ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

Ingeniería Técnica Informática de Sistemas
Curso 2010-2011

Mobile Collaborative Argument and Trace Tool

Autor: Gustavo E. Astudillo Romero
Tutor: Luis Miguel Serrano Cámara

Agradecimientos

- A Fco. Román González Gómez y Miguel Ángel Domínguez Coloma por sus proyectos TACAC y MoCAS en los que el actual proyecto fin de carrera se basa como punto de partida.

- A mis padres, por su paciencia.

RESUMEN

El objetivo del presente proyecto ha sido la creación de un entorno colaborativo de aprendizaje de la programación. El trabajo de este proyecto fin de carrera (PFC) se basa en la fusión de funcionalidades de dos PFCs anteriores en el campo de la enseñanza de la programación de forma colaborativa. Además de la fusión de funcionalidades se ha realizado una adaptación tecnológica para mejorar su accesibilidad y un análisis sobre la usabilidad de los proyectos existentes que nos ha permitido identificar mejoras a implementar en el nuevo PFC.

Este PFC crea un entorno de trabajo para los alumnos en el que puedan resolver dos tipos de ejercicios de manera colaborativa, uno relacionado con la vigencia y ámbito de las variables de un programa y el otro con la depuración paso a paso de su ejecución.

Más detalladamente, la aplicación consta de dos funcionalidades según el actor para el que están diseñadas:

Primeramente la Aplicación del profesor: Esta aplicación permite al profesor gestionar los grupos de usuarios, así como los proyectos que están asociados a cada uno. Se ha mejorado tanto el proceso de generación de traza para depuración, permitiendo al profesor plantear soluciones más completas que hagan uso de la entrada por teclado a los programas; así también se ha corregido la generación automática de soluciones para ejercicios de visibilidad de variables, ya que generaba soluciones erróneas cuando la declaración de variables no se realizaba de una forma determinada que excluía muchos casos válidos aceptados por el compilador.

Una segunda funcionalidad es el cliente para el alumno: Se ha desarrollado una aplicación web que, comunicándose con la aplicación del profesor mediante servicios web, permite la interacción de los usuarios en los proyectos asignados por el profesor. La herramienta desarrollada proporciona una ayuda a la instrucción de las clases colaborativas en el aula y fuera de ésta. Proporciona a los estudiantes un entorno de aprendizaje colaborativo asistido por ordenador y a los profesores los dota de una herramienta de ayuda a la instrucción de clases colaborativas, donde se canaliza la discusión de los alumnos y se registra su participación.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Motivación

La razón de ser de este proyecto parte de las pruebas realizadas sobre TACAC y MoCAS, dos aplicaciones dependientes del entorno Windows Mobile y que sólo pueden ser ejecutadas en PDAs o SmartPhones que funcionen con dicho entorno.

Esto implica la necesidad de que los alumnos tengan este tipo concreto de dispositivo, y limita en gran medida lo que la aplicación puede ofrecer, por plataforma y por hardware.

Tanto TACAC como MoCAS cuentan con un cliente independiente (aunque comparten un mismo servidor), lo cual limita las posibilidades con las que cuenta el profesor para plantear ejercicios conjuntos.

Resulta necesario realizar un nuevo software que unifique ambas herramientas y esté desarrollado en una tecnología independiente del sistema operativo en el que se va a ejecutar, mejorando también el servidor para adaptarse a las necesidades de este nuevo software.

El presente PFC consiste en un software llamado MoCAT que realiza dichas funciones.

1.2. Objetivo

El objetivo del presente proyecto fin de cerrera (PFC) consiste en la fusión de dos prototipos actuales relacionados con el aprendizaje de la programación, uno con el ámbito y vigencia de operadores en un código fuente dado, y el otro con la traza de depuración de un programa.

Además de esta integración se realiza un cambio tecnológico hacia un entorno web que facilite el acceso a la herramienta vía diferentes plataformas hardware, eliminando la limitación existente en los prototipos anteriores con las PDA.

Analizamos los proyectos de partida mediante ingeniería inversa para localizar sus puntos débiles, mejorar la usabilidad del sistema y unir la aplicación TACAC (Traza de programas en ambientes colaborativos de aprendizaje con computación móvil) y MoCAS (Mobile Collaborative Argument Support) en un único cliente que permita al profesor plantear proyectos comunes y compartidos entre las dos herramientas.

TACAC es una aplicación ideada para desarrollar ejercicios de depuración de código de forma colaborativa entre varios alumnos, mediante el uso de un dispositivo PDA. Esta aplicación desarrolla una arquitectura cliente servidor, con una aplicación para el profesor (servidor) que mediante servicios web conecta con aplicaciones de los alumnos para PDA (clientes).

Por otro lado, MoCAS es un proyecto desarrollado a partir de TACAC que, haciendo uso de su estructura, amplía la funcionalidad de la aplicación del profesor y crea un nuevo cliente para la realización de ejercicios sobre el ámbito y vigencia de las

variables de un programa, creando un espacio de negociación (tabla de identificadores) donde los alumnos pueden proponer identificadores y discutir sobre las propuestas.

Al analizar la aplicación del profesor se ha concluido que, aunque es perfectamente aprovechable, deben solucionarse una serie de problemas para poder dar un servicio eficiente al profesor y a los alumnos, de esta forma se ha modificado lo siguiente puntos:

- Implementación de los servicios webs de comunicación con los clientes (aspecto básico de los proyectos anteriores) dado que en grupos amplios su uso era imposible por la ralentización que suponía que dichos servicios estuviesen serializados.
- La depuración de códigos, al no admitir entrada de datos externa generaba una limitación para las posibles trazas de depuración a generar, por lo que se modifica para admitir entradas externas.
- Al realizar pruebas con proyectos conjuntos de la aplicación de depuración y de visibilidad se observa que las soluciones automáticas generadas en código con "cuerpo" (no sólo estructura, también instrucciones) no son correctas. Se ha modificado por tanto el parseador de código para que permita este tipo de códigos de entrada.

En el software cliente (para los alumnos), el principal problema encontrado al analizar los proyectos anteriores es el empleo de la plataforma Windows Mobile para su uso por los alumnos.

Los dispositivos Windows Mobile no se encuentran entre los más usados en la actualidad. Las PDA, aun siendo asequibles en precio, no se encuentran entre las prioridades de un alumno genérico. La utilización de Windows Mobile en teléfonos móviles (muchos de la marca HTC, como la Diamond o el HD) se encuentra ahora más extendida, pero, aunque se puede ejecutar la aplicación en dichos terminales, sus características no son las más adecuadas para el objetivo del proyecto debido entre otras restricciones al tamaño de la pantalla, y a la limitación del .NET Compact Framework.

La tecnología .NET de Microsoft, aunque es un potente framework, no deja de ser software propietario con todas las restricciones que ello supone, por lo que también podría plantearse el abrir la aplicación a otras tecnologías más abiertas como puede ser Java. No en vano, no son necesarias licencias para desarrollar en Java, pudiendo descargar un Entorno Integrado de Desarrollo (IDE) de la calidad de Eclipse o NetBeans, mientas que para .NET la licencia más barata de la última versión de su IDE asciende a los 600 €.

Por tanto, una vez analizados los prototipos existentes se han identificado un conjunto de carencias a nivel tecnológico que necesitan ser solventadas. Estas carencias, la encorseta la tecnología utilizada en TACAC y MoCAS y nos llevan a considerar la realización de un nuevo prototipo que integre las actuales funcionalidades de MoCAS y TACAC en un entorno común basado en tecnología Web.

Estos objetivos desde el punto de vista tecnológico nos abren nuevas posibilidades en el plano funcional, ya que al no estar ceñidos a la tecnología .NET ni al entorno de las PDAs podemos dar al alumno una serie de funciones que antes era impensable, con el

fin de flexibilizar y potenciar la herramienta, y facilitar el intercambio de opiniones entre ellos a través de la aplicación.

1.3. Estructura de la memoria

Vamos a exponer a continuación como está organizada la presente memoria.

El capítulo segundo, Fundamentos, describe las tecnologías implicadas en los trabajos realizados. Hablamos en primer lugar de .NET, tecnología utilizada en los PFC en los que nos basamos, para después explicar las tecnologías que vamos a usar para el desarrollo del nuevo cliente, deteniéndonos especialmente en la plataforma Java y en Ajax, por ser las que permiten, por sus características, muchas de las funcionalidades que vamos a implementar.

En el tercer capítulo, Desarrollo, explicamos el proceso seguido desde que surge la idea de este proyecto hasta su finalización, desde el análisis de los prototipos TACAC y MoCAS hasta las pruebas realizadas sobre la implementación de la nueva aplicación.

El cuarto capítulo, Conclusiones, está dedicado a las reflexiones que hemos extraído tras este desarrollo y hacia donde deberían dirigirse el futuro trabajo que se realice tras él.

Por último, se han incluido cuatro anexos en los que se exponen los diagramas de clase del sistema, el diseño de la base de datos, cómo instalar la aplicación y una breve guía de uso.

2. FUNDAMENTOS

Como paso previo a la realización de un PFC es necesario evaluar las diferentes tecnologías disponibles en el mercado. En esta sección vamos a analizar brevemente dichas tecnologías prestando especial atención a las que mejor se adaptan a las funcionalidades de nuestro proyecto.

2.1. Framework .NET

Microsoft Framework .NET es la tecnología sobre las que están construidos los proyectos iniciales TACAC y MoCAS.

La tecnología .NET surge como una respuesta de Microsoft a la pujante Plataforma Java, pretendiendo ofrecer un entorno de trabajo en el que realizar aplicaciones de forma rápida, sencilla y económica que sean accesibles desde cualquier tipo de dispositivo.

Microsoft pone a disposición del desarrollador un kit de desarrollo (Software Development Kit, SDK) con el que ofrece todas las herramientas necesarias para la creación de una aplicación desde cero. Mediante su interfaz de usuario podemos crear proyectos (soluciones, dentro del mundo .NET) utilizando cualquiera de los lenguajes soportados por el framework, entre los que se incluyen C# (utilizado en los prototipos TACAC y MoCAS), C++, o Visual Basic, además de adaptaciones a .NET de otros lenguajes como Java, Fortran, o Ruby; es decir, Microsoft ha creado una plataforma en la que a partir de una gran variedad de lenguajes y paradigmas de programación podamos crear una aplicación que siga sus estándares.

Toda la arquitectura del framework .NET se sostiene gracias a la Common Language Infrastructure (CLI), que a partir del código compilado de cualquier de los lenguajes soportados por .NET es capaz de generar el código ensamblado que da lugar a los ejecutables que formarán la aplicación, como se describe en Figura 2.1.

Esta potente herramienta permite que el framework .NET sea tan versátil, ya que programadores de diferentes lenguajes de programación pueden usarla para sus desarrollos.

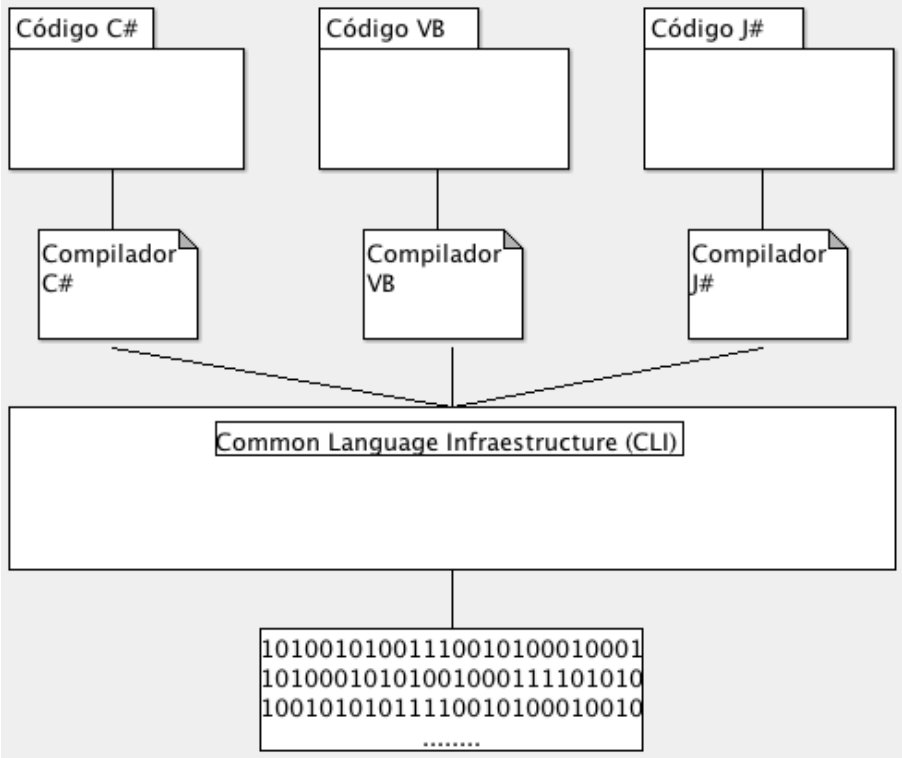

Figura 2.1 Common Language Infrastructure

2.2. Java EE

Java Enterprise Edition es una plataforma de programación especialmente preparada para el desarrollo y la ejecución de aplicaciones web mediante el lenguaje Java. Forma parte de la "Plataforma Java" desarrollada por Sun Microsystems (propiedad de Oracle), junto a Java SE (Standard Edition) y Java ME (Micro Edition).

La característica principal de esta plataforma se explica en su lema "*write once, runanywhere*" ("escribe una vez, ejecuta donde quieras"), dando prioridad a la portabilidad del código y permitiendo que una aplicación Java pueda ser ejecutada en los principales sistemas, como Windows, Linux o Mac. En el caso de las aplicaciones web, como la que nos ocupa, esto permite una fácil migración entre servidores o máquinas.

Java EE incluye gran cantidad de APIs que permiten desarrollar y coordinar fácilmente servicios comunes a otras plataformas, como el acceso a base de datos, envío de emails o el uso de servicios web. Además, cuenta con ciertas especificaciones propias, únicas de la plataforma Java, como Enterprise JavaBeans (EJB), Servlets y Java Server Pages (JSP). Todo esto facilita la labor del desarrollador a la hora de implementar una aplicación mediante esta plataforma.

La plataforma Java dispone de un compilador incluido en el kit de desarrollo (Java Development Kit, JDK) que traduce el código fuente Java a código interpretable por una máquina virtual de Java (en lo sucesivo JVM). Cada sistema dispone de una JVM propia, capaz de ejecutar este código intermedio en la máquina en la que está instalada, siendo esta la base de la portabilidad de la plataforma.

2.2.1. Entorno de ejecución para Java (JRE)

Existen dos componentes básicos e imprescindibles que debe tener cualquier sistema para poder ejecutar una aplicación Java: una máquina virtual de Java (Java Virtual Machine, o JVM) y un conjunto estándar de librerías. Sun Microsystems proporciona su propia máquina virtual junto con su propia implementación de estas librerías estándar, esto es lo que se conoce como entorno en tiempo de ejecución para Java (Java Runtime Enviroment, JRE).

Una Máquina Virtual Java (JVM) es una máquina virtual capaz de ejecutar *Java bytecode*, las instrucciones generadas por el compilador Java a partir del código fuente de la aplicación.

La JVM es una pieza básica dentro de la plataforma Java, puesto que gracias a ella es posible mantener la portabilidad entre sistemas. Una máquina virtual Java puede estar creada utilizando hardware, software, o a través de un navegador, el único requisito para que una implementación de JVM sea aceptada como válida por Sun es que sea capaz de ejecutar cualquier clase que cumpla la especificación.

Existen JVM para los principales sistemas operativos, incluyendo Windows, Linux o Mac (cuya JVM es desarrollada directamente por Apple), por lo que un desarrollador que trabaje sobre uno de estos sistemas puede compartir su trabajo con otro que trabaje en uno distinto sólo enviando sus .class (código Java compilado), razón por la que Java resulta tan versátil.

El otro componente básico de un JRE son las bibliotecas de Java estándar. Actualmente, la gran mayoría de lenguajes de programación recurren a bibliotecas dinámicas del sistema para realizar tareas bastante comunes como el acceso a ficheros o a la red.

La plataforma Java, con el fin de mantener la premisa de la portabilidad, suministra una serie de bibliotecas estándar con la mayoría de funciones que desempeñan los sistemas operativos actuales, desvinculándose así del sistema operativo donde ha de ejecutarse el código, y ofreciendo mayor libertad al desarrollador.

2.2.2. Métodos dinámicos para la creación de páginas web

Existen dos formas de generar páginas web de forma dinámica mediante el lenguaje Java, los Java Servlets y los JSP (Java Server Pages). Vamos a explicar las características de cada método y como se integran en la estructura de una aplicación web Java.

Una aplicación web Java está formada por una colección de ficheros que generan el contenido dinámico de la aplicación (JSPs y Servlets) y el contenido estático (ficheros HTML, CSS y JavaScript). Todos estos ficheros se presentan en un empaquetado único denominado WAR (Web Application Archive) que contiene todo lo necesario para ejecutar la aplicación.

Todo el contenido estático (HTML, CSS, JavaScript) y los JSP se ubicarán a partir del directorio raíz del WAR, pudiendo crear la estructura de directorios que el programador necesite con el fin de separar cada tipo de archivo, pero sin ser imprescindible hacerlo.

En el raíz del WAR ha de existir un directorio llamado WEB-INF, en él habrá un fichero llamado web.xml, que contiene elementos de seguridad de la aplicación así como información sobre los servlets que serán utilizados dentro de la misma, definiendo las URLs por las que se les podrá invocar.

Dentro del directorio WEB-INF se colocarán todos los servlets (ficheros Java compilados, .class) que contenga la aplicación, en un subdirectorio llamado classes. Si la aplicación necesita alguna librería Java no estándar es posible adjuntarla al WAR dentro de otro subdirectorio de WEB-INF llamado libs.

Una vez explicada la estructura de una aplicación web Java vamos a centrarnos en la generación de contenido dinámico. Como hemos dicho, el fichero web.xml contiene las rutas por las que es posible invocar cada uno de los servlets; los JSP podrán ser invocados como si se tratase de un fichero HTML, escribiendo directamente su ruta en el navegador.

Java Servlets

Los Java Servlets son objetos Java (ficheros .class) que generan contenido (generalmente HTML) de forma dinámica. Se ejecutan dentro de un contenedor de servlets (como Tomcat o Resin). El término servlet deriva de otro anterior, applet, que hacía referencia a pequeños programas Java que se ejecutaban dentro del contexto del navegador web, en contraposición a los servlets que se ejecutan en el del servidor.

Al incluir un servlet en una aplicación web se deben especificar las rutas a través de las que dará servicio en el fichero web.xml.

Para su funcionamiento, un servlet implementa la interfaz javax.servlet.Servlet, a través de la cual se accede a la implementación de cada JVM que permite aceptar peticiones a través de la red y responder a ellas.

Esta interfaz permite a la clase Java interpretar dos objetos que son parte de su implementación, los de tipo HttpServletRequest y HttpServletResponse, que contienen la información de la página que ha invocado al servlet.

Una vez invocado, el servlet recibe la petición como un objeto HttpServletRequest y procesa la información necesaria, ya sea extrayéndola de base de datos, almacenándola, o invocando servicios web, con el fin de poder devolver como retorno un objeto HttpServletResponse con la salida adecuada a la petición, que puede ser contenido de todo tipo.

JSP

Los JSP son una forma alternativa y simplificada de crear servlets. Estas páginas permiten la ejecución de código Java incrustado en ella mediante scripts, por lo que, al igual que los servlets, deben ser interpretadas por un contenedor de servlets o un servidor de aplicaciones.

La diferencia entre los servlets y los JSP radica en el enfoque de la programación, ya que mientras un JSP es una página web con etiquetas especiales y código Java

incrustado, un servlet es un programa Java puro que recibe peticiones y genera a partir de ellas una página web.

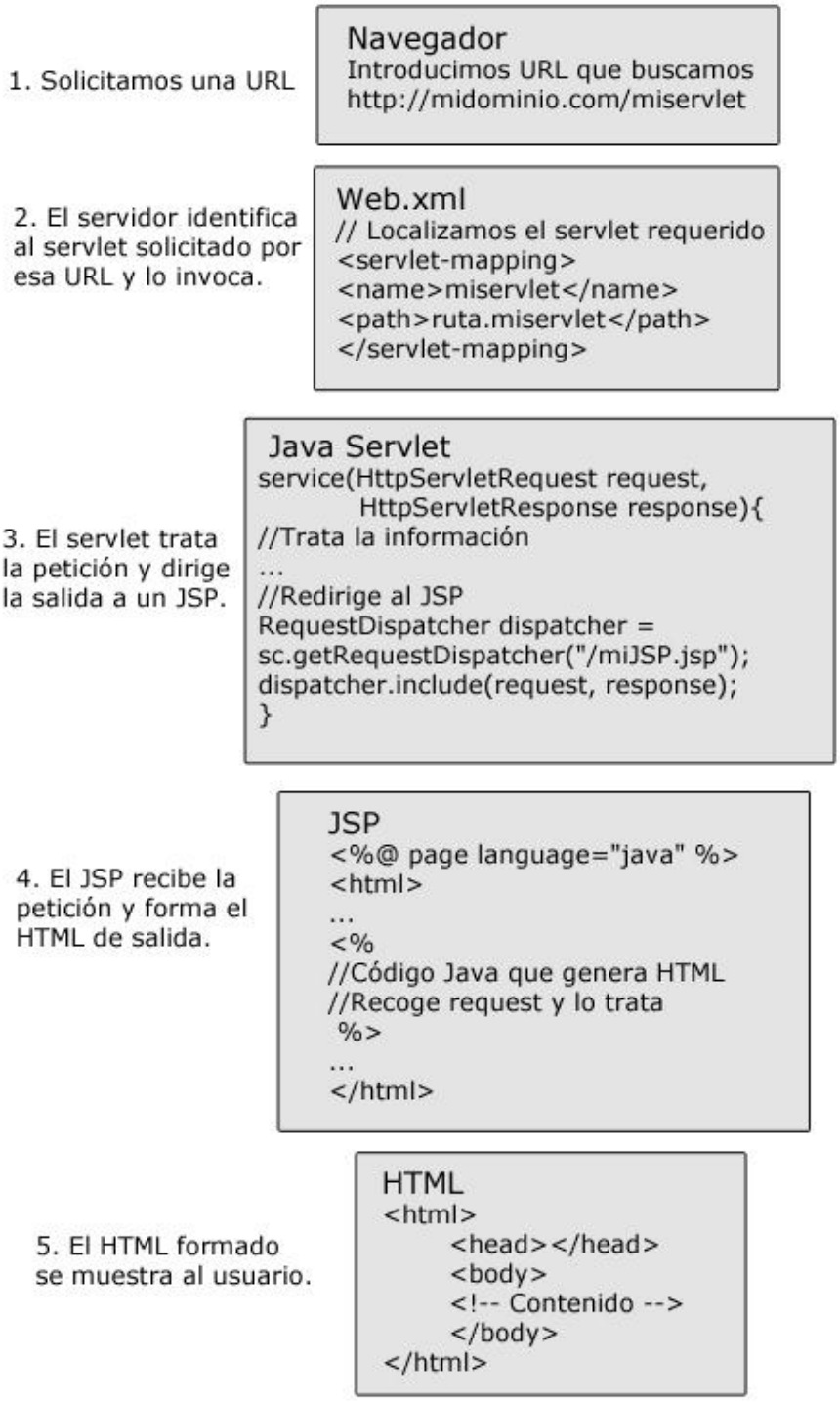

Figura 2.2 Esquema de generación de HTML con J2EE

Aunque es posible generar contenido HTML utilizando exclusivamente un servlet, estos suelen usarse en combinación con los JSP. Es habitual y una práctica recomendable tratar peticiones con un servlet y dirigir la salida hacia un JSP, generando el HTML final en dos etapas, como puede verse en Figura 2.2.

De esta forma aprovechamos la comodidad que nos da el servlet para acceder a base de datos o tratar la información recibida directamente en un fichero Java, y redirigimos una petición más clara al JSP, el cual tiene la ventaja de integrarse mejor con el HTML y hacernos más fácil construir la salida final de la petición.

2.3. Servicios Web SOAP

Un servicio web (en inglés web service, WS) es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos.

Para este proyecto se ha utilizado el protocolo SOAP (siglas de Simple Object Access Protocol), protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por David Winer en 1998 (Wikipedia, última visita junio 2011), llamado XML-RPC. SOAP que fue creado por Microsoft, IBM y otros y está actualmente bajo el auspicio de la W3C.

2.4. XML

Esta tecnología es un metalenguaje extensible de etiquetas (eXtensible Markup Language, XML) desarrollado por el World Wide Web Consortium (W3C) en 1996, aprobándose la especificación 1.0 del XML en febrero de 1998 (w3schools, última visita junio 2011).

Nace con el propósito de facilitar el intercambio de información entre diferentes plataformas de forma estructurada, por lo que resulta especialmente útil en Internet, donde, rodeado de otras tecnologías que hacen uso del mismo, permite abstraerse de los sistemas para los que se está desarrollando cualquier aplicación.

