Project Presentation: Distributed System for Data Validation and Persistence

Daniela Cadena
Eladio Huamani

National University of Engineering
Faculty of Sciences

CC4P1 - Concurrent Programming and Distributed Systems

1. Introduction

- **Project Goal:** Implement a distributed system that simulates a real-world scenario of asynchronous data validation and persistence, using messaging middleware.
- **Key Technologies:**
  - Node.js (Client)
  - Java (Validation Service)
  - Python (Persistence Service)
  - RabbitMQ (Middleware)
  - PostgreSQL (Main DB)
  - MariaDB (Validation DB)
  - Docker & Docker Compose
- **Simulation:** A flow where user data is sent, validated (e.g., DNI), and then stored in a decoupled and robust manner.

2. System Architecture (Part 1)

- **Overview:**
  - A system composed of multiple independent services that collaborate through a message bus.
-
- **Main Components:**
  - **Client (Node.js):** Interface for data entry. Sends data to RabbitMQ.
  - **DNI Validator Service (Java):** Consumes from RabbitMQ, validates DNI against MariaDB, and resends if valid.
  - **Persistence Service (Python):** Consumes validated messages and stores them in PostgreSQL.

2. System Architecture (Part 2)

- Databases:
  - **DB1 (PostgreSQL):** Stores validated user information (ID, name, email, DNI, etc.).
  - **DB2 (MariaDB):** Contains reference records for DNI validation.

3. Communication Flow: RabbitMQ (Part 1)

- **What is RabbitMQ?**
  - An open-source message broker that acts as an intermediary between producers and consumers.
  - Enables asynchronous and decoupled communication.
- **Detailed Message Flow:**
  1. **Client (Node.js)** sends user data to the `validate_dni` queue.
  2. **Validator Service (Java)** consumes from `validate_dni`. Validates DNI against **DB2 (MariaDB)**.
  3. If DNI is valid, it publishes the message to `validate_save_user`.
  4. **Persistence Service (Python)** consumes from `validate_save_user`. Saves user to **DB1 (PostgreSQL)**.

3. Communication Flow: RabbitMQ (Part 2)

- **Key RabbitMQ Concepts Used:**
  - **Producer:** Node.js Client.
  - **Consumers:** Java and Python Services.
  - **Queues:** `validate_dni`, `validate_save_user` (durable).
  - **Persistent Messages:** To prevent message loss.
  - **Acknowledgements (ACKs):** Confirmation of processing.
- **Benefits of RabbitMQ:**
  - Decoupling, Scalability, Fault Tolerance, Asynchronous Processing.

Visual Suggestion

4. Implementation Details (Part 1)

- **Client (Node.js):**
  - Use of `inquirer` for interactive capture.
  - Connection to RabbitMQ (`amqplib`), sending to `validate_dni`.
  - Messages as `persistent: true`.
- **DNI Validator (Java):**
  - Connection to RabbitMQ (`com.rabbitmq.client`) with retries.
  - Consumption from `validate_dni`.
  - DNI extraction (regex), validation against MariaDB (JDBC).
  - Publishing to `validate_save_user` if successful.
  - Handling of `basicAck`.

4. Implementation Details (Part 2)

- **Consumer and Persistence (Python):**
  - Connection to RabbitMQ (`pika`) with `BlockingConnection`.
  - Consumption of messages from `validate_save_user`.
  - Data processing and saving to PostgreSQL (`psycopg2`).
  - **Concurrency:** Use of threads (`threading`) to process multiple messages and improve performance.
  - Handling of `basic_ack`.

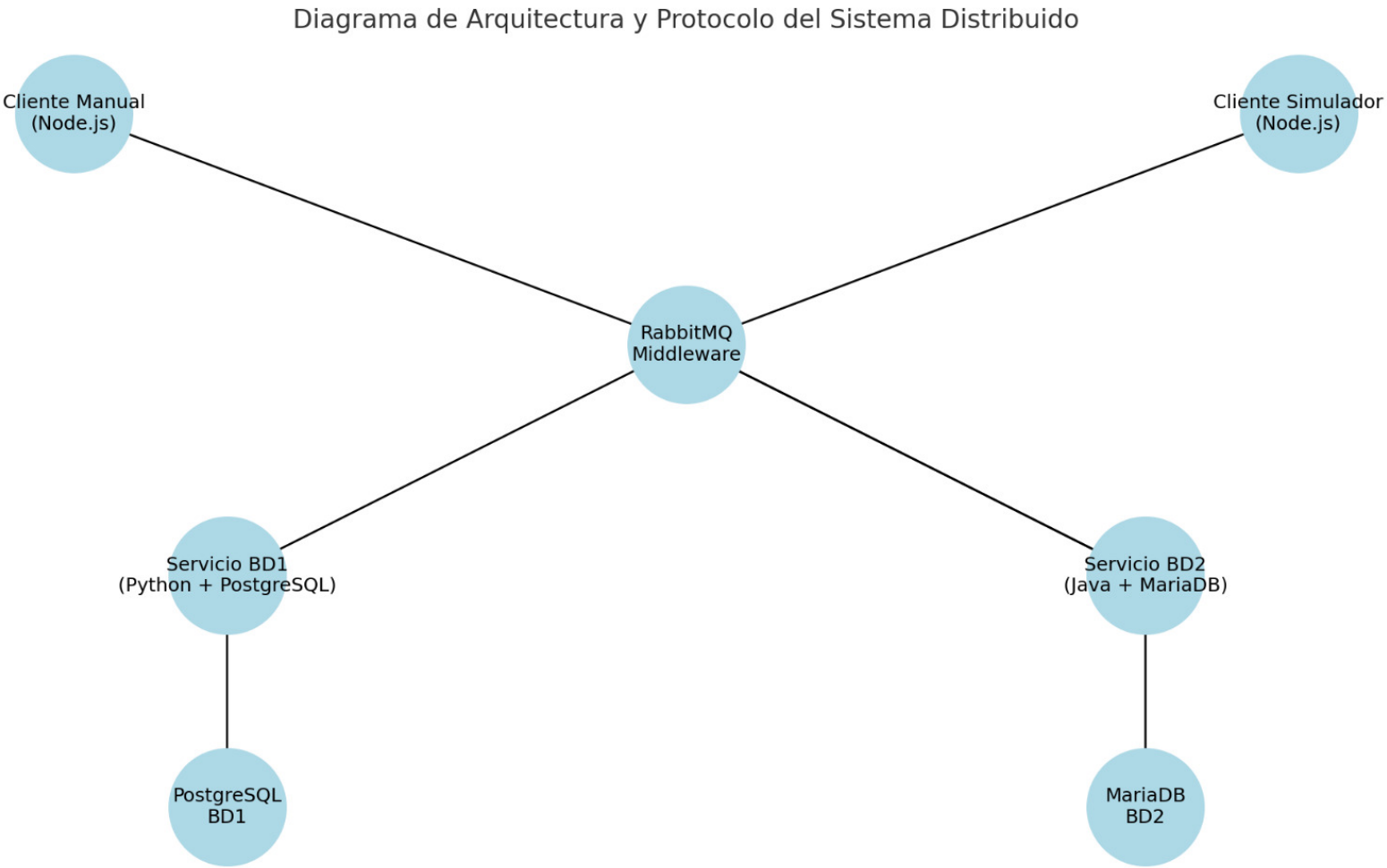

Figura 1: Architechture Diagram

5. Docker and Docker Compose (Part 1)

- **Why Docker?**
  - Consistent Environments.
  - Service Isolation.
  - Simple Scalability.
  - Simplified Deployment (`docker-compose up`).
- **Orchestration with Docker Compose (`docker-compose.yml`):**
  - Service definition: `rabbitmq`, `bd1` (PostgreSQL), `bd2` (MariaDB), `client-simulator` (Node.js), `service_bd1` (Python), `service_bd2` (Java).
  - Configuration of internal Docker networks.

5. Docker and Docker Compose (Part 2)

- **The Role of Dockerfiles:**
  - `client-simulator` **(Node.js):** Base `node:18-alpine`, `npm install`.
  - `service_bd2` **(Java):** Base `openjdk:17`, compiles, RabbitMQ retries.
  - `service_bd1` **(Python):** Base `python:3.11-slim`, installs `pika`, `psycopg2`.
  - Lightweight and specific images.
- **Lifecycle with** `docker-compose up`**:**
  1. Start `rabbitmq`, `bd1`, `bd2`.
  2. SQL scripts to initialize DBs.
  3. Java/Python services wait for RabbitMQ and listen to queues.
  4. Interactive Node.js for data input.

5. Docker and Docker Compose (Part 3)

- **Robustness Thanks to Docker:**
  - Retry logic in services to handle race conditions at startup.
  - Volume mounting for SQL scripts.
  - Secure and fluid communication over the Docker network.

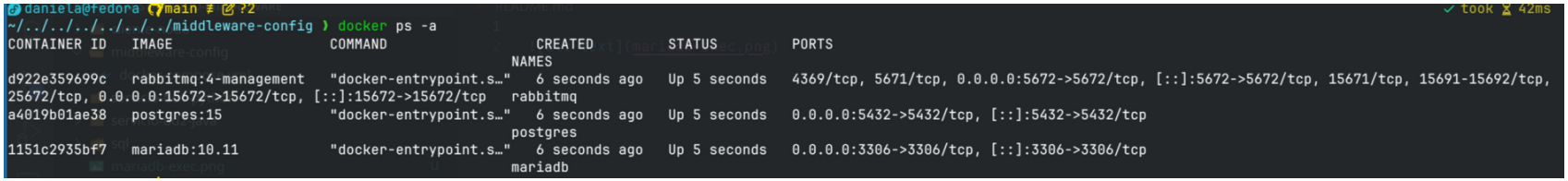

Figura 2: Containers

6. Testing and Evaluation

- **Performance Evaluation:**
  - Stress test script (1000 random entries) to evaluate performance and resilience.
- **Data Validation:**
  - Queries to DB1 (PostgreSQL) and DB2 (MariaDB) to verify correct insertion and validation logic.
- **Manual Entry:**
  - System supports interactive user registration from the client terminal.

