# Respuestas Bak-end Test de Rappi

## Coding Challenge

El coding challenge se resolvió con dos aproximaciones, ambas web en una misma aplicación bajo el framework Laravel.

Se le permite al usuario seleccionar que aproximación quiere tomar apenas entra a la aplicación, una aproximación cumple al pie de la letra las especificaciones entregadas en el coding challenge, esta opción se llamo “**standard way**”, la otra es una aproximación más interactiva en la que el usuario debe primero entrar el tamaño de la matriz, una vez creada puede hacer varias actualizaciones y querys a la matriz, esta aproximación se llamó “**interactive way**”

Capas de la aproximación “standard way”

1. Persistencia: no hay persistencia de datos, cada llamada a la aplicación realiza la operación de creación, actualización y querys a la matriz y devuelve los resultados de las querys
2. Vista: se compone de una única vista “standar.blade.php” con un único input en el que se ingresa un string con las reglas descritas en [hacker Rank](https://www.hackerrank.com/challenges/cube-summation) y si hay algún resultado lo muestra
3. Aplicación: El formulario es recibido por la clase “StandarCubeController”, esta tiene un método llamado “test” que recibe la petición y delega la realización de las operaciones al método “perform\_operations” el cual crea una instancia de la clase “Cube” y realiza operaciones sobre esta y devuelve los resultados para ser presentados.

Capas de la aproximación “interactive way”

1. Persistencia: Se utiliza la clase “Session” nativa de laravel para guardar los datos de la matriz entre diferentes llamadas al servidor
2. Vista: se compone de única vista “cube.blade.php” que muestra solamente la opción de crear una matriz inicialmente, y cuando esta creada muestra los formularios de actualización y consulta a la matriz (cada acción tiene su propio formulario aparte y el formulario para crear una nueva matriz es el que se muestra en todo momento)
3. Aplicación: todas las rutas se direccionan a la clase “CubeController”, esta se encarga de validar los inputs, crea y opera una instancia de la clase “Cube” que es guardada en la Session de laravel para la persistencia de datos entre diferentes llamadas a la aplicación.

## Code Refactoring

Las malas prácticas encontradas fueron:

1. No hacer validación de los input
2. Nombres no descriptivos de variables , ej: llamar una variable con el nombre id y no id\_service
3. Dejar pedazos de código usado para pruebas comentado
4. variables sin usar, ej: guardar el valor de return de $push a android o a ios cuando no se usa
5. Hacer updates del mismo modelo varias veces por separado pudiendo hacerlas una sola vez
6. Hacer uso varias veces del método find() del modelo service sin necesidad de hacerlo más de una vez
7. Nombrar variables en español cuando los modelos y demás variables y atributos externos al método esta en ingles
8. Implementar validaciones en una serie de if anidados (los if anidados deberían evitarse al máximo en especial si el código interno va a ser largo)

El refactor hecho evita todas las anteriores malas prácticas y delega funciones complejas como por ejemplo hacer la notificación Push, o hacer el update de los driver a funciones privadas facilitando la lectura del código, su posible testeo unitario y su mantenimiento en consecuencia, adicional se comenta el código de forma descriptiva para ayudar a etender mas rápido el código