Introducción

CAPÍTULO 1

CAPÍTULO 1 - Introducción.

En la actualidad hay una gran cantidad de repositorios en los que se puede alojar código fuente para poder compartirlo con los usuarios que visiten el sitio. Esta abundancia de repositorios dificulta o incluso impide encontrar la información que realmente interesa. En algunos de estos sitios, para acceder a todo su contenido hay que pagar una membresía, otros centran su contenido en información irrelevante, insuficiente o en el peor de los casos poco confiable, provocando una inversión inútil de tiempo. Por el contrario son pocos lo que proporcionan los códigos fuente de manera clara y ordenada por ejemplo: http://www.koders.com, http://www.planet-source-code.com. Koders proporciona una funcionalidad muy importante que es la de poder hacer consultas directamente en el código fuente y por su parte Planet-Souce proporcionan la capacidad de consulta de código agrupado en categorías así como un *ranking* que el usuario le da dependiendo de el grado de calidad y beneficio que le haya aportado el programa que descargó.

Actualmente los estudiantes de Ingeniería en Sistemas Computaciones de la Universidad de las Américas Puebla, generalmente, al concluir un programa este queda olvidado en algún lugar. Por otro lado, la idea de que estos puedan colaborar para ayudar a la comprensión de funciones o conceptos de programación a alguien más, es lo que motiva la propuesta de crear una herramienta para poder almacenar y consultar código fuente, proporcionando la facilidad de ser visto y analizado por muchos estudiantes, independientes al profesor y el autor del programa.

En el contexto actual de estudiantes de ingeniería en sistemas, estos antes o durante la realización de un programa se sienten en la necesidad de consultar fuentes que les aporten ideas suficientes para solucionar su problema, indiscutiblemente leen o modifican código fuente para mejorarlo y adaptarlo a sus necesidades; la filosofía *open source* afirma la premisa de compartir el código tiende a provocar que el programa resultante sea de calidad superior [Wikipedia, 2006]. Sin embargo no se pretende solamente poner los programas disponibles,

sino convocar a los estudiantes a exponer sus conocimientos, discutirlos y generar nuevas ideas y propuestas. [Proal, 2003].

1.1 Definición del problema.

En la actualidad la Universidad de las Américas Puebla, cuenta con una colección de tesis digitales de los estudiantes, de manera similar se encuentra la necesidad de tener una colección de los programas realizados por los estudiantes de Ingeniería en Sistemas, de tal manera que el código fuente de estos programas puedan ser consultados de manera conjunta. En la actualidad los estudiantes tienen sus programas dispersos en las cuentas asignadas por la Universidad, lo que hace necesario el desarrollo de una herramienta que ayude a la integración de los programas en un solo sitio.

1.2 Justificación

Como se mencionó anteriormente el compartir el código fuente de los programas sería muy útil, debido a que los programas desarrollados podrían ayudar a resolver dudas de implementación a otros estudiantes, brindándoles una referencia para el desarrollo de sus aplicaciones. No se pretende fomentar el plagio sino por el contrario, se quiere motivar al estudiante a realizar trabajos de calidad y alcance en sus proyectos, porque se les daría una alternativa para consultar código fuente de programas de contenido fiable y seguro.

1.3 Objetivo General

Desarrollar una aplicación web para la creación de un repositorio de programas de Java que permita realizar consultas de un conjunto de programas a través de un enunciado de búsqueda usando tecnologías de indexación, almacenamiento y recuperación, utilizando principalmente el ORM *Hibernate* (*Object/ Relational Mapping*).

1.4 Objetivos Específicos

- Capacidad de almacenar y recuperar código fuente de java, utilizando la herramienta de ORM Hibernate para el *mapping* entre los dos modelos (orientado a objetos y relacional), con el fin de probar los beneficios que ofrecen los ORM principalmente el reducir la complejidad de la aplicación y el tiempo de desarrollo.

- Capacidad de compilación del código en el momento de almacenarlo para "garantizar" su funcionamiento. El compilar no garantiza que los programas hayan sido desarrollados adecuadamente, que haya métodos duplicados o variables sin usar, pero es una manera muy útil de probar que el programa funciona y no solo es código inservible.

- Capacidad de obtener los archivos java por dos tipos de búsqueda (por palabras o fragmentos de código y por clasificación). Este es el punto más importante debido a que el usuario que va a hacer la consulta necesita alguna manera de filtrar los resultados para llegar a los programa que él necesita, el encontrar los resultados de una manera rápida es lo que impulsaría al usuario a ocupar el sistema.

- Capacidad de visualizar el código fuente coloreado en formato html. El poder visualizarlo facilitaría el entendimiento entre lo que estamos buscando y lo que estamos viendo es necesario mostrarlo de una manera agradable a la vista y con distintos colores (como si estuviéramos visualizando el código en un IDE)

- Interfaz web para acceder a la aplicación. En el contexto de esta aplicación cliente-servidor es necesaria la interfaz web. Por otro lado se busca que está interfaz se diseñe de una manera minimalista y agradable a la vista. El diseño minimalista se eligió para no complicar el uso del sistema a los usuario y de está manera no entretenerlo mucho y le resulte fastidioso.

- Control de usuarios y clasificaciones existentes. El administrador debe de ser capaz de eliminar del repositorio a usuarios que por criterios personales no deban de usar más el repositorio, de misma manera pasa con las categorías.

1.5 Alcances

- Aplicación web disponible de manera local. No se considera la instalación en un servido de la Universidad debido al difícil acceso que se tiene a ellos en este momento. Aunque la aplicación no se liga a un sistema operativo en particular.

- No se considera el manejo de paquetes y dependencias para los programas del repositorio. El manejo de paquetes y dependencias implicaría una serie de revisión de dependencias que no está consideras para este proyecto.

- Indexación de código fuente para su consulta. La indexación es necesaria para la realización de las consultas a través de un enunciado proporcionado por el usuario estudiante.

- Asignación de categorías a los códigos (clasificación). El clasificar a los programas ayuda a tenerlos organizados para su recuperación.

- Obtención de códigos fuente a través de la comunicación de Hibernate con la base de datos y el sistema de archivos generado. El mantener los programas en la base de datos le restaría velocidad a la aplicación, teniendo un sistema de archivos se tiene más velocidad de recuperación.

- Manejo de usuarios sin considerar la interacción con la base de datos institucional de estudiantes de la Universidad. Como se mencionó anteriormente el acceso a los servidores de la Universidad por el momento está muy restringida.

1.6 Justificación de tecnología a utilizar

En la actualidad las bases de datos más comerciales y con mayor estabilidad se enfocan hacia un modelo relacional, mientras que la programación por el contrario tiende a ser orientada a objetos. Al tratarse de dos modelos distintos se dificulta el intercambio de

información del lenguaje de programación con la base de datos; en consecuencia han surgido las bases de datos orientadas a objetos. Debido a su reciente aparición estas se encuentran inmaduras para poder brindar la seguridad de mantener la consistencia y persistencia adecuada de los datos, independientemente de no tener compatibilidad entre ellas, haciendo imposible la migración de una base de datos a otra. Por lo que para trabajar la persistencia de manera transparente entre los dos modelos se desarrollaron las herramientas de ORM *(Object/Relational mapping),* como por ejemplo: *Hibernate*, *Torque, TopLink* entre otros.

1.6.1 Hibernate

Hibernate es una herramienta *Open Source*, que realiza el mapeo entre las aplicaciones orientadas a objetos y la entidad-relación de las bases de datos en entorno Java. Mantiene características fundamentales como: asociación, herencia, polimorfismo, composición y colecciones [Hibernate, 2006]

Hibernate realiza la transición de una representación de datos de un modelo orientado a objetos a un modelo relacional, proporcionándole capacidades para el almacenamiento y recuperación de información en una base de datos relacional. En el capítulo 2 se profundizará sobre estos conceptos.

1.7 Software

- Hibernate 3.0.5
    - Comunicación entre base de datos y java siguiendo modelo orientado a objetos.

- Apache Tomcat 5
    - Contenedor de *servlets* y JSPs

- MySQL 4.1
    - Base de datos relacional, en donde se almacenará información persistente

- XML
    - Lenguaje para intercambio de datos entre aplicaciones (uso en nuestro contexto), utilizado principalmente para mapear los objetos a ser persistentes de Hibernate a la base de datos.

- Apache Lucene 1.4.3
    - API para la indexación y búsqueda de documentos creada en java, utilizado para la búsqueda de programas.

- Apache FileUpload 1.1
    - API utilizado para subir el archivo al servidor. Para utilizar éste es necesario commons-io-1.1.jar

- Java2Html 5 de Markus Gebhard
    - API utilizado para generar el archivo html con el código fuente coloreado a partir de el archivo .java

- Java jdk1.5 release3
    - Lenguaje de programación para el desarrollo de la aplicación y la compilación de los programas que poblan el repositorio.

- Javascript
    - Lenguaje de programación para la validación de los formularios.

- CCS
    - Hoja de estilo

- Macromedia Dreamweaver MX 2004
    - Editor de paginas web, utilizado para creación de JSPs

- Borland JBuilder 2005
    - Entorno de desarrollo (IDE)

1.8 Organización de este documento

Este documento está organizado como se explica a continuación: el capítulo 2 introduce los conceptos generales para comprender la base teórica y técnica de Hibernate como los requerimientos mínimos para su uso y algunos ejemplos básicos para su implementación, por otro lado se dan los conceptos generales del indexador Lucene para entender su funcionamiento. El capítulo 3 se muestra un análisis de los requerimientos del sistema desarrollado, así como las especificaciones de proceso y la descripción de interfaces. En el capítulo 4 se describen los componentes usados para el desarrollo y las clases que se utilizaron. En el capítulo 5 se muestran las pruebas realizadas al sistema y los correspondientes resultados obtenidos. Finalmente en el capítulo 6 termina el documento mostrando las conclusiones y las posibilidades de ampliación al sistema.