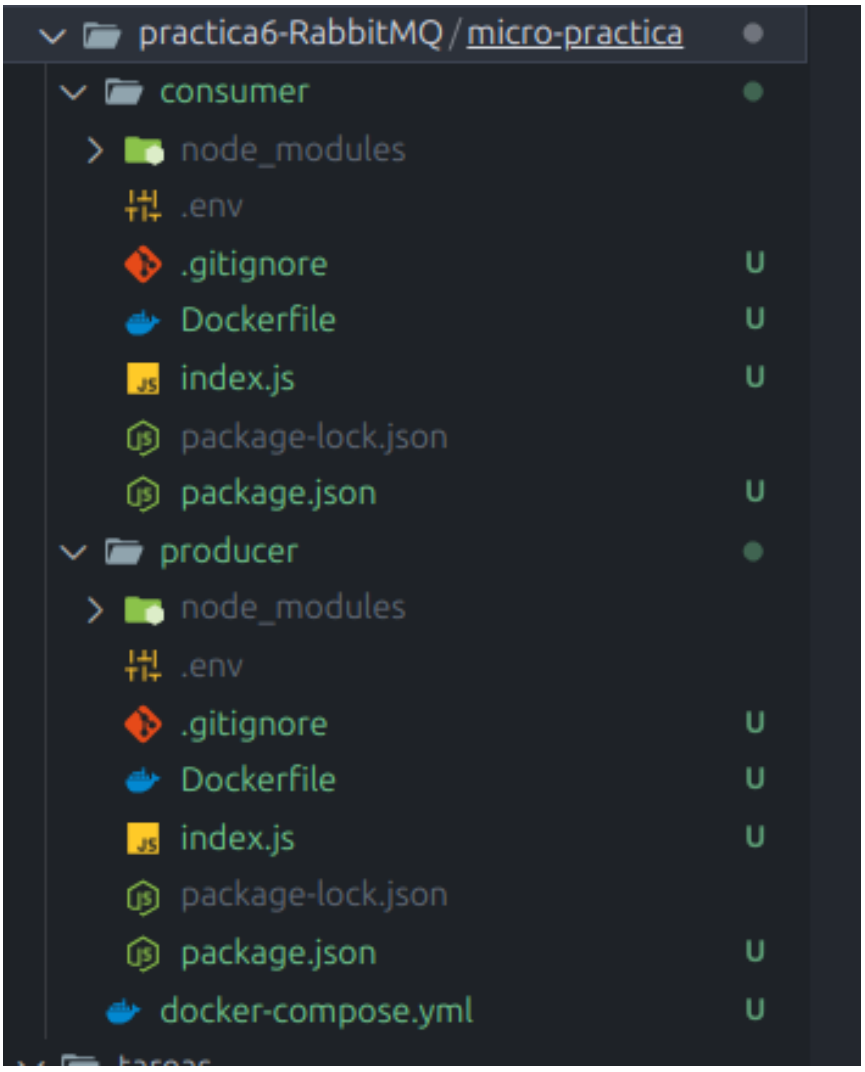
Práctica guiada microservicios con RabbitMQ

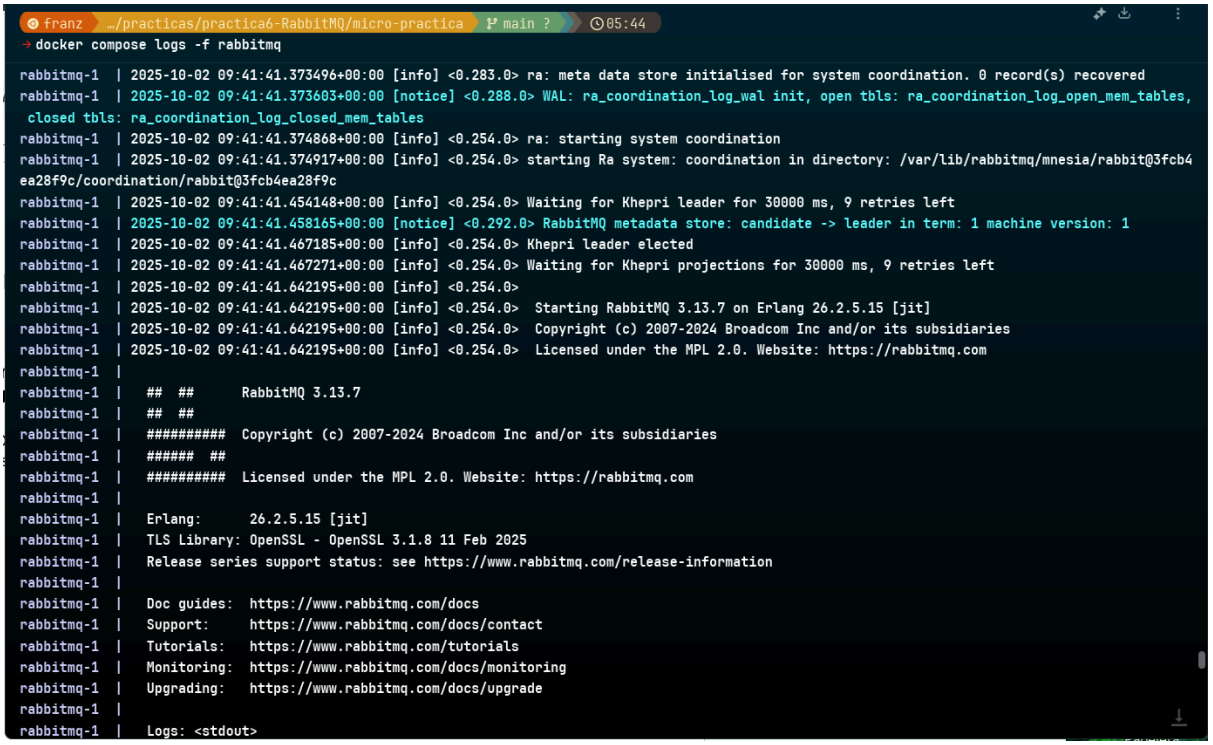
(Docker) microservicios en Node.js:

Nombre: Gonzales Suyo Franz Reinaldo
C.U. 111-500 - 35-5335

PROBAR PRUEBAS DE LAS PRÁCTICAS
ESTRUCTURA DE LA PRÁCTICA



1. Asegúrate RabbitMQ levantado



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

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

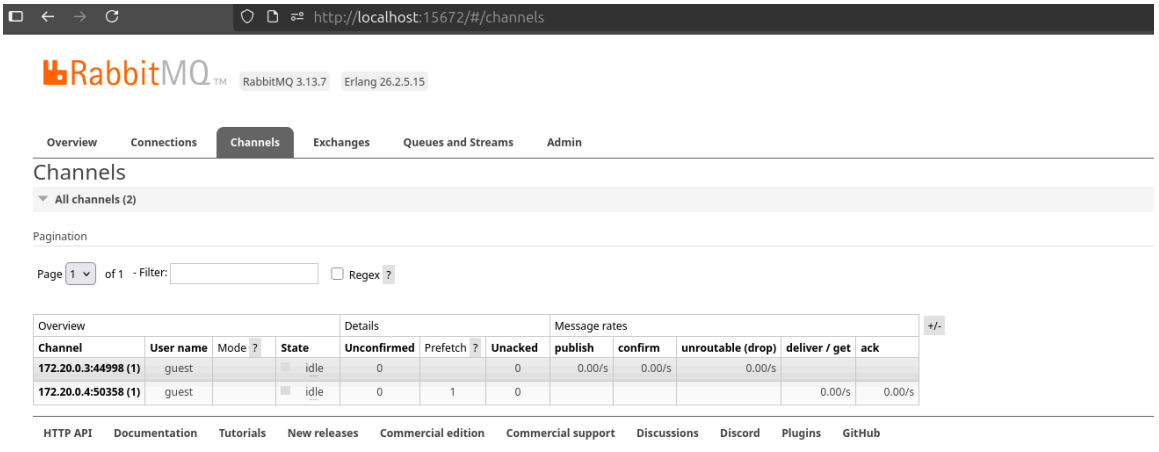
4. Registrar un usuario (ejemplo curl):

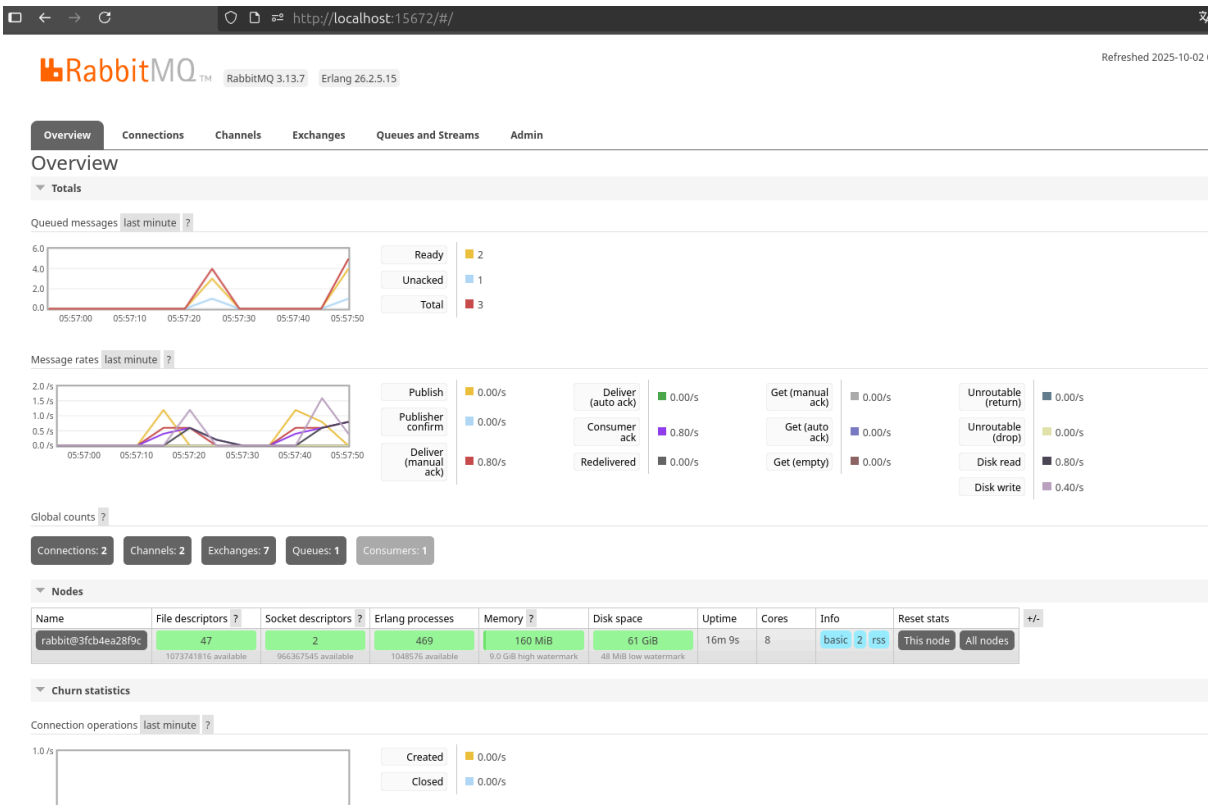
```
franz ~ 05:45
→ curl -X POST http://localhost:3000/register -H "Content-Type: application/json" -d '{"name":"Carlitos2","email":"juan.carlitos@example.com","cell"
:"781133000"}'
{"ok":true,"user":{"id":1759398301103,"name":"Carlitos2","email":"juan.carlitos@example.com","cell":"781133000","createdAt":"2025-10-02T09:45:01.103
Z"}}

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":"2025-10-02T09:45:02.062
Z"}}

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":"2025-10-02T09:45:02.662
Z"}}

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":"2025-10-02T09:45:03.565
Z"}}
```

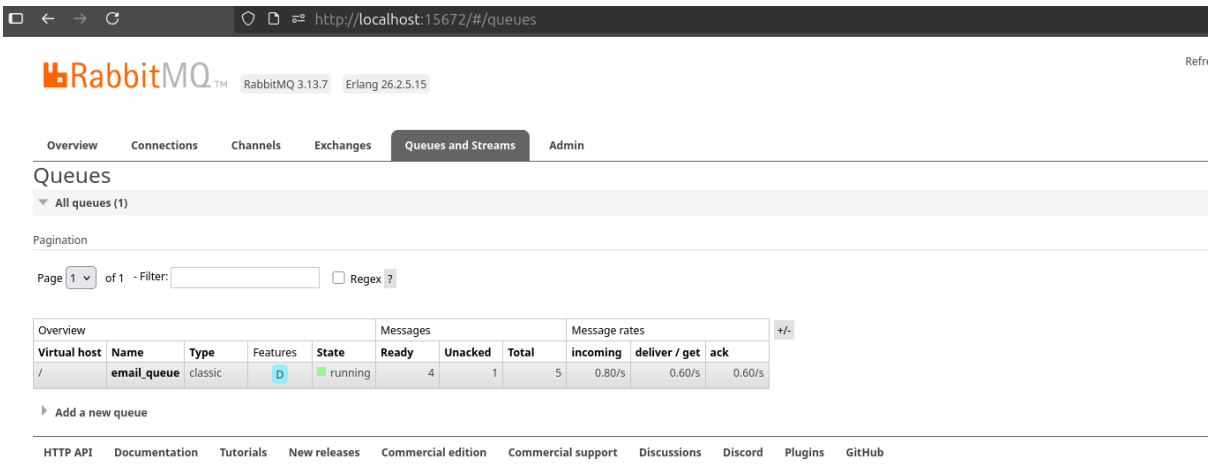
```
franz …/practica6-RabbitMQ/micro-practica/producer main ? v24.7.0 05:52
→ docker compose logs -f producer
producer-1  | Esperando 5000ms antes del siguiente intento...
producer-1  | Intento 2/10 de conexión a RabbitMQ...
producer-1  | Conectado a RabbitMQ exitosamente
producer-1  | Cola asegurada: 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

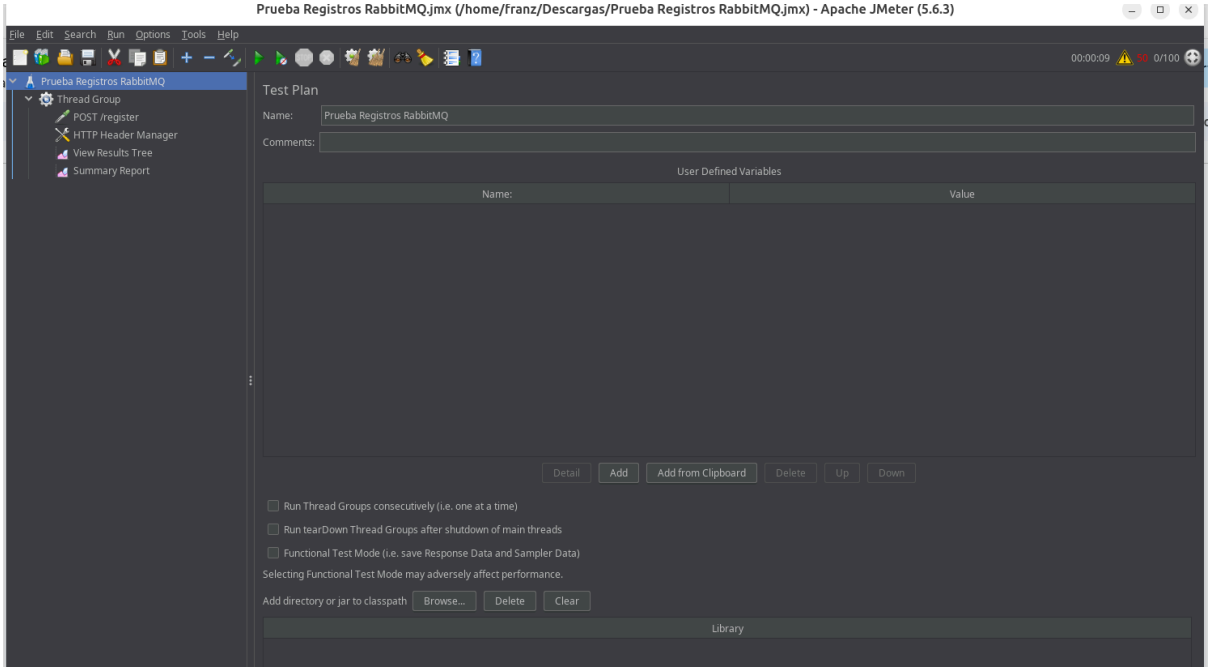



http://localhost:15672/#/
RabbitMQ
RabbitMQ 3.13.7
Erlang 26.2.5.15
Refreshed 2025-10-02
Overview
Connections
Channels
Exchanges
Queues and Streams
Admin
Overview
Totals
Queued messages last minute
Ready 2
Unacked 1
Total 3
Message rates last minute
Publish 0.00/s
Publisher confirm 0.00/s
