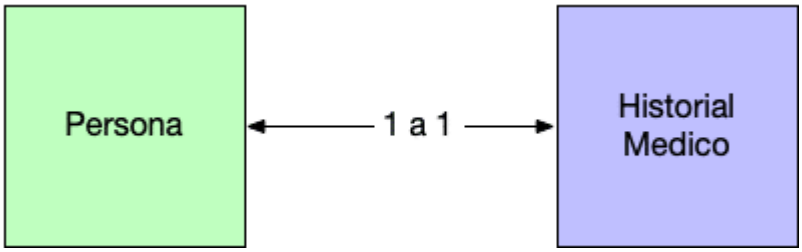
¿Java Relaciones Unidireccionales o Bidireccionales?. Esta es una buena pregunta que me realizan muchas personas. Cuando nosotros generamos relaciones entre clases estas pueden ser bidireccionales o unidireccionales y en muchas ocasiones se tienen dudas de como enfocar. Vamos a hablar un poco a detalle de ello.Supongamos que tenemos dos Entidades Persona e Historial médico. Se trata de una relación de uno a uno . Es de las relaciones más sencillas.

- [ihc-hide-content ihc_mb_type=”show” ihc_mb_who=”4″ ihc_mb_template=”1″]



¿Java Relaciones Bidireccionales?

Normalmente cuando relacionamos clases lo más lógico es que la relación se construya en ambas direcciones . Es decir una Persona contiene una referencia a su historial médico y viceversa un Historial Médico contiene una relación con la Persona a la que pertenece. El código sería algo similar a esto:

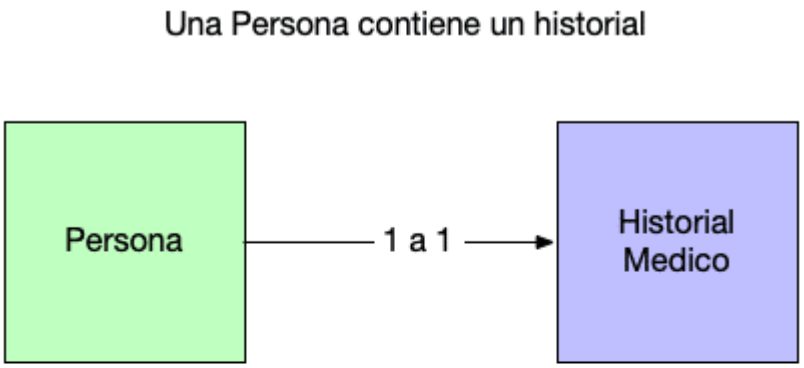
```
package com.arquitecturajava;

public class Persona {
```

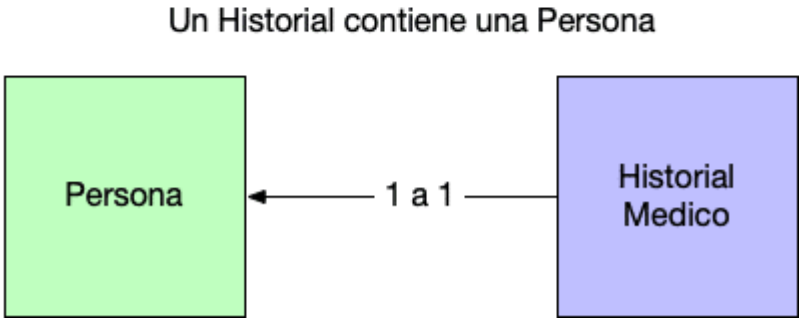
```
        private String nombre;
        private int edad;
        private HistorialMedico historialMedico;
        public Persona(String nombre, int edad) {
                super();
                this.nombre = nombre;
                this.edad = edad;
        }
        public String getNombre() {
                return nombre;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public int getEdad() {
                return edad;
        }
        public void setEdad(int edad) {
                this.edad = edad;
        }
        public HistorialMedico getHistorialMedico() {
                return historialMedico;
        }
        public void setHistorialMedico(HistorialMedico
historialMedico) {
                this.historialMedico = historialMedico;
        }
}


package com.arquitecturajava;
```

```
import java.util.ArrayList;
import java.util.List;

public class HistorialMedico {

	private int numero;
	private List<String> enfermedades= new ArrayList<String>();
	private Persona persona;
	public int getNumero() {
		return numero;
	}
	public void setNumero(int numero) {
		this.numero = numero;
	}
	public List<String> getEnfermedades() {
		return enfermedades;
	}
	public void setEnfermedades(List<String> enfermedades) {
		this.enfermedades = enfermedades;
	}
	public Persona getPersona() {
		return persona;
	}
	public void setPersona(Persona persona) {
		this.persona = persona;
	}
}
```

Otras opciones

Existen otras dos opciones podemos hacer que la Persona quede relacionada con el Historial Medico o que el Historial Médico quede relacionado con la Persona. Ambas son relaciones unidireccionales .

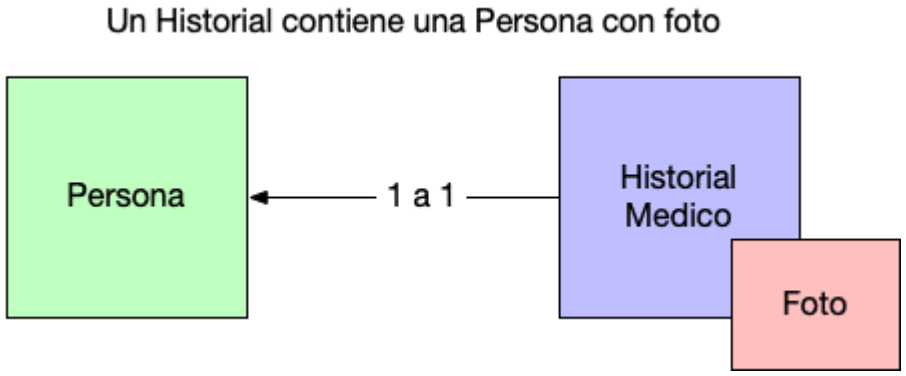




¿Tiene sentido crearlas , es mejor quedarse con la relación bidireccional? . Vamos a hablar un poco más de ello. En una aplicación sencilla lo más habitual es disponer de una relación bidireccional y ahí se termina el tema no hay más que hacer . Es lo más natural .

Java Relaciones y Problemas

Cuando estamos ante aplicaciones complejas en las cuales por ejemplo el Historial Medico incluye una foto o un PDF que es algo “pesado” a nivel de rendimiento.