El XML es uno de los componentes de la web semántica, o web de los datos, cuya idea es añadir metadatos semánticos a la Word Wide Web (WWW), aumentando la interoperabilidad entre sistemas a través de “agentes inteligentes”, aplicaciones informáticas que recaban información sin operadores humanos. Un gran ejemplo de estos operadores es la tecnología de publicidad utilizada por Google, Google Ads, que a partir de la navegación de un usuario es capaz de mostrar anuncios adaptados al mismo.

Como lenguaje de etiquetas, XML resulta especialmente flexible, ya que permite añadir nueva etiquetas sin ninguna complicación, siendo esta una de sus grandes ventajas, ya que permite adaptarlo a cualquier comunicación, tan sólo respetando una sintaxis sobre su estructura, lo que define un documento XML bien formado.

Esta tecnología es especialmente importante para el proyecto debido a su uso en el intercambio de información entre cliente y servidor.

2.5. JavaScript

JavaScript es un lenguaje creado por Netscape en colaboración con Sun dada la necesidad de un término intermedio entre la sencillez de HTML y la complejidad de Java. JavaScript tiene una sintaxis similar a Java, aunque aparte de esto no tiene nada en común con dicho lenguaje.

Las principales ventajas que aporta son:

- Reduce el tráfico de comunicaciones en la red.
- Realiza operaciones en el propio navegador del cliente.
- Puede obtener información del navegador.

Se trata de un lenguaje interpretado, por lo que no existen problemas de compilación o de memoria. El código se transmite dentro del HTML, entre etiquetas <script>, y se ejecuta directamente en su navegador.

JavaScript es especialmente importante en el día de hoy por las capacidades que permite ofrecer a una aplicación web, sin las cuales no sería posible el avance que ha ocurrido dentro del mundo de internet, y todo lo que se ha llamado Web 2.0.

2.6. Ajax

Ajax, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una serie de técnicas para la creación de aplicaciones web interactivas que se ejecutan en el cliente, manteniendo una comunicación asíncrona en segundo plano con el servidor.

Estas aplicaciones, al ejecutarse en el cliente, permiten realizar tareas dinámicas que no serían posibles sin ellas, haciendo que el usuario tenga una experiencia más rica al acceder a un sitio web.

Una de las características de Ajax es que no requiere plugins de ningún tipo para funcionar, tan sólo depende de las características y versión del navegador en el que se esté ejecutando para funcionar. Hoy en día Ajax es soportado por la gran mayoría de navegadores, como Internet Explorer (desde la versión 5.0), Mozilla Firefox o Safari.

El funcionamiento de Ajax se sustenta en la comunicación asíncrona, posible gracias al objeto XMLHttpRequest (como se puede observar en Figura 2.3), por lo que vamos a explicar ambos conceptos a continuación, sin los cuales no es posible entender esta tecnología.

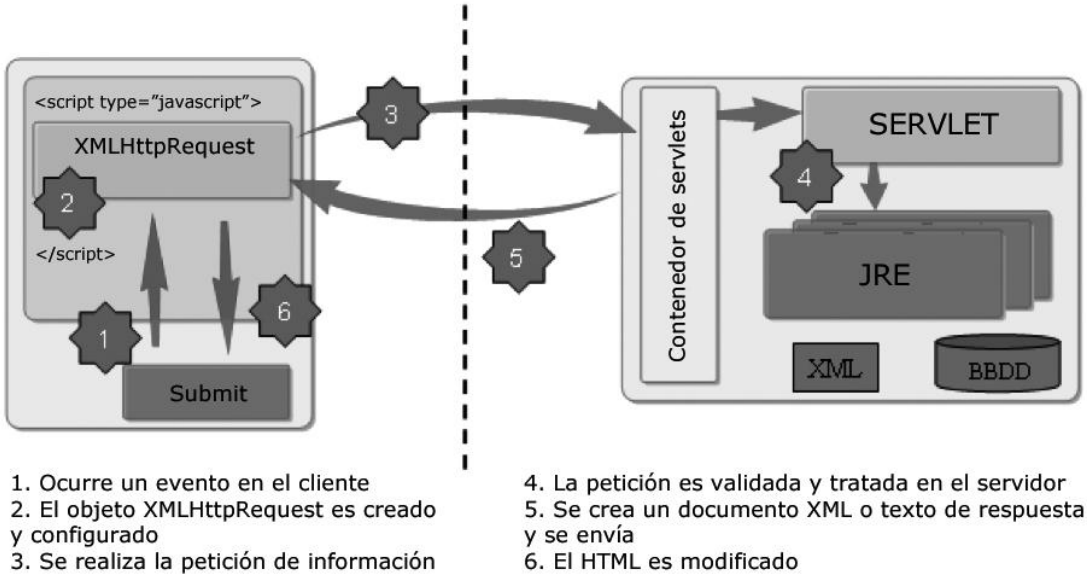

Figura 2.3 Esquema Ajax

2.6.1. Funcionamiento asíncrono

Ajax funciona bajo una estructura cliente-servidor, un modelo típico de las aplicaciones web, pero, como ya se ha indicado, funcionando de forma asíncrona. En aplicaciones web clásicas cliente-servidor el funcionamiento es síncrono, lo cual supone que el usuario hace peticiones al servidor de forma continua y debe esperar la respuesta de éste para continuar y recargar los datos solicitados. En este tipo de aplicaciones, cuando se realizan muchas peticiones el cliente tiene mucho tiempo de espera lo cual genera una sensación de mal funcionamiento en los usuarios finales.

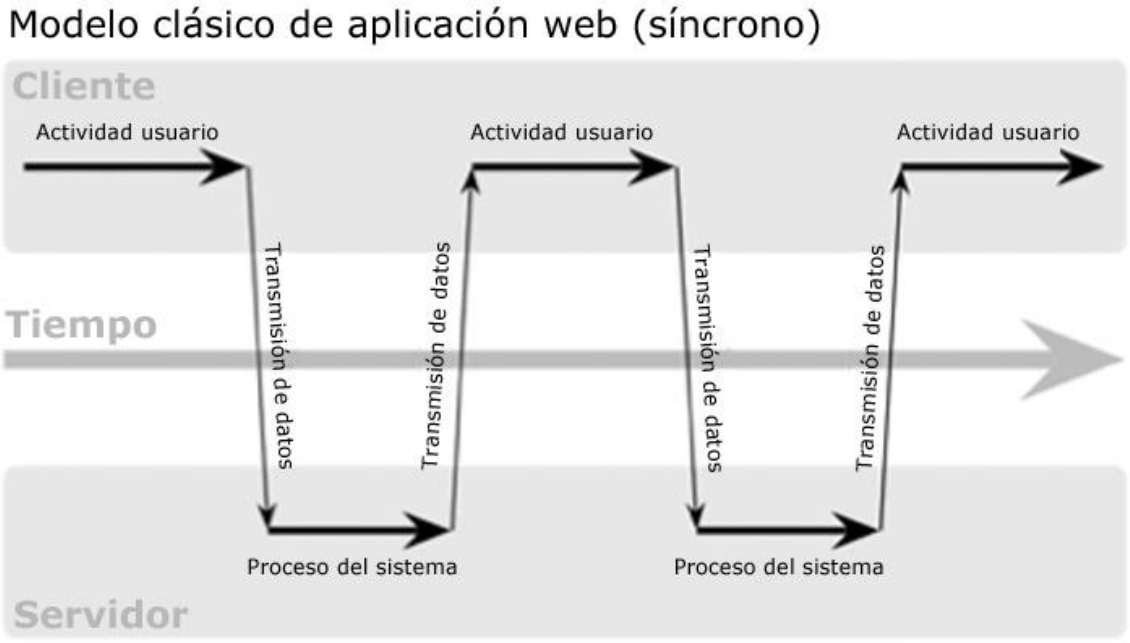

Figura 2.4 Modelo clásico síncrono

Ajax permite mejorar la comunicación entre cliente y servidor, consiguiendo que la interoperabilidad entre usuario y aplicación sea más fluida y más activa, todo ello mediante comunicaciones asíncronas que evitan la recarga continua de la página para poder actualizar los datos que muestra.

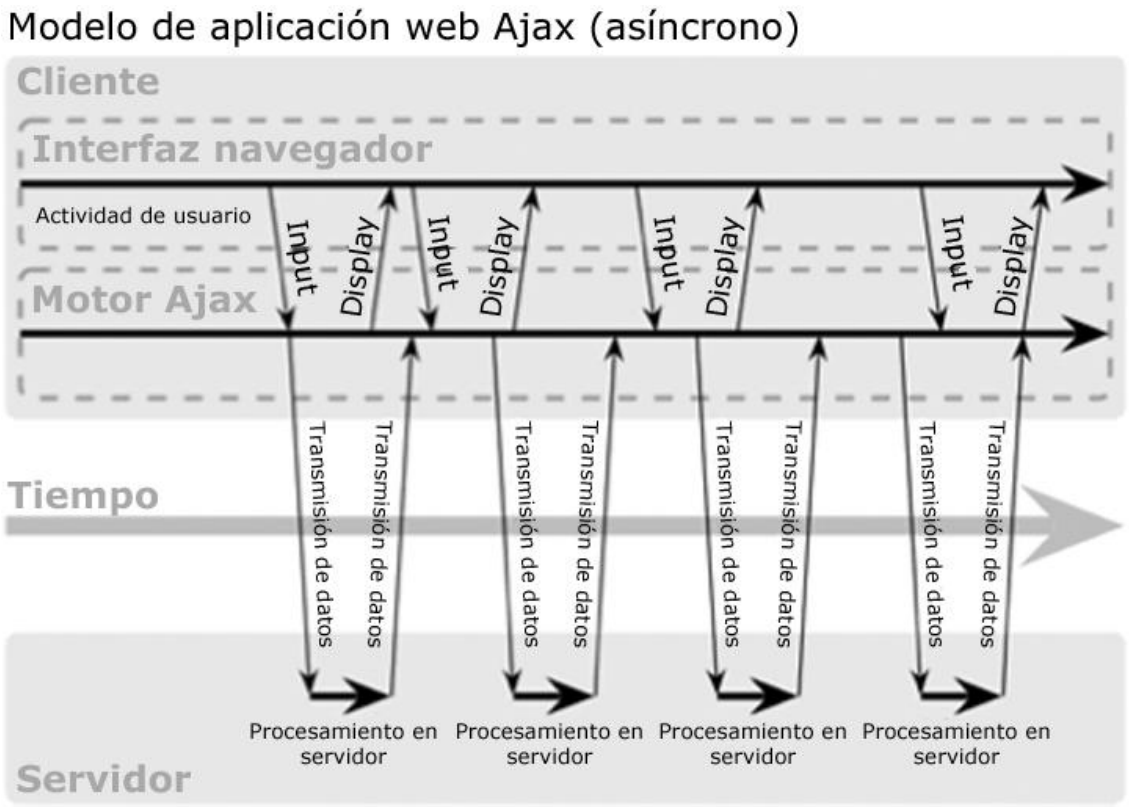

Figura 2.5 Modelo Ajax asíncrono

Gracias a esto Ajax se ha impuesto a tecnologías como Flash y Applets, existiendo multitud de aplicaciones que hacen uso de ella, como por ejemplo, GMail.

2.6.2. XMLHttpRequest

El objeto XMLHttpRequest es la piedra fundamental de las comunicaciones entre cliente y servidor en una aplicación Ajax. Sus primeras versiones fueron desarrolladas por Microsoft para Internet Explorer 5.0 como un ActiveX (componente software reutilizable), pero fue incorporándose con el paso del tiempo a otros navegadores como Firefox hasta que finalmente la W3C lo presentó en 2006 como una interfaz estándar, aunque aún sin aprobar.

La funcionalidad de objeto es realizar comunicaciones cliente-servidor de manera asíncrona o síncrona, a elección del desarrollador, permitiendo emitir y recoger mensajes de ambas formas. Su funcionamiento se describe en Figura 2.6.

La creación del objeto varía entre navegadores, por lo que debe controlarse donde se está ejecutando la aplicación, pero una vez creado el objeto todos se comportan de la misma forma respecto al objeto y no es necesario diferenciar ninguno de ellos.

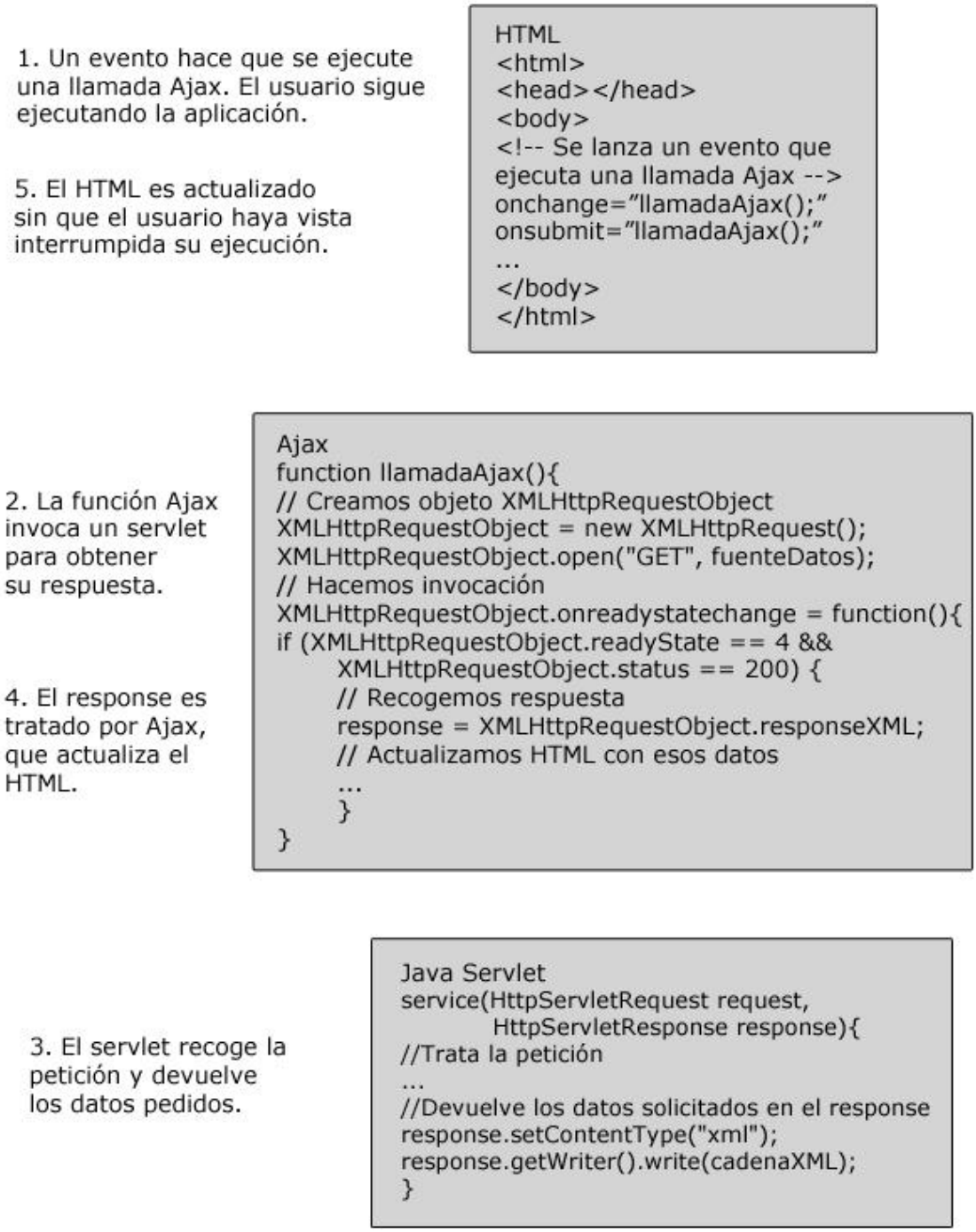

Figura 2.6 Esquema de funcionamiento Ajax

2.6.3. Prototype

Para facilitar el desarrollo de aplicaciones con Ajax existen varias librerías de JavaScript que facilitan tareas como la creación del objeto XMLHttpRequest, realizando todas las validaciones necesarias acerca del navegador en el que se ejecuta y tareas

similares. Gracias a esto el programador puede abstraerse de dichas tareas y centrarse en la funcionalidad a conseguir, por lo que resultan de gran ayuda.

Prototype es un framework escrito en JavaScript por Sam Stephenson (Dupont, 2008). Está compuesto por varias librerías, que además de resolver muchos problemas de compatibilidad entre navegadores da soporte a otras librerías de efectos (como script.aculo.us).

Además, Prototype simplifica muchas secuencias de JavaScript y facilita enormemente trabajar con el DOM (Modelo de Objetos del Documento), por lo que es una herramienta verdaderamente útil para cualquier desarrollador de aplicaciones web.

3. DESARROLLO

Una vez hemos enumerado el conjunto de tecnologías disponibles para implementar nuestro PFC, hemos decidido combinar el uso de la plataforma Java con Ajax por los siguientes motivos:

1. La portabilidad de la plataforma Java es un factor importante, ya que queremos lograr que la aplicación sea completamente accesible, y esto nos da la capacidad de montar la estructura del cliente en cualquier tipo de máquina.

2. Java nos da las herramientas necesarias para acceder fácilmente a los datos del sistema, ya sea mediante servicios web o directamente a la base de datos, y prepararlos para ser mostrados en la interfaz del usuario.

3. Por lo extendido que está Ajax hoy en día en todos los navegadores lograremos que el cliente sea accesible desde todo tipo de dispositivos, ya que no se precisa ningún otro tipo de instalación.

4. Ajax nos facilitará reducir el tráfico en la red, algo imprescindible dado que la comunicación entre los clientes debe ser lo más fluida posible, sin que ello repercuta en el rendimiento de la aplicación para el usuario.

5. Por último, Java y Ajax se complementan perfectamente, ya que Ajax nos permite realizar fácilmente las llamadas al servidor desde el lado del cliente y Java da funcionalidad a esas llamadas.

Tras explicar las tecnologías utilizadas vamos, a continuación, a enumerar los pasos que hemos dado hasta la solución final, partiendo del análisis de los prototipos iniciales TACAC y MoCAS.

3.1. Análisis de la situación actual

El primer paso de nuestro desarrollo ha sido analizar las aplicaciones que ya teníamos de PFC anteriores buscando los puntos que debían mejorarse, ya sea por fallos encontrados en su ejecución o para mejorar la usabilidad de los mismos.

Por ello, hemos realizado un trabajo de ingeniería inversa para entender como se había realizado la implementación de los PFC anteriores. Hemos analizado la usabilidad del sistema para definir sus puntos débiles a mejorar, y a partir de ello hemos propuesto una solución.

3.1.1. Ingeniería inversa

La ingeniería inversa es un proceso por el que, partiendo de un producto acabado, se analiza con el fin de entender su funcionamiento y, en este caso, tratar de mejorarlo. Es una técnica usada en diversos campos, pero especialmente común en la informática

Partiendo de los manuales de instalación de los proyectos anteriores hemos creado un entorno de pruebas, y ayudados por herramientas de captura de tráfico de red (como WinPcap (WinPcap, última visita junio 2011)) se ha analizado la comunicación entre la aplicación del profesor y el cliente PDA.

Durante la configuración del entorno de pruebas encontramos complicaciones en el momento de conectar el servidor con una PDA. Para establecer la comunicación ha sido necesario configurar los puertos de cada dispositivo como abiertos, debido a la implementación del cliente PDA. Consideramos esto una fuente de posibles errores cuando se quiera utilizar en una clase real con un número elevado de alumnos.

Se han realizado un conjunto de pruebas atendiendo al número de dispositivos involucrados en las mismas:

a) Pruebas con PDA+ Servidor: Hemos podido observar un funcionamiento normal y conseguimos realizar un ejercicio completo de manera satisfactoria. Al analizar las conexiones realizadas entre cliente y servidor vemos que la aplicación del profesor va a condicionar demasiado el rendimiento del sistema, y que puede ser un cuello de botella en la comunicación entre clientes.

b) Prueba de carga con N PDA + Servidor: En esta prueba hemos podido constatar cómo los problemas detectados en la prueba inicial (apartado a) se han confirmado, ya que hemos encontrado números problemas de transmisión de mensajes entre las PDA. Hemos preparado un ejercicio de depuración con dos dispositivos, una PDA y un SmartPhone HTC Diamond, con el objetivo de probar no sólo las comunicaciones entre cliente y servidor, sino entre los clientes. Queremos validar el funcionamiento del chat, del control de la traza, y de las variables. Los resultados que obtuvimos no fueron satisfactorios, ya que encontramos multitud de incoherencias como mensajes de chat que no llegaban a su destino, avances en el paso de traza que no se reflejaban en el resto de usuarios, control de traza dividido cuando esto no debería estar permitido.

Estudiando estos problemas concluimos que la comunicación cliente/servidor implementada no es apropiada, y es la responsable de estas pérdidas, error importante y que se debe corregir para que la herramienta sea funcional.

3.1.2. Informe de usabilidad

La usabilidad es la facilidad con que las personas pueden utilizar una herramienta particular con el fin de alcanzar un objetivo concreto (Wikipedia, última visita junio 2011).

En el ámbito persona-ordenador, ésta se refiere a la claridad y elegancia con que se diseña la interacción con un programa de ordenador o un sitio web.

El grado de usabilidad de un sistema es, por su parte, una medida empírica y relativa de la usabilidad del mismo.

- Empírica porque no se basa en opiniones o sensaciones, sino en pruebas de usabilidad realizadas en laboratorio u observadas mediante trabajo de campo.

- Relativa porque el resultado no es ni bueno ni malo, sino que depende de las metas planteadas o de una comparación con otros sistemas similares.

Vamos a exponer los principios generales de la usabilidad aplicada al software, y analizar cómo se ajusta la aplicación a cada uno de esos puntos, como se puede ver en Tabla 3.1.

Principio general	**Definición**	**Análisis realizado**
Facilidad de aprendizaje	Mide el tiempo requerido por un usuario para alcanzar un uso productivo de la aplicación.	La interface de usuario de la aplicación del profesor es muy clara y sencilla de usar. Tiene detalles mejorables que harían más intuitiva la aplicación, pero cualquier usuario genérico puede hacerse fácilmente con el control de la misma sin necesidad de un aprendizaje o tutorial previo. La del alumno, sin embargo, es más complicada de usar y precisa que el usuario reciba unos conceptos básicos sobre su funcionamiento antes de usarla. Esto implica que para un grupo de usuarios normales se necesitará un tiempo dedicado al aprendizaje de la aplicación.
Consistencia	Un sistema es consistente si todos los mecanismos que se utilizan son siempre usados de la misma manera.	En la aplicación del alumno, dado que no todos los botones deben funcionar igual dependiendo de si el alumno tiene o no el control de la depuración en ese momento, tenemos situaciones que no se adaptan a estas características. Se debería deshabilitar dichas opciones, o diferenciar claramente en la interface del usuario estas situaciones con el fin de no inducir al usuario a error, y de no crear situaciones en las que distintos usuarios tengas percepciones diferentes de lo que debe hacer la aplicación en un momento dado.
Flexibilidad	Multiplicidad de maneras en que el usuario y el sistema intercambian información.	La aplicación resulta ser muy rígida, ya que las formas en que se intercambia información son limitadas y acotadas, con lo que se consigue un mayor control sobre las acciones que puede realizar el usuario, a costa de la flexibilidad y de la experiencia educativa que podrían conseguir los alumnos, ya que las limitaciones que marca la aplicación no permiten una verdadera resolución

		conjunta de un problema, un trabajo en grupo, sino un trabajo revisado, en el que los usuarios pueden ir "pasando el cuaderno" entre ellos, pero combinar sus esfuerzos de forma conjunta.
Robustez	Capacidad de cubrir las características para poder cumplir sus objetivos y su asesoramiento.	Desde el punto de vista funcional, los objetivos mínimos necesarios para que la aplicación sea útil, entendiendo por útil el permitir una depuración en común por diversos alumnos en una clase, están cumplidos, sin olvidar que, como hemos expuesto en el estudio de ingeniería inversa del proyecto, estos no estén implementados correctamente y se encuentren errores. No obstante, existen diversas características que aumentarían las posibilidades de la aplicación, por lo que la robustez de la aplicación consideramos que es mejorable.
Recuperabilidad	Grado de facilidad que una aplicación permite al usuario para corregir una acción una vez está reconocido un error.	Éste es un factor a mejorar en la aplicación, principalmente en la parte del alumno, introduciendo un mayor control de errores en tiempo de ejecución, dando mayores posibilidades al alumno para volver sobre sus pasos en caso de equivocarse y más información sobre la ejecución.
Tiempo de respuesta	Tiempo que necesita el sistema para expresar los cambios de estado del usuario.	Los tiempos de respuesta en las pruebas no han sido correctos, ya que en muchas ocasiones se han perdido mensajes y ha resultado en incoherencia de datos entre los clientes conectados. Es un punto crítico a mejorar.
Adecuación de las tareas	Grado en que los servicios del sistema soportan todas las tareas que el usuario quiere hacer y la manera en que éstas las comprenden.	Es innegable que el usuario va a querer realizar acciones que la aplicación no le va a permitir, tanto a la hora de comunicarse con el resto de usuarios con los que comparte esa ejecución, algo básico si queremos que se tenga una experiencia compartida, como en la participación de la resolución del problema, ya que sólo un alumno tiene el control en cada momento. Esto no quiere decir que las tareas realizadas por la aplicación no se completen de forma correcta y adecuada,

		pero existen tareas que pueden ser interesantes y necesarias tanto para el alumno como para el profesor que no están implementadas en el sistema.
Disminución de la carga cognitiva	Los usuarios tienen que confiar más en los reconocimientos que en los recuerdos y no necesitar recordar abreviaciones y códigos muy complicados.	El alumno deberá recordar el aprendizaje realizado acerca de la aplicación que va a usar, dado que existen una serie de botones en la interface que no están etiquetados de una forma clara, pero no son de ningún modo una cantidad preocupante.

Tabla 3.1 Informe de usabilidad

3.1.3. Propuesta de proyecto

Por todo lo expuesto anteriormente, podemos concluir que es apropiado tomar como punto de partida los proyectos anteriores e iniciar el desarrollo de un nuevo cliente que corrija los errores encontrados, recoja la funcionalidad ofrecida actualmente y aporte nuevas funcionalidades y mayor usabilidad.

Dada la arquitectura existente, se propone aprovechar la aplicación del profesor para la gestión de grupos y usuarios, comunicándose con el nuevo cliente mediante los servicios web. Esto nos permite cambiar de tecnología, y no estar limitados a .NET y a las PDAs.

En este contexto, una aplicación web es sin duda la solución óptima como medio para la interacción de estudiante y sistema, puesto que se evitan una serie de problemas que tendríamos con una aplicación de escritorio, como incompatibilidades por sistema operativo o problemas de configuración (Boroni, 2001).

En definitiva, esta propuesta pretende desarrollar un sistema que, corriendo en un sistema de Windows (limitación dada por la aplicación del profesor) pueda ser accedido desde prácticamente cualquier máquina que use el alumno, independientemente del sistema que utilice (Windows, Mac, Linux, Android…), como se expone en Figura 3.1. Esta portabilidad resulta especialmente útil para permitir al usuario trabajar con su propio equipo, o más adelante, y con la infraestructura adecuada, trabajar en grupos descentralizados, cada cual en un campus o en sus domicilios, fomentando el trabajo en grupo a distancia.

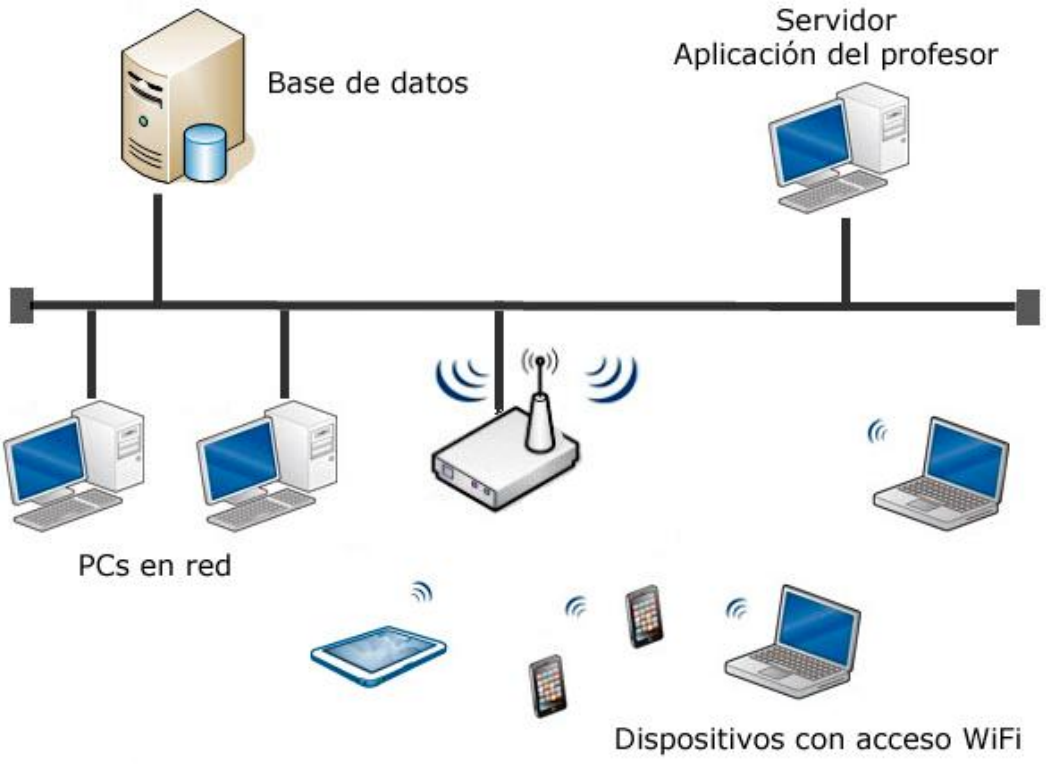

Figura 3.1 Esquema del sistema

3.2. Requisitos

La toma de requisitos es el proceso por el que usuario y analista/desarrollador reúnen las características que debe cumplir una aplicación software. Es importante la implicación de las dos partes en este proceso, ya que mientras el usuario puede, y debe, aportar toda la información necesaria acerca de que espera de la aplicación, el analista/desarrollador tiene una visión técnica para analizar la viabilidad de las peticiones del usuario, y, en muchos casos, añadir funcionalidades útiles a partir de su experiencia en el desarrollo, aunque no tenga tanta experiencia en el campo de negocio de la aplicación como el usuario.

3.2.1. Descripción del problema

El objetivo de este desarrollo es mejorar la aplicación del profesor ya existente, e implementar un nuevo cliente para los alumnos que englobe tanto la parte de depuración de código (TACAC) como la visibilidad de variables (MoCAS).

Aplicación del profesor:

Los cambios a realizar sobre la aplicación del profesor tienen el fin de mejorar el rendimiento del sistema y ampliar la funcionalidad del mismo. Se marcan las siguientes necesidades y carencias:

- Mejora de rendimiento, cambios en el tratamiento de los servicios web para evitar la serialización.

- Ampliación de funcionalidad en la depuración, se procederá a mejorar el sistema de depuración de código para que al generar una traza de depuración se acepten entradas externas, para aumentar las posibilidades del profesor a la hora de plantear un ejercicio.
- Corrección de método para la obtención de la solución automática, para conseguir que en programas con cuerpo, y no sólo estructura, se consigan los valores correctos.

Cliente para el alumno:

El cliente que va a desarrollarse para el alumno es completamente nuevo. Aunará las aplicaciones existentes para los ejercicios de depuración y visibilidad en una sola, y permitirá combinar ejercicios conjuntos entre ellas.

Necesidades generales:

- El tipo de cliente será una aplicación web. Es prioritario ampliar al máximo el tipo de dispositivos que puedan acceder al mismo para llegar a todos los alumnos posibles.
- El alumno accederá a la aplicación mediante una URL que le presentará la pantalla de login, donde deberá identificarse en el sistema. No podrá acceder a ninguna parte de la aplicación si no se identifica correctamente.
- Una vez validado, el usuario deberá elegir entre varios combos enlazados (Group, Project, SubGroup, Inputs) con los que se va a trabajar.
- Un proyecto de depuración valido estará definido por grupo, proyecto y entrada de datos; uno de visibilidad por grupo, subgrupo y proyecto.
- Según los combos que rellene el usuario se le presentará el acceso a una u otra aplicación, o a ambas si procede.
- Si el usuario elige valores para los cuatro combos se le permitirá acceder tanto a la sección de depuración como a la de visibilidad, pudiendo saltar de una a otra, compartiendo datos comunes entre ellas, tales como comentarios o líneas de chat.

Necesidades depuración:

- La interfaz de usuario mostrará el código a depurar, la información de los valores de cada variable en cada paso de la traza, un marco de chat con los usuarios conectados, e información acerca de la traza depurada (descripción del ejercicio por el profesor y entrada de datos).
- El alumno podrá dejar comentarios en cada línea de código, haciendo un clic de ratón en la línea deseada. La línea mostrará entonces un icono para indicar que tiene comentarios añadidos.
- Para visualizar los comentarios bastará con poner el cursor del ratón sobre el icono de comentarios de una línea.
- El recuadro del chat mostrará las aportaciones de los usuarios conectados a la misma traza que el cliente, junto a un listado de todos los usuarios conectados.
- El cuadro de mandos informará acerca del paso de traza actual y línea de ejecución actual, así como disponer botones para avanzar en el paso de traza (no se permitirá volver atrás), y ampliar y disminuir el marco de variables.

Necesidades visibilidad:

- El usuario podrá visualizar código de ejercicio, ventana de chat, y cuadro de aportaciones del grupo.
- Deberán compartirse las aportaciones del grupo en tiempo real, permitiendo así una resolución colaborativa del ejercicio.
- El alumno podrá aportar valores a la solución, e igualmente podrá confirmar o desconfirmar los valores aportados por otros miembros del grupo.
- El alumno dispondrá de comentarios en las líneas de código con el mismo funcionamiento que en la parte de depuración, compartiendo los comentarios entre ambas aplicaciones.
- El chat entre usuarios tendrá el mismo comportamiento que los comentarios.

3.2.2. Especificación de requisitos

Tras definir junto al usuario una serie de necesidades que ha de satisfacer la aplicación vamos a establecer los requisitos de la aplicación, a partir de los cuales se realizará el análisis y el diseño que nos llevará a la implementación deseada.

Existen tres tipos de requisitos:

- Funcional (RF): Aquellos requisitos que el usuario necesita que realice la aplicación, tales como emisión de mensajes, o como gestionar determinada información.
- No funcional (RNF): Se incluyen en este apartado los recursos técnicos necesarios para que la aplicación desempeñe su función, como tipo de BBDD, o servidores a usar.
- Organizacionales (RO): Definen el marco sobre el que se va a implantar la aplicación para establecer un objetivo, como puede ser maximizar el número de usuario que puedan acceder a la aplicación.

La creación del listado final de requisitos es un proceso largo que precisa de mucha comunicación entre cliente y proveedor. Partiendo de las necesidades definidas anteriormente el proveedor define un primer listado que debe ser debatido y corregido con el cliente para consensuar un listado final que recoja de manera óptima todas las necesidades del usuario y que técnicamente sea una solución buena; es por ello imprescindible que las dos partes se impliquen concienzudamente en este punto del desarrollo para prevenir futuros cambios de alcance o mantenimientos problemáticos.

Exponemos en las siguientes tablas los requisitos finales definidos para este proyecto.

Los requisitos que se han planteado están enfocados hacia la unión de los dos prototipos anteriores, TACAC y MoCAS, en un nuevo cliente web.

Dado que el presente proyecto incluye las funcionalidades ofrecidas por TACAC y MoCAS algunos de los requisitos son comunes a los de dichos prototipos. Marcaremos en cursiva estos casos, manteniendo en texto normal los nuevos requisitos de este PFC:

Requisito	Descripción
RNF1	Los servicios web ofrecidos no deben ser serializados para eliminar problemas de rendimiento con un número de usuarios elevado.
RF1	La generación de trazas debe aceptar entrada de datos para ofrecer más opciones al profesor.
RF2	La solución automática debe generarse correctamente para cualquier código que compile sin errores.

Tabla 3.2 Requisitos aplicación del profesor

Los tres requisitos planteados para la aplicación del profesor en Tabla 3.2no existían en los prototipos anteriores.

Requisito	Descripción
RO1	El cliente debe ser una aplicación web, debe ser accesible desde cualquier dispositivo capaz de navegar por la red.
RF1	*Ningún usuario podrá acceder al sistema sin identificarse mediante su usuario y contraseña.*
RF2	Las secciones de Depuración y Visibility deben poder compartir información común tal como comentarios o chat entre usuario.
RF3	El sistema ofrecerá acceso a cada una de las secciones, o a ambas, en función de la selección de Grupo, Proyecto, Subgrupo y Entrada de Datos efectuada por el usuario.
RF4	El sistema almacenará toda la información generada en la resolución de un ejercicio, incluyendo comentarios de código, líneas de chat, aportaciones sobre valores de variables, y sobre valores de una solución.
RNF1	*El sistema accederá a los datos de la aplicación del profesor exclusivamente mediante servicios web.*
RNF2	El sistema contará con tablas propias en una BBDD MySQL.

Tabla 3.3 Requisitos generales del cliente

El nuevo cliente web comparte con los prototipos anteriores dos requisitos expuestos en Tabla 3.3, el RF1 y el RNF1. En cuanto al RF1, se mantiene porque el nuevo cliente no va a permitir ninguna acción sin identificarse en el sistema, algo imprescindible. El RNF1 hemos decidido mantenerlo para que el cambio de cliente resulte invisible a la aplicación del profesor, para ello sólo accederemos a sus datos mediante los servicios web con los que nos provee.

Requisito	Descripción
RF1	*El cliente podrá acceder a cualquier proyecto creado en la aplicación del profesor.*
RF2	La interfaz de usuario permitirá intercambiar información de forma sencilla con otros usuarios, a través de comentarios sobre el código, o mediante chat.
RF3	*Cualquier acción sobre la traza a depurar, o información añadida por cualquier usuario, debe ser radiada en tiempo real al resto.*
RF4	La interfaz permitirá pasar a la sección de Visibility desde Depuración si los datos seleccionado por el usuario (Grupo, Proyecto, Subgrupo) así lo permiten.

Tabla 3.4 Requisitos sección Depuración

Requisito	Descripción
RF1	*El cliente podrá acceder a cualquier solución creada en la aplicación del profesor.*
RF2	La interfaz de usuario mostrará toda la información relativa a la solución, tales como valores de la misma, diferenciación de aportaciones propias o del resto de alumnos, y totales de confirmaciones y desconfirmaciones.
RF3	*Cualquier acción realizada sobre la solución por un alumno deberá ser transmitida al resto en tiempo real.*
RF4	La interfaz permitirá pasar a la sección de Depuración desde Visibility si los datos seleccionado por el usuario (Grupo, Proyecto, Entrada de datos) así lo permiten.
RF5	Los comentarios sobre el código y el chat entre usuarios será compartido entre secciones de Depuración y Visibility si es posible pasar de una a otra.

Tabla 3.5 Requisitos sección Visibility

En cuanto a las secciones de los ejercicios, exponemos sus requisitos en Tabla 3.4 y Tabla 3.5, Depuración y Visibility respectivamente.

Los requisitos RF1 y RF3 de dichas tablas son los mismos y ya existían en TACAC y MoCAS. Se mantienen para indicar la necesidad de que el nuevo cliente pueda acceder a cualquier ejercicio propuesto desde la aplicación del profesor (RF1) y que cualquier cambio realizado por un alumno en un ejercicio debe ser transmitido a sus compañeros de grupo (RF3).

3.2.3. Casos de uso

Los diagramas de casos de uso se usan para definir las interacciones entre los usuarios y el sistema.

En un caso de uso existen distintos elementos:

- Actor: Cualquier entidad externa al sistema que interactúe con el mismo. Se representa con una figura esquemática humana.
- Caso de uso: Funcionalidad del sistema. Es representada con una elipse.
- Relaciones: Asociaciones existentes entre actores y el sistema. Representadas por flechas.

Mediante los diagramas vamos a representar de forma gráfica los requisitos que hemos definido, los límites del sistema, y los diferentes tipos de usuario que pueden acceder y usar la aplicación.

Diagrama de casos de uso del sistema

Vamos a definir el funcionamiento global del sistema, incluyendo la aplicación del profesor y el nuevo cliente. El sistema consta de dos partes bien diferenciadas, la anteriormente existente aplicación del profesor, de la que se obtendrán datos mediante servicios web, y el nuevo cliente que agrupa la funcionalidad de Depuración junto a la de Visibility, acoplándolas e incluyendo nuevas formas en que los alumnos pueden trabajar juntos. Todo ello define un entorno de trabajo fácil de usar, agradable, e intuitivo.

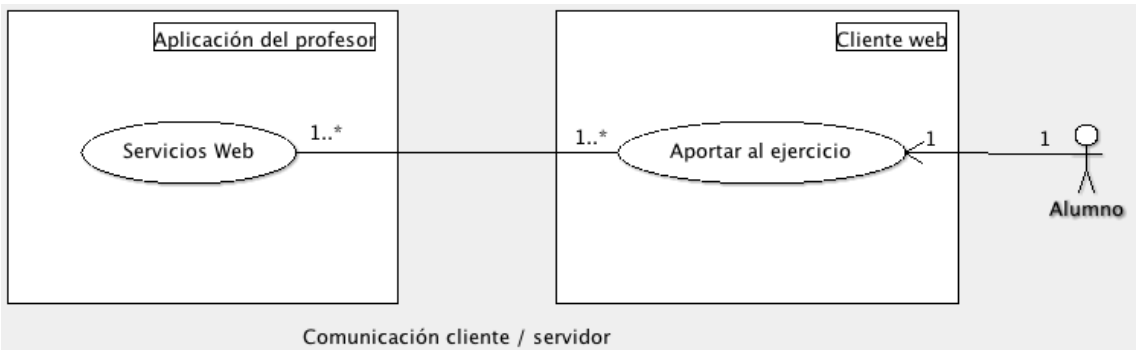

Figura 3.2 Casos de uso del sistema

La comunicación que representamos en Figura 3.2 corresponde a las invocaciones desde el cliente a los servicios web de la aplicación del profesor, sistema que, como se explicó

en la toma de requisitos, se ha corregido para optimizar su funcionamiento con un número elevado de alumnos conectados. No obstante, el funcionamiento exterior es idéntico al de TACAC y MoCAS, con lo que la aplicación del profesor no notará diferencia alguna.

Diagrama de casos de uso de la aplicación del profesor

En la Figura 3.3 vamos a mostrar los casos de uso que se han añadido o modificado a la aplicación del profesor para satisfacer los requisitos acordados con el usuario. Se trata de dos casos de uso ya existentes en las versiones previas de la aplicación que han sido mejorados y corregidos para ampliar su funcionalidad.

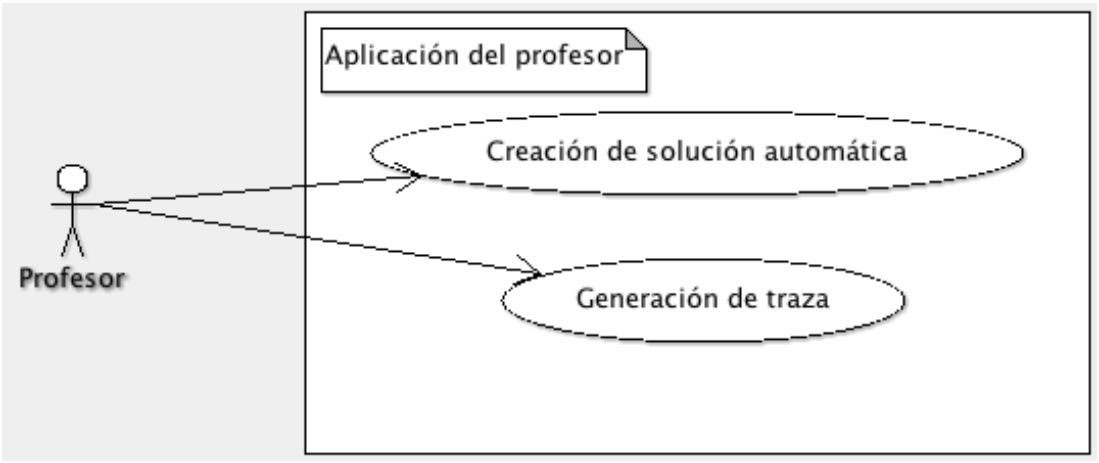

Figura 3.3 Casos de uso de la aplicación del profesor

Diagrama de casos de uso del cliente

Estos diagramas definen los casos de uso del nuevo cliente para los alumnos, mostrando como se relacionan usuario y aplicación. El único rol posible para los actores de este caso de uso es el de “Alumno”, puesto que el cliente no contempla ninguna sección exclusiva del profesor.

El primer paso para el alumno será identificarse en el sistema con su usuario y password, lo cual le mostrará la pantalla de selección de grupo, proyecto, subgrupo y entrada de datos, con combos cargados en función del usuario. Según su selección se habilitará el paso a las secciones de Depuración y de Visibility, pudiendo acceder a ambas si el proyecto de trabajo y sus selecciones lo permiten.

Una vez que el usuario entra a una de las secciones se le mostrará la interfaz de la misma, en la que encontrará herramientas comunes a ambas, como son el poder dejar comentarios en una línea de código o la ventana de chat, junto a las propias de cada sección.

En la de Depuración encontrará un cuadro de mandos en el que ver las variables en cada paso de traza, pudiendo asignar valores a las mismas y viendo los valores que les asignan otros usuarios, avanzar en el paso de traza (lo cual corrige los valores de pasos anteriores, ofreciendo los resultados) y, en definitiva, pudiendo depurar un programa de forma compartida y paso a paso.

En la sección de Visibility se mostrará el cuadro de valores asignados a la solución, junto a información acerca de si la aportación es propia o de otro usuario, cuantos han confirmado el valor y cuantos lo han desconfirmado. Pinchando en un valor el usuario podrá fácilmente participar en la resolución del ejercicio, confirmando o desconfirmando valores del resto, o eliminando sus aportaciones si lo cree oportuno, o podrá añadir nuevos valores.

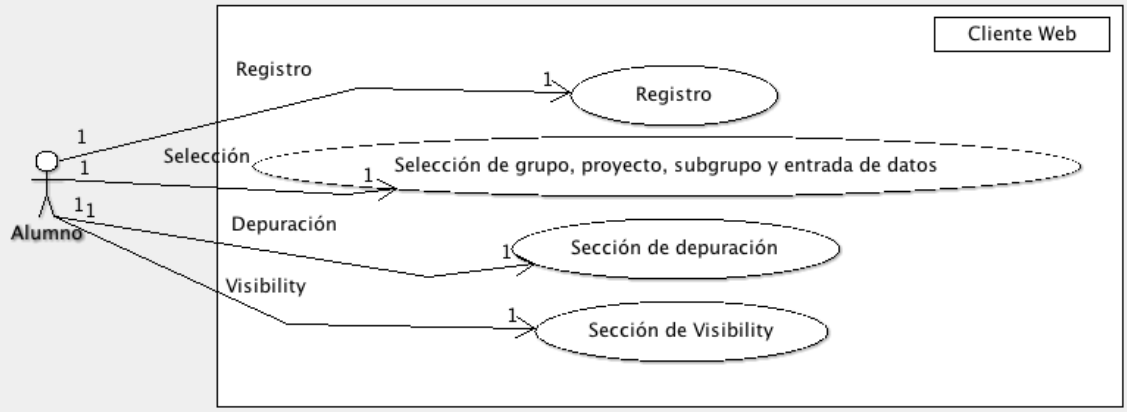

Figura 3.4 Casos de uso del cliente

3.2.4. Diagramas de estado

Un diagrama de estados es un método por el que identificar las rutas que puede seguir el flujo de información en una ejecución de la aplicación en función de los eventos que puedan suceder. Todo diagrama de estados comienza con un punto de partida, desde el que se van sucediendo los eventos que dirigirán la transición entre estados.

Un diagrama de estados consta de:

- Acciones: Estas están asociadas con transiciones, son procesos que no pueden interrumpirse y que pueden modificar el estado del sistema.
- Actividades: Las actividades se asocian con los estados, son procesos más largos que las acciones, susceptibles de ser interrumpidos por cualquier evento.
- Punto de partida: Estado inicial del sistema, desde el que se suceden los eventos y se describe el flujo de datos.

Diagramas de estado para la aplicación del profesor

Vamos a exponer los diagramas de estado de la aplicación del profesor donde se engloban las mejoras efectuadas sobre dicha parte del proyecto, que afecta a la creación de proyectos, donde se definirá el método de depuración que acepta entrada de datos y generará la solución automática.

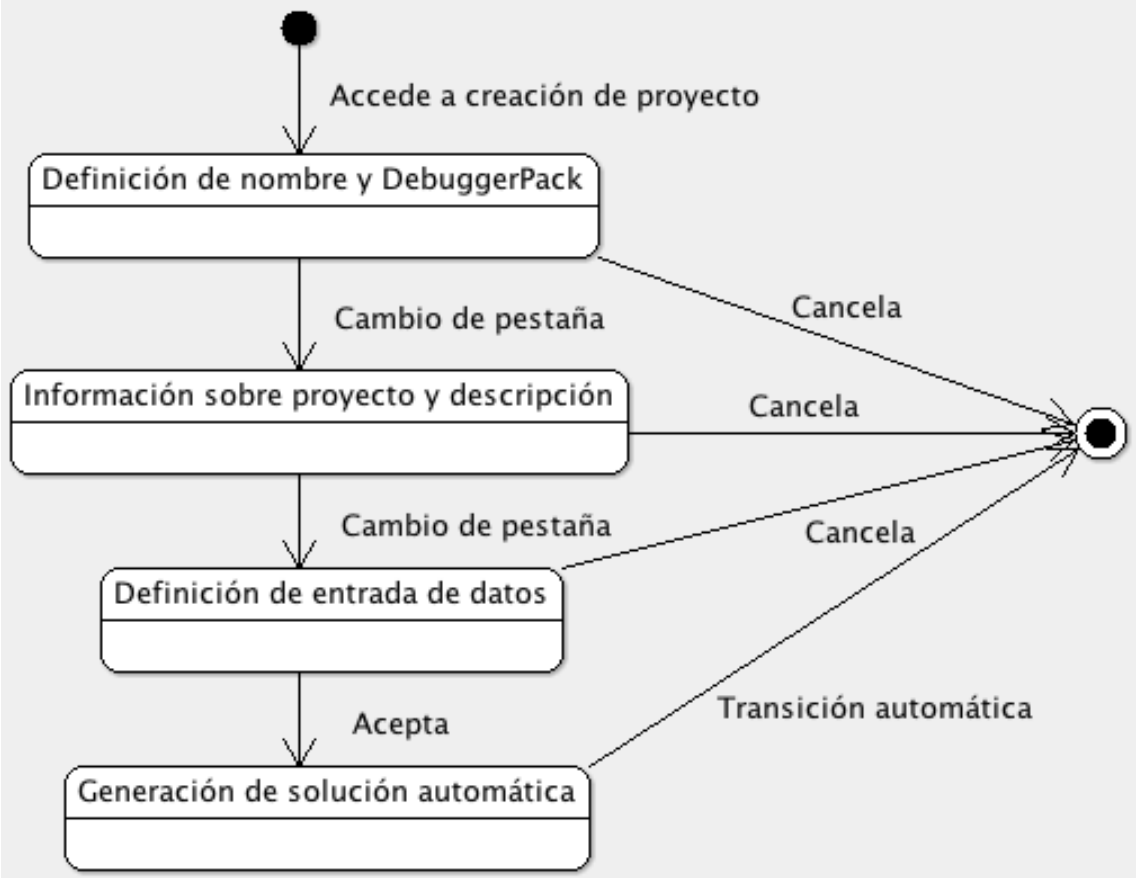

Figura 3.5 Diagrama de estado de generación de proyecto

La Figura 3.5 representa la creación de un nuevo proyecto. Como puede verse en el diagrama, el profesor no ha de realizar ninguna nueva tarea para tener acceso a las mejoras que se han añadido, son cambios completamente transparentes a él pero que consiguen subsanar los errores encontrados en la generación de soluciones automáticas de MoCAS y en la generación de trazas de depuración de TACAC.

Diagramas de estado del cliente

Expondremos en este aportado los diagramas de estado de los casos de uso definidos previamente para el cliente web.

Registro

El registro en la aplicación es el primer paso que debe realizar un alumno para poder hacer uso del sistema. El usuario deberá acceder a la URL que se le proporcione e introducir su nombre de usuario y contraseña, que será validado contra la base de datos haciendo uso de los servicios web.

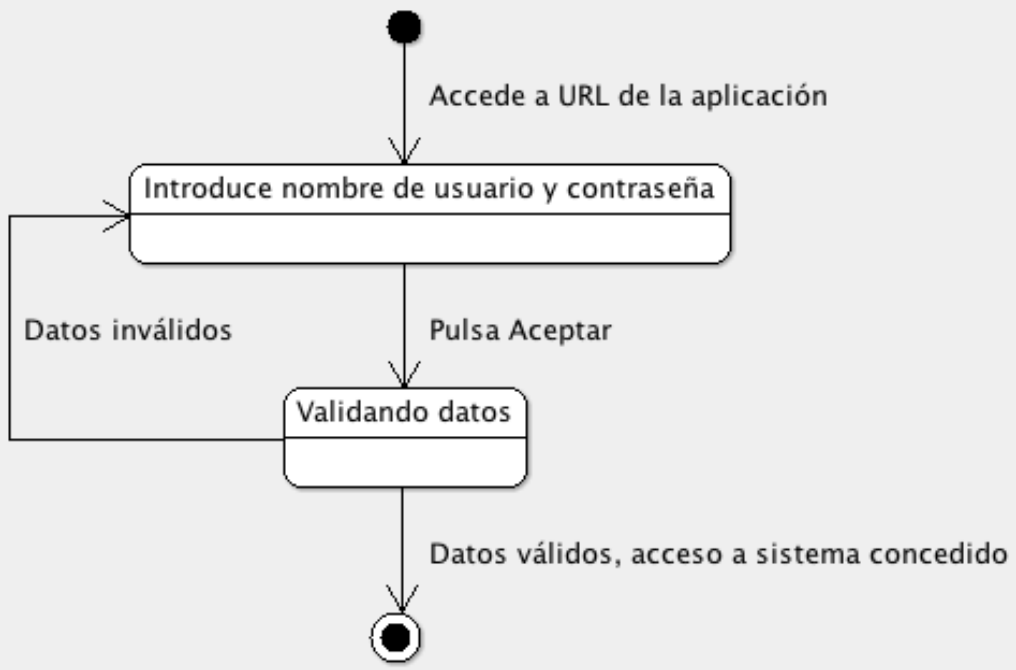

Figura 3.6 Registro de usuario

No es posible acceder a ningún servicio del sistema sin identificarse previamente, tal como se expone en la Figura 3.6.

Selección de grupo, proyecto, subgrupo, y entrada de datos

Una vez concedido el acceso al sistema el usuario debe seleccionar de entre varias listas de valores aquellos con los que va a entrar en las secciones de Depuración y de Visibility. Los valores en cuestión (grupo, proyecto, subgrupo, y entrada de datos) son definidos por el profesor para cada alumno en el momento de darle de alta, y definen en que ejercicios pueden participar y con que otros alumnos lo harán.

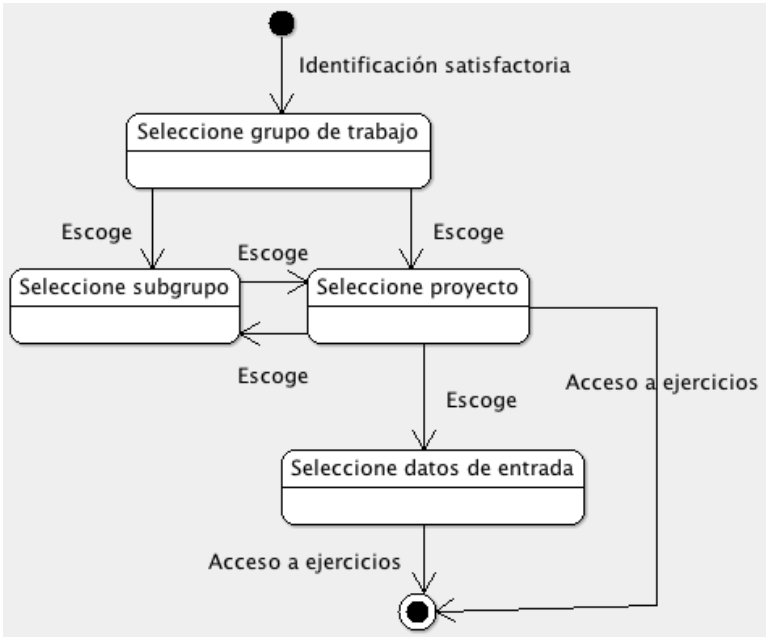

Figura 3.7 Selección de grupo, proyecto, subgrupo y entrada de datos

Desde esta pantalla de selección se accede a los ejercicios propiamente dichos, habilitando el enlace a los mismos según el usuario escoge. Al ser los valores de grupo, proyecto, subgrupo y entrada de datos dependientes unos de otros las listas que los contienen se recargarán automáticamente en función de las elecciones que vaya tomando el usuario.

Los valores a elegir son la fusión de los necesarios para definir un ejercicio de TACAC y uno de MoCAS. Creamos una pantalla de selección única y compartida que dará acceso a las funcionalidades de ambas y que permitirá que compartan información entre ellas. No sólo se han implementado las funcionalidades de los prototipos que ya existían, se ha creado un entorno compartido en el que ambos ejercicios se realizan sobre el mismo código, compartiendo datos, y profundizando en el aprendizaje de la programación.

A continuación expondremos las secciones de resolución de ejercicios, las cuales mantienen la funcionalidad ofrecida por los prototipos de partida, añadiéndoles las mejoras creadas para este PFC sobre la comunicación entre alumnos y utilizando una nueva implementación para la sincronización de datos entre usuarios.

Depuración

La sección de Depuración permite al usuario resolver de forma conjunta un ejercicio de depuración, junto a otros alumnos, en el que, además de ver paso a paso la ejecución de un programa, podrá discutir sobre los valores que van a tomar las diferentes variables del programa, asignarles un valor (que será visto por el resto de usuarios, como si estuviésemos resolviendo un ejercicio en la pizarra de una clase), y ser corregido por la aplicación para verificar su resultado.

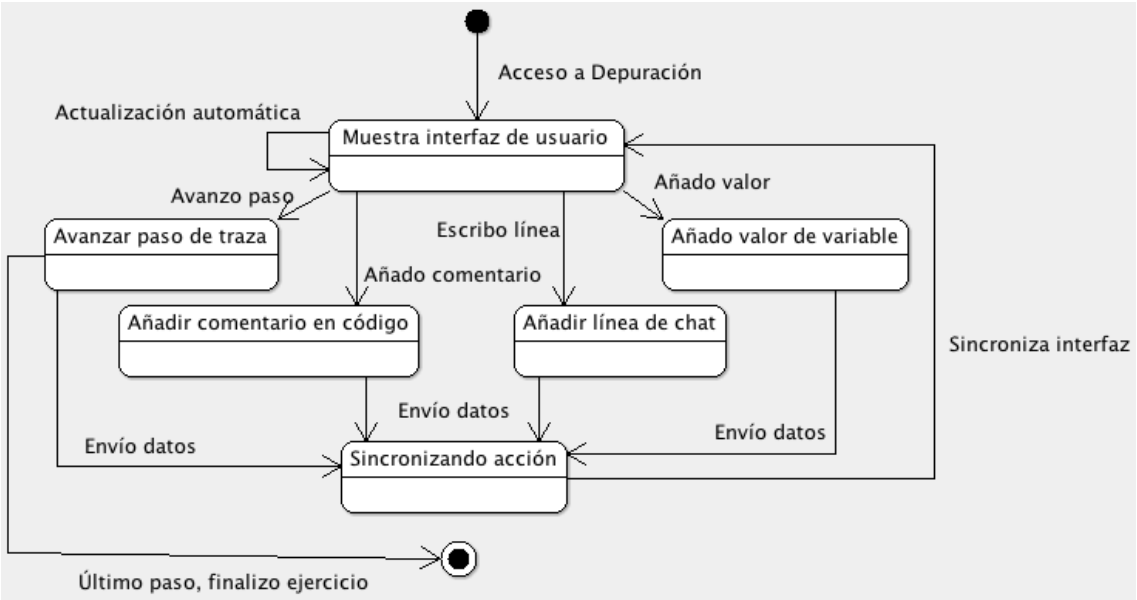

Figura 3.8 Depuración de código

Esta funcionalidad se implementó en el prototipo TACAC. En él se recorría la traza de depuración conjuntamente entre todos los usuarios, pero sólo uno de ellos tenía el control para realizar avances. También se permitía guardar valores de variables pero no eran corregidos ni radiados al resto de usuarios.

Hemos creado la corrección de variables para que se realice de forma conjunta entre todos los alumnos. Se puede comparar su funcionamiento con una pizarra en la que se escriben los resultados propuestos por todos y van siendo corregidos según avanza el ejercicio.

Otra cambio importante es que el control ya no pertenece sólo a un alumno, es compartido entre todos.

Además disponemos de los métodos de comunicación nuevos, como son el chat y los comentarios de línea.

Visibility

En esta sección de la aplicación añadiremos la funcionalidad para realizar ejercicios conjuntos de visibilidad de variables. Partiendo de un código dado, que se mostrará en la interfaz del usuario, el usuario podrá, junto al resto de alumnos conectados a dicho ejercicio, discutir acerca del ámbito de las variables de ese programa, aportando sus ideas y debatiendo las de los demás. Se ha pretendido dar un paso más en este desarrollo y no quedarse en el intercambio mudo de valores, aprovechando las funcionalidades ya creadas en la sección de Depuración, estas han sido añadidas en este sección, permitiendo una resolución del ejercicio más activa, que involucre más al alumno y que no exija cercanía física entre ellos. Al igual que en todo el desarrollo del cliente, se pretende que este desarrollo permita un trabajo conjunto independientemente de la plataforma que decida usar cada uno, o del lugar de trabajo en el que se encuentre. Tal como mostramos en la Figura 3.9, la interfaz de usuario está pensada para conseguir unir todas estas ideas de forma intuitiva y sencilla.

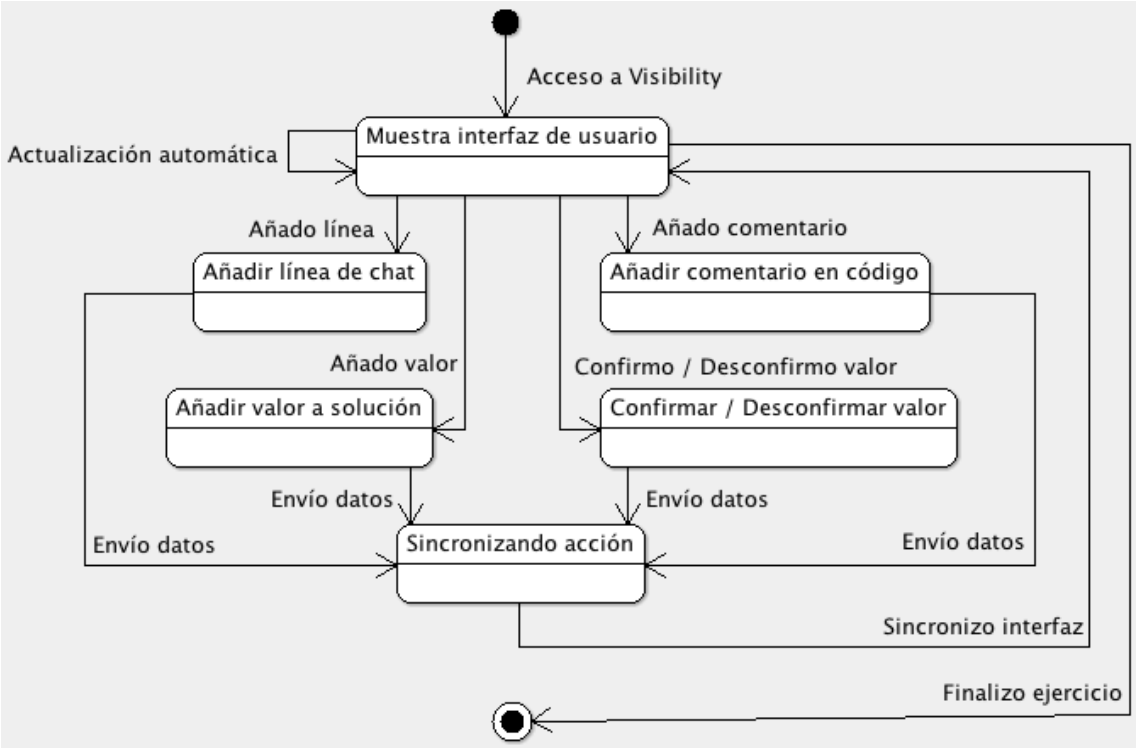

Figura 3.9 Visibility

En esta sección hemos recogido la funcionalidad creada en MoCAS, que permite proponer valores al resto de usuarios, y confirmar y desconfirmar estos valores.

La añadimos la comunicación mediante comentarios en líneas de código y el chat, y mejoramos la interfaz de usuario aprovechando la nueva implementación de aplicación web, permitiendo al alumno ver a simple vista toda la información necesaria acerca de la solución del ejercicio.

3.3. Análisis

Una vez definida la funcionalidad que debe ser implementada en el proyecto vamos a proceder a realizar una especificación más profunda de cada una de las tareas que deben ser realizadas por la aplicación, entrando, por tanto, en la fase de análisis.

Para ello, vamos a realizar una serie de diagramas que, profundizando en cada una de las tareas que previamente hemos ido delimitando, describa su funcionamiento interno y las relaciones existentes entre todas las partes del sistema.

3.3.1. Diagramas de colaboración y secuencia

Mediante los diagramas de colaboración mostraremos las interacciones existentes en el sistema al llevar a cabo las tareas de cada caso de uso. Los elementos que componen este tipo de diagramas se llaman clases, existen tres tipos de diferentes:

- Interfaz: Usadas para representar las interacciones del sistema con el usuario, comprenden todas las partes del sistema a partir de las que el actor puede iniciar una ejecución.
- Control: Esta clase representa la lógica de la aplicación, la parte de la misma que realiza las tareas encomendadas por el usuario para luego mostrarle un resultado.
- Entidad: Usadas para representar todo tipo de información, como bases de datos, a las que puede acceder la aplicación.

La representación de las clases explicadas se muestra en la Figura 3.10.

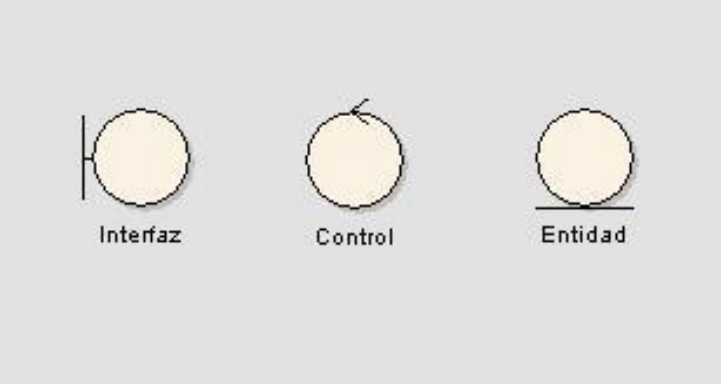

Figura 3.10 Tipos de clases en diagramas de colaboración

Al ser este un sistema con múltiples conexiones entre sus diferentes partes vamos a mostrar, además de los diagramas de colaboración, diagramas de secuencia para explicar las conexiones del mismo, un tipo de diagramas en el que vamos a definir como se extienden estas relaciones en el tiempo en toda la ejecución del mismo para ciertos casos de uso.

Las clases que vamos a utilizar en estos diagramas son las siguientes:

- Alumno, como actor principal y único del sistema
- Interfaz del sistema, como capa externa con la que se relaciona el alumno
- Gestor del cliente, capa interna del cliente web que realiza tareas invisibles al usuario.
- Gestor de la aplicación del profesor, capa interna de la aplicación del profesor con la que se comunica el cliente web.
- BBDD, la base de datos que almacena toda la información del sistema.

Mediante estas clases y las conexiones existentes entre ellas explicaremos el funcionamiento del sistema para caso de uso.

Registro en el sistema

Cuando un usuario accede al sistema se le muestra únicamente la pantalla de login, en la que debe identificarse con su usuario y contraseña. Estos datos deben ser validados contra la BBDD antes de permitir el acceso, para lo cual el cliente debe conectar con la aplicación del profesor mediante los servicios web que esta ofrece, y una vez validado, mostrar al usuario el acceso al sistema.

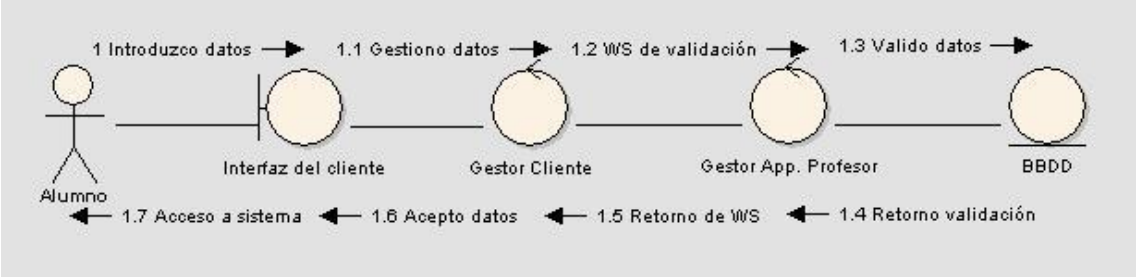

Figura 3.11 Registro de usuario en sistema

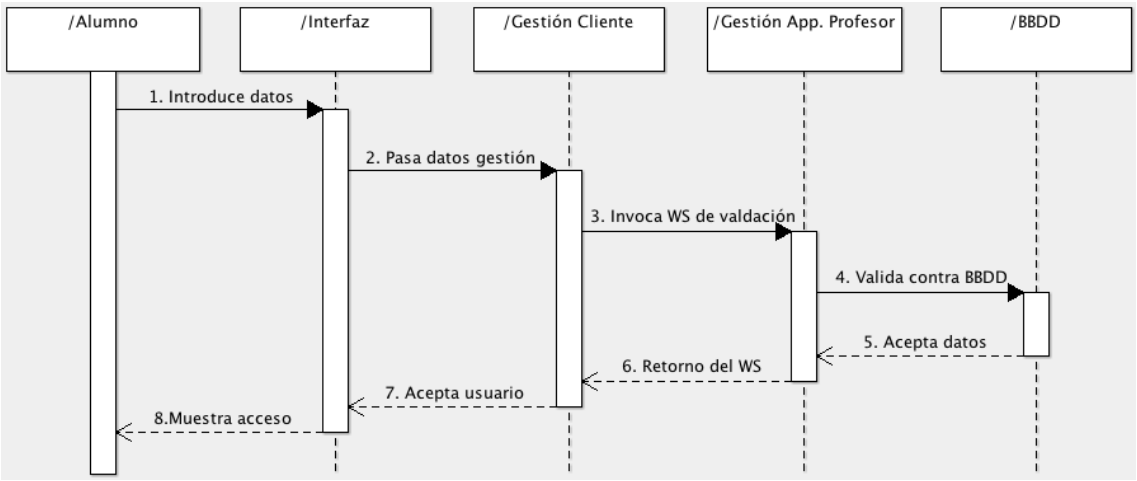

Figura 3.12 Secuencia en Registro

Selección de grupo, proyecto, subgrupo y entrada de datos

Una vez que el usuario ha sido validado en el sistema debe seleccionar los datos necesarios para participar en los ejercicios. Al ser dependientes unos de otros (proyectos dependientes de grupos, entradas de datos de proyectos…) la selección de estos valores no resulta trivial, puesto que exige al sistema recargar las listas de valores en función de las selecciones que vaya realizando el usuario. El sistema por el que se cargan estos datos y se habilita el acceso a las secciones de Depuración y Visibility se explica en Figura 3.13 y Figura 3.14.

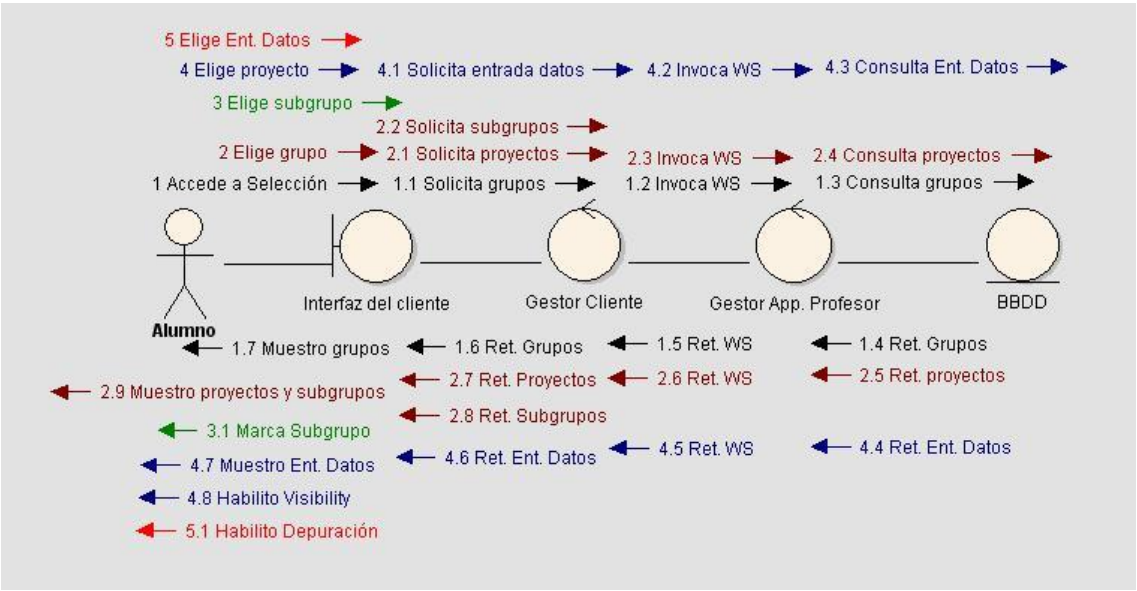

Figura 3.13 Selección de datos

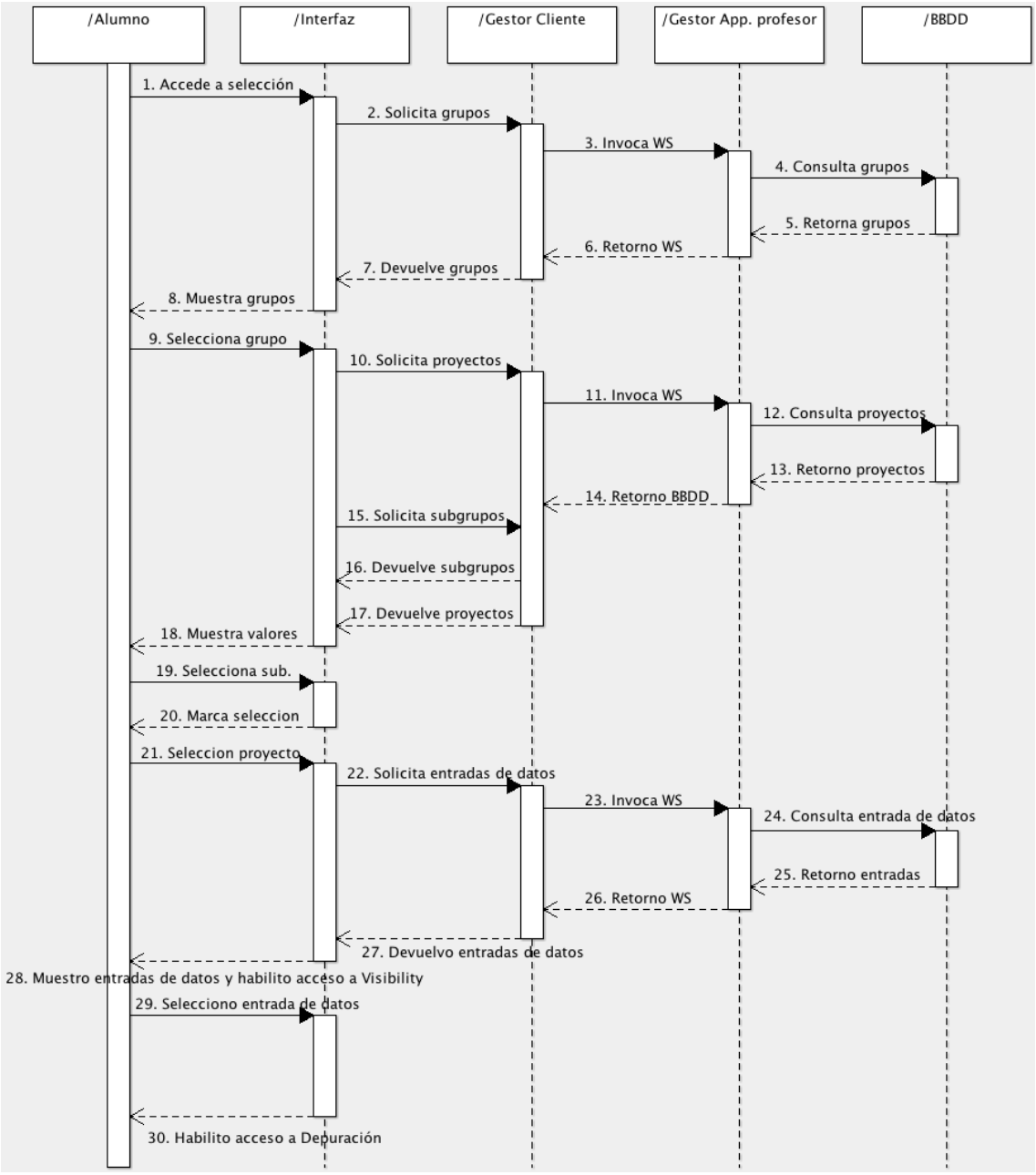

Figura 3.14 Secuencia de selección de grupo, subgrupo, proyecto y entrada de datos

Depuración y Visibility

Una vez seleccionados los datos identificadores del ejercicio, la aplicación habilita el acceso a las secciones de Depuración y Visibility, según proceda, según se explica en Figura 3.15 y Figura 3.16. Es importante destacar que en el momento en el que el usuario accede a las interfaces de usuario de cualquiera de los dos tipos de ejercicio se ejecuta periódicamente la actualización automática que, de forma invisible al usuario, refresca la interfaz para que las aportaciones de todos los usuarios se transmitan al resto. Este proceso se describe en Figura 3.17.

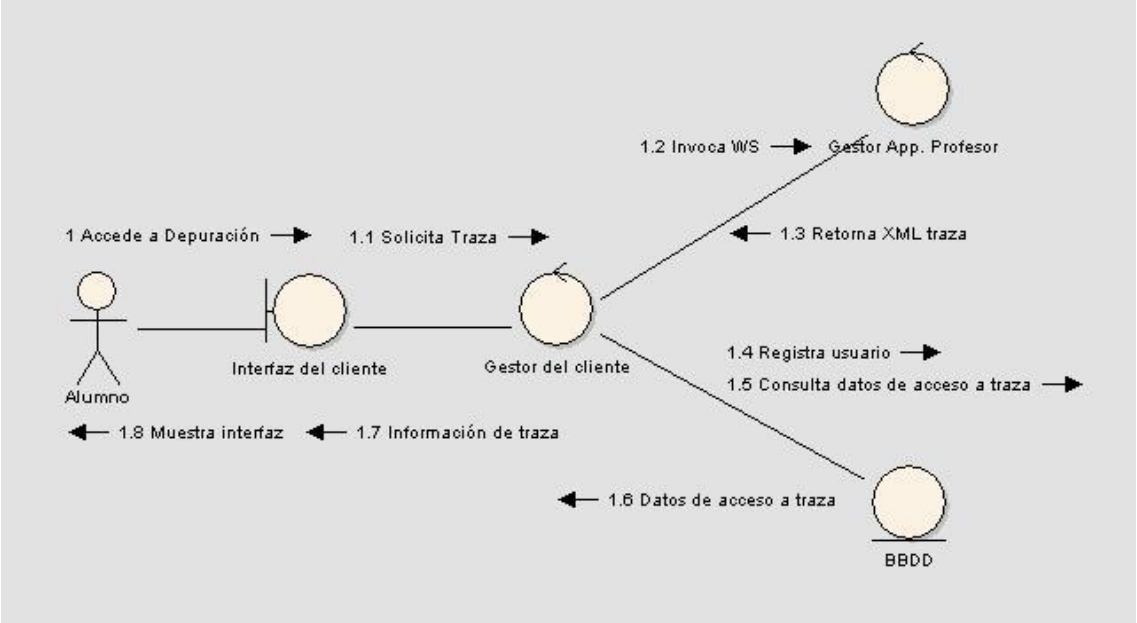

Figura 3.15 Acceso a depuración

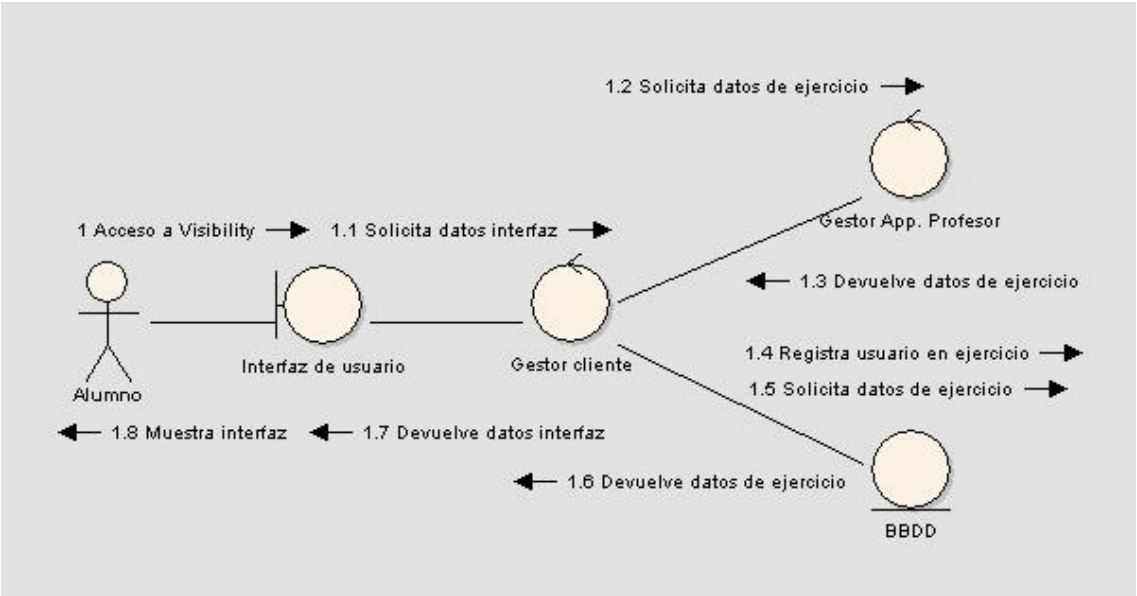

Figura 3.16 Acceso a Visibility

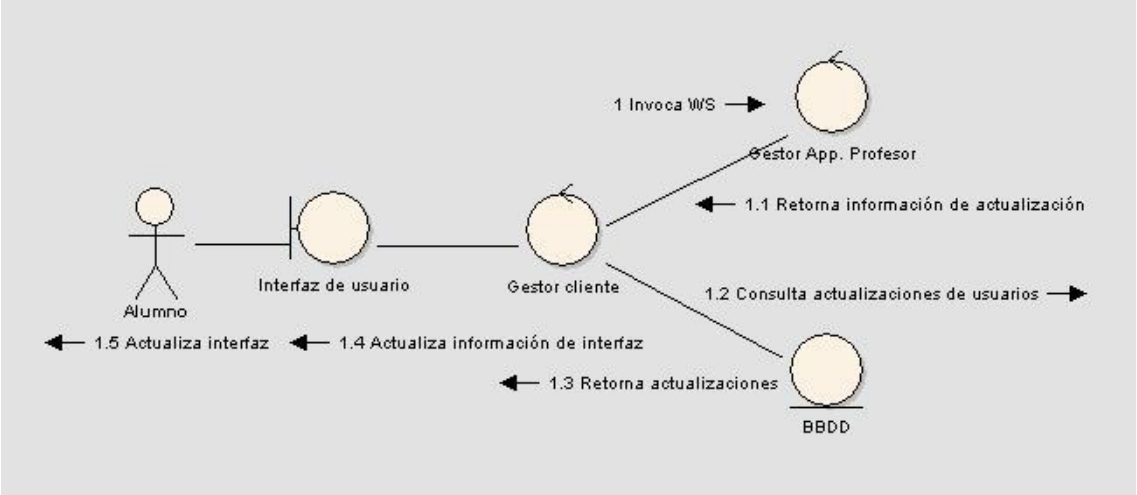

Figura 3.17 Actualización automática

En ambas vamos a poder realizar una serie de tareas, algunas propias de cada una de ellas, y otras comunes, como son la función de añadir comentarios al código y la conversación por chat. Dado que ambas secciones son bastante extensas vamos a dividir estas funciones para explicar lo mejor posible su funcionamiento y las conexiones precisas para que este se lleve a cabo.

Comenzaremos por las secciones comunes a cada una de ellas, pasando después a las propias de cada sección. Es importante destacar que no sólo son funciones que puedan llevarse a cabo en los dos tipos de ejercicios, sino que en los casos en que es posible cambiar de uno a otro estos comentarios y líneas de chat son compartidos, por lo que estarán visibles en cualquiera de las dos secciones una vez puestos. Ver Figura 3.18.

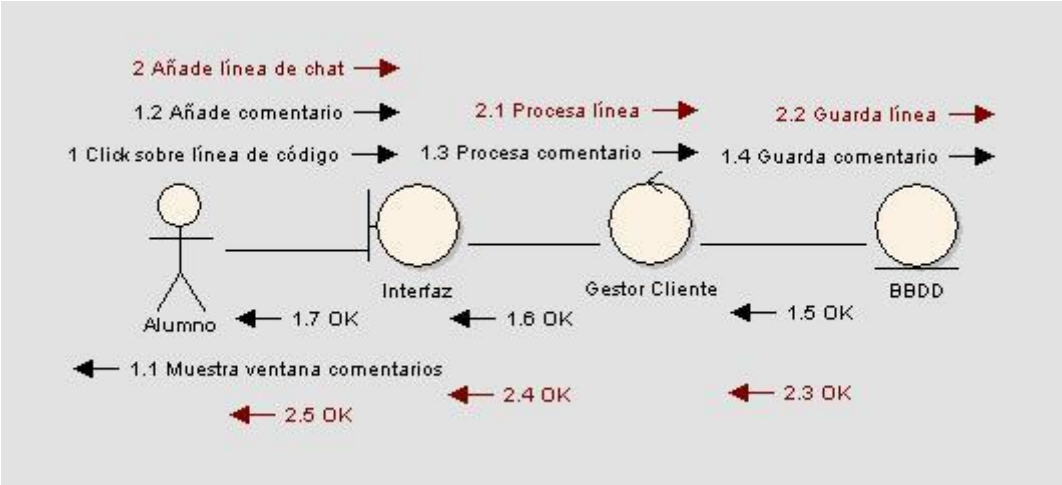

Figura 3.18 Procesamiento de chat y comentarios

Comentarios en líneas de código

La base de los dos tipos de ejercicio es el código fuente, en éste tendrá que fijarse el alumno para poder participar y resolver los problemas planteados de forma satisfactoria, por lo que creamos esta funcionalidad y se hace compartida entre las dos secciones.

Para ello, el usuario sólo debe hacer clic en una línea de código, esto abrirá un cuadro de texto en el que, aparte de ver el comentario existente, si lo hubiese, podrá añadir nuevo texto. Una vez editado y cuando el usuario guarde los cambios, el texto se almacena en BBDD.

Conversación por chat

Esta función añade la capacidad de mantener una conversación por chat entre los usuarios, con el fin de fomentar y facilitar la comunicación entre ellos. Dado que el sistema podría ser ejecutado por usuarios que no tienen por qué estar físicamente cerca, esta función resulta especialmente útil para dichos casos.

En las interfaces de usuario de los dos tipos de ejercicio existe una ventana de texto con la conversación actual, un listado de usuarios conectados, y una entrada de texto por la que el alumno puede introducir su mensaje.

Depuración

La sección de Depuración incluye, además de las funcionalidades comunes antes explicadas, otras propias a ella necesarias para resolver los ejercicios. Ver Figura 3.19.

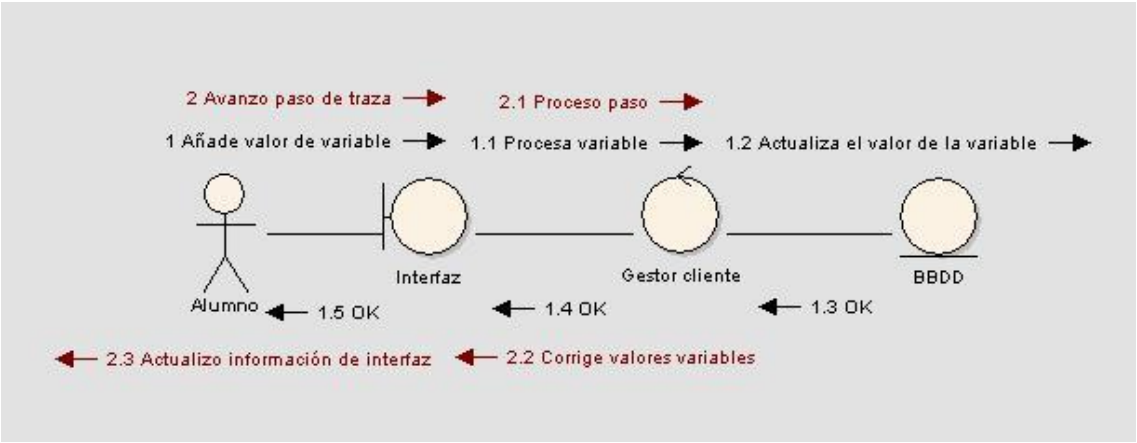

Figura 3.19 Actualización y corrección de variables

Como tales tenemos:

- La corrección de variables, por la que el usuario puede asignar valores a cada variable del programa en cada paso de ejecución de la traza y observar el valor corregido al avanzar la ejecución.
- Avance de paso de traza, por el que avanzaremos la ejecución de la depuración del programa, obteniendo los datos acerca de la línea actual en que se encuentra, y veremos los valores a las variables de pasos anteriores corregidos.

Corrección de variables

En la interfaz de usuario se mostrará una tabla de variables en la que se verán todas las variables del programa por cada paso de traza. Para los pasos de traza anteriores al actual el valor de la variable estará relleno, y corregido si los usuarios habían realizado

alguna propuesta. Los valores del paso de traza actual y posteriores dispondrán de un cuadro de texto en el que añadir un posible valor, el cual se guardará al pulsar ‘enter’ o abandonar el cuadro de texto.

Avance de paso de traza

La interfaz de usuario proporcionará un botón que permitirá a cualquier alumno avanzar la ejecución de la depuración del programa, lo cual corregirá las variables añadidas en el paso actual y mostrará la nueva línea en la que se encuentra la ejecución del mismo.

Visibility

En esta sección podremos resolver conjuntamente a otros usuarios ejercicios de visibilidad de variables, para lo que disponemos de las siguientes funcionalidades:

- Añadir valor de solución
- Confirmar / Desconfirmar / Eliminar valores de solución

La interfaz de usuario mostrará una tabla con todos los subprogramas del ejercicio y con los valores que hayan sido asignados ya a cada uno de ellos, junto a información de si es una aportación propia o del resto de alumnos y cuantas confirmaciones o desconfirmaciones tiene.

Añadir valor de solución

Mediante un combo desplegable el alumno podrá ver todas las variables del sistema, identificadas por su nombre y la línea en la que aparecen. Eligiendo una de ellas junto a un subprograma de la tabla de subprogramas de la interfaz podrá añadir ese valor a la solución. Este valor aparecerá en la tabla de todos los alumnos conectados, pudiendo cada uno de ellos confirmar o desconfirmar el valor, y el alumno que lo ha puesto pudiendo eliminarlo si lo ve procedente.

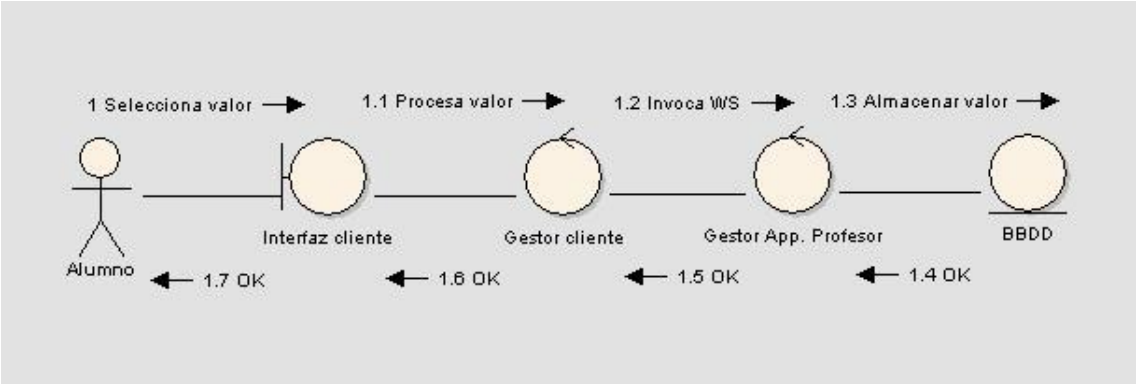

Figura 3.20 Añadir valor de solución

Confirmar / Desconfirmar / Eliminar valor de solución

Cuando el alumno pinche en cualquiera de los valores de la solución se desplegará una ventana en la que, además de observar información sobre el valor en cuestión, tal como que usuario la ha aportado y número de confirmaciones y desconfirmaciones con las

que cuenta, observará tres botones que le permitirán Confirmar o Desconfirmar el valor si no es el autor de la aportación, o Eliminar el valor si es el autor del mismo.

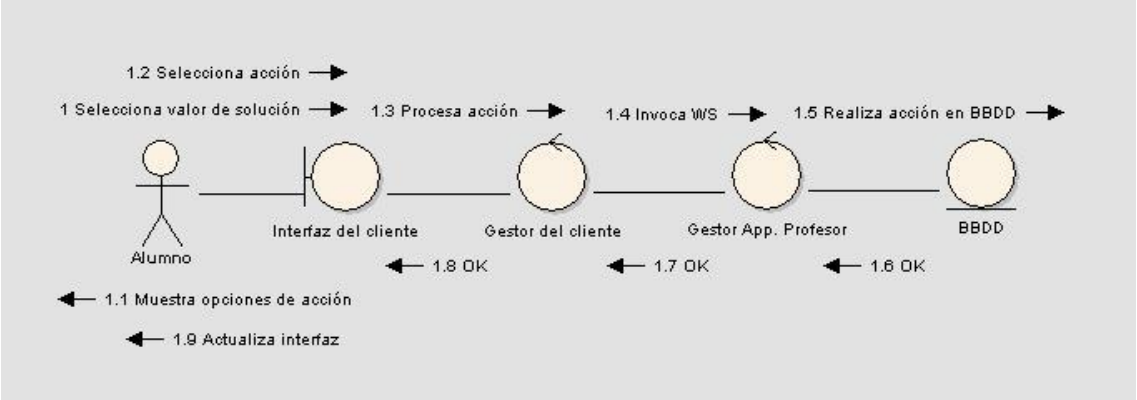

Figura 3.21 Confirmar / Desconfirmar / Eliminar valor

3.4. Diseño

Tras realizar la toma de requisitos del proyecto y el análisis del proyecto que queremos realizar vamos a pasar a la fase de Diseño, en la que utilizando todos los datos obtenidos en fases anteriores vamos a definir la implementación del proyecto.

Partimos de un sistema ya existente, que cuenta con dos aplicaciones bien diferenciadas:

- Aplicación del profesor
- Aplicaciones del alumno (TACAC y MoCAS), PDA

La aplicación del profesor incluye la implementación de los servicios web con los que se comunica con la PDA. Estos servicios web serán aprovechados por el nuevo sistema para extraer los datos que sean necesarios, y dado como están implementados esto será un proceso completamente transparente para la aplicación del profesor, que no distinguirá si el cliente que le invoca es una PDA o la nueva aplicación web.

Por tanto, mantendremos la aplicación del profesor intacta, incluyendo en ella las mejoras que se han incluido en este proyecto, pero como una fase evolutiva del mismo, no como una nueva aplicación.

La aplicación de la PDA pasará a ser sustituida por el nuevo cliente web, que incluirá las funcionalidades de las dos aplicaciones de PDA existentes, como ya se ha explicado, y añadirá nuevas capacidades que mejoren la experiencia del usuario y que combinen las dos aplicaciones de PDA en una única aplicación más potente y flexible.

Por tanto, nuestro sistema constará de las herramientas siguientes:

- Aplicación del profesor
- Cliente Web

Estas dos herramientas se comunicarán entre sí mediante servicios web, mostrando una arquitectura cliente / servidor, y manteniendo ambas comunicación directa con la base

de datos. No obstante, el cliente web en ningún caso accederá directamente al esquema de datos usado por la aplicación del profesor, siendo éste una caja negra para él, con el fin de preservar la integridad del mismo.

El cliente contará con un esquema propio, sólo accedido por él, del que se servirá de apoyo para mantener todos los nuevos datos de los que consta. Se toma esta decisión con el fin de no sobrecargar la comunicación mediante los servicios web, ya que, como se explicó en el Análisis de la situación actual, la aplicación del profesor estaba funcionando como un *cuello de botella* en el sistema anterior, algo que hemos querido evitar en esta implementación.

3.4.1. Diagramas de clases

Los diagramas de clases son un tipo de diagramas usados para representar la estructura y comportamiento de los objetos del sistema, y las relaciones existentes entre dichos elementos. No se pretende describir los aspectos dinámicos, sino los estáticos, para dejar lo más claro posible todas las relaciones existentes en la información contenida por la aplicación.

Por restricciones de espacio en la memoria mostraremos los diagramas de clase de la aplicación en ANEXO A: Diagramas de clases.

3.4.2. Diseño de la base de datos

Como se ha descrito en fases anteriores, el sistema hace uso de una base de datos para almacenar información de ejecución y de los usuarios.

Todo el diseño de la base de datos puede verse en ANEXO B: Diseño de la base de datos.

3.4.3. Arquitectura Cliente / Servidor

Por todo lo explicado anteriormente, podemos observar que la arquitectura de nuestro sistema corresponde con la de Cliente / Servidor. Este tipo de arquitectura consiste en una aplicación (cliente) que hace solicitudes a otra (servidor), la cual le da respuesta. Es un tipo de arquitectura que, aun pudiéndose ejecutar en un entorno de un solo equipo, con cliente y servidor en la misma máquina, está pensado para sistemas multiusuario funcionando con una red de computadores. Es por ello que pensamos que es la arquitectura que encaja perfectamente en nuestro prototipo.

El diseño que vamos a implementar constará de la aplicación del profesor, tal y como los prototipos anteriores TACAC y MoCAS, que funcionará como servidor para nuestros clientes web.

Aplicación del profesor

La aplicación del profesor conservará las funciones que tenía en los proyectos anteriores, esto es, servirá al profesor para gestionar alumnos, grupos y proyectos, podrá preparar ejercicios a realizar y disponerlos para la clase. No obstante, se descarga de trabajo al mismo, con el fin de evitar los problemas de conexión existentes en MoCAS cuando el número de usuarios era elevado.

El funcionamiento de TACAC tan sólo necesitará contactar con la aplicación del profesor mediante los servicios web para corroborar los permisos que tiene un usuario y los datos del ejercicio a depurar, pero toda la carga del ejercicio se llevará desde el cliente, que contactará con su esquema propio para gestionar la información que necesite.

Para el funcionamiento de MoCAS utilizaremos los servicios web para todas las operaciones que desee realizar el usuario sobre el ejercicio, pero la comunicación entre ellos también correrá por parte del cliente.

Cabe destacar las modificaciones realizadas sobre la aplicación del profesor para evitar la serialización de los servicios web, que producía los problemas de conexión ante un número de alumnos elevado. De esta forma, descargando a la aplicación del profesor de carga de trabajo y optimizando el funcionamiento de sus servicios web conseguiremos una mejora en los tiempos de las labores que debe llevar a cabo, lo cual resulta especialmente importante para el objetivo marcado de ampliar la escalabilidad de todo el sistema.

Cliente Web

El cliente web, siendo la parte visible del sistema para el alumno, dispondrá de las funcionalidades creadas en TACAC y MoCAS junto a las mejoras ya identificadas en las fases anteriores.

Para desplegar la aplicación web usaremos un contenedor de servlets (Resin, o Tomcat), que pongan a disposición de los usuarios una URL a la que conectar. Un contenedor web de este tipo puede ser fácilmente instalado en cualquier equipo, por lo que un profesor puede tener en su portátil todo el sistema, y con sólo conectarse a la red de la universidad dar servicio a sus alumnos. No obstante, y pensando en futuras ampliaciones, también es posible incluir un contenedor de servlets en un servidor más potente, como pueda ser el propio de una universidad, siendo además programas open source.

Por tanto, el cliente web conectará directamente con la base de datos para acceder a su propio esquema, y a través de los servicios web se comunicará con la aplicación del profesor para extraer la información que necesite para llevar a cabo la ejecución de los ejercicios.

El cliente deberá garantizar el acceso al sistema únicamente a usuarios identificados, para lo cual la primera función que cumplirá será validar el nombre de usuario y la contraseña contra la aplicación del profesor, informando al usuario si la validación no ha sido satisfactoria, o dándole acceso si lo ha sido.

Figura 3.22 Registro de usuario

La siguiente función que deberá realizar, una vez validado un usuario en el sistema es permitirle escoger a que ejercicio quiere conectarse, esto se realizará de nuevo conectándose a la aplicación del profesor, que será la que informa de los grupos y proyectos disponibles para el alumno en cuestión.

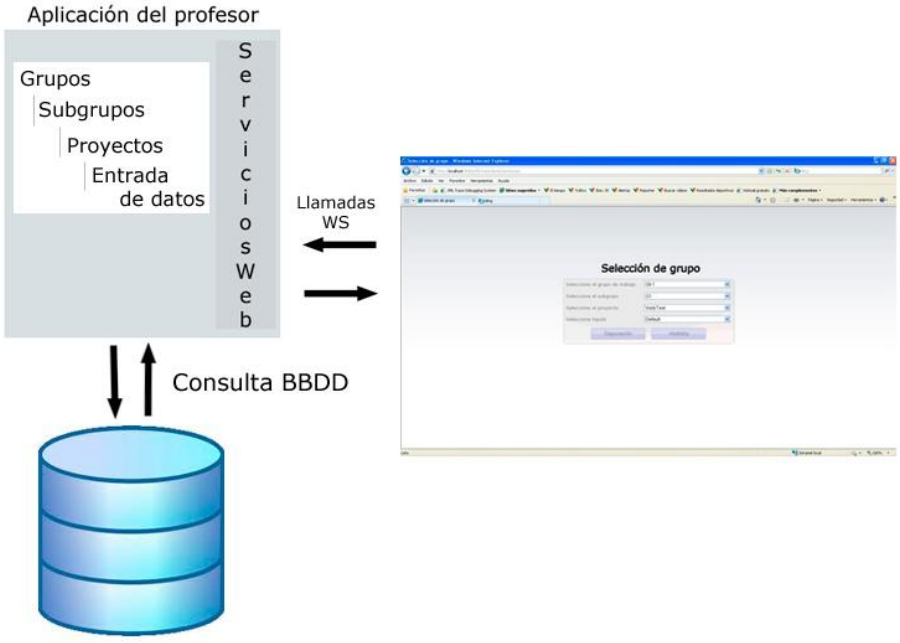

Figura 3.23 Selección de datos de ejercicio

Una vez haya escogido se pasa a las secciones de los ejercicios, de depuración y de ámbito de variables, como hemos dicho. Es en esta sección donde las comunicaciones del cliente toman mayor importancia, ya que para que los alumnos puedan trabajar en tiempo real, viendo en todo momento las aportaciones realizadas por sus compañeros y compartiendo las suyas propias, hemos creado un sistema, utilizando la tecnología Ajax, que es capaz de realizar estas actualizaciones automáticas resultando en un proceso invisible para el usuario.

Como se explica en la Figura 3.24, la capa de Ajax realiza invocaciones de forma automática y periódica a determinados servlets (paso 1); estos servlets solicitan la información a la base de datos (pasos 2' y 3') o a los servicios web (pasos 2 y 3), según proceda y preparan el envoltorio XML de respuesta. Dicha respuesta se devuelve a la capa de Ajax (paso 4), la cual actualiza la interfaz de usuario con esos datos (paso 5).

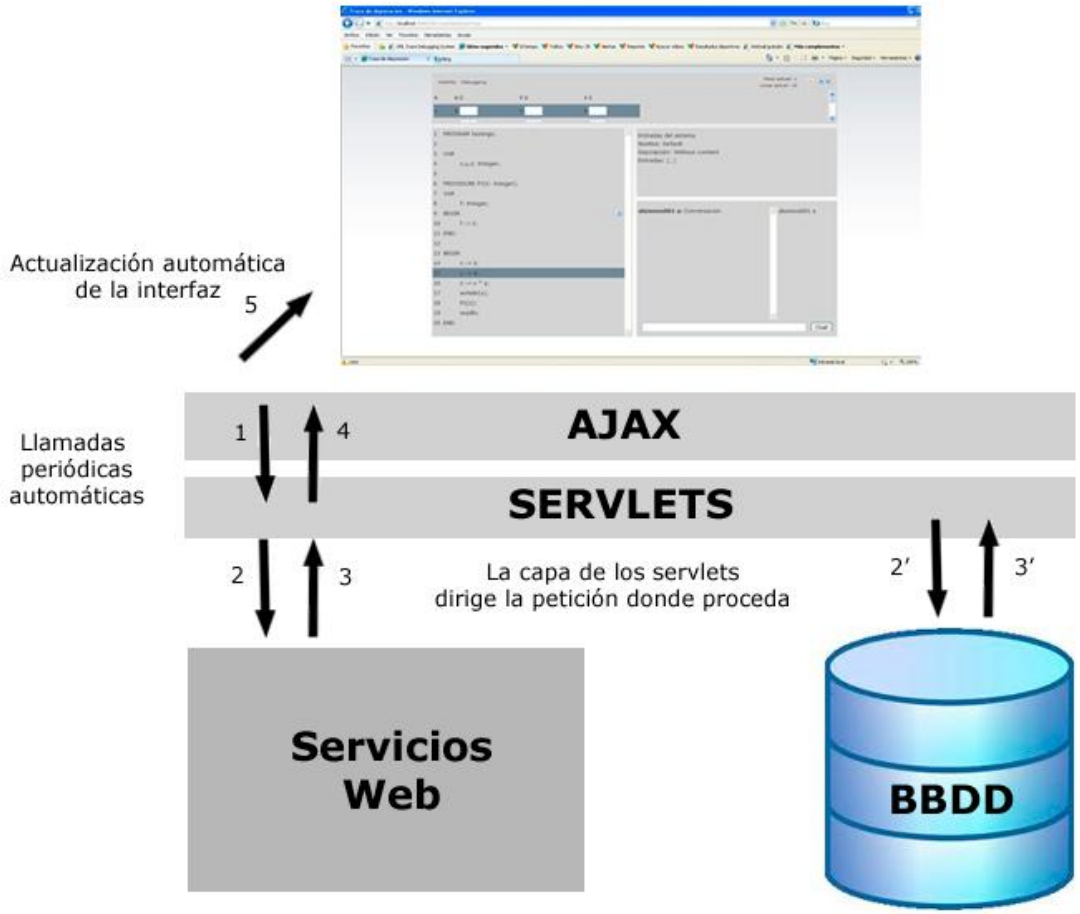

Figura 3.24 Actualizaciones automáticas

Las actualizaciones de un usuario se radian al resto de una forma similar.

Cuando el usuario realiza una acción que debe ser transmitida se ejecuta una invocación Ajax, como se puede observar en la Figura 3.25.

Al realizar la aportación se ejecuta una función Ajax (paso 1), esta función enviará los datos que deban ser guardados a un servlet (paso 2), el cual los almacenará invocando directamente a la base de datos (paso 3') o a los servicios web (paso 3).

Es importante remarcar que durante todo el proceso de guardado de la aportación el usuario puede seguir haciendo uso de la aplicación, puesto que ese proceso es completamente transparente para él.

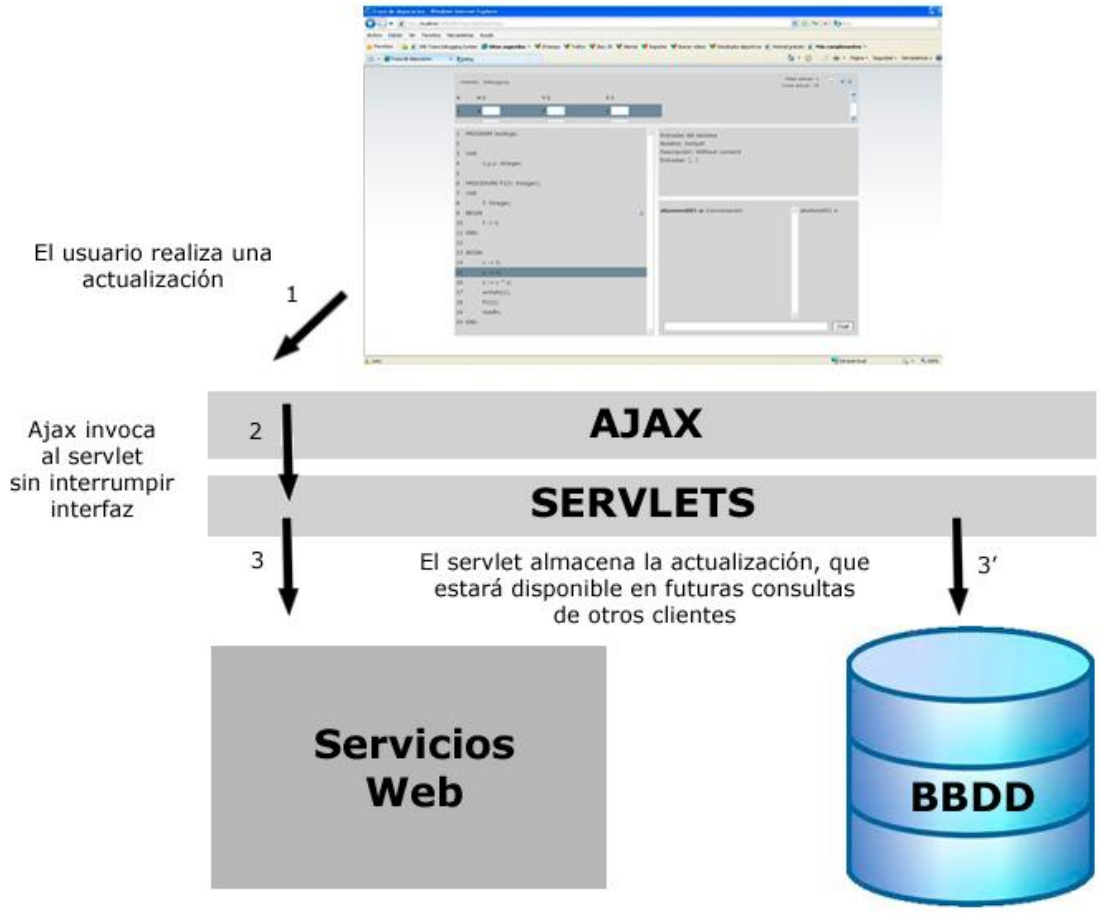

Figura 3.25 Guarda aportación de usuario

De esta forma, liberamos al servidor de responsabilidad a la hora de informar a los clientes, ya que serán estos los encargados de preguntar por los cambios que deben recoger. También de esta forma dividimos los esfuerzos necesarios para realizar las actualizaciones, ya que cada punto del sistema realiza una parte.

3.5. Pruebas del sistema

La fase de pruebas es una parte esencial de cualquier modelo de ciclo de desarrollo de software. En ella se debe comprobar que la funcionalidad que se ha descrito en el análisis y diseño de la aplicación se ajusta a la implementación creada.

Para la realización de la fase de pruebas distinguiremos dos tipos de errores:

- Errores de programación, estos son aquellos causados por un fallo en la semántica del código escrito por el desarrollador, que puede llevar a un defecto de forma en determinadas circunstancias, pero cuya causa ha sido debida a una programación incorrecta y no a fallos en el diseño.
- Defectos de forma, son aquellos en los que el sistema no realiza las acciones que el usuario espera en un momento determinado. Como ya hemos dicho, pueden ser causadas por errores de programación, pero la causa puede subyacer en las fases de análisis y diseño, y han de ser cuidadosamente estudiados.

Existen diversos tipos de pruebas en el desarrollo de software. Para nuestro sistema, dadas sus características, vamos a realizar:

- Pruebas unitarias: Método de pruebas por el que se valida por separado cada módulo de software del sistema para comprobar que su funcionamiento es correcto.
- Pruebas de integración: Estas pruebas parten de unas pruebas unitarias con resultado satisfactorio, y tratan de conjugar todos los módulos del sistema para validar el funcionamiento correcto del mismo.

Las pruebas unitarias se han llevado a cabo partiendo del análisis del sistema y los casos de uso descritos en el mismo. De tal forma, se han validado los módulos siguientes:

- Registro de usuario en el sistema
  La identificación resulta correcta, de igual manera que al introducir un usuario que no existe, o con contraseña errónea, el resultado es el esperado. No obstante, en este punto se decide diferenciar los casos en que la identificación no es correcta con aquellos en los que ha fallado la comunicación con base de datos, añadiendo este nuevo tratamiento de errores.

- Selección de grupo, subgrupo, proyecto y entrada de datos
  Comprobamos que la carga de combos concuerde con los datos existentes en la aplicación del profesor y que el cambiar los valores de estos combos enlazados no tiene resultados incoherentes o imposibles (selecciones de grupos con proyectos que nos les pertenecen, por ejemplo).

- Módulo de comentarios en código
  Introducimos comentarios en una línea y comprobamos que se actualiza en la interfaz de usuario el icono que lo indica. En esta fase se decide añadir el globo de texto al pasar el ratón sobre el icono para mostrar el comentario.

- Módulo de chat
  Probamos la introducción de línea y su actualización en la interfaz del usuario.

- Módulo de variables en ejercicio de depuración
  En las primeras pruebas del módulo de variables se decide añadir la opción de maximizar el área de variables, dado que en trazas con muchas variables resulta incómodo un espacio pequeño y es precisa una visión global. Se comprueba que la aplicación almacena correctamente las variables.

- Paso de traza en ejercicio de depuración
  Comprobamos que la secuencia de pasos es correcta, y que en cada avance de paso de traza se marcan como corregidas las variables rellenas.

- Añadir valor a solución
  Se añade un valor a una solución seleccionando subprograma y variable, observamos que el resultado en pantalla es correcto.

- Corregir valor de solución
  Seleccionando un valor de la solución aportado por un usuario diferente al nuestro comprobamos que nos permite Confirmar y Desconfirmar variable, pero no Eliminar. Probamos ambas opciones y vemos el resultado en la interfaz de usuario.

- Eliminar valor de solución
  Tras aportar un valor a la solución probamos a eliminarlo, comprobamos que desparece de la interfaz de usuario.

- Generación de proyectos compartidos
  Con el fin de generar proyectos conjuntos para los ejercicios de depuración y visibility nos encontramos con un error heredado de los proyectos anteriores, ya que la generación de la solución automática falla con código fuente que tenga algo más que el "esqueleto". Planteamos esto como un mantenimiento correctivo y lo resolvemos para no disminuir la funcionalidad planteada.

- Generación de proyectos de depuración con entradas de datos
  Una de las limitaciones que encontramos en TACAC era la imposibilidad de depurar programas con entrada de datos externa al programa, esto se ha corregido y comprobado.

Una vez completadas las pruebas unitarias nos disponemos a probar el sistema en su conjunto con las pruebas de integración. Para ello, planteamos una serie de proyectos a los que acceder desde tres dispositivos diferentes y resolviendo los ejercicios de forma conjunta.

En esta fase de pruebas nos centraremos en la actualización automática de la interfaz de usuario y en las comunicaciones entre ellos, siendo estas partes críticas del proyecto:

- Propagación de comentarios de código y líneas de chat.
- Paso de traza conjunto, comprobamos que no se solapen peticiones.
- Modificaciones de variables, vemos que las actualizaciones se propagan.
- Añadir valores a la solución con un usuario, siendo visibles al instante por los otros.
- Correcciones de valores, modificándose en la interfaz de cada usuario los valores de Confirmados y Desconfirmados.
- Intercambio de información entre usuarios que, estando trabajando en un proyecto que contiene ejercicio de depuración y de visibility, se encuentran en secciones distintas.

Tras realizar las pruebas expuestas concluimos que el funcionamiento del sistema se adapta a las funcionalidades recogidas en este proyecto, aun no pudiendo realizar una puesta en producción al no disponer de los medios técnicos necesarios.

4. CONCLUSIONES

4.1. Conclusiones

Este PFC comenzó analizando los prototipos anteriores TACAC y MoCAS. Dichas aplicaciones compartían un punto común, la aplicación del profesor, y ofrecían al alumno dos clientes diferentes, ambos utilizando una PDA, con los que resolver dos tipos de ejercicio distintos. TACAC permite realizar un ejercicio de depuración de código y MoCAS uno acerca de la vigencia y ámbito de las variables de un programa, todos ellos de forma colaborativa.

Creemos que la base propuesta por estos dos prototipos es buena, útil para el aprendizaje colaborativo, pero mejorable en ciertos aspectos. Es por ello que los primeros pasos que nos planteamos son aplicar ingeniería inversa para entender por completo el funcionamiento del sistema y analizar la usabilidad de los prototipos para identificar las áreas a mejorar.

En este estudio encontramos una fuerte dependencia de la plataforma .NET que impone una limitación a los tipos de dispositivos que pueden conectarse al sistema, junto con errores de comunicación entre cliente y servidor que hacen que el uso de las aplicaciones en un entorno real sea muy complicado.

A la vista de estos resultados decidimos mejorar un nuevo cliente multiplataforma que englobe las funcionalidades ya existentes y también aporte nuevas funcionalidades, teniendo como prioridad la escalabilidad y flexibilidad del sistema.

Hemos eliminado la limitación que suponía depender de la plataforma Windows Mobile con el fin de abrir la herramienta a cualquier usuario con acceso a la red con su propio equipo, sin depender de un dispositivo concreto que podría no estar en disposición de adquirir o que podría no querer hacerlo, al ser algo de un uso tan exclusivo. Creemos que esto era una prioridad para seguir avanzando en la idea en la que se basa este proyecto y los prototipos anteriores.

La nueva aplicación web incluye todas las funcionalidades anteriores, ampliando algunas de ellas, dado que se han eliminado las limitaciones que marcaban tanto el entorno de trabajo de .NET como los dispositivos donde se debía ejecutar. Además, se ha potenciado la comunicación entre los usuarios para que el trabajo en grupo resulte más real y más satisfactorio, algo realmente importante para conseguir la implicación de los alumnos.

Se han combinado las funcionalidades de TACAC y MoCAS para que puedan ser utilizadas de forma simultánea, compartiendo chat y comentarios en el código entre las dos.

Hemos creado, por tanto, un entorno de trabajo para el alumno fácil de usar, que integra las herramientas que tenemos en una única aplicación, y que permite realizar un trabajo en grupo sobre los ejercicios asignados, requisitos marcados por Michael Köllingen su tesis como necesarios para entornos orientados a alumnos de primer curso (Kölling, 1999).

Las mejoras que hemos aportado también permiten realizar un estudio sobre el uso de la aplicación, puesto que cada acción realizada queda marcada en la base de datos, con el fin de que el profesor pueda analizar cómo se ha realizado el ejercicio, como han participado los alumnos, que puntos deben mejorarse o donde incidir en el futuro.

Eliminamos los problemas existentes en la comunicación entre la aplicación del profesor y cualquier cliente que se conectase a ella, destacando la serialización de los servicios web con los que se comunica. Corregir esta limitación responde a la necesidad de potenciar la escalabilidad de la aplicación, no sólo por poder dar servicio a un mayor número de alumnos, sino porque aunque sea un número reducido el rendimiento ofrecido sea óptimo.

Finalmente, las pruebas realizadas han sido satisfactorias, indicándonos que se han cumplido los objetivos marcados. Se ha creado una base sobre la que potenciar el trabajo lectivo a través de internet y la colaboración entre alumnos y profesores que por razones de tiempo o distancia física no pueden realizar estos trabajos de forma común en una clase. Creemos poder concluir que se ha conseguido dar el primer paso en esta dirección.

4.2. Trabajo futuro

Los siguientes pasos que creemos deben darse son seguir potenciando la escalabilidad del sistema, ampliar el cliente para que tenga un funcionamiento autónomo que permita conectar a varias “aplicaciones de profesor”, y no solo a una, permitiendo a la aplicación ser un punto de encuentre entre alumnos y profesores. Dada la arquitectura del sistema actual creemos que podría adaptarse fácilmente a la comunicación con diversos servidores, permitiendo un registro de profesores con sus respectivas aplicaciones, que ofrezcan ejercicios a los alumnos de sus grupos.

Es decir, el camino recorrido hasta ahora ha dispuesto a la aplicación del profesor como centro del sistema, y a los clientes como sus satélites. Podría moverse este centro hacia la aplicación web, permitiendo a esta gestionar las comunicaciones entre alumnos y profesores, y no limitándose a ofrecer comunicación con un único profesor.

De esta forma, el cliente estaría siempre disponible en una URL de internet, dispondría de una pestaña de administración donde los profesores identificasen su equipo, al que se podría conectar el cliente, y poner a disposición de los alumnos sus ejercicios.

El cliente pasaría a ser la parte central del sistema, siendo una herramienta ya no sólo para los alumnos, sino también para el profesor, o profesores, que podrían utilizarla a conveniencia.

5. BIBLIOGRAFÍA

Boroni, C. G. (2001). Engaging Students with Active Learning Resources: Hypertextbooks for the Web. *Procs. of the 32nd SIGCSE Technical Symposium on Computer Science Education.*

Dupont, A. (2008). *Practical Prototype and script.aculo.us.* Apress.

Kölling, M. (1999). The design of an object-oriented environment and language for teaching. *Dissertation of DPh. Department of Computer Science, University of Sydney.* Sydney.

w3schools. (última visita junio 2011). Obtenido de w3schools: http://www.w3schools.com/

Wikipedia. (última visita junio 2011). Obtenido de http://www.wikipedia.com

WinPcap. (última visita junio 2011). Obtenido de WinPcap: http://www.winpcap.org/

6. ANEXO A: Diagramas de clases

A continuación vamos a mostrar los diagramas más relevantes de nuestro proyecto.

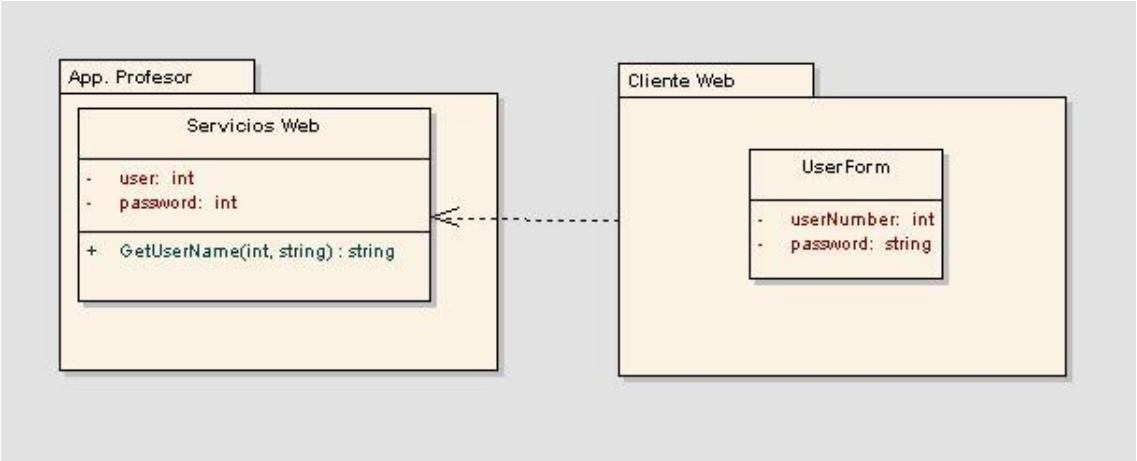

Figura 6.1 Registro en sistema

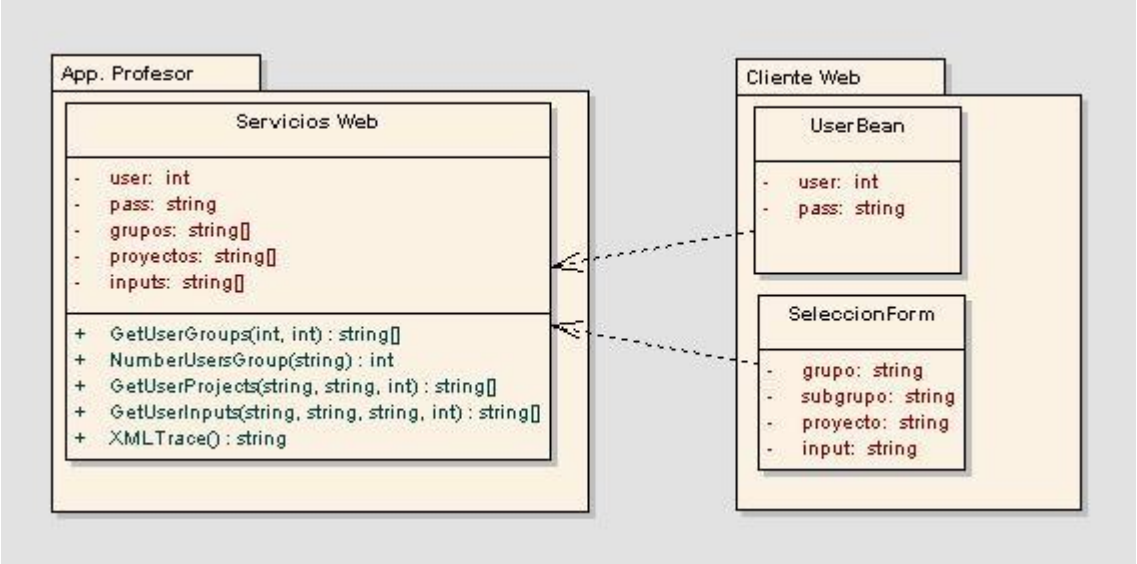

Figura 6.2 Selección de datos de ejercicio

Anexo A: Diagramas de clases

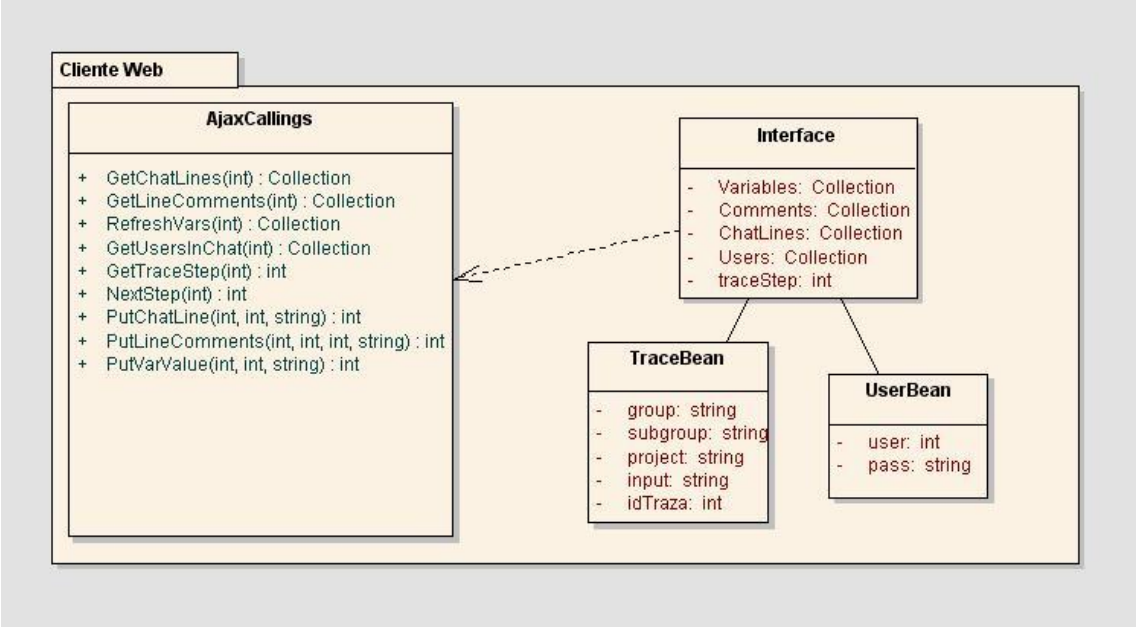

Figura 6.3 Depuración de código

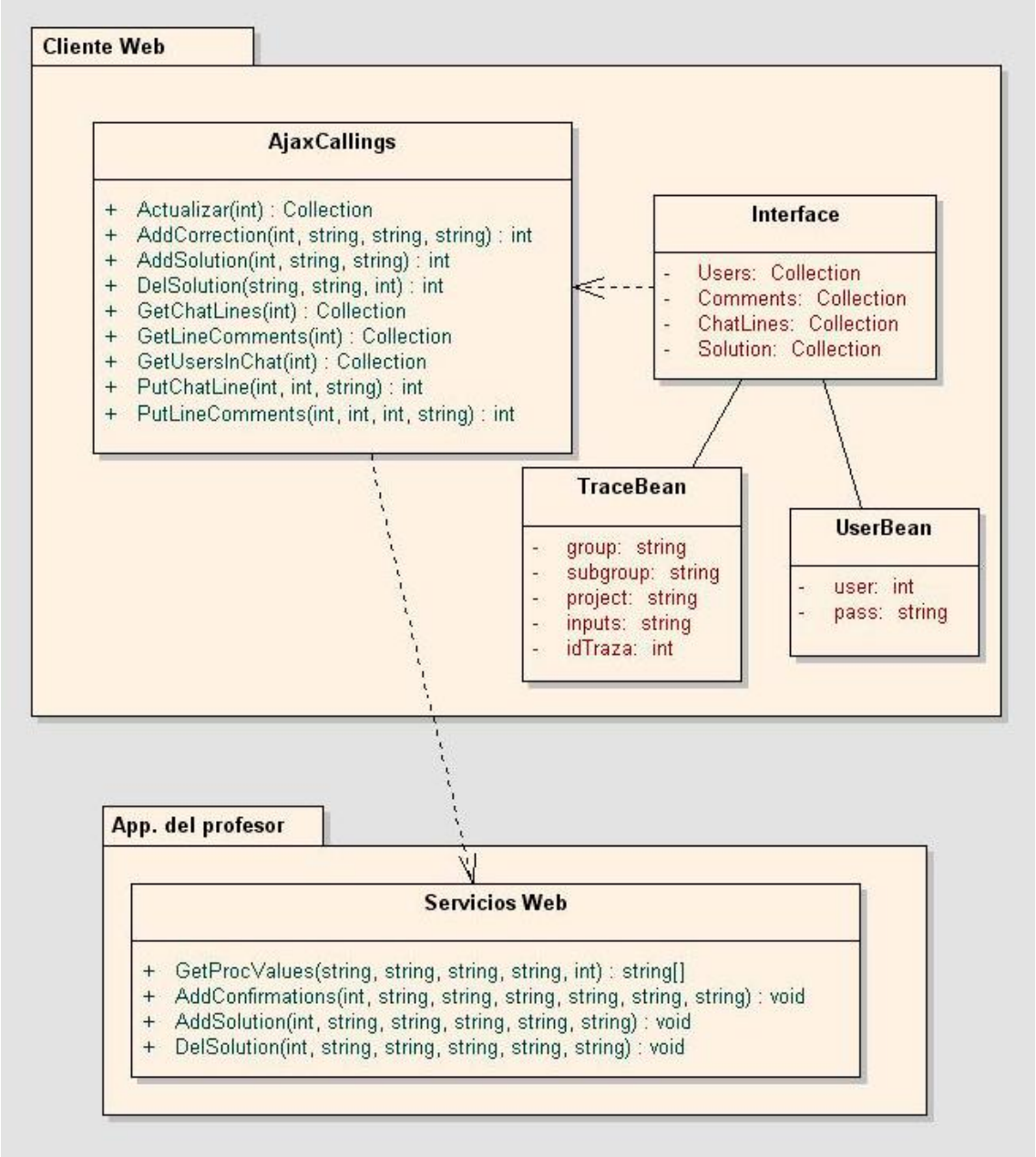

Figura 6.4 Ejercicio de Visibility

7. ANEXO B: Diseño de la base de datos.

Las aplicaciones de las que partimos, TACAC y MoCAS, compartían un esquema de base de datos en el que se mezclaban y funcionaban juntos los procesos de ambas. Este sistema era apropiado para las implementaciones que se llevaron a cabo de ambos proyectos. Primero TACAC creó el esquema que contendría toda la gestión de usuario realizada por la aplicación del profesor y la de ejecución de la PDA, y posteriormente MoCAS añadiría nuevas tablas al esquema para el desarrollo de sus funcionalidad, funcionando de forma conjunta e indisoluble con TACAC.

Por el contrario, nuestra aplicación va a desvincularse del esquema existente en la medida de lo posible. Deberemos acceder a los datos contenidos en él, pero sólo lo haremos mediante los servicios web puestos a disposición por la aplicación del profesor, por lo que el esquema existente será una caja negra para nuestra aplicación, fomentando de esta forma la encapsulación del sistema y favoreciendo que futuros cambios tanto en un lado como en otro puedan ser realizados de forma independiente.

Además de acceder al esquema existente mediante los servicios web, nuestra aplicación contará con un esquema propio, solamente accesible por el nuevo cliente, en el que se almacenará la información de ejecución necesaria, junto a las relaciones existentes entre alumnos y ejercicios realizados.

El nuevo esquema consta de las tablas siguientes:

- **jc_trazas**: La presente tabla es el centro del sistema, en ella se crea una nueva entrada cada vez que un usuario accede a alguno de los ejercicios. Definida por la relación grupo-proyecto-subgrupo-entrada, que define cada ejercicio, como hemos explicado anteriormente, añade una columna más, id_traza, que identificará de forma única a cada entrada de la tabla y a cada ejercicio, siendo este identificador el que nos va a permitir conectar todos los datos de una ejecución. También tendremos un campo fecha con el que validaremos que la entrada en cuestión sea un ejercicio aún activo en el tiempo o no, pudiendo de esta forma tener un registro de un mismo ejercicio realizado en días diferentes.
- **jc_comments**: En esta tabla almacenaremos los comentarios añadidos al código por los alumnos, junto con el autor del comentario. Relacionada con la tabla central por el atributo id_traza.
- **jc_chat**: Tabla con la información referente al chat entre usuarios, de nuevo relacionada con la tabla central por id_traza.
- **jc_usersprojects**: Esta tabla guarda información de todos los usuarios conectados a un ejercicio, ya sean ejercicios en ejecución o antiguos, con el fin de llevar un histórico. Además de la relación con la tabla centra mediante id_traza cuenta también con un campo fecha que nos permitirá saber si el usuario está activo en el sistema, o lo ha abandonado.
- **jc_variables**: En esta tabla guardamos la información de las variables referentes a los ejercicios de depuración, teniendo toda la información necesaria para la corrección de un ejercicio. Nos permitirá saber los valores aportados por los usuarios, la corrección de los mismos, y, por supuesto, que usuario es el que ha realizado la aportación.

Como puede observarse, el nuevo esquema de base datos, además de utilizarse para la gestión de acceso de los usuarios a los ejercicios, sirve para ampliar las funcionalidades existentes en la aplicación TACAC. Mientras que para incluir las funcionalidades de MoCAS nos hemos servido de los servicios web que la misma nos proveía, para TACAC hemos realizado una implementación nueva, respetando y compartiendo su idea inicial, pero ampliándola y mejorándola, razón por la que se han necesitado estas nuevas tablas.

A continuación mostramos en la Figura 7.1 el esquema de base de datos usado por nuestro cliente web:

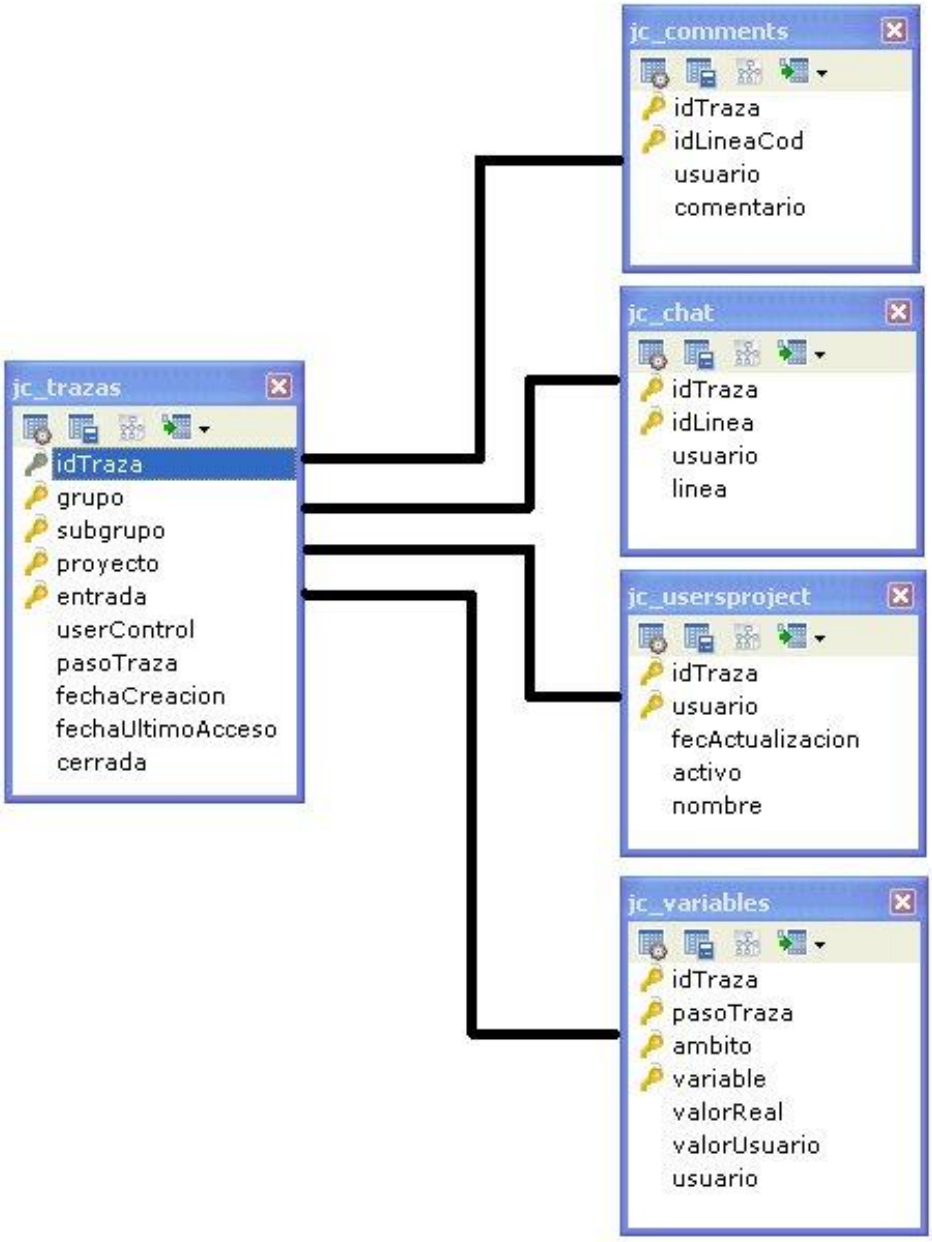

Figura 7.1 Esquema de base de datos

8. ANEXO C: Instalación del sistema

Para poder ofrecer servicio a través del cliente debemos realizar los siguientes pasos. En primer lugar, necesitaremos tener la aplicación del profesor operativa en un equipo de la red en la que vayamos a instalar la aplicación web, tras esto prepararemos un contenedor de servlets para responder a las peticiones que se hagan sobre el cliente.

Tomaremos como partida los proyectos de Visual Studio anexos al presente documento y el proyecto J2EE que contiene la aplicación web. La versión utilizada de Visual Studio es la 2008, la versión de Eclipse para el desarrollo Java utilizada es "Eclipse IDE for Java EE Developers - Helios", y el SO sobre el que se realiza el montaje un Windows XP con ServicePack 3.

1.- Instalación de aplicaciones necesarias

Instalaremos las aplicaciones FreePascal 2.2.2 y AppServ 2.5.10 sin modificar las ubicaciones ofrecidas por defecto en sus programas de instalación.

Será necesaria la instalación del Framework de .NET en el Windows XP donde se vaya a usar la aplicación del profesor.

Instalaremos el contenedor de aplicaciones Resin 4.0.6 en su versión GPL, sin realizar modificaciones frente a la configuración propuesta por el instalador. Resin dispone de múltiples variables configurables, entre las cuales debemos prestar especial atención al puerto desde el que ofreceremos servicio, configurable en resin.xml.

Resin 4.0.6 necesita "Java SE Development Kit 6 Update 26" instalado en el sistema para funcionar. Se ha incluido el instalador en el CD del PFC.

2.- Puesta a punto de BBDD

Una vez instalado el AppServ accederemos a la siguiente URL, http://localhost//phpmyadmin, lo cual abrirá el gestor de bases de datos MySql instalado. Accederemos poniendo el usuario y password que indicásemos en la instalación de AppServ. Deberemos crear una base de datos, cuyo nombre más adelante especificaremos en un fichero de configuración de la aplicación, la llamaremos visualdebugger. Necesitaremos también un usuario con permisos de lectura y escritura sobre la base de datos creada, en los proyectos estamos utilizando el usuario root/root. Dicho usuario también será especificado en la configuración de la aplicación.

Una vez creada la base de datos, lanzaremos los ficheros de carga bbdd.sql y jc_clienteweb.sql desde el phpMyAdmin, que crearán las tablas de la aplicación e introducirán los datos mínimos necesarios para comenzar a usar el sistema.

Es preciso realizar dicha carga importando el fichero SQL, y no copiando y pegando su contenido, ya que, dada la codificación del mismo, podríamos encontrarnos problemas en la ejecución del script.

3.- Importación de proyectos en Visual Studio 2008

En el fichero zip adjunto encontraremos 2 proyectos de Visual Studio: XMLTraceAdministrator y XMLTraceDebugPlugins. Copiaremos las 2 carpetas en

donde queramos mantener el proyecto, el Visual Studio generará en el raíz del directorio donde las copiemos una nueva carpeta, de nombre bin, donde generará los ejecutables y dll's de la aplicación.

XMLTraceAdministrator: En esta ubicación encontraremos el proyecto del profesor. Deberemos modificar el fichero appConfiguration.xml para incluir la configuración adecuada al equipo donde va a ser ejecutada la aplicación:

<DB_Hostname>localhost</DB_Hostname>

<DB_Database>visualdebugger</DB_Database>

<DB_Username>root</DB_Username>

<DB_Password>root</DB_Password>

Estas etiquetas son las que deben contener los datos que hemos definido previamente, el host sobre el que corre la base de datos, su nombre, usuario y contraseña.

El fichero Server.config contiene información acerca de la configuración del servidor de la aplicación, como el puerto por el que escucharemos peticiones del WebService. Para esta primera puesta a punto no modificaremos ningún parámetro del mismo.

XMLTraceDebugPlugins: Este proyecto generará la DLL con la información sobre el compilador que utiliza la aplicación para generar las trazas.

5.- Generación de soluciones

En primer lugar accederemos al proyecto XMLTraceAdministrator, realizaremos las modificaciones que consideremos necesarias en appConfiguration.xml, y generaremos solución. Tras una breve espera el Visual Studio debería informarnos de que la compilación ha sido correcta.

Tras esto, abriremos XMLTraceDebugPlugins, y sin realizar modificaciones generaremos solución. De la misma manera que en el proyecto anterior, Visual Studio debe indicarnos que todo ha sido correcto.

Con esto, habremos generado todo lo necesario para la aplicación del profesor. Ahora, si vamos a la carpeta bin que mencionamos en el apartado de importación de los proyectos, veremos que en su interior hay una carpeta llamada PC, la cual contiene los ejecutables del profesor. Ejecutaremos "VisualDebugger Administrator.exe", lo cual, si los pasos anteriores han ido bien, abrirá dos nuevas ventanas, una en la que veremos las trazas de logs generadas por la aplicación, y otra con la aplicación en sí.

Entre los errores de compilación que se han encontrado cabe destacar que, al cambiar de una versión de Visual Studio a otra, una referencia utilizada en el proyecto, MySql.Data, no se encontraba. Esto se resuelve eliminándola del proyecto y añadiéndola de nuevo, con lo que es de suponer que el problema es debido a las versiones de la librería.

6.- Generación de la aplicación web

Utilizando Eclipse abriremos el workspace adjunto a este documento, en el que se contiene el código fuente de la aplicación.

Deberemos editar el fichero webservices.properties para indicar la URL con la que se conectará a los servicios web de la aplicación del profesor y el archivo datos_bbdd.properties, donde se configura la cadena de conexión a la base de datos donde se encuentra el esquema propio del cliente. Para el ejemplo actual hemos utilizado la misma base de datos para ambas aplicaciones.

Una vez hecho esto, procederemos a reconstruir el workspace y a exportar el proyecto XMLTraceClient como un archivo WAR, el cual es un encapsulado con todos los archivos necesarios para la ejecución.

Dicho archivo WAR lo ubicaremos en la carpeta de Resin webapps, desde la que, una vez arranquemos Resin, estará disponible la aplicación.

7.- Ejecución y prueba

Una vez realizados todos los pasos anteriores tendremos los ejecutables de las aplicaciones del profesor y del alumno listos para su prueba en nuestras máquinas.

En la carpeta bin encontraremos la carpeta PC del profesor. La aplicación del profesor podemos lanzarla desde ese mismo directorio.

Para ejecutar el cliente web lanzaremos Resin utilizando resin.exe y pasándole el parámetrogui por línea de comandos (se recomienda crear un bat que facilite esta tarea). Esto pondrá la aplicación web a disposición de los usuarios en http://localhost:XXXX/XMLTraceClient/, donde XXXX corresponde al puerto que se haya configurado como disponible en el fichero resin.xml (etiqueta <http address="*" port="8090"/>).

9. ANEXO D: Guía rápida

9.1. Acceso al sistema

Para acceder al nuevo cliente debemos utilizar la URL que se haya configurado mediante el contenedor de servlets utilizado. Siguiendo el ejemplo del anexo anterior, en dicho caso sería http://localhost:XXXX/XMLTraceClient.

Una vez que accedemos, la primera pantalla que se le presenta al usuario es la de registro en el sistema, como puede verse en Figura 9.1. No se permite acceder a ninguna de las funcionalidades del mismo sin antes identificarse.

