# Diagrama de clases

Para realizar la implementación de las principales funcionalidades del proyecto hicimos un previo análisis para determinar que patrón utilizar, el mismo fue **MVC** y **DAO**, pero...

# ¿Por qué MVC?

Para el diseño de aplicaciones con sofisticadas interfaces es muy recomendable usar el patrón de diseño Modelo-Vista-Controlador.

La lógica de una interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Si realizamos un diseño pobre, es decir que mezcle los componentes de la interfaz y del negocio, entonces la consecuencia será que, cuando necesitemos cambiar la interfaz, tendremos que modificar arduamente los componentes de negocio lo cual generará mayor trabajo y más riesgo de error.

* **Modelo**: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento del controlador o la vista.
* **Vista**: Muestra la información a través de una interfaz de usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador con el que interactúa.
* **Controlador**: Reciben las entradas de las vistas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio para el modelo o la vista.

# ¿Por qué DAO?

El problema que viene a resolver este patrón es el de incluir diversas fuentes de datos (base de datos, archivos, servicios externos, etc) a la lógica de negocio y a su vez encapsulando la forma de acceder a dichas fuentes.

# Ahora sí, diagrama de clases.

Sabiendo las ventajas que ofrecen estos patrones, los combinamos y utilizamos, quedando el diagrama de clases de la siguiente manera

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente