UNIVERSIDAD DE ANTIOQUIA
1803

PATRÓN MVC

POR
ASTRID CAROLINA GOMEZ
LUIS CARLOS MARÍN CAMPOS
ANDRES DARIO HIGUITA PEREZ
MARIO DE JESUS PUENTES

Udea
Educación Virtual

¿Qué es?

Es un patrón arquitectónico desarrollado en 1979 que separa la lógica de una aplicación, manteniendo distintas capas que se encargan de hacer una tarea muy concreta, esto permite realizar proyectos escalables y modulables.

¿Qué es?

Es uno de los patrones más conocidos en la web, ya que muchos framework lo usan.

Ruby on rail, Sinatra - Ruby

Flask - Python

CakePhp, Laravel - PHP

¿Cuándo es necesario usarlo?

Se usa inicialmente en sistemas donde se requiere el uso de interfaces de usuario, aunque en la práctica el mismo patrón de arquitectura se puede utilizar para distintos tipos de aplicaciones.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores.

¿Cuáles son sus capas y funcionamiento?

Modelo

Esta es la capa donde se trabaja con los datos, contiene los diferentes mecanismos con los cuales se puede acceder a la información y actualizar sus estados.

Si se trabaja con bases de datos, en esta es donde se realizan las funciones con las cuales se hacen consultas, búsquedas, filtros y actualizaciones hacia las tablas.

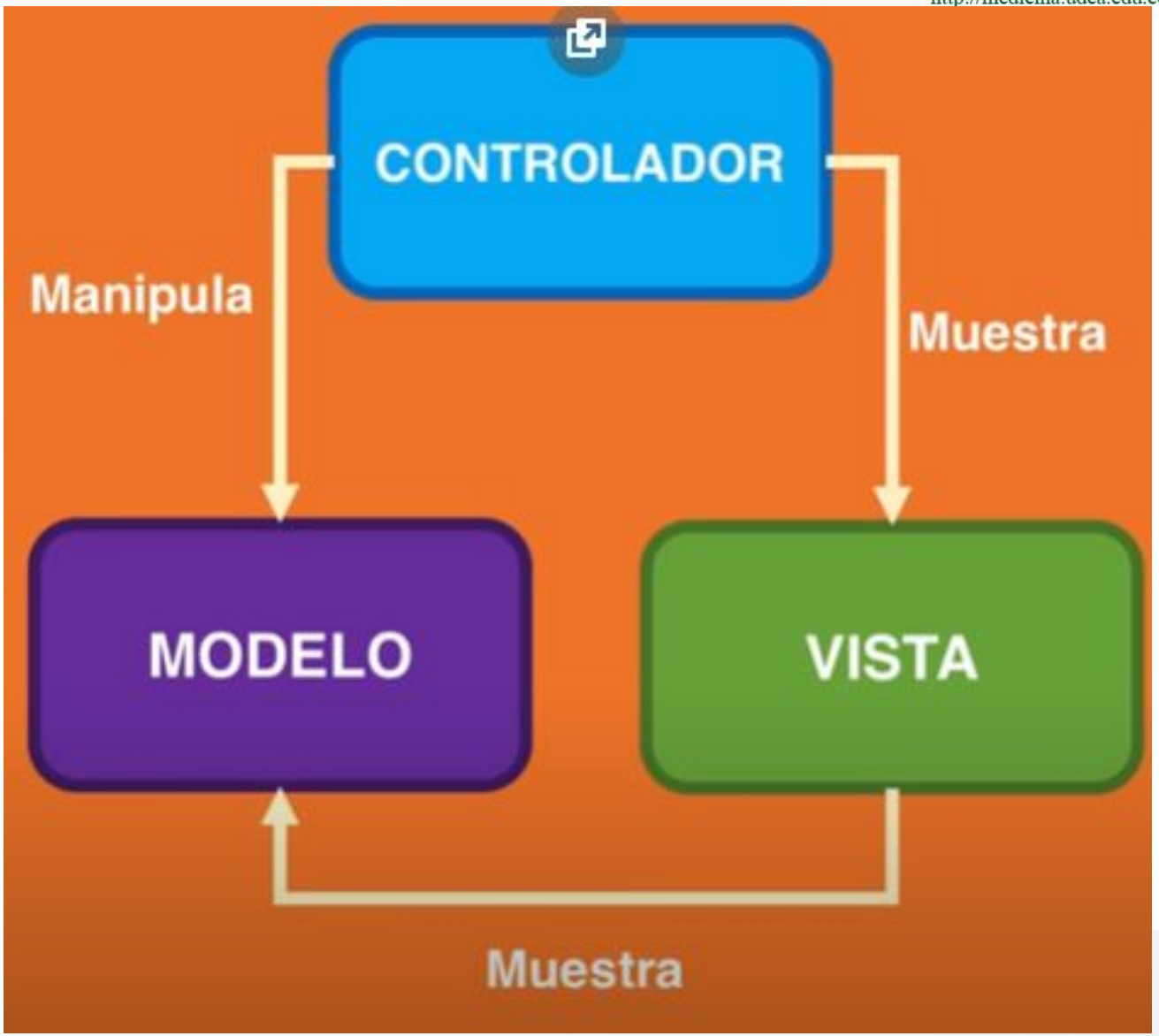
Vista

En esta capa se encuentra todo los que tenga que ver con la interfaz gráfica, por lo tanto, ni la capa de modelo, ni el controlador se preocupan de cómo se ven los datos, siendo esta una responsabilidad exclusiva de la capa de vista.

Para nosotros los desarrolladores y programadores esta capa se traduciría como en el componente que se encarga del frontend de una aplicación, puede ser para una aplicación de escritorio, como también lo puede ser para páginas web (CSS, HTML, HTML5 y JavaScript).

Controlador

Esta capa la podemos imaginar que se encuentre posicionada entre la capa de modelo y la capa de vista, entre ellas esta capa se encuentra gestionando instrucciones que se reciben, atenderlas y procesarlas. Esta capa se encuentra como medio de comunicación entre estas capas solicitando los datos necesarios, para manipularlos y así obtener resultados para que por último sean entregados a la vista para que puedan ser mostrados.


CONTROLADOR
Manipula
Muestra
MODELO
VISTA
Muestra

¿Por qué es importante usar MVC?

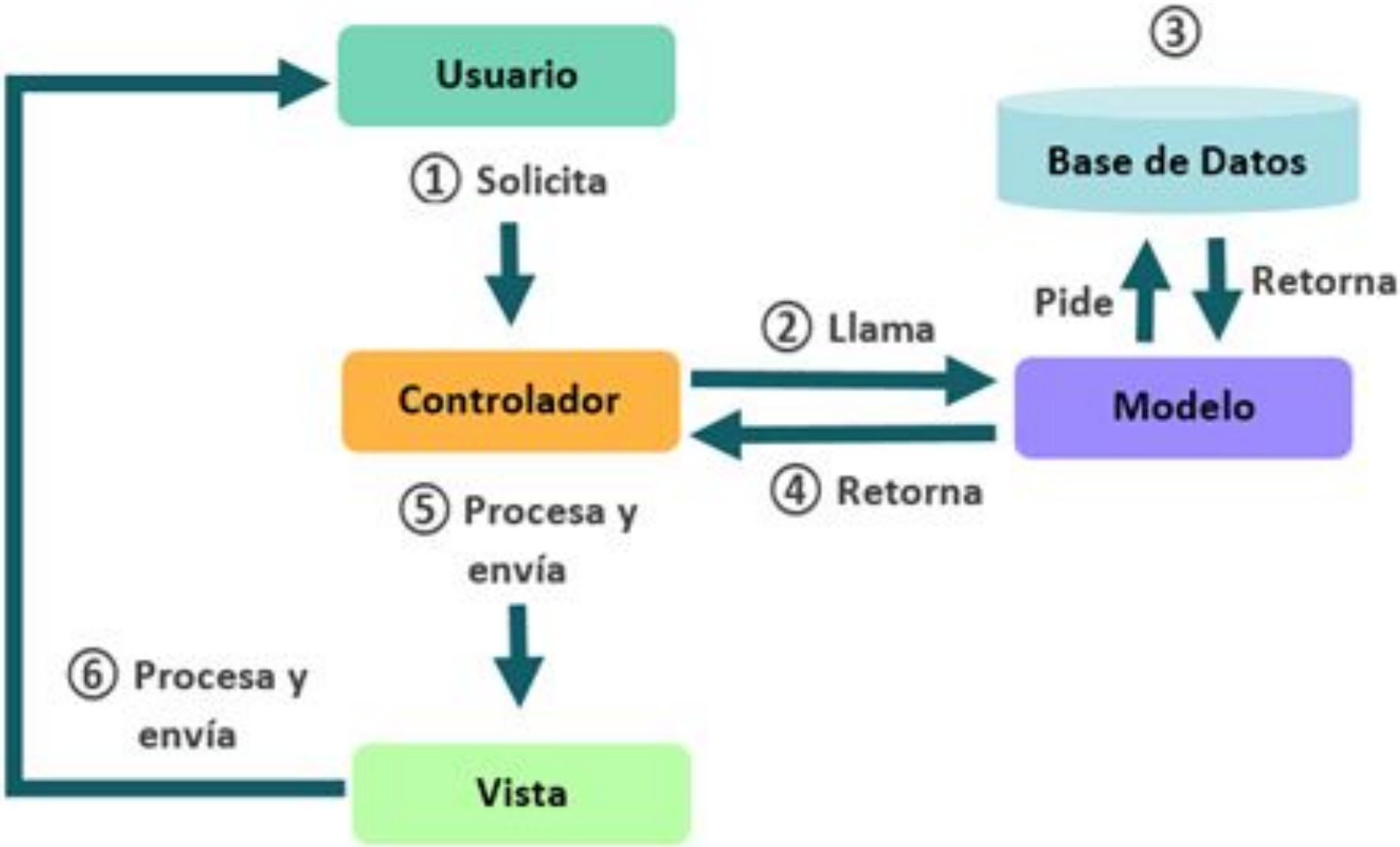
Proceso de desarrollo más rápido: MVC apoya el desarrollo rápido y paralelo, ya que, al utilizar el patrón, se desarrolla de una forma más eficiente debido a que una persona puede trabajar en la vista, mientras que otra puede trabajar en el controlador y así crear la lógica empresarial de la aplicación.

Modularidad: Nos permite separar los componentes de nuestra aplicación dependiendo de la responsabilidad que tienen. Si modificamos nuestra Base de Datos, sólo deberíamos modificar el modelo que es quién se encarga de los datos y el resto de la aplicación debería permanecer intacta. Esto respeta el principio de la responsabilidad única.

Soporte para la técnica asincrónica: es compatible con la técnica asíncrona, la cual ayuda al programador a desarrollar, y permite que la aplicación pueda tener un rendimiento superior al cargar su contenido.

Ciclo de vida del MVC.

- El usuario realiza una petición.
- El controlador captura la petición del usuario.
- El controlador llama al modelo.
- El modelo interactúa con la base de datos, y retorna la información al controlador.
- El controlador recibe la información y la envía a la vista.
- La vista procesa la información recibida y la entrega de una manera visualmente entendible al usuario.


Usuario
① Solicita
Controlador
② Llama
Modelo
③
Base de Datos
Pide
Retorna
④ Retorna
⑤ Procesa y envía
⑥ Procesa y envía
Vista

Ventajas de MVC

Las principales ventajas del uso del patrón MVC son:

- La separación del Modelo y la Vista, lo cual logra separar los datos, de su representación visual.
- Facilita el manejo de errores.
- Permite que el sistema sea escalable si es requerido.
- Es posible agregar múltiples representaciones de los datos.

Desventajas de MVC

Las principales desventajas del uso del patrón MVC son:

- La cantidad de archivos que se deben mantener incrementa considerablemente.
- La curva de aprendizaje es más alta que utilizando otros modelos.
- Su separación en capas, aumenta la complejidad del sistema.

Implementación del patrón de diseño MVC.

Para comprender mejor cómo funcionan los componentes del patrón de diseño MVC, se realizará el siguiente ejercicio.

1. Crear modelo

```java
public class Intern {
    private String name;
    private String lastname;
    private String city;
    private String phone;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }
}
```

UNIVERSIDAD DE ANTIOQUIA
1803

2. Crear vista

```java
public class InternView {
    public void printInternDetails(String internName, String internLastname){
        System.out.println("Intern: ");
        System.out.println("Name: " + internName);
        System.out.println("Lastname: " + internLastname);
    }
}
```

UNIVERSIDAD DE ANTIOQUIA
1803

3. Crear controlador

```java
public class InternController {
    private Intern model;
    private InternView view;

    public InternController(Intern model, InternView view){
       this.model = model;
       this.view = view;
    }

    public String getInternName() {
        return model.getName();
     }

     public void setInternName(String name){
        model.setName(name);
     }

     public String getInternLastname() {
        return model.getLastname();
     }

     public void setInternLastname(String lastname) {
        model.setLastname(lastname);
     }

     public void updateView(){
       view.printInternDetails(model.getName(), model.getLastname());
    }
 }
```

UNIVERSIDAD DE ANTIOQUIA
1803

4. Crear clase principal

```
1  public class MainMVC{
2      public static void main(String[] args) {
3        Intern model = retriveInternFromDatabase();
4         InternView view = new InternView();
5         InternController controller = new InternController(model, view);
6         controller.updateView();
7         controller.setInternName("John");
8         controller.updateView();
9      }
10
11     public static Intern retriveInternFromDatabase(){
12        Intern intern = new Intern();
13        intern.setName("Jane");
14        intern.setLastname("Doe");
15        return intern;
16     }
17  }
```

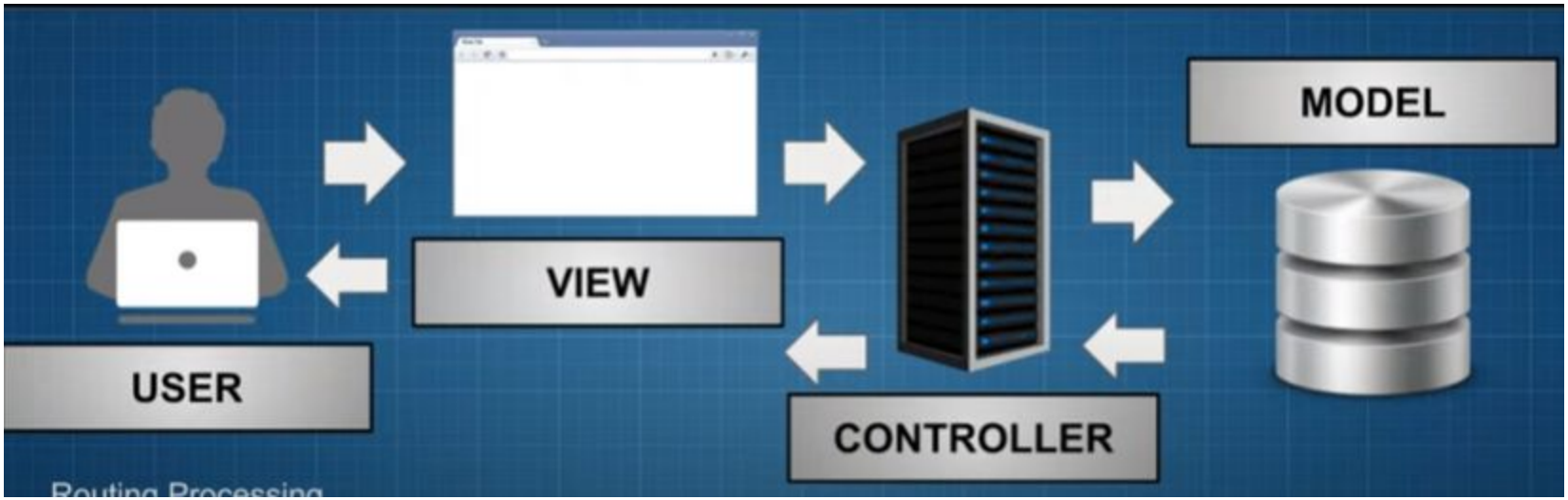
UNIVERSIDAD DE ANTIOQUIA
1803

5. Verificar resultado

```
→  MVCExample java MainMVC
Intern:
Name: Jane
Lastname: Doe
Intern:
Name: John
Lastname: Doe
→  MVCExample
```

Resultados de la implementación del patrón de diseño MVC.

UNIVERSIDAD DE ANTIOQUIA
1803


MODEL
VIEW
USER
CONTROLLER
Routing Processing