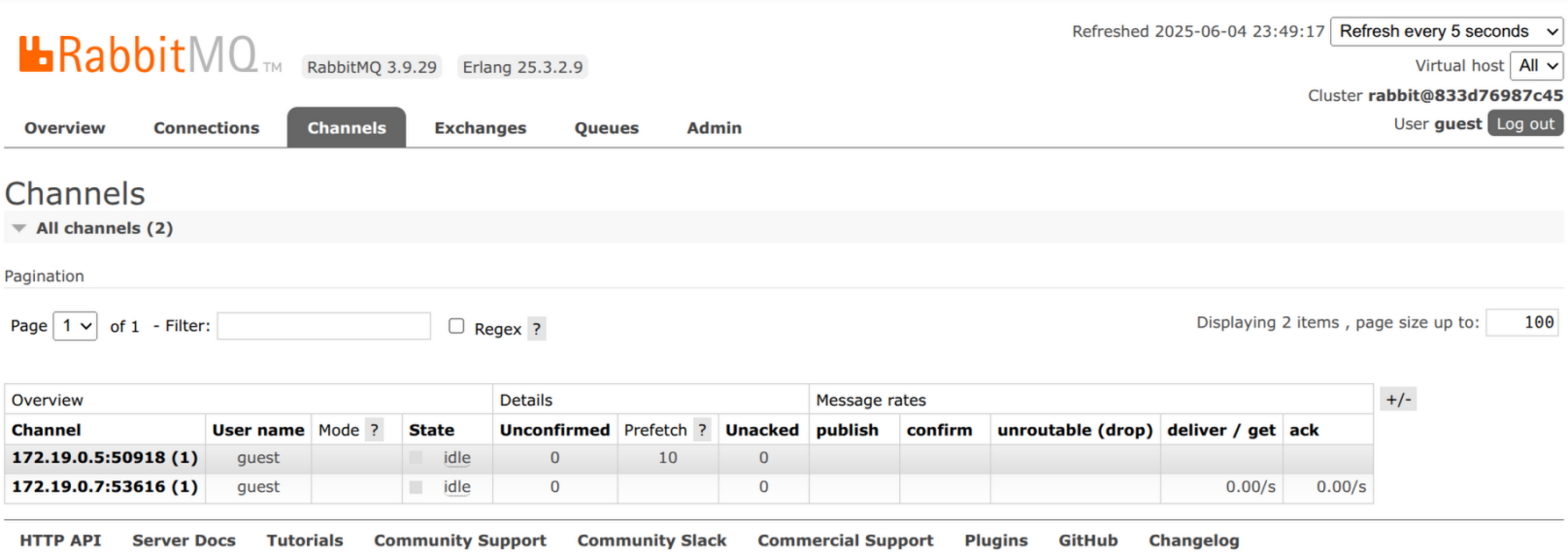

Figura 3: Channels

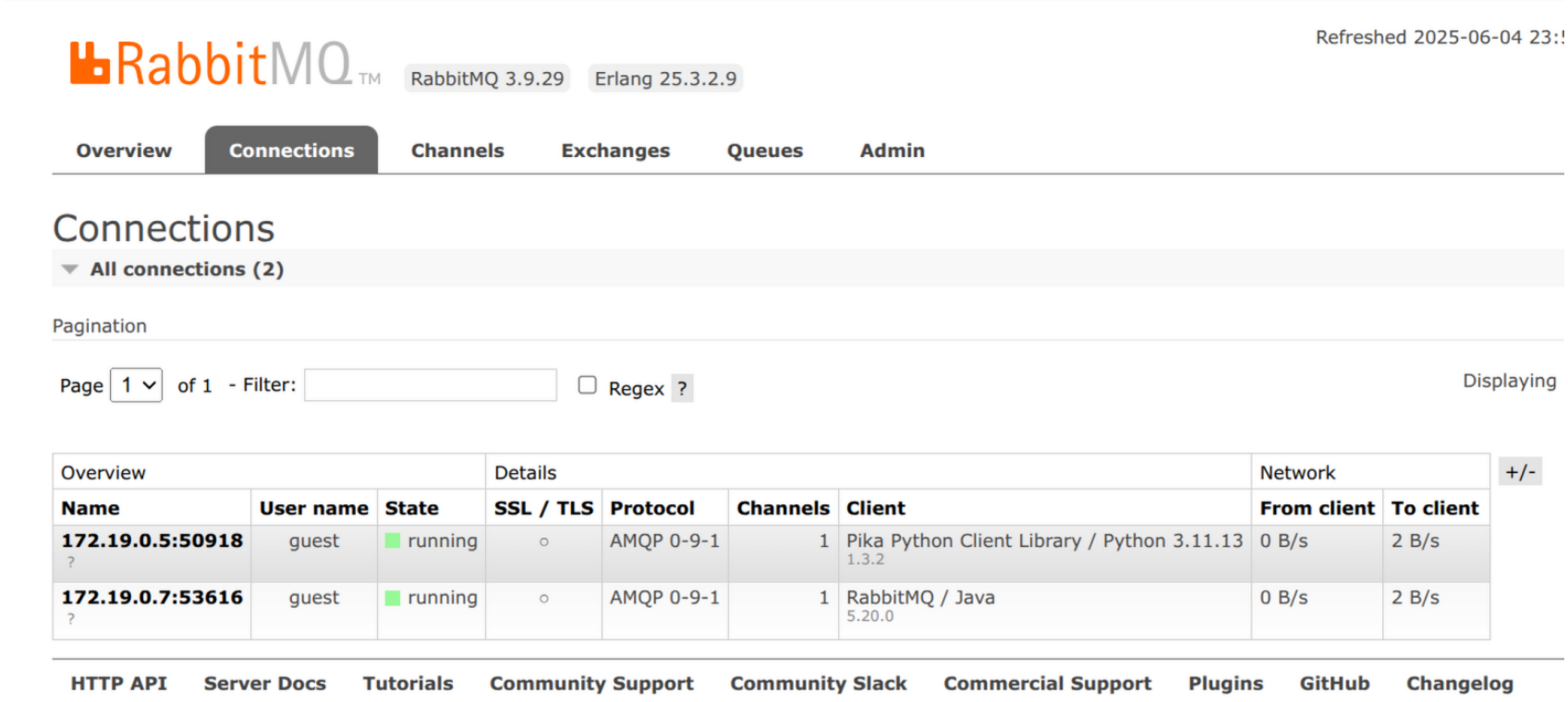

Figura 4: Connections services

Figura 5: Up Containers

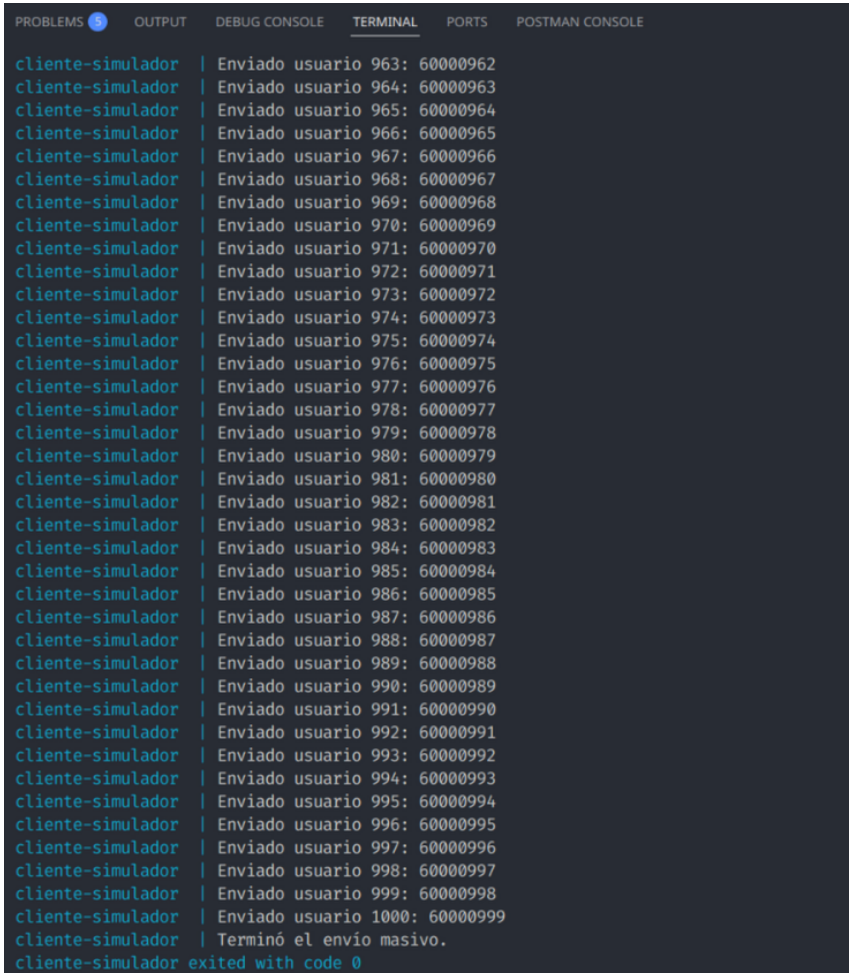

Figura 6: Send data

```
~/../../../../../sistema-distribuido-middleware ❯ docker exec -i postgres psql -U user -d bd1 -c "SELECT * FROM usuarios;"
 id |   nombre   |        correo         |      clave      |   dni    | telefono  |    amigos
----+------------+-----------------------+-----------------+----------+-----------+--------------
  1 | Maye       | usuario24@correo.com  | L07l1KtqX8syZrT | 60000024 | 900000024 | {}
  2 | Alphonso   | usuario57@correo.com  | SzEUmu5DJc21X3w | 60000057 | 900000057 | {}
  3 | Yasmin     | usuario100@correo.com | uZNoSyurJZTBu0U | 60000100 | 900000100 | {2}
  4 | Madyson    | usuario26@correo.com  | wVGXdZ7Kd4IWNEA | 60000026 | 900000026 | {}
  5 | Wyatt      | usuario108@correo.com | PGkfVYoaHYBPPdh | 60000108 | 900000108 | {3,4}
  6 | Benedict   | usuario111@correo.com | wmUXYLE7aPb9eOf | 60000111 | 900000111 | {}
  7 | Abbie      | usuario78@correo.com  | VkdNggNvimz4grg | 60000078 | 900000078 | {}
  8 | Emanuel    | usuario5@correo.com   | fShi8W2ViUg1LLW | 60000005 | 900000005 | {6}
  9 | Monica     | usuario31@correo.com  | yUENegWWhnJ3uDO | 60000031 | 900000031 | {}
 10 | Frieda     | usuario30@correo.com  | K4q8Nbd4lQzb4FB | 60000030 | 900000030 | {}
 11 | Ezequiel   | usuario118@correo.com | QVNsWwX_dncI5r2 | 60000118 | 900000118 | {2}
 12 | Josiane    | usuario35@correo.com  | WpYzhg4he5nvHoQ | 60000035 | 900000035 | {}
 13 | Caleb      | usuario28@correo.com  | Vb_ADr47xctkcyx | 60000028 | 900000028 | {}
 14 | Alexandrine | usuario10@correo.com | 40bDu8Gs1uZmJwm | 60000010 | 900000010 | {12}
 15 | Rocio      | usuario133@correo.com | vSapCL0qf2cXBtp | 60000133 | 900000133 | {}
 16 | Roberto    | usuario0@correo.com   | notEzERY1ZMJP8W | 60000000 | 900000000 | {4}
 17 | Randi      | usuario2@correo.com   | PkZEZnsuva06IaQ | 60000002 | 900000002 | {7}
 18 | Maegan     | usuario130@correo.com | w0b0H7_DXxIpJ3v | 60000130 | 900000130 | {}
 19 | Alexzander | usuario14@correo.com  | pO3PgYYF1lZiRrW | 60000014 | 900000014 | {}
 20 | Nyah       | usuario20@correo.com  | Sfzj4_MJSKH1_o8 | 60000020 | 900000020 | {11,18}
 21 | Sonia      | usuario37@correo.com  | mIQ7RMLBZFt7rKP | 60000037 | 900000037 | {16}
 22 | Adolphus   | usuario33@correo.com  | 3ZJMjwAsqhFh3lw | 60000033 | 900000033 | {9,11}
 23 | Garnet     | usuario117@correo.com | mg0KbjAR6KKv8vf | 60000117 | 900000117 | {6}
 24 | Reggie     | usuario149@correo.com | i2W_xcph9q_YTND | 60000149 | 900000149 | {20}
 25 | Lenna      | usuario34@correo.com  | T4eZWustnmjiOtJ | 60000034 | 900000034 | {13,20}
 26 | Laurianne  | usuario135@correo.com | 7T7JqQSyKY5RuTl | 60000135 | 900000135 | {8,15}
 27 | Willie     | usuario91@correo.com  | JXuBxMQHBJK1DEH | 60000091 | 900000091 | {16}
 28 | Virgie     | usuario22@correo.com  | gPwH2EhyJC9Z_ig | 60000022 | 900000022 | {}
 29 | Ines       | usuario12@correo.com  | 7YYIcgWr_2JzltW | 60000012 | 900000012 | {9}
 30 | Nickolas   | usuario23@correo.com  | nsMh7piil3bLjI4 | 60000023 | 900000023 | {}
 31 | Hubert     | usuario25@correo.com  | gsZgJ9DIGIy_gdt | 60000025 | 900000025 | {11,23}
```

