



Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😊 

➤ El patrón de diseño MVC

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

- ✓ Patrón de diseño DAO
- ✓ Patrón de diseño DTO

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

5.5

El patrón de diseño MVC

Modelo Vista-Controlador

MVC

Qué es un patrón de diseño
En qué consiste el patrón MVC
Beneficios de utilizar MVC

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



Comprender el concepto de *Patrón de Diseño*



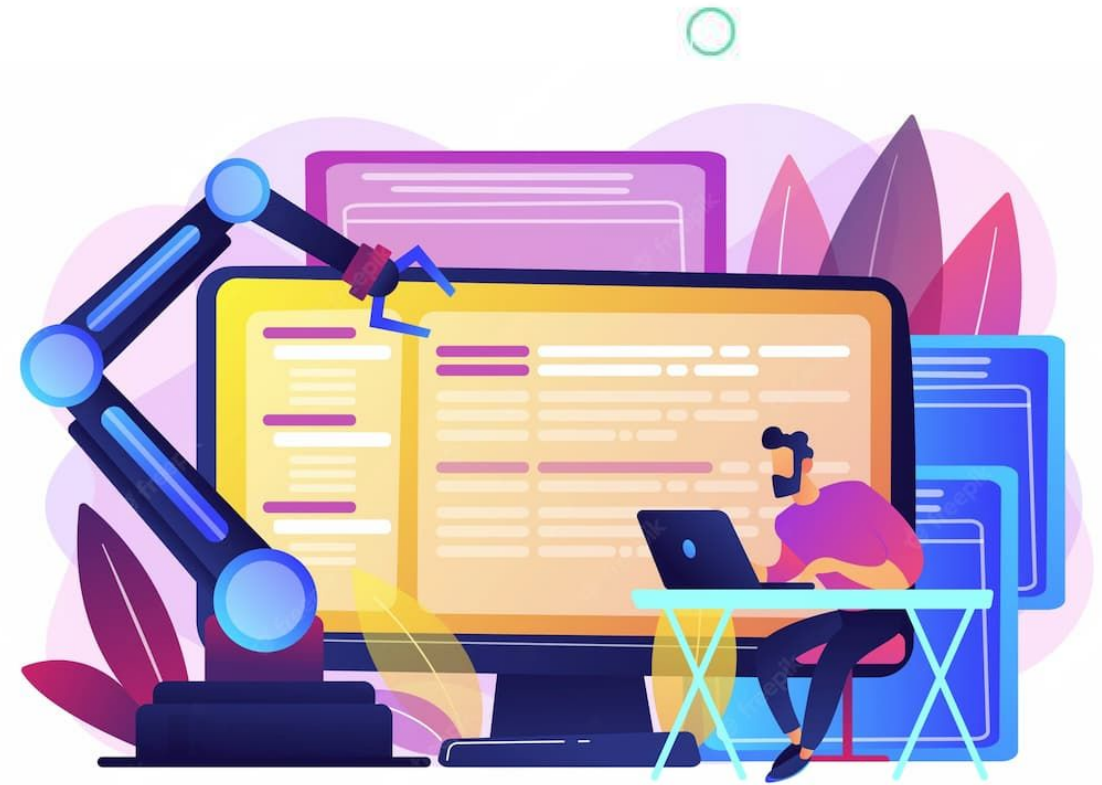
**Aprender en qué consiste el modelo
*vista controlador***



› ¿Qué es un patrón de diseño?

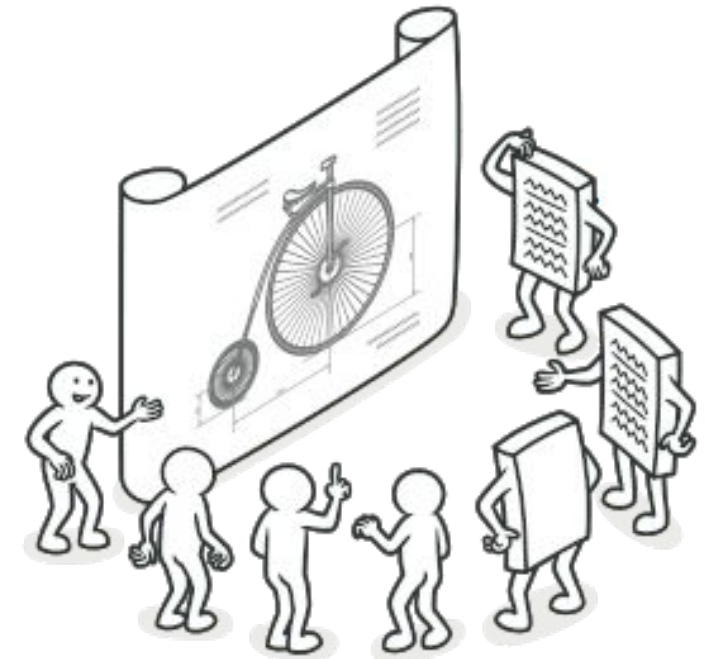
¿Qué es un patrón de diseño?

Un patrón de diseño es una solución reutilizable y general para un problema recurrente en el desarrollo de software. Los patrones de diseño proveen templates predefinidos para resolver situaciones comunes en la construcción de aplicaciones, permitiendo resolver estos problemas de manera eficiente y probada.



¿Qué es un patrón de diseño?

Los patrones de diseño surgieron a partir de la necesidad de contar con soluciones estandarizadas para problemas habituales como creación de objetos, estructuración de clases e interfaces o relaciones entre componentes. En lugar de resolver estas cuestiones una y otra vez, los patrones capturan las mejores prácticas en formatos reutilizables.



¿En qué consiste el patrón?

La mayoría de los patrones se describe con mucha formalidad para que la gente pueda reproducirlos en muchos contextos. Aquí tienes las secciones que suelen estar presentes en la descripción de un patrón:

- El propósito del patrón explica brevemente el problema y la solución.
- La motivación explica en más detalle el problema y la solución que brinda el patrón.
- La estructura de las clases muestra cada una de las partes del patrón y el modo en que se relacionan.
- El ejemplo de código en uno de los lenguajes de programación populares facilita la asimilación de la idea que se esconde tras el patrón.

Algunos catálogos de patrones enumeran otros detalles útiles, como la aplicabilidad del patrón, los pasos de implementación y las relaciones con otros patrones.



Características principales



Algunas características de los patrones de diseño:

- Son reutilizables en distintos contextos y lenguajes de programación.
- Resuelven problemas recurrentes en el desarrollo de software.
- Facilitan el uso de buenas prácticas probadas y estandarizadas.
- Se basan en principios de diseño orientado a objetos.
- Mejoran la legibilidad y mantenimiento del código.



En resumen, los patrones de diseño como MVC, Singleton, Observer y otros tantos, nos permiten resolver de forma óptima y reutilizable problemas comunes de diseño en nuestras aplicaciones.





¿Por qué debería aprender sobre patrones?



La realidad es que podrías trabajar durante años como programador sin conocer un solo patrón. Mucha gente lo hace. Incluso en ese caso, podrías estar implementando patrones sin saberlo. Así que, ¿por qué dedicar tiempo a aprenderlos?





- Los patrones de diseño son un juego de herramientas de soluciones comprobadas a problemas habituales en el diseño de software. Incluso aunque nunca te encuentres con estos problemas, conocer los patrones sigue siendo de utilidad, porque te enseña a resolver todo tipo de problemas utilizando principios del diseño orientado a objetos.





¿Por qué debería aprender sobre patrones?

- 
- Los patrones de diseño definen un lenguaje común que puedes utilizar con tus compañeros de equipo para comunicaros de forma más eficiente. Podrías decir: “Oh, utiliza un singleton para eso”, y todos entenderían la idea de tu sugerencia. No habría necesidad de explicar qué es un singleton si conocen el patrón y su nombre.
- 





Clasificación de los patrones

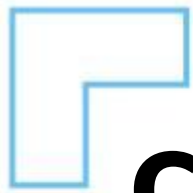


Los patrones de diseño varían en su complejidad, nivel de detalle y escala de aplicabilidad al sistema completo que se diseña. Los patrones más básicos y de más bajo nivel suelen llamarse **idioms**. Normalmente se aplican a un único lenguaje de programación.



Los patrones más universales y de más alto nivel son los patrones de arquitectura. Los desarrolladores pueden implementar estos patrones prácticamente en cualquier lenguaje. Al contrario que otros patrones, pueden utilizarse para diseñar la arquitectura de una aplicación completa.





Clasificación de los patrones



Además, todos los patrones pueden clasificarse por su propósito. Veamos tres grupos generales de patrones:

- Los **patrones creacionales** proporcionan mecanismos de creación de objetos que incrementan la flexibilidad y la reutilización de código existente.
- Los **patrones estructurales** explican cómo ensamblar objetos y clases en estructuras más grandes a la vez que se mantiene la flexibilidad y eficiencia de la estructura.
- Los **patrones de comportamiento** se encargan de una comunicación efectiva y la asignación de responsabilidades entre objetos.



➤ El patrón MVC



Patrón MVC



MVC era inicialmente un patrón arquitectural, un modelo o guía que expresa cómo organizar y estructurar los componentes de un sistema software, sus responsabilidades y las relaciones existentes entre cada uno de ellos.



Su nombre, MVC, parte de las iniciales de **Modelo-Vista-Controlador** (Model-View-Controller, en inglés), que son las capas o grupos de componentes en los que organizaremos nuestras aplicaciones bajo este paradigma.

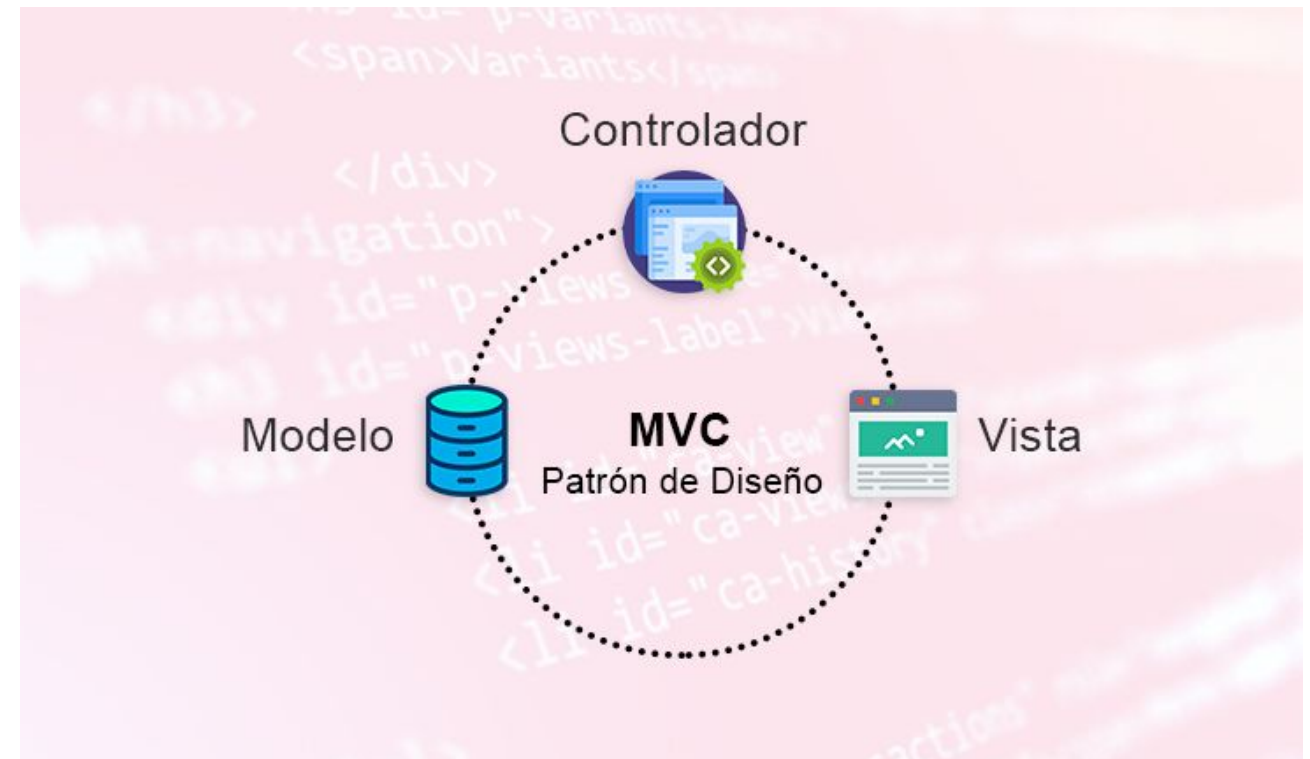
Es a menudo considerado también un patrón de diseño de la capa de presentación, pues define la forma en que se organizan los componentes de presentación en sistemas distribuidos.





Patrón MVC

La arquitectura MVC propone, independientemente de las tecnologías o entornos en los que se base el sistema a desarrollar, la separación de los componentes de una aplicación en tres grupos (o capas) principales: el modelo, la vista, y el controlador, y describe cómo se relacionarán entre ellos para mantener una estructura organizada, limpia y con un acoplamiento mínimo entre las distintas capas.





Patrón MVC

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.



Las tres partes del patrón de diseño de software MVC se pueden describir de la siguiente manera:



- **Modelo:** Maneja datos y lógica de negocios.
- **Vista:** Se encarga del diseño y presentación.
- **Controlador:** Enruta comandos a los modelos y vistas.





Beneficios de utilizar MVC



Utilizar el patrón MVC en aplicaciones web aporta varios beneficios:

- Permite separar la lógica de negocio de la interfaz de usuario, resultando en una mejor organización del código.
- Promueve la reutilización de código, por ejemplo se puede usar el mismo modelo para distintas interfaces.
- Facilita el mantenimiento y actualización, ya que se pueden modificar el modelo, vista o controlador de forma independiente.
- El desarrollo se puede dividir por roles: programadores de backend en el modelo, diseñadores frontend en la vista, etc.





Beneficios de utilizar MVC



Utilizar el patrón MVC en aplicaciones web aporta varios beneficios:

- Los cambios tienen menor impacto al estar desacoplados en capas separadas.
- Permite escalar cada componente de forma independiente.
- Es más fácil realizar pruebas unitarias enfocadas en cada capa.
- La interfaz visual queda abstracta del modelo de datos subyacente.
- Promueve un acoplamiento más débil entre los componentes.
- Siguiendo estándares resulta en un código más claro y mantenible.



Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro...

Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase.





Ejercicio N° 1

MVC

MVC



Manos a la obra: 🙌

En este ejercicio vamos diseñar las clases necesarias para interactuar con las operaciones de acceso a datos (DAO) de usuarios, cuentas y divisas. Para ello usaremos una nueva base de datos llamada "Wallet", con la cual deberás crear las clases correspondientes para gestionar la interacción con los UserDAO, AccountDAO y CurrencyDAO.

Consigna: 📝

- 1- Crear las clases necesarias para interactuar con UserDAO, AccountDAO y CurrencyDAO .
- 2- Crear una nueva BD Wallet e insertar registros.
- 3- Modificar el nombre de al menos dos usuarios.

Tiempo 🕒: 30 minutos

○

¿Alguna consulta?

+



RESUMEN

¿Qué logramos en esta clase?

- ✓ **Comprender la implementación del patrón DAO**
- ✓ **Aprender el concepto de los DTOs.**



#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 📌 📌 📌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
 - a. *Lectura Modulo 5, Lección 5: páginas 1 - 3*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 🙌



Momento: ✚

Time-out!

🕒 5 min.

