Sistema recomendador orientado a la educación terapéutica del paciente diabético

Memoria de Proyecto de Sistemas Informáticos

Alberto Felipe Carrillo de Comas
Jose María Llovet Rodríguez
Israel Rodríguez Paulete

Departamento de Arquitectura de Computadores y Automática (DACYA)
Facultad de Informática (FDI)
Universidad Complutense de Madrid (UCM)
Dirigido por:
José Ignacio Hidalgo Pérez
Esther Maqueda Villaizán

Curso 2010-2011

Autorización

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.

Fdo.: Alberto Felipe Carrillo de Comas

Fdo.: Jose María Llovet Rodríguez

Fdo.: Israel Rodríguez Paulete

Agradecimientos

Lo primero agradecer la paciencia que ha tenido Esther para orientarnos en todo momento y conseguir adentrarnos en este campo médico. Fue realmente difícil la fase previa de análisis ante nuestro desconocimiento total de la diabetes y su aportación fue necesaria para la captura de requisitos iniciales y aclaración de conceptos médicos complejos.

También, ha sido clave en el desarrollo del proyecto el profesorado de la Facultad de Informática de la UCM. Desde nuestro director de proyecto D. José Ignacio Hidalgo, siempre disponible y nexo de unión entre nosotros y el especialista endocrino. D. José Luis Risco, director del desarrollo de la parte A, que también nos formó en el uso de tecnologías que desconocíamos y Juan Recio, del Grupo de Investigación GAIA, que nos solucionó dudas de funcionamiento de la herramienta jColibri.

Palabras clave

Diabetes Mellitus, e-Learning, Sistema Recomendador, Formación Terapéutica, CBR, Moodle, Google Health, jCOLIBRI

Resumen

La Diabetes Mellitus es una enfermedad común en la sociedad actual. A pesar de esto, es una afección con un tratamiento complejo y muchos de los pacientes que la padecen pueden presentar un cierto desconocimiento sobre algunos conceptos relativos a ella, y con los cuáles es conveniente familiarizarse para mantener un correcto control y evitar situaciones de riesgo. El sistema que se expone pretende ayudar al paciente recientemente diagnosticado, facilitando la primera toma de contacto con la enfermedad y ayudando a resolver los problemas derivados de ella.

La herramienta desarrollada utiliza datos y mediciones de analíticas del historial del paciente, en base a los cuáles recomienda una serie de consejos y contenidos formativos. Los contenidos se encuentran en un curso de formación online en el que se podrán realizar test para evaluar la progresiva adquisición de conocimiento sobre diferentes aspectos. El objetivo final es ayudar al paciente a mantener una buena salud y aumentar su calidad de vida.

Abstract

Diabetes Mellitus is a common illness in our current society. In spite of that, it's a hard affection with a complex treatment and lots of patients who are diagnosed with it can show some lack of knowledge about concepts related to it. It is important taking contact with these concepts to keep the control over the illness and avoid dangerous situations. The system which is being expounded tries to help patients, who have just been diagnosed of Diabetes, making easier the first experience with it and helping solving problems arising from it.

The developed application requests for information of patient's analytics, these measures are used by the system in order to generate a list of recommendations and formative contents. All contents are located in an on-line course where tests could be made in order to assess the progressive acquisition of knowledge about different aspects. The final purpose is helping patients to keep a good health and quality of life.

Índice general

Capítulo 1

Introducción

1.1. Breve descripción del problema

La diabetes Mellitus es una enfermedad autoinmune que afecta a más de 300 millones de personas –aproximadamente el 4,5% de la población mundial padece diabetes- y provoca en todo el mundo 3,8 millones de muertes al año [**1**]. Tras un primer análisis podríamos pensar que el motivo de que esta desorbitada cifra siga creciendo anualmente puede ser debido a la falta del necesario conocimiento de la enfermedad por parte del paciente diabético pero la realidad no es esa: se conocen perfectamente las causas y cómo realizar su conveniente tratamiento. Y entonces, ¿cómo es posible que los avances médicos no consigan reducir estos elevados porcentajes? Varios son los motivos entre los que podemos destacar los que exponemos a continuación.

Con el advenimiento y aplicación de las nuevas tecnologías, se ha avanzado considerablemente en el control, prevención y tratamiento de la diabetes. Desde que se elaboraron en la década de los 60 las primeras máquinas rudimentarias para la medición personal del nivel de glucosa en sangre –conocidos como glucómetros- hasta la evolución de los últimos modelos de páncreas artificial que autocontrolan la cantidad de insulina necesaria, han surgido un sinfín de proyectos de investigación con base tecnológica cuyo fin último era la mejora de la calidad y la esperanza de vida del diabético. A día de hoy podemos afirmar que estos avances han aportado un notable aumento en el bienestar del paciente porque le ayudan a sobrellevar las dificultades que se le presentan diariamente.

Existe un alto grado de implicación de los estamentos públicos y privados, nacionales e internacionales: desde la OMS –Organización Mundial de la Salud- hasta la más pequeña sociedad de diabetes tratan de formar a los diabéticos y de concienciar a la sociedad en general –a sabiendas de prevenir

una enfermedad que puede aparecer en edad adulta-. Se utilizan ingentes y diversos medios de difusión –notas de prensa, artículos especializados, revistas digitales,…- para determinar las pautas de la educación terapéutica del paciente diabético.

Ahora, si la medicina avanza en este campo y el nivel de formación de médicos y pacientes es elevado, ¿cuál es el problema real? El problema subyace en que existen factores de riesgo que potencian los efectos como puede ser la ausencia de hábitos de vida saludable –dieta y ejercicio físico- o las situaciones de estrés profesional y personal. Estas situaciones de la vida cotidiana, que afectan a la salud de cualquier individuo –pudiendo derivar en edad adulta en una diabetes de tipo 2-, tienen un mayor impacto en la diabetes; si bien es cierto que dependerá de la fisionomía y psicología de cada paciente ya que si algo hemos aprendido de esta enfermedad es que cada persona es un mundo y es difícil generalizar patrones de comportamiento aplicables a todos.

Ante la importancia de la educación para el riguroso control y la prevención de la diabetes y la necesidad de llegar al mayor número de pacientes, este entorno educativo debe ser lo más accesible posible. Garantizar estas características es posible a través de la red de redes, Internet. A día de hoy, la familiaridad que existe en la navegación Web y las posibilidades que ofrece –disponibilidad en hogares, dispositivos móviles,…- nos conduce a ofrecer formación a través de la Web.

La educación virtual, más conocida como *e-learning*, tiene un fuerte impacto en el ámbito de aplicación de nuestro proyecto. La creciente evolución en estos últimos años de las tecnologías aplicadas al aprendizaje y en particular, el desarrollo de los sistemas de gestión del aprendizaje –*LMS, Learning Management Systems*- [**2**] han provocado un profundo cambio en las estrategias educativas. A día de hoy la mayoría de instituciones educativas tratan de implantar estos sistemas como herramientas de apoyo a los sistemas educativos convencionales de manera que coexistan ambos modelos educativos. Algunas instituciones o centros, han optado incluso por ir más allá y ofertan cursos en los que se excluye el aprendizaje presencial de manera que se convierten en cursos a distancia exclusivamente "virtuales". Aprovechando el auge de estas herramientas de aprendizaje, utilizaremos el LMS de código abierto desarrollado por Moodle –que también está catalogado como CMS, Course Management System o VLE, Virtual Learning Environment-[1] en el contexto de la formación terapéutica de la diabetes. Nuestro sistema se sirve

[1] *http://moodle.org*

de la herramienta Moodle para entender la relación *"profesor-alumno-asignatura"* que existe dentro del ámbito académico como *"médico-paciente-curso"*, dentro del campo de la diabetes. Estas dos relaciones son funcionalmente idénticas en cuanto al enfoque formativo para el alumno (paciente) y el seguimiento del profesor (médico o especialista).

Más allá de la importante revolución en la educación, también cobra especial relevancia los avances tecnológicos en relación a la diabetes. A día de hoy, los diabéticos disponen de sencillos pero sofisticados glucómetros capaces de realizar análisis y almacenarlos periódicamente, y ser conectados como periféricos –mediante puertos USB, conexiones inalámbricas, infrarrojos o bluetooth- a computadoras con el objetivo de recoger o enviar la información al médico que lleva el seguimiento del paciente. La evolución de los *"smartphones"* –dispositivos móviles de última generación o móviles inteligentes- han permitido el desarrollo de aplicaciones que permiten introducir de manera manual y sencilla desde cualquier parte, los análisis de glucosa diarios[2].

Aunando las posibilidades que nos ofrecen estos avances, nuestro proyecto se enmarca dentro de un sistema global en colaboración con otros grupos de sistemas informáticos. El sistema en su totalidad se divide en tres módulos claramente diferenciados (ver Diagrama 1):

La parte A, *"Entorno de Obtención de datos y generación de informes"* –adjudicada a otro grupo del año académico en curso y con la participación de un alumno diabético- consiste en el desarrollo de diversas interfaces dedicadas a la recogida de información desde algunos modelos específicos de glucómetros y a su respectivo almacenamiento en la base de datos sanitaria de Google Health mediante el uso de una la API que provee Google de código abierto para desarrolladores y dedicada a tal efecto[3].

La parte B, *"Módulo de educación sanitaria y evaluación de conocimientos del paciente diabético"* –de la que somos responsables nosotros- consiste en proporcionar un sistema recomendador de la educación terapéutica de un paciente diabético debutante, es decir, aquel que ha sido recientemente diagnosticado. Utilizando la interfaz desarrollada por el grupo dedicado a la parte A podremos acceder a los datos de Google Health y mediante Moodle,

[2] "*Listado de aplicaciones para iPhone de la diabetes*" http://www.sugarfree.com.es/?p=936

[3] http://code.google.com/intl/es/apis/health/

una '*Herramienta de Evaluación On-Line'* (HEOL), el sistema será capaz de generar recomendaciones sobre el seguimiento y la formación del paciente. De esta manera trataremos de mejorar la comunicación entre médico y paciente, y en cierto modo simular un doctor virtual capaz de detectar hábitos de conducta equivocados que recomienden la necesidad de profundizar en la formación terapéutica del paciente.

El desarrollo de la parte C, "*Sistema de ayuda al profesional para la toma de decisiones en el tratamiento del paciente diabético*" quedará pendiente de asignar a un grupo de sistemas informáticos del próximo curso académico 2011-2012. Consistirá, mediante el uso de los datos y funcionalidades aportadas por las partes A y B y la inteligencia artificial, en un Sistema Experto capaz de modelar patrones de comportamiento del diabético.

Figura 1. Esquema general del sistema global.

1.2. Objetivos

El objetivo final del proyecto en su conjunto –incluyendo las 3 partes A, B y C anteriormente mencionadas- es crear un sistema capaz de realizar las funcionalidades descritas para mejorar el tratamiento y la calidad de vida del paciente diabético para ser implantado de forma experimental en los Centros de Salud dependientes del Hospital Virgen de la Salud de Toledo y que sirva de herramienta de control y seguimiento del paciente para los médicos de familia y especialistas dedicados y como herramienta formativa para el paciente diabético debutante. Los dos primeros módulos –parte A y parte B-

en fase de desarrollo durante el presente curso académico, estarán supervisadas por la Doctora Esther Maqueda, especialista del Servicio de Endocrinología y Nutrición del Hospital Virgen de la Salud de Toledo, encargada de orientarnos en el campo médico de la diabetes y de orientarnos durante la fase de análisis del proyecto.

Concretando sobre el módulo parte B, que es la que nos concierne, los dos objetivos principales son los siguientes:

1. Crear un Sistema de Formación On-Line. Como primer objetivo tendríamos que elegir una herramienta ya existente para parametrizar nuestra Herramienta de Evaluación On-Line orientada al diabético. La primera decisión sería descartar los LMS comerciales que hay en el mercado ya que es necesaria la adquisición de licencias. Tomada esta decisión, dentro de las posibles LMS de Software Libre que nos ofertó el profesorado –Sakai, Claroline- decidimos utilizar Moodle debido a que todos los componentes del grupo conocíamos de antemano su manejo a nivel de usuario al ser uno de los soportes utilizados por el Campus Virtual de la UCM para virtualizar asignaturas que hemos cursado a lo largo de la carrera. Nuestro objetivo por tanto será profundizar en la herramienta Moodle a nivel de administrador para ser capaces de parametrizar, publicar y alojar en un servidor de la Facultad de Informática un curso orientado a la educación terapéutica del diabético. La gestión de los contenidos, del temario y de los tests de evaluación correrá a cargo del especialista endocrino. Y nosotros realizaremos el diseño interno del curso. Será necesaria la realización de tutoriales visuales para formar a ambos perfiles de usuario, tanto al médico como al paciente, en la gestión y el uso del curso.

2. Desarrollar un Sistema Recomendador. Ofrecer al paciente una serie de recomendaciones formativas de acuerdo a la información que dispone del mismo relativa a:

- La recogida de datos inicial mediante formulario. A través de la propia aplicación y tras activar el módulo de formación y seguimiento, el usuario suministrará información inicial al sistema –desde conocimientos de la diabetes hasta conocimientos informáticos- para realizar las recomendaciones iniciales.

- Los datos suministrados por la Parte A del proyecto. Mediante la interfaz que han elaborado tendremos que acceder a la información del historial médico

del paciente en Google Health y dependiendo de los datos referentes a análisis, dosis de insulina, o realización de ejercicio recomendar la lectura de ciertos contenidos del temario al detectar la falta de formación tras el análisis de los mismos [**3**].

- La información relativa al curso en Moodle. Desde la aplicación, se tendrá que realizar la conveniente gestión de perfiles de usuarios, con sus respectivos permisos para facilitar el acceso. También, la aplicación tendrá que ser capaz de obtener la información del resultado de los tests realizados por los pacientes en el curso Moodle para conocer las preguntas realizadas por tema y las respuestas acertadas para poder dar las recomendaciones pertinentes.

Como objetivo global, la aplicación deberá ser una interfaz amigable con facilidad de uso. Es importante mencionar que la orquilla de edades de un diabético puede ser muy variada y tendrá que estar adaptada a todas, aunque principalmente esté enfocado al paciente debutante en temprana edad de diagnóstico.

1.3. Estructura de la memoria

Como el lector habrá podido comprobar este primer punto introductorio describe el marco de la diabetes en relación con las nuevas tecnologías, cómo estos avances han sido clave en la evolución de la diabetes en los últimos años y cómo actúa la sociedad para intentar paliar esta enfermedad que muchos expertos vaticinan que será la que tendrá mayor mortalidad en el siglo XXI. Se exponen también los objetivos globales del proyecto e individuales de la parte que nos corresponde y de qué manera tanto las recomendaciones del profesorado y la especialista como la fase previa de análisis del marco de la diabetes expuesto nos han ayudado a tomar ciertas decisiones de diseño.

En el segundo capítulo se hace una breve descripción de la enfermedad de la diabetes para conocer más de cerca el ámbito de aplicación y poder entender después el comportamiento del sistema.

En el capítulo tercero, citamos y presentamos las funcionalidades de las diferentes herramientas o tecnologías que hemos utilizado en el sistema, indicando cuáles han sido los motivos por los que hemos seleccionado éstas entre las diferentes posibilidades que se nos presentaban. En referencia a la

implementación de la aplicación hemos elegido el IDE NetBeans. Comentaremos las librerías necesarias para realizar las diferentes conexiones con el API de Google y el curso Moodle, la interfaz JDBC (Java Database Connectivity). Comentaremos las tecnologías utilizadas en la parametrización del curso de Moodle -phpMyAdmin v.5- y en la instalación del mismo en un servidor virtual Debian. En lo que se refiere al sistema recomendador hemos decidido utilizar una herramienta desarrollada por el GAIA –Group of Artificial Intelligence Applications del departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la UCM- llamada jColibrí que nos permitirá realizar nuestras recomendaciones mediante razonamiento basado en casos.

En el capítulo 4 se realiza una descripción detallada del sistema mediante casos de uso, mostraremos también la visión y arquitectura global, y desarrollaremos las funcionalidades tanto del sistema recomendador como del sistema educativo con sus respectivos detalles técnicos de implementación.

El capítulo 5 lo dedicamos a resumir las conclusiones del trabajo e indicaremos las pautas a seguir para completar los objetivos globales del sistema completo, cómo integrar los módulos y cómo encaminar el desarrollo pendiente de la parte C, el sistema experto.

Nota: Las referencias bibliográficas a artículos de investigación se detallan de manera numerada en la Bibliografía. Las referencias a páginas Web de documentación se realizan como pie de página.

Capítulo 2

Diabetes Mellitus

La Diabetes Mellitus (DM) es una enfermedad que se caracteriza por una hiperglucemia mantenida acompañada de alteraciones en el metabolismo de los hidratos de carbono, la grasa y las proteínas como consecuencia del defecto de secreción de insulina, de la acción de la insulina o de ambas. Existen diversos tipos de diabetes que fundamentalmente se pueden agrupar de la siguiente forma:

Tipo 1 (DM 1): Destrucción de células beta, que habitualmente provoca insuficiencia absoluta de insulina. A su vez puede ser autoinmunitaria o idiopática.

Tipo 2 (DM 2): Puede variar de una forma en la que predomine la resistencia a la insulina con insuficiencia relativa de insulina a una forma en la que predomine el defecto de la secreción con o sin resistencia a la insulina.

Otros tipos específicos:

- Defectos genéticos de la función de las células beta.
- Defectos genéticos de la acción de la insulina.
- Enfermedades del páncreas exocrino.
- Endocrinopatías.
- Inducidas por fármacos o sustancias químicas.
- Infecciones.
- Formas infrecuentes de diabetes de mediación inmunitaria.
- Otros síndromes genéticos que en ocasiones se asocian a diabetes.
- Diabetes gestacional.

2.1. Regulación de la secreción de Insulina

Existen distintos factores que intervienen en la regulación de la secreción de insulina desde el páncreas. Estos factores tienen distinta naturaleza, entre ellos podemos mencionar los siguientes:

- Nutrientes hidrocarbonados: La sustancia fisiológica más importante en la regulación de la secreción de insulina es la glucosa. La secreción de insulina no responde como función lineal a la concentración de glucosa. La relación entre la concentración de glucosa y la tasa de liberación de insulina sigue una curva sigmoidea, con un umbral que corresponde a la glucemia observada normalmente en condiciones de ayuno y con el segmento de la pendiente de la curva de dosis respuesta que corresponde al intervalo de glucemias alcanzadas normalmente en el estado postprandial. Existen otros azúcares, derivados de los azúcares alcoholes que también tienen efecto sobre la célula beta.
- Nutrientes no hidrocarbonados: se ha demostrado que los aminoácidos estimulan la secreción de insulina en ausencia de glucosa; los secretagogos más potentes son los aminoácidos esenciales arginina, leucina y lisina.
- Factores hormonales: la liberación de insulina por las células beta después de una comida está facilitada por una serie de hormonas tales como el GLP1, GIP, y la colecistoquinina. Estas hormonas no son secretagogas por sí mismas y sus efectos solo son evidentes en presencia de hiperglucemia. La respuesta secretora postprandial de la insulina también podría estar influida por otras hormonas intestinales peptídicas como el VIP, la secretina y la gastrina. Otras hormonas que ejercen un efecto estimulador de la insulina son la hormona de crecimiento, los glucocorticoides, la prolactina, el lactógeno placentario y los esteroides sexuales. Aunque las hormonas mencionadas pueden estimular la secreción de insulina indirectamente induciendo un estado de resistencia a la insulina, algunas pueden actuar directamente sobre las células beta, posiblemente para aumentar su sensibilidad a la glucosa.
- Factores nerviosos: los islotes están inervados por las ramas colinérgicas y adrenérgicos del sistema nervioso autónomo. L a estimulación simpática y parasimpática potencia la secreción de glucagón, mientras que la de insulina está estimulada por fibras del nervio vago e inhibida por las fibras nerviosas simpáticas. Sin

embargo, en los estudios no está clara la importancia del sistema parasimpático para mantener la tolerancia a la glucosa. También se ha cuestionado si las fibras nerviosas simpáticas que inervan los islotes ejercen una influencia importante en las respuestas secretoras insulínicas basal y postprandial. Es probable que la inervación simpática de los islotes explique la inhibición de la secreción insuliníca y el incremento del glucagón en el ejercicio y en respuesta a la hipoglucemia. La inhibición de la secreción de insulina por el sistema nervioso simpático puede explicar parcialmente el deterioro del control de la glucemia en sujetos sometidos a un estrés grave.

2.2. Patrón cronológico de la secreción de insulina

Se ha calculado que en 24 horas el 50% de la insulina total secretada se secreta en condiciones basales y el resto, en respuesta a la ingesta. Las tasas basales estimadas oscilan entre 18 y 32 unidades por 24 horas. La secreción de insulina es pulsátil; los principales pulsos se observan cada 1,5-2 horas. Estos pulsos se observan también cuando se administra glucosa intravenosa, por lo que no se relaciona con la absorción intermitente de alimentos en el intestino. Se han descrito variaciones circadianas en la secreción de insulina. Las respuestas postprandiales máximas se observan tras el desayuno. En diferentes estudios se observó que los sujetos presentaban respuestas secretoras máximas de insulina por la mañana y más bajas por la tarde y la noche. Se ha propuesto que esto pueda ser debido a una reducción de la reactividad de las células beta por la tarde y la noche. Los análogos de insulina de acción prolongada e intermedia disponibles en el momento actual intentan imitar el patrón de secreción basal pancreática, así como los análogos ultrarrápidos y las insulinas humanas de acción rápida intentar imitar el patrón de secreción tras la ingesta.

2.3. Justificación para mantener un buen control glucémico. Objetivos de control

Se debe mantener un buen control glucémico en primer lugar para evitar las complicaciones agudas y en segundo lugar para evitar las complicaciones crónicas (nefropatía, retinopatía, microangiopatía y macroangiopatía). Las complicaciones agudas son:

- Cetoacidosis diabética

- Hipoglucemia definida como un valor de glucemia inferior a 70. Esta a su vez se puede clasificar en leve (el sujeto es capaz de reconocerla y actuar para corregirla), moderada (el sujeto la reconoce, pero precisa ayuda para actuar por disminución del nivel de consciencia) y severa (el paciente está en coma y/o presenta convulsiones).

Múltiples estudios (UKPDS2, DCCT3) han demostrado que el buen control glucémico previene estas complicaciones. Posteriormente el estudio EDIC (Epidemilogy of Diabetes Inter ventions and Complications) que continuó al DCCT demostró que además la terapia intensiva era capaz de disminuir la morbilidad y mortalidad cardiovascular [11]. En los últimos años se ha demostrado que el control estricto de la glucemia en los pacientes críticos mejora los resultados [10], [13], [5], [7], [8], [6] y reduce los costes [9].

Para valorar el control glucémico se utiliza la cantidad de hemoglobina glucosilada o HbA1C que refleja el control de la glucemia durante el período de 6-8 semanas previo a la extracción de la muestra de sangre (ya que la vida media del eritrocito es de 120 días).

Dependiendo de las diferentes sociedades se establecen diferentes grados de control. Así la ADA y la EASD establecen que el control debe ser < 7 % mientras que las sociedades británicas bajan la HbA1C a $< 6, 5$ % si el paciente está solo con un antidiabético oral o $< 7, 5$ % si está con varios o insulina. Cuanto mejor sea el control más riesgo existen de hipoglucemias. En general se acepta que el control preprandial debe estar entre $70 - 110$ y el postprandial < 150.

2.4. Pautas actuales de tratamiento en DM tipo 1

El tratamiento intensivo con insulina es la terapia de elección en los pacientes con DM 1. Existen dos formas básicas el tratamiento convencional y el tratamiento intensivo, ambas opciones necesitan de un ajuste periódico de las pautas acorde a los controles de ingesta y ejercicio. El tratamiento insulínico convencional no refleja una terapia fisiológica como lo hace el tratamiento intensivo. Consiste en una a tres inyecciones diarias, normalmente con mezclas de insulina que se administran antes del desayuno y de la cena y a veces en comida. La terapia intensiva imita mejor la acción fisiológica de la insulina a través de varias inyecciones al día o con bomba de infusión continua. Se administra, por tanto una insulina basal y una insulina prandial (rápida o

ultrarrápida) antes de cada comida. Esto supone una media de 4 inyecciones diarias (una basal y mínimo 3 prandiales). A pesar de que se recomienda el tratamiento intensivo en la DM tipo 1 hay que tener en cuenta las siguientes consideraciones:

- Precisa una educación diabetológica exhaustiva, conociendo la dieta por raciones, el autoajuste diario de la insulina prandial para adecuarla a las raciones de hidratos de carbono así como a la actividad física prevista de acuerdo a unos objetivo glucémicos.
- Necesidad de autocontroles de glucemia capilar frecuentes (4 a 7 diarios) para el autoajuste de dosis.
- Mayor riesgo de hipoglucemias.
- Mejor calidad de vida.
- Mayor coste que la terapia convencional a corto plazo (aproximadamente tres veces superior) según refleja el DCCT [4] pero más eficaz al prevenir y/o retrasar las complicaciones crónicas [11].

A pesar de los avances en la terapia intensiva todavía existen pacientes que no consiguen el grado de control adecuado o que presentan múltiples hipoglucemias que ponen en riesgo su vida o disminuyen la calidad de la misma. La importancia del control metabólico y la dificultad para conseguirlo con los medios actuales provoca la necesidad de intentar desarrollar herramientas que ayuden a mantener un buen control glucémico y mejorar la educación sanitaria del paciente para favorecer su independencia con garantías.

Capítulo 3

Herramientas de SW utilizadas

3.1. Moodle

3.1.1. Introducción

Como comentamos en los objetivos del proyecto (punto 1.2) decidimos usar como herramienta para crear el curso la plataforma Moodle, porque aunaba las características que considerábamos más necesarias. Entre ellas, la de poder acceder externamente a la base de datos del curso para poder dar de alta automáticamente a los pacientes desde el momento en que estos soliciten el alta en el curso desde la aplicación.

Estudiamos la posibilidad de utilizar una herramienta de evaluación online desarrollada por el grupo HEOL, grupo de la Escuela de Ingeniería Técnica Informática de Sistemas del CES "Felipe II". Dicha herramienta es iTEST[4], y permite la generación aleatoria y corrección automática de tests online. Debido a las dificultades para poder acceder a la base de datos de esta herramienta y dar de alta a usuarios desde fuera (facilitando el trabajo al médico) y recoger los resultados de los tests realizados, decidimos finalmente decantarnos por Moodle.

Moodle es un Ambiente Educativo Virtual, el cuál proporciona un sistema de gestión de cursos, de distribución libre, que ayuda a los educadores a crear comunidades de aprendizaje en línea. Este tipo de plataformas tecnológicas también se conoce como LMS (Learning Management System). Es un

[4] *http://dosi.itis.cesfelipesegundo.com/heol/doku.php*

proyecto, en constante desarrollo, diseñado para dar soporte a un marco de educación social constructivista[5].

La filosofía planteada por Moodle incluye una aproximación constructiva basada en el constructivismo social de la educación, enfatizando que los estudiantes (y no sólo los profesores) pueden contribuir a la experiencia educativa en muchas formas. Las características de Moodle reflejan esto en varios aspectos, como hacer posible que los estudiantes puedan comentar en entradas de bases de datos (o inclusive contribuir con entradas ellos mismos), o trabajar colaborativamente en un Wiki.

Esta filosofía comúnmente se denomina "pedagogía construccionista social". Aún así, Moodle es lo suficientemente flexible para permitir una amplia gama de modos de enseñanza, utilizándose para generar contenido tanto de manera básica como avanzada (por ejemplo páginas Web).

Moodle actualmente no sólo se usa en las universidades, también se usa en enseñanza secundaria, enseñanza primaria, organizaciones sin ánimo de lucro, empresas privadas, profesores independientes e incluso padres de alumnos, y en todas las áreas de conocimiento incluyendo arte, idiomas, humanidades y matemáticas. Se ha establecido por sí mismo en el mundo del aprendizaje a lo largo del ciclo vital.

Un número cada vez mayor de personas de todo el mundo contribuye al desarrollo de Moodle de varias maneras. Como no hay pagos por licencias o límites de crecimiento, una institución puede añadir los servidores Moodle que necesite. Moodle se distribuye gratuitamente como Software libre (Open Source) (bajo la Licencia Pública GNU). El usuario puede copiar, usar y modificar Moodle siempre que acepte la siguiente condición: proporcionar el código fuente a otros, no modificar o eliminar la licencia original y los derechos de autor, y aplicar esta misma licencia a cualquier trabajo derivado de él. De esta manera, el desarrollo de Moodle continúa como un proyecto de software libre apoyado por un equipo de programadores y una comunidad de usuarios internacional, quienes solicitan contribuciones a Moodle Community que alienta el debate.

La instalación es sencilla requiriendo una plataforma que soporte PHP y la disponibilidad de una base de datos. Moodle tiene una capa de abstracción de bases de datos por lo que soporta los principales sistemas gestores de bases de

[5] *http://es.scribd.com/doc/6012729/Que-es-Moodle*

datos. En nuestro caso los técnicos de la facultad nos proporcionaron espacio en un servidor Debian, y allí nos instalaron tanto el curso, como el sistema de base de datos que utilizamos en el proyecto y que explicaremos más adelante: MySQL.

3.1.2. Características

Una importante característica del proyecto Moodle es la página Web moodle.org, que proporciona un punto central de información, discusión y colaboración entre los usuarios de Moodle, incluyendo administradores de sistemas, profesores, investigadores, diseñadores de sistemas de formación y, por supuesto, desarrolladores. Al igual que Moodle, esta Web está continuamente evolucionando para ajustarse a las necesidades de la comunidad, y al igual que Moodle, siempre será libre. Toda lo documentación de moodle se encuentra en Moodle Docs, para facilitar a los usuarios el uso de la aplicación, escrita por la comunidad de Moodle.

A continuación se detallaran de forma resumida las principales características que presenta Moodle en los 3 niveles de relevancia:

1. A nivel General:

   - **Interoperabilidad**: Debido a que el sistema Moodle se distribuye bajo la licencia GNU, propicia el intercambio de información gracias a la utilización de los "estándares abiertos de la industria para implementaciones Web" (SOAP, XML…). Al usar un lenguaje Web popular como PHP y MySQL como base de datos, es posible ejecutarlo en los diversos entornos para los cuales están disponibles estás herramientas tales como Windows, Linux, Mac, etc.
   - **Escalable**: Se adapta a las necesidades que aparecen en el transcurso del tiempo. Tanto en organizaciones pequeñas como grandes se pueden utilizar la arquitectura Web que presenta Moodle.
   - **Personalizable**: Moodle se puede modificar de acuerdo a los requerimientos específicos de una institución o empresa. Por defecto incluye un panel de configuración desde el cual se pueden activar o cambiar muchas de sus funcionalidades.
   - **Económico**: En comparación a otros sistemas propietarios Moodle es gratuito, su uso no implica el pago de licencias u otro mecanismo de pago.

- **Seguro**: Implementa mecanismos de seguridad a lo largo de toda su interfase, tanto en los elementos de aprendizaje como evaluación.

2. A nivel Pedagógico.

   - **Pedagógicamente flexible**: Aunque Moodle promueve una pedagogía constructivista social (colaboración, actividades, reflexión crítica, etc.), es factible usarlo con otros modelos pedagógicos.
   - **Permite realizar un seguimiento y monitoreo** sobre el alumno o estudiante.

3. A nivel funcional.

   - **Facilidad de uso**.
   - **Permite la Gestión de Perfiles de Usuario**. Permite almacenar cualquier dato que se desee sobre el alumno o profesor, no solo los que aparecen por defecto.
   - **Facilidad de Administración**. Cuenta con un panel de control central desde el cual se puede monitorear el correcto funcionamiento y configuración del sistema.
   - **Permite realizar exámenes en línea**, es decir publicar una lista de preguntas dentro de un horario establecido y recibir las respuestas de los alumnos. En el caso de las preguntas con alternativas o simples, es posible obtener las notas de manera inmediata ya que el sistema se encarga de calificar los exámenes. Las preguntas se almacenan en una base de datos, permitiendo crear bancos de preguntas a lo largo del tiempo.
   - **Permite la presentación de cualquier contenido digital**. Se puede publicar todo tipo de contenido multimedia como texto, imagen, audio y video para su uso dentro de Moodle como material didáctico.
   - **Permite la gestión de tareas**. Los profesores pueden asignar tareas o trabajo prácticos de todo tipo, gestionar el horario y fecha su recepción, evaluarlo y transmitir al alumno la retroalimentación respectiva. Los alumnos pueden verificar en línea su calificación y las notas o comentarios sobre su trabajo.
   - **Permite la implementación de aulas virtuales**. Mediante el uso del Chat o sala de conversación incorporada en Moodle, se pueden realizar sesiones o clases virtuales, en las cuales el profesor podría plantear y resolver interrogantes, mientras que los alumnos aprovechan la dinámica para interactuar tanto con el profesor así como con otros alumnos.

- **Permite la implementación de foros de debate o consulta**. Esta característica se puede usar para promover la participación del alumnado en colectivo hacia el debate y reflexión. Así como colaboración alumno a alumno hacia la resolución de interrogantes. El profesor podría evaluar la dinámica grupal y calificar el desarrollo de cada alumno.
- **Permite la importación de contenidos de diversos formatos**. Moodle permite integrar los contenidos en un gran número de formatos diferentes, incluyendo SCORM, Flash, Mp3s y canales RSS.
- **Permite la inclusión de nuevas funcionalidades**. La arquitectura del sistema permite incluir de forma posterior funcionalidades o características nuevas, permitiendo su actualización a nuevas necesidades o requerimientos.

3.1.3. Ventajas y desventajas

En los dos puntos anteriores hemos visto algunas de las razones por las que moodle se está convirtiendo en una de las herramientas de e-learning más valoradas e importantes. En este apartado indicamos las ventajas e inconvenientes que esta plataforma presenta[6].

Las principales ventajas de Moodle con respecto a otras plataformas e-learning son:

- Bajos costes.
- Alto nivel de seguridad.
- Revisión y mantenimiento constante.
- Gran flexibilidad.
- Capacidad de personalizarlo a tus necesidades.
- Código fuente disponible.
- Soporte técnico comunitario (documentación en línea).
- Constantes actualizaciones y plugins.
- Desarrollado en multitud de idiomas.
- Facilita la comunicación.
- Estabilidad.
- Interoperabilidad.

[6] - *http://blog.pucp.edu.pe/item/26621/ventajas-y-desventajas-del-moodle-aplicado-en-el-aprendizaje-cooperativo*
- *http://lobos.dcs.fi.uva.es/absi02/index.php/moodle-ventajas-y-desventajas/*

- Facilidad de uso.
- Escalabilidad.
- Alta Disponibilidad.

Aunque a primera vista parezca que el ahorro en costes pueda ser la mayor ventaja de moodle frente a sus competidores, la verdad es que sus herramientas, los estándares, su interoperabilidad, la gestión de usuarios y su atención online son realmente la clave de su éxito. No solo esto, sino que moodle está provocando una verdadera revolución en el e-learning ya que permite que comunidades con intereses comunes puedan trabajar juntos y compartir sus conocimientos.

A continuación exponemos las principales desventajas de Moodle con respecto a otras plataformas e-Learning:

- Algunas actividades pueden ser un poco mecánicas.
- Falta mejorar su interfaz de una manera más sencilla.
- Hay desventajas asociadas a la seguridad, según dónde se esté alojando la instalación de Moodle y cuáles sean las políticas de seguridad y la infraestructura tecnológica con la cual se cuente durante la instalación.
- Al ser una plataforma de tecnología abierta y por lo tanto gratuita, no se incluyen servicios gratuitos de soporte por lo que los costos de consultoría y soporte técnico están sujetos a firmas y entidades externas.
- La gestión de roles de las categorías de usuarios es insuficiente. Se muestran los mismos contenidos a todos los usuarios.
- No incorpora algunas de las herramientas pedagógicas más utilizadas. Por ejemplo, el seguimiento evolutivo de los estudiantes.
- Es necesario contar con un programador en PHP para su administración y programación personalizada.
- Minimiza el trabajo docente, se rompe el vínculo afectivo, imposibilita las interrelaciones presenciales entre docente y maestro.
- Sensación de aislamiento. Es muy importante que la plataforma a utilizar provea de distintas formas de fomentar la comunicación y colaboración (tanto de los estudiantes entre sí, como con los profesores y/o tutores) para suplir la falta de convivencia real.
- Para los docentes es muy difícil realizar un seguimiento a cada estudiante continuamente, y, por otro lado, es muy complicado explicar determinado tema de forma tal que sea comprensible por estudiantes con distintos niveles de aprendizaje.

Muchas de estas desventajas comentadas anteriormente las comparten el resto de plataformas e-Learning, sobre todo aquellas en las que se hace hincapié en la distancia existente entre el alumno y el maestro.

3.1.4. Futuras Directrices

Moodle es un proyecto activo y en constante evolución. El desarrollo de Moodle continúa como un proyecto de software libre apoyado por un equipo de programadores y una comunidad de usuarios internacional.

Según Moodle va madurando, su evolución cada vez está más influenciada por la comunidad de desarrolladores y usuarios. Una base de datos dinámica de nuevas funcionalidades y su estado puede consultarse en moodle.org/bugs. Las contribuciones de los usuarios en forma de ideas, código, información de uso y promoción son claves para el desarrollo de Moodle. También se puede pagar para que algunas funcionalidades se desarrollen más rápidamente (moodle.com/development).

Esta es la situación actual sobre el futuro de Moodle, teniendo siempre en cuenta que todo está sujeto a múltiples cambios dependiendo de los patrocinadores y de los desarrolladores.

3.2. jCOLIBRI 2.1

3.2.1. Introducción

Antes de abordar jCOLIBRI es necesario conocer un concepto esencial que da origen y forma a todo este software. Se trata del razonamiento basado en casos o "CBR", el cual pasaremos a explicar brevemente.

Un sistema CBR se basa en la resolución de problemas o situaciones propuestas por el usuario (casos) a partir del conocimiento adquirido anteriormente a través de la resolución de casos similares. La clave del éxito de estos sistemas es el conocimiento que posean sobre la problemática a tratar.

Este conocimiento se puede medir en función de la complejidad y detalle de los casos resueltos y del número de casos resueltos que "recuerde" el sistema.

De una manera simplificada, el proceso consiste en que dado un problema concreto, y una lista de problemas resueltos, el sistema CBR buscará el más parecido al nuestro y nos ofrecerá la solución que tiene asociada a él. En la siguiente imagen podemos apreciar de manera visual este comportamiento.

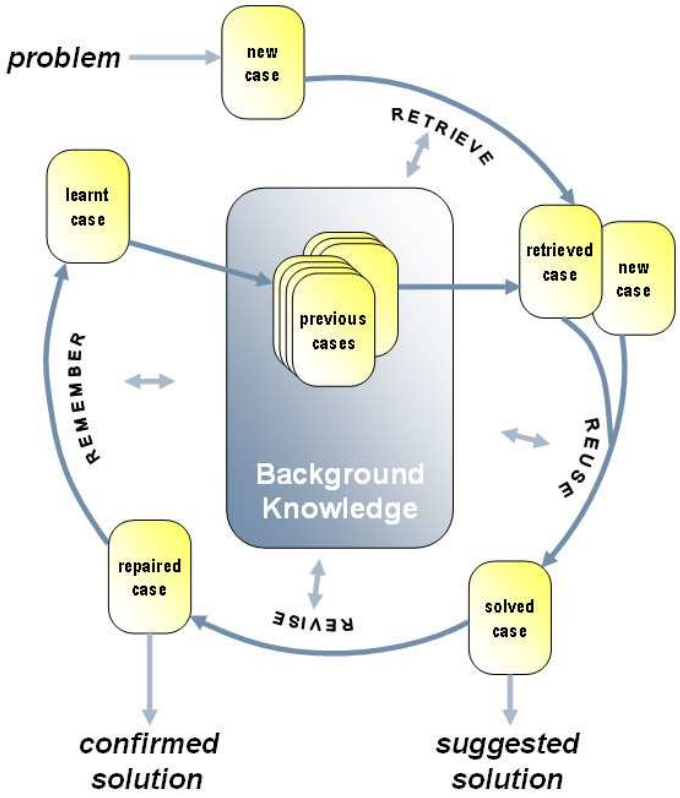

Figura 2. Sistema CBR.

Una vez nos hemos aproximado al concepto de CBR, podemos lanzarnos a comprobar la potencia qué la herramienta JCOLIBRI nos proporciona en este ámbito.

jCOLIBRI 2.1 se puede definir como un armazón o "framework" creado bajo entorno JAVA y cuyo objetivo es proporcionar diversas funcionalidades para facilitar el desarrollo de aplicaciones basadas en CBR's. Es una herramienta de código abierto (licencia LGPL) y ha sido desarrollado en su totalidad por el "Group for Artificial Intelligence Applications" (GAIA[7]), del Departamento de

[7] *jCOLIBRI, página web de GAIA* http://gaia.fdi.ucm.es

Ingeniería del Software e Inteligencia Artificial de la Universidad Complutense de Madrid.

El proyecto jCOLIBRI tiene sus orígenes en el año 2002, cuando se crea la herramienta COLIBRI, la cual estaba orientada hacia sistemas CBR con conocimiento intensivo (K-I CBR). Esta primera versión utilizaba una ontología de conceptos comunes en sistemas CBR denominada CBROnto y estaba desarrollada en LISP utilizando LOOM como tecnología para la representación del conocimiento.

El principal inconveniente de esta arquitectura era su dificultad y complejidad. Como consecuencia de esto y para hacerla más accesible a la comunidad de desarrolladores surge JCOLIBRI y sus sucesivas versiones.

A continuación se describirán las principales características de jCOLIBRI y se realizará una breve descripción de su arquitectura y modo de uso, incidiendo especialmente en su utilidad para la creación de sistemas recomendadores.

3.2.2. Arquitectura

jCOLIBRI 2.1 tiene dos ámbitos de uso claramente diferenciados. El primero de ellos está orientado a diseñadores, los cuáles a través de una herramienta visual podrán crear de una forma sencilla e intuitiva sus sistemas CBR. De esta forma el aprendizaje será más sencillo ya que no necesitarán comprender el núcleo de clases que compone el framework.

El segundo de ellos, y sobre el cual se profundizará más adelante está orientado a programadores. No posee ningún tipo de herramienta gráfica pero tiene acceso a todas y cada una de las interfaces y clases, para su uso y adaptación a la aplicación deseada.
Una de las principales innovaciones de jCOLIBRI 2.1 respecto a sus anteriores versiones es la representación de los casos como “Java Beans”.

En la siguiente imagen podemos ver de una forma más clara estas dos perspectivas de desarrollo.

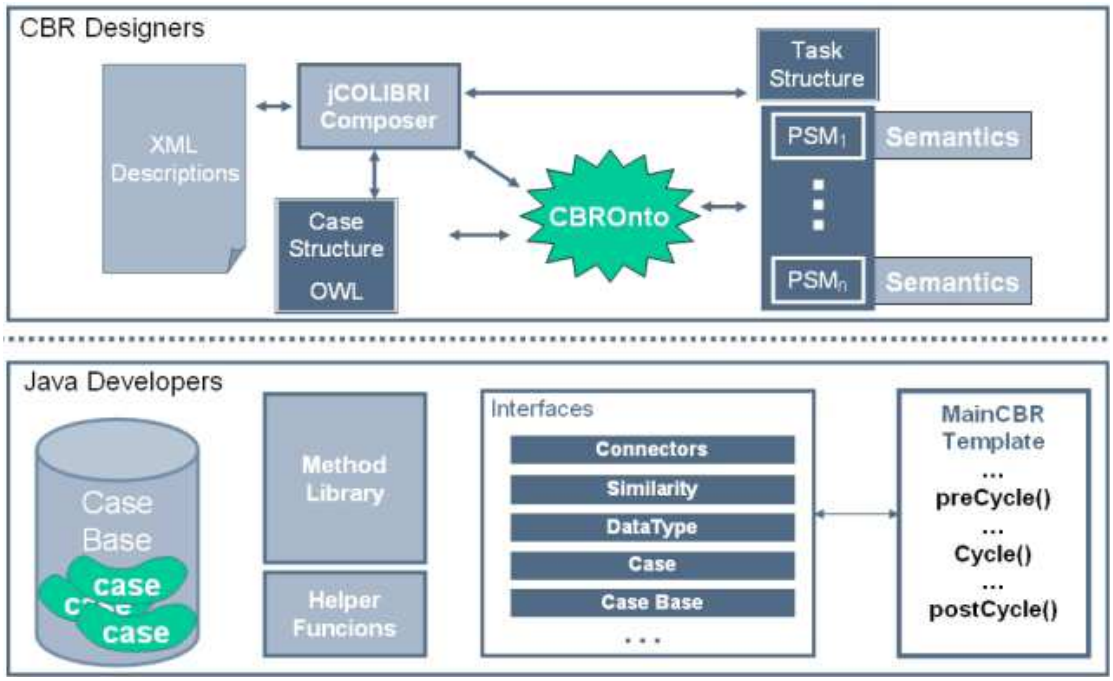

Figura 3. Perfiles de desarrollo.

Haremos especial hincapié en la arquitectura destinada a desarrolladores. Como podemos observar en la imagen inferior, jCOLIBRI se organiza en las siguientes secciones:

Base de casos: se trata del lugar donde está almacenada toda la información de los casos solucionados disponibles. Actualmente jCOLIBRI nos permite los siguientes sistemas de almacenamiento:

- Base de datos: podemos almacenar los datos de nuestros casos en una base de datos relacional, la cual podremos gestionar a través del paquete Hibernate (incluido en el framework), que nos permitirá obtener la información de una forma sencilla y rápida.

- Texto plano: es el método más sencillo para contener nuestros casos. Basta con crear un archivo de texto en el que introduzcamos una a una la descripción de cada caso con su solución.

- Ontología: podemos crear una ontología para representar nuestra base de casos. Posteriormente jCOLIBRI 2.1 nos proporciona la librería OntoBridge, también desarrollada por el grupo GAIA y que nos facilita las conexiones entre la ontología creada y nuestra aplicación.

Interfaces y métodos: el framework ofrece una gran cantidad de interfaces y métodos que deberemos implementar para desarrollar nuestro sistema CBR. A continuación se explicarán las más relevantes:

- **Connector:** implementando esta interfaz podremos crear nuestro propio conector para adecuarlo de la mejor manera posible a nuestras necesidades. Sin embargo, en el sistema de clases se nos ofrecen tres tipos de conectores definidos y que nos permiten un fácil y sencillo acceso a los tres tipos de almacenamiento indicados anteriormente (base de datos, archivo de texto y ontología). A continuación explicaremos más a fondo el conector destinado para la base de datos.

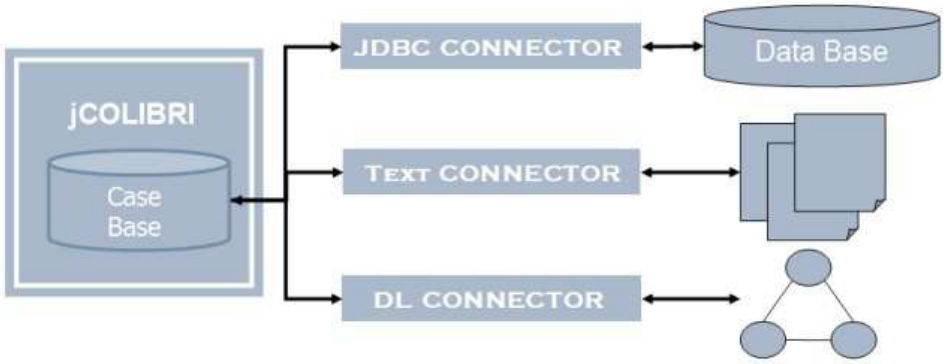

Figura 4. Tipos de conector.

La clase DataBaseConector está implementada por Hibernate, y para configurar la conexión se requerirá la creación de varios archivos xml.

- DataBaseConfig.xml: es el primer archivo procesado por el conector, actúa a modo de índice para encontrar el resto archivos de configuración. Además del resto de xml's, en los valores de DescriptionClassName y SolutionClassName se especifican las clases que representan la descripción y la solución de un caso. Estas clases serán contra las que se tendrán que mapear los datos adquiridos de la base de datos. La estructura del fichero es la siguiente:

```
<DataBaseConfiguration>
        <HibernateConfigFile>
                Ruta/hibernate.cfg.xml
        </HibernateConfigFile>
        <DescriptionMappingFile>
                Ruta/Description.hbm.xml
        </DescriptionMappingFile>
        <DescriptionClassName>
                Paquete.Description
        </DescriptionClassName>
        <SolutionMappingFile>
                Ruta/Solution.hbm.xml
        </SolutionMappingFile>
        <SolutionClassName>
                Paquete.Solution
        </SolutionClassName>
</DataBaseConfiguration>
```

- Hibernate.cfg.xml: este fichero es el encargado de proporcionar dos los datos y parámetros para establecer la conexión a nuestra base de datos.

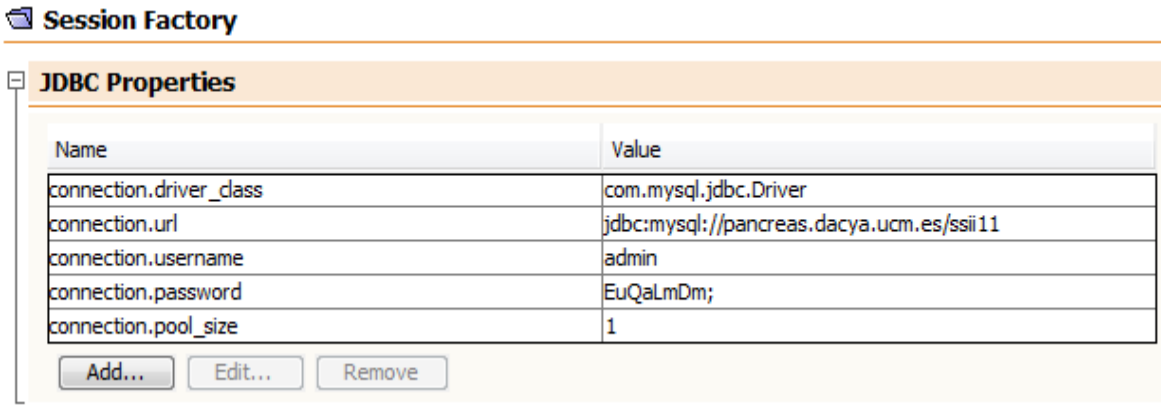

Figura 5. Archivo de configuración Hibernate.

- Solution.hbm.xml y Description.hbm.xml: contienen la información relativa a la correspondencia entre los atributos que forman la descripción y la solución (Java Beans) de un caso con los distintos campos de las tablas de la base de datos. Cuando se carga de la base de datos toda la información, necesitamos saber los campos que forman parte de cada clase.
- Si por ejemplo se tiene una tabla en la que existen 20 filas, cada una de ellas representa un caso.
- En la siguiente imagen se puede observar un ejemplo en el que el caso está formado por una descripción (verde) y una solución (naranja). En la parte inferior se encuentra

la base de datos en la cual cada campo está mapeado con la descripción o la solución dependiendo del color. Vemos que el campo, caseID está sombreado de azul, ya que simplemente es un identificador para poder diferenciar unos casos de otros.

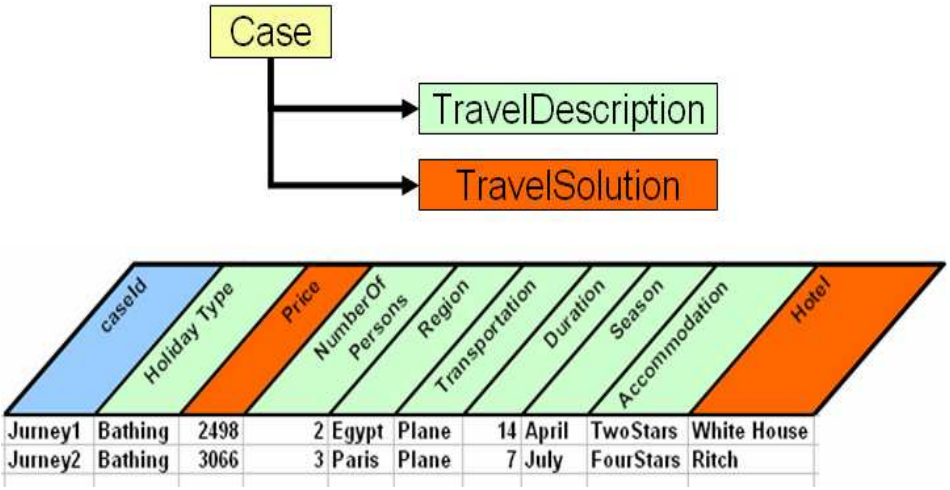

Figura 6. Mapeo de los atributos de un caso.

- La estructura de los xml's de mapeo es la siguiente:

```
<? xml v e r s i on =" 1 . 0 " ?>
< !DOCTYPE h i b e r n a t e - mapping PUBLIC
" -// Hi b e r n a t e / Hi b e r n a t e Mapping DTD/ / EN"
" h t t p : / / h i b e r n a t e . s o u r c e f o r g e . n e t / h i b e r n a t e -
mapping -3.0. d t d ">
< h i b e r n a t e -mapping de f aul t-l a z y =" f a l s e ">
< classname=" Paquete.NombreClase "
t a b l e =" NombreTabla ">
< i d name=" atributoID " column=" campoID ">
< / i d >
< p r o p e r t y name=" atributo " column=" campo " / >
< p r o p e r t y name=" atributo " column=" campo " / >
.....
< / c l a s s >
< / h i b e r n a t e -mapping>
```

- **Main CBR Template (StandardCBRApplication):** interfaz principal que debe implementar la aplicación. Proporciona los métodos esenciales para el funcionamiento del sistema CBR. Son los siguientes:

- Configure: nos sirve para configurar la aplicación y el conector.
- Precycle: una vez está configurada la conexión, este método carga la base de casos en memoria, creando un objeto de tipo CaseBase.
- Cycle: se le debe proveer de una descripción de un caso (query) como parámetro de entrada. A partir de este problema de entrada busca la solución o soluciones más apropiadas.
- Postcycle: cierra la base de casos y la conexión.

- **CaseBase:** mediante la creación de este objeto se cargará la base de casos y nos permitirá buscar la solución más adecuada a nuestra consulta. jCOLIBRI ofrece tres tipos de bases de casos predefinidas:

  - LinealCaseBase**:** los casos están representados a través de una lista.
  - IdIndexedLinealCaseBase: es una modificación de la anterior. En lugar de ser una lista, internamente está implementado como una tabla hash, en la cuál la clave es el "id" del caso.
  - CachedLinealCaseBase:

- **CBRQuery**: se trata de un objeto formado por un solo componente llamado descripción. Esta descripción debe implementar la interfaz CaseComponent, que veremos más adelante. La descripción de este objeto debe contener el problema o situación para la que se necesita buscar una solución.

  El usuario debe introducir una serie de datos que especifiquen su la descripción de su problema, y conforme a ellos se crea este objeto.

- **CBRCase:** extiende a CBRQuery. Además de tener un CaseComponent de tipo descripción, posee otro que contiene una solución. Estos objetos son los que componen la lista de casos que mantiene la aplicación en memoria.

- **CaseComponent:** interfaz que deben implementar las descripciones y soluciones asociadas a un caso o consulta.

- **NNConfig:** en esta clase configuraremos las distintas maneras de evaluar la similitud entre las descripciones de casos y consultas. Proporciona una puntuación a cada caso de nuestro CBR mediante la comparación de atributos.

  Esta puntuación se obtiene mediante el uso de funciones globales, que se aplican a objetos de tipo compuesto (CaseComponents y atributos compuestos) y funciones locales que se aplican a los atributos.

  También podemos otorgar a cada atributo un "peso" en función de la importancia que le queramos dar dentro de la evaluación. Si es un atributo muy determinante, podemos darle más valor que a otros que carezcan de relevancia.

  Un ejemplo podría ser el de una descripción (CaseComponent) a la que aplicamos una función global de tipo "calcular la media", y funciones locales de igualdad a sus atributos. De esta forma si un atributo cumple la función de igualdad le daremos una puntuación y después se calculara la media sumando todas las puntuaciones para otorgar un valor final.

- **NNScoringMethod**: clase que contiene el método evaluateSimilarity. Este método es el encargado hacer la búsqueda de casos devolviendo una colección de objetos de tipo Retrieval Result. A este método se le deberán pasar como parámetros la lista de casos de la que disponemos (CBRCase), la consulta o descripción proporcionada por el usuario (CBRQuery) y la configuración de las funciones del método de puntuación (NNConfig).

- **RetrievalResult**: objeto que representa el resultado del método evaluateSimilarity. Contiene un CBRCase, con su descripción y solución y una puntuación asociada.

- **Funciones de similitud local y global**: jCOLIBRI provee al programador de una gran cantidad de funciones de similitud. Cada función local solo puede ser aplicada a ciertos tipos de datos (enteros, strings, caracteres, etc.).
  También se podrán crear nuevas funciones implementando la interfaz *NNretrieval.similarity.GlobalSimilarityFunction* y NNretrieval.similarity. LocalSimilarityFunction.

Algunas de las que se proporcionan y cuya relevancia hemos considerado más importantes son las siguientes:

- Average (Global): como se indica en el nombre, calcula la media de las puntuaciones obtenidas de aplicar las diversas funciones locales a atributos simples.
- Interval (Local): requiere que se le pase un valor entero (intervalo) como parámetro. La puntuación calculada se obtiene siguiendo la siguiente fórmula: 1-(|v1-v2|/distancia), donde v1 y v2 son los valores que queremos comparar del atributo
- Equal (Local): devuelve la puntuación máxima en caso de ser iguales ambos atributos y cero en otro caso.
- EnumDistance (Local): calcula la distancia absoluta existente entre los valores de dos atributos, y en función de este valor devuelve la puntuación.
- MaxString (Local): compara dos cadenas de caracteres y calcula la puntuación en función de la máxima subcadena común que esté contenida en ambas.
- EqualsStringIgnoreCase (Local): otorga la máxima puntuación en caso de que ambas cadenas sean iguales, sin tener en cuenta letras mayúsculas y minúsculas.

Para concluir este apartado podemos apreciar de forma visual a través del diagrama de la siguiente página cómo se organiza nuestro sistema CBR utilizando las clases y métodos que se han explicado hasta ahora:

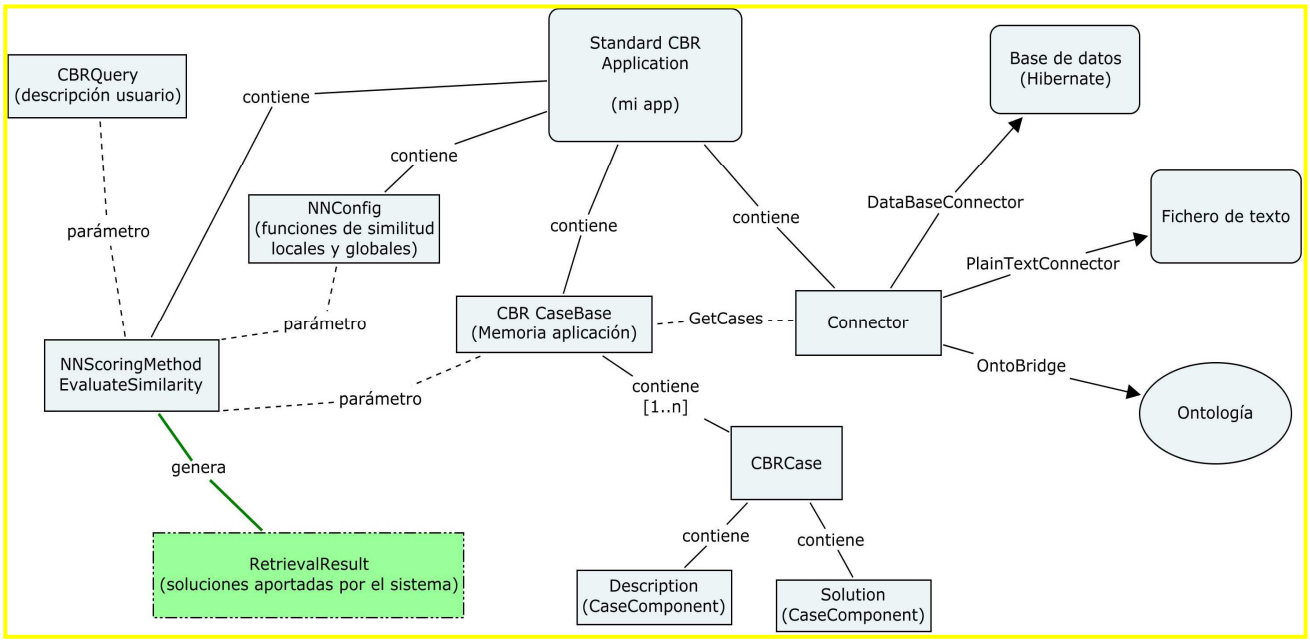

Figura 7. Estructura del recomendador.

3.2.3. Modo de uso y compatibilidad con NetBeans IDE 6.9.1

En este apartado explicaremos mediante sencillos pasos como integrar la herramienta jCOLIBRI 2.1 en nuestro entorno de desarrollo JAVA, en este caso NetBeans IDE 6.9.1.

Descarga e importación de la herramienta.

El software jCOLIBRI se puede descargar a través de la página del grupo GAIA de la Facultad de Informática de la Universidad Complutense de Madrid mediante la siguiente url: http://gaia.fdi.ucm.es/projects/jcolibri . Aquí encontraremos una amplia documentación que nos permitirá conocer mucho más a fondo la herramienta.

Descargaremos el paquete autoinstalable (.exe) y lo ejecutaremos siguiendo los pasos de instalación.

jCOLIBRI requiere Java 1.6 o posterior para su funcionamiento.

Una vez el framework esté instalado, lanzaremos Eclipse IDE creando un nuevo workspace y seleccionaremos la opción importar proyecto existente a nuestro workspace. Debemos seleccionar para la importación el proyecto

jCOLIBRI, por defecto se instala en C:/Archivos de Programa. Marcaremos también la opción de copiar las carpetas del proyecto al workspace.
Una vez finalizado este proceso, ya está preparado para su uso en Eclipse. Pero si queremos utilizarlo en NetBeans IDE tendremos que realizar un último paso.

Lanzaremos NetBeans IDE 6.9.1 y seleccionaremos la opción importar proyecto desde Eclipse. A continuación seleccionaremos Importar proyecto ignorando dependencias de Proyecto, y rellenaremos los campos con la ruta en la que se encontraba el workspace de jCOLIBRI en Eclipse y la ruta a la cual queremos copiar este proyecto para su uso en NetBeans. Pulsamos en finalizar y ya tendremos jCOLIBRI lista para ser utilizado.

Es posible que al concluir el proceso aparezca un error en la siguiente clase: */src/jcolibri/method/maintenance/CaseResult.java*

Para solucionar este error, debemos renombrar el siguiente método:

```
public static List sortResults(boolean ascending, List<CaseResult> toSort)
```

Una vez renombrado, debemos modificar también las llamadas que se hagan al método en el resto de clases del framework. Finalmente, añadiremos una referencia a jCOLIBRI en el apartado de librerías de nuestro proyecto, para de esta manera poder disfrutar de toda las funcionalidades que nos ofrece.

Pasos esenciales para la construcción de nuestro CBR.

Como resumen, a continuación se indican unas breves directrices de cómo crear una aplicación CBR a través de jCOLIBRI.

1) Crear una clase que implemente la interface StandardCBRApplication.

2) Crear clases para las descripciones y soluciones que deberán contener los casos de nuestro sistema. Deben implementar la interface Casecomponent.

3) Crear la base de casos a través de una base de datos, fichero de texto u ontología.

4) Una vez decidido el método de almacenamiento de la base de casos, se deberá proceder a la configuración del conector y sus xml's y mapeos asociados.

5) Implementamos los métodos configure(), precycle(), cycle() y postcycle() que determinarán el funcionamiento de nuestro sistema.

3.2.4. Futuro y evolución de JCOLIBRI

Para conseguir que jCOLIBRI sea realmente una plataforma de desarrollo de sistemas CBR utilizada por un amplio número de desarrolladores, se le deben proporcionar los mecanismos adecuados.

Partiendo de esta base, una posible evolución nos llevaría a una arquitectura basada en componentes distribuidos. De esta manera se podría alcanzar mayores niveles de colaboración e integración entre la comunidad de desarrolladores. Un desarrollador podría desarrollar una nueva funcionalidad a integrar en la herramienta y ponerla al servicio de la comunidad.

El principal problema para implementar esta idea es la dificultad del uso remoto de ciertas funcionalidades a través de la red, que ralentizarían la ejecución de una aplicación en comparación con un uso plenamente local.

3.3. Hibernate

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones[8]. En la siguiente imagen vemos un pequeño diagrama de su arquitectura interna:

[8] *http://www.javahispano.org/contenidos/es/manual_hibernate/*

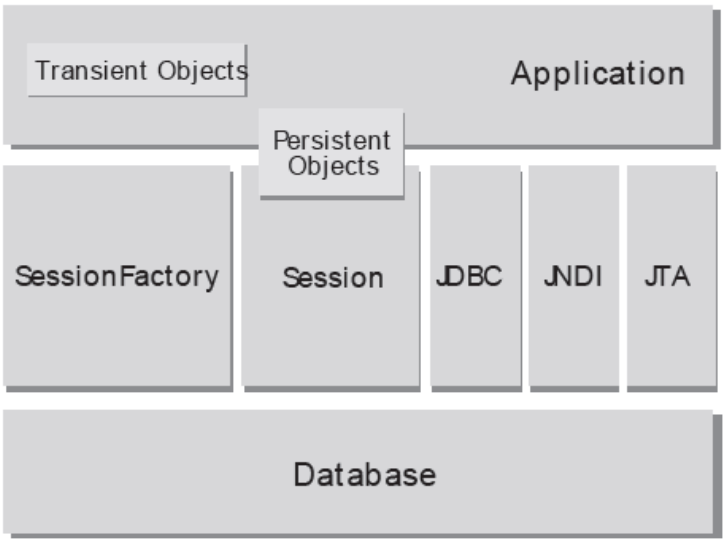

Figura 8. Arquitectura de Hibernate.

Hibernate es el gestor de base de datos que utiliza jCOLIBRI internamente, proporcionando una fácil configuración de la conexión.

El principal objetivo que persigue es resolver la diferencia entre el modelo de datos relacional existente en las base de datos y el modelo de datos que se representa en Java mediante los distintos objetos.

Hibernate resuelve este problema mediante mapeos especificados a través de ficheros xml. En estos ficheros se indica la clase Java a la cual queremos cargar los datos, y la tabla o tablas de las cuales los leeremos.

Entre sus principales características podemos resaltar las siguientes[9]:

- Modelo de programación natural. Hibernate es una capa de persistencia objeto/relacional que e permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos.
- Gran escalabilidad
- Software libre. Está bajo licencia LGPL (Lesser GNU Public License).
- EJB 3.0. Hibernate implementa la gestión de la API de la persistencia Java y el mapeado objeto-relacional.
- Persistencia transparente. Ofrece soporte para un amplio conjunto de las colecciones de Java, propiedades del estilo de persistencia de JavaBeans, etc. que abstraen al usuario.

[9] *http://www.hibernate.org/*

- Mapeado flexible gracias a las asociaciones bidireccionales, la persistencia transitiva, colecciones de tipos básicos, etc. el mapeado resulta mucho más flexible.
- Facilidades en consultas. Debido en parte a que se realizan en un potente lenguaje de consultas orientado a objetos.
- Facilidades en metadatos. Soporta el formato del mapeado de XML, diseñado para ser editado a mano y el mapeado basado en anotaciones. Además de Validación basada en anotaciones.
- Incluye un lenguaje propio para consultas llamado HQL (Hibernate Query Language).

3.4. phpMyAdmin

3.4.1. Introducción

Para la conexión e interrelación con nuestra base de datos nos hemos servido de una tecnología Java llamada JDBC. Así realizamos las consultas, inserciones, modificaciones, cancelaciones,… desde el código de la aplicación. Veremos esta tecnología más en detalle en el apartado 3.5.4.

Pero para poder controlar la correcta conexión con la base de datos, así como verificar su correcto funcionamiento a la hora de realizar las distintas operaciones sobre ella desde el código, y para poder hacer pruebas, nos hemos servido de la herramienta phpMyAdmin[10].

Generalmente en una aplicación Web (en nuestro caso Moodle), disponemos de un servidor remoto que tiene instalado el motor de base de datos MySQL y el lenguaje PHP, entre otras funcionalidades. La base de datos MySQL se puede administrar perfectamente conectándose al servidor en forma remota, mediante **SSH**, y desde ahí ingresar utilizando el comando MySQL.

Como herramienta administrativa, MySQL vía SSH, se vuelve bastante poco funcional, sobre todo para un usuario no experimentado, ya que como se puede observar, el manejo es similar al de una pantalla del sistema operativo **DOS** (Figura 9). Si además de considerar eso, contemplamos que no muchos

[10] *http://www.phpmyadmin.net/home_page/index.php*

proveedores de hosting, nos permiten acceder vía SSH al servidor, tendremos el problema de que no podremos ingresar a configurar nuestra base de datos.

Para solucionar este tema, se comenzaron a desarrollar distintas aplicaciones Web, que nos permiten manejar desde el navegador Web nuestras bases de datos. Entre estas aplicaciones que se desarrollaron, tenemos a phpMyAdmin. No sólo es un programa OpenSource, sino que es "el programa" con mayúsculas para la administración de bases de datos MySQL en forma remota.

También debemos considerar que para administrar una base de datos, necesitaremos saber algo de **SQL** (*Structured Query Language*), que es el lenguaje que se utiliza para realizar una consulta a la base de datos. Gracias a phpMyAdmin esto es bastante más sencillo debido a que al ser una interfaz gráfica, escrita en PHP, estamos en disposición de interaccionar con la base de datos sin la necesidad de escribir alguna línea de código SQL.

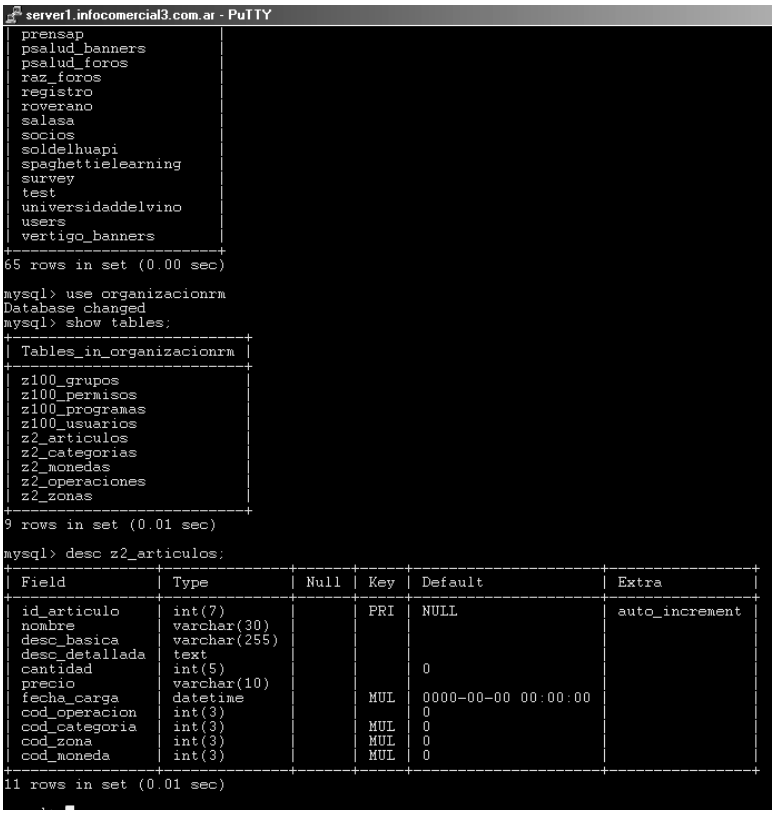

Figura 9. Estructura de una tabla de una base de datos.

Otro aspecto a destacar es que la versión de phpMyAdmin está disponible en 62 idiomas diferentes, y se puede cambiar entre los distintos lenguajes

disponibles, simplemente seleccionando el lenguaje de nuestro interés desde una lista desplegable, lo cual la hace sumamente adaptable a muchos usuarios y países.

La pantalla principal de phpMyAdmin (Figura 10), está compuesta por una página de marcos, dividida en dos partes: por un lado el lateral izquierdo donde podrá observar todas las bases de datos a las cuales tiene acceso y por el otro, el lateral derecho donde le mostrará en principio todas las opciones a las cuales puede acceder según sus privilegios de usuario, y posteriormente, luego de haber seleccionado una base de datos desde el marco izquierdo, podrá ver las propiedades de la misma, como cantidad de tablas, registros, etc.

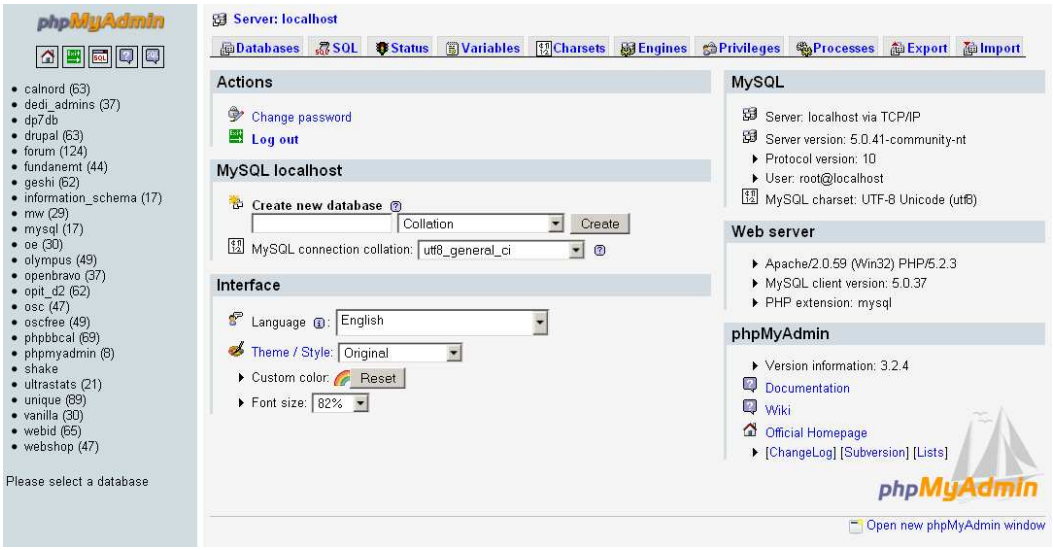

Figura 10. Aspecto de la herramienta phpMyAdmin.

3.4.2. Instalación

Los requerimientos para poder instalar phpMyAdmin son:

- **PHP** versión 5.2.0 o posterior, con soporte de sesión, la extensión "Standard PHP Library" (SPL) y soporte JSON.
- **MySQL** 5.0 o superior.
- Navegador Web con cookies activadas.

Para instalarlo basta con descargar la última versión estable del sitio Web www.phpmyadmin.net/home_page/downloads.php y seguir las instrucciones http://www.phpmyadmin.net/localized_docs/es/Documentation.html.

El programa se maneja desde una interfaz Web y es normal encontrarlo pre-instalado en casi todos los servicios de hosting. De hecho instalando **XAMPP** (paquete de instalación de un servidor PHP - Apache - MySQL) se podrá ver como ya viene configurado y listo para usar.

Este paquete es el que utilizaron cuando nos instalaron Moodle en un servidor del *Departamento de Arquitectura de Computadores y Automática.* De esta manera tenemos acceso a las tablas de la base datos de Moodle, necesarias para la obtención de datos de los usuarios, y también disponíamos de un espacio en un servidor donde crear nuestra propia base de datos para la gestión interna de la aplicación.

3.4.3. Características

Este programa permite acceder a todas las funciones típicas de la base de datos MySQL de forma muy intuitiva. Básicamente no se necesita tener conocimientos previos de base de datos para comenzar a crear tablas y agregar contenido. Alguna de las características de phpMyAdmin más importante son las siguientes:

1. Administración completa de las Bases de Datos.
2. Administración completa de las Tablas.
3. Ejecuta sentencias SQL.
4. Exporta datos a diferentes formatos.
5. Administra usuarios y privilegios de MySQL.
6. Es un Administrador Multiplataforma.
7. Esta herramienta se encuentra bajo la Licencia GNU/GPL (GNU General Public License).
8. Escrito en el lenguaje de programación PHP4 compatible con PHP5.
9. Proporciona una Interfaz Web intuitiva.
10. Importar datos y estructuras de MySQL desde hojas de cálculo de Microsoft Excel y OpenDocument, así como de ficheros XML, CSV, y SQL.
11. Administrar múltiples servidores.
12. Gestionar privilegios y usuarios de MySQL.
13. Transformar los datos almacenados a cualquier formato usando un conjunto de funciones predefinidas, como mostrar objetos binarios (BLOBs) como imágenes o enlaces de descarga.

3.4.4. Ventajas de phpMyAdmin

Primero de todo, decir que Para la administración de base de datos MySQL existen varias alternativas. Una de las más populares es un software llamado **SQLyog**[11]. Es un programa muy potente que permite conectarse con varias bases de datos y guardar los accesos en el mismo programa para poder administrarlas fácilmente. La desventaja que tiene es que **NO es gratuito**.

Una de las ventajas que tiene es que al tener una interfaz Web no se necesita estar en la computadora de cada uno, el programa se instala en el mismo lugar del posting. Así desde cualquier computadora con acceso a Internet se pueden realizar consultas, y acciones sobre la base de datos que se esté utilizando para trabajar.

Otra ventaja es que es muy intuitivo y se encuentra instalado en casi todos los servicios de hosting y estar presente en casi todos los proveedores de hosting a nivel mundial, además de haber ganado numerosos premios. No solamente por las funcionalidades que nos ofrece y que van mejorando con cada nueva versión, sino también porque a lo largo de los años ha sabido ganarse su lugar.

Las principales operaciones que podemos realizar con phpMyAdmin son:

- Crear y eliminar una base de datos
- Crear, modificar, vaciar y eliminar una tabla.
- Insertar, modificar y borrar campos de una tabla.
- Ejecutar sentencias SQL, sobre la base de datos.
- Realizar una copia de seguridad de la base de datos.

3.5. Java y NetBeans

3.5.1. Introducción

A la hora de desarrollar nuestra aplicación decidimos decantarnos por la utilización del lenguaje Java, utilizando el entorno gráfico que nos proporciona la plataforma NetBeans. En gran parte se debe a que tanto el lenguaje como

[11] http://www.elwebmaster.com/editorial/taller-de-php-mysql-phpmyadmin

la plataforma son objeto de estudio durante la carrera. Las razones de esta decisión las exponemos brevemente a continuación[12].

Java es un lenguaje de programación orientado a objetos que comparte gran parte de su sintaxis con C y C++ (lenguajes también vistos en la carrera). Java es uno de los lenguajes más poderosos que existen actualmente y desde la versión 6 es un proyecto OpenSource, por lo que tiene el soporte de toda una comunidad de programadores, además de Sun Microsystems que fueron los creadores originales. En 2005 Java se utilizaba en uno de cada cuatro proyectos, casi diez por ciento por encima de su siguiente competidor (C++) y ha seguido creciendo. Se estima que un noventa por ciento de las computadoras cuentan con una máquina virtual de Java, además de que todos los celulares y una gran cantidad de dispositivos móviles también cuentan con Java. Esto supone una gran cuota de mercado.

NetBeans es un entorno de desarrollo integrado (IDE por sus siglas en inglés). Esto quiere decir que integra todas las herramientas que necesitamos para poder desarrollar. Originalmente la programación en Java era algo complicada porque Java cuenta con una enorme cantidad de librearías y funciones que era preciso aprenderse de memoria, viendo esto muchas compañías construyeron diferentes entornos de programación para facilitar la tarea del programador. Entre los más populares surgió Eclipse que reinó como el único y más importante IDE de Java durante varios años. Sun Microsystems desarrollo su propio IDE, que tenía la ventaja de que fue creado por las mismas personas que crearon Java años antes, este IDE fue NetBeans y después de varios años de desarrollo ha llegado a ser tan útil y poderoso como Eclipse o quizás un poco más.

3.5.2. Características de NetBeans

- Es un IDE multilenguaje completo y modular:
  - Soporte para Java SE, Java EE, Java ME
  - Gran cantidad de módulos de terceros (plugins)
  - Gran cantidad de módulos de terceros (plugins)
  - Desarrollo intuitivo drag-and-drop
  - Debugger, Profiler, Refactoring, Completa código
  - Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial

[12] http://www.magusoft.net/compuv/01_Netbeans.html

(manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

- Es gratis y Open Source
  - Gran comunidad de usuarios y desarrolladores
- Una plataforma para construir aplicaciones completas para el cliente
- El IDE ha sido desarrollado para distintas plataformas:
  - Linux
  - MacOS X
  - Solaris
  - Windows

3.5.3. Java y bases de datos

Uno de los retos a los que nos enfrentamos por primera vez a la hora de crear nuestra aplicación fue la conexión e interrelación con una base de datos externa. En este caso, como vimos en el apartado 3.4, una base de datos MySQL.

Para ello nos servimos de la **tecnología JDBC:**

- *Java DataBase Connectivity* es la tecnología Java que permite a las aplicaciones interactuar directamente con motores de base de datos relacionales.
- La **API JDBC** es una parte integral de la plataforma Java, por lo tanto no es necesario descargar ningún paquete adicional para usarla.
- JDBC provee una interfase única, que independiza a las aplicaciones del motor de la base de datos.
- Un **driver JDBC** es usado por la JVM para traducir las invocaciones JDBC en invocaciones que la base de datos entiende.

Existen drivers JDBC para la mayoría de los motores de base de datos. Típicamente, los fabricantes de bases de datos proveen el driver JDBC para su motor.

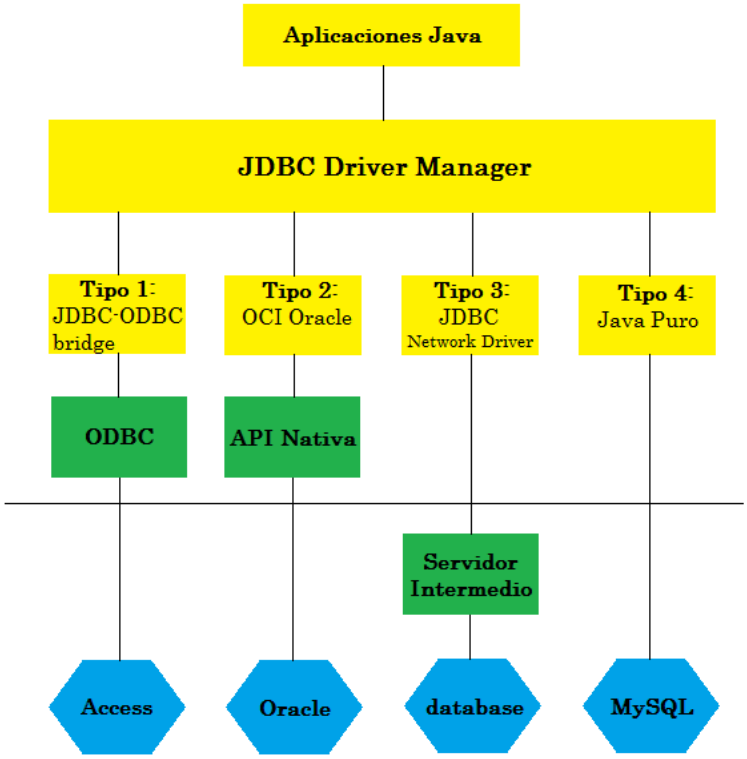

Figura 11. Arquitectura del JDBC

Como podemos observar en la figura anterior, hay 4 tipos de driver. Nosotros utilizamos el driver de tipo 4, por las siguientes razones:

- Es un driver Java Puro, que habla directamente con la base de datos.
- Es el método más eficiente de acceso a bases de datos.
- No requiere de ninguna librería adicional ni de la instalación de un *middleware*, con lo cual es de *deployment* más simple.
- La mayoría de los fabricantes, proveen drivers JDBC de tipo 4 para sus bases de datos.

Figura 12. Driver tipo 4 (100% Java)

Capítulo 4

Descripción del sistema

4.1. Arquitectura del sistema

4.1.1. Organización de Componentes (Partes A, B y C)

En la introducción mostrábamos una visión del sistema global, citábamos las tres partes en que se dividía el proyecto y mencionábamos cuáles eran los objetivos a gran escala de cada una de ellas. Ahora, desde un punto de vista más técnico, explicaremos la manera en que se van a interconectar estos 3 componentes.

La parte A, *"Entorno de Obtención de datos y generación de informes"* desarrolla diversas interfaces de entrada-salida: la entrada, datos de medidas de glucemia –cantidad de insulina en sangre- desde algunos modelos específicos de glucómetros e información introducida a mano por parte del usuario sobre análisis o información médica de interés para completar el historial del paciente. La salida, información ordenada para ser guardada de forma persistente en la base de datos de Google Health. La aplicación que han desarrollado nos ofrece una API en Java con los servicios necesarios para conectarnos a esta base de datos y disponer de la información que necesitamos.

En nuestra parte B, inicialmente titulada como *"Módulo de educación sanitaria y evaluación de conocimientos del paciente diabético"*, tenemos que desarrollar un curso de formación en Moodle, y nuestro sistema será capaz de ofrecer contenidos y evaluar al paciente para luego recoger la información del curso y generar las pertinentes recomendaciones sobre la formación adquirida.

También nuestro sistema se adentra en el inicio del desarrollo de la parte C, "*Sistema de ayuda al profesional para la toma de decisiones en el tratamiento*

del paciente diabético". En este año académico y durante esta fase experimental del proyecto, inicialmente las recomendaciones de seguimiento se programaron de manera mecánica y sencilla –comparaciones sistemáticas de los datos- pero al final hemos realizado una pequeña mejora y hemos incluido las recomendaciones dentro de una herramienta ya desarrollada para aplicar técnicas de inteligencia artificial de manera que sirva de base para el futuro desarrollo del complejo modelo de doctor virtual que ser quiere simular.

En la Figura 13 se muestra cómo interactúan los componentes dentro de nuestro sistema, la Parte B del proyecto cuya aplicación ya implementada a la que hemos dado el nombre de GlucoAprende.

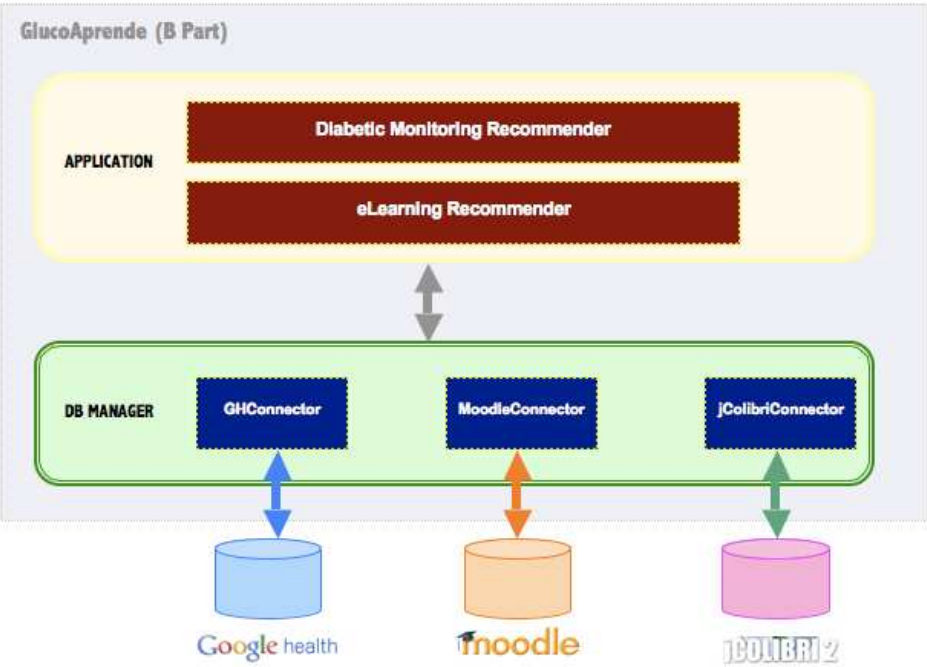

Figura 13. Visión global. Componentes.

Desde una perspectiva amplia, la capa de aplicación (APPLICATION) ofrece recomendaciones. Según sea la naturaleza de los datos –académicos o médicos- o el origen de los mismos –Moodle o Google Health respectivamente- dividimos las funcionalidades en dos: la primera, llevar un seguimiento del paciente de los datos obtenidos de Google Health (Diabetic Monitoring Recommender) para evaluarlos y recomendar sobre los mismos; la segunda realizar las recomendaciones acerca de la información suministrada por el curso de formación implantado en Moodle (e-Learning Recommender).

La capa de aplicación se sirve de los diferentes conectores implementados en la capa de gestión de bases de datos (DB MANAGER) que nos permiten acceder

a la información externa a nuestro sistema. En el caso del GHConnector, integramos la API desarrollada por la Parte A del proyecto y utilizamos sus servicios para acceder a los datos del historial médico del paciente en Google Health. El jColibriConnector nos permite acceder a la base de casos de recomendaciones previas almacenadas y a partir de los datos evaluados del historial (RecomendadorSEG) y del formulario inicial (RecomendadorINI) realizará recomendaciones sobre el seguimiento del paciente –los detalles sobre cada uno de los recomendadores se explicarán con más detalle en el punto 4.1.2.4.-. A través del MoodleConnector, accederemos a la información del curso y realizaremos recomendaciones mecánicas sobre su formación, pero no a través de la herramienta jColibrí como en las recomendaciones anteriores.

NOTA: Dos de las tres bases de datos externas –excluyendo la de Google Health- las almacenamos de manera independiente en un servidor virtual que nos ha creado para tal efecto durante esta fase experimental del proyecto el Departamento de Arquitectura de Computadores y Automática –DACYA- de la Facultad de Informática. Nos han proporcionado tanto el acceso remoto a la máquina con permisos de administrador como acceso mediante protocolo http a las bases de datos a través de la herramienta phpMyAdmin.

Tras esta breve exposición del sistema global, a continuación procederemos a la descripción detallada de la Parte B, que es la que nos compete.

4.1.2. Arquitectura Parte B

La estructura de la aplicación está organizada en diferentes paquetes, agrupando las clases según su funcionalidad.

Como observamos en la Figura 14, contamos con cinco paquetes. El paquete *GUI* es el paquete dedicado a la interfaz, en él se definen las clases que permitirán crear e interactuar con la aplicación. En el paquete *DB*, se definen los conectores utilizados para acceder a bases de datos externas. El paquete *Data* es para definir la estructura de datos de un usuario. Por último, los paquetes *RecomendadorINI* y *RecomendadorSEG*, agrupan las clases que implementan el sistema recomendador basado en jColibri.

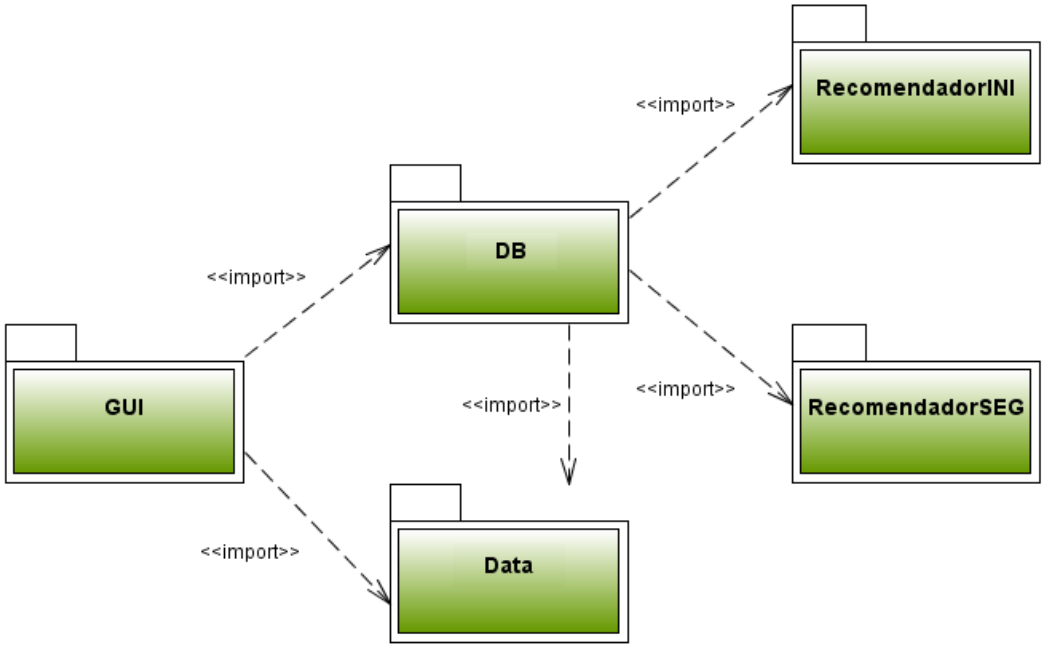

Figura 14. Diagrama de dependencias entre paquetes de GlucoAprende.

A continuación describimos cada uno de los paquetes con las clases que agrupan cada uno de ellos.

4.1.2.1. Paquete GUI

El paquete GUI (ver Figura 15) a su vez está dividido en otros tres paquetes, y mediante la interacción de todos ellos obtenemos el aspecto visual del sistema. Los distintos paquetes son:

a) Interfaces
b) Auxiliares
c) Images

A continuación se enumeran las clases, frames y paneles contenidos en este paquete:

a) Interfaces:

Está compuesto por la ventana principal de la aplicación (*FramePrincipal*) y todos los paneles que se irán cargando sobre la ventana, a medida que el usuario interaccione con los distintas opciones de la aplicación. *FramePrincipal* es el contenedor para insertar cada

panel y también contiene los menús principales que nos permitirán interactuar con la aplicación. Los distintos paneles son:

i. *PanelInicio:* es el panel que aparece al iniciar la aplicación. Es el home, al que podremos volver en cualquier momento.
ii. *PanelAltaPaciente*: contiene la interfaz para que un administrador pueda dar de alta a un médico nuevo.
iii. *PanelAltaPaciente*: contiene la interfaz para que un médico pueda dar de alta a un paciente nuevo.
iv. *PanelAsistente*: un panel que guiará al usuario paciente por la aplicación.
v. *PanelFORM:* aquí encontramos un formulario inicial para darnos de alta en el curso online.
vi. *PanelSEG:* muestra en una tabla los distintos mensajes que el sistema recomendador ha mandado al usuario. Tiene el funcionamiento de una bandeja de correo entrante.
vii. *PanelMoodle:* nos proporciona un link que nos redirige externamente a través del explorador a la página de acceso al curso.
viii. *PanelAñadirCasoFORM*: panel para que un médico pueda añadir un nuevo caso relacionado con el formulario inicial a la base de casos utilizada por el recomendador.
ix. *PanelAñadirCasoSEG*: panel para que un médico pueda añadir un nuevo caso relacionado los datos del seguimiento médico a la base de casos utilizada por el recomendador.
x. *PanelNuevaRec:* panel para que un médico pueda añadir una nueva recomendación y almacenarla en la base de datos.

b) Auxiliares:

Dispone de cuatro clases que ayudarán a la hora de dar formato tanto a la ventana principal como a los distintos paneles, tablas…. Las clases *MiRender* y *MiModelo* nos permiten crear la tabla, con el formato que queremos, donde se mostrarán los distintos mensajes del usuario. La clase *PanelFondo* da formato y proporciona la imagen que hemos elegido como fondo a cada uno de los paneles. La clase *Elem* permite mantener una correlación entre cada uno de los checkbox que se pueden ver en la pantalla con el id de la recomendación asociada a la selección de cada uno de ellos.

c) Images:

Este subpaquete (carpeta) contiene todas las imágenes que hemos utilizado para crear el aspecto visual de la aplicación: botones, iconos, links…

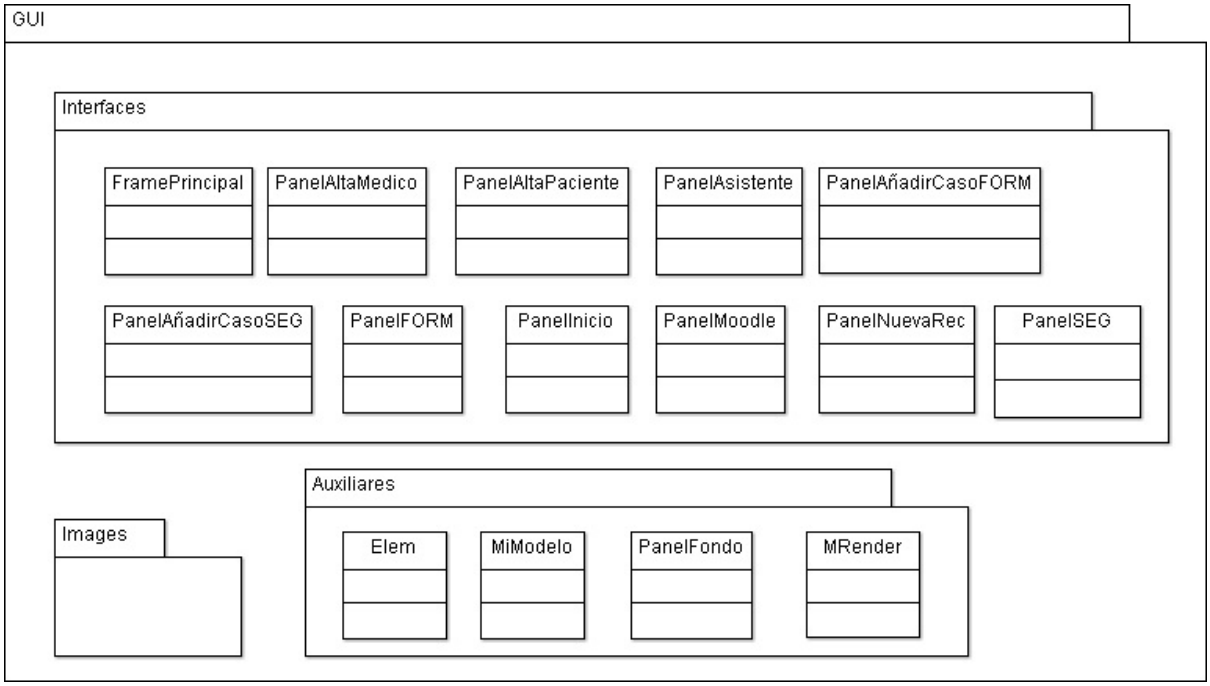

Figura 15. Paquete GUI

4.1.2.2. Paquete DB

En el paquete DB implementamos todo lo relacionado con las conexiones a las bases de datos externas, para la obtención y actualización de datos necesarios para la correcta ejecución de la aplicación por parte de los usuarios.

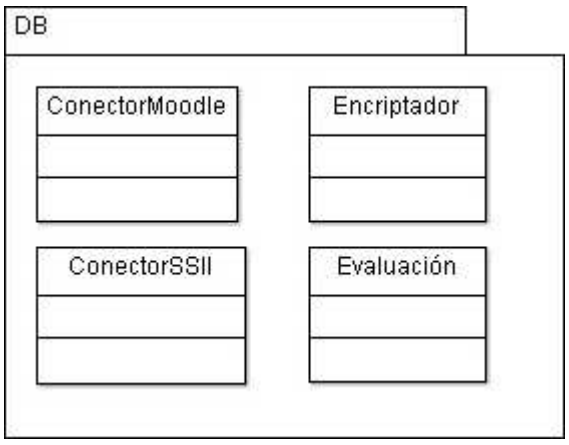

Figura 16. Paquete DB

Pasamos a la descripción de las cuatro clases:

a) *ConectorMoodle:* mediante esta clase obtenemos un objeto con el que poder conectarnos a la base de datos del curso online desarrollado bajo la plataforma Moodle. Con los métodos implementados en su interior podemos realizar escrituras en la BD para dar de alta a los usuarios en el curso desde la misma aplicación. Y también obtener los resultados de los tests realizados por los usuarios para realizar una revisión y que el sistema recomendador ofrezca sus soluciones.

b) *ConectorSSII:* con esta otra, cuya base, es prácticamente análoga a la anterior, podemos conectarnos a la BD "*ssii*" que hemos creado para almacenar la información relativa de cada uno de los usuarios: datos personales, historial de mensajes, control de evaluaciones...Podremos insertar nuevos usuarios en la BD para darlos de alta en la aplicación, así como actualizar ciertos campos y recoger información necesaria para mostrarla en la aplicación.

c) *Evaluación:* en esta clase se realiza la captación y procesamiento de datos tanto de la BD de Moodle (tests) como de la BD de Google Health. Una vez recogidos los datos médicos de GoogleHealth se creará un caso y se procederá a llamar al sistema recomendador para que a tenor de los valores de los distintos parámetros del caso nos ofrezca una solución, en forma de mensaje de recomendación. Lo mismo pasa con los resultados recogidos del test del curso online.

d) *Encriptador:* es una clase auxiliar, que nos ofrece un método para poder encriptar las contraseñas de acceso a la aplicación de los usuarios. En un principio nos vimos obligados a utilizarlo porque para dar de alta a un nuevo usuario en Moodle se necesita encriptar la contraseña con el formato "MD5". Esta clase nos posibilita la encriptación de la contraseña con este formato, y así aprovechamos también para encriptar la contraseña de acceso a la aplicación con ese mismo formato, y así dotar al sistema (a los usuarios en concreto) de seguridad.

4.1.2.3. Paquete Data

Este paquete está dedicado a la implementación de las estructuras de datos utilizadas para el almacenamiento de datos de los usuarios, recogidos desde las bases de datos.

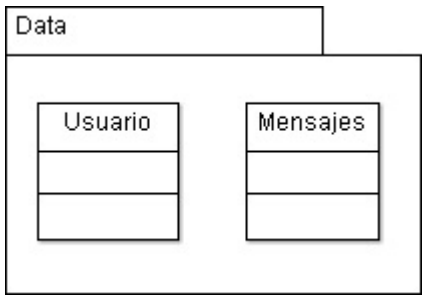

Figura 17. Paquete Data

Las clases que integran este paquete son:

a) *Usuario:* con esta clase creamos una estructura de datos para almacenar toda la información relativa a cada usuario que hayamos obtenido desde la BD. Contiene todos los campos de información tanto personal como para realizar el seguimiento y las evaluaciones, que son almacenados por cada usuario en la BD.

b) *Mensajes:* esta otra clase también especifica una estructura de datos en la cual se va a almacenar el historial de mensajes del usuario que se haya obtenido previamente de la BD. Con esta estructura se facilita la posterior visualización de los mensajes en la aplicación.

4.1.2.4. Paquete RecomendadorINI y RecomendadorSEG

Debido a la similitud entre ambos paquetes, se explicarán conjuntamente.
En estos paquetes se encuentran ubicadas las distintas clases y ficheros de configuración que utilizar el recomendador para la encuesta inicial que se realiza al paciente ante de su inscripción en el curso de Moodle.

Para la realización del recomendador se ha utilizado la herramienta jColibri, por lo que las clases que aquí se indican implementan distintas interfaces del Framework proporcionado por la herramienta.

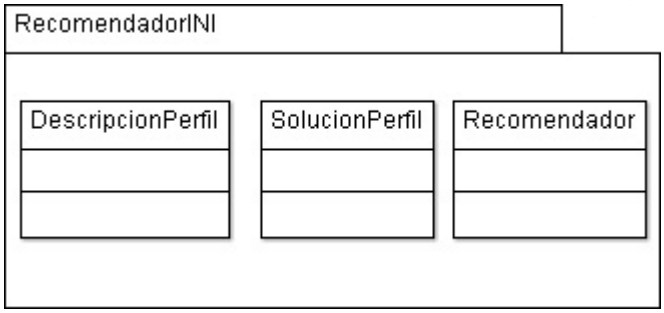

Figura 18. Paquete RecomendadorINI

A continuación se explicará en detalle cada uno de los componentes del paquete.

a) *Recomendador*: se trata de la clase principal. Implementa la interface StandardCBRApplication. Contiene un atributo de tipo connector y otro de tipo caseBase. Además contiene los siguientes métodos, los cuáles gestionan todo el sistema de recomendación:

   Configure(): inicializa el recomendador y la base de casos, además de configurar la conexión mediante distintos ficheros .xml's con la base de datos en la cual está almacenada la base de casos.

   Precycle(): una vez hemos configurado la conexión, mediante este método realizaremos la carga en memoria de la base de casos. El objeto caseBase almacenará toda esta información. Esta lectura caso de la base de datos está gestionada por Hibernate, realizando una serie de mapeos que relacionan los distintos campos de las tablas en base de datos con los atributos de las descripciones y soluciones de cada caso.

   Cycle(): recibe como parámetro la un objeto de tipo descripción. Realiza una búsqueda de entre la base de casos de caso que tenga una descripción con mayor similitud a la pasada por parámetro. Esta similitud es configurable a través de varios tipos de funciones y algoritmos. Como resultado se devuelven 1 o más casos.

   Postcycle(): se encarga de cerrar la conexión y de devolver al recomendador al estado inicial.

b) *DescripcionPerfil*: se trata de la clase que contiene los Atributos que conforman la descripción de los casos que se tratan en la aplicación. Estos atributos son tanto de tipo booleano como enteros.

c) *SoluciónPerfil*: contiene los atributos que conforman la solución de cada caso.

d) *Databaseconfig.xml*: es un archivo de configuración que a modo de índice, proporciona a la aplicación la ubicación del resto de ficheros de configuración.

e) *Hibérnate.cfg.xml*: archivo que sirve para configurar la conexión con la base de datos del servidor proporcionado. Como se ha mencionado anteriormente, esta conexión con la base de datos externa a la aplicación se realiza a través de Hibernate. En este fichero indicaremos la url del servidor así como el usuario y contraseña para acceder a él

f) *SolucionPerfil.hbm.xml y DescripciónPerfil.hbm.xml*: indican los mapeos existentes entre los campos que representan la información de un caso en la base de datos con los atributos de los objetos en Java. Se debe ir indicando campo a campo, con que atributo de la clase descripción o solución se corresponde.

4.1.3. Base de datos

4.1.3.1. Base de datos propia: SSII

La creación de una base de datos para nuestra aplicación viene de la necesidad de guardar cierta información de los usuarios para que el sistema pudiese operar correctamente.

Inicialmente, lo primordial para toda aplicación con la que interactúa un grupo de usuarios, es almacenar un perfil con cierta información personal sobre cada uno de ellos. De ahí que diseñáramos la tabla *usuarios*, con atributos como nombre, apellidos, userid, contraseña, médico, e-mail…

Para la implementación de las soluciones del sistema recomendador decidimos que éstas fueran en forma de mensaje de correo interno indicando las

recomendaciones necesarias. Así diseñamos la tabla *mensajes_usuario* dónde almacenar un historial de las recomendaciones asignadas a cada usuario. Y a su vez decidimos diseñar una tabla *recomendaciones,* donde almacenar todas las posibles soluciones del sistema recomendador.

El paciente diabético es principalmente sobre quien se centra la aplicación pues es al que está dirigido el sistema recomendador. Por ello, mientras implementábamos la aplicación y surgía la necesidad de controlar las glucemias, los análisis y los tests que ya se habían evaluado tuvimos que diseñar la tabla *evaluacion_usuarios.* Esta tabla contiene las fechas de los últimos datos evaluados, para no volver a evaluar los mismos datos.

Y por último, tras el estudio del funcionamiento de la herramienta jColibri, diseñamos las dos tablas más importantes para que funcionara el sistema recomendador: *casos_seguimiento* y *casos_formacion.* Estas dos tablas componen la base de casos de la cual se nutre el recomendador para poder evaluar al paciente.

Así con todas estas necesidades de almacenar información creamos nuestra base de datos llamada "ssii". A continuación se exponen las distintas entidades con sus respectivos atributos. Para los atributos no tan obvios expondremos una breve descripción.

- **usuarios:** para almacenar un perfil a cada uno de los usuarios de la aplicación. Contiene los siguientes atributos:
  - nombre
  - apellidos
  - userid: identificador para entrar en la aplicación.
  - password: contraseña de acceso a la aplicación.
  - email
  - medico: userid del médico que realiza el seguimiento su historia clínica.
  - googleId: identificador interno de Google Health.
  - moodleId: identificador interno del curso online en Moodle.
  - altacurso: indicador de si está dado de alta en el curso.
  - rolcurso: identificador del rol que desempeña dentro del curso.

- **mensajes_usuario:** para almacenar el historial de mensajes que le ha enviado el sistema recomendador. Contiene los siguientes atributos:
  - userid: indica a qué usuario pertenece el mensaje.

  - mensajeid: valor autoincrement para registrar todos los mensajes.
  - recomendacionid: indica la recomendación, que se obtiene de la tabla *recomendaciones,* que muestra como mensaje.
  - fecha
  - nuevo: parámetro que indica si el mensaje es nuevo o no.

- **recomendaciones:** tabla donde se almacenan todas las posibles recomendaciones que puede ofrecer como solución el sistema recomendador. Contiene los atributos esenciales de un correo:
  - id: valor autoincrement para registrar todas las recomendaciones.
  - seccion: indica si la recomendación es de formación o de seguimiento.
  - asunto: contiene el objeto de la recomendación.
  - texto: contiene el texto de la recomendación.

- **evaluacion_usuarios:** donde se almacenan las fechas de los últimos análisis y evaluaciones realizadas. Contiene los siguientes atributos:
  - userid: indica a qué usuario pertenece la evaluación.
  - evaluacion: fecha en que el sistema recomendador realizó la última evaluación al paciente.
  - examen: fecha en que se evaluó el último test realizado en el curso.
  - cetonicos: fecha del último análisis de cuerpos cetónicos del paciente.
  - hba1c: fecha del último análisis HbA1c del paciente.
  - fondoOjos: fecha del último fondo de ojos realizado al paciente.
  - cuidadoPies: fecha de la última vez que se le recomendó el cuidado de los pies al paciente.
  - imc: fecha del último valor de IMC del paciente.

- **casos_formacion:** tabla en la que se almacenan los parámetros necesarios referentes al formulario inicial.

- **casos_seguimiento:** tabla en la que se almacenan los parámetros necesarios referentes al seguimiento médico.

Los atributos de estas dos últimas tablas, así cómo el diagrama Entidad-Relación de la base de datos los podemos ver en la Figura 19.

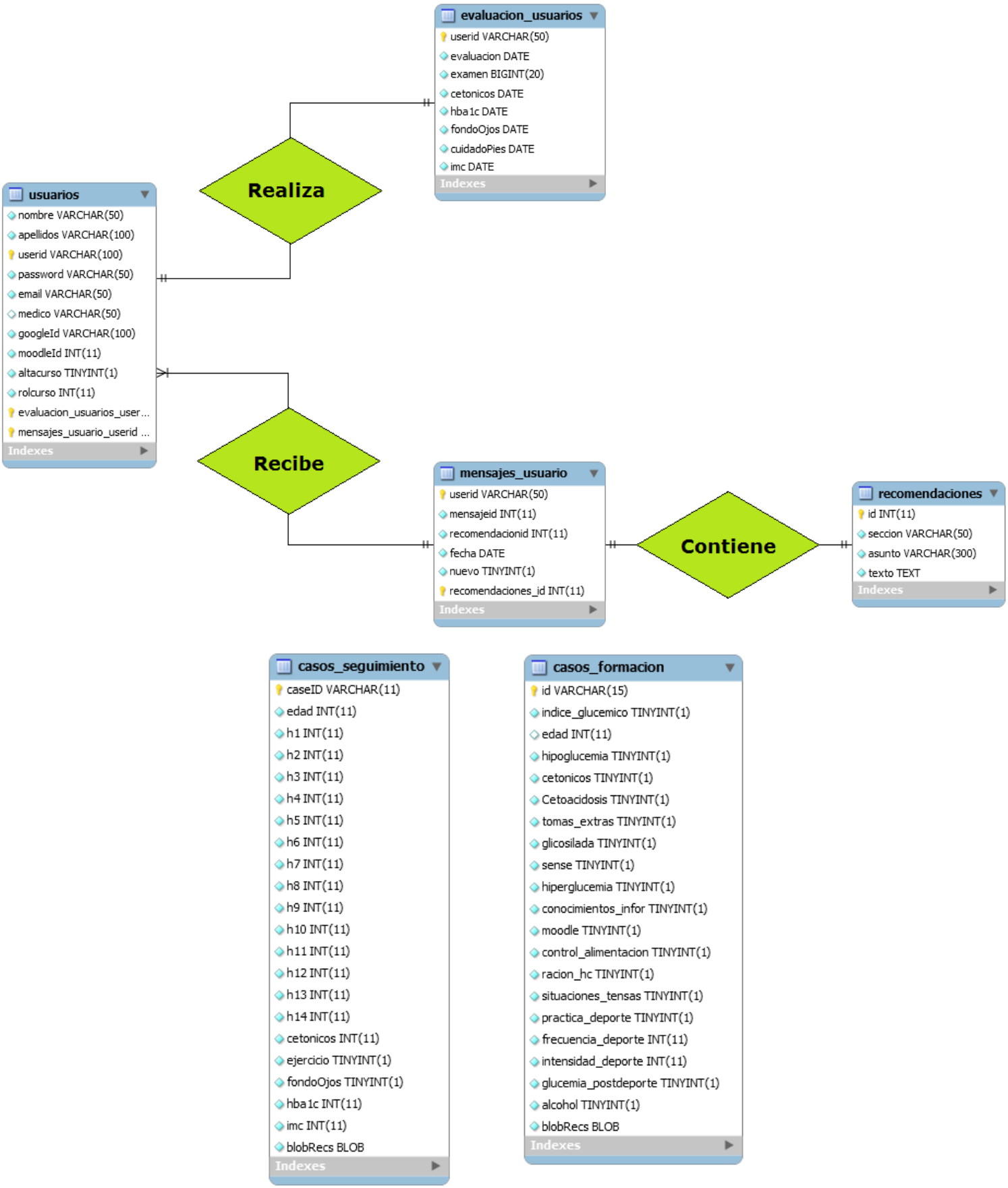

Figura 19. Diagrama ER de la base de datos SSII

4.1.3.2. Base de datos Moodle

Una de las ideas principales para evitar que el médico tuviera que dar de alta manualmente a cada paciente (a su vez dado de alta en la aplicación) en el curso online, y así facilitar su trabajo, fue la consideración de realizar el alta automática desde fuera del curso, cuando el paciente rellenara un formulario inicial para inscribirse en el curso.

Para realizar esta función de dar de alta a un paciente en el curso (Moodle) externamente desde la aplicación, y otras como asignar el rol del paciente en el curso y recoger los resultados de los tests realizados por el paciente tuvimos que estudiar las tablas sobre las que debíamos realizar escrituras y/o consultas, de la base de datos propia del curso.

Fue una tarea un poco ardua, pues esta base consta casi de 200 tablas. Por lo que tuvimos que analizar cada una de ellas para saber a cuáles afectaban los cambios necesarios para realizar las funciones anteriores. Por fortuna, no afectaban a muchas.

A continuación pasamos a enumerarlas:

- **mdl_user**: tabla que contiene todos el perfil de cada usuario con todos sus datos personales y otras opciones.
- **mdl_role_assignments**: tabla que contiene el rol de cada uno de los usuarios dentro del curso online.
- **mdl_quiz_attempts**: aquí se almacenan los intentos que ha realizado cada usuario con los tests del curso.
- **mdl_question**: almacena los enunciados de cada una de las preguntas y la categoría a la que pertenecen.
- **mdl_question_categories**: contiene el título de cada uno de los temas a los que están asociadas cada una de las preguntas.
- **mdl_question_states**: contiene el resultado obtenido, por los usuarios, de cada una de las preguntas del tests, de cada uno de los intentos realizados.

Las dos primeras tablas son necesarias para dar de alta a un usuario en el curso. Se escribirá sobre ellas desde la aplicación para insertar al usuario en la tabla de usuarios del curso y para asignarle el rol de estudiante, para que pueda navegar por el curso con los permisos de que disponga un estudiante.

Las últimas cuatro tablas son necesarias para la recogida de resultados de los tests realizados por el usuario estudiante (paciente). Obtendremos el último

intento de un test realizado por el usuario, el resultado de cada una de las preguntas, y el enunciado y el tema de cada una de las preguntas falladas. De esto último nos serviremos para elaborar la recomendación sobre estudiar los distintos temas en los que haya fallado el usuario.

En cuanto al diagrama y los campos de estas tablas, no se adjuntan ni de forma escrita ni en forma de gráfico pues alguna de ellas contiene un número de atributos elevado, que supondría una gran cantidad de espacio dentro de la memoria, que consideramos irrelevante exponer.

4.2. Funcionamiento y modo de uso del sistema

4.2.1. Perfiles de usuario

Los usuarios del sistema o usuarios cliente son personas que se conectan al sistema para hacer uso de los servicios que este les proporciona. Dentro de los usuarios del sistema podemos distinguir diferentes perfiles o niveles de usuario.

En GlucoAprende distinguimos tres perfiles de usuario, los cuales podrán llevar a cabo diferentes tareas y funciones:

- Paciente
- Médico
- Administrador

Para que un *Paciente* pueda acceder a la aplicación, primero, tiene que haber sido creado dentro de la aplicación por un *Médico*, y a su vez el *Médico* será creado por un *Administrador*. Una vez que ya existen los usuarios, estos tendrán que insertar su identificador de usuario y su contraseña en la pantalla de inicio para poder trabajar con GlucoAprende.

No todos los usuarios tienen acceso a todas las opciones de la aplicación, depende del perfil de usuario. Por lo tanto las opciones de menú que se activan en los distintos perfiles de usuario son distintas.

Veamos cuáles son las opciones y operaciones que pueden realizar cada uno de los tres tipos de usuario.

4.2.1.1. Paciente

El usuario *Paciente* tendrá acceso al menú denominado de la misma manera. Allí encontrará dos opciones:

- Formación
- Mensajes Recibidos

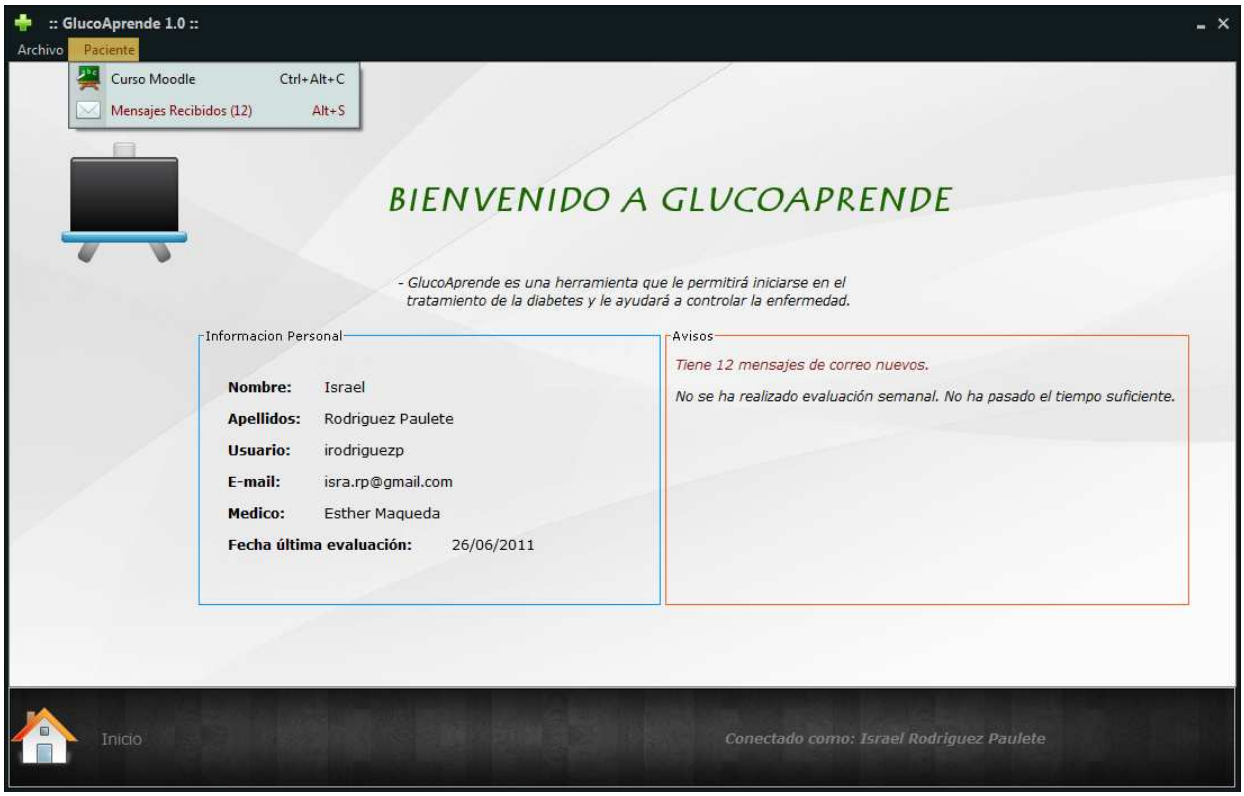

Figura 20. Pantalla principal del usuario Paciente.

Formación

El *Paciente*, desde su vista, tendrá la posibilidad de acceder al curso de formación online, al que será relanzado desde la aplicación.

Mensajes Recibidos

Con esta opción podrá consultar si tiene mensajes (recomendaciones del sistema) nuevos, así como su historial de mensajes, todo ello a través de la implementación de una bandeja de entrada de correo.

4.2.1.2. Medico

El usuario *Medico* tendrá acceso al menú denominado de la misma manera. Allí encontrará cuatro opciones:

- Alta Paciente
- Nuevo Caso Formación
- Nuevo Caso Seguimiento
- Nueva Recomendación

Figura 21. Pantalla principal del usuario Medico.

Alta Paciente

La primera de las opciones es la que habilita al *Medico* para poder dar de alta a un usuario *Paciente* en la aplicación. Es la única figura que tiene potestad para crear e insertar nuevos usuarios *Paciente* en la aplicación.

Nuevo Caso Formación

Esta opción permite crear nuevos casos (relacionados con el formulario inicial de formación) que serán agregados a la base de casos de la aplicación.

Nuevo Caso Seguimiento

Al igual que la anterior, permite la creación de nuevos casos (relacionados con los datos de seguimiento de GoogleHealth) que serán agregados a la base de casos de la aplicación.