Figura 7: active users

Figura 8: Validar Usuarios

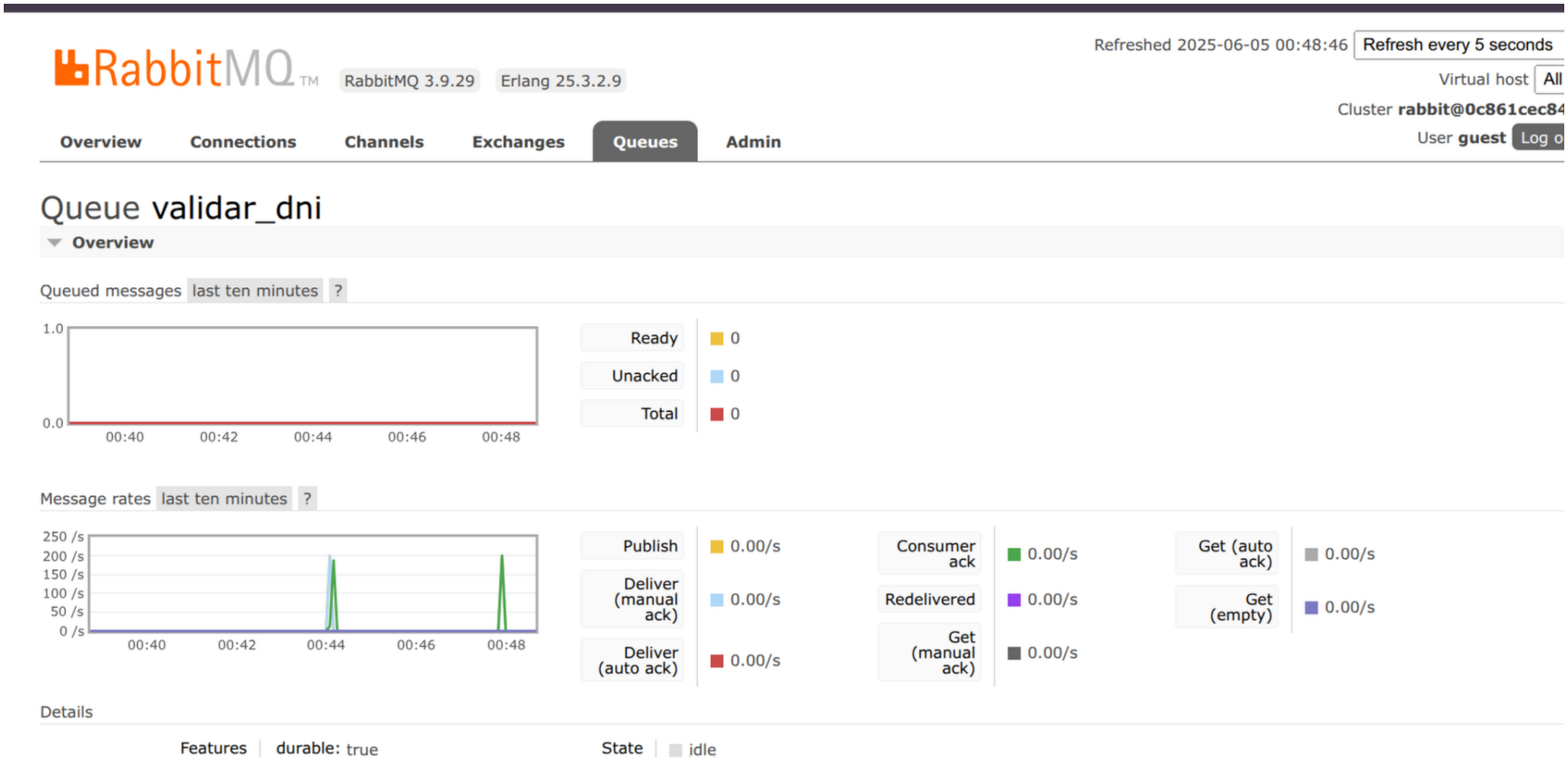

Figura 9: Queue validated DNI

7. Conclusions

- System successfully demonstrates middleware-based communication and asynchronous processing.
- Validation of real-world scenarios: data integrity, messaging, multi-threaded persistence.
- RabbitMQ fundamental for decoupling, scalability, and fault tolerance.
- Docker and Docker Compose simplified development, deployment, and management.
- Practical and robust application of concurrent programming and distributed systems concepts.