTO(problema),
modelo.getId());
        if (nuevoId != null) {
            problema.setIdInterno(nuevoId);
            modelo.setEnfermedades(enfermedad, problema);
            modeloHasChange = true;
        }
}
else {
        if (!estado && modelo.getEnfermedades().get(enfermedad) != null) {
            ProblemsDAO dao = new ProblemsDAOGH();
            //Si se borra el valor actual correctamente
            if (dao.removeProblem(new ProblemsDTO(modelo.getEnfermedades(enfermedad)),
modelo.getId())){
                //Lo cambiamos en el modelo
                modelo.removeEnfermedades(enfermedad);
                modeloHasChange = true;
            }
        }
}
```

Paquete Modelos

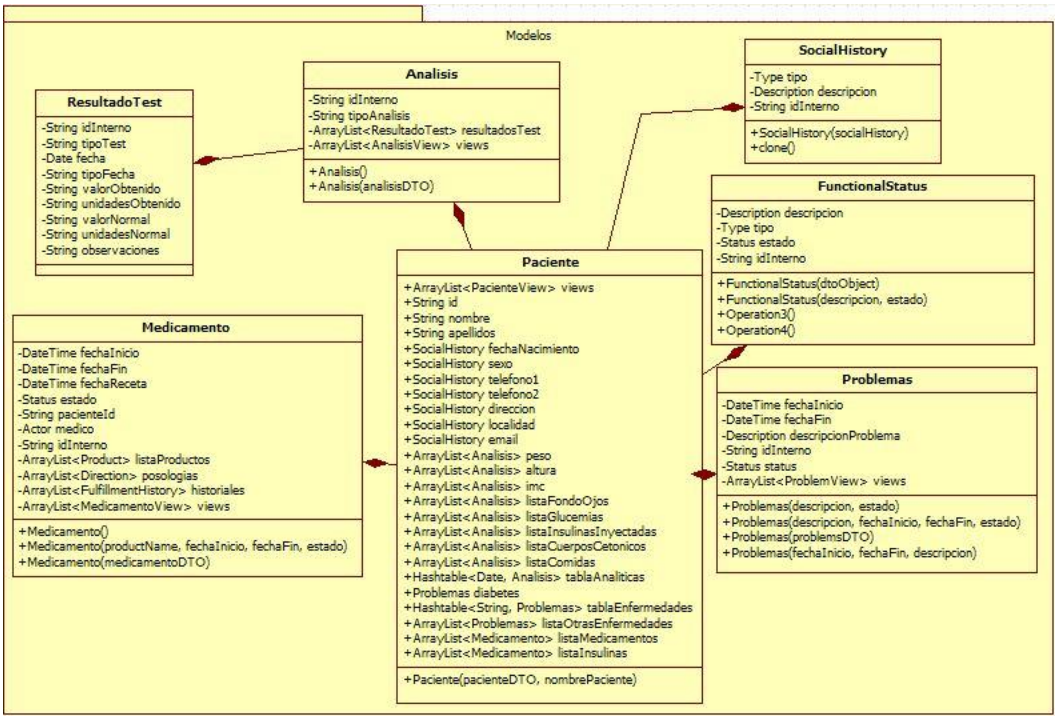


Figura 5-18 Contenido del paquete Modelos

El paquete modelo tiene como objeto que está siendo observado por una o varias vistas (implementa la interfaz *Observable*), todas las clases contiene toda la lógica necesaria para editar y acceder a sus atributos.

Como podemos observar en la Figura 5-18 el paquete modelos está formado por 7 clases: Medicamento, Problemas, FunctionalStatus, SocialHistory, Analisis, ResultadoTest y Paciente. Cada una ellas, excepto la clase Paciente, están compuestas siguiente en función de nuestra necesidad el

formato CCR. Hemos eliminado ciertos atributos innecesarios para nuestra aplicación.

La clase paciente está formada por unos atributos que hemos necesitados en nuestra aplicación, pero podrían ser aumentados o disminuidos en cualquier momento si fuese necesario.

En este caso la clase tiene como atributos: nombre, apellidos, fecha de nacimiento, sexo, teléfonos, dirección, localidad, email, pesos, alturas, IMC, lista de fondo de ojos, lista de glucemias, lista de insulinas inyectas, lista de cuerpos cetónicos, una tabla de las analíticas, tipo de diabetes, tabla de enfermedades, lista de medicamentos y listas de insulina. La manera de guardar estos datos en la aplicación, ha sido por decisión propia y sopesando los pro y contra de cada uno de ellos.

Como podemos observar en la Figura 5-18 la fecha de nacimiento, el sexo del paciente, los teléfonos, dirección, localidad y el email son atributos de tipo *SocialHistory*. Ya que estos atributos solo necesitan almacenar un valor que guardaremos en el campo *Type* del SocialHistory y en el campo Description se indicara el tipo de atributo (sexo, teléfono1, teléfono2,...).

Como observamos en la Figura 5-18 las medidas de peso, altura, IMC, fondos de ojos, glucemias, insulinas inyectadas, cuerpos cetónicos y comidas son atributos de tipo Análisis, estos atributos se almacenan de la siguiente manera: guardaremos un array de *ResultTest* con sus unidades, valores, observaciones de los distintos Análisis, y, para poder diferenciarlos en la lectura de los datos, indicaremos en el atributo Description la clase de Análisis.

Además tendremos una tabla de analíticas ordenadas por fecha y una tabla de enfermedades de tipo Problemas. Por último tendremos una lista de medicamento y de insulinas que serán de tipo Medicamento. De este modo podremos guardar las fechas de inicio y de fin, el estado, etc.

El código siguiente se corresponde con el proceso de carga de todos los datos de un paciente:

```
public Paciente(PacienteDTO pacienteDTO, String nombrePaciente){
id = pacienteDTO.getId();
inicializarArrayList();
if (nombrePaciente.contains(",")) {
        apellidos = nombrePaciente.split(",")[0];
        nombre = nombrePaciente.split(",")[1];
}else {
        nombre = nombrePaciente;
}
Iterator it;
//Rellenamos la información personal del paciente
if (pacienteDTO.getSocialHistories() != null) {
        it = pacienteDTO.getSocialHistories().iterator();
        SocialHistoryDTO socialHistory;
        //Clasificamos cada elem SocialHistory según el contenido de Description
        while (it.hasNext()) {
           socialHistory = (SocialHistoryDTO) it.next();
           if (socialHistory.getDescription().getText().equalsIgnoreCase("Telefono1")) {
              telefono1 = new SocialHistory(socialHistory);
           }else if(socialHistory.getDescription().getText().equalsIgnoreCase("Telefono2")){
              telefono2 = new SocialHistory(socialHistory);
           }
           [...]
           }else
if(socialHistory.getDescription().getText().equalsIgnoreCase("FechaNacimiento")){
              fechaNacimiento = new SocialHistory(socialHistory);
           }
        }
}
//Rellenamos las medidas registradas del paciente (glucemias, insulinas, etc.)
if (pacienteDTO.getAnalisis() != null) {
        it = pacienteDTO.getAnalisis().iterator();
        AnalisisDTO analisisDTO;
        //Clasificamos cada elem de tipo Análisis según el contenido de su campo Description
        while (it.hasNext()) {
           analisisDTO = (AnalisisDTO) it.next();
           if (analisisDTO.getDescripcion().getText().equalsIgnoreCase("Glucemia")) {
              insertarNuevoElemEnLista(new Analisis(analisisDTO), listaGlucemias);
           }else if(analisisDTO.getDescripcion().getText().equalsIgnoreCase("Insulina")){
              insertarNuevoElemEnLista(new Analisis(analisisDTO),listaInsulinasInyectadas);
           }
           [...]
           }else if(analisisDTO.getDescripcion().getText().equalsIgnoreCase("Comidas")){
              insertarNuevoElemEnLista(new Analisis(analisisDTO), listaComidas);
}
        }
}
//Rellenamos la información de las dolencias y enfermedades del paciente
if (pacienteDTO.getProblemas() != null) {
        it = pacienteDTO.getProblemas().iterator();
        ProblemsDTO problemDTO;
        [...]
}
//Rellenamos la información de las dolencias y enfermedades del paciente
if (pacienteDTO.getMedicamentos() != null) {
        it = pacienteDTO.getMedicamentos().iterator();
        MedicamentoDTO medicamentoDTO;
        [...]
}
}
```

A continuación en la Figura 5-19 se muestra un diagrama de secuencia de una inserción de un medicamento.

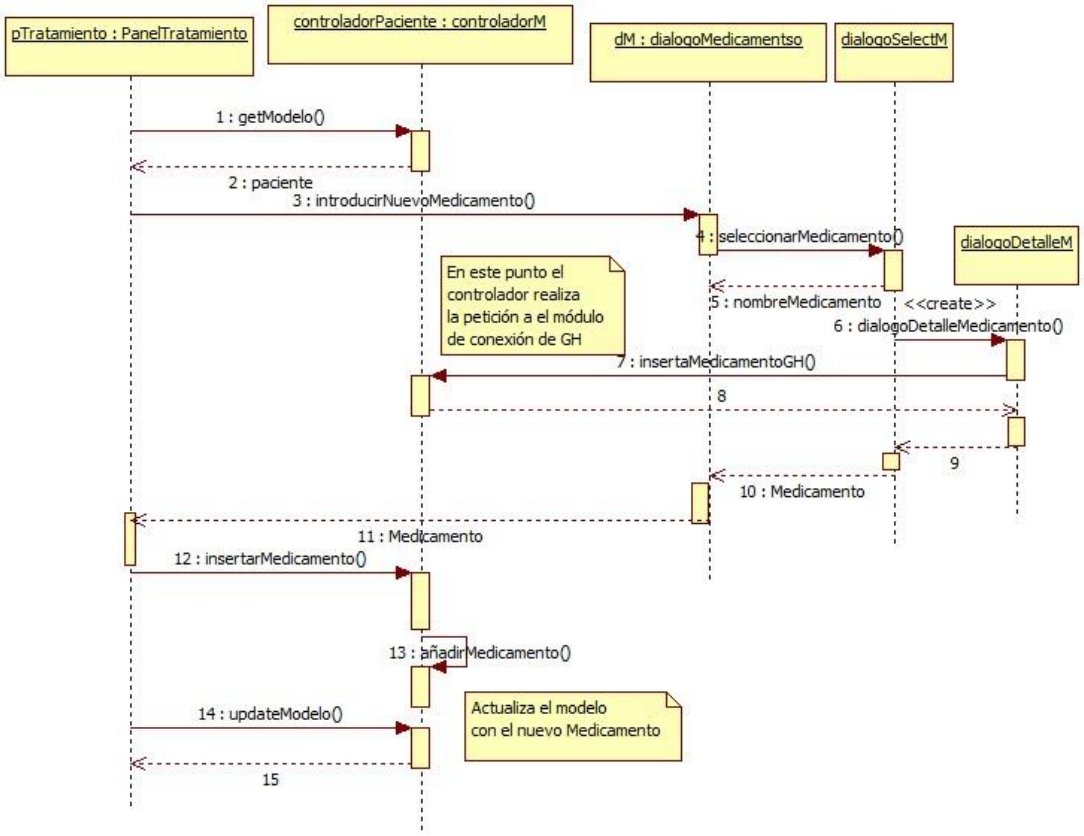


Figura 5-19 Diagrama de secuencia para inserción de un nuevo medicamento

En la figura anterior, observamos el diagrama de secuencia de la inserción de un medicamento. La inserción de otro elemento funcionaría de manera parecida. El panel donde el usuario está trabajando recibe una petición de inserción de un nuevo medicamento. Posteriormente tras obtener el modelo del Paciente para pasarlo como parámetro al nuevo diálogo realizamos una petición al *dialogoMedicamentos*. Éste dialogo realiza una llamada al dialogo de selección del medicamento que deseamos introducir, devolviendo el nombre del medicamento seleccionado. Posteriormente se crea un dialogo para rellenar los campos del medicamento. Una vez validada la creación se sirve la petición de inserción a GH del medicamento a través del controlador. Este último utilizará el módulo de conexión a GH para insertarlo en GH. Tras la inserción en GH se devuelve el objeto Medicamento insertado hasta llegar al panel inicial. En este panel se le solicita al conector que actualice el modelo añadiéndole al paciente el nuevo medicamento.

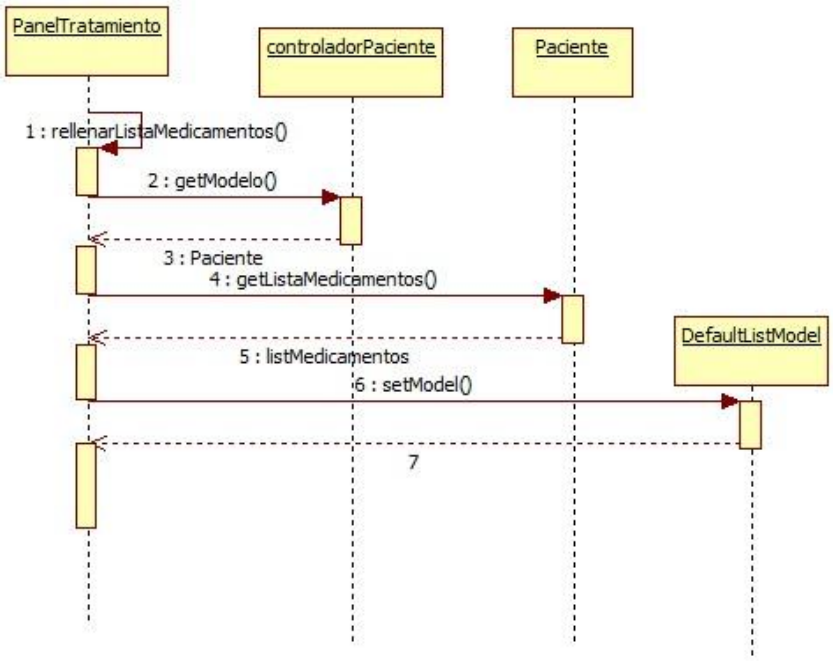


Figura 5-20 Diagrama de secuencia para rellenar una vista con los datos del modelo

En esta otra figura (Figura 5-20) observamos la operación inversa. Se rellena la tabla con los medicamentos presentes en el modelo del paciente. Esta operación se lleva a cabo cuando se accede el panel de tratamiento. En la constructora del panel se llama al método *rellenarListaMedicamentos*() el cual se encarga de establecer el modelo (DefaultListModel) de la tabla de medicamentos. Como observamos cuando se accede a un panel nuevo no se descarga la información si no que sólo se descarga al iniciar la aplicación, posteriormente cada vez que se necesite algún dato sobre el paciente se obtendrá del objeto modelo paciente.

Paquete Herramientas

En el paquete de herramientas nos encontramos con distintas clases auxiliares que son utilizadas en otros paquetes.

La clase *GlucemiasXML.java* se utiliza cuando extraemos los datos del XML de los glucómetros. Todos los datos que iremos obteniendo lo guardaremos en un ArrayList de tipo *GlucemiasXML*. Esta clase está formada por 3 atributos: fecha, periodo y glucemia.

La clase *CargadorDatosFichero.java*, tiene dos funciones. La primera extraer información desde un fichero y la segunda buscar un elemento dado en un listado.

Las clases Gráfica, GraficaPesoAlturaImc y GraficaGlucemias son las encargadas de llevar a cabo la gestión de las gráficas de la aplicación. A continuación se muestra un diagrama de la arquitectura del módulo:

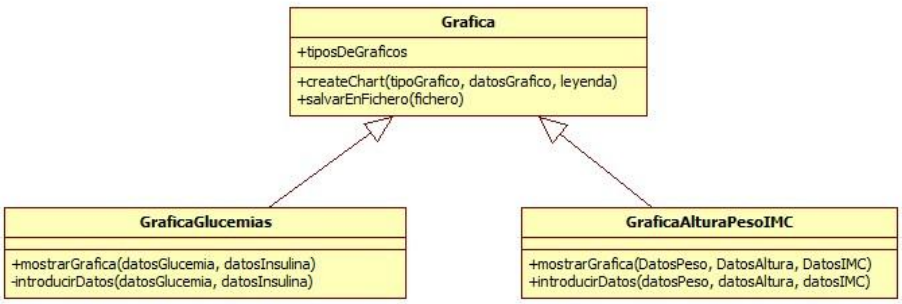


Figura 5-21 Estructura de las clases usadas para las gráficas

Como vemos en la Figura 5-21, existen 2 clases diferentes para representar gráficas. Una es *GráficasGlucemias.java*, encargada de representar las gráficas para las glucemias e insulinas, y *GráficaAlturaPesoIMC.java*, que realiza la misma labor con el peso, la altura y el IMC. Ambas clases extienden a la superclase Gráfica que contiene una serie de métodos auxiliares del que se benefician todas las clases que la extiendan. Para representar las gráficas se ha utilizado la librería *jFreeChart*. Toda clase que quiera representar una gráfica deberá extender la clase Gráfica e importar esta librería. El flujo para representación de una gráfica es el siguiente:

La clase correspondiente recibe una petición a través del método *mostrarGráfica* recibiendo como parámetro los datos de los elementos a mostrar. Este método delega la petición a *introducirDatos* que se encarga de crear un objeto *DefaultCategoryDataset* con los datos y nombres de los elementos introducidos. Posteriormente el método *mostrarGrafica* se encargará de crear un panel con los datos del *objetoDefaultCategoryDataset* y de tipo *ChartPanel*. Este panel será devuelto a la aplicación para que se pueda visualizar donde se considere oportuno.

Una visión esquemática del código podría ser la siguiente:

```
public static ChartPanel mostrarGrafica(ArrayList<Analisis> datos1, ArrayList<Analisis> datos2){
        // Introduzco los datos en el objeto dt de tipo DefaultCategoryDataset
        DefaultCategoryDataset dt = introducirDatos(datos1, datos2);
        //Creo el objeto Chart con el metodo de la superclase
        JFreeChart  chart = createChart(dt, tipoElegido, titulo, columna, fila);
        // Creo el panel a partir del Grafico
        ChartPanel panel = new ChartPanel(chart);
        // Lo devuelvo para que pueda ser mostrado donde se desee
        return panel
}
```

Las clases *CellRender* se usarán para configurar el aspecto visual que tendrán las tablas usadas en los paneles de Dieta, paneles principales tanto para medico como para paciente y en el panel de Insulinas y glucemias. Con *CellRenderInsulinasGlucemias.java* determinaremos el color de fondo y el color de los valores en caso de híper/hipoglucemias. Con *CellRenderComidas.java* determinaremos el color de fondo para una celda según el número de fila y

finalmente con *CellRenderJButton.java* y *CellRenderIcon.java* podremos crear tablas con botones y etiquetas (respectivamente) dentro de sus celdas.

La clase *ColumnGroup.java*, permite crear tablas con doble cabecera como puede observarse en el panel de insulinas y glucemias.

CommonTools.java contiene una serie funciones comunes usadas desde múltiples clases de la GUI, tales como centrarVentana(...), formatearFecha(...), etc.

A continuación se va a mostrar unos diagramas de secuencia para explicar los distintos pasos que se dan para dar de alta a un paciente en la aplicación, entrada del paciente en la aplicación.

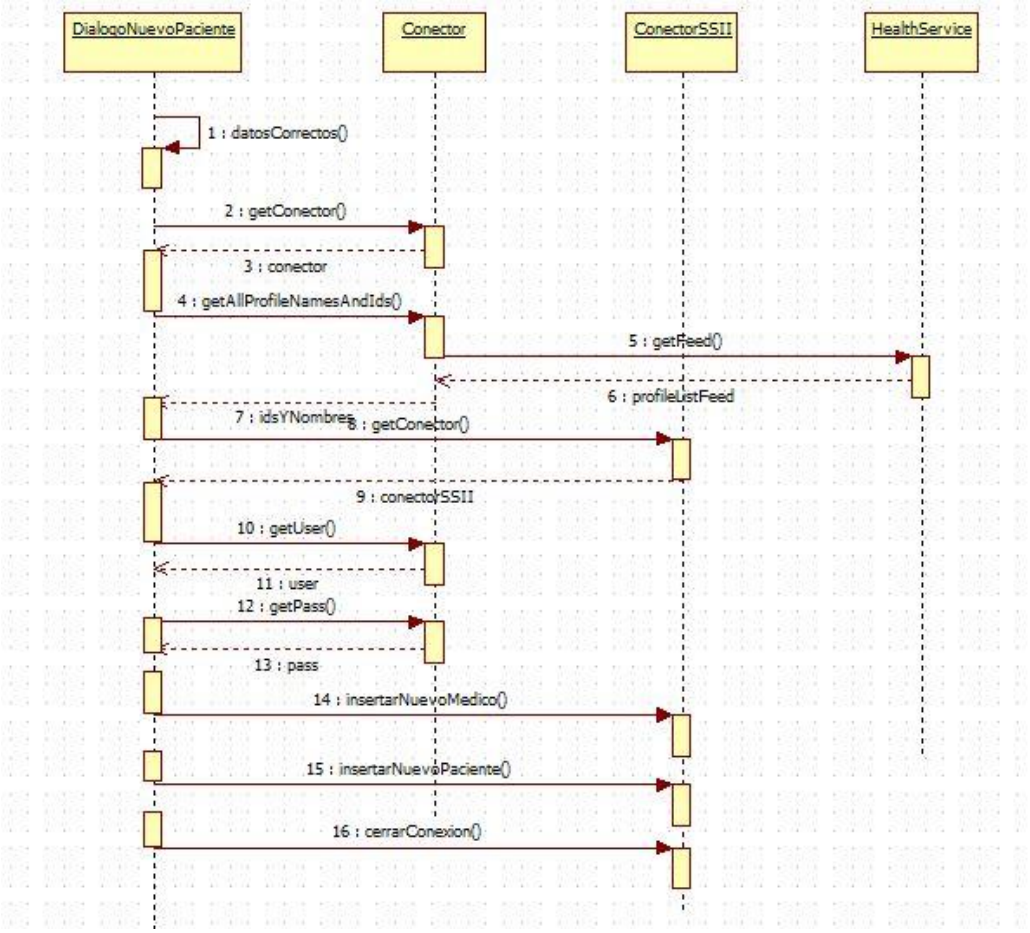


Figura 5-22 Diagrama de secuencia para alta de paciente

Este diagrama de Figura 5-22 representa el proceso de registro de un nuevo paciente (y del propio médico si fuese necesario) en la base de datos. Para ello, el médico deberá introducir en un panel el nombre, apellidos, email, usuario de la aplicación y contraseña para el nuevo paciente. Inicialmente se comprueba que estos datos se han introducido correctamente. Por motivos ajenos a nosotros, GH no permite dar de alta a un paciente a través del API de Java, por este motivo hemos necesitado crearnos una base de datos temporal donde almacenamos tanto los médico como los paciente. El segundo paso que realizamos es buscar el identificador interno del paciente que vamos a añadir a

nuestra base de datos externa. Para ello lo que hacemos es obtener todos los nombres y usuarios de GH y sus identificadores internos con el método *getAllProfileNamesAndIds*. Comparamos el nombre de los usuarios con el que ha rellenado el médico en el panel y obtenemos el identificador interno de Google. Esto es necesario realizarlo así, ya que para la conexión del paciente es necesario conocer el identificador interno. El paso siguiente es introducir al paciente en la base de datos y al médico. El médico solo lo introduciremos en el caso de que no exista con anterioridad y así evitarnos tener médicos duplicados. Por último cerraremos la conexión con la base de datos.

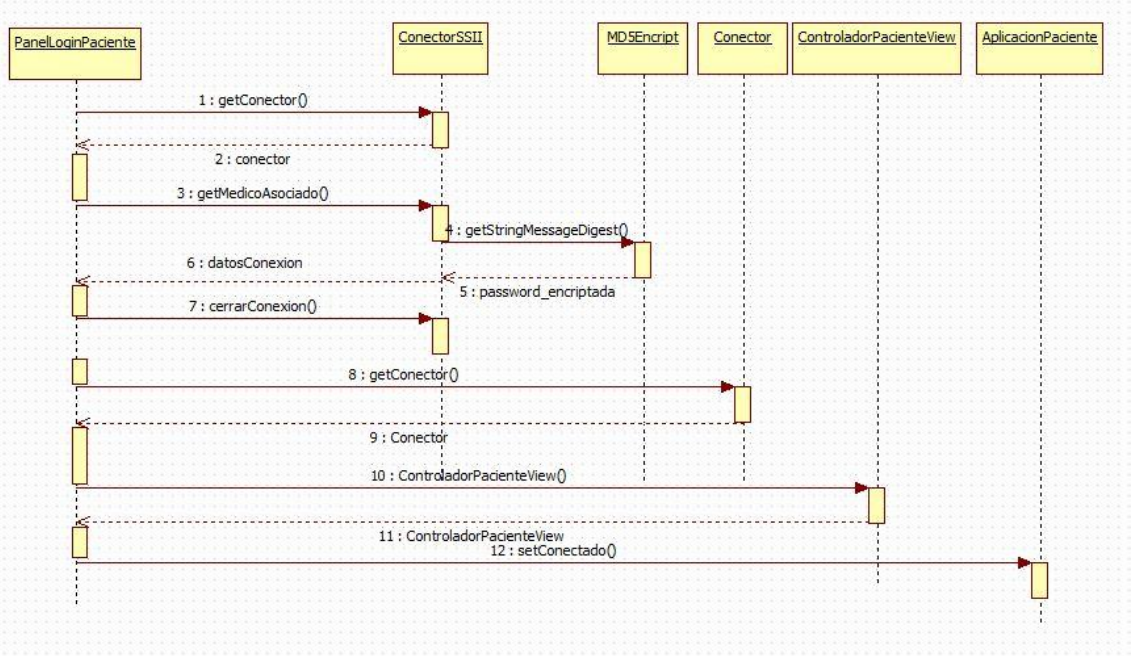


Figura 5-23 Diagrama de secuencia para conexión del paciente a la aplicación

Como podemos observar en la Figura 5-23 lo primero que se realiza es la conexión con la base de datos. Para ello como hemos nombrado en este documento con anterioridad, aplicamos el patrón *Singleton* por tanto si el conector no estaba creado todavía, se crea y se devuelve el conector en caso contrario simplemente devolvemos el conector. El siguiente paso es obtener a partir del usuario y contraseña del paciente el medico asociado a él. Para ello llamamos a la función *getMedicoAsociado* que realiza dos consultas a la tabla de usuarios y devolverá 5 campos: Nombre y apellidos del paciente, Usuario y contraseña del médico asociado y el identificador de GH del paciente. El paso siguiente será cerrar la conexión con la base de datos y crear una nueva conexión con la base de datos de GH. Como tenemos los datos necesarios que son el usuario y contraseña del médico, simplemente será hacer un *getConector* con estos campos y nos devolverá el conector. Por último crearemos un *controladorPacienteView* y ya estaremos autentificados en la aplicación.

5.3.3 Descripción de la GUI

A la hora de diseñar la parte visual de la aplicación, se hizo especial hincapié en que el resultado fuera un entorno simple, completo y, especialmente en la parte diseñada para el paciente, intuitivo. Además, aunque de manera secundaria, se intentó compaginar lo anterior con un diseño visualmente atractivo.

Con este fin, se buscó la opinión de posibles usuarios reales como un endocrinólogo y un diabético tipo 1, cuyas críticas, observaciones y experiencias, sirvieron para determinar no sólo el mejor aspecto visual sino aquellas las funcionalidades básicas que nuestra aplicación ofrece.

Como resultado de este estudio, se decidió:

Dividir la aplicación en dos aplicaciones: una para el médico (con todas las funcionalidades) y otra para el paciente (minimizando las funcionalidades). Desde su experiencia, el usuario diabético nos indicó que la gran mayoría de las aplicaciones para el control de la diabetes ofrecían excesivas opciones, lo cual se traduce en aplicaciones demasiado complejas y poco atractivas para un usuario con conocimientos básicos de informática. Por otro lado, el médico endocrinólogo hizo una observación muy importante: gran parte de los pacientes con diabetes Tipo 2 superan los 50 años y, a diferencia de las nuevas generaciones, su conocimientos de informática suelen ser muy básicos. Ante esta situación, se ha procurado que la aplicación para el paciente fuera lo más simple e intuitiva posible

Usar un diseño estilo “aplicación Web”. Pese a tratarse de una aplicación Java, se ha intentado dotar a la GUI de un aspecto de aplicación Web. La razón la encontramos en que la mayoría de los usuarios de ordenador están más acostumbrados a este escenario, que a aplicaciones de escritorio.

Finalmente el diseño logrado para ambas aplicaciones puede observarse en la Figura 5-24 Aspecto de la versión de GluControl para el médico y en la Figura 5-25 Aspecto de la versión de GluControl para el paciente.

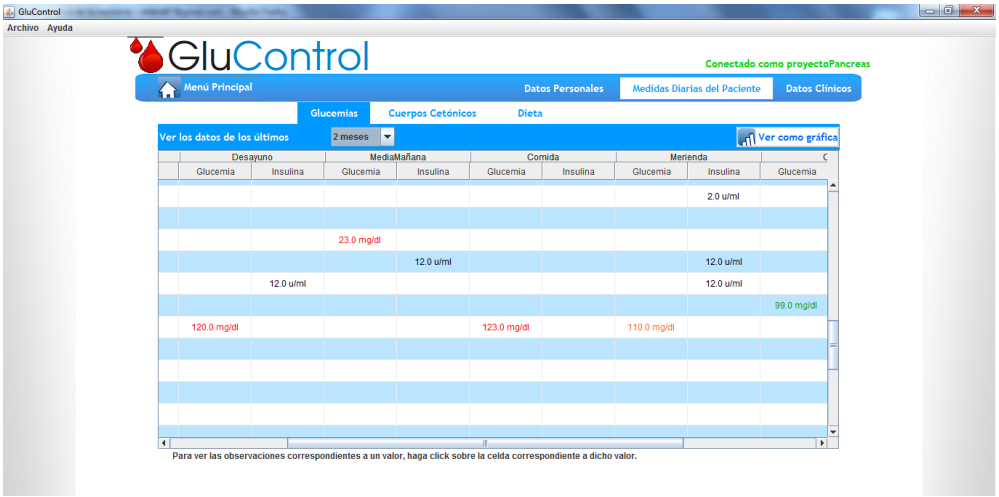


Figura 5-24 Aspecto de la versión de GluControl para el médico

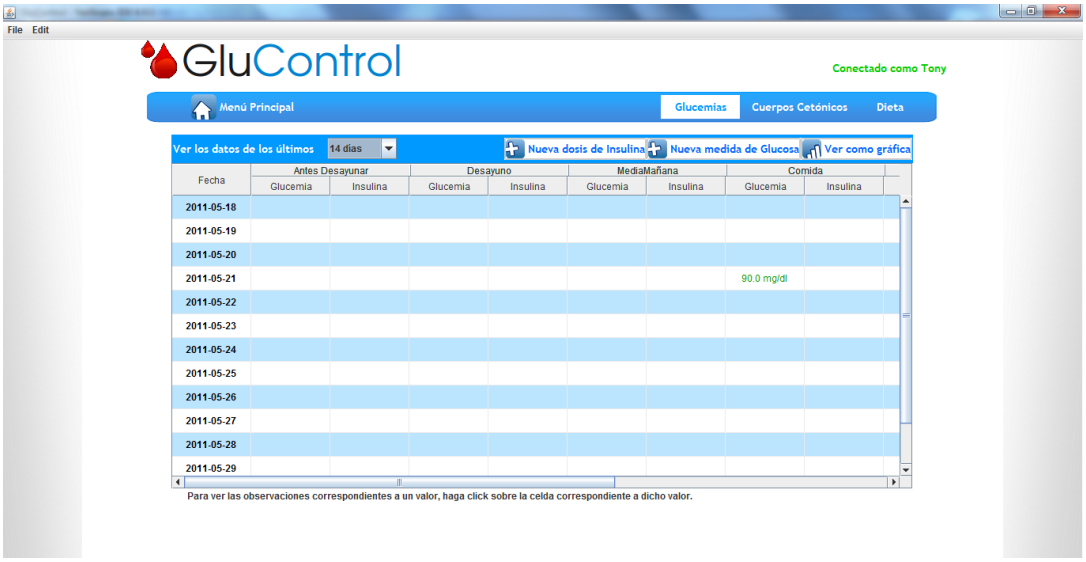


Figura 5-25 Aspecto de la versión de GluControl para el paciente

Descripción de la interfaz

Pese a haber dividido la aplicación en 2, una para el paciente y otra para el médico, ambas partes comparten la gran parte de los paneles y diálogos, diferenciándose en las opciones disponibles en cada caso. A continuación explicaremos aquellas vistas más representativas de nuestra aplicación:

Vista para Datos Personales

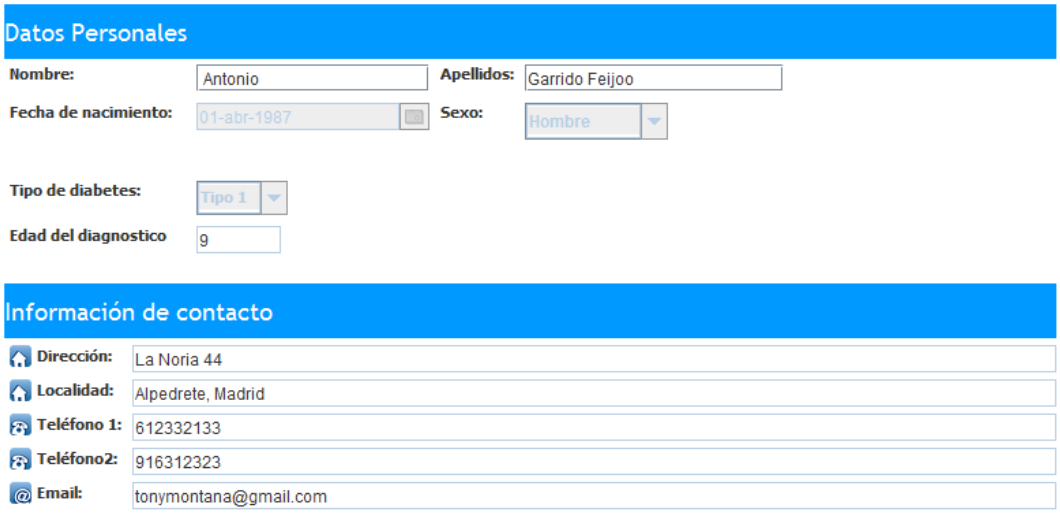


Figura 5-26 Panel con los datos personales del paciente

Como puede observarse en la Figura 5-26, este panel ofrece información personal de un paciente. Además de la información de contacto (nombre completo, edad, domicilio, teléfono, correo electrónico, etc.), indica otra información importante como el tipo de diabetes que padece y la fecha de diagnóstico. Este panel se ha omitido para el paciente pues sólo aumentaría la complejidad de la aplicación.

Vistas para el control de Glucemias e Insulinas

Por medio de este panel, representaremos las glucemias e insulinas administradas de cierto paciente. Tanto el médico como el paciente podrán observar estos valores ya sea por medio de una tabla de valores (en intervalos que van desde 1 semana a 2 meses atrás) como a través de gráficas, no obstante, sólo el paciente dispondrá además de opciones para introducir nuevos registros de insulinas y glucemias.

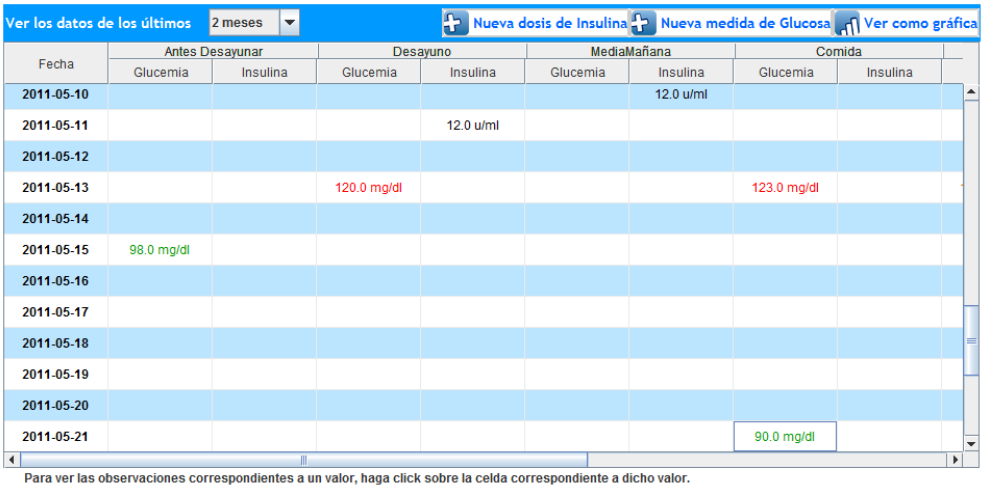


Figura 5-27 Panel de Glucemias e Insulinas para la aplicación del paciente

En la Figura 5-27 podemos observar una tabla con los valores de insulinas y glucemias del paciente autentificado. Obsérvese que, a diferencia de la vista para el médico aparecen las opciones “Insertar nueva glucemia” e “Insertar nueva insulina” (Figura 5-28), que dan acceso a los diálogos para registrar nuevas Glucemias (Figura 5-29) y nuevas Insulinas (Figura 5-30), respectivamente.

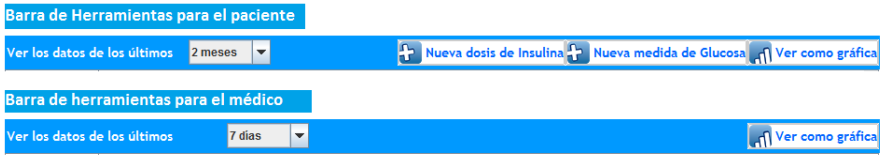


Figura 5-28 Diferencia entre las opciones ofrecidas al paciente y las ofrecidas al medico

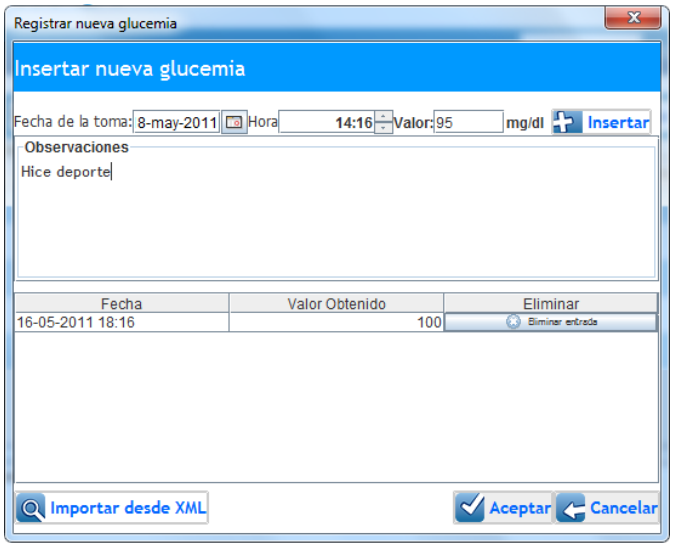


Figura 5-29 Dialogo para registrar nuevas glucemias

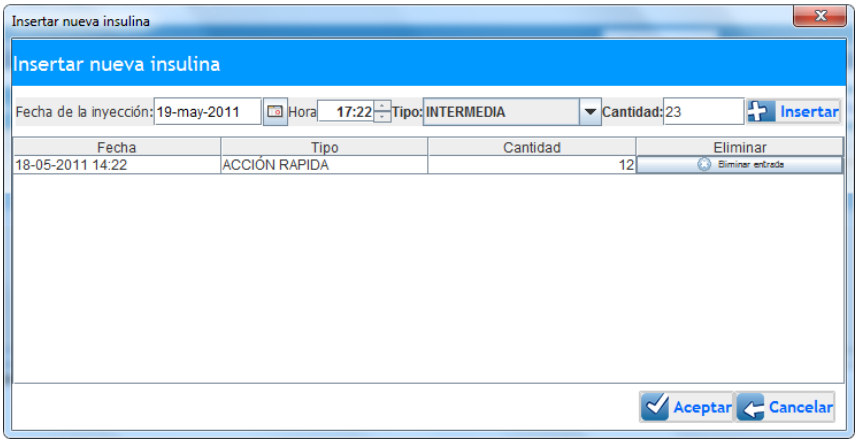


Figura 5-30 Dialogo para registrar nuevas dosis de insulina

Vistas para medidas de Cuerpos Cetónicos

Al igual que en el caso de la glucemias, este panel es compartido entre la aplicación del médico y la del paciente. La principal diferencia la encontramos en que el médico sólo puede observar aquellos valores registrados por el paciente. En el panel principal, se listarán aquellos valores de Cuerpos cetónicos registrados, ordenados por fecha y se ofrecerán las opciones “Ver en detalle” e “Insertar nueva medida” (sólo para el paciente). Ambas opciones darán acceso al diálogo de detalles de cuerpos cetónicos, ya sea para consultar un valor anteriormente registrado o para rellenar los datos de una nueva medida de cuerpos cetónicos. En la Figura 5-31 se puede observar el aspecto del panel de cuerpos cetónicos para el paciente y en la Figura 5-32 un ejemplo de registro de una nueva medida mediante el diálogo de detalles.

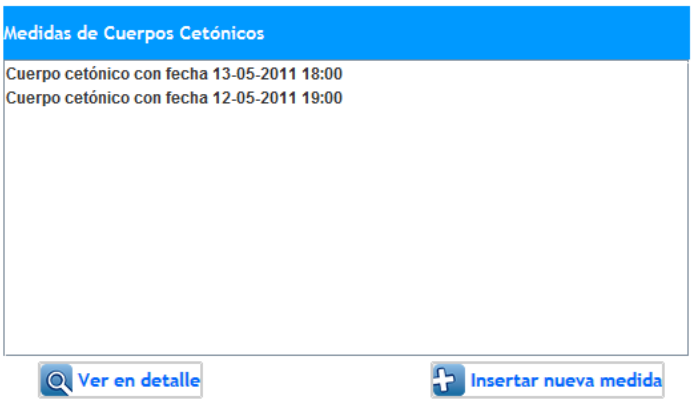


Figura 5-31 Panel de medidas de cuerpos cetónicos para paciente

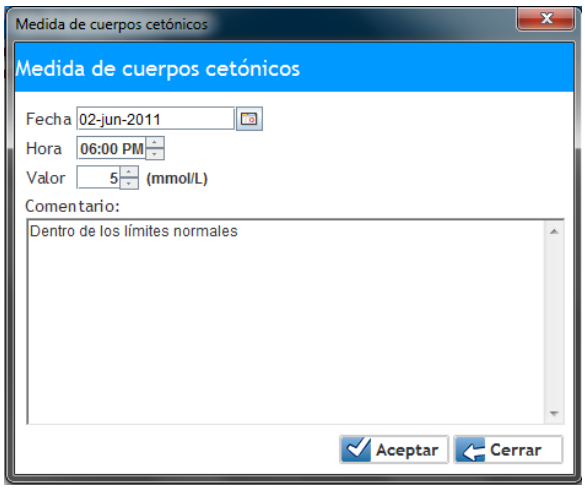


Figura 5-32 Diálogo para registrar una nueva medida de cuerpos cetónicos

Vistas para la dieta

Como podemos observar en la Figura 5-33, mediante esta tabla presentamos un registro de los hidratos de carbono ingeridos por el paciente a lo largo de día y las observaciones.

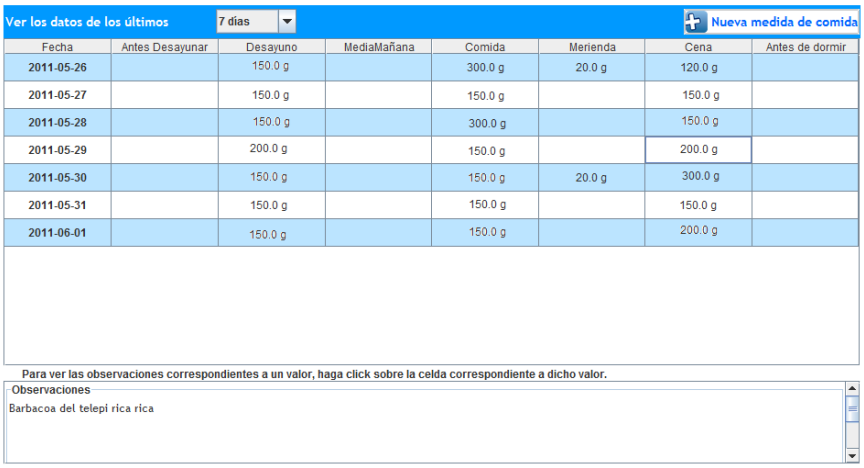


Figura 5-33 Panel con los valores de hidratos de carbono ingeridos durante la semana

Vistas para Pruebas Complementarias

Por medio de esta vista se representarán los listados de todas aquellas pruebas realizadas al paciente, tales como analíticas, exámenes de fondo de ojo y medidas de peso, talla e *índice de masa corporal* (IMC) (Figura 5-34). Tanto las analíticas como los fondos de Ojo pueden ser inspeccionados más en detalle o introducir nuevos resultados por medio de los diálogos Detalles de analítica (Figura 5-35) y Detalles de Fondo de Ojo (Figura 5-36), respectivamente. Las medidas de peso, talla e IMC pueden observarse por medio de gráficas (Figura 5-37). Esta vista sólo es accesible desde la aplicación del médico.

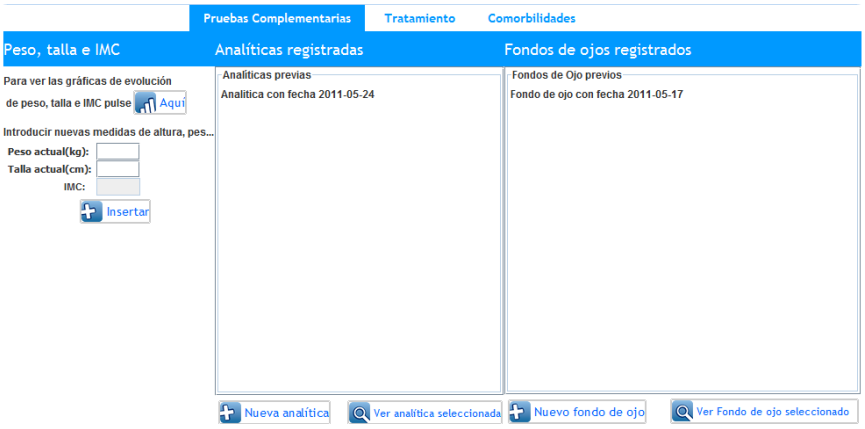


Figura 5-34 Panel para pruebas complementarias

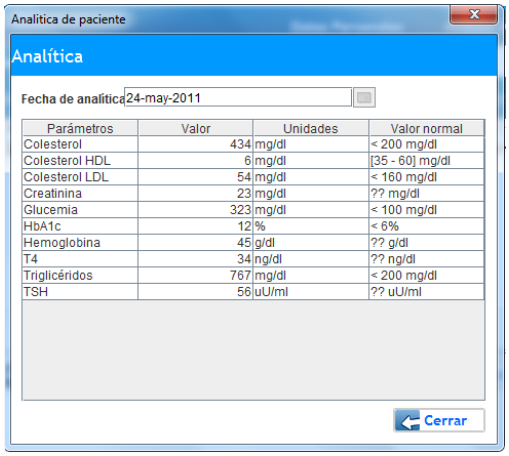


Figura 5-35 Dialogo con detalles de una analítica

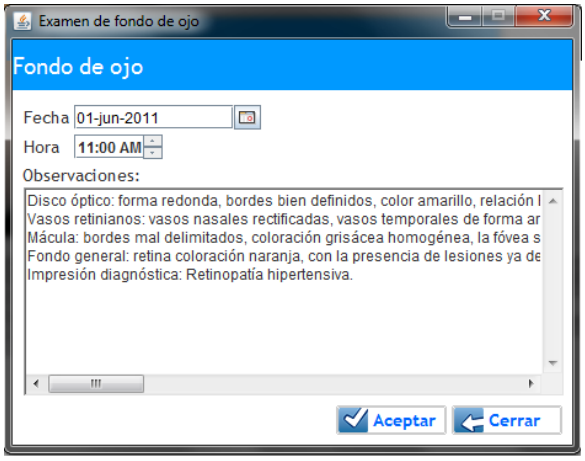


Figura 5-36 Diálogo con detalles de un examen de fondo de ojo

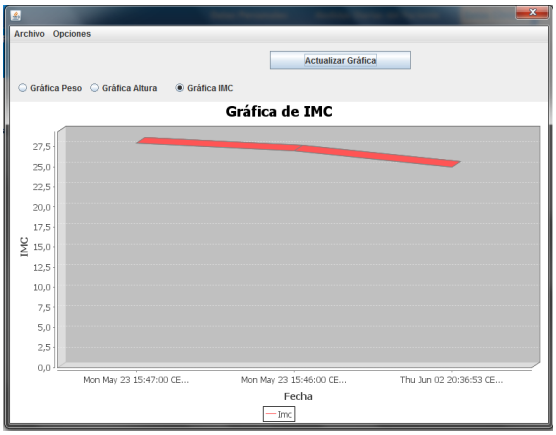


Figura 5-37 Gráfica con los valores de IMC calculados para un paciente

Vista de Tratamiento

Usada para listar aquellos medicamentos e insulinas que el paciente esté tomando o haya tomado y que sean relevantes para el tratamiento de la diabetes (Figura 5-38). En el caso de los medicamentos podrán ser observados más en detalle o registrar nuevos medicamentos usando por medio del Dialogo detalles del medicamento (Figura 5-39). Como se puede observar en la Figura 5-40, la aplicación ofrece un extenso listado de principios activos a la hora de registrar un nuevo medicamento.

Esta vista sólo es accesible desde la aplicación del médico.

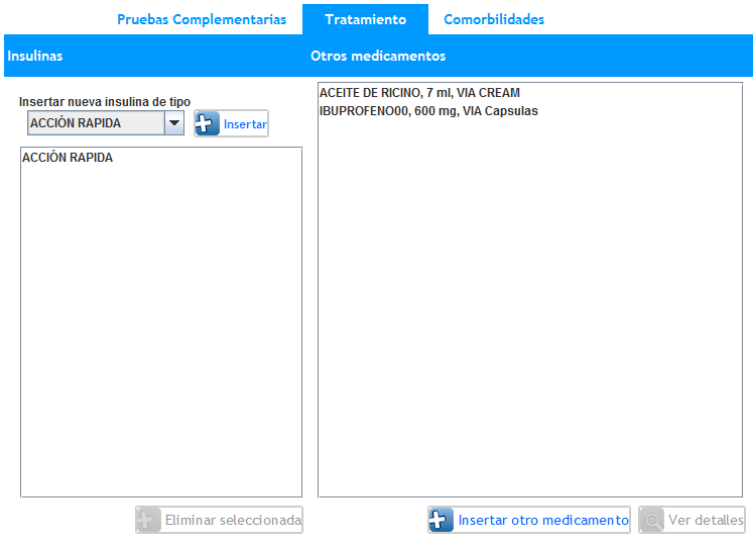


Figura 5-38 Panel con los datos del tratamiento para un paciente

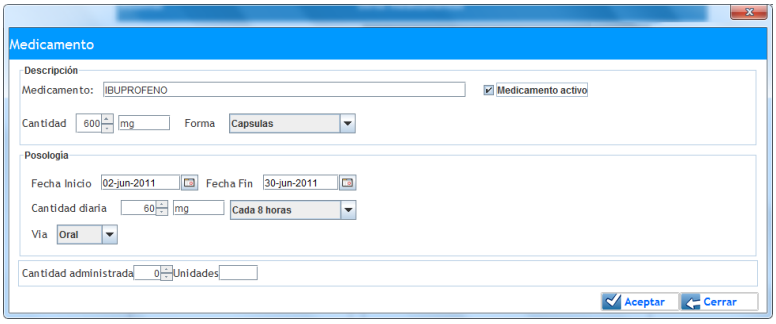


Figura 5-39 Diálogo con detalles de un medicamento

Figura 5-40 Listado de principios activos

Vista Comorbilidades

Por medio del panel de la Figura 5-41 se presentan todas aquellas enfermedades, dolencias y factores de riesgo relevantes para el paciente diabético. En el lado izquierdo se presenta un listado con los factores de riesgo, enfermedades autoinmunes y complicaciones propias de la diabetes. En el lado derecho encontramos un lista de otras enfermedades que padezca o haya padecido el paciente y que el médico considere relevantes para el tratamiento.

Dichas enfermedades podrán observarse con más detalle mediante el dialogo de Detalles de enfermedad (Figura 5-42), accesible mediante la opción “Ver detalles”. Este diálogo también se usará para rellenar los datos de las nuevas enfermedades que se registren. En la Figura 5-43 se muestra el amplio listado de enfermedades que se ofrece al seleccionar la opción “Añadir otra enfermedad”.

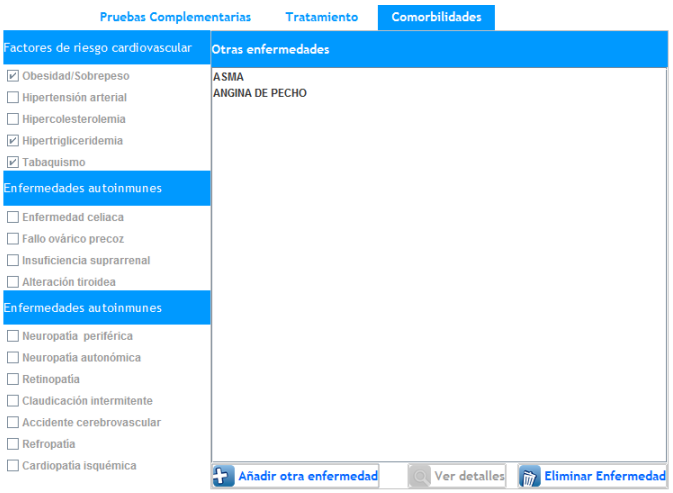


Figura 5-41 Panel de comorbilidades

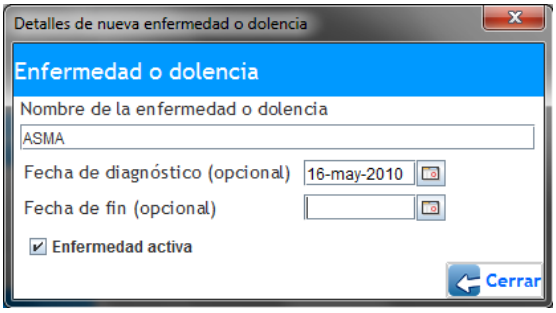


Figura 5-42 Dialogo con detalles de una enfermedad

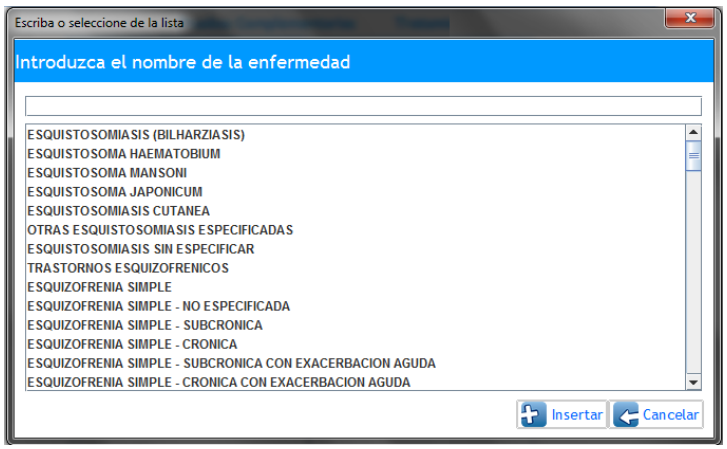


Figura 5-43 Listado de enfermedades y dolencias

Esta vista sólo es accesible desde la aplicación del médico.

Aplicación para el médico

Una vez autentificados, la aplicación del médico ofrecerá el menú de la Figura 5-44 en el que podremos elegir entre Dar de alta a un nuevo paciente y Consultar los datos de un paciente ya registrado.

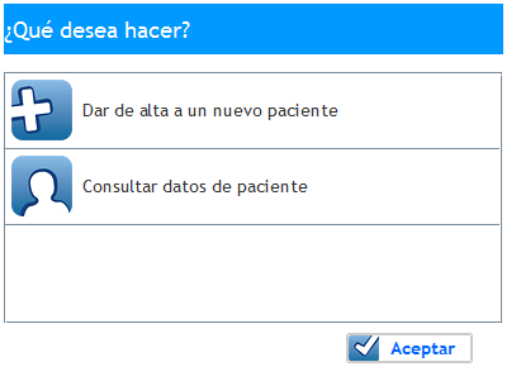


Figura 5-44 Menú principal para el médico

Eligiendo la primera opción, tendremos acceso al Dialogo de Alta de nuevo paciente, en el cual, mediante un breve tutorial, se nos irá guiando en el proceso de Alta de un paciente (ejemplo del proceso en la Figura 5-45).

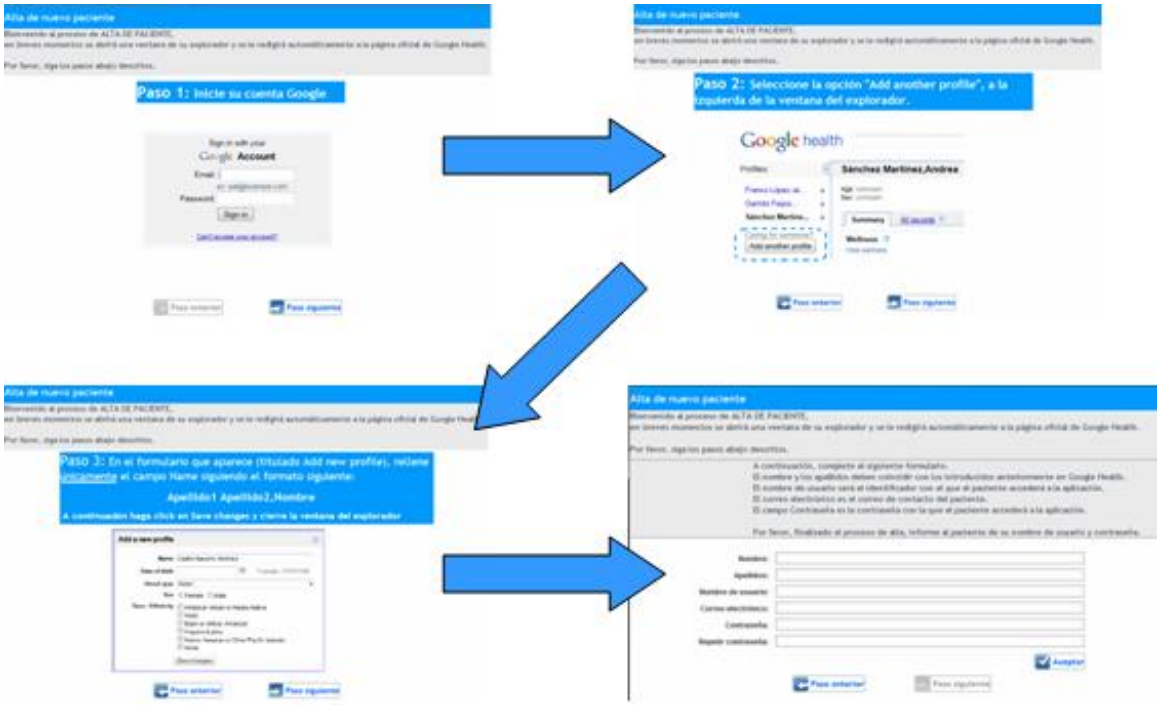


Figura 5-45 Proceso de alta de paciente

Mediante la segunda opción, accederemos al listado de todos los pacientes registrados para el médico (Figura 5-46). Eligiendo uno de ellos, podremos acceder a la información detallada de dicho paciente.

Figura 5-46 Aspecto del listado de pacientes registrados para el médico

La aplicación del médico, a diferencia de la del paciente, ofrece acceso a todas las vistas anteriormente descritas (salvo aquellas usadas para introducir nuevas medidas de insulina, glucosa, dietas y cuerpos cetónicos), que se han agrupado de la siguiente manera:

Datos personales, permite acceder al panel de Datos personales del paciente.

Medidas diarias, engloba aquellas vistas que ofrecen medidas registradas por el paciente de manera periódica: glucemias, insulinas administradas, dieta y cuerpos cetónicos.

Datos clínicos, incluye toda la información clínica relevante: pruebas complementarias (peso, talla e IMC, analíticas y fondos de ojo), tratamiento (insulinas y otros medicamentos) y comorbilidades (factores de riesgo cardiovascular, enfermedades o dolencias, etc.).

Aplicación del paciente

En el caso de la aplicación para el paciente, se han omitido gran parte de las vistas anteriores, en pos de reducir la complejidad. Por tanto, las opciones principales para el paciente se reducirán a aquellas que dan acceso a las vistas de Glucemias e insulinas, Dieta y cuerpos cetónicos. En la Figura 5-47 podemos ver el aspecto del menú principal para un paciente.

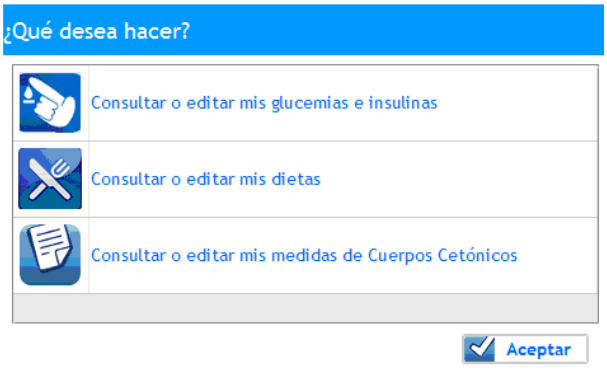


Figura 5-47 Aspecto del menú principal para el paciente

5.4 Integración en dispositivos móviles

Una de los principales objetivos de la telemedicina es facilitar y mejorar la calidad de vida del paciente. Para pacientes crónicos, como el caso de los diabéticos, este hecho adquiere aún más relevancia.

La irrupción de acceso a Internet vía teléfono móvil ha abierto una nueva puerta en este campo. En nuestro caso, nos hemos servido de esta tecnología con el objetivo de facilitar el acceso a la aplicación desde cualquier lugar y en cualquier momento. Un paciente con acceso a un móvil con conexión a Internet podrá introducir medidas de glucemia, insulina, las últimas comidas, etc. Esto supone una mejora importante en la facilidad de seguimiento de medidas de un paciente; mientras que si dispone de teléfono móvil con acceso a Internet podrá insertar medidas en cualquier momento, si no dispusiera, tendría que insertarlas al llegar a su ordenador personal.

Con la utilización de la aplicación en dispositivos móviles se consiguen dos cosas fundamentales, mejorar la calidad de vida del paciente y, por otra parte, facilitar el seguimiento de la enfermedad por parte del médico. El médico agradecerá que su paciente inserte las medidas nada más realizarse éstas y no, que se inserten con horas o días de retraso.

Hemos elegido Android como plataforma para la integración de la aplicación en sistemas móviles.

5.4.1 GluControl en Android.

Para Android hemos desarrollado una versión simplificada de la aplicación. Se trata de una versión que permite introducir, listar y borrar medidas de glucemia, insulina, comidas. La aplicación para Android ha sido desarrollada partiendo de un ejemplo que los desarrolladores de GH han elaborado. Se trata pues, de una aplicación que interactúa con GH descargando e insertando medidas desde el dispositivo Android.

Descripción de la aplicación Android.

Como puede observarse en la Figura 5-48, la aplicación consta de dos actividades. La primera *HealthAndroid.java* es la actividad principal y se encarga de la gestión general de la aplicación. Esta actividad representa el menú principal en el que aparece un listado de las medidas del paciente seleccionado. En este menú podremos seleccionar la opción que deseemos realizar: insertar medida nueva, borrar alguna existente o actualizar la lista de medidas. La otra actividad, *AddResultActivity.java*, es lanzada cuando deseamos introducir una nueva medida, se trata de un diálogo donde rellenaremos los datos y el tipo de la medida. Toda medida tendrá asociada un valor, una fecha y el tipo de medida que es.

Figura 5-48 Actividades en las que se divide la aplicación Android

Se utiliza autenticación automática, esto se realiza un chequeo de las cuentas Google (xxx@gmail.com) sincronizadas en el dispositivo móvil y, tras elegir la que se desea utilizar, realiza un acceso a GH a través de esa cuenta. De esta tarea se encarga el paquete Auth. Este paquete contiene las clases AccountChooser.java y AuthManager.java. La primera se encarga del proceso de selección de la cuenta que se desea utilizar, AuthManager se encarga de gestionar todo el proceso de selección y autentificación con la cuenta en GH.

Por otra parte la gestión de la conexión con GH se realiza a través del paquete gdata. La clase GDataHealthClient.java que hereda de HealthClient.java es la encarga de gestionar el intercambio de datos con GH. Labores como petición de pacientes, obtención de medidas de un paciente, envío de nuevas medidas, etc. Se realizan través de esta clase. Por otra parte, la clase TestResult.java y Result.java que heredan de CCRObject.java representan objetos de GH guardados en la aplicación.

Por último la clase HealthGDataContentHandler.java es utilizada por la actividad principal, HealthAndroid.java, para la gestión de las medidas de la aplicación. En la Figura 5-49 puede observarse la arquitectura de la aplicación resultante.

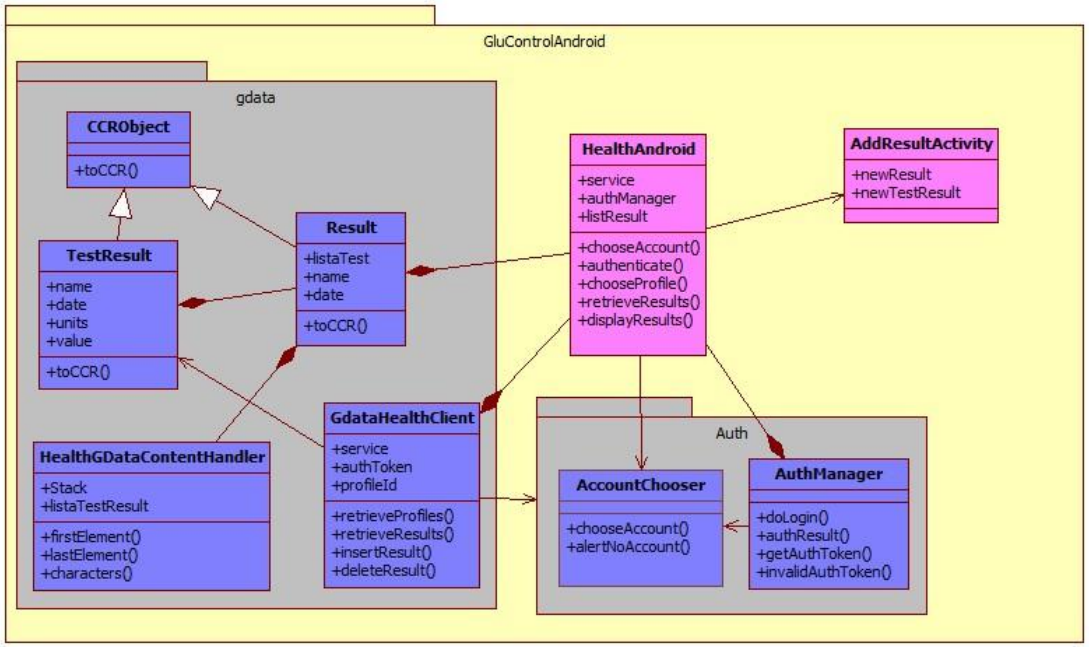


Figura 5-49 Arquitectura de la aplicación para Android

Funcionamiento de la aplicación.

El funcionamiento de la aplicación móvil es el siguiente:

Una vez iniciada la aplicación seleccionamos la cuenta Gmail que queremos utilizar por medio del menú de la Figura 5-50.

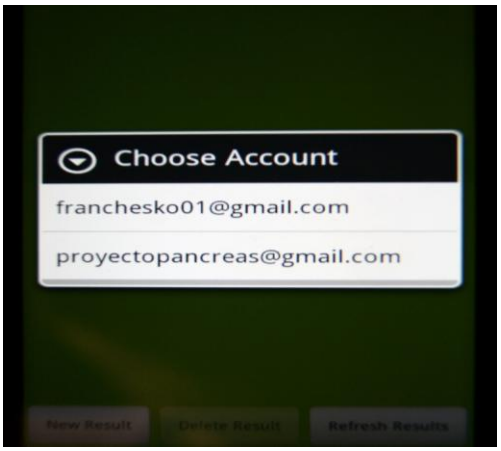


Figura 5-50 Menú de selección de la cuenta

Se cargarán los pacientes de dicha cuenta, seleccionamos el paciente deseado.

Para el paciente se cargarán todas las medidas, en este punto podremos:

Insertar una nueva medida. Si deseamos insertar una medida introduciremos el valor de la medida, la fecha y hora de la toma (usando los selectores de la Figura 5-51 y Figura 5-52) y el tipo de medida. Finalmente validaremos la inserción (Figura 5-53).

Figura 5-51 Selector de fecha para una nueva medida

Figura 5-52 Selector de hora para una nueva medida

Figura 5-53 Nueva glucemia lista para insertar en Google Health

Borrar una medida. Seleccionaremos la medida y validaremos la eliminación.

Actualizar la lista de medidas. Seleccionaremos dicha opción en el menú contextual.

5.5 Casos de uso: médicos y pacientes

Para poder utilizar la aplicación, tanto el médico como el paciente, tienen que tener acceso a Internet. Además el médico deberá tener una cuenta de Gmail donde tendrá registrados a todos sus pacientes.

5.5.1 Caso de uso del Médico

El médico comenzará a usar la aplicación introduciendo su usuario de Gmail y su contraseña. Le aparecerá un panel con dos opciones: (1) Dar de alta a un nuevo paciente, (2) Consultar datos de un paciente.

El primer paso que tiene que realizar el médico es “dar de alta al paciente”, esto le va a permitir después consultar los datos del paciente y a su vez que el paciente pueda introducir sus datos.

El alta de un paciente se realiza de manera guiada en 4 pasos. Primero accederá a Google Health. Se le abrirá en su explorador la página *www.google.com/health/* y tendrá que introducir su usuario y contraseña en el panel que muestra la Figura 5-54. En este momento podrá comenzar a dar de alta a los pacientes, para ello pulsaremos en la opción *“Add another profile”* (Figura 5-56). Automáticamente le aparecerá un formulario donde tendrá que rellenar únicamente el campo *Name*. Como muestra la Figura 5-57, la manera de rellenarlo debe tener el siguiente formato: apellido1 apellido2, nombre. A continuación hará *click* en el botón *Save Change* y cerrará el explorador. De esta manera habrá dado de alta a un paciente en Google Health.

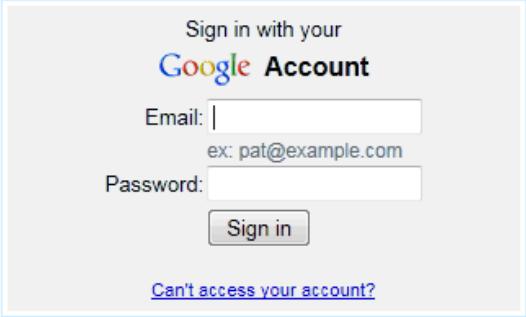


Figura 5-54 Panel para iniciar sesión en Google Health

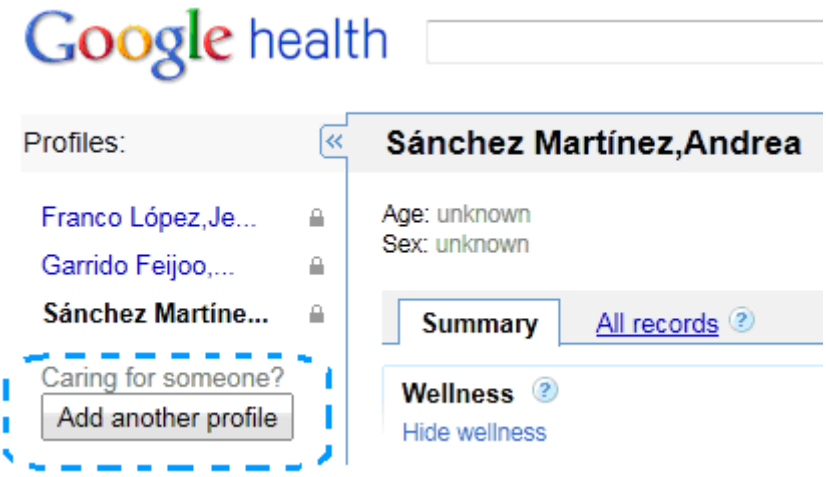


Figura 5-55 Botón para creación de nuevos perfiles en Google Health

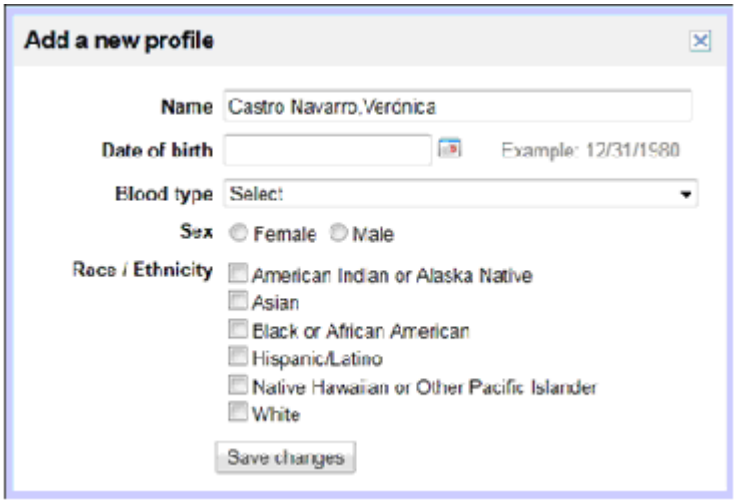


Figura 5-56 Dialogo de creación de nuevo perfil en Google Health

El último paso es rellenar un formulario para asignarle un usuario y contraseña al paciente para que pueda acceder a la aplicación. El médico tendrá que rellenar los siguientes campos: nombre, apellidos, nombre de usuario, correo electrónico y contraseña. El nombre y los apellidos deberán coincidir con los introducidos anteriormente en Google Health; el nombre de usuario que será el identificador con el que el paciente accederá a la aplicación; el campo Contraseña que es la contraseña con la que el paciente accederá a la aplicación y por último de manera complementaria el correo electrónico. Al finalizar este proceso de alta, el médico ya puede informar al paciente de sus datos de autenticación (nombre de usuario y contraseña) para acceder a la aplicación.

La otra opción que puede realizar el médico es consultar datos de un paciente. Para ello el médico seleccionará un paciente y podrá realizar las siguientes consultas o visualizar el estado de:

Datos Personales

Medidas diarias del paciente donde se puede consultar: Insulina y glucemias, cuerpos cetónicos y la dieta

Datos clínicos en la que se encuentran las pruebas complementarias, el tratamiento y comorbilidades.

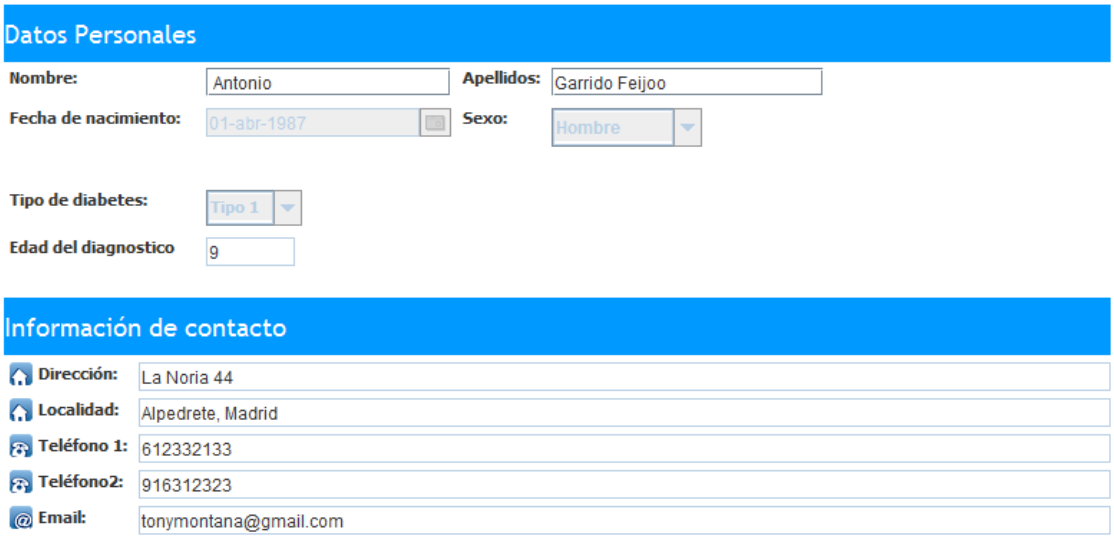


Figura 5-57 Pantalla con los datos personales del paciente

La Figura 5-57 muestra la ficha del paciente, en la cual podremos rellenar los siguientes campos: nombre, apellidos, fecha de nacimiento, sexo, tipo de diabetes, año del diagnóstico, dirección, localidad, teléfonos y email.

En el bloque de medidas diarias de un paciente (Figura 5-58), el médico solo podrá visualizar los datos. En ningún momento podrá añadir ninguna glucemia, bolo de insulina, cuerpo cetónico o alimentación. La parte más importante para poder llevar un buen control es el de glucemias e insulina. Aquí el médico podrá ver todos los valores que el paciente ha ido introduciendo a lo largo de las semanas y los comentarios de éste si los hay. Además para un análisis más claro, podrá ver en modo gráfica las glucemias introducidas y los bolos de insulina. De esta manera podrá ver la evolución que ha tenido el paciente a lo largo de los días.

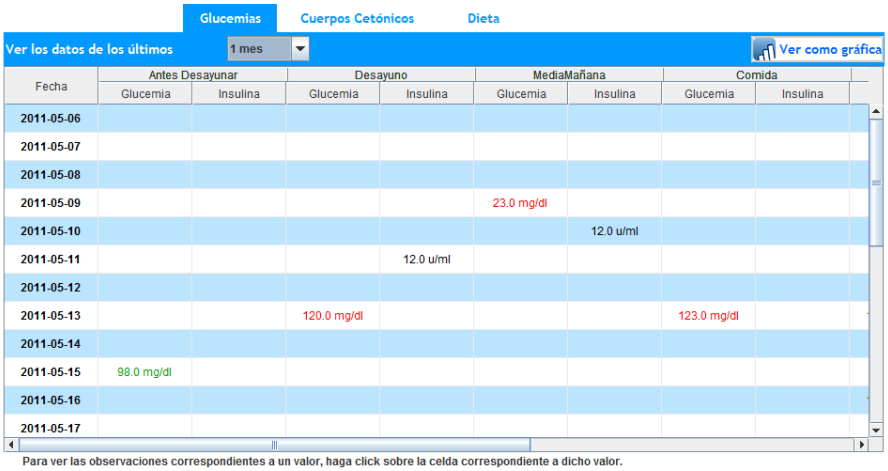


Figura 5-58 Aspecto del bloque de medidas diarias del paciente para el médico

Desde el bloque de medidas diarias, el médico también podrá llevar un registro de los valores de los cuerpos cetónicos medidos por paciente. El médico podrá analizar, por fecha y hora, qué valores ha tenido el paciente. Por último un campo importante en el control del paciente diabético es la alimentación. Por medio del panel de la Figura 5-59, podrá ver las cantidades de hidratos de carbono y el tipo de comida que el paciente ha ingerido a lo largo de los días.

Fecha	Antes Desayunar	Desayuno	MediaMañana	Comida	Merienda	Cena	Antes de dormir
2011-05-27	200.0 g	150.0 g					
2011-05-28							
2011-05-29						150.0 g	
2011-05-30							
2011-05-31							
2011-06-01							
2011-06-02							

Figura 5-59 Panel con el registro de dieta del pacietnte

En el bloque de datos clínicos nos encontramos con las comorbilidades que son las enfermedades más habituales de un paciente diabético. El médico podrá marcar las que padezca el paciente y/o añadir otras enfermedades que vayan surgiendo y que no estén en la lista. Para ello ofrecemos un listado de gran cantidad de enfermedades registradas (obtenidas partir de la Clasificación Internacional de Enfermedades, 10ª edición). El médico podrá seleccionar aquella dolencia que el paciente padezca y rellenar un formulario indicando la fecha de diagnóstico, la fecha de fin y si es una enfermedad activa. De esta manera tendrá en Google Health el historial médico al completo del paciente.

También podrá guardar el médico las pruebas más habituales de un paciente diabético que son el fondo de ojo, el análisis de sangre y orina. Como datos complementarios introducirá el peso, talla, pudiendo ver la evolución de forma gráfica tanto del peso como de la talla. Para los análisis de sangre y de orina, nos aparecerá un formulario en el que rellenaremos los siguientes campos: para análisis de sangre serán: Hba1c, Glucemia, Colesterol (HDL - LDL -Triglicéridos), TSH, T4, Hemoglobina glicosilada, VCM. Para análisis de orina serán los campos: microalbuminuria, creatinina. Por último podrá añadir los resultados del fondo de ojo, simplemente introduciremos la fecha y los resultados en forma de texto del fondo de ojo.

Hemos dicho que la parte de datos clínicos estaba formada por tres bloques. El último es el tratamiento que sigue el paciente, este será tanto la insulina, como los medicamentos que está tomando. Para añadir un medicamento nos aparecerán una lista de todos los medicamentos y solo deberá seleccionar uno y rellenar un formulario que es el que podemos observar en la Figura 5-60.

Figura 5-60 Detalles de un medicamento

5.5.2 Caso de uso del paciente

El paciente para entrar en la aplicación, tendrá que introducir su usuario y contraseña que previamente le habrá sido notificado por su médico. Si los datos son correctos estará dentro de la aplicación.

Cuando entre en su perfil, el paciente podrá realizar las siguientes acciones:

Consultar o editar las insulinas y glucemias

Consultar o editar los cuerpos cetónicos

Consultar o editar la dieta.

La parte más significativa para poder llevar un buen control de las glucemias, tanto para el paciente como para el médico es la introducción de insulinas y bolos de insulina. Tanto para introducir una medida de glucemia o un bolo de insulina, resulta muy sencillo: seleccionará una fecha y hora y la cantidad obtenida o suministrada. Además en el caso de introducir las glucemias podrá añadir un comentario como por ejemplo: ejercicio físico elevado o examen en la universidad, etc. Estos comentarios serán factores externos que el paciente ha sufrido a lo largo del día y pueden haber provocado una glucemia irregular. Una opción que se le permite al paciente, es que podrá visualizar a modo de gráfica sus evoluciones. Esto le puede ayudar mucho para poder mejorar sus glucemias.

Una información complementaría es introducir los valores de cuerpos cetónicos. Introducirá el valor en mmol/l indicando la fecha y hora de cuando ha obtenido ese valor. También de manera opcional tendrá la posibilidad de introducir cualquier comentario relativo a este dato

Por último, un campo importante en el control del paciente diabético es la alimentación. Podrá introducir las cantidades de hidratos de carbono y el tipo de comida que el paciente realiza a lo largo de los días. Para ello deberá

rellenar los campos de fecha, hora tipo de alimento, cantidad de hidratos de carbono y un comentario si fuese necesario.

De esta manera el paciente de una manera fácil y rápida, tendrá en su perfil todos los datos guardados. De esta manera, el médico podrá indicarle cambios en sus tratamientos para mejorar su diabetes.

6 Conclusiones

El haber trabajado en este proyecto ha sido una experiencia gratificante en primer lugar el trabajar en un campo como es la salud y concretamente la telemedicina nos ha enriquecido mucho y nos ha proporcionado una visión global de este mundo y las aplicaciones médicas de la informática. Esto supone una experiencia en el desarrollo de aplicaciones para en el campo de la medicina a distancia. Además el hecho de haber trabajado con un médico y de contar con una persona diabética entre los integrantes del proyecto ha proporcionado un mayor realismo al resultado.

La diabetes es una enfermedad que exige, por parte del paciente, una gran dosis de auto control ya que afecta a su vida diaria, a sus hábitos y sus costumbres. Un paciente diabético debe ser riguroso en el control de comidas, en la administración de insulina inyectada, en el ejercicio realizado,… Por ello, el trabajo realizado se enfoca en facilitar tanto la labor del endocrino como la de los pacientes, a la hora de controlar de manera exhaustiva la diabetes. Para ello el paciente diabético podrá registrar sus glucemias, dosis de insulina, ejercicio físico, dieta, etc. de la manera más simple posible y el endocrino podrá visualizar cómodamente estos datos y guiar al paciente de una manera más personal.

Una de las decisiones más importante a tomar, fue qué proveedor de servicios de la salud utilizar. Finalmente nos decantamos por Google Health por varios motivos: el principal fue que, al igual que la mayoría de los servicios ofrecidos por esta empresa, Google Health es una herramienta muy potente y que, con mucha seguridad, se mantenga como gratuita. Además, Google Health supera con creces a sus competidores en el número de empresas colaboradoras y servicios opcionales que ofrece, lo cual la convierte en la aplicación más grande y potente del sector. Por último un detalle muy importante a nivel técnico, Google Health ofrece un *Interfaz de programación de aplicaciones* (API) para el desarrollo de aplicaciones en Java y la información almacenada sigue el formato estándar *Continuity Care of Record* (CCR).

No obstante, esta elección también trajo consigo nuevas dificultades, motivadas sobretodo por el hecho de que Google Health se encuentre en fase *beta* de desarrollo: funcionalidades aún no terminadas (como la creación de pacientes desde Java, la compartición de información entre varios usuarios, entre otras) y la falta absoluta de mantenimiento de la documentación de la API (poco clara, sin traducir y muchas veces inexistente para algunos servicios ofrecidos). Por esta razón, decidimos desarrollar un API para Java que envolviese toda la comunicación con Google Health, convirtiéndolo en un proceso fácil e intuitivo, así como un manual para interactuar con Google Health a través de Java (manual que ha sido enviado a Google con el objetivo

de que cualquier futuros desarrolladores puedan tener acceso a una documentación simple y en castellano).

También nos gustaría indicar que no todo han sido éxitos. Es el caso del módulo de conexión con los medidores de glucosa. En la idea inicial del proyecto, se estudió la posibilidad de introducir medidas de glucosa en la aplicación directamente desde los glucómetros. Sin embargo, ante la falta de colaboración por parte de los laboratorios quienes no proporcionan ningún tipo de API ni documentación que facilite el proceso de envío/descarga de información con sus glucómetros; y a que no usan ningún protocolo estandarizado de conexión USB finalmente se decidió desechar esta funcionalidad, considerando el grado de esfuerzo necesario para gestionar las conexiones por otros método. En su lugar, se puede introducir medidas desde ficheros XML que proporcionan ciertos glucómetros. Una posible mejora que podría llevarse a cabo en un futuro, es desarrollar el módulo de gestión de glucómetros.

Por otra parte, otra ampliación que podría llevarse a cabo es introducir un módulo de inteligencia artificial que se encargara de realizar una estimación de la cantidad de insulina necesaria en función del paciente. Este módulo estaría supervisado por un médico pero daría unos valores aproximados de la cantidad de insulina que necesita un paciente en un determinado momento.

Para finalizar, nos gustaría mostrar nuestra satisfacción y orgullo por ayudar a mejorar la calidad de vida de los pacientes diabéticos proporcionándoles una aplicación que les facilite la tarea de controlar sus medidas. Además, los médicos agradecerán el poder llevar un control más exhaustivo de las medidas de sus pacientes. A su vez, el haber desarrollado un módulo para gestionar la conexión a Google Health proporciona muchas facilidades a la hora de desarrollar una aplicación que utilice Google Health como proveedor de servicios ya que simplemente ha de centrarse en el desarrollo de su aplicación y no en el modo de conectarla con GH.

Por otra parte a nivel personal, hemos aprendido mucho gracias a los consejos de nuestro tutor José Luís Risco Martín y José Ignacio Hidalgo Pérez, también gracias a los conocimientos de la doctora Esther Maqueda quien nos guió en el desarrollo de la aplicación médica.

7 Glosario

***Diabetes mellitus tipo 1*. E**nfermedad caracterizada por la nula producción de insulina debida a la destrucción autoinmune de las células β de los Islotes de *Langerhans* del páncreas mediadas por las células. La diabetes tipo I se clasifica en dos subtipos: *1a* (diabetes juvenil) y *1b* (diabetes asociada a otras numerosas y variadas alteraciones endocrinas).

Diabetes mellitus tipo 2. Enfermedad caracterizada por el déficit relativo de la producción de insulina, y una deficiente utilización periférica por los tejidos de glucosa. Esto quiere decir que el receptor de la insulina de las células que se encargan de facilitar la entrada de la glucosa a la propia célula está dañado.

Glucemia. Es el nivel de glucosa en sangre de una persona. En condiciones normales este nivel esta entre 75 y 175 mg/dl. En una persona diabética estos niveles pueden varias en función de aspecto externos como la dieta, ejercicio físico, exámenes, etc. Por ello, es muy importante tener unos niveles de glucemias.

Hemoglobina glicosilada. Es un valor importante en las analíticas que muestra el nivel promedio de glucosa en sangre en las últimas 6 a 8 semanas. La hemoglobina glicosilada refleja todas las subidas y bajadas del azúcar en su sangre en las pasadas 8 o más semanas. La diferencia con una medida de glucosa es que esta última solo muestra el estado de su control de la diabetes en un momento determinado, mientras que la hemoglobina glicosilada refleja el estado del paciente diabético en las últimas semanas.

Hipoglucemia. Término médico para acuñar los niveles de glucemias inferiores a 70 mg/dl. Cuando sufre una hipoglucemia, el paciente diabético comenzará a sentirse mal, con diversos síntomas como, por ejemplo, mareo, sudoración en la palma de las manos, mal humor, etc.

***Hiperglucemia*.** Término médico para acuñar los niveles de glucemias superiores a 180 mg/dl produciendo en el paciente diabético sensación de sed, hambre y cansancio.

Bolo de insulina*.* Se define como una inyección de insulina que el paciente diabético se pincha en los diferentes periodos de tiempo estipulados por el médico. Normalmente esta inyección se realiza antes de la ingesta de alimento.

Cuerpo cetónico*.* Sustancia química que produce el organismo cuando no hay suficiente insulina en la sangre y tiene que descomponer las grasas para obtener energía. Los cuerpos cetónicos pueden envenenar y hasta destruir células corporales. Al carecer el organismo de la ayuda de la insulina, los cuerpos cetónicos se van acumulando en la sangre y luego se "derraman" en la orina para poderse eliminar.

API. Siglas de *Application Programming Interface* (interfaz de programación de aplicaciones). Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

GUI: Siglas de *Graphical User Interface*, designa al entorno visual sencillo y cómodo para el usuario que permite una comunicación a través del ordenador.

CCR. Siglas de Continuity of Care Record, se trata de estándar desarrollado por ASTM International para estructurar los documentos con información sanitaria de manera electrónica, creado a partir del lenguaje XML.

JSON. Siglas de *JavaScript Object Notation*. JSON es un formato de intercambio de datos ligero cuya simplicidad ha favorecido su uso generalizado entre los desarrolladores Web.

Secure Sockets Layer. Protocolo criptográfico que proporciona comunicaciones seguras por una red, comúnmente Internet.

HIPAA. Siglas de *Health Insurance Portability and Accountability Act o Ley de* Portabilidad y Responsabilidad del seguro médico, dictada en 1996 por el congreso de EEUU. En ella se establecen las condiciones mínimas de confidencialidad y seguridad que deben cumplir los datos médicos de los pacientes.

EHR. Siglas de *Electronic History Record* o historia clínica electrónica en castellano. Es el resultado de aplicar las nuevas tecnologías de información para obtener un sopote electrónico en el que almacenar toda la información clínica de un paciente.

PHR. Siglas de *Personal History Record,* es decir historia clínica personal. A diferencia del EHR, que es creado y mantenido por profesionales de la salud, el PHR es iniciado y gestionado estrictamente por el paciente

8 Bibliografía

Para la introducción

iTelemedicina (2011), Introducción a la telemedicina,
http://www.itelemedicina.com/index.asp?p=intro/intro.asp

Jaime Prats (2009), "Telemedicina contra las listas de espera", El País,
http://www.elpais.com/articulo/sociedad/Telemedicina/listas/espera/elpepisoc/20091019elpepisoc_1/Tes

Laboratorios Abbot (2011), CoPilot - Sistema de Gestión de Datos,
http://www.abbottdiabetescare.es/productos/prod_10.asp

MyCareTeam Inc. (2011), Quick Tour,
http://www.mycareteam.com/app/quicktour/quicktour6.aspx

Karen E. Smith, Betty A. Levine, Stephen C. Clement, Ming-Jye Hu, Adil Alaoui, Y Seong K. Mun (2004), "Impact of MyCareTeam™ for Poorly Controlled Diabetes Mellitus",
http://www.mycareteam.com/documents/SmithDiabetesTech_TherArticle.pdf

Sobre la diabetes en España

Godoy, A., Rev. Esp Cardiol (2002), "Epidemiología de la diabetes y sus complicaciones no coronarias".

Pantaleón, Ana (Abril 2009), "Medio millón de inmigrantes sufren diabetes en España, El País.

Valdés S, Rojo-Martínez G, Soriguer F. (2007), "Evolución de la prevalencia de la diabetes tipo 2 en población adulta española".

Sobre definición de Diabetes

Biblioteca Nacional de Medicina de EE.UU., Institutos Nacionales de la Salud (2010), "Diabetes Tipo 1",
http://www.nlm.nih.gov/medlineplus/spanish/ency/article/000305.htm

Sobre tratamiento de la diabetes

Ministerio de Sanidad, Política Social e Igualdad, Gobierno de España, "ORDEN SCO/710/2004, DE 12 DE MARZO, POR LA QUE SE AUTORIZA LA FINANCIACIÓN DE DETERMINADOS EFECTOS Y ACCESORIOS CON FONDOS PÚBLICOS",
http://www.mspsi.gob.es/profesionales/CarteraDeServicios/ContenidoCS/3AtencionEspecializada/docs/d10Apartado3Orden710de2004.pdf

Sobre Salud 2.0

Benjamin Hughes, Indra Joshi, Jonathan Wareham, "Health 2.0 and Medicine 2.0: Tensions and Controversies in the Field"
http://www.jmir.org/2008/3/e23/

The Health 2.0 Conference (2011), "Defining Health 2.0",
http://www.health2con.com/about-us/defining-health-2-0/

The Health 2.0 Org (2011), "Health 2.0 Definition",
http://health20.org/wiki/Health_2.0_Definition

Sobre Personal Health Record

The PHR Reviews (2008), “What is a PHR?”,
http://www.phrreviews.com/what-is-a-phr

Children's Hospital Informatics Program, “History of the Personally Controlled Health Record”,
http://indivohealth.org/research.

Sociedad Española de Informática de la Salud (2003), “De la historia clínica a la historia de salud electrónica”,
http://www.conganat.org/seis/informes/2003/PDF/CAPITULO1.pdf

National Institutes of Health National Center for Research Resources (2006), “Electronic Health Records Overview”,
http://www.ncrr.nih.gov/publications/informatics/EHR.pdf

Tracy D Gunter and Nicolas P Terr (2005), “The Emergence of National Electronic Health Record Architectures in the United States and Australia: Models, Costs, and Questions”, Journal of Medical Internet Research,
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1550638/

Lohr, Steve (2009), “Little Benefit Seen, So Far, in Electronic Patient Records”, The New York Times,
http://www.nytimes.com/2009/11/16/business/16records.html?_r=1

Mason, Moya K. (2011), “What Can We Learn from the Rest of the World? A Look at International Electronic Health Record Best Practices”,
http://www.moyak.com/papers/best-practices-ehr.html

Robert Wood Johnson Foundation (2006), “Health Information Technology in the United States: The Information Base for Progress”,
http://www.rwjf.org/files/publications/other/EHRReport0609.pdf

Carnicero Giménez, Javier (2004), “De la historia clínica a la historia de salud electrónica (resumen)”,
http://www.conganat.org/seis/informes/2003/PDF/CAPITULO1.pdf

Bravo Toledo, R., Gervás Camacho, J., Bonís Sanz J. (2009), “Influencia de la informatización de la Atención Primaria en el trabajo de los profesionales y en la salud de la población”, XXVIII CONGRESO DE MEDICINA DE FAMILIA Y COMUNITARIA.
http://www.equipocesca.org/wp-content/uploads/2009/10/electronica-mesa-semfyc-2008.pdf

National Institutes of Health National Center for Research Resources (2006), “Electronic Health Records Overview”,
http://www.ncrr.nih.gov/publications/informatics/EHR.pdf

Gérvas, Juan (2000), “Expectación excesiva acerca de la pronta implantación de la historia clínica electrónica”,
http://www.equipocesca.org/wp-content/uploads/2009/04/expectacion-excesiva-acerca-de-la-pronta-implantacion-de-la-historia-clinica-electronica.pdf.

Szolovits, Peter (2001), “Guardian Angel Personal Lifelong Active Medical Assistant“, The Guardian Angel Consortium,
http://groups.csail.mit.edu/medg/projects/ga/

Sobre Google Health

Sama, Sameer (2009), “Google Health: helping you better coordinate your care”, The Official Google Blog,
http://googleblog.blogspot.com/2009/03/google-health-helping-you-better.html

Arora, Maneesh (2009), “CVS joins Google Health Rx network: millions can access medication records online”, The Official Google Blog.
http://googleblog.blogspot.com/2009/04/cvs-joins-google-health-rx-network.html

Spector, Alfred (2010), "Update from the Google Health Team", The Official Google Blog.
http://googleblog.blogspot.com/2010/03/update-from-google-health-team.html

Brow, Aaron (2010), "A Google Health update", The Official Google Blog,
http://googleblog.blogspot.com/2010/09/google-health-update.html

Lenssen, Philipp (2007), “First Google Health Screenshots“,
http://blogoscoped.com/archive/2007-08-14-n43.html

Estándar CCR

Enright, Cicely (2011), “La tecnología conecta las historias clínicas”,
http://www.astm.org/SNEWS/SPANISH/SPMA08/enright_spma08.html

Tessier, Claudia (2011), “El Comité Informático del Cuidado de la Salud publica la norma de la continuidad del registro del cuidado”,
http://www.astm.org/SNEWS/SPANISH/Q106/q106ccr.html

Arquitectura del API de Google Health

Google Inc. (2011), “API de datos de Google“,
http://code.google.com/intl/es-ES/apis/gdata/

Google Inc. (2011), “API de cuentas de Google”,
http://code.google.com/intl/es-ES/apis/accounts/

Google Inc. (2011), “API de datos de Google Health”,
http://code.google.com/intl/es-ES/apis/health/

OAuth (2011), “Getting Started“,
http://oauth.net/documentation/getting-started/

OpenId Foundation (2011), “What is OpenId?”,
http://openid.net/get-an-openid/what-is-openid/

Seguridad de los PHR

Andérez González, Alberto, "Aspectos legales de la Historia Clínica informatizada",
http://www.conganat.org/seis/informes/2003/PDF/CAPITULO8.pdf

Garbayo Sánchez, José Antonio, Sanz Ureta, Jokin, Carnicero Giménez, Javier, Sánchez García, Carlos (2003), "La seguridad, confidencialidad y disponibilidad de la información clínica",
http://www.conganat.org/seis/informes/2003/PDF/CAPITULO9.pdf

Privacidad de Google Health

Google Inc. (2011), “Google Health Privacy”,
http://www.google.com/intl/en_us/health/about/privacy.html

Google Inc. (2011), “Política de Privacidad”,
http://www.google.es/intl/es/privacy/privacy-policy.html

Google Inc. (2011), “Condiciones del Servicio Google”,
https://www.google.com/accounts/TOS?loc=ES&hl=es

Rizzi, Andrea, Prados, Luís (2009), “La cúpula del Partido Comunista Chino dirigió el ataque a Google”, El País
http://www.elpais.com/articulo/internacional/cupula/Partido/Comunista/Chino/dirigio/ataque/Google/elpepuint/20101204elpepuint_8/T

APÉNDICE I Manual API de Google Health para Java

I.1 Resumen breve de la estructura de Google Health

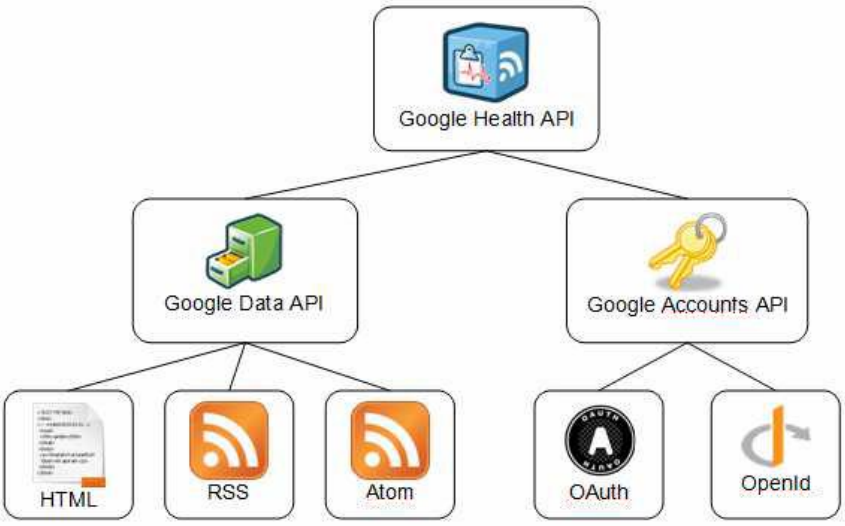


Figura I - 1 Pilares básicos de la estructura de Google Health

Como podemos observar en la Figura I-1, Google Health, al igual que el resto de las aplicaciones ofrecidas por Google se construye en torno a los servicios ofrecidos por la interfaz de programación de aplicaciones (API en inglés) de Google Data (envío/recepción de datos, etc.).

Google Data API

El API de datos de Google proporciona los servicios básicos para leer e introducir datos en la Web siguiendo el Protocolo de Datos de Google. Por tratarse, en su mayoría, de servicios Web, el API de Google Data depende directamente de los estándares que son parte de la Web: HTTP y XML. En concreto, se basa en *Atom Publishing Protocol* (*AtomPub*), utilizando el formato de sindicación *Atom* estándar para representar datos y HTTP para administrar la comunicación (servicios GET y PUT). Además, amplía los servicios ofrecidos por AtomPub, introduciendo autenticación, consultas e incluso permitiendo elegir el formato de salida (JSON, RSS).

Google Accounts API

El API de Google Accounts permite que las aplicaciones externas tengan un acceso limitado a la cuenta de Google de un usuario para ciertos tipos de actividad. Todas las solicitudes de acceso deben tener la aprobación del titular de la cuenta de Google. Actualmente, los API de cuentas de Google proporcionan autenticación para las siguientes actividades:

Intercambio de datos de usuario entre aplicaciones externas y servicios de Google,

Permiso para que los usuarios puedan acceder a aplicaciones externas mediante su cuenta de Google.

El API de Google Accounts facilita el proceso de autenticación de aplicaciones externas mediante un mecanismo de solicitud y recepción de la autenticación. Además, es totalmente compatible con el API de datos de Google. Para el acceso a los datos y la autenticación, Google Accounts ofrece los siguientes protocolos:

OAuth: se trata de un protocolo abierto desarrollado con el objetivo de ofrecer una vía de autenticación segura para aplicaciones de escritorio, móviles y Web. El API de Google Accounts ofrece dos versiones de este protocolo:

OAuth 1.0, versión estándar del protocolo y compatible con todos los APIs de Google.

OAuth 2.0 (estado experimental), versión simplificada del protocolo de autenticación compatible con todos los APIs de Google. OAuth 2.0 se basa en el protocolo criptográfico Secure Sockets Layer (SSL).

OpenId: estándar de identificación digital descentralizado, con el que un usuario puede identificarse en una página Web a través de una URL y puede ser verificado por cualquier servidor que soporte el protocolo. La versión de *OpenId* usada en Google Accounts está basada en el protocolo *OpenId* 2.0, permite a los usuarios el acceso a tu sitio o aplicación Web mediante su cuenta de Google. Cuando Google autentica la cuenta de un usuario, devuelve una ID de usuario a tu aplicación, lo que te permite recopilar y almacenar la información de usuario.

Protocolo Híbrido: ofrece ambos sistemas de autentificación y autorización para aplicaciones Web, permitiendo a los usuarios conectarse y acceder a sus datos. Este protocolo usa *OpenId* para los servicios de autenticación y *OAuth* para los de autorización.

AuthSub: el protocolo *AuthSub* de Google ofrece una alternativa a *OAuth* con niveles de seguridad distintos. Auque es soportado por la mayoría de los APIs de Google, Google recomienda usar *OAuth*.

ClientLogin: proporciona a las aplicaciones de dispositivos móviles o de escritorio la capacidad de incorporar un inicio de sesión automático en su interfaz. ClientLogin es la alternativa preferible para enviar las credenciales de inicio de sesión de un usuario con cada solicitud, lo que permite un rendimiento mayor y más seguridad.

Formato CCR

Google Health utiliza un subconjunto del estándar *Continuity Care of Record* (CCR) para representar los datos. El estándar CCR define la estructura de la información clínica de un paciente usando para ello documentos XML. Un documento CCR estará dividido en tres secciones principales:

Header (Cabecera): describe el formato del documento y aporta información como la fecha de creación, etc.

Body (Cuerpo): en esta sección se encuentran la información clínica del paciente.

Footer (Píe): contiene información demográfica básica del paciente (nombre, edad y sexo).

Cada una de estas secciones se encuentra dividida a su vez en subsecciones, dando lugar a la siguiente estructura (Figura I - 2):

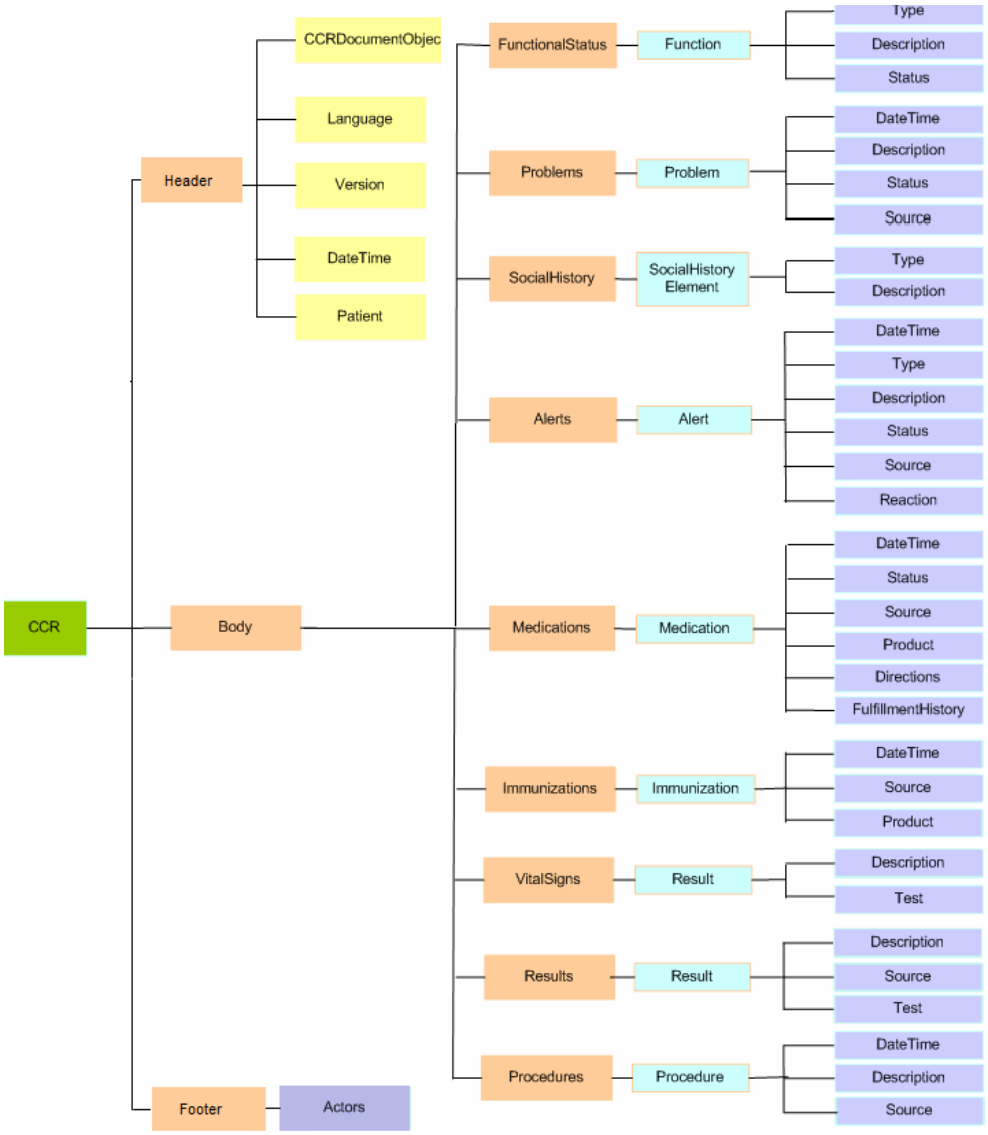


Figura I - 2 Estructura del documento CCR para Google Health

Se observa que existe un subconjunto de elementos que no están formados por otros subelementos (hojas del árbol). Estos elementos constituyen los tipos de datos básicos y son:

<Type>

Especifica un tipo de dato. Por ejemplo, dentro de un elemento de tipo DateTime especificará qué tipo de fecha se representa (Start date, Stop date, Prescription date, etc.).

Sintaxis

```
<Type>
        <Text>...</Text>
</Type>
```

<Description>

Describe el elemento. Contiene un campo para la descripción textual y otro para el código (para clasificar el elemento según el sistema de código usado).

Sintaxis

```
<Description>
        <Text>...</Text>
        <Code>...</Code>
</Description>
```

<Code>

Especifica un código de un elemento y el sistema de códigos usado (existen diferentes sistemas en la comunidad médica [1]). Ej. La diabetes es el elemento 250.0 según el sistema ICD9.

Sintaxis

```
<Code>
        <Value>...</Value>
        <CodingSystem>...</CodingSystem>
</Code>
```

<Status>

Especifica el estado de un elemento. Ej. Una enfermedad puede estar Activa o Inactiva.

Sintaxis

```
<Status><Text>...</Text></Status>
```

<DateTime>

Se usa para especificar cuándo ocurrió un evento y que tipo de evento (elemento de tipo **Type**). La fecha se introduce siguiendo el formato ISO-8601 [2]. Ej. Un medicamento tiene una fecha de inicio de toma (Start date) y otra de fin de toma (Stop date).

Sintaxis

```
<DateTime>
        <Type><Text>...</Text></Type>
        <ExactDateTime>...</ExactDateTime>
</DateTime>
```

<Source>

Especifica la fuente de cierta información. Está formada por uno o más elementos de tipo **Actor** que a su vez está formado por dos elementos de tipo Text: ActorId (Identificador del actor ej. Nombre) y ActorRole (role del actor, ej. Médico, Paciente, etc.)**.** Ej. Un medicamento preescrito por un médico, tendrá en su fuente a dicho médico (con nombre del médico y rol *Médico*).

Sintaxis

```
<Source>
        <Actor>
           <ActorID>...</ActorID>
           <ActorRole>...</ActorRole>
        </Actor>
</Source>
```

<Reaction>

Descripción textual de una reacción ante un evento. Ej. Si el paciente es alérgico a un medicamento podremos describir la reacción que tendría al tomarlo.

Esta formado a su vez por un elemento de tipo <Severity> que determina la magnitud de la reacción (con texto).

Sintaxis

```
<Reaction>
        <Severity><Text>Severe</Text></Severity>
</Reaction>
```

<Direction>

Usado para describir la posología de un medicamento: Dosis, Ruta y Frecuencia. Las dosis son a su vez elementos de tipo **<Dose>** formados por un valor (**<Value>**) y una o más unidades (**<Unit>**). La ruta (**<Route>**) y la frecuencia (**<Frequency>**) son Text, pero existen tablas con valores más comunes ([3] y [4], respectivamente.).

<FullfillmentHistory>

Se usa para llevar un registro de las prescripciones de medicamentos (veces que cierto medicamento ha sido recetado por el médico) indicando la cantidad (en algunos países los medicamentos no se venden en cajas si no en capsulas sueltas).

Está formado por elementos de tipo **<Fullfillment>** formados a su vez por un elemento **<Quantity>** con un valor y unas unidades y un **<DateTime>**.

Sintaxis

```
<FulfillmentHistory>
        <Fulfillment>
           <Quantity>...</Quantity>
           <DateTime>...</DateTime>
        </Fulfillment>
</FulfillmentHistory>
```

El resto de los elementos del árbol son elementos compuestos por estos elementos “básicos”. Una breve descripción de cada uno de ellos:

<FunctionalStatus>

Usado para registrar estados transitorios del paciente (embarazo, amamantando, etc.). Formado por elementos de tipo Function, que a su vez están compuesto por un Type, un Description y un Status.

<Problems>

Esta sección se usará para registrar enfermedades y dolencias del paciente. Estará formado por elementos de tipo **<Problem>** descritos mediante una fecha de inicio de la enfermedad <**DateTime**>, una descripción <**Description**>, un estado <**Status**> y actores relacionados <**Source**>.

<SocialHistory>

Esta sección sólo se usará para indicar la raza del paciente. Esta compuesta por elementos de tipo **<SocialHistoryElement>** descritos mediante un <**Type**> y un <**Description**>.

<Alerts>

Usada para indicar cualquier alergia o evento a tener en cuenta. Cada elemento de tipo <Alert> tendrá una fecha (<**DateTime**>), un tipo (<**Type**>), un estado (<**Status**>), una descripción (<**Description**>), una reacción (<**Reaction**>) y un conjunto de actores relacionados (<**Source**>).

<Product>

Describe un medicamento: Nombre del producto <**ProductName**>, concentración <**Strength**> (sólo en medicamentos) y forma <**Form**> (sólo en medicamentos). Ej. Ibuprofeno 600mg en pastillas.

<**ProductName**> estará formado a su vez por un texto y un código del medicamento, <**Strength**> por un <**Value**> y unas <**Units**> y <**Form**> por un texto [5].

<Medications>

Listado de los medicamentos que el paciente esté tomando o haya tomado. Cada elemento de tipo <**Medication**> vendrá descrito por un estado (<**Status**>), una fecha de prescripción (<**DateTime**>) , una descripción del producto (<**Product**>) , unas indicaciones de toma (<**Directions**>), un registro de las prescripciones (<**FulfillmentHistory**>) y un conjunto de actores relacionados (<**Source**>).

<Immunizations>

Vacunas y tratamientos recibidos por el paciente. Cada elemento **<Inmunization>** vendrá descrito por una fecha (<**DateTime**>), una descripción del producto (<**Product**>) y un conjunto de actores relacionados (<**Source**>).

<Test>

Describe un resultado de una prueba médica y vendrá descrito por una fecha de realización (<**DateTime**>), una descripción de la prueba (<**Description**>), un resultado del test (<**TestResult**>), un valor normal de dicho test (<**NormalResult**>) y un conjunto de actores relacionados (<**Source**>).

Tanto <**TestResult**> como <**NormalResult**> estarán formados por un valor <**Value**> y unas unidades <**Units**>.

<VitalSigns>

Se usa para registrar medidas tales como la altura, el peso etc. Estará formado por uno o varios elementos de tipo <**Result**> formados a su vez por elementos de tipo <**Test**>. El objetivo es agrupar dentro de cada <**Result**> resultados de test distintos para una misma prueba (ej. todas las medidas de peso estarían juntas dentro de un <**Result**>, las de altura en otro <**Result**>,...).

<Results>

Igual que <**VitalSigns**> pero para registrar otro tipo de pruebas (analíticas, glucemias, etc.).

<Procedures>

Registrará mediante elementos de tipo <**Procedure**> aquellas operaciones que haya sufrido el paciente. Cada elemento <**Procedure**> estará formado por una fecha <**DateTime**>, una descripción <**Description**> y un conjunto de actores relacionados (<**Source**>).

Google Health ofrece un ejemplo de un documento CCR completo en la página oficial del API de Google Health [6].

I.2 Ejemplos de uso del API Google Health para Java 1.5

Autentificación usando HealthService y ClientLogin

A continuación explicaremos cómo realizar la autenticación de nuestra aplicación de escritorio a Google Health usando para ello HealthService y el protocolo ClientLogin.

El primer paso es crearnos un objeto de clase HealthService. Los objetos de esta clase, extienden la clase GoogleService de Gdata y contienen la funcionalidad necesaria para acceder a los datos de Google (Google Health en este caso).

Creamos el objeto HealthService

```
HealthService healthService = new HealthService("Ejemplo");
```

Una vez creado, es necesario configurar las credenciales para poder autentificarse. Para ello es necesario una cuenta Google y su correspondiente contraseña:

Configuramos las credenciales

```
healthService.setUserCredentials("ejemplo@gmail.com", "password");
```

Una vez autentificados, ya podemos trabajar con la información almacenada en Google Health.

Listar los perfiles registrados en Google Health y sus identificadores internos

Un usuario puede tener registrados datos médicos de varios perfiles y, por tanto, los datos almacenados en Google Health se archivan según el sujeto al que pertenecen, encapsulados en una entrada *Atom*. Cada entrada *Atom* estará identificada mediante un identificador alfanumérico único y un nombre del perfil representado. Para poder consultar los datos de un perfil necesitamos conocer el identificador interno que Google Health le asignó en el momento de su creación.

Por lo tanto, necesitamos obtener primero el identificador del perfil a consultar. Para listar los perfiles registrados y sus respectivos identificadores usaremos:

```
String PROFILE_LIST_URL = "http://www.google.com/health/feeds/profile/list/";
Feed profileListFeed = healthService.getFeed(new URL PROFILE_LIST_URL,
Feed.class);
List<Entry> entries = profileListFeed.getEntries();
for (Entry profileListEntry : entries) {
  System.out.println("Profile name: " +
profileListEntry.getTitle().getPlainText());
  System.out.println("Profile id: " + profileListEntry.getPlainTextContent());
}
```

Obtener información clínica para cierto perfil

Una vez encontrado el identificador del perfil que queremos consultar, podemos obtener su información clínica en formato CCR. Para ello:

```
String profileID = “...”;
String PROFILE_FEED_PATH = "https://www.google.com/health/feeds/profile/ui/";
String url = PROFILE_FEED_PATH + profileID;
ProfileFeed profileFeed = healthService.getFeed(new URL(url),
ProfileFeed.class);
for (ProfileEntry entry : profileFeed.getEntries()) {
  System.out.println(entry.getContinuityOfCareRecord().getXmlBlob().getBlob());
}
```

La función getFeed(…) de HealthService nos devuelve un objeto de tipo Feed, que contiene a su vez objetos de tipo Entry. Estos objetos de clase Entry son entradas Atom que contienen el código XML con la información CCR del perfil. Sin embargo, para un mismo perfil se nos puede devolver numerosos objetos Entry, lo cual puede ser engorroso y poco útil. Podemos solucionarlo cambiando el modo en que recuperamos los datos usando consultas (*queries*). Una *query* puede ser configurada antes de ser lanzada con distintos parámetros (existe una tabla de parámetros [7]). En este caso usaremos el parámetro *digest* que unifica todas las Entries en una sola:

```
String profileID = “...”;
String PROFILE_FEED_PATH = "https://www.google.com/health/feeds/profile/ui/";
String url = PROFILE_FEED_PATH + profileID;
Query query = new Query(url);
query.addCustomParameter(new Query.CustomParameter("digest", "true"));
ProfileFeed profileFeed = healthService.getFeed(query, ProfileFeed.class);
System.out.println(profileFeed.getEntries().get(0).getContinuityOfCareRecord().
getXmlBlob().getBlob());
```

Obtener información específica de cierto perfil

Obtener sólo una parte de los datos médicos del perfil (por ejemplo los medicamentos), es sencillo usando queries. Para ello debemos configurar la query especificando la categoría de los datos que queremos consultar. Existen 8 categorías posibles ():

Labtest.
Usada para obtener los datos de la sección <Results> registrados al perfil.

Medication
Usada para obtener los datos de la sección <Medications> registrados al perfil.

Condition
Usada para obtener los datos de la sección <Problems> registrados al perfil.

Payer
Usada para obtener los <Payer> registrados al perfil (información del seguro, etc.).

Procedure
Usada para obtener los datos de la sección <Procedures> registrados al perfil.

Immunization
Usada para obtener los datos de la sección <Procedures> registrados al perfil.

Allergy
Usada para obtener los datos de la sección <Alerts> registrados al perfil.

Demographics
Usada para obtener los datos de la sección <Actor>, <VitalSigns>, <FunctionalStatus> y <SocialHistory> registrados al perfil.

Si por ejemplo quisiéramos obtener todos los medicamentos que tiene registrado un perfil escribiremos:

```
String profileID = “...”;
String PROFILE_FEED_PATH = "https://www.google.com/health/feeds/profile/ui/";
String url = PROFILE_FEED_PATH + profileID;
Query query = new Query(feedUri);
query.addCustomParameter(new Query.CustomParameter("digest", "true"));
query.addCategoryFilter(new Query.CategoryFilter(new Category(“Medication”)));
ProfileFeed profileFeed = healthService.getFeed(query, ProfileFeed.class);
System.out.println(profleFeed.getEntries().get(0).getContinuityOfCareRecord().g
etXmlBlob().getBlob());
```

Sobre obtención datos del perfil:

Cada elemento introducido en un perfil (<Result>, <Medication>, etc.) tiene asignado un identificador único que *Google Health* asigna automáticamente al ser introducido. Este identificador aparece en el código XML devuelto tras una consulta, en concreto al principio del código de cada elemento, identificado por la etiqueta XML <CCRDataObjectID>. Los identificadores se dividen en dos partes: un código alfanumérico y un número, separados mediante un guión. El código alfanumérico es el identificador del objeto mientras que el número tendrá distinto significado según el elemento:

Si es un elemento que puede estar compuesto por otros subelementos, indicará el número de subelementos que contiene. Por ejemplo, un elemento de tipo **<Result>** puede estar formado por uno o más **TestResult** (en teoría). Por lo tanto si el **Result** está formado por 4 **TestResult**, su identificador será algo como: *0vmUUlTTrFw-4.*

En elementos que son subelementos de otros (por ejemplo los **TestResult**), el número indicará el índice dentro del elemento padre (empezando por 0). Así, sobre el ejemplo anterior, el elemento de tipo **<Result>** con id. 0vmUUlTTrFw-4, contendrá los elementos 0vmUUlTTrFw-0, 0vmUUlTTrFw-1, 0vmUUlTTrFw-2 y 0vmUUlTTrFw-3.

No es obligatorio que aparezca la segunda parte del identificador, limitándose a un código alfanumérico (ej. _zfEpnPn4CQ).

Insertar nuevos datos en un perfil

Si queremos introducir nuevos datos para un perfil de Google Health, debemos empezar generando el código XML con los nuevos datos siguiendo el formato CCR. Por ejemplo si queremos introducir un nuevo medicamento con las siguientes propiedades:

Nombre	Ibuprofeno, 600 mg, tabletas
Posología	Vía Oral, 600 mg, 2 veces al día
Fecha de prescripción	04/05/2011 18:00

El código XML generado sería:

```
<urn:Body xmlns:urn="urn:astm-org:CCR">
        <urn:Medications>
            <urn:Medication>
                <urn:Product>
                    <urn:ProductName>
                        <urn:Text>Ibuprofeno</urn:Text>
                    </urn:ProductName>
                    <urn:Strength>
                        <urn:Value>600</urn:Value>
                        <urn:Units><urn:Unit>mg</urn:Unit></urn:Units>
                    </urn:Strength>
                    <urn:Form><urn:Text>Tableta</urn:Text></urn:Form>
                </urn:Product>
                <urn:Directions>
                    <urn:Direction>
                        <urn:Route><urn:Text>Oral</urn:Text></urn:Route>
                        <urn:Dose><urn:Value>600</urn:Value></urn:Dose>
                        <urn:Frequency><urn:Value>2 times per
day</urn:Value></urn:Frequency>
                    </urn:Direction>
                </urn:Directions>
                <urn:DateTime>
                    <urn:Type><urn:Text>Prescription date</urn:Text></urn:Type>
                    <urn:ExactDateTime>2011-05-04T18:00:00Z</urn:ExactDateTime>
                </urn:DateTime>
                <urn:Status><urn:Text>ACTIVE</urn:Text></urn:Status>
            </urn:Medication>
        </urn:Medications>
</urn:Body>
```

Una vez generado el código XML, podemos proceder a introducirlo en el sistema. Para ello, recuperaremos el ProfileFeed para el perfil y a través de su función createEntry(), generaremos la estructura para una la Entry. A continuación inicializaremos el campo ContinuityCareOfRecord de la Entry y configuraremos su código XML como el código generado anteriormente. Para finalizar, insertaremos la nueva Entry por medio de la función insert(…) del HealthService que nos devolverá la nueva Entry creada con su respectivo nuevo identificador asociado:

```
String datos = “<urn:Body xmlns:urn="urn:...”;
String profileID = "...";
String PROFILE_FEED_PATH = "https://www.google.com/health/feeds/profile/ui/";
String url = PROFILE_FEED_PATH + profileID;
//Obtenemos el ProfileFeed
ProfileFeed profileFeed = healthService.getFeed(new URL(url),
ProfileFeed.class);
//Creamos una nueva Entry
ProfileEntry entry2 = profileFeed.createEntry();
//Inicializamos y configuramos los datos de la Entry
entry2.setContinuityOfCareRecord(new ContinuityOfCareRecord());
entry2.getContinuityOfCareRecord().setXmlBlob(new XmlBlob());
entry2.getContinuityOfCareRecord().getXmlBlob().setBlob(datos);
//Introducimos los datos
entry2 = healthService.insert(new URL(url), entry2);
```

Notas sobre Insertar nuevos datos:

A fecha de 2011-05-06, los elementos de tipo **<Result>** no admiten tener más de un **<TestResult>** dentro (por ejemplo una analítica: un único documento con varios resultados de test dentro). La única vía posible es introducir cada test en un elemento de tipo **<Result>** individual.

A fecha de 2011-05-06, no es posible introducir varios elementos a la vez (por ejemplo 2 elementos **<Result>**), se deben introducir de uno en uno.

A fecha de 2011-05-06, es conveniente respetar el formato de las fechas en los elementos de tipo **<Datetime> [8]**

Actualizar datos del perfil

El API de Google Health también ofrece la posibilidad de actualizar los datos de cierto elemento introducido en un perfil. Para ello necesitaremos el código original del elemento, es decir, el código introducido previamente obtenido al recuperar el elemento de Google Health. Sobre este código XML que contiene la información del elemento, podremos realizar algunos cambios en los valores encerrados entre la etiquetas XML.

Ejemplo: Modificación del valor de una toma de glucemia (<Result>), previamente recuperado de Google Health (Figura I - 3).

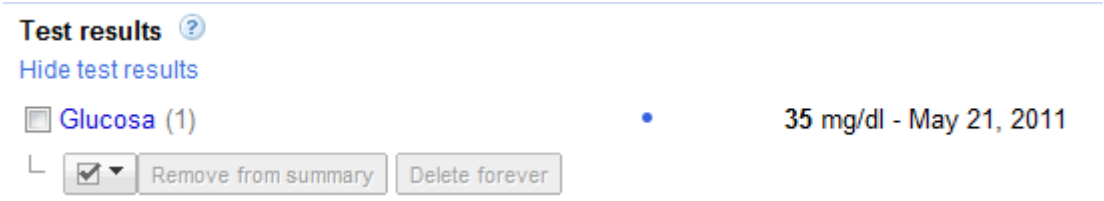


Figura I - 3 Toma de glucosa antes de la actualización

Código recuperado del Google Health:

```
<Body>
        [...]
        <Results>
        [...]
<Result>
        <CCRDataObjectID>bcu2WA9xB4g-1</CCRDataObjectID>
        <DateTime>
           <Type>
              <Text>Collection start date</Text>
           </Type>
           <ExactDateTime>2011-05-21T10:10:00Z</ExactDateTime>
        </DateTime>
        <Description>
           <Text>Glucosa</Text>
        </Description>
        <Source> [...]</Source>
        <Test>
           <CCRDataObjectID>bcu2WA9xB4g-0</CCRDataObjectID>
           <DateTime>
                 <Type><Text>Collection start date</Text></Type>
                 <ExactDateTime>2011-05-21T10:10:00Z</ExactDateTime>
           </DateTime>
           <Description>
              <Text>Glucosa</Text>
           </Description>
           <Source>[...]</Source>
           <TestResult>
              <Value>35</Value>
              <Units>
                 <Unit>mg/dl</Unit>
              </Units>
           </TestResult>
           <NormalResult><Normal><Source>[...]</Source></Normal></NormalResult>
        </Test>
</Result>
</Results>
</Body>
```

Ya que queremos actualizar el valor de la toma (campo <Value> de la sección <Result><Test><TestResult>), el código XML usado en para la actualización sólo contendrá aquellas partes afectadas por el cambio, así como las imprescindibles para diferenciar el elemento del resto de los <Result> introducidos (Fecha, descripciones, etc.). Para actualizar, por tanto, usaremos el siguiente código:

```
<urn:Body xmlns:urn="urn:astm-org:CCR">
        <urn:Results>
            <urn:Result>
                <urn:Description>
                    <urn:Text>Glucosa</urn:Text>
                </urn:Description>
                <urn:Test>
                    <urn:DateTime>
                        <urn:Type><urn:Text>Collection start
date</urn:Text></urn:Type>
                        <urn:ExactDateTime>2011-05-
21T10:10:00Z</urn:ExactDateTime>
                    </urn:DateTime>
                    <urn:Description>
                        <urn:Text>Glucosa</urn:Text>
                    </urn:Description>
                    <urn:TestResult>
                        <urn:Value>90</urn:Value>
                        <urn:Units><urn:Unit>mg/dl</urn:Unit></urn:Units>
                    </urn:TestResult>
                </urn:Test>
            </urn:Result>
        </urn:Results>
</urn:Body>"
```

Una vez generado este código XML, se enviara a Google Health usando la función update del API de Google Health, para lo cual necesitaremos el identificador del perfil y el identificador del objeto a actualizar (ambos identificadores los encontraremos en el código XML recuperado):

```
//Código XML generado
String data = “<urn:Body xmlns:urn="urn:...”;
//Identificador del perfil
String profileID = "...";
//Identificador del objeto a actualizar
String objectID = "...";
String PROFILE_FEED_PATH = "https://www.google.com/health/feeds/profile/ui/";
String url = PROFILE_FEED_PATH + profileID;
//Obtenemos el ProfileFeed
ProfileFeed profileFeed = healthService.getFeed(new URL(url),
ProfileFeed.class);
//Creamos una nueva Entry
ProfileEntry entry2 = profileFeed.createEntry();
//Inicializamos y configuramos los datos de la Entry
entry2.setContinuityOfCareRecord(new ContinuityOfCareRecord());
entry2.getContinuityOfCareRecord().setXmlBlob(new XmlBlob());
entry2.getContinuityOfCareRecord().getXmlBlob().setBlob(datos);
//Introducimos los datos
url = PROFILE_FEED_PATH + profileID + "/" + objectID;
healthService.update(new URL(url), entry);
```

Podremos observar cómo en Google Health el valor de la toma de Glucosa se actualiza pasados unos segundos (Figura I - 461 I - 4)

Figura I - 461 Valor de la toma de Glucosa tras la actualización

Sobre Actualizar los datos del perfil

A fecha de Mayo de 2011, no se pueden modificar el contenido de las secciones <Description>

Eliminar datos del perfil

El API de Google Health también nos ofrece la posibilidad de borrar la mayoría de los elementos introducidos en un perfil. Para ello, únicamente necesitaremos el identificador interno asignado por Google Health al elemento (que en el código XML entre las etiquetas <CCRDataObjectID>) y el identificador del perfil al que pertenece el elemento. Al eliminar un elemento se eliminarán también todos aquellos subelementos que lo compongan. El código para borrar un elemento dado su identificador interno es el siguiente:

```
//Identificador del perfil
String profileID = "...";
//Identificador del objeto a eliminar
String objectID = "...";
String PROFILE_FEED_PATH =
"https://www.google.com/health/feeds/profile/ui/";
String url = PROFILE_FEED_PATH + profileID + "/" + objectID;
//Eliminamos
healthService.delete(new URL(url));
```

I.3 Referencias

[1]Sistemas de códigos médicos, http://code.google.com/intl/es-ES/apis/health/ccrg_reference.html#codingsystem

[2] Formato de fechas según el estándar ISO-8601, http://code.google.com/intl/es-ES/apis/health/ccrg_reference.html#exactdatetime

[3] Tabla de rutas (<Route>) más comunes, http://code.google.com/intl/es-ES/apis/health/ccrg_reference.html#route

[4] Tabla de frecuencias (<Frequency>) más comunes, http://code.google.com/intl/es-ES/apis/health/ccrg_reference.html#frequency

[5] Tabla con las formas más comunes (<Form>), http://code.google.com/intl/es-ES/apis/health/ccrg_reference.html#form

[6] Ejemplo de documento CCR completo, http://code.google.com/p/googlehealthsamples/source/browse/trunk/CCR_samples/CompleteProfile.xml

[7] Parámetros para las queries, http://code.google.com/intl/es-ES/apis/health/docs/2.0/reference.html#Parameters

[8] Formato para las fechas (<DateTime>), http://code.google.com/intl/es-ES/apis/health/ccrg_reference.html#exactdatetime

Página oficial del formato CCR para Google Health, secciones principales

<Actors>
<Actor>
<Alerts>
<Alert>
<Code>
<CodingSystem>
<DateOfBirth>
<DateTime>
<Description>
<Direction>
<Directions>
<DisplayName>
<Dose>
<ExactDateTime>
<Form>
<Frequency>
<Fulfillment>
<FulfillmentHistory>
<Function>
<FunctionalStatus>
<Gender>
<Immunization>
<Immunizations>
<Language>
<Medication>
<Medications>
<Name>
<NormalResult>
<Person>
<Problem>
<Problems>
<Procedure>
<Procedures>
<Product>
<ProductName>
<Products>
<Quantity>
<Reaction>
<Result>
<Results>
<Route>
<Severity>
<SocialHistory>
<SocialHistoryElement>
<Source>
<Status>
<Strength>
<Test>
<TestResult>
<Type>
<Unit>
<Units>
<Value>
<VitalSigns>

APÉNDICE II Bugs de Google Health

Durante el desarrollo de la aplicación *GluControl* encontramos numerosos fallos en el API de Google Health. Estos fallos, compresibles ya que Google Health todavía se encuentra fase Beta, han sido registrados y enviados a Google con el fin de que sean subsanados en futuras actualizaciones.

A continuación, listaremos aquellos fallos (bugs) que hemos encontrado:

Creación de nuevos perfiles vía API. Actualmente la única vía posible para crear nuevos perfiles para un cuenta de Google Health es desde la página Web de Google Health. Como ha ocurrido en nuestro caso, no poder dar de alta a nuevos perfiles desde la propia aplicación, obliga a tener que buscar alternativas que aumentan la complejidad del proceso de alta.

Compartición de perfiles. Google Health sólo permite compartir los perfiles de una cuenta Google con otra en modo *sólo lectura*. Este hecho supone un gran obstáculo para el desarrollo de aplicaciones como la nuestra, en las que un solo perfil sea compartido y editado por varios usuarios (por ejemplo médicos y pacientes), que obliga a recurrir a alternativas que aumentan considerablemente la complejidad.

Agrupar medidas de análisis. En la versión actual del API, no es posible agrupar medidas de un mismo tipo de análisis. Este hecho complica mucho el trabajo con los datos del paciente ya que, por ejemplo, si el paciente tiene muchas glucemias registradas, al obtener esta información de Google Health, tendrá que recorrer todos los elementos de tipo *Result* para recuperar sólo las glucemias.

Envío simultáneo de elementos. Con la versión actual del API no es posible enviar a Google Health más de un elemento a la vez. Si por ejemplo, se desean registrar varias glucemias, es necesario obtener el código XML, crear un documento CCR y enviar a Google cada una de las medidas de manera independiente. Esto no sólo hace que el código sea más complejo, sino que aumenta considerablemente el número de transferencias con Google y por tanto enlentece el proceso.

Omisión de campos del CCR. Aunque por lo general el subconjunto CCR adoptado por Google es suficiente para almacenar la información del paciente, en ciertos momentos se hacen necesarios otros campos como los relativos a la información personal del paciente (campos para domicilio, email, teléfono de contacto, etc.) y, especialmente, un campo destinado al almacenamiento de observaciones.

APÉNDICE III Investigaciones sobre Glucómetros

I.1 Introducción

Como parte de nuestro proyecto se decidió intentar incorporar a la aplicación GluControl de un sistema que permitiera importar las glucemias tomadas mediante glucómetros de última generación (con puertos USB). No obstante, debido a las dificultades encontradas, finalmente se decidió no incluir este módulo en la aplicación final. A continuación explicaremos los resultados obtenidos con vistas a que se vuelva a intentar en el futuro.

I.2 Sobre USB

El bus universal en serie (*Universal Serial Bus* en inglés), abreviado comúnmente USB, es un puerto físico que permite conectar periféricos a un ordenador aprovechando las capacidades *plug-and-play* que permite conectar/desconectar dispositivos sin necesidad de reiniciar el ordenador. A continuación se exponen de una manera muy básica, los principales aspectos del USB.

Componentes del USB

Controlador (Host)

Reside dentro del computador y es responsable de las comunicaciones entre los periféricos USB y el CPU del computador. Es también responsable de la admisión de los periféricos dentro del bus, tanto si se detecta una conexión como una desconexión.

Concentradores (Hubs)

Permiten la conexión de hasta 127 dispositivos. Además del controlador, el computador también contiene el concentrador raíz. Éste es el primer concentrador de toda la cadena que permite a los datos y a la energía pasar a uno o dos conectores USB del PC y, de allí, a los 127 periféricos.

Dispositivos (Functions)

Se conectan a los *Hubs* (de manera externa o interna) y aportan alguna funcionalidad (ratones, teclados, etc.).

Una aproximación gráfica para esta estructura puede verse en la Figura III-1

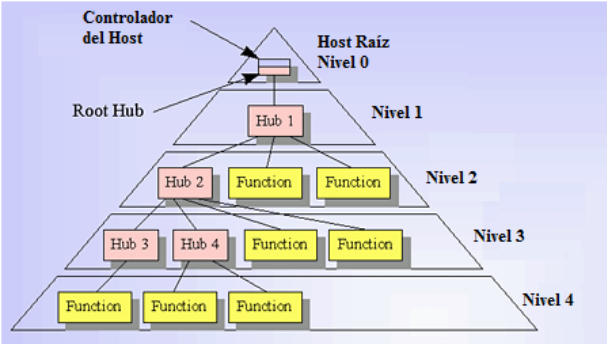


Figura III-1 Estructura piramidal de los componentes del USB

Configuraciones, interfaces, y extremos

Un dispositivo puede tener una o más *configuraciones* que controlan su comportamiento. Las configuraciones pueden diferir en la cantidad de energía que consumen, etc.

Cada configuración contiene una o más *interfaces* que especifican cómo accede el software al hardware.

Cada interfaz tiene uno o más *endpoints* (*extremos)* que utiliza como fuentes o destinos de transferencia de datos. Cada *endpoint* se caracteriza por:

Frecuencia de acceso al bus requerida

Ancho de banda requerido

Número de *endpoint*

Tratamiento de errores requerido

Máximo tamaño de paquete que el *endpoint* puede enviar o recibir

Tipo de transferencia para el *endpoint*

Orientación en la que se transmiten los datos

Tuberías

Una tubería USB es una asociación entre uno o dos *endpoints* en un dispositivo, y el software en el *host*. Las tuberías permiten mover datos entre software en el *host*, a través de un buffer, y un *endpoint* en un dispositivo

Tipos de transferencia

La interpretación de los datos que se transmiten a través de las tuberías depende del dispositivo receptor o del software. No obstante, USB proporciona cuatro tipos de transferencia de datos sobre las tuberías para optimizar la utilización del bus en función del tipo de servicio:

Transferencias de control, se utilizan para controlar el dispositivo físico.

Transferencias isócronas, se utilizan para transferir en tiempo real, audio, vídeo y otros tipos de datos no estructurados a una velocidad garantizada.

Transferencias de interrupción, son utilizados por los dispositivos de señalamiento, teclados, y cualquier otros dispositivos físicos que necesitan una respuesta rápida garantizada

Transferencias Bulk, se utilizan para transferir archivos y realizar otros tipos de transferencias que involucran a grandes bloques de datos no estructurados. Estas transferencias de garantizar que todos los datos serán entregados.

Descriptores

Los dispositivos USB disponen además de unos datos estructurados conocidos como descriptores. Estas estructuras de datos permiten al dispositivo identificarse al software que se ejecuta en el *host.* Dentro de estos datos, destacan los descriptores *de dispositivo,* descriptores *de configuración,* descriptores *de interfaz*, descriptores *de endpoint y de*scriptores *String que* contienen cadenas Unicode legibles para describir un dispositivo, una configuración, una interfaz o un *endpoint*.

I.3 Sobre Java y USB, bibliotecas existentes

A pesar de que el USB es una parte totalmente integrada en los ordenadores actuales, java no da soporte oficial a este tipo de dispositivos. Lograr que una aplicación Java interactúe con dispositivos USB, requiere el uso de API desarrolladas por terceros. Actualmente existen dos bibliotecas mayoritariamente extendidas: JSR-80 y JUSB.

JSR-80

El objetivo de este proyecto es desarrollar una interfaz USB para la plataforma Java que permita el acceso pleno al sistema USB desde cualquier aplicación Java. Fue creado en 1999 para IBM y en 2001 fue aceptado como una extensión candidata para el lenguaje Java. Existen dos implementaciones,: una para Linux, que ha sido certificada con los test de Sun; y otra para Windows, en todavía en desarrollo.

JUSB

El objetivo de este proyecto era proporcionar un API open-source para Java que permitiera acceder a los dispositivos USB en plataformas UNIX. Este API proporciona acceso múltiple dispositivos USB y da soporte para transferencias de control, transferencias de tipo BULK y transferencias de interrupción. Las transferencias isócronas no son compatibles. Al igual que JSR-80, también dispone de una implementación para Windows que se encuentra aún en una etapa muy temprana.

Si se trabaja en Windows, es imprescindible introducir la biblioteca dinámica jusb.dll, suministrada en la distribución de la biblioteca JUSB, en la carpeta Windows/System32/.

I.4 Pruebas realizadas con los medidores de glucosa

Medidor *Optium XCeed* de Abbot

El medidor de glucosa Optium Xceed de laboratorios Abbott se conecta al PC por medio de un cable adaptador de la Figura III- III-2.

Figura III-2 Glucómetro Optium Xceed y cable USB de conexión

Una vez conectado e instalados los controladores necesarios, se realizaron las una serie de pruebas, en Windows 7 y con la biblioteca JUSB, que se explican a continuación.

Identificación del dispositivo

Antes de nada, es necesario conocer los identificadores de fabricante y producto del dispositivo USB. Para ello, en la ventana de Sistema → Administrador de Dispositivos, buscamos el dispositivo identificado como "Abbot Strip Port Cable" y exploramos sus propiedades. Sus identificadores (Identificador del fabricante = 1A61 e identificador de producto = 3420) se encuentran en la opción de la Figura III-3.

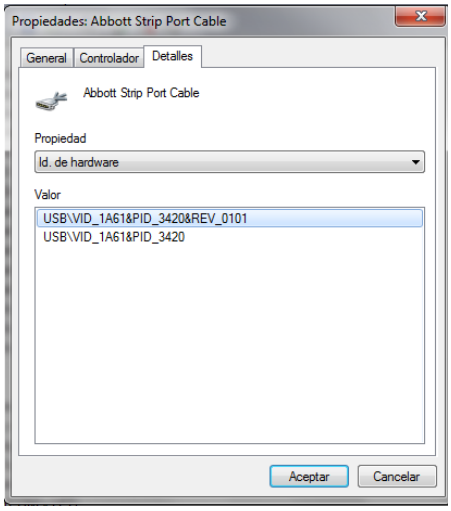


Figura III -3 Propiedades del dispositivo USB

Una vez identificado, el siguiente código nos permite saber si se encuentra conectado y, en caso de estarlo, se nos devuelve la información relativa al dispositivo (número de interfaces, tipos de *endpoints*, etc.).

```
DeviceImpl dev;
//Vamos recorriendo todos los host del bus
for (int k = 0; k < busses.length; k++) {
        //Por cada host, identificamos cada uno de los dispositivos conectados
        System.out.println("\n\nBus[ " + ((USB) busses[k]).getBusNum() + " ]  ");
        for (int i = 0; i < 5; i++) {
           dev = (DeviceImpl) busses[k].getDevice(i);
           if (dev != null) {
              System.out.print(" [ " + i + " ] :  ");
              if (dev.getAddress() > 0) {
                 printDeviceDescriptor(dev);
                 DeviceDescriptor dd = dev.getDeviceDescriptor();
                 //Comprobamos si el dispositivo es un Optium Xceed
                 if(Integer.toHexString(dd.getVendorId()).equalsIgnoreCase("1A61")
                     &&
Integer.toHexString(dd.getProductId()).equalsIgnoreCase("3420")) {
                    System.out.println("Encontrado un dispositivo Optium Xceed");
                    //Imprimimos todas las configuraciones posibles del
dispositivo
                    printConfigurationDescriptor(dev);
                    //Imprimimos todas las interfaces posibles del dispositivo
                    printInterfaceDescriptors(dev);.
                    //Imprimimos todos los endpoints posibles del dispositivo
                    printEndpointDescriptors(dev);
                 }
```

Tras ejecutarlo, se obtiene la siguiente información:

```
ID: 1a61
IDProd: 3420
Configuration Descriptor:
        wTotalLength      : 39
        bNumInterfaces    : 1
        bConfigurationValue: 2
        bmAttributes       : 128
        bMaxPower         : 50  ( 100mA)

Interface Descriptor[0]:
        bInterfaceNumber        : 0
        bAlternateSetting       : 0
        bNumEndpoints           : 3
        bInterfaceClass         : 255
        bInterfaceSubClass      : 0
        bInterfaceProtocol      : 0
        iInterface              : 0
        Interface[0]  Endpoint Descriptor[0]:
           bEndpointAdress      : 1    IN Pipe Data flows from Device to Host
           bmAttributes         : 2    Type: bulk
           bMaxPacketSize       : 64
           bInterval            : 0
           getEndpoint          : 129  81
        Interface[0]  Endpoint Descriptor[1]:
           bEndpointAdress      : 1 OUT Pipe Data flows from Host to Device
           bmAttributes         : 2    Type: bulk
           bMaxPacketSize       : 64
           bInterval            : 0
           getEndpoint          : 1  1
        Interface[0]  Endpoint Descriptor[2]:
```

```
bEndpointAdress      : 3    IN Pipe Data flows from Device to Host
bmAttributes         : 3    Type: interrupt
bMaxPacketSize       : 2
bInterval            : 1
getEndpoint          : 131  83
```

Como se observa en la información anterior, el dispositivo tiene una sola configuración con una sola interfaz. Dentro de esta interfaz, existen tres *endpoints*: 2 de tipo *bulk,* que serán usadas para el envío y la recepción de datos, y una de tipo interrupción. En nuestro caso nos interesan las de tipo Bulk y, en concreto, aquella en la que el sentido de la información va desde el dispositivo hacia el *host* (para la descarga de datos).

Una vez que tenemos identificado el dispositivo y localizado el endpoint sobre el deberíamos trabajar, podría pensarse que acceder al dispositivo y realizar una petición de volcado de datos usando una transferencia de tipo *bulk*, podría ser posible. La idea es la siguiente:

```
//device es el dispositivo Optium Xceed, identificado mediante el código
anterior
//Obtenemos la configuración del dispositivo y la única interfaz q tiene
Configuration config = device.getConfiguration();
Interface interface = config.getInterface(0, 0);
//Obtenemos el endpoint con el que vamos a trabajar
Endpoint ep = itf.getEndpoint(0);
//Si es de lectura dispositivo → Host, se leerá de la misma manera que un
fichero
if (ep.isInput()) {
        InputStream in;
        in = ep.getInputStream();
        // Read in data here
```

No obstante, el código anterior no funciona debido a que a fecha de Marzo de 2011, el código para lecturas de tipo *Bulk* no ha sido implementado por los autores de la biblioteca.

Uso de la biblioteca JaWin

Después de investigar más en profundidad el problema anterior, se llegó a la conclusión de que el principal obstáculo para utilizar las bibliotecas anteriores es que dichas bibliotecas están pensadas para trabajar con dispositivos USB cuyo comportamiento siga un protocolo estándar (por ejemplo, las memorias USB). Sin embargo, estos medidores de glucosa tienen un comportamiento no estandarizado, suponemos que debido a que a los fabricantes les interesa que sólo se pueda acceder a ellos usando el propio software del fabricante.

Ante esta idea, decidimos cambiar el enfoque y tratar de acceder a los dispositivos usando algún tipo de API propia del laboratorio. De nuevo descubrimos que los fabricantes no ofrecen ningún tipo de biblioteca o documentación con este objetivo. Finalmente se decidió intentar aprovechar las propias bibliotecas dinámicas (DLL) que usa el programa oficial de Abbott, de

manera que desde la aplicación Java se pudieran ejecutar las funciones de estas DLL correspondientes a la descarga de la información. Para ello, se utilizó la biblioteca JaWin, un proyecto *open source* que permite cargar bibliotecas dinámicas y ejecutarlas desde aplicaciones Java.

Por medio de un proceso de ensayo y error, se consiguió identificar las bibliotecas dll necesarias para la comunicación suministradas junto al programa oficial de *Abbott Copilot Management System*.

De entre estas bibliotecas, destaca *Importer.dll,* que tras un análisis usando con el programa *DLL Export Viewer*, pudimos ver que esta biblioteca exportaba 4 funciones interesantes: *GetDataEx*, *GetDeviceImageName*, *GetDeviceList* y *StopRead*.

Con todo esto, se desarrolló el siguiente código:

```
//Se cargan todas las bibliotecas necesarias para el intercambio de datos
//Se tiene que tener en cuenta el orden de las bibliotecas por tema de
dependencias
System.load(Windows.class.getResource("./lib/jawin.dll").getPath());
System.load(Windows.class.getResource("AbbotHandler.dll").getPath());

//Bibliotecas importadas por Importer.dll, es necesario cargarlas antes
System.load(Windows.class.getResource("./lib/c4dll.dll").getPath());
System.load(Windows.class.getResource("./lib/HSAPI.dll").getPath());
System.load(Windows.class.getResource("./lib/ib_util.dll").getPath());
System.load(Windows.class.getResource("./lib/Instaide.dll").getPath());
System.load(Windows.class.getResource("./lib/libeay32.dll").getPath());
System.load(Windows.class.getResource("./lib/NucleusDB.dll").getPath());
System.load(Windows.class.getResource("./lib/ssleay32.dll").getPath());
System.load(Windows.class.getResource("./lib/borlndmm.dll").getPath());

System.load(Windows.class.getResource("./lib/Importer.dll").getPath());

//Una vez cargada la biblioteca, podemos, usando JaWin, llamar a las funciones
exportadas
```

Al percatarnos de que no se pueden usar las bibliotecas *DLL* tan fácilmente y tras investigar posibles maneras de deducir los parámetros (llegando incluso a estudiar el código objeto de estas bibliotecas), se llegó a la conclusión de que, por el momento, conseguir conectar los glucómetros con la aplicación java era una tarea demasiado complicada con trabajo suficiente como para constituir un proyecto por sí solo. Ante este marco, se decidió abandonar este módulo, no si antes dejar todo documentado por si en un futuro alguien decide retomar estas investigaciones.

I.5 Referencias

Jeff Friesen (2006), Java and USB,
http://today.java.net/pub/a/today/2006/07/06/java-and-usb.html.

The Jawin Porject (2011), Introduction to Jawin,
http://jawinproject.sourceforge.net/jawin.html

Nir Soler (2011), DLL Export Viewer v1.45,
http://www.nirsoft.net/utils/dll_export_viewer.html

Meneses Sánchez, Jesús David; Marín Martínez, Juan David (2008), Java y Usb,
http://code.google.com/p/conexion-aplicaciones-java-puerto-usb/downloads/list

Jojo , Mojo; Brownell, David (2011), About jUSB,
http://jusb.sourceforge.net/

Streetman, Dan (2006), JSR80 API Specification,
http://javax-usb.org/jsr80.pdf

Anexo I. Google Health and HIPAA

A diferencia de otras aplicaciones similares, Google Health no está regulado por la *Health Insurance Portability and Accountability Act* (HIPAA), una ley federal que establece las normas de confidencialidad de los datos de información de salud del paciente aprobada por el Congreso de EEUU en 1996. Pese a ello, la empresa norteamericana afirma estar comprometida con la privacidad de los usuarios, contar con estrictas políticas de seguridad de datos y asegurar que sean sólo los usuarios quienes controlen el acceso a su información. Además aseguran ser transparentes sobre cuál es la información que recopila cuando se usa Google Health y de cómo va a ser usada y se asegura que no se venderá o compartirá la información salvo autorización expresa del usuario.

Todo ello queda recogido en el documento *Google Health and HIPAA*, que se puede encontrar en la Página oficial de Google Health y que hemos recogido a continuación.

Google Health and HIPAA

Unlike a doctor or health plan, Google Health is not regulated by the Health Insurance Portability and Accountability Act (HIPAA), a federal law that establishes data confidentiality standards for patient health information. This is because Google does not store data on behalf of health care providers. Instead, our primary relationship is with the user. Under HIPAA, patients have a right to obtain a copy of their medical records. If they choose to use Google Health, we'll help them store and manage their medical records online.

Although Google Health is not covered by HIPAA, we are committed to user privacy and have in place strict data security policies and measures, and ensure that users control access to their information. We let users know what information we collect when they use Google Health, how we use it, and how we keep it safe. Users choose who views or adds information to their profile, and they can revoke access at any time.

There is no advertising in Google Health. We do not sell user health information, and we do not share it with other individuals or services unless a user explicitly authorizes us to do so, or in the limited circumstances described in our privacy policy. A user's personal medical records are stored in their secure account and cannot be accessed by others through a search on Google.com. Also, no personal or medical information stored in a user's Google Health profile is used to customize their Google.com search results.

The information below describes how Google Health's data confidentiality practices compare to those mandated by HIPAA.

For more information on Google Health's privacy practices, see our privacy policy.

For more detailed information on HIPAA, see:
http://www.hhs.gov/ocr/hipaa/
http://www.hipaadvisory.com/REGS/HIPAAprimer.htm

	HIPAA	Google Health
Do individuals have access to their medical records and health information?	Under HIPAA, patients can request a copy of their medical records from their health care	In Google Health, users have free and immediate web access at all times to the

	provider. This typically requires completing release paperwork and may require a printing or copying fee. In some circumstances, availability of certain records may be limited.	medical records and health information they store in their account.
Are individuals informed of how their information is used and protected?	Health care providers must provide patients with written notice of their HIPAA privacy rights.	Google provides users with a privacy policy when they sign up for Google Health. The policy is also posted online, along with Frequently Asked Questions, allowing users to reference it at any time.
What information is protected?	Under HIPAA, personally identifiable information is protected. De-identified patient information is not protected. Aggregate, de-identified patient information can be published and shared with third parties.	Under the Google Health privacy policy, personally identifiable information is protected. De-identified information, including our anonymous logs data, is restricted and cannot be shared with third parties. Aggregate, de-identified user information can be used to publish trends.
When is information sharing permitted?	Health care providers may share information with patient authorization, and may share without authorization, for certain purposes, such as: • When doctors or other health care providers share information to treat patients, like when faxing patient records for a referral • When used for payment, including	Google Health may share information with explicit user authorization, and may share without authorization in certain limited circumstances, such as: • With contractors and vendors operating solely on Google's behalf (subject to security and confidentiality requirements)

	sharing with insurance companies to pay for care • When employers face workplace injury claims • When public health researchers need aggregate information for studies • For health care operations, including to contractors and vendors operating on a provider's behalf (subject to security and confidentiality requirements)	• To protect against imminent harm to the rights, property or safety of Google, its users or the public, or to address fraud or violations of the Terms of Service
When is information sharing required?	Under various federal and state laws, health care providers must share patient information to comply with court orders and subpoenas. HIPAA itself also allows health care providers to voluntarily share patient information with law enforcement without a subpoena and without permission from or notice to the patient.	Under various federal and state laws, Google must share user information to comply with court orders and subpoenas. When possible, we notify the user in order to give them the opportunity to object. Under the Electronic Communications Privacy Act (ECPA), Google may not voluntarily share most user information with law enforcement.
How does the individual authorize sharing?	Patient authorization is not required for institutions to share information in the case of certain permitted disclosures, described above. When authorization is required, patients provide consent to share information through a written authorization form that must satisfy certain HIPAA	Users must request and give Google permission to share information through electronic authorization in their Google Health account. Sharing is revocable at any time.

Can individuals find out who has viewed or added information to their records?	Patients can request to see to whom their information has been disclosed in the last six years by requesting this information in writing from their health care provider. However, most disclosures, such as those for treatment, payment, and health care operations, do not have to be reported in response to such a request.	Every time data is added to a user's profile, the user is updated with a 'notice' on the main page of their profile. Users can see their full list of notices at any time. Users can view a full list of anyone that can currently view or add information to their account at any time in the settings tab of their Google Health account. This list does not include those who previously had access but from whom the user later revoked reading or editing privileges. Additionally, individual items that have been added to a user's account include a source--the name of the health care provider or institution that added the information--even if the source no longer has reading or editing privileges on the account.
How is information kept secure?	HIPAA requires that health care providers and other services maintain a minimum standard of "reasonable and appropriate safeguards to prevent intentional or unintentional use or disclosure of health information".	Google Health secures information by: • Using electronic security measures such as Secure Socket Layer (SSL) encryption, back-up systems, and other cutting-edge information security technology • Strongly restricting information access to a limited number of necessary personnel

Who enforces privacy protections?	Under HIPAA, the Department of Health and Human Services enforces HIPAA privacy protections through civil and criminal penalties. Read more information about HIPAA enforcement from the HHS Office of Civil Rights.	Under Section 5 of the Federal Trade Commission Act, the FTC enforces privacy protections in the Google Health privacy policy through civil and criminal penalties. State attorneys general and district attorneys have similar authority under general consumer protection laws.