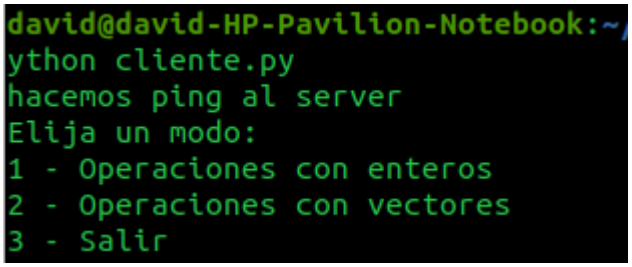
Desarrollo de Sistemas Distribuidos - Práctica 2: Apache Thrift

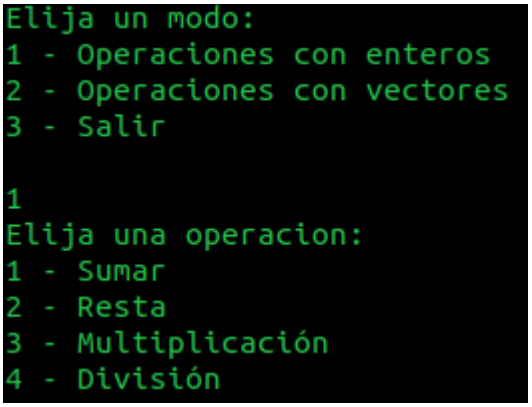
David Jesús Ruiz de Valdivia Torres 3º, 78006825-W

Mi propuesta de calculadora tiene dos modos de ejecución, operaciones con enteros y con vectores tridimensionales. Se deberá ejecutar el servidor en una terminal para que en la otra se pueda ejecutar el cliente. El programa cliente hará un ping al servidor y recibirá al usuario con un menú textual:



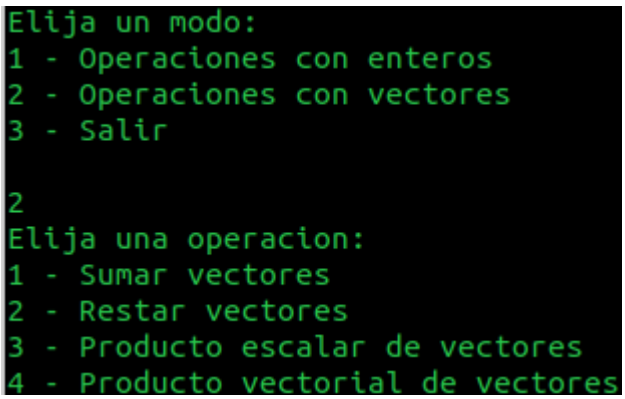
Escribiremos el número de la opción que queramos ejecutar (Si escribimos otra opción el programa nos notificará de nuestro error).

Probemos escribir la opción 1 para operaciones con enteros:
Se nos pedirá que ingresemos primero el primer operando y luego el segundo, por último también nos pedirán la operación a realizar de las disponibles (suma, resta, producto, división). Si no ponemos una operación reconocida, el programa finalizará.



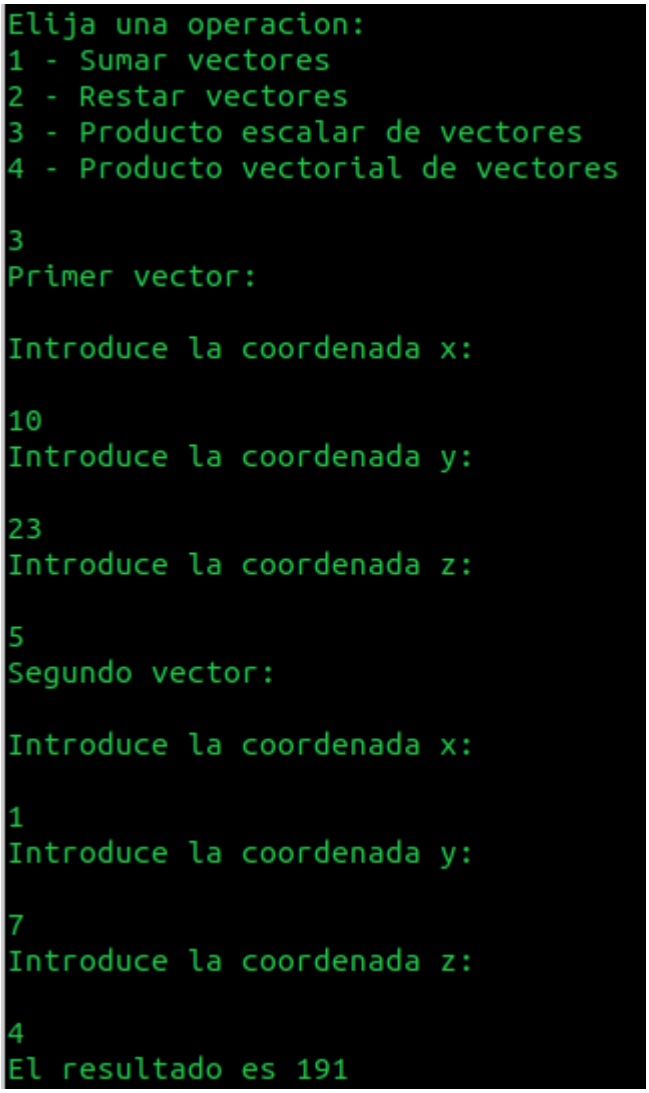
El resto de operaciones funcionan como se espera. (Añado que la división no te permite dividir entre 0, el programa acabaría).

Probemos ahora la opción 2 para operaciones con vectores. Se nos presentará este menú textual:



Al elegir una opción se nos pedirá que ingresemos las coordenadas de los dos vectores una a una (véase función **inputVector()** en cliente.py).

Ejemplo de producto escalar:



Y su implementación en el lado del servidor:

```
def producto_escalar(self, vec1, vec2):
    result = 0
    print("producto vectorial de ") + str(vec1) + " con " + str(vec2)
    for i in range(len(vec1)):
        result += vec1[i] * vec2[i]

    return result
```

Aquí el ejemplo de producto vectorial entre dos vectores:

```
Elija una operacion:
1 - Sumar vectores
2 - Restar vectores
3 - Producto escalar de vectores
4 - Producto vectorial de vectores

4
Primer vector:

Introduce la coordenada x:

10
Introduce la coordenada y:

2
Introduce la coordenada z:

4
Segundo vector:

Introduce la coordenada x:

1
Introduce la coordenada y:

5
Introduce la coordenada z:

3
El resultado es [-14, -26, 48]
```

Y su implementación en el lado del servidor:

```
def producto_vectorial(self, vec1, vec2):
    result = []
    print("producto escalar de ") + str(vec1) + " con " + str(vec2)
    result.append( (vec1[1] * vec2[2]) - (vec1[2] * vec2[1]) )
    result.append( (vec1[2] * vec2[0]) - (vec1[0] * vec2[2]) )
    result.append( (vec1[0] * vec2[1]) - (vec1[1] * vec2[0]) )
    return result
```

El servidor va imprimiendo en su lado las operaciones que va realizando:

```
iniciando servidor...
me han hecho ping()
me han hecho ping()
me han hecho ping()
producto vectorial de [10, 23, 5] con [1, 7, 4]
me han hecho ping()
producto escalar de [10, 2, 4] con [1, 5, 3]
```

Y por último, el archivo .thrift a partir del que hemos generado los demás:

```
service Calculadora{
   void ping(),
   i32 suma(1:i32 num1, 2:i32 num2),
   i32 resta(1:i32 num1, 2:i32 num2),
   i32 multiplicacion(1:i32 num1, 2:i32 num2),
   i32 division(1:i32 num1, 2:i32 num2),
   list<i32> suma_vectores(1:list<i32> vec1, 2:list<i32> vec2),
   list<i32> resta_vectores(1:list<i32> vec1, 2:list<i32> vec2),
   i32 producto_escalar(1:list<i32> vec1, 2:list<i32> vec2),
   list<i32> producto_vectorial(1:list<i32> vec1, 2:list<i32> vec2),
}
```