

Taller 2 - Remote Procedure Call



Pontificia Universidad
JAVERIANA
Colombia

Maria Paula Rodríguez Ruiz

Daniel Felipe Castro Moreno

Juan Enrique Rozo Tarache

Eliana Katherine Cepeda González

Introducción a Sistemas Distribuidos

Dpto. de Ingeniería de Sistemas

Pontificia Universidad Javeriana

Bogotá, Colombia

14 de marzo del 2025

Introducción:

El taller tiene como objetivo demostrar un escenario de computación distribuida donde una máquina virtual central (cliente) recibe dos entradas representando los catetos de un triángulo rectángulo. Esta máquina delega el cálculo del cuadrado de cada cateto a dos máquinas virtuales distintas (servidores). Una vez que la máquina central recibe los resultados de los cuadrados, realiza el cálculo final de la hipotenusa. La comunicación entre estas máquinas se establece mediante gRPC, que actúa como la infraestructura de comunicación que permite que la máquina central envíe solicitudes a las máquinas de cálculo y reciba sus respuestas de manera estructurada y eficiente.

Arquitectura

La arquitectura que se maneja en este taller consiste en 4 máquinas conectadas de la siguiente manera:

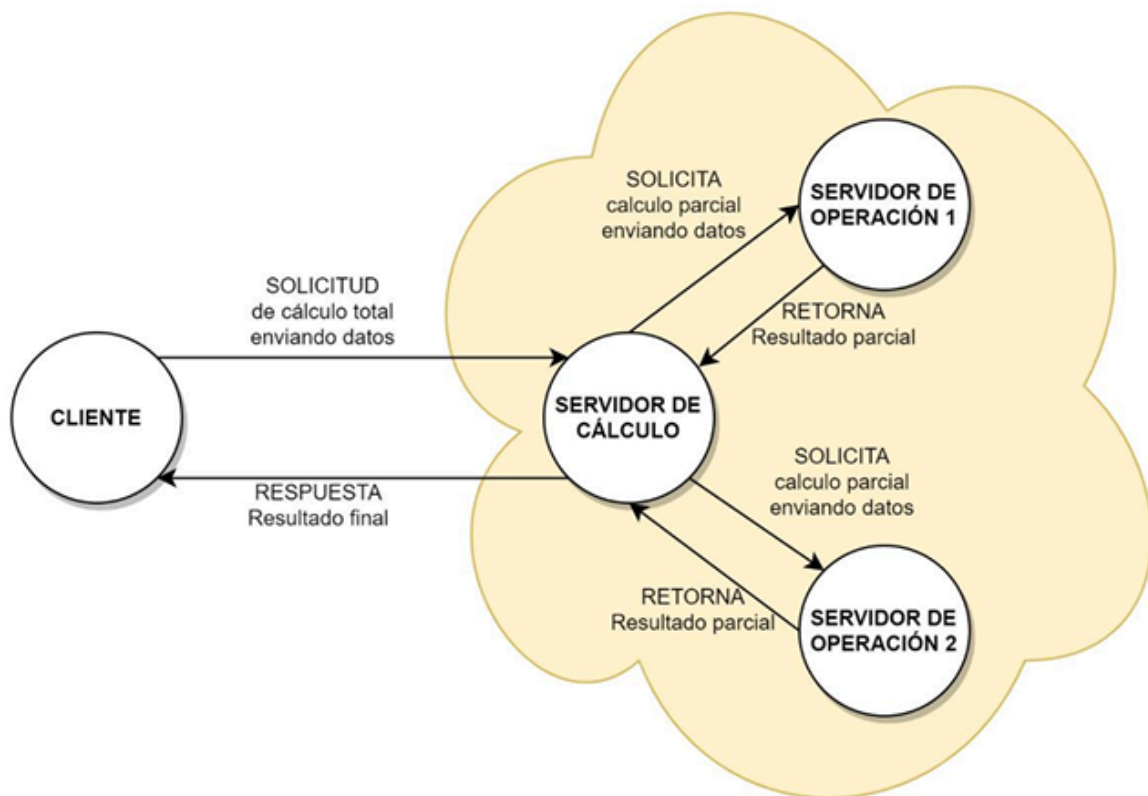


Figura 1. Arquitectura básica

Protocol Buffers y el Archivo calc.proto

En gRPC, se utilizan Protocol Buffers para definir tanto la estructura de los datos que se transmiten como la interfaz de los servicios. Para este taller estamos usando las siguientes estructuras:

```
1  syntax = "proto3";
2
3  package calc;
4
5  // Message definitions for OperationService
6  message OperationRequest {
7      double value = 1;
8  }
9
10 message OperationReply {
11     double result = 1;
12 }
13
14 // Service to square a number
15 service OperationService {
16     rpc Square(OperationRequest) returns (OperationReply);
17 }
18
19 // Message definitions for CalculationService
20 message CalculationRequest {
21     double a = 1;
22     double b = 2;
23 }
24
25 message CalculationReply {
26     double hypotenuse = 1;
27 }
28
29 // Service to calculate the hypotenuse using two remote square operations
30 service CalculationService {
31     rpc Calculate(CalculationRequest) returns (CalculationReply);
32 }
```

Figura 2. Archivo *calc.proto*

Mensajes:

- OperationRequest: Este mensaje contiene un único campo (value) de tipo double, utilizado para enviar el valor de un cateto a una máquina de cálculo para ser elevado al cuadrado.
- OperationReply: Este mensaje contiene un campo (result) de tipo double, que representa el resultado del cálculo del cuadrado enviado por la máquina de cálculo.
- CalculationRequest: Este mensaje contiene dos campos (a y b) de tipo double, que representan las longitudes de los dos catetos proporcionadas a la máquina central.
- CalculationReply: Este mensaje contiene un campo (hypotenuse) de tipo double, que representa el resultado final del cálculo de la hipotenusa.

Servicios:

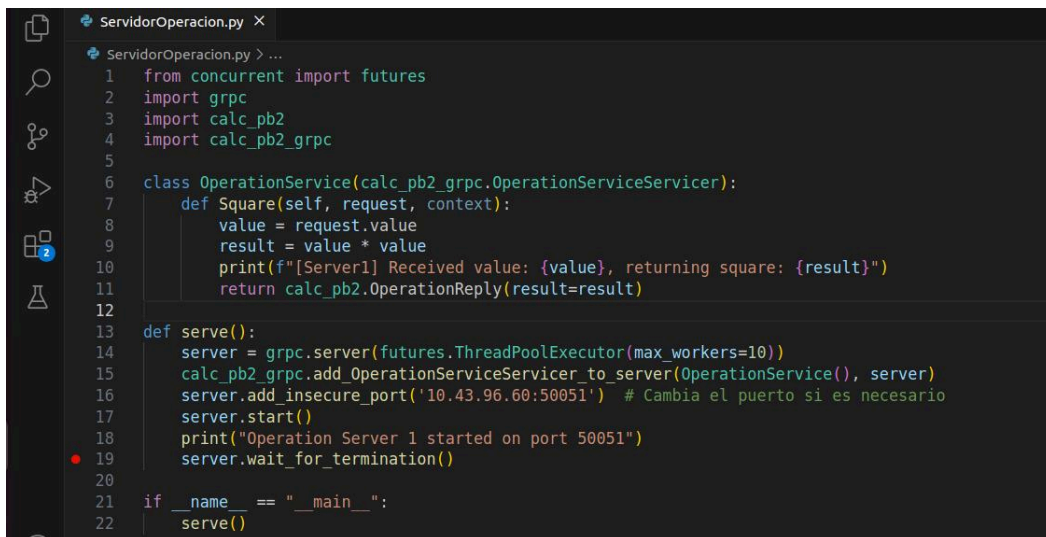
- OperationService: Este servicio define una operación RPC llamada Square. Esta operación toma un *OperationRequest* como entrada y devuelve un *OperationReply*. Este

servicio se implementa en las dos máquinas virtuales dedicadas al cálculo del cuadrado de los catetos.

- **CalculationService:** Este servicio define una operación RPC llamada Calculate. Esta operación toma un *CalculationRequest* como entrada y devuelve un *CalculationReply*. Este servicio se implementa en la máquina virtual central, encargada de recibir las entradas y coordinar el cálculo de la hipotenusa.

Cliente y Servidores

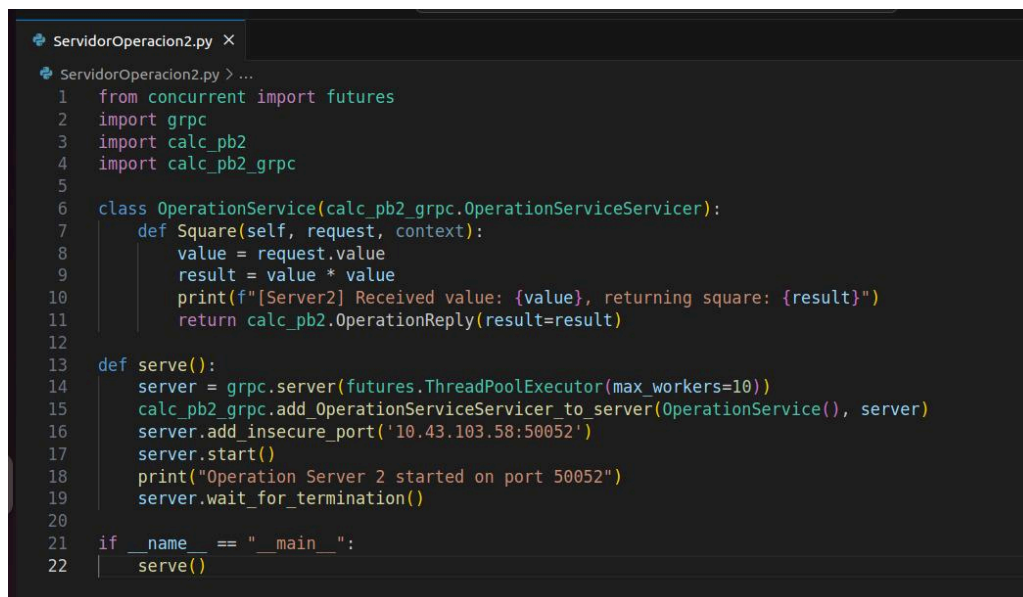
Servidor Operación (10.43.96.60):



```
1 from concurrent import futures
2 import grpc
3 import calc_pb2
4 import calc_pb2_grpc
5
6 class OperationService(calc_pb2_grpc.OperationServiceServicer):
7     def Square(self, request, context):
8         value = request.value
9         result = value * value
10        print(f"[Server1] Received value: {value}, returning square: {result}")
11        return calc_pb2.OperationReply(result=result)
12
13 def serve():
14     server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
15     calc_pb2_grpc.add_OperationServiceServicer_to_server(OperationService(), server)
16     server.add_insecure_port('10.43.96.60:50051') # Cambia el puerto si es necesario
17     server.start()
18     print("Operation Server 1 started on port 50051")
19     server.wait_for_termination()
20
21 if __name__ == "__main__":
22     serve()
```

Figura 3. Programa del Servidor de Operación con ip 10.43.96.60

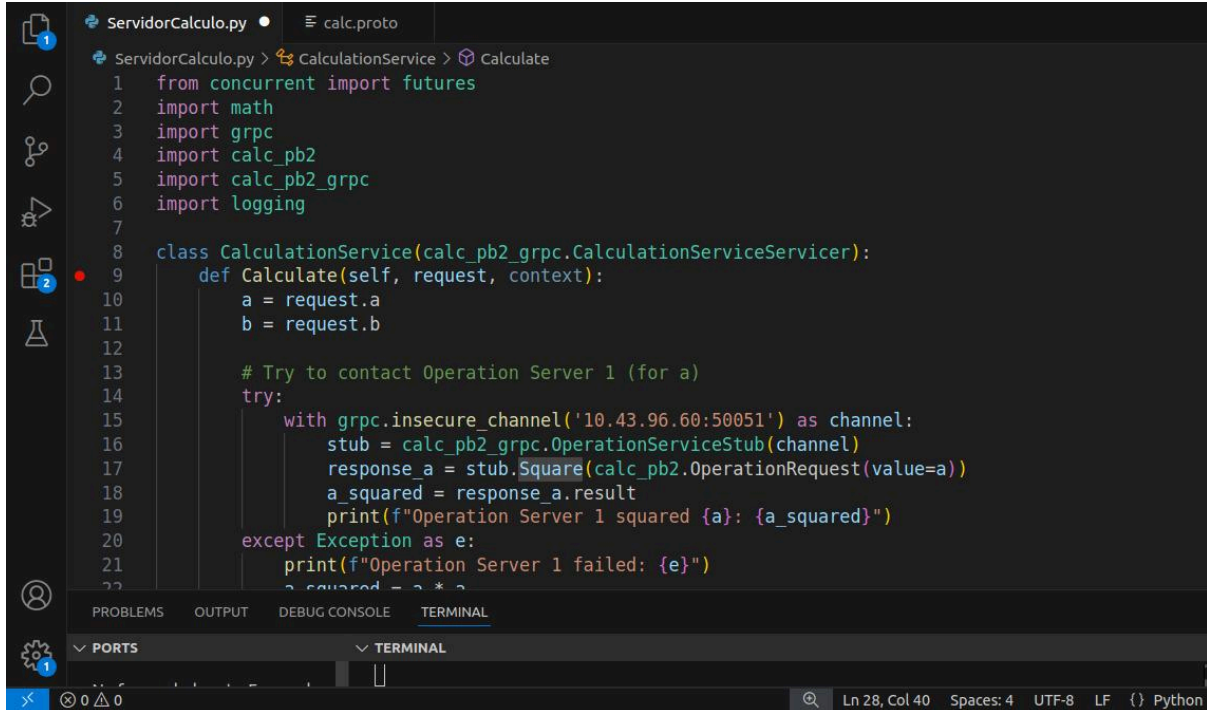
Servidor Operación 2 (10.43.103.58):



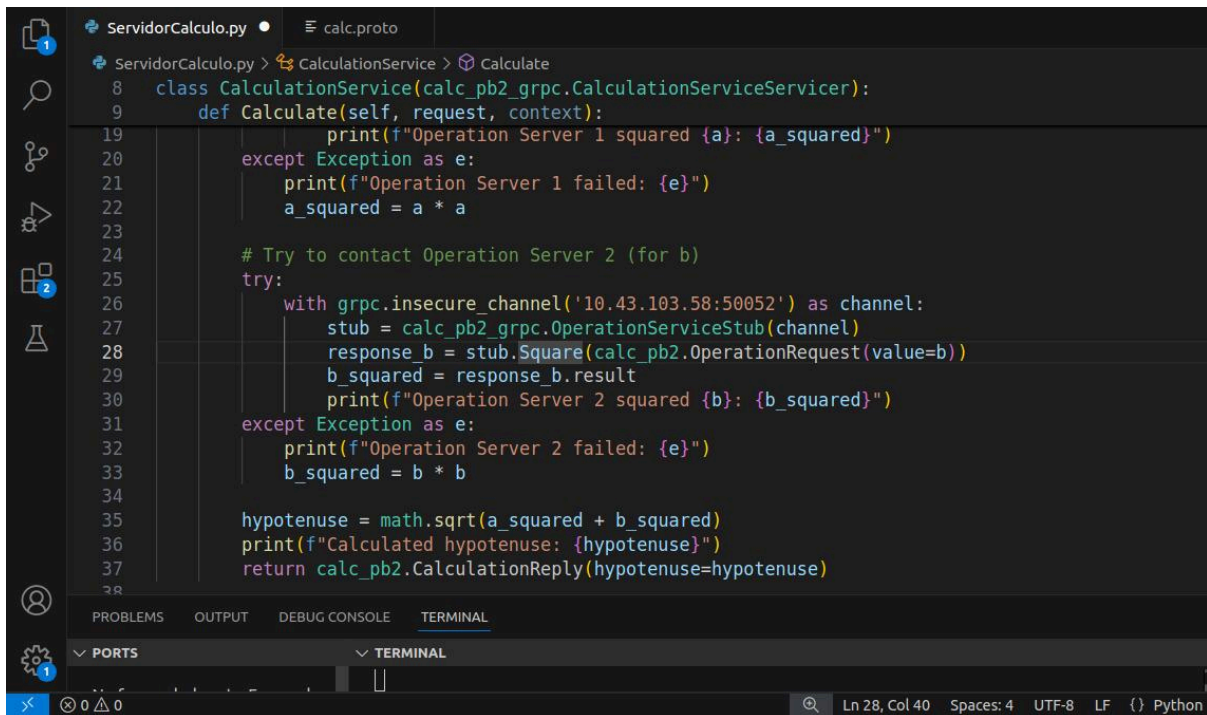
```
1 from concurrent import futures
2 import grpc
3 import calc_pb2
4 import calc_pb2_grpc
5
6 class OperationService(calc_pb2_grpc.OperationServiceServicer):
7     def Square(self, request, context):
8         value = request.value
9         result = value * value
10        print(f"[Server2] Received value: {value}, returning square: {result}")
11        return calc_pb2.OperationReply(result=result)
12
13 def serve():
14     server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
15     calc_pb2_grpc.add_OperationServiceServicer_to_server(OperationService(), server)
16     server.add_insecure_port('10.43.103.58:50052')
17     server.start()
18     print("Operation Server 2 started on port 50052")
19     server.wait_for_termination()
20
21 if __name__ == "__main__":
22     serve()
```

Figura 4. Programa del Servidor de Operación 2 con ip 10.43.103.58

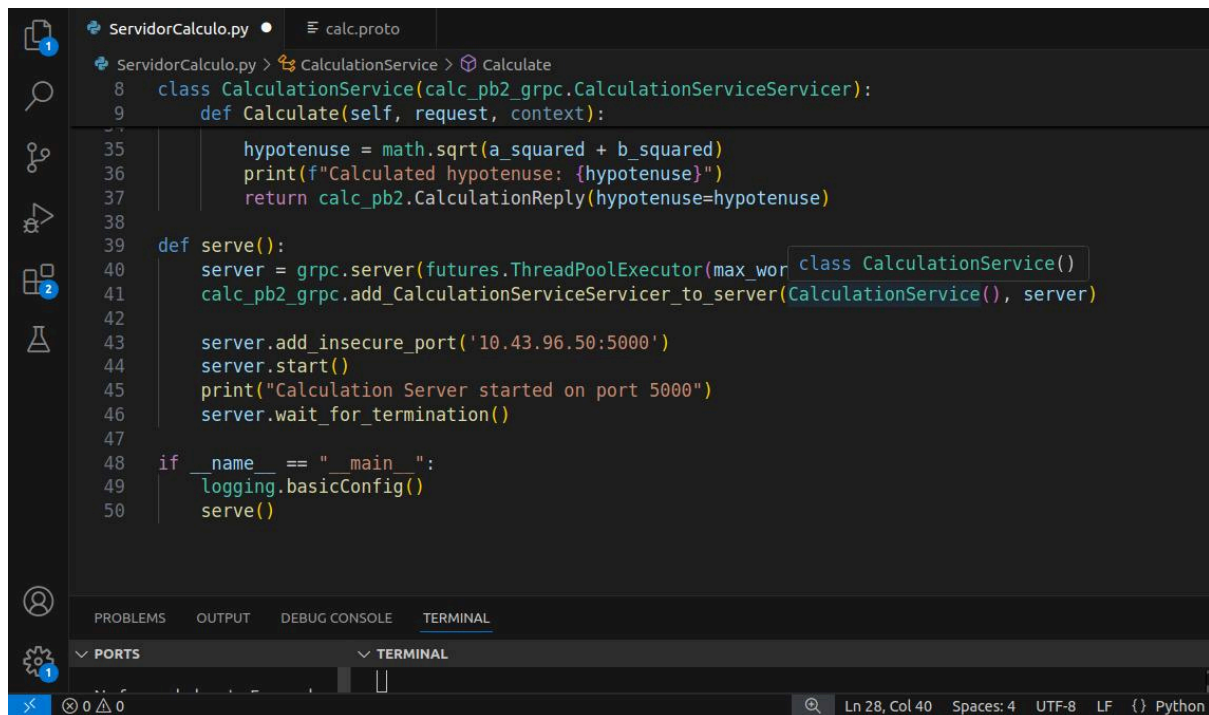
Servidor Cálculo (10.43.96.50)



```
1 from concurrent import futures
2 import math
3 import grpc
4 import calc_pb2
5 import calc_pb2_grpc
6 import logging
7
8 class CalculationService(calc_pb2_grpc.CalculationServiceServicer):
9     def Calculate(self, request, context):
10         a = request.a
11         b = request.b
12
13         # Try to contact Operation Server 1 (for a)
14         try:
15             with grpc.insecure_channel('10.43.96.60:50051') as channel:
16                 stub = calc_pb2_grpc.OperationServiceStub(channel)
17                 response_a = stub.Square(calc_pb2.OperationRequest(value=a))
18                 a_squared = response_a.result
19                 print(f"Operation Server 1 squared {a}: {a_squared}")
20         except Exception as e:
21             print(f"Operation Server 1 failed: {e}")
22             a_squared = a * a
```



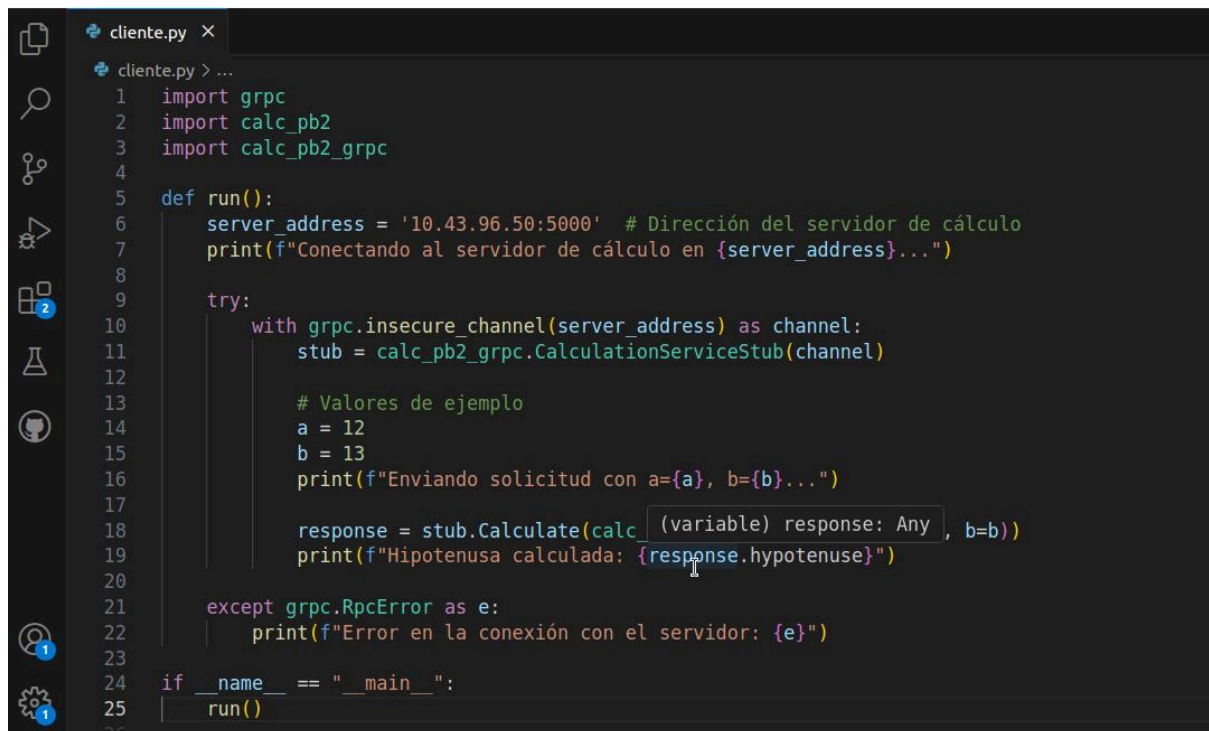
```
19         print(f"Operation Server 1 squared {a}: {a_squared}")
20     except Exception as e:
21         print(f"Operation Server 1 failed: {e}")
22         a_squared = a * a
23
24     # Try to contact Operation Server 2 (for b)
25     try:
26         with grpc.insecure_channel('10.43.103.58:50052') as channel:
27             stub = calc_pb2_grpc.OperationServiceStub(channel)
28             response_b = stub.Square(calc_pb2.OperationRequest(value=b))
29             b_squared = response_b.result
30             print(f"Operation Server 2 squared {b}: {b_squared}")
31     except Exception as e:
32         print(f"Operation Server 2 failed: {e}")
33         b_squared = b * b
34
35     hypotenuse = math.sqrt(a_squared + b_squared)
36     print(f"Calculated hypotenuse: {hypotenuse}")
37     return calc_pb2.CalculationReply(hypotenuse=hypotenuse)
```



```
8 class CalculationService(calc_pb2_grpc.CalculationServiceServicer):
9     def Calculate(self, request, context):
10
11         hypotenuse = math.sqrt(a_squared + b_squared)
12         print(f"Calculated hypotenuse: {hypotenuse}")
13         return calc_pb2.CalculationReply(hypotenuse=hypotenuse)
14
15 def serve():
16     server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
17     calc_pb2_grpc.add_CalculationServiceServicer_to_server(CalculationService(), server)
18
19     server.add_insecure_port('10.43.96.50:5000')
20     server.start()
21     print("Calculation Server started on port 5000")
22     server.wait_for_termination()
23
24 if __name__ == "__main__":
25     logging.basicConfig()
26     serve()
```

Figura 5. Programa del Servidor de Calculo con ip 10.43.96.50

Cliente (10.43.103.51):



```
1 import grpc
2 import calc_pb2
3 import calc_pb2_grpc
4
5 def run():
6     server_address = '10.43.96.50:5000' # Dirección del servidor de cálculo
7     print(f"Conectando al servidor de cálculo en {server_address}...")
8
9     try:
10         with grpc.insecure_channel(server_address) as channel:
11             stub = calc_pb2_grpc.CalculationServiceStub(channel)
12
13             # Valores de ejemplo
14             a = 12
15             b = 13
16             print(f"Enviando solicitud con a={a}, b={b}...")
17
18             response = stub.Calculate(calc_pb2.CalculationRequest(a_squared=a**2, b_squared=b**2))
19             print(f"Hipotenusa calculada: {response.hypotenuse}")
20
21     except grpc.RpcError as e:
22         print(f"Error en la conexión con el servidor: {e}")
23
24 if __name__ == "__main__":
25     run()
```

Figura 6. Programa del Cliente con ip 10.43.103.51

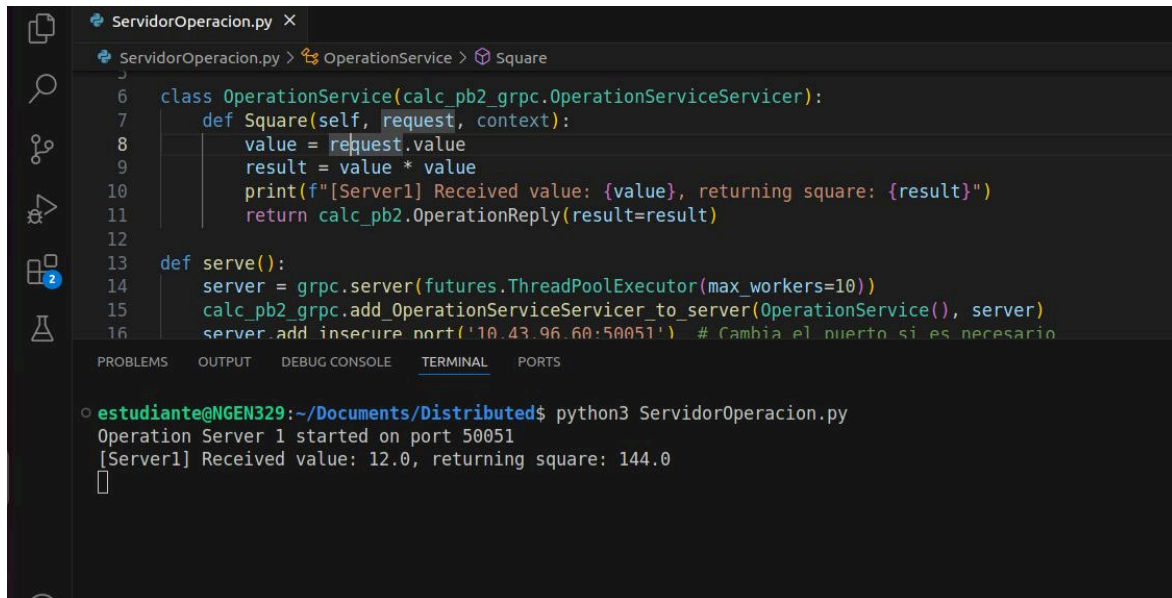
Pruebas

Caso 1: Todo funcionando correctamente (Configuración completa)

Resultado esperado:

El cliente envía a y b , los servidores de operación calculan a^2 y b^2 , y el servidor de cálculo devuelve la hipotenusa correctamente.

Salida de Servidor Operación:



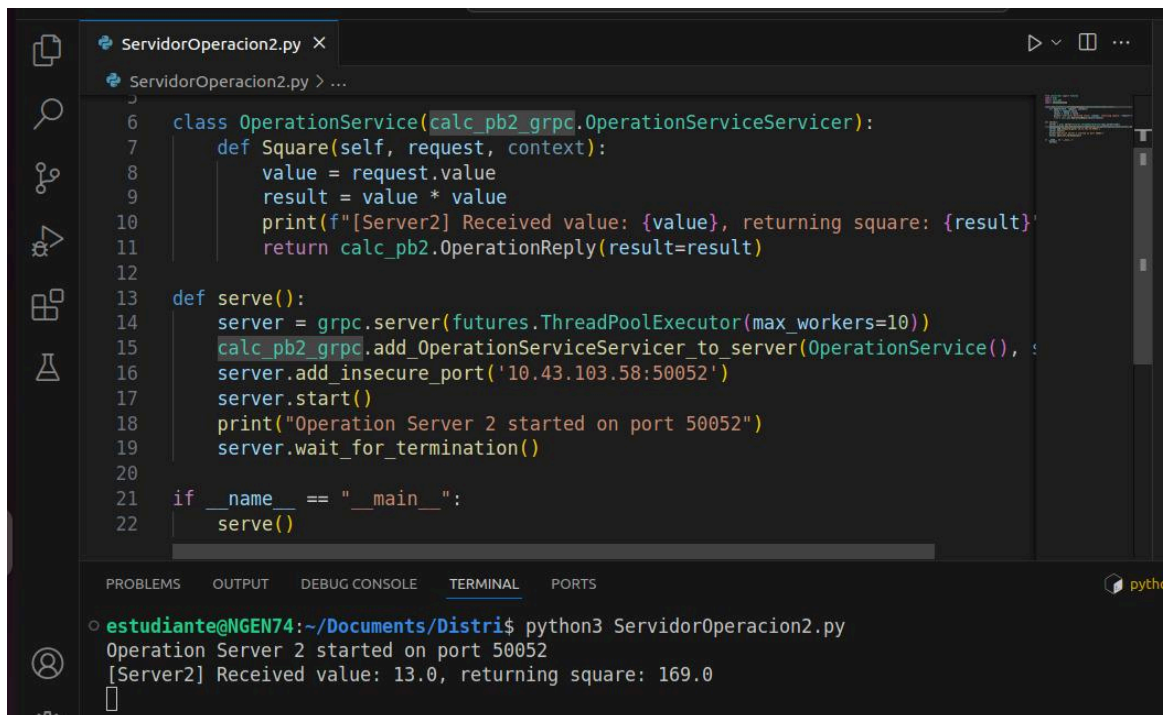
```

ServidorOperacion.py X
ServidorOperacion.py > OperationService > Square
6 class OperationService(calc_pb2_grpc.OperationServiceServicer):
7     def Square(self, request, context):
8         value = request.value
9         result = value * value
10        print(f"[Server1] Received value: {value}, returning square: {result}")
11        return calc_pb2.OperationReply(result=result)
12
13    def serve():
14        server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
15        calc_pb2_grpc.add_OperationServiceServicer_to_server(OperationService(), server)
16        server.add_insecure_port('10.43.96.60:50051') # Cambia el puerto si es necesario
17
18    if __name__ == '__main__':
19        serve()
20
21    if __name__ == '__main__':
22        serve()
23
24    if __name__ == '__main__':
25        serve()
26
27    if __name__ == '__main__':
28        serve()
29
30    if __name__ == '__main__':
31        serve()
32
33    if __name__ == '__main__':
34        serve()
35
36    if __name__ == '__main__':
37        serve()
38
39    if __name__ == '__main__':
40        serve()
41
42    if __name__ == '__main__':
43        serve()
44
45    if __name__ == '__main__':
46        serve()
47
48    if __name__ == '__main__':
49        serve()
50
51    if __name__ == '__main__':
52        serve()
53
54    if __name__ == '__main__':
55        serve()
56
57    if __name__ == '__main__':
58        serve()
59
60    if __name__ == '__main__':
61        serve()
62
63    if __name__ == '__main__':
64        serve()
65
66    if __name__ == '__main__':
67        serve()
68
69    if __name__ == '__main__':
70        serve()
71
72    if __name__ == '__main__':
73        serve()
74
75    if __name__ == '__main__':
76        serve()
77
78    if __name__ == '__main__':
79        serve()
80
81    if __name__ == '__main__':
82        serve()
83
84    if __name__ == '__main__':
85        serve()
86
87    if __name__ == '__main__':
88        serve()
89
90    if __name__ == '__main__':
91        serve()
92
93    if __name__ == '__main__':
94        serve()
95
96    if __name__ == '__main__':
97        serve()
98
99    if __name__ == '__main__':
100       serve()
101
102    if __name__ == '__main__':
103       serve()
104
105    if __name__ == '__main__':
106       serve()
107
108    if __name__ == '__main__':
109       serve()
110
111    if __name__ == '__main__':
112       serve()
113
114    if __name__ == '__main__':
115       serve()
116
117    if __name__ == '__main__':
118       serve()
119
120    if __name__ == '__main__':
121       serve()
122
123    if __name__ == '__main__':
124       serve()
125
126    if __name__ == '__main__':
127       serve()
128
129    if __name__ == '__main__':
130       serve()
131
132    if __name__ == '__main__':
133       serve()
134
135    if __name__ == '__main__':
136       serve()
137
138    if __name__ == '__main__':
139       serve()
140
141    if __name__ == '__main__':
142       serve()
143
144    if __name__ == '__main__':
145       serve()
146
147    if __name__ == '__main__':
148       serve()
149
150    if __name__ == '__main__':
151       serve()
152
153    if __name__ == '__main__':
154       serve()
155
156    if __name__ == '__main__':
157       serve()
158
159    if __name__ == '__main__':
160       serve()
161
162    if __name__ == '__main__':
163       serve()
164
165    if __name__ == '__main__':
166       serve()
167
168    if __name__ == '__main__':
169       serve()
170
171    if __name__ == '__main__':
172       serve()
173
174    if __name__ == '__main__':
175       serve()
176
177    if __name__ == '__main__':
178       serve()
179
180    if __name__ == '__main__':
181       serve()
182
183    if __name__ == '__main__':
184       serve()
185
186    if __name__ == '__main__':
187       serve()
188
189    if __name__ == '__main__':
190       serve()
191
192    if __name__ == '__main__':
193       serve()
194
195    if __name__ == '__main__':
196       serve()
197
198    if __name__ == '__main__':
199       serve()
200
201    if __name__ == '__main__':
202       serve()
203
204    if __name__ == '__main__':
205       serve()
206
207    if __name__ == '__main__':
208       serve()
209
210    if __name__ == '__main__':
211       serve()
212
213    if __name__ == '__main__':
214       serve()
215
216    if __name__ == '__main__':
217       serve()
218
219    if __name__ == '__main__':
220       serve()
221
222    if __name__ == '__main__':
223       serve()
224
225    if __name__ == '__main__':
226       serve()
227
228    if __name__ == '__main__':
229       serve()
230
231    if __name__ == '__main__':
232       serve()
233
234    if __name__ == '__main__':
235       serve()
236
237    if __name__ == '__main__':
238       serve()
239
240    if __name__ == '__main__':
241       serve()
242
243    if __name__ == '__main__':
244       serve()
245
246    if __name__ == '__main__':
247       serve()
248
249    if __name__ == '__main__':
250       serve()
251
252    if __name__ == '__main__':
253       serve()
254
255    if __name__ == '__main__':
256       serve()
257
258    if __name__ == '__main__':
259       serve()
260
261    if __name__ == '__main__':
262       serve()
263
264    if __name__ == '__main__':
265       serve()
266
267    if __name__ == '__main__':
268       serve()
269
270    if __name__ == '__main__':
271       serve()
272
273    if __name__ == '__main__':
274       serve()
275
276    if __name__ == '__main__':
277       serve()
278
279    if __name__ == '__main__':
280       serve()
281
282    if __name__ == '__main__':
283       serve()
284
285    if __name__ == '__main__':
286       serve()
287
288    if __name__ == '__main__':
289       serve()
290
291    if __name__ == '__main__':
292       serve()
293
294    if __name__ == '__main__':
295       serve()
296
297    if __name__ == '__main__':
298       serve()
299
300    if __name__ == '__main__':
301       serve()
302
303    if __name__ == '__main__':
304       serve()
305
306    if __name__ == '__main__':
307       serve()
308
309    if __name__ == '__main__':
310       serve()
311
312    if __name__ == '__main__':
313       serve()
314
315    if __name__ == '__main__':
316       serve()
317
318    if __name__ == '__main__':
319       serve()
320
321    if __name__ == '__main__':
322       serve()
323
324    if __name__ == '__main__':
325       serve()
326
327    if __name__ == '__main__':
328       serve()
329
330    if __name__ == '__main__':
331       serve()
332
333    if __name__ == '__main__':
334       serve()
335
336    if __name__ == '__main__':
337       serve()
338
339    if __name__ == '__main__':
340       serve()
341
342    if __name__ == '__main__':
343       serve()
344
345    if __name__ == '__main__':
346       serve()
347
348    if __name__ == '__main__':
349       serve()
350
351    if __name__ == '__main__':
352       serve()
353
354    if __name__ == '__main__':
355       serve()
356
357    if __name__ == '__main__':
358       serve()
359
360    if __name__ == '__main__':
361       serve()
362
363    if __name__ == '__main__':
364       serve()
365
366    if __name__ == '__main__':
367       serve()
368
369    if __name__ == '__main__':
370       serve()
371
372    if __name__ == '__main__':
373       serve()
374
375    if __name__ == '__main__':
376       serve()
377
378    if __name__ == '__main__':
379       serve()
380
381    if __name__ == '__main__':
382       serve()
383
384    if __name__ == '__main__':
385       serve()
386
387    if __name__ == '__main__':
388       serve()
389
390    if __name__ == '__main__':
391       serve()
392
393    if __name__ == '__main__':
394       serve()
395
396    if __name__ == '__main__':
397       serve()
398
399    if __name__ == '__main__':
400       serve()
401
402    if __name__ == '__main__':
403       serve()
404
405    if __name__ == '__main__':
406       serve()
407
408    if __name__ == '__main__':
409       serve()
410
411    if __name__ == '__main__':
412       serve()
413
414    if __name__ == '__main__':
415       serve()
416
417    if __name__ == '__main__':
418       serve()
419
420    if __name__ == '__main__':
421       serve()
422
423    if __name__ == '__main__':
424       serve()
425
426    if __name__ == '__main__':
427       serve()
428
429    if __name__ == '__main__':
430       serve()
431
432    if __name__ == '__main__':
433       serve()
434
435    if __name__ == '__main__':
436       serve()
437
438    if __name__ == '__main__':
439       serve()
440
441    if __name__ == '__main__':
442       serve()
443
444    if __name__ == '__main__':
445       serve()
446
447    if __name__ == '__main__':
448       serve()
449
450    if __name__ == '__main__':
451       serve()
452
453    if __name__ == '__main__':
454       serve()
455
456    if __name__ == '__main__':
457       serve()
458
459    if __name__ == '__main__':
460       serve()
461
462    if __name__ == '__main__':
463       serve()
464
465    if __name__ == '__main__':
466       serve()
467
468    if __name__ == '__main__':
469       serve()
470
471    if __name__ == '__main__':
472       serve()
473
474    if __name__ == '__main__':
475       serve()
476
477    if __name__ == '__main__':
478       serve()
479
480    if __name__ == '__main__':
481       serve()
482
483    if __name__ == '__main__':
484       serve()
485
486    if __name__ == '__main__':
487       serve()
488
489    if __name__ == '__main__':
490       serve()
491
492    if __name__ == '__main__':
493       serve()
494
495    if __name__ == '__main__':
496       serve()
497
498    if __name__ == '__main__':
499       serve()
500
501    if __name__ == '__main__':
502       serve()
503
504    if __name__ == '__main__':
505       serve()
506
507    if __name__ == '__main__':
508       serve()
509
510    if __name__ == '__main__':
511       serve()
512
513    if __name__ == '__main__':
514       serve()
515
516    if __name__ == '__main__':
517       serve()
518
519    if __name__ == '__main__':
520       serve()
521
522    if __name__ == '__main__':
523       serve()
524
525    if __name__ == '__main__':
526       serve()
527
528    if __name__ == '__main__':
529       serve()
530
531    if __name__ == '__main__':
532       serve()
533
534    if __name__ == '__main__':
535       serve()
536
537    if __name__ == '__main__':
538       serve()
539
540    if __name__ == '__main__':
541       serve()
542
543    if __name__ == '__main__':
544       serve()
545
546    if __name__ == '__main__':
547       serve()
548
549    if __name__ == '__main__':
550       serve()
551
552    if __name__ == '__main__':
553       serve()
554
555    if __name__ == '__main__':
556       serve()
557
558    if __name__ == '__main__':
559       serve()
560
561    if __name__ == '__main__':
562       serve()
563
564    if __name__ == '__main__':
565       serve()
566
567    if __name__ == '__main__':
568       serve()
569
570    if __name__ == '__main__':
571       serve()
572
573    if __name__ == '__main__':
574       serve()
575
576    if __name__ == '__main__':
577       serve()
578
579    if __name__ == '__main__':
580       serve()
581
582    if __name__ == '__main__':
583       serve()
584
585    if __name__ == '__main__':
586       serve()
587
588    if __name__ == '__main__':
589       serve()
590
591    if __name__ == '__main__':
592       serve()
593
594    if __name__ == '__main__':
595       serve()
596
597    if __name__ == '__main__':
598       serve()
599
600    if __name__ == '__main__':
601       serve()
602
603    if __name__ == '__main__':
604       serve()
605
606    if __name__ == '__main__':
607       serve()
608
609    if __name__ == '__main__':
610       serve()
611
612    if __name__ == '__main__':
613       serve()
614
615    if __name__ == '__main__':
616       serve()
617
618    if __name__ == '__main__':
619       serve()
620
621    if __name__ == '__main__':
622       serve()
623
624    if __name__ == '__main__':
625       serve()
626
627    if __name__ == '__main__':
628       serve()
629
630    if __name__ == '__main__':
631       serve()
632
633    if __name__ == '__main__':
634       serve()
635
636    if __name__ == '__main__':
637       serve()
638
639    if __name__ == '__main__':
640       serve()
641
642    if __name__ == '__main__':
643       serve()
644
645    if __name__ == '__main__':
646       serve()
647
648    if __name__ == '__main__':
649       serve()
650
651    if __name__ == '__main__':
652       serve()
653
654    if __name__ == '__main__':
655       serve()
656
657    if __name__ == '__main__':
658       serve()
659
660    if __name__ == '__main__':
661       serve()
662
663    if __name__ == '__main__':
664       serve()
665
666    if __name__ == '__main__':
667       serve()
668
669    if __name__ == '__main__':
670       serve()
671
672    if __name__ == '__main__':
673       serve()
674
675    if __name__ == '__main__':
676       serve()
677
678    if __name__ == '__main__':
679       serve()
680
681    if __name__ == '__main__':
682       serve()
683
684    if __name__ == '__main__':
685       serve()
686
687    if __name__ == '__main__':
688       serve()
689
690    if __name__ == '__main__':
691       serve()
692
693    if __name__ == '__main__':
694       serve()
695
696    if __name__ == '__main__':
697       serve()
698
699    if __name__ == '__main__':
700       serve()
701
702    if __name__ == '__main__':
703       serve()
704
705    if __name__ == '__main__':
706       serve()
707
708    if __name__ == '__main__':
709       serve()
710
711    if __name__ == '__main__':
712       serve()
713
714    if __name__ == '__main__':
715       serve()
716
717    if __name__ == '__main__':
718       serve()
719
720    if __name__ == '__main__':
721       serve()
722
723    if __name__ == '__main__':
724       serve()
725
726    if __name__ == '__main__':
727       serve()
728
729    if __name__ == '__main__':
730       serve()
731
732    if __name__ == '__main__':
733       serve()
734
735    if __name__ == '__main__':
736       serve()
737
738    if __name__ == '__main__':
739       serve()
740
741    if __name__ == '__main__':
742       serve()
743
744    if __name__ == '__main__':
745       serve()
746
747    if __name__ == '__main__':
748       serve()
749
750    if __name__ == '__main__':
751       serve()
752
753    if __name__ == '__main__':
754       serve()
755
756    if __name__ == '__main__':
757       serve()
758
759    if __name__ == '__main__':
760       serve()
761
762    if __name__ == '__main__':
763       serve()
764
765    if __name__ == '__main__':
766       serve()
767
768    if __name__ == '__main__':
769       serve()
770
771    if __name__ == '__main__':
772       serve()
773
774    if __name__ == '__main__':
775       serve()
776
777    if __name__ == '__main__':
778       serve()
779
780    if __name__ == '__main__':
781       serve()
782
783    if __name__ == '__main__':
784       serve()
785
786    if __name__ == '__main__':
787       serve()
788
789    if __name__ == '__main__':
790       serve()
791
792    if __name__ == '__main__':
793       serve()
794
795    if __name__ == '__main__':
796       serve()
797
798    if __name__ == '__main__':
799       serve()
800
801    if __name__ == '__main__':
802       serve()
803
804    if __name__ == '__main__':
805       serve()
806
807    if __name__ == '__main__':
808       serve()
809
810    if __name__ == '__main__':
811       serve()
812
813    if __name__ == '__main__':
814       serve()
815
816    if __name__ == '__main__':
817       serve()
818
819    if __name__ == '__main__':
820       serve()
821
822    if __name__ == '__main__':
823       serve()
824
825    if __name__ == '__main__':
826       serve()
827
828    if __name__ == '__main__':
829       serve()
830
831    if __name__ == '__main__':
832       serve()
833
834    if __name__ == '__main__':
835       serve()
836
837    if __name__ == '__main__':
838       serve()
839
840    if __name__ == '__main__':
841       serve()
842
843    if __name__ == '__main__':
844       serve()
845
846    if __name__ == '__main__':
847       serve()
848
849    if __name__ == '__main__':
850       serve()
851
852    if __name__ == '__main__':
853       serve()
854
855    if __name__ == '__main__':
856       serve()
857
858    if __name__ == '__main__':
859       serve()
860
861    if __name__ == '__main__':
862       serve()
863
864    if __name__ == '__main__':
865       serve()
866
867    if __name__ == '__main__':
868       serve()
869
870    if __name__ == '__main__':
871       serve()
872
873    if __name__ == '__main__':
874       serve()
875
876    if __name__ == '__main__':
877       serve()
878
879    if __name__ == '__main__':
880       serve()
881
882    if __name__ == '__main__':
883       serve()
884
885    if __name__ == '__main__':
886       serve()
887
888    if __name__ == '__main__':
889       serve()
890
891    if __name__ == '__main__':
892       serve()
893
894    if __name__ == '__main__':
895       serve()
896
897    if __name__ == '__main__':
898       serve()
899
900    if __name__ == '__main__':
901       serve()
902
903    if __name__ == '__main__':
904       serve()
905
906    if __name__ == '__main__':
907       serve()
908
909    if __name__ == '__main__':
910       serve()
911
912    if __name__ == '__main__':
913       serve()
914
915    if __name__ == '__main__':
916       serve()
917
918    if __name__ == '__main__':
919       serve()
920
921    if __name__ == '__main__':
922       serve()
923
924    if __name__ == '__main__':
925       serve()
926
927    if __name__ == '__main__':
928       serve()
929
930    if __name__ == '__main__':
931       serve()
932
933    if __name__ == '__main__':
934       serve()
935
936    if __name__ == '__main__':
937       serve()
938
939    if __name__ == '__main__':
940       serve()
941
942    if __name__ == '__main__':
943       serve()
944
945    if __name__ == '__main__':
946       serve()
947
948    if __name__ == '__main__':
949       serve()
950
951    if __name__ == '__main__':
952       serve()
953
954    if __name__ == '__main__':
955       serve()
956
957    if __name__ == '__main__':
958       serve()
959
960    if __name__ == '__main__':
961       serve()
962
963    if __name__ == '__main__':
964       serve()
965
966    if __name__ == '__main__':
967       serve()
968
969    if __name__ == '__main__':
970       serve()
971
972    if __name__ == '__main__':
973       serve()
974
975    if __name__ == '__main__':
976       serve()
977
978    if __name__ == '__main__':
979       serve()
980
981    if __name__ == '__main__':
982       serve()
983
984    if __name__ == '__main__':
985       serve()
986
987    if __name__ == '__main__':
988       serve()
989
990    if __name__ == '__main__':
991       serve()
992
993    if __name__ == '__main__':
994       serve()
995
996    if __name__ == '__main__':
997       serve()
998
999    if __name__ == '__main__':
1000      serve()

```

Figura 7. Ejecución del Servidor de Operación 1

Salida de Servidor Operación 2:



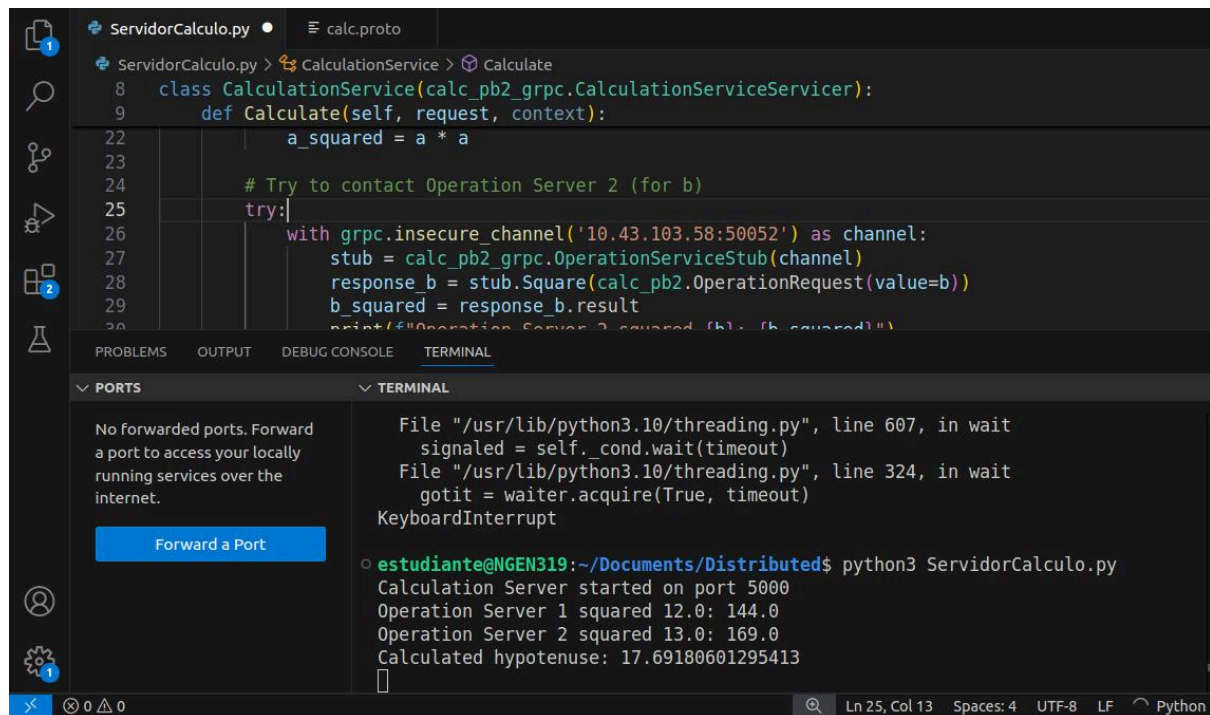
```

ServidorOperacion2.py X
ServidorOperacion2.py > ...
6 class OperationService(calc_pb2_grpc.OperationServiceServicer):
7     def Square(self, request, context):
8         value = request.value
9         result = value * value
10        print(f"[Server2] Received value: {value}, returning square: {result}")
11        return calc_pb2.OperationReply(result=result)
12
13    def serve():
14        server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
15        calc_pb2_grpc.add_OperationServiceServicer_to_server(OperationService(), server)
16        server.add_insecure_port('10.43.103.58:50052')
17        server.start()
18        print("Operation Server 2 started on port 50052")
19        server.wait_for_termination()
20
21    if __name__ == '__main__':
22        serve()
23
24    if __name__ == '__main__':
25        serve()
26
27    if __name__ == '__main__':
28        serve()
29
30    if __name__ == '__main__':
31        serve()
32
33    if __name__ == '__main__':
34        serve()
35
36    if __name__ == '__main__':
37        serve()
38
39    if __name__ == '__main__':
40        serve()
41
42    if __name__ == '__main__':
43        serve()
44
45    if __name__ == '__main__':
46        serve()
47
48    if __name__ == '__main__':
49        serve()
49
50    if __name__ == '__main__':
51        serve()
52
53    if __name__ == '__main__':
54        serve()
55
56    if __name__ == '__main__':
57        serve()
58
59    if __name__ == '__main__':
60        serve()
61
62    if __name__ == '__main__':
63        serve()
64
65    if __name__ == '__main__':
66        serve()
67
68    if __name__ == '__main__':
69        serve()
69
70    if __name__ == '__main__':
71        serve()
72
73    if __name__ == '__main__':
74        serve()
75
76    if __name__ == '__main__':
77        serve()
78
79    if __name__ == '__main__':
80        serve()
81
82    if __name__ == '__main__':
83        serve()
84
85    if __name__ == '__main__':
86        serve()
87
88    if __name__ == '__main__':
89        serve()
89
90    if __name__ == '__main__':
91        serve()
92
93    if __name__ == '__main__':
94        serve()
95
96    if __name__ == '__main__':
97        serve()
98
99    if __name__ == '__main__':
100       serve()

```

Figura 8. Ejecución del Servidor de Operación 2

Salida de Servidor Cálculo:



```
ServerCalc.py • calc.proto
ServerCalc.py > CalculationService > Calculate
8 class CalculationService(calc_pb2_grpc.CalculationServiceServicer):
9     def Calculate(self, request, context):
22         a_squared = a * a
23
24         # Try to contact Operation Server 2 (for b)
25         try:
26             with grpc.insecure_channel('10.43.103.58:50052') as channel:
27                 stub = calc_pb2_grpc.OperationServiceStub(channel)
28                 response_b = stub.Square(calc_pb2.OperationRequest(value=b))
29                 b_squared = response_b.result
30                 print(f"Operation Server 2 squared {b}: {b_squared}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PORTS

No forwarded ports. Forward a port to access your locally running services over the internet.

Forward a Port

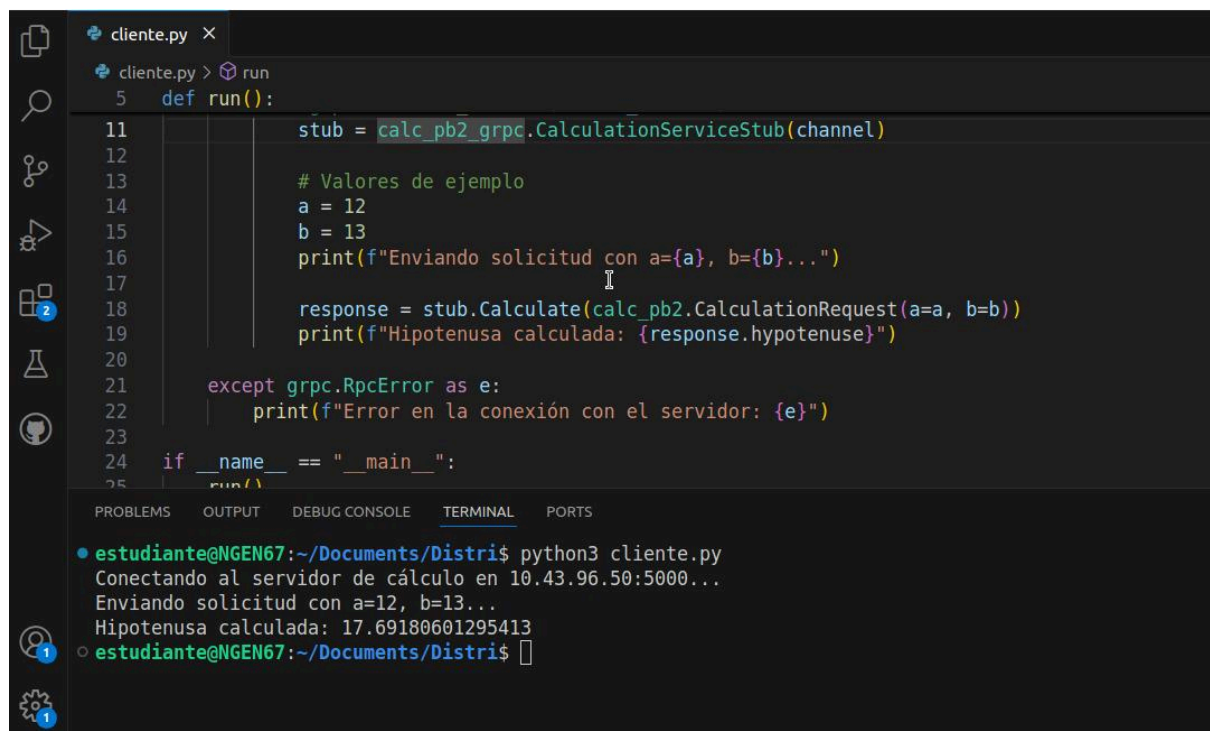
TERMINAL

```
File "/usr/lib/python3.10/threading.py", line 607, in wait
    signaled = self._cond.wait(timeout)
File "/usr/lib/python3.10/threading.py", line 324, in wait
    gotit = waiter.acquire(True, timeout)
KeyboardInterrupt

estudiante@NGEN319:~/Documents/Distributed$ python3 ServidorCalculo.py
Calculation Server started on port 5000
Operation Server 1 squared 12.0: 144.0
Operation Server 2 squared 13.0: 169.0
Calculated hypotenuse: 17.69180601295413
```

Figura 9. Ejecución del Servidor de Cálculo

Salida de cliente:



```
cliente.py x
cliente.py > run
5 def run():
11     stub = calc_pb2_grpc.CalculationServiceStub(channel)
12
13     # Valores de ejemplo
14     a = 12
15     b = 13
16     print(f"Enviando solicitud con a={a}, b={b}...")
17
18     response = stub.Calculate(calc_pb2.CalculationRequest(a=a, b=b))
19     print(f"Hipotenusa calculada: {response.hypotenuse}")
20
21 except grpc.RpcError as e:
22     print(f"Error en la conexión con el servidor: {e}")
23
24 if __name__ == "__main__":
25     run()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
estudiante@NGEN67:~/Documents/Distri$ python3 cliente.py
Conectando al servidor de cálculo en 10.43.96.50:5000...
Enviando solicitud con a=12, b=13...
Hipotenusa calculada: 17.69180601295413
estudiante@NGEN67:~/Documents/Distri$
```

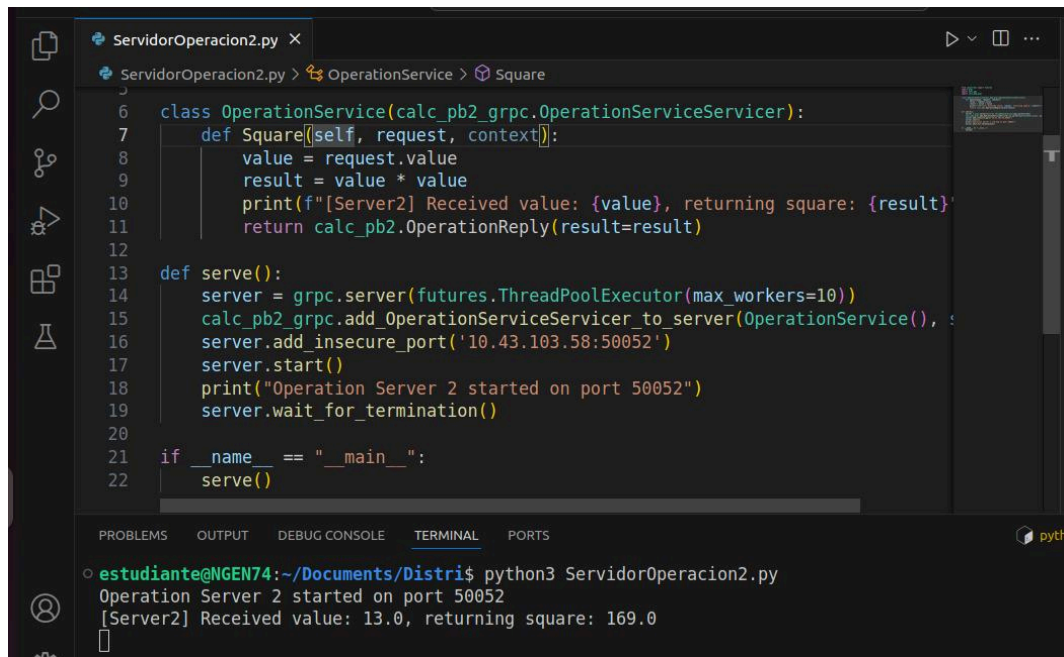
Figura 10. Ejecución del Cliente

Caso 2: Servidor Operación apagado.

Resultado esperado:

Error en el Servidor Cálculo, pero el cálculo sigue funcionando, porque en Servidor Cálculo se usa a * a en lugar de consultar a Servidor Operación

Salida de Servidor Operación 2



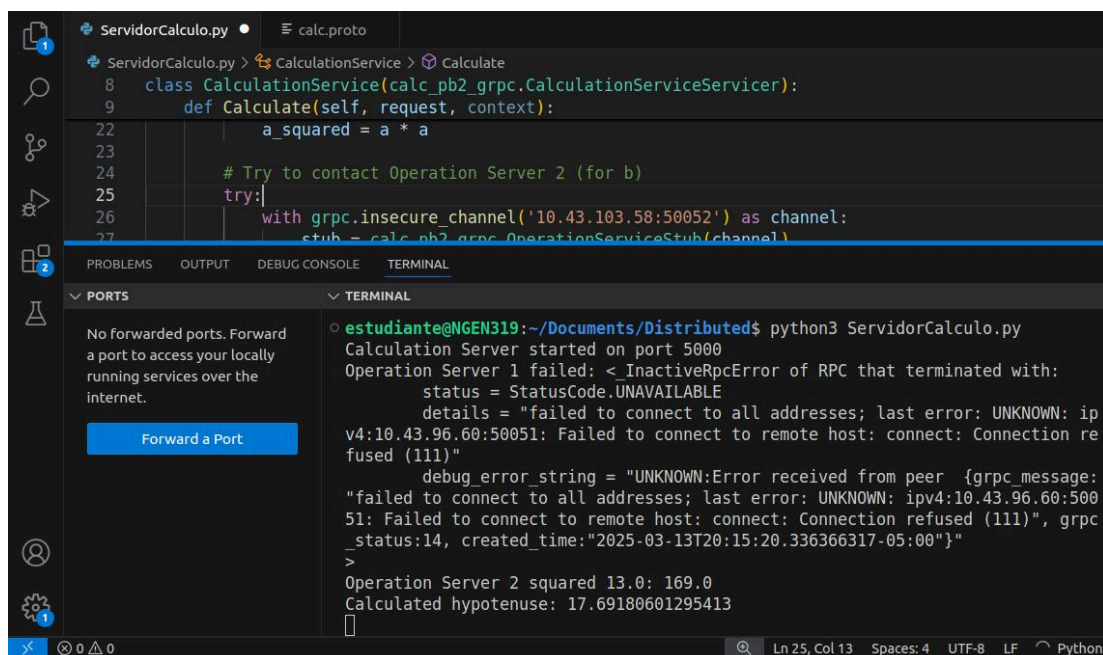
```
6 class OperationService(calc_pb2_grpc.OperationServiceServicer):
7     def Square(self, request, context):
8         value = request.value
9         result = value * value
10        print(f"[Server2] Received value: {value}, returning square: {result}")
11        return calc_pb2.OperationReply(result=result)
12
13    def serve():
14        server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
15        calc_pb2_grpc.add_OperationServiceServicer_to_server(OperationService(), server)
16        server.add_insecure_port('10.43.103.58:50052')
17        server.start()
18        print("Operation Server 2 started on port 50052")
19        server.wait_for_termination()
20
21    if __name__ == "__main__":
22        serve()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
estudiante@NGEN74:~/Documents/Distri$ python3 ServidorOperacion2.py
Operation Server 2 started on port 50052
[Server2] Received value: 13.0, returning square: 169.0
```

Figura 11. Ejecución del Servidor de Operación 2

Salida de Servidor Cálculo:



```
8 class CalculationService(calc_pb2_grpc.CalculationServiceServicer):
9     def Calculate(self, request, context):
10        a_squared = a * a
11
12        # Try to contact Operation Server 2 (for b)
13        try:
14            with grpc.insecure_channel('10.43.103.58:50052') as channel:
15                stub = calc_pb2_grpc.OperationServiceStub(channel)
16                b_squared = stub.Square(request=request).result
17        except:
18            pass
19
20    def serve():
21        server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
22        calc_pb2_grpc.add_CalculationServiceServicer_to_server(CalculationService(), server)
23        server.add_insecure_port('10.43.103.58:5000')
24        server.start()
25        print("Calculation Server 1 started on port 5000")
26        server.wait_for_termination()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

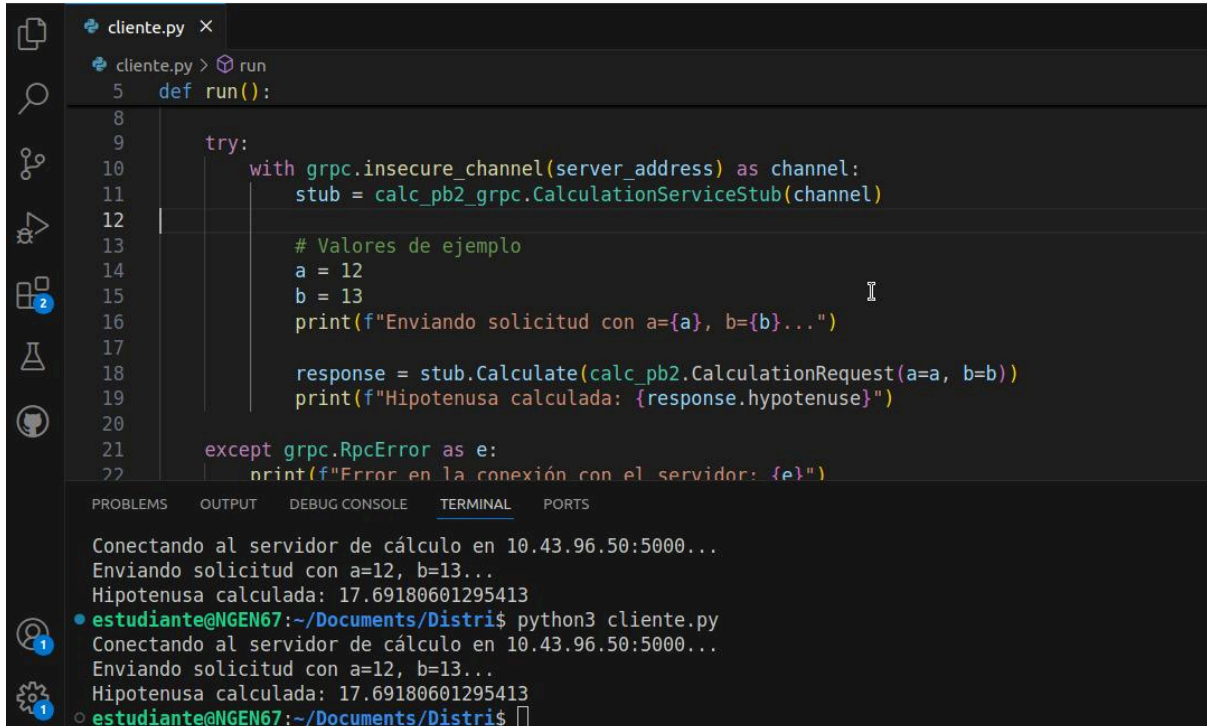
No forwarded ports. Forward a port to access your locally running services over the internet.

Forward a Port

```
estudiante@NGEN319:~/Documents/Distributed$ python3 ServidorCalculo.py
Calculation Server 1 started on port 5000
Operation Server 1 failed: < InactiveRpcError of RPC that terminated with:
  status = StatusCode.UNAVAILABLE
  details = "failed to connect to all addresses; last error: UNKNOWN: ip
v4:10.43.96.60:50051: Failed to connect to remote host: connect: Connection re
fused (111)"
  debug_error_string = "UNKNOWN:Error received from peer {grpc_message:
"failed to connect to all addresses; last error: UNKNOWN: ipv4:10.43.96.60:500
51: Failed to connect to remote host: connect: Connection refused (111)", grpc
_status:14, created_time:"2025-03-13T20:15:20.336366317-05:00"}"
>
Operation Server 2 squared 13.0: 169.0
Calculated hypotenuse: 17.69180601295413
```

Figura 12. Ejecución del Servidor de Cálculo

Salida de Cliente:



```
cliente.py X
cliente.py > run
5 def run():
8
9     try:
10         with grpc.insecure_channel(server_address) as channel:
11             stub = calc_pb2_grpc.CalculationServiceStub(channel)
12
13             # Valores de ejemplo
14             a = 12
15             b = 13
16             print(f"Enviando solicitud con a={a}, b={b}...")
17
18             response = stub.Calculate(calc_pb2.CalculationRequest(a=a, b=b))
19             print(f"Hipotenusa calculada: {response.hypotenuse}")
20
21         except grpc.RpcError as e:
22             print(f"Error en la conexión con el servidor: {e}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Conectando al servidor de cálculo en 10.43.96.50:5000...
Enviando solicitud con a=12, b=13...
Hipotenusa calculada: 17.69180601295413
● estudiante@NGEN67:~/Documents/Distri$ python3 cliente.py
Conectando al servidor de cálculo en 10.43.96.50:5000...
Enviando solicitud con a=12, b=13...
Hipotenusa calculada: 17.69180601295413
○ estudiante@NGEN67:~/Documents/Distri$
```

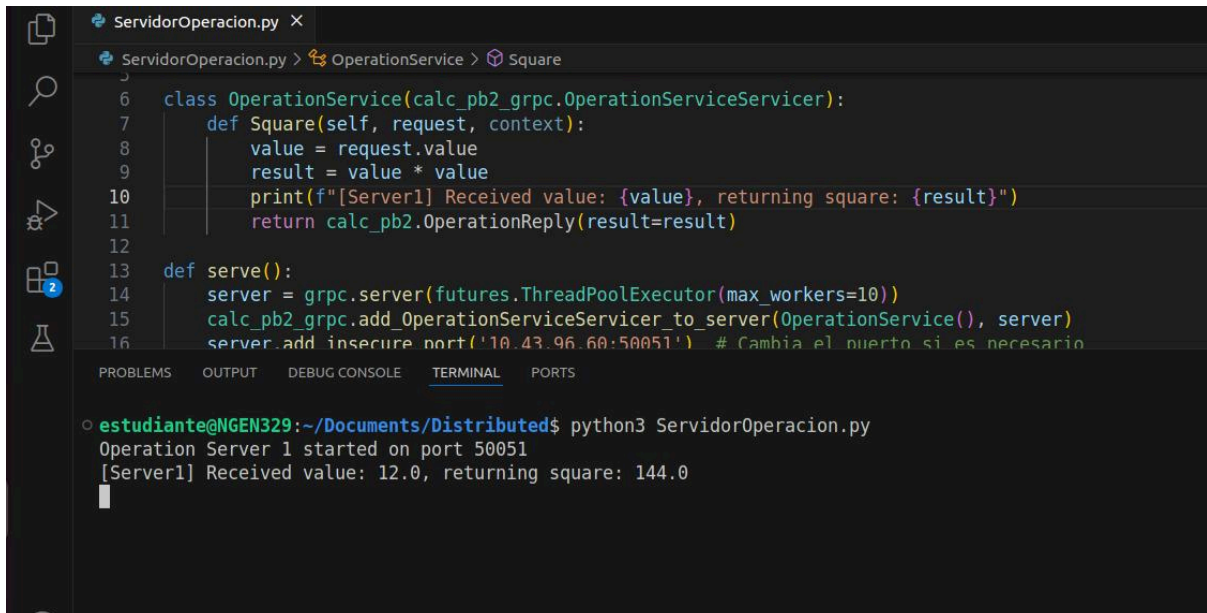
Figura 13. Ejecución del Cliente

Caso 3: Servidor Operación 2 apagado.

Resultado esperado:

Error en el Servidor Cálculo, pero el cálculo sigue funcionando, porque en Servidor Cálculo se usa $b * b$ en lugar de consultar a Servidor Operación 2

Salida de Servidor Operación:



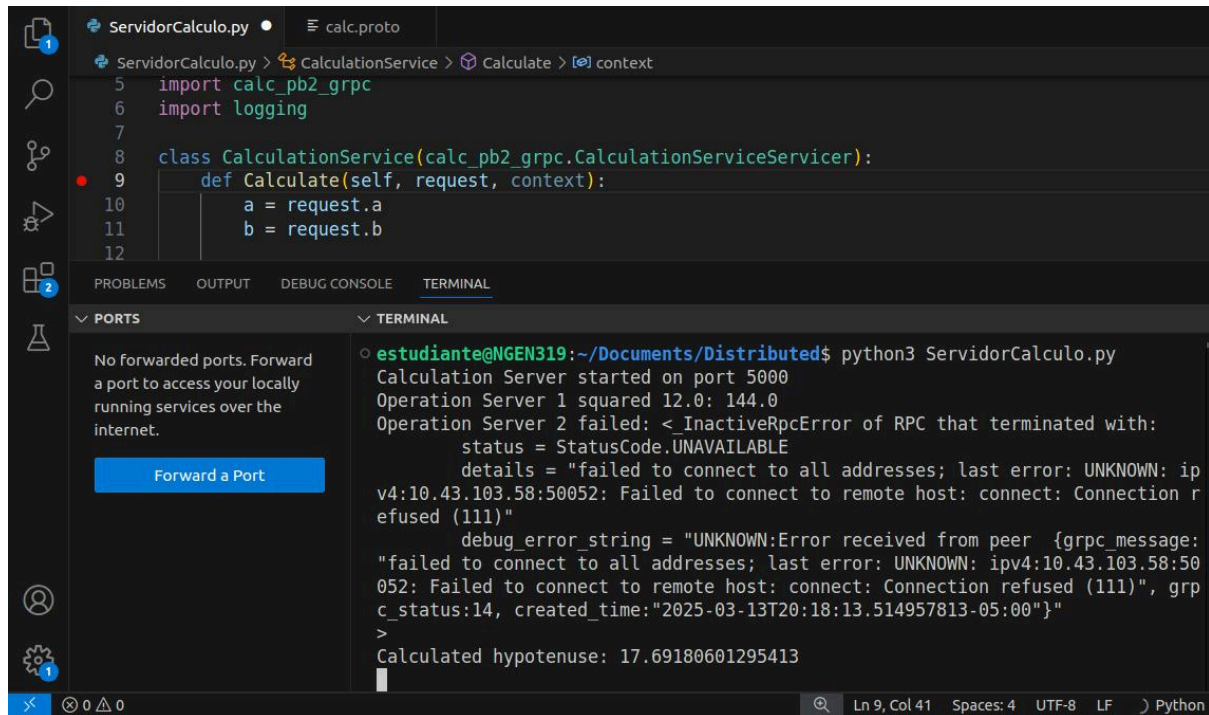
```
ServidorOperacion.py X
ServidorOperacion.py > OperationService > Square
5
6 class OperationService(calc_pb2_grpc.OperationServiceServicer):
7     def Square(self, request, context):
8         value = request.value
9         result = value * value
10        print(f"[Server1] Received value: {value}, returning square: {result}")
11        return calc_pb2.OperationReply(result=result)
12
13    def serve():
14        server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
15        calc_pb2_grpc.add_OperationServiceServicer_to_server(OperationService(), server)
16        server.add_insecure_port('10.43.96.60:50051') # Cambia el puerto si es necesario

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

○ estudiante@NGEN329:~/Documents/Distributed$ python3 ServidorOperacion.py
Operation Server 1 started on port 50051
[Server1] Received value: 12.0, returning square: 144.0
```

Figura 14. Ejecución del Servidor de Operación

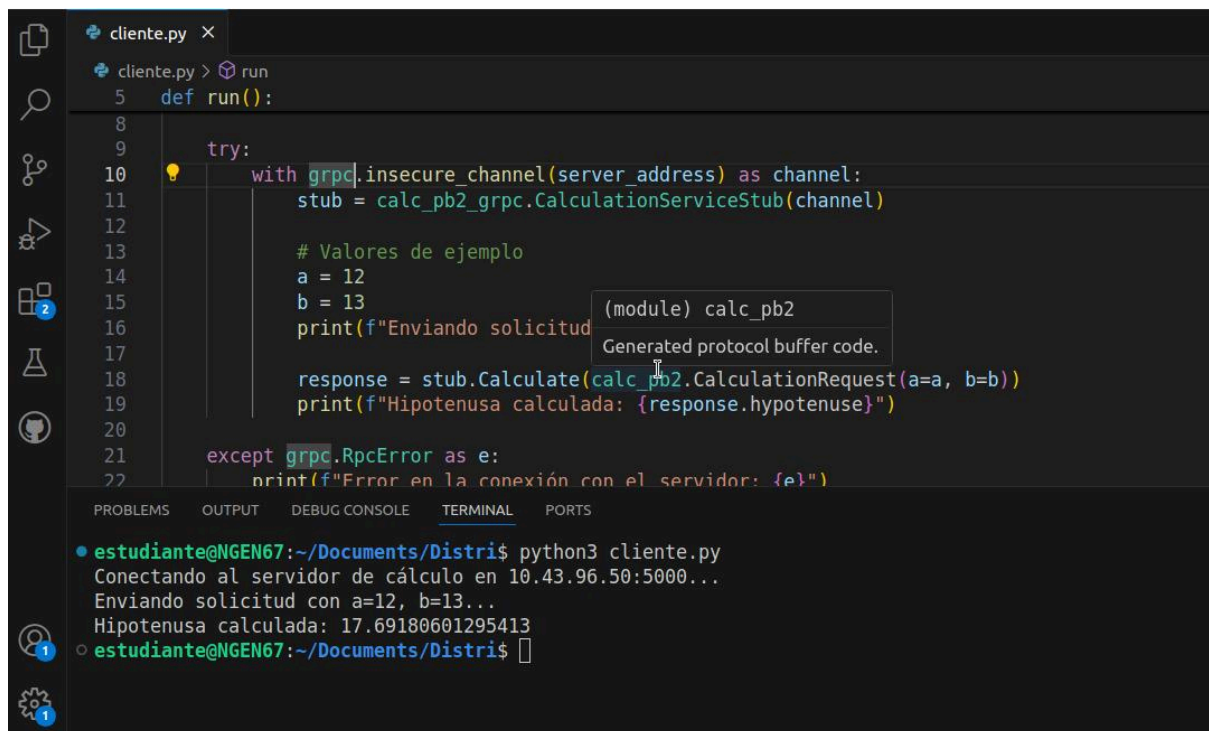
Salida de Servidor Cálculo:



```
ServerCalculo.py • calc.proto
ServerCalculo.py > CalculationService > Calculate > context
5 import calc_pb2_grpc
6 import logging
7
8 class CalculationService(calc_pb2_grpc.CalculationServiceServicer):
9     def Calculate(self, request, context):
10         a = request.a
11         b = request.b
12
13 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
14 No forwarded ports. Forward a port to access your locally running services over the internet.
15 Forward a Port
16
17 estudiante@NGEN319:~/Documents/Distributed$ python3 ServidorCalculo.py
18 Calculation Server started on port 5000
19 Operation Server 1 squared 12.0: 144.0
20 Operation Server 2 failed: <InactiveRpcError of RPC that terminated with:
21     status = StatusCode.UNAVAILABLE
22     details = "failed to connect to all addresses; last error: UNKNOWN: ip
23 v4:10.43.103.58:50052: Failed to connect to remote host: connect: Connection r
24 eferred (111)"
25     debug_error_string = "UNKNOWN:Error received from peer {grpc_message:
26 "failed to connect to all addresses; last error: UNKNOWN: ipv4:10.43.103.58:50
27 052: Failed to connect to remote host: connect: Connection refused (111)", gr
28 p_c_status:14, created_time:"2025-03-13T20:18:13.514957813-05:00"}"
29 >
30 Calculated hypotenuse: 17.69180601295413
```

Figura 15. Ejecución del Servidor de Cálculo

Salida de Cliente:



```
cliente.py X
cliente.py > run
5 def run():
6
7
8
9     try:
10         with grpc.insecure_channel(server_address) as channel:
11             stub = calc_pb2_grpc.CalculationServiceStub(channel)
12
13             # Valores de ejemplo
14             a = 12
15             b = 13
16             print(f"Enviando solicitud")
17
18             response = stub.Calculate(calc_pb2.CalculationRequest(a=a, b=b))
19             print(f"Hipotenusa calculada: {response.hypotenuse}")
20
21     except grpc.RpcError as e:
22         print(f"Error en la conexión con el servidor: {e}")
23
24 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
25 estudiante@NGEN67:~/Documents/Distri$ python3 cliente.py
26 Conectando al servidor de cálculo en 10.43.96.50:5000...
27 Enviando solicitud con a=12, b=13...
28 Hipotenusa calculada: 17.69180601295413
29 estudiante@NGEN67:~/Documents/Distri$
```

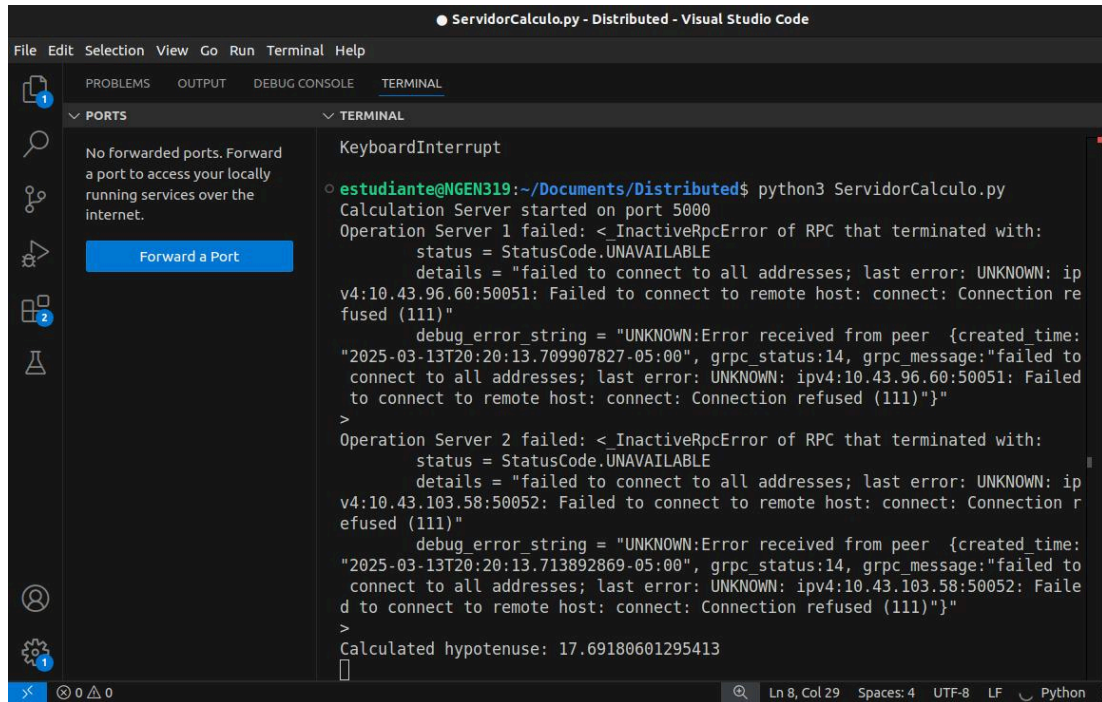
Figura 16. Ejecución del Cliente

Caso 4: Ambos servidores de operación están apagados.

Resultado esperado:

Error en el Servidor Cálculo, pero el cálculo sigue funcionando, porque en Servidor Cálculo se usa a * a + b * directamente sin llamar a los servidores de operación.

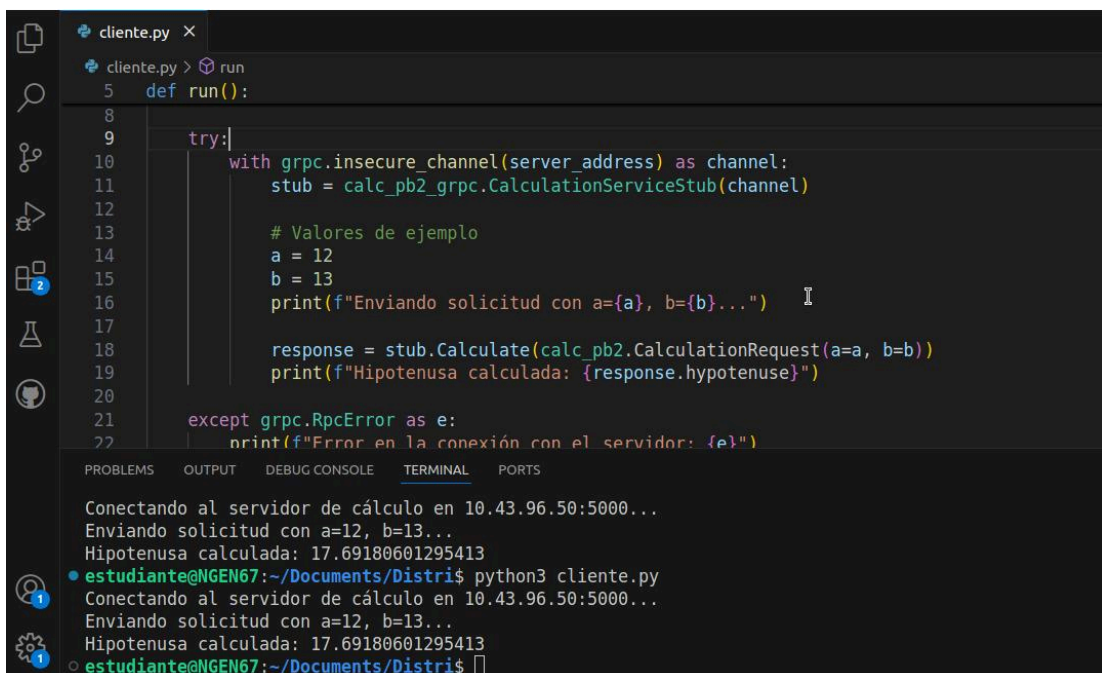
Salida de Servidor Cálculo:



```
● ServidorCalculo.py - Distributed - Visual Studio Code
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
No forwarded ports. Forward a port to access your locally running services over the internet.
Forward a Port
KeyboardInterrupt
○ estudiante@NGEN319:~/Documents/Distributed$ python3 ServidorCalculo.py
Calculation Server started on port 5000
Operation Server 1 failed: <InactiveRpcError of RPC that terminated with:
  status = StatusCode.UNAVAILABLE
  details = "failed to connect to all addresses; last error: UNKNOWN: ip
v4:10.43.96.60:50051: Failed to connect to remote host: connect: Connection re
fused (111)"
  debug_error_string = "UNKNOWN:Error received from peer {created time:
"2025-03-13T20:20:13.709907827-05:00", grpc_status:14, grpc_message:"failed to
connect to all addresses; last error: UNKNOWN: ipv4:10.43.96.60:50051: Failed
to connect to remote host: connect: Connection refused (111)}"
>
Operation Server 2 failed: <InactiveRpcError of RPC that terminated with:
  status = StatusCode.UNAVAILABLE
  details = "failed to connect to all addresses; last error: UNKNOWN: ip
v4:10.43.103.58:50052: Failed to connect to remote host: connect: Connection r
efused (111)"
  debug_error_string = "UNKNOWN:Error received from peer {created time:
"2025-03-13T20:20:13.713892869-05:00", grpc_status:14, grpc_message:"failed to
connect to all addresses; last error: UNKNOWN: ipv4:10.43.103.58:50052: Faile
d to connect to remote host: connect: Connection refused (111)}"
>
Calculated hypotenuse: 17.69180601295413
█
```

Figura 17. Ejecución del Servidor de Cálculo

Salida de Cliente:



```
cliente.py x
cliente.py > run
5 def run():
6
7
8
9     try:
10         with grpc.insecure_channel(server_address) as channel:
11             stub = calc_pb2_grpc.CalculationServiceStub(channel)
12
13             # Valores de ejemplo
14             a = 12
15             b = 13
16             print(f"Enviando solicitud con a={a}, b={b}...")
17
18             response = stub.Calculate(calc_pb2.CalculationRequest(a=a, b=b))
19             print(f"Hipotenusa calculada: {response.hypotenuse}")
20
21         except grpc.RpcError as e:
22             print(f"Error en la conexión con el servidor: {e}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Conectando al servidor de cálculo en 10.43.96.50:5000...
Enviando solicitud con a=12, b=13...
Hipotenusa calculada: 17.69180601295413
● estudiante@NGEN67:~/Documents/Distri$ python3 cliente.py
Conectando al servidor de cálculo en 10.43.96.50:5000...
Enviando solicitud con a=12, b=13...
Hipotenusa calculada: 17.69180601295413
○ estudiante@NGEN67:~/Documents/Distri$ █
```

Figura 18. Ejecución del Cliente