# Actualización de la Arquitectura

Luego de realizar una investigación más intensiva sobre la tecnología GWT, se entendió que se debía modificar la arquitectura inicialmente propuesta para lograr una mejor performance y eliminar framework no requeridos.

El siguiente diagrama muestra la arquitectura actualizada para el sistema Tempore.

**<GWT>**

**<Client>**

Widgets – CSS – Image - JavaScript

**<Server>**

Servlet

**<Services>**

Java Class

**<DAOs>**

ORM - Hibernate

**<Base de Datos>**

Tablas

**<DTO>**

POJO Class

**<Entity>**

POJO Class Mapping

La principal modificación a la propuesta inicial fue la eliminación del framework Spring como implementador del patrón MVC.

Esta eliminación tiene como objetivo lograr una aceleración en la respuesta de la aplicación al no requerir una capa de framework tan robusta como la presta Spring. En reemplazo a esta eliminación se decide utilizar el mismo componente inicialmente propuesto, GWT (Google Web Toolkit),

# Responsabilidad de cada Layer

GWT

#### Client

Capa responsable del control dinamico de la vista. Apartir de componentes como widgets, css e imágenes, es la encargada de generar las pantallas que serán visualizadas. Ademas será la encargada de ciertas validaciones básicas y de controlar las peticiones a la capa se servico, de manera que administrara el patrón MVC para esta acción.

La comunicación con el servidor será por medio de XML-RPC

#### Server

Esta capa será la responsable de la administración de los servicios. De esta manera queda aislada la vista del modelo. Esta capa estará compuesta por Servlets que servirán de interfaz para el *Front-End*.

Services

Servicio encargado de la validación, lógica de negocio y transaccionabilidad de las reglas de negocio.

DAOs

Data acces object. Capa encargada de canalizar la persistencia. Esta capa es la única que tendrá acceso a la administración de datos de la base de datos.

Base de Datos

Entidad encargada de la persistencia física de la información y estados de la aplicación.

Entity

Entidad utilizada por los DAOs para realizar los mapeos entre las clases Java y las tablas de la base de datos.

DTOs

Clases Pojos que tiene la responsabilidad de comunicar el front-end con el back-end. Para nuestro caso esta capa será la encargada de pasar la información de la vista hacia la capa de servicios.

# Patrones

Ya hemos mencionado **MVC** como patrón utilizado por GWT.

Asimismo nos hemos valido de los patrones como el **decorator** para extender la funcionalidad de los componentes de GWT (TaskTreePanel para extender TreeGrid, TaskTabPanel, ProjectTabPanel, UserConfigurationPanel, AlertConfigurationPanel, etc. para extender Canvas) **command** y **singleton** para los servicios, **observer** para las notificaciones del panel de proyectos, **facade** para compartir una interfaz común para los paneles de la aplicación.