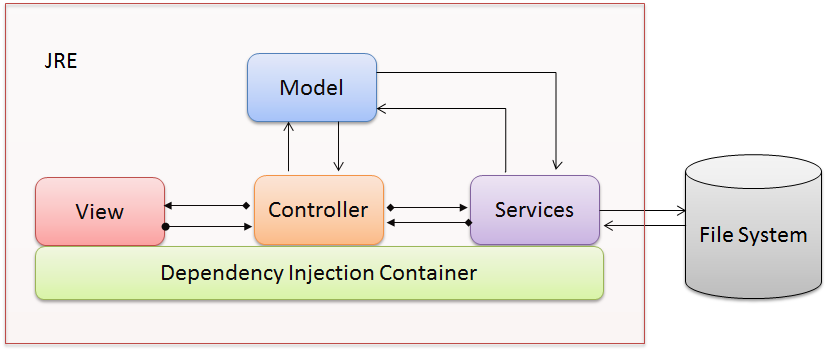
ERM Development Environment

Este documento explica la metodología usada para el trabajo práctico, el diseño de la aplicación y las decisiones que llevaron al mismo y también lista las principales librerías utilizadas para realizar la aplicación.

El objetivo es resumir las principales características del trabajo realizado.

# 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](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller). Las razones para ello fueron:

* El patrón permite abstraer a lógica visual y la de control, simplificando las cada componente y clarificando sus responsabilidades.
* Mediante el desarrollo orientado a interfaces en vez de implementaciones concretas permite realizar pruebas unitarias con "[objetos mock](http://en.wikipedia.org/wiki/Mock_object)".
* Comunicación desacoplada, comunicando a la vista y controlador utilizando interfaces o el patrón [Observer](http://en.wikipedia.org/wiki/Observer_pattern).
* 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](http://en.wikipedia.org/wiki/Dependency_injection), [PicoContainer](http://picocontainer.codehaus.org/). El mismo se configura mediante un [Bootstrapper](http://en.wikipedia.org/wiki/Bootstrapping).

# Librerías utilizadas

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

* [JUnit 4](http://www.junit.org/): Utilizada para las pruebas unitarias creadas para los componentes de la aplicación.
* [JGraph](http://www.jgraph.com/): Utilizada para realizar los modelos de Entidades y Relaciones.
* [Velocity Engine 1.7](http://velocity.apache.org/engine/releases/velocity-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](http://en.wikipedia.org/wiki/Test-driven_development) junto con [JUnit 4](http://www.junit.org/) 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.
* 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

## Repositorio SVN

## Backlog