Deliver (manual 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
2
469
160 MiB
61 GiB
16m 9s
8
basic 2 rss
This node
All nodes
Churn statistics
Connection operations last minute
Created 0.00/s
Closed 0.00/s

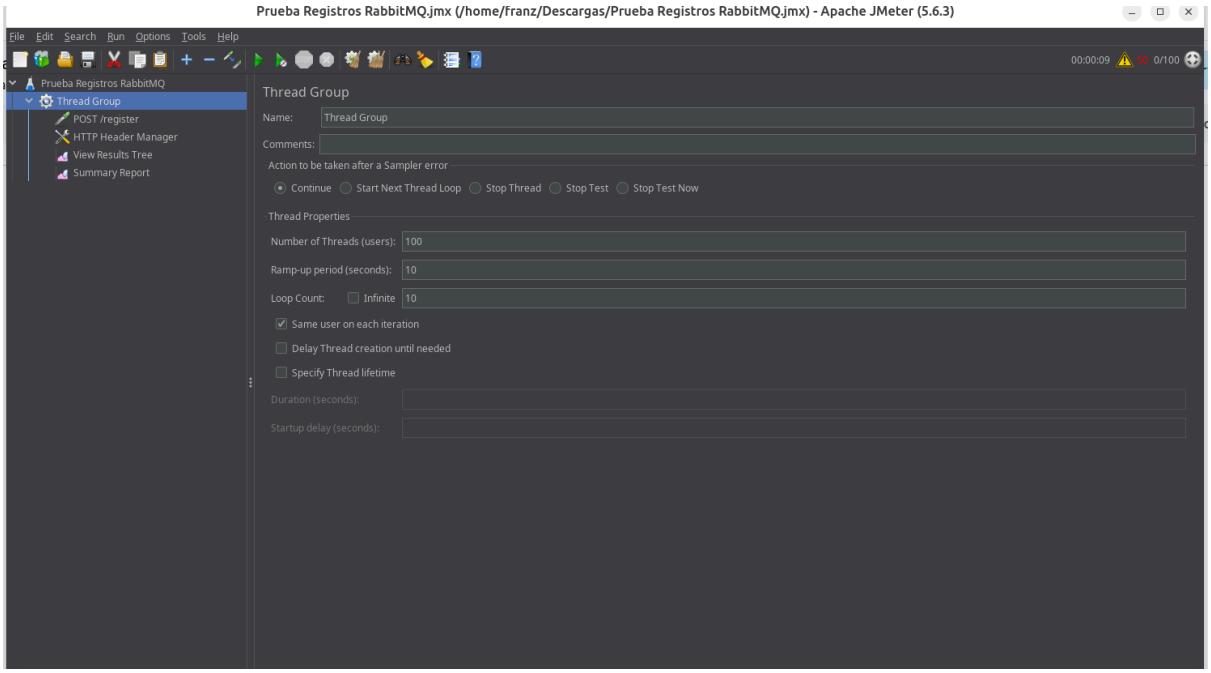

http://localhost:15672/#/queues
RabbitMQ
RabbitMQ 3.13.7
Erlang 26.2.5.15
Overview
Connections
Channels
Exchanges
Queues and Streams
Admin
Queues
All queues (1)
Pagination
Page 1 of 1 - Filter:
Regex
Overview
Messages
Message rates
Virtual host
Name
Type
Features
