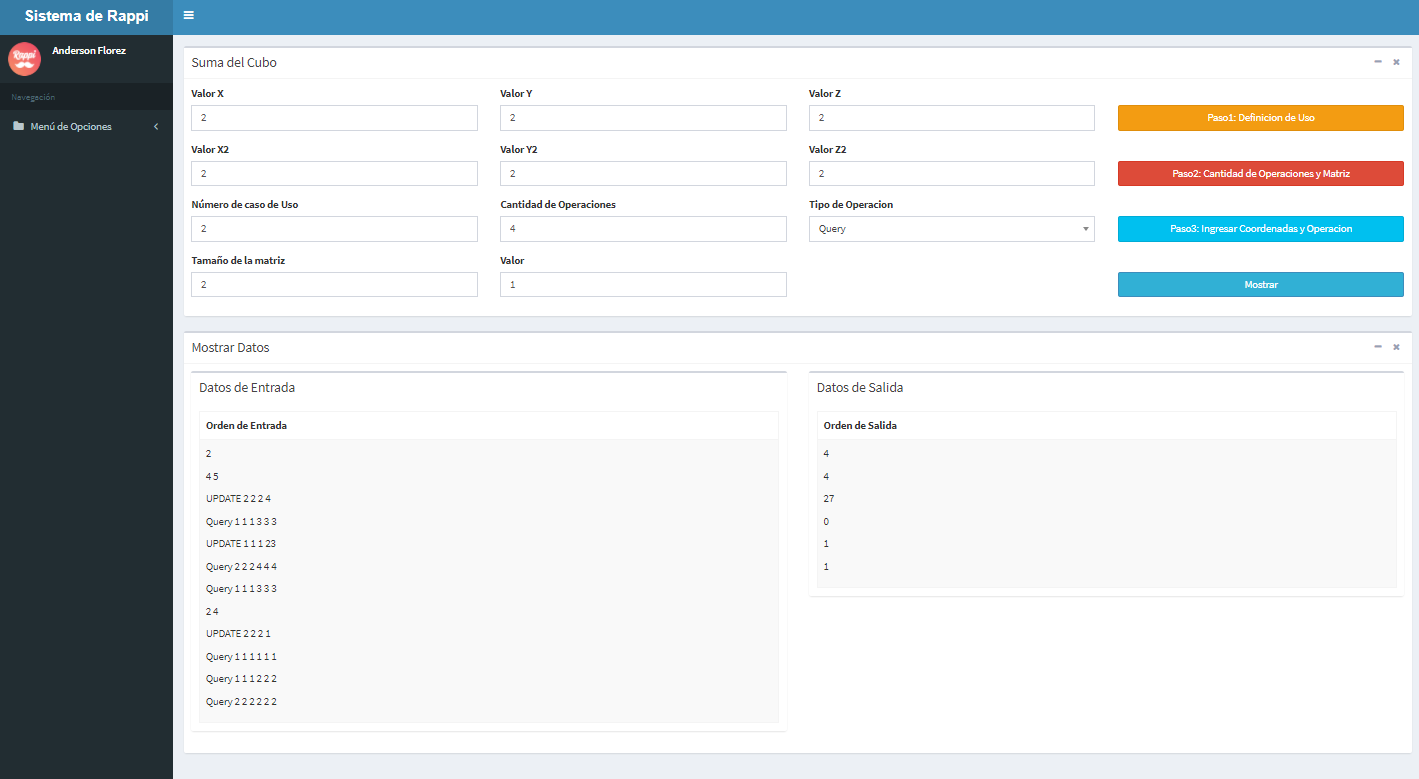
Suma del Cubo



INDICE GENERAL

[**Parte I: CODING CHALLENGE** 3](#_Toc503279439)

[**Uso del Sistema** 3](#_Toc503279440)

[Parte 1: Definir uso 3](#_Toc503279441)

[Parte 2 Definir cantidad de operaciones y Matriz 3](#_Toc503279442)

[Parte 3 Definir Coordenadas para el update o query. 4](#_Toc503279443)

[**Validaciones** 5](#_Toc503279444)

[**Capas de la aplicación** 6](#_Toc503279445)

[Vista 6](#_Toc503279446)

[Controlador 7](#_Toc503279447)

[**Pruebas Unitarias.** 7](#_Toc503279448)

[**Parte II: CODE REFACTORING** 9](#_Toc503279449)

[Las malas prácticas de programación que en su criterio son evidenciadas en el código. 9](#_Toc503279450)

[Cómo su refactorización supera las malas prácticas de programación.  9](#_Toc503279451)

[**Parte III: PREGUNTAS** 12](#_Toc503279452)

[¿En qué consiste el principio de responsabilidad única? ¿Cuál es su propósito? 12](#_Toc503279453)

[¿Qué características tiene según su opinión “buen” código o código limpio?   12](#_Toc503279454)

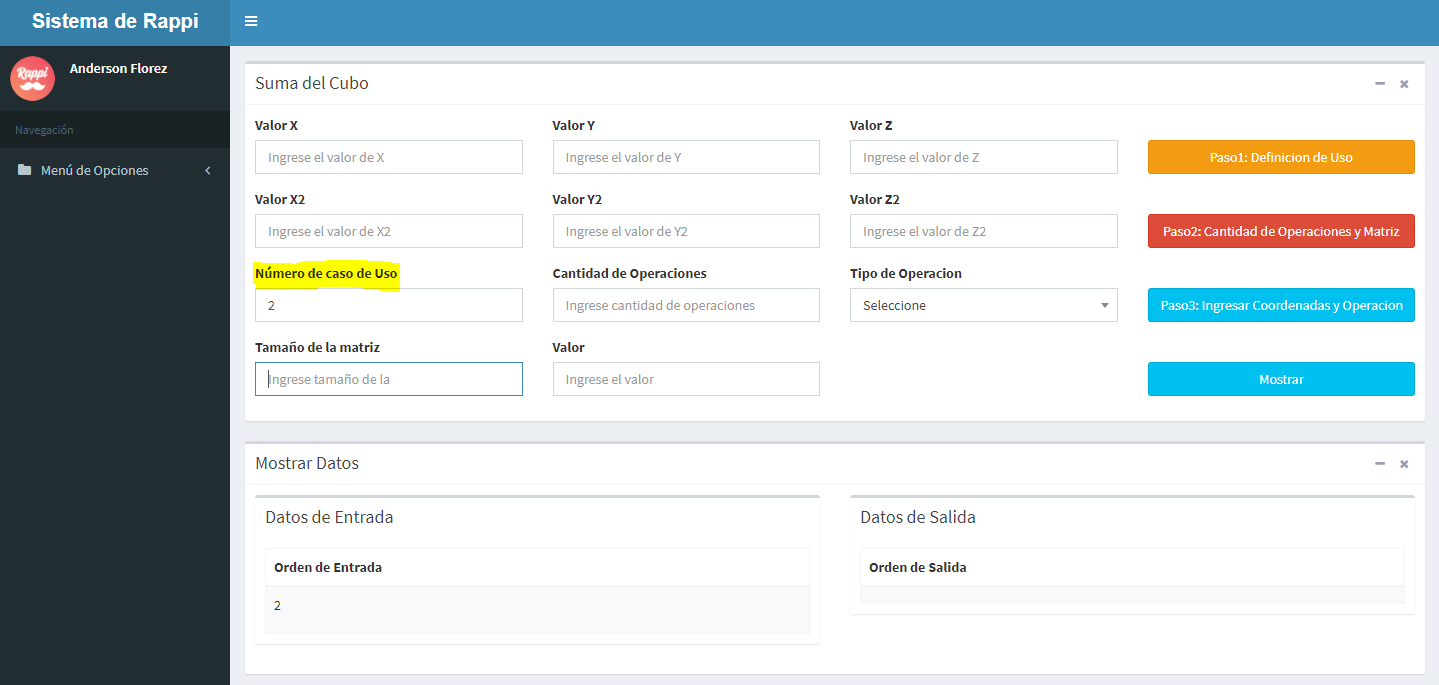
[¿Qué es un microservicio?, ventajas y desventajas de los micorservicios 12](#_Toc503279455)

