Code Challenge

1. El programa se diseño siguiendo paso a paso los instructivos de la página, fue diseñado en Java 7, actualmente el programa permite crear una matriz 3d con las restricciones asignadas al ejercicio, es decir, el número de caso será menor a 50, la matriz no puede superar el tamaño, es decir no puede ser mayor a una matriz de 100\*100\*100, de igual manera el número de operaciones está restringido de manera que no puede exceder las 1000 variadas entre los Update y los Query.

El programa parte de un menú base o una interfaz de usuario donde se le solicitan los datos paso a paso a petición del ejercicio, una vez hecho el ingreso de los datos necesarios para la elaboración de la matriz, se inicializa la matriz en ceros (0), una vez llenada la matriz solicita que se seleccione una de las opciones del menú para poder escoger una de las operaciones existentes, de esta manera se puede actualizar la matriz o hacer la consulta que genera la suma de las dos coordenadas a ingresar.

Una vez seleccionada la operación procedemos al ingreso de los datos en las coordenadas para poder asignarle nuevos valores, el programa imprime constantemente la matriz almacenando los datos ingresados previamente.

El programa fue diseñado con un tercer botón que permite finalizar el proceso de inserción de datos, o se cierra y muestra una alerta cuando los valores ingresados son superiores a los restringidos.

1. El programa es un aplicativo en Java por lo tanto las capas de persistencia no están presentes en el código, la capa de aplicación o vista está dada por la consola del IDE que se usa, en este caso NetBeans IDE 8.1 en conjunto con el menú interfaz diseñado para el programa.
2. Se manejaron dos clases para la elaboración del programa, cuenta con un Main que permite que contiene la interfaz de usuario, y la clase Matriz 3D que contiene, la captura de los datos, y el procesamiento de los mismos, consta de la elaboración de la matriz, inicialización de Elementos, asignación de elementos, Asignación de los elementos a las coordenadas, Query Elementos, que permite realizar la suma de los elementos definidos por dos coordenadas más no los elementos que estén en el rango de las coordenadas, imprimir elementos, que contribuye a la parte visual del programa.

**Code Refactoring.**

Public function post confirm () {

/\*Declaración de variables globales\*/

$id = Input ::('get service\_id');

&servicio = Service::find($id);

dd($servicio);

/\*Solicitud de servicio\*/

if ($servicio !=NULL) {

if ($servicio ->status\_id == '6') {

return response::json(array('error' => '2');

}

/\*verificación del usuario y del conductor\*/

if ($servicio>driver\_id == NULL && $servicio->status\_id =='1') {

$servicio = Service::update($id, array('driver\_id' => Input::get('dirver\_id'), status\_id => '2', 'pwd => md5(Input::get('pwd)));

Driver::update(Input::get('driver\_id'), array("avalible" => '0'));

$driverTmp = Driver::find(Input::get('driver\_id'));

Service::update($id, array('car\_id' => $driverTmp->car\_id, 'pwd'=> md5 (Input::get('pwd'))));

/\*Notificar Usuario!!\*/

$pushMessage= 'tu servicio ha sido confirmado!';

$servicio = Service::find($id);

$psuh = Push::make();

if ($servicio->user->type == '1') {

&pushAns = $psuh->ios($servicio->user->uuid, $pushMessage);

} else {

$pushAns = $push->andriod($servicio->user->uuid, $pushMessage);

}

}

$servicio = Service::find($id);

$push = Push::make();

if ($servicio->user->uuid=='') {

return Response::json(array('error'=> '0'));

}

/\*Usuarios Iphone Mensaje\*/

if ($servicio->user->type == '1'){

$result = $push-ios($servicio->user-uuid, $pushMessage, 1, 'honk.hav','Open', array('serviceId' =>$servicio->id));

} else {

/\* Usuarios Android Mensaje\*/

result = $push-android($servicio->user-uuid, $pushMessage, 1, 'default','Open', array('serviceId' =>$servicio->id));

}

return Response::json(array('error' => '0'));

} else {

return Response::json(array('error' => '1'));

}

} else {

return Response::json(array('error' =>'3'));

}

}

1. El Código tenía problemas de orden, le faltaban signos de puntuación y parte de los comentarios que existían en el código original son parte del programa, a su vez los comentarios no estaban colocados en el lenguaje de programación que se está manejando, la manera en que estaban ingresados los comentarios corresponden a Java, mientras que los en este caso los comentarios deben abrirse con /\* y cerrarse con \*/.
2. La recodificación no fue compleja, se redujeron la cantidad de líneas manteniendo el orden necesario para leer cada bloque del programa, el uso de los caracteres adecuados permite la organización de los bloques del programa con el fin de identificar cual es la funcionalidad de cada uno.

**Preguntas**

1. El principio de responsabilidad única consiste en la codificación de las clases basándonos en la única funcionalidad para dicha clase con el fin de evitar múltiples motivos por los cuales pueda ser alterada, entendiendo por esto, tiene como propósito la estructuración de software con el fin de crear módulos con tareas y elementos funcionales propios y no con funciones separadas, que en dado caso que requieran ser editados o corregidos no se alteren más funcionalidades del programa, por otro lado permite la reutilización de código y por lo tanto la optimización del mismo código.
2. Un código limpio consiste en la generación y aplicación de los principios de programación; dentro de las prácticas como programador está la reducción de líneas de código de manera que sea amigable al lector, la estructuración e identificación de los módulos del programa con comentarios que identifiquen las funciones de cada bloque o módulo, elaboración de test para cada uno de los bloques y las funciones, el código debe ser enfocado a proveer soluciones generales y no condicionar el código para realizar bloques para cada uno de los casos que generen problemas.