Conclusiones

CAPÍTULO 6

CAPÍTULO 6 – Conclusiones

A lo largo de este proyecto, se puede concluir que migrar las aplicaciones a bases de datos orientados a objetos sería una decisión muy arriesgada, debido a su reciente aparición no se encuentran totalmente probadas. En sistemas críticos podrían ocasionar inconsistencias de la información que provocaría gastos de recuperación muy altos. Por lo que en la actualidad se siguen utilizando los dos modelos en las aplicaciones.

Las herramientas de ORM funcionan como traductores entre los dos modelos, por lo cual los programadores centran la atención en desarrollar las funcionalidades y delegan la tarea de comunicación con la base de datos a la herramienta. A pesar de que está herramienta está volviéndose muy popular para los desarrolladores de software, hay una escasa documentación sobre algunos elementos para el *mapping* entre la base de datos y la aplicación, por lo cual la gran cantidad de foros que se encuentran en la red constituyen un soporte valioso cuando se está desarrollando una aplicación utilizando esta *framework*.

El repositorio de programas fue desarrollando integrando diversas tecnologías existentes, aunque se centró en el uso de *Hibernate* un *framework* robusto y estable, se pudo comprobar la considerable reducción de LOC cuando se delega la persistencia a un ORM, la facilidad de configuración de la base de datos es muy sencilla. El *mapping* de los *beans* hacia la base de datos es el punto fundamental para generar la persistencia automatizada, para entender el nuevo concepto es un poco complicado al principio, debido a la costumbre que existe entre los desarrolladores a programar usando los dos modelos. Al usar el *framework* y generar lo necesario para su funcionamiento es donde se clarifica la manera en que se comunica la aplicación con la base de datos. El *mapping* ofrece atributos muy poderosos como el de borrar en cascada (borra al objeto que se envía así como a los objetos que dependen de él). Otro atributo muy funcional es el de recuperación de los objetos, ya que la herramienta es capaz de recuperar el objeto que se busca incluyendo a los objetos que tienen alguna relación con este.

A lo largo del desarrollo del proyecto se obtuvieron comentarios muy favores con respecto a la necesidad de un repositorio de código fuente de programas en la Universidad lo cual se puede resumir en que en una comunidad de programadores es muy útil contar con un repositorio de programas para poder consultar ejemplos de código fuente, así como el compartirlos para contribuir a una alza en la calidad y alcance de los proyectos.

El uso del patrón de diseño MVC facilita la integración de nuevas funcionalidades al sistema, debido a la separación que existen entre los módulos de programación solo es necesario desarrollar la funcionalidad e integrarlo a las clases correspondientes del paquete *model* , sin tener que hacer modificaciones a las clases de los otros paquetes.

El repositorio para programas de Java implementado da una alternativa para el intercambio de código fuente desarrollado en Java, durante su desarrollo se busco seguir un diseño minimalista, con el fin de proporcionar rapidez y facilidad en las tareas a realizar en el sistema, es evidente que un sistema que pide muchos requisitos y entretiene al usuario mucho tiempo para realizar su función resulta aburrido, frustrante y abrumador. Se desarrolló una funcionalidad que no se ha visto en otro repositorio como es el de compilar el código fuente en el momento de subirlo al servidor.

6.1 Trabajo a futuro.

Al repositorio para programas de Java es una aplicación que permite la extensión a diversas funcionalidades algunas de ellas poden ser:

- La instalación en un servidor de la Universidad de las Américas Puebla para que los estudiantes tengan una nueva alternativa para subir sus programas y realizar las consultas que necesiten.

- La integración con la base de datos estudiantil para el uso del id y contraseña institucional. De esta manera se evitaría el doble registro de los estudiantes y se tendría un mejor control sobre los usuarios del sistema.

- La implementación de los módulos necesarios para ser capaz de manejar paquetes y dependencias entre las clases; para poder subir proyectos grandes incluso tesis de los estudiantes y de está manera aportar más cantidad de código a los estudiantes consultores en el sistema.

- La creación de un manual en línea para resolver las dudas de los usuarios, el sistema tiene un diseño minimalista pero sería un complemento el contar con un manual para resolver las dudas que pudieran aparecer durante su uso.

- La actualización de la información que el usuario le asigna a sus programas por el momento está de manera estática, con esto se quiere decir que el usuario no puede modificar las palabras claves o la descripción hecha sobre algún programa ya registrado, la actualización de la descripción o palabras clave sería de gran utilidad para poder enriquecer la descripción de determinado programa en caso de ser necesario, sin necesidad de borrarlo y volverlo a guardar.

- Hacer pruebas de rendimiento para probar la velocidad con la que realiza los procesos. De esta manera se puede hacer una comparativa entre el sistema usando el ORM *Hibernate* y el sistema sin usar el ORM, para probar la diferencia que pudiera haber entre cada uno.