RabbitMQ Chat Application: Activity

Felix Edesa

December 2, 2024

Activity: Enhancing a Chat Application with RabbitMQ

In this activity, you will enhance a simple chat application using RabbitMQ, which includes a producer (sender) and a consumer (receiver). Complete the following tasks by modifying the provided `producer.py` and `consumer.py` code.

Task 1: Modify the `producer.py` (Sender)

1. **How can you add a timestamp to each message before sending it?**
2. **How would you ensure that the length of each message does not exceed 250 characters?**
3. **Can you modify the code to send an automatic "Goodbye" message when the user types 'quit'?**
4. **How would you add error handling to gracefully close the connection to RabbitMQ if there is any issue with sending the message?**

Task 2: Modify the `consumer.py` (Receiver)

1. **How can you display the sender's username along with the message?**
2. **Can you add a command to stop listening for messages and exit the consumer application?**
3. **How would you implement storing received messages in a log file?**
4. **Can you add a "Welcome" message after binding the queue for the consumer?**

Task 3: Bonus - Extend the System

1. **How can you implement a broadcast feature so that the producer sends a message to all users at once?**
2. **How could you add a feature that allows the consumer to retrieve a chat history of recent messages?**

Submission Requirements

- Submit the modified `producer.py` and `consumer.py` files with all of the above features implemented.
- Make sure to comment your code explaining the changes you've made and how the new functionality works.

Expected Outcomes

By completing this activity, you will:

- Learn how to interact with RabbitMQ using Python and the Pika library.
- Gain experience in modifying and extending existing code to add new functionality.
- Understand how to manage RabbitMQ queues, exchanges, and messages in a Python environment.
- Enhance your problem-solving skills by working with message-oriented middleware.

Hints for Success

- Use Python's `datetime` module to add timestamps to your messages.
- For logging, use Python's built-in `open()` function in append mode (`'a'`) to write received messages to a file.
- To implement broadcast, maintain a list of active users' queues and iterate through them when sending messages.
- For chat history, you could save each message in a list or file and allow consumers to retrieve them when needed.