# **Parte I: CODING CHALLENGE**

# **Uso del Sistema**

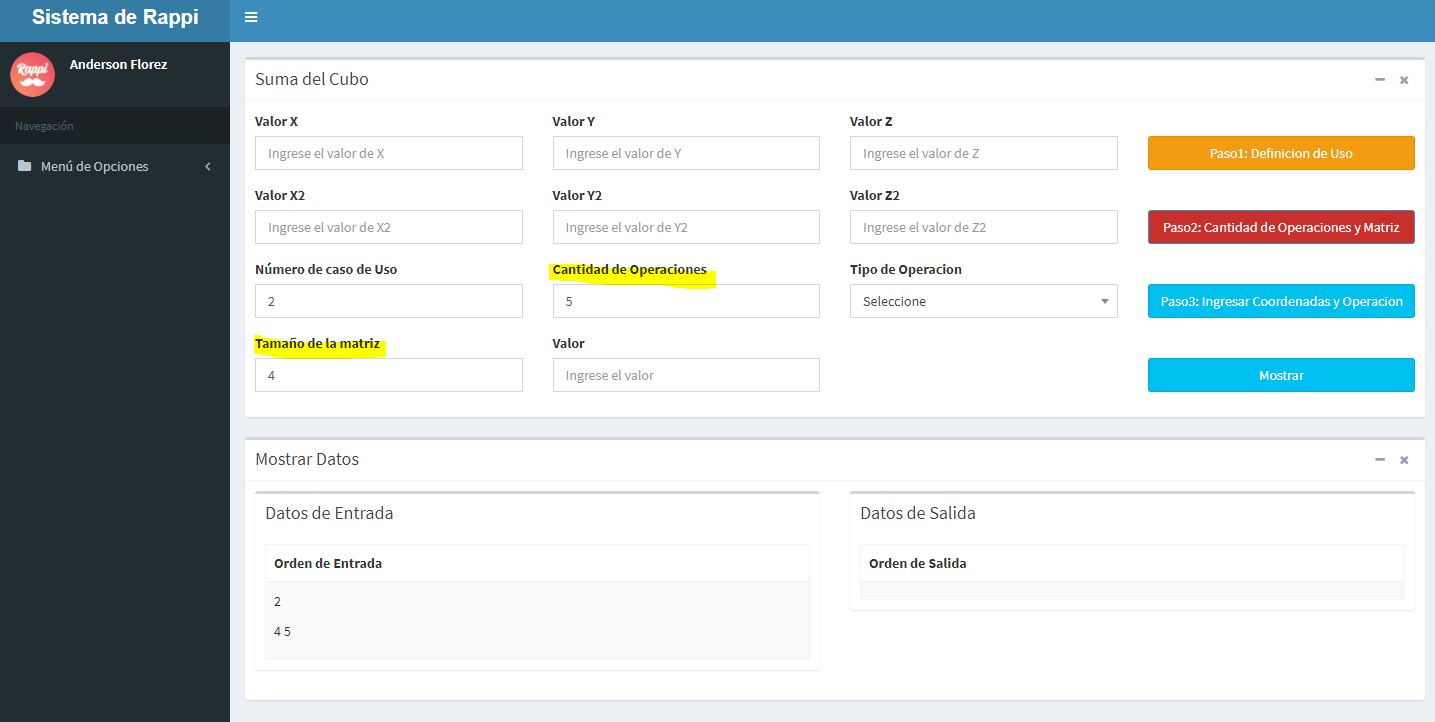
## Parte 1: Definir uso

Defina cuantas veces utilizara el sistema y presione sobre el botón “Definición de uso” para ingresar los valores.



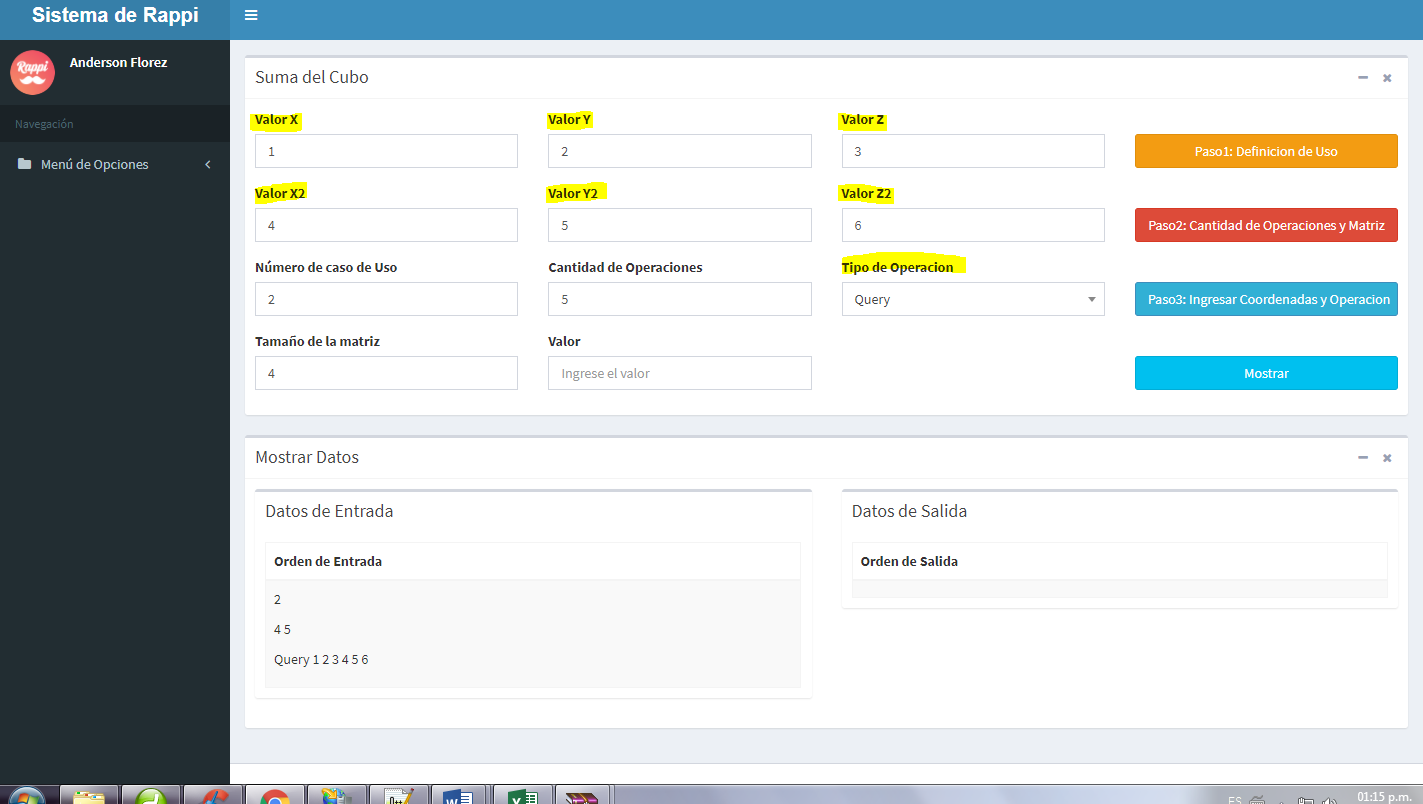
## Parte 2 Definir cantidad de operaciones y Matriz

Defina cuantas operaciones va a realizar en cada uso y el tamaño de la matriz y presione sobre el botón “Cantidad de operaciones y matriz” para ingresar los valores.

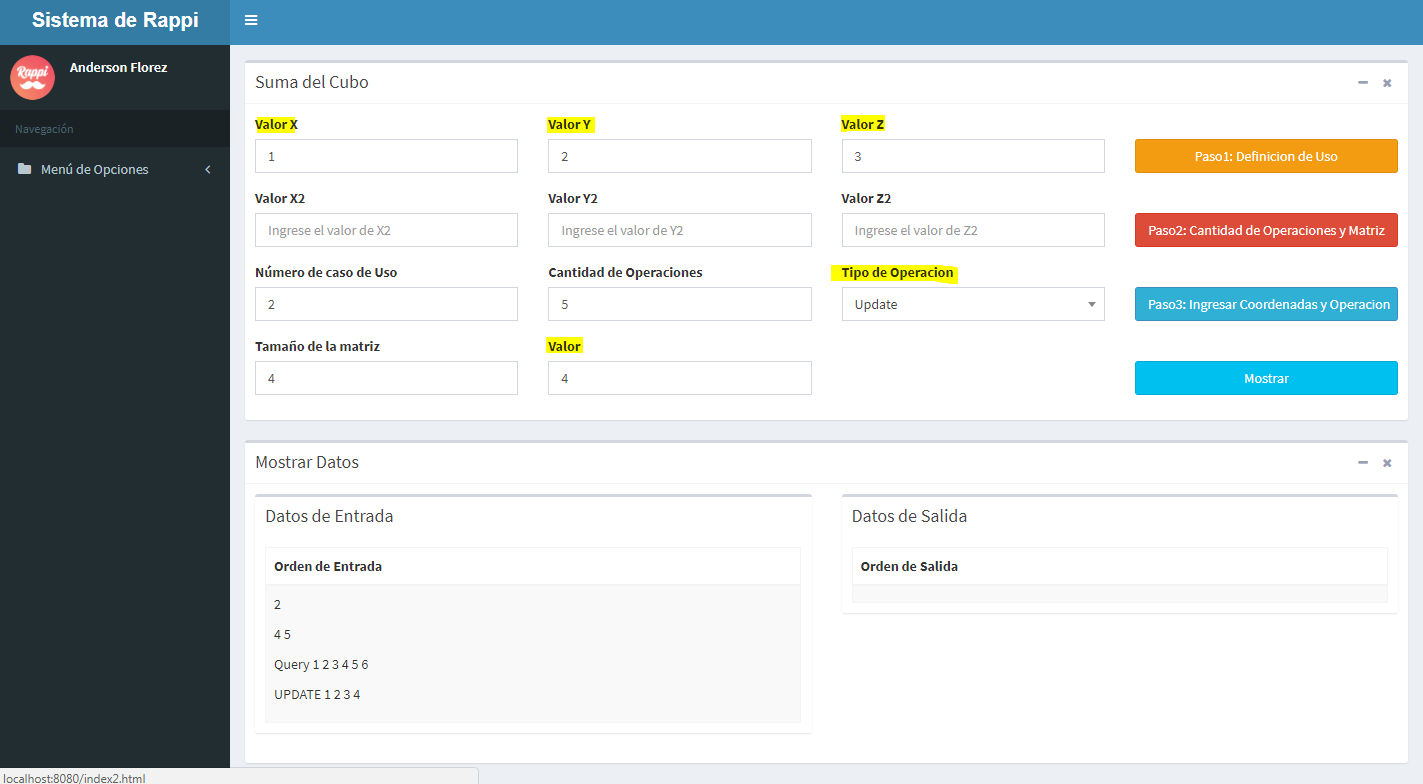


## Parte 3 Definir Coordenadas para el update o query.

* Ingrese las coordenadas (x,y,z) y (x1,y1,z1) y el tipo de operación query y presione sobre el botón “ingresar coordenadas y operación”.

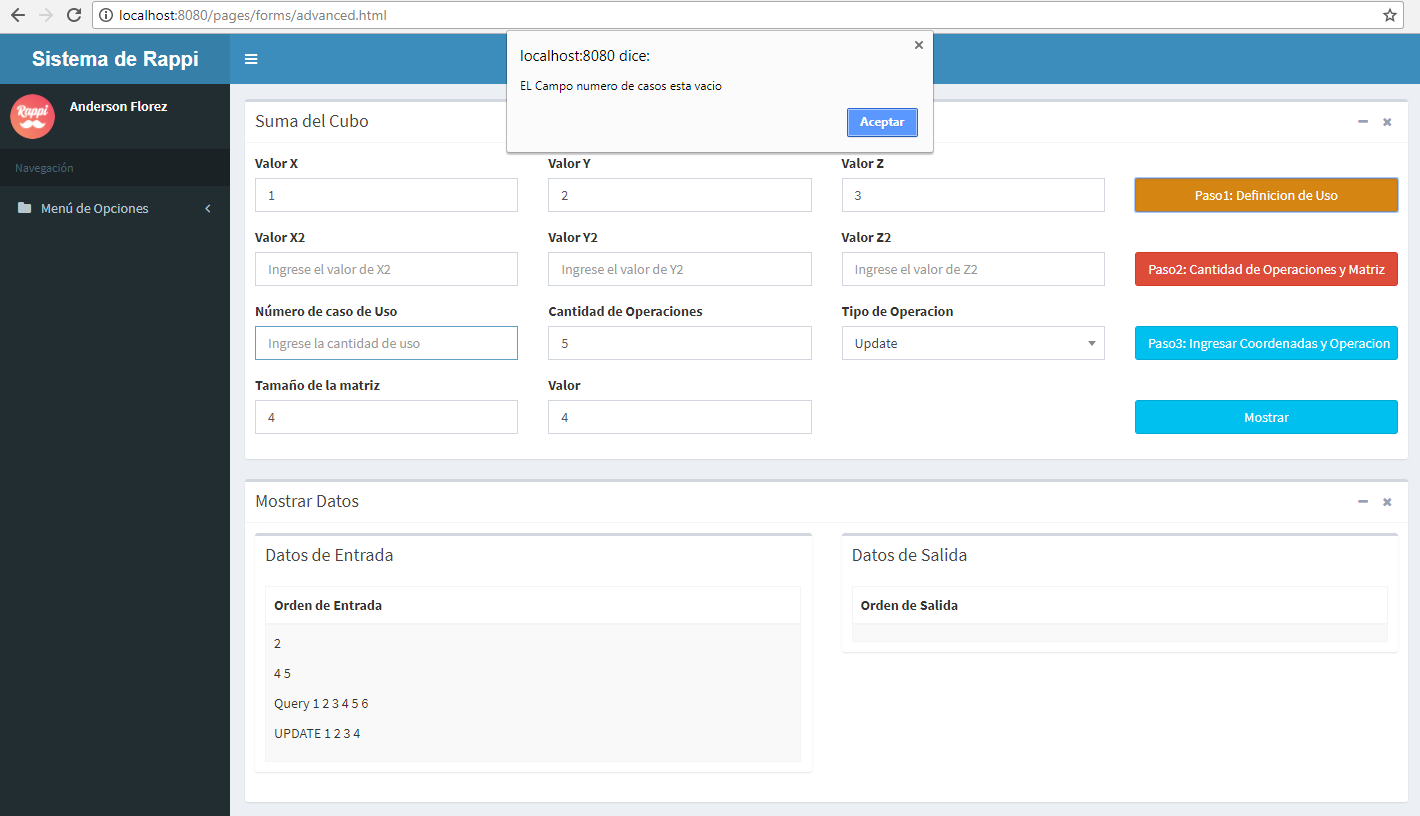


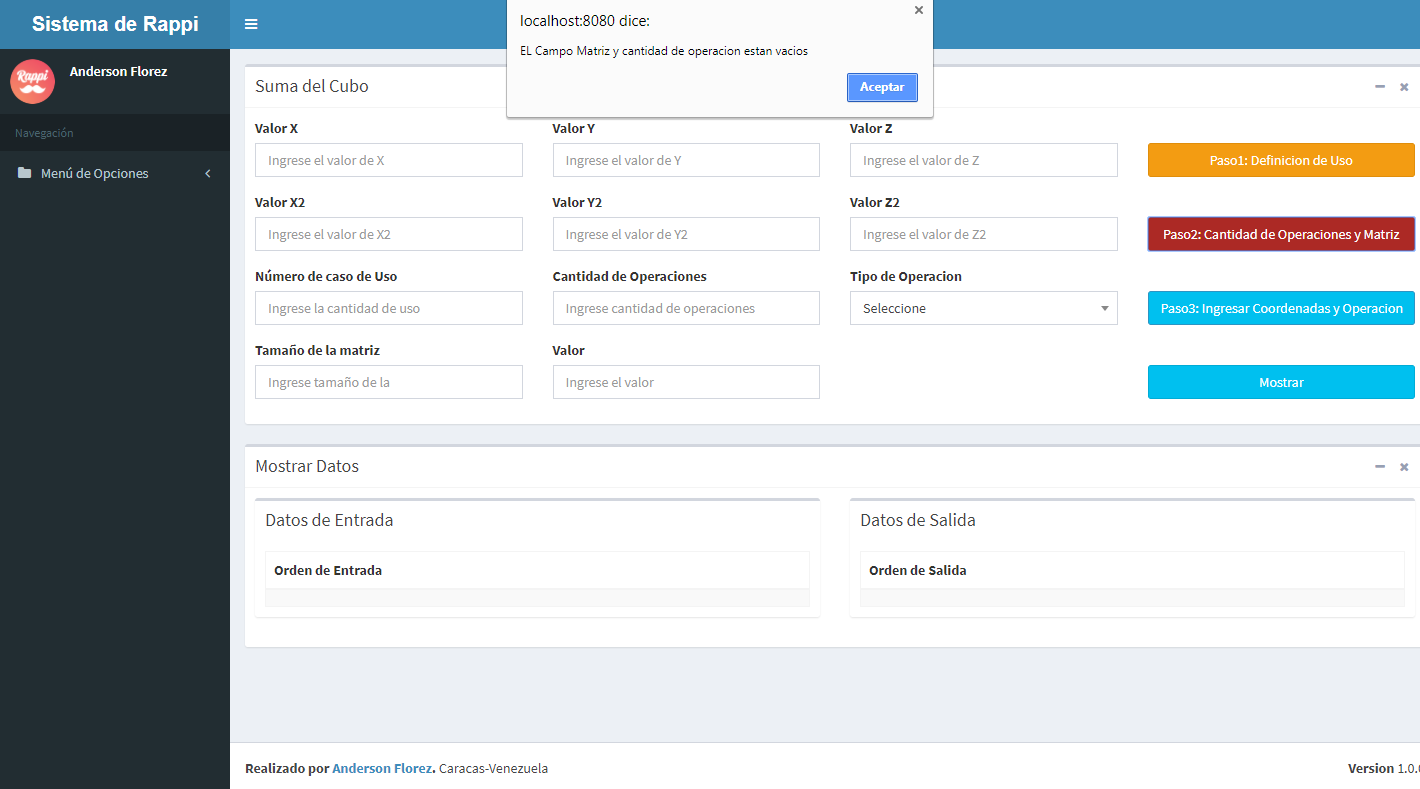
* Ingrese las coordenadas (x,y,z), el valor nuevo y el tipo de operación update y presione sobre el botón “ingresar coordenadas y operación”.



# **Validaciones**

1. No se permite entradas de valores que no contengan información y letras dentro de los campos de texto.

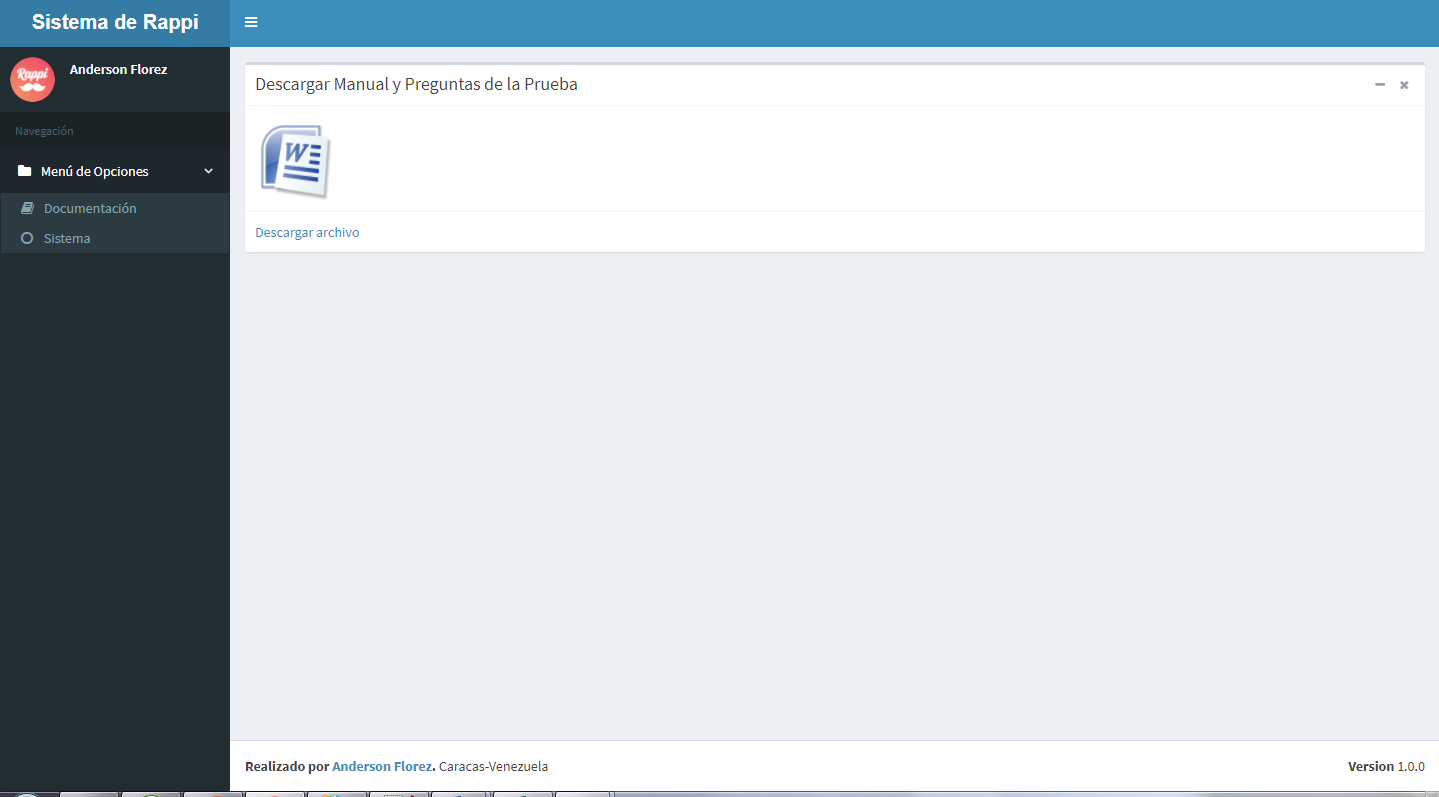




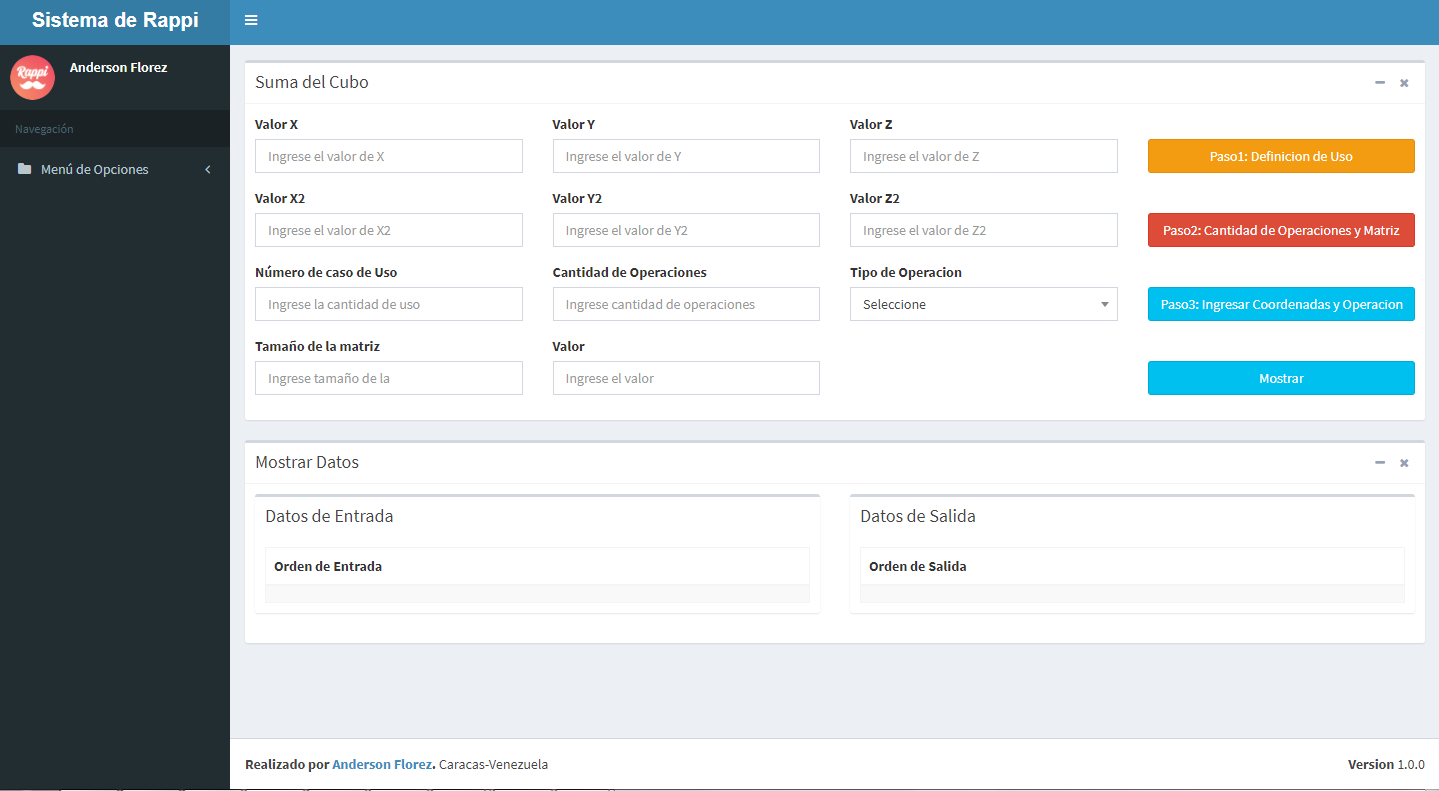
# **Capas de la aplicación**

## Vista

Index.html: contiene el menú donde se encontrará la documentación y la funcionalidad del cubo



pages/forms/advanced.html: Contiene la página donde se ejecutará las funcionalidades del sistema y que se comunicará con el controlado.



## Controlador

Contiene las funcionalidades que hace posible la ejecución del sistema a continuación se detallan:

* DefinirUso() : Agrega la cantidad de uso del sistema.
* TamanoMatriz(): Agrega el tamaño de la matriz y cantidad de operaciones.
* IngresarOperaciones(): Agrega las coordenadas y tipo de operación.
* limpia (): limpia los valores del sistema.
* GenerarSalida(): Genera la respuesta de los valores ingresados.
* suma\_cubo(): Suma los valores cada vez que se ejecuta un query.
* cablear(): Contiene valores ya pre cargados que permiten validar que el flujo del sistema es el correcto.

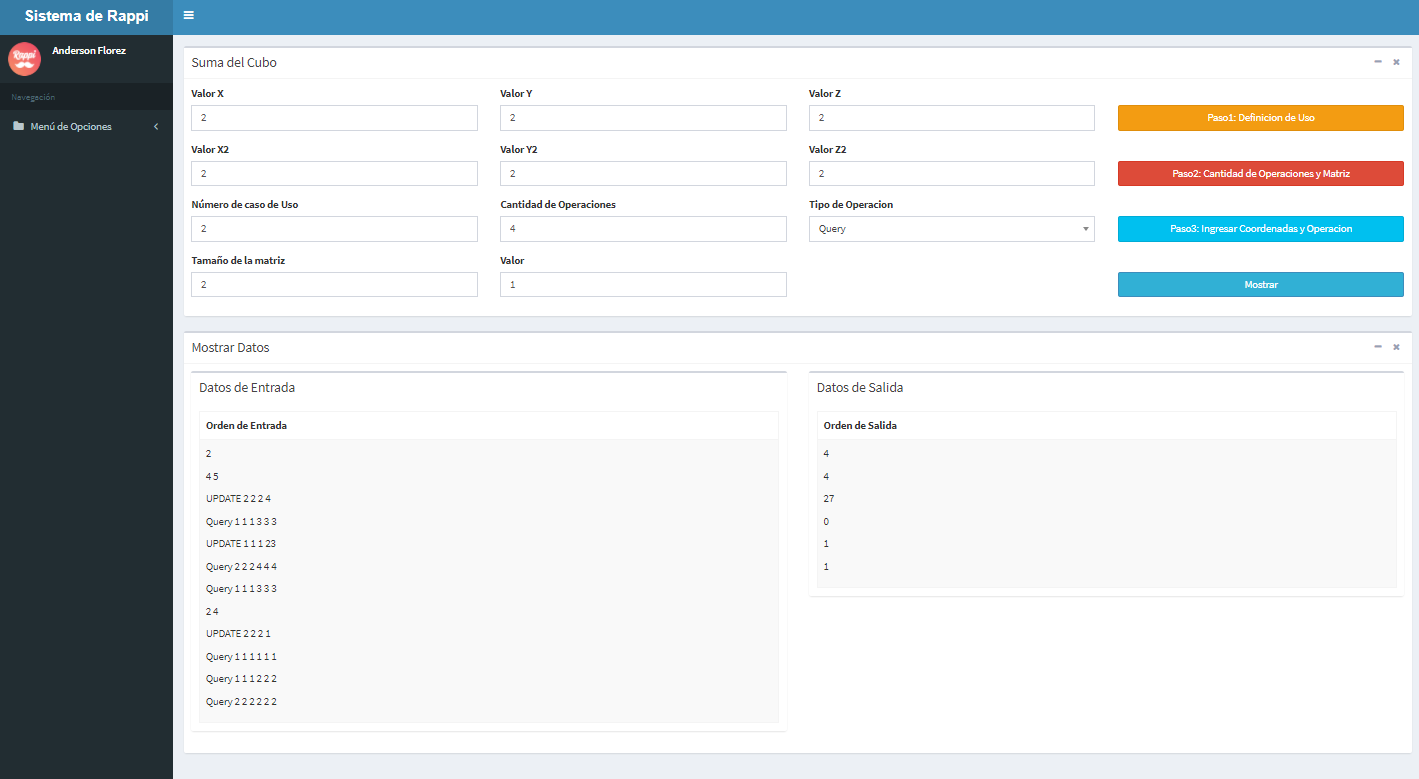
El controlador se encuentra en la siguiente ruta

<script src="../../controlador/js.js"></script>

# **Pruebas Unitarias.**

Ingresar la secuencia indicada en el sistema.

* 2
* 4 5
* UPDATE 2 2 2 4
* Query 1 1 1 3 3 3
* UPDATE 1 1 1 23
* Query 2 2 2 4 4 4
* Query 1 1 1 3 3 3
* 2 4
* UPDATE 2 2 2 1
* Query 1 1 1 1 1 1
* Query 1 1 1 2 2 2
* Query 2 2 2 2 2 2



# **Parte II: CODE REFACTORING**

## Las malas prácticas de programación que en su criterio son evidenciadas en el código.

1. Se puede observar en el código líneas comentadas de modificaciones realizadas anteriormente y que ahora ya no se utilizan generando código basura y que el programa tenga comentarios que puedan dificultar la legibilidad del código.

Ejemplo: //Up carro

//, 'pwd => md5(Imput::get('pwd'))

1. También se observa que no en todo el código se maneja el criterio de excepciones para cuando no se consiguen los valores.

Ejemplo: Driver::update(Imput::get('driver\_id'),array(

"available"=>'0'

));

## Cómo su refactorización supera las malas prácticas de programación.

<?php

public function post\_confirm

{

//CAPTURAR VARIABLES

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

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

if($servicio!=null)//RESPUESTA DEL SERVICIO

{

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

{

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

}

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

{

$servicio=Service::update($id,array(

'driver\_id' => Imput::get ('driver\_id'),

'status\_id' => '2'

));

Driver::update(Imput::get('driver\_id'),array(

"available"=>'0'

));

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

service::update($id,array(

'car\_id'=>$driverTmp ->car\_id

));

//Notificar a usuario!!!

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

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

$push =Push::make();

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

{

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

}

if($servicio->user->type=='1') //Respuesta IOS//

{

$result =$push ->ios($servicio->user->uuid,$pushMessage,1,'honk.wav','open',array('serviceid'=> $servicio->id));

}

else //Respuesta ANDROID//

{

$result =$push ->android2($servicio->user->uuid,$pushMessage,1,'default','open',array('serviceid'=> $servicio->id));

}

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

}

else

{

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

}

}//EL SERVICIO RESPONDIO NULO

else

{

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

}

}

?>

# **Parte III: PREGUNTAS**

## ¿En qué consiste el principio de responsabilidad única? ¿Cuál es su propósito?

El principio de responsabilidad establece que cada módulo o clase creada debe tener responsabilidad sobre una sola parte de la funcionalidad software y no del programa completo, haciendo posible la modificación, validación, y reingeniería del código fuente.

## ¿Qué características tiene según su opinión “buen” código o código limpio?

* Líneas de código comentada para definir qué clase y objetos se crean y su funcionalidad.
* Sin importar el lenguaje de programación desarrollar funcionalidades que no exceden en líneas de códigos para que pueda permitirse modificar y hacer reingeneria del código en el futuro por otros programadores.
* Se debe crear métodos que permitan reutiliza el código sin necesidad de escribirlo varias veces.
* El software desarrollado debe trabajar bajo una arquitectura MVC para permitir la legibilidad del código y posterior modificación.

## ¿Qué es un microservicio?, ventajas y desventajas de los micorservicios

Son pequeños servicios (tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones) los cuales se ejecutan de forma autónoma y se comunican entre sí.

Ventajas

* Se pueden usar diferentes lenguajes de programación.
* Fácil integración y despliegue.
* Reducción de tiempo de desarrollo.

Desventajas

* Alto consumo de memoria.
* Si se cuenta con un gran número de servicios, integrarlos y gestionarlos puede resultar muy complejo.