March 2019

1 Introduction

Message bus is a many-to-many communication mechanism between services. Message buses have become a popular architecture used behind the scene in administration systems, web-based services, and enterprise systems. A message bus is a mechanism whereby services send messages to "*channels*", like a radio channel, and other services listen to the channels they need. A service that sends message is called a **publisher** and the receivers are known as **subscribers**. A service can be a publisher or subscriber of a given channel, or it can simply ignore the channel. Messages are the entities used by systems to communicate with each other when using messaging channels. Messages flow in one direction from a sender to a receiver, as illustrated in Figure 1. Channels may be open to everyone, or they can be tightly controlled with access control list (ACL) determining who can subscribed.

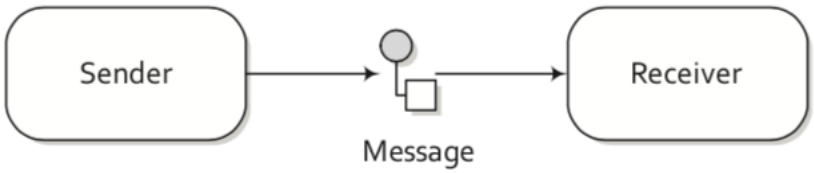

Figure 1: Messages are entities used to send data from one system to another

Apache Camel (*camel.apache.org*) is a routing engine that allows the developers to decide from which sources to accept messages, and determine how to process and send those messages to other destinations. Thus, in Camel, messages are moved around different channels, based on routes' rules.

2 Introducing the *World News* Production Process

The World News publishes in its portal articles written by journalists, opinions written by columnists, social and economic indexes produced by different agencies, and an overview of the trending topics discussed in Twitter. Its editorial production process comprises three major phases (i) content gathering, (ii) content curation, and (iii) content publication. Content gathering starts with journalists getting their stories ready and sending them electronically to the editor. In this case, each journalist creates a text file with his/her story and (s)he puts it in a directory named *inputs*. Each text file comprises a title, an author, a submission date, and a content with the text to be reviewed, as depicted in Figure 2. Likewise, a file must be named according to the following convention: author's name and the date of submission in the US date format (i.e., yyyy-mm-dd), separated by a dash (-). For

Figure 2: Excerpt of an article submitted for publication

title: House Republicans Fret About Winning Their Health Care Suit
author: Carl Hulse
date: 2016-12-31
content: Congressional Republicans have a new fear when it comes to their health care lawsuit against the Obama administration: They might win. ···

example, *carl-hulse-2016-12-31.txt*, which is the name of a file submitted by Carl Hulse on December 31^{th}, 2016.

After a story has been sent, the editors curate it, which include fixing grammar and typographical errors, rephrasing statements, or rejecting the whole story. When the story is accepted, a JSON (JavaScript Object Notation) is created and stored in a directory named *accepted* to be lately included in a database, and the rejected ones are moved to another directory named (*rejected*). The attributes of the JSON file include title, author, submission date, content, accepted date, and reviewer name.

Currently, the newspaper publishes the women speech rate of French's channel stations (*bit.ly/2UvzK1t*) and the IAS national syndrome flu (*bit.ly/2TNrYUu*) indexes, which are daily updated by their corresponding agencies. Finally, every hour, the newspaper collects the top-ten Twitter trending topics and stores them in a database.

3 Technical work

The Chief Technical Officer (CTO) of the newspaper hired you to automate the whