API REST CALCULATOR

Prueba Técnica para APPGATE

Autor: José Luis Caicedo González

04-04-2022

Contenido

1. INTRODUCCION

En el documento se registra la información técnica acerca de la API REST útil para realizar operaciones matemáticas, donde en la interacción parte de obtener un código para cada usuario, el cual le permite identificarlo para ir agregando los números (operandos) y finalmente realizar el cálculo matemático con los valores suministrados. Las operaciones que permite realizar son: suma, resta, multiplicación, división y potenciación.

Este documento se organiza en los capítulos:

- Análisis
- Diseño: clases principales entre capas, diseño de clases para la calculadora,
- Diseño de servicio: descripción de recursos
- Requisitos: tecnologías utilizadas
- Distribución: ubicación en Github
- Empaquetado y despliegue: orientan en cómo ejecutar el programa
- Pruebas: pruebas de unidad, pruebas funcionales
- Arquitectura despliegue: propuesta para disponibilidad, escalamiento y elástico

2. ANÁLISIS

El cliente requiere realizar las siguientes funciones:

- Obtener un código para identificar sus datos
- Agregar números y puede agregar N números para operación
- Realizar las siguientes operaciones matemáticas con los números agregados:
- Suma, resta, multiplica, división, potenciación
- Indicar que quiere agregar el resultado después de la operación para utilizar en otra operación
- Consultar números
- Eliminar un numero
- Eliminar todos los números
- Eliminar el código de identificación

3. DISEÑO

El diagrama presenta el diseño de las clases principales entre las capas de la aplicación, utilizando el patrón fachada para desacoplar el acceso al almacenamiento de datos.

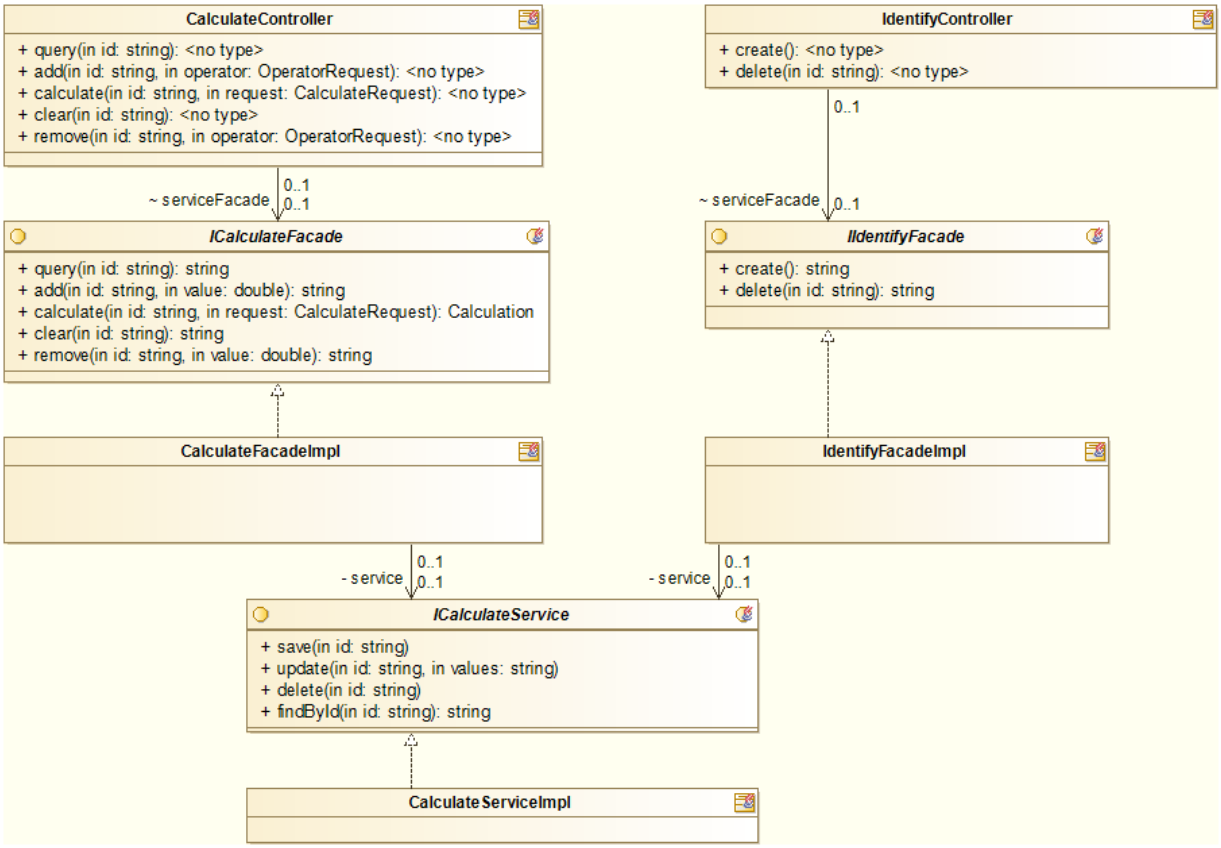


Figura 1. Modelo Clases de capas

Clase	Descripción
IdentifyController	Clase controlador para exponer operaciones de generación de identificador para el usuario y eliminar el identificador
CalculateController	Clase controlador para exponer operaciones de adición, eliminar, eliminar todos números y realizar calculo matemático
ICalculateFacade	Interface para exponer el servicio de cálculos. Se aplica el patrón fachada
IIdentifyFacade	Interface para exponer el servicio de código de identificación fachada
CalculateFacadeImpl	Implementa la lógica de interacción con almacenamiento y reglas generales para cálculos
IdentifyFacadeImpl	Implementa la lógica de interacción con almacenamiento para generar código identificación y eliminación
ICalculateService	Interface que expone el servicio de almacenamiento
CalculateServiceImpl	Clase encargada de realizar operaciones sobre el almacenamiento de redis

El diagrama de clases figura 2, presenta el diseño de las clases para implementar la calculadora, utilizando el patrón método fabrica "Factory Method", se implementa un método para la construcción de objetos operación como: Addition (adición), Sustraction (Sustracción), Multiplication (multiplicación), Division (división), Exponential (exponencial). También patrón reflexión para desacoplar la referencia a clases concretas permitiendo cambios estructurales de forma dinámica, de esta manera en el caso de necesitar la adición de una nueva operación matemática, solo implicara agregar la nueva clase al paquete de operaciones.

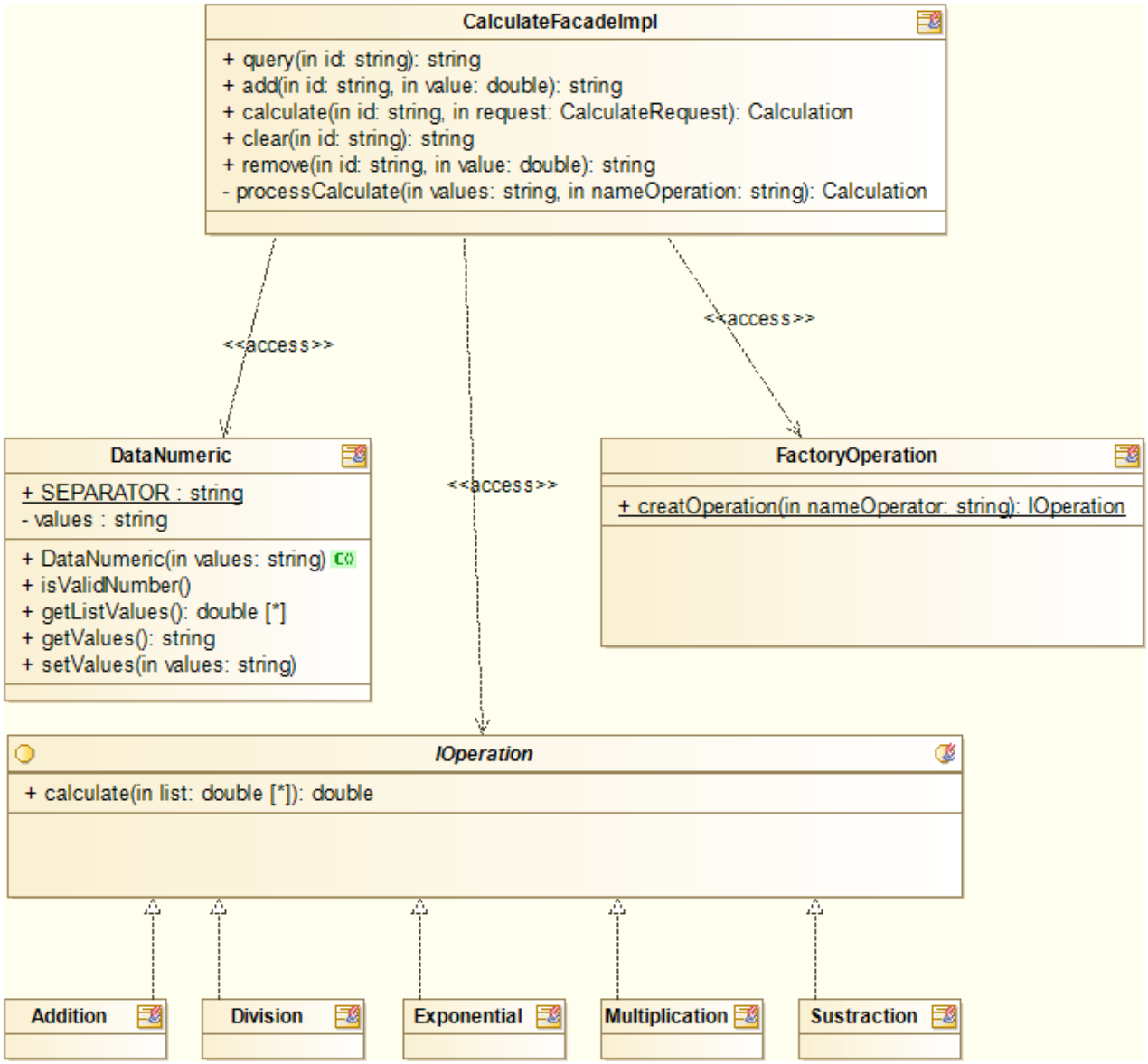


Figura 2. Modelo Clases Calculadora

Clase	Descripción
DataNumeric	Clase encapsula los valores utilizados para realizar el calculo y realizar la transformación a tipos numéricos para disponerlos a las clase de operación matemática
FactoryOperation	Clase aplica el patrón método de fábrica para instanciar clases que implementan y realizan la operación matemática. También utiliza el patrón reflexión para obtener la instancia de forma dinámica.

IOperation	Interface a través de la cual se generaliza la operación de cálculo matemático mediante la declaración del método calculate
Addition	Clase implementan el cálculo para realizar la operación matemática de adición (suma)
Sustraction	Clase implementan el cálculo para realizar la operación matemática de resta o sustracción
Multiplication	Clase implementan el cálculo para realizar la operación matemática de multiplicación
Division	Clase implementan el cálculo para realizar la operación matemática de división
Exponential	Clase implementan el cálculo para realizar la operación matemática de aplicar potencia

4. DISEÑO DE SERVICIO

METODO	RECURSO	DESCRIPCION
POST	http://localhost:8080/identify	Crear el identificador
DELETE	http://localhost:8080/identify	Elimina el identificador Parámetro en cuerpo: id es Identificador de registro a eliminar
POST	http://localhost:8080/calculator/number	Agregar número (operando), Parámetro en cabecera: id es Identificador de registro En el cuerpo: parámetro “value” el valor a agregar. Retorna mensaje de confirmación con los números agregados hasta el momento.
PUT	http://localhost:8080/calculator/number	Actualiza los números agregados, removiendo el valor suministrando Parámetro id Identificador de registro en la cabecera En el cuerpo parámetro “value” es el valor a eliminar Retorna números disponibles después de eliminar
DELETE	http://localhost:8080/calculator/number	Eliminar todos los valores agregados Parámetro id Identificador de registro en la cabecera Retorna mensaje de confirmación
GET	http://localhost:8080/calculator/number	Consulta los números agregados hasta el momento Parámetro en cabecera: id es Identificador de registro Retorna lista números agregados hasta el momento
POST	http://localhost:8080/calculator/operation	Realizar el cálculo matemático Parámetro id Identificador de registro en la cabecera En el cuerpo el parámetro operation : es nombre de la operación matemática a realizar: Addition Sustraction Multiplication Division Exponential isAddResult : es valor booleano para indicar true: eliminar los números actuales y se agregar el resultado para el siguiente calculo false: no agrega el resultado y conserva los numero agregados

5. REQUISITOS

El empaquetado y despliegue requiere tener instalado y configurado las siguientes:

- JDK 11
- Maven versión 3.8.1
- Git 2.31.1
- Docker 20.10.13
- Spring Tool Suite

6. DISTRIBUCIÓN

El proyecto se encuentra en GitHub bajo un repositorio público el enlace:
https://github.com/jcaicedo7/calculator.git

7. EMPAQUETADO Y DESPLIEGUE

1. Descargar imagen redis:
docker pull redis

2. Desplegar redis:
docker run -d --name redis -p 6379:6379 redis
En caso de tener el puerto ocupado cambiarlo

3. Clonar el proyecto calculator :
git clone https://github.com/jcaicedo7/calculator.git
En caso de haber cambiado el puerto de redis modificar colocando el puerto e IP (si es necesario) en el siguiente el archivo
\calculator\src\main\resources\ application.properties

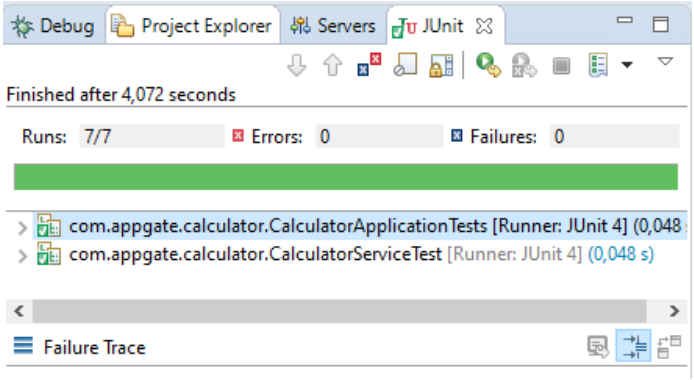
4. Instalar
mvn install

5. Desplegar
mvn spring-boot:run

8. PRUEBAS

Pruebas de Unidad

Mediante la herramienta “Spring Tool Suite” sobre el paquete test se dispone de código de pruebas de unidad implementación en Junit, con 5 casos, a continuación se presenta el resultado de ejecusiòn:

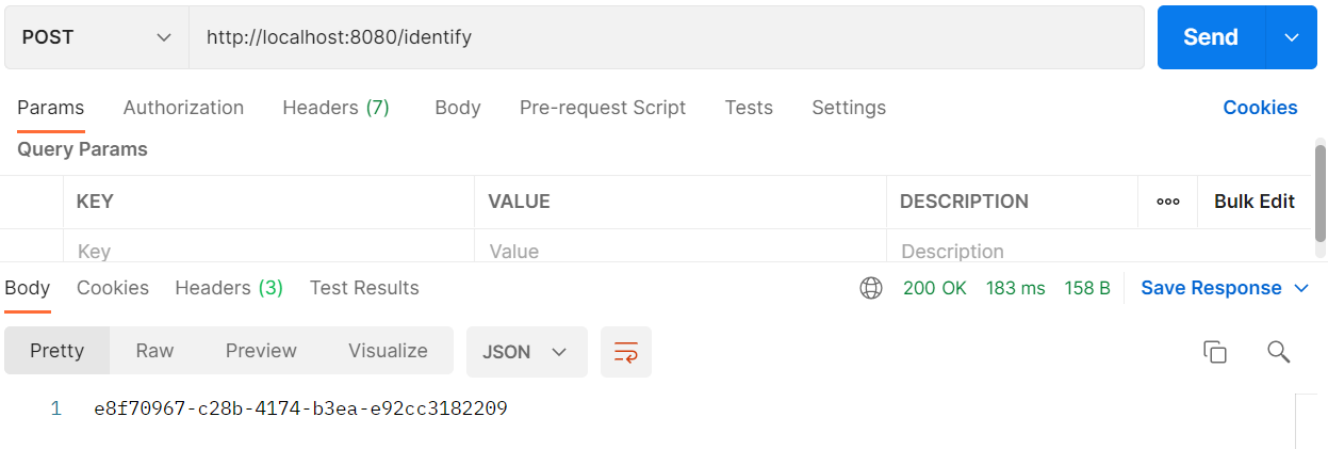


Pruebas Funcionales

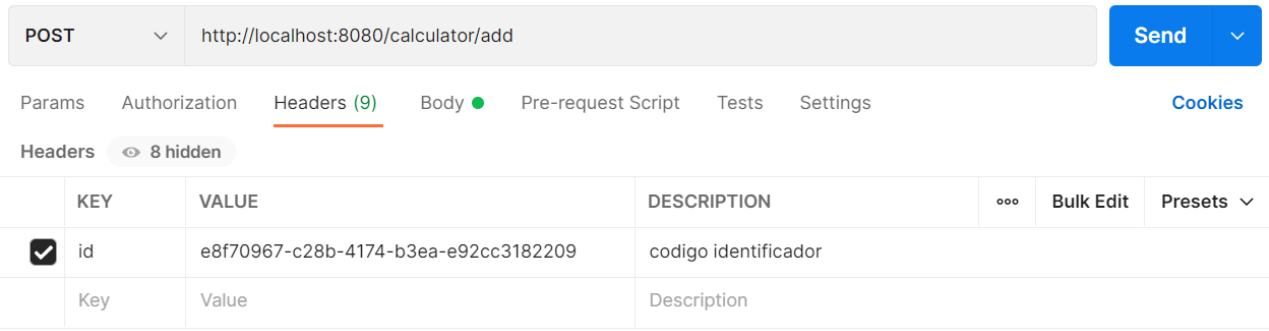
CASO DE PRUEBA 1

Objetivo: Obtener código identificador, agregar números, realizar operación de suma y consultar números agregados.

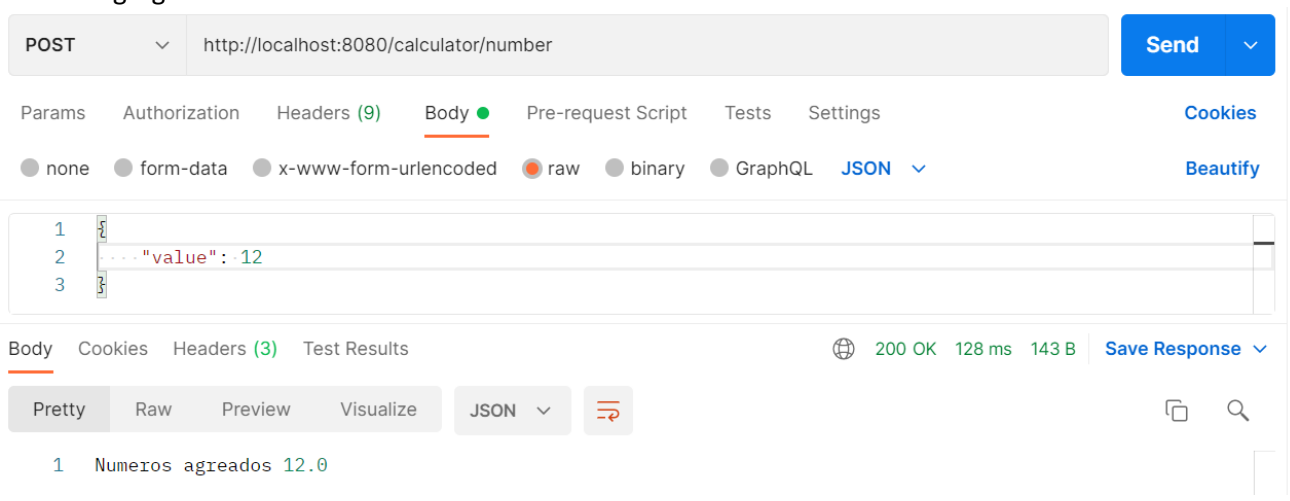
1. Obtener código identificador



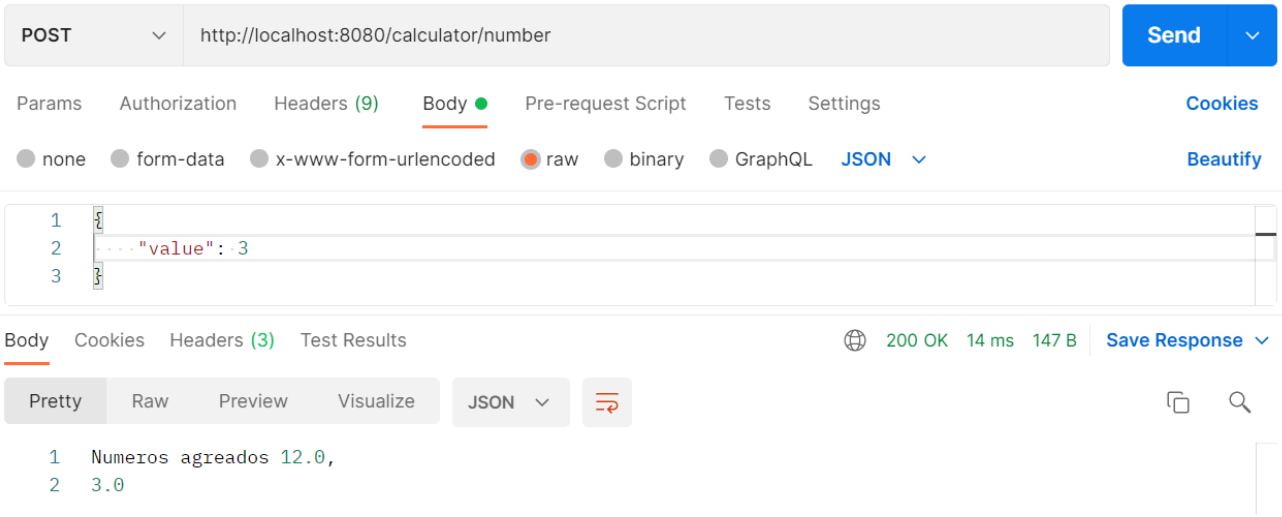
2. Agregar código a la cabecera



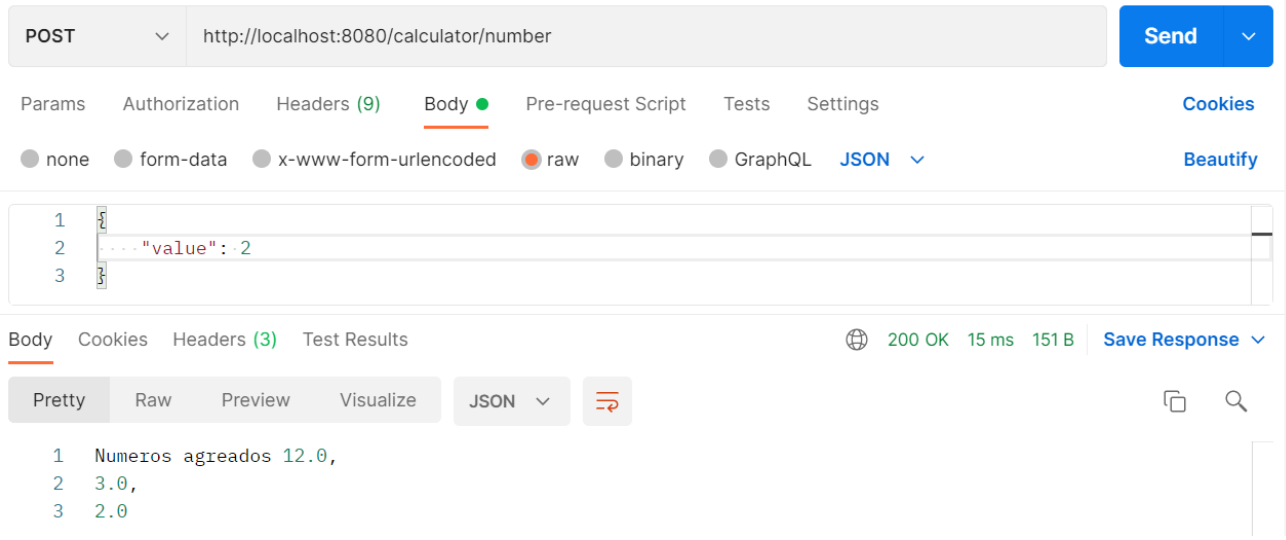
3. Agregar número 12



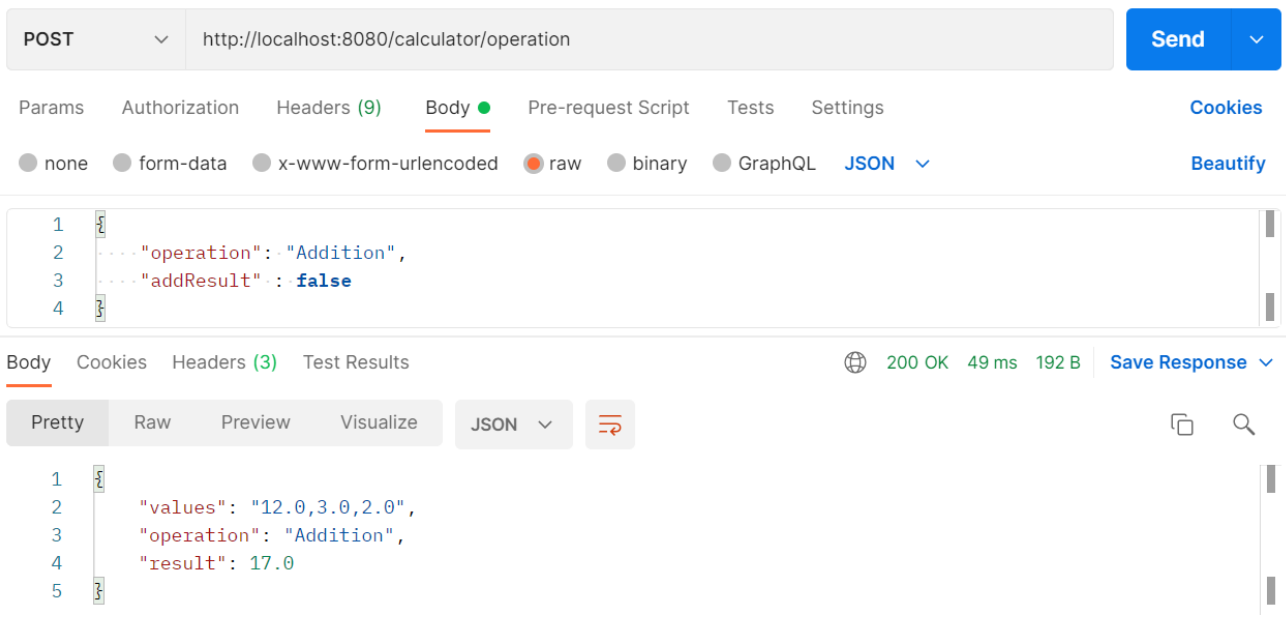
4. Agregar número 3



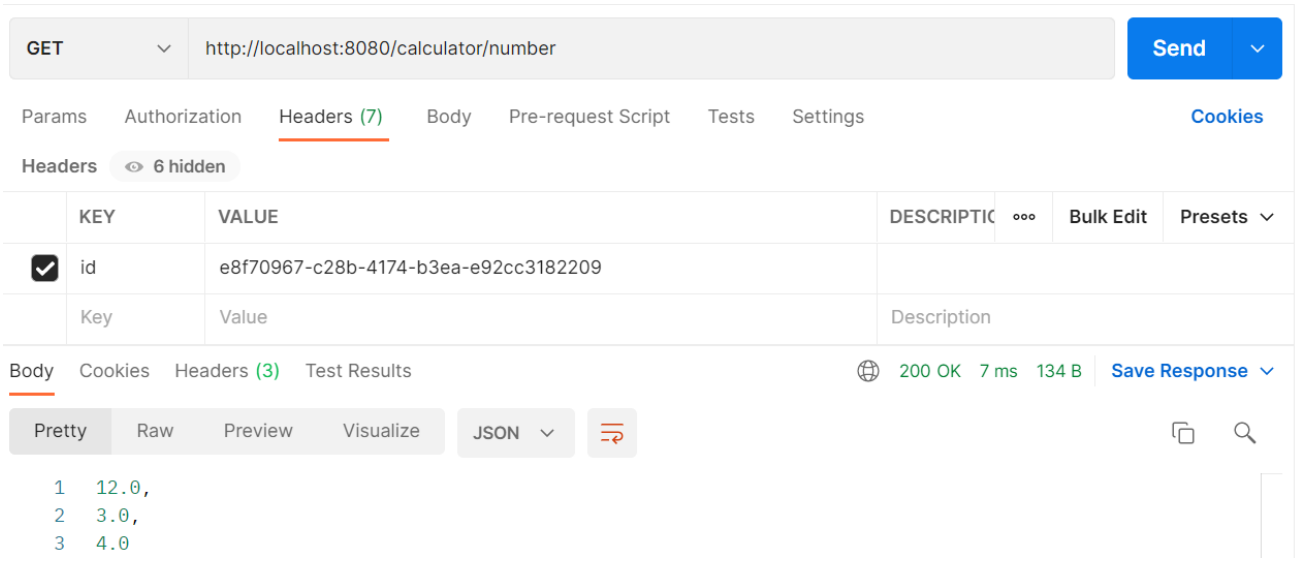
5. Agregar número 2



6. Realizar suma

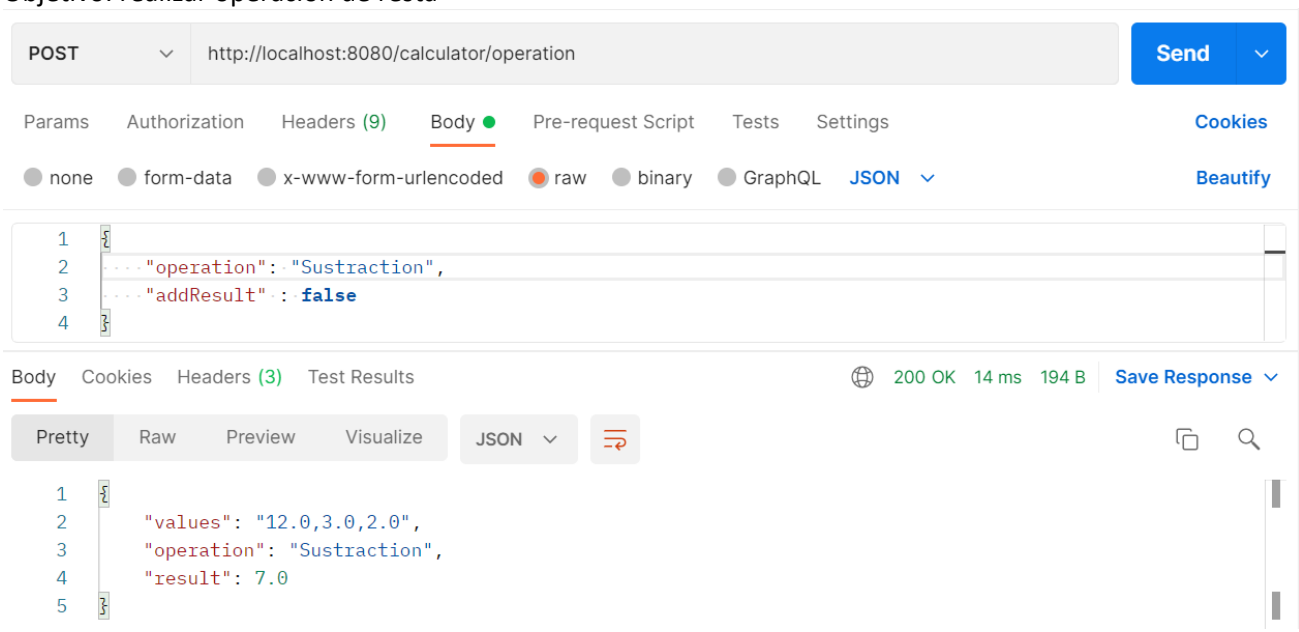


7. Realizar consulta



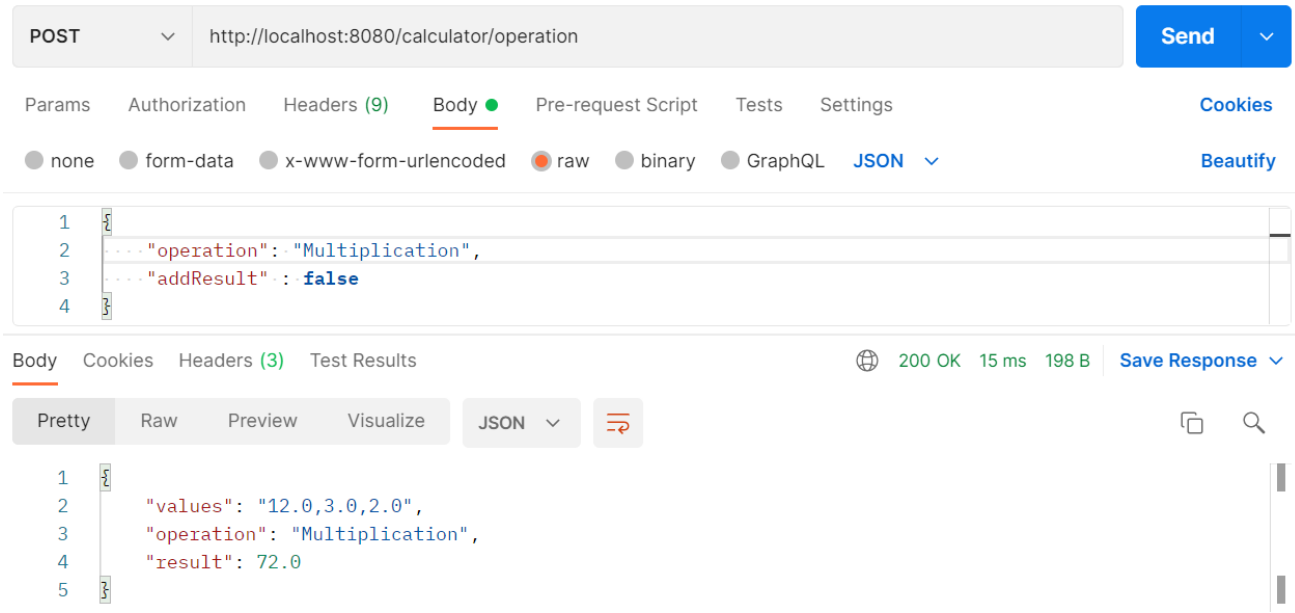
CASO DE PRUEBA 2

Objetivo: realizar operación de resta



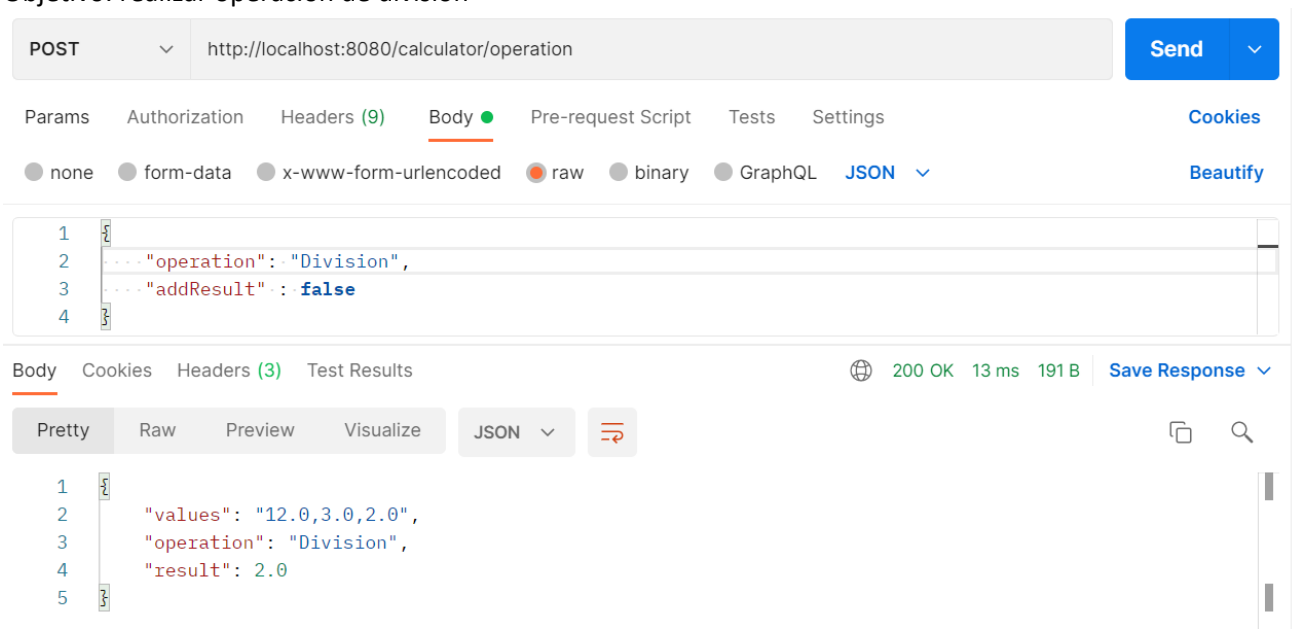
CASO DE PRUEBA 3

Objetivo: realizar operación de multiplicación



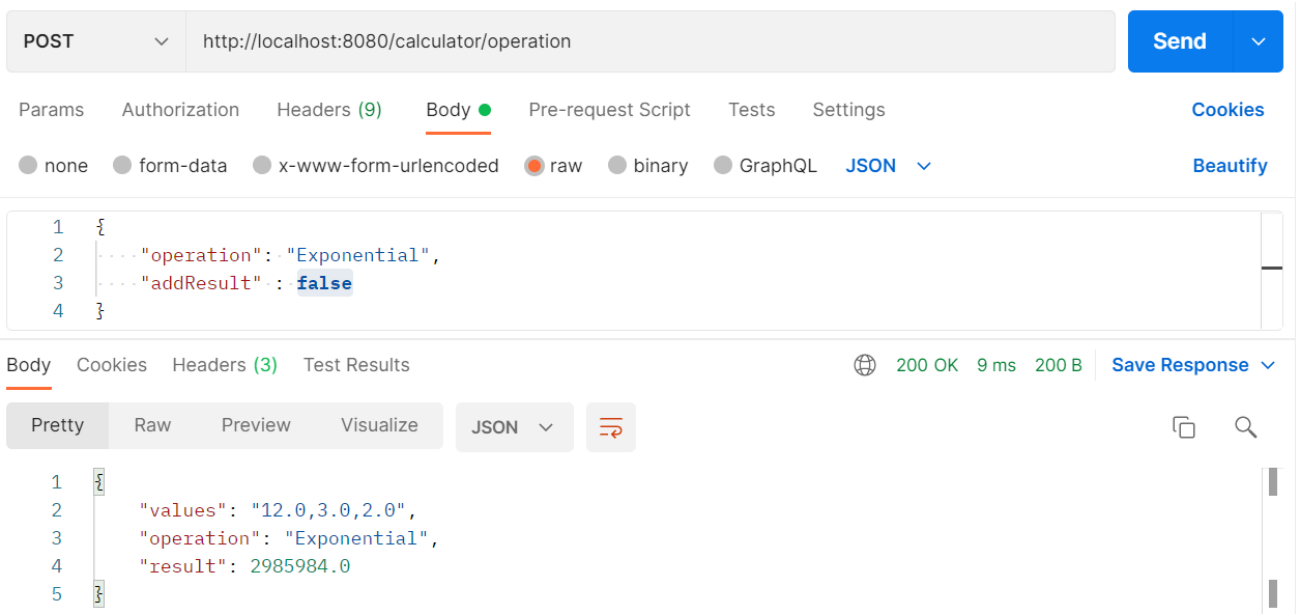
CASO DE PRUEBA 4

Objetivo: realizar operación de división



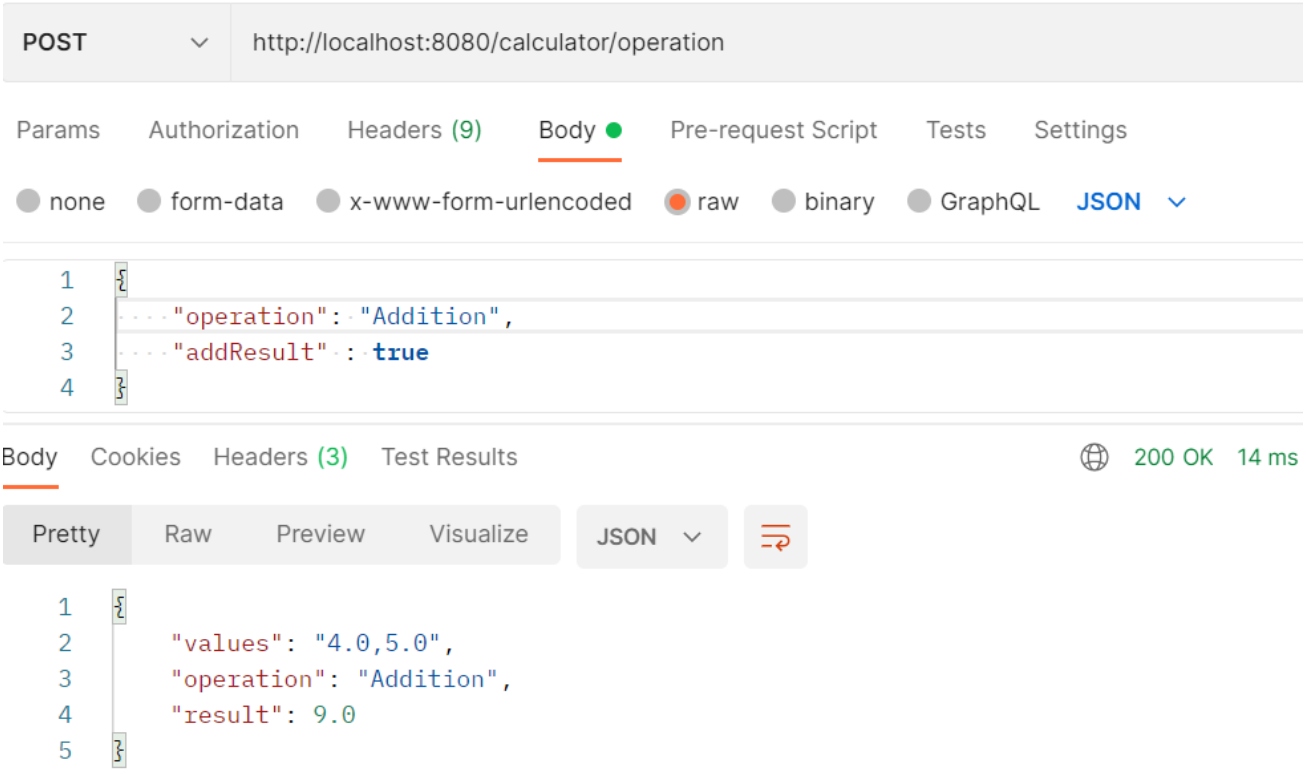
CASO DE PRUEBA 5

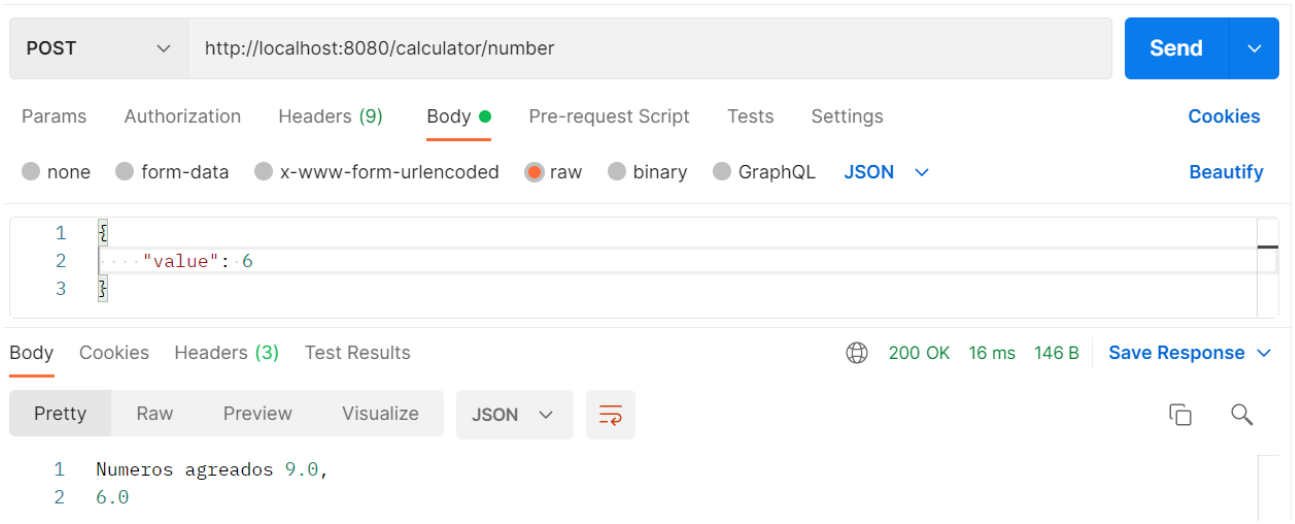
Objetivo: realizar operación de exponencial



CASO DE PRUEBA 6

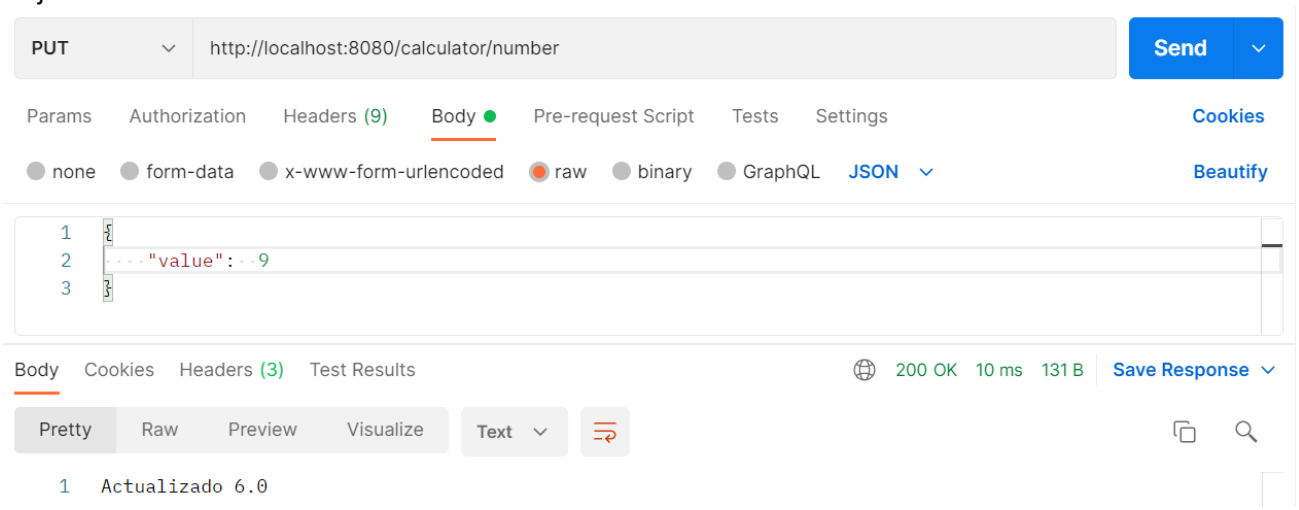
Objetivo: Agregados los números 4 y 5, agregar el resultado colocando addResult en true, luego agregar 6 e identificar que se dispone de resultado (9) y el nuevo numero





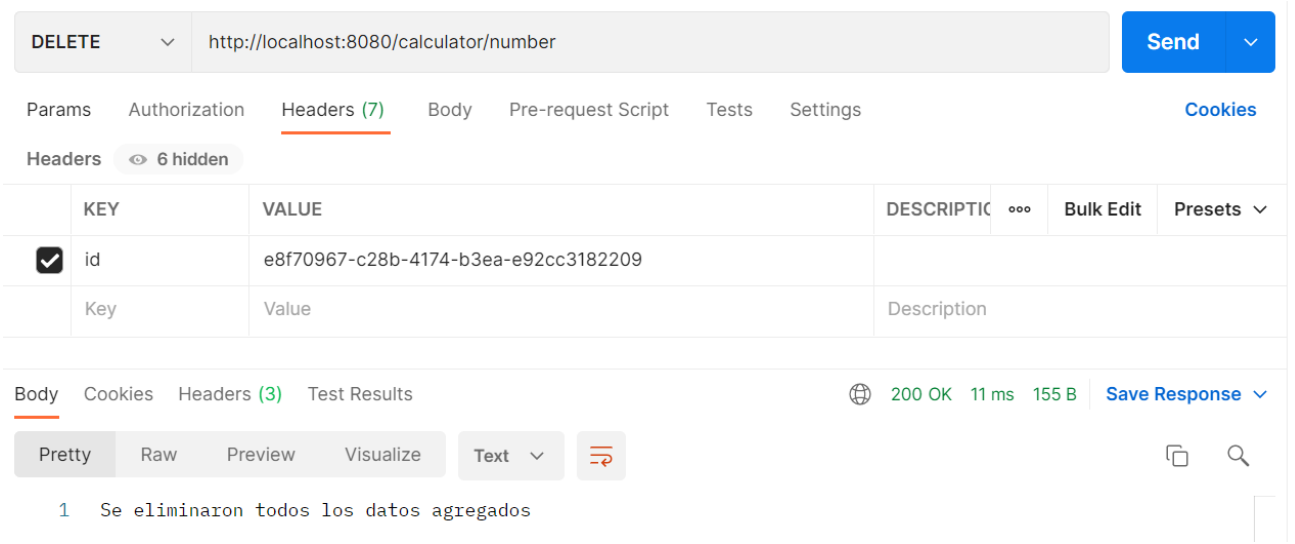
CASO DE PRUEBA 7

Objetivo: Remover un numero



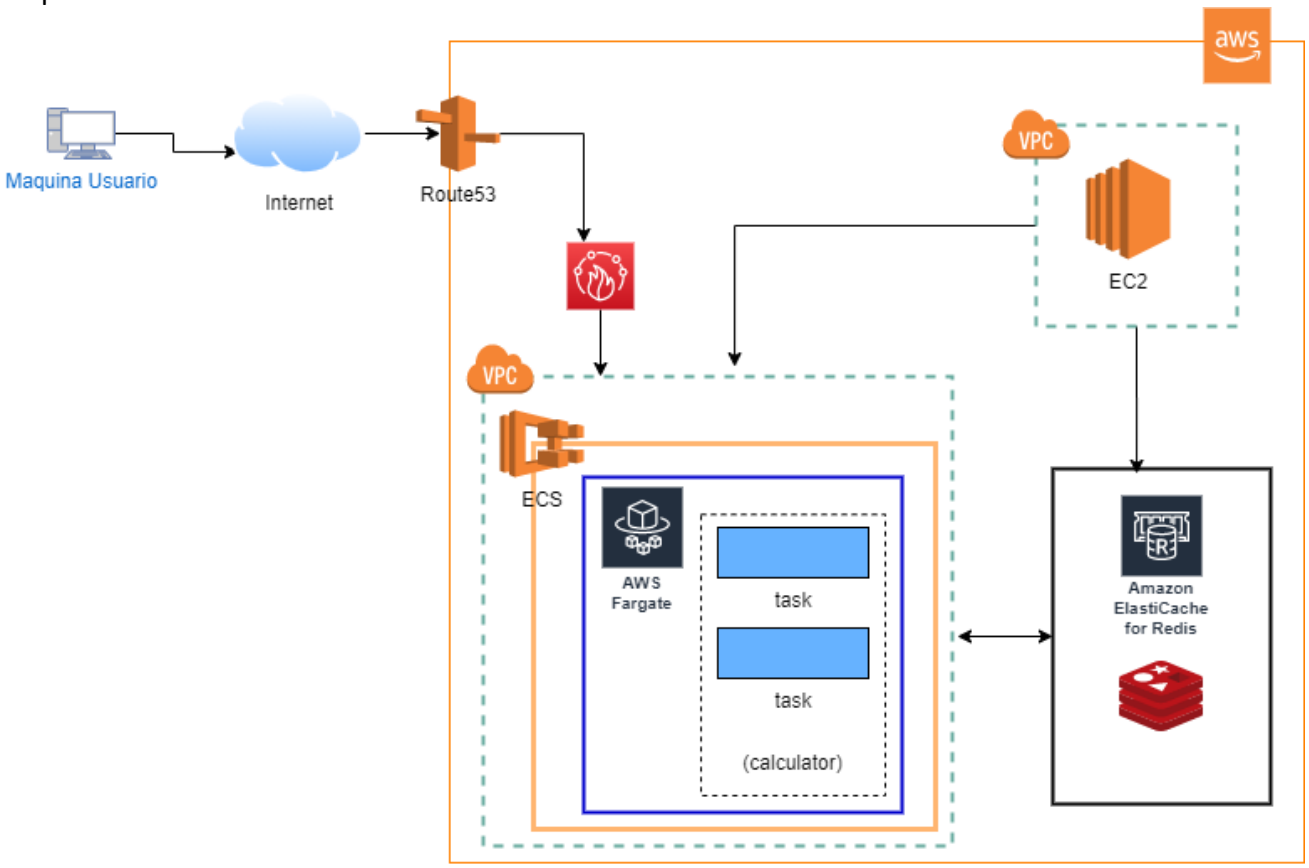
CASO DE PRUEBA 8

Objetivo: Eliminar todos los números



9. ARQUITECTURA DESPLIEGUE

Considerando los posibles requisitos no funcionales para tener alta disponibilidad, escalable, y elasticidad, a continuación se presenta un diagrama de despliegue con el cual se propone la arquitectura basada en los servicio de AWS.



La aplicación “calculator” se ejecuta bajo el contenedor docker orquestado por el servico El actual contenedor docker se desplegaría bajo el orquestador de contenedores ECS (Elastic Container Service) de esta forma dispone de elasticidad en el consumo de recursos, también mediante la capacidad de incrementar o disminuir instancias ofrece escalabilidad horizontal, en cuanto a la infraestructura utiliza aws fargate para disponer de infraestructura auto administrada. El almacenamiento utilizaría el servicio de “Elastic Cache” para Redis y mediante una instancia de EC2 se implementaría un bastión para el mantenimiento.