**Service Pattern**

El patrón Service es muy utilizado en la programación y su nombre es muy conocido. Pero a pesar de ser un nombre único, Service puede ser interpretado de varias maneras: puede ser un caso de uso (Application Service); un Domain Service, que tiene reglas de su dominio; un Infrastructure Service, que utiliza algún paquete externo para realizar tareas; etc.

A pesar de que la interpretación puede ocurrir de varias formas, la idea detrás del patrón es separar las reglas de negocio, las reglas de la aplicación y las reglas de presentación para que puedan ser fácilmente probadas y reutilizadas en otras partes del sistema.

Existen dos formas más utilizadas para crear Services. Puede crear Services más genéricos, responsables de todas las asignaciones de un Controller; o ser aún más específico, aplicando así la S del SOLID: Single Responsibility Principle (Principio de Responsabilidad Única). Este principio nos dice que una clase/función/archivo debe tener sólo una única responsabilidad.

Piense en un sistema de ventas, en el que probablemente tendríamos algunas funciones como: Registrar usuario, Iniciar sesión, Buscar productos, Buscar producto por nombre, etc. Entonces, podríamos crear los siguientes Services: RegistroDeUsuarioService, IniciarSesionService, BusquedaDeProductosService, etc.

Pero es importante estar atentos, ya que muchas veces no es necesario crear un Service y, por lo tanto, agregar otra capa y complejidad innecesarias a nuestra aplicación. Una regla que podemos utilizar es la siguiente: si no hay reglas de negocio, simplemente podemos realizar la comunicación directa entre los controllers y los repositories de la aplicación.