Capítulo 5. Sistema SICODE

A partir de los requisitos establecidos en el Capítulo 4 y las conclusiones obtenidas en el Capítulo 3, en este capítulo y los siguientes abordamos la descripción de los prototipos implementados. En concreto, en este capítulo establecemos los planteamientos generales del sistema SICODE (SIstema COlaborativo de DEsarrollo), las decisiones básicas que se han tomado en su desarrollo [Pérez 2004a] y una descripción del proceso de desarrollo seguido. Por último, realizaremos una panorámica general del sistema en su conjunto.

5.1 Planteamiento general

El planteamiento es implementar el sistema definido por los requisitos del capítulo anterior para poder evaluar las ventajas proporcionadas en la práctica.

SICODE debe de ser fundamentalmente una herramienta que sirva para el desarrollo de aplicaciones de tamaño medio; para ello integra un conjunto de herramientas que facilitan este trabajo a los desarrolladores. Además, SICODE es un entorno orientado a la mejora de la calidad del código de sus usuarios; para realizar esto la herramienta realizará un exhaustivo examen del código mediante procesadores de lenguaje y analizará los resultados de estas herramientas para proporcionar a los usuarios información que le permita evitar errores futuros.

5.2 Decisiones básicas de diseño

A la hora de desarrollar los distintos módulos del sistema se han tomado varias decisiones que condicionan el esquema general del sistema. En los siguientes apartados se hablará de tres decisiones que tienen gran influencia sobre el resto del sistema: el lenguaje de programación al que se dará soporte, la arquitectura general del sistema y el proceso de mejora del código centrado en los errores.

5.2.1 Lenguaje que soportará el entorno

Discutiremos en este apartado la elección del lenguaje de programación que soportará el entorno que desarrollaremos. El diseño general del sistema es que sea un entorno abierto a cualquier lenguaje, ya que esto puede conllevar muchas ventajas como reseña el Dr. Labra [Labra 2003]; sin embargo, en la construcción de los prototipos de evaluación del sistema hemos decidido decantarnos por un lenguaje; aunque siempre dejando abierta la posibilidad de una configuración que permita el uso de otros.

Requisitos del lenguaje de programación

Han sido muchos los artículos que han estudiado los requisitos necesarios de un buen lenguaje de programación en un contexto general. Sin embargo, un lenguaje no es bueno o malo en si mismo sino que depende del propósito para el que se aplique; por tanto, debemos analizar los requisitos en el contexto de nuestro sistema:

- El primer objetivo general de nuestro entorno es la mejora de la calidad del código escrito por los programadores, entendiendo por calidad de código que haga lo que tenga que hacer y sea fácilmente legible y mantenible. Por tanto, orientaremos los requisitos más hacia la calidad del código fuente que hacia el rendimiento o la optimización del código objeto generado por el entorno de compilación del lenguaje.
- El segundo propósito general es tratar de favorecer la transición de estudiantes o desarrolladores con poca experiencia desde pequeños proyectos a proyectos de software de mayor tamaño. Para ello el sistema debe disponer de herramientas avanzadas y que faciliten la colaboración por varios programadores.
- El tercer objetivo es disponer de un lenguaje que facilite la búsqueda de errores en el código fuente.

Teniendo en cuenta estos objetivos y los requisitos que plantea Kölling [Kölling 1999c] para los lenguajes para la enseñanza de la programación, planteamos los siguientes requisitos que nos guiarán en la búsqueda del lenguaje de programación más adecuado:

1. **Orientación a objetos pura**. El lenguaje que elijamos debe de ser orientado a objetos ya que todas las técnicas y métodos actuales se sustentan sobre este paradigma; por tanto, debemos crear el modelo mental correcto y entrenar a los programadores en este paradigma y es la única forma de que nuestro entorno pueda ser capaz de soportar proyectos reales.

   Además, el lenguaje debería ser orientado a objetos puro y no un lenguaje híbrido. Hay mucha gente que argumenta que los lenguajes híbridos permiten una transición más fácil desde lenguajes que siguen el paradigma estructurado como C. El problema de los lenguajes híbridos es que no fomentan el cambio de estilo de los programadores con experiencia previa en otros paradigmas. Al contrario, permiten seguir escribiendo programas creyendo que son orientados a objeto y sin embargo olvidarse de los conceptos clave. Los programadores, pueden considerar que cuando un programa compila sin errores ya está bien programado. De la misma forma, consideran que cuando el compilador de C++ no da errores ya tienen un programa orientado a objetos, lo cual puede ser totalmente falso. Esto puede ser un inconveniente a la hora de motivar a los programadores a mejorar su calidad de código. El lenguaje debe ser el instrumento que fuerce a los alumnos a escribir un buen código orientado a objetos y es la base para evitar futuros defectos en el software.

2. **Seguridad**. El principio de seguridad consiste en que se detecten el mayor número de errores posible bien por el compilador o por el entorno de ejecución. Más aún, deberían detectarse lo antes posible y deberían proporcionarse mensajes sobre la causa de estos. Por otra parte, deberían evitarse las construcciones del lenguaje que sean propensas a errores.

   En la línea de este requisito están los lenguajes fuertemente tipados. Lenguajes como C++ no son fuertemente tipados y pueden producirse errores en tiempo de ejecución difíciles de encontrar. Los lenguajes con tipos dinámicos como Smalltalk

tienen inconvenientes similares. Todo esto provoca que el punto de localización del error esté muy lejano de la fuente del problema y sea difícil asociarlas y por tanto, comprender y eliminar el error.

Otra característica que también proporciona seguridad es la comprobación de los límites de un array o la eliminación de construcciones problemáticas como los punteros explícitos, varios lenguajes permiten programar sin utilizarlos. Muchas veces se utiliza el rendimiento como argumento en contra de estas características, sin embargo, cada vez prevalece más la escritura de un código de alta calidad, libre de errores y que permita un mantenimiento simple, que el rendimiento puro en tiempo de ejecución de una aplicación.

3. **Alto nivel**. El programador debería poder despreocuparse de la estructura interna del sistema. El compilador y el entorno de ejecución pueden encargarse de esta tarea. Un claro ejemplo que va en contra de este requisito es, la gestión explícita de la memoria dinámica por parte del programador. Poner la gestión de memoria dinámica en manos de un programador principiante es una fuente segura de frustraciones ya que los errores de punteros perdidos, doble liberación u otras formas de corrupción de memoria son uno de los tipos de errores de los más difíciles de depurar. Todo esto se puede evitar si se utiliza un sistema de recolección de basura. La utilización de estos sistemas se considera tan positiva que varios lenguajes los incorporan en sus entornos de producción. De nuevo, está el balance entre el rendimiento y la calidad del código que cada vez se inclina más hacia el segundo planteamiento.
4. **Sintaxis legible**. La sintaxis debería de ser consistente y fácil de leer. Este parece un requisito menor; pero hay varias razones que lo justifican. Un programa fácilmente legible también es más fácil de corregir. Es verdad que, una sintaxis más legible no garantiza la corrección del código; sin embargo, determinados casos de error se producen porque un programador no entiende otra parte del código preexistente. Si asumimos que la legibilidad incrementa la comprensibilidad, entonces es clara su importancia.

   Uno de los aspectos más importantes es que el lenguaje favorezca las palabras reservadas sobre los símbolos. Las palabras son mucho más intuitivas que los símbolos. Esto hace que el lenguaje sea más fácil de aprender y sus programas más fáciles de leer. El C++ es el más claro ejemplo de cómo el uso de símbolos puede ir en detrimento de la claridad, en este lenguaje se ha puesto mucho énfasis en tener un número lo más pequeño posible de palabras reservadas. Esto conduce a que haya palabras reservadas que se utilicen en distintas construcciones con distintos significado semántico; por ejemplo: static o el uso de símbolos como "= 0" después de una cabecera (lo cual podría sugerir una asignación) para indicar que es un método abstracto. Incluso programadores con experiencia tienen problemas con esto.

   Mucha gente para aprender un lenguaje o programa lee directamente código fuente de programas bien sean ejemplos o programas para modificar; no lee el texto explicativo de los programas o los comentarios que, si existen, muchas veces son incompletos. Este aprendizaje basado en ejemplos es una forma de aprender muy potente que todos hemos aplicado alguna vez.

   Otro aspecto de la legibilidad es la consistencia. La misma sintaxis debería ser usada para la misma semántica y diferente sintaxis para diferentes semánticas. Actualmente los programadores invierten mucho más tiempo leyendo código para refactorizarlo, corregirlo o ampliarlo que escribiendo código nuevo; si tenemos en

cuenta esto a la hora de elegir un lenguaje no debemos quedarnos con el que permita escribir código de forma más rápida sino con el que proporcione la forma más clara de lectura. Además, a la hora de escribir código los programadores deberían pensar en el futuro lector de este código y hacer las cosas lo más evidentes posibles y cuando no sea así añadir un buen comentario.

5. **Fácil transición**. Cuando se plantea un lenguaje para el aprendizaje de la programación, uno de los factores que hay que tener en cuenta es que las competencias adquiridas sean relevantes para el futuro profesional de los programadores. Aunque no hay que confundir el hecho de aprender un lenguaje de programación con aprender los conceptos, técnicas y habilidades necesarias para programar. Por tanto, la importancia está en los principios que sigue ese lenguaje y no tanto en los detalles. Esto proporciona a los programadores una gran versatilidad y capacidad de adaptación a futuros lenguajes, cosa muy necesaria con la velocidad actual de cambio de la industria del software. Por tanto, los conocimientos y habilidades adquiridas deben de ser fácilmente transferibles a otras situaciones que se darán en el futuro.
6. **Mecanismos para garantizar la corrección**. Es importante adiestrar a los programadores para que integren técnicas y buenas prácticas de ingeniería del software que les ayudarán a obtener un código de alta calidad. Entre otras técnicas están aquellas que permiten integrar en el código comprobaciones sobre su validez como técnicas de programación por contrato: precondiciones, postcondiciones e invariantes. Es muy importante que el lenguaje permita integrar mediante asertos estas técnicas: esto permite utilizarlos desde el principio, e integrarlos en el propio proceso de programación.

Razones de la elección del lenguaje Java como lenguaje para los prototipos

Teniendo en cuenta los requisitos generales enunciados en el apartado anterior se ha elegido Java como lenguaje soportado por los prototipos desarrollados.

Java [Arnold 1996] es uno de lenguajes orientados a objetos más populares actualmente. A continuación estudiamos este lenguaje bajo la óptica de los requisitos anteriores.

1. **Orientación a objetos pura**. Java es básicamente un lenguaje orientado a objetos puro. Todo el código es parte de una clase y las clases son la unidad estructural de código. Con la excepción de los métodos estáticos con el método main a la cabeza necesario para lanzar la ejecución.
2. **Seguridad**. Java es un lenguaje fuertemente tipado que combina el chequeo estático con el dinámico. La mayoría de las construcciones se comprueban en compilación, mientras que alguna se comprueba en tiempo de ejecución pudiendo lanzar una excepción. Java fuerza la comprobación de que los índices al acceder a un array permanezcan dentro de las dimensiones definidas para este.

   Un problema que puede surgir en relación a esto en Java es el tema relativo a la implementación de genericidad como base para las colecciones. En Java el mecanismo de genericidad se implementa mediante el uso de clases genéricas y el polimorfismo; lo cual no permite comprobar los tipos de los objetos que metemos en una colección y no proporciona seguridad al extraer un objeto de esta teniendo que hacer un cast a ciegas. Sin embargo, esta desventaja ha sido eliminada en la versión 1.5 de Java 2 con la introducción de plantillas (templates) que permiten

parametrizar el tipo de los elementos de una colección y así dan la posibilidad al compilador realizar una comprobación estática de tipos.

3. **Alto nivel**. Las construcciones en este lenguaje permiten un alto nivel de abstracción permitiendo al programador despreocuparse por los detalles de bajo nivel. La faceta más destacada es la eliminación de todo el trabajo explícito con punteros y la utilización de un recolector de basura que evita al programador tener que trabajar manualmente con la memoria dinámica.
4. **Sintaxis legible**. La sintaxis es un punto débil de Java. Esta sintaxis estilo C reduce la legibilidad de varias formas: en primer lugar la preferencia por los símbolos respecto a las palabras clave, hace que la lectura de los programas sea a veces poco intuitiva. En segundo lugar el estilo flexible de la sintaxis; permite que los programadores utilicen distintos estilos que si se mezclan dificultan el mantenimiento del código fuente. Esto obliga a la introducción de guías de estilo, que son susceptibles de no ser cumplidas, ya que el compilador no comprueba estas guías.
5. **Fácil transición**. El propio lenguaje Java está siendo cada vez más utilizado en la industria con lo que realmente no sería necesaria ninguna transición para el estado actual de la industria. De todas formas, Java recoge los principios modernos de desarrollo de software con lo que facilita la transición a otros lenguajes más específicos o a lenguajes que puedan surgir en el futuro.
6. **Mecanismos para garantizar la corrección**. A partir de la versión de Java 2 1.4 el lenguaje soporta asertos que permiten implementar pre, postcondiciones e invariantes de clase, con lo que podemos realizar una programación por contrato integrada dentro del proceso de desarrollo de una aplicación en este lenguaje.

Vemos que el lenguaje Java cumple de forma muy destacada los requisitos exigidos, en líneas generales vemos que es un lenguaje de alto nivel que implementa de forma clara los principios de la orientación a objetos, en el que se proporciona seguridad al programador en la detección de errores que proporciona mecanismos para garantizar la corrección y que es un lenguaje cada vez más aceptado en la industria y de todas formas permite dar el salto de forma sencilla a otros lenguajes orientados a objetos. Los problemas pueden venir por una sintaxis heredada del C que en algunos casos puede resultar difícil de leer, de todas formas esto puede ser atenuado estableciendo un buen estilo a la hora de escribir código que podrá ser reforzado por un entorno que incluya un buen editor de código.

A estas características podemos añadir tres más que son relevantes para nuestro entorno:

- Gracias a la comprobación estática de tipos es posible la captura de muchos errores en el código. Por otro lado, la utilización de excepciones como forma estándar notificación de errores en tiempo de ejecución permite su captura y tratamiento automáticos.
- Existen gran cantidad de herramientas para este lenguaje que permiten la comprobación del código de forma estática y que permiten reforzar las comprobaciones del compilador de código.
- Actualmente es el lenguaje que se emplea en la mayoría de las asignaturas de programación en la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo, sobre los cuales realizaremos la evaluación del prototipo.

5.2.2 Arquitectura del sistema: centrado en Internet

Actualmente estamos bastante cerca de aquella utopía planteada por Sun a finales de los 90: el "net – computer". Ya existen muchas aplicaciones donde el ordenador local debe disponer de unos recursos mínimos básicamente un navegador Web y una conexión a Internet, ya que el almacenamiento y el procesamiento lo realiza la o las máquinas remotas al otro lado de la red. Un ejemplo lo tenemos en el correo de Google (gmail[3]) no consiste en un simple lector Web de correo electrónico sino que está pensado para que los correo queden almacenados siempre en el servidor; y más aún que no haga falta borrar ninguno. En el campo del desarrollo de software tenemos alguna experiencia en esta línea como es Sourceforge[4]. Sourceforge es un gran "portal" para proyectos. Permite gestionar un proyecto software completo desde la ideas, el reclutamiento de desarrolladores, el desarrollo incluido el almacenamiento de la versión común de los fuentes, y el lanzamiento para los usuarios. Como servicios complementarios de comunicación incluye listas de correo de distintos tipos.

Hemos tomado la decisión de que SICODE vamos un poco más allá que SourceForge Internet no sólo permitirá que los desarrolladores puedan conectarse para acceder a la versión compartida de los archivos sino que la edición y la compilación también se realizarán en red.

5.2.3 Los errores centran el proceso de mejora del código

SICODE plantea un proceso de mejora de código no como algo teórico, donde se estudian los posibles defectos que pueden surgir al escribir código en un lenguaje determinado; sino como algo totalmente práctico, desde la idea de que "a programar se aprende programando" y que la experiencia es muy valiosa, SICODE es un sistema que permite desarrollar aplicaciones; pero en el momento en el que se detecta un posible error SICODE se convierte en **una herramienta que permite aprender de los errores propios y de la experiencia de los demás programadores**.

Para que esto funcione de forma adecuada debemos tomar varias precauciones:

- No debemos conformarnos con los errores que puede proporcionar un compilador, hay que realizar un análisis más profundo. Por eso recurrimos a otro tipo de procesadores de lenguaje, los **analizadores estáticos**, y los usamos en combinación con el compilador. De esta forma, podemos realizar un análisis más profundo del código y detectar distintos tipos de problemas.
- **Historia de compilación**. No sólo mostramos al usuario el resultado del análisis del código sino que lo guardamos para posteriormente procesarlo.
- **Historia activa**. Muchas veces no sólo tiene interés ver el estado actual de un proyecto, puede tener incluso, más interés ver la evolución de un proyecto. Esto es lo que nos proporciona la historia activa poder recorrer y comparar distintas versiones de un proyecto.
- **Análisis de errores de la historia de compilación**, el sistema calcula estadísticas sobre la información contenida en la historia de compilación: errores más frecuentes, evolución de errores. Esto permite a los desarrolladores darse cuenta de forma más fácil de cuales son sus puntos débiles y fuertes.

[3] Se puede acceder a esta aplicación de correo electrónico en http://www.gmail.com

[4] SourceForge: http://www.sourceforge.net

- **Base de conocimientos colaborativa**. Es la clave para que el desarrollador no vuelva a repetir errores y la forma de aprender de la experiencia de otros programadores ya que todos apartan su experiencia añadiendo casos concretos o ejemplos en la base de conocimientos y creando relaciones entre errores que pueden tener puntos en común.

5.3 Proceso de desarrollo del sistema

En los siguientes apartados describimos a grandes rasgos las fases de desarrollo del sistema SICODE.

5.3.1 Planteamiento iterativo

En el capítulo anterior se plantearon los requisitos del sistema; sin embargo, estos requisitos se han obtenido de forma iterativa. Dentro de una línea general se han planteado unos requisitos y se ha implementado un prototipo, que se ha evaluado para a partir de esto refinar los requisitos del sistema y construir nuevos prototipos.

5.3.2 Prototipo Compilador Web SICODE

Tras una primera obtención de requisitos se ha implementado el Compilador Web SICODE que pone en práctica los requisitos fundamentales del sistema y permite evaluar cómo funcionan todos integrados.

5.3.3 División en subsistemas

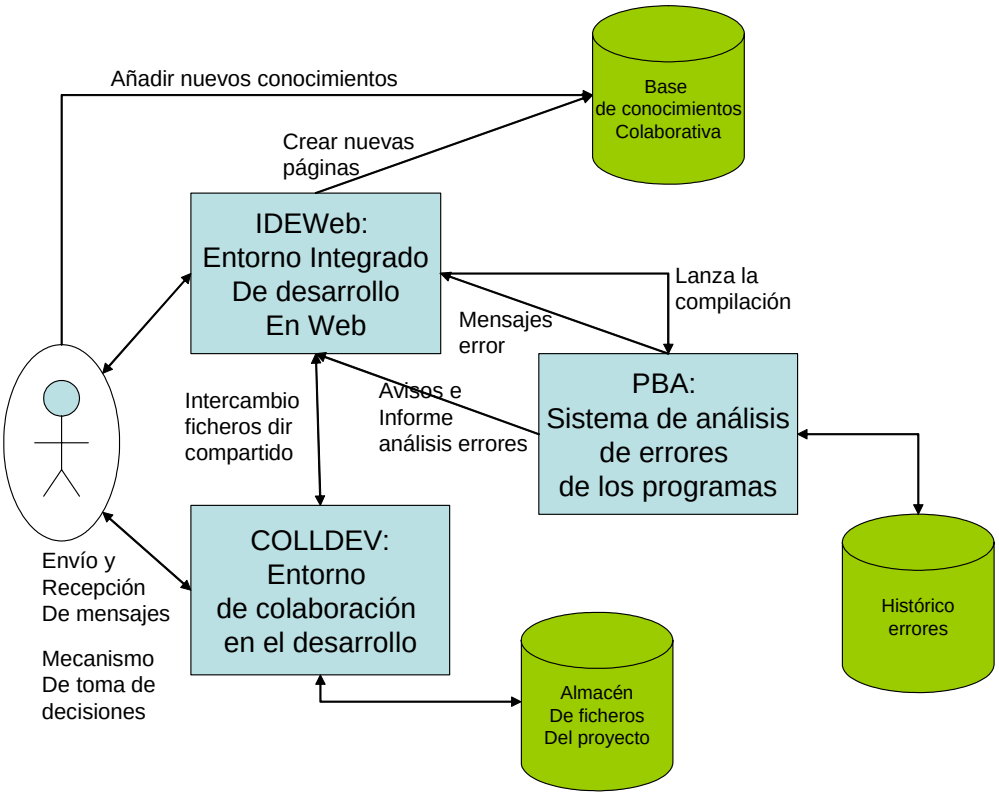


Figura 8. Diagrama de bloques que representa los subsistemas y la interacción entre ellos y con el desarrollador

Una vez desarrollado el primer prototipo, se decidió dividir el sistema en varios subsistemas que se centraran en diferentes aspectos, de forma que estos fueran modulares e independientes. Una vez concluidos estos prototipos podrían unirse para resolver el problema global y poder formar una herramienta útil para los profesores y los alumnos. A continuación se presenta un breve resumen de las tareas que realiza cada uno de los subsistemas.

- **Entorno de desarrollo integrado en Web (IDEWeb)**. Entorno de edición, compilación y depuración sobre Web y base de conocimientos colaborativa que permite introducir ayuda sobre los errores, buenas prácticas y en general ayuda a los errores de programación (errores de compilación y excepciones en ejecución) de forma colaborativa y una forma automática de enlazar esto con cada uno de los errores obtenidos en una compilación. Este es el módulo del sistema con el que el usuario va a interactuar la mayor parte del tiempo.
- **Sistema para la Colaboración en el Desarrollo de Aplicaciones (COLLDEV, Collaborative Development)**: Sistema colaborativo que gestiona grupos de trabajo, maneja los usuarios, el grupo al que pertenecen y se encarga de gestionar las sesiones que abren para interactuar con el sistema, así como facilitar la coordinación entre los miembros del grupos. Gestiona el almacenamiento compartido del proyecto y permite el trabajo simultáneo de varios usuarios con los archivos, implementa un control de versiones para realizar un seguimiento del proceso de desarrollo mediante una historia activa del trabajo.
- **Sistema de análisis de errores de programas (PBA, Program Bug Analysis)**. Sistema de gestión de errores tanto en tiempo de compilación, como en tiempo de ejecución. El tratamiento incluye captura, clasificación, almacenamiento y procesamiento de los errores. El sistema debe de ser capaz de realizar estadísticas sobre los errores, para facilitar a los usuarios la comprensión del tipo de errores que comete.

5.4 Una panorámica general

En este apartado queremos reflejar un escenario típico de interacción entre un desarrollador y el sistema SICODE.

5.4.1 Requisitos iniciales del sistema

Para comenzar a trabajar con el sistema SICODE el desarrollador sólo debe disponer en su máquina local de un navegador Web con el plug-in de Java para poder ejecutar el applet que soporta el editor de código ampliado con el coloreado de sintaxis; no es necesario haber instalado ningún software especial.

5.4.2 Entrando en sesión

Para trabajar en el sistema, el desarrollador, debe disponer de un usuario registrado que le permita identificarse y autenticarse. Si se cumple esta condición el desarrollador simplemente se conectará a la dirección Web del sistema e introducirá su nombre de usuario y su contraseña; si estos datos son correctos el sistema abrirá una nueva sesión para el usuario.

5.4.3 Preparación para empezar a trabajar con el código fuente del proyecto

El sistema también tiene información sobre los grupos de trabajo y los proyectos que pertenecen a cada uno de ellos; por tanto el usuario al iniciar la sesión tendrá disponibles todos los proyectos que pertenecen a su grupo de trabajo.

Antes de empezar a trabajar sobre ellos el usuario debería comprobar que trabajo han realizado el resto de miembros del grupo sobre el proyecto; esto puede comprobarlo utilizando las opciones de "directorio de grupo" dentro de "Espacios de proyecto" del Sistema para la Colaboración en el Desarrollo de Aplicaciones (COLLDEV, Collaborative Development).

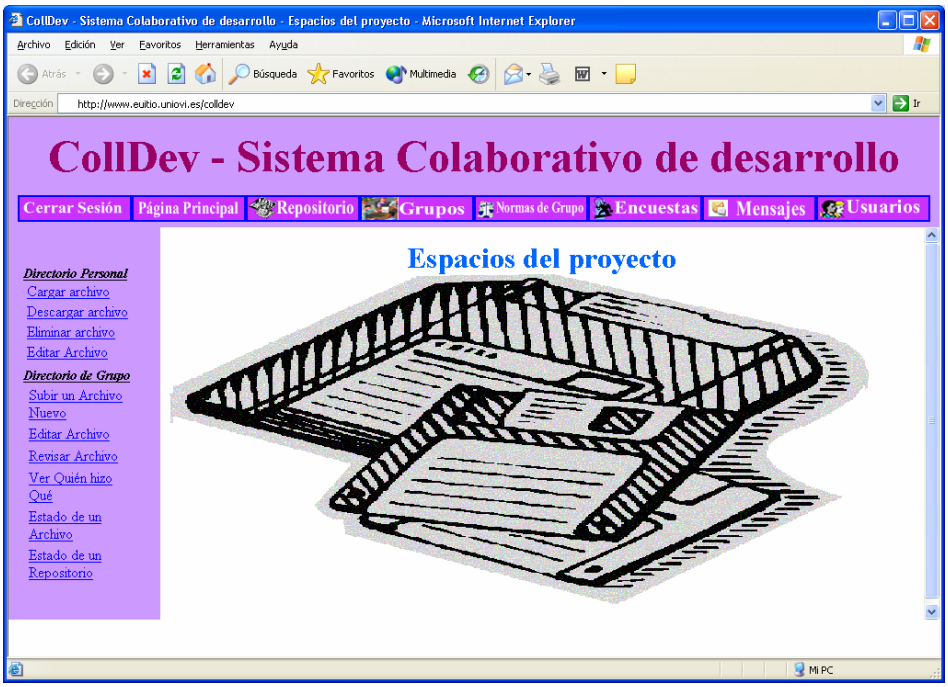


Figura 9. Interfaz del Sistema para la Colaboración en el Desarrollo de Aplicaciones (COLLDEV)

El desarrollador deberá actualizar en su directorio personal las versiones de los archivos del proyecto para disponer de las últimas y empezar a modificar los archivos correspondientes. El sistema se encarga de comunicarse con un sistema de control de versiones (CVS) para permitir que los desarrolladores puedan editar concurrentemente cualquier archivo del proyecto y juntar los cambios al final del trabajo, sin necesidad de que un desarrollador espere a que otro termine.

5.4.4 Comunicación entre los desarrolladores

Muchas veces es necesario que los desarrolladores se comuniquen decisiones que toman en la escritura de código o en la implementación de determinada parte del diseño de una aplicación. El Sistema para la Colaboración en el Desarrollo de Aplicaciones (COLLDEV, Collaborative Development) proporciona una forma sencilla de comunicación interna en formato texto entre los distintos miembros del grupo de desarrollo de un proyecto.

5.4.5 Edición del código fuente de un archivo del proyecto

Para editar un archivo concreto de nuestro proyecto podemos cargarlo desde el espacio de trabajo personal utilizando el gestor de archivos del Entorno de desarrollo integrado en Web (IDEWeb). Este subsistema constituye un entorno de desarrollo que integra distintas

herramientas necesarias para el desarrollo; pero con una interfaz Web, permitiendo su utilización desde cualquier sistema que tenga disponible un navegador.

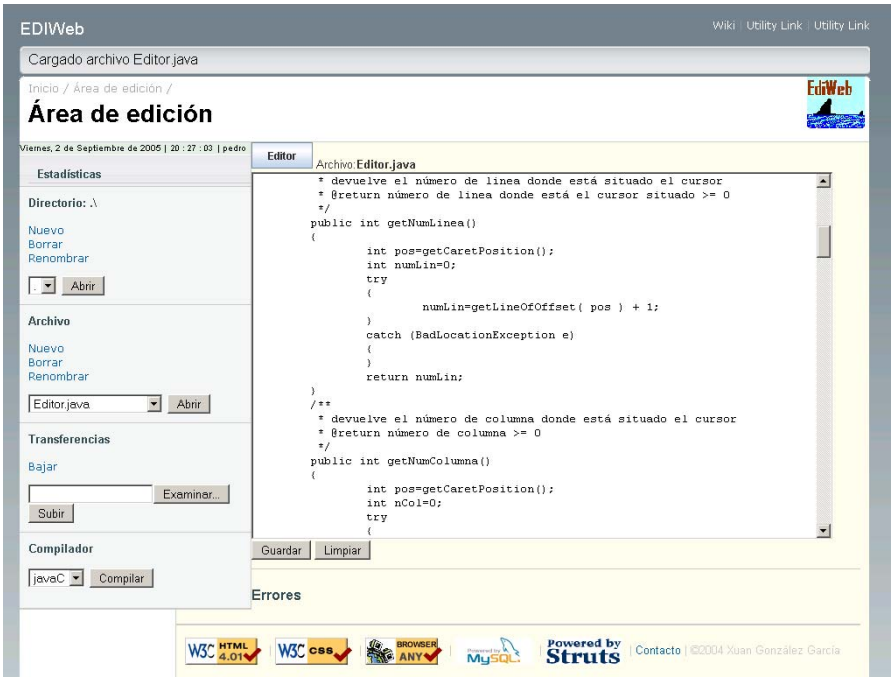


Figura 10. Vista del área de trabajo de usuario en el entorno IDEWeb

El área de trabajo de IDEWeb dispone de distintas opciones de gestión de los archivos del proyecto, además de las opciones de configuración de la compilación remota. Dispone de una gran área de edición donde se muestra el código fuente del archivo que queremos modificar y permite realizar sobre él los cambios que el desarrollador estime conveniente.

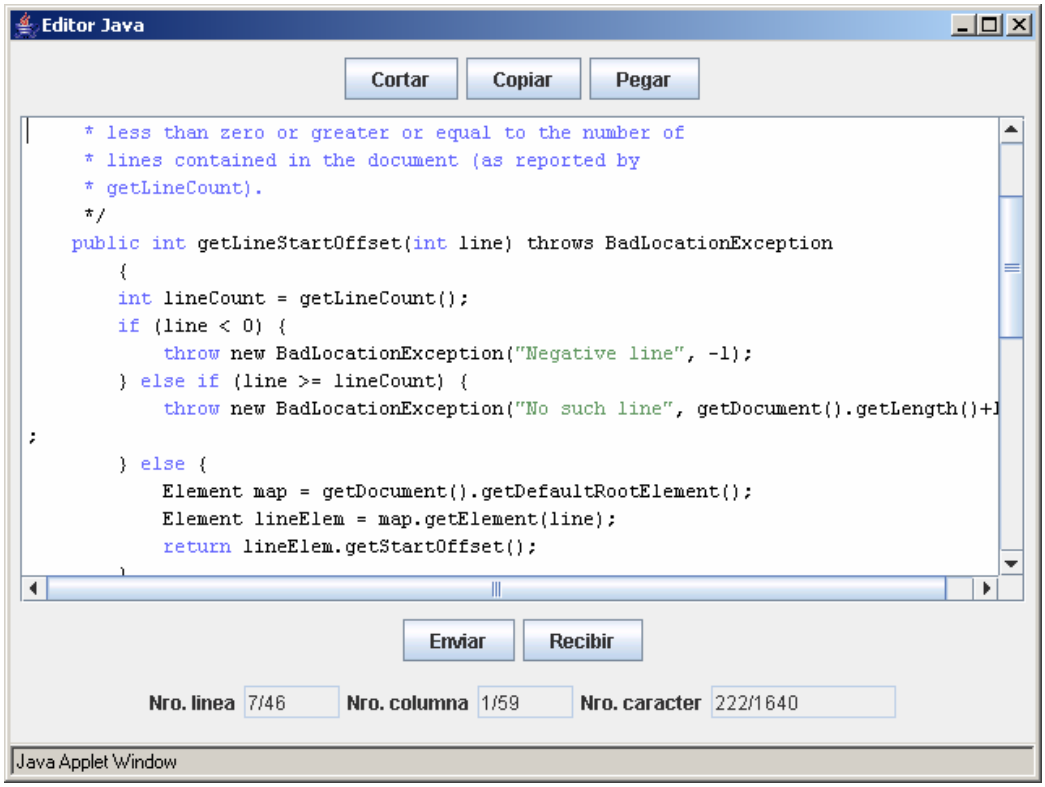


Figura 11. Applet Editor del IDEWeb

Para facilitar la edición de archivos fuente, IDEWeb dispone de un applet editor que dispone de coloreado de sintaxis de los archivos en Java y su forma de trabajar es más parecida a los editores estándar ya que se puede utilizar la tecla de tabulación.

5.4.6 Compilación

Desde IDEWeb se puede invocar una unidad de compilación perteneciente al Sistema de análisis de errores de programas (Program Bug Analysis) PBA. Las unidades de compilación están configuradas por los desarrolladores y especifican las herramientas que se van a utilizar para analizar el código fuente, las opciones de cada una de esta herramientas y el orden en el que se utilizan. Las unidades de configuración utilizan archivos XML compatibles con el estándar de Ant para configurar esto.

Las unidades de compilación proporcionan gran flexibilidad al sistema ya que se puede utilizar simplemente el compilador estándar de Java o se pueden combinar varias herramientas de análisis sintáctico para buscar problemas en el código de forma más exhaustiva.

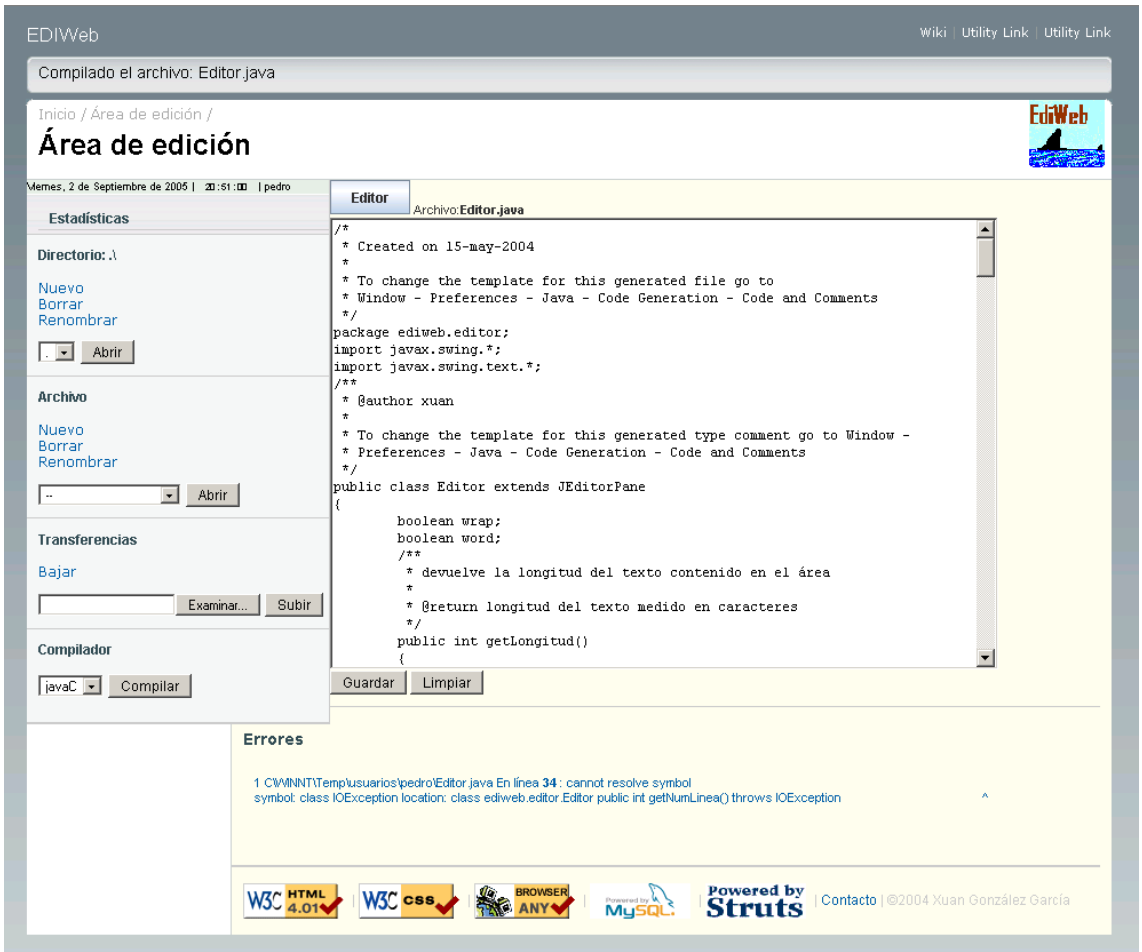


Figura 12. IDEWeb mostrando un mensaje de error después de compilar

El sistema PBA devuelve una vez que a finalizado su trabajo todos los mensajes de error a IDEWeb para que el entorno se los muestre al desarrollador y pueda estudiarlos y repararlos.

5.4.7 Búsqueda en la base de conocimientos para reparar un error

El sistema dispone de una base de conocimientos que permite acceder a información ampliada sobre el error, sus posibles causas, los distintos contextos en los que se puede dar y las alternativas para darles solución. La base de conocimientos también incluye hiperenlaces entre mensajes de error que tengan relación.

El desarrollador puede acceder a la información concreta sobre el error en la base de conocimientos utilizando el hiperenlace que tiene el propio mensaje de error que muestra IDEWeb.

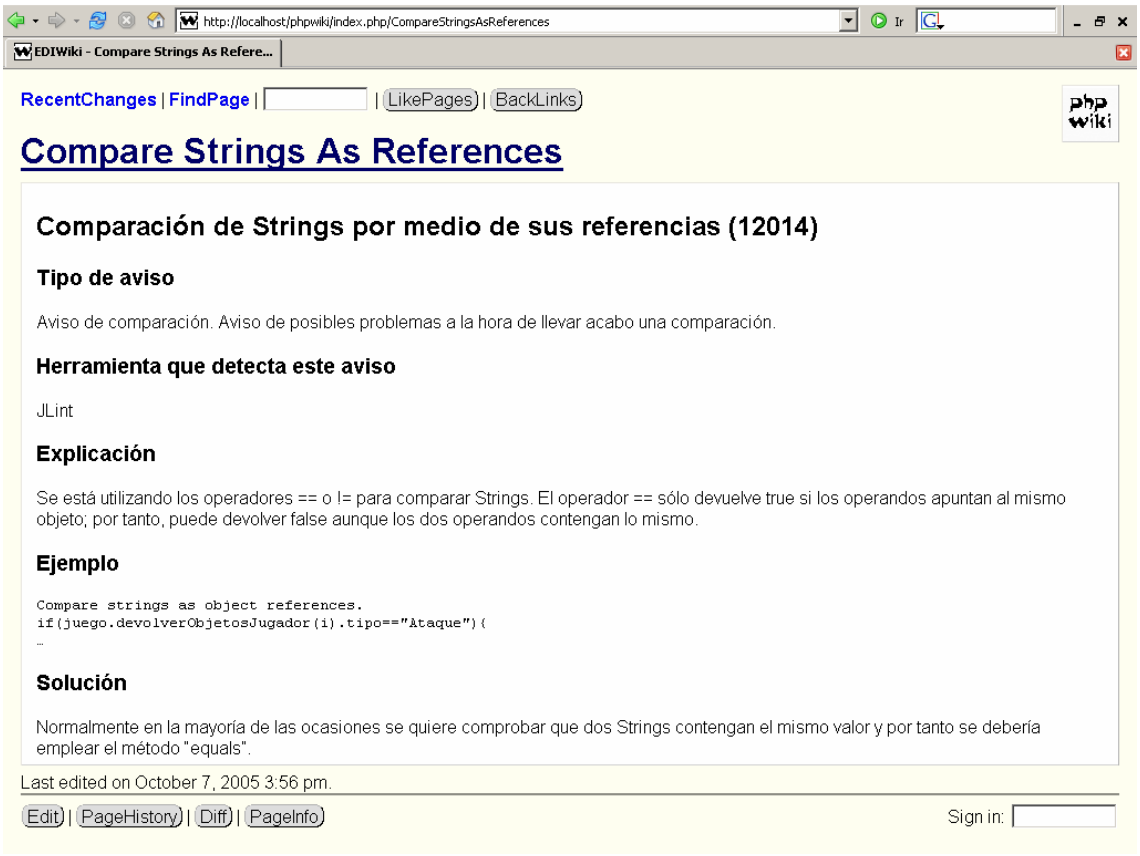


Figura 13. Vista de la página Web con toda la información sobre un aviso.

5.4.8 Almacenamiento de los mensajes en la historia de compilación

El Sistema de análisis de errores de programas (PBA) organiza la ejecución de las distintas herramientas de análisis estático mediante las unidades de compilación y recoge los mensajes generados por estas. Por un lado informa al desarrollador de los problemas encontrados a través de IDEWeb, como se ha descrito en el apartado anterior, y por otro envía estos mensajes a una base de datos para crear un histórico de errores relacionados con cada desarrollador perteneciente al proyecto, es lo que denominamos "historia de compilación".

5.4.9 Toma de decisiones en la corrección de un error

A veces, para solucionar un problema es necesario que los desarrolladores se comuniquen y tomen la decisión correcta sobre cual es la forma de afrontarlo. Tanto en problemas que requieren un rediseño del código como en errores las decisiones conjuntas entre varios programadores proporcionan varias ventajas: la toma de decisiones sobre una base más amplia suele dar como resultado mejores decisiones, permite el intercambio de experiencias

entre los distintos desarrolladores y por tanto el conocimiento de nuevos puntos de vista a la hora de afrontar determinados problemas que siempre resulta enriquecedor y permite separar quién escribió originalmente el código de quién realiza la corrección con lo que aumenta la flexibilidad del equipo. El inconveniente de este sistema es que se emplea mayor cantidad de tiempo para solucionar el error que si se hiciera individualmente; pero esto a largo plazo se compensa con un conocimiento más profundo de los errores y una eficacia mayor en su corrección.

5.4.10 Añadir contenidos a la base de conocimientos

Los desarrolladores utilizan la base de conocimientos para buscar información sobre los distintos errores que pueden aparecer; pero esta base de conocimientos, como las que tratan sobre un tema relativamente amplio, está siempre evolucionando ya que puede que sea necesario añadir información sobre una determinada causa de un error o el error detectado se ha producido en un contexto no previsto entonces el desarrollador tendrá que utilizar sus conocimientos para interpretar la nueva causa del error y buscar nuevas soluciones; pero una vez que consiga la solución, debería compartir estos nuevos conocimientos con el resto del grupo en la base de conocimientos. Para facilitar que se añadan conocimientos se ha intentado facilitar al máximo esta tarea y no se requieren herramientas especiales, el mismo navegador que utilizamos para consultar la base de conocimientos nos sirve para realizar modificaciones sobre ella: añadir nuevo contenido, corregir algún error o borrar datos desactualizados. De esta forma la base de conocimientos irá creciendo a medida que se vaya utilizando.

Por otra parte, el subsistema IDEWeb dispone de un módulo que permite crear nuevas páginas en la base de conocimientos cuando estas no existen. Por ejemplo, cuando se incorpora una nueva herramienta aparecen nuevos mensajes de error que hasta ese momento no existían, no hace falta que se vaya creando página a página las correspondientes a todos los errores sino que el sistema al ir a consultar un mensaje que todavía no tiene página la crea con la información básica sobre ese error para que no haya que empezar desde cero.

5.4.11 Análisis y elaboración de los avisos sobre errores

Código de Error	4	13019	34	10013	11019	13017	12027	11002
Frecuencia	3,333	2,222	1,111	0,667	0,444	0,444	0,222	0,222

Número de compilaciones : 9

Figura 14. Tabla de frecuencia de errores elaborada por PBA

El Sistema de análisis de errores de programas (PBA) trabaja sobre la historia de compilación de un usuario para elaborar avisos e informes que le puedan ser útiles al usuario para que se de cuenta de los errores que comete al programar, aprender sobre sus causas y así poder prevenir nuevos errores cuando escriba más código. Es decir, pretendemos trabajar sobre el estilo de programación y no solamente sobre los errores concretos de una compilación determinada. De forma análoga a como trabajamos con los estilos de aprendizaje en el trabajo del autor junto con la Dr. Paule Ruiz [Paule 2003], pretendemos aquí actuar sobre los estilos de programación.

El sistema permite muchas posibilidades de configuración para elaborar las estadísticas que permitirán seleccionar los avisos que se envíen al programador. Esta configuración se

especifica mediante archivos XML con una estructura claramente definida con un esquema XML que permite su validación automática. Se puede configurar el intervalo temporal del histórico sobre el que se aplica la estadística y el tipo de errores tratados siguiendo los criterios de herramienta, código interno, tipología, etc. De esta forma podemos obtener, por ejemplo, los errores más frecuentes de un usuario en las compilaciones de la última semana o los errores más frecuentes en el último mes. A partir de las estadísticas se seleccionarán los avisos que se mostrarán a cada usuario a través de IDEWeb.

5.4.12 Seguimientos de la historia de trabajo del proyecto

Tanto en la vida académica como profesional es necesario revisar no sólo el resultado final de un proyecto sino estados intermedios para encontrar las causas a determinados errores o conocer la forma en que fueron solucionados, esto es lo que denominamos historia de trabajo.

El Sistema para la Colaboración en el Desarrollo de Aplicaciones (COLLDEV) permite no sólo examinar los proyectos en su estado final sino volver atrás y comprobar que miembro del equipo hizo determinada modificación y de esta forma explorar la evolución del proyecto.

5.4.13 Análisis global de los errores y su evolución

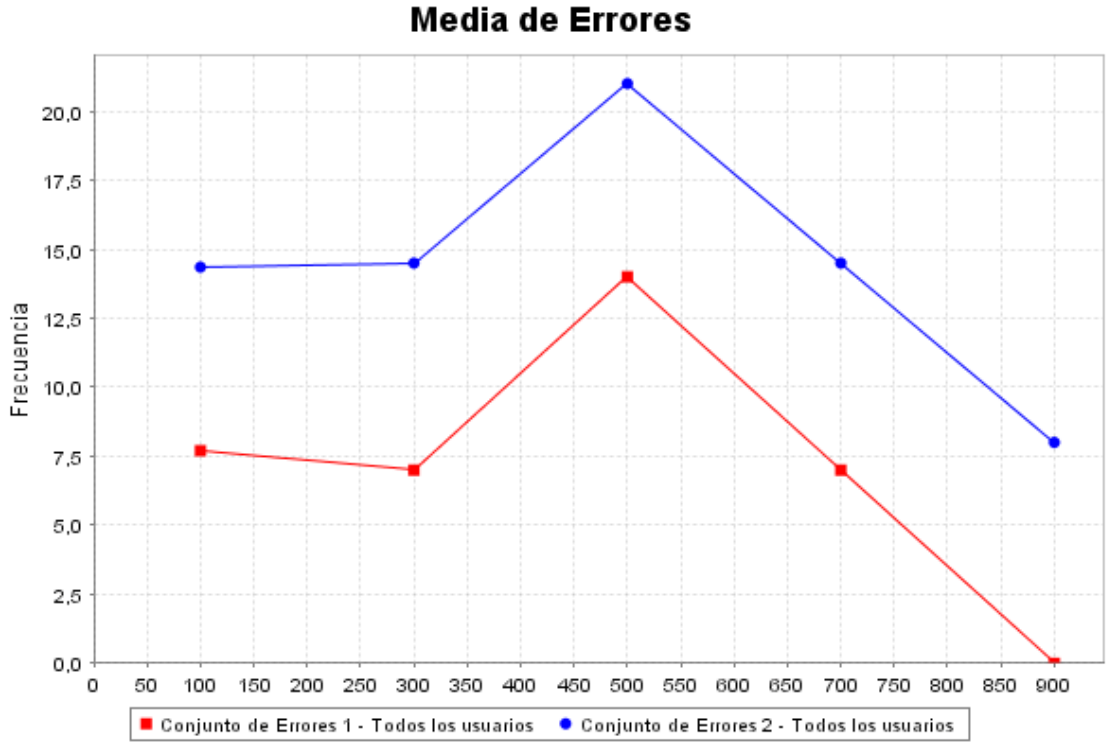


Figura 15. Gráfica de la evolución media de errores para dos conjuntos de errores distintos

De cara al seguimiento global del proyecto nos interesan unas estadísticas diferentes a las proporcionadas a los usuarios individuales, de esta forma es interesante disponer de estadísticas de evolución de errores para un determinado proyecto. El Sistema de análisis de errores de programas (PBA) permite configurar estadísticas que muestren esta evolución. El objetivo es que el estilo de programación evolucione para evitar errores antes de cometerlos y conseguir un código de calidad, y esto nos permite ver gráficamente la evolución de un desarrollador o de un grupo.

Capítulo 6. Compilador Web SICODE

En este capítulo, se describe un prototipo que aborda la implementación de los requisitos esenciales del sistema. Estos requisitos han sido planteados en el Capítulo 4. En esta primera fase de implementación se da prioridad a los requisitos que permiten la escritura de código fuente y compilar utilizando la Web como interfaz. Otro requisito abordado es la mejora del código centrada en el análisis de los errores de compilación, generando mensajes a partir de este análisis y posibilitando la creación de una ayuda semántica asociada a cada error.

Este prototipo se llevó a cabo en el año 2002 [Pérez 2003a][Pérez 2003b] y como el resto de los prototipos soporta Java [Arnold 1996] como lenguaje de desarrollo.

6.1 Objetivos

Los objetos que se plantearon a la hora de crear este prototipo fueron los siguientes:

- Crear un entorno de edición, compilación y corrección de errores, que permita a los usuarios escribir y llevar a cabo la compilación de sus archivos escritos en lenguaje Java, a través de un navegador Web, sin necesidad de tener que instalar en el equipo local ningún tipo de software.
- Permitir la gestión de los archivos que forman el proyecto que se está desarrollando, de forma que el entorno no limite el desarrollo a pequeños proyectos, sino que se puedan llevar a cabo proyectos con un número de archivos grande y para facilitar su gestión se puedan estructurar en paquetes.
- Generar para cada usuario un conjunto de avisos personalizados de manera automática, orientados a evitar errores de programación. Para realizar esto, se analizarán los errores previos que ha cometiendo propio usuario y el resto de los usuarios. Para conseguir una mayor efectividad, se analizarán en tiempo real los datos de las compilaciones y se generarán los avisos basados en este análisis.
- El entorno debe tener la posibilidad de asociar ayuda a cada tipo de error, y que sea fácilmente accesible por parte del usuario cuando esté corrigiendo los errores. Además, la ayuda debe ser fácilmente ampliable y modificable para adaptarla a los usuarios que están utilizando el entorno, en función de los errores frecuentes de cometen y del código que están escribiendo. Esta ayuda permitirá complementar la información de referencia que incluyen típicamente las plataformas de desarrollo, y debe ayudar al usuario a analizar cuál es el problema real que esta causando el mensaje de error, y cómo dar solución a este problema.

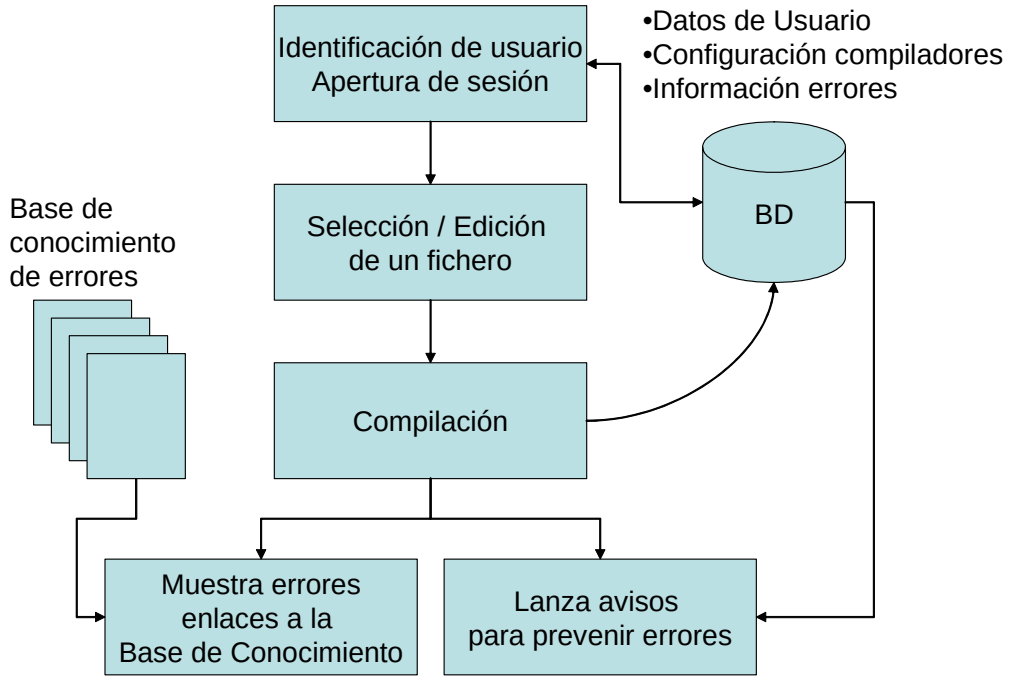


Figura 16. Modelo general del prototipo de Compilador Web SICODE

- Permitir a determinados usuarios (administrador) realizar un seguimiento de los errores de compilación cometidos por un usuario o conjunto de usuarios.

6.2 Requisitos que cumple el prototipo

A partir de los objetivos definidos en el apartado anterior definimos una serie de requisitos tanto funcionales como no funcionales, que especificamos a continuación.

6.2.1 Requisitos funcionales

Los usuarios de la aplicación se deben identificar y autenticar para poder hacer uso del sistema.

Almacenamiento de archivos en la carpeta personal del usuario. El usuario dispondrá de una carpeta personal donde se almacenarán todos los archivos que forman el proyecto o proyectos con los que trabaja el usuario.

Gestión de archivos. La aplicación permitirá realizar una gestión de archivos básica sobre los archivos del usuario: creación de nuevos archivos, selección de archivos existentes. También permitirá intercambiar archivos entre la carpeta personal y otros repositorios externos. Por último, permitirá la lectura y edición de un archivo seleccionado con el editor integrado en el sistema.

Compilación. El sistema permite configurar el compilador que debe utilizar el sistema que será el mismo para todos los usuarios. En la opción de compilación, el sistema utilizará la configuración actual para compilar el código fuente contenido en el archivo seleccionado en el gestor de archivos o que está actualmente en edición. El código objeto generado será almacenado en la carpeta personal. Si el compilador genera errores en la compilación, estos

se capturarán y se mostrarán al usuario para que pueda corregirlos y, por otro lado, se almacenarán para poder procesarlos posteriormente, creando la *historia de compilación*.

Visualización del archivo donde se localiza un error. El sistema debe de ser capaz de localizar el archivo y la línea relacionada con un determinado mensaje de error y mostrar el archivo correspondiente en el editor integrado.

Visualización de la ayuda asociada a un error. El sistema debe de ser capaz de buscar la ayuda hipertextual asociada a un mensaje de error determinado en la base de conocimientos.

Gestión de avisos o mensajes a los usuarios. El sistema realizará un análisis a partir de los mensajes de error almacenados en la historia de compilación y buscará los errores más frecuentes y los últimos cometidos, tanto por el usuario individual como por el conjunto de usuarios de la aplicación. El sistema mostrará información personalizada al usuario sobre sus errores y también sobre los del conjunto de usuarios.

6.2.2 Requisitos no funcionales

El sistema debe evitar que el usuario final tenga que realizar cualquier tipo de instalación y configuración del software.

El sistema debe poder ser utilizado por un grupo de usuarios simultáneamente, cada uno con su conjunto de archivos de la carpeta personal y utilizando conjuntamente los recursos comunes, como la ayuda sobre los errores y los mensajes de grupo, sin que se produzcan interferencias entre ellos.

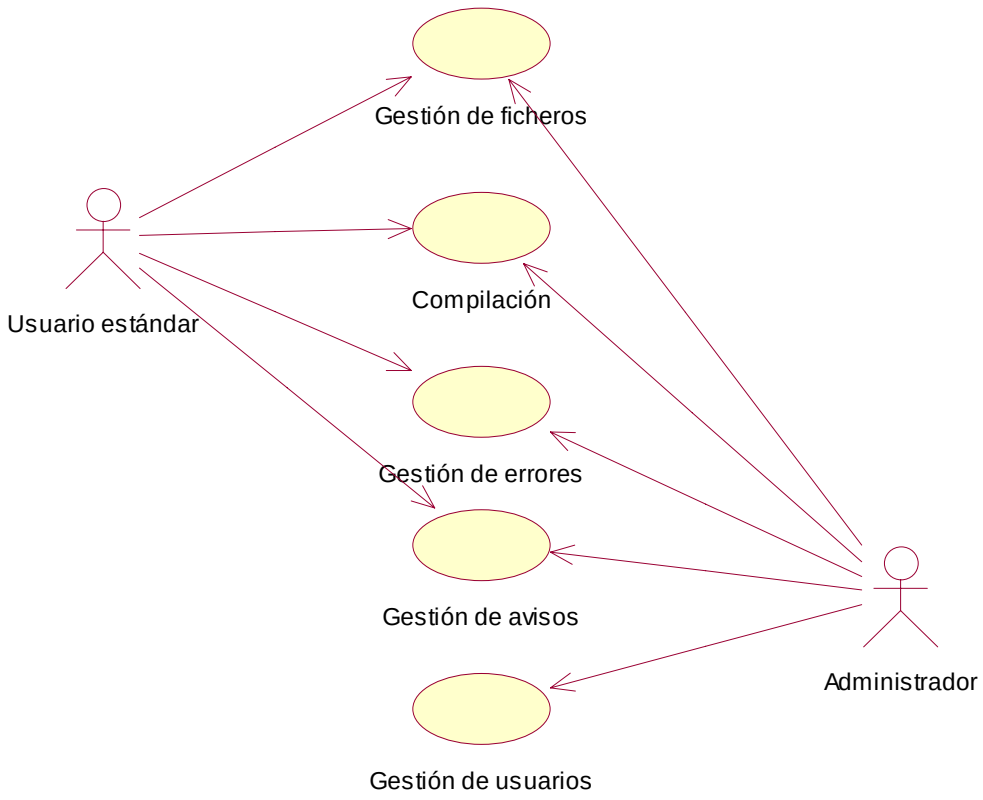


Figura 17. Diagrama de casos de uso del prototipo de Compilador Web SICODE

Se debe almacenar la ayuda de forma modular y en un formato que permita fácilmente su modificación manual para adaptarla a los usuarios que están utilizando la herramienta. Se pretende aquí incluir técnicas de adaptación conceptual [Fernández 2003].

6.3 Casos de uso

En este apartado describimos cada uno de los casos de uso y los actores del sistema, esto lo representamos en la Figura 17.

Administrador. Se encarga de administrar los usuarios del sistema, puede acceder a los ficheros de la base de conocimientos y además, es también usuario estándar de la aplicación. Es el encargado de hacer las altas, bajas, modificaciones y consulta de usuarios. Como usuario de la aplicación podrá abrir, cargar, descargar y compilar archivos, solicitar ayuda y visualizar el archivo que contiene un determinado error.

Usuario estándar. Podrá abrir, cargar, descargar y compilar archivos solicitar ayuda y visualizar el archivo que contiene un determinado error. No tendrá acceso a la información (cuentas y archivos) del resto de usuarios del sistema.

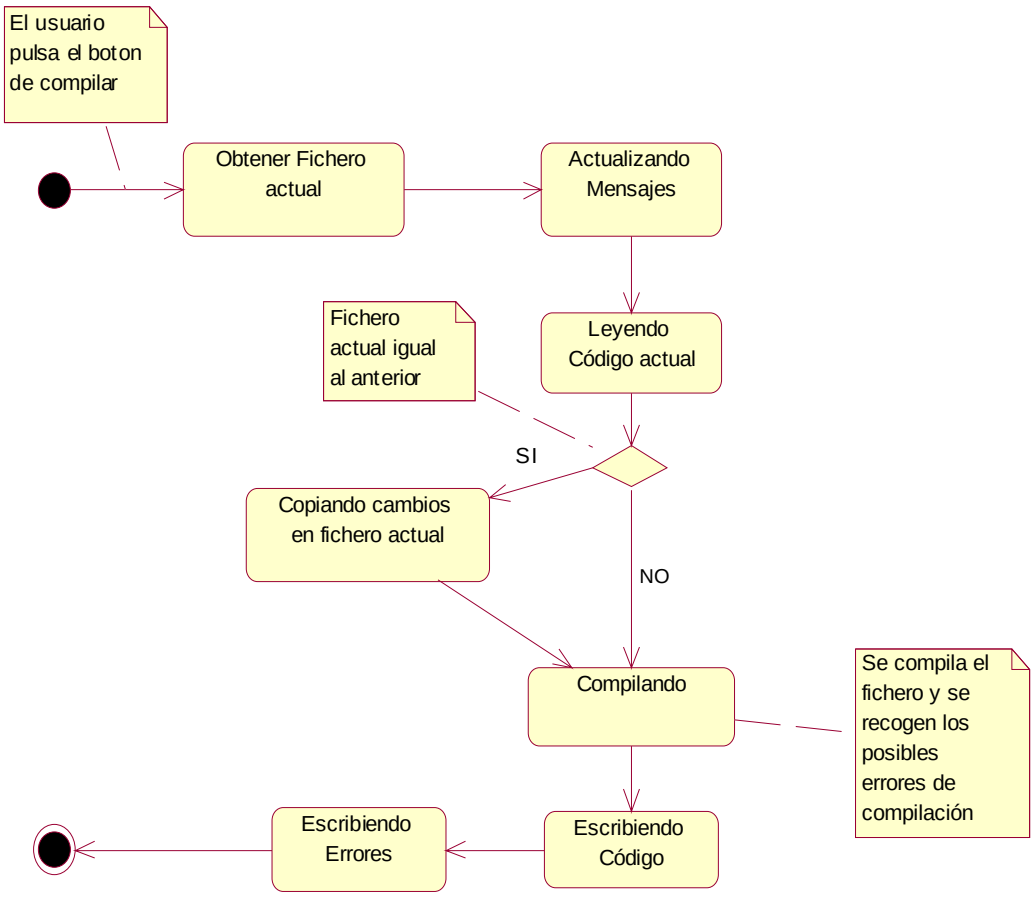


Figura 18. Diagrama de actividades del proceso de compilación de un archivo fuente

Entrada de un usuario a la aplicación. Tanto los Administradores como los usuarios estándar están identificados por su nombre de usuario y contraseña. El usuario deberá introducir su nombre de usuario y contraseña.

Gestión de usuarios. Se crean, borran, modifican o consultan las cuentas de los usuarios. Sólo los Administradores pueden realizar este tipo de operaciones.

Gestión de archivos. Un usuario puede abrir, enviar un archivo al directorio personal y extraerlo de ella. La carpeta personal es siempre el centro de estas operaciones. Por cada usuario se creará una carpeta personal, cada carpeta estará identificada por el nombre de usuario y contraseña.

Compilación. Permite tanto a un usuario estándar como a un administrador realizar una compilación de un archivo o conjunto de archivos del proyecto obteniendo un archivo objeto si la compilación fue correcta o un conjunto de errores si hubo problemas.

Gestión de errores. El sistema permitirá visualizar el error concreto de compilación y su localización dentro de un archivo de los que componen el proyecto; así mismo permitirá visualizar una página de ayuda para comprender las causas del error y abordar su solución. Por otro lado, el sistema almacenará en una base de datos la información sobre los errores encontrados.

Gestión de avisos. Envío de avisos a los usuarios elaborados por el sistema a partir del análisis de los errores previos del usuario almacenados por la gestión de errores. Serán de dos tipos: generales y personales. Contendrán información sobre los errores de compilación.

6.4 Actividad de compilación de un archivo

Describimos detalladamente el escenario de una compilación mediante un diagrama de actividad (Figura 18) para aclarar el proceso que debe realizar la aplicación.

El usuario selecciona el archivo que desee de su carpeta personal y elige la opción de compilar. Inicialmente se obtiene el archivo que el usuario desea compilar y se actualizan los mensajes de ayuda que se muestran al usuario con los datos de las anteriores compilaciones. A continuación se obtiene el código actual que se le está mostrando al usuario y una vez hecho esto se comprueba si el archivo actual, que es el que el usuario desea compilar, es el mismo que el usuario compiló la vez anterior (si es la primera compilación será nulo), en caso afirmativo se sobrescribe el archivo actual con el código actual que contiene las últimas modificaciones hechas por el usuario con lo que el archivo actual estará actualizado. En caso negativo no es necesario hacer ningún paso previo antes de compilar. Seguidamente se compila el archivo actual y se recogen los posibles errores de compilación y finalmente se muestra al usuario el código del archivo compilado y los errores de compilación que se han producido.

6.5 Diseño

6.5.1 Diseño de la arquitectura de la aplicación

Teniendo en cuenta los requisitos establecidos para el prototipo, se ha diseñado una arquitectura de Aplicación Web centrada en el servidor (Figura 19). Donde toda la lógica de compilación, gestión y almacenamiento de archivos del proyecto y gestión de errores y avisos se realiza en el servidor. Todo el espacio de almacenamiento de archivos residirá en el servidor haciendo que estos archivos estén disponibles independientemente del ordenador desde el que se conecte el usuario.

Proporcionaremos una interfaz Web que permite que el cliente esté compuesto, simplemente, por un navegador Web estándar que permitirá acceder a todas las opciones del entorno.

Con esta arquitectura se cumple el requisito de evitar la instalación de software en el ordenador del usuario final. Además, permite una configuración centralizada del sistema que permite que el administrador la realice para todo el sistema, evitando a los usuarios estándar preocuparse de estos temas. Proporciona a los usuarios independencia del lugar desde donde se conectan debido a que todos los recursos y archivos residen en el servidor.

Una tecnología que se adapta perfectamente a estas necesidades es la combinación de servlets / JSP basados en la plataforma Java de Sun Microsystems [Coward 2001] [Pelegrí-Llopart 2001] y es la que hemos empleado para la implementación.

6.5.2 Diseño de la navegación

La Figura 20 constituye un diagrama donde representa el mapa general de navegación y se muestran las posibilidades de navegación de los distintos usuarios a través del entorno. En el esquema se pueden ver las distintas partes de la aplicación, así como las relaciones entre las mismas. En dicho esquema, los rectángulos redondeados representan procesos que se realizan mediante servlets, los rectángulos representan páginas HTML.

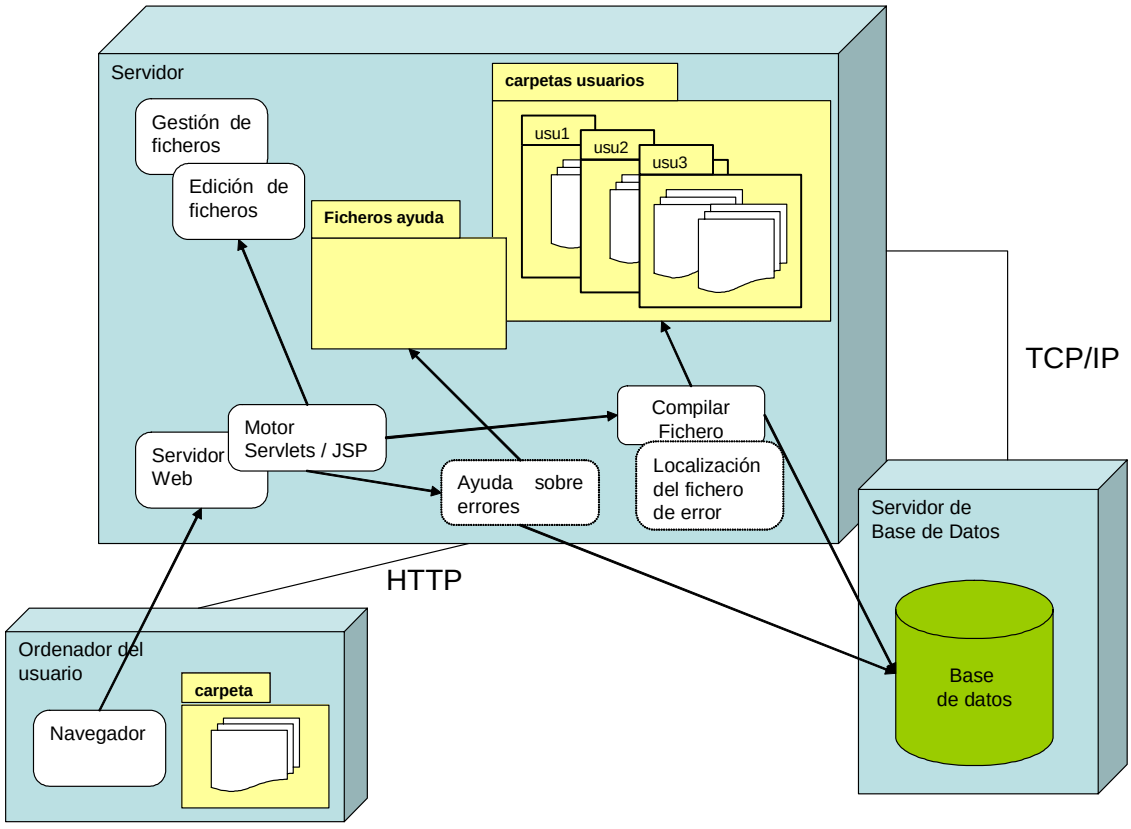


Figura 19. Arquitectura general del sistema

En el módulo de acceso tenemos una página HTML en la que el usuario introduce su nombre de usuario y su contraseña para poder utilizar la aplicación, dichos datos serán validados con los almacenados en la base de datos. En caso de no ser correctos se informará al usuario mediante la página No registrado ("NoRegistrado.html") de que no dispone de acceso, si son correctos, el usuario tendrá acceso a la aplicación.

Una vez iniciada la sesión y dependiendo de si se trata de un usuario estándar o de un administrador, se le mostrará la página principal de compilación, creándose dinámicamente (de ahí que tenga un contorno discontinuo) mediante un servlet; o la página opciones de administrador que es una página HTML normal.

Vamos a suponer que el acceso lo ha realizado un usuario estándar. La página principal reúne la funcionalidad de un entorno integrado sencillo, en la que el usuario podrá realizar la gestión de archivos, la edición y compilación de código; además de ser donde se muestran al usuario los avisos de prevención de errores. En dicha página, podrá abrir un archivo, crear un nuevo archivo, importar un archivo desde el equipo del usuario a su carpeta personal y exportar un archivo desde su carpeta personal a su equipo, en todo momento el usuario podrá cancelar la operación seleccionada, en cuyo caso volverá a mostrársele la página principal para que pueda seleccionar otra opción.

En el módulo de compilación, tenemos una primera fase en la que se seleccionan los mensajes de ayuda que se muestran al usuario en función del análisis de la historia de compilación (registro de los mensajes generados en todas las compilaciones precedentes); y una segunda fase en la que se lleva a la compilación propiamente dicha, recogiendo los posibles errores de compilación y actualizando los datos de la base de datos.

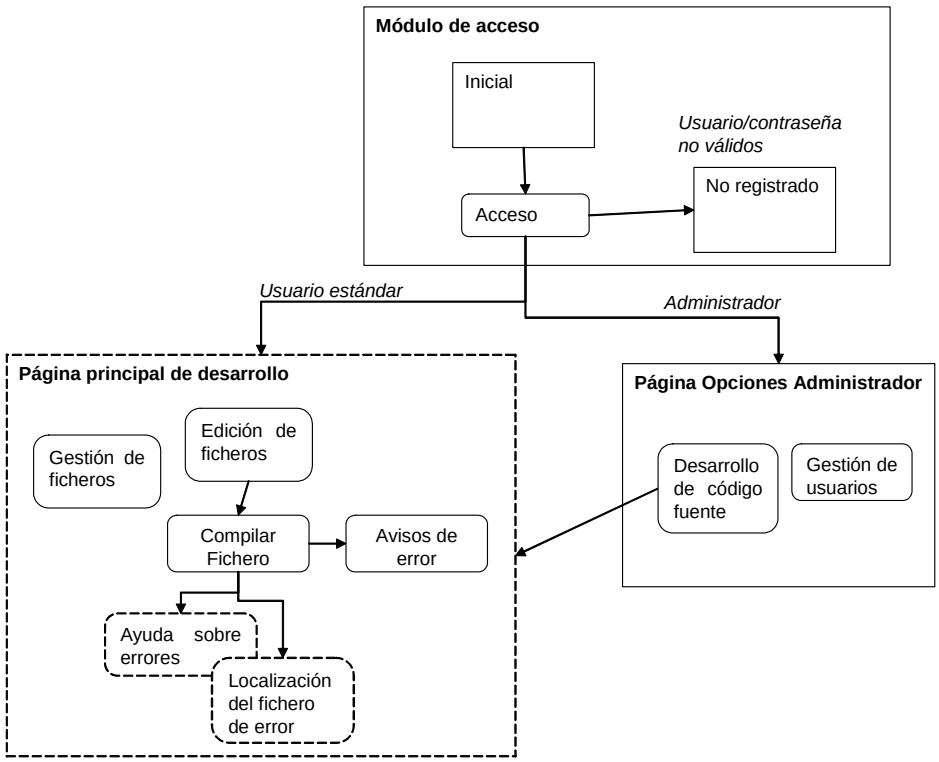


Figura 20. Diagrama de navegación de la interfaz del sistema

Además de lo anterior, en la página principal de compilación también intervienen dos módulos que sólo estarán activos si en la compilación se ha producido algún error de compilación (de ahí que tengan un contorno discontinuo) y que le permitirán al usuario: acceder a la ayuda sobre un determinado error, en cuyo caso extrae de una carpeta incluida en la aplicación el archivo HTML de ayuda; o abrir el archivo que contiene un determinado error de compilación si este se ha producido en un archivo distinto al actualmente activo.

Si el acceso lo realiza un administrador se le mostrará una página HTML donde le aparecerán las opciones de administrador relacionadas con la gestión de usuarios: dar de alta, dar de baja, realizar modificación y realizar consulta. Como ocurría en el caso de la gestión de archivos, una vez seleccionada una opción el administrador podrá cancelar la ejecución de la misma volviendo a la página de opciones del administrador. Además, contará en dicha pantalla con una opción que le permite utilizar la aplicación como si fuera un usuario estándar para poder realizar la compilación de sus archivos. Las opciones de administrador tendrán repercusión tanto en la base de datos como en las carpetas personales de los usuarios ya que ambas deben de reflejar los cambios realizados por el administrador.

6.5.3 Diseño de la interfaz de la página principal de desarrollo

En esta sección describiremos detalladamente la interfaz correspondiente a la página principal de la aplicación (Figura 21) que el usuario utilizará en el proceso de desarrollo. Se trata, de reunir en una misma página toda la funcionalidad e información necesaria para que el usuario desarrolle una aplicación; sin necesidad de navegar por la aplicación buscando diferentes opciones.

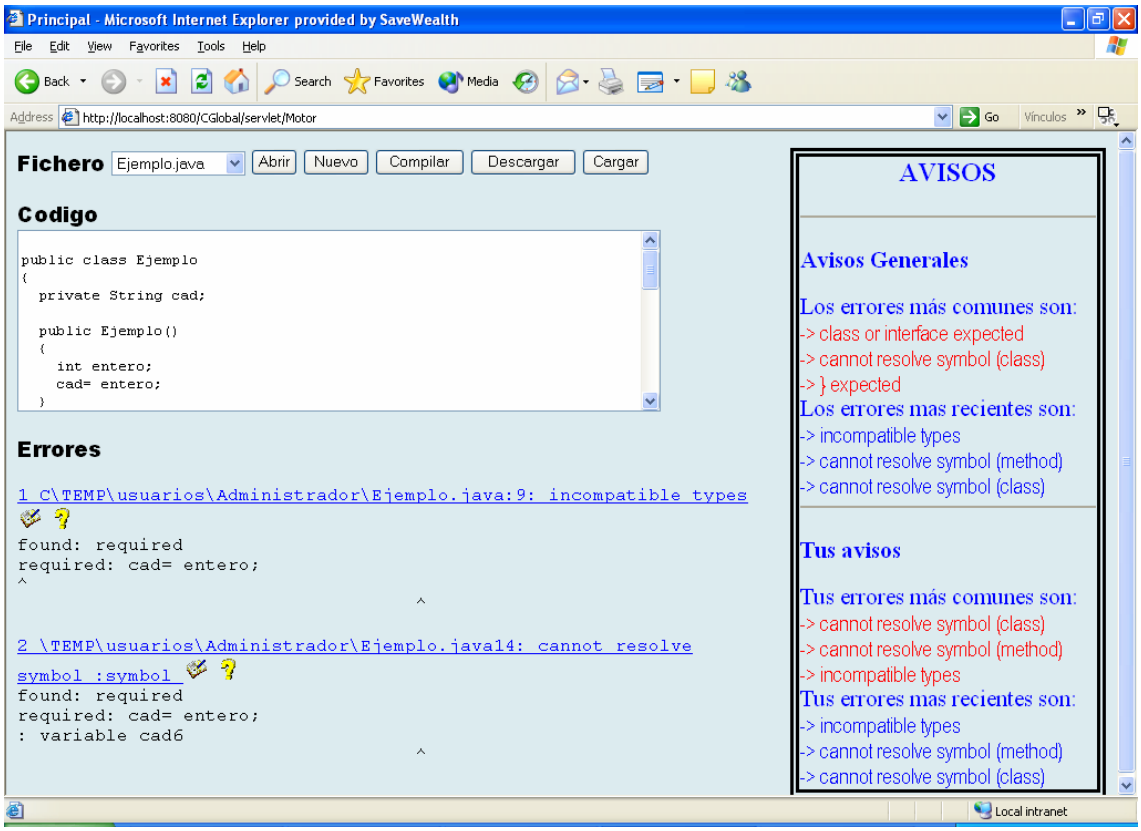


Figura 21. Pantalla del entorno de desarrollo en Web

En la pantalla principal destaca un **área de edición** etiquetada como "Código" donde el usuario puede escribir el código fuente de los ficheros y modificar los ficheros existentes.

En la pantalla principal de la aplicación el usuario podrá realizar las operaciones que se describirán a continuación y todas son accesibles a través del **menú de desarrollo** (Figura 22).

Abrir: Al pulsar sobre esta opción se abrirá el archivo actualmente activo en el campo "Fichero" y se mostrará al usuario el código de dicho archivo.

Nuevo: Al pulsar en esta opción se mostrará una pantalla en la que el usuario podrá introducir un nombre para un nuevo archivo que se creará en su carpeta personal.

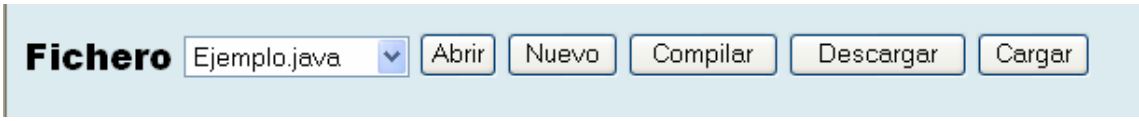


Figura 22. Detalle del menú de desarrollo del sistema

Compilar: Al pulsar sobre este botón se compilará el archivo previamente abierto con el botón Abrir. Si el archivo que se compila contiene errores de compilación, al pulsar Compilar aparecerá cada uno de dichos errores el en **área de notificación de errores** (Figura 23) e irán acompañados de dos iconos: al pulsar en el primero, abrirá el archivo que contiene ese error; con el segundo icono obtendrá una ayuda sobre ese tipo de error.

- Abrir el archivo que contiene el error: Al pinchar sobre el enlace que indica la ubicación del error o sobre el icono del libro (ver Figura 23) se abrirá el archivo que contiene el error para que el usuario pueda modificarlo.
- Solicitar ayuda: Al pinchar sobre el icono de la interrogación (ver Figura 23) se le mostrará al usuario una página con la ayuda correspondiente a ese tipo de error o bien se le indicará que no hay ayuda disponible para ese tipo de error.

Cargar: Al pulsar sobre este botón se mostrará una pantalla que permitirá al usuario seleccionar un archivo para que se copie de su equipo a su carpeta personal.

Descargar: Este botón se le mostrará al usuario una pantalla en la que aparecerá todos los archivos que haya actualmente en su carpeta personal y en la que podrá seleccionar aquel que desea copiar desde su carpeta personal a su equipo.

Figura 23. Detalle del área de notificación de errores de la página principal del sistema

Además, de las zonas ya descritas la parte derecha de la pantalla está reservada para el **área de avisos** (Figura 24). Aquí pretendemos ofrecer al usuario avisos cortos que le sirvan de

recordatorio para no volver a repetir los errores a los cuales es más propenso. Esta área está subdividida en dos partes con dos categorías cada una:

- Avisos generales. Avisos referidos a todo el grupo de usuarios de la aplicación; pero que pueden tener validez para este usuario individual. Se muestran los tres errores que con más frecuencia cometen los usuarios de la aplicación en su conjunto y los últimos errores cometidos, que al haber varios usuarios trabajando en paralelo, no tienen que coincidir con los del usuario actual.
- Tus avisos. Avisos referidos al desarrollador individual. Se muestran los tres errores más frecuentes en la historia de compilación y los tres más recientes para el usuario que está en sesión.

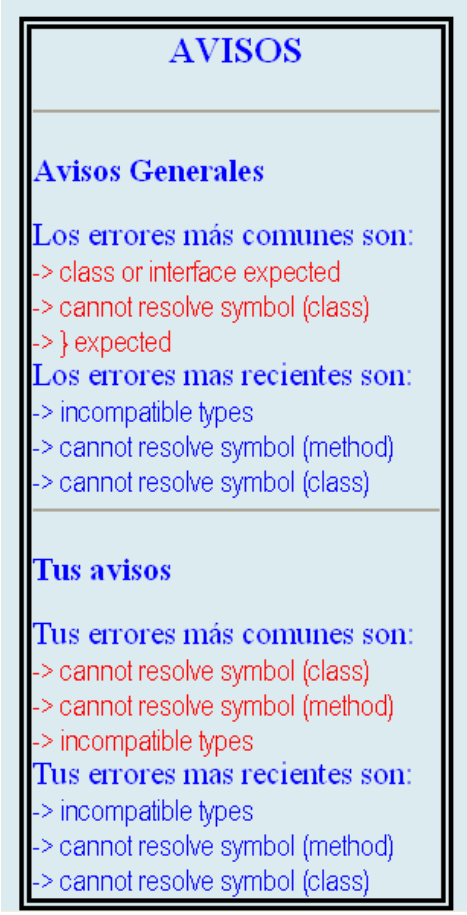


Figura 24. Área de avisos

6.5.4 Modelado de datos

Mostramos el diagrama entidad – relación (Figura 25) para mostrar la forma en la que se guardan la información sobre los errores, como se asocian a los usuarios y los datos auxiliares para realizar las estadísticas. En el diagrama se ven las entidades de información que guarda el entorno, se podrían clasificar estás entidades en tres grupos: información relacionada con los usuarios, información relacionada con los errores e información relacionada con el análisis de errores (estadísticas).

A continuación explicamos cada una de las entidades y relaciones que forman el diagrama:

Entidad: **usuarios** (Contiene el login, password y grupo de todos los usuarios de la aplicación). Se utiliza para validar los datos de acceso de los usuarios a la aplicación y en ella se reflejan las altas, bajas y modificaciones realizadas por los administradores.

Entidad: **claves_errores** (Contiene las palabras clave que identifican a los errores, el código que se corresponde con cada error y el mensaje que se corresponde con cada código). Se utiliza para analizar la cadena del mensaje de error que proporciona el compilador y clasificarlo según un tipo de error para poder contabilizarlo y almacenarlo así como para obtener los mensajes de ayuda para mostrarselos al usuario.

Entidad: **estadísticas_errores** (Contiene información acerca de los errores cometidos por un determinado usuario). Se utiliza para saber los errores más frecuentes y más recientes cometidos por cada usuario y nos indicará los códigos de los errores cuyos avisos debemos mostrar al usuario. Dichos avisos serán de tipo personal.

Entidad: **estadística_total** (Contiene información acerca de los errores cometidos por todos los usuarios de la aplicación). Se utiliza para saber los errores más frecuentes y más recientes cometidos por todos los usuarios y nos indicará los códigos de los errores cuyos mensajes debemos mostrar al usuario. Dichos mensajes serán de tipo general.

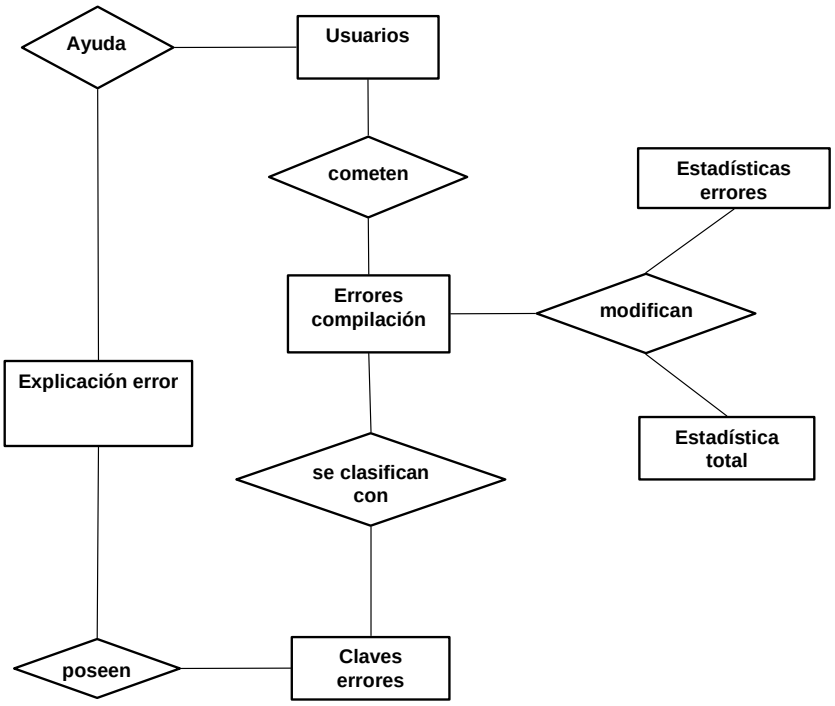


Figura 25. Diagrama entidad - relación de la base de datos del sistema

Entidad: **explicación_error** Sirve para obtener la dirección URL en la que se encuentra el archivo de ayuda correspondiente a un determinado error para poder mostrar dicha ayuda al usuario.

Entidad: **errores_compilación** (Almacena para cada usuario, la información de los mensajes de error generados por el javac al compilar). Se utiliza para almacenar los datos obtenidos de la salida del compilador javac y a partir de los cuales se actualizarán los datos de las entidades estadistica_errores y estadistica_total.

Relación: cometen (Asocia cada usuario con los errores de compilación que comete).

Relación: modifican (Asocia cada error cometido por el usuario con el número de veces y frecuencia con que se comete dicho error).

Relación: se_clasifican_con (Asocia cada error de compilación con el código y mensaje de error correspondiente).

Relación: poseen (Asocia cada código de error con el archivo HTML donde está la ayuda correspondiente a ese error).

Relación: ayuda (Asocia cada usuario con el archivo de ayuda que se le debe mostrar).

6.6 Descripción del prototipo

El prototipo trata de desarrollar una herramienta que facilite al usuario el aprendizaje activo de distintas técnicas de programación, utilizando el lenguaje Java [Arnold 1996], para ello le permite la visualización, modificación y compilación de sus propias aplicaciones realizadas en dicho lenguaje, y realiza una gestión de errores avanzada para ayudar al programador a no repetir errores.

6.6.1 Roles y apertura de una sesión en la aplicación

Se distinguen dos tipos de usuarios de la aplicación. Por un lado, los usuarios estándar que puede ser cualquier programador que quiera realizar un proyecto con la herramienta, y por otro lado están los administradores que dispondrán de toda la funcionalidad de los usuarios estándar y alguna extra como la gestión de cuentas de usuario, la modificación y ampliación de la ayuda y el seguimiento de los errores de un usuario. Al tratarse de una herramienta enfocada hacia el ámbito de la enseñanza de un lenguaje de programación los usuarios estándar pueden identificarse con los alumnos y los administradores con los profesores.

La forma de acceder a la aplicación será similar para los usuarios estándar y los administradores y se realiza mediante la introducción de un nombre de usuario y una contraseña que los identifica de forma única. La información que se les muestra y las tareas que pueden realizar unos y otros varían dependiendo del tipo de usuario que se trate. La comunicación entre los usuarios estándar y los administradores para lo relacionado con la solicitud de altas, bajas y modificaciones queda fuera del ámbito de este prototipo. La evaluación de los proyectos se lleva a cabo con herramientas externas, por ejemplo vía e-mail.

6.6.2 Gestión de archivos y espacio de almacenamiento

Por cada usuario, tanto si se trata de un administrador como si se trata de un usuario estándar, se crea una carpeta personal en el servidor, identificada por el nombre de usuario de los mismos, en la cual se almacenan sus archivos fuente Java con los que trabajan y los ".class" resultado de las compilaciones. Los usuarios pueden acceder a los archivos de su carpeta personal pero no a los de los demás usuarios. Esto ocurre, para este prototipo, incluso en el caso del Administrador, que sólo podrá acceder a sus archivos.

El usuario puede importar (cargar) archivos que se encuentren en su equipo local con lo que estos se copian en su carpeta personal para poder trabajar con ellos y de forma similar puede exportar los archivos de carpeta personal a su equipo local. Así mismo, se pueden crear nuevos archivos y manipular los ya existentes en la carpeta personal.

6.6.3 Gestión de cuentas de usuario

Los administradores llevan a cabo todo lo relacionado con la gestión de todos los usuarios, tanto los usuarios estándar como de los propios administradores, es decir, pueden dar de alta a un nuevo usuario, eliminar un usuario existente, modificar los datos de un usuario o consultar los datos de los usuarios actualmente registrados. Al dar de alta a un usuario se crea en el servidor una nueva carpeta personal para ese usuario. Al eliminar una cuenta de usuario se borra del servidor su carpeta personal y todo su contenido. Al modificar los datos de un usuario, si se modifica su nombre de usuario, su carpeta personal pasará a identificarse por el nuevo nombre de usuario.

Otra característica de los administradores es que también tienen la posibilidad de utilizar la aplicación como si fueran usuarios estándar, es decir, crear sus propios archivos para llevar a cabo la compilación, sin necesidad de tener otro nombre de usuario y contraseña de usuario estándar.

6.6.4 Compilación, gestión de avisos y ayuda a la corrección

En la parte de la aplicación dedicada a la compilación propiamente dicha, el usuario tiene la posibilidad de abrir un archivo de su carpeta personal y modificarlo o crear uno nuevo desde cero para luego compilarlo, es decir unas tareas básicas de cualquier entorno de compilación estándar. Podemos compilar un archivo de dos formas: si el archivo se abre previamente se compilará el código del archivo que es visible en ese momento en pantalla; si no hay un archivo abierto, se compilará el archivo que está actualmente seleccionado en el gestor de archivos.

Un usuario solicita la compilación de un archivo y el entorno lo compila mostrándole los errores de compilación que se produjeron, si hubo alguno. Además, estos errores se almacenan en una base de datos, clasificados por tipo de error y relacionándolos con el usuario que realizó la compilación del archivo, para posteriormente analizarlos y generar los avisos que se explican a continuación.

Otro punto importante, es lo relacionado con los avisos que se les muestran a los usuarios. Aparte de visualizar los mensajes de error que se producen al compilar un archivo tal cual los genera el compilador, se le muestran al usuario una serie de avisos de ayuda referidos a los errores de compilación y que serán de dos tipos:

- Avisos Generales: Son avisos acerca de los errores más comunes y los más recientes cometidos por todo el conjunto de usuarios de la aplicación.
- Avisos Personales: En este caso se trata de los errores más comunes y los más recientes cometidos por el usuario. Este tipo de avisos están totalmente personalizados para cada usuario.

Estos avisos están visibles permanentemente mientras el usuario se encuentre en la parte de la aplicación destinada a la edición y se renuevan cada vez que el usuario entra en la aplicación y cada vez que se realice la compilación de un archivo, haciendo de este modo que en todo momento el usuario tenga una información acerca de los errores lo más actualizada posible. A la hora de decidir que avisos se deben mostrar al usuario en cada momento, se utilizaran una serie de datos referidos a los errores de compilación previamente cometidos. Dichos datos se almacenaran en una base de datos y nos permitirán seleccionar los avisos correspondientes a los errores más frecuentes y numerosos.

Otro punto a destacar en la aplicación, es lo concerniente a la ayuda para depurar los errores de compilación. Para un subconjunto de todos los errores de compilación que se

pueden producir, se mostrará una ayuda en formato de hipertexto, lo más completa posible donde se indicarán las posibles causas del error, las posibles soluciones, y más datos que facilitará a los usuarios la tarea de corrección de código. Un usuario solicita visualizar la ayuda respecto a un determinado error de compilación y si esta existe, se le muestra al usuario. Esta ayuda es fácilmente modificable y ampliable por el administrador, ya que básicamente se compone de un conjunto de páginas HTML.

Relacionado con esta tarea de depuración de código, el usuario tendrá la posibilidad de, cuando se visualizan los errores de compilación que se han producido, acceder fácilmente al archivo en el que se ha producido un determinado error si este se produce en un archivo diferente al que actualmente se está mostrando al usuario.

6.6.5 Clasificación de los mensajes de error por tipos para facilitar la gestión de errores

Una de las tareas centrales que debe realizar el sistema es la de gestionar datos sobre los errores cometidos. Para llevar a cabo esta tarea, se almacenarán una serie de datos respecto a los errores cometidos por los usuarios y se elaborarán unas estadísticas para informar a los mismos, el problema está en: ¿Qué datos de los errores se almacenan? Dependiendo del tipo de error se almacenan datos asociados diferentes. Por tanto, es importante clasificar los errores para poder agruparlos y para saber que información tenemos que guardar sobre cada uno. El compilador *javac* puede indicar un número de tipos de error de compilación considerable, además el mensaje de error incorpora palabras correspondientes a los identificadores de los elementos relacionados con el error: nombres de archivos, clases, métodos, variables, etc. La posibilidad de que dos errores de compilación sean exactamente iguales es muy reducida; por tanto, la clasificación de los errores no es directa. Veamos un ejemplo:

Supongamos que al compilar un archivo llamado HolaMundo.java genere el siguiente error:

```
HolaMundo.java:7:cannot resolve symbol
symbol  : class Usuario
location: class HolaMundo
Usuario user = new Usuario ();
^
```

Y ahora supongamos que se compila un archivo llamado Helloword.java y que le produce el siguiente error:

```
Helloword.java:8:cannot resolve symbol
symbol  : class Admin
location: class Helloword
Admin ad = new    Admin ();
^
```

En este ejemplo, se ve claramente que se trata del mismo tipo de error pero los datos de ambos son distintos salvo ciertas palabras, y estas son en las que nos apoyaremos para llevar a cabo la comparación. Estas palabras se repiten en todos los errores que son del mismo tipo. En este caso, por ejemplo, estas palabras son, en la primera línea: "**cannot resolve symbol**", y en la segunda línea: **class**. Estas palabras "clave" son las que nos permitirán determinar que nos encontramos ante el mismo tipo de error. Por cada tipo de error se buscarían dichas palabras y se almacenarían de manera que ante cada error de compilación que se produzca, se realice una comparación con todo el conjunto de palabras clave almacenadas y si se produce una coincidencia de palabras "clave" entre las del error y las guardadas sabremos que se trata de un determinado tipo de error y por lo tanto podremos cuantificarlo para su posterior uso en la generación de estadísticas y avisos que sirvan de ayuda al usuario.

En este momento surge otra pregunta importante: ¿cómo obtener las palabras clave de todos los tipos de errores cuyo número, como se dijo, es muy considerable? Lo que se ha

hecho en este prototipo del Compilador Web SICODE es acotar el número de tipos de errores a estudiar y reducirlo a un conjunto de los más comunes, sobre los que se ha hecho un estudio lo más exhaustivo posible, tratando de proporcionar la mejor ayuda posible al usuario y con el resto lo que se hace es simplemente mostrar el error al usuario pero sin guardar ninguna información específica del mismo.

Una vez que se ha clasificado los errores esta información se utiliza para varias tareas dentro del proceso del procesamiento de errores (Figura 26):

- La clasificación del error permite saber que información se debe almacenar en la base de datos.
- Además, se utiliza en el análisis de errores para hacer métricas por tipos de error. De esta forma se generan avisos relacionados con los tipos de errores.
- Por último, se utiliza para enlazar de forma automática cada uno de los errores mostrados al usuario con la información de la base de conocimientos correspondiente a ese tipo de error.

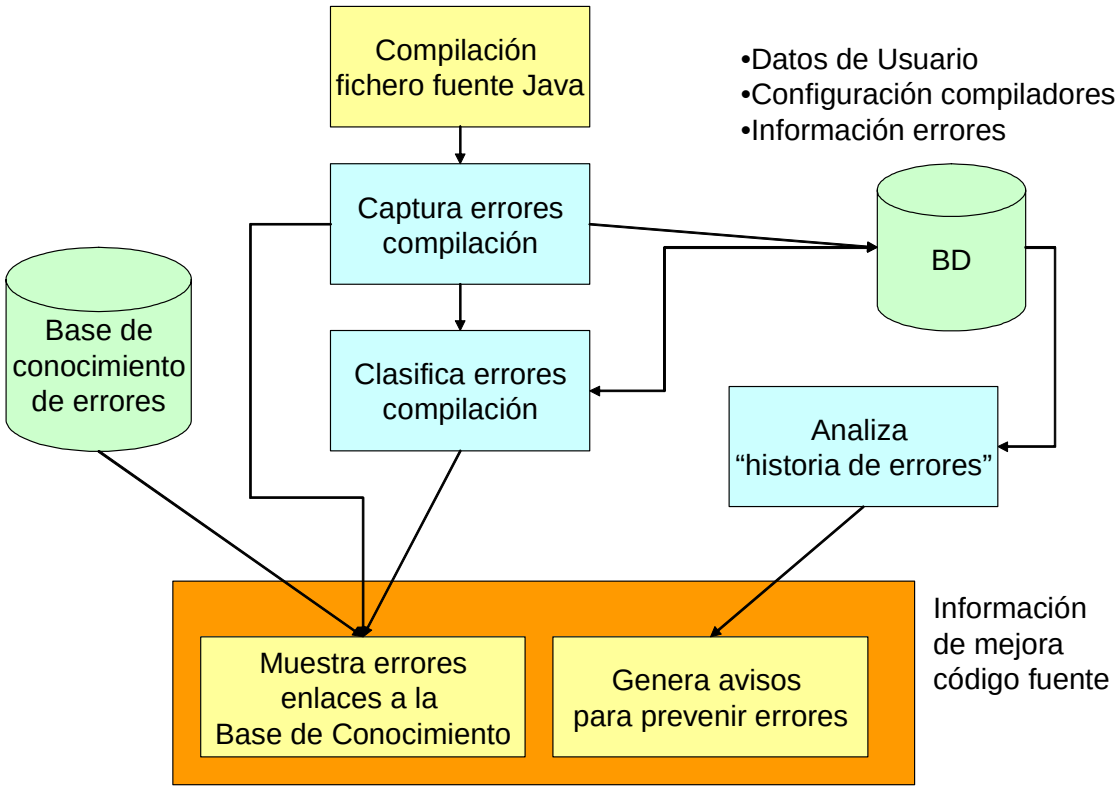


Figura 26. Modelo de la gestión de errores

6.7 Limitaciones del prototipo

El prototipo realizado tiene una serie de limitaciones en cuanto a los requisitos generales planteados en Capítulo 4 para un sistema de este tipo. Enumeraremos a continuación estas limitaciones que serán resueltas, en su mayor parte, por los siguientes prototipos desarrollados.

El análisis de código fuente se limita al que realiza un compilador, actualmente existen gran cantidad de herramientas que permiten un análisis de código fuente más exhaustivo y que proporcionará mayor información a los programadores; además, sería muy interesante poder combinar varias herramientas de este tipo para mejorar el análisis estático.

Sólo se captura información de errores en tiempo de compilación, no se proporciona ninguna forma de realizar las ejecuciones y capturar las excepciones en tiempo de ejecución generadas. Si un programador quiere realizar pruebas de ejecución de una clase (unitarias) o de un conjunto de ellas (de integración) para comprobar su buen funcionamiento, el único camino para realizarlo es descargando el archivo ".class" generado en las compilaciones; ya que el sistema no dispone de ninguna forma de realizar las ejecuciones en el servidor. Además, no tendríamos almacenadas las excepciones que puedan surgir en las pruebas.

Las métricas y estadísticas que se proporcionan sobre los errores únicamente se proporcionan como datos numéricos y solamente clasificadas por tipo de error, esto puede ser suficiente para generar avisos hacia un programador; pero para hacer una evaluación general o un diagnóstico sobre la calidad del código y su evolución estos datos son insuficientes, es necesaria una mayor flexibilidad y una presentación gráfica de los datos.

La lógica de clasificación de errores se encuentra inmersa en el código lo cuál dificulta la ampliación y extensión del sistema a otros compiladores y herramientas, es necesario disponer de un sistema de codificación y poder guardar en la base de datos las reglas para realizar este trabajo.

Desarrollo de aplicaciones en este entorno se realiza de forma individual. No hay herramientas para el apoyo al trabajo en grupo lo cual limita esta forma de trabajo.

La comunicación entre los usuarios estándar y los administradores para realizar operaciones de gestión queda fuera del ámbito de esta aplicación y se lleva a cabo externamente a ella, por ejemplo vía correo electrónico. Esto impide la simplificación de tareas específicas del desarrollo de aplicaciones, como puede ser la revisión de un archivo fuente, la toma de decisiones para solucionar un error en el código fuente, etc.

Modificación y ampliación de la ayuda se debe realizar manualmente directamente sobre los archivos HTML del servidor. Es necesario el planteamiento de un sistema colaborativo que facilite y motive a los usuarios a aportar información a la base de conocimientos común.

No es posible realizar un seguimiento del proceso de desarrollo del proyecto sino que sólo se puede examinar el resultado actual del trabajo. Para pequeños proyectos no hay problema en la revisión del estado final de este; pero para grandes proyectos es fundamental a la hora de analizar las causas de un error poder analizar los pasos en el proceso de desarrollo de un proyecto, por tanto necesitamos un sistema que soporte este requisito.

La revisión de los mensajes de error de los usuarios también debe realizarse manualmente sobre la base de datos que almacena todos los mensajes de error de cada usuario. Es necesario un sistema que permita un seguimiento cómodo del historial de errores.

6.8 Conclusiones sobre el prototipo de Compilador Web SICODE

Con este prototipo se han cumplido los dos objetivos fundamentales:

- Desarrollar un entorno que se pudiese ejecutar en cualquier ordenador sin necesidad de instalación y fácil de utilizar. Para lo cual se ha utilizado una

arquitectura Web y una interfaz de un entorno integrado de desarrollo sobre un navegador.

- Desarrollar un modelo centrado en los errores para la mejora de la calidad del código. Captura de errores de compilación, clasificación y almacenamiento en una base de datos, para su posterior análisis que permite una emisión de avisos preventivos que visualizará el entorno y por otro lado la base de conocimientos basada en página HTML, asociada a los distintos tipos de error.
- El entorno permite a los desarrolladores gestionar proyectos reales gracias a la posibilidad de configuración de distintos compiladores y bibliotecas. Además, el tamaño de los proyectos puede crecer ya que se dispone de un gestor de archivos flexible y también integrado en el entorno.

En los siguientes capítulos propondremos prototipos más evolucionados para cada uno de los subsistemas de este Compilador Global SICODE que nos permitan acercarnos más a los requisitos ideales especificados en el Capítulo 4.

Capítulo 7. Sistema de análisis de errores de programas: PBA

Como ya se comentó en el Capítulo 5 la política de desarrollo del sistema ha consistido en implementar un prototipo del Compilador Web SICODE con la funcionalidad básica para luego dividirlo en subsistemas relativamente independientes para poder profundizar más en la funcionalidad de cada uno de ellos. En este capítulo presentamos el Sistema de análisis de errores de programas (Program Bug Analysis) PBA, que se centra en la captura y procesamiento de errores del proceso de desarrollo de software para facilitar la mejora en la calidad del código fuente de los programas.

7.1 Objetivos del prototipo

La meta de este prototipo es obtener un sistema capaz de capturar, analizar y clasificar los errores de los desarrolladores a la hora de programar. Para conocer los errores, emplearemos como información base la proporcionada por el compilador e intentaremos ampliar el grado de detección de errores de los compiladores utilizando herramientas de análisis adicionales. Dichas herramientas, mediante análisis del código fuente o del código objeto generado, permitirán detectar errores que los compiladores no detectan. Toda la información obtenida se guardará, clasificada por el tipo de error y estará asociada al usuario que haya escrito el código correspondiente, para ser consultada y obtener información estadística a partir de ella.

No es objetivo de este prototipo el crear nuevas herramientas o técnicas para el análisis de código sino que se utilizarán las herramientas existentes vistas en el apartado 3.8.4. Herramientas análisis estático de código del Capítulo 3. Utilizando los resultados que generan, combinándolos y calculando métricas para presentarlas de forma gráfica.

El sistema será fácilmente configurable para ser capaz de trabajar con diferentes lenguajes de programación, aunque en nuestro prototipo nos centraremos en el análisis de código y de aplicaciones en el lenguaje Java. Se dejará la posibilidad del uso de diferentes compiladores y diferentes versiones para facilitar la actualización de los mismos. El sistema también estará preparado para utilizar herramientas de análisis de código distintas de las que hay configuradas.

También se le permitirá al usuario del sistema ejecutar el código compilado, siendo capaz de interactuar con las aplicaciones, pudiendo de esta manera hacer pruebas para detectar errores. En caso de que durante la ejecución se produzca una excepción, se dará al usuario información sobre cual es el error que puede haberla provocado.

El sistema dispondrá de un módulo que proporcionará información estadística al usuario sobre sus errores. Cada desarrollador podrá consultar información referida a los datos sobre sus errores, y a los datos del total de los desarrolladores.

A la hora del uso de la herramienta, habrá que diferenciar entre dos tipos de usuarios: los supervisores y los desarrolladores. El sistema de gestión de errores no será idéntico para todos los tipos de usuarios. La diferencia esta en la confección de las estadísticas, donde los usuarios supervisores, tendrán las mismas capacidades que los desarrolladores y además podrán usar los datos individuales de todos los usuarios del sistema para confeccionar estadísticas.

7.2 Ámbito del prototipo y relación con el resto de prototipos

Este prototipo se encarga exclusivamente del tratamiento de errores. Está concebido como un módulo que funcionará en segundo plano, sin interfaz de usuario, con la excepción de generación de páginas Web de resultados personalizados para cada usuario; serán otros módulos los que realicen llamadas a este, en función de las acciones del usuario.

El subsistema utiliza la información de usuarios para poder realizar métricas en relación a esta información; aunque no incluye ninguna gestión de grupos o usuarios que delega en otros subsistemas. Al carecer de interfaz con el usuario es otro el módulo que proporciona el entorno de edición de código y colaboración entre usuarios para el desarrollo de software. Este módulo trabaja con los ficheros fuente ya creados realizando la compilación y el procesamiento estático y una vez creados los ficheros objeto permite su ejecución y captura de excepciones.

7.3 Requisitos que cumple el prototipo

A continuación, se enumerar los requisitos de este prototipo, dividiéndolos en tres módulos.

7.3.1 Módulo de análisis estático

El módulo de análisis realiza el análisis estático de los ficheros fuente, captura de mensajes de error y almacenamiento los resultados del análisis del código que se realiza mediante un compilador y otros procesadores de lenguaje que realizan este análisis estático del código.

El sistema debe ser configurable:

- El sistema debe ser capaz de trabajar con diferentes lenguajes de programación.
- El sistema debe permitir el uso de diferentes compiladores y diferentes herramientas de detección de errores en un mismo análisis.
- Combinar la ejecución de distintas herramientas.

Se debe de permitir al usuario la elección de la forma de realizar los análisis: configuración de las herramientas que se van a utilizar, control de las dependencias que pueden tener unas respecto a otras y configurar las opciones de ejecución de cada una de las herramientas. Todo esto de una forma sencilla y homogénea para todas las herramientas, sin que tenga que conocer la forma específica en que se organizan las opciones necesarias en cada herramienta.

Se permitirá cambiar y añadir herramientas de una forma sencilla, con lo que lograremos que el sistema se adapte fácilmente a los cambios. Se permitirá la ampliación de las herramientas que son usadas en el sistema, pudiendo instalar nuevas herramientas de una forma sencilla.

Todos los errores o avisos detectados en un análisis serán guardados en una base de datos para su posterior uso creando un registro que denominaremos "historia de compilación".

Se utilizarán patrones búsqueda para clasificar los mensajes de error según el tipo al que pertenecen.

7.3.2 Módulo de análisis de la ejecución

El sistema debe permitir al usuario ejecutar las clases Java compiladas. Lo cual permitirá a los usuarios hacer pruebas unitarias para comprobar que su clase se ejecuta correctamente sin tener que exportar las clases fuera del sistema.

En caso de que se produzca una excepción durante la ejecución de la clase se capturará y se almacenará en la base de datos para poder realizar un seguimiento posterior de estas excepciones.

7.3.3 Módulo de elaboración de resultados

El módulo de elaboración de resultados realiza estadísticas sobre los errores almacenados por el sistema en el registro ("historia de compilación"). Los resultados se presentarán tanto de forma numérica como gráfica.

Se platearán varios tipos de estadísticas que tienen utilidad para los supervisores y los desarrolladores que usen la herramienta, y hacer que estén disponibles para los usuarios. Se debe hacer que las estadísticas sean fáciles de interpretar, para ello se pueden emplear gráficas que reflejen de una forma clara los resultados obtenidos en las estadísticas.

El sistema de estadísticas debe de ser fácilmente ampliable y configurable.

A la hora de realizar las estadísticas, se tendrá en cuenta el tipo de usuario. Si el usuario es un desarrollador sólo podrá usar sus datos y los del conjunto de los usuarios. Mientras que un supervisor además podrá tener acceso a los datos de cualquier usuario de forma individual.

7.4 Descripción de los actores y casos de uso

7.4.1 Actor Desarrollador

El actor Desarrollador representa a un usuario que realiza todo el ciclo de desarrollo de aplicaciones. Las acciones que tendrá a su disposición serán las de compilar un archivo, ejecutar sus aplicaciones. Podrá realizar estadísticas y gráficas sobre sus errores, los errores de su grupo de trabajo y los que tuvieron todos los usuarios del grupo desarrolladores colectivamente, pero no tendrá acceso a las del resto de desarrolladores individualmente, ni a la de los supervisores.

7.4.2 Actor Supervisor

El actor Supervisor representa a un supervisor del trabajo de los desarrolladores. Tendrá a su disposición todas las posibilidades de un usuario desarrollador, y además tendrán acceso a todas las estadísticas sobre los errores individuales de todos los usuarios del sistema.

7.4.3 Análisis estático

Este módulo permite realizar una exploración del código fuente para verificar su corrección. Utiliza el compilador y otros procesadores de lenguaje para realizar el análisis estático del código fuente.

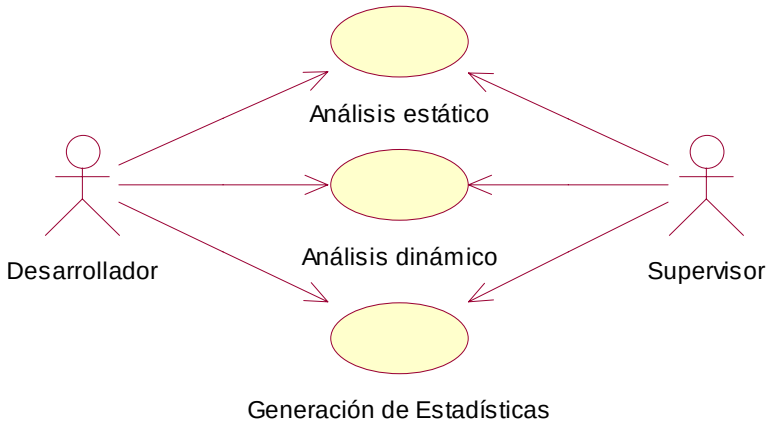


Figura 27. Diagrama de casos de uso del prototipo de análisis de errores de programas (PBA)

Los análisis son lanzados por los usuarios del sistema. Es técnicamente posible lanzar este análisis desde línea de comandos; pero lo normal es que sea otro subsistema del sistema SICODE el que lo haga. Como entrada para la ejecución de este módulo se deberá especificar:

- La unidad de compilación con la que se quiere compilar los archivos.
- La ruta donde están los archivos fuente del proyecto que se desea analizar.

Una unidad de compilación, permite indicar al sistema los directorios donde se encuentran los archivos fuente y donde tiene que dejar los resultados; las herramientas que debe utilizar el sistema, las opciones de estas herramientas y en que orden debe utilizarlas; incluso permite condicionar el uso de determinadas herramientas a que otras proporcionen determinados resultados. Entre las herramientas utilizadas siempre se incluye un compilador; de ahí el nombre de unidad de compilación. Se pueden crear tantas unidades de compilación como se crean convenientes y cada una tiene un identificador único con lo que se pueden utilizar como bloque que unifica todo el trabajo que debe realizar el sistema sobre el conjunto de archivos de un proyecto.

En caso de que durante el proceso de análisis se detecte errores en el código fuente, se guardará toda la información sobre el mismo en la BD en el registro de "historia de compilación", para que se pueda realizar un estudio sobre ella y mostrar los resultados al usuario a través de una interfaz.

7.4.4 Análisis dinámico

Un usuario lanza la ejecución de una clase. Se permitirá al usuario interactuar con ella haciendo llamadas a los métodos con distintos valores en sus parámetros, en caso de producirse una excepción se guardará toda la información sobre esta excepción en la BD. El análisis dinámico está pensado para hacer pruebas de ejecución de las clases sin tener que extraer estas clases del sistema.

7.4.5 Generación de estadísticas

Esta operación será lanzada por los usuarios del sistema. El usuario deberá indicar un archivo de configuración de estadísticas, donde se indicará: los datos que se deben analizar, la forma de confeccionarlas, y el archivo donde se guardará el resultado de las mismas.

Si el usuario que realiza la operación es un desarrollador, podrá usar para la confección de las estadísticas sus datos y los datos de todos los usuarios tomados en conjunto. Si el usuario es un supervisor, además podrá usar para las estadísticas los datos de todos los usuarios del sistema tomados individualmente.

7.5 Escenarios del prototipo

7.5.1 Análisis estático

El usuario ordena el análisis de los archivos fuente de un proyecto concreto con una unidad de compilación determinada. En esa unidad de compilación, además del compilador se podrán configurar herramientas para la comprobación del código fuente, que se puede usar para indagar posibles errores en la aplicación no detectados por el compilador. El resultado del análisis será uno de los siguientes:

1. Los parámetros del análisis son erróneos. Se indicará al usuario que error hay en los parámetros y además se le mostrará la forma de uso correcta.
2. Al realizar el proceso de compilación y ejecución de las herramientas de comprobación de código se produce un error de ejecución por una configuración errónea. Se mostrará al usuario que se ha producido un error durante este proceso, así como el mensaje de la excepción.
3. El proyecto compila correctamente y no tiene ningún aviso. La compilación finaliza correctamente y las herramientas adicionales no encuentran ningún error, por lo que se le indicará al usuario que no hay errores de compilación y que no se produjo ningún aviso.
4. El proyecto compila correctamente, pero tiene avisos. Se muestra un mensaje de compilación correcta, pero se han detectado algún posible error en la aplicación, y se da un aviso sobre él. Se le ofrecerá al usuario un enlace a la información sobre ese aviso.
5. Algún archivo del proyecto no compila. Se indica al usuario que el archivo tiene errores de compilación, y se indican los errores detectados, así como un enlace a información sobre los mismos.

Para almacenar información sobre el análisis y poder consultarla y procesarla posteriormente a cada análisis se la asignará un código de identificación. El código del análisis se hará corresponder con el orden de los análisis para ese usuario, es decir, el primer análisis de un usuario deberá ser el análisis 0 y de ahí en adelante se les dará de código a cada uno de forma correlativa.

7.5.2 Análisis dinámico

Un usuario podrá lanzar la ejecución de sus aplicaciones. El sistema se ocupará de capturar la salida de la aplicación y de los errores que se pudieran provocar durante la ejecución.

1. Los parámetros de la ejecución son erróneos. Se indicará al usuario que error hay en los parámetros y además se le mostrará la forma de uso correcta.

2. La clase se ejecuta sin lanzar excepciones. En caso de que no haya ninguna excepción se mostrará la salida de la aplicación, así como un mensaje indicando que la ejecución no produjo excepciones.
3. La clase finaliza con una excepción. Se mostrará la salida de la ejecución de la aplicación hasta el momento en el que se produjo la excepción. Además se mostrará un mensaje con la excepción producida y un enlace a la información sobre dicha excepción.

7.5.3 Generación de estadísticas

El usuario deberá indicar un archivo de configuración de estadísticas, donde se indicará la forma de confeccionarlas, y el archivo donde se guardará el resultado de las mismas. A partir de esta información y los datos de errores y avisos almacenados en la base de datos el módulo genera informes con información numérica y gráfica dependiendo de la configuración elegida.

Escenarios según el tipo de estadísticas que queremos generar:

1. Generación de una estadística de Media de Errores en un Periodo. En el archivo de configuración se manda generar unas estadísticas sobre la media de errores de algunos usuarios durante un periodo dado. Los datos requeridos por el usuario son accesibles para él y se puede generar las estadísticas que se presentarán en la página Web resultante.
2. Generación de una estadística de evolución de la Media de Errores. En el archivo de configuración se manda generar unas estadísticas sobre la evolución de la media de errores. Los datos requeridos por el usuario son accesibles para él y se puede generar las estadísticas que se presentarán en la página Web resultante.
3. Generación de una estadística de Errores más Frecuentes en un Periodo. En el archivo de configuración se manda generar unas estadísticas sobre los errores más frecuentes de algunos usuarios durante un periodo dado. Los datos requeridos por el usuario son accesibles para él y se puede generar las estadísticas que se presentarán en la página Web resultante.

En la generación de estadísticas se pueden dar los siguientes escenarios de error:

4. Los parámetros de la estadística son erróneos. Se indicará al usuario que error hay en los parámetros y además se mostrará al usuario la forma de uso correcta.
5. El archivo de configuración contiene errores. Se indicará al usuario que en el archivo se han detectado errores en el archivo de configuración, indicando la línea donde se produce el error y el tipo de error encontrado.
6. Un desarrollador intenta acceder a la información individualizada de otros usuarios del sistema. En caso de que un desarrollador intente obtener información de otros desarrolladores de forma individual, se parará la ejecución del archivo de configuración y se avisará de que se está intentado acceder a una información para la cual no se tienen permisos.

7.6 Herramientas de búsqueda de errores utilizadas en el proyecto

En el apartado 3.8.4. Herramientas análisis estático de código del Capítulo 3 se ha realizado un análisis detallado de las herramientas de análisis de código fuente más utilizadas en la

actualidad. En el desarrollo de este prototipo hemos seleccionado algunas de estas herramientas para su integración en el sistema. A continuación enumeramos estas herramientas y las razones por las que las hemos elegido:

- **Jlint** [Knizhnik 2003]. Los avisos que generados por esta herramienta son interesantes, permiten la detección de algunos errores difíciles de encontrar. Está implementada en C++, lo cual hace que sea muy eficiente respecto las que están implementadas en Java. Su código fuente es compilable en Linux y Windows, lo que hace que pueda ser usada en varias plataformas.
- **Antic** [Knizhnik 2003]. Los errores detectados por esta herramienta son errores muy simples que se suelen presentar sobre todo en las primeras fases de aprendizaje de la programación. Al igual que la anterior es muy eficiente puesto que está implementada en C, pero puede ser compilada bajo Windows y Linux. Además de Java, puede analizar código C y C++.
- **Findbugs** [Hovemeyer 2004a]. Esta herramienta usa el concepto de Patrón de Error, lo que la hace interesante desde el punto de vista teórico. Comprueba la existencia de los patrones en el código fuente. Algunos de estos patrones no son aplicables en el ámbito del aprendizaje de la programación, pero permite que estos no sean usados en el análisis, seleccionando los que sean de interés.
- **PMD** [PMD]. Esta herramienta también permite definir patrones de error para su detección y comparte algunos de los tipos de análisis que realizan FindBugs y Jlint, pero proporciona muchos tipos de avisos que no son tratados por las otras como avisos sobre estilo o sobre código innecesario o ineficiente.

En el estudio aparecían más herramientas que finalmente hemos descartado, exponemos a continuación las razones para descartarlas:

- **Jwiz** [Geis 1999]. Fue descartada debido a que proporciona una documentación pobre.
- **DoctorJ** [DoctorJ]. Descartada al no tener ningún tipo de análisis distinto de los que hace Jlint. Además está implementada en C++ y sólo dispone de una versión para Linux.
- **Checkstyle** [Checkstyle] y **JCSC** [JCSC]. Estas dos herramientas tienen el mismo sistema de funcionamiento que PMD, y en muchos casos realizan el mismo tipo de análisis de código. La elección de PMD frente a las otras dos se debe a que PMD se centra más en generar avisos sobre posibles errores, mientras que Checkstyle y JCSC se centran más en comprobar que el documento sigue un buen estilo.

7.7 Estadísticas de análisis de errores

Es muy importante disponer de una gran flexibilidad a la hora de generar estadísticas; por ello, los usuarios dispondrán de la posibilidad de configurar una serie de estadísticas sobre sus errores de compilación o avisos detectados por el sistema al realizar el análisis estático. Todas las métricas y estadísticas se podrán elaborar para un usuario individual, o para todos los usuarios del sistema conjuntamente. En el caso de desarrolladores el usuario individual será el propio usuario; pero en el caso de los supervisores podrán seleccionar el usuario sobre el cual elaborarlas.

7.7.1 Consideraciones generales para las estadísticas

Para realizar las estadísticas se necesita un periodo que delimite los datos que van a ser analizados. En el sistema se permitirán dos formas de indicar el periodo. Por un lado se podrá especificar de forma temporal, y por otro se podrá indicar por el código de análisis. Por ejemplo se permitirá expresar los periodos de las siguientes formas:

- 4 de Agosto de 2003 a 16 de Agosto de 2003
- Análisis 23 hasta el análisis 65.

Además se permitirá usar periodos temporales relativos, es decir, que no se basen en una fecha en concreto. Por ejemplo, se podrá indicar que el periodo es:

- Desde hace quince días hasta hoy.

En cuanto a las medidas de los periodos, para los periodos temporales se usará el día como unidad de medida, mientras que para los análisis se usarán el número de estas.

Se podrán crear conjuntos de errores en los cuales se podrá indicar los errores que pertenecen o no pertenecen por medio de su código, su tipo o la herramienta que los genera. Por ejemplo, se permitirá expresar los siguientes conjuntos de errores:

- Errores de Compilación.
- Error 122, Error 345 y Error 346 (por el código de error).
- Avisos de sincronismo que no han sido generados por Findbugs (por el nombre de la herramienta).
- Avisos de PMD menos el error 12330 (todos excepto el error del código dado).
- Todos los avisos que no han sido generados por Antic (excepciones por herramienta).

De la misma forma, se podrá especificar que se usen sólo los datos de unas determinadas unidades de compilación para generar las métricas.

7.7.2 Tipos de estadísticas disponibles

Para crear los tipos de estadísticas hemos intentado ante todo que tengan utilidad para los desarrolladores y los supervisores de forma que se puedan usar para prevenir errores típicos y poder conocer la evolución de los desarrolladores a lo largo de un periodo.

Cada una de las estadísticas proporcionadas tendrá una gráfica asociada que simplifica la lectura de los datos. Estas gráficas deberán tener una leyenda para mejorar su comprensión por parte de los usuarios.

Errores más frecuentes durante un periodo

Esta métrica nos permite conocer cuales han sidos los errores más frecuentes de un usuario en un determinado periodo. Los resultados se proporcionarán en errores por análisis. Desde el punto de vista de los desarrolladores puede servir de recordatorio de sus fallos más comunes de forma que puedan prevenirlos mediante el estudio de sus soluciones. Desde el punto de vista de los supervisores puede servir para tener una visión general de los errores más frecuentes de los diferentes desarrolladores y de todo el sistema en general. Con esta información puede preparar mejor las clases futuras sabiendo de antemano donde están los mayores problemas.

Los parámetros de este tipo de estadística serán:

- El número de errores más frecuentes. Deberá ser un entero entre 1 y 10. Por defecto 5.
- El periodo que se va a analizar.
- Los usuarios sobre los que se va a realizar.
- Los conjuntos de errores que van a intervenir.
- El periodo de los análisis que se van a usar.

Ejemplos:

- 5 errores de compilación más frecuentes durante la semana pasada.
- 10 avisos de Antic más frecuentes entre los análisis 340 y la 360.
- El aviso de Jlint más frecuente de todos los análisis.
- 6 errores de g++ más frecuentes de las compilaciones de la unidad de compilación "compila_cpp".

La gráfica asociada a esta métrica será una gráfica de barras donde se represente el nivel de errores por análisis de cada error. En el informe de ejemplo del apartado 7.9.2 la cuarta estadística es de este tipo, que se corresponde con los 8 errores más frecuentes de cualquier tipo entre los análisis 0 y 10. En la Figura 44 se representa la gráfica de la estadística y en la Figura 45 la tabla con los datos numéricos.

Media de errores en un periodo

Se calculará para un conjunto de errores la media de errores por análisis en un determinado periodo. Esta métrica permitirá la comparación de los niveles de error de los usuarios en el sistema. Además permite conocer cuales son las herramientas más útiles o los conjuntos de errores que más nivel de error tienen.

Los parámetros de este tipo de estadística serán:

- El periodo donde se va a realizar.
- Los usuarios sobre el que se va a realizar.
- Los conjuntos de errores que van a intervenir.
- El periodo de los análisis que se van a usar.

Ejemplos:

- Media de errores de compilación en el mes pasado.
- Media de avisos de diseño de FindBugs desde el análisis 200 hasta el análisis 220.
- Media de avisos de PMD del 10 al 13 de Junio.

La gráfica asociada a la media de errores en un periodo será una gráfica de barras donde se represente el porcentaje de errores por análisis de cada conjunto de error. En el informe de ejemplo del apartado 7.9.2 la tercera estadística es de este tipo, representa la frecuencia para dos conjuntos de errores: todos los errores excepto los de la herramienta JLint, todos los errores que sean de JLint o sean del tipo de sincronización; el periodo de análisis empieza el 10 de septiembre de 2004 y finaliza el 17 de septiembre de 2004; además se calcula para poder compararlo los errores para un usuario individual y para todos los usuarios. En la

Figura 41 aparece la representación gráfica y en la Figura 42 aparece la tabla con los datos numéricos.

Evolución de media de errores

Básicamente es la misma idea que la métrica anterior, sólo que tomada en varios periodos consecutivos. Por lo tanto permite tener una visión de como se comporta la media de errores en el tiempo (ya sea medio en unidades temporales o en el número de análisis).

Los parámetros de este tipo de estadística serán:

- Número de periodos en los que se va a realizar.
- Tamaño del periodo.
- Los usuarios sobre los que se va a realizar.
- Los conjuntos de errores que van a intervenir.
- El periodo de los análisis que se van a usar.

Ejemplos:

- Media de errores de compilación de cada mes en los últimos 5 meses.
- Media de avisos de sincronismo cada 20 análisis desde el análisis 350.
- Media de avisos de Jlint en esta semana y en las 10 semanas anteriores.

La gráfica asociada será una gráfica de líneas donde cada punto represente un periodo. Estos puntos estarán unidos por líneas. En el informe de ejemplo del apartado 7.9.2 la segunda estadística es de este tipo, representa la evolución de la media de errores para un conjunto de errores: todos los errores de tipo error de compilación; el periodo total del análisis empieza el 26 de agosto de 2004 y finaliza el 9 de septiembre de 2004, dividiendo todo en subperiodos de 200 análisis. En la Figura 38 aparece la representación gráfica y en la Figura 39 aparece la tabla con los datos numéricos.

7.8 Diseño

7.8.1 Definición de la secuencia de utilización de las herramientas de análisis

Uno de los requisitos fundamentales de este sistema es la organización de la ejecución del compilador y de las herramientas de análisis estático a la hora de realizar un análisis. Normalmente deberíamos poder configurar varias herramientas de análisis y permitir configurar un número indeterminado de ellas. Además, debemos definir una secuencia de ejecución de estas herramientas, ya que algunas utilizan los resultados de otras. Por otra parte, es necesario definir las opciones de ejecución de cada una de las herramientas. Además, es necesario gestionar los resultados situándolos en un directorio temporal. Todo esto respetando la portabilidad del sistema para que pueda ser ejecutado, tanto en máquinas finales de los usuarios como en servidores. Por último, es necesario controlar que no haya problemas en el proceso y si los hay tener un mecanismo para notificarlos.

Para solucionar este problema se pensó en un primer momento en guardar la información sobre como organizar los análisis en la Base de Datos del sistema. Frente a esta solución se pensó en el uso de Ant [Ant 2003]. Como pudimos comprobar, Ant es una herramienta de

gran utilidad a la hora de organizar la ejecución de las tareas. Al final, se uso Ant, puesto que se adaptaba mucho mejor a los requisitos del proceso que debemos de realizar.

La herramienta Ant [Ant 2003] es una herramienta de software libre desarrollada por Apache Software Foundation. Esta es una herramienta del tipo de "make" cuya función es permitir la definición de una secuencia de ejecución de comandos necesarios en la compilación, despliegue, prueba y distribución de una aplicación. Pero que supere parte de los inconvenientes que tiene el "make" original, como la falta de portabilidad y la fuerte dependencia del sistema operativo. Esta completamente implementada en Java, por lo que puede ser usada en cualquier plataforma. Para especificar los objetivos y el orden de los mismos se usa un archivo de construcción codificado en XML.

Cada archivo de construcción define la forma de tratar el proyecto de una aplicación, que está compuesto por objetivos (targets), los cuales pueden depender unos de otros. En cada uno de estos objetivos se realizan tareas (tasks), las cuales están implementadas por una clase Java. Se dispone de muchas tareas predefinidas, relacionadas con las principales tareas de trabajo con archivos y compilación en el proceso de desarrollo. También se pueden establecer propiedades dentro del archivo de construcción, pasárselas como parámetros mediante la línea de comandos, y usar las propiedades de sistema.

Ant dispone de varias características que lo hacen muy versátil:

- Gran cantidad de tareas (Task) predefinidas.
- Fácilmente ampliable. La creación de nuevas tareas y la extensión de los existentes es algo tan sencillo como la programación de una clase Java.
- Uso desde Java. Las tareas de Ant pueden ser usadas como una clase más desde nuestro código Java, permitiendo la reutilización del código existente.
- Buena documentación. Ant dispone de una buena documentación, lo que facilita tremendamente su uso.

7.8.2 Empleo de la herramienta Ant en el análisis estático

La herramienta Ant es una de las piezas clave en el sistema de análisis, puesto que se encarga de todo lo referente a la ejecución de distintas herramientas especificadas en las unidades de compilación.

En un primer momento el sistema de análisis ejecutaba Ant desde la línea de comandos mediante la clase Runtime, esto presentaba un serio problema de comunicación entre las clases, puesto que en caso de producirse un error en la ejecución de Ant, no se tenía acceso a la información sobre el error, es más, ni siquiera se tenía constancia de que se hubiese producido tal error. Tras estudiar el código fuente y la documentación de Ant, se llegó a la conclusión de que se podía ejecutar directamente un archivo Ant desde el código Java del proyecto usando la tarea "Ant". Por lo tanto, la solución a este problema fue crear en la clase Compilacion que agrega clases del proyecto de Ant y permitir ejecutar la tarea "Ant" con los parámetros adecuados para realizar la ejecución del script y acceder a todos los errores que se produzcan en la ejecución de Ant.

Algunas de las herramientas utilizadas proporcionan una tarea Ant. Viendo que la herramienta Ant ofrecía muchas posibilidades, se buscó la forma poder ejecutar también, las aplicaciones que no tenían interfaz Ant. En un primer momento, se usó para ello la tarea "Exec" de Ant, la cual permite la ejecución de comandos. Sin embargo, se pensó que era mucho más elegante el crear unas tareas para estas herramientas, lo cual simplifica mucho la edición de los archivos Ant de las Unidades de Compilación. Por ello, se estudió la forma en que se pueden crear tareas Ant. En ese momento se crearon las clases

AnticTask, JlintTask y AnalizadorTask. Esta última hereda de la clase Task, mientras que las dos primeras heredan de la clase MatchingTask. Se eligió esa clase porque está pensada para que hereden de ellas las clases que quieran trabajar con paths y conjuntos de archivos.

El último problema que se tuvo que resolver en el Ant fue que la tarea predefinida "Javac" no ofrece la posibilidad de redirigir la salida del compilador hacia un archivo. Esto hace que no sea posible el manejo de los errores detectados por el compilador. La solución que se dio a este problema fue la creación de una nueva tarea, JavacTask, que permitiera realizar las compilaciones y además guardar la salida a un archivo. Para construir esta clase se emplearon los conocimientos adquiridos en la implementación de AnticTask y JlintTask.

Diagrama de clases de la Clase Compilación

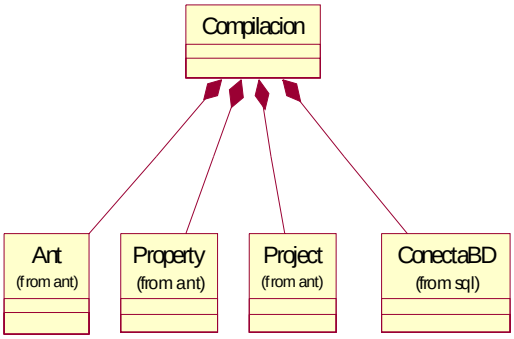


Figura 28. Diagrama de clases de las relaciones de la clase compilación

La clase Compilación (Figura 28) se encargará de realizar los análisis en el sistema. Para ello usa clases ajenas procedentes de la herramienta Ant, las cuales son usadas para la ejecución del script Ant con el que se configura la Unidad de Compilación que se le indicó a la clase Compilacion. La clase ConectaBD es una clase de nuestro proyecto la cual se usa para la comunicación con la Base de Datos.

Diagrama de clases del paquete proyecto.utilidad.ant

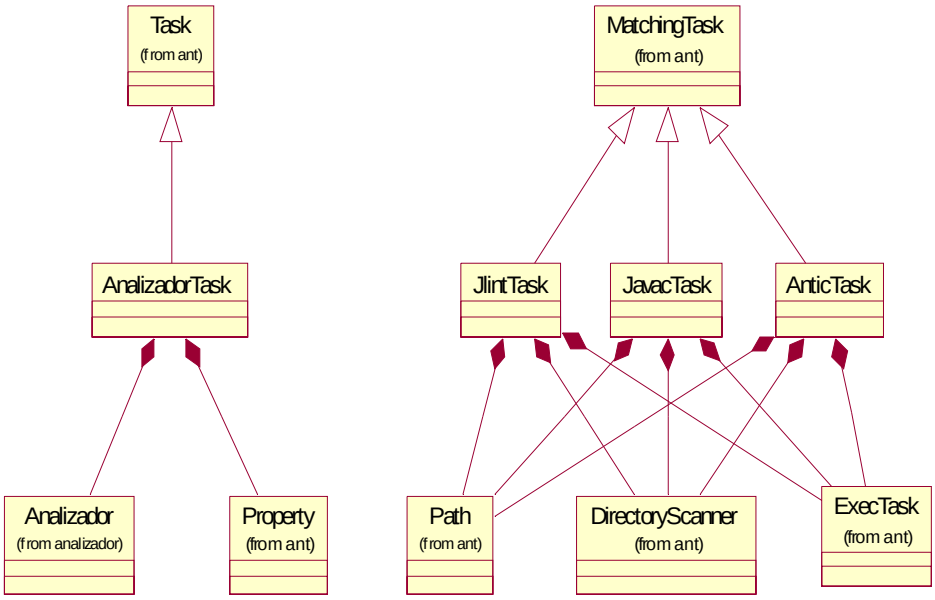


Figura 29. Diagrama de Clases del paquete proyecto.utilidad.ant

AnalizadorTask (Figura 29) es una clase que hereda de Task, lo que permite su ejecución desde un script Ant. El objetivo de esta clase es el proporcionar un interfaz Ant a la clase Analizador del paquete proyecto.analizador. Para este trabajo usa las clases Analizador, que se encarga de realizar el análisis al archivo indicado, y la clase Property de Ant, la que hace posible el uso desde el script Ant del número de errores o avisos que había en el archivo analizado.

Las clases JlintTask, JavacTask y AnticTask son muy similares. Las tres tratan de ser interfaces para que una herramienta sea ejecutada desde Ant. Estas herramientas son Jlint, Javac y Antic respectivamente. Estas clases heredan de MatchingTask, por lo que, además de poder ser ejecutadas desde Ant, heredan de una serie de métodos y campos muy útiles. En estas clases se usarán clases de Ant para simplificar las tareas que se tienen que realizar, como la localización de los archivos fuentes en un árbol de directorios o la propia ejecución de la herramienta.

Diagrama de clases del paquete proyecto.analizador

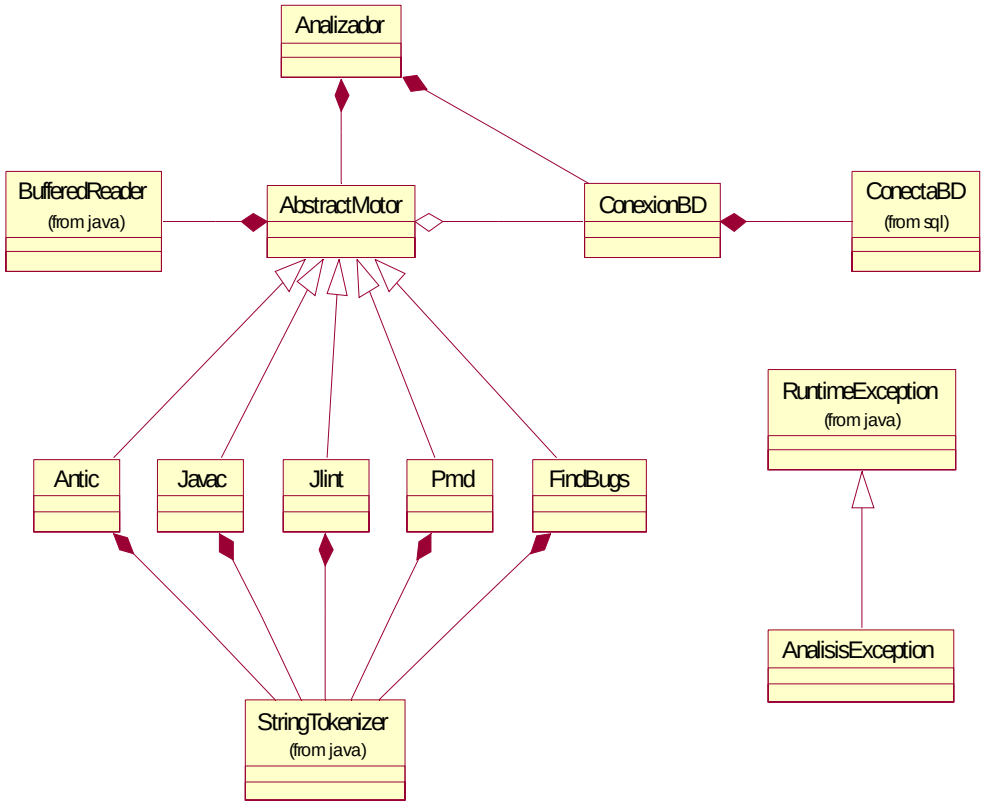


Figura 30. Diagrama de clases del paquete proyecto.analizador

En el diagrama de la Figura 30, se pueden apreciar todas las relaciones que tienen las clases del paquete proyecto.analizador la cuales resuelven el problema del análisis de los archivos de errores o avisos de varias herramientas.

Para la conexión con la Base de Datos se usa la clase ConexionBD que proporciona un interfaz simple para las operaciones que realizan las clases del paquete con la Base de Datos. Esta clase usa ConectaBD del paquete proyecto.utilidad.sql para realizar la conexión propiamente dicha.

En el diagrama se puede ver una jerarquía de herencia que tiene como cúspide la clase AbstractMotor. Ésta es una clase abstracta que se usa para simplificar la tarea de crear

motores de análisis de archivos de errores o avisos de las herramientas. Esta clase usa la clase BufferedReader de la API de Java para poder leer el archivo. Además utiliza ConexionBD para guardar los errores analizados. Todos los motores de análisis de errores deberán heredar de esta clase.

Cada uno de sus hijos son clases que implementan motores de análisis de archivos de error de herramientas. Cada una de estas clases lleva el nombre de la herramienta para la que está diseñada, de forma que sean fácilmente reconocibles. Todas estas clases usan la clase StringTokenizer de la API estándar de Java.

La clase principal del paquete es Analizador, la cual servirá de interfaz con el resto del proyecto. Esta usa las clases que heredan de AbstractMotor para realizar el análisis y también usa la clase ConexionBD para poder realizar consultas en la Base de Datos.

Por último hay una excepción AnalisisException, la cual hereda de RuntimeException. En esta excepción se guardarán los problemas que se puedan producir durante el análisis y se lanzarán para ser tratados por entidades superiores.

Diagrama de secuencia para el escenario: El archivo compila correctamente, pero tiene avisos

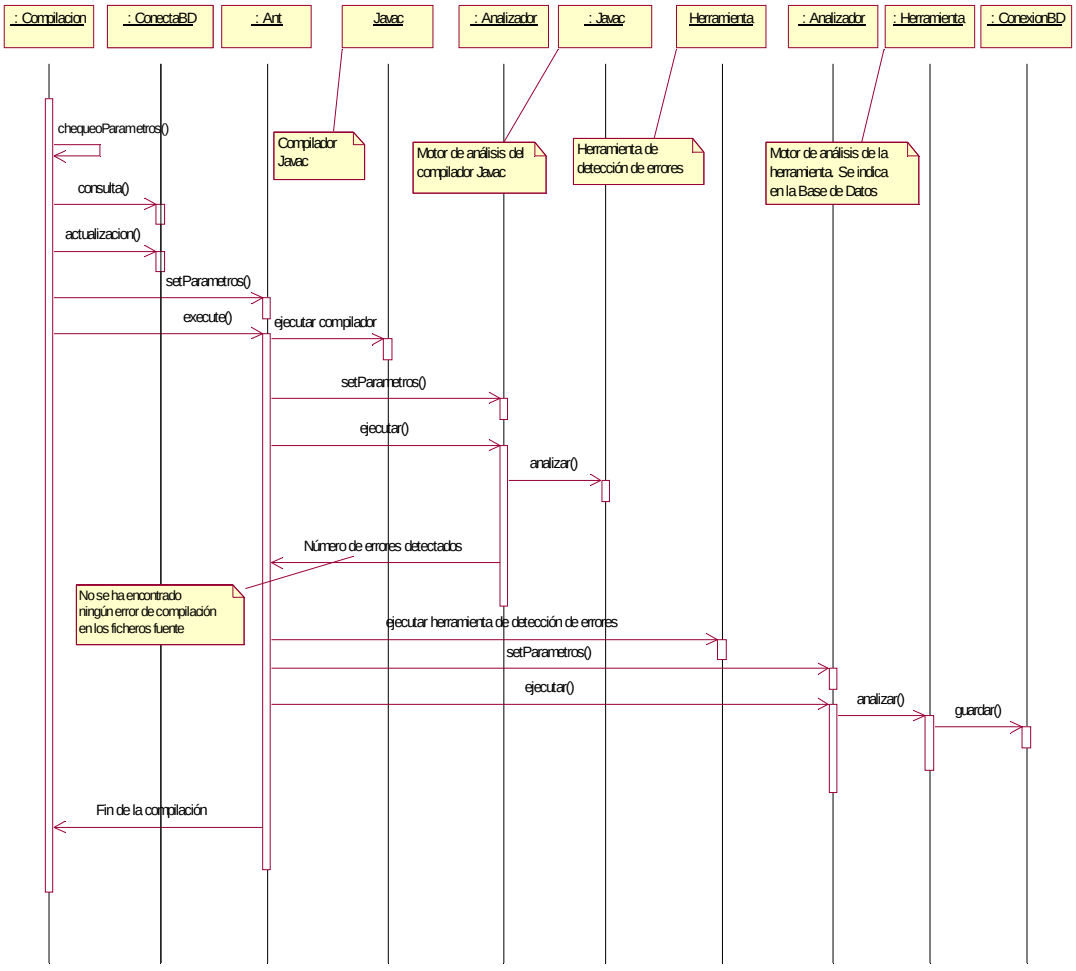


Figura 31. Diagrama de secuencia de un archivo que compila sin errores; pero otras herramientas de análisis generan avisos

En este caso tenemos un análisis en la que el compilador no detecta errores de compilación, lo cual hace que se puedan ejecutar las herramientas de detección de errores, las que detectan posibles errores en el código fuente. Esta información será guardada en la Base de Datos.

7.8.3 Empleo de la herramienta Ant en el análisis dinámico

El sistema de ejecución o análisis dinámico utiliza una filosofía muy similar para ejecutar las clases de los usuarios. Anteriormente se pensó en la creación de clases que trabajaban con hilos de ejecución para llevar a cabo la ejecución de las clases de los usuarios. El problema es que esta solución requería conocimientos sobre aplicaciones multihilo y sincronismo lo cual es muy costoso en tiempo. Por lo tanto, se buscó otra vía para solucionar el problema y esta solución se encontró otra vez en Ant. En este caso, se usa la tarea predefinida "Java". Esta tarea proporciona muchas funcionalidades y permite un tratamiento sencillo de las excepciones lanzadas durante la ejecución de la clase del usuario.

7.8.4 Diseño arquitectónico

Este subsistema está diseñado para su funcionamiento en segundo plano invocado por otros módulos del sistema SICODE, en concreto este subsistema será utilizado por el entorno de desarrollo integrado en Web (IDEWeb) (ver Capítulo 9), a través del cual, el usuario lanzará el análisis y la compilación para obtener los archivos en código objeto a partir del código fuente escrito. También, se utilizará este mecanismo para la fase de pruebas de ejecución, donde el entorno de desarrollo permitirá invocar el código ya compilado para realizar las pruebas oportunas. Todos los resultados, tanto de análisis estático del código fuente como dinámico de la ejecución, se almacenarán en una base de datos. Estos datos serán analizados por este subsistema y los resultados de este análisis podrán ser mostrados al usuario, bien a través de un navegador en una página Web independiente, o bien a través del entorno de desarrollo integrado (IDEWeb), de nuevo.

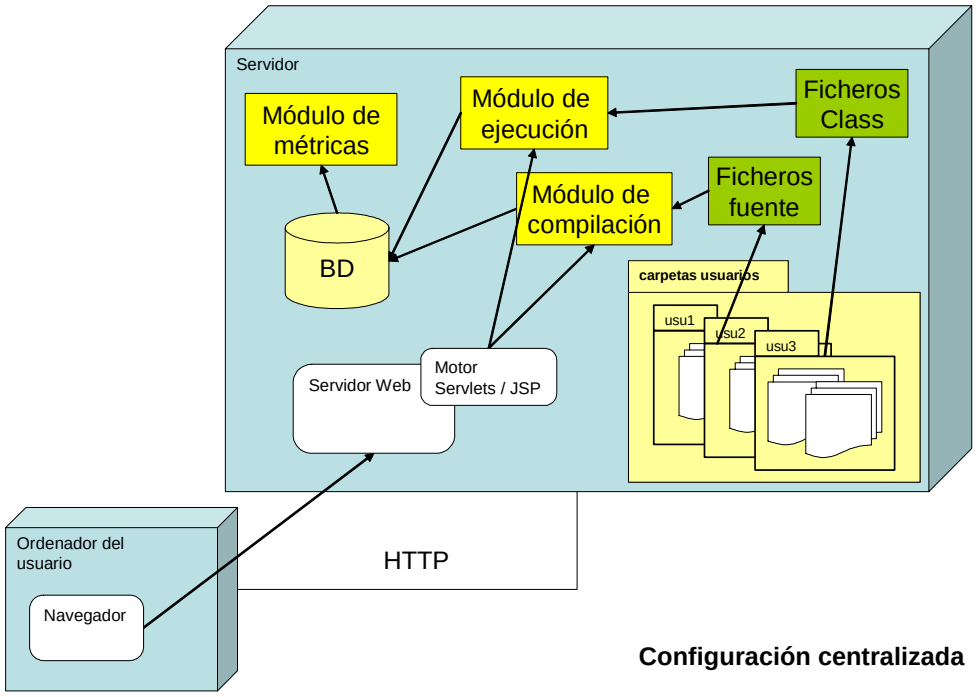


Figura 32. Diagrama que representa la arquitectura del sistema en una configuración centralizada

La independencia con la interfaz de usuario y la separación entre los módulos de análisis y el de generación de métricas permiten una gran flexibilidad en la configuración de la arquitectura del sistema. Planteamos aquí dos alternativas posibles en esta configuración:

- Arquitectura centralizada (Figura 32). En esta configuración todos los módulos residen en el servidor del sistema. Reciben las peticiones desde el entorno de desarrollo, que también podría residir en el servidor, y la interfaz con el usuario consistiría en un simple navegador, que se comunicaría con el servidor utilizando un protocolo HTTP. En esta configuración, los archivos de los proyectos que están desarrollando los usuarios, también residen en el servidor. Esta arquitectura permite una independencia total del usuario respecto al lugar de conexión, simplemente con una conexión a Internet y un navegador estándar, el usuario podría acceder a todos los servicios del sistema y al estado actual del trabajo que está desarrollando, sin necesidad de instalaciones de software adicional, ni de llevar en algún tipo de almacenamiento extraíble los ficheros del proyecto.

- Arquitectura distribuida (Figura 33). Gracias al planteamiento de módulos independientes, se puede optar por distribuir los módulos de análisis junto con un entorno de desarrollo a los ordenadores de los usuarios. Permitiendo la utilización de entornos de desarrollo más ricos y potentes, y enviando al servidor solamente los resultados del análisis para su almacenamiento en la base de datos. Esta comunicación puede realizarse mediante Servicios Web [Champbell 2001], que sobre un protocolo HTTP permita transmitir los resultados del análisis estático o dinámico que se realiza en el cliente. Es importante mantener la base de datos centralizada debido a la importancia de elaborar resultados de grupo y hacer análisis no sólo a nivel individual. El módulo de métricas que ofrece los resultados estadísticos seguirá situándose en el servidor ya que hace un uso intensivo de los datos almacenados en la base de datos, y no necesita de una interacción intensiva con el usuarios, simplemente la visualización de resultados gráficos y numéricos.

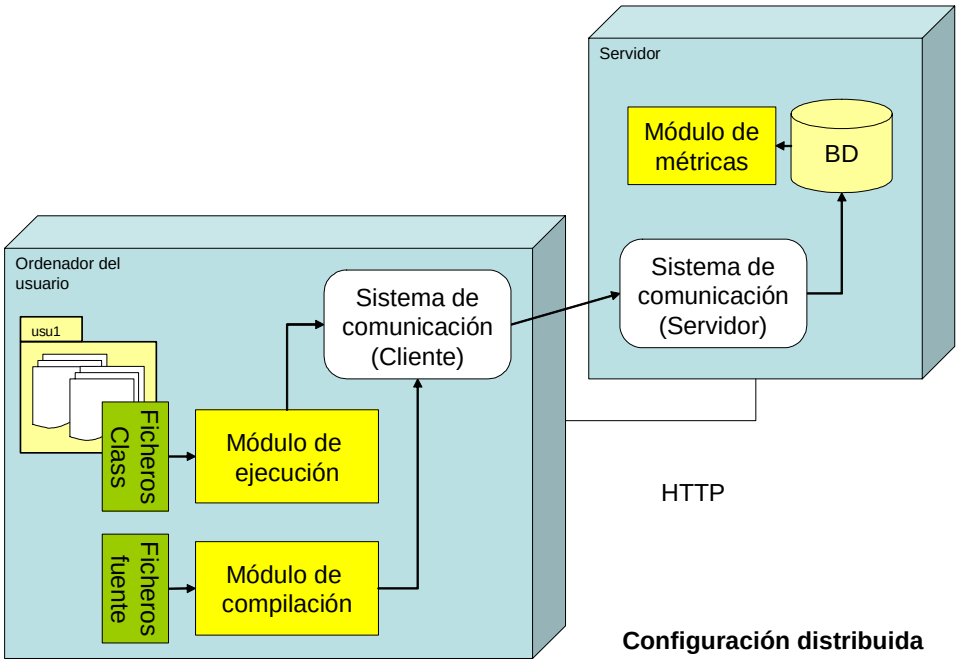


Figura 33. Diagrama que representa la arquitectura del sistema en una configuración distribuida

El sistema SICODE presentado en este documento utiliza la arquitectura centralizada; sin embargo, ya se dispone de prototipos que utilizan la segunda arquitectura presentada y se propone como una de las líneas de trabajo futuro (Capítulo 11) el potenciar esta arquitectura y explorar las mejoras que podemos obtener con ella.

7.8.5 Modelado de datos

Se considera importante reflejar el modelo de datos elegido para almacenar la información tanto sobre las herramientas como sobre los propios avisos almacenados, ya que sobre ella basaremos el análisis posterior y la construcción de los informes.

Debemos de tener en cuenta varios puntos a la hora de diseñar el modelo de datos. Por un lado, debemos de tener a nuestra disposición la información sobre cada una de las operaciones que puede llevar a cabo un usuario en el sistema: análisis estático utilizando diferentes unidades de compilación, análisis dinámico. Por otro lado, debemos de disponer de información que nos permita reconocer y clasificar los diferentes avisos y mensajes de error que pueden ocurrir al realizar estas operaciones. Por último, debemos relacionar todo, de forma que toda la información sea accesible.

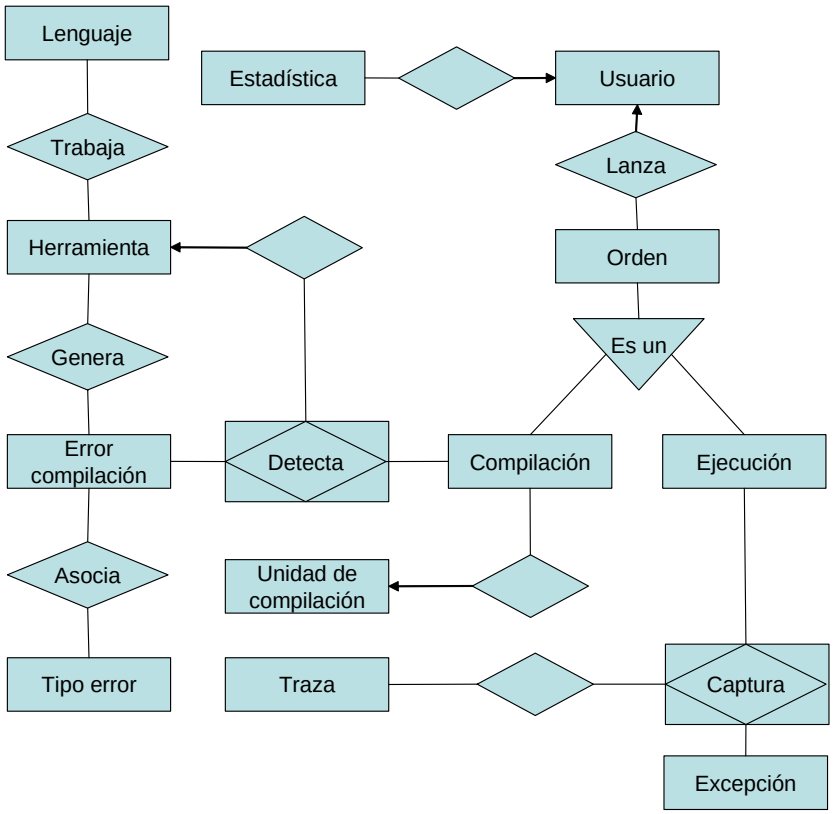


Figura 34. Diagrama entidad-relación subsistema de análisis de errores

Información sobre la actividad de los usuarios

El usuario puede realizar dos operaciones con el sistema: un análisis estático o una ejecución. De cada una de ellas, se deberá guardar información específica de cuándo y cómo se ha realizado, así como los mensajes que se han generado en cada caso. En un análisis, se guardan todos los errores de compilación o avisos detectados por la unidad de compilación en los ficheros de código fuente de un proyecto; mientras que en una ejecución, se guardan las excepciones que se producen durante la ejecución de la aplicación Java.

Información sobre la configuración del sistema

El sistema debe de tener constancia de los lenguajes y las herramientas con las que puede trabajar, así como la información sobre cada uno de los errores que pueden ser generados por cada una de las herramientas. Es esencial, disponer de información para poder clasificar correctamente los mensajes de error detectados para, posteriormente poder realizar estadísticas sobre los tipos de error. Por otro lado, se debe de disponer de una lista de las excepciones que puede detectar nuestro sistema.

Además, se deberá decidir como se almacenarán la información sobre los errores detectados en los análisis y las ejecuciones. También se debe almacenar la información sobre las herramientas que puede usar el sistema, los lenguajes de que se dispone e información sobre los usuarios. En la Figura 34 aparece el diagrama entidad – relación con todas las entidades y las relaciones existentes entre ellas.

En primer lugar, debemos relacionar la información que genera este subsistema con el resto de la base de datos, fundamentalmente la información sobre los usuarios. Esta relación se lleva a cabo a través de la entidad 'Usuario'.

La entidad 'Usuario' está relacionada con la entidad 'Estadística' mediante una relación de dependencia por existencia. La misión de esta entidad es el almacenamiento de algunos datos que pueden ser útiles para hacer estadísticas sobre los errores del usuario, o sobre el uso de la herramienta.

A su vez, la entidad 'Usuario' estará relacionada con la entidad 'Orden', la cual representa a las ordenes lanzadas por los usuarios. Estas órdenes podrán ser de dos tipos distintos, análisis estáticos representados por la entidad 'Compilación' y análisis dinámicos representados por la entidad 'Ejecución'. Las dos relaciones anteriores serían entidades hijas de la entidad 'Orden' en una relación de herencia.

La entidad ejecución estará relacionada con la entidad 'Excepción' mediante la relación 'Captura'. La entidad Excepción guardará la información sobre los distintos tipos de excepciones que pueden ser lanzadas por una aplicación. La relación 'Captura' guardará la información sobre la captura de excepciones durante la ejecución de una aplicación.

Mediante una agregación la relación 'Captura' se relaciona con la entidad 'Traza'. Esta última sirve para guardar la información sobre el volcado de la pila de llamadas por las que se fue propagando la excepción capturada.

La entidad 'Compilación' estará relacionada con la entidad 'Unidad de compilación' por una relación de dependencia por existencia. La 'Unidad de compilación' guardará la información sobre que pasos deben seguirse para llevar a cabo el análisis.

Por otro lado, la relación 'Compilación' se relaciona por medio de la relación 'Detecta' con la entidad 'Error de compilación'. La relación 'Detecta' guarda información sobre cada error de compilación o aviso encontrado por alguna herramienta durante un análisis determinado. 'Detecta' forma un agregado que se relaciona por relación de bitácora con la entidad 'Herramienta'.

La entidad 'Error de compilación' guarda la información sobre los errores que pueden ser detectados con nuestro sistema. Además de estar relacionada con 'Compilación', lo está también con la entidad 'Herramienta' a través de la relación 'Genera' y con la entidad 'Tipo Error' por medio de la relación 'Asocia'.

La entidad 'Tipo Error' guarda la información sobre los diferentes tipos en que se clasifican los errores y avisos detectados en el análisis.

La entidad 'Herramienta' guarda la información sobre los procesadores de lenguaje que emplea el sistema, como pueden ser compiladores o analizadores estáticos. Esta entidad tiene una relación con la relación 'Lenguaje' la cual guarda información sobre los diferentes lenguajes que pueden ser utilizados en la herramienta.

7.9 Archivos de estadísticas: Configuración y ejemplos

En este apartado, especificaremos la forma en la que se han resulto los requisitos establecidos para la generación de informes de estadísticas establecidos en el apartado 7.3.3 que se concreta en el escenario descrito en 7.5.3 y se detalla en 7.7.

7.9.1 Descripción de archivos XML para la configuración de las estadísticas

Los archivos de configuración de estadísticas proporcionan a este módulo una gran versatilidad. Están creados para que un usuario pueda crear fácilmente consultas complejas y avanzadas que se adapten a sus necesidades y luego pueda reutilizarlo siempre que lo necesite. En este apartado daremos es una visión general sobre las posibilidades que ofrece la configuración de estadísticas a los usuarios.

Empleo de XML y XML Schema

Cuando se pensó en la forma de configurar las estadísticas, se trato de buscar una forma de que estas fueran sencillas de utilizar para los usuarios y de procesar para el sistema. Por ello, se eligió XML [XML 2004] para guardar esa información. Puesto que un usuario puede leer fácilmente la configuración de las estadísticas y modificarlas, incluso con un simple editor de texto y de la misma manera el tratamiento de esa información se hace de forma estándar; además es posible validar el fichero.

Una vez elegido XML, se pensó en las posibles formas de validar los datos contenidos en el archivo. Las dos tecnologías más usadas actualmente son DTD y XML-Schema [XML Schema 2004]. La elección de XML-Schema se debe a que tiene características más avanzadas que DTD y parece que se va a imponer como estándar en un futuro próximo. Si se sabe diseñar bien el XML-Schema, uno puede ser capaz de validar los archivos XML con mayor precisión que un DTD. Desde el punto de vista del diseño del módulo de estadísticas esto presenta una ventaja enorme, puesto que no se necesita crear clases para el tratamiento de los errores del archivo, lo que es un ahorro de tiempo y esfuerzo, pero además, hace que el sistema sea menos dependiente de la forma que tengan los archivos de configuración de estadísticas. Esto hace que si se introducen nuevos tipos de estadísticas, no se deba modificar ninguna de las clases existentes, tan sólo se deberá de crear una nueva clase que interprete ese nuevo tipo.

XML-Schema es una tecnología relativamente nueva, por lo que se dispone de pocos analizadores (parsers) de XML que sean capaces de implementar la recomendación XML-Schema al completo. El único analizador que cumple la especificación XML-Schema al completo en el momento de desarrollar el prototipo es Xerces [Xerces 2004]. Se usó esta herramienta para chequear que los archivos XML validen el XML-Schema.

Después de la validación, se utiliza la tecnología DOM [DOM 1998] para acceder a los datos de los archivos, se eligió esta tecnología porque es muy sencilla de usar. Esta tecnología genera un árbol a partir del documento, el cual puede ser navegado usando métodos que permiten tratar con la estructura de datos.

Archivo de configuración de estadísticas

El archivo de configuración de estadísticas (ver apartados A.1 y A.2 del Apéndice A para ver dos ejemplos), es un archivo cuyo formato cumple la especificación XML 1.0 [XML 2004] y dispone de un archivo de XML-Schema (ver apartado A.3. para ver un ejemplo de este tipo de archivo) que permite su validación utilizando un analizador XML como Xerces.

El elemento *estadisticas* es el elemento raíz de los archivos, el cual puede tener anidados uno o más elementos que representan las diferentes estadísticas que queremos representar en el informe. Cada una de estas estadísticas está especificada por un elemento distinto. Para este prototipo se concibieron dos tipos de estadísticas distintas; pero el sistema permite que estos puedan ser ampliables en un futuro:

- Errores más frecuentes.
- Media de errores.

Hacemos, a continuación, una descripción de los atributos de cada estadística:

1. La estadística de Errores más frecuentes posee un atributo "numero" que representa la cantidad de tipos de errores más frecuentes que se desean mostrar, por defecto es 5. Su otro atributo "manejador" indica al sistema la clase Java que debe ser empleada para generar esta estadística. El valor por defecto es "proyecto.estadisticas.ErroresFrecuentes", pero puede ser cambiada por los usuarios.
2. La estadística de "Media de Errores" posee un atributo "evolucion" que representa el número de periodos para los que se desea conocer la media de errores, por defecto es 1. Su otro atributo "manejador" indica al sistema la clase que se debe emplear para generar esta estadística. El valor que tiene por defecto es "proyecto.estadisticas.MediaErrores", pero puede ser cambiada por los usuarios.

Los dos tipos de estadísticas anteriores tienen la misma estructura, que está compuesta por los cuatro elementos que se describen a continuación, siguiendo este orden:

1. Definición del elemento “periodo”. Indica los límites inicial y final de los datos que analizamos. Debe aparecer sólo en una vez.

   Habrá tres formas posibles de indicar los periodos:

   - Mediante un intervalo temporal relativo (unidad día). El final es hoy menos los días que se indiquen como retardo.
   - Mediante un intervalo temporal absoluto (unidad día). El final es la fecha indicada. El formato de la fecha es: AAAA-MM-DD (año - mes – día)
   - Mediante un intervalo entre análisis absoluto (unidad compilación). El final es el análisis indicado.

   Siempre habrá que especificar la duración del periodo, cuya unidad será el día o cantidad de análisis dependiendo del método que se use para indicar el periodo. El inicio de un periodo de final "f" y de duración "d" será "f-d"

2. Definición del elemento “usuarios”. El elemento “usuarios” sirve para indicar los usuarios para los cuales se va a realizar el estudio estadístico. Al igual que el anterior, debe aparecer una vez.

   La forma de incluirlos es indicando dentro de elemento anidado “usuario” el código de usuario que se desee estudiar. El elemento usuarios tiene dos atributos booleanos:

- "autor". Se usa para indicar que se haga la estadística para el usuario que lanza el estudio estadístico. Por defecto es "true".
- "todos". Se usa para indicar que se haga la estadística a todos los usuarios del sistema. Por defecto es "false".

3. Definición del elemento “errores”. Permite especificar que tipos de errores se incluirán en el análisis. Este elemento debe aparecer al menos en una ocasión, pero puede aparecer más de una vez.

Se puede indicar que errores se quiere usar en la muestra de estadísticas. En primer lugar se pueden meter condiciones de dos tipos:

- elementos "incluir", para usar los errores que cumplan una condición.
- elementos "excluir", para usar los errores que no cumplan una condición.

Si se especifica más de una condición se deberán indicar como se relacionan esas condiciones, para ello se emplean los elementos "and" y "or".

Las condiciones pueden ser de tres tipos:

- "codigo". Donde se indica el código del error.
- "tipo". Donde se indica un tipo de errores.
- "herramienta". Donde se indica la herramienta que generó los errores.

4. Definición del elemento “compilaciones”. Indica que unidades de compilación se quieren usar a la hora de hacer las estadísticas. Puede aparecer una vez o ninguna vez.

Como se puede ver la forma de configurar los archivos de estadísticas ofrece muchas posibilidades, pero de la misma forma es simple e intuitiva. En el Apéndice IV se presenta un archivo de ejemplo de como configurar las estadísticas, además ese archivo fue el usado para generar los resultados que se presentan en el siguiente apartado.

7.9.2 Descripción del informe de estadísticas

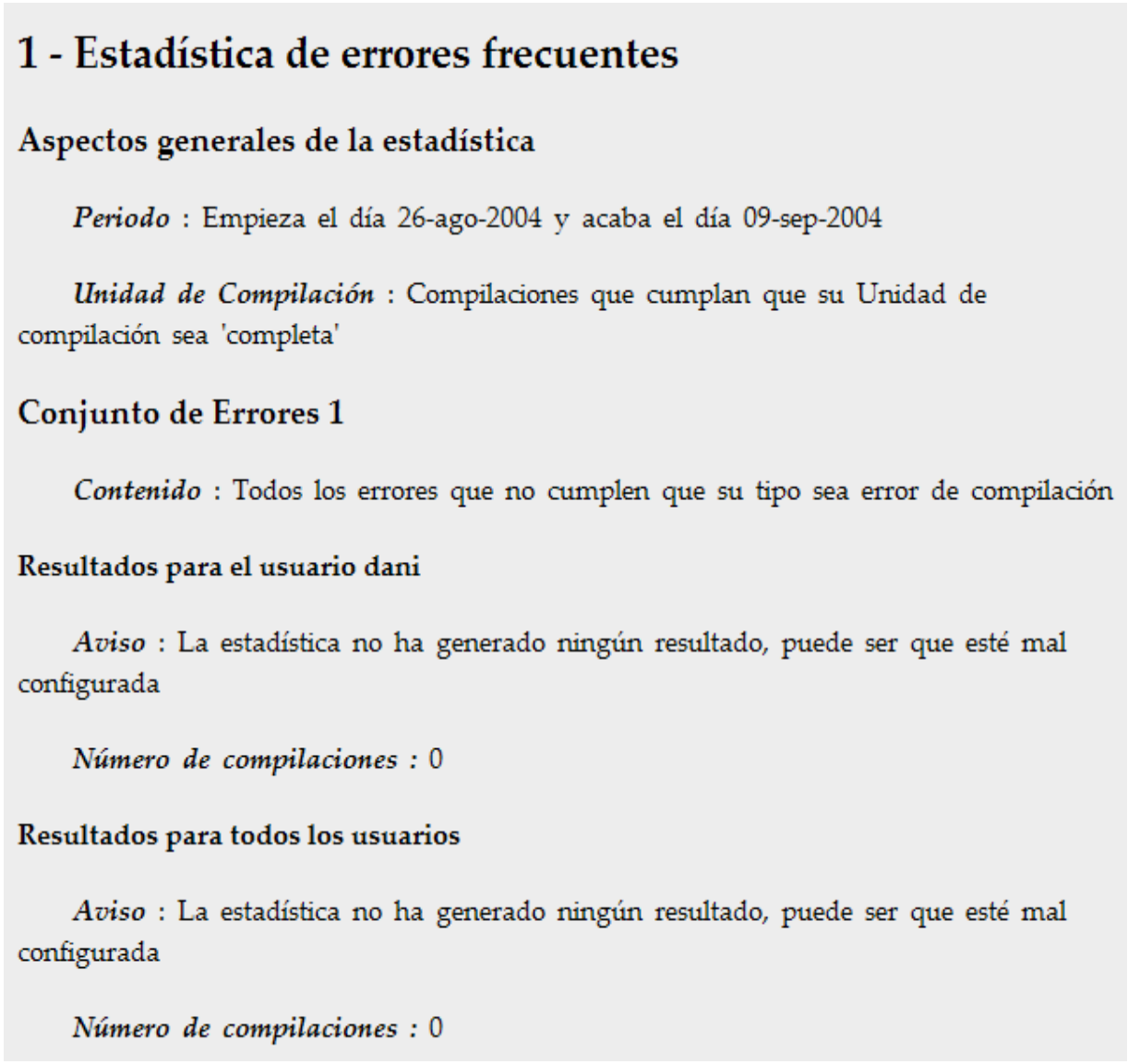
En este apartado hablamos de como se deben interpretar los informes que contienen los resultados de las estadísticas. Vamos a verlo con un informe de ejemplo resultado del procesamiento archivo de configuración de estadísticas que aparece en el apartado A.4. Archivo ejemplo de configuración de estadísticas del Apéndice A.

Figura 35. Esquema general del informe de estadísticas

Un esquema de la estructura del informe (Figura 35) es el siguiente:

- En primer lugar, el título principal del informe.
- Cada una de las estadísticas está numerada y aparecen consecutivamente, no hay límite en el número de estadísticas para un informe.
- Al comienzo se indica el tipo de la estadística: errores frecuentes o media de errores (en un periodo o la evolución a través de varios periodos).
- A continuación, se presentan una serie de datos sobre la estadística que dependen del tipo. Para errores frecuentes aparece: periodo, conjunto de errores y usuario; para media de errores: conjunto de errores y si no es una evolución, el periodo.
- Gráfica de los resultados del análisis, utilizando el tipo de gráfica adecuado al tipo de estadística realizado.
- Si la estadística realizada es una evolución de la media de errores, después de la gráfica, se presenta una descripción de los periodos para los que fueron realizadas las medias (Figura 39).
- Por último, se dan los resultados de la estadística de forma numérica mediante una tabla.

A continuación vamos a ir describiendo más detalladamente los aspectos destacables de cada una de las estadísticas del informe de ejemplo.



1 - Estadística de errores frecuentes

Aspectos generales de la estadística

Periodo : Empieza el día 26-ago-2004 y acaba el día 09-sep-2004

Unidad de Compilación : Compilaciones que cumplan que su Unidad de compilación sea 'completa'

Conjunto de Errores 1

Contenido : Todos los errores que no cumplen que su tipo sea error de compilación

Resultados para el usuario dani

Aviso : La estadística no ha generado ningún resultado, puede ser que esté mal configurada

Número de compilaciones : 0

Resultados para todos los usuarios

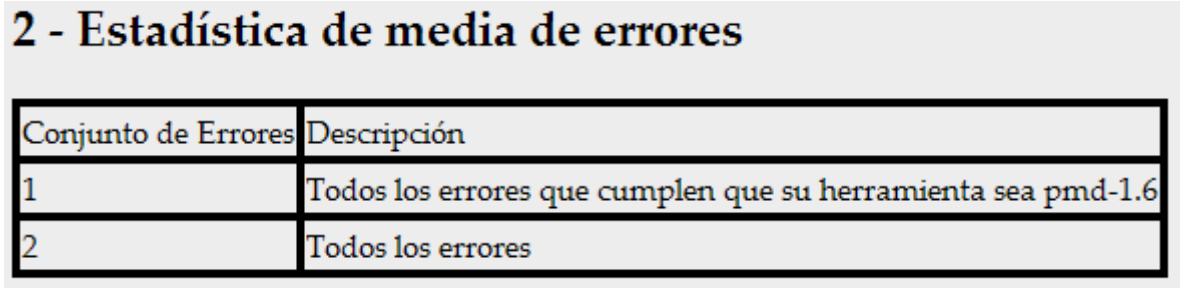
Aviso : La estadística no ha generado ningún resultado, puede ser que esté mal configurada

Número de compilaciones : 0

Figura 36. Resultados estadística errores frecuentes (primera) con posible error de configuración

En el informe la primera estadística que se intenta mostrar es del tipo de errores frecuentes (Figura 36). Sin embargo, este es un caso especial, puesto que las consultas que se han llevado a cabo no han proporcionado ningún resultado, lo que puede significar que la estadística esté mal configurada; aunque sintácticamente sea correcta, por lo tanto, se avisa al usuario para que compruebe que todo es correcto.

La siguiente estadística es de evolución de la media de errores, en ella se compara dos conjuntos de errores, que se indican en la tabla que hay detrás del título (Figura 37).



2 - Estadística de media de errores

Conjunto de Errores	Descripción
1	Todos los errores que cumplen que su herramienta sea pmd-1.6
2	Todos los errores

Figura 37. Información general de la segunda estadística (evolución media de errores)

La gráfica de líneas (Figura 38) muestra como se ha desarrollado la evolución de la media de estos dos conjuntos de errores, en los periodos que se establecieron en la configuración. En la leyenda se puede ver que colores se corresponden a cada pareja conjunto de errores – usuario.

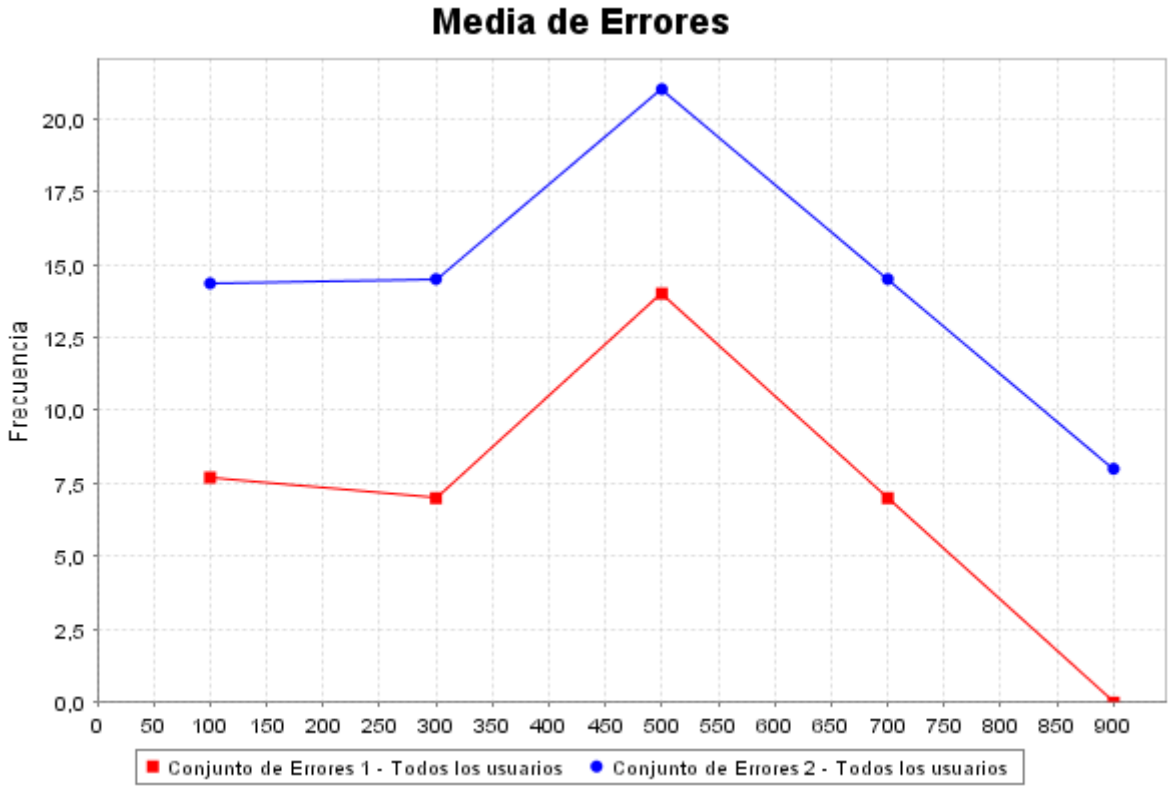


Figura 38. Gráfica de líneas de la evolución de la media de errores (segunda estadística)

Después de la gráfica, se presenta una descripción de los periodos para los que fueron realizadas las medias (Figura 39). Puede comprobarse que a la hora de representar la gráfica se tomó el punto medio del periodo como representante de todo el intervalo siguiendo los criterios estadísticos. A continuación, se dan los resultados de la estadística de forma numérica mediante una tabla.

Periodos

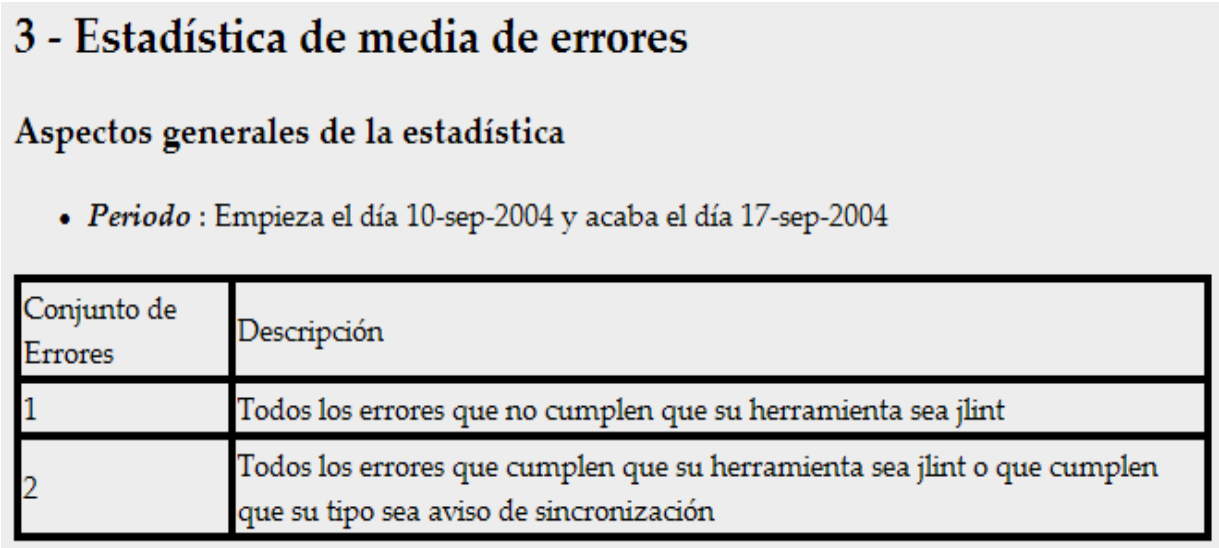
P1	P2	P3	P4	P5
Empieza en la compilación 0 y acaba en la compilación 200	Empieza en la compilación 200 y acaba en la compilación 400	Empieza en la compilación 400 y acaba en la compilación 600	Empieza en la compilación 600 y acaba en la compilación 800	Empieza en la compilación 800 y acaba en la compilación 1000

Tabla de Resultados para todos los usuarios

	P1	P2	P3	P4	P5
Conjunto de Errores 1	7,70	7,00	14,00	7,00	0,00
Conjunto de Errores 2	14,35	14,50	21,00	14,50	8,00

Figura 39. Tabla con la descripción de los periodos y tabla con los resultados de forma numérica

La siguiente estadística es también una estadística de media de errores; pero en este caso, no hay evolución, por lo tanto sólo se estudian los avisos para un periodo y este se indica al comienzo, en la información general de la estadística. Además, aquí se describen los conjuntos de errores que intervienen en la estadística (Figura 40).



3 - Estadística de media de errores

Aspectos generales de la estadística

- ***Periodo*** : Empieza el día 10-sep-2004 y acaba el día 17-sep-2004

Conjunto de Errores	Descripción
1	Todos los errores que no cumplen que su herramienta sea jlint
2	Todos los errores que cumplen que su herramienta sea jlint o que cumplen que su tipo sea aviso de sincronización

Figura 40. Comienzo de la tercera estadística

Al tratarse de una gráfica de media de errores en un periodo, la gráfica que se genera es una gráfica de barras (Figura 41). En la que aparece una barra para cada conjunto de errores y además se duplican para poder comparar la información para un usuario individual con la del conjunto de todos los usuarios.

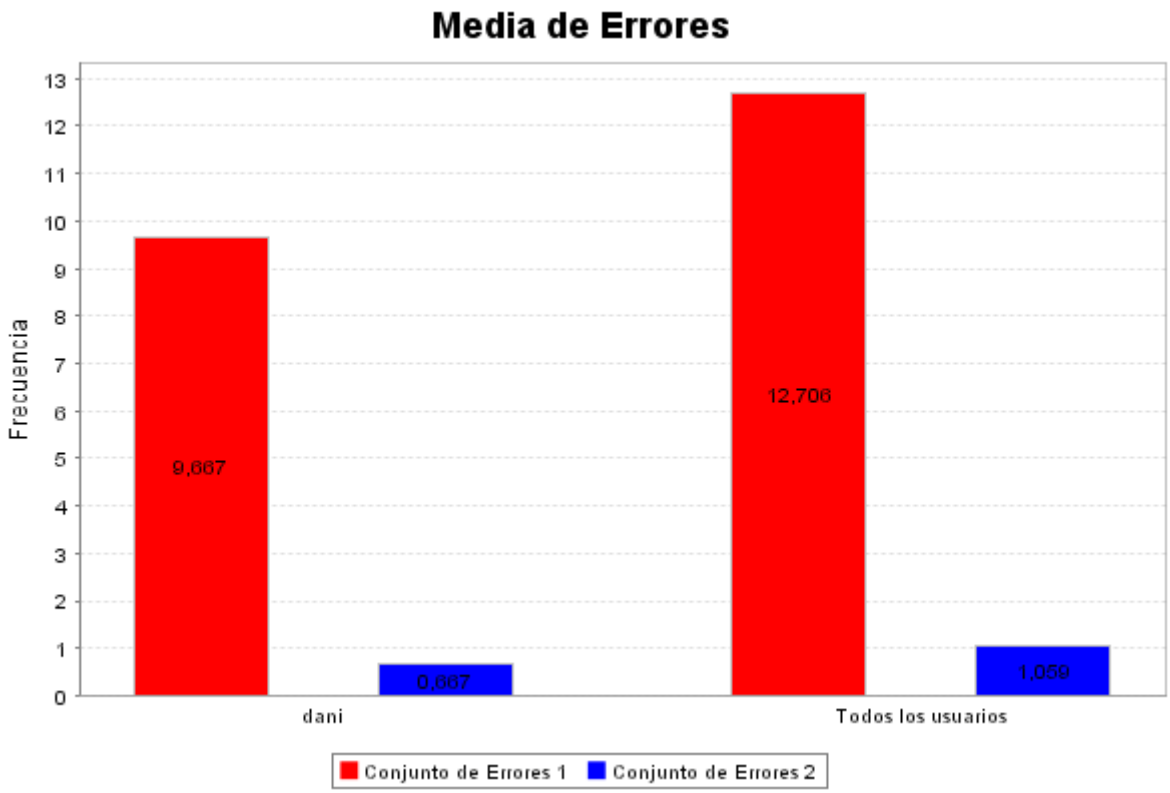


Figura 41. Gráfica de barras para una media de errores

Por último se presentan los resultados de forma numérica en una tabla (Figura 42).

	Conjunto de Errores 1	*Conjunto de Errores 2*
dani	9,667	0,667
Todos los usuarios	12,706	1,059

Figura 42. Resultados numéricos de la media de errores (tercera estadística)

La siguiente estadística, es una estadística de errores frecuentes, en este caso no hay ningún problema a la hora de realizar las consultas. Por lo que, además de la información general, se generarán las gráficas y los resultados numéricos.

Al principio de la estadística, se presentan los datos generales sobre la estadística como el periodo, el conjunto de errores que intervienen en la estadística y el usuario afectado (Figura 43).

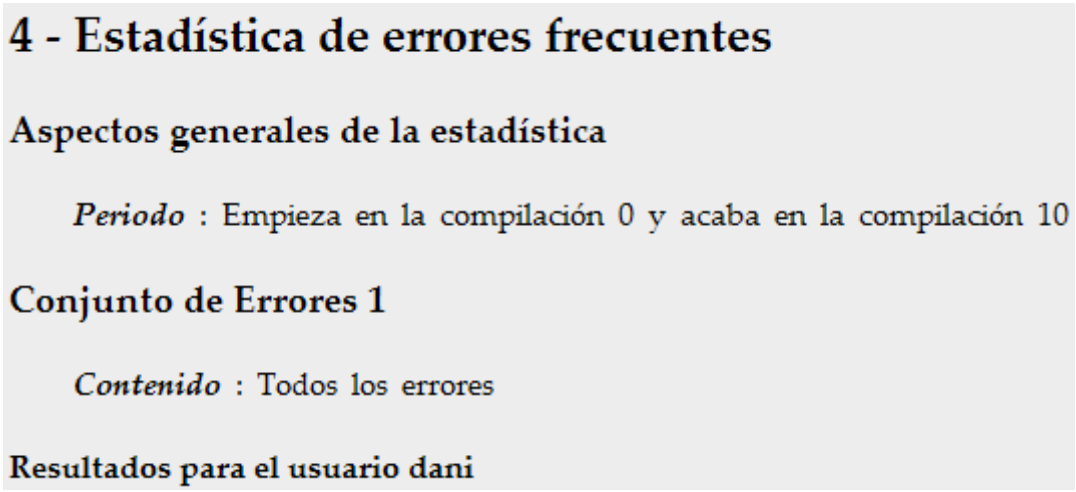


Figura 43. Descripción de los aspectos generales de la cuarta estadística

Para mostrar gráficamente los resultados se utiliza una gráfica de barras. Los errores frecuentes están ordenados desde el más frecuente hasta el menos frecuente. En la leyenda se puede ver a que errores corresponde cada barra (Figura 44).

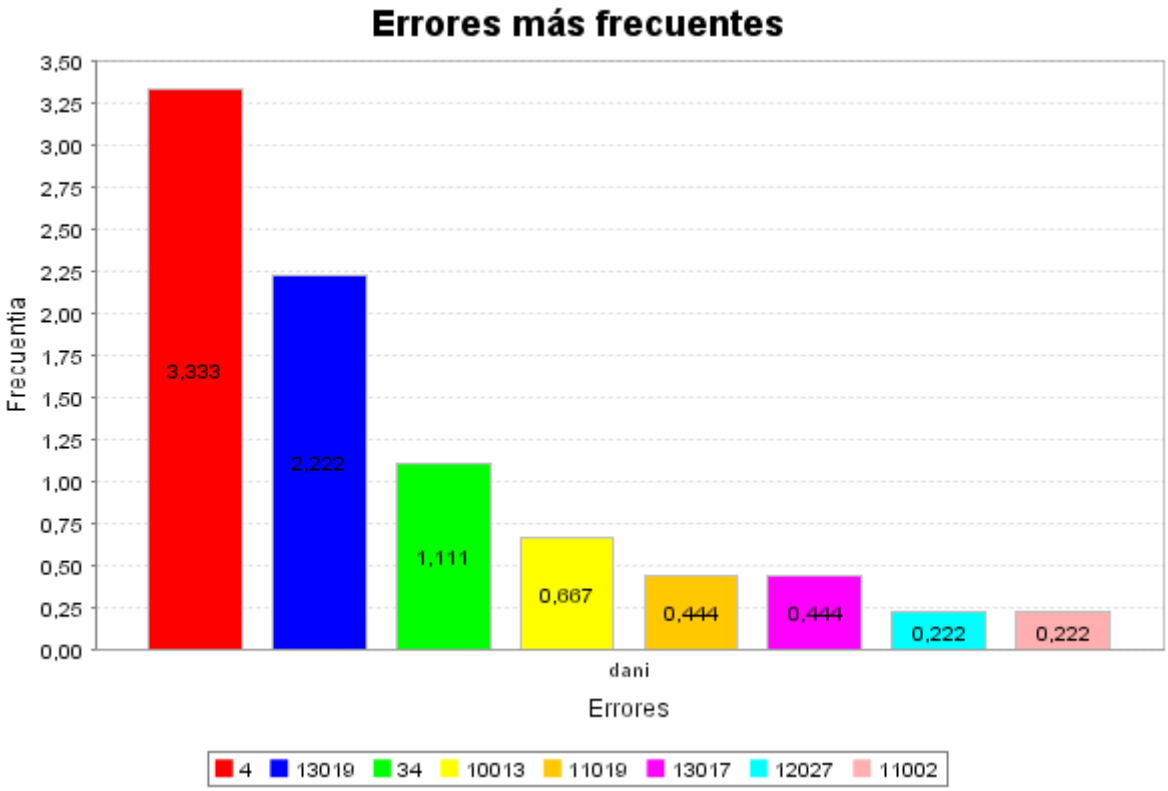


Figura 44. Gráfica de barras de errores frecuentes (cuarta estadística)

Después de la gráfica, se presenta una tabla con los resultados de forma numérica (Figura 45). Además del valor número, se puede ver que el código del error tiene un hiperenlace a la página Web de la base de conocimientos de errores (Capítulo 9. Entorno de desarrollo integrado en Web y base de conocimientos colaborativa: IDEWeb) donde está la información semántica sobre el error, lo cual permite que se pueda consultar esta información con facilidad.

Código de Error	4	13019	34	10013	11019	13017	12027	11002
Frecuencia	3,333	2,222	1,111	0,667	0,444	0,444	0,222	0,222

Número de compilaciones : 9

Figura 45. Tabla de errores más frecuentes (cuarta estadística)

Por último, tenemos una estadística de errores frecuentes que presenta un problema (Figura 46). La estadística está configurada para obtener los cinco errores más frecuentes de un conjunto de errores, pero en el análisis del conjunto de errores sólo se encuentran dos errores (Si se comprueba el contenido del conjunto de errores 1, se verá que es lógico que sólo haya dos). Para avisar al usuario se pone un mensaje de aviso en la sección de información general de la estadística.

5 - Estadística de errores frecuentes

Aspectos generales de la estadística

Periodo : Empieza en la compilación 0 y acaba en la compilación 10

Conjunto de Errores 1

Contenido : Todos los errores que cumplen que su código sea 13019 o que cumplen que su código sea 13000

Resultados para el usuario dani

Aviso : La estadística está configurada para dar 5 de errores pero sólo hay resultados de 2 errores.

Figura 46. Estadística de errores frecuentes con menos errores de los necesarios

Como se puede ver en la Figura 47, la gráfica de errores frecuentes en vez de ser de cinco errores frecuentes pasa a ser de los dos errores más frecuentes.

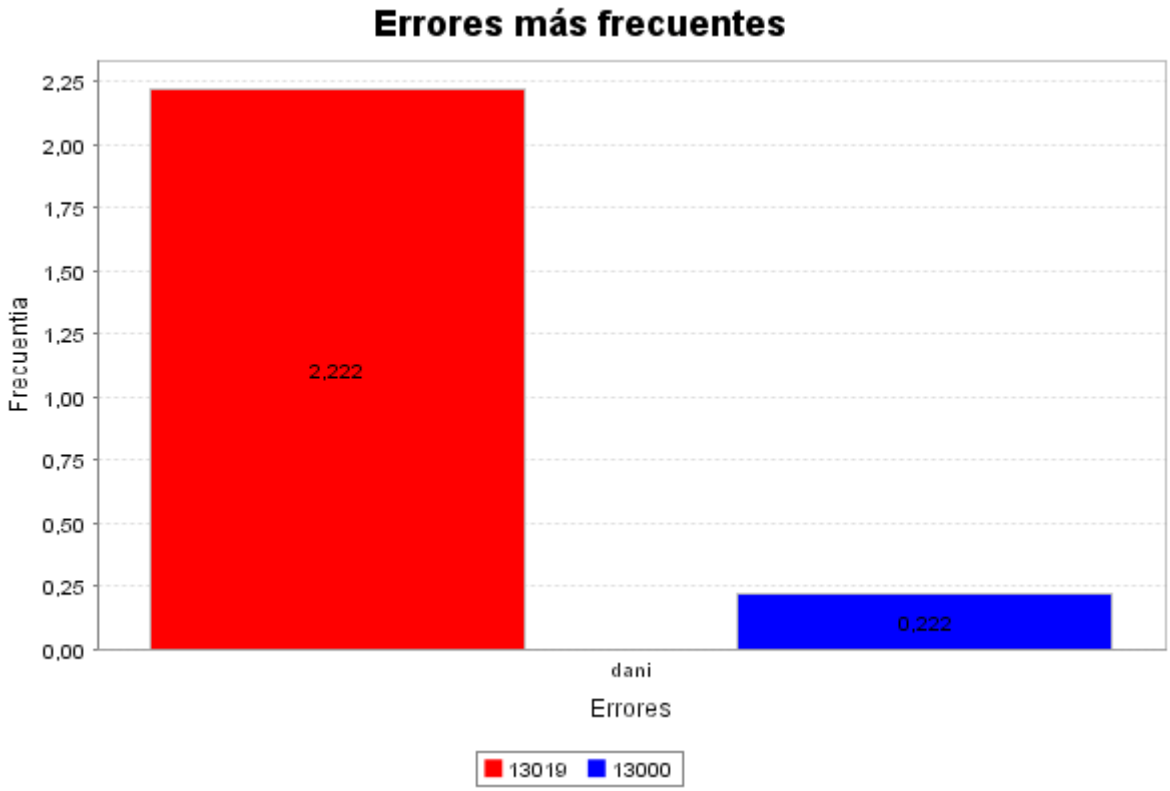


Figura 47. Gráfica de errores frecuentes con menos errores de lo normal

Por último, se pone la tabla con los resultados para los dos errores con su frecuencia y los enlaces a sus páginas Web (Figura 48).

Las estadísticas de errores frecuentes siempre ponen al final de sus resultados el número de análisis que se emplearon para realizar la estadística, así se puede tener una idea del uso del sistema en ese periodo.

Código de Error	13019	13000
Frecuencia	2,222	0,222

Número de compilaciones : 9

Figura 48. Tabla de resultados de la quinta estadística

Para finalizar el informe de resultados, se incluyen los logotipos del W3C: xhtml y CSS informando de que esta página cumple los estándares y permite su comprobación.

7.10 Evaluación del Sistema de análisis de programas

Para comprobar la viabilidad de la utilización de este subsistema con proyectos reales y en condiciones parecidas a las que tendría en un entorno real, se llevó a cabo un experimento [Pérez 2004c]. Para ello empleamos diez proyectos elegidos al azar, entre todos los disponibles de la asignatura de Metodología a la Programación de primer curso (segundo cuatrimestre) de la Escuela Universitaria Ingeniería Técnica de Informática de Oviedo[5].

[5] Página Web de la asignatura de Metodología de la Programación: http://www.euitio.uniovi.es/~mp

Para desarrollar los proyectos de esta asignatura se utilizó el entorno de desarrollo de Borland JBuilder 9.0.

Estos proyectos fueron analizados usando la unidad de compilación "completa", que empleaba todas las herramientas de detección de errores configuradas en el sistema, es decir Antic, Jlint, FindBugs y PMD. Se guardaron todos los avisos en la base de datos del proyecto y se comprobó que fuesen errores reales y que no constituyesen falsos positivos.

Tabla 7. Contabilización y medias de los avisos generados por cada una de las herramientas.

	Código error	**Práctica**										
		A1	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**A8**	**a9**	**a10**	**media**
Antic	10012	0	0	0	0	1	0	0	0	1	1	0,3
	10013	2	0	0	0	0	0	0	1	0	0	0,3
	10016	0	0	0	0	1	0	0	0	0	0	0,1
	Total	2	0	0	0	2	0	0	1	1	1	0,7
FindBugs	11004	0	0	0	0	0	1	2	0	0	0	0,3
	11010	1	0	0	0	0	0	0	0	0	0	0,1
	11014	0	0	0	1	0	0	0	0	7	0	0,8
	11015	0	0	0	0	0	0	0	0	1	0	0,1
	11020	0	0	6	0	0	0	0	0	7	0	1,3
	11021	2	0	0	0	0	0	0	0	3	0	0,5
	11022	0	6	0	0	0	0	0	3	0	0	0,9
	11023	0	0	0	0	0	1	2	4	0	0	0,7
	Total	3	6	6	1	0	2	4	7	18	0	4,7
Jlint	12002	1	3	0	5	7	3	1	2	1	0	2,3
	12004	0	0	0	0	0	0	0	0	0	1	0,1
	12005	1	0	0	0	0	0	0	0	0	0	0,1
	12014	0	0	0	0	0	11	3	0	0	0	1,4
	Total	2	3	0	5	7	14	4	2	1	1	3,9
PMD	13036	0	0	0	0	0	0	0	0	0	1	0,1
	13038	0	0	2	0	0	0	0	0	0	0	0,2
	13039	0	3	0	8	17	6	15	9	1	3	5,9
	13040	1	1	3	0	2	0	0	4	6	0	1,7
	13042	0	0	0	0	1	1	0	0	0	0	0,2
	13043	0	0	0	1	2	0	1	0	1	0	0,5
	13044	4	3	6	0	14	0	2	6	2	0	3,7
	13046	2	0	0	0	0	0	0	0	3	0	0,5
	13061	0	0	1	0	0	0	0	0	0	0	0,1
	13071	0	0	1	0	0	10	0	0	0	0	1,1
	13072	0	1	0	0	0	1	0	0	0	0	0,2
	13073	2	1	1	0	0	0	1	0	2	0	0,7
	13080	2	2	2	2	5	2	2	2	6	3	2,8
	13083	0	0	0	0	0	0	0	0	1	0	0,1
	13084	2	2	1	2	2	7	0	4	2	1	2,3
	Total	13	13	17	13	43	27	21	25	24	8	20,1
	Totales	20	22	23	19	52	43	29	35	44	10	

Los resultados de este experimento pueden verse en la Tabla 7 donde se presentan el número y tipo de avisos generados por las herramientas, para cada uno de los proyectos, así como la media de cada uno de ellos. Los avisos están representados por su código en la base de datos del proyecto. Para poder conocer el significado de cada aviso se muestra una segunda tabla Tabla 8 donde se asocia a cada aviso una descripción breve sobre su significado.

Tabla 8. Correspondencia de cada código con el mensaje de aviso.

Código	Descripción
10012	May be wrong assumption about IF body
10013	May be wrong assumption about ELSE branch association
10016	Possible miss of BREAK before CASE/DEFAULT
11004	Comparison of String objects using == or !=
11010	Possible null pointer dereference
11014	Uninitialized read
11015	Useless control flow
11020	Unwritten field
11021	Unread field: should this field be static?
11022	Unused field
11023	Unread field
12002	Local variable '*name*' shadows component of class '*name*'.
12004	Method *name* can be invoked with NULL as *number* parameter and this parameter is used without check for NULL.
12005	Value of referenced variable '*name*' may be NULL.
12014	Compare strings as object references.
13036	All methods are static. Consider using Singleton instead.
13038	Avoid unnecessary if..then..else statements when returning a boolean
13039	Avoid unnecessary comparisons in boolean expressions
13040	Switch statements should have a default label
13042	Avoid reassigning parameters such as '*name*'
13043	A high ratio of statements to labels in a switch statement. Consider refactoring.
13044	Avoid calls to overridable methods during construction
13046	This final field could be made static
13061	Avoid unused imports such as '*name*'
13071	Method name does not begin with a lower case character.
13072	Class names should begin with an uppercase character and not include underscores
13073	Abstract classes should be named 'AbstractXXX'
13080	The same String literal appears *number* times in this file; the first occurrence is on line *number*
13083	Avoid unused private fields such as '*name*'
13084	Avoid unused local variables such as '*name*'

A continuación se presentan los comentarios sobre los resultados obtenidos de cada una de las herramientas de detección de errores que se usaron en el análisis.

Antic: Se puede ver claramente que detecta pocos errores, la explicación de esta escasez de avisos es el uso de un entorno de desarrollo, en este caso JBuilder. Un IDE facilita que los archivos fuentes tengan una buena indentación. Como el análisis de Antic se centra en

errores provocados por una mala indentación, apenas se generan errores. Su ejecución en casi instantánea.

Jlint: Genera avisos importantes que claramente son síntomas de errores, como las comparaciones de String como una referencia o la ocultación de variables en una herencia. Su ejecución en casi instantánea.

FindBugs: Al igual que Jlint los errores generados son importantes, y casi siempre se complementan los que indican Jlint y FindBugs. Su ejecución es bastante lenta.

PMD: La herramienta que más avisos da es PMD. Entre los avisos hay algunos que no son demasiado importantes, pero que pueden llevar a confusiones, como los nombres de clases y métodos que no cumplen las normas habituales de nombramiento. Algunos de los avisos están destinados a indicar zonas de código donde sería una buena medida llevar a cabo una refactorización. La ejecución de esta herramienta es bastante lenta.

Las prácticas constaban de unas 1100 líneas de código repartidas en 14 archivos fuente java. En la ejecución de estas se tardó una media de aproximadamente 7 segundos.

7.11 Limitaciones del prototipo

Una de las líneas de mejora de este sistema es la creación de herramientas que faciliten el mantenimiento y la ampliación del sistema, para simplificar el trabajo del administrador del sistema. Entre esas mejoras se incluirían las herramientas que faciliten:

- La creación de nuevos scripts Ant para implementar nuevas Unidades de Compilación.
- La creación de nuevos archivos de configuración de estadísticas.
- La ampliación del archivo XML-Schema para crear nuevos tipos de estadísticas.
- El registro de nuevas herramientas en el sistema.

El sistema actual permite realizar todas estas operaciones; pero se realizan manualmente sobre los archivos de configuración, sin ningún apoyo de ninguna herramienta y requieren un cierto grado de preparación, si se quiere hacer que el sistema pueda ser administrado más fácilmente se deberá crear herramientas que simplifiquen estas tareas.

La parte de ejecución permite realizar la ejecución de clases para realizar pruebas de funcionamiento, estas pruebas podrían ser mucho más completas si integrásemos este módulo con una herramienta de pruebas como puede ser JUnit [JUnit 2002].

En el módulo de estadísticas sería necesario mejorar la usabilidad del informe de estadísticas y dejarlo en un formato que pueda ser procesado por otros servicios.

7.12 Conclusiones

En el sistema realizado se cumplen los objetivos planteados en la introducción y el análisis. Se ha logrado crear un sistema capaz de:

- Gestionar los errores cometidos por los usuarios a la hora de escribir código, los errores no sólo servirán como indicación de algo que hay que corregir; sino una oportunidad para aprender a hacer las cosas mejor para obtener un código fuente de mayor calidad.

- Detectar automáticamente estos errores al realizar cada compilación y también al ejecutar las clases en las pruebas, para ello se utiliza el compilador y otras herramientas de análisis estático y sus resultados son guardado en una base de datos.
- Crear una historia de compilación, compilación tras compilación se van acumulando los resultados de esos análisis con lo que podremos realizar un seguimiento de los errores de un usuario o de un grupo de usuarios trabajando en un proyecto.
- Analizar los datos del análisis actual y de la historia de compilación, guardados en la base de datos para generar estadísticas. Para proporcionar información al usuario no sólo se procesan los datos del último análisis sino que se recoge toda la historia de compilación con lo que podemos obtener no sólo frecuencia de errores sino también evolución de estos errores.
- Se presenta la información de forma que el usuario pueda obtener conocimiento. Se proporciona información numérica; pero también gráfica adaptada a cada usuario, cada usuario obtiene información sobre sus compilaciones; pero además puede configurarla para poder analizar ciertos datos que le pueden interesar a él y no a otro usuario.

Los ámbitos donde este proyecto tiene utilidad práctica son:

- El aprendizaje de la programación. En este contexto el papel de supervisor lo desempeñaría un profesor y el de desarrollador los estudiantes. Estos últimos disponen de un sistema de análisis avanzado capaz de detectar más errores que un simple compilador, proporcionándoles una información más precisa acerca de la corrección de su código. El sistema de ejecución capaz de capturar las excepciones, proporciona la posibilidad de realizar pruebas unitarias y almacenar los resultados. El módulo de generación de estadísticas hace que tanto desarrolladores como supervisores tengan todos los datos a su alcance de una forma sencilla y así poder utilizarlos para ir aprendiendo a realizar un código de mayor calidad y evitar errores los estudiantes y ver los puntos en donde el grupo necesita más apoyo, los profesores.
- El desarrollo de aplicaciones en general. El sistema es útil sobre todo desde el punto de vista de los desarrolladores. Se trata de obtener un código, no sólo que cumpla los requisitos funcionales, sino también, que sea fácilmente modificable y mantenible. Por tanto, podría ser usado por desarrolladores que tengan conocimientos avanzados de programación; pero quieran mejorar la calidad de su código fuente. Para ello, el sistema permite realizar un análisis más profundo y proporcionar las pautas de mejora.

El sistema de análisis es lo suficientemente flexible para que pueda ser ampliado sin necesidad de tener que hacer cambios en el modelo existente. El sistema está preparado para la ampliación de las herramientas registradas, ya sean compiladores o otras herramientas de detección de errores. También es sencillo crear nuevas Unidades de Compilación que amplíen la funcionalidad del sistema.

El sistema de generación de estadísticas al igual que el anterior puede ser ampliado fácilmente. Usando la tecnología XML-Schema el sistema puede ser ampliado y modificado sin que sea necesaria la modificación del núcleo del sistema.

Todas estas características hacen que este subsistema sea potente y flexible; pero para conseguir toda la funcionalidad de SICODE necesitamos una gestión de grupos y

colaboración sobre los ficheros del proyecto que proporcionará el subsistema COLLDEV que veremos a continuación en el Capítulo 8 y un entorno para la escritura de los programas y que proporcione una base de conocimientos como es IDEWeb que veremos en el Capítulo 9.

Capítulo 8. Sistema para la colaboración en el desarrollo de aplicaciones: COLLDEV

En este capítulo describimos el prototipo de Sistema para la Colaboración en el Desarrollo de Aplicaciones (COLLDEV, Collaborative Development). Este prototipo trata de desarrollar un Sistema Colaborativo que gestione los grupos de trabajo para el desarrollo de aplicaciones de forma asíncrona a través de la Web, permitiendo que se compartan los archivos fuente del proyecto en desarrollo, y que proporcione, a su vez, una "historia activa" del trabajo realizado hasta el momento; para permitir el seguimiento del proceso de desarrollo. Este es uno de los subsistemas que junto con PBA e IDEWeb forman el sistema SICODE.

8.1 Objetivos

COLLDEV [Pérez, 2004b] es un sistema colaborativo orientado al aprendizaje (Computer Supported Collaborative Learning, CSCL) vía Web, que facilita la coordinación y colaboración de grupos de trabajo para trabajar sobre un proyecto software y proporciona una *"historia activa"* del trabajo realizado por los miembros del grupo.

Los sistemas colaborativos tratan de encontrar un modelo que englobe los distintos tipos de participantes, las tareas a realizar y los modos de colaboración, siendo tema principal de gran número de investigaciones y artículos. Además presentan una gran funcionalidad y basan su arquitectura en cuatro principios básicos citados por Barros [Barros 2000]:

- Construcción conjunta de la solución a un problema siguiendo algún método de colaboración.
- Coordinación de los miembros del grupo para organizar el trabajo.
- Semiestructuración de los mecanismos que soportan la discusión argumentativa y la consecución de acuerdos entre los participantes.
- Interés tanto en el proceso de aprendizaje como en el resultado del trabajo en grupo, y por tanto, representación explícita de los procesos de producción e integración.

A partir de estos principios, se ha desarrollado un sistema CSCL, enfocado al desarrollo de proyectos software en grupo, que intenta solucionar las barreras con las que se encuentran los estudiantes de programación o desarrolladores en general, en este tipo de tareas. Es el Sistema para la Colaboración en el Desarrollo de Aplicaciones, cuyas siglas del inglés *Collaborative Development* son COLLDEV.

Con este sistema se ha tratado de resolver los inconvenientes que surgen a la hora de trabajar en grupo de forma presencial para desarrollar un proyecto software, como es la

distancia física que puede separar a los desarrolladores, y la incompatibilidad entre los horarios de trabajo.

8.1.1 Facilitar la comunicación y la colaboración en el desarrollo de software

El principal objetivo es permitir el desarrollo de software en colaboración por parte de un grupo de usuarios, siendo, por tanto, un objetivo muy importante a cumplir por este sistema, facilitar la comunicación y el trabajo colaborativo entre desarrolladores, y de esta forma permitir que los miembros del grupo colaboren en el objetivo común para hacer más eficiente el trabajo de todos. Este objetivo tiene relación con el primer y segundo principios de los sistemas CSCL citados anteriormente.

Para ello, se ha creado una aplicación que, de forma fácil, permitiese a los desarrolladores compartir los archivos fuente, permitiendo el trabajo en paralelo sobre los archivos del proyecto, a través de un espacio de trabajo compartido, y facilitar la comunicación, tanto entre ellos como con su supervisor, que será el encargado de dirigir el proyecto, mediante el envío de mensajes asíncronos, sin tener que recurrir a otros sistemas de comunicación más genéricos y menos eficientes. Esta comunicación es particularmente interesante en el planteamiento de tareas, la resolución de dudas y la revisión de código.

El objetivo del trabajo en paralelo sobre los archivos del proyecto consiste en que dos o más desarrolladores puedan modificar a la vez archivos pertenecientes al mismo proyecto, incluso el mismo archivo y que sea el propio sistema el que se encargue de juntar automáticamente y de forma asíncrona las modificaciones. Esto es viable y relativamente sencillo cuando los cambios no coinciden en la misma localización del archivo; cuando existe coincidencia habrá que habilitar un sistema de resolución de conflictos de código donde serán los desarrolladores los que manualmente hagan la mezcla de las modificaciones que entran en conflicto.

8.1.2 Seguimiento continuo del proceso de construcción de software

El segundo objetivo a cumplir por este sistema, y no menos importante, es permitir al supervisor un seguimiento continuo del proyecto, basado en la "historia activa", es decir, en las distintas versiones que se van generando de los archivos fuente del proyecto cuando los desarrolladores realizan las distintas tareas que se le plantean. La "historia activa" también debe facilitar la realización de comentarios sobre el trabajo realizado por los desarrolladores. Esto permite centrar el interés tanto en el proceso de desarrollo como en el resultado final del trabajo en grupo, cuarto principio de los sistemas CSCL.

Además de los comentarios realizados sobre los trabajos de los desarrolladores, los *supervisores* podrán responder, a las dudas planteadas por los desarrolladores e incluso colaborar en el desarrollo mediante el envío de algún documento o archivo.

8.1.3 Ayuda a la toma de decisiones en grupo

Un tercer objetivo perseguido en la realización de esta aplicación, es ayudar a los desarrolladores en la consecución de acuerdos, por lo que se pone a su disposición un sistema de encuestas que los propios desarrolladores irán creando cuando necesiten tomar alguna decisión, del tipo que sea, cumpliendo así el tercer principio de los sistemas CSCL.

8.2 Ámbito del prototipo y relación con el resto de prototipos

El sistema COLLDEV proporciona toda la funcionalidad relativa a la gestión de grupos de trabajo y almacenamiento compartido de los proyectos. Permite asignar uno o más proyectos a un grupo, refleja que miembros han trabajado sobre los ficheros de un proyecto y permite seguir la evolución de este proyecto a través de sus distintas versiones. Delega en IDEWeb para realizar la creación, edición y modificación de nuevos ficheros de un proyecto y para gestionar los errores y su corrección. Por otra parte, este sistema tampoco se encarga de la compilación y análisis de errores de los ficheros fuente del proyecto que realizará el subsistema PBA, sólo permite la gestión de las distintas versiones del proyecto.

8.3 Requisitos del prototipo

8.3.1 Gestión de roles, grupos y tareas

El prototipo distinguirá tres tipos usuarios.

- **Desarrolladores**, los cuales realizarán las tareas de desarrollo de forma individual o en grupos; y podrán editar, y por lo tanto, modificar, los archivos del proyecto almacenados en el repositorio de su grupo y revisar las versiones anteriores, así como comunicarse con los demás usuarios del sistema y leer los comentarios realizados por supervisores y otros desarrolladores en las revisiones del código fuente.
- **Supervisores**, su principal función (aunque no exclusiva, ya que pueden realizarla también desarrolladores) es revisar el código fuente realizado por sus desarrolladores, teniendo la posibilidad de realizar comentarios sobre el mismo, así como de responder a las dudas expuestas por los desarrolladores y proporcionar la ayuda que sea necesaria.
- **Administrador**, que realiza las tareas de un supervisor y además, hará las veces de administrador de la aplicación, aunque uno de los objetivos de esta aplicación es reducir las tareas que tiene que llevar a cabo este para no centralizar el trabajo en un usuario, siendo la mayoría de ellas automatizadas.

Usuarios

La forma de acceder a la aplicación será la misma para todos los tipos de usuarios, es decir, en cualquier caso, deberán introducir un nombre de usuario y una contraseña secreta que los identifique de forma única, para poder realizar esto es condición necesaria que el usuario se haya registrado previamente en el sistema, posteriormente el sistema muestra la información correspondiente al usuario que ha entrado en sesión.

El alta de un usuario desarrollador en el sistema se realiza de la siguiente forma: él mismo registra sus datos en el sistema, para lo cual deberá rellenar un formulario público con los datos necesarios (nombre, apellidos,…); esta acción es notificada al usuario administrador, que autoriza o deniega la incorporación del usuario al sistema.

Cada usuario supervisor podrá acceder a los archivos de sus desarrolladores para revisarlos y realizar los comentarios que sean oportunos, así como proveer a los desarrolladores de las especificaciones de las tareas a realizar, pudiendo aportar archivos, que serán expuestos en un espacio de trabajo compartido de los desarrolladores correspondientes. El usuario

administrador deberá llevar a cabo también las tareas propias de un administrador en lo que concierne a la gestión de usuarios (autorización de usuarios nuevos, creación de reglas para los grupos de trabajo, creación de grupos de trabajo, etc.).

Grupos de trabajo

Para poder trabajar en el sistema, un desarrollador deberá de elegir un grupo de trabajo al que incorporarse de entre los existentes en el sistema. Esta inscripción se verificará automáticamente mediante reglas. Una vez que el desarrollador se inscriba en un grupo se notificará al administrador, para que este tenga notificación de los usuarios inscritos en los grupos. La notificación sólo se produce cuando se ha incorporado a un grupo al cumplir las reglas preestablecidas, de otra forma el usuario es notificado en el momento que no puede darse de alta al incumplir una de las reglas.

El sistema permite la definición de reglas para la verificación automática de la correcta creación de nuevos grupos de trabajo. Estas son gestionadas por el administrador. Todo grupo existente en la aplicación deberá cumplir estas reglas. Ejemplos de estas reglas son: cada grupo de trabajo debe tener asignado algún supervisor, cada grupo de trabajo tendrá un número mínimo y máximo de desarrolladores inscritos a él, cada desarrollador sólo podrá aparecer inscrito en un grupo de trabajo.

Los grupos de trabajo serán creados a partir de las peticiones de los desarrolladores a la hora de formarlos. El sistema también permite la incorporación de un desarrollador a un grupo de trabajo o el cambio de un desarrollador de un grupo de trabajo a otro. Para realizar todas estas operaciones se sigue el mismo mecanismo: son los propios usuarios los que desencadenan la acción que se verifica automáticamente mediante las reglas de verificación definidas a priori y el administrador recibirá la notificación de la operación realizada.

Estos grupos poseerán un espacio de trabajo colaborativo (identificado por el nombre del grupo) en el cual los desarrolladores irán escribiendo y probando los archivos fuente en relación con las tareas que tienen asignadas; estos archivos quedarán disponibles para el resto del grupo, así como para el supervisor que tengan asignado, cuando se vayan finalizando las tareas se realizará un nuevo reparto de tareas en el grupo.

El proceso de inscripción de los supervisores a los grupos es el mismo que el de los desarrolladores, el supervisor hace la petición que tiene que cumplir las reglas establecidas y se notificará al administrador.

8.3.2 Espacio compartido de colaboración - espacio personal

Cada grupo tiene un espacio de trabajo colaborativo asociado. En este espacio se mantendrán los archivos correspondientes a los proyectos que el grupo está desarrollando. Se pueden crear nuevos proyectos en el espacio de trabajo colaborativo para que todo el grupo pueda trabajar sobre ellos. Cada vez que un usuario quiera trabajar sobre un archivo almacenado en el espacio compartido de su grupo, primero se realizará una copia del mismo en el espacio de trabajo personal, sobre esta copia se realizará el trabajo correspondiente y una vez finalizado, el archivo se enviará de nuevo al espacio de trabajo colaborativo combinando los cambios realizados.

Cada usuario tendrá a su disposición un espacio de trabajo personal, que será creado al entrar en el sistema como usuario registrado. A este espacio personal sólo tendrá acceso el propio usuario. Este espacio de trabajo será identificado por el nombre del usuario y se utilizará para el trabajo individual que desarrolle el usuario.

Tanto el espacio compartido de colaboración, como el espacio de trabajo personal estarán situados en el servidor, con lo que estarán siempre disponibles independientemente del lugar desde el que se acceda. Los usuarios podrán intercambiar archivos entre su espacio de trabajo personal (en el servidor) y el sistema local.

Cada desarrollador podrá editar los archivos existentes correspondientes a un determinado proyecto y guardar de nuevo los cambios realizados. Para poder realizar un seguimiento se guardan distintas versiones del archivo a medida que los usuarios lo van modificando.

8.3.3 Navegación a través de las versiones

La historia activa permite una navegación entre las diferentes versiones que se han ido generando de un trabajo, artículo, archivo, etc. a lo largo de su proceso de creación, permitiendo la introducción de comentarios y sugerencias en cualquiera de las versiones.

En nuestro caso, la historia activa hará referencia a las distintas versiones que los desarrolladores han ido generando de los archivos para la realización del proyecto, las cuales ayudarán a los supervisores a llevar un seguimiento de la evolución del trabajo de sus desarrolladores.

Se podrán realizar revisiones sobre los archivos, y todas sus versiones, existentes en su espacio de trabajo colaborativo del grupo al que pertenece y anexar comentarios a los mismos. Cada usuario podrá revisar: qué miembro del grupo hizo determinados cambios y cuáles fueron estos; todo esto en cada una de las versiones de un archivo; de esta forma, podrá ponerse en contacto con él para hacer comentarios o pedir aclaraciones sobre un fragmento de código.

8.3.4 Comunicación, coordinación y toma de decisiones conjunta por parte de los usuarios

En lo que se refiere a la comunicación asíncrona entre desarrolladores y supervisores o entre múltiples desarrolladores, se distinguen tres tipos de mensajes: "preguntas y respuestas", comentarios y tareas.

- Mensajes de tipo preguntas y respuestas. Este tipo de mensajes, como su propio nombre indica, pueden contener preguntas sobre cualquier tipo de duda, o su respuesta. Pueden ser enviados y recibidos tanto por supervisores como desarrolladores y su finalidad es establecer una comunicación para facilitar la resolución de las dudas que puedan surgir. Ante este tipo de mensaje, el receptor podrá enviar otro al emisor del mismo, como respuesta a sus dudas o planteando otra pregunta relacionada con la anterior.

- Mensajes de tipo comentarios. Estos mensajes contienen comentarios (sugerencias, errores, etc.) realizados, bien por supervisores, bien por desarrolladores, que surgen tras la revisión de algún archivo de otro desarrollador. Además, deberán contener también, el nombre del archivo al que hacen referencia. En el caso de los desarrolladores, estos sólo podrán realizar comentarios sobre los archivos que se encuentran en el espacio de trabajo colaborativo de su grupo, ya que al resto no tiene acceso. Este tipo de mensajes también puede ser enviados de forma automática por el sistema, indicando el registro de un nuevo usuario, la creación de un nuevo grupo, etc. Ante este tipo de mensaje el receptor puede responder con mensajes, matizando los comentarios, o dar solución al error o problema notificado.

- Mensajes de tipo tarea. En estos mensajes unos usuarios indican a otros las tareas que han de realizar, y en el caso de los supervisores podrán ir acompañados de archivos que han depositado en el espacio de trabajo de sus desarrolladores. Ante estos mensajes el receptor podría responder con la aceptación o el rechazo de una determinada tarea.

Todos los mensajes deberán ser accesibles cuando el usuario receptor de los mismos esté trabajando con la aplicación, visualizándose cuando son recibidos en un espacio de la aplicación dedicado especialmente para ello y donde se ubicarán los mensajes más recientes. De esta forma la transmisión de mensajes entre usuarios es casi instantanea. También se podrá acceder al resto de los mensajes recibidos con anterioridad a petición del usuario. Los mensajes leídos deberán ser distinguidos de aquellos que el usuario aun no ha leído o marcado como tal.

Otra herramienta del sistema COLLDEV pensada para la coordinación es la realización de votaciones o encuestas que ayudará a los desarrolladores a ponerse de acuerdo en determinadas decisiones. Estas siempre se llevarán a cabo entre los miembros integrantes de un grupo. Cualquier miembro que desee someter a consulta una decisión puede crear una encuesta, proponiendo una pregunta y una serie abierta de alternativas. Cuando se crea una encuesta, el resto de desarrolladores del grupo recibirán un mensaje de tipo Tarea. Todos los miembros del grupo pueden votar en la encuesta, en caso de no estar de acuerdo con las alternativas propuestas en la encuesta, puede aportar su propia alternativa, a la cual le dará su voto. Una encuesta será evaluada cuando todos los miembros del grupo hayan votado. Se pueden consultar los resultados obtenidos tras las encuestas, ya que quedarán almacenados.

8.4 Descripción de Actores y casos de uso

En este apartado describimos cada uno de los actores del sistema, además de la definición, se explica, las operaciones que puede realizar cada uno. Y los casos de uso dando una breve descripción de lo que comprenden.

8.4.1 Actores

Supervisor. Usuario encargado del reparto de tareas en el grupo y la revisión del trabajo realizado por los desarrolladores. Puede enviar cualquier tipo de mensajes.

Administrador. Encargado de realizar las tareas del administrador de un sistema. Recibe la notificación del registro de nuevos usuarios. Es el único que puede modificar cualquier tipo dato de cualquier usuario. Es el encargado de crear las reglas a cumplir por los grupos de trabajo. Recibe la notificación de la creación de nuevos grupos. Recibe la notificación de la incorporación de usuarios a los grupos. Indica a los desarrolladores si han de cambiarse a otro grupo, si se elimina el actual.

Desarrollador. Usuario que interacciona con la aplicación para realizar el desarrollo de la aplicación colaborando con otros. Puede realizar cualquier operación sobre sus archivos. Recibe y envía mensajes de preguntas y respuestas e, incluso de tareas. Recibe mensajes de comentarios. Tiene acceso a los archivos de su grupo. No tendrá acceso a la información (cuentas y archivos) de otros usuarios del sistema.

8.4.2 Casos de uso

Gestión de usuarios y grupos. Es una función exclusiva del administrador. Autorización de registro de un usuario. Modificación y consulta de usuarios. Creación de reglas para la creación de grupos, creación y eliminación de grupos de trabajo.

Administración de proyectos y espacios de trabajo. Acceso a los archivos de los proyectos de todos los grupos, configuración de los repositorios de versiones.

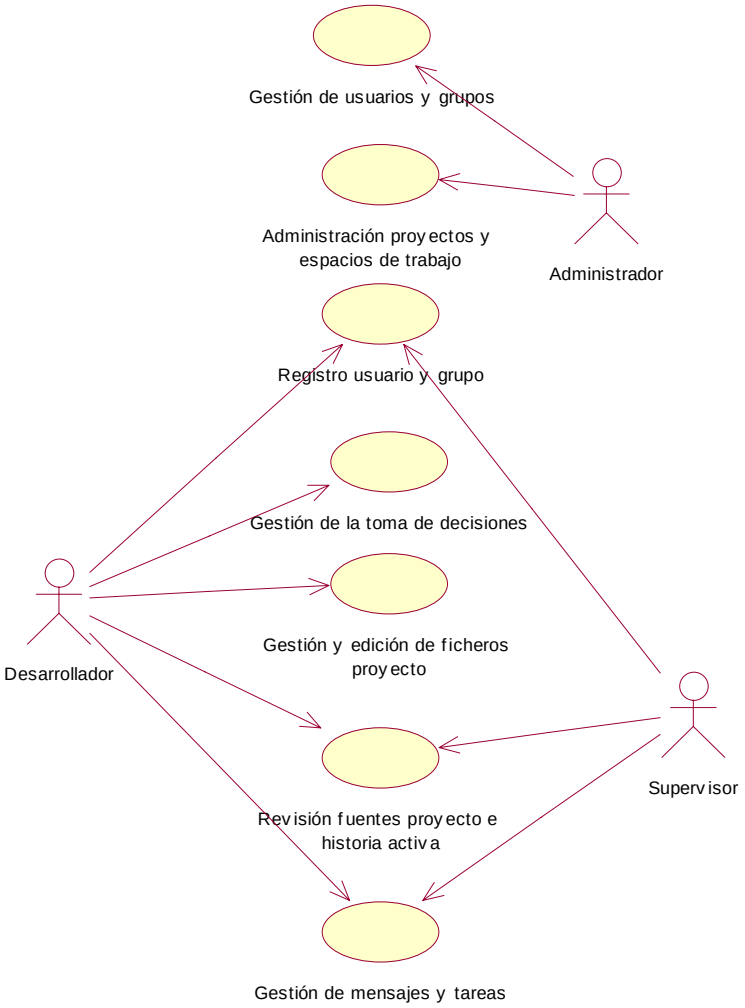


Figura 49. Diagrama de casos de uso del prototipo de colaboración

Registro de usuario y grupo. Tanto desarrolladores como supervisores pueden acceder a este caso de uso. Registro pro-activo (por parte del propio usuario) de un usuario en la aplicación. Incorporación de usuarios a los mismos siguiendo un modelo pro-activo.

Gestión y edición de archivos del proyecto. Creación, edición y borrado de archivos de un proyecto. Gestión del intercambio de archivos entre los dos espacios y con el exterior, generación de versiones para permitir un seguimiento mediante la historia activa.

Revisión de fuentes del proyecto e historia activa. Visualización, revisión y realización de comentarios sobre los archivos fuente. Acceso a versiones anteriores de los archivos permitiendo seguir el proceso de construcción del proyecto.

Gestión de mensajes y tareas. Envío y recepción de mensajes entre usuarios. Generación automática de mensajes asociados a determinadas tareas.

Gestión de toma de decisiones. Creación y gestión de encuestas para ayudar a la toma de decisiones.

8.5 Diseño

8.5.1 Diseño del espacio de trabajo compartido basado en un sistema concurrente de versiones (CVS)

Uno de los puntos clave del sistema, como se ha dicho en el análisis de requisitos, es el establecimiento de un espacio compartido para facilitar la colaboración de los desarrolladores, en la escritura de código, en la revisión, en la corrección de errores o en la mejora del diseño.

Para obtener un espacio de trabajo colaborativo entre un grupo de desarrolladores se ha utilizado un sistema concurrente de versiones (CVS) [Cederqvist 1993], que, como su propio nombre indica, es un sistema de control de versiones automático para facilitar el trabajo en grupos desarrollo.

CVS centra su misión en llevar a cabo el control de versiones de los archivos de un proyecto, que es realizado por un grupo de desarrolladores, facilitando el control de cambios realizados en el mismo por los distintos componentes del grupo de desarrollo. No tiene ninguna faceta de comunicación con lo que no aporta características para el envío de mensajes o la comunicación directa entre usuarios.

Una de las características más destacadas que aporta CVS es que no bloquea los archivos que están siendo modificados. El usuario que desea acceder a un archivo debe realizar una copia local del archivo original, que se encuentra en el repositorio, que es "la copia maestra en la que CVS guarda el historial de revisiones al completo efectuadas en un proyecto". Sobre esta copia local el desarrollador realizará las modificaciones oportunas, sin entorpecer el trabajo de los demás, ya que cualquier otro desarrollador, de forma simultánea, también puede hacer una copia del archivo original y ponerse a modificarlo. Una vez hechos los cambios, esta copia, será enviada de nuevo al repositorio, dando lugar a una nueva versión del archivo, que CVS se encargará de combinar con la copia maestra de forma automática.

Al hacer esta actualización de la copia maestra pueden suceder tres casos distintos:

1. Que la actualización se realice sin ningún tipo de problemas, porque la copia maestra no haya cambiado.
2. Que mientras un desarrollador ha estado realizando modificaciones otro haya enviado sus cambios al repositorio. Y los cambios hayan sido realizados en fragmentos distintos del archivo, con lo que CVS será capaz de combinar ambos cambios de forma automática.
3. Que mientras un desarrollador ha estado realizando modificaciones otro haya enviado sus cambios al repositorio. Y que estos cambios hayan sido realizados en el mismo fragmento del archivo por los dos desarrolladores. CVS detectará un conflicto, que deberán resolver los desarrolladores manualmente.

Antes de comenzar a trabajar con la copia local, es importante actualizarla, por si se han producido modificaciones en el repositorio que nuestra copia actual aun no recogía. De

este modo, los usuarios podrán realizar su trabajo reduciendo el riesgo de aparición de conflictos.

Uno de los inconvenientes de CVS es que no es posible crear archivo en el repositorio a partir de la nada. Para crear un nuevo archivo en el repositorio, este deberá ser importado de algún directorio local.

CVS permite realizar todas estas operaciones mediante instrucciones para ejecutar en línea de comandos. Las instrucciones más utilizadas son:

- **Update.** Realiza una actualización de la copia de trabajo a partir del repositorio. Es la primera operación que debe realizar un desarrollador a la hora de ponerse a modificar un archivo del proyecto; para actualizar la versión que tiene del mismo con los cambios realizados por el resto de los desarrolladores.
- **Dic.** Compara las copias de trabajo con sus homónimos en el repositorio. Y muestra gráficamente los cambios entre un archivo y otro.
- **Commit**: Envía las modificaciones al repositorio. Cuando el desarrollador ha finalizado los cambios y ha comprobado que todo funciona correctamente, realiza esta operación para que el resto del grupo pueda acceder a sus cambios.
- **Import**: Importa un nuevo proyecto al repositorio. Hay que asegurarse de que ese directorio sólo está lo que realmente que realmente queremos copiar.
- **Checkout**: Obtiene una copia de trabajo del archivo o proyecto indicado, para poder comenzar a trabajar.

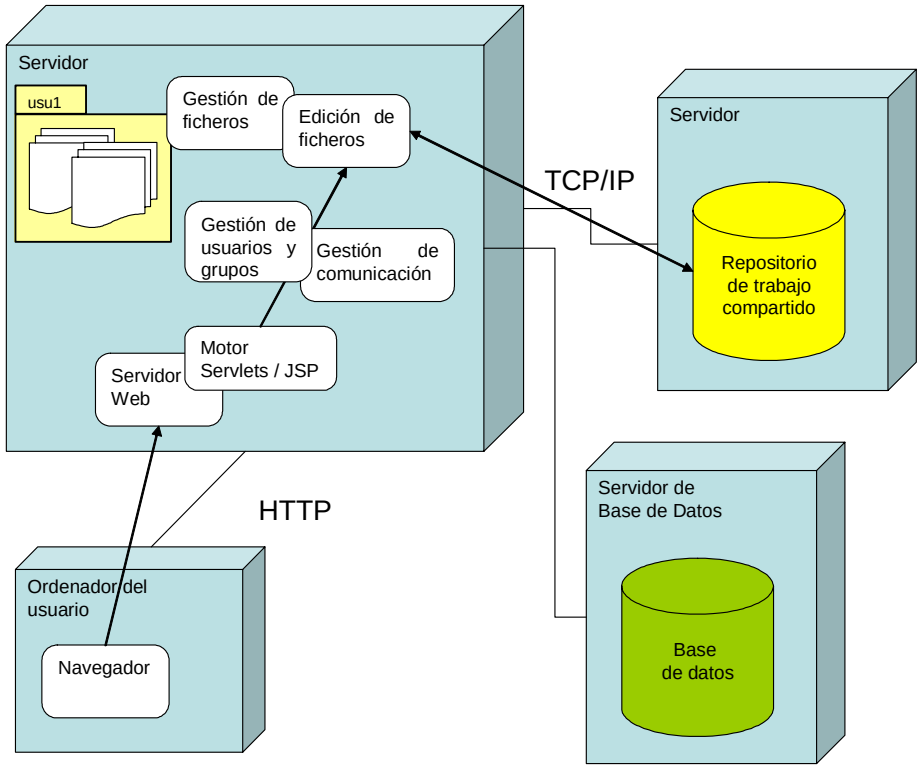


Figura 50. Arquitectura del sistema

Estos comandos tienen diferentes opciones que permiten flexibilizar su funcionalidad.

Para gestionar el espacio de trabajo colaborativo de COLLDEV utilizamos CVS, ya que ofrece un control de versiones automático, y esto nos facilitará la creación de la historia activa necesaria para realizar un seguimiento del progreso de los desarrolladores dentro un grupo de trabajo. Sobre este CVS de base añadimos una funcionalidad extra que automatice algunas de las tareas que tendrían que hacer los desarrolladores a mano; además también de forma automática agrupamos las versiones de diferentes ficheros de forma automática para conseguir versiones del proyecto completo para poder examinar todos los ficheros en conjunto.

8.5.2 Diseño de la arquitectura de la aplicación

COLLDEV es una aplicación Web basada en una arquitectura en la que toda la carga de la aplicación recae sobre el servidor, y el ordenador cliente únicamente debe ejecutar un navegador estándar que hará de interfaz con el desarrollador (ver Figura 50).

El espacio de trabajo compartido se encuentra en un servidor que gestiona el acceso concurrente a los archivos, al cual se accede mediante un cliente CVS, y en él se almacena el conjunto de archivos de los proyectos correspondientes a cada grupo de trabajo al que sólo pueden acceder los desarrolladores que lo componen y los supervisores que lo supervisan.

El espacio de trabajo personal contiene un directorio, personal y restringido, por cada usuario registrado en el sistema, por lo que cada usuario sólo tendrá acceso a su directorio, y nunca al del resto se sus compañeros.

Por otro lado, en la base de datos se encuentra toda la información relativa a usuarios, grupos y mensajes, que la aplicación necesita para funcionar correctamente.

8.5.3 Diseño de la interfaz

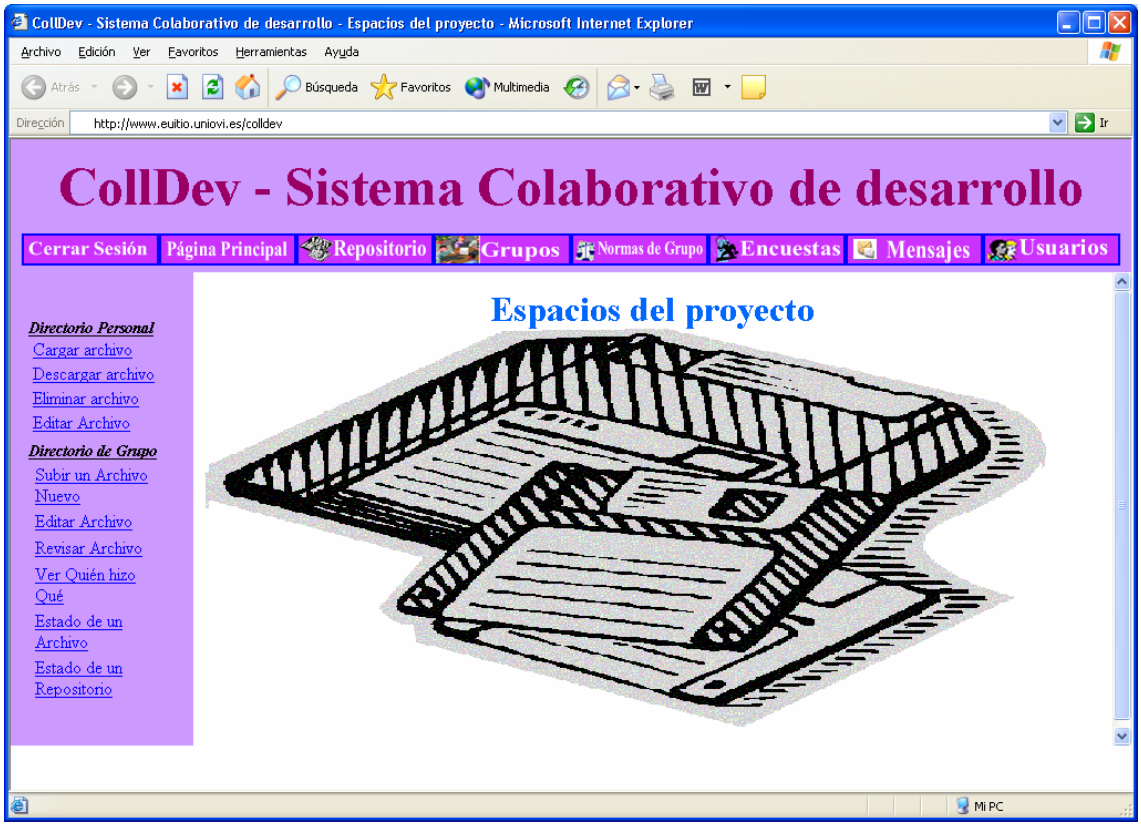


Figura 51. Página de espacios del proyecto del sistema colaborativo de desarrollo

En la Figura 51 se puede ver la página que permite acceder a la funcionalidad de espacios del proyecto, tanto el personal como el compartido. Todas las páginas de la aplicación siguen la misma estructura, analizaremos esta estructura general de la interfaz del sistema sobre esta página.

Los elementos que componen una página en el sistema COLLDEV son:

- Cabecera con el nombre del sistema, identifica claramente la aplicación.
- Menú horizontal (Figura 52), siempre está visible. Permite acceder en todo momento a toda la funcionalidad del sistema.
- Menú de la opción, menú vertical en el que se recoge todas las funciones de la opción elegida, en este caso "Espacios del proyecto".
- Área principal de interacción, se encuentra a la derecha del menú de la opción y es el área más amplia de la página, permite presentar al usuario información relacionada con la opción que ha seleccionado; también permite mostrar los formularios para recoger datos adicionales.
- Área de avisos del sistema, la parte inferior se reserva a avisos del sistema al usuario, informándole de operaciones erróneas o cualquier otra información útil en la interacción.

Figura 52. Barra horizontal con el menú principal de la aplicación

8.5.4 Modelado de datos

La base de datos es utilizada para recoger información de forma persistente sobre los distintos usuarios, grupos, mensajes y encuestas que tienen lugar dentro del sistema, así como de las relaciones creadas entre ellos. Además, la base de datos permite compartir información con los otros dos subsistemas: PBA e IDEWeb, con los cuales se formará el sistema SICODE que, entre otras cosas, facilita la realización de proyectos software en grupo.

Una parte común a todos los subsistemas es la entidad Usuario, la cual representa a todos los usuarios, reales o potenciales, del sistema y permite autenticar a los usuarios que quieren iniciar una sesión en el sistema. La entidad Estadistica, encargada de almacenar datos para la obtención de estadísticas de errores de los usuarios, es el punto de unión con el subsistema PBA (Capítulo 7) que se encarga de esta funcionalidad.

Usuarios y grupos

Entidad Usuario. Almacena toda la información que el sistema necesita en relación a los usuarios, en general. Permite utilizar estos datos en el proceso de autenticación.

Entidades Desarrollador, Supervisor, Administrador. Son distintos tipos de usuario. Almacenan la información asociada a cada uno de estos tipos.

Entidad Grupo. Almacena información relevante a cerca de los grupos de trabajo formados por los desarrolladores y autorizados por supervisores.

Entidad NormaGrupo. Almacena información a cerca de las normas que han de cumplir los grupos de trabajo en su composición. Esta información se utiliza para que el sistema verifique automáticamente la formación de grupos.

Relación Pertenece Grupo. Representa la relación existente entre un desarrollador y un grupo de trabajo. Un grupo podrá tener varios desarrolladores, pero un desarrollador únicamente podrá pertenecer a un grupo.

Relación Tutoriza Grupo. Representa la relación existente entre un grupo y un supervisor. Un supervisor podrá tutorizar muchos grupos, y un grupo podrá ser autorizado por varios supervisores.

Relación CumpleNormas. Representa la relación existente entre un grupo y la norma que cumple. Un grupo podrá cumplir muchas normas, y una norma podrá ser cumplida por varios grupos.

Relación Crea. Representa la relación existente entre una norma y el administrador que la crea. Debido a que en el sistema sólo existe un único administrador, esta relación carece de representación en forma de tabla, tratándose únicamente de una característica funcional.

Mensajes

Entidad Mensaje. Almacena información relativa a los mensajes enviados entre los diferentes usuario, incluido por el sistema.

Relación Envia. Representa la relación de envío de mensajes existente entre un usuario y el mensaje que envía. Un usuario podrá enviar muchos mensajes, pero un mensaje únicamente podrá ser enviado por un usuario.

Relación Recibe. Representa la relación de recepción de mensajes existente entre un usuario y el mensaje que recibe. Un usuario podrá recibir muchos mensajes, y un mensaje podrá ser recibido por varios usuarios.

Encuestas

Entidad Encuesta. Almacena datos de las encuestas que los desarrolladores realizan para poder llegar a acuerdos.

Entidad Idea. Almacena datos de las sugerencias que han ido surgiendo en las encuestas. Tiene una relación de dependencia por existencia con la entidad encuesta.

Relación CreaEncuesta. Representa la relación existente entre un grupo y la encuesta creada por uno de sus miembros. Un grupo podrá crear muchas encuestas, pero una encuesta sólo podrá ser creada por un grupo.

Relación Voto. Representa la relación existente entre un desarrollador y la encuesta en la cual ha participado con su voto. Un desarrollador podrá votar en muchas encuestas, y una encuesta podrá ser votada por muchos desarrolladores.

Relación Contiene Idea. Representa la relación existente entre una encuesta y sus ideas o sugerencias. Una encuesta podrá contener muchas ideas, pero una idea sólo podrá pertenecer a una encuesta.

Versiones de archivos y estadísticas

Entidad VersionArchivo. Almacena datos referentes a una versión de un archivo del espacio de trabajo colaborativo.

Relación CreaVersion. Representa la relación existente entre una versión de un archivo y el desarrollador que la creo. Un desarrollador podrá crear muchas versiones de un archivo, pero una versión sólo podrá ser creada por un desarrollador.

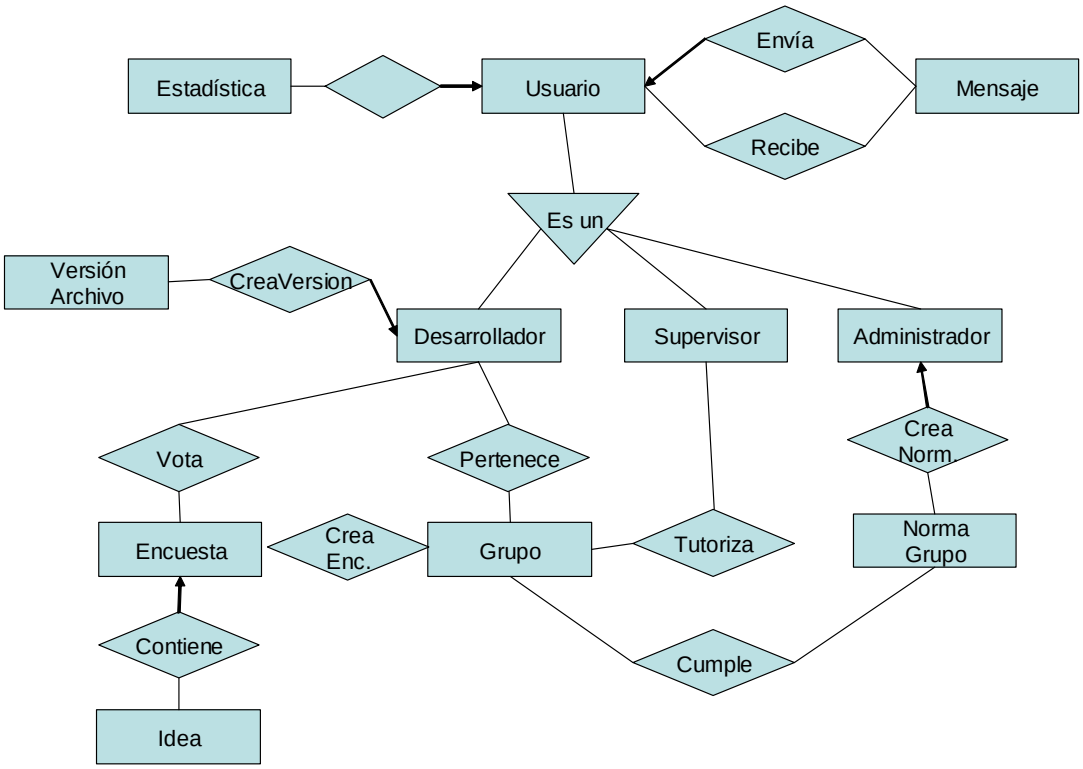


Figura 53. Diagrama entidad - relación del sistema

Entidad Estadística. Almacena datos que pueden ser de utilidad a la hora de hacer estadísticas sobre la actuación del usuario, sirviendo de enlace entre este proyecto y otros. Tiene una relación de dependencia por existencia con la entidad Usuario. En este diagrama (Figura 53) toda la información estadística está representada por una entidad; sin embargo, realmente hay un conjunto de entidades que permiten almacenar esta información; para una información más detallada ir al apartado correspondiente del subsistema PBA (apartado 7.8.5. Modelado de datos del Capítulo 7).

8.6 Limitaciones del prototipo

El prototipo implementa una historia activa de los proyectos en desarrollo permitiendo la revisión de cualquier versión de un archivo y compararlo con otra versiones para comprobar cuales han sido los cambios; sin embargo la navegación a través de la versiones del proyecto y se carece de una visión general de la evolución del proyecto, por lo que un supervisor tendrá que invertir bastante tiempo para tener una idea clara de la evolución de un proyecto.

Actualmente el sistema permite el intercambio de distintos tipos de mensajes entre los usuarios; sin embargo la construcción de los mensajes está poco automatiza, por ejemplo en los comentarios sobre un error debemos escribir el comentario y luego adjuntar el fichero que contiene el error; el sistema debería crear automáticamente el mensaje cuando el usuario marcara un error en área de edición del archivo fuente.

8.7 Conclusiones

En la introducción de este documento se señalaban tres objetivos clave. El resultado final, tras la realización del sistema COLLDEV, ha sido el siguiente:

- **Permitir la comunicación y el trabajo colaborativo entre desarrolladores y supervisores, para facilitar el desarrollo colaborativo de software.** El sistema permite la comunicación mediante mensajes especializados según su función. Permite compartir y colaborar en la escritura de los archivos del proyecto a través de un espacio de trabajo colaborativo basado en un sistema de control de versiones.
- **Permitir al supervisor un seguimiento continuo de sus desarrolladores basado en la "historia activa".** Gestión automática de las versiones de los archivos del proyecto en el espacio de trabajo colaborativo y navegación a través de esas versiones facilitando la revisión y el seguimiento del proceso de desarrollo de una aplicación.
- **Ayudar a los desarrolladores en la toma de decisiones.** Creación y gestión de encuestas que permiten recoger planteamientos y elegir uno de ellos por mayoría.

De esta manera, se consigue disminuir los obstáculos a los que se enfrentan los desarrolladores a la hora de realizar sus proyectos software en grupo permitiendo una obtención de aplicaciones con una mayor calidad de código.

En el siguiente capítulo veremos el prototipo para el último subsistema de los que componen el sistema SICODE.

Capítulo 9. Entorno de desarrollo integrado en Web y base de conocimientos colaborativa: IDEWeb

En este capítulo abordamos el subsistema de SICODE que constituye el entorno de desarrollo integrado en Web (IDEWeb). Junto con los subsistemas PBA y COLLDEV (analizados en los capítulos anteriores) constituyen el sistema completo SICODE. Este entorno permitirá la edición de código y permitirá la invocación al compilador y las herramientas de análisis estático (subsistema PBA). Por otra parte, permite trabajar con los mensajes de error de varias formas: muestra los errores y facilita su corrección mostrando el archivo fuente donde se localizó, muestra avisos en base a las estadísticas realizadas por PBA para prevenir la repetición de errores y sobre todo proporciona enlaces a la base de conocimientos colaborativa que permitirá obtener mayor información sobre el error para solucionarlo y aprender a evitarlo.

9.1 Planteamiento general

El prototipo IDEWeb (Integrated Development Environment on Web) trata de conseguir un entorno de desarrollo que funcione a través de Web. El objetivo es mejorar las destrezas de los desarrolladores en la comprensión y solución de los errores y el aprendizaje para evitarlos en futuras ocasiones, con ello se consigue la mejora de la calidad del código fuente.

Cuando un programador tiene poca experiencia en un lenguaje de programación comete errores que aunque simples, no todos son de obvia solución, ya que sin esa experiencia es difícil relacionar el mensaje de un error con su causa. Algunas veces, en estos casos, una persona con conocimientos le proporciona algún patrón para solucionar el error; o bien el propio programador invierte tiempo hasta que encuentra la causa. Ese mismo error puede sucederle a otros desarrolladores. La experiencia acumulada por una persona que ha programado mucho en un lenguaje actualmente está desaprovechada, solo sirve para esa misma persona. Si conseguimos que el entorno en el que trabaja pueda guardar esas experiencias y compartirlas con el resto de desarrolladores, se aprovecharán las experiencias individuales de cada uno de ellos, resultando una adquisición de experiencia acelerada para todos los usuarios y una mejora en la calidad del código más rápida que si cada programador trabajase por su cuenta.

El entorno debe permitir la edición de código, y proporcionar ayudas en la edición por ejemplo el resaltado de sintaxis

Hemos decidido realizar el entorno de desarrollo para que funcionase en la Web por varios motivos:

- Al funcionar sobre un navegador Web estándar, el usuario final puede ponerse a programar sin necesidad de instalar nada, simplemente debe arrancar su navegador preferido y conectarse a la URL de IDEWeb, con esto evitamos el paso previo de instalación y configuración del entorno que siempre requiere emplear un tiempo y a veces conlleva también complicaciones.
- Una arquitectura Web nos permite centralizar, de forma sencilla, la información que se recoge en la monitorización del proceso de desarrollo de los distintos usuarios.
- En el campo del aprendizaje, un entorno Web puede ser integrado de forma relativamente fácil y natural con un sistema de e-learning que contenga la parte contenidos sobre la programación.
- Por último, el hecho de que funcione a través de Web, posibilita que los distintos usuarios trabajen a su ritmo sin la necesidad de que haya presencia física coincidente en lugar y tiempo de desarrolladores y supervisores, proporcionando mucha más flexibilidad de horarios.

Consideramos la base de conocimientos como una pieza clave de este sistema. Muchos errores son difíciles de corregir por una mala comprensión del mensaje de error por parte del desarrollador. Además, muchas veces es más importante obtener una ayuda con ejemplos concretos y casos en los que se haya detectado y corregido el error que una definición teórica de en que consiste el error. En estos momentos, existen distintos sistemas para gestionar conocimiento que son muy populares; pero la característica que deberíamos establecer para este módulo es la posibilidad de incluir contenido de forma muy sencilla para los usuarios y de forma colaborativa, es decir que todos los usuarios puedan dejar información en la base de conocimientos de una forma muy sencilla y sin la necesidad de herramientas especializadas. Además, otro factor importante es que el propio sistema extraiga conocimiento de forma automática de las acciones de los usuarios.

9.2 Objetivos del prototipo

La creación de este prototipo tiene dos objetivos fundamentales:

- Desarrollo un **entorno de desarrollo integrado con una interfaz Web**,
    - El entorno debe ser lo más abierto y flexible posible para que pueda ser utilizado con distintos lenguajes y compiladores.
    - La instalación de elementos en la parte cliente debe ser mínima, idealmente, solo un navegador Web.
    - Debe proporcionar herramientas de desarrollo tales como un editor de código, diversos compiladores y un gestor de archivos y directorios del proyecto, para poder afrontar proyectos software de tamaño medio.
- El entorno debe disponer de una **base de conocimientos colaborativa**, esta base de conocimiento será creada a partir de la experiencia de todos los usuarios
    - Debe permitir a todos los usuarios consultar información semántica a cerca de los errores detectados en el programa.
    - Cualquier usuario debe poder introducir, de forma sencilla y rápida información sobre los errores detectados o solucionados.
    - Así mismo, el sistema debe enlazar automáticamente esta información con el sistema de compilación para facilitar la consulta del conocimiento

asociado a cada error, proporcionando ayudas a la programación y a la resolución de errores.

Entre los objetivos también incluimos:

- La **sencillez de uso**. Es importante que sea sencillo de utilizar, instalar y mantener.
- Que el entorno pueda ser ejecutado y sea accesible desde un **gran número de plataformas**.

9.3 Ámbito del prototipo y relación con el resto de prototipos

Este subsistema cumple la función de interfaz con el usuario y permite controlar todas las herramientas necesarias en el proceso de escritura y depuración de código desde una única pantalla. El trabajo con este subsistema siempre se realiza sobre los archivos en el espacio personal del usuario y delega en el sistema COLLDEV el intercambio entre este espacio personal y el espacio compartido colaborativo disponible para el grupo que trabaja sobre un proyecto. El módulo IDEWeb tiene una relación muy estrecha con el subsistema PBA, ya que desde IDEWeb se invoca la compilación y el análisis estático que realiza PBA y se recoge la información generada por este tras realizar el análisis de los datos almacenados en la base de datos.

9.4 Requisitos del prototipo

A continuación, se enumerar los requisitos de este prototipo.

9.4.1 Gestión de usuarios (identificación y autenticación)

Para abrir una nueva sesión en el entorno de desarrollo los usuarios deben de estar registrados. El entorno permitirá autenticar a los usuarios para poder acceder a los archivos del proyecto que están desarrollando, a los servicios que ofrece y personalizar el acceso a la base de conocimientos.

9.4.2 Gestión del desarrollo de proyectos

Cada proyecto tiene reservado un espacio de trabajo en el servidor. Una vez que el usuario abre una nueva sesión de trabajo, se le presenta su espacio de trabajo, con una lista de los archivos que le pertenecen, y opciones para la gestión de los mismos. Dentro del espacio del servidor, así mismo, se dispondrá de utilidades para intercambiar archivos entre este espacio en el servidor y los sistemas locales de usuario.

Al elegir la opción de edición de un archivo, se cargará el archivo seleccionado en el área de edición, en caso de seleccionar nuevo archivo se creará en blanco para su edición. El entorno dispondrá de utilidades de ayuda a la edición, como el resaltado de sintaxis dependiendo del lenguaje seleccionado.

El sistema permitirá compilar los archivos en el espacio del servidor y recoger los binarios resultantes.

9.4.3 Gestión de errores

Los errores de compilación se mostrarán al usuario inmediatamente; pero además, serán clasificados por tipos y almacenados en la base de datos de la aplicación junto al enlace a la

página correspondiente de la base de conocimientos que explica la solución al mismo. Estas páginas de la base de conocimientos son creadas y enlazadas automáticamente por el sistema, pero los propios usuarios añaden más información y las modifican para enriquecerlas con su experiencia.

9.4.4 Base de conocimientos colaborativa

Se pretende aprovechar los conocimientos adquiridos por todos los usuarios del sistema para crear una base de conocimientos acerca del lenguaje de programación.

Un mismo error puede tener varias soluciones, dependiendo de las causas que lo provoquen, y a veces resulta frustrante, para el alumno, comprobar que la "ayuda" del compilador no aporta nada para solucionar el problema.

Pretendemos que en esta base de conocimiento, cada usuario del sistema vaya incluyendo ejemplos reales que permitan resolver distintos problemas, ya que los ejemplos de código es la parte de la ayuda que aporta más luz a los programadores. Con este sistema la experiencia de los usuarios van quedando reflejada en esos archivos de ayuda, las cuales con el tiempo van formando una excelente "enciclopedia" de programación.

La base de conocimientos colaborativa dispone de páginas HTML en las que se recoge información sobre un error de compilación concreto, que cualquier usuario puede crear y modificar para puntualizar alguna parte del contenido, para corregir algún error en el contenido, añadir más datos, etc. Pretendemos que todos los usuarios colaboren entre si para crear páginas que recojan contenidos amplios, correctos y de calidad; más que cada usuario aporte su solución de forma independiente.

Todo este sistema estará conectado tanto con el sistema de captura de errores que permitirá acceder fácilmente a este sistema para profundizar en la información sobre el error capturado y también con el entorno de desarrollo en el cual los mensajes visualizados serán extraídos de esta base de conocimientos.

9.4.5 Arquitectura portable y multiplataforma

Aplicación que funciona según el modelo de aplicación Web. La función del cliente es la de proporcionar una interfaz al usuario, dejando toda la parte del proceso en el lado servidor. La comunicación entre ambos será a través del protocolo HTTP, usándose en el cliente cualquier navegador Web estándar.

El sistema está pensado para ser portable entre plataformas, por lo cual las rutas a archivos, llamadas a ejecutables, llamadas a bases de datos, etc., deben situarse en un archivo de configuración esto proporcionará mucha flexibilidad para adaptar al sistema a nuevas situaciones. Por la misma razón, la salida HTML que proporcione deberá ser estrictamente compatible con los estándares (W3C, World Wide Web Consortium) para que pueda visualizarse en cualquier navegador.

9.5 Actores y casos de uso

Administrador. Lleva el control de la aplicación, configura nuevos compiladores, gestiona usuarios, modifica índices de la base de conocimiento.

Desarrollador. Sólo tiene acceso a su espacio personal, en él puede importar archivos, crearlos, compilarlos, eliminarlos. Puede trabajar con la base de conocimientos colaborativa creando nuevas entradas o añadiendo y modificando contenidos a entradas ya existentes.

Supervisor. Puede realizar todas las operaciones del actor desarrollador y además, invocar

análisis sobre cualquier proyecto, de los que supervisa, bien finalizado o en desarrollo.

Gestión de usuarios. Creación, modificación y eliminación de usuarios.

Validación de usuario. Si el usuario es correcto y la clave de usuario es correcta, entonces se inicia la sesión del usuario y se le muestra el menú adecuado según el rol: desarrollador, supervisor, administrador.

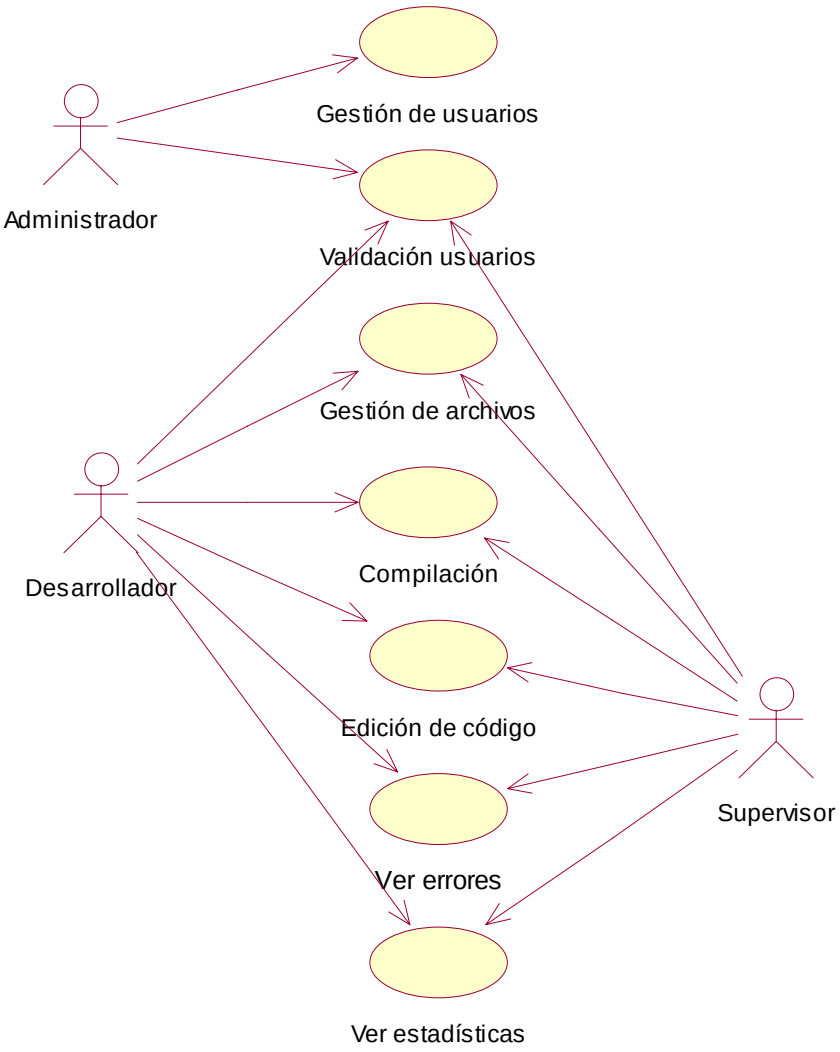


Figura 54. Diagrama de casos de uso para el prototipo de entorno integrado

Gestión de archivos. Nuevo archivo, existen dos formas de crear un nuevo archivo en espacio de trabajo: importándolo del ordenador cliente o editándolo en el sistema. En ambos casos se guardará en el directorio de trabajo del usuario, o en un subdirectorio. Intercambio de archivos entre servidor y sistema local, el usuario puede seleccionar un archivo existente en su espacio de trabajo, y este se transfiere a la máquina cliente y viceversa.

Edición de código. Para editar código IDEWeb proporcionará dos posibilidades, un área de texto HTML, y un editor Java con más opciones, que se desplegará en forma de applet.

Compilación. Compilar un archivo: Se selecciona un archivo del espacio de trabajo del usuario, un lenguaje y un compilador de entre todos los disponibles. La salida se mostrará en un área del entorno incluyendo todos los mensajes con errores de compilación y otros posibles errores.

Gestión de errores. El sistema localizará automáticamente cada error en el archivo fuente correspondiente. Además, por cada error se creará un enlace a la base de conocimientos según el compilador y lenguaje utilizados. El usuario podrá consultar la información asociada a cada error y revisar los enlaces relacionados. Si no existe la entrada en la base de conocimientos se creará automáticamente con la información básica.

Gestión de estadísticas. Muestra las estadísticas sobre frecuencia y evolución de errores dentro del entorno integrado para ayudar al usuario a no repetirlos. La elaboración de estas estadísticas la realiza el subsistema PBA (Capítulo 7).

9.6 Diseño

9.6.1 Diseño del subsistema de edición y compilación

Diagrama de clases

Para desarrollar este entorno como aplicación Web se ha utilizado el framework Struts®. En este framework la funcionalidad se descompone en "acciones" muy simples que están implementadas por clases que heredan de la clase "Action". Por otra parte, los formularios que permiten la invocación de las acciones desde la "vista" de la aplicación heredan de la clase "ActionForm".

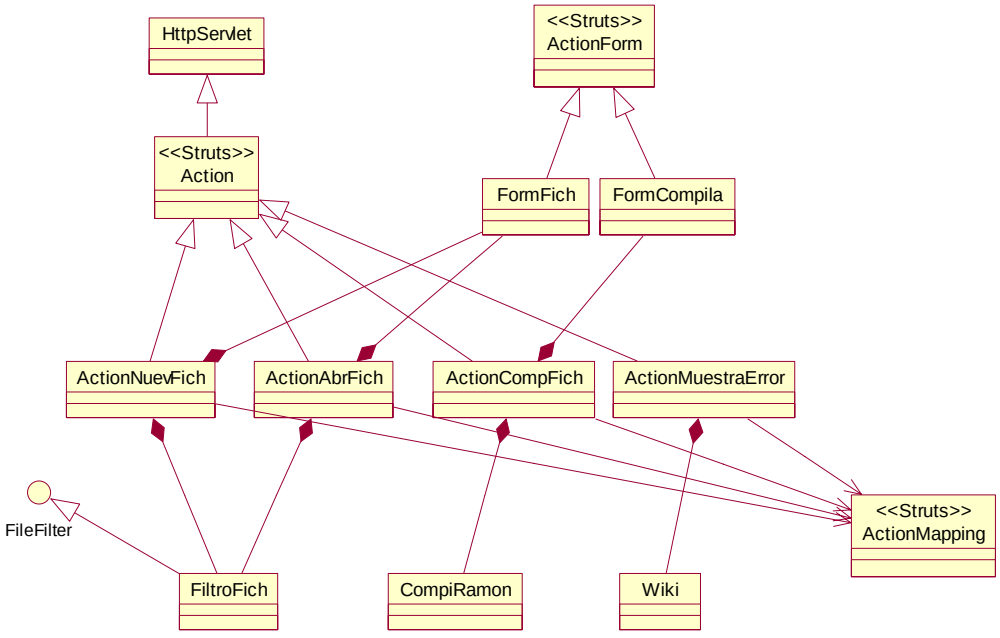


Figura 55. Vista parcial del diagrama de clases del sistema

Diagrama de secuencia de edición Web de un nuevo archivo

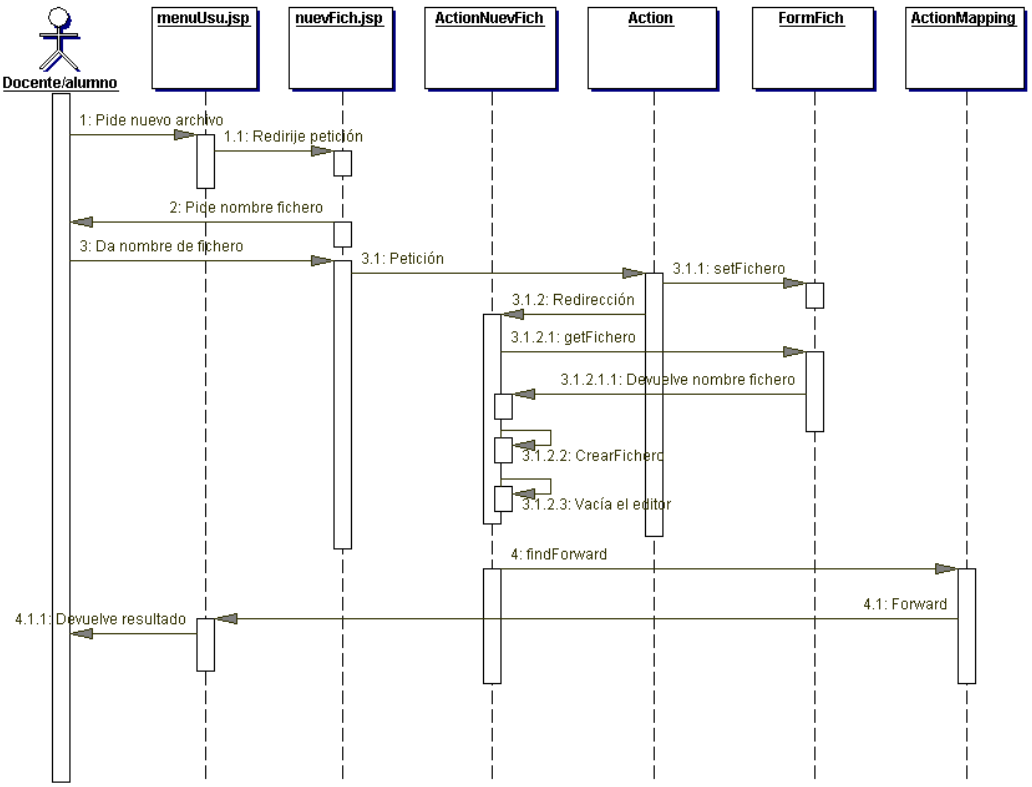


Figura 56. Diagrama de secuencia de Gestión de archivos, edición de un nuevo archivo

Diagrama de secuencia de la compilación de un archivo

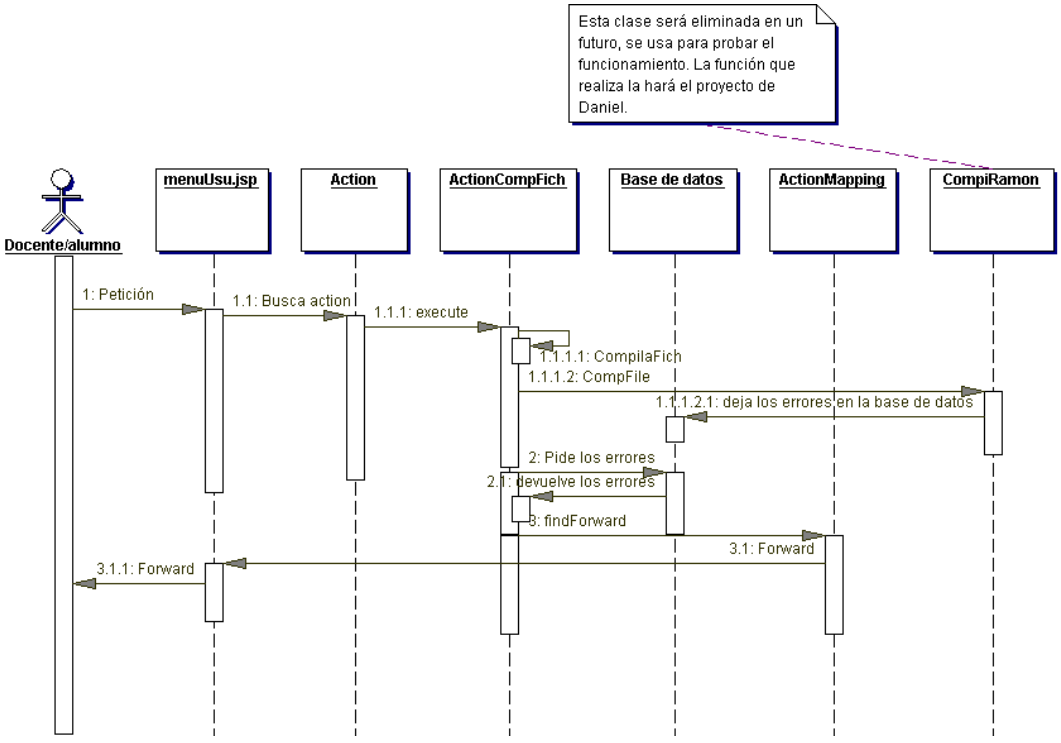


Figura 57. Diagrama de secuencia de Gestión de las compilaciones

Diagrama de secuencia de la creación de una nueva página asociada a un error

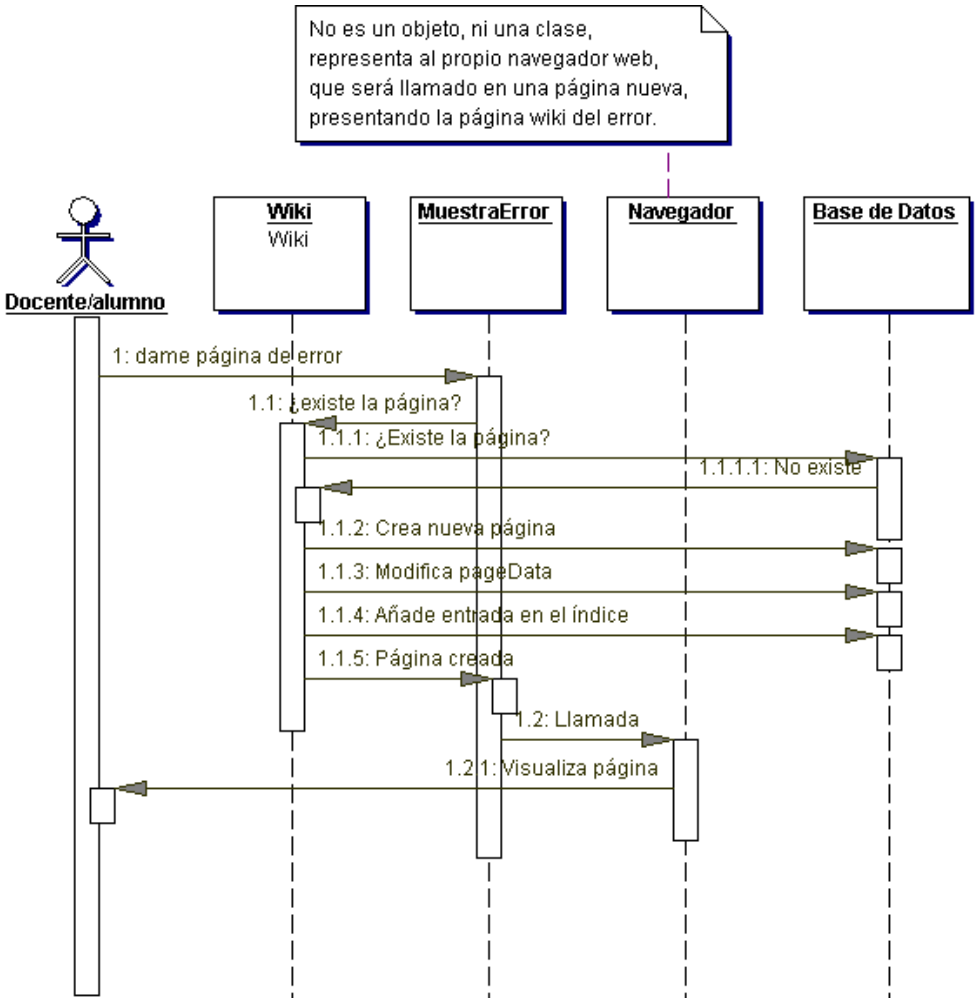


Figura 58. . Diagrama de secuencia de Gestión de errores, creando una nueva entrada de error en la base de conocimiento

9.6.2 Diseño subsistema de la base de conocimientos colaborativa

Una parte fundamental del sistema IDEWeb es la base de conocimientos en la que el usuario del sistema podrá obtener información y ayuda en el manejo del lenguaje, del compilador y de los errores de programación que vaya cometiendo. Se pretende que sean los propios usuarios quienes compartan sus conocimientos, ayuden a resolver problemas, informen de cómo consiguieron arreglar un determinado error para implementar esta base de conocimientos se han estudiado varias posibilidades y la que se ha elegido ha sido el sistema Wiki.

¿Qué es Wiki?

El término WikiWiki ("wiki wiki" significa "rápido" en la lengua hawaiana, "wee kee wee kee") se utiliza para referirse a un programa que permite tanto leer documentos hipertexto como publicarlos en la Web, este sistema es típicamente colaborativo ya que distintos autores describen un tema con ayuda de enlaces a otros temas relacionados creados por otros autores, además en una determinada página pueden intervenir varios autores a la vez. A las páginas publicadas por este sistema se las denomina Wiki Wiki Web, pero suele abreviarse como “wiki”.

Wiki es una herramienta colaborativa que permite a los usuarios publicar en Web sin tener conocimientos de HTML ni permisos de acceso a las carpetas que contienen los archivos de la página. El tiempo necesario para aprender a manejar esta tecnología es muy corto. Fue creado en 1.995 por Ward Cunningham quien creó uno de los primeros sitios con esta tecnología[6], y además ha escrito un libro al respecto [Leuf 2001]. Wiki no es un nuevo protocolo de Internet, las páginas generadas por un wiki deberían de seguir los estándares HTML. Para hacer un sitio wiki es necesaria la generación de páginas dinámicas, con cualquier tecnología que nos permita hacer esto y una base de datos que guarde dichas páginas.

Su funcionamiento es simple. Cada página tiene un botón de edición, al pulsarlo sale una nueva ventana con un área de edición de texto que nos permite modificar la página, escribiéndola en un lenguaje de marcado estándar para los wiki mucho más sencillo que HTML, con solo añadir enlaces a páginas que aún no existen, podemos crear nuevas páginas wiki. Wiki nos permite ver las diferencias entre versiones y realiza automáticamente un registro histórico de todos los cambios.

Cómo se utiliza un Wiki

El modo de empleo del wiki es muy sencillo, para consultar una entrada, no hay que hacer nada en especial, se consulta como cualquier otra página HTML, en cualquier navegador: introduciendo su URL o bien utilizando los enlaces hasta llegar a la entrada buscada. Para editar una página wiki sólo hace falta pulsar en el botón (o enlace) "editar" que hay en cada página, saldrá un área de edición de texto con el contenido de la página, en lenguaje wiki. Este lenguaje es simple y permite un marcado sencillo. Para crear una nueva página sólo hay que hacer un enlace a ella y cuando se pulsa sobre él, si no existía la página, se crea una nueva, si la página ya existía se edita.

Para darnos cuenta de la ventaja que supone esto sobre la edición convencional de página HTML, pensemos en como se edita una página Web típica:

- Se obtiene copia de las fuentes de la página a modificar, mediante ftp, o directamente al leerla en un navegador.
- Se edita el archivo HTML con todas sus etiquetas, para lo cual se requiere unos conocimientos sobre las etiquetas y la estructura del HTML. Normalmente se utilizan herramientas de edición especializadas para realizar esta tarea.
- Se sube este archivo al servidor que contiene la página Web, normalmente mediante el servicio ftp, necesitando para ello un acceso de lectura-escritura en el directorio.

Todos estos pasos se simplifican en Wiki a pulsar el botón de edición, escribir sobre el navegador y pulsar el botón de grabar. Además, debemos de pensar que wiki está pensado para que muchos usuarios colaboren, evita la obtención de las fuentes habría que asegurarse siempre que es la última versión, la sustitución de la página antigua por la nueva con el riesgo de sobrescribir cambios, el uso de ftp, y la necesidad de tener cuentas de usuario con permisos de lectura-escritura en el directorio para todos los usuarios que colaboren.

Casi todos los wikis tienen botones (o enlaces) diff e info. Diff indica los cambios realizados entre las distintas versiones de la página, e Info da información sobre la página,

[6] Wiki mantenido por Ward Cunningham, el creador del Wiki, con muchos enlaces a motores wiki, sintaxis… Fue el primer wiki de la historia y sigue siendo uno de los más visitados: http://c2.com/cgi/wiki

versión, última modificación, etc. Esto es muy importante ya que al ser unas páginas abiertas a la libre modificación conviene llevar un control de esos cambios realizados, para poder dejar la página en un estado previo si la modificación no fue conveniente, ver quien realizó las modificaciones y llevar estadísticas de su uso.

Funcionamiento interno de un wiki

Un wiki puede programarse en cualquier lenguaje, a través de CGI, aunque abundan los realizados en php y java, dada su mayor orientación a la red. Los más sofisticados usan bases de datos para almacenar las páginas e incluyen más opciones de sintaxis, posibilidad de añadir etiquetas HTML, si bien es importante la estandarización en la mayoría de las funciones. Un wiki tiene una página principal, generada automáticamente, a partir de la cual se construirá el resto del sitio, con el sistema de crear enlaces vacíos e irlos rellenando. El programa va generando las nuevas páginas y las guarda en archivos, o base de datos, guardando además información de los cambios, fechas, históricos, quien realizó la modificación, y demás datos necesarios.

Seguridad en wiki

En principio cualquier persona puede modificar las páginas de un Wiki, esto puede parecer un fallo de seguridad muy grande, pero en la práctica funciona; así lo demuestran proyectos como Wikipedia[7], una enciclopedia basada en este sistema. Hay que situar a wiki en su contexto, wiki nace como herramienta de colaboración y para ello debe estar abierta a todos. Se basa en el hecho de que si alguien se molesta en editar una página, lo va a hacer para aportar algo al wiki y no para destruirlo. Evidentemente, cualquiera puede hacerlo, no supone un reto para nadie, precisamente por ello los wikis no suelen ser objetivos de hackers. Otra cuestión es que se introduzcan datos incorrectos en el wiki ya sea de forma accidental o intencionada, eso puede ocurrir en un wiki o en cualquier página HTML convencional, y en un wiki tenemos la ventaja de que algún visitante de la página puede percatarse del error y corregirlo de forma inmediata.

Aplicación de wiki a la base de conocimientos colaborativa de IDEWeb

Una vez vistas las características de los sistemas wiki y su forma de trabajar, comprobamos que encaja perfectamente con los requisitos establecidos para la base de conocimientos colaborativa de IDEWeb, ya que:

- Facilita la consulta de la información ya que podemos acceder a ella mediante un hiperenlace, de la misma forma podemos acceder también a información relacionada.
- Permite la construcción y edición de páginas Web completas, con el formato deseado y con muy pocas restricciones, todo ello de una forma muy cómoda, sin necesidad de conocer HTML en profundidad.
- La única herramienta para realizar todo esto es un navegador Web por lo que evitamos la instalación de herramientas más sofisticadas.
- Está preparado para agilizar la colaboración de diferentes personas en la incorporación de nuevo contenido y refinamiento del existente.

[7] Wikipedia, es un proyecto que trata de construir una enciclopedia en múltiples idiomas con la colaboración de todo el mundo que quiera aportar algo, su dirección es: www.wikipedia.org.

La concepción del wiki que permite que todo el mundo introduzca contenido en el sistema es un cambio en la concepción respecto a muchos sistemas en los que únicamente un experto o un grupo de expertos reducido puede aportar conocimiento al sistema y todos los demás únicamente pueden consultar. En este nuevo planteamiento todos pueden aportar con lo que la base de conocimientos se enriquece más rápidamente, y aunque pueda haber ciertos errores en el contenido estos pueden ser reparados rápidamente.

Para integrar este sistema dentro de IDEWeb hemos trabajado fundamentalmente a nivel de la base de datos del wiki. A través, de la base de datos disponemos del índice del wiki y la URL de cada una de sus páginas, por lo que podemos crear enlaces dentro del entorno de desarrollo a la entrada del wiki correspondiente. Por ejemplo, en cada mensaje de error, cuando se visualizan en el entorno después de realizar una compilación, incluimos un enlace a la información correspondiente a ese error en el Wiki. Además, si no existe entrada para ese error en el Wiki el sistema crea una nueva entrada en la base de datos e introduce en la página la información básica; para que posteriormente los usuarios vayas añadiendo sus experiencias.

Como se vio en el apartado anterior, el problema de la seguridad puede solucionarse evitando que modifiquen páginas albergadas en el wiki. Los usuarios que no se hayan identificado y abierto una sesión en el sistema, con lo cual se sabrá siempre quien ha modificado una página.

9.6.3 Diseño de la arquitectura de la aplicación

La aplicación se divide en tres bloques claramente diferenciados (Figura 59):

- El entorno sigue la arquitectura de una aplicación Web modelo 2 clásica. Donde todo el procesamiento se lleva a cabo en el lado servidor. El núcleo principal del proyecto, está constituido por el entorno que integra una amplia funcionalidad: edición, compilación, gestión de archivos, gestión de la base de conocimientos, etc.
- Un applet que se ejecuta en la parte cliente y cuyo único objetivo es editar texto de la forma más cómoda posible y que permite el coloreado de sintaxis.
- Por último la base de conocimientos colaborativa, que almacena las páginas de ayuda, es totalmente independiente del entorno y simplemente necesitan una conexión TCP/IP para comunicarse, por tanto podrán estar en máquinas distintas para mejorar el rendimiento general del sistema.

Entorno: Se seguirá el patrón arquitectónico MVC (Modelo-Vista-Controlador). Este patrón promueve la separación de las funciones:

- Modelo: Lógica del negocio, son las operaciones propias de la aplicación, en nuestro caso compilación de archivos, gestión de usuarios, gestión de archivos, gestión de errores, manejo de páginas de ayuda. También se incluyen en este nivel las clases que permiten el acceso a la base de datos.
- Vista: Es el interfaz con el usuario, la entrada-salida de la aplicación, en IDEWeb son las páginas Web presentadas en el navegador y los formularios de entrada de datos.
- Controlador: Son las funciones de control, recogen los datos de entrada del usuario, se los envían al "Modelo", y redirigen la salida adecuada al usuario llamando a la "Vista".

Trabajando con servlets/JSP, el patrón MVC también se denomina Modelo2. El tamaño del presente proyecto es de la suficiente envergadura para que el Modelo2 comience a

mostrar todas sus ventajas. Se ha utilizado el framework Struts [Cavaness 2004] como base para implementar el sistema. Struts implementa el patrón MVC mediante un servlet como controlador y páginas JSP que conforman la vista. El modelo estará formado por diversas clases Java, algunas de ellas implementadas como Beans para permitir un acceso "limpio" desde los JSP.

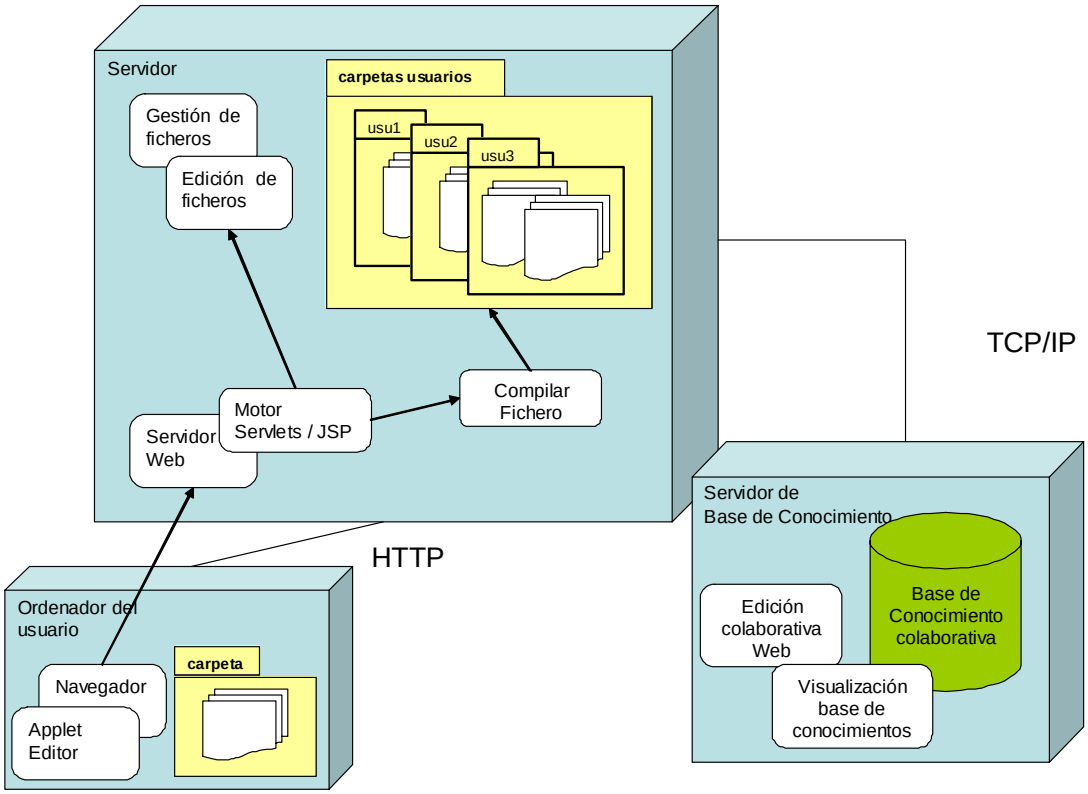


Figura 59. Arquitectura del sistema

Base de conocimientos colaborativa: Es independiente del entorno. Para su interconexión el entorno hace consultas sobre la base de datos que soporta a la base de conocimiento, de esta forma puede recuperar la dirección de cada una de las entradas de la base de conocimientos para insertar enlaces a estas entradas y crear nuevas páginas modificando directamente la base de datos, añadiendo un enlace a la nueva página en la página índice, y creando la nueva página, con una plantilla que servirá como base para que el usuario edite el error más fácilmente.

Editor: El editor es un programa Java implementado como un applet para que pueda ser ejecutado desde el navegador Web. Con la tecnología del plug-in de Sun podemos utilizar las últimas versiones del runtime de Java con lo que hemos decidido utilizar componentes Swing para la interfaz del editor. Este editor se comunica con el navegador para intercambiar el contenido del área de edición en el applet y en el navegador.

9.6.4 Diseño de la navegación

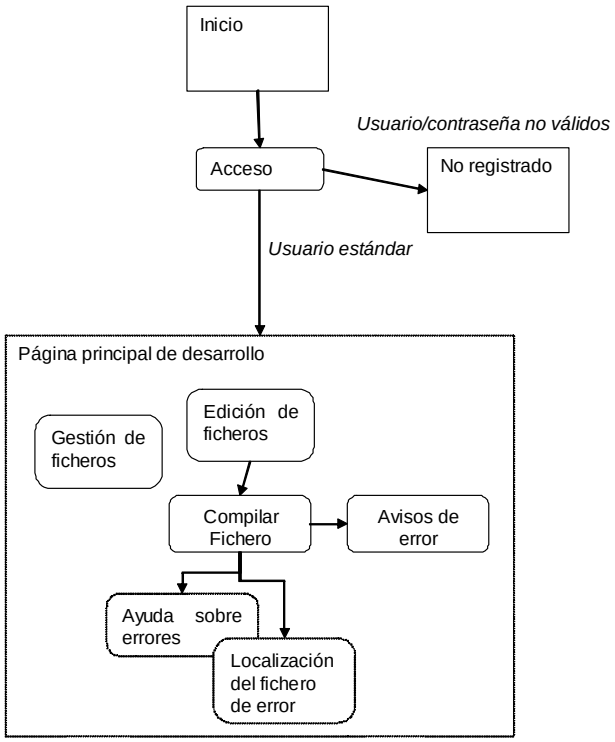


Figura 60. Esquema del modelo navegacional de la aplicación

9.6.5 Diseño de la interfaz

Primer prototipo de interfaz

Se ha realizado una aproximación iterativa para el diseño de la interfaz realizando varios prototipos sobre el papel y en HTML. En la primera aproximación al diseño considerado óptimo: una amplia zona de edición en la zona central, y los controles necesarios en los laterales. Se intentó minimizar el número de páginas necesarias, idealmente se manejarán todas las opciones principales desde una única página. Se busca una apariencia similar a un entorno de desarrollo tradicional. Por supuesto, el código HTML debe ajustarse a los estándares del W3C para evitar incompatibilidades entre navegadores. Se prima el espacio dedicado al área de texto que servirá para editar el código fuente, esto es importante ya que va a ser lo más utilizado y debe poder leerse con la mayor comodidad posible.

Se han utilizado técnicas de internacionalización para que el idioma de la interfaz pueda adaptarse a la configuración del navegador, con lo que el usuario dispondrá del idioma de su preferencia. IDEWeb estará disponible en inglés y castellano de forma inicial, añadir nuevos idiomas es extremadamente sencillo y no requiere cambios en el código de la aplicación ni su recompilación.

Las utilidades de archivo, en la primera aproximación, se sitúan en la parte derecha de la pantalla, en una columna. Los errores obtenidos en la compilación se situarían debajo de la ventana del texto, y se dispone de una barra de avisos donde van apareciendo mensajes según las operaciones que realice el usuario. En la parte superior se indica la fecha, y se deja espacio para enlaces que puedan ser interesantes.

Editor de código avanzado

Este es el elemento que más flexibilidad tiene a la hora del diseño, ya que al ser un applet, puede usarse toda la potencia de los componentes swing de Java en interfaces de usuario, y no estamos limitados al código HTML como en el resto de la aplicación. No obstante se opta por un diseño sencillo, teniendo en cuenta que aunque no estemos limitados por la potencia del lenguaje, si lo estamos por la velocidad de transmisión a través de la red, el editor se despliega en forma de applet, por lo que debe ser descargado de la máquina servidora a la máquina cliente a través de la red. Pese a que el enfoque de la aplicación es su uso en Intranets, es perfectamente posible su uso en Internet, que suelen ser bastante limitadas en cuanto a ancho de banda de comunicación.

Versión final de la interfaz

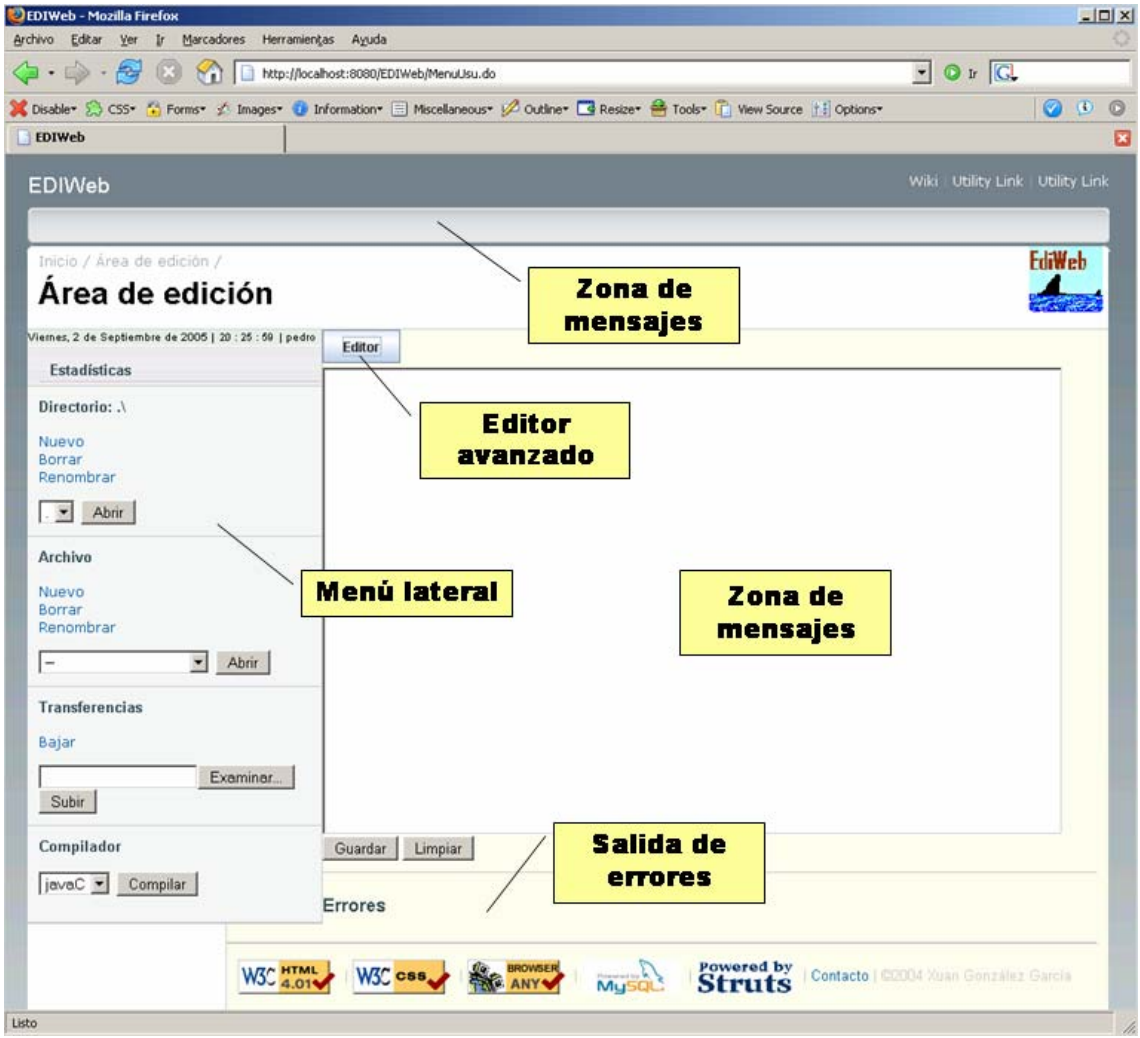


Figura 61. Partes de la interfaz de usuario

Con la primera aproximación se obtuvieron buenos resultados, un interfaz sencillo y fácilmente comprensible. Para la versión final (Figura 61) se mantuvo la estructura básica, pero se profesionalizó el diseño para darle una apariencia más seria. Se cambiaron los controles de archivos a la parte izquierda de la pantalla, dado que también los entornos de

desarrollo (Netbeans, Eclipse…), y los exploradores de archivos más conocidos (Internet Explorer de Windows, Nautilus, Xffm…) también lo sitúan ahí.

La barra de mensajes que en el prototipo se situó en la parte inferior, ahora se ha colocado en la parte superior. Las herramientas la sitúan normalmente en la parte inferior, pero al tratarse de una aplicación Web, no contener un marco de aplicación definido, y poder funcionar en ordenadores con muy distinta resolución de pantalla (aunque se recomienda 1024x768 como mínimo para trabajar cómodamente), esta barra podría quedar oculta en la parte inferior y ser molesto el tener que bajar la barra de scroll para verla. Por ello aún estando en un sitio no habitual, se ha preferido colocar en la parte superior.

Se ha comprobado su correcto funcionamiento con los dos navegadores más populares, el Mozilla Firefox (Figura 61), y el Microsoft Internet Explorer (Figura 62). El motor del Mozilla (Gecko) es usado a su vez en muchos otros navegadores, con lo que se supone igual apariencia en todos ellos. También se probó de forma satisfactoria en el navegador Konqueror, que dispone de un motor de renderizado propio, aunque en este navegador mostraba un pequeño error, que afecta estéticamente; pero no funcionalmente, y es que el tamaño del elemento de selección de archivo, es bastante superior a los otros navegadores, y se "sale" un poco de la cuadrícula del menú. Como se dijo, esto no afecta a la funcionalidad, solo en la apariencia. Se conseguido mantener un diseño sin frames, de acuerdo a los principios de usabilidad propuestos por Jakob Nielsen [Nielsen 1999].

Además del correcto funcionamiento, se ha intentado que la apariencia en ambos sea lo más parecida posible, esto no resulta sencillo ya que tienen un tratamiento distinto de CSS, distintos tamaños de elementos y distintas políticas de colocación.

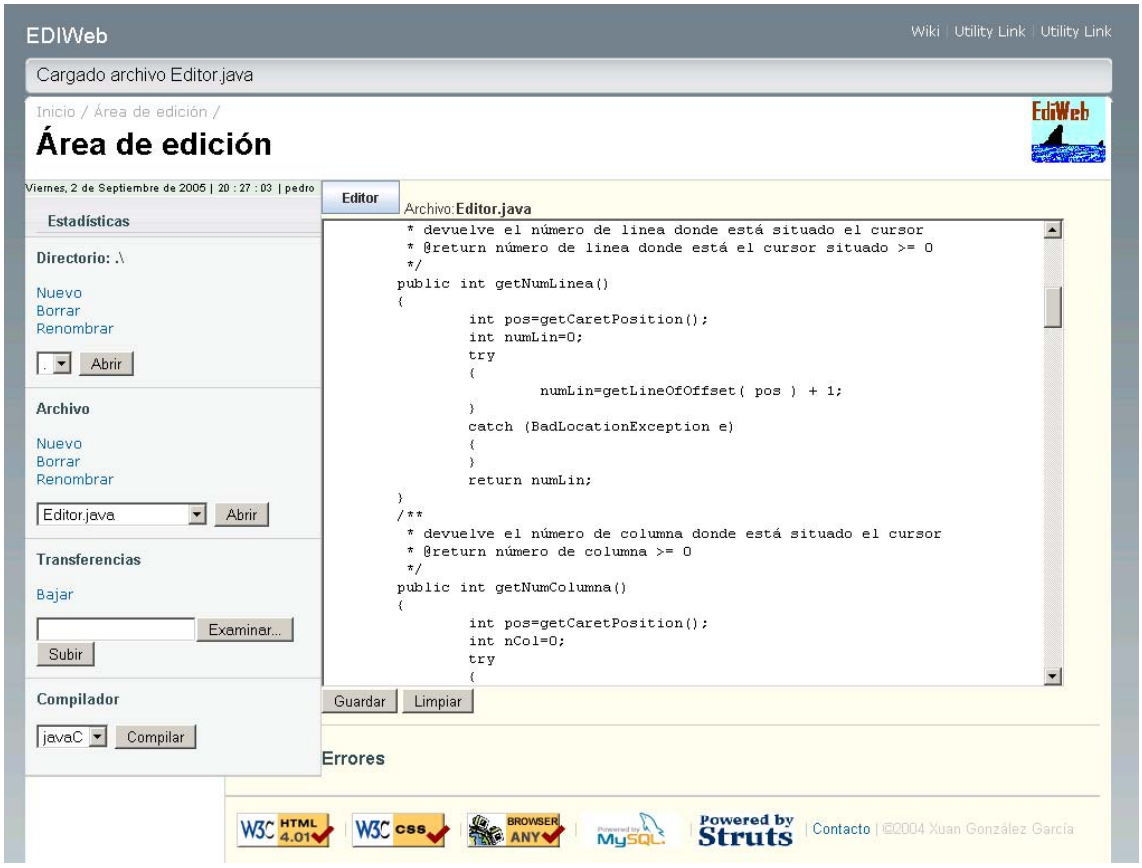


Figura 62: Vista del área de trabajo de usuario con el navegador Explorer 6 en castellano.

Se ha mantenido la misma estructura en todas las pantallas de la aplicación, solo variando el contenido del área de trabajo, el contenido del menú lateral, y la barra de navegación (esta no estaba presente en el prototipo).

Al editor avanzado (Figura 63), desplegado en forma de applet Java, se le han añadido funcionalidades de posicionamiento de cursor, indicando en todo momento en qué línea, columna, y número de carácter se encuentra, y cual es el total, esto ayuda a la hora de buscar un error, ya que el compilador informa del número de línea donde se encuentra. También se cambiaron las barras de desplazamiento, del tipo AWT, al tipo swing, esto permite desplazar el texto con los cursores, que antes no era posible, solo se podía mover la barra de desplazamiento con el ratón, lo cual hacía su uso muy molesto. Esto ha cambiado la apariencia de dicha barra, que se muestra con el aspecto del resto de componentes swing, en lugar de adoptar el estilo del sistema operativo. La apariencia es correcta en ambos casos, por lo que estéticamente no se considera que se haya mejorado o empeorado, pero funcionalmente si se ha mejorado su comportamiento.

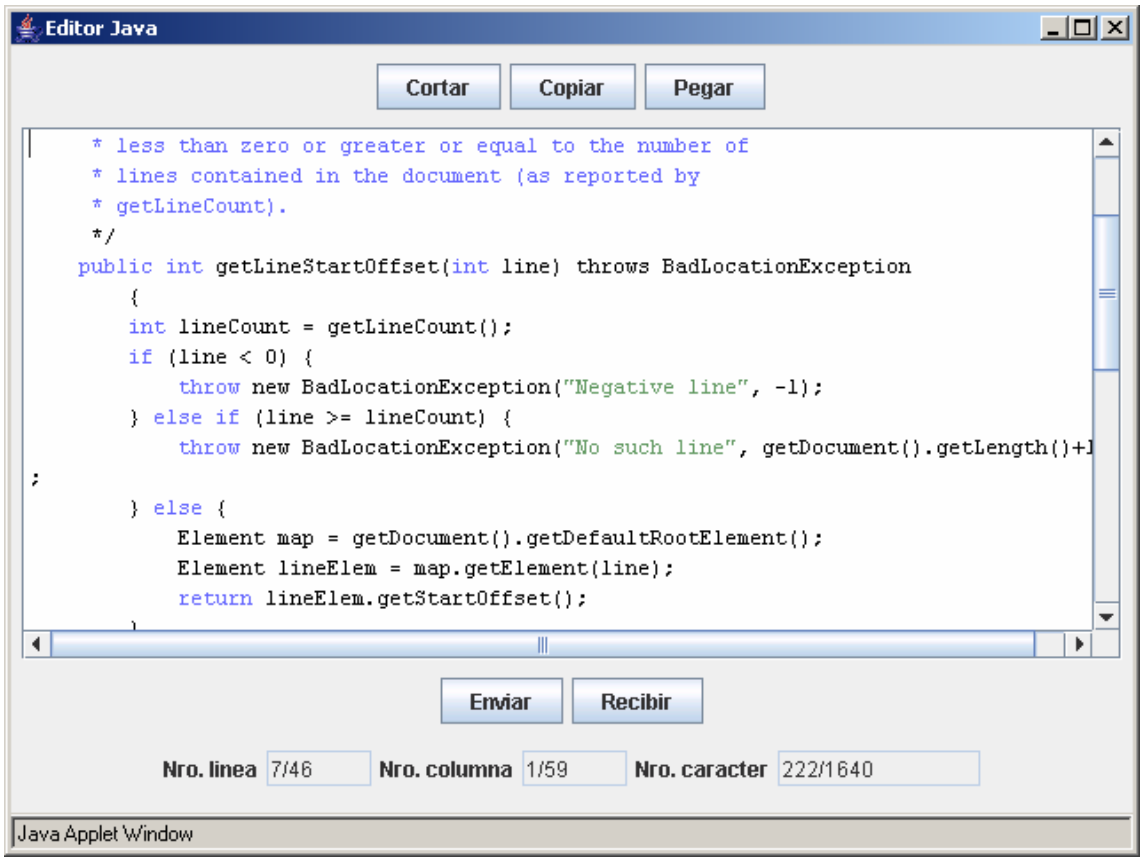


Figura 63: Versión final del applet Editor.

9.7 Evaluación de la base de conocimientos colaborativa

Para evaluar la idoneidad de la base de conocimientos colaborativa en el cometido de ayuda a la programación, se ha efectuado una prueba real en un entorno académico, con los

alumnos de la asignatura Estructura de Datos y de la Información de la Escuela de Ingeniería Técnica Informática de Oviedo[8].

Esta prueba ha consistido en probar la parte colaborativa en la construcción de la base de conocimiento, el prototipo que se utilizó se basa en un sistema Wiki, para ello se instaló el phpwiki [phpwiki], y se usó como herramienta de ayuda en los grupos de prácticas de la asignatura (Figura 64). El Wiki se administró de forma totalmente manual, el objetivo de la prueba era demostrar la idoneidad de una herramienta colaborativa para gestionar los conocimientos relacionados con el desarrollo de proyectos de programación de tamaño medio.

Figura 64. Pantalla principal del Wiki de pruebas.

En esas prácticas se les pide a los alumnos que realicen una aplicación Java que resuelva un determinado problema. Las prácticas duraron 3 meses aproximadamente, el número de alumnos implicado fue de unas 20 personas por grupo, 5 grupos.

Para resolverlo, pueden usar distintas estructuras de datos, y se pretende que el Wiki sirva como tablón de exposición de las distintas opciones, todas ellas aportadas, ampliadas y corregidas por los usuarios.

9.7.1 Encuesta

Para evaluar la utilidad del Wiki se pasaron 2 encuestas, una de respuestas libres en el propio Wiki, para conocer las preferencias de configuración y la forma concreta de utilizar

[8] http://petra.euitio.uniovi.es/asignaturas/edi/ediweb/

la herramienta, y otra en papel con preguntas tipo test sobre la actitud ante el Wiki que se evaluó según la escala de Likert (Figura 65). La encuesta consta de 2 bloques, uno de actitud, que mide la predisposición del usuario a las tecnologías en la educación, y otro de utilidad que trata de medir la utilidad real de la herramienta en el experimento.

Test EDIWiki

Edad……

Actitud general:

Valore el uso de tecnologías web en la enseñanza (1 nada positivo, 5 muy positivo)

1	2	3	4	5

Valore la tecnología wiki (1 nada positiva, 5 muy positiva)

1	2	3	4	5

Valore el presente wiki de 1 a 5

1	2	3	4	5

Aprendizaje con el wiki

Valore la organización del wiki (1 mal organizado, 5 bien organizado)

1	2	3	4	5

¿Le ha sido de utilidad en estas prácticas? (1 nada útil, 5 muy útil)

1	2	3	4	5

¿Resulta sencillo de usar? (1 muy difícil de usar, 5 muy fácil de usar)

1	2	3	4	5

¿Más o menos válido, para el uso que se le ha dado, que una lista de correo? (1 mucho menos válido, 5 mucho más válido)

1	2	3	4	5

¿Qué mejoraría? (Respuesta libre)

Figura 65. Encuesta

Si se puede demostrar que los resultados de ambos bloques de la encuesta se ajustan a la curva normal, con un intervalo de confianza de al menos un 95 % podríamos asumir que individuos con la misma predisposición hacia las tecnologías en la educación también tendrían una opinión positiva del uso del Wiki en la educación. Para ello primero se comprueba que ambas variables estén relacionadas, usando la prueba de Levene para

evaluar la homogeneidad de ambas varianzas. Los resultados de la encuesta se han transcrito de forma numérica y se ha hecho la media de puntuaciones de cada grupo (actitud y utilidad), por cada individuo del experimento ().

9.7.2 Resultados encuesta

Tabla 9. Resultados de la encuesta Test EDIWiki

Preguntas **Edad**	**1**	**2**	**3**	**4**	**promedio**	**5**	**6**	**7**	**8**	**promedio**
20	4	4	3	2	3,25	4	5	4	3	4
20	4	4	5	2	3,75	3	2	3	5	3,25
24	3	3	3	2	2,75	3	4	2	3	3
21	4	4	3	2	3,25	3	1	4	3	2,75
19	5	5	4	2	4	4	2	4	4	3,5
25	5	5	5	2	4,25	4	4	4	3	3,75
21	5	4	4	2	3,75	3	4	2	3	3
21	2	2	1	1	1,5	3	1	2	3	2,25
	4	4	4	3	3,75	3	3	3	4	3,25
20	5	5	5	2	4,25	4	2	5	4	3,75
24	4	4	4	4	4	3	4	5	4	4
24	4	4	4	4	4	3	3	4	3	3,25
24	5	3	3	3	3,5	4	4	4	3	3,75
	5	4	4	4	4,25	3	5	5	4	4,25
23	4	5	5	3	4,25	4	5	4	4	4,25
23	4	3	3	3	3,25	4	4	5	4	4,25
24	5	4	5	3	4,25	4	4	5	5	4,5
	3	3	3	2	2,75	3	3	4	3	3,25
22	5	4	3	2	3,5	3	3	4	3	3,25
21	5	4	3	3	3,75	4	4	4	4	4
24	5	4	3	2	3,5	2	1	2	3	2
	4	4	4	2	3,5	4	4	5	5	4,5
25	5	5	4	4	4,5	4	4	4	5	4,25
21	5	5	5	4	4,75	5	5	4	5	4,75
21	4	4	3	3	3,5	3	4	3	4	3,5
23	4	3	3	5	3,75	3	2	5	3	3,25
22	5	4	4	3	4	3	3	2	2	2,5
22	4	4	3	3	3,5	2	4	3	4	3,25
21	4	5	3	3	3,75	4	5	2	3	3,5
19	4	5	3	2	3,5	3	5	3	4	3,75
19	5	4	4	2	3,75	3	4	5	4	4
19	5	4	4	3	4	3	3	4	4	3,5
20	4	4	4	3	3,75	3	4	4	3	3,5
22	5	5	4	4	4,5	4	4	3	4	3,75

Una encuesta se eliminó de forma preliminar (casillas sombreadas), al errar en la pregunta de control, en la que se preguntaba si había utilizado el Wiki (la cuarta del bloque de

actitud). No había utilizado el Wiki por lo tanto sus respuestas no son válidas. El resto de las encuestas quedaron todas dentro del estudio. Los datos se trataron con el paquete estadístico SPSS® versión 11.5.1.

Tabla 10

	BLOQUE	Casos					
		Válidos		Perdidos		Total	
		N	Porcentaje	N	Porcentaje	N	Porcentaje
PROMEDIO	Actitud	33	100,0%	0	,0%	33	100,0%
	Utilidad	33	100,0%	0	,0%	33	100,0%

Prueba de homogeneidad de la varianza

		Estadístico de Levene	gl1	gl2	Sig.
PROMEDIO	Basándose en la media	2,159	1	64	,147
	Basándose en la mediana.	2,079	1	64	,154
	Basándose en la mediana y con gl corregido	2,079	1	60,745	,154
	Basándose en la media recortada	2,127	1	64	,150

Para que Levene demuestre la homogeneidad de las varianzas, el resultado Sigma debe ser >0,05. El resultado de la prueba de Levene es positivo.

Para demostrar que ambos bloques se ajustan a la normal, empleamos el test de Shapiro-Wilk, al igual que la prueba de Levene debe de tener una sigma >0,05 para que quede demostrada la hipótesis con un nivel de significancia del 95%.

Pruebas de normalidad

	BLOQUE	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Estadístico	gl	Sig.	Estadístico	gl	Sig.
PROMEDIO	Actitud	,132	33	,153	,963	33	,307
	Utilidad	,125	33	,200(*)	,970	33	,489

* Este es un límite inferior de la significación verdadera.

Corrección de la significación de Lilliefors

Vemos que ambas Sigmas están bastante por encima del 0,05, con lo que demostramos la hipótesis.

9.8 Limitaciones del prototipo

IDEWeb resuelve muchos de los problemas que afectaban al Compilador Web SICODE, pero aún es claramente mejorable.

La primera mejora consistiría en integrar el entorno IDEWeb con el Sistema para la colaboración en el desarrollo de aplicaciones: COLLDEV, así como con el Sistema de análisis de errores en programas: PBA.

El applet editor puede ampliarse para incluir más funcionalidades que ayuden en la edición, al estilo de los modernos editores de código (autocompletado de código, detección de errores de sintaxis, detección avanzada de errores…). Las ampliaciones a IDEWeb son sencillas de hacer, dada la descomposición modular de que goza su código.

Son importantes todas las mejoras que se puedan hacer en el interfaz, este debería incorporar las facilidades de los entornos de programación tradicionales (Netbeans, Eclipse…), permitir abrir varios archivos, usando pestañas, e integrar el mayor número de herramientas posible.

Haría falta aumentar el número de lenguajes soportados por el editor avanzado, permitiendo resaltar la sintaxis a otros lenguajes, actualmente solo se soporta el lenguaje Java.

También se puede aumentar el número de estadísticas ofrecidas al usuario, incluyendo gráficos de las mismas.

9.9 Conclusiones

Hemos creado un entorno de desarrollo sobre Web que permite utilizar desde un navegador Web las herramientas esenciales en el proceso de desarrollo: editor, gestor de archivos, compilador y analizador de código y gestor de errores.

El entorno Web es interesante dada su universalidad, permite un interfaz común sin importar la tecnología que haya detrás, en la parte servidor. El hecho de poder acceder a aplicaciones desde un simple navegador Web es una ventaja que permite a los usuarios ser independientes del puesto de conexión que utilicen.

Una de las limitaciones del entorno es la capacidad limitada en la edición; para superar esta limitación se ha desarrollado un editor que se implementa como un applet para que el entorno se pueda seguir utilizando con un simple navegador.

La base de conocimientos colaborativa es una aportación esencial de este sistema. Su necesidad está justificada porque la interpretación de los mensajes de error de los compiladores se basa muchas veces, en la experiencia del desarrollador. Para acelerar la adquisición de esta experiencia buscamos que todos los desarrolladores aporten la suya en una base de conocimientos que está centrada en la comprensión y corrección de errores a través de ejemplos.

Para implementar esto hemos utilizado la tecnología Wiki, que ha demostrado su valor en otros campos. La concepción del Wiki permite que todo el mundo introduzca contenido en el sistema y supone un cambio en la concepción respecto a muchos sistemas en los que únicamente un experto o un grupo de expertos reducido puede aportar conocimiento al sistema y todos los demás únicamente pueden consultar. En este nuevo planteamiento todos pueden aportar con lo que la base de conocimientos se enriquece más rápidamente, y aunque pueda haber ciertos errores en el contenido estos pueden ser reparados rápidamente.

Wiki además permite añadir contenido desde el propio navegador sin herramientas especiales. Además, en nuestro sistema los usuarios no crean las páginas desde cero sino que el propio sistema introduce la información básica y a partir de ella los usuarios introducen sus experiencias concretas.