

Práctica guiada microservicios con RabbitMQ

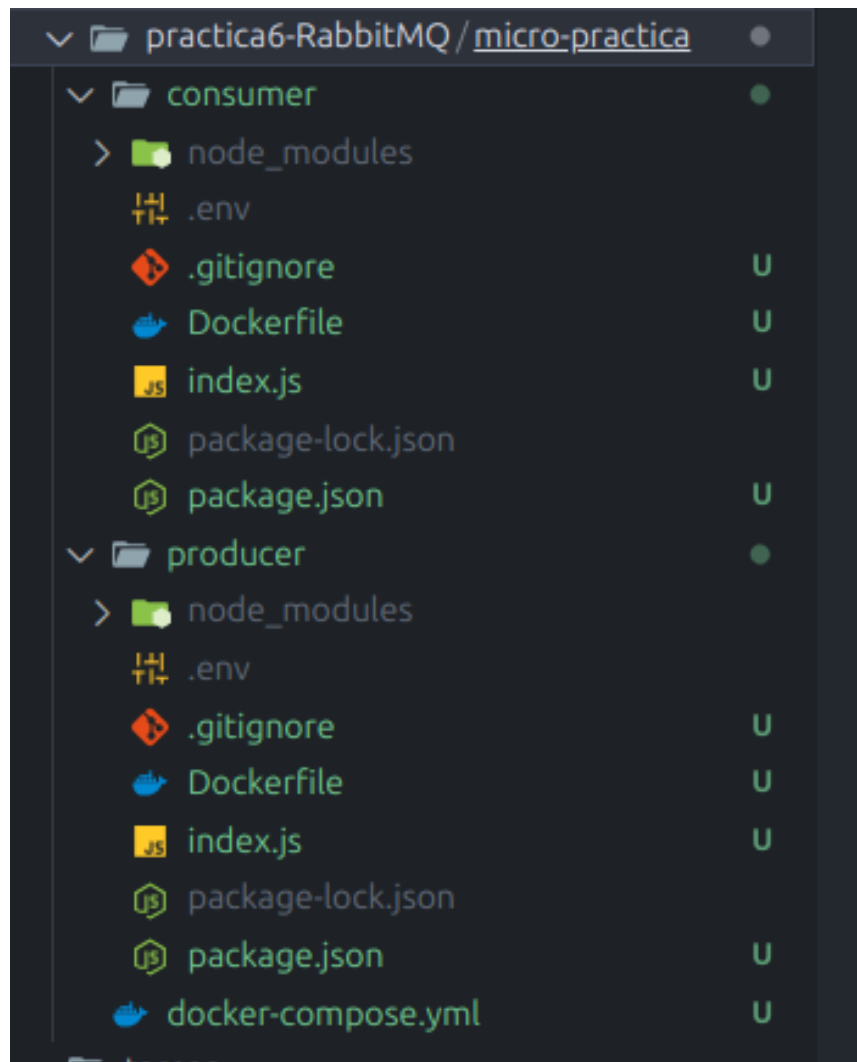
(Docker) microservicios en Node.js:

Nombre: Gonzales Suyo Franz Reinaldo

C.U. 111-500 - 35-5335

PROBAR PRUEBAS DE LAS PRÁCTICAS

ESTRUCTURA DE LA PRÁCTICA



1. Asegúrate RabbitMQ levantado

```
franz ~/practicas/practica6-RabbitMQ/micro-practica P main ? 05:44
→ docker compose logs -f rabbitmq

rabbitmq-1 | 2025-10-02 09:41:41.373496+00:00 [info] <0.283.0> ra: meta data store initialised for system coordination. 0 record(s) recovered
rabbitmq-1 | 2025-10-02 09:41:41.373603+00:00 [notice] <0.288.0> WAL: ra_coordination_log_wal init, open tbls: ra_coordination_log_open_mem_tables,
closed tbls: ra_coordination_log_closed_mem_tables
rabbitmq-1 | 2025-10-02 09:41:41.374868+00:00 [info] <0.254.0> ra: starting system coordination
rabbitmq-1 | 2025-10-02 09:41:41.374917+00:00 [info] <0.254.0> starting Ra system: coordination in directory: /var/lib/rabbitmq/mnesia/rabbit@3fcb4
ea28f9c/coordination/rabbit@3fcb4ea28f9c
rabbitmq-1 | 2025-10-02 09:41:41.454148+00:00 [info] <0.254.0> Waiting for Khepri leader for 30000 ms, 9 retries left
rabbitmq-1 | 2025-10-02 09:41:41.458165+00:00 [notice] <0.292.0> RabbitMQ metadata store: candidate -> leader in term: 1 machine version: 1
rabbitmq-1 | 2025-10-02 09:41:41.467185+00:00 [info] <0.254.0> Khepri leader elected
rabbitmq-1 | 2025-10-02 09:41:41.467271+00:00 [info] <0.254.0> Waiting for Khepri projections for 30000 ms, 9 retries left
rabbitmq-1 | 2025-10-02 09:41:41.642195+00:00 [info] <0.254.0>
rabbitmq-1 | 2025-10-02 09:41:41.642195+00:00 [info] <0.254.0> Starting RabbitMQ 3.13.7 on Erlang 26.2.5.15 [jit]
rabbitmq-1 | 2025-10-02 09:41:41.642195+00:00 [info] <0.254.0> Copyright (c) 2007-2024 Broadcom Inc and/or its subsidiaries
rabbitmq-1 | 2025-10-02 09:41:41.642195+00:00 [info] <0.254.0> Licensed under the MPL 2.0. Website: https://rabbitmq.com

rabbitmq-1 |
rabbitmq-1 | ## ##      RabbitMQ 3.13.7
rabbitmq-1 | ## ##
rabbitmq-1 | ##### Copyright (c) 2007-2024 Broadcom Inc and/or its subsidiaries
rabbitmq-1 | ##### ##
rabbitmq-1 | ##### Licensed under the MPL 2.0. Website: https://rabbitmq.com
rabbitmq-1 |
rabbitmq-1 | Erlang:      26.2.5.15 [jit]
rabbitmq-1 | TLS Library: OpenSSL - OpenSSL 3.1.8 11 Feb 2025
rabbitmq-1 | Release series support status: see https://www.rabbitmq.com/release-information
rabbitmq-1 |
rabbitmq-1 | Doc guides:  https://www.rabbitmq.com/docs
rabbitmq-1 | Support:     https://www.rabbitmq.com/docs/contact
rabbitmq-1 | Tutorials:   https://www.rabbitmq.com/tutorials
rabbitmq-1 | Monitoring:  https://www.rabbitmq.com/docs/monitoring
rabbitmq-1 | Upgrading:   https://www.rabbitmq.com/docs/upgrade
rabbitmq-1 |
rabbitmq-1 | Logs: <stdout>
```

2. En producer/: npm start (o npm run dev).

```
franz ~/practica6-RabbitMQ/micro-practica/producer P main ? v24.7.0 05:52
→ docker compose logs -f producer

WARN[0000] /home/franz/workspace/CICO/Micro-Servicios-6SFR/practicas/practica6-RabbitMQ/micro-practica/docker-compose.yml: the attribute
s obsolete, it will be ignored, please remove it to avoid potential confusion
producer-1 | [dotenv@17.2.3] injecting env (0) from .env -- tip: ☞ enable debug logging with { debug: true }
producer-1 | Intento 1/10 de conexión a RabbitMQ...
producer-1 | Error en intento 1/10: connect ECONNREFUSED 172.20.0.2:5672
producer-1 | Esperando 5000ms antes del siguiente intento...
producer-1 | Intento 2/10 de conexión a RabbitMQ...
producer-1 | Conectado a RabbitMQ exitosamente
producer-1 | Cola asegurada: email_queue
producer-1 | Producer API en http://localhost:3000
producer-1 | Health check: http://localhost:3000/health
```

3. En consumer/: npm start (o npm run dev).

```
franz ~/practica6-RabbitMQ/micro-practica/consumer P main ? v24.7.0 05:33
→ docker compose logs -f consumer

WARN[0000] /home/franz/workspace/CICO/Micro-Servicios-6SFR/practicas/practica6-RabbitMQ/micro-practica/docker-compose.yml: the attribute
s obsolete, it will be ignored, please remove it to avoid potential confusion
consumer-1 | [dotenv@17.2.3] injecting env (1) from .env -- tip: 📖 add secrets lifecycle management: https://dotenvx.com/ops
consumer-1 | Intento 1/10 de conexión a RabbitMQ...
consumer-1 | Error en intento 1/10: connect ECONNREFUSED 172.20.0.2:5672
consumer-1 | Esperando 5000ms antes del siguiente intento...
consumer-1 | Intento 2/10 de conexión a RabbitMQ...
consumer-1 | Conectado a RabbitMQ exitosamente
consumer-1 | Esperando mensajes en la cola: email_queue
```

4. Registrar un usuario (ejemplo curl):

```
franz ~ 05:45
→ curl -X POST http://localhost:3000/register -H "Content-Type: application/json" -d '{"name":"Carlitos2","email":"juan.carlitos@example.com","cell":"781133000"}'

{"ok":true,"user":{"id":"1759398301103","name":"Carlitos2","email":"juan.carlitos@example.com","cell":"781133000","createdAt":"2025-10-02T09:45:01.103Z"}}

franz ~ 05:45
→ curl -X POST http://localhost:3000/register -H "Content-Type: application/json" -d '{"name":"Carlitos2","email":"juan.carlitos@example.com","cell":"781133000"}'

{"ok":true,"user":{"id":"1759398302062","name":"Carlitos2","email":"juan.carlitos@example.com","cell":"781133000","createdAt":"2025-10-02T09:45:02.062Z"}}

franz ~ 05:45
→ curl -X POST http://localhost:3000/register -H "Content-Type: application/json" -d '{"name":"Carlitos2","email":"juan.carlitos@example.com","cell":"781133000"}'

{"ok":true,"user":{"id":"1759398302662","name":"Carlitos2","email":"juan.carlitos@example.com","cell":"781133000","createdAt":"2025-10-02T09:45:02.662Z"}}

franz ~ 05:45
→ curl -X POST http://localhost:3000/register -H "Content-Type: application/json" -d '{"name":"Carlitos2","email":"juan.carlitos@example.com","cell":"781133000"}'

{"ok":true,"user":{"id":"1759398303565","name":"Carlitos2","email":"juan.carlitos@example.com","cell":"781133000","createdAt":"2025-10-02T09:45:03.565Z"}}
```

```
franz ~/practica6-RabbitMQ/micro-practica/producer 1? main ? v24.7.0 05:52
→ docker compose logs -f producer

producer-1 | Esperando 5000ms antes del siguiente intento...
producer-1 | Intento 2/10 de conexión a RabbitMQ...
producer-1 | Conectado a RabbitMQ exitosamente
producer-1 | Cola asegurada: email_queue
producer-1 | Producer API en http://localhost:3000
producer-1 | Health check: http://localhost:3000/health
producer-1 | Mensaje enviado a la cola: {
producer-1 |   type: 'NEW_USER',
producer-1 |   user: {
producer-1 |     id: 1759398299430,
producer-1 |     name: 'Carlitos2',
producer-1 |     email: 'juan.carlitos@example.com',
producer-1 |     cell: '781133000',
producer-1 |     createdAt: '2025-10-02T09:44:59.430Z'
producer-1 |   }
producer-1 | }
producer-1 | Mensaje enviado a la cola: {
producer-1 |   type: 'NEW_USER',
producer-1 |   user: {
producer-1 |     id: 1759398301103,
producer-1 |     name: 'Carlitos2',
producer-1 |     email: 'juan.carlitos@example.com',
producer-1 |     cell: '781133000',
producer-1 |     createdAt: '2025-10-02T09:45:01.103Z'
producer-1 |   }
producer-1 | }
producer-1 | Mensaje enviado a la cola: {
producer-1 |   type: 'NEW_USER',
producer-1 |   user: {
producer-1 |     id: 1759398302062,
producer-1 |     name: 'Carlitos2',
producer-1 |     email: 'juan.carlitos@example.com',
producer-1 |     cell: '781133000',
producer-1 |     createdAt: '2025-10-02T09:45:02.062Z'
producer-1 |   }
producer-1 | }
```

```
franz ~/practica6-RabbitMQ/micro-practica/consumer ? main ? v24.7.0 05:33
→ docker compose logs -f consumer
consumer-1 | Esperando mensajes en la cola: email_queue
consumer-1 | Mensaje recibido: {
consumer-1 |   type: 'NEW_USER',
consumer-1 |   user: {
consumer-1 |     id: 1759398299430,
consumer-1 |     name: 'Carlitos2',
consumer-1 |     email: 'juan.carlitos@example.com',
consumer-1 |     cell: '781133000',
consumer-1 |     createdAt: '2025-10-02T09:44:59.430Z'
consumer-1 |   }
consumer-1 | }
consumer-1 | Simulando envío de correo a: juan.carlitos@example.com (nombre: Carlitos2)
consumer-1 | Correo "enviado" a juan.carlitos@example.com (simulado)
consumer-1 | Mensaje recibido: {
consumer-1 |   type: 'NEW_USER',
consumer-1 |   user: {
consumer-1 |     id: 1759398301103,
consumer-1 |     name: 'Carlitos2',
consumer-1 |     email: 'juan.carlitos@example.com',
consumer-1 |     cell: '781133000',
consumer-1 |     createdAt: '2025-10-02T09:45:01.103Z'
consumer-1 |   }
consumer-1 | }
consumer-1 | Simulando envío de correo a: juan.carlitos@example.com (nombre: Carlitos2)
consumer-1 | Correo "enviado" a juan.carlitos@example.com (simulado)
consumer-1 | Mensaje recibido: {
consumer-1 |   type: 'NEW_USER',
consumer-1 |   user: {
consumer-1 |     id: 1759398302062,
consumer-1 |     name: 'Carlitos2',
consumer-1 |     email: 'juan.carlitos@example.com',
consumer-1 |     cell: '781133000',
consumer-1 |     createdAt: '2025-10-02T09:45:02.062Z'
consumer-1 |   }
consumer-1 | }
consumer-1 | Simulando envío de correo a: juan.carlitos@example.com (nombre: Carlitos2)
```

INTERFAZ DE RABBITMQ

← → ↺

🔍 http://localhost:15672/#/channels

RabbitMQ™

RabbitMQ 3.13.7 Erlang 26.2.5.15

Overview

Connections

Channels

Exchanges

Queues and Streams

Admin

Channels

▼ All channels (2)

Pagination

Page 1 of 1 - Filter: ☐ Regex ?

Overview				Details			Message rates					
Channel	User name	Mode ?	State	Unconfirmed	Prefetch ?	Unacked	publish	confirm	unroutable (drop)	deliver / get	ack	
172.20.0.3:44998 (1)	guest		idle	0		0	0.00/s	0.00/s	0.00/s			
172.20.0.4:50358 (1)	guest		idle	0	1	0				0.00/s	0.00/s	

HTTP API

Documentation

Tutorials

New releases

Commercial edition

Commercial support

Discussions

Discord

Plugins

GitHub

Overview

Connections

Channels

Exchanges

Queues and Streams

Admin

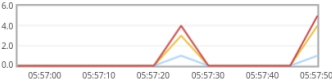
Overview

▼ Totals

Queued messages

last minute

?



Ready

2

Unacked

1

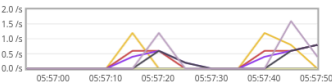
Total

3

Message rates

last minute

?



Publish

0.00/s

Publisher confirm

0.00/s

Deliver (manual ack)

0.80/s

Deliver (auto ack)

0.00/s

Consumer ack

0.80/s

Redelivered

0.00/s

Get (manual ack)

0.00/s

Get (auto ack)

0.00/s

Get (empty)

0.00/s

Unroutable (return)

0.00/s

Unroutable (drop)

0.00/s

Disk read

0.80/s

Disk write

0.40/s

Global counts

?

Connections: 2

Channels: 2

Exchanges: 7

Queues: 1

Consumers: 1

▼ Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Cores	Info	Reset stats	+/-
rabbit@3fcb4ea28f9c	47 1073741816 available	2 966367545 available	469 1048576 available	160 MiB 9.0 GiB high watermark	61 GiB 48 MiB low watermark	16m 9s	8	basic 2 rss	This node All nodes	

▼ Churn statistics

Connection operations

last minute

?



Created

0.00/s

Closed

0.00/s

Overview

Connections

Channels

Exchanges

Queues and Streams

Admin

Queues

▼ All queues (1)

Pagination

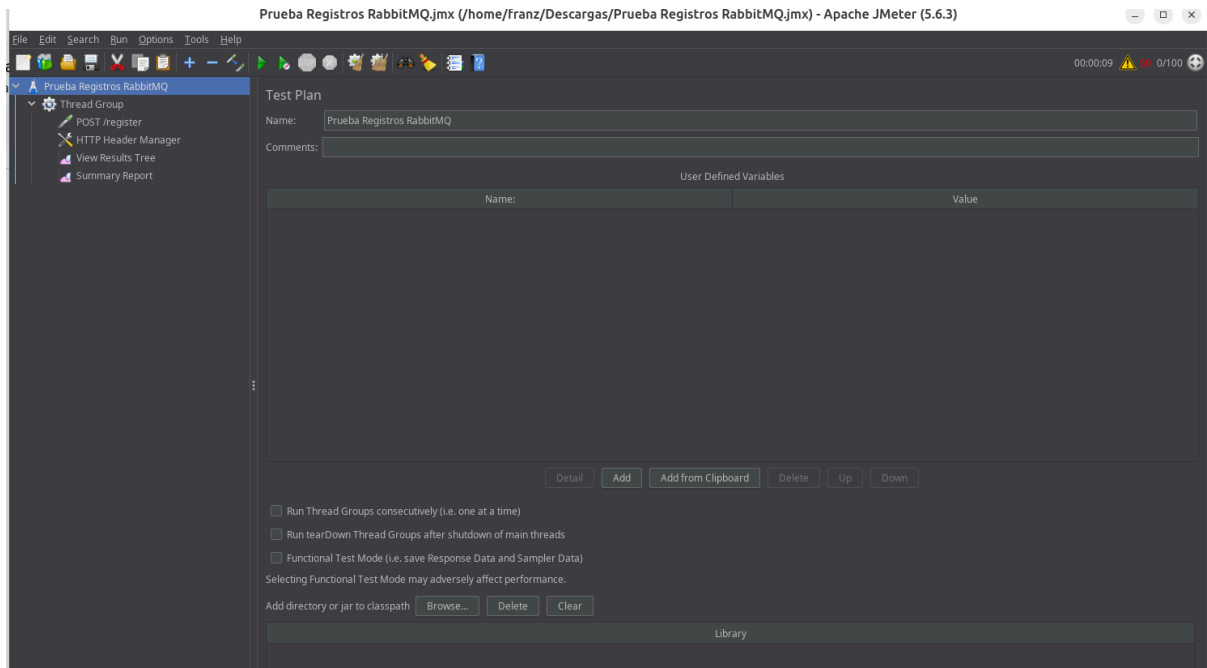
Page 1 of 1 - Filter: ☐ Regex ?

Overview					Messages			Message rates			+/-
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
/	email_queue	classic	D	running	4	1	5	0.80/s	0.00/s	0.00/s	

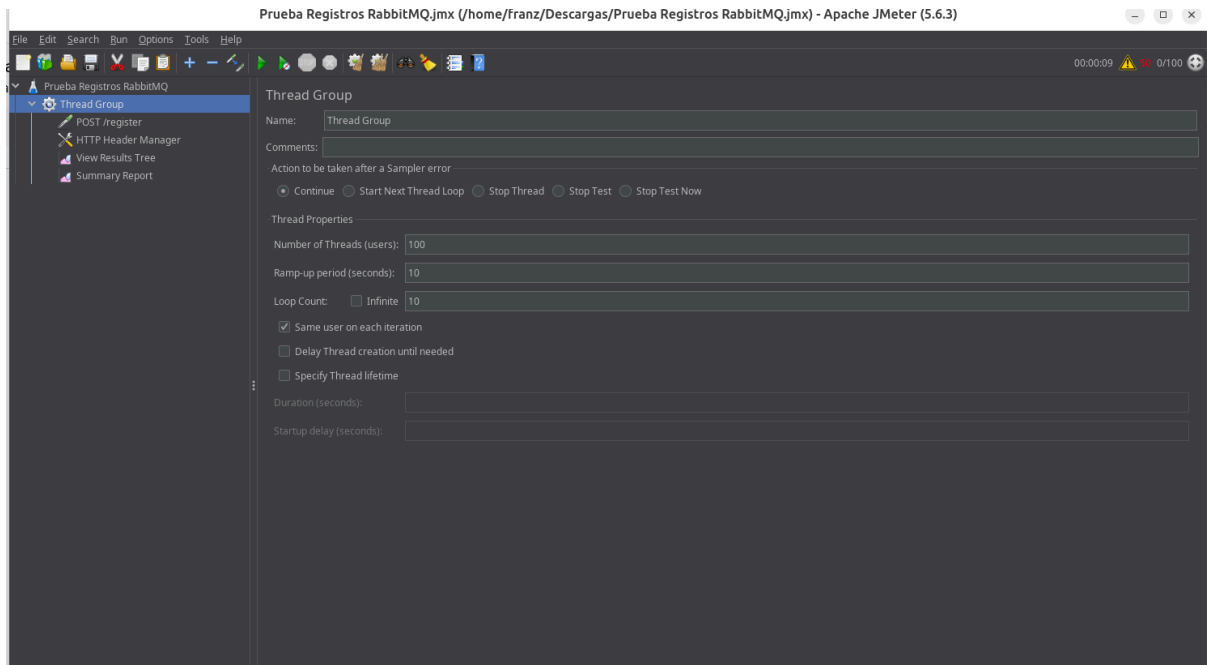
▶ Add a new queue

PRUEBAS CON 1000 REGISTROS CON JMETER

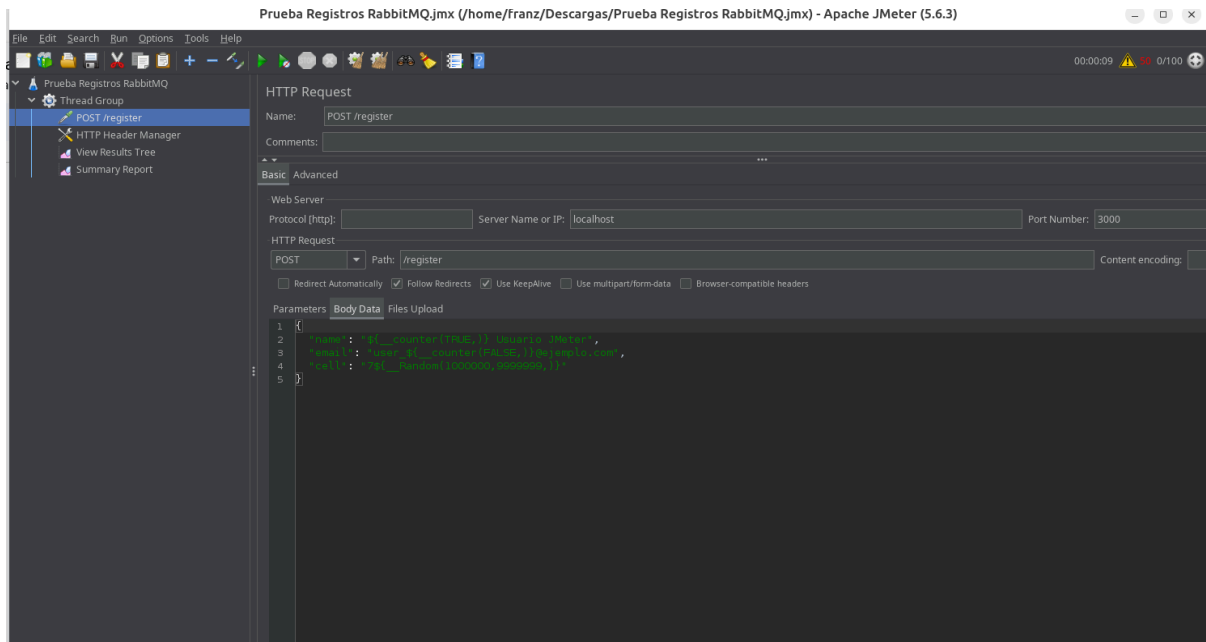
1: Crear un Nuevo Test Plan en JMeter



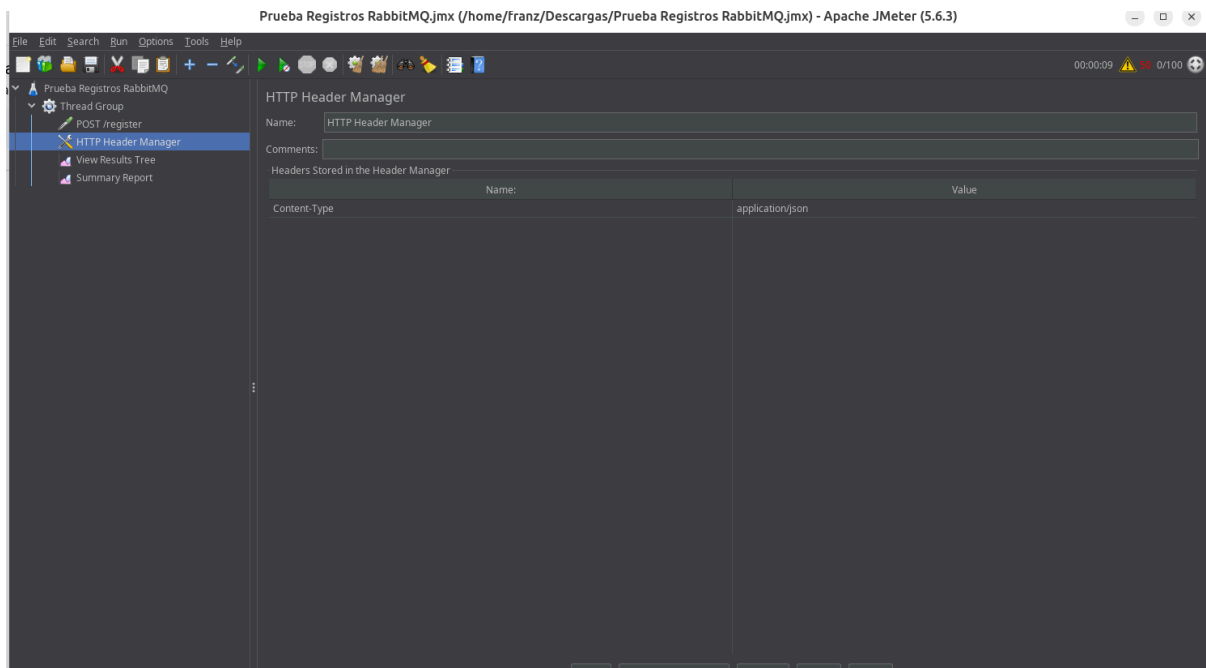
2: Configurar el Thread Group (Simulación de Usuarios)



3: Agregar un HTTP Request Sampler (La Solicitud POST)



4: Configurar Headers para JSON (Opcional pero Recomendado)



5: Agregar Listeners para Ver Resultados

Los listeners muestran métricas en tiempo real y al final.

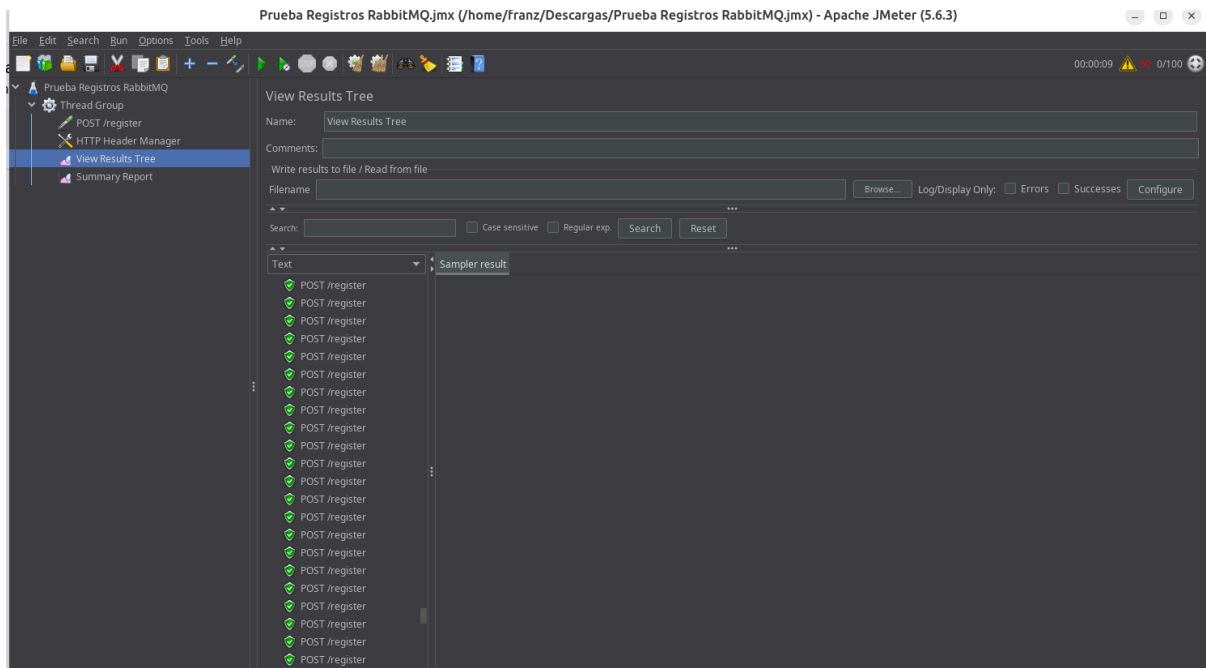
1. Haz clic derecho en **Thread Group** > **Add** > **Listener** > **View Results Tree** (para ver detalles de cada solicitud, como respuesta JSON del producer).
 - Útil para debug: Verás el body de respuesta (e.g., { "ok": true, "user": {...} }).

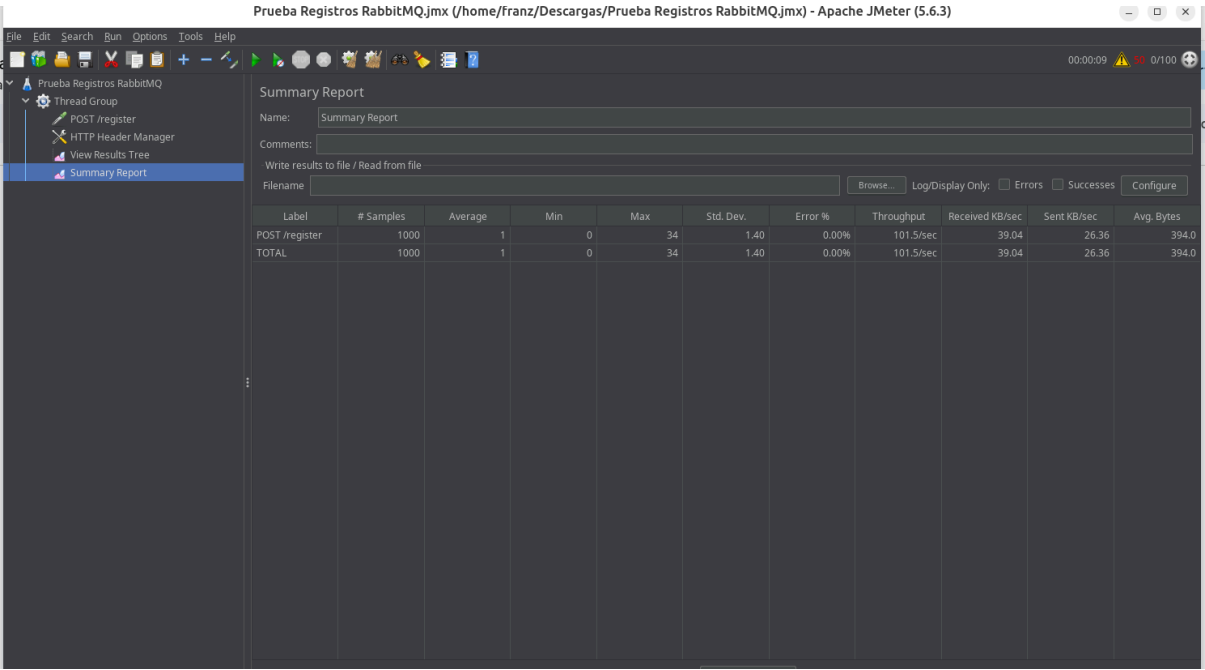
2. Haz clic derecho en **Thread Group** > **Add** > **Listener** > **Summary Report** (resumen simple: # Samples, Average time, Error %, Throughput).
3. Opcional: **Aggregate Report** para métricas detalladas (90% Line, Min/Max time, etc.).
 - Haz clic derecho en **Thread Group** > **Add** > **Listener** > **Aggregate Report**.

