Queue Exercise

This task simulates that you are creating a service which receives data from a hotel reservation system (PMS). In order to simplify things a bit you can assume that the data you get from the hotel is just a simple string. No JSON or XML document etc. is necessary. It is not important WHAT you get, this task is more about HOW you deal with it.

In general, you can assume that hotels are sending reservations in real-time. Sometimes we get multiple modifications to the same reservation within a few seconds (e.g update number of persons, update stay dates, update price). That means it is important to process the incoming messages in the right order. After you get the message you need to process it. This needs to happen asynchronously as the sender won't wait until we are finished. In reality, these messages contain correlation IDs which allow us to notify the sender in an asynchronous way whether the message was processed successfully or not, but the asynchronous response is out of scope for this task.

Task

Create two services. One service contains a REST API that can receive these messages/strings. The second service is a processor that will process the incoming messages. Both services are connected by a message queue. In your case the processor doesn't need to do anything useful, just log that it was processed. Assume that multiple API services can run in parallel on different hosts, and there is only one processor instance. Please also consider the case that the processor could be unreachable for some time but you shouldn't lose messages.

What you can/should use:

Please use Java or Kotlin and the Spring Framework. You can use Spring Boot if you like but it depends on your preference. Since you are not really processing these messages, you don't have to save them to a database. But you will have to make sure that you queue them in some way so that nothing gets lost when the processor is not reachable.
Your solution doesn't have to be production ready and super-flexible. You can make meaningful assumptions and use any existing tools/libraries that help you to solve this task.

Questions

1. Create a README documenting the necessary steps to run the program.

2. In the README, please explain design decisions and assumptions you have made. Consider aspects such as Maintainability, Scalability, Performance, etc. If you decide to use external libraries, make sure to justify why you picked them.
3. Do you see any problems with this setup?
4. What kind of data does the sender's message HAVE TO contain to ensure they are imported in the correct order?
5. Are there any optimisations you see but didn't implement?

Notes

Please upload your solution to github (or other) repository and send us the link.