Nueva Recomendación

Tendrá la posibilidad de crear nuevos mensajes (recomendaciones) que podrán ser asignados como solución a alguno de los nuevos casos creados anteriormente. Estos mensajes, se almacenarán en la base de datos.

4.2.1.3. Administrador

El usuario *Administrador,* es una figura más externa al sistema. La mayoría de las funciones que desempeña las realizará fuera de la aplicación. Entre ellas cabe destacar:

- Gestionar los recursos del servidor Web.
- Acceso directo a la base de datos y sus tablas.
- Administrar el curso de formación online.
- Corregir errores del sistema.

Figura 22. Pantalla principal del usuario Administrador

En cuanto a la aplicación tendrá acceso al menú denominado de la misma manera. Allí encontrará una sola opción:

- Alta Médico (ver Figura 22)

.

Alta Medico

Esta opción es la que habilita al *Administrador* para poder dar de alta a un usuario *Medico* en la aplicación. Es la única figura que tiene potestad para crear e insertar nuevos usuarios *Medico* en la aplicación.

4.2.2. Casos de uso

Los casos de uso que se explicarán representan una visión global de las funcionalidades que proporciona el sistema. Cada caso de uso es realizado por uno o varios actores, los cuáles son los distintos perfiles de usuario explicados previamente.

En la siguiente imagen podemos observar de manera resumida los principales casos de uso de la aplicación, así como las relaciones entre ellos y con los actores/usuarios.

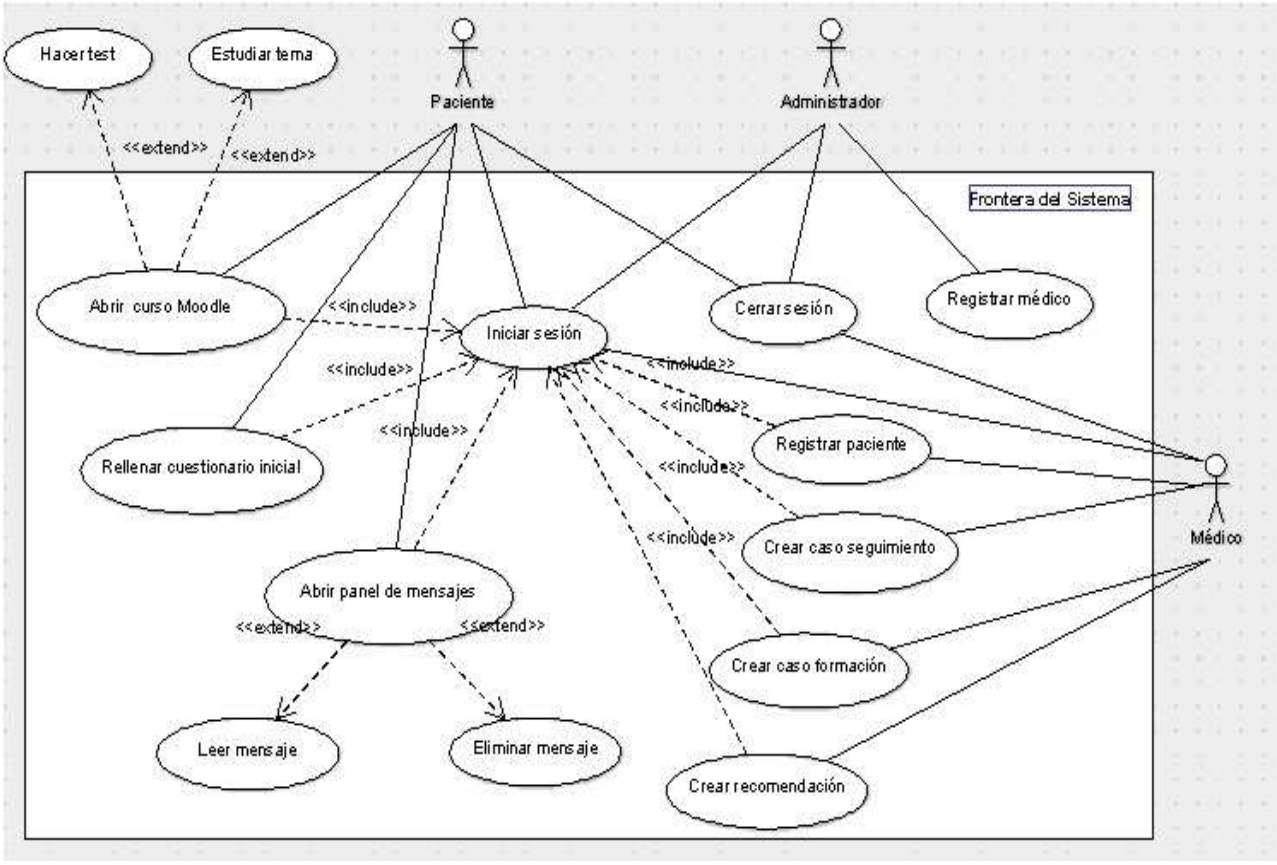

Figura 23. Diagrama de casos de uso.

A continuación se verá al detalle cada uno de ellos:

CU-01	REGISTRAR PACIENTE	
Objetivos asociados	Gestión de usuarios	
Actor	Médico	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un médico quiera registrar un nuevo paciente.	
Precondición	El usuario debe ser un médico previamente registrado en la aplicación. El usuario debe haber iniciado sesión y encontrarse en la pantalla inicial.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre el botón ***Médico*** → ***Alta Paciente*** de la barra de menú.
	2	Pulsar sobre el botón ***GOOGLE HEALTH*** para acceder a la Web.
	3	Introducir usuario y contraseña de nuestra cuenta.
	4	Pulsar sobre ***Add another profile .***
	5	Rellenar los datos del paciente
	6	Pulsar en ***Save changes*** y cerrar el explorador para volver al sistema.
	7	Rellenar los campos de Nombre, Apellidos, nombre de usuario, email y contraseña. Los campos nombre y apellidos deben coincidir con los introducidos en Google Health.
	8	Pulsar sobre ***Aceptar*** y el paciente será registrado en el sistema.
Postcondición	El sistema contiene un nuevo paciente, el cual tendrá que rellenar el formulario inicial para poder acceder al curso online.	
Excepciones	**Paso**	**Acción**
	8	Si al pulsar sobre aceptar recibimos algún mensaje de error, volveremos al punto 7 y verificaremos los campos.
Frecuencia esperada	Alta	
Comentarios	Al introducir el nombre y apellidos del nuevo usuario en Google Health, se deberá realizar siguiendo el siguiente patrón: 'apellidos,nombre'.	

CU-02	NUEVO MENSAJE DE RECOMENDACION	
Objetivos asociados	Mantenimiento base de casos.	
Actor	Médico	
Descripción	El sistema incorporará una nueva solución disponible para los casos almacenados en base de datos.	
Precondición	El usuario debe ser un Médico registrado en la aplicación, y debe haber iniciado sesión previamente.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre el botón ***Médico*** → ***Nueva Recomendación*** de la barra de menú.
	2	Rellenar los campos de asunto y texto del mensaje.
	3	Pulsar sobre ***Insertar***
Postcondición	La nueva solución queda almacenada en el sistema, preparada para ser asociado a los nuevos casos que queramos agregar.	
Frecuencia esperada	Media	
Comentarios	ninguno	

CU-03	INSERTAR CASO FORMACION	
Objetivos asociados	Mantenimiento base de casos.	
Actor	Médico	
Descripción	Un nuevo caso será almacenado para el recomendador de formación inicial.	
Precondición	El usuario debe ser un Médico registrado en la aplicación, y debe haber iniciado sesión previamente.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre el botón ***Médico*** → ***Nuev0 Caso Formación*** de la barra de menú.
	2	Responder y rellenar todas las preguntas.
	3	Pulsar sobre el botón ***Seleccionar Soluciones***
	4	Marcar los checkbox de las soluciones requeridas.
	5	Pulsar sobre el botón ***Confirmar***
	6	Pulsar sobre el botón ***Añadir Caso*** para finalizar el proceso.
Postcondición	El caso es añadido a la base de datos.	
Excepciones	**Paso**	**Acción**
Frecuencia esperada	Baja	
Comentarios	ninguno	

CU-04	INSERTAR CASO SEGUIMIENTO	
Objetivos asociados	Mantenimiento base de casos.	
Actor	Médico	
Descripción	Un nuevo caso será almacenado para el recomendador encargado de evaluar y realizar un seguimiento del paciente.	
Precondición	El usuario debe ser un Médico registrado en la aplicación, y debe haber iniciado sesión previamente.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre el botón ***Médico*** → ***Nuev0 Caso Seguimiento*** de la barra de menú.
	2	Responder y rellenar todas las preguntas.
	3	Pulsar sobre el botón ***Seleccionar Soluciones***
	4	Marcar los checkbox de las soluciones requeridas.
	5	Pulsar sobre el botón ***Confirmar***
	6	Pulsar sobre el botón ***Guardar Caso*** para finalizar el proceso.
Postcondición	El caso es añadido a la base de datos.	
Excepciones	**Paso**	**Acción**
Frecuencia esperada	Baja	
Comentarios	ninguno	

CU-05	INICIAR SESION	
Objetivos asociados	Acceder a las funcionalidades del sistema.	
Actor	Usuario de la aplicación (médico, paciente, administrador)	
Descripción	El sistema dará acceso a la aplicación al usuario en función de su perfil.	
Precondición	El usuario debe haber sido registrado previamente en la aplicación.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre ***Archivo -> Conectar***
	2	Rellenar correctamente los campos de usuario y contraseña
	3	Pulsar sobre el botón ***Aceptar***
	4	Pulsar sobre el botón ***Cancelar***
Postcondición	El usuario inicia la sesión correctamente en la aplicación.	
Excepciones	**Paso**	**Acción**
	3	Si el usuario o la contraseña introducidos no son correctos se debe volver al paso 2.
	4	Si pulsamos sobre ***Cancelar*** volveremos al paso 1.
Frecuencia esperada	Alta	

Comentarios	Ninguno

CU-06	**CERRAR SESION**	
Objetivos asociados	Cambio de usuario sin cerrar la aplicación.	
Actor	Usuario de la aplicación (médico, paciente, administrador)	
Descripción	El sistema desconectará al usuario.	
Precondición	El usuario debe haber iniciado sesión previamente.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre ***Archivo -> Desconectar***
Postcondición	El sistema queda preparado para un nuevo inicio de sesión.	
Excepciones	**Paso**	**Acción**
Frecuencia esperada	Alta	
Comentarios	Ninguno	

CU-07	**REGISTRAR MEDICO**	
Objetivos asociados	Gestión de usuarios	
Actor	Administrador	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el administrador quiera registrar un nuevo médico.	
Precondición	El usuario debe ser el administrador de la aplicación. El usuario debe haber iniciado sesión y encontrarse en la pantalla inicial.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre el botón ***Administrador*** → ***Alta Médico*** de la barra de menú.
	2	Rellenar los campos requeridos
	3	Pulsar sobre ***Aceptar***
Postcondición	El sistema contiene un nuevo médico registrado.	
Excepciones	**Paso**	**Acción**
	3	Si el nombre de usuario ya existe o las contraseñas no coinciden se lanzará un mensaje de error y volveremos al paso 2.
Frecuencia esperada	Baja	
Comentarios	El correo del médico y la contraseña del correo deben ser los de la cuenta de Google Health que deseamos asociar a la aplicación.	

CU-08	**RELLENAR CUESTIONARIO INICIAL**
Objetivos asociados	Gestión del curso Moodle
Actor	Paciente

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el paciente acceda al curso Moodle por primera vez.	
Precondición	El usuario debe ser un paciente previamente registrado y debe haber iniciado sesión. Además, debe ser la primera vez que trate de acceder al curso Moodle.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre el botón ***Paciente*** → ***Curso Moodle*** de la barra de menú.
	2	Contestar a las preguntas solicitadas.
	3	Pulsar sobre el botón ***Confirmar***
	4	Pulsar sobre el botón ***Si*** para aceptar los términos y concluir con el caso de uso.
Postcondición	El paciente es dado de alta en el curso Moodle, y se le asignan unos primeros temas para estudiar.	
Excepciones	**Paso**	**Acción**
	4	Si pulsamos sobre el botón ***No*** volveremos al paso 3
Frecuencia esperada	Media	
Comentarios	Es necesario aceptar los términos de uso de la aplicación ya que se manipularán datos personales del paciente.	

CU-09	**ESTUDIAR TEMA EN MOODLE**	
Objetivos asociados	Acceso contenidos formativos	
Actor	Paciente	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el paciente quiera acceder al curso online para el estudio de un tema.	
Precondición	El usuario debe ser un paciente previamente registrado en la aplicación y debe haber iniciado sesión. Además tiene que haber sido dado de alta en el curso Moodle.	
Secuencia Normal	Paso	Acción
	1	Pulsar sobre el botón ***Paciente*** → ***Curso Moodle*** de la barra de menú.
	2	Pulsar sobre el logo de Moodle
	3	Introducir nuestro usuario y contraseña para el curso (mismo que para la aplicación).
	4	Pulsar sobre el botón ***Entrar***
	5	Pulsar sobre cada uno de los títulos de cada tema para acceder al contenido pdf
Postcondición	El paciente tiene acceso al pdf del tema que quiere estudiar.	
Excepciones	**Paso**	**Acción**

Frecuencia esperada	Alta
Comentarios	

CU-10	**REALIZAR TEST EN MOODLE**	
Objetivos asociados	Gestión del curso Moodle	
Actor	Paciente	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el paciente quiera acceder al curso online para realizar un test.	
Precondición	El usuario debe ser un paciente previamente registrado en la aplicación y debe haber iniciado sesión. Además tiene que haber sido dado de alta en el curso Moodle.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre el botón ***Paciente*** → ***Curso Moodle*** de la barra de menú.
	2	Pulsar sobre el logo de Moodle
	3	Introducir nuestro usuario y contraseña para el curso (mismo que para la aplicación).
	4	Pulsar sobre el botón ***Entrar***
	5	Ir al tema 10 y pulsar sobre Test **Diabetes Mellitus tipo 1**
	6	Pulsar el botón **Comenzar**
Postcondición	El paciente realizar el test online obteniendo unos resultados.	
Excepciones	**Paso**	**Acción**
	4	Si pulsamos sobre el botón ***No*** volveremos al paso 3
Frecuencia esperada	Media	
Comentarios		

CU-11	**LEER MENSAJE**	
Objetivos asociados	Gestión panel de mensajes.	
Actor	Paciente	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el paciente quiera abrir leer una recomendación de su panel de mensajes.	
Precondición	El usuario debe ser un paciente previamente registrado en la aplicación y debe haber iniciado sesión.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre el botón ***Paciente*** → ***Mensajes recibidos*** de la barra de menú.
	2	Hacer doble click sobre el mensaje deseado.
Postcondición	El paciente lee una recomendación.	

Excepciones	Paso	Acción
Frecuencia esperada	Media	
Comentarios	ninguno	

CU-12	ELIMINAR MENSAJE	
Objetivos asociados	Gestión panel de mensajes.	
Actor	Paciente	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el paciente quiera eliminar una recomendación de su panel de mensajes.	
Precondición	El usuario debe ser un paciente previamente registrado en la aplicación y debe haber iniciado sesión.	
Secuencia Normal	**Paso**	**Acción**
	1	Pulsar sobre el botón ***Paciente*** → ***Mensajes Recibidos*** de la barra de menú.
	2	Seleccionar el mensaje a eliminar.
	1	Pulsar sobre el botón ***Eliminar.***
Postcondición	El mensaje del paciente queda eliminado.	
Excepciones	**Paso**	**Acción**
Frecuencia esperada	Alta	
Comentarios		

4.2.3. Aplicación

En este punto vamos a explicar el funcionamiento de la aplicación exponiendo cuál sería el flujo de trabajo tanto para el paciente como para el médico.

Login

El proceso para identificarse es el siguiente:

1. Acceder al menú Archivo -> Conectar.
2. Insertar los datos en los campos Usuario y Contraseña.
3. Pulsar sobre el botón Aceptar.

A continuación se puede ver la pantalla inicial de la aplicación. Al pulsar sobre el menú Conectar nos aparece una nueva ventana en la que debemos insertar los datos para acceder:

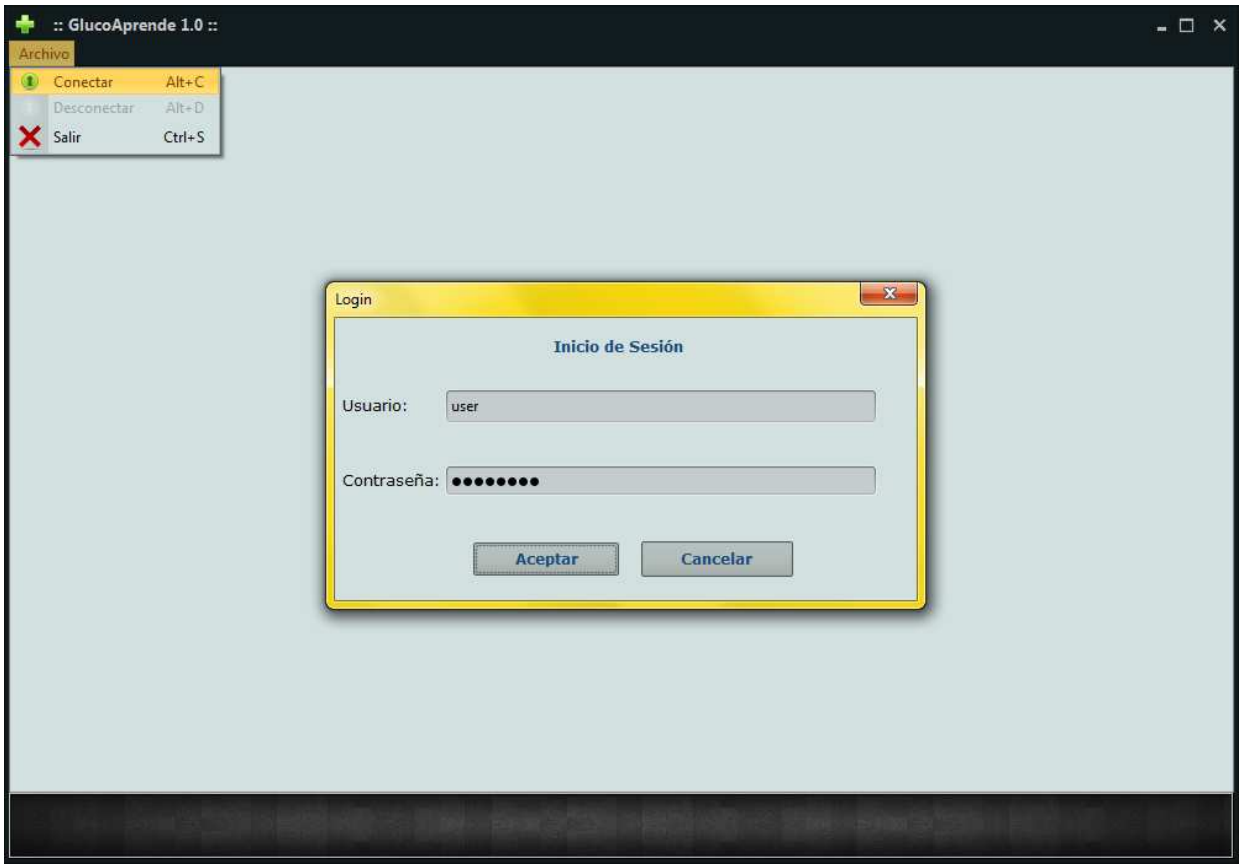

Figura 24. Pantalla de inicio de sesión.

Con los datos recogidos se realiza una búsqueda en la base de datos para comprobar que el usuario exista. Si existe, la contraseña es encriptada y se comprueba con la almacenada para ese usuario en la BD. Si los datos fueron ingresados de forma correcta entonces se mostrará el home del sistema con las opciones según el perfil que tenga definido dicho usuario en el sistema. En caso de haber introducido mal la contraseña o el nombre de usuario, se le notificará la causa del error y se le pedirá de nuevo el ingreso de datos (ver Figura 25).

Home

Una vez que ha ingresado los datos correspondientes y son validados correctamente, el usuario accede a la pantalla principal de la aplicación. En la barra superior de menús se listan las funcionalidades a las cuales tiene acceso el usuario según el perfil (visto en el punto 4.2.1).

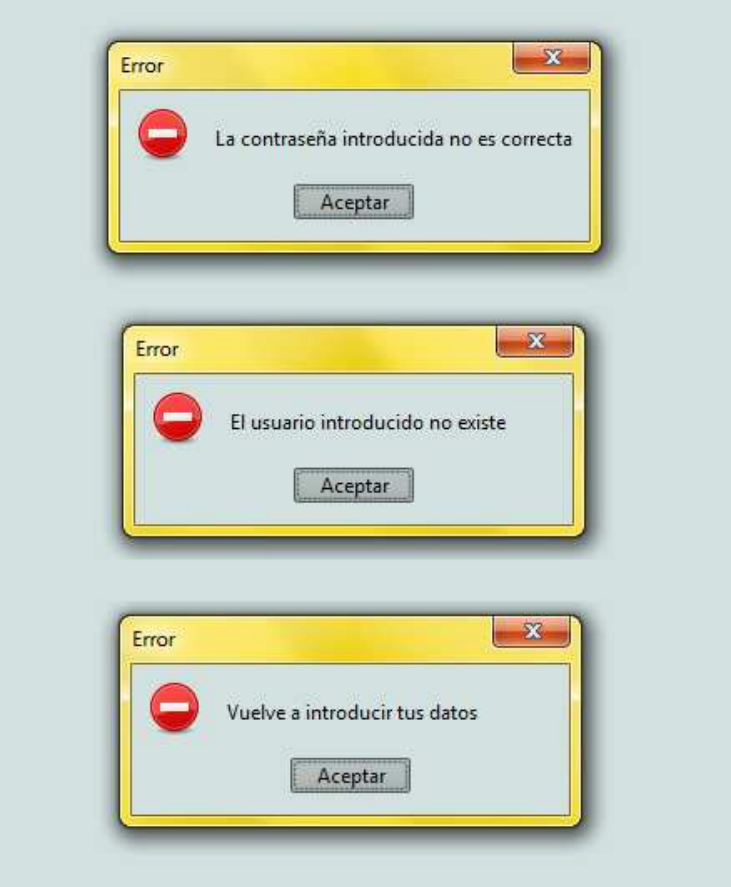

Figura 25. Mensajes de error al iniciar sesión.

4.2.3.1. Funcionamiento Paciente

Una vez se haya iniciado sesión y el usuario sea un paciente, se comenzará con la evaluación de éste. La evaluación está programada para que se ejecute automáticamente después del ingreso del paciente, con un intervalo de una semana entre evaluación y evaluación. Es decir, se ejecutará una evaluación semanal.

Para saber si se lleva a caso la evaluación el sistema comprobará la fecha de su última evaluación. En caso de que la evaluación anterior hubiese sido realizada hace una semana o más, se procederá a la conexión con Google Health para obtener las últimas analíticas y mediciones del paciente.

Aquí entra en juego el componente A del proyecto general. Mediante el uso de su conector y la implementación de la recogida de datos de Google Health, crearemos un objeto de tipo Paciente (una de las clases de la parte A), y tendremos acceso al historial del paciente con sus datos médicos.

La clase Evaluador se encarga de gestionar todo este proceso acorde a los siguientes pasos:

- Comprueba la fecha de la última evaluación del paciente. Si hace más de una semana continúa con el proceso de evaluación.

- Establece la conexión con Google Health para traer los datos del paciente. Estos datos se obtienen a través de una estructura de distintos ArrayLists que contienen tanto las mediciones de glucemias como los distintos resultados de análisis clasificados por día y hora.

- Una vez tenemos estos datos, buscamos las glucemias de la última semana así como los últimos análisis realizados recientemente (último mes) y de los cuáles no se haya generado recomendación alguna. Para saber si las analíticas han sido evluadas, nos conectaremos a la base de datos para recoger las fechas, de última evaluación, de cada analítica y las compararemos con las recogidas de Google Health. Así evitamos posibles reevaluaciones.

- Con los datos obtenidos de Google Health crearemos un nuevo caso. A continuación se utilizará el recomendador de seguimiento, el cual se conectará a la base de datos de la aplicación para obtener la base de casos. Después, buscará la descripción de caso con una mayor similitud al nuevo caso creado.

- Finalmente, se ofrecerá la solución del caso que obtenido más puntuación de similitud. Esta solución se insertará en la tabla que contiene los mensajes de recomendación del paciente. Al usuario le aparecerán los mensajes nuevos en el panel Seguimiento de la aplicación.

Otra evaluación que realiza el sistema es la del test del curso online. Igual que en el proceso anterior, comprobaremos que la fecha del último test realizado en el curso es distinta a la fecha en que evaluamos la última vez el test, y que se encuentra almacenada en la base de datos.

Si fuera mayor (indicador de que ha realizado de nuevo el test) se procederá a la recogida de los resultados del test. En función de estos resultados se le recomendará al paciente que estudie de nuevo las partes del temario falladas o que profundice en algunos puntos en concreto.

Una vez se haya completado todo el proceso de evaluación, el usuario podrá seguir interactuando con la aplicación.

La primera de las opciones de la barra de menú es Formación. A través de esta opción se podrá acceder al curso Moodle, siempre y cuando haya realizado el formulario inicial y haya sido dado de alta en el curso. Si por el contrario es la primera vez que se intenta acceder al curso aparecerá el siguiente formulario para darnos de alta.

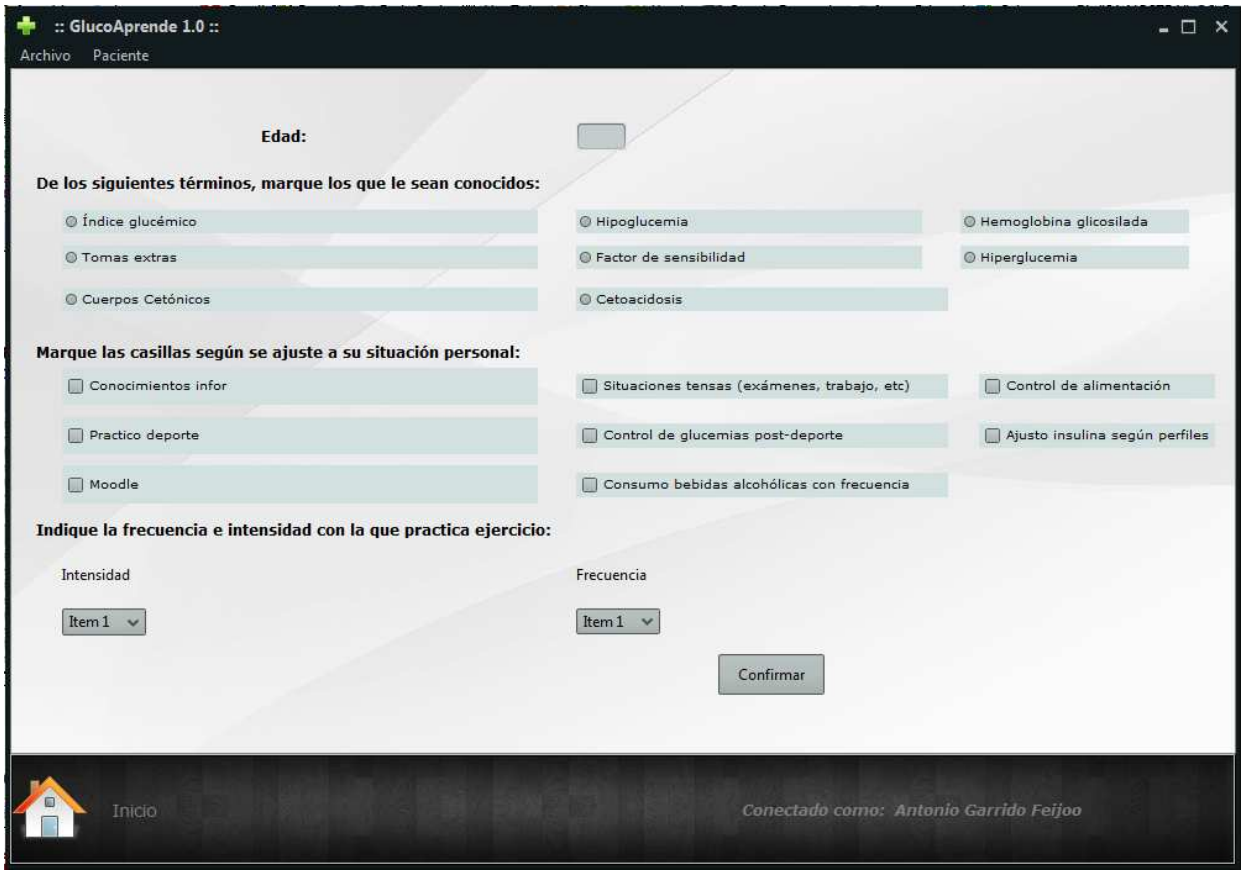

Figura 26. Panel Formulario Inicial.

Cuando se haya rellenado el formulario y se pulse en aceptar se deberán confirmar los términos de uso, ya que consentiremos la manipulación de datos personales para dar de alta en el curso.

Una vez aceptados los términos se lanzará el recomendador de formación inicial, el cuál, evaluará las opciones marcadas por el usuario sobre el conocimiento previo de la enfermedad y le propondrá una serie de recomendaciones iniciales en forma de mensaje. Estas recomendaciones orientarán al paciente dentro del curso para que comience a estudiar los temas

en los que tiene menos conocimientos. También generará un mensaje de bienvenida con algunas instrucciones para acceder al curso.

Por último se dará de alta al usuario en el curso online creado en Moodle. La aplicación accederá a la base de datos de moodle para realizar las escrituras necesarias en la tabla de usuarios y roles de usuarios, otorgándole el mismo usuario y contraseña que en la aplicación y el rol de estudiante. Una vez dado de alta, las próximas veces que acceda al menú Formación aparecerá el siguiente panel, desde el cuál, pulsando sobre el link de moodle, accederemos al curso en una ventana a parte a través del explorador.

Figura 27. Panel de acceso al curso.

La segunda de las opciones del menú es Seguimiento. Pulsando sobre esta opción se cargará el panel seguimiento. El paciente tendrá acceso a su historial de mensajes mediante este panel que implementa una bandeja de correo entrante.

Una vez dentro del panel podrá gestionar su historial de mensajes eliminándolos o consultando el contenido de los mismos, que contienen las

recomendaciones creadas por el sistema. En la siguiente imagen podemos ver el panel de mensajes.

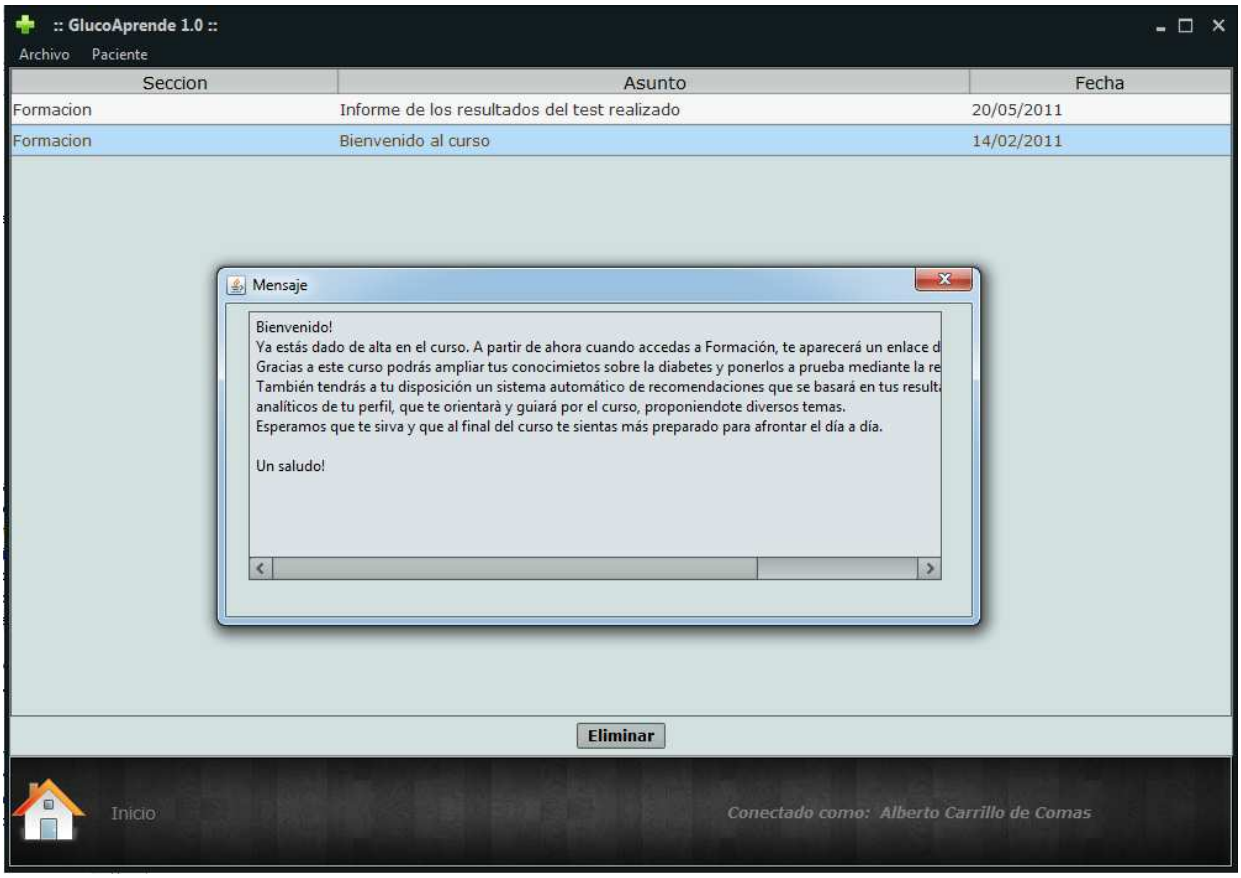

Figura 28. Panel Seguimiento.

En la tabla se podrán observar dos tipos de mensaje. Los leídos y los no leídos. Se diferencian en que los no leídos están resaltados con un fondo de color blanco mientras que los leídos tienen el fondo de color gris. Para acceder al contenido de un mensaje el usuario deberá pulsar dos veces sobre el mensaje. Se abrirá entonces un cuadro mostrando el texto. Si el mensaje fuese nuevo se actualizará su fondo a color gris.

Para eliminar un mensaje, se deberá seleccionar la fila de la tabla que contenga el mensaje a eliminar, y a continuación pulsar sobre el botón eliminar. Entonces se procederá al borrado del mensaje de la tabla mensajes_usuario de la base de datos y se actualizará la vista de la bandeja de entrada.

4.2.3.2. Funcionamiento Médico

Para el caso de que el usuario sea un médico no se lanzará el evaluador como si se hizo con el paciente, ya que el sistema recomendador va dirigido a éste último.

Podremos visualizar las distintas funcionalidades que se ofrecen al médico a través de la barra superior de menú.

La primera de la que hablaremos y una de las más importantes, por su funcionalidad, es la de dar de alta a un paciente:

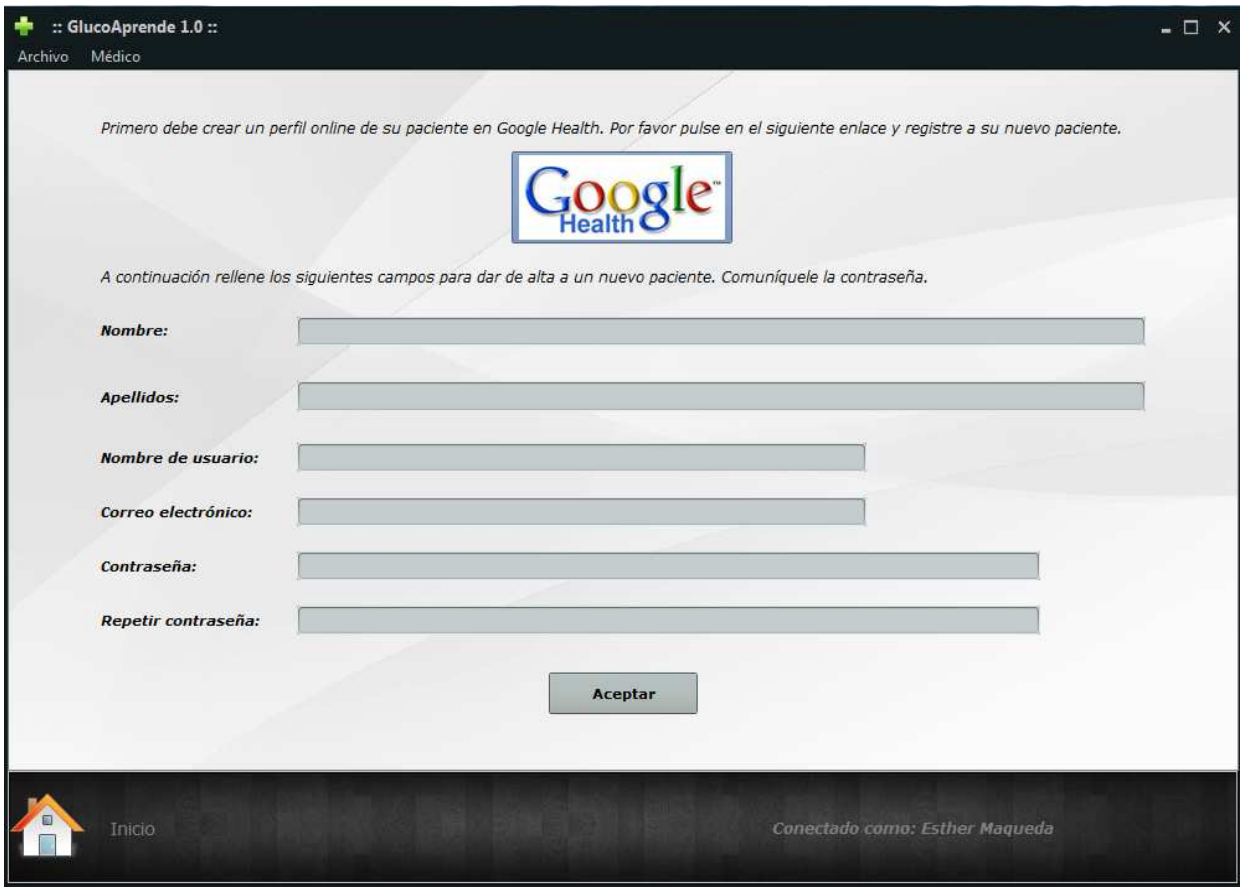

Figura 29. Panel para añadir paciente nuevo.

Como se indica en el formulario de alta, primero debemos añadir al paciente en la lista de pacientes de la cuenta del médico de Google Health. Para ello el médico pulsará el icono de Google Health y será redirigido en una ventana nueva a través del explorador. Después de haber accedido al servicio de Google con sus credenciales procederá a añadir a un paciente nuevo. Hay que destacar que el formato que debe seguir el campo nombre en el alta de

paciente en Google Health debe ser el siguiente: “apellidos,nombre” (sin comillas).

Después rellenaremos el resto de campos del formulario de la aplicación, poniendo especial atención al nombre y apellidos, los cuales deberán coincidir exactamente con los del usuario añadido Google Health. Importantes el nombre de usuario y la contraseña para acceder a la aplicación, para poder entregárselas al paciente.

Finalizaremos el proceso de alta confirmando la operación y aceptando los términos de privacidad que se indican. Al aceptar los términos, se procederá a insertar un nuevo usuario en la tabla de usuarios de la base de datos. A partir de ahora, el nuevo paciente tendrá acceso a la aplicación.

Otras de las opciones de que dispone un médico en la barra de menú son las de ayudar a enriquecer la base de casos introduciendo nuevos casos y generando nuevas recomendaciones.

Para ello utilizaremos los formularios de añadir caso seguimiento, añadir caso formación y añadir recomendación.

Empezaremos a explicar el panel en que se añaden nuevas recomendaciones. Como vemos en la imagen (Figura 30) se deben rellenar los campos asunto y texto, como si de un mensaje de correo electrónico se tratase. Cuando pulsemos sobre insertar, se añadirá una nueva entrada en la tabla *recomendaciones* representando esta nueva recomendación. Esta a su vez, podrá ser seleccionada a la hora de adjudicar una recomendación a un caso nuevo que cree el médico.

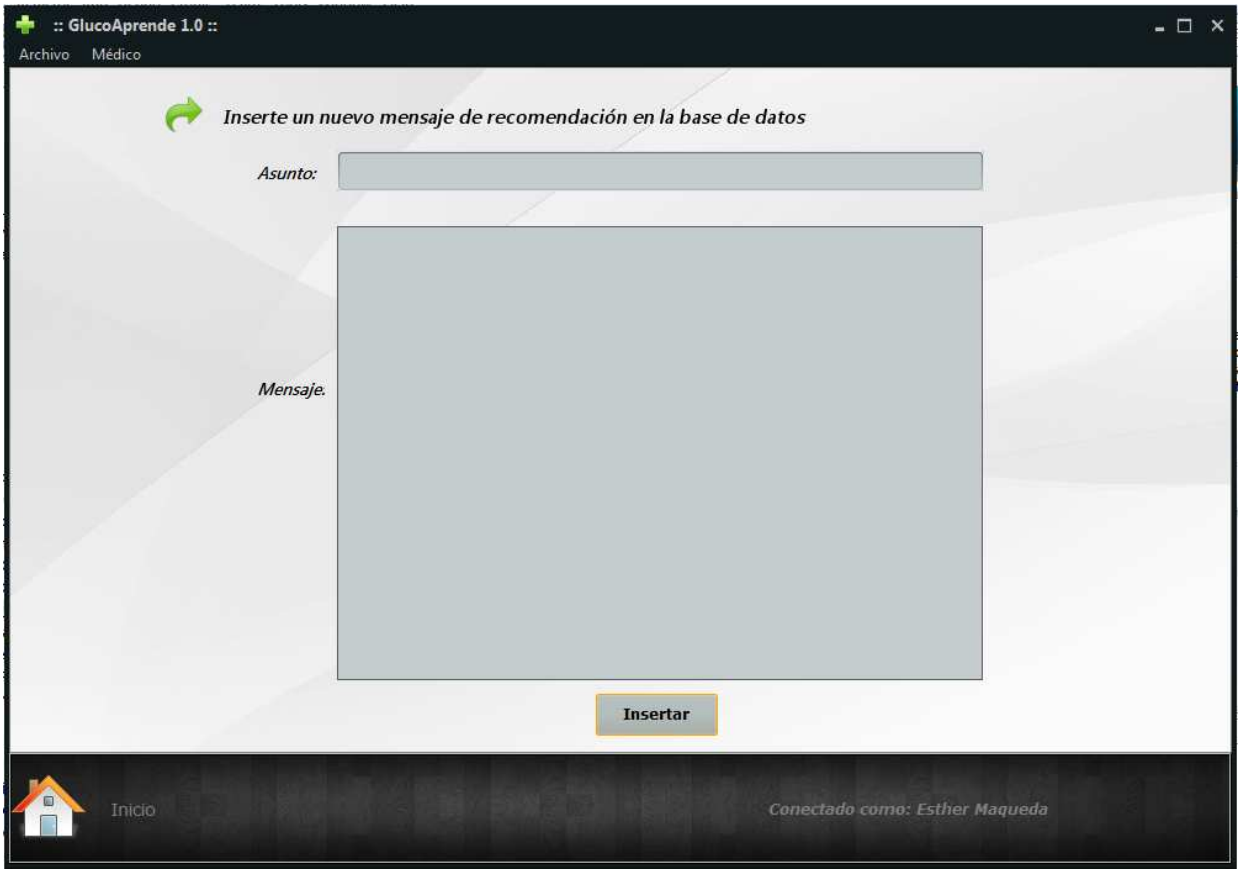

Figura 30. Panel Añadir Nueva Recomendación.

Para los paneles de añadir casos de formación o seguimiento procederemos de manera similar. Accederemos a través del menú a los paneles correspondientes. Una vez dentro de ellos debemos rellenar los diferentes parámetros para configurar la descripción del caso que vamos a añadir. En la Figura 31 se pueden observar ambos paneles.

Posteriormente debemos seleccionar las soluciones que queremos asociar a esta descripción. Entre estas soluciones se encontrarán las que había predefinidas en la base de datos y las que hayamos ido añadiendo desde el panel nueva recomendación (ver Figura 32).

Una vez seleccionemos la solución, pulsamos el botón guardar caso, y se añadirá una nueva entrada a la tabla que contiene la información de los casos en la base de datos. Es decir, en las tablas *casos_seguimiento* y *casos_formacion*.

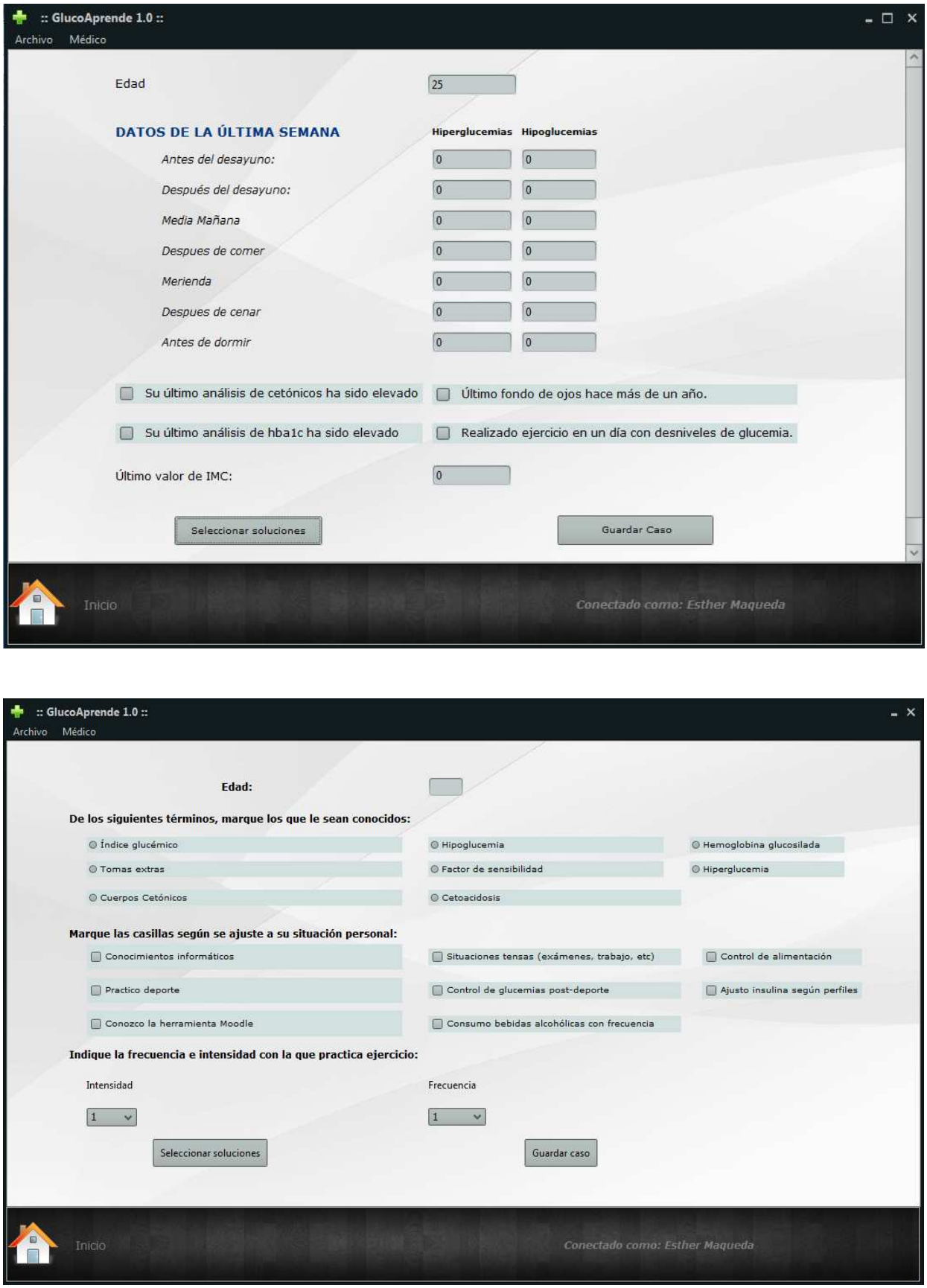

Figura 31. Paneles Añadir Casos.