State
Ready
Unacked
Total
incoming
deliver / get
ack
/
email_queue
classic
D
running
4
1
5
0.80/s
0.60/s
0.60/s
Add a new queue
HTTP API
Documentation
Tutorials
New releases
Commercial edition
Commercial support
Discussions
Discord
Plugins
GitHub

PRUEBAS CON 1000 REGISTROS CON JMETER

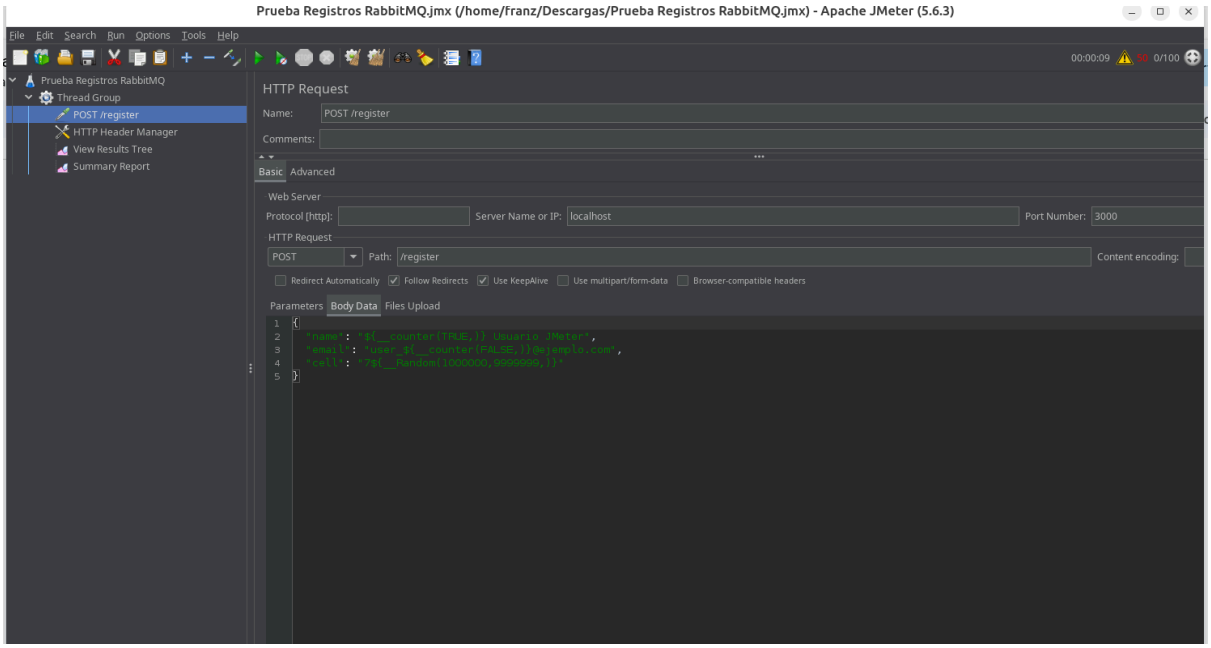
1: Crear un Nuevo Test Plan en JMeter



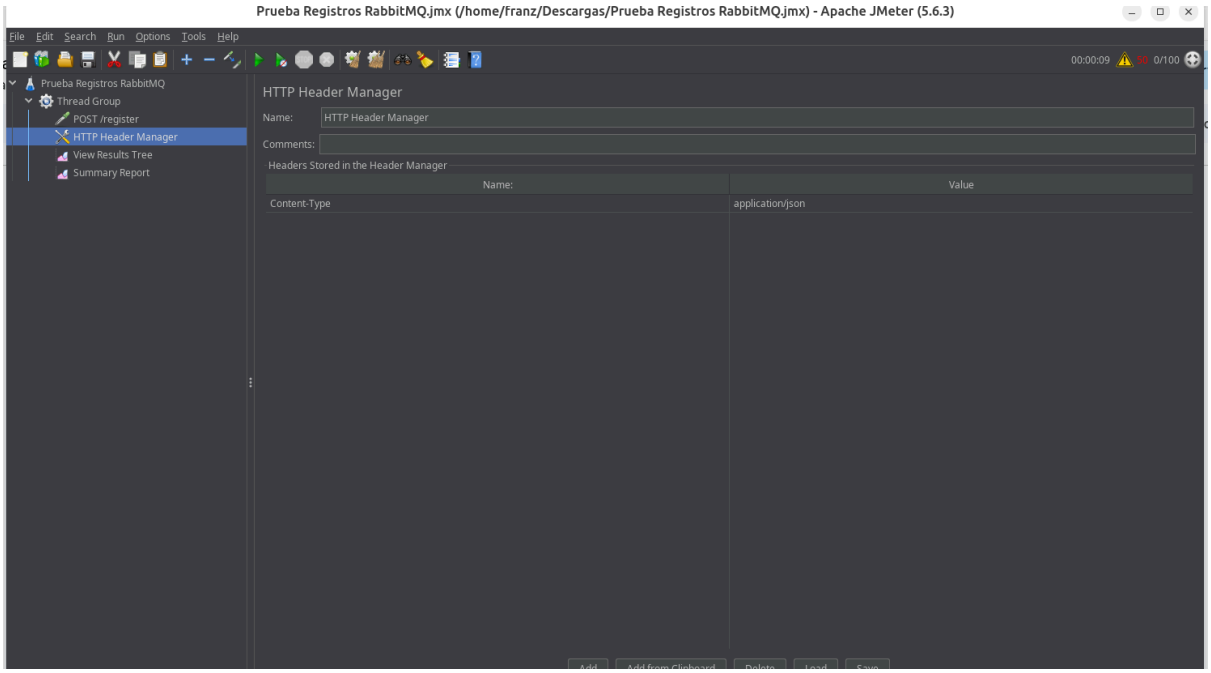
2: Configurar el Thread Group (Simulación de Usuarios)



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.

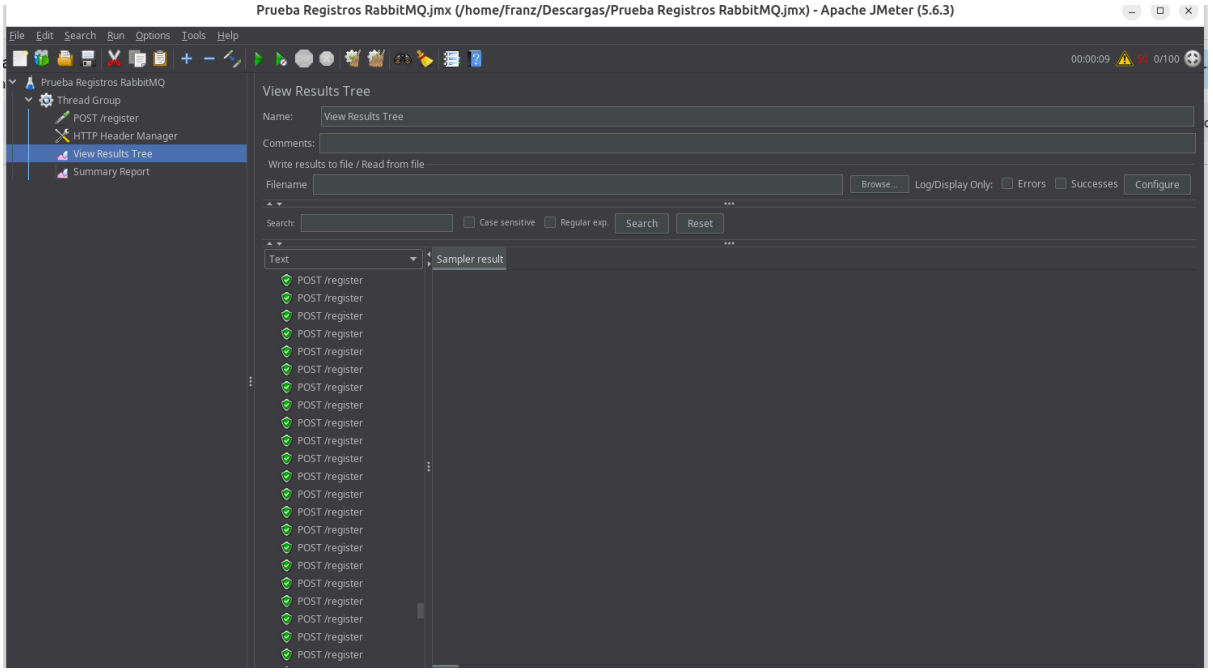
1. Haz clic derecho en **Thread Group** > **Add > Listener > View Results Tree** (para ver detalles de cada solicitud, como respuesta JSON del producer).
    - Útil para debug: Verás el body de respuesta (e.g., { "ok": true, "user": {...} }).

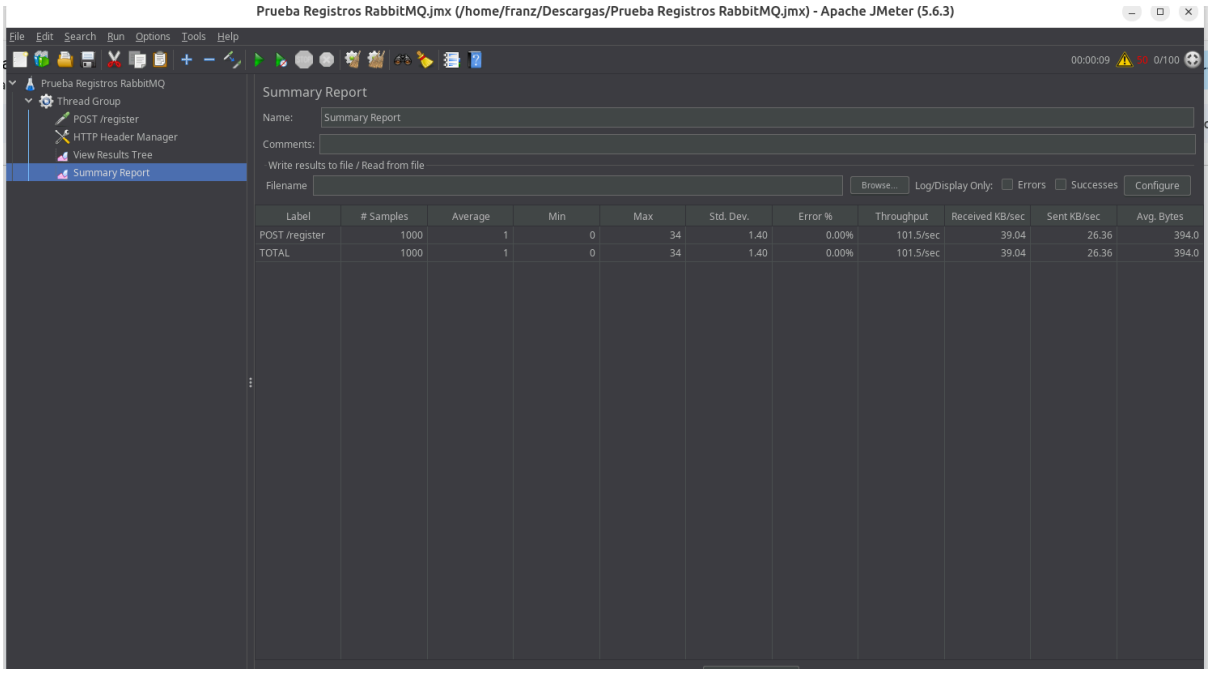
2. Haz clic derecho en **Thread Group** > **Add > Listener > Summary Report** (resumen simple: # Samples, Average time, Error %, Throughput).
3. Opcional: **Aggregate Report** para métricas detalladas (90% Line, Min/Max time, etc.).
    - Haz clic derecho en **Thread Group** > **Add > Listener > Aggregate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

6: Configuraciones Adicionales del Test Plan

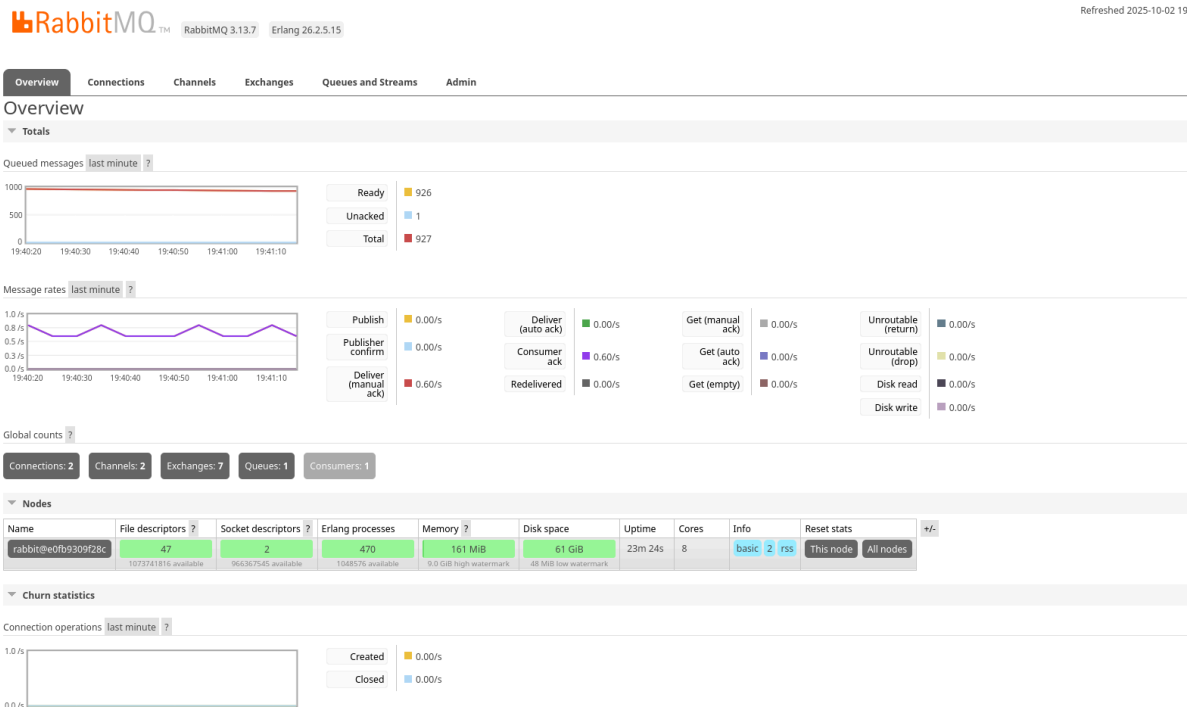
1. En **Test Plan** (nivel raíz):
    - Marca **"Run Thread Groups consecutively"** si quieres pruebas secuenciales, pero déjalo off para concurrente.
    - **User Defined Variables**: No necesario aquí.
2. Guarda el plan: **File > Save** como prueba-1000-registros-rabbitmq.jmx.

7: Ejecutar la Prueba





Muestras de las pruebas en RabbitMQ



RabbitMQ™ RabbitMQ 3.13.7 Erlang 26.2.5.15

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	918	1	919	0.00/s	0.80/s	0.80/s	

Add a new queue

HTTP API | Documentation | Tutorials | New releases | Commercial edition | Commercial support | Discussions | Discord | Plugins | GitHub