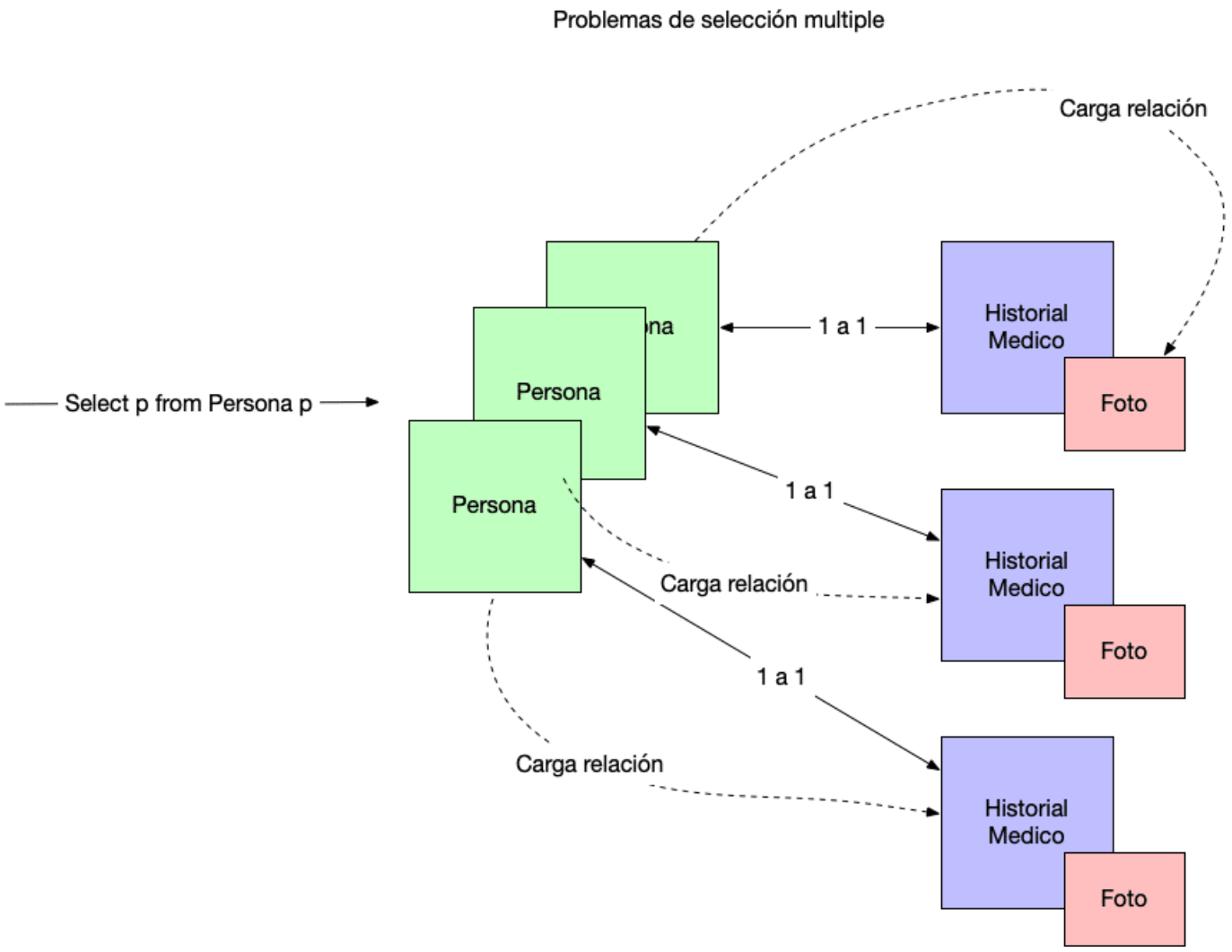


Veamos un posible código

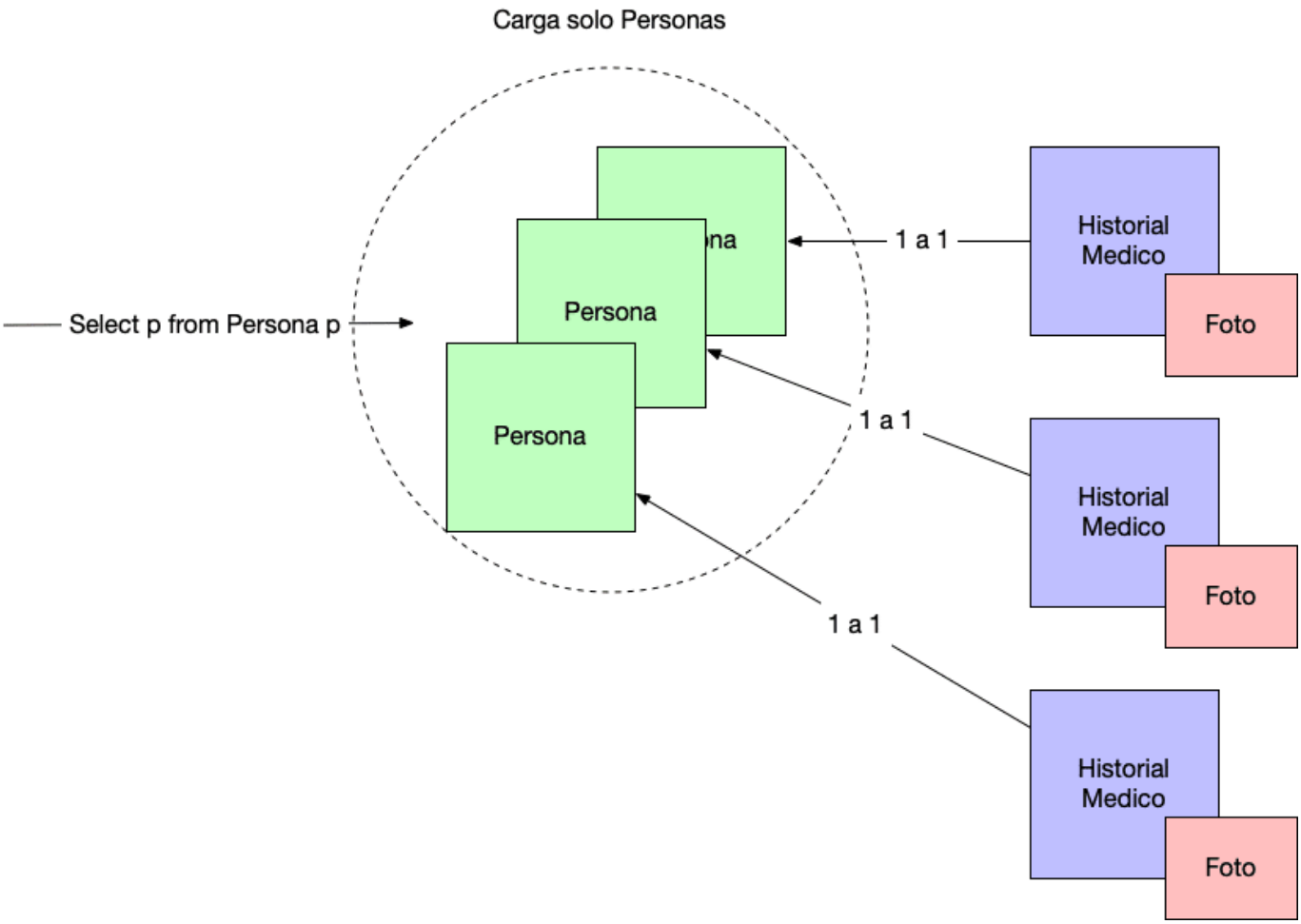
```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;

public class HistorialMedico {

	private int numero;
	private List<String> enfermedades= new ArrayList<String>();
	private Persona persona;
	private byte[] foto= new byte[20000];
	public byte[] getFoto() {
		return foto;
```

```
	}
	public void setFoto(byte[] foto) {
		this.foto = foto;
	}
	public int getNumero() {
		return numero;
	}
	public void setNumero(int numero) {
		this.numero = numero;
	}
	public List<String> getEnfermedades() {
		return enfermedades;
	}
	public void setEnfermedades(List<String> enfermedades) {
		this.enfermedades = enfermedades;
	}
	public Persona getPersona() {
		return persona;
	}
	public void setPersona(Persona persona) {
		this.persona = persona;
	}
}
```

En estas situaciones puede ser interesante componer una relación unidireccional de tal forma que la Personas no tenga acceso directo al Historial Medico ya que a nivel de aplicación se selecciona un listado de Personas es muy probable que un framework de Persistencia como Hibernate seleccione de forma automática también los Historiales algo que cargaría mucho la consulta.



En cambio sino generamos la relación de forma bidireccional , el framework de Persistencia nunca podrá cargar los Historiales de las Personas de forma directa ya que no habrá relación . Solo se podrá construir algo de ese estilo si primero seleccionamos el Historial en ese caso el programador tiene claro que ha de gestionar la foto.



Por lo tanto en este caso sería mejor que la clase Persona no tuviera una relación con la clase HistorialMedico . La clase Persona quedaría así :

```
package com.arquitecturajava;

public class Persona {

	private String nombre;
	private int edad;
	public Persona(String nombre, int edad) {
		super();
```

```
                this.nombre = nombre;
                this.edad = edad;
        }
        public String getNombre() {
                return nombre;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public int getEdad() {
                return edad;
        }
        public void setEdad(int edad) {
                this.edad = edad;
        }
}
```

Se accedería a la relación siempre a través de la clase HistorialMedico.

Conclusiones

No siempre las relaciones bidireccionales son las mejores a nivel de código real y puede haber casuísticas que por rendimiento sea mejor construir la relación en un solo sentido

Otros artículos relacionados

- JPA @OneToOne y relaciones 1 a 1
- n+1 Queries y sus problemas

- JPA Orphan Removal y como usarlo

[/ihc-hide-content]