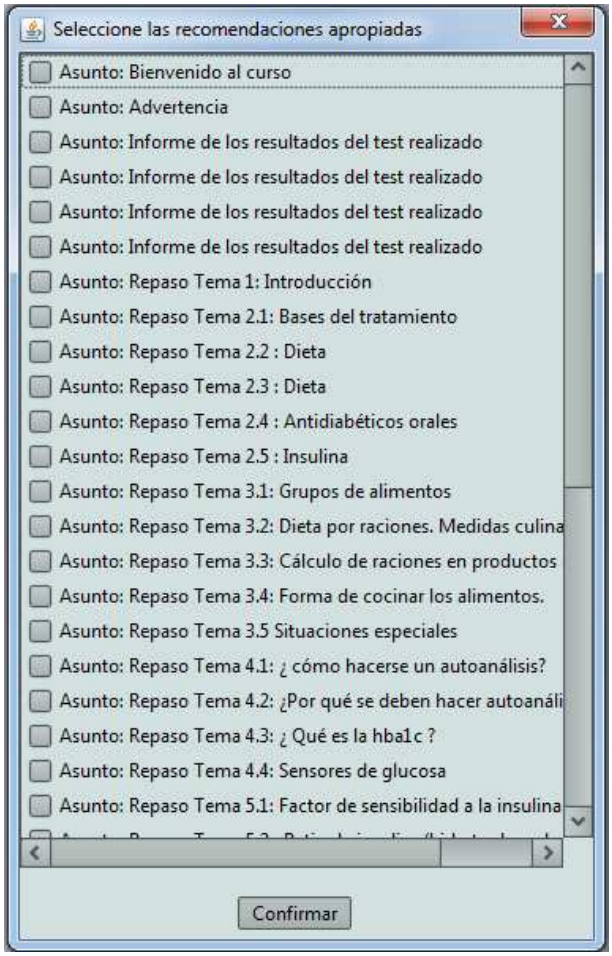

Figura 32. Selección de recomendaciones.

4.2.3.3. Funcionamiento Administrador

Como ya comentamos en el apartado 4.2 de los perfiles de usuario el administrador dispone solo de la función relacionada con la interacción de la aplicación. Esta es la de añadir un nuevo médico al sistema.

Para ello, se dispone de un formulario de alta en el cual se deberán introducir los campos mostrados en la Figura 33. Es importante, y condición necesaria, que el médico al que se va a dar de alta disponga de una cuenta de acceso a Google Health, pues el sistema está orientado a los pacientes que estén dados de alta en este servicio para poder recoger sus datos médicos y activar el recomendador evaluando los mismos. Si no, no tendría sentido.

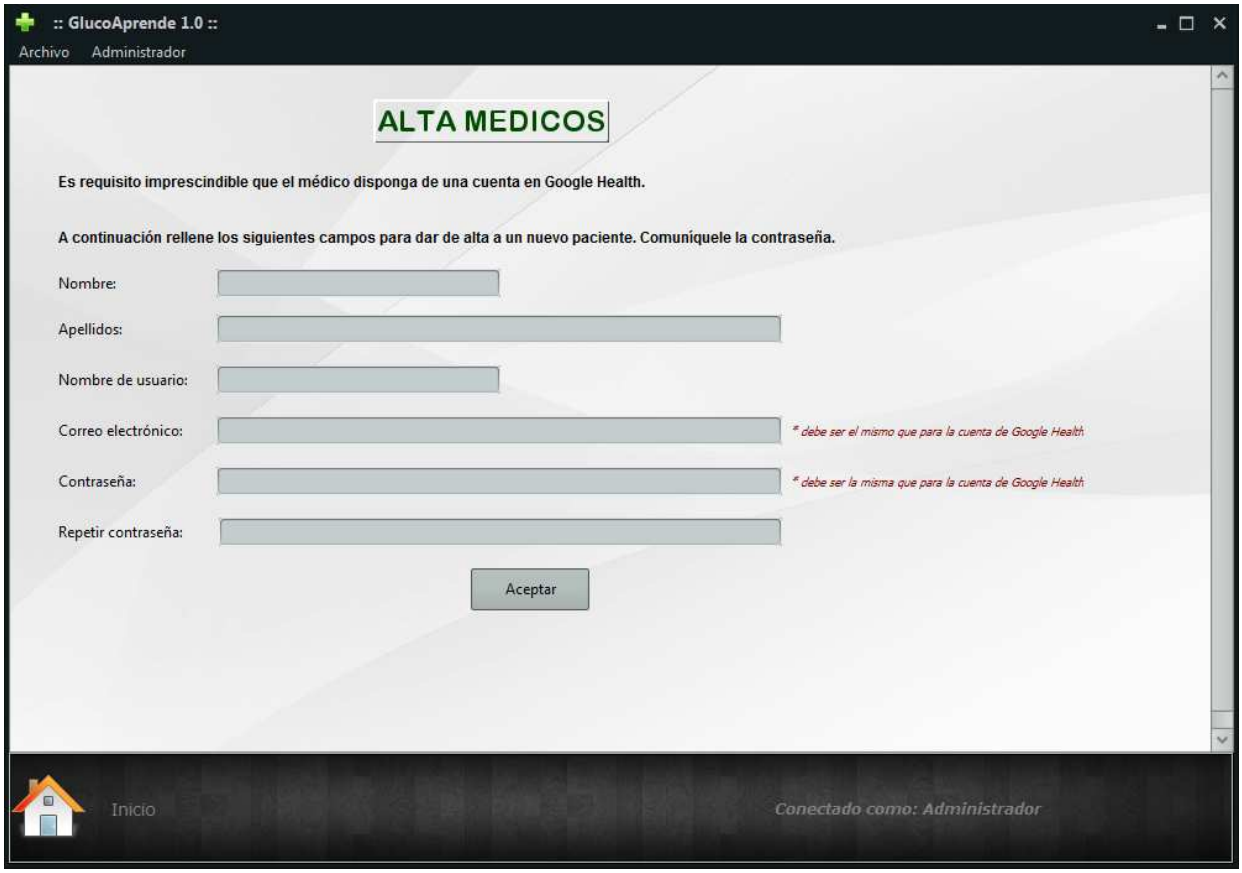

Figura 33. Panel Alta Médico.

4.3. Curso formación online: Proyecto Páncreas

4.3.1. Perfiles de usuario

Cuando el sistema da de alta a un usuario en el curso de Moodle, tiene en cuenta el perfil que tiene en la aplicación para asignarle el correspondiente en el curso. Así el perfil de paciente recibirá el rol de *estudiante*, y el perfil de médico el de *profesor*. El *administrador* principal del curso seguirá siendo el mismo que administra la aplicación y bases de datos.

Pasamos a ver los permisos y características más destacadas de cada uno de los perfiles, dentro del curso:

- **Estudiante**

  Los usuarios con este rol pueden acceder a los contenidos, realizar las actividades de que disponga el curso (quedando sus acciones registradas en el sistema, a disposición del profesor), modificar la información personal de su perfil, acceder a los foros para interactuar con otros usuarios y/o el profesor, y consultar las entradas del glosario.

- **Profesor**

  Un usuario con este rol puede modificar la estructura del curso, agregar y/o modificar los contenidos y las actividades, crear preguntas y cuestionarios, aumentar las definiciones del glosario, calificar y revisar las actividades (cuestionarios) realizados por los alumnos e inscribir alumnos nuevos en el curso.

- **Administrador**

  Un administrador en Moodle gestiona todo el sitio. Normalmente, el administrador supervisa la apariencia y la sensación que produce el sitio Web y que lo hacen único. Este rol ocupa el nivel más alto en Moodle, en lo que a privilegios de usuario se refiere y podrá gestionar los permisos y privilegios de cada uno de los roles anteriores.

4.3.2. Estructura y funcionamiento del curso

Como comentamos anteriormente, una de las funcionalidades del perfil paciente en la aplicación, es el acceso directo al curso online. Pulsando sobre el enlace que aparece en el Panel Moodle se lanzará el navegador web predeterminado redirigiendonos a la página inicial del curso http://pancreas.dacya.ucm.es/ (este proceso se puede realizar manualmente escribiendo la URL en la barra de direcciones del navegador). En la siguiente imagen (Figura 34) podemos observar la home del curso.

Figura 34. Home del curso.

Desde esta página podremos acceder a través del botón entrar que se encuentra en la parte superior derecha. Una vez hayamos introducido las credenciales y completado el proceso de login podremos pulsar sobre el enlace al curso *Educación para la Diabetes* y se mostrar la pantalla principal del curso en Moodle. Explicaremos a continuación

A continuación pasamos a explicar las diferentes secciones numeradas en las que está organizado el curso, y que podemos observar en la Figura 35.

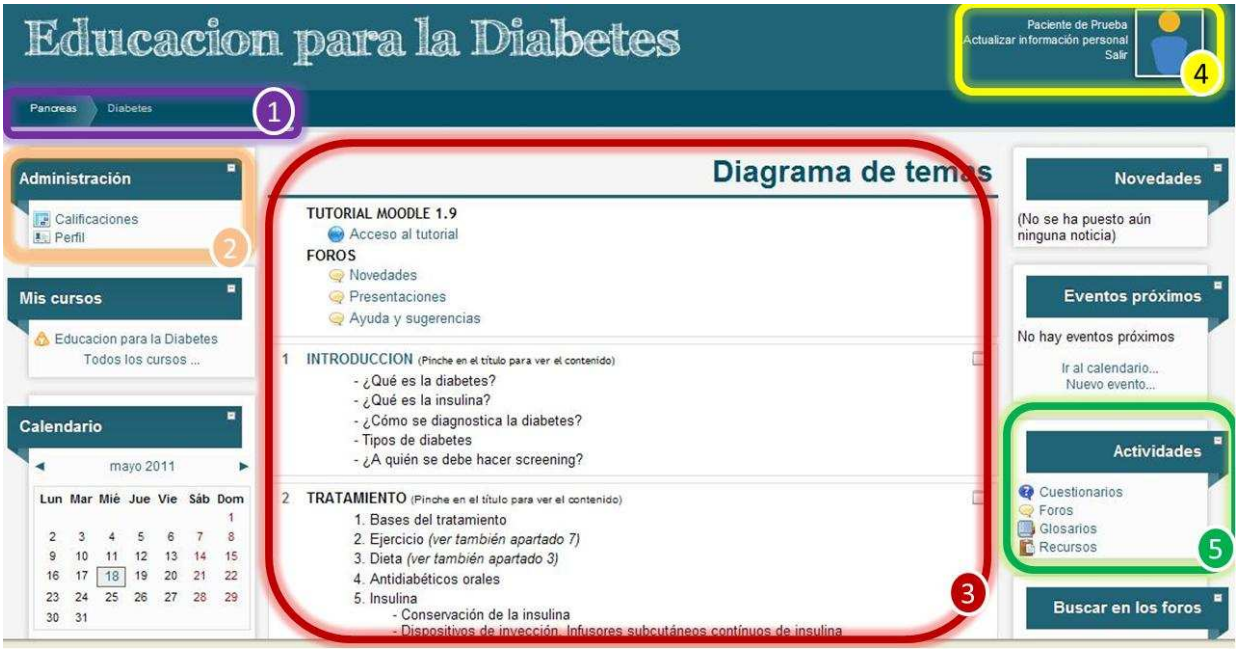

Figura 35. Secciones del curso.

La *sección 1*, es una barra de localización. El enlace situado más a la derecha nos indica en qué punto del curso nos encontramos, y la totalidad de enlaces de la barra es la ruta absoluta que hemos seguido por el mapa Web para acceder a dicho punto. Si pulsamos sobre cualquier punto intermedio nos llevará a la página indicada.

La *sección 2*, sirve para administrar el perfil de usuario. Si pulsamos sobre el enlace 'Perfil' accederemos al mismo para realizar tareas de consulta o edición de: la información de contacto, las entradas de mensajes y debates introducidos por el usuario, las entradas del blog personal y el informe de actividad del alumno en el curso.

La *sección 3* es la principal, localizada en el centro de la pantalla donde se muestran todos los recursos –foros, tutoriales, temario y contenido, glosario- subidos por el profesor del curso y visibles para el alumno. Para acceder al fichero de contenido correspondiente a cada tema bastará pulsar en el título del mismo y el enlace nos llevará al archivo .pdf con la información didáctica del tema. Debajo de cada título de tema y a modo informativo –no son enlaces-, aparece un breve índice de contenidos del tema.

En la *sección 4* se muestra la información de sesión. Aparece el nombre de usuario conectado al curso, su foto –en caso de no haber introducido ninguna en el perfil sale un icono- y el botón para '*Salir'* del curso. Si pulsamos sobre el enlace 'Actualizar información personal' accederemos a la misma página que

obteníamos tras seguir los enlaces de la ruta *"Páncreas-> Diabetes-> Participantes-> Nombre -> Editar Información".* Podemos observar e introducir algunos campos de opciones avanzadas que son irrelevantes para el curso.

La *sección 5* contiene las diferentes actividades habilitadas por el profesor a las que el alumno tiene acceso salvo el contenido didáctico de cada tema que como hemos mencionado está incluido en la sección central 3. Podremos acceder a '*Cuestionarios'* para realizar la evaluación de los contenidos y consultar los resultados de cada intento. En 'Foros' podremos acceder a los tres foros temáticos incluidos para mejorar la comunicación entre alumnado y profesorado. El acceso directo a 'Recursos' te permite acceder al tutorial del curso. Por último, el 'Glosario' nos enlaza al índice de entradas del mismo.

Nota: A parte de las secciones aquí expuestas, dentro del curso se adjunta un tutorial específico para el alumno que explica más en detalle las funcionalidades del curso y cómo moverse dentro de él.

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo hemos presentado un sistema recomendador para mejorar la educación terapéutica del paciente diabético, especialmente el de diagnóstico reciente. Este proyecto se enmarca dentro de un entorno global que consta de 3 partes y cuya finalidad es la de crear una arquitectura online que suponga una herramienta global para la gestión del paciente (ver sección 1.2).

Nuestro trabajo se compone de dos partes fundamentales:

- sistema recomendador
- curso de e-learning

En este capitulo analizamos las conclusiones obtenidas durante el desarrollo de nuestro trabajo y expondremos las características principales del sistema desarrollado. Incluimos además unas propuestas de líneas futuras de trabajo.

5.1. Conclusiones

Como se puede apreciar en el Capítulo 1, Sección 2, los objetivos de este proyecto se enmarcan en un sistema global, completo y cerrado de ayuda a los pacientes diabéticos y a sus endocrinos. Este macro-sistema consta de tres bloques, de los cuales, nuestro proyecto abarca el subsistema de formación y recomendación. Para abordar esto, hemos creado una interfaz amigable que acelera los procesos de aprendizaje. Es fácil de usar y se puede adaptar a cualquier edad. Todo este sistema se construye con base en una serie de módulos software ya implementados, en todos los casos disponibles y de uso libre en Internet.

Una gran parte del resultado de todo este esfuerzo, lo que se refiere al curso de formación, se encuentra disponible en la URL http://pancreas.dacya.ucm.es.

Al tratarse de un trabajo equivalente a un proyecto fin de carrera, las principales conclusiones obtenidas provienen de las dificultades encontradas y decisiones tomadas durante el desarrollo del mismo y que pasamos a explicar.

El trabajo propuesto tenía definidas las funcionalidades del sistema pero lo cierto es que al tratarse de un proyecto coordinado con otros grupos de trabajo e interdisciplinar, se planteó como un trabajo dinámico en cuanto a su especificación. Durante el proceso inicial de especificación de requisitos y durante el proceso de desarrollo a lo largo de todo el curso académico, se han debido tomar ciertas decisiones y proponer ciertas modificaciones sobre el planteamiento inicial para llegar a la consecución de los mismos.

Las principales decisiones se establecieron inicialmente en la elección de las aplicaciones a utilizar tanto para el sistema recomendador como para el curso de educación. Realizamos la primera fase de análisis concienciados de que tendríamos que adquirir cierto conocimiento de la materia, la diabetes, para poder plasmar el conocimiento médico de la enfermedad y del especialista sobre las decisiones de diseño más importantes del proyecto. Tras esta fase de estudio, demás de adentrarnos en este campo médico concreto, realizamos un análisis de las diferentes soluciones software en relación con la educación virtual. Al tratarse de un proyecto en fase experimental y con posibilidad de implantarse en un centro médico real, decidimos utilizar una herramienta libre para evitar el pago de licencias y descartamos otras alternativas como la plataforma WebCT. Aunque es cierto que las herramientas de pago pueden ofrecer más posibilidades a los desarrolladores, programadores, administradores, o profesores, a la hora de personalizar o parametrizar los cursos, convenimos que las funcionalidades de nuestro curso de formación podían llevarse a cabo con la herramienta ya conocida y que finalmente elegimos, Moodle, descartando otros LMS gratuitos como Sakai o Claroline. Con Moodle hemos conseguido crear un curso formativo orientado al diabético, intuitivo y sencillo en su manejo para cualquier edad y accesible desde cualquier sitio al ser un portal Web. Dentro de las posibilidades que ofrece el entorno, hemos puesto especial hincapié en la elección de los temas o en el diseño y organización de contenidos para facilitar o animar al aprendizaje dentro del curso. La fase de pruebas del curso la hemos podido desempeñar sobre un escenario virtual gracias a la colaboración del Departamento DACYA, que nos ha dispuesto un servidor virtual para la

instalación y configuración del curso de manera que podamos simular situaciones y probar la interoperabilidad de cada perfil de usuario, acercándonos notablemente a un posible escenario de su implantación real sobre un servidor físico.

La fase del proceso de desarrollo más relevante y con un mayor tiempo de dedicación asignado ha sido conseguir integrar desde nuestra aplicación los dos flujos de información externos para realizar las recomendaciones. Por una parte, extraer la información de evaluación del paciente de Moodle para las recomendaciones de formación y, por otra, disponer de la información del historial médico de Google Health para las recomendaciones de seguimiento. Dentro de éstas últimas han sido de vital importancia las reuniones con la especialista para determinar, por ejemplo, los intervalos concretos de valores de análisis que pueden producir situaciones anómalas en los pacientes diabéticos.

Como hemos indicado, la segunda parte es un sistema recomendador y para este tipo de recomendaciones de seguimiento semanal decidimos utilizar igualmente una herramienta de software libre, jCOLIBRI, que aplica técnicas de razonamiento basado en casos y que servirá de base para crear el modelo inteligente que es el objetivo final del proyecto. Las razones fundamentales para esta elección fueron por un lado, su gratuidad y por otro, nuestro conocimiento previo de la misma: tras analizar más a fondo la herramienta jCOLIBRI, concluimos que sus funcionalidades nos proporcionaban la manera de representar las recomendaciones sobre el objeto de estudio, la diabetes en nuestro caso, con cierta facilidad de parametrización. Los principales inconvenientes que hemos tenido con el uso de la herramienta han sido la configuración del conector a través de Hibernate, el mapeo de la base de casos con la memoria del sistemas a través de los distintos ficheros XML y la integración de jCOLIBRI en el entorno de desarrollo NetBeans ya que en el manual de la herramienta tan sólo viene explicado su utilización y uso en el entorno Eclipse.

5.2. Trabajo futuro

En la introducción mencionábamos la visión global del proyecto y citábamos cada una de las tres partes en las que se componían. Durante el presente curso académico, se ha trabajado de manera paralela tanto sobre la Parte A, la recogida de información y almacenaje persistente, como nuestra Parte B,

el entorno educativo y recomendador. Es cierto que ha habido comunicación entre los dos grupos para utilizar la API desarrollada pero al ser un trabajo independiente y en paralelo, uno de los grandes esfuerzos del futuro deberá ser enfocado a conseguir integrar estas dos partes de manera coherente a través de la misma interfaz. Es posible que al ser la suya una aplicación de presentación de información, tenga un impacto visual más atractivo y sea más conveniente integrar nuestras funcionalidades en su sistema. De ser así, no sería necesario guardar a los usuarios con perfil de Médico en la tabla de usuarios en nuestra base de datos, ya que ellos comprueban si es médico mediante la conexión a la cuenta de Google Health. También, será importante crear una buena y completa base de casos para conseguir que la herramienta ofrezca recomendaciones de la manera más fiable posible. Esto pasa por disponer de un importante número de pacientes/alumnos que utilicen la aplicación, que tengan sus datos de análisis actualizados al menos semanalmente para que se puedan realizar las recomendaciones de seguimiento y que el médico se ocupe también de introducir manualmente casos interesantes de historiales de pacientes reales o ficticios que puedan llegar a producirse.

En cuanto al curso implantado en Moodle, la introducción de más contenidos o el desarrollo de más preguntas o nuevos test corren a cargo también del profesor o médico. Se ha diseñado y organizado el temario, los tests, sus preguntas y los foros temáticos, delegando al profesor la tarea de la actualización y subida de contenidos, para una formación completa. En referencia a las pruebas realizadas sobre el servidor virtual dispuesto del departamento DACYA, hemos detectado tiempos de acceso al curso bastante elevados, y por tanto una bajada del rendimiento, dependiendo del número de administradores, profesores o alumnos funcionando en el sistema o incluso dependiendo de la cercanía o no del Host cliente; ha sido necesario y de gran utilidad disponer del servidor para desarrollar el plan de pruebas en la fase experimental pero sería necesaria una inversión en una máquina física exclusivamente dedicada al alojamiento del curso en caso de que finalmente el proyecto llegue a fase de producción.

Y por último, algo muy importante, la seguridad y privacidad de los datos. Lo más cercano que hemos hecho para cubrir la parte de seguridad es la de encriptar las contraseñas de acceso de los usuarios, mediante algoritmos de encriptación. Posiblemente sea necesario un paquete de medidas para asegurar la utilización de los datos y su acceso.

Bibliografía

[1] SANZ BOU, M. A., SANCHEZ MILLA, J. J. La diabetes ¿criterio de selección para el trabajo?: el médico del trabajo ante el trabajador diabético. Revista de la Sociedad Española de Salud Laboral en la Administración Pública, ISSN 1575-8524, Vol. 2, Nº 11, Págs. 11-19 (2011).

[2] FARLEY ORTIZ, LUIS, Campus virtual. La educación más allá del LMS. RUSC, Revista de Universidad y Sociedad del Conocimiento, (Oct 2006).

[3] GARCÍA SALCINES, E., ROMERO MORALES, C., VENTURA SOTO, S., DE CASTRO LOZANO, C. Sistema recomendador colaborativo usando minería de datos distribuida para la mejora continua de cursos e-learning. IEEE-RITA Vol. 3, Núm. 1, (May 2008)

[4] THE DIABETES CONTROL AND COMPLICATIONS RESEARCH GROUP. The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. N Engl J Med 329, 14 (Sep 1993), 977–986.

[5] ABEREGG, S. K. Intensive insulin therapy in the medical icu. N Engl J Med 354, 19 (May 2006), 2069–2071.

[6] DELLINGE R, R.P., CARLET, J.M., MASUR, H., GERLACH, H., CALA NDRA, T., COHEN, J., GEA-BANACLOCHE, J., KEH, D., MARSHALL, J.C., PA RKER, M.M., RAMSAY, G., ZIMMERMAN, J.L., VINCENT, J.L., ANDLEVY, M. M. Surviving sepsis campaign guidelines for management of severe sepsis and septic shock. Intensive Care Med 30, 4 (Apr 2004), 536–555.

[7] FINNEY, S.J., ZEKVELD, C., ELIA, A., AND EVANS, T.W. Glucose control and mortality in critically ill patients. JAMA 290, 15 (Oct 2003), 2041–2047.

[8] KRINSLEY, J. S. Effect of an intensive glucose management protocol on the mortality of critically ill adult patients. Mayo Clin Proc 79, 8 (Aug 2004), 992–1000.

[9] KRINSLEY, J.S., AND JONES, R. L. Cost analysis of intensive glycemic control in critically ill adult patients. Chest 129, 3 (Mar 2006), 644–650.

[10] MALMBERG, K. Prospective randomized study of intensive insulin treatment on long term survival after acute myocardial infarction in patients with diabetes mellitus. Digami (diabetes mellitus, insulin glucose infusion in acute myocardial infarction) study group. BMJ 314, 7093 (May 1997), 1512–1515.

[11] NATHAN, D.M., CLEARY, P.A., BACKLUND, J. -Y.C., GENUTH, S.M., LACHIN, J.M., ORCHARD, T.J., RASKIN, P., AND ZINMAN, B. Intensive diabetes treatment and cardiovascular disease in patients with type 1 diabetes. N Engl J Med 353, 25 (2005), 2643–2653.

[12] NISHIDA, K., SHIMODA, S., ICHINOSE, K., ARAKI, E., AND SHICHIRI, M. What is artificial endocrine pancreas? Mechanism and history. World J Gastroenterol 15, 33 (2009), 4105–4110.

[13] VAN DEN BERGHE, G., WOUTERS, P., WEEKERS, F., VERWAEST, C., BRUYNINCKX, F., SCHETZ, M., VLASSELAERS, D., FERDINANDE, P., LAUWERS, P., AND BOUILLON, R. Intensive insulin therapy in the critically ill patients. N Engl J Med 345, 19 (Nov 2001), 1359–1367.

[14] RECIO GARCÍA, J.A., DÍAZ AGUDO, B., GONZÁLEZ CALERO, P. jColibri 2 Tutorial versión 1.2, (Sep 2008).

[15] MONSO MARTÍN, R. Capitulo1. phpMyAdmin. Software Libre para sitios Web, (Nov 2004)