PRUEBA TÉCNICA PARA EL BACKEND DE GRABILITY

Ernesto Soto Meléndez
CC 1.015.429.099

Bogotá D.C.
Abril 24 2017

Contents

Code Refactoring

Código refactorizado

El código refactorizado se encuentra en la carpeta Code_Refactoring de este mismo repositorio.

Malas prácticas

En el código entregado en la prueba, se identificaron varias prácticas que llevan al desorden y a que el código no se entienda fácilmente. Estas son:

- No se mantiene la identación, ya que en algunas partes se dejan varios espacios y en otras sólo un espacio de tabulación.
- No se mantiene la notación. Se combinan comillas dobles y comillas sencillas en diferentes partes del mismo método.
- Se deja código comentado. Esto es código muerto y muestra desorden. Para encontrar código que anteriormente se utilizaba es mejor un sistema de integración continua, que cuenta con versionamiento, como Git, SVN, etc., pues guardan historial.
- Accede varias veces al parámetro *driver*_id a través de Input, en lugar de utilizar una variable para almacenar el valor. Es desordenado, y si se utiliza varias veces es mejor que esté directamente en memoria con una variable.
- Accesos innecesarios a la persistencia. El objeto *$servicio* ya se tiene antes de declarar la notificación al usuario y en este punto (después de inicializar *$pushMessage*), se hace nuevamente un find.

Cómo la refactorización supera las malas prácticas

La refactorización corrige las malas prácticas mencionadas de la siguiente manera:

- Se unifica la identación para dar más orden al código. Es más sencillo identificar lo que está anidado.
- Se unifica la notación. Todos los arrays se escriben de la misma forma y todos los nombres en *Strings* se dejan con comillas sencillas, para evitar confusiones.
- Se elimina el código comentado. Esto deja sólo comentarios descriptivos y código funcional.

- Se utiliza una variable para guardar el parámetro *driver_id* y utilizarlo las veces que sea necesario.
- Se elimina el acceso innecesario a persistencia. Se deja sólo el *Service::find* del principio, que inicializa la variable *$servicio*.

De esta forma el código es más ordenado, más limpio e incluso más eficiente evitando un acceso a persistencia.