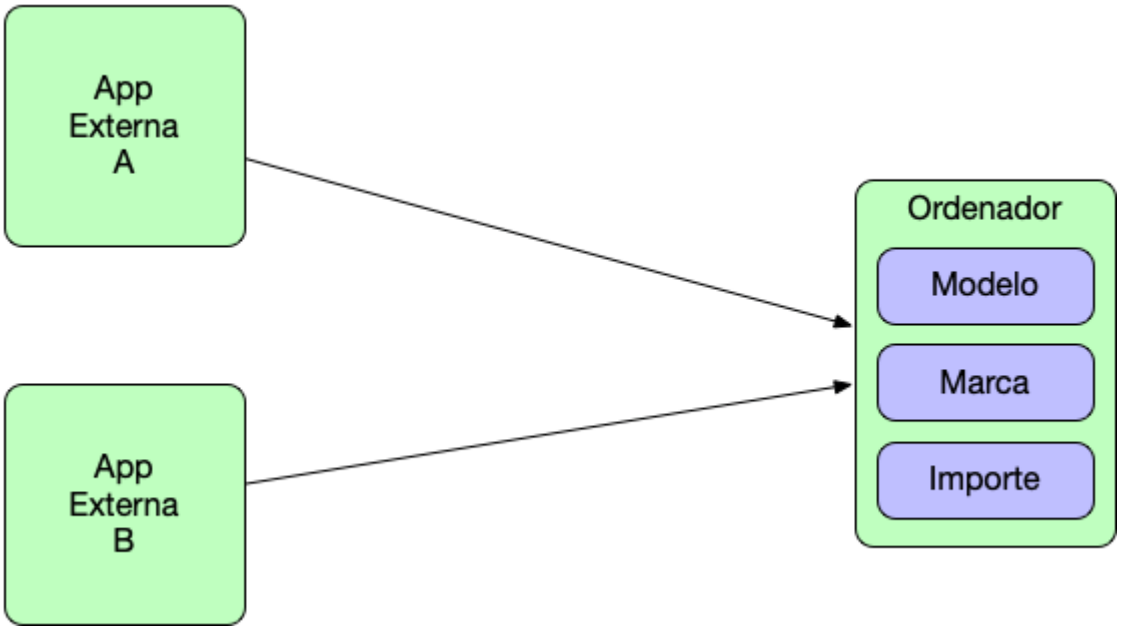
DTO o Data Transfer Oject es uno de los conceptos más clásicos de programación a nivel de patrones de diseño . Un DTO nos permite aislar las entidades de una aplicación del entorno externo que se conecta a ella. Por ejemplo imaginemos que tenemos una clase que se denomina Ordenador.

```
package com.arquitecturajava;

public class Ordenador {

	private String modelo;
	private String marca;
	private double importe;
	public String getModelo() {
		return modelo;
	}
	public void setModelo(String modelo) {
		this.modelo = modelo;
	}
	public String getMarca() {
		return marca;
	}
	public void setMarca(String marca) {
		this.marca = marca;
	}
	public double getImporte() {
		return importe;
	}
	public void setImporte(double importe) {
		this.importe = importe;
	}
	public Ordenador(String modelo, String marca, double importe)
```

```
{
                super();
                this.modelo = modelo;
                this.marca = marca;
                this.importe = importe;
        }
        public Ordenador() {
                super();
        }
}
```

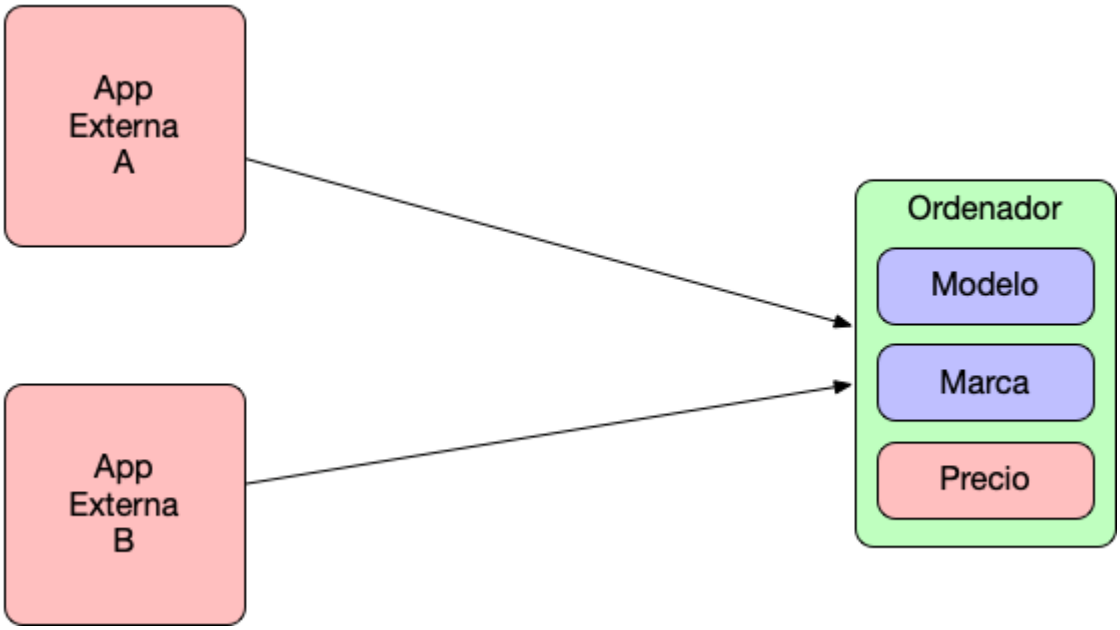
Entidades y Acoplamiento

El manejo y construcción de Data Transfer Objects es relacionado con el acoplamiento ya que por ejemplo tenemos la clase Ordenador como objeto de negocio y ligamos aplicaciones externas a este concepto tendremos un problema ya que si el concepto de DTO cambia , como por ejemplo puede ser una propiedad o algo similar todas las aplicaciones se verán afectadas de forma inmediata.



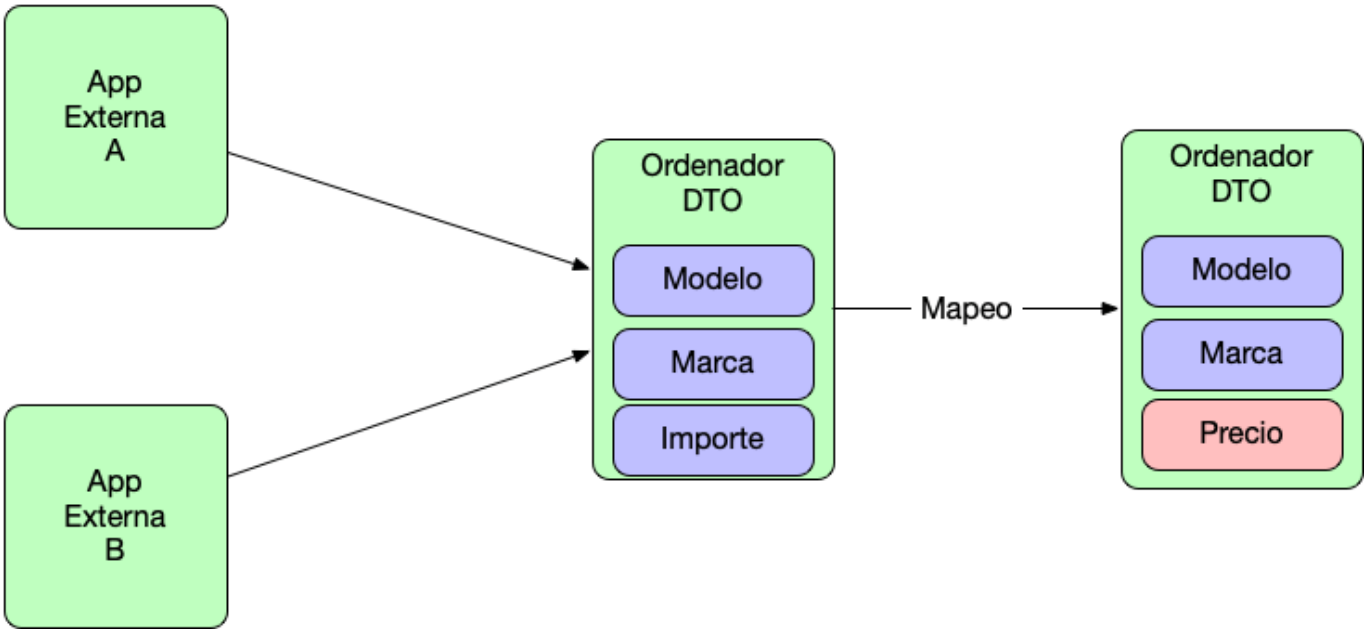
El problema es que si en algún momento alguna de las propiedades del Ordenador cambia

nos veremos obligados a actualizar todas las aplicaciones relacionadas con ella.



DTO al Rescate

Podemos generar una capa de DTO de indirección que haga que nuestras aplicaciones se conecten a un DTO y no directamente a nuestras clases de Entidad.



Veamos un código sencillo de Data Transfer Object en el cual vemos que comparte las mismas propiedades con la Entidad.

```
package com.arquitecturajava;

public class OrdenadorDTO {

	private String marca;
	private String modelo;
	private double importe;
	public String getMarca() {
		return marca;
	}
	public void setMarca(String marca) {
		this.marca = marca;
	}
	public String getModelo() {
		return modelo;
	}
	public void setModelo(String modelo) {
		this.modelo = modelo;
	}
	public double getImporte() {
		return importe;
	}
	public void setImporte(double importe) {
		this.importe = importe;
	}
	public OrdenadorDTO (Ordenador ordenador) {
		this.modelo=ordenador.getModelo();
		this.marca=ordenador.getMarca();
```

```
                this.importe=ordenador.getImporte();
        }
}
```

Si tuviéramos que realizar algún mapeo entre propiedades podríamos realizarlo en el propio constructor. Aqui se muestra un ejemplo entre precio e importe.

```
        public OrdenadorDTO (Ordenador ordenador) {
                this.modelo=ordenador.getModelo();
                this.marca=ordenador.getMarca();
                this.importe=ordenador.getPrecio();
        }
```

En algunas ocasiones para simplificar el manejo de Data Transfer Objects algunos de ellos pueden terminar siendo clases inmutables ya que su uso es muy concreto sirven para generar ese nivel de indirección.

```
package com.arquitecturajava;

public class OrdenadorDTO {

        private final String marca;
        private final String modelo;
        private final double importe;
        public String getMarca() {
                return marca;
        }
        public String getModelo() {
                return modelo;
        }
        public double getImporte() {
                return importe;
```

```
        }
        public OrdenadorDTO (Ordenador ordenador) {
                this.modelo=ordenador.getModelo();
                this.marca=ordenador.getMarca();
                this.importe=ordenador.getImporte();
        }
}
```

En este caso es suficiente con declarar las propiedades final y eliminar los métodos set. De esta forma el DTO queda fuertemente simplificado . Este modelo de Data Transfer Object es muy típico en aplicaciones CQRS

Otros artículos relacionados

- REST DTO y JSON Arquitecturas Web y objetos
- Entity to DTO y Java 8
- Java Encapsulamiento y reutilización
- JPA DTO (Data Transfer Object) y JPQL
- Wikipedia