Ejercicio 15 – Uso de DTO

En ejercicios anteriores hemos definido las clases entidad de nuestro modelo con todos sus atributos y las relaciones entre ellas. Además, hemos utilizado estas clases para cualquier operación que requiera la presencia de un objeto determinado.

En aplicaciones que requieren de intercambiar mensajes con terceros, una de las problemáticas más comunes es utilizar las clases de entidades en la capa de aplicación, lo que ocasiona que retornemos más datos de los necesarios o expongamos detalles poco convenientes de la base de datos.

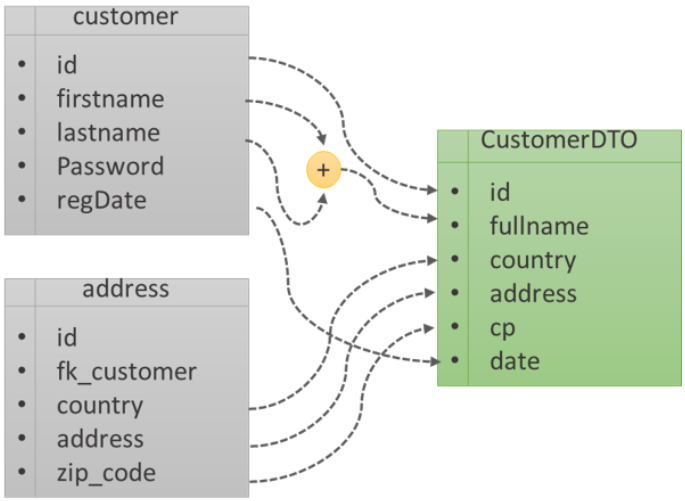
¿Qué es un DTO?

Un ***DTO*** (Data Transfer Object) es un patrón de arquitectura propuesto por Martin Fowler en su libro *Patterns of Enterprise Application Architecture*, cuya finalidad es crear objetos que ofrezcan una versión "resumida" de las entidades según las necesidades de las entradas y salidas de las operaciones.

Entre los beneficios de utilizar el patrón DTO encontramos:

- Reducir la cantidad de información que transferimos entre las capas.
- Adaptar los datos a una salida específica.
- Desacoplar los modelos de dominio de la capa de presentación.

Ejemplo de DTO



En la imagen vemos como dos entidades que mapean dos tablas de la base de datos, la tabla Customer y la tabla Address. La tabla Address tiene una columna que hace referencia al Customer, formando una relación One To One.

Podemos ver que hemos creado un nuevo objeto llamado CustomerDTO, en el cual podemos agregar libremente cuantos atributo requiramos, incluso, podemos asignarle valores de diferentes fuentes de datos.

Debido a que el DTO es una clase creada únicamente para una determinada respuesta, es posible modificarla sin mucho problema, pues no tiene un impacto en la capa de servicios o de datos, ya que en estas capas se trabaja con las Entidades.

Características de un DTO

Si bien un DTO es simplemente un objeto plano, sí que tiene que cumplir algunas reglas para poder considerar que hemos creado un DTO correctamente implementado:

- ***Solo lectura***: Dado que el objetivo de un DTO es utilizarlo como un objeto de transferencia entre el cliente y el servidor, solo deberemos de tener *getters* y *setters* de los atributos del DTO.

- ***Serializable***: Si los objetos tendrán que viajar por la red, el propio objeto y sus atributos deberán de poder ser serializables.

En este contexto, es interesante el uso de los Java ***Records***, un tipo especial de clase que actúa como un objeto de datos y que es inmutable.