ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMATICA DE SISTEMAS

Curso académico 2010/2011

Proyecto de Fin de Carrera

Fisca, un agente que te enseña a resolver problemas de Física y Química

Autor: Antonio Boza España

Tutora: Diana Pérez

RESUMEN

El motivo principal de este proyecto es el desarrollo de un sistema informático catalogado como Agente Conversacional Pedagógico (ACP). El objetivo de este ACP es ser capaz de presentar a los estudiantes problemas de física y química adaptados a su nivel de aprendizaje. Este proceso se realiza mediante la interacción con la aplicación en lenguaje natural basada en una serie de pasos y fórmulas previamente introducidos por el profesor.

Más concretamente, este proyecto se centra en el área de conocimiento de *e-learning*. Son sistemas que ayudan a aprender y repasar, de forma sencilla e intuitiva haciendo uso de las Tecnologías de la Información presentes.

La aplicación está orientada a alumnos en edad escolar para servir de apoyo en el estudio. Es una herramienta simple, estilo "Messenger", accesible para cualquier usuario que necesite repasar la materia de física y química, sin necesidad de tener elevados conocimientos informáticos.

La aplicación se divide en dos partes: Profesor y Alumno. La aplicación Profesor está orientada al uso por parte de los profesores. Los profesores a través de ella, generarán problemas de física y química dejándolos preparados para que los alumnos los puedan resolver. La aplicación Alumno está orientada a los alumnos. En ella los alumnos resolverán los problemas que los profesores hayan creado anteriormente.

La aplicación se ha implementado mediante un lenguaje de programación orientado a objetos, el lenguaje Java, diseñando su interfaz a través de la clase JSwing. El proyecto cuenta con un diseño basado en la arquitectura Modelo-Vista-Controlador. Para almacenar la información necesaria para la aplicación se cuenta con una base de datos relacional gestionada a través de un sistema de gestión de bases de datos relacional, multihilo y multiusuario, MySQL.

Las pruebas realizadas de la aplicación se han realizado sobre un grupo de personas con diferentes edades y conocimientos. Estas pruebas confirman que la aplicación cumple con el objetivo de poder ser usada por personas con conocimientos básicos de informática, ampliando así el rango de acción de la herramienta. Los conocimientos necesarios, pero no obligatorios, para utilizar esta herramienta son los de física y química. Aunque la herramienta también resulte atractiva para usuarios que aún no hayan cursado la asignatura de física y química.

AGRADECIMIENTOS

A todo el que ha sido capaz de aguantarme durante estos años y entender lo duro que es, sobre todo en los momentos malos, mis amigos.

A los profesores que de verdad se han esforzado en que aprendiéramos.

A mi tutora Diana, que ha conseguido que por fin me sienta ingeniero.

A mi familia, que siempre se han sentido orgullosa.

Y por supuesto, a mi señora esposa, que sin ella habría sido imposible llegar al final.

ÍNDICE

ÍNDICE FIGURAS

ÍNDICE TABLAS

1. INTRODUCCIÓN

En este proyecto de fin carrera se trata el tema de la tecnología de la información aplicada al aprendizaje. Se basa en cómo nos podemos ayudar de la tecnología para aprender, repasar y estudiar conceptos vistos en clase o nuevos. Para ello, se desarrollará un agente conversacional pedagógico. Un agente conversacional pedagógico (ACP) se puede definir como una personificación o representación en un ordenador de la figura del profesor o del estudiante. El agente puede mostrar características humanas, mediante voz, texto o gráficos. El objetivo de este agente será por tanto el apoyo al proceso educativo.

La materia de trabajo será la asignatura de Física y Química. Se usará el agente como medio para que el usuario mejore sus conocimientos mediante la resolución de problemas de física y química. Los problemas serán de física básica y de formulación y nomenclatura para la química.

Hoy en día, la mayoría de los niños tienen la opción de estudiar, al menos en el primer mundo. Por ello, existe un gran abanico de estudiantes y no todos con el mismo interés y capacidad para el estudio. Muchos de ellos necesitan ayuda adicional, o simplemente una opción de repaso de conceptos. La tecnología es una opción cada vez mas asumida para la enseñanza y formación de los niños, como se puede observar en varios ámbitos educativos. Por lo tanto estamos usando la tecnología para dar la posibilidad de estudio desde cualquier dispositivo conectado a internet con flexibilidad espacio – temporal y al ritmo de cada estudiante.

Este proyecto de fin de carrera pretende aportar una herramienta libre. Cualquier niño la podrá utilizar para su formación académica, pudiendo ser usado tanto a través de un sistema *e-learning* (usando únicamente la herramienta informática para el estudio), como un sistema *b-learning* (mezclando el uso de la herramienta informática con la ayuda de un profesor). Aumentando así su versatilidad, teniendo la capacidad de ser usada en diferentes entornos.

El Agente Conversacional Pedagógico (ACP) se ha diseñado como si fuera una aplicación tipo Messenger, donde habrá dos usuarios (profesor y alumno) definidos mediante una aplicación cada uno. El ACP simulará la relación entre alumno y profesor. El profesor será quien maneje la creación de los problemas, problemas que serán resueltos por el alumno. El alumno recibirá la ayuda que anteriormente haya dispuesto el profesor. Serán frases de ánimo, explicación de problemas y motivación, fomentando así el efecto persona, es decir, que la herramienta haga sentir al alumno que está siendo evaluado por un profesor de carne y hueso.

En cuanto a la aplicación profesor, será usada para insertar problemas, modificarlos y borrarlos o dar de alta alumnos nuevos. La herramienta irá guiando a los profesores para crear los

problemas de forma correcta con sus pasos y parámetros, que posteriormente podrán seguir los estudiantes cuando entren en la aplicación estudiante.

La aplicación estudiante se utiliza para la resolución de problemas seleccionados según las necesidades del usuario. La Figura 1 muestra un ejemplo de la interfaz alumno desarrollada mediante el lenguaje orientado a objetos Java.

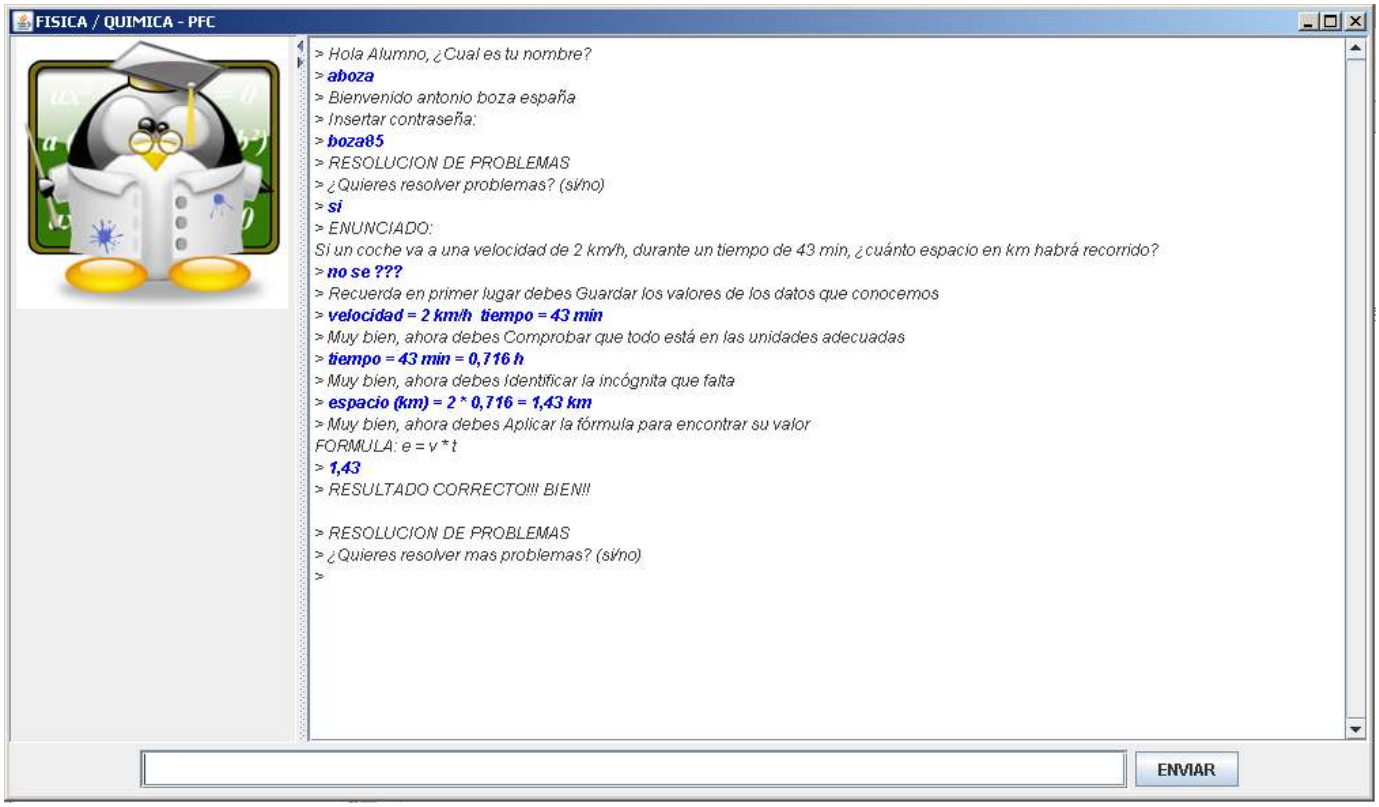

Figura 1: Ejemplo de interfaz

Una vez que el alumno haya registrado y haya introducido su nombre de usuario y contraseña, ya puede acceder a resolver problemas, como se observa en la Figura 1. El alumno se enfrentará a problemas creados mediante un conjunto de pasos por el profesor. Estos pasos serán los que guíen al alumno durante la resolución, si no fuese capaz de resolverlos.

Este documento se organiza en 5 capítulos, tras esta introducción, el capítulo 2 presenta los objetivos del trabajo. El capítulo 3 consiste en la descripción del sistema informático desde su análisis hasta la estructura del código implementado. El capítulo 4 se centra en las pruebas realizadas desde las unitarias hasta las de aceptación con un grupo de usuarios. Finalmente el capítulo 5 presenta las conclusiones y posibles líneas de trabajo futuro.

2.OBJETIVOS

2.1. DESCRIPCION DEL PROBLEMA

La resolución de problemas constituye uno de los ejes fundamentales donde se asienta la didáctica de la Ciencias Experimentales. El álgebra, la física o la química, son asignaturas que tradicionalmente cuentan con una alta tasa de suspensos y por lo tanto, con alumnos desmotivados que en algunos casos llegan incluso a considerar que es demasiado difícil afrontarlas.

Analizando los resultados obtenidos al estudiar las dificultades que poseen distintos grupos de alumnos de ESO en la resolución de problemas de lápiz y papel [1]. Se muestra que la carencia de estrategias adecuadas de resolución y la falta de aprendizaje significativo son las principales fuentes de fracaso en los alumnos de ESO. Sería conveniente, desde la formación inicial, incidir en los aspectos procedimentales de la resolución de problemas para paliar esta carencia.

Como posible solución existen los sistemas de repaso interactivo y más concretamente el Agente Conversacional Pedagógico (ACP). El ACP [2] es un sistema informático que interacciona con el estudiante en lenguaje natural, proporcionando un repaso de los temas vistos en clase. El uso de este tipo de agente es normal en metodologías de Aprendizaje Híbrido (Blended Learning, b-learning) según las cuales el profesor combina las clases presenciales con el uso de sistemas informáticos como apoyo durante las clases, o bien para ser usados en tareas no presenciales, como son los e-learning. El agente alberga una serie de características necesarias, las más importantes son la adaptabilidad, el soporte afectivo y la capacidad de evolución como se comenta a continuación.

La adaptabilidad, que se refiere a que el agente puede ajustar los contenidos según las necesidades o conocimientos del alumno. En este proyecto el propio agente será capaz de ir mostrando los problemas de Física y Química según el estudiante los haya afrontado anteriormente, mediante una estrategia de proponer al alumno siempre los problemas que más le cueste resolver. Consiguiendo así que el agente se adapte al alumno.

Otra característica destacable sería el soporte afectivo, el agente debe intentar animar al estudiante y mantener su atención. Para ello debe usar un lenguaje cercano y positivo. A la hora de mantener la atención del estudiante le ofreceremos al alumno un formato conocido y usado, como es el formato de la conocida aplicación “Messenger”, donde a través de una foto identificamos quien está al otro lado de la pantalla. Por lo tanto al ser un soporte cotidiano, ya que hoy en día todos los niños usan chats para hablar con sus amigos, se espera que el alumno se sienta cómodo al usar la herramienta.

Como última característica general destacamos la capacidad de evolución, el agente puede ir aprendiendo del estudiante experimentando una evolución según los resultados obtenidos. Estas características son fundamentales para que el proyecto sea considerado un ACP, como se puede comprobar en el capítulo 4.3.Pruebas de validación, cumpliéndose los requisitos esperados.

Un ACP se puede utilizar mediante varios modelos de aprendizaje, sistemas *e/b-learning*. El *e-learning* equivale a la transformación de todas las formas de educación y aprendizaje en el siglo XX mediante el uso de las tecnologías. Su objetivo es la mejorar el aprendizaje y el rendimiento de los estudiantes. El *e-learning*, permite construir y poner a disposición del usuario, cursos educativos y de entrenamiento de alta calidad [3].

La formación tradicional es una metodología a veces insuficiente para atender la creciente demanda de formación. Se cuenta con infraestructura limitada, costes elevados y además, los estudiantes carecen del tiempo necesario.

Por lo tanto la utilización de plataformas de *e-learning* como instrumento de formación personal presentan ventajas como:

- La adaptabilidad al ritmo individual de cada alumno.
- Fomento del trabajo autónomo del alumno.
- La reducción de costes, habitualmente por el desplazamiento, alojamiento, etc.
- La eliminación de barreras espaciales.
- La flexibilidad temporal.

Por otro lado, el *B-Learning* [4] se puede considerar una especie de evolución del *e-learning*, un curso dictado en este formato (*B-Learning*) incluirá tanto clases presenciales como actividades de *e-learning*. Este modelo de formación hace uso de las ventajas de la formación 100% on-line y la formación presencial, combinándolas en un solo tipo de formación que agiliza la labor tanto del profesor como del alumno. Las ventajas que se suelen atribuir a esta modalidad de aprendizaje son las que se atribuyen al modelo *e-learning* más las del modelo presencial *(b-learning)*:

- Aplicación de los conocimientos.
- Interacción física.
- Facilita el establecimiento de vínculos, ofreciendo la posibilidad de realizar actividades algo más complicadas de realizar de manera puramente virtual.

2.2. OBJETIVOS

El objetivo principal de este proyecto fin de carrera es desarrollar un Agente Pedagógico Conversacional capaz de presentar a los estudiantes problemas de física y química adaptados a su nivel de aprendizaje y siguiendo una interacción en lenguaje natural basada en una serie de pasos y fórmulas previamente introducidos por el profesor.

Para cumplir este objetivo, se han desarrollado dos programas. El primero de ello va destinado a los profesores que podrán crear cualquier tipo de problema de física y química. El segundo está destinado a los estudiantes que podrán resolver los problemas que se les va mostrando en una interfaz tipo Messenger intuitiva y amigable para que puedan repasar de una forma amena.

El profesor será el encargado de diseñar el problema interactuando con el agente. El agente le irá pidiendo a cada paso la información necesaria según el propio diseño que vaya generando el profesor. Deberá definir el enunciado, parámetros y pasos, añadiéndole una fórmula final para que el problema esté bien definido y pueda ser almacenado.

Realizar el mantenimiento de sus propias creaciones, borrando y modificando la información almacenada, además de permitir o no el acceso a los estudiantes a la aplicación, son otras de las acciones que puede realizar el profesor.

En cuanto al grupo de los alumnos, el objetivo principal es la realización de problemas de Física y Química para repaso y estudio, y por tanto la mejora de sus conocimientos de la materia.

De esta forma, se espera que mediante la realización de los problemas que vayan apareciendo en la aplicación, los estudiantes vayan aprendiendo la metodología de resolución. El alumno deberá apoyarse en la información que se vaya mostrando tras cada inserción de un resultado erróneo que serán los pasos que el profesor haya definido anteriormente. Para así aprender que pasos de deben realizar para una adecuada resolución de los problemas.

2.3 ESTUDIO DE ALTERNATIVAS

2.3.1 PARADIGMAS DE PROGRAMACION

Un paradigma de programación [5] provee y determina la visión y métodos de un programador en la construcción de un programa. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de problemas. Existen múltiples paradigmas, difícilmente un lenguaje de programación pueda clasificarse solamente en un paradigma. La Figura 2 muestra los tipos de paradigmas de programación más comunes:

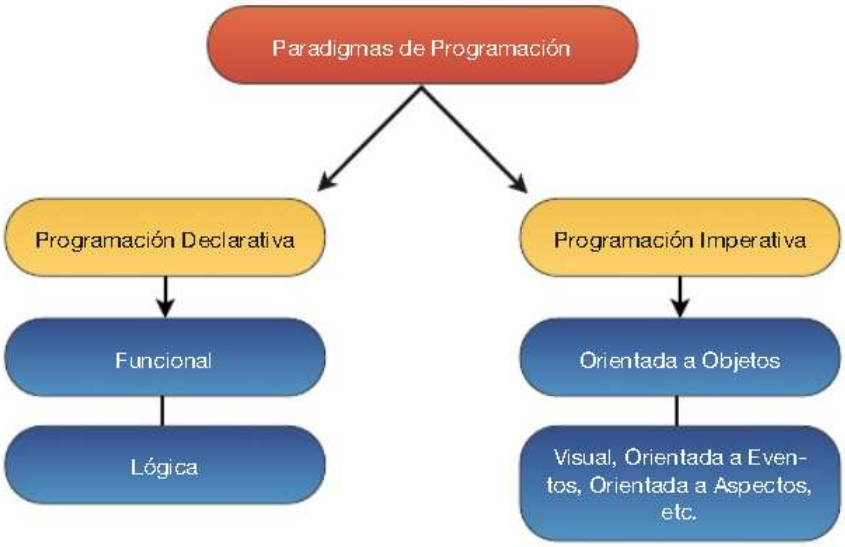

Figura 2: Esquema de Paradigmas de programación.

Probablemente el paradigma de programación que actualmente es el más usado a todos los niveles es la orientación a objeto (POO). El núcleo central de este paradigma es la unión de datos y procesamiento en una entidad llamada "objeto", relacionable a su vez con otras entidades "objeto". Con la orientación a objetos y características como el encapsulado, polimorfismo o la herencia se permitió un avance significativo en el desarrollo de software a cualquier escala de producción.

En consecuencia, el proyecto se realizará bajo el paradigma de programación orientado a objetos. Los aspectos que afectan a la decisión de un lenguaje orientado a objetos son los siguientes:

- Claridad. Al ligar de forma evidente la estructura de la información con los procedimientos que la manipulan, los programas ganan en claridad a la hora de desarrollarlos y mantenerlos. Esto supone una ventaja frente a los lenguajes

procedurales, aunque éstos podrían suplir esta deficiencia mediante una correcta elección de los nombres de las variables y funciones, mediante una oportuna codificación.

- Complejidad. La complejidad de un problema es abarcable por una sola persona y resolverlo con una herramienta u otra no va a aportar grandes ventajas. Pero al realizar el desarrollo por un equipo grande, debe existir una forma para aislar partes de problema y repartir el trabajo, y aunque en este proyecto no sea un criterio decisivo, es una buena manera de programar. De hecho las bases de datos orientadas a objetos suponen un gran adelanto, ya que aúnan la flexibilidad en la manipulación de los OOP con la capacidad de consulta de un DBMS (Data Base Management System).

- Relación entre datos. Por el mismo motivo se verán beneficiados aquellos programas que impliquen una relación compleja entre los datos. Este tipo de complejidad permite la utilización de todas las ventajas de los lenguajes de programación orientados a objetos. Propiedades como la herencia (donde los objetos pueden heredar estructura y operaciones de objetos predecesores), la encapsulación, etc. Muestran en este tipo de programas todas sus ventajas.

- Gestión de recursos. Las aplicaciones orientadas a objetos demandan normalmente más recursos del sistema que las aplicaciones procedurales. La creación dinámica de objetos, que ocupa un lugar en la memoria del ordenador, puede acarrear graves problemas. Una de las soluciones, que incluye alguno de los lenguajes OOP, es liberar a menudo el espacio que los objetos dejan de utilizar. Este procedimiento de optimización como “garbage collection” (recolección de basura, implementado en java), minimiza los efecto de la creación dinámica de objetos.

- El interface de usuario es uno de los aspectos más importantes en la programación actual. La aparición de sistemas de explotación que soportan un interface gráfico de usuario como Windows, X-Windows o Presentation Manager hace que la mayoría de los usuarios prefieran que sus programas corran bajo este tipo de interface. Este es uno de los puntos fuertes para la elección de un lenguaje OOP. La mayoría de los interfaces gráficos actuales han sido diseñados o rediseñados en base a la OOP. Existen en el mercado librerías de clases que soportan todos los dispositivos de control de ventanas como menús, combo box, listas, barras de herramientas, etc. Siendo éste uno de los aspectos primordiales de la elección de este paradigma para desarrollar el proyecto.

2.3.2 LENGUAJES DE PROGRAMACION

Los lenguajes de programación que se evalúan con el fin de determinar el adecuado para el desarrollo del proyecto son los lenguajes orientados a objetos. Los lenguajes OOP implementan de manera distinta los conceptos de programación orientada a objetos. No existe el lenguaje perfecto capaz de satisfacer todas las necesidades y que se adapte a todos los estilos, por lo tanto a la hora de la elección del lenguaje de programación adecuado, debemos tener en cuenta varias características:

Eficiencia. Los lenguajes OOP arrastraron en un principio la reputación de ser ineficaces. Esto se debía en gran medida a que los primeros lenguajes (como Smalltalk) eran interpretados y no compilados. La existencia de compiladores permite a los desarrolladores ganar rapidez. Actualmente, usando un buen lenguaje orientado a objetos como C++, Java, etc., junto con las librerías apropiadas para la realización de un programa, puede que se ejecute más rápidamente que el mismo programa compilado con un lenguaje procedural.

Asignación de tipos**.** Los lenguajes orientados a objetos varían de forma sustancial la forma por la que se aproximan a la asignación de tipos. Por asignación de tipos entendemos que cada variable sea identificada como perteneciente a una clase (asignación fuerte) o sea simplemente un objeto indeterminado (asignación débil). Eiffel y C son dos lenguajes basados en la asignación fuerte, frente a Smalltalk, en el que todas las variables definidas pertenecen a una clase indeterminada. La asignación fuerte sirve a dos propósitos. Por una parte para que el desarrollador pueda identificar a que clase pertenece cada operación. Por otra, permite al compilador un mayor grado de optimización, ya que conoce en todo momento el espacio que ha de asignar.

Manejo de memoria. Los OOP son lenguajes que utilizan de manera intensiva la memoria de la computadora. Hay dos tipos de aproximación a la gestión de memoria. En el primero el sistema en tiempo de ejecución libera la memoria automáticamente a medida que los objetos dejan de utilizarse. En el segundo el sistema tiene instrucciones concretas para liberar la memoria explícitamente. Este es el enfoque adoptado por lenguajes como C++, que aportan dos operadores: crear y destruir. El primero reserva automáticamente memoria, mientras que el segundo la libera.

Encapsulación, que consiste en separar aquellos atributos del objeto que deben ser conocidos por el resto, de aquellos necesarios para su funcionamiento propio.

Ejemplo de lenguajes basados en este paradigma son: Simula, Smalltalk, C++, Eiffel, Java Script, Java, PHP, Visual Basic .NET, etc. A continuación, una pequeña panorámica de los lenguajes orientados a objetos:

SMALLTALK

Fue el primer lenguaje de programación orientado a objetos. En sus primeras implementaciones no ofrece solamente un intérprete, sino que es mucho más ambicioso, integrando intérprete on-line y otros aspectos que le convierten en un pseudo-sistema operativo. Es el primero en aportar la arquitectura de Modelo/Vista/Controlador, utilizada en este proyecto. El MVC será explicado en el capítulo 3.2.

Ventajas

- Smalltalk es un lenguaje puro orientado a objetos
- La implementación a través de un intérprete facilita la labor de desarrollo de programas. Las clases son añadidas, corregidas y depuradas de forma interactiva.
- Tiene una sintaxis simple, donde las variables y los atributos no necesitan tener un tipo asociado. Todo está definido en principio como objeto, incluyendo las propias clases.

Inconvenientes

- Es un lenguaje interpretado, lo que reduce su rendimiento y dificulta su comercialización.
- Al proporcionar su propio entorno operativo, interactúa mal con otro tipo de software o hardware

EIFFEL

Al contrario que Smalltalk, incluye un preprocesador que permite la traducción de código Eiffel a Lenguaje C. Es ideal para la ingeniería de software, que permite la encapsulación, control de acceso y ámbito de las modificaciones. Como lenguaje orientado a objetos puro, es presumiblemente el mejor por sus capacidades técnicas.

El punto primordial de un programa Eiffel es la declaración de clases, que asocia atributos. Ambos, clases y atributos, son accesibles a partir de la implementación de un concepto

llamado característica. Una característica es, por tanto, una agrupación de datos y unas formas típicas de tratarlos.

Ventajas

- Es un lenguaje orientado a objetos puro
- Eiffel es un lenguaje de programación orientado hacia el diseño de grandes aplicaciones. Ideal para el diseño de aplicaciones en grupos de trabajo.
- Su compatibilidad con C asegura también su portabilidad hacia otros S.O.

Desventajas

- El manejo de la memoria, un punto delicado en todos los lenguajes orientados a objetos no es transparente como en el caso de Smalltalk.
- Las librerías de clases son reducidas
- El rendimiento es mayor que el de Smalltalk, pero al tener que incluir un módulo Run-time dentro del ejecutable, su tamaño crece y su rendimiento baja.

C++

Es un lenguaje de uso general que deriva del C. Añade a su predecesor una serie de características que le convierten en un lenguaje orientado a objetos: La abstracción de datos y la programación orientada a objetos, ya que permite asociar a los datos las funciones que los manipulan. C++ conserva todas las capacidades de su predecesor C. Los aspectos más importantes que hacen del C++ un lenguaje orientado a objetos son:

- La mayor contribución que realiza C++ al C es la introducción del tipo clase. Las clases permiten definir conjunto de datos y las funciones que los manipulan.
- La ocultación de datos es el mecanismo para implementar la abstracción de datos. Permite al programador olvidar el funcionamiento interno de una clase.
- La herencia extiende el concepto de abstracción de datos al permitir la construcción de clases a partir de otras (sus antecesores).
- Los operadores definidos por el usuario permiten un tratamiento homogéneo entre los tipos predefinidos por el lenguaje y los desarrollados por el programador.

PHP

PHP no es un lenguaje orientado a objetos puro pero tiene muchas características comunes que hacen que aparezca en el informe. Originalmente considerado como un simple conjunto de scripts de Perl para guiar a los usuarios en sus páginas. Las cuatro características principales de PHP son las siguientes:

- Velocidad: A parte de velocidad de ejecución, no crea demoras en la máquina. Por esta razón no debe requerir demasiados recursos de sistema.
- Estabilidad: PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- Seguridad: El sistema debe poseer protecciones contra ataques. PHP provee diferentes niveles de seguridad.
- Simplicidad: Se les debe permitir a los programadores generar código en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.

Otra característica a tener en cuenta seria la conectividad. PHP dispone de una amplia gama de librerías, y agregarle extensiones es muy fácil. Esto le permite al PHP ser utilizado en muchas áreas diferentes, tales como encriptado, gráficos, XML y otras.

JAVA

Sun Microsystems (1991). El objetivo era utilizarlo en un set-top box, un tipo de dispositivo que encarga de la recepción y la decodificación de la señal televisiva. El primer nombre del lenguaje fue Oak, luego se conoció como Green y finamente adoptó la denominación de Java.

La intención de Sun era crear un lenguaje con una estructura y una sintaxis similar a C y C++, aunque con un modelo de objetos más simple y eliminando las herramientas de bajo nivel. Los pilares en los que se sustenta Java son cinco:

- Debería usar la metodología de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería incluir por defecto soporte para trabajo en red.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Lo habitual es que las aplicaciones Java se encuentren compiladas en un bytecode (un fichero binario que tiene un programa ejecutable), aunque también pueden estar compiladas en código máquina nativo.

Sun controla las especificaciones y el desarrollo del lenguaje, los compiladores, las máquinas virtuales y las bibliotecas de clases a través del Java Community Process. En los últimos años, la empresa (que fue adquirida por Oracle) ha liberado gran parte de las tecnologías Java bajo la licencia GNU GPL.

La aplicación de Java es muy amplia. El lenguaje se utiliza en una gran variedad de dispositivos móviles, como teléfonos y pequeños electrodomésticos. En los navegadores web, Java permite desarrollar pequeñas aplicaciones conocidas como applets que se incrustan en el código HTML de las páginas. Mediante un plug-in se ejecutarán las aplicaciones Java en el navegador.

Para conseguir la ejecución de código remoto y el soporte de red, los programadores de Java a veces recurren a extensiones como CORBA (Common Object Request Broker Architecture), Internet Communications Engine o OSGi respectivamente.

2.3.3 GESTORES DE BASES DE DATOS

Los sistemas gestores de base de datos [7] se pueden definir como el intermediario o la interfaz entre la base de datos, el usuario y el programa. Su propósito es el manejo de la información de la base de datos, permitiendo introducir, organizar y recuperar la información de las bases de datos, en definitiva, administrarlas.

Un sistema gestor de bases de datos (SGBD) debe definir la Base de Datos mediante el Lenguaje de Definición de Datos, el cual permite especificar la estructura, tipo de datos y las restricciones sobre los datos, almacenándolo todo en la base de datos.

Un SGBD presenta separación entre la descripción y manipulación de la data, permitiendo un mayor entendimiento de los objetos, además de flexibilidad de consulta y actualización de los datos. Permite la inserción, eliminación, actualización, consulta de los datos mediante el Lenguaje de Manejo de Datos, lo que permite resolver el problema que presentan los sistemas de archivos, donde hay que trabajar con un conjunto fijo de consultas o la necesidad de tener muchos programas de aplicaciones. Existen dos tipos de programas de Manejo de Datos, los cuales se diferencian por la forma en que acceden a los datos:

Lenguajes procedurales: manipulan la base de datos registro a registro y se deben especificar las operaciones a realizar para obtener los datos.

Lenguajes no procedurales: manipulan la base de datos en conjuntos de registros y se especifican qué datos deben obtenerse como resultado sin plantear las forma deshacerlo. El lenguaje no procedural más utilizado es SQL (Structure Query Languaje) que se ha convertido en un estándar y el lenguaje por defecto de los SGBD relacionales.

La existencia de estos sistemas (SGBD) provocan tanto ventajas como inconvenientes a la hora de manejar los datos. Como ventajas podemos destacar:

• Mejora en la integridad de datos: Se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar, aplicadas tanto a los datos, como a sus relaciones.

• Mejora en la seguridad: Los SGBD permiten mantener la seguridad mediante el establecimiento de claves para identificar al personal autorizado a utilizar la base de datos.

• Mejora en la accesibilidad a los datos: Suelen proporcionar lenguajes de consultas o generadores de informes permitiendo al usuario hacer cualquier tipo de consulta sobre datos.

• Mejora en la productividad: A nivel básico, el SGBD proporciona todas las rutinas de manejo de archivos típicas de los programas de aplicación, sin tener que preocuparse de los detalles de implementación de bajo nivel.

• Mejora en el mantenimiento gracias a la independencia de datos: Los SGBD separan las descripciones de los datos de las aplicaciones.

• Aumento de la concurrencia: La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas en el acceso de múltiples usuarios.

• Mejora en los servicios de copias de seguridad y de recuperación ante fallos: Los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

Pero también presentan inconvenientes:

• Complejidad: Son conjuntos de programas muy complejos con una gran funcionalidad. Es preciso comprender muy bien esta funcionalidad para poder sacar un buen partido de ellos.

• Tamaño: Los SGBD son programas complejos y muy extensos que requieren una gran cantidad de espacio en disco y de memoria para trabajar de forma eficiente.

• Coste económico del SGBD: El coste de un SGBD varía dependiendo del entorno y de la funcionalidad que ofrece. Además, hay que pagar una cuota anual de mantenimiento.

• Costo del equipamiento adicional: Para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir una máquina con más prestaciones o una máquina sola para la BBDD.

• Prestaciones: Los SGBD están escritos para ser más generales y ser útiles en muchas aplicaciones, provocando que algunas no sean tan rápidas como en los sistemas de archivos.

• Vulnerable a los fallos: El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse.

Existen distintos tipos de gestores de bases de datos: relacional, jerárquico, red, etc. El modelo relacional es el utilizado por casi todos los gestores de bases de datos. El modelo relacional (SGBDR) es un software que almacena los datos en forma de tablas, que es la manera en la que se almacena la información representativa en este proyecto.

El principal objetivo es encontrar el SGBD que sea capaz de responder adecuadamente al conjunto de aplicaciones y a las exigencias de los usuarios. Se debe tener en cuenta varias características que definen un SGBD:

• Modelo de datos: relacional, jerárquico o en red.

• Lenguajes de definición y manipulación de datos (SQL)

• Herramientas de ayuda para el desarrollo

Además de aspectos que diferencien los distintos SGBD:

• Capacidad de almacenamiento y recuperación de datos: velocidad de ejecución de consultas, creación de índices, importación de datos, etc.

• Protección de datos: acceso simultáneo de varios usuarios, etc.

• Control de accesos de los usuarios: intentos de acceso de usuarios no autorizados, etc.

• Consumo de recursos: memoria RAM, etc. (algunos SGBD incorporan monitores que permiten realizar un seguimiento de los recursos consumidos)

Con la salida al mercado de múltiples entornos de desarrollo, la preocupación está en conocer las características, ventajas y desventajas de cada herramienta que ofrece el mercado. Los productos que más se destacan como son Oracle, Microsoft SQL Server y Borland Interbase que comercialmente son los más fuertes, sin embargo en el mundo del software libre, se aprecian opciones tan completas como MySQL, y postgreSQL.

POSTGRESQL

PostgreSQL se diseño como una base de datos orientada a objetos, es decir, una ORDBMS. Esto significa, que las tablas no son tablas, sino objetos, y las tuplas son instancias de ese objeto. Se pueden crear nuevos tipos de datos y hacer herencias entre objetos. PostgreSQL tiene transacciones, integridad referencial, vista y multitud de funcionalidades, pero es lento y pesado.

Han incorporado la llamada MVCC (multiversion concurrency control) con lo que los bloqueos de escritura actúan sólo en la sesión del cliente, no en las de los demás clientes. También tiene soporte de Full-Text-indexing .PostgreSQL provee nativamente soporte para: Números de precisión arbitraria, texto de largo ilimitado, figuras geométricas (con una variedad de funciones asociadas), direcciones IP (IPv4 e IPv6), bloques de direcciones estilo CIDR, direcciones MAC, arrays.

Elementos que caracterizan a PostgreSQL pueden ser tanto la presencia de claves ajenas también denominadas Llaves ajenas o Claves Foráneas (*foreign keys*). Como la presencia de disparadores (*triggers*), con una amplia funcionalidad.

Otras características pueden ser las vistas, integridad transaccional, herencia de tablas, tipos de *datos y operaciones geométricas. Y por último el soporte para transacciones distribuidas.*

MYSQL SERVER

MySQL Server es la base de datos de código fuente abierto más usada del mundo desarrollado y proporcionado por MySQL AB. Su origen se debió a la búsqueda por parte de los fundadores de crear un manejador de bases de datos que fuera "rápido".

El servidor MySQL fue desarrollado originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes y ha estado siendo usado exitosamente en ambientes de producción sumamente exigentes por varios años.

Aunque se encuentra en desarrollo constante, el servidor MySQL ofrece hoy un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL un servidor bastante apropiado para acceder a bases de datos en Internet. Algunas características son:

• Escrito en C y C++.

•Trabaja bajo múltiples plataformas: AIX, Amiga, BSDI, HP-UX, Linux, Mac OS, Solaris, SunOS y Windows 9x, Me, NT, 2000, XP, 2003.

• Desarrollo de APIs para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.

• Procesos MultiHilo. Capacidad de trabajar servidores con varios procesadores.

• Provee sistema transaccional con la tabla Innodb.

• Velocidad cuando se manipula datos con el tipo de tabla Myisam.

• Velocidad en la utilización de joins y procesos de optimización.

• Soporta muchos tipos de columnas para las tablas: FLOAT, DOUBLE, CHAR,VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM y OpenGIS.

• Manejo de la memoria a través de manejo del buffer y cache.

INTERBASE

Este es un buen gestor de base de datos con 16 años de experiencia en el sector de las bases de datos comerciales, desarrollado y comercializado por la compañía Borland Software Corporation y actualmente desarrollado por su ex-filial CodeGear. Existen muchas herramientas de pago para Interbase. Casi todos los gestores de Backup soportan Interbase (como ArcServe).

Tiene también gestores remotos para Windows de muy alta calidad, como Maratón, todas estas herramientas hacen de Interbase un producto muy profesional preparado para cualquier proyecto medio que necesite de una base de datos realmente fiable. Interbase cuenta tiene la mayoría de funcionalidades de una base de datos comercial: triggers, tratamiento especial de blobs, backup Online, gran escalabilidad, bases de datos de solo lectura (para ponerlas en CDRom), integridad referencial en cascada o el autotunning. También dispone de soporte directo para PHP.

ORACLE

Sin duda alguna la actual unión entre Dell y Oracle constituye uno de los principales encuentros tecnológicos al servicio de las necesidades empresariales actuales, tras alcanzar más de 22.000 instalaciones de software Oracle en equipo Dell, las empresas han demostrado un sólido éxito en la tarea de entregar mayor beneficio empresarial a una amplia gama de clientes, entre ellos el Lighting Group de Acuity Brands, Electronic Arts, Menasha Corporation, el Centro Mercedes-Benz de Ayuda al Cliente y Precisión Response Corporación.

Dell ofrecerá una plataforma de almacenamiento optimizada para Oracle9i Database con Real Application Clusters para Red Hat(r) Linux Advanced Server y los entornos Microsoft Windows. Ahora, la pequeña y mediana empresa podrá aprovechar el rendimiento, la disponibilidad y la flexibilidad de escala de los clusters de servidores Dell PowerEdge respaldados por redes de almacenamiento Dell / EMC CX200 de nivel básico o bóvedas de discos Dell PowerVault SCSI. Dell Services está asociándose con Oracle(r) Consulting para ofrecer un conjunto de servicios profesionales que reducirá los costos y facilitará el despliegue a los clientes que migran de arquitecturas legado y propietarias de bases de datos a Oracle9i Database con Real Application Clusters en plataformas de servidores y almacenamiento Dell basadas en estándares.

Las propuestas de precio fijo incluyen: servicios de migración para los clientes que proceden de UNIX a Linux, servicios de implementación para ayudar a los clientes a desplegar rápidamente Oracle9i Database con Real Application Clusters, afinamiento del rendimiento y de la capacidad, así como replicación en espejo de las bases de datos y planificación de la recuperación de emergencia.

SQL SERVER 2000

SQL Server es el sistema de gestión de base de datos representativa de la firma mundialmente conocida Microsoft, En la actualidad, las compañías demandan una clase diferente de solución de base de datos. El rendimiento, la escalabilidad y la confiabilidad son esenciales y la anticipación al mercado es crítica. Aparte de estas cualidades empresariales fundamentales, SQL Server 2000 proporciona agilidad a sus operaciones de análisis y administración de datos al permitir a su organización adaptarse rápida y fácilmente para obtener ventaja competitiva en un entorno de cambios constantes.

Desde una perspectiva de administración de datos y análisis, resulta crítico transformar los datos sin procesar en inteligencia empresarial y aprovechar las oportunidades que presenta el Web. SQL Server 2000 es un paquete completo de base de datos y análisis de datos que abre las puertas al rápido desarrollo de una nueva generación de aplicaciones comerciales de nivel empresarial, que pueden proporcionar a su compañía una ventaja competitiva crítica. SQL Server 2000 ha obtenido importantes galardones en pruebas de referencia por su escalabilidad y velocidad. Es un producto de base de datos totalmente habilitado para Web que proporciona una compatibilidad fundamental con el Lenguaje de marcado extensible (XML, Extensible Markup Language) y la capacidad para realizar consultas en Internet y por encima del servidor de seguridad.

Otros SGBD comúnmente usados de carácter comercial son:

• DB2, Informix (IBM)

• dBase (dBI)

• Paradox (Borland)

• Access (MS)

• FoxPro (MS)

2.3.4. TOMA DE DECISIONES

El proyecto está basado en el paradigma de programación orientado a objetos. Es el más usado en la actualidad y el que mayor abanico de posibilidades funcionales me ofrece. Presenta como lenguaje de programación un lenguaje orientado a objetos acorde con el paradigma de programación, Java. Junto con el gestor de bases de datos MySQL.

Se ha elegido Java [6] como lenguaje de programación por características y ventajas como: simplicidad, atractivo, alto rendimiento y portabilidad.

El lenguaje Java es multiplataforma, es decir, no depende del sistema operativo en el que se haya compilado, es capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, sin importar el tipo de hardware, tal como reza el axioma de Java, ‘’write once, run anywhere’’. Cuando compilamos en java lo hacemos a un lenguaje más o menos unificado que se traduce en tiempo real por una maquina virtual, que no es más que un intérprete, desde el bitecode de java a las sentencias maquina de cada ordenador, pudiendo utilizarse en GNU/linux, en Windows, en MacOS, etc.

La portabilidad no se limita a ordenadores. Sun (la empresa que lo creó) pensó en java como una especie de lenguaje universal, y para ello creó una versión más pequeña y minimalista de su API pensada exclusivamente para dispositivos pequeños y móviles. Esta característica es muy importante ya que permitiría la futura migración de esta herramienta de repaso a diferentes arquitecturas y dispositivos móviles.

Comparando con C (o su equivalente en POO, C++), se escoge Java ya que, aunque java ha sido escrito en C/C++ , su sintaxis es infinitamente más clara e intuitiva, además desaparecen tanto los punteros (con los problemas asociados que causan) como la basura que generan los objetos y que hay que recoger a mano en C++ , de esto se encarga un recolector de basura especifico que se encuentra en la maquina virtual de java, evitando en gran medida las fugas de memoria.

Crear la interfaz gráfica para este proyecto de java es más fácil que con otros lenguajes. Java presenta bibliotecas y funciones predefinidas para facilitarnos el desarrollo gráfico. En java solo existen el AWT y el Swing, y una es casi una pequeña ampliación de la otra, es decir, a efectos prácticos, no se necesita más que un bloc de notas para empezar a programar, nada de paquetes gráficos que hay que instalar aparte, solo programar y listo, ya que esta todo instalado en el JDK.

JDBC (Java Database Connectivity) es el API para conectividad a Bases de datos mediante lenguaje java. Esta API permite conectarse a la BD para efectuar consultas, actualizaciones, etc., empleado para ello el Lenguaje Estructurado de Consulta, SQL (estándar de la industria para el acceso a bases de datos relacionales). JDBC brinda un conjunto de interfaces y clases para acceder a cualquier motor de base de datos que lo implemente. JDBC se abstrae de los detalles específicos del motor, permitiendo así conectarse prácticamente de la misma manera a cualquier base de datos.

Java y JDBC poseen una ventaja esencial con respecto a otros entornos de programación para base de datos: los programas desarrollados con java y JDBC son independientes de la plataforma y del fabricante.

Para MySQL, el driver se llama Connector/J y está actualmente en su versión 5.1. La elección de MySQL frente a otros gestores de datos se debe En primer lugar, el económico. MySQL es un producto OpenSource o productos licenciados libremente por los fabricantes. En segundo lugar, es un producto con reconocido prestigio, fiabilidad, velocidad, rendimiento, facilidad de administración y conexión con otros productos, bien documentados, con una buena evolución y soporte. También presenta facilidad para obtener información, con buenas herramientas y tutoriales.

Estos gestores de bases de datos OpenSource hace tiempo que dejaron de ser un experimento y ya son una alternativa real, son productos cada vez más evolucionados y como principales productos destacan MySQL y Postgre, pero Mysql es mucho más comercial y la comunidad es mucho más grande, por lo que se hace más fácil encontrar soporte en foros y en las páginas oficiales.

Por último, en cuanto a la plataforma, el proyecto está desarrollado en Windows, ya que es la plataforma más usada en el entorno escolar, pero sobre todo en el ámbito del hogar. Una de las características de Java, anteriormente señalada, es que es multiplataforma y por ello la portabilidad a otros sistemas operativos como Linux no supone ningún problema.

2.4 METODOLOGÍA EMPLEADA

Para desarrollar el sistema informático objetivo se ha seguido una metodología en cascada con las siguientes fases:

1. Análisis
2. Diseño
3. Codificación
4. Pruebas (unitarias y de integración)
5. Instalación y paso a Producción
6. Mantenimiento

Para representar estas etapas se usa un diagrama del cido de vida del software. Este diagrama define el orden de las tareas o actividades involucradas en el desarrollo, definiendo también la coordinación entre ellas.

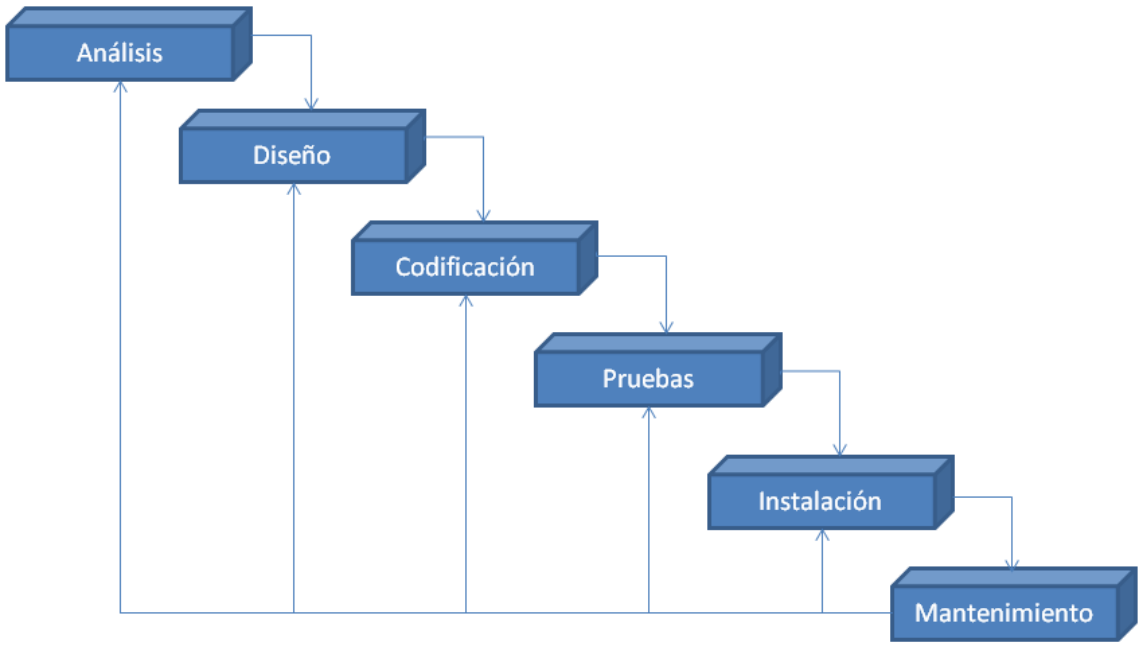

Figura 3: Esquema de ciclo de vida del software

En la Figura 3 se muestra el cido de vida del software a través del modelo en cascada realimentado. El modelo en cascada en algunas de sus variantes es actualmente uno de los más utilizados por su eficacia y simplicidad. El modelo cascada destaca en la representación de software pequeño y mediano, pero nunca (o muy rara vez) se usa en su forma pura. En el modelo en cascada siempre se produce alguna realimentación entre etapas, que no es

completamente predecible ni rígida. Esto da oportunidad al desarrollo de productos software en los cuales hay ciertas incertezas, cambios o evoluciones durante el cido de vida.

La metodología de Ingeniería del Software clásico se ha combinado con Diseño Centrado en el Usuario. El Diseño Centrado en el Usuario es una filosofía de diseño que tiene por objetivo la creación de productos que resuelvan necesidades concretas de sus usuarios finales, consiguiendo la mayor satisfacción y mejor experiencia de uso posible con el mínimo esfuerzo de su parte. Los pasos habituales del método de diseño centrado en el usuario:

1. Definir los usuarios
2. Analizar las necesidades de los usuarios
3. Diseñar y evaluar el artefacto
4. Evaluar el proceso

Durante el desarrollo no se ha seguido un modelo lineal, sino que se ha intentado involucrar al usuario desde el primer prototipo de forma incremental. El proyecto está centrado en el usuario, por lo que inicialmente debo conocer las necesidades de este usuario final, con el fin de diseñar un producto que satisfaga esas necesidades teniendo en cuanta las cualidades de dicho sujeto. La Figura 4 muestra el diagrama de diseño centrado en el usuario seguido en el desarrollo de esta aplicación con los 4 factores principales que se han tenido en cuenta: definir los usuarios, analizar las necesidades de los usuarios, diseñar y evaluar el artefacto y evaluar el proceso.

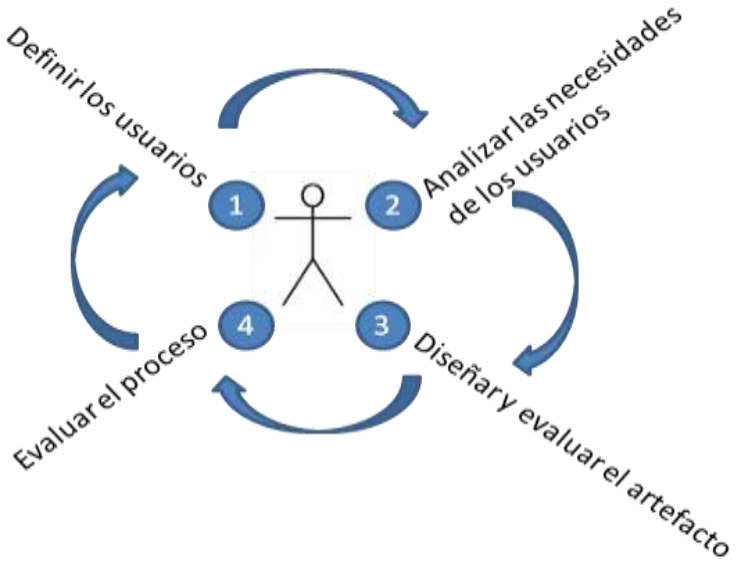

Figura 4: Diagrama del diseño Centrado en el Usuario

3. DESCRIPCIÓN INFORMÁTICA

3.1 ESPECIFICACIÓN

3.1.1 ANÁLISIS DE REQUISITOS

REQUISITOS FUNCIONALES

Son todos aquellos requisitos que especifican una funcionalidad que debe realizar un sistema o un componente. A continuación se muestra una lista con los requisitos identificados:

RF1. El administrador da de alta en la BBDD a los profesores permitiendo a éstos la utilización de la aplicación.

RF2. El profesor crea especificaciones de problemas de física y química a través de un editor de escritorio local que ira solicitando la información necesaria para la creación.

RF3. El profesor debe asignar a cada problema un enunciado, parámetros y pasos, para que la creación del problema sea correcta.

RF4. El profesor puede modificar el problema

- RF4.1. EL profesor puede modificar el enunciado
- RF4.2. EL profesor puede modificar los parámetros
- RF4.3. EL profesor puede modificar los pasos

RF5. El profesor puede borrar problemas creados por él mismo.

RF6. El profesor se encarga de dar de alta a los estudiantes, quienes podrán realizar los problemas creados por el profesor.

RF7. El profesor definirá el usuario del alumno, pero no su contraseña que será definida por cada alumno.

RF8. El estudiante deberá identificarse con usuario y contraseña para poder acceder a la aplicación mediante un diálogo inicial.

RF9. Los problemas que le aparecen al estudiante inicialmente se muestran de manera aleatoria, pero más adelante, a medida que vayamos obteniendo resultados, se mostrarán siguiendo el principio: “repasar más aquellos en los que más falle”.

RF10. El estudiante irá resolviendo los problemas en lenguaje natural guiado paso a paso por la aplicación.

RF11. Tanto la aplicación del estudiante como la del profesor presentarán una figura a la izquierda.

RF12. El estudiante podrá cerrar la aplicación en cualquier momento, y los datos deberán persistir en la BBDD.

REQUISITOS NO FUNCIONALES

Son los requisitos que describen no lo que hará el software, sino cómo lo hará.

RNF1. La aplicación será amigable según los principios IPO (Integración Persona – Ordenador).

RNF2. Se seguirá un diseño centrado en el usuario.

RNF3. La velocidad de respuesta de la aplicación no superara en ningún caso 1 minuto por interacción.

3.1.2 CASOS DE USO

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

Aunque este proyecto está dividido en dos partes diferenciadas que son la aplicación profesor y estudiante, la representación del esquema de casos de uso se va a definir de forma combinada. Se compondrá de una sola caja representando una única aplicación llamada profesor-estudiante.

El esquema usado es bastante representativo definiendo los casos de uso están en el interior de la caja del sistema, y los actores (profesor y estudiante) en el exterior. Los actores estarán unidos a los usos que pueden hacer en la aplicación.

El actor estudiante utiliza la herramienta únicamente para resolver problemas, mientras que el profesor, presenta un mayor abanico, pudiendo crear, modificar o borrar problemas, además de dar de alta alumnos que utilizarán la aplicación. La Figura 5 muestra el comportamiento de los actores en el diagrama de casos de uso.

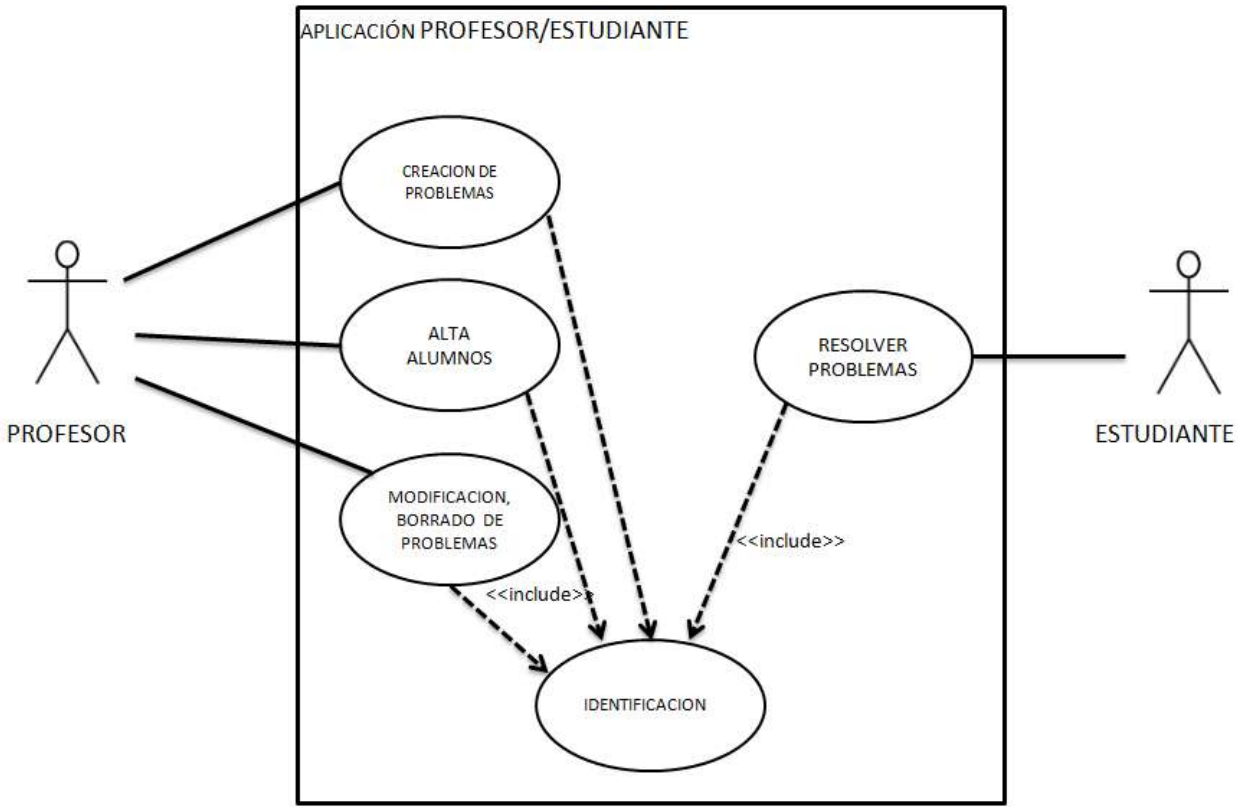

Figura 5: Diagrama de casos de uso

Para que los actores accedan a cualquier uso de la aplicación, antes deben de identificarse. En el diagrama anterior se observa como todos los casos de uso presentan relaciones <<include>>. Estas relaciones representan que si no se transita antes por el estado "identificación" no se podrá acceder al resto de casos.

3.1.3 DIAGRAMAS DE INTERACCIÓN

En el diagrama de interacción se muestra un patrón de interacción entre los objetos que forman el proyecto para cierto comportamiento. El diagrama de interacción captará el comportamiento de los caso de uso anteriores. Hay dos tipos de diagramas de interacción: diagramas de secuencia y diagramas de colaboración.

Para la representación del comportamiento el tipo de diagrama elegido será el diagrama de secuencia, debido a que es fácil apreciar el orden en el que ocurren las cosas y presenta gran simplicidad. Buscando la relación con los casos de uso se usará el modo "instancia". Un

diagrama de de secuencia en modo instancia describe un escenario específico que es una instancia de la ejecución de un caso de uso. Primero se describen las clases de análisis, que representan las clases principales del proyecto:

- Interfaz (Clase límite): encargado de la interacción entre los actores y el sistema. Representa la interfaz del sistema a nivel conceptual. Muestra los mensajes entre el actor y el sistema.

- Gestor Profesor/Alumno (Clase control): encargada de la relación entre objetos y de contener el flujo de control del caso de uso. Su cometido es la gestión interna siendo el nexo de unión entre objetos, pero sin interacción con actores o almacenamiento de información.

- Base Datos (Clase Entidad): se encarga de gestionar la información relativa a un caso de uso, almacena los datos que se usarán, siendo en este caso, el objeto que gestiona la Base de datos.

A partir de aquí se mostrarán algunos diagramas que se generarán para los casos de usos anteriores, donde se pueden visualizar cómo interaccionan las clases antes definidas. En concreto, se mostrará cómo se identifican cada uno de los actores interactuando con la aplicación a través de las 3 clases definidas anteriormente.

A continuación se muestra el Diagrama de interacción para el caso de uso Identificación del profesor. La Figura 6 muestra un diagrama en el que se describe como el profesor (actor) deberá introducir su nombre para que la herramienta le permita el acceso.

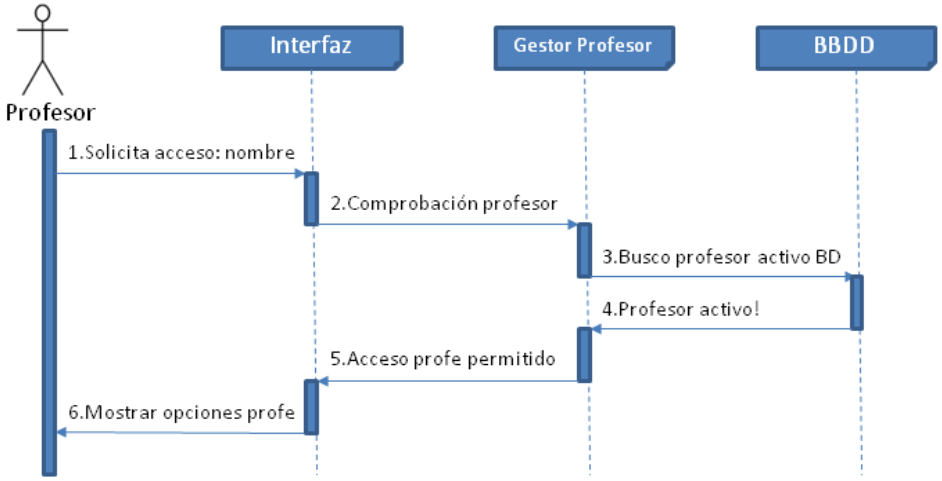

Figura 6: Diagrama de interacción Identificación del profesor

La Figura 7 recoge el diagrama de interacción para el caso de uso Identificación del estudiante, en el que el estudiante (actor) deberá introducir su nombre y contraseña para que la herramienta le permita el acceso para la resolución de los problemas.

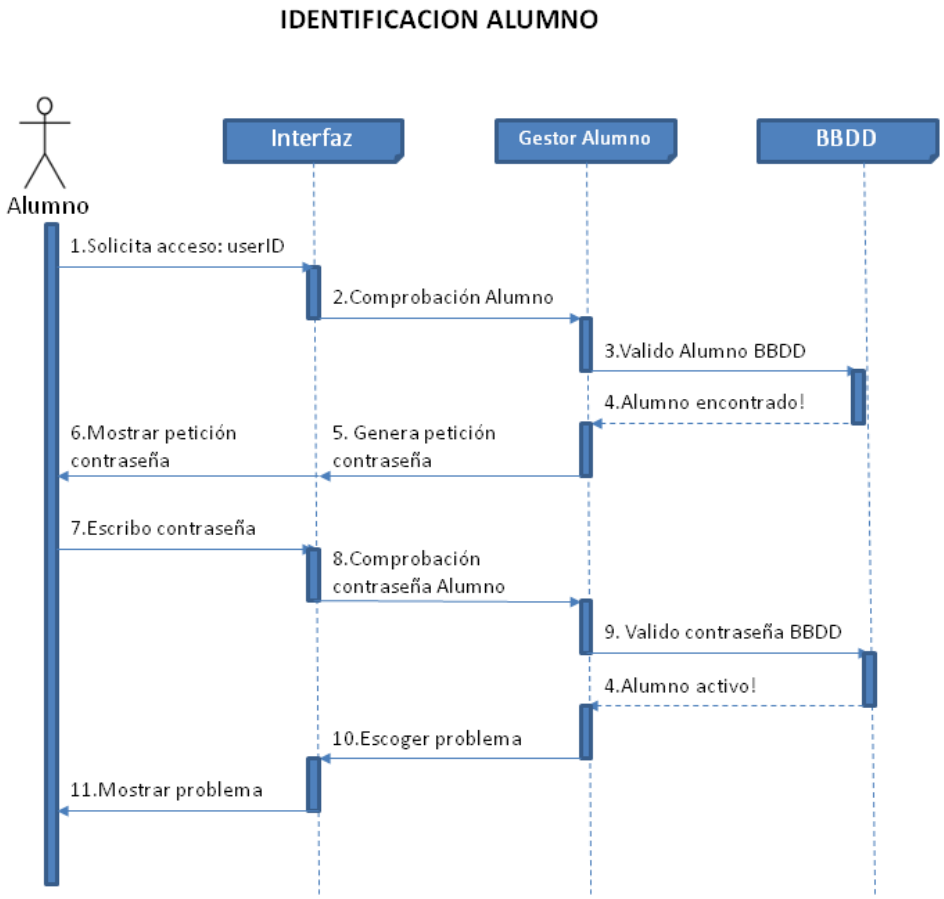

Figura 7: Diagrama de interacción Identificación del alumno

3.2. ARQUITECTURA DE ALTO NIVEL

El proyecto está basado en la arquitectura Modelo/Vista/Controlador, más conocida como MVC. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio. En este caso, el modelo se encargará de gestionar las respuestas según las entradas introducidas por el usuario. El controlador es el responsable de recibir los eventos de entrada desde la vista, más concretamente, cada vez que se pulse "Enter" o "Enviar". Por último la vista ofrece la interfaz del usuario. El Modelo, las Vistas y los Controladores se tratan como entidades separadas, esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

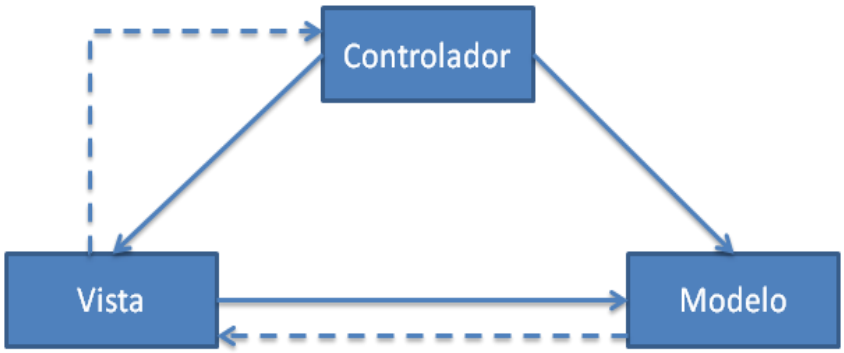

Figura 8: Representación del MVC

En la Figura 8 se observan las relaciones entre los distintos módulos de la arquitectura MVC, el flujo habitual del sistema suele ser el siguiente:

1. El usuario interactúa con la interfaz de usuario, en este caso, mediante la pulsación de la tecla "enter" o mediante un click en el botón "enviar".

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, en este caso a través de ActionListener.

3. El controlador accede al modelo para actualizarlo, pasándole los datos para que el modelo genere una respuesta.

4. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo.

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

Una vez se ha entendido como funciona un sistema basado en la arquitectura MVC, se definirá como se va a relacionar esta arquitectura con la del proyecto.

Se utiliza un diagrama de módulos para mostrar la asignación de clases y objetos a módulos en el diseño físico de un sistema. Un solo diagrama de módulos representa una vista de la estructura de módulos de un sistema. Los dos elementos esenciales de un diagrama de módulos son los módulos y sus dependencias, representando con cajas cada clase y mediante flechas las relaciones. Además, se muestra la relación que estos módulos puedan tener con la base de datos.

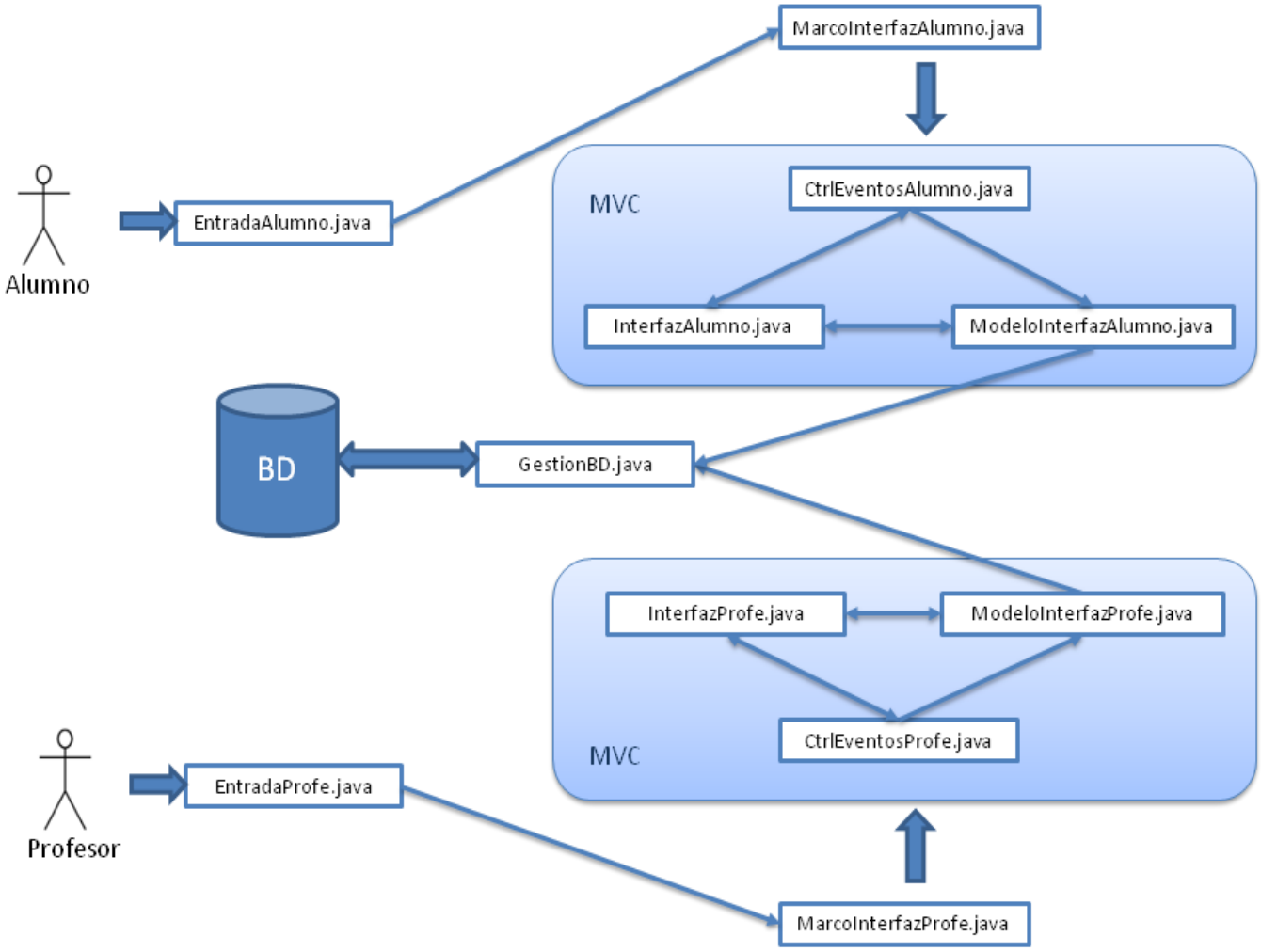

Figura 9: Diagrama de módulos

En resumen, tras analizar el diagrama de módulos, se extrae que la única clase que se relaciona con la base de datos es GestionBD.java. Para entender la relación entre la arquitectura MVC y

el resto de módulos de la aplicación se elegirá uno de los actores, por ejemplo el profesor, aunque el diagrama sea idéntico para ambos. La clase ModeloInterfazProfe.java es la que se encarga de gestionar todo sistema, es decir, todo lo que la aplicación realiza tras el lanzamiento de un evento, siendo el único que puede acceder a la base de datos a través de GestionBD.java. Para la recepción de esos eventos esta la clase CtrlEventosProfe.java. Y para mostrar toda la información al usuario se utilizará la clase InterfazProfe.java.

Estas tres clases forman el patrón MVC, Modelo es ModeloInterfazProfe.java, Vista es InterfazProfe.java y Controlador es CtrlEventosProfe.java. A parte se tiene la clase MarcoInterfazProfe.java que se encarga de la invocación de los 3 objetos que forman el MVC.

3.3 DIAGRAMA E/R

Los diagramas de Entidad - Relación son un lenguaje gráfico utilizado para describir conceptos, se utilizará para describir cómo se almacena la información en la base de datos, la cual interactuará con la aplicación.

Como su nombre indica están formados por entidades y relaciones o dependencias entre estas entidades. Cada entidad va a representar un objeto del mundo real. En este caso hay definidas varias entidades como: estudiante, profesor, etc. Cada una de ellas con una serie de atributos.

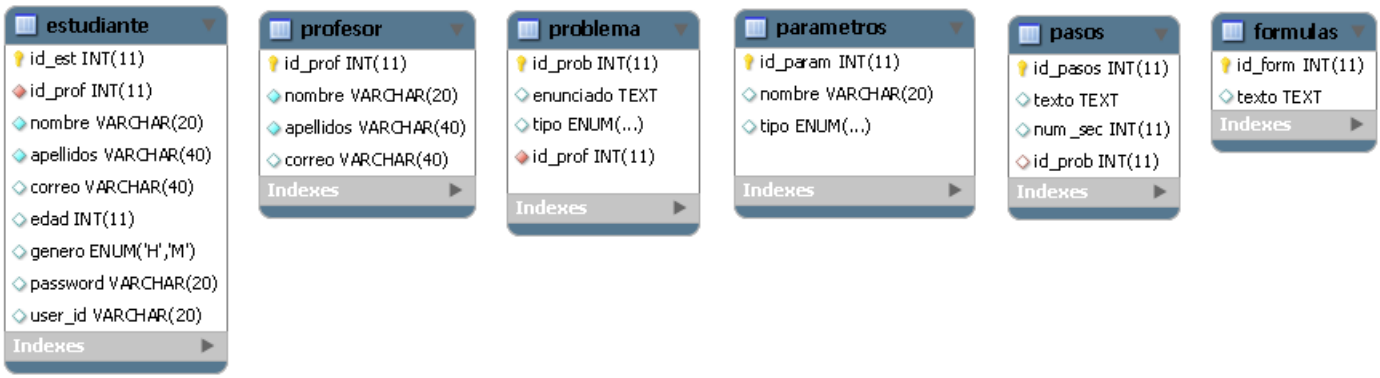

Figura 10: Entidades del modelo E-R

En la Figura 10 se puede observar las entidades que representan los objetos reales. Estudiante representa cada estudiante con su nombre, edad, correo, etc. Profesor representa a cada profesor con nombre, apellidos, etc. Problema almacena cada problema de física y química. Para almacenar los parámetros de cada problema usamos la tabla parámetros, dentro de cada problema se tiene una serie de pasos que se almacenan en la tabla pasos y por último cada fórmula necesaria en la resolución del problema se almacena en formulas.

ESQUEMA DEL DIAGRAMA DE ENTIDAD – RELACION DE MI BASE DE DATOS

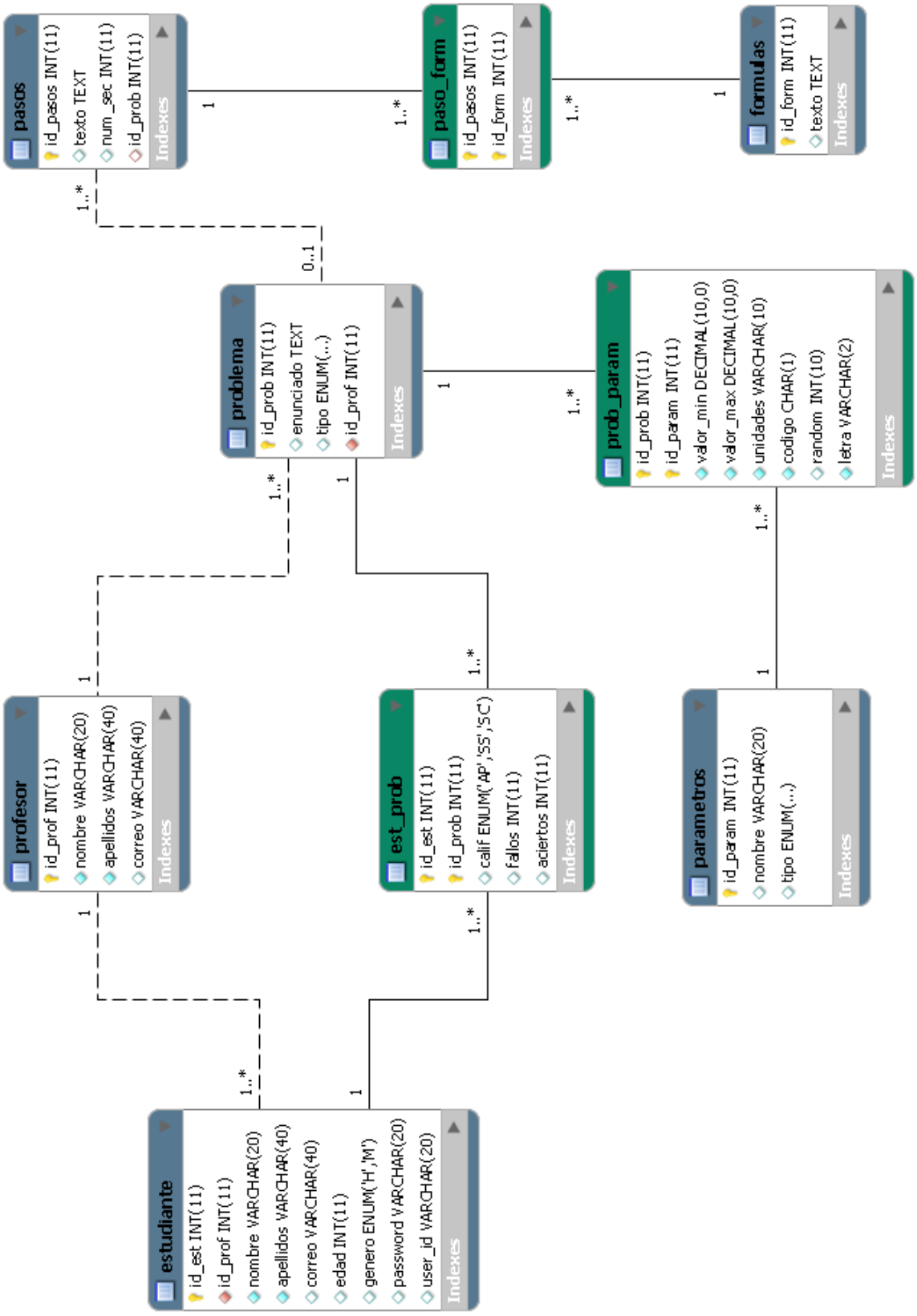

Figura 11: Diagrama de Entidad-Relación (E-R)

Esto en cuanto a las entidades, consideradas tablas y representadas con etiqueta azul, pero las relaciones manejan la interacción de estas entidades. Las relaciones describen cierta dependencia entre entidades o permiten la asociación de las mismas. A cada relación se le asigna un nombre para poder distinguirla de las demás y saber su función dentro del modelo entidad-relación. Otra característica es el grado de relación, siendo las de grado 1 relaciones que solo relacionan una entidad consigo misma. Las de grado 2 son relaciones que asocian dos entidades distintas, y las de grado n (* en la figura 11), que se trata de relaciones que unen más de dos entidades.

Las relaciones se representas gráficamente con rombos, dentro de ellas se coloca el nombre de la relación. En este caso se muestra con etiqueta verde las que además presentan atributos y con un simple nombre de relación o rombo las que no. En el caso de las relaciones con atributos se almacenarán cono una tabla más.

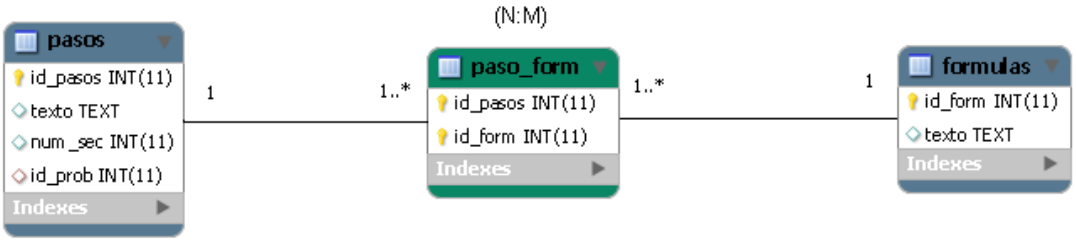

Figura 12: Representación relación

Otra característica es el tipo de correspondencia entre dos relaciones:

- 1:1. Uno a uno, a cada ocurrencia de una entidad le corresponde como máximo una ocurrencia de la otra entidad relacionada.

- 1:N. Uno a Mucho, a cada ocurrencia de la entidad A le pueden corresponder varias de la entidad B.

- N:M. Muchos a muchos, cada ocurrencia de una entidad puede contener varias de la otra entidad relacionada y viceversa. Aunque de este tipo mi diagrama no presenta ninguna.

Para finalizar las características de las relaciones se define la cardinalidad que representa el número máximo y mínimo de ocurrencias de cada tipo de entidad. Se constituye con los valores máximo y mínimo encerrados entre paréntesis encima de la relación (Máximo, mínimo). Como se observa en la Figura 12, con una cardinalidad de (N:M).

Como último concepto del diagrama están los atributos. Se define como cada una de las propiedades de una entidad o relación. Cada atributo tiene un nombre y todos los posibles valores que puede tener. Dentro de una entidad tiene que haber un atributo principal que identifica a la entidad y su valor tiene que ser único. Un ejemplo de atributo principal seria el id_pasos dentro de la entidad pasos, según la Figura 12.

Para describir mejor las relaciones de la base de datos, se muestra el sistema de almacenamiento utilizado en el proyecto a través del modelo relacional estático. Propuesto por E. Codd, es un modelo basado en la teoría matemática de las relaciones, donde los datos se estructuran lógicamente, en forma de relaciones (tablas) siendo un objetivo fundamental del modelo mantener la independencia de esta estructura lógica respecto al modo de almacenamiento y otras características de tipo físico.

La forma de representación será estática. Codd introduce el concepto de relación como estructura básica del modelo. Todos los datos de una base de datos se representan en forma de relaciones cuyo contenido varía con el tiempo. Una relación en terminología relacional, es un conjunto de filas (tuplas) con unas determinadas características.

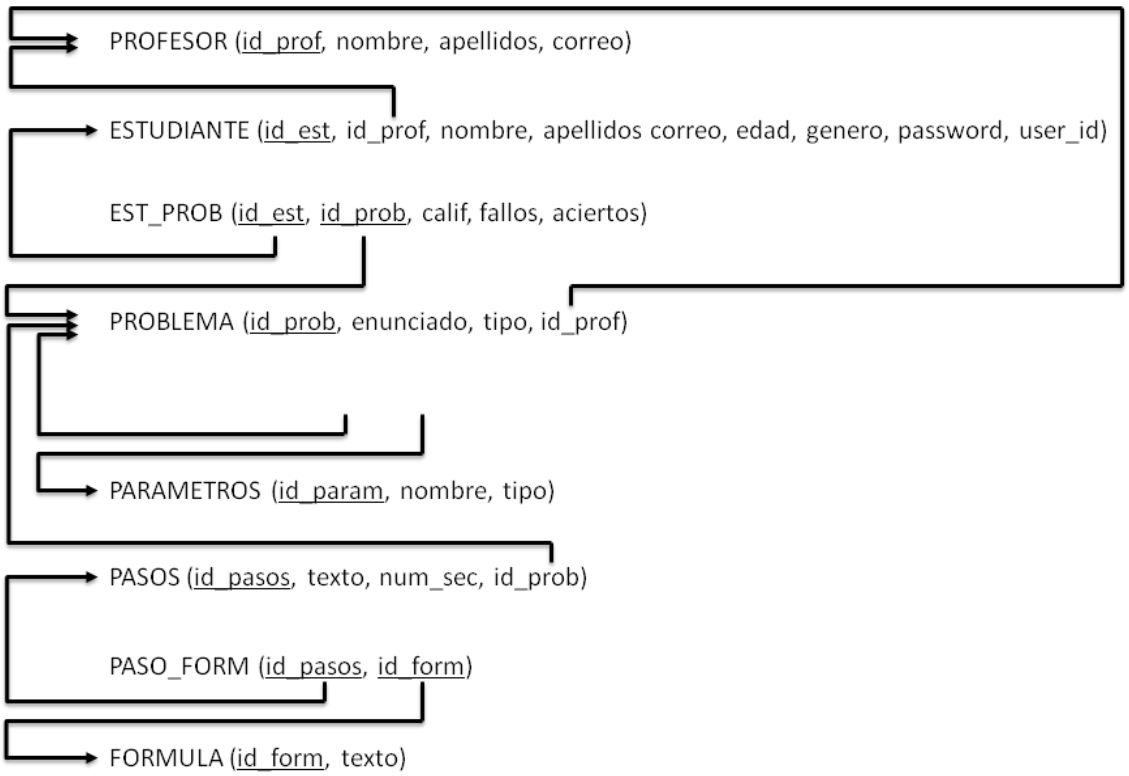

Figura 13: Modelo relacional estático.

Seguidamente se describe las tablas que forman la base de datos del proyecto, indicando cual será su uso y los atributos que las componen. Acompañadas de un ejemplo de fila como ejemplo ilustrativo de los tipos de datos que almacenan.

PROFESOR: Tabla que almacena los datos personales relativos a cada profesor (nombre, apellidos, correo), donde su clave principal es id_prof.

id_prof	nombre	apellidos	correo
1	Soraya	Trejo Molina	s.trejo@urjc.es
2	Antonio	Boza España	a.boza.es@gmail.com

Tabla 1: tabla profesor.

ESTUDIANTE: Tabla que almacena los datos personales referentes a cada alumno (nombre, apellidos, correo, edad, genero), el usuario y contraseña de acceso a la aplicación (user_id, password) y el identificador del profesor (id_prof, clave ajena) que reflejará la relación estudiante-profesor, a través de la que el alumno podrá realizar los problemas definidos por el profesor indicado. La clave primaria es id_est.

id_est	id_prof	nombre	apellidos	correo	edad	gen...	password	user_id
14	2	antonio	boza españa	NULL	NULL	NULL	boza85	aboza
15	2	daniel	boza españa	NULL	NULL	NULL	dani92	dboza

Tabla 2: tabla estudiante.

PROBLEMA: Tabla que almacena la descripción inicial del problema con el enunciado y el tipo de problema (enunciado, tipo), además del profesor que lo ha realizado (id_prof, clave ajena), donde la clave principal es id_est.

id_prob	enunciado	tipo	id_prof
92	Formulacion de Hidróxido de litio	Quimica	2
93	¿cual es la formula del oxido de hierro?	Quimica	2
94	Un coche lleva una velocidad de X km/...	Fisica	2

Tabla 3: tabla problema.

EST_PROB: Tabla que define la relación entre estudiante y problema, almacenando también la calificación(SS,SC,AP), número de fallos y aciertos (calif, fallos, aciertos) de cada alumno en cada problema, siendo las claves primarias id_est y id_prob.

id_est	id_prob	calif	fallos	aciertos
14	92	AP	4	3
14	93	SC	0	0
14	94	SS	1	0

Tabla 4: tabla – relación est_prob.

PARAMETROS: Tabla que define cada parámetro en cuanto a nombre y tipo, pudiendo ser decimal o entero. Su clave primaria es id_param.

id_param	nombre	tipo
71	velocidad	decimal
72	tiempo	decimal

Tabla 5: tabla problema.

PROB_PARAM: Tabla que define la relación entre problemas y sus parámetros. Define el valor mínimo y máximo que puede alcanzar cada parámetro, sus unidades, el código que se utilizará. Para ser representado en el enunciado, su valor y la letra que identifica el parámetro a la hora de la sustitución en la formula (valor_min, valor_max, unidades, código, random y letra respectivamente). Las claves primarias son id_prob, id_param.

id_prob	id_param	valor_min	valor_max	unidades	codigo	random	letra
94	71	1	200	km/h	X	159	v
94	72	0	1000	min	Y	812	t

Tabla 6: tabla – relación prob_param.

PASOS: Tabla que define los pasos de los problemas con el contenido del paso que se mostrará al alumno, el numero del paso y el problema al que pertenece el paso que es clave ajena (texto, num_sec, id_prob respectivamente). La clave primaria es id_pasos.

id_pasos	texto	num_sec	id_prob
89	recordad que el litio es LI	1	92
90	combinar con hidroxido	2	92
91	la solucion es:	1	93
92	hay que comprobar los parametros de en...	1	94
93	transformar las unidades	2	94
94	aplicar la formula espacio = velocidad * ti...	3	94

Tabla 7: tabla pasos.

FORMULA: Tabla que almacena las formulas de la que saldrá el resultado en el caso de problemas de física y la nomenclatura de un compuesto en el caso de química bajo la columna texto. La clave primaria es id_form.

id_form	texto
60	LiOH
61	FeO
62	x=v*t

Tabla 8: tabla fórmula.

PASO_FORM: Tabla que define la relación entre los pasos y la formula que se muestre en cada paso. Siendo las claves primarias las referencias de pasos y problemas (id_pasos, id_form).

id_pasos	id_form
90	60
91	61

Tabla 9: tabla – relación paso_form.

3.4. ALGORITMO

En esta sección se expondrá el algoritmo utilizado con ayuda de pantallazos y diagramas de estados. El algoritmo principal está situado en el módulo ModeloInterfazProfe.java en el caso del profesor, y en ModeloInterfazAlumno.java en el caso del algoritmo del alumno. Estos dos módulos comparten estructura pero presentan diferencias en el diseño. A través de ellos se controla el funcionamiento de la aplicación. Los demás módulos son menos complejos y una descripción basta para comprender su comportamiento.

Estos 2 módulos (ModeloInterfazProfe.java y ModeloInterfazAlumno.java) aunque están definidos para profesores y alumnos de forma separada, ambos comparten la principal función de la aplicación que es la de agente conversacional pedagógico, objetivo del proyecto.

Tanto para el profesor como para el alumno, el algoritmo cuenta con un modulo principal llamado RecogidaDatos(String), en el que se recibe la información introducida por el usuario y dependiendo del estado que esté transitando, pasará a otro o seguirá en el mismo, hasta finalizar la acción requerida.

El envío de esta información por parte del usuario hacia la aplicación se hará mediante la ventana estilo Messenger anteriormente nombrada, que presenta una superficie para que el usuario introduzca el texto que quiera. Este texto será analizado por el algoritmo cada vez que se pulse la tecla “Enter” o se haga click en el botón “Enviar” dispuesto en la ventana.

Es preciso resaltar que la introducción del texto “salir” o “cerrar” por parte del usuario, provocará un cierre de sesión del usuario o el cierre de la aplicación respectivamente. A partir de aquí, la parte alumno y profesor se empiezan a diferenciar. Por eso tratará de forma distinta cada uno de los algoritmos.

En el anexo A se muestra un esquema práctico de generación de problema donde se explica los estados por donde transita paso a paso.

ALGORITMO PROFESOR

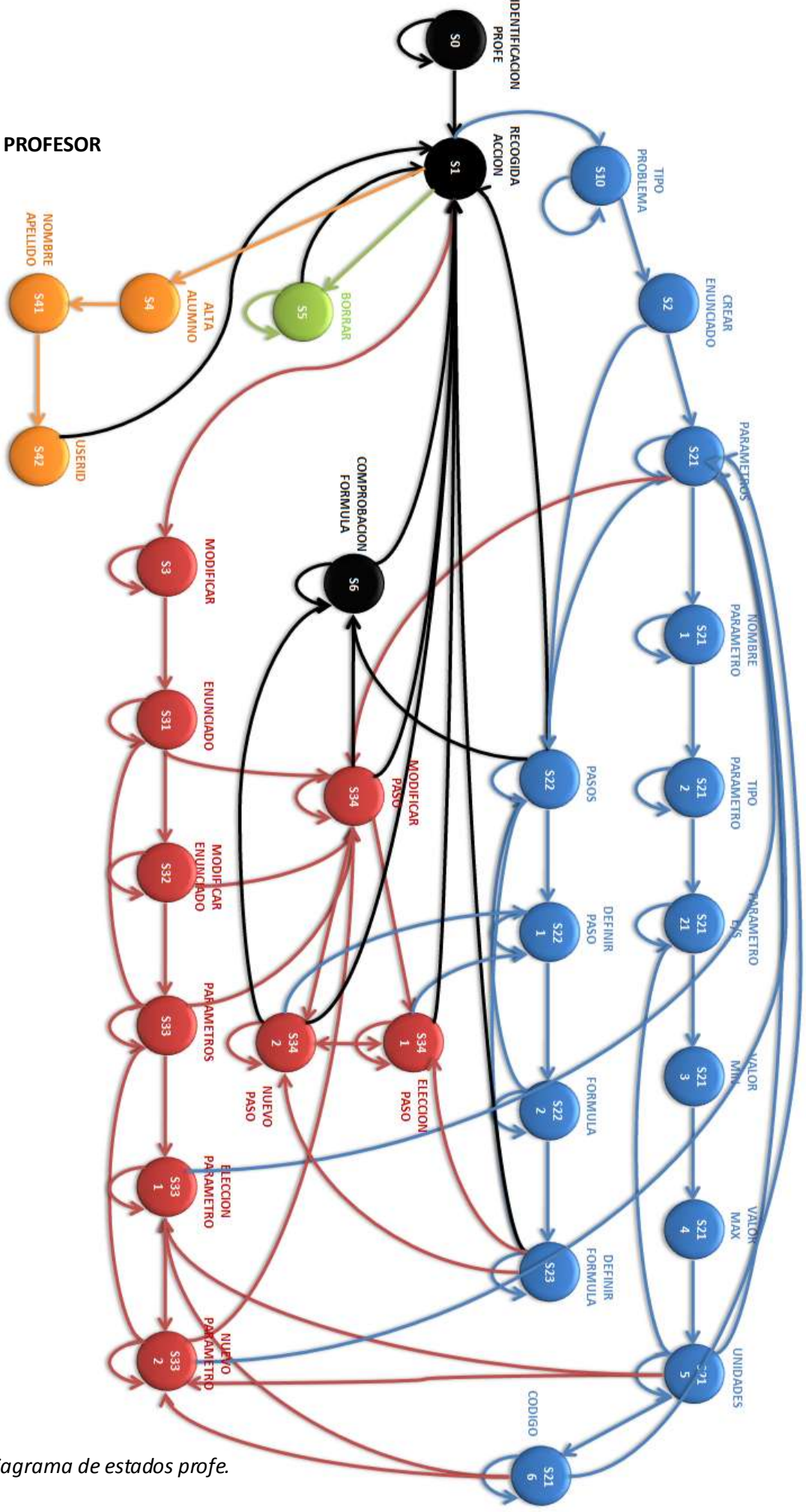

Figura 14: Diagrama de estados profe.

El anterior diagrama de estados muestra el comportamiento del algoritmo en el caso del profesor. Mediante la transición entre estados se mostrará el comportamiento de la herramienta según las situaciones. (ModeloInterfazProfe.java)

Tras la identificación correcta del profesor (AccesoProfe())en la que se requerirá el nombre de un profesor activo en la base de datos, se pasa al estado de recogida de acción donde se elegirá una de estas opciones: crear problema, modificar problema, borrar problema y dar de alta alumnos. Se definen con los colores azul, rojo, verde y naranja respectivamente en el diagrama de estados. En color negro los estados comunes a todas opciones.

Se empezará con la creación del problema una vez el usuario ha introducido la palabra "CREAR" durante el estado "S1". Se pasa al estado "S10" donde el usuario elegirá realizar un problema de física o química introduciendo 1 o 2 respectivamente, ya que la creación de un problema de física respecto a uno de química es diferente. Durante el recorrido del grafo se notarán las diferencias.

Tras la selección de física o química, se pedirá al profesor que introduzca el enunciado del problema, pasando al estado "S2". En "S2" se almacena el enunciado, pasando a introducir los parámetros ("S21") en el caso de física o directamente los pasos ("S22") en el caso de química, la herramienta permite no definir parámetros en un problema de física, pero de hacerlo, más adelante será cancelada la creación del problema.

La introducción de los parámetros se realiza siguiendo una serie de pasos para recorrer los estados y completar la creación. Se introduce el nombre del parámetro ("S211"), se define su tipo ("S212") ya sea entero o decimal. También se debe definir si es un parámetro de entrada o salida ("S2121"), si es de entrada habrá que darle un rango para generar el número aleatorio a la hora de generar el problema. El valor aleatorio tendrá un valor mínimo y otro máximo ("S213" y "S214") que el usuario introducirá. Pero si es de salida, la introducción del rango no se realiza, ya que no hay que generar valores aleatorios. El siguiente paso es la introducción de las unidades ("S215"), que mas adelante serán comprobadas. Por último si es un parámetro de entrada se le asignará un código para que sea sustituido en el enunciado ("S216").

Después de los parámetros se introducen los pasos, una vez se haya decidido la no introducción de nuevos parámetros. En el estado "S221" se introducen los pasos. Para cada paso se insertará o no una formula ("S221") y de ser afirmativo la respuesta a la inserción, se insertará la fórmula en el estado "S23". Una vez que no se quieran introducir más pasos se pasa al estado "S6" para comprobar si el problema está bien especificado, si es así, habremos finalizado la creación. De estar mal especificado, el problema se borrará. Los problemas que se almacenen deben sestar perfectamente definidos.

La siguiente opción es "MODIFICAR" ("S3"), que se encarga de la modificación de problemas almacenados en BBDD. Para modificar irán apareciendo problemas hasta que uno sea el que se quiera modificar pasando al siguiente estado ("S31") para decidir si se modifica o no el enunciado, de ser "si" la respuesta, se almacenará en el estado "S32". Teniendo en cuenta que si se acaba la lista de problemas almacenados volveré al estado "S1" mostrando el menú, como podemos observar en la Figura 15. Se realizará el mismo protocolo que en la creación para parámetros y pasos, pudiéndose en estos dos añadir nuevos e ir eligiendo cuales quiero modificar. A la hora de modificar usaré los estados de creación del problema anteriores volviendo al estado "S21"en el caso de parámetros y "S22" en el caso de pasos. Al igual que en la creación antes de dar por terminado el problema se debe pasar por el estado "S6" que comprobará si el problema está bien definido.

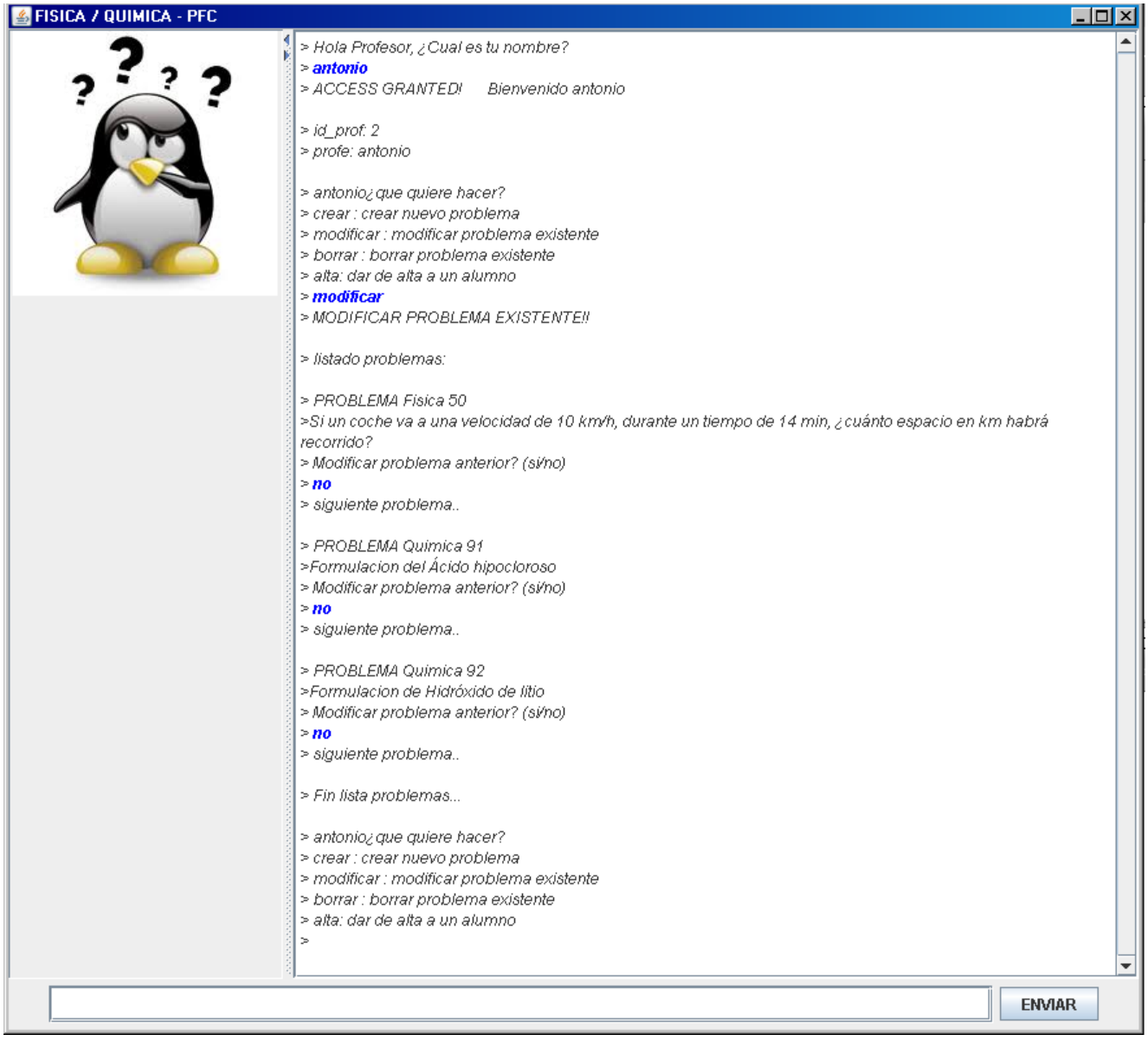

Figura 15: lista de problemas a modificar.

La opción de "BORRAR" ("S5"), que eliminará de la base de datos los problemas seleccionados, empezará como en la modificación, mostrando uno a uno cada problema para elegir el que quiera eliminar.

Y por último la opción "ALTA ALUMNO", en la que se dará acceso a los alumnos que vayan a utilizar la herramienta. Introduciendo nombre, apellidos y userID para que el alumno pueda acceder a la herramienta recorriendo los estados "S5", "S51" y "S52" respectivamente.

Tras ejecutar la opción requerida siempre se vuelve al punto de partida, estado "S1", donde se volverán a mostrar todas las posibles opciones de nuevo, formando un bucle que solo se romperá con la introducción de "salir", "cerrar" o haciendo click en la "X" de la ventana.

ALGORITMO ESTUDIANTE (ModeloInterfazAlumno.java)

Al igual que para el profesor, el alumno para acceder a la aplicación necesita identificarse, para ello en el estado "S0" el alumno debe introducir el nombre de usuario (userID), el cual ha debido de asignar anteriormente el profesor.

A partir de aquí tenemos dos caminos, dependiendo si es la primera vez que el alumno accede a la aplicación o no. Si es la primera vez, aun no tiene asignada ninguna contraseña. Por lo tanto deberá asignar la contraseña, que solo conocerá el usuario, y deberá utilizar cada vez que quiera entrar a resolver problemas. La contraseña se almacenará en el estado "S2" volviendo al estado "S0" para realizar de nuevo el acceso teniendo ya la contraseña almacenada.

El camino alternativo es la transición al estado "S1", que se produce cuando el alumno ya ha definido su contraseña anteriormente. Una vez que es autorizado para entrar en la aplicación deberá decidir si realiza ejercicios de física o química en el estado "S20", para después pasar a la elección del problema de la categoría antes indicada ("S3"). A través del método "SeleccionarProblema(int estudiante, int profesor, String tipo)", que se encargará de elegir un problema del tipo seleccionado, basándose en las calificaciones que el alumno haya conseguido anteriormente. El algoritmo de este método funciona eligiendo primero los problemas con más fallos si tiene problemas suspensos, si no tiene fallos pero tiene problemas sin calificar estos serían la segunda opción y como tercera, si no tiene suspensos y son todo aprobados, elegirá los problemas con menos aciertos, es decir, el alumno hará mayor esfuerzo en los problemas que peor se le den. A continuación se muestra en la figura 16 como se seleccionan tanto problemas de física como de química siguiendo esta doctrina.

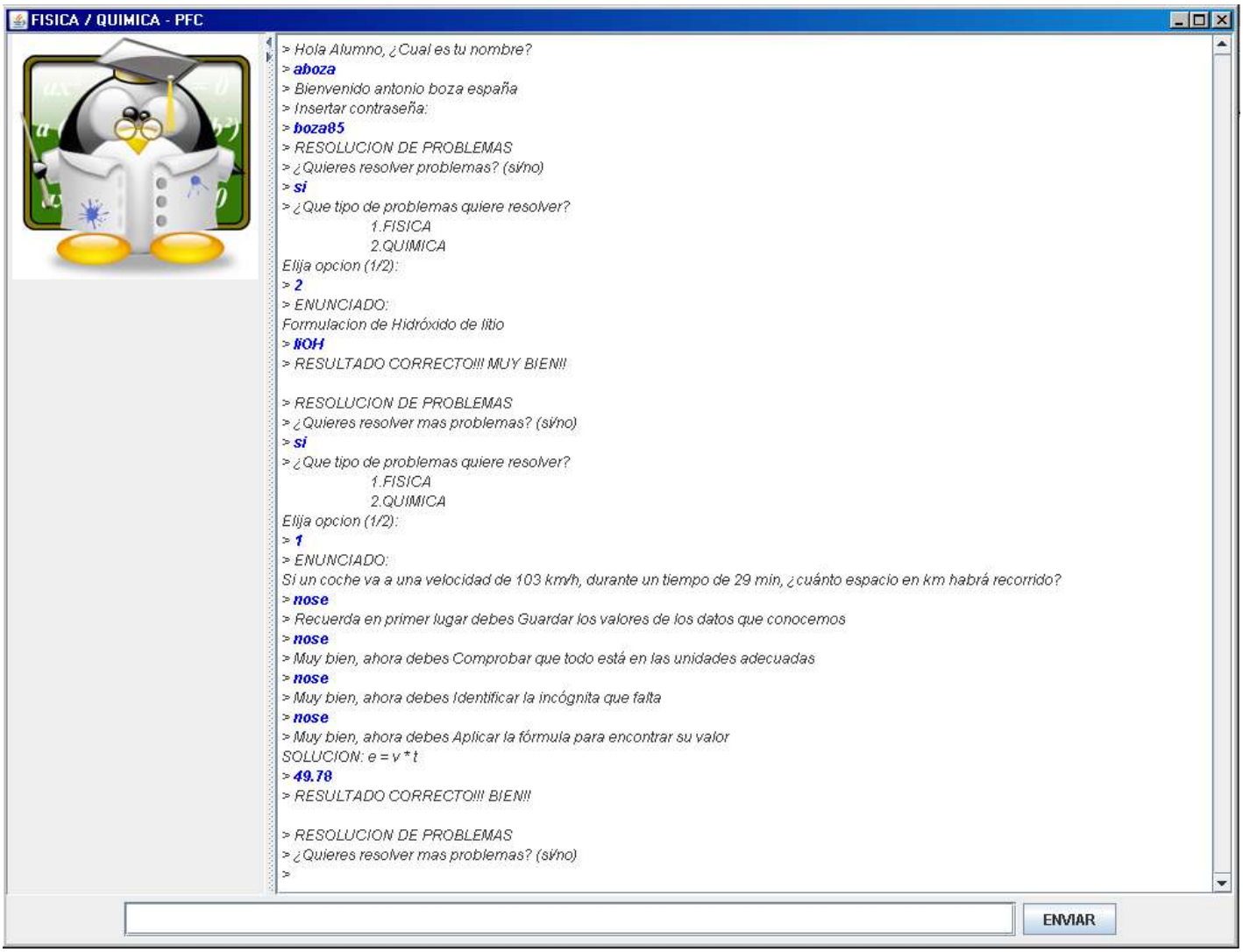

Figura 16: Ejemplo selección de problemas alumno

Durante este estado "S3" también se le mostrará el enunciado del problema para pasar al estado "S4" y esperar la solución correcta. Dependiendo si el problema es de física o química el algoritmo actuará de forma distinta. En el caso de física, al mostrar el enunciado, el algoritmo calcula la solución y espera la respuesta del alumno. Cada vez que el alumno introduzca una respuesta se comprobará si la solución introducida es igual a la anteriormente calculada. Si el alumno no acierta a la primera se considerará suspenso y se irán mostrando pasos hasta que el alumno acierte o se acaben los pasos mostrándose la solución acompañada de la fórmula. Para que se considere aprobado debe acertar a la primera, realizándose una comparación de números reales, siendo necesarias para acertar que el error producido en la solución insertada por el alumno sea menor que 0.1 respecto del resultado calculado por la herramienta.

Sin embargo el algoritmo de obtención de resultado en los problemas de química es más sencillo ya que el profesor debe introducir la solución, luego no hay nada que calcular, simplemente comparar los strings e ir mostrando los pasos como para física.

DIAGRAMA DE ESTADOS ALUMNO

Figura 17: Diagrama de estado alumno

En el Anexo B se muestra un esquema práctico de resolución de problema donde se explica los estados por donde transita paso a paso.

Existen otros módulos, que aunque de menor complejidad también tienen una importante función:

GestionBD.java

Es el más extenso después de los anteriores, su función principal es la de acceder a la base de datos tanto para insertar, como para borrar o extraer información necesaria en la aplicación. Tiene como método más complejo el "CalcularSolucion(int prob, String[] tipos, String[] parametros, String[] unidades, double[] valores, String[] nombreParam, int num, String operacion)", que se encarga de obtener la solución a los problemas de física creados de manera aleatoria. En resumen este módulo es el nexo de unión entre la aplicación y la base de datos siendo compartido tanto por PROFESOR como por ALUMNO.

InterfazALumno.java InterfazProfe.java

Este módulo se encarga de construir la interfaz visual para alumnos y profesores respectivamente. Diferenciándose entre ellas simplemente en la foto que hace referencia a quien está al otro lado para causar el efecto persona anteriormente mencionado.

En la Figura 18 se puede ver cómo tanto la interfaz del profesor como la del estudiante siguen la estructura tipo chat del Messenger para facilitar su manejo:

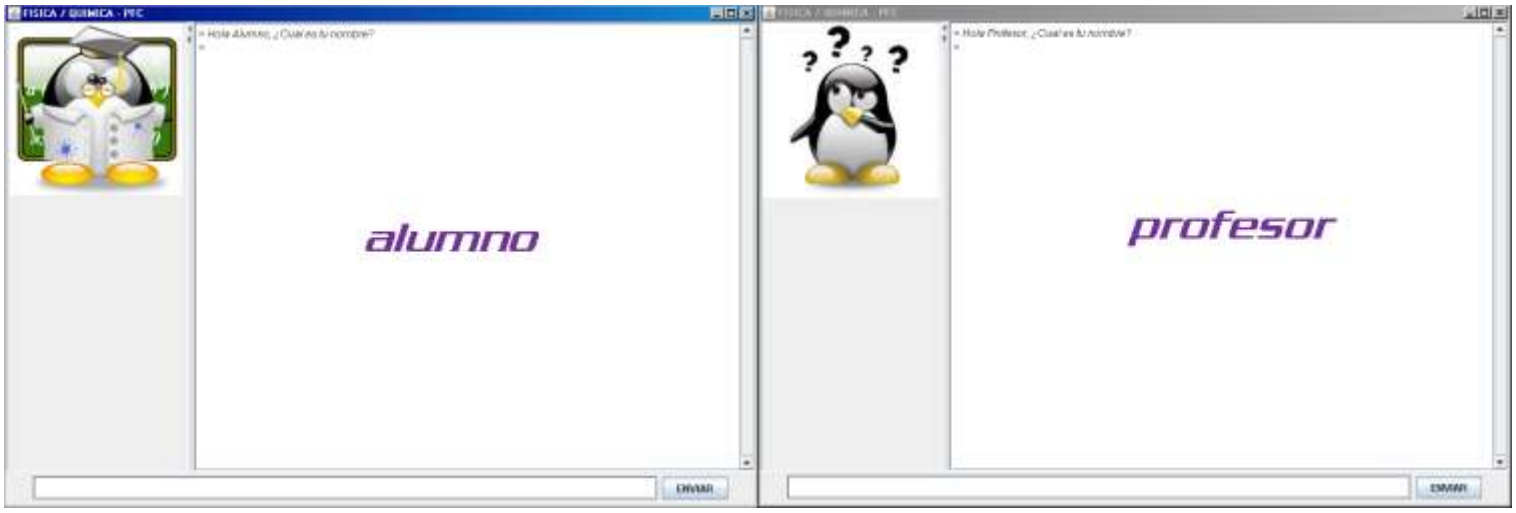

Figura 18: Diferencia Interfaz Alumno/Profe

CrlEventosProfe.java CtrlEventosAlumno.java

Este módulo se encarga de procesar la información relativa a la producción de un evento, este evento como anteriormente se dijo, se debe a la pulsación de la tecla "Enter" captada como "\n" o el click en el botón "Enviar" de la interfaz. El algoritmo es el siguiente:

Si (evento = "enviar") o (evento = "\n") entonces
Empezar
Leer texto introducido por usuario.
Representar lo escrito en la interfaz.
Enviar la información al modeloInterfaz según sea alumno o profe.
Borrar el texto introducido.
Fin
Sino
Error
Fin Si

MarcoInterfazProfe.java MarcoInterfazAlumno.java

Estos módulos tienen como función relacionar los distintos módulos siguiendo la arquitectura del Modelo – Vista – Controlador.

EntradaProfe.java EntardaAlumno.java

En estos módulos reside el método MAIN, se crearán todos los objetos referenciando los módulos anteriores y se arrancará la aplicación con una bienvenida y preguntando quién quiere acceder a la herramienta.

4. PRUEBAS

En este capítulo se presentan las pruebas realizadas.

4.1. PRUEBAS UNITARIAS

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esta prueba se realiza a nivel de módulo independiente. El propósito es encontrar discrepancias entre la especificación de la interfaz del módulo y su comportamiento real. En concreto se muestran dos tipos de pruebas unitarias, de caja blanca y caja negra.

4.1.1 CAJA BLANCA

Es un tipo de pruebas de software que se realiza sobre las funciones internas de un módulo. La caja es "blanca", se puede ver lo que hay dentro. Se va a realizar únicamente sobre el módulo más complejo de la aplicación que en este caso va a ser ModeloInterfazProfesor.java, cuya función es la de guiar en la creación, borrado y mantenimiento de problemas. El módulo se define a través de un solo método: *public void RecogidaDatos(String datos).* En concreto se va a visualizar un ejemplo del funcionamiento de este módulo, la creación de un problema de física.

Bloque 1 Identificación profesor

El profesor debe introducir su nombre para acceder a la aplicación, de no ser correcto no se dará acceso y se volverá a pedir el nombre.

Bloque 2 Crear problema

El profesor deberá elegir la opción crear, de no escribirlo correctamente se volverá a pedir que introduzca la frase “crear”.

Bloque 3 Tipo de problema FISICA

Aparecerán dos opciones, física o química y el profesor deberá introducir un “1”, de no introducir lo requerido (1/2) volverá a pedir el tipo de problema a crear.

Bloque 4 Introducción enunciado

El profesor deberá escribir el enunciado, teniendo en cuenta que donde quiera que se representen los valores de los parámetros deberá especificar una letra clave para que sea sustituido por el valor aleatorio al mostrar el problema al alumno. De no escribir nada habrá que volver a introducir el enunciado.

Bloque 5 Introducción parámetros

Introducción de parámetros siguiendo el proceso de nombre, tipo, rango, unidades y código si es de entrada y volviendo a introducir nuevos parámetros hasta que el profesor decida que son suficientes.

Boque 6 Introducción pasos y formula

Introducción de tantos pasos como oportunidades tenga el alumno de escribir el resultado correcto, aunque una vez que haya fallado la primera oportunidad la calificación será suspenso. Con el último paso introducirá la fórmula, aunque podrá introducir fórmula en cada paso produciendo que el problema sea cancelado durante la comprobación.

Bloque 7 Comprobación solución

Una vez se decide no escribir más pasos se comprueba que el problema está bien definido para obtener la solución, debe tener más de 1 parámetro, además de 1 paso y 1 fórmula mínimo, dependiendo si es correcto o no seguirá diferentes caminos.

Bloque 8 Borrado problema mal definido

Si el problema no pasa la comprobación será borrado de la base de datos no dejando ningún rastro de la creación y volviendo a mostrar el menú para elegir acción.

Una vez definidos los bloques de forma genérica se muestra el comportamiento de éstos mediante un diagrama de caja blanca que se muestra en la Figura 19. Se observan todos los posibles caminos que puede seguir el método RecogidaDatos() a la hora de crear un problema nuevo.

Observando el diagrama se concreta que no existen caminos aislados ni bloques a los que no se pueda llegar, o bucles de los que no se pueda salir, quedando de esta forma constatado que el método más complejo del proyecto está correctamente definido. Siempre volviendo al bloque 1, donde se muestran las opciones que se pueden realizar, entre ellas las de salir, que aunque no lo represente en el diagrama se puede elegir en cualquier momento del flujo del diagrama.

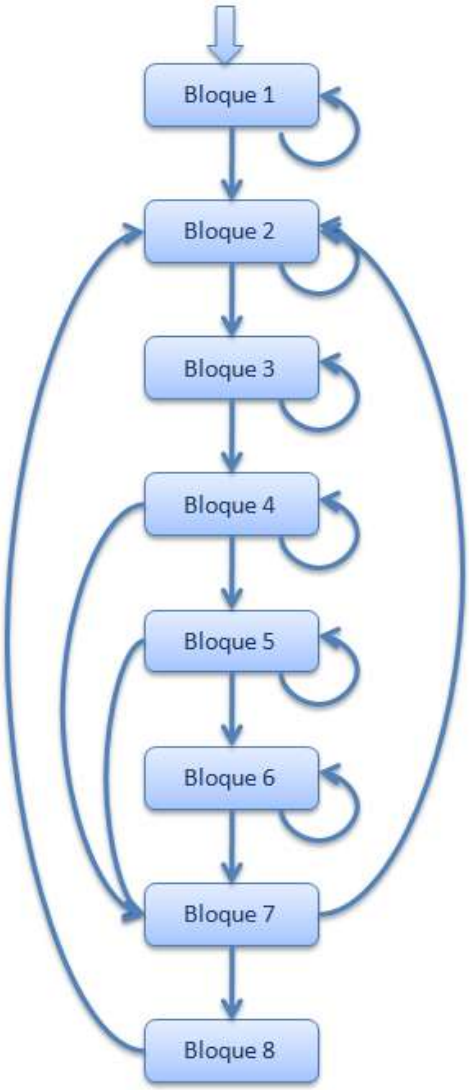

Figura 19: Diagrama de caja blanca

4.1.2. CAJA NEGRA

En programación, se denomina caja negra a aquel componente que es probado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. La caja es "negra", luego no se puede ver lo que hay dentro, justo al contrario del estudio de la caja “blanca”.

Para realizar las pruebas se usarán valores límites, esto es, valores extremos para medir la robustez del código. Se muestran algunas pruebas de caja negra realizadas para los módulos ModeloInterfazAlumno.java, ModeloInterfazProfe.java y GestionBD.java.

Durante el funcionamiento de la aplicación, la primera inserción de posibles valores límites va a producirse en el acceso tanto del profesor como del alumno a la herramienta, mediante la introducción de nombre o contraseña.

Prueba Caja negra: Identificación profesor. (ModeloInterfazProfe.java y GestionBD.java)
Para el acceso del profesor, éste deberá introducir su nombre, que se validará comparándolo con el almacenado en la base de datos.

Prueba 1. Identificación profesor.

Clases java	Entrada	Descripción	Salida esperada	Salida real
ModeloInterfaz-Porfe.java y GestionBD.java	campo vacío	Identificación de un profesor mediante su nombre pero insertando un vacío	Deberá mostrar un mensaje pidiendo que vuelva a introducir el nombre	Devuelve que el profesor no existe "ACCESS DENIED!", accediendo a la BBDD a buscar un vacío. **INCORRECTO**

Tabla 10. Prueba 1 Caja negra

El error no es grave pero un acceso a la base de datos a buscar un vacío puede provocar resultados no deseados. Para solucionarlo se comprueba siempre el tamaño de la cadena insertada y de ser igual a 0, se pide al profesor que vuelva a introducir el nombre evitando así el acceso a la Base de datos.

Prueba 2. Identificación profesor.

Clases java	Entrada	Descripción	Salida esperada	Salida real
ModeloInterfaz-Porfe.java y GestionBD.java	Inserción del nombre del profesor sin tener en cuenta las Mayúsculas	Identificación de un profesor mediante su nombre en mayúsculas cuando está almacenado en minúsculas	Deberá comprobar en la BBDD que es un profesor activo y permitirle el acceso	Devuelve que el profesor no existe "ACCESS DENIED!", **INCORRECTO**

Tabla 11. Prueba 2 Caja negra

Este error se evita contemplando la cadena insertada por el profesor con todas las posibles combinaciones de mayúsculas y minúsculas para que "antonio" y "ANTOnio" sirvan para que el profesor llamado Antonio acceda a la aplicación.

Prueba 3. Identificación profesor.

Clases java	Entrada	Descripción	Salida esperada	Salida real
ModeloInterfaz-Porfe.java y GestionBD.java	Inserción del nombre del profesor con cualquier conjunto de caracteres	Identificación de un profesor mediante un conjunto irregular de caracteres	Deberá comprobar en la BBDD que no es un profesor activo y no permitirle el acceso	Devuelve que el profesor no existe "ACCESS DENIED!", **CORRECTO**

Tabla 12. Prueba 3 Caja negra

Prueba Caja negra nº 2: Identificación alumno. (ModeloInterfazAlumno.java y GestionBD.java)
Para el acceso del alumno, éste deberá introducir su nombre de usuario (userid) más su contraseña, que se validará comparándolo con los datos almacenados en la base de datos.

Prueba 4. Identificación alumno.

Clases java	Entrada	Descripción	Salida esperada	Salida real
ModeloInterfaz-Alumno.java y GestionBD.java	Inserción de la contraseña al azar repetidas veces	Identificación del alumno mediante una contraseña al azar que no permitirá nunca el acceso	Tras un número limitado de intentos se rechazará el acceso del alumno	Se produce un bucle infinito que no acabará hasta que se acierte la contraseña. **INCORRECTO**

Tabla 13. Prueba 4 Caja negra

El error se evita controlando el número de intentos en la inserción de la contraseña. Cuando se sobrepase este número, el usuario será rechazado y deberá volver a introducir su nombre de usuario para tener opción a insertar la contraseña.

Prueba negra nº 3: Menú. (ModeloInterfazProfe.java)
En esta prueba el profesor deberá introducir la opción que quiera realizar escribiendo el texto relacionado con la acción a llevar a cabo: crear, modificar, borrar, alta.

Prueba 5. Menú profesor.

Clases java	Entrada	Descripción	Salida esperada	Salida real
ModeloInterfaz-Profe.java	Inserción de un vacío	El profesor intenta escoger una opción del menú introduciendo un vacío	La aplicación no entiende lo insertado y pide que se repita	Se produce la salida "No entiendo, repita por favor..." **CORRECTO**

Tabla 14. Prueba 5 Caja negra

Prueba 6. Menú profesor.

Clases java	Entrada	Descripción	Salida esperada	Salida real
ModeloInterfaz-Profe.java	Inserción de una cadena de caracteres aleatoria	El profesor intenta escoger una opción del menú introduciendo caracteres al azar	La aplicación no entiende lo insertado y pide que se repita	Se produce la salida "No entiendo, repita por favor..." **CORRECTO**

Tabla 15. Prueba 6 Caja negra

Prueba Caja negra nº 4: Elección Física/Química. (ModeloInterfazAlumno.java y ModeloInterfazAlummno.java) El usuario deberá elegir si quiere crear o realizar problemas de una u otra materia, mediante la introducción de un número (1=física/2=química).

Prueba 7.Elección Física/Química.

Clases java	Entrada	Descripción	Salida esperada	Salida real
ModeloInterfaz-Alumno.java y ModeloInterfaz-Profe.java	Inserción de un carácter no numérico	Se elegirá el tipo de problema mediante la inserción de un número	Se volverá a pedir la introducción del tipo de problema	Se produce la salida: "Vuelva a introducir opciones (1:fisica/ 2:quimica)" **CORRECTO**

Tabla 16. Prueba 7 Caja negra

4.2. PRUEBAS DE INTEGRACION

Las pruebas de integración hacen referencia a la correcta interrelación entre todos los módulos que dependen unos de otros dentro del proyecto. La integración entre módulos se ha ido comprobando a lo largo del desarrollo del código. Desde un primer momento la correcta relación entre módulos se ha ido comprobando continuamente para que a medida que el código de los módulos vaya creciendo en tamaño, no llegase el momento en el que podamos tener errores complejos en la interrelación de los módulos.

Para una visión global de la interrelación de los módulos se puede usar el diagrama de clases de alto nivel de la Figura 9. Como se observa en el diagrama, la realización del proyecto está basado en la arquitectura MVC (Modelo – Vista – Controlador), luego la interrelación de estos tres módulos es esencial que esté definida desde el principio. La primera clase que se creó fue la Interfaz, que posteriormente se dividiría en InterfazAlumno e InterfazProfe. Ambos se encargan de mostrar la información relevante en forma de ventana. Más tarde se crea el controlador, mediante la clase CntrlEventos que posteriormente se dividiría en CntrlEventosAlumno y ContrlEventosProfe. Ambas se encargan de gestionar los eventos producidos en la vista para evaluarlos y tratarlos en el modelo, siguiente módulo en realizarse. Estas relaciones se pueden observar en la Figura 20:

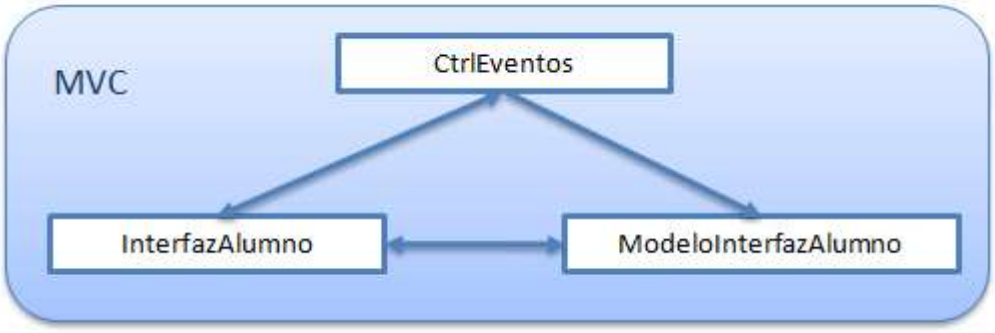

Figura 20: Esquema MVC

Una vez construido el esquema MVC con sus tres clases correctamente interrelacionadas, se integra la clase Marco, ya sea MarcoInterfazAlumno o MarcoInterfazProfe, encargándose de englobar en una entidad el MVC y a su vez relacionándolo con la clase Main() instaurado en EntradaAlumno o EntradaProfe.

Esta relación con la clase Main() es necesaria desde el primer momento para ir probando todos los objetos creados, necesitando una perfecta conexión para que los resultados de aplicar el método main sean los esperados. A continuación se muestra el esquema:

Figura 21: Pruebas integración Main.

Por último, se relaciona todo lo que haya construido hasta ahora con la base de datos y para ello se crea la clase GestionBD que se relaciona directamente con la clase ModeloInterfaz.

Una vez que se tienen todas las anteriores relaciones bien definidas, el diseño completo del sistema estará construido, y del correcto funcionamiento de estas relaciones dependerá el funcionamiento general de la aplicación. En el siguiente esquema de la Figura 22 se visualiza como queda el sistema:

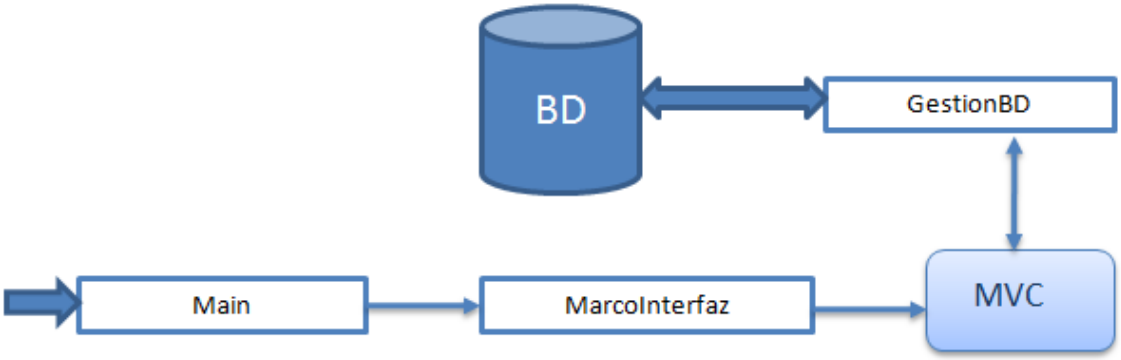

Figura 22: Pruebas integración sistema completo

4.3 PRUEBAS DE VALIDACIÓN

Son pruebas funcionales sobre el sistema completo que buscan una cobertura de la especificación de requisitos y del manual del usuario. Se mostrará cómo se han cumplido todos los requisitos con pantallazos de la aplicación completa.

Los requisitos principales del profesor van a ser las acciones que puede realizar. Vamos a ver una simulación de cada una de ellas.

Pruebas validación profesor

Prueba de validación de la creación de un problema por parte del profesor. En concreto la creación de un problema de química de formulación del oxido de hierro (FeO):

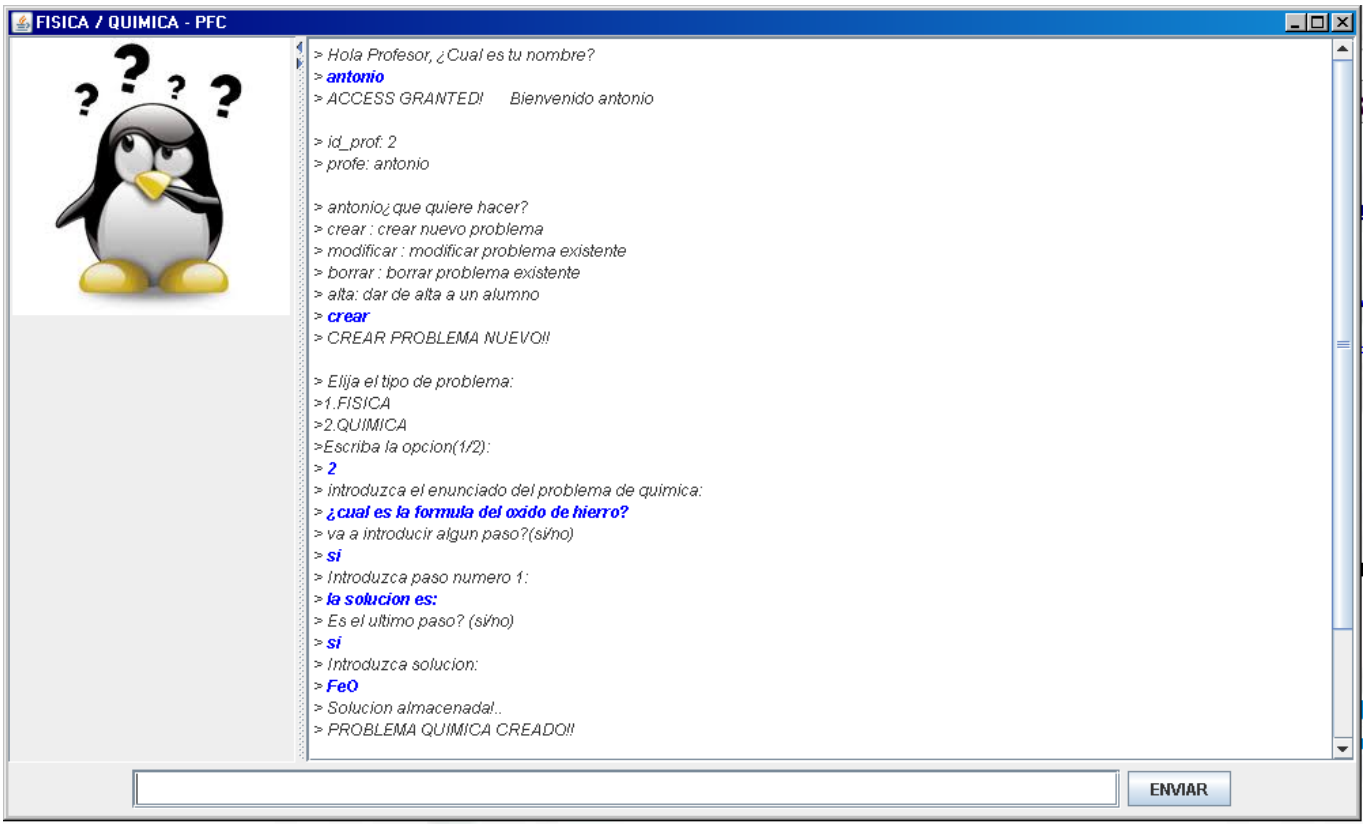

Figura 23: Prueba validación – crear.

Prueba de validación de la modificación de un problema por parte del profesor. En concreto la modificación de un problema de física modificando uno de sus pasos, en concreto el primero y solo el primero:

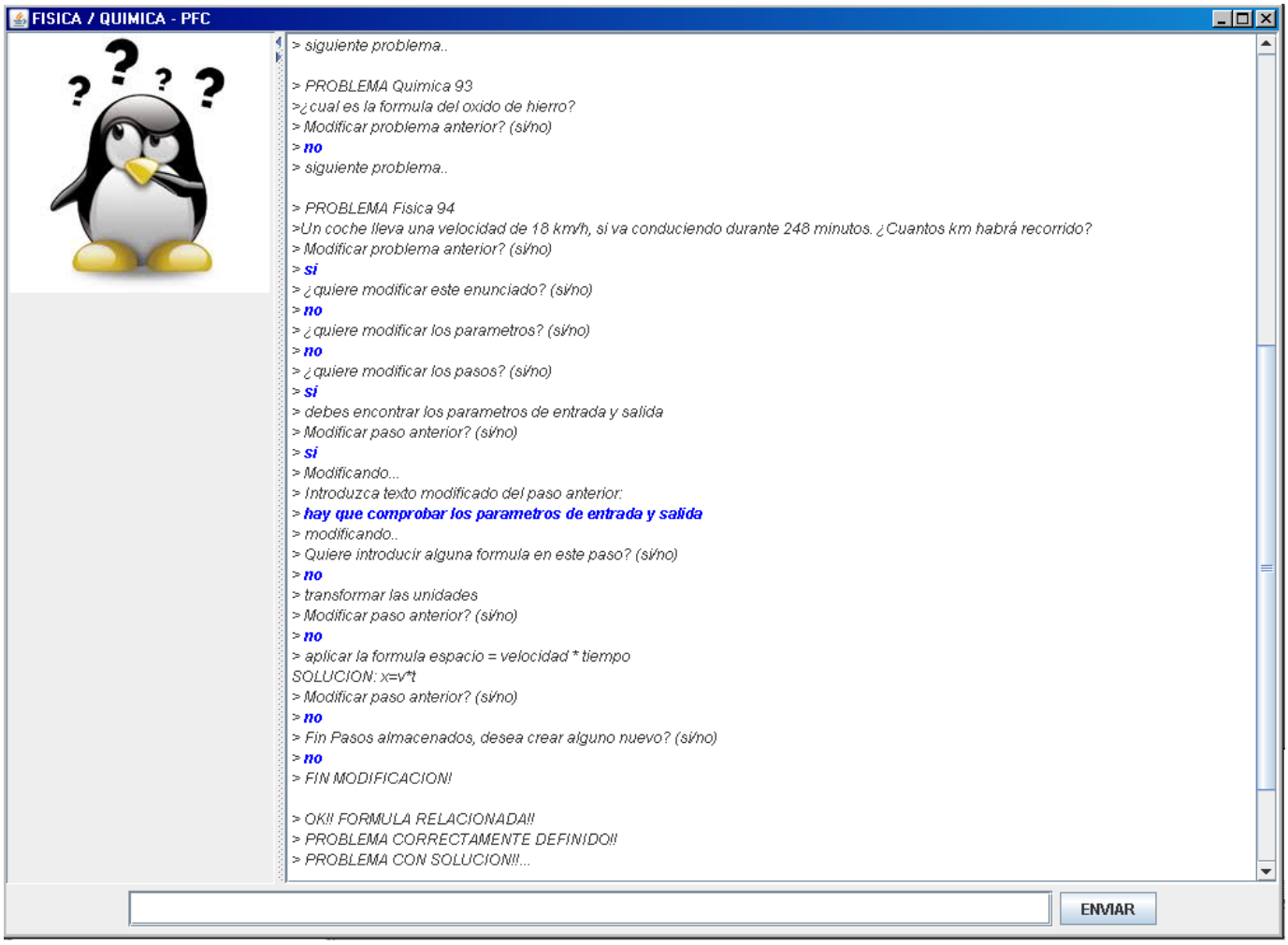

Figura 24: Prueba validación – modificar

Prueba de validación del borrado de un problema por parte del profesor. Para que un profesor pueda borrar un problema, éste debe haber sido creado por él.

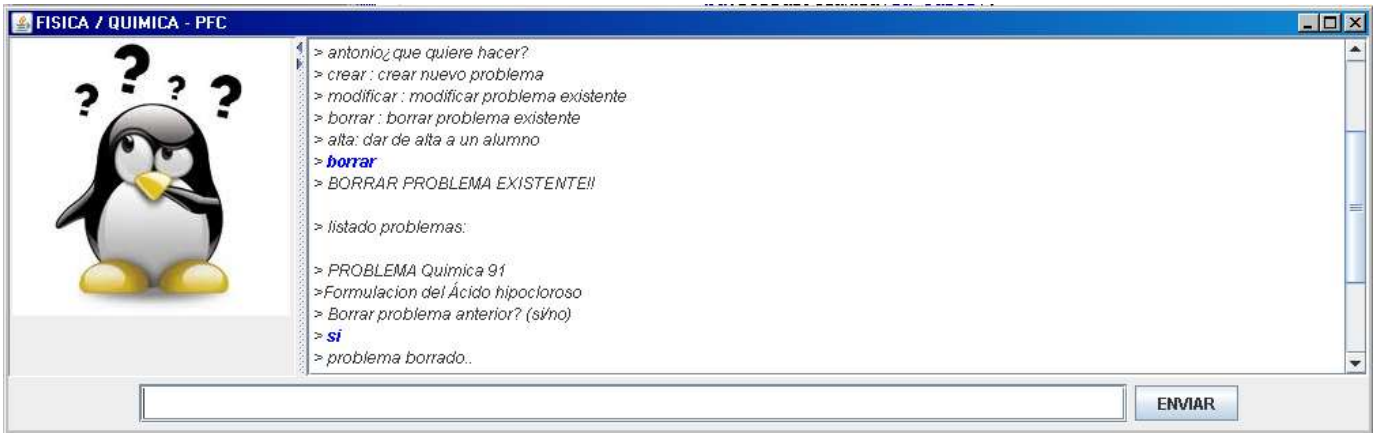

Figura 25: Prueba validación – borrar

Prueba de validación del alta de un alumno por parte del profesor. Solo se da de alta en la base de datos con un "userid" que el alumno deberá usar para asignarse el mismo su propia contraseña.

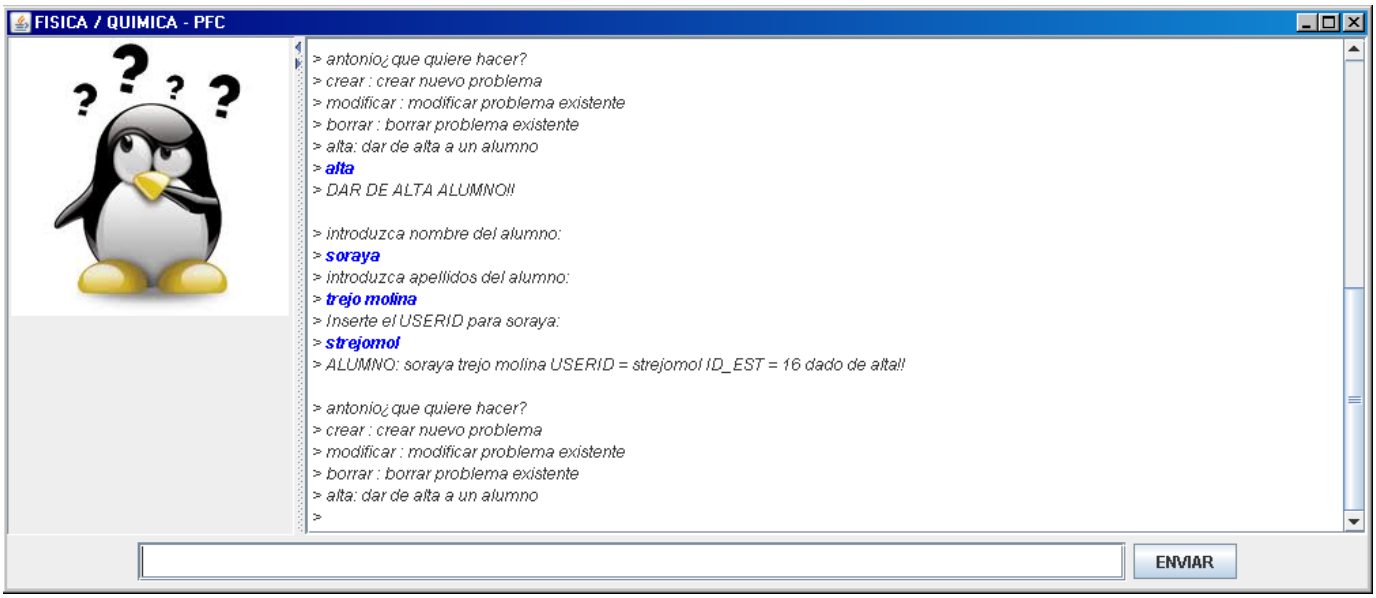

Figura 26: Prueba de validación – alta

Pruebas validación alumno

Prueba de validación del acceso de un alumno nuevo a la aplicación previamente dado de alta por el profesor, indicando su "userid". El alumno deberá definir él mismo su propia contraseña, la cual deberá introducir siempre que quiera resolver problemas.

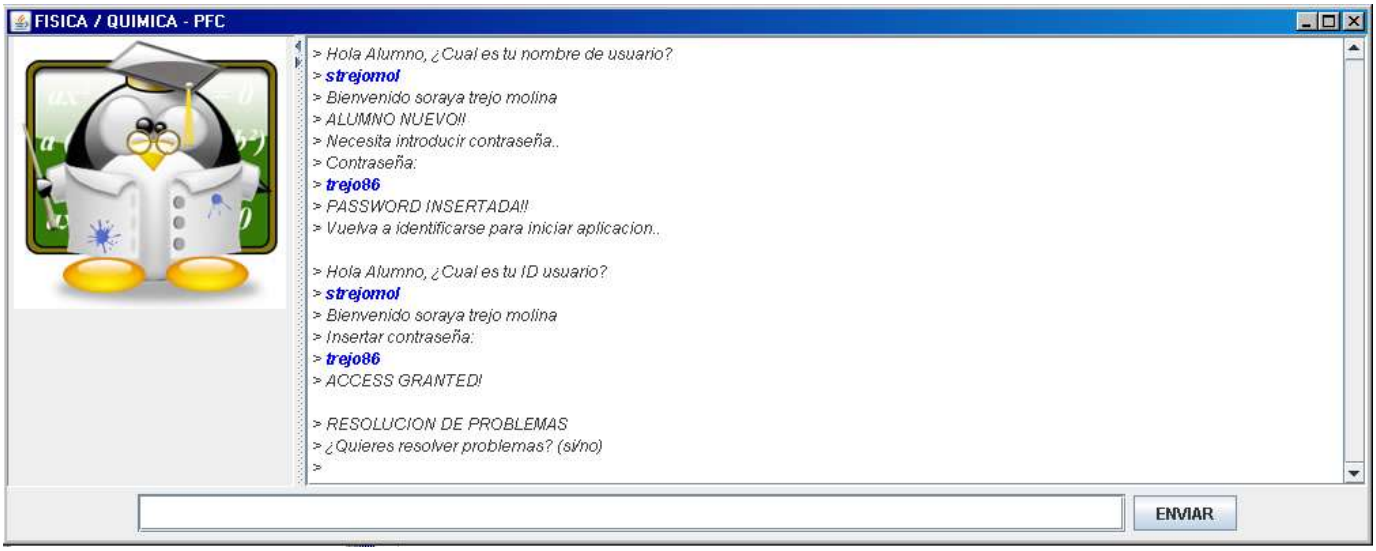

Figura 27: Prueba de validación – acceso alumno.

Prueba de validación de la generación de problemas de física y de química resolviéndolos de forma correcta. Resolución de un problema de cada tipo a la primera, obteniendo la calificación de aprobado en ambos y siendo almacenada en la BBDD.

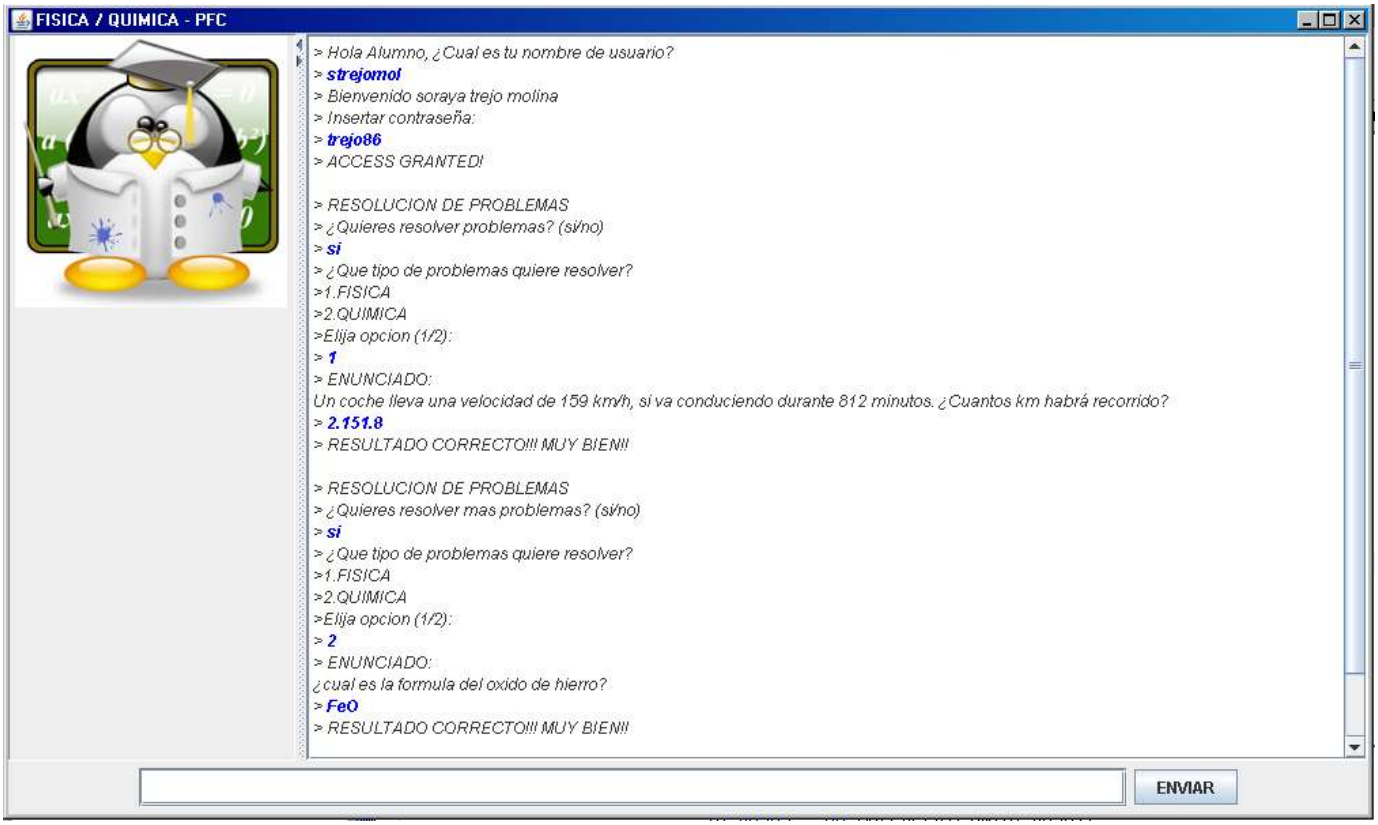

Figura 28: Prueba de validación – resolución problemas alumno.

4.4. PRUEBAS DE ACEPTACION

En esta prueba se evalúa el grado de calidad del software con relación al cumplimiento de los objetivos para usuarios reales. Por lo tanto, estas pruebas están centradas en las opiniones y actuaciones de los usuarios con el software desarrollado.

La selección de los usuarios se ha realizado con el objetivo de agrupar las opiniones de candidatos con la mayor diversidad posible en cuanto a formación y conocimientos informáticos. Aunque la herramienta es para niños en edad escolar, cualquiera con edad superior debería poder usar la aplicación sin problemas. Los perfiles de usuarios probadores se muestran en la siguiente tabla:

USUARIOS	EDAD	ESTUDIOS	APLICACIÓN
USUARIO A	19	Educación Secundaria Obligatoria	Estudiante
USUARIO B	17	Bachillerato	Estudiante
USUARIO C	24	Carrera universitaria sin conocimientos informáticos	Profesor
USUARIO D	16	Educación Secundaria Obligatoria	Estudiante
USUARIO E	12	Educación primaria	Estudiante
USUARIO F	54	Educación primaria	Profesor
USUARIO G	44	Carrera de magisterio	Profesor
USUARIO H	15	Educación Secundaria Obligatoria	Estudiante
USUARIO I	25	Carrera universitaria informática	Profesor

Tabla 17: Perfiles usuarios probadores

Dependiendo de las edades y conocimientos de los usuarios estarán destinados a probar la aplicación estudiante o la aplicación profesor.

Todos los usuarios antes de enfrentarse a esta prueba habían experimentado el uso de ordenadores y aplicaciones estilo "Messenger". En el caso de los alumnos, tras una rápida introducción de cómo funciona la aplicación, estaban preparados para el desarrollo de la prueba. Sin embargo en el caso del profesor es necesario explicar cómo se deben insertar los enunciados teniendo en cuenta que debemos definir variables en el enunciado que serán sustituidas por los valores de los parámetros generados aleatoriamente, por ejemplo. El tamaño de las fórmulas o la necesidad de fijar un número determinado de parámetros de entrada y salida entre otros, hacen que sea necesaria una explicación más extensa. Toda esta información está detallada en el manual de usuario, Anexo D.

Tras la un determinado espacio de tiempo interactuando con la herramienta, a cada usuario se le pide realizar una encuesta donde podrán dar su opinión mediante puntuaciones de 1 a 10 acerca de las sensaciones que les ha provocado la aplicación, si les ha gustado o no, si les ha parecido sencillo, etc. La Tabla 18 muestra los resultados obtenidos.

USUARIOS	DISEÑO	RAPIDEZ	FACILIDAD MANEJO	OPINION GENERAL
USUARIO A	5	8	7	6
USUARIO B	6	8	7	7
USUARIO C	4	9	9	8
USUARIO D	8	8	8	8
USUARIO E	10	9	9	9
USUARIO F	8	9	6	8
USUARIO G	7	8	7	7
USUARIO H	8	9	8	8
USUARIO I	6	8	7	7

Tabla 18: Puntuaciones pruebas usuario en una escala de 0-10.

Y la media de los resultados nos dará una idea más general de la aceptación de la herramienta entre los usuarios encuestados:

	DISEÑO	RAPIDEZ	FACILIDAD MANEJO	OPINION GENERAL
MEDIA	6,9	8,4	7,6	7, 6

Tabla 19: Puntuaciones medias pruebas usuario

Los datos obtenidos son buenos en general. El aspecto que ha recibido una menor puntuación es el diseño, refiriéndose al diseño de la interfaz. El diseño es simple por lo que los usuarios más jóvenes lo ven demasiado plano, con la excepción del usuario E, que valoró de forma sobresaliente el diseño con caricatura del pingüino – profesor.

La característica mejor puntuada es la rapidez, ya que las pruebas se han realizado sobre una base de datos muy poco cargada y un pc Intel Core2 Duo a 2.27Ghz con 4.00GB de memoria RAM. Por lo que todos los usuarios destacaron la rapidez de la herramienta.

Por último la opinión general que la herramienta merece por parte de los usuarios es alta, con una media mayor que 7,5. La mayoría de los usuarios comparten que es una aplicación fácil de utilizar y cercana por su simplicidad. En general se comparte que es una muy buena ayuda para el repaso de la asignatura de física y química.

5. CONCLUSIONES Y TRABAJO FUTURO

El principal objetivo del proyecto era la creación del agente conversacional pedagógico para la enseñanza de problemas de física y química se ha cumplido con la implementación del agente. A continuación, se revisan cómo se han cumplido cada uno de los requisitos identificados.

REQUISITOS FUNCIONALES

RF1. El administrador da de alta en la BBDD a los profesores, permitiendo a éstos la utilización de la aplicación.

Es necesario que el administrador de la herramienta dé de alta a cada profesor que quiera manejar la aplicación.

RF2. El profesor crea especificaciones de problemas de física y química a través de un editor de escritorio local que ira solicitando la información necesaria para la creación.

El profesor mediante la opción "crear" es capaz de definir cualquier problema simple de física y química. Se puede observar en la Figura 23.

RF3. El profesor debe asignar a cada problema un enunciado, parámetros y pasos, para que la creación del problema sea correcta.

La aplicación irá guiando al profesor en el desarrollo del problema para que éste quede correctamente definido y se pueda obtener el resultado. Para una correcta definición se debe utilizar el manual de usuario.

RF4. El profesor puede modificar el problema.

Mediante la opción "modificar" el profesor será capaz de modificar enunciado, parámetros o pasos de cada problema anteriormente insertados por él mismo. Se puede observar en la Figura 24.

RF5. El profesor puede borrar problemas creados por él.

Mediante la opción "borrar" el profesor será capaz de eliminar enunciado, parámetros o pasos de cada problema anteriormente realizado por él mismo, no dejando ningún residuo en la base de datos. Se observa en la Figura 25.

RF6. El profesor se encarga de dar de alta a los estudiantes, quienes podrán realizar los problemas creados por el profesor.

RF7. El profesor definirá el usuario, pero contraseña que utilizará el alumno para acceder a la aplicación la definirá el alumno.

Mediante la opción "alta" el profesor será capaz de permitir a los alumnos el acceso a la aplicación. Se puede observar en la Figura 26.

RF8. El estudiante deberá identificarse con usuario y contraseña para poder acceder a la aplicación.

Anteriormente deberá haber sido definido por el profesor, la identificación la realizará introduciendo el usuario y la contraseña mediante texto. La aplicación irá pidiendo estos datos. Se puede observar en la Figura 27.

RF9. Los problemas que le aparecen al estudiante inicialmente se muestran de manera aleatoria, pero más adelante, a medida que vayamos obteniendo resultados, se mostrarán siguiendo el principio: "repasar más aquellos en los que más falle".

La aplicación almacenará los aciertos y fallos que cada alumno haya tenido en cada problema. Seleccionando siempre el problema con mayor número de fallos dejando como último problema en salir al que más veces haya resuelto bien el alumno.

RF10. El estudiante irá resolviendo los problemas en lenguaje natural guiado paso a paso por la aplicación.

Esto será posible gracias a los pasos creados previamente por el profesor para ir ayudando al alumno a encarar de mejor manera los problemas.

RF11. Tanto la aplicación del estudiante como la del profesor presentará una figura a la izquierda.

Se ha usado una ventana estilo "Messenger" para que sea más familiar para los alumnos y una foto para que cada usuario sepa con quien están interaccionando.

RF12. El estudiante podrá cerrar la aplicación en cualquier momento, los datos hasta ahora definidos deberán persistir en la BBDD.

Mediante la inserción del String "cerrar" o haciendo click en la "x" de la ventana.

REQUISITOS NO FUNCIONALES

RNF1. La aplicación será amigable según los principios Interacción Persona-Ordenador.

Como antes se ha referenciado, la arquitectura estilo “Messenger” hará que los alumnos, a quien está dirigida la aplicación, se sientan cómodos a la hora de usar la aplicación y no pierdan interés. Este requisito es compartido por los usuarios que probaron la herramienta.

RNF2. Se seguirá un diseño centrado en el usuario.

Ya que todo lo desarrollado en este proyecto está dirigido a los usuarios, los resultados de las encuestas realizadas a los usuarios validan este requisito también.

RNF3. La velocidad de respuesta de la aplicación no superara en ningún caso 1 minuto por interacción.

Lo único que podría provocar este retraso sería bucles infinitos o acceso saturados a la base de datos, situaciones que se han estudiado para que no puedan producirse. Además, la rapidez de la aplicación ha sido una de las características mejor puntuadas durante las pruebas de aceptación.

En cuanto al trabajo futuro, se puede destacar que este tipo de aplicaciones pueden tener un gran potencial en general. De hecho, cada vez la tecnología está más a la orden del día, y las nuevas generaciones las dominan y usan continuamente, por lo que se estima que el siguiente paso que se podría dar es desarrollar el agente para poder ser ejecutado en dispositivos móviles.

Otro de los aspectos a mejorar sería el nivel de dificultad de los ejercicios de física y química. En esta primera versión se han desarrollado ejercicios básicos con ecuaciones simples, y el siguiente paso que se podría dar es introducir también la posibilidad de resolver ejercicios con sistemas de ecuaciones o complejos conjuntos de fórmulas para que un futuro se pudiese plantear también el uso del agente en el nivel universitario.

6. BIBLIOGRAFÍA

REFERENCIAS

[1] Roque Jiménez Pérez, Bartolomé Vázquez Bernal. Departamento de Didáctica de las Ciencias y Filosofía. Universidad de Huelva Av. Fuerzas Armadas, s. 21071.H. Estudio de dificultades en resolución de problemas de física y química: desde la percepción de profesores de formación inicial y de alumnos de eso, 2002.

[2] Pérez Marín, Diana. 2010. Seminario de Investigación en Tecnologías de la Información Aplicadas a la Educación, SITIAE.

[3] del Pilar Soler Gordils, M. Sistemas e-learning inteligentes. Pedagoga, 2005.

[4] Bartolomé, A. 2004. Conceptos básicos de Blended Learning. Universidad de Barcelona, Píxel-Bit. Revista de Medios y Educación, 23, pp. 7-20.

[5] Zárate Rea, H. 2008. Paradigmas de Programación Universidad Nacional Autónoma de México Facultad de Ingeniería Ciudad de México.

[6] Pimentel, F. 2007. Programación orientada a objetos con java. ed. Thomson-

[7] Facultad Experimental de Ciencias y Tecnología. Departamento de Computación. Unidad Académica Base de Datos. 2005. SISTEMAS DE GESTIÓN DE BASE DE DATOS SGBD / DBMS

TUTORIALES

- *JDBC™ 4.0 Specification JSR 221 Lance Andersen, Specification Lead November 7, 2006 November 2006*
- *MySQL Query Browser Manual.pdf*
- *MySQL Workbench Manual.pdf*
- (CoreJava2VI) Cornell, C.S.H.G. *Core Java 2. Volumen I. Fundamentos*.
- Pearson *(ed.)* Sun Microsistems, **2006**
- (CoreJava2VII) Cornell, C.S.H.G. *Core Java 2. Volumen II. Características avanzadas* Pearson *(ed.)* Sun Microsistems, **2006**
- (MySQL) Gallardo, J.D.G. *MySQL* Anaya *(ed.)* **2004**
- ORACLE *The Java Tutorials*

ANEXO A: Explicación caso práctico profesor

Explicación de las transiciones entre estados en un caso práctico. El caso práctico es la creación de un problema de física como este: "*Un coche va a una velocidad de 59 km/h, durante un tiempo de 45 min, ¿cuánto espacio en Kilómetros habrá recorrido?*". El proceso de generación de problemas como el del ejemplo pero con valores aleatorios está representado por la siguiente figura:

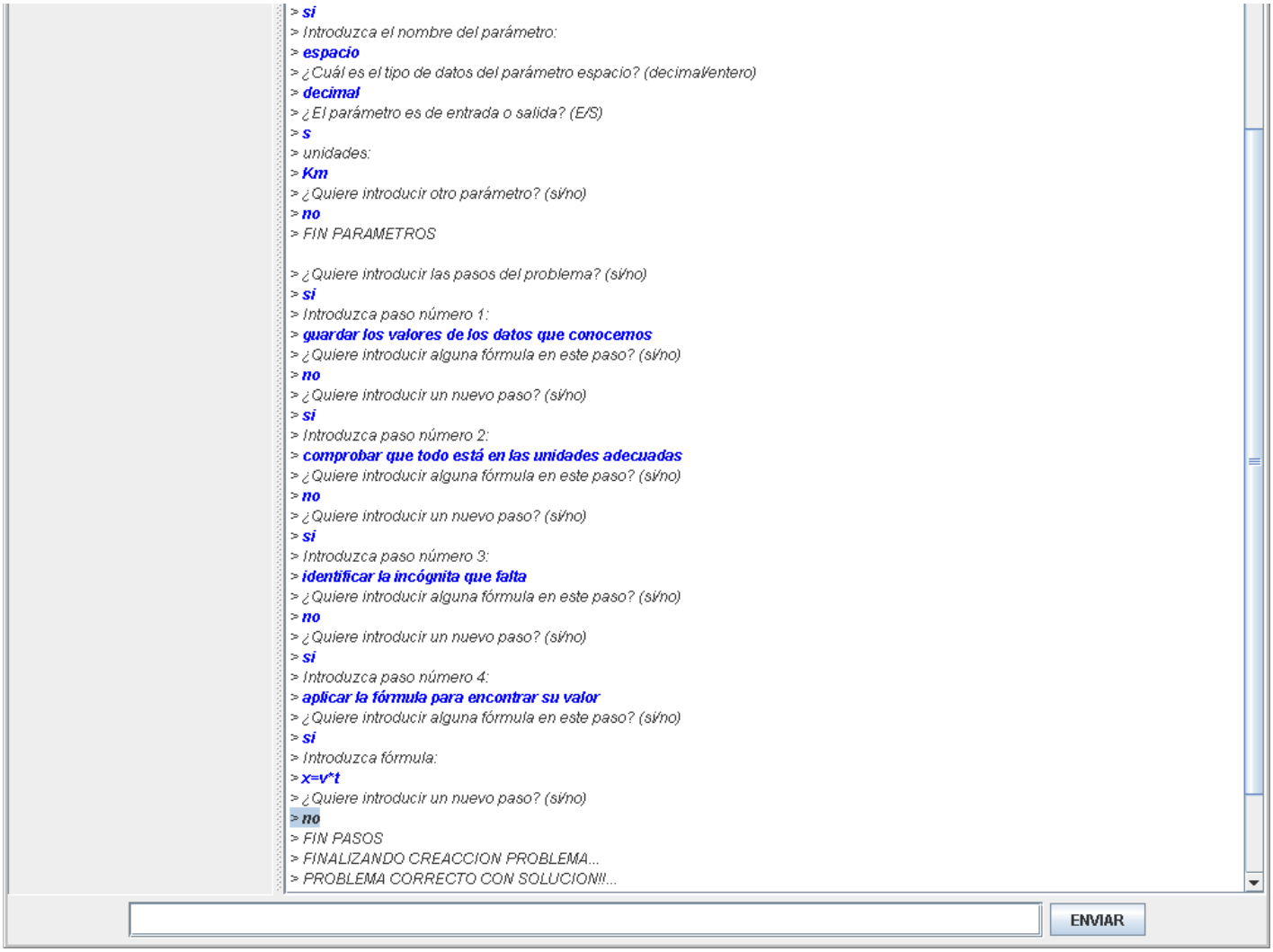

Figura 29. Caso práctico – creación de problema.

Al arrancar la aplicación, se recibe el saludo inicial: *">Hola Profesor, ¿Cuál es tu nombre?"*. Cuando el usuario introduce un nombre cualquiera pasamos al estado "S0". En este estado se está a la espera de recibir el nombre del profesor para comprobar si está dado de alta en la base de datos, permitiéndole así el acceso.

El usuario introduce "**Antonio**" como nombre y tras acceder a la Base de datos, se confirma que está dado de alta y se mostrará el menú de acciones que puede realizar el profesor:

"> Antonio ¿qué quiere hacer?

> CREAR: crear nuevo problema

> MODIFICAR: modificar problema existente

> BORRAR: borrar problema existente

> ALTA: dar de alta a un alumno"

Tras mostrar el menú se transita al siguiente estado "S1" con la inserción de la opción de acción por parte del profesor.

El usuario introduce "**CREAR**" con el fin de crear un problema nuevo. Antes de transitar al siguiente estado se mostrará las opciones de tipos de problemas que se pueden elegir:

"> Elija el tipo de problema:

>1. FISICA

>2. QUIMICA

>Escriba la opción (1/2):"

Ahora si se pasa al estado "S10" tras la inserción de una nueva cadena por parte del usuario. En este estado la aplicación almacena el tipo de problema (física o química) mediante la inserción del número correspondiente.

El usuario introduce la opción "**1**" para crear un problema de física. Seguidamente se muestra la frase: "*> introduzca el enunciado del problema de física:*", pasando al estado "S2".

En "S2" la aplicación espera que el usuario inserte el enunciado del problema siguiendo una serie de pautas. Es necesario que todos los valores de parámetros de entrada que se quieran representar en el enunciado deban ser referenciaos mediante un código, por ejemplo: "X". Cada parámetro obtendrá un valor aleatorio que sustituirá en la zona del enunciado donde se encuentre con este código. Se recomienda siempre escribir letras mayúsculas. Veamos cómo sería la inserción correcta por parte del usuario: "**Si un coche va a una velocidad de X km/h, durante un tiempo de Y min, ¿cuánto espacio en Km habrá recorrido?**". El enunciado se almacena y se transita al siguiente estado "S21" mostrando antes: "*> ¿va a introducir algún parámetro? (si/no)*".

En "S21" se está a la espera de que el usuario inserte sí o no para pasar a introducir los parámetros o directamente los pasos. No introducir parámetros conllevará que el problema esté mal definido. El usuario introduce: "**si**", pasando por ello al estado "S211" tras la impresión de la frase: "*> Introduzca el nombre del parámetro:*"

En "S211" almacenamos el nombre del parámetro después de que el usuario haya lo haya insertado, en este caso: "**Velocidad**". También es ente estado se muestra la frase: "*> ¿Cuál es el tipo de datos del parámetro Velocidad? (decimal/entero)*" y se transita al siguiente estado.

El siguiente estado es "S212". En él se recibe el tipo de notación que manejará el parámetro, ya sea decimal o entero. El usuario introduce: "**decimal**", como la comprobación es correcta se produce una nueva transición, mostrando: "*¿El parámetro es de entrada o salida? (E/S)*".

El siguiente estado es "S2121" se define el tipo de parámetro. En este caso como el usuario introduce "**E**", el parámetro será de entrada y habrá que asignarle un rango de valores. Tras almacenar que es un parámetro de entrada transitamos de estado mostrando:

"*> Introduzca el rango de valores del parámetro Velocidad:*

> valor mínimo:"

Durante el estado "S213" se almacena el valor mínimo que pueda tener el parámetro. En este caso el usuario introduce: "**0**". Y se pide que se introduzca el valor máximo que pueda alcanzar el parámetro: "*> valor máximo:*", pasando al siguiente estado.

El siguiente estado es "S214", donde se almacena el valor máximo, siendo "**110**" la cantidad introducida por el usuario. Tras almacenar el valor se pasa al estado "S215", mostrando: "*> unidades:*".

En "S215" se almacenan las unidades del parámetro tras la inserción de "**Km/h**" por parte del usuario, notificando que Kilómetros por hora son las unidades. El último paso en la definición de un parámetro es establecer el código mediante el que se relacionará con el enunciado para que pueda ser mostrado de forma aleatoria para que ningún problema sea igual al anterior.

Tras la frase: "*> introduzca el código del parámetro:*" se pasa al estado "S216" donde el usuario inserta: "**X**" quedando así el parámetro relacionado con el enunciado. Tras esta inserción volvemos al estado anterior "S21" mostrando: "*> ¿Quiere introducir otro parámetro? (si/no)*"

En el ejemplo se introducen dos parámetros más, Tiempo y Espacio. El primero de entrada y el segundo de salida. Se realizará el mismo camino que el anterior, salvo por que el de salida no se insertan ni valor mínimo, ni valor máximo, ni código. Una vez definidos estos dos parámetros, estando en el estado "S21" el usuario introduce "**no**" respondiendo a si quiere introducir más parámetros, pasando con esto a la inserción de los pasos.

Antes de la transición al estado "S22", se muestra:

"*> FIN PARAMETROS*

> ¿Quiere introducir las pasos del problema? (si/no)"

El usuario introduce "**si**", ya que sin pasos y sin fórmula el problema estará mal definido y será borrado. Tras confirmar que se van a generar nuevos pasos se muestra: "*> Introduzca paso número 1:*" pasando al estado "S221" donde se espera la inserción del paso 1 por parte del usuario.

En "S221" almacenamos el texto que define el paso número 1: "**guardar los valores de los datos que conocemos**", que es la frase introducida por el usuario y pasamos al estado "S222" mostrando: *"> ¿Quiere introducir alguna fórmula en este paso? (si/no)"*. En la creación correcta de un problema solo se creará una fórmula y deberá almacenarse junto con el último paso. Luego el usuario en este caso escribirá "**no**", volviendo al estado "S21" para realizar de nuevo el mismo proceso con los pasos 2,3 y 4.

Paso2: "**comprobar que todo está en las unidades adecuadas**".
Paso3: "**identificar la incógnita que falta**".
Paso4: "**aplicar la fórmula para encontrar su valor**".

En el caso del paso 4 cuando estemos en el estado "S222" el usuario insertará "**si**" para poder introducir la fórmula del problema, pasando al estado "S223" mostrando: "*> Introduzca fórmula:*".

En "S223" se almacena esta fórmula: "**x=v*t**", insertada por el usuario, necesaria para calcular la solución del problema.

Para finalizar la creación del problema volvemos al estado "S22" y a la pregunta: "*¿Quiere introducir un nuevo paso? (si/no)*" el usuario responde "**no**". Antes de dar por concluida la creación pasamos al estado "S6", donde se comprobara que la fórmula este perfectamente relacionada con los parámetros. Si todo va bien se mostrará:

"> FIN PASOS

> FINALIZANDO CREACCION PROBLEMA…

> PROBLEMA CORRECTO CON SOLUCION!!..."

Y volviendo al estado inicial "S1" para mostrar el menú y poder realizar nuevas acciones:

> Antonio ¿qué quiere hacer?

> CREAR: crear nuevo problema

> MODIFICAR: modificar problema existente

> BORRAR: borrar problema existente

> ALTA: dar de alta a un alumno"

ANEXO B: Explicación caso práctico alumno

Explicación de las transiciones entre estados en un caso práctico. El caso práctico es la resolución de un problema de física como este: "*Un coche va a una velocidad de X km/h, durante un tiempo de X min, ¿cuánto espacio en Kilómetros habrá recorrido?*". El sistema de selección de problemas según las calificaciones lo descartamos en este caso para realizar como ejemplo un problema determinado. El proceso de la resolución de este problema determinado se ve en la siguiente figura:

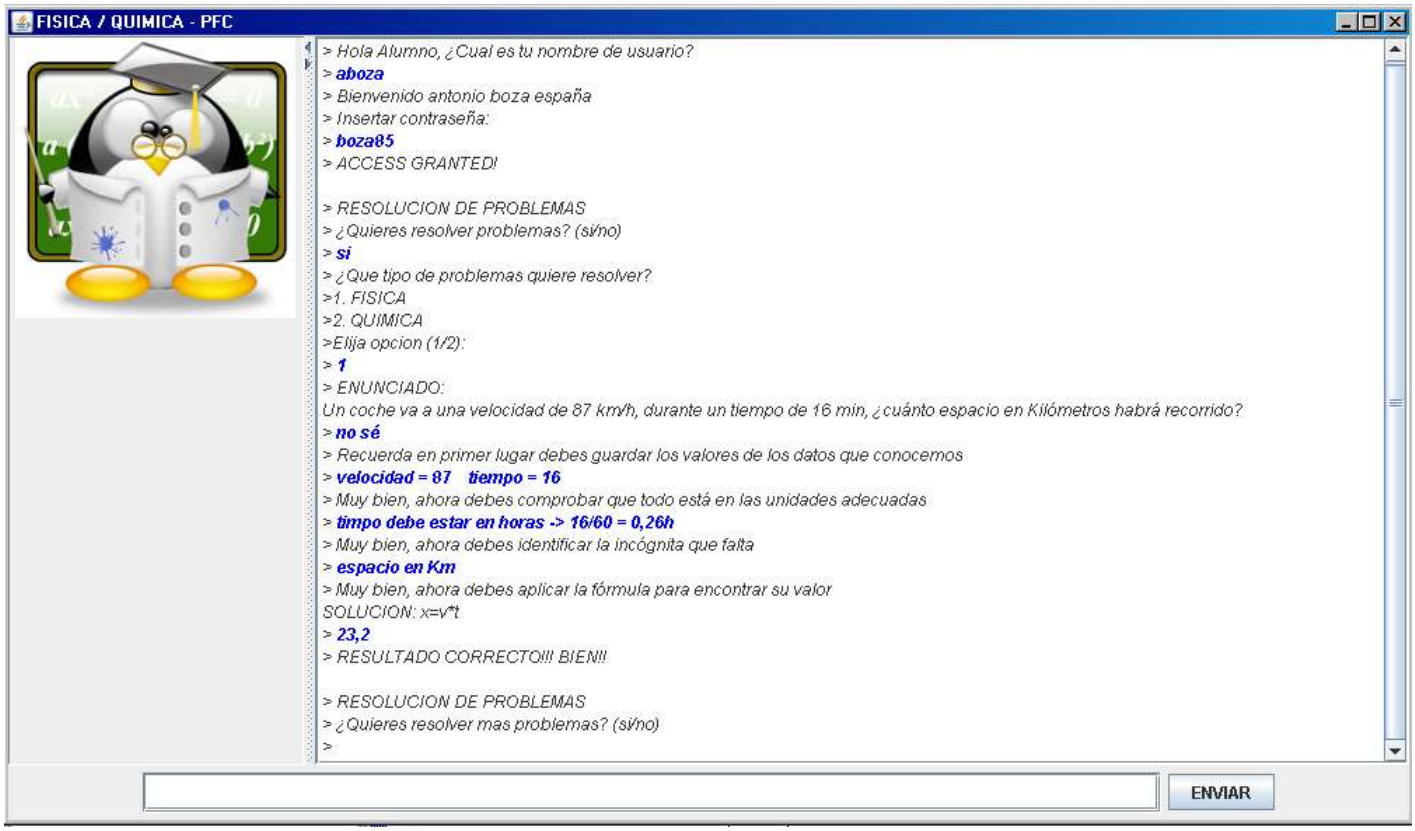

Figura 30. Caso práctico – resolución de problema

Al arrancar la aplicación, se recibe el saludo inicial: "*> Hola Alumno, ¿Cuál es tu nombre de usuario?*". Cuando el usuario introduce un nombre cualquiera pasamos al estado "S0". En este estado se está a la espera de recibir el nombre de usuario del alumno (userid), para comprobar si está dado de alta en la base de datos.

El usuario introduce "**aboza**" como nombre de usuario y tras acceder a la Base de datos, se confirma que está dado de alta y se pasará a la introducción de la contraseña, mostrando:

"*> Bienvenido antonio boza españa*
> Insertar contraseña:"

La introducción de la contraseña del alumno se produce en el estado"S1", donde el alumno inserta: "**boza85**". Se comprueba que es correcto accediendo a la base de datos y se transita al siguiente estado mostrando:

"*> ACCESS GRANTED!*
> RESOLUCION DE PROBLEMAS
> ¿Quieres resolver problemas? (si/no)"

Pasamos al estado "S20" donde el usuario introduce "**si**", provocando el paso al siguiente estado donde deberá elegir si quiere realizar ejercicios de física o química, mostrando la frase:

"*> ¿Qué tipo de problemas quiere resolver?*
>1. FISICA
>2. QUIMICA
>Elija opción (1/2):"

Tras la inserción del tipo de problemas, en este caso "**1**". Pasamos al estado "S3", donde se mostrarán los problemas, en concreto se mostrarán los problemas según un algoritmo que elegirá los problemas en los que peores resultados haya tenido el alumno. El problema se generará con valores aleatorios según los rangos definidos por el profesor:

"*> ENUNCIADO:*
Un coche va a una velocidad de 87 km/h, durante un tiempo de 16 min, ¿cuánto espacio en Kilómetros habrá recorrido?"

A partir de aquí el usuario debe introducir la solución al problema, para pasar al estado "S4", donde comprobaremos si ha acertado. En el estado "S4" recibimos esta respuesta, si no es correcta: "**no sé**", la aplicación irá generando los pasos que previamente habrá definido el profesor, para ayudar a resolver y sobre todo a aprender a resolver el problema. Después de cada paso se volverá a comprobar si se ha acertado, mostrando el siguiente paso de no ser la solución esperada.

"*> Recuerda en primer lugar debes guardar los valores de los datos que conocemos*
> Muy bien, ahora debes comprobar que todo está en las unidades adecuadas
> Muy bien, ahora debes identificar la incógnita que falta
> Muy bien, ahora debes aplicar la fórmula para encontrar su valor
*SOLUCION: x=v*t*"

En este caso práctico la solución correcta se alcanza en el último paso mediante la inserción por parte del usuario de: "**23,2**". Al ser correcto el resultado, provocará la finalización del proceso de resolución del este problema, volviendo al estado "S20" y generando la siguiente salida:

"> RESULTADO CORRECTO!!! BIEN!!
> RESOLUCION DE PROBLEMAS
> ¿Quieres resolver más problemas? (si/no)"

El sistema continuará si el usuario quiere continuar resolviendo problemas.

ANEXO C: Manual de Instalación

Para el correcto funcionamiento de la herramienta, se deben tener instalados dos programas, un gestor de bases de datos (MySQL) y la Máquina Virtual de Java (JMV), además del conector de MySQL para Java. Estos 3 elementos son necesarios para que se puedan ejecutar los ".jar".

Instalación de MySql:

- Bajarse el programa en la página: http://www.mysql.com/downloads/.
- Seleccionar la descarga MySQL Community Server (Current Generally Available Release: 5.5.13)
- Descargar el fichero "Mysql-5.5.13-win32.msi" en el caso de ser una plataforma Windows con de 32 bytes. Se podrá elegir otras opciones.
- Ejecutar "Mysql-5.5.13-win32". Y aparecerá el MySQL Server 5.5 setup Wizard, pulsar "Next".
- Tras aceptar la licencia, seleccionar "Typical" en la ventana Choose setup type.
- Pulsar "Install" y el proceso de instalación comenzará.

Configuración de MySql:

- Tras la instalación se abrirá la ventana MySQL Enterprise, pulsar Next para continuar.
- Se ejecutará MySQLInstanceConfig.exe para configurar MySQL
- Aparecerá la ventana de configuración del servidor, marcamos Detailed Configuration y pulsamos Next.
- Aparecerá la configuración de instancia, marcar Developer Machine y pulsar Next.
- En la siguiente ventana marcar Multifunctional Database y pulsar Next.
- En la siguiente ventana pulsar Next.
- Marcar Decision Support (DSS)/OLAP y pulsar Next.
- Tras aceptar Enable TCP/IP Networking y Enable Strinct Mode pulsar Next.
- Marcar Standart Character Set y pulsar Next.
- En la siguiente ventana marcar, Install As Windows Service y Include Bin Directory in Windows PATH y pulsar Next.
- Para Finalizar pulsar Execute y FInish y MYSQL estará preparado.

Instalación de la maquina virtual de java

- Bajarse el programa en la página: www.java.com.es
- Hacer click en "Descarga gratuita de Java"
- Descargar Java para Windows: Recomendado Version 6 Update 26 (tamaño del archivo: ~ 11 MB)
- Ejecutar el archivo descargado "pxpiinstall.exe"
- La Maquina virtual de Java quedará instalada y preparada.

Instalando el MySQL Connector J

- Lo primero es ir a la http://www.mysql.com/downloads/connector/j/ y descargar la última versión (si es necesario).
- Escoger el tipo de archivo que convenga (por ejemplo el tar.gz si usamos Gnu/Linux, o zip si usamos Windows). Para una plataforma Windows mysql-connector-java-5.1.16.zip.
- Al descomprimir el archivo y se observa:
  - Los binarios ejecutables
  - Los archivos fuente
  - La documentación del conector
- Copiar el archivo JAR ejecutable, en el siguiente path: "\ruta_instalacion_java\jre\lib\ext". En Windows podría ser en "*C:\Program Files\Java\jre1.6.0_06\lib\ext*"
- Después de copiar dicho archivo, ya se puede usar el conector de MySQL desde la aplicación Java.

ANEXO D: Manual de Usuario

APLICACIÓN PROFESOR

PASO1. ACCESO PROFE

El usuario debe escribir su nombre correctamente en el campo de texto y pulsar ENTER o ENVIAR para poder acceder a la aplicación.

PASO2. ELEGIR OPCION

El usuario deberá elegir una de las siguientes opciones:

CREAR: crear nuevo problema
MODIFICAR: modificar problema existente
BORRAR: borrar problema existente
ALTA: dar de alta a un alumno

Escribiendo en el campo de texto una de las 4 opciones anteriores en el campo de texto, sin importar que sean mayúsculas o minúsculas.

PASO3. CREAR PROBLEMA

Para la correcta creación de un problema nuevo hay que seguir una serie de pasos. Se deben cumplimentar teniendo en cuenta una serie de reglas.

REGLA1.TIPO PROBLEMA

La elección del tipo se hará mediante la inserción de los números 1 ó 2.

REGLA2.ENUNCIADO

En el enunciado por cada parámetro de entrada que tenga el problema debe representarse un código a través del que se sustituirá el valor de dicho parámetro. Los códigos deben ser caracteres en mayúsculas en orden inverso al abecedario, es decir: Z, X, Y, W, etc. También será necesario definir en el enunciado las unidades del parámetro de salida, para que el alumno sepa en qué unidades debe representar el resultado.

REGLA3. PARAMERTROS

El número de parámetros de salida creados debe ser siempre 1.
El número de parámetros de entrada deberá ser como mínimo 2.
Para que el valor del parámetro sea representado en el enunciado el código definido deberá ser caracteres en mayúsculas en orden inverso al abecedario, es decir: Z, X, Y, W, etc. Y así relacionarse con los escritos en el enunciado.

REGLA4. PASOS

El número mínimo de pasos deberá ser 1, y en este paso estará definida la fórmula.
El último paso será en el que debe estar definida la fórmula.

REGLA5. FORMULA

Sólo podrá estar definida una fórmula.
El formato de la fórmula debe ser:

x = a operación b [operación c]

PASO4.MODIFICAR PROBLEMA

Para la correcta modificación de un problema almacenado hay que seguir una serie de pasos. Se deben cumplimentar teniendo en cuenta una serie de reglas. Estas reglas son las mismas que en la creación de problemas nuevos.

En cuanto a la elección del problema a modificar, irán apareciendo enunciados de problemas en orden de creación. Para seleccionarlos habrá que introducir "si" en el campo de texto cuando la aplicación lo requiera, con un "no", otro problema almacenado se mostrará. Una vez que la lista de problemas almacenados llegue a su fin, si no hemos elegido ninguno volveremos al menú inicial.

PASO5. BORRAR PROBLEMA

Se elegirán los problemas a borrara del mismo modo que la elección de los que se van a modificar, mediante la inserción de un "si" en el campo de texto, el problema mostrado será eliminado.

PASO6. ALTA ALUMNO

Para crear un perfil a un alumno que desee utilizar la herramienta habrá que introducir en el campo de texto el nombre, apellido y nombre de usuario de forma correcta según lo vaya exigiendo la aplicación.

APLICACIÓN ALUMNO

PASO1. ACCESO ALUMNO

El usuario debe escribir su nombre de usuario correctamente en el campo de texto y pulsar ENTER o ENVIAR para poder acceder a la aplicación. Una vez validado el nombre de usuario insertará la contraseña y si todo es correcto obtendrá el acceso.

Si es la primera vez que el usuario entra en la aplicación deberá definirse el mismo su contraseña. Deberá escribir en el campo de texto la contraseña elegida y tras pulsar ENTER o ENVIAR quedará asignada.

PASO2. RESOLVER PROBLEMAS

Para acceder o no a la aplicación será necesario introducir "si" y pulsar ENTER o ENVIAR

PASO3.TIPO PROBLEMA

La elección del tipo se hará mediante la inserción de los números 1 ó 2.

PASO4. RESULTADO

Tras aparecer el problema el usuario deberá escribir el resultado en el campo de texto. Si el resultado fuese decimal la inserción de la coma se podrá realizar mediante un punto o una coma indistintamente. Permitiendo al usuario un rango de error de menor que 0.1 en la obtención del resultado para considerar el problema correcto.

El problema sólo se considerará correcto si es resuelto a la primera, sin necesidad de mostrar ningún paso.

Una vez resuelto el problema se vuelve al paso 2.

ANEXO E: Cuestionario de satisfacción

Nombre: _____

Estudios: _____

Nivel informático: _____

Edad: ____

Aplicación:

- ☐ Profesor
- ☐ Alumno

Responda a las siguientes preguntas:

¿Qué le ha parecido la aplicación?

¿Qué aspectos destacaría de la aplicación?

¿Qué cambiaría de la aplicación?

¿Le ha resultado fácil manejar la aplicación?

¿Cree que es útil esta aplicación?

Indique la valoración de 1 a 10 de las siguientes preguntas:

	DISEÑO	RAPIDEZ	FACILIDAD MANEJO	OPINION GENERAL
VALORACION				