**MODELO – VISTA – CONTROLADOR**

(Patrón de arquitectura de software)

MVC propone la construcción de tres componentes distintos que son el **modelo**, la **vista** y el **controlador**, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

**DESCRIPCION:**

**MODELO**

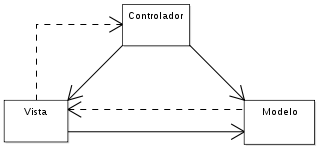
* Gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación
* Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'

**CONTROLADOR**

* Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información.
* Podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo‘.

**VISTA**

* Presenta el 'modelo' (información y *lógica de negocio*) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.



**OBJET POOL**

(Piscina de objetos)

El patrón **object pool** es un patrón de diseño de software que usa un conjunto de objetos inicializados preparados para su uso. Esto es más efectivo normalmente que creando y destruyendo los objetos bajo demanda.

* Los objetos no se crean (salvo la primera vez) ni se destruyen, simplemente se van reciclando.
* Este patrón de diseño suele mejorar significativamente el rendimiento en situaciones donde el coste de iniciar una instancia de clase es alto y cuando el número de instancias simultáneas no es muy alto.
* Es común usar los Object Pools para la creación de recursos gráficos, conexiones a base de datos, sockets y en general cualquier objeto cuya creación es costosa. En ciertas ocasiones el uso de este patrón es más costoso.

**PROS**

* Cuando es necesario trabajar con un gran número de objetos que son particularmente costosos para instanciar y cada objeto solo se necesita durante un corto período de tiempo, el rendimiento de una aplicación completa puede verse afectada negativamente. Un patrón de diseño object pool puede resultar útil en casos como estos.

