ERM Development Environment

Este documento detalla:

- Las metodologías y prácticas utilizadas durante la realización del trabajo práctico.
- El diseño de la aplicación y las decisiones que llevaron al mismo.
- Las principales librerías utilizadas para desarrollar la aplicación.

El objetivo es resumir las principales características del trabajo realizado, proveyendo una visión general de los aspectos más relevantes.

Integrantes del Grupo

- Festa Gastón
- Schenkelman Damián
- Soler Francisco
- Storti Santiago

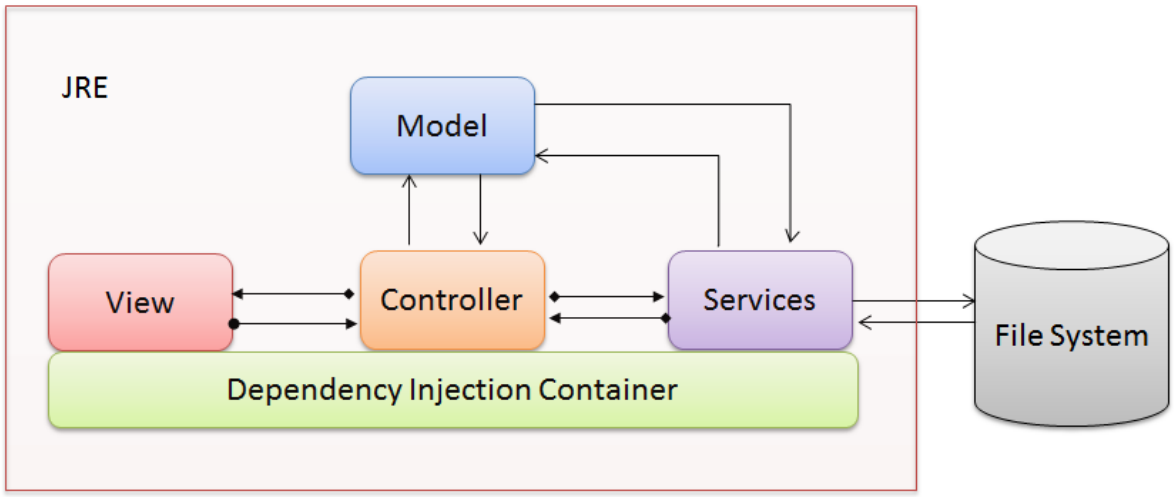
Docente a Cargo

- Lic. Arturo Servetto

Índice

Diseño de la Aplicación

La siguiente figura provee una visualización abstracta del diseño de la aplicación.



Desde el comienzo se determino utilizar el patrón Model View Controller. Las razones para ello fueron:

- El patrón permite abstraer a lógica visual y la de control, simplificando cada componente y clarificando sus responsabilidades.
- Mediante el desarrollo orientado a interfaces en vez de implementaciones concretas permite realizar pruebas unitarias con "objetos mock".
- Comunicación desacoplada, comunicando a la vista y controlador utilizando interfaces o el patrón Observer.
- Reutilización de vistas, como el caso de la vista de Atributos, la cuál se permitió desarrollar una única vista que sirva para administrar los atributos de las entidades y relaciones.

Adicionalmente, como uno de los objetivos del desarrollo era lograr el desacoplamiento de componentes, cada clase recibe en su constructor interfaces, no implementaciones abstractas. Para facilitar la construcción de estos objetos y poder modificar de manera simple y rápida la implementación a utilizar, se decidió utilizar un Dependency Injection Container, PicoContainer. El mismo se configura mediante un Bootstrapper.

Librerías utilizadas

Para el desarrollo de la aplicación se utilizaron las siguientes librerías:

- JUnit 4: Utilizada para las pruebas unitarias creadas para los componentes de la aplicación.
- JGraph: Utilizada para realizar los modelos de Entidades y Relaciones.
- Velocity Engine 1.7: Utilizada para generar reportes HTML de errores y advertencias a partir de templates predefinidos.

Prácticas de desarrollo

Durante el transcurso del trabajo práctico utilizamos las siguientes prácticas.

Test Driven Development (TDD)

La razón principal para utilizar TDD junto con JUnit 4 fue poder tener un conjunto de pruebas automatizadas que nos permitiesen realizar de forma simple regresiones cada vez que se modificaba alguno de los componentes de la aplicación. Esto nos pareció especialmente importante en equipos un equipo distribuido (no trabajamos siempre todos en el mismo lugar, horario) ya que:

- Al realizar un cambió de una funcionalidad desarrollada por otra persona, se puede saber si la funcionalidad existente sigue funcionando.
- Permite evitar largas sesiones de debugging.
- Fuerza a los miembros del equipo a subir código que funciona y con pruebas al repositorio. Durante proyectos anteriores en la facultad, a todos los integrantes del equipo nos ocurrió que algún miembro del equipo subió código nuevo que, o bien no funcionaba, o bien rompía el funcionamiento de código previamente existente.

Creemos que fue una decisión acertada, ya que a lo largo del desarrollo realizamos múltiples cambios a la funcionalidad y el conjunto de pruebas nos permitió mantener constantemente una base de código funcional.

Repositorio SVN

Una buena práctica, que todos los miembros del equipo ya habíamos utilizado, fue el usar un repositorio centralizado de control de versiones. Para trabajos en equipo que involucran gran cantidad de archivos de código fuente, tener forma de acceder al código desde Internet, así como también tener control de los cambios de los archivos es algo fundamental.

Utilizamos el hosting gratis de Google Code con un repositorio SVN.

Backlog Online

Para mantener control de las prioridades y las asignaciones de tareas utilizamos un backlog online, provisto por Asana. Esta herramienta nos permitió enfocarnos constantemente en la tarea más importante, tener trazabilidad de las actividades realizadas y aquellas pendientes, y mantener

claras las responsabilidades de cada uno de los integrantes del equipo.