Objetivo: recapitulación Back-OOP sesión I
Autoras: Miriam García y Marcos Chotsouurian
Fecha: 11/01/20023

Recapitulación Back-OOP, sesión I

A fin de pensar la arquitectura del back end estuvimos un rato imaginando cómo es una aplicación bancaria tipo. ¿Qué funciones debe tener? Consultar saldo, realizar transferencias, comprobar usuario y contraseña, enviar emails, enviar notificaciones y muchas otras cosas más.

Para funcionar bien y organizadamente, la aplicación está dividida en capas. Cada capa tiene una responsabilidad distinta. Es muy importante en este ejercicio imaginativo tener presente que estamos pensando siempre en la aplicación bancaria, en la aplicación web como tal. No estamos pensando en el banco como empresa ni como edificio.

La aplicación bancaria cuenta con las siguientes capas:

***FRONT**: es la cara visible de la aplicación y aquello con lo que interactúa el usuario.*

***BACK**:*

***Controlador**: funciona como "mesa de entrada" de la aplicación. Tiene definidas las solicitudes o "trámites" que puede resolver la aplicación.*

***Servicios**: funcionan como "operadores o resolvedores de trámites" de la aplicación. Tienen definidos los procesos puntuales a través de los cuales esos trámites se llevan a cabo. Pueden ser internos a la aplicación o externos.*

***Bases de datos**: contienen el universo de datos que necesita la aplicación para funcionar correctamente. Se organizan en tablas. Las entidades con las cuales opera la aplicación van a verse reflejadas en tablas. Cada columna de cada tabla refleja una propiedad o atributo.*

Clases, clases abstractas e interfaces:

***CLASE**: Una clase es un tipo de objeto. Es importante tomarlo de esta manera tan general porque puede cumplir funciones distintas a lo largo del código.*

¿Cuándo utilizar una clase o qué función puede cumplir en la arquitectura? Por ejemplos:

-Las entidades de la base de datos van a estar representadas por clases.

-Los objetos que implementen los Servicios también.

-Los objetos que sean los Controladores también.

-Si necesitamos definir algún tipo de dato extra en nuestra aplicación, sea o no una entidad en la base de datos, también.

***Herencia**: Las clases pueden heredarse las unas a las otras. De esa manera se organizan en "árboles de familias" con padres e hijos. Un padre es más general que el hijo. Todos los atributos y métodos del padres son también métodos y atributos del hijo.*

***CLASE ABSTRACTA**: Es una clase que puede tener elementos abstractos, es decir, elementos no definidos aún y que serán precisados en la definición de sus hijos.*

***INTERFAZ**: Define un **rol**. Las clases que **implementen** una interfaz son aptas de cumplir ese rol. En principio, de lo que venimos viendo y de lo que necesitamos trabajar por el momento, solamente usaremos interfaces para los Servicios. Esto es porque la aplicación no debe depender de tal o cual Servicio concreto, sino de un rol en abstracto. Por ejemplo, las solicitudes o "trámites" relacionadas con emails no puede depender de **un** proveedor concreto de servicio de emails, sino que debe poder funcionar con **cualquier** proveedor de servicio de emails.*