

Distributed Systems

Assignment 2

**Teaching Assistant:** GabrielAntonesi **Year:** 2023 **Student Name:** Bacea Tudor Ovidiu

Abstract: A containerized Java Spring Boot listener (consumer) for a queue from a cloud instance of rabbitMQ, together with a sender comprised of a Java Spring Boot application that feeds data into the queue from a .csv file.

Contents

[1. Deployment Diagram 3](#_Toc151534854)

[2. Conceptual Architecture 4](#_Toc151534855)

[3. Project Details 5](#_Toc151534856)

[Front-End: React.js 5](#_Toc151534857)

[Back-End Microservices 5](#_Toc151534858)

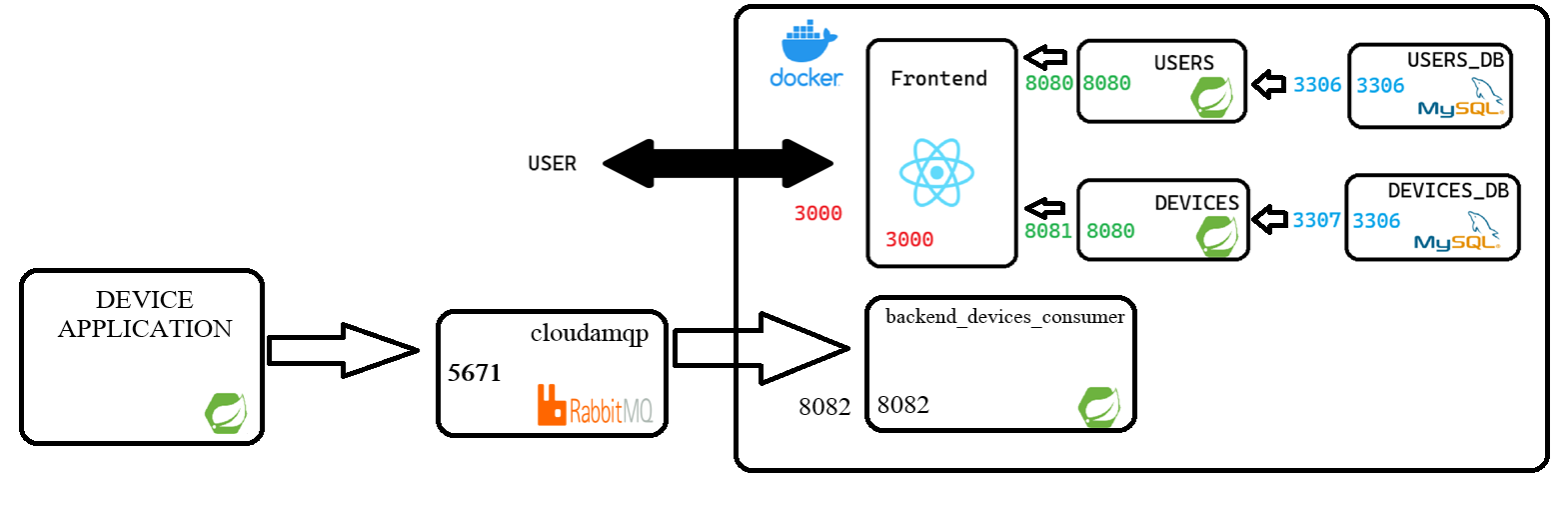
[User Management Microservice 5](#_Toc151534859)

[Device Management Microservice 5](#_Toc151534860)

[Communication Between Microservices 5](#_Toc151534861)

[Conclusion 5](#_Toc151534862)

# Deployment Diagram



# Conceptual Architecture

A screenshot of a computer screen

Description automatically generated

# Project Details

## Producer

The application is designed to interact with RabbitMQ, a message broker, by utilizing the MessageProducer component. It employs Spring Boot annotations, such as @SpringBootApplication, to configure the application. Notably, the exclusion of DataSourceAutoConfiguration.class suggests that the application is not configured to connect to a database. Additionally, the @ComponentScan annotation is used to specify the base package for component scanning, ensuring that Spring locates and manages the relevant components in the specified package (com.example.backend).

The core functionality of the application involves reading data from a CSV file located at a specified path (csvFilePath). The Apache Commons CSV library is utilized for parsing the CSV file, and each record is processed within a loop. For each record, assumed to contain a single value, a message is extracted and sent to a RabbitMQ queue using the messageProducer.sendMessageFromMain(message) method. This implies that the application serves as a message producer, generating and dispatching messages to a RabbitMQ queue. An optional delay of one second (TimeUnit.SECONDS.sleep(1)) is introduced between sending messages, potentially to control the rate of message production. Overall, this code establishes the foundation for a Spring Boot application responsible for processing CSV data and producing messages to a RabbitMQ queue.

## cloudamqp

In CloudAMQP, the queue named "my\_queue\_1" operates as a central component for sending and receiving messages. This queue, managed by CloudAMQP's RabbitMQ service, facilitates seamless communication between different components of a distributed system. Utilizing port 5671, a secure connection is established for the exchange of messages, ensuring a robust and encrypted channel for data transmission.

## Consumer

The provided Java code defines a MessageConsumer class, annotated as a Spring Component, indicating its role as a managed bean within the Spring application context. The class includes a method, annotated with @RabbitListener, specifically designed to listen for messages from the RabbitMQ queue named "my\_queue\_1." When a message is received on this queue, the listen method is triggered, printing the message content to the console. This class showcases the implementation of a message consumer in a RabbitMQ-based system, efficiently handling messages arriving on the specified queue.

# Conclusions

In conclusion, the presented system exhibits a well-structured and modular architecture for message processing utilizing RabbitMQ in a Spring Boot application. The producer component efficiently reads data from a CSV file, transforming it into messages sent to the designated RabbitMQ queue. CloudAMQP provides a robust infrastructure, managing the central message queue securely on port 5671. The consumer component, represented by the MessageConsumer class, seamlessly integrates with the RabbitMQ queue, demonstrating an effective implementation for handling incoming messages. This cohesive system showcases the synergy between the producer, CloudAMQP, and consumer components, establishing a reliable and scalable foundation for message-based communication in a distributed environment.