

Patrones de diseño de software.

POO



25 de octubre de 2021

Fundación universitaria compensar

OSCAR JAVIER GALVIS

# Introducción

En el mundo de la programación, es común el termino de resolución de problemas, día a día se trabaja en ello para brindar una mejor manera de ejecutar tareas o procesos. Por esto es imprescindible que todo desarrollador conozca, comprenda y tenga la facultad de implementar patrones de diseño de software, de esta manera, la comunicación entre desarrolladores será más comprensible ya que independiente de lo que se esté creando, se trabajará bajo el mismo marco.

Objetivos

* Investigar el concepto y estructura de los patrones de diseño de software.
* Comprender la importancia que representa el trabajar con las diferentes técnicas que otorgan dichos patrones.
* Fomentar la inclusión de dichos patrones, con el fin de obtener un código más claro, limpio y mejor estructurado.

Definición

Se entiende por patrones de diseño, a una serie de técnicas que permiten resolver problemáticas del común en lo que respecta al desarrollo de software. Lo ideal de estos patrones es dejar una huella en cada implementación de código, mediante la cual, otros desarrolladores puedan reutilizar y extraer la solución que se implementó en principio para resolver dicho problema. En la programación orientada a objetos, se evidencian tres tipos de patrones, lo cuales se explican de manera breve a continuación:

Patrones creacionales, que hacen referencia a la instanciación directa de los objetos mediante su constructor:

* Abstract factory
* Builder
* Factory Method
* Prototype
* Singleton
* Model View Controller

Patrones estructurales, que se encargan de la creación de clases y objetos, creando independencia dentro del proyecto, de tal manera que, si alguna de las otras partes es modificada, estas permanezcan intactas:

* Adapter
* Composite
* Bridge
* Decorator
* Facade
* Flyweight
* Proxy
* Module

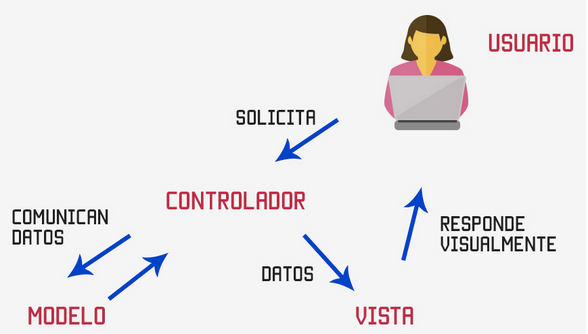
patrones de comportamiento, que definen la manera en que interactúan las clases y objetos:

* Chain of responsability
* Command
* Iterator
* Interpreter
* Memento
* Mediator
* Observer
* State
* Strategy
* Template method
* Visitor

El ideal es tener una noción de cómo utilizar o implementar estos patrones según el propósito, lo cual requiere un poco más de énfasis en cada tipología listada y estudiada a fondo. De momento se toma como ejemplo el patrón MVC de tipo creacional, en el que se hará énfasis hacia la programación orientada a objetos con Java.

Patrón MVC

El MVC se define como un estándar de arquitectura de software que separa la lógica de una aplicación, de la lógica de vista en una aplicación utilizando 3 componentes. vistas, modelos y controladores. El modelo, alberga la representación de toda la información; la vista contiene todas aquellas interacciones que el usuario tiene con el sistema, y el controlador, es el encargado de recibir dichas interacciones y comunicarlas con el modelo. De esta manera cuando se realiza algún cambio en el código, no afecte otra parte de este, respetando el principio de responsabilidad única que se basa en afirmar que una parte del código no debe saber qué es lo que hace toda una aplicación y solamente tener una responsabilidad.

  
tomada de: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

# DTO y DAO

Dentro de las buenas prácticas integradas a nivel general para la construcción de capa de negocio, es importante tomar en cuenta algunas pautas que determinan dicho proceso. Con ello se pretende obtener la separación entre controlador y modelo, previniendo así cualquier cambio o modificación que se pueda dar en la forma que se accede a los datos.

DAO: Mediante este componente de software accedemos de manera controlada y única a los datos a través de métodos definidos previamente en esta estructura. Además, actúa como un adaptador entre el componente y la fuente de datos

DTO: Se consideran como un tipo de objeto que permite transportar los datos, allí se almacenan las propiedades de los objetos, proveyendo así, acceso a las propiedades del bean. Sus características principales son:

**Serializable**: Todos los objetos y sus atributos debe ser fácilmente codificables, debido a que tendrán que viajar por la red.

**Solo lectura:** Esto implica que no se realice ninguna operación directamente sobre los datos, por ello es importante tener definidos y declarados los métodos set y get de sus respectivos atributos.

Conclusiones

* A través de la investigación realizada, se logra tener una claridad conceptual de lo que compone el tema de patrones de diseño de software.
* Se consigue comprender el objetivo que tienen los patrones de diseño, los cuales se enfocan en obtener un mejor código con más claridad y fácil de interpretar, a medida que se avanza e incursiona en la práctica.

Referencias Bibliograficas