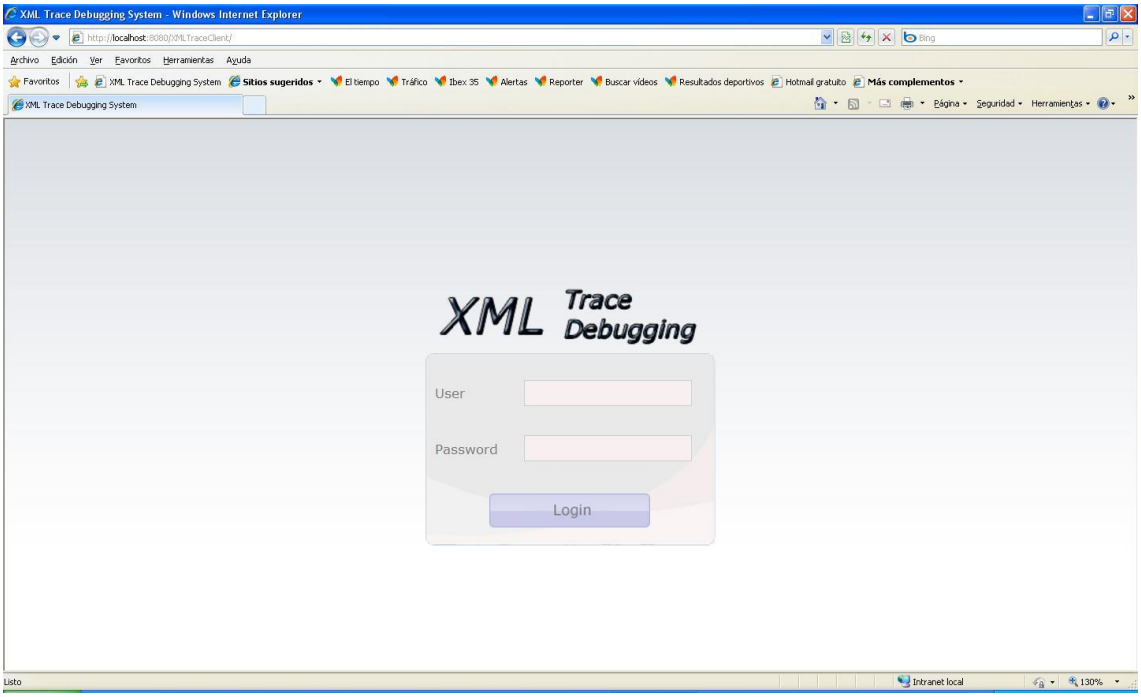

Figura 9.1 Acceso al sistema

En los casos en que el usuario introduzca un usuario y contraseña no válidos, o que no pueda establecerse conexión con la base de datos, se notificará al usuario como se puede ver en Figura 9.2 y Figura 9.3 respectivamente.

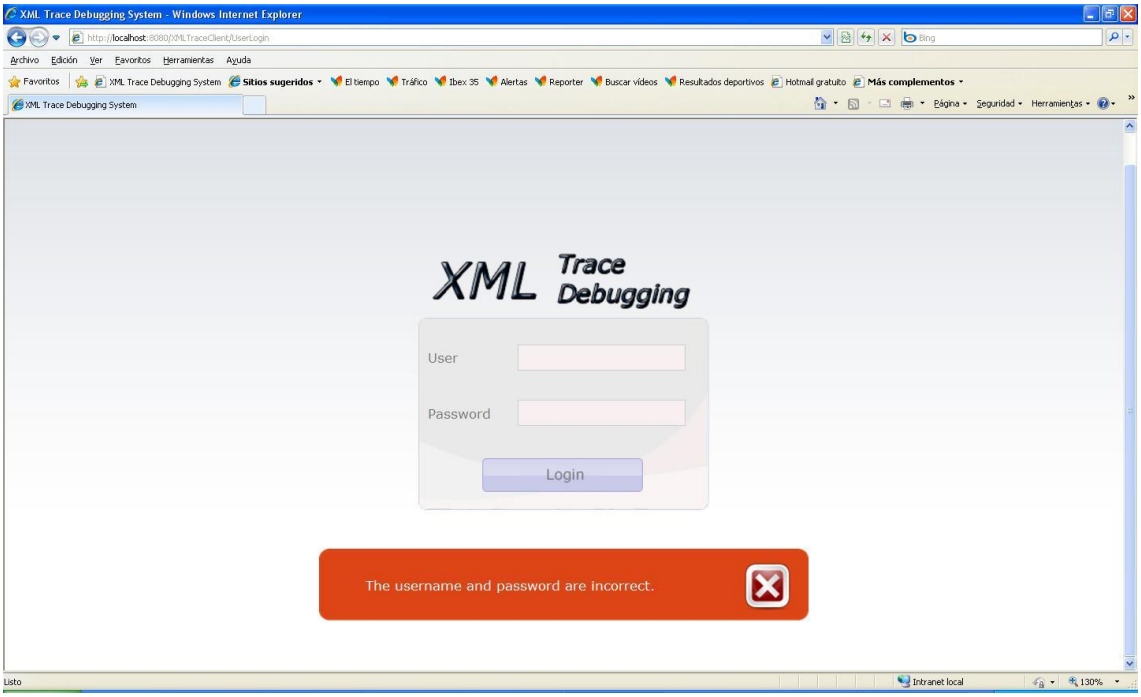

Figura 9.2 Error de acceso, usuario / password incorrecto

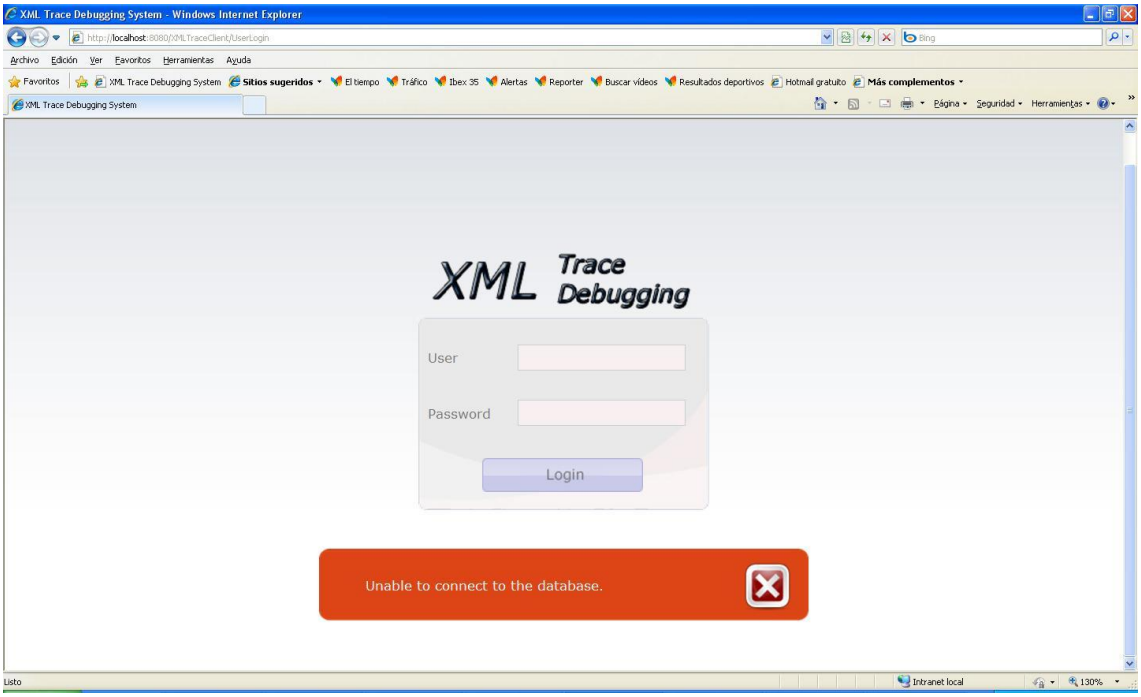

Figura 9.3 Error de conexión con BBDD

Una vez que el usuario se ha identificado en el sistema se accede a la pantalla de selección de datos del ejercicio, en la que, según los grupos en los que esté asignado, podrá elegir en que ejercicio quiere participar. Esta pantalla consta de cuatro combos enlazados, cuyos valores se cargan en función de las selecciones del usuario, y que definen el ejercicio a realizar.

Como ya hemos dicho anteriormente, si el usuario elige Grupo, Proyecto y Entrada de datos, se le concederá acceso a la sección de Depuración; si escoge Grupo, SubGrupo y

Proyecto podrá acceder a la sección de Visibility; y si escoge valores en los cuatro combos podrá acceder a las dos secciones, y moverse entre ellas, compartiendo datos comunes entre ambos ejercicios.

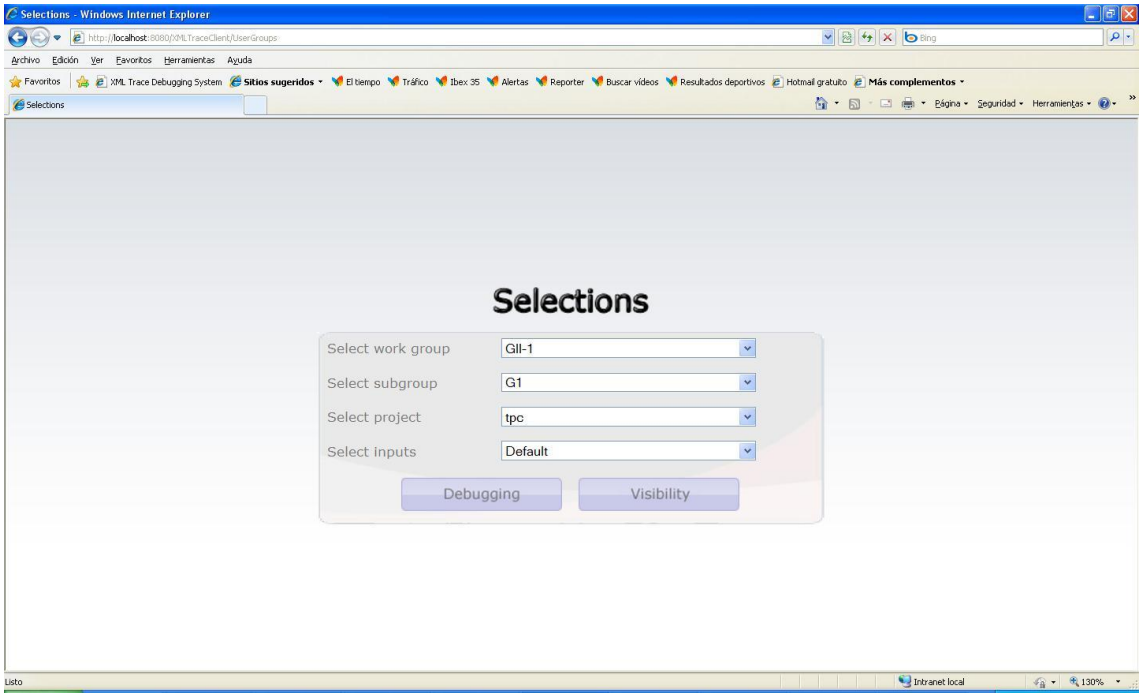

Figura 9.4 Selección de grupo de trabajo

9.2. Depuración

Al acceder a la sección de depuración se nos mostrará una interfaz en la que tendremos toda la información del ejercicio que se está realizando (ver Figura 9.5), tal como código del ejercicio, línea de ejecución, o datos de las variables en ejecución; además, tendremos información del ejercicio, como sus entradas de datos y descripción, e información sobre el resto de alumnos con los que lo estamos resolviendo.

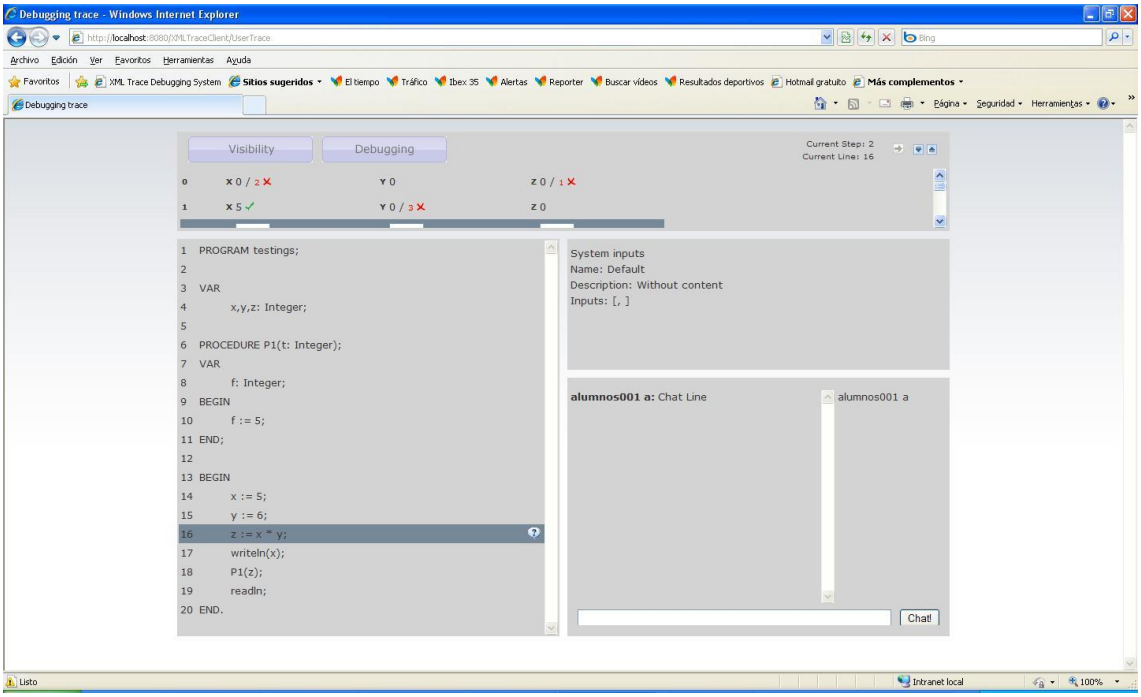

Figura 9.5 Interfaz de depuración

A continuación vamos a desgranar el funcionamiento de la interfaz de depuración, para dar una visión más amplia de todo lo que podemos hacer con ella.

En la parte superior tendremos un cuadro de mandos en el que observamos varios botones. Los dos situados a la izquierda nos llevarán, en caso de que las selecciones hechas sobre los datos de ejercicio lo permitan, de un tipo de ejercicio a otro.

En la parte derecha, además de ver el paso actual de la traza y la línea de ejecución en que se encuentra la depuración, tendremos tres botones. El botón de color gris, con una flecha hacia la derecha, nos permite avanzar la ejecución del programa. Cualquier usuario, en cualquier momento, puede avanzar el paso de traza, pero el sistema controla que no se solapen solicitudes de paso, con el fin de que dos usuarios que quieran pasar del paso 3 al 4, no pasen involuntariamente al 5. Los otros dos botones nos permitirá desplegar el área de variables (ver Figura 9.6), para poder ver todas las variables de cada paso de traza (en negrita las globales y en cursiva las locales) junto a una entrada de texto para proponer valores que serán corregidos según avance la traza, o el valor final, corregido según se avance con un aspa roja o un tick verde, dejando el valor en texto normal si no se propuso ninguno por parte de los alumnos.

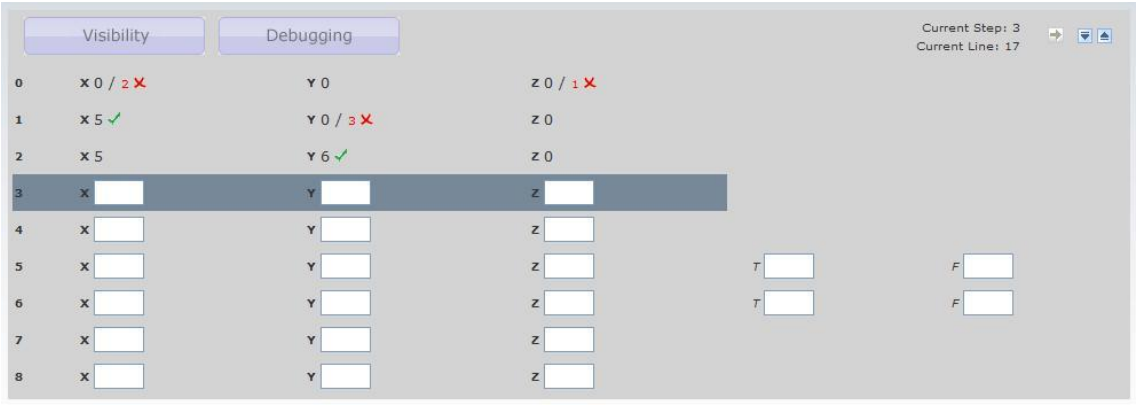

Figura 9.6 Área de variables desplegada

El proceso por el que un usuario propone un valor es simple, tan sólo ha de escribir un valor en una de las entradas de texto, y en cuanto abandone dicha entrada o pulse 'enter' el valor se propagará al resto de usuarios.

9.3. Visibility

Al acceder a esta sección se nos mostrará su interfaz de usuario (ver Figura 9.7), que comparte, como ahora veremos, ciertas secciones comunes con la de depuración. Tendremos un área de código y otra de chat, que comparten datos en común con la sección de Depuración. Además, tendremos un marco en el que se mostrarán todas las aportaciones realizadas a la solución por los usuarios, pudiendo ver a simple vista los valores asignados a cada subprograma, así como si esa aportación es nuestra o de otros usuarios (mediante los iconos en cada valor) y el número de confirmaciones o desconfirmaciones que tiene.

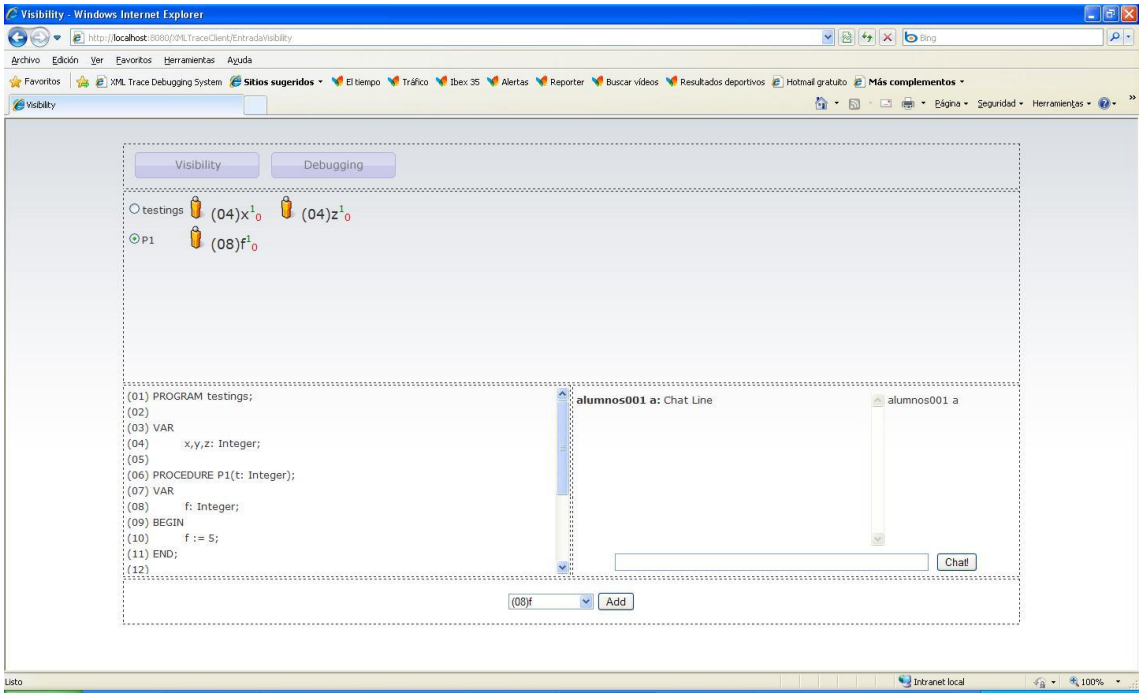

Figura 9.7 Interfaz de Visibility

Si pulsamos en cualquier de los valores de la solución se desplegará una ventana en la que podremos, además de ver cierta información acerca del valor, realizar una serie de acciones sobre dicho valor en función de si somos el autor o no del mismo. Si lo somos, podremos eliminarlo de la solución y si no lo somos podremos confirmar o desconfirmar el valor, como se ve en Figura 9.8.

Figura 9.8 Correcciones sobre valor

Para aportar un nuevo valor debemos escoger uno de la lista desplegable que tenemos en la parte baja de la interfaz, y uno de los procedimientos de la parte izquierda. Tras ello pulsaremos el botón "Add", que guardará el valor como parte de la solución.

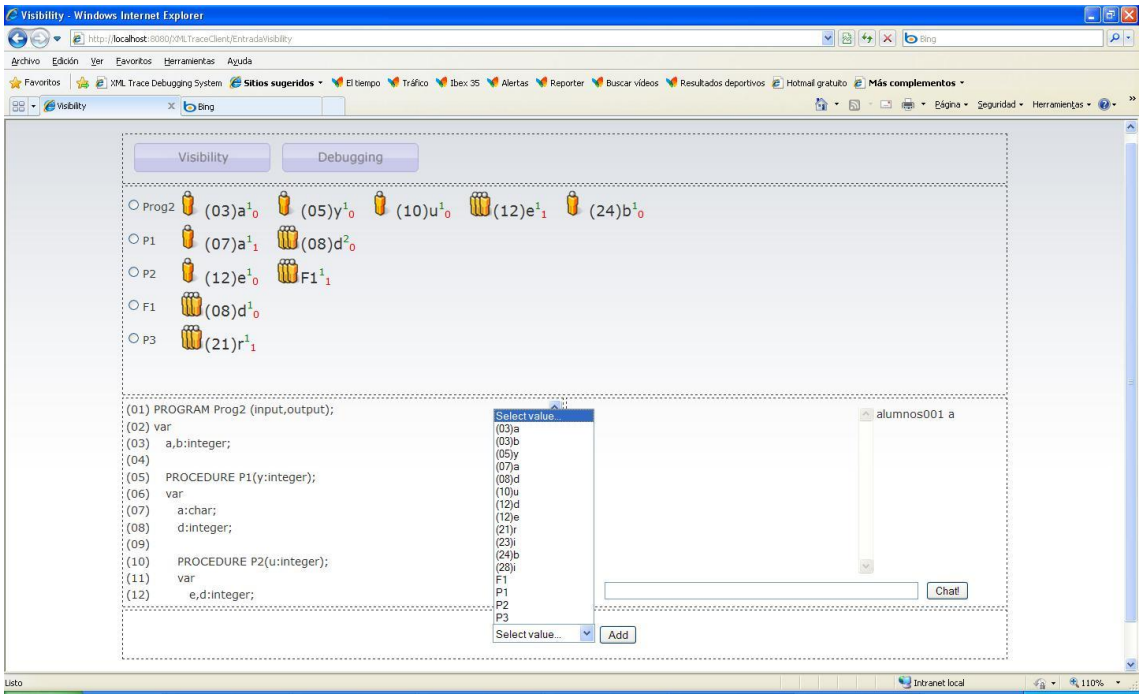

Figura 9.9 Aportar valor a la solución

9.4. Áreas comunes

Ambos tipos de ejercicio de la aplicación comparten dos características comunes, con datos compartidos en determinadas ocasiones. Se trata de la capacidad para dejar comentarios en las líneas de código y el área de chat, funcionalidades creadas para fomentar la resolución en común de los ejercicios.

En cuanto al área de chat (ver Figura 9.10), su funcionamiento es sencillo, cuenta con una columna donde se muestran los usuarios conectados, otra donde se verán los mensajes de los usuarios, y una entrada de texto en la que escribir el mensaje.

Figura 9.10 Área de chat

Para dejar un comentario en una línea de código debemos hacer click sobre la línea deseada, lo cual nos abrirá una ventana en la que escribir el comentario. Ver Figura 9.11.

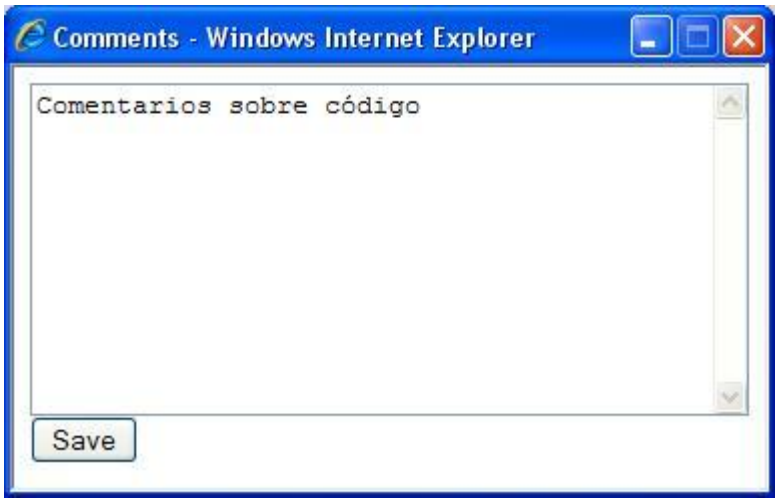

Figura 9.11 Ventana de comentarios

De igual modo, si la línea en cuestión ya tenía un comentario, podremos editarlo y actualizarlo según consideremos oportuno.

Una vez que se hemos puesto el comentario, en la línea de código aparecerá un icono que nos indicará la presencia de comentarios, y que con poner el ratón sobre él mostrará un cuadro de texto con su contenido. Ver Figura 9.12.

Figura 9.12 Línea de código con comentarios