Justificaciones de diseño - Modelo de datos

En esta entrega, optamos por embeber algunas entidades dentro de otras para evitar joins innecesarios, especialmente en casos como las direcciones o áreas dentro de las entidades de colaborador, técnico y heladera. Al hacer esto, reducimos la complejidad de las consultas y mejoramos el rendimiento general del sistema.

Otra decisión importante fue utilizar la estrategia Single Table para mapear los diferentes tipos de registro de uso de tarjeta. Esto fue posible porque las clases solo tenían una mínima diferencia en un atributo que podría quedar en null. Para manejar esta distinción, agregamos un atributo discriminador llamado "tipo", que puede tomar los valores de distribución, consumo o colocación. Esta estrategia nos permite gestionar diferentes tipos de registros dentro de una única tabla, simplificando la estructura de datos.

En cuanto a las relaciones muchos a muchos, hemos recurrido varias veces a tablas intermedias para gestionar estos vínculos, como en los casos de heladera-registroUsoDeTarjeta, oferta-producto y colaborador-suscripción. Estas tablas no necesitan ser representadas a nivel de objetos en el código, ya que no contienen atributos adicionales, lo que nos permitió implementar las relaciones mediante manyToMany.

Además, aprovechamos algunos objetos ya existentes en el modelo de datos para que funcionen como tablas intermedias, como es el caso de canje, que actúa como intermediario entre la billetera de puntos y las ofertas. Esto nos permitió evitar la creación de nuevas entidades y reutilizar las que ya teníamos, optimizando así el modelo.

En el diseño del modelo de datos, decidimos embeber los *value objects* en lugar de crear tablas separadas por las siguientes razones:

1. Los *value objects* no tienen un peso significativo en términos de volumen de datos ni requieren gestión independiente. Embebidos, simplifican el modelo sin agregar complejidad innecesaria.
2. Evitamos *joins* complejos, mejorando el rendimiento de las consultas y operaciones en la base de datos. Esto facilita el acceso rápido a los datos relacionados.

A los principales que embebimos fue a la clase direcciones, modelo de heladera, entre otros, en los cuales no había un comportamiento y no eran lo suficientemente grandes como para justificar que tengan una tabla aparte.