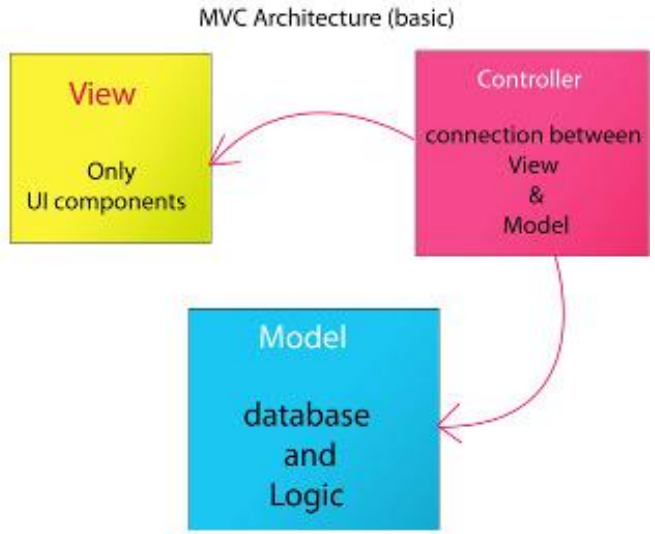
Tema 1: Ficheros

Repaso Modelo Vista Controlador (MVC)

Patrón Modelo - Vista – Controlador (MVC)

- Patrón de diseño mediante el cual en una aplicación gráfica:
  - Se separa la lógica de la presentación de los datos
  - Facilita la reutilización de código y el mantenimiento de la aplicación.

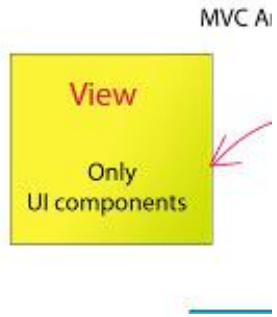


Se separa la aplicación en 3 capas:

- **Vista:** Referencia a la ventana principal de la aplicación (GUI). En MVC, la vista solo contiene el código referente a la construcción y organización de los componentes gráficos.
- **Modelo:** es la capa que contiene los datos de la aplicación y los gestiona. El modelo contiene todos los métodos para realizar las operaciones de nuestra aplicación. Dar de alta elementos, eliminar, buscar, guardar, cargar, etc.
- **Controlador:** es la capa que comunica la vista y el modelo. En su constructor, se le pasa como parámetro una instancia de la vista y una instancia del modelo. El controlador es quien tiene implementados los manejadores de eventos y se los añade a los componentes de la vista indicados, y también es quien ejecuta las operaciones del modelo en respuesta a esos eventos.

Patrón Modelo - Vista – Controlador (MVC)

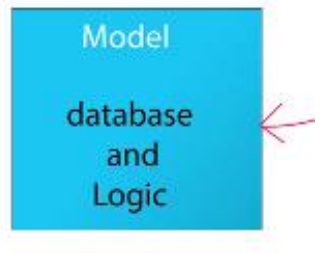
- Vista
  - Contiene todos los elementos gráficos (botones, campos de texto…)




```java
public class Vista extends JFrame {

	JPanel contentPane;
	JTextField textNombre;
	JTextField textApellidos;
	... //Todos los componentes swing que gestione desde el controlador

	public Vista(){
		initComponents();
	}

	public void initComponentes(){

		setResizable(false);
		setTitle("Mi Aplicacion");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		... //resto del código generado por WindowBuilder

		set visible(true);
	}
}
```

Patrón Modelo - Vista – Controlador (MVC)

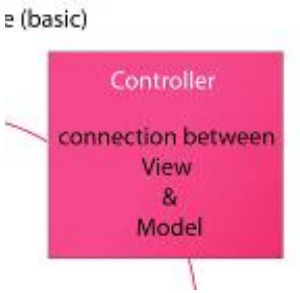
- Modelo
  - El modelo es la clase que maneja los datos de la aplicación y permite las operaciones.




```java
public class Modelo{

	//Lista con los datos que gestiona mi aplicación

	private ArrayList<Persona> listaPersonas;
	...
	public Modelo(){
		listaPersonas = new ArrayList<Persona>;
	}
	 //Métodos para acceder a los datos
	public ArrayList<Persona> obtenerPersonas(){
		return listaPersonas;
	}

	//Operaciones sobre los datos
	public void altaPersona(String nombre, int edad){
		Persona nuevaPersona = new Persona(nombre, edad);
		listaPersonas.add(nuevaPersona);
	}
	...
}
```

Patrón Modelo - Vista – Controlador (MVC)

- Controlador
  - La clase Controlador añade los manejadores de eventos (listeners) a cada elemento de la vista que lo provoque.




```java
public class Controlador{
	private Vista vista;
	private Modelo modelo;

	//Constructor recibe instancia de la vista y del modelo
	public Controlador(Vista vista, Modelo modelo){
		this.vista = vista;
		this.modelo = modelo;
		initEventHandlers();
	}

	//Método para añadir los listeners a los elementos que las generan (botones)
	public void initEventHandlers(){
		vista.btnAceptar.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
			... //Código que quiero que ocurra al pulsar el botón de aceptar
			}
		});

	//Añadir tantos manejadores de eventos como acciones pueda realizar
	...
	}
}
```

Patrón Modelo - Vista – Controlador (MVC)

- Arrancar una aplicación MVC
  - Para arrancar la aplicación es necesario llamar al constructor del **controlador**, que recibe como parámetros una instancia de la vista y una instancia del **modelo**

```
public class Principal(){

	public static void main(String[] args){

		Vista vista = new Vista();
		Modelo modelo = new Modelo();
		Controlador controlador = new Controlador(vista, modelo);
	}
}
```