6: Configuraciones Adicionales del Test Plan

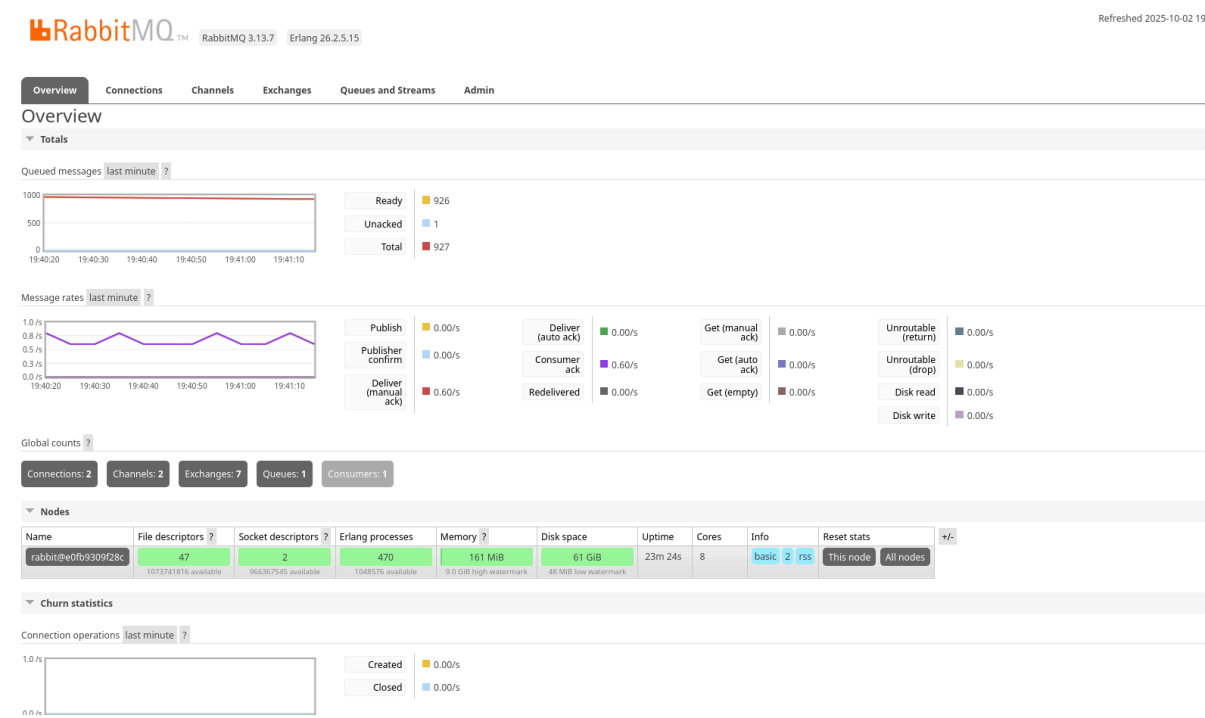
1. En **Test Plan** (nivel raíz):
 - Marca **"Run Thread Groups consecutively"** si quieres pruebas secuenciales, pero déjalo off para concurrente.
 - **User Defined Variables**: No necesario aquí.
2. Guarda el plan: **File** > **Save** como prueba-1000-registros-rabbitmq.jmx.

7: Ejecutar la Prueba





Muestras de las pruebas en RabbitMQ



Queues

▼ All queues (1)

Pagination

Page

1

 of 1 - Filter: ☐ Regexp

?

Overview					Messages			Message rates				+/-
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack		
/	email_queue	classic	<div>D</div>	<div>running</div>	918	1	919	0.00/s	0.80/s	0.80/s		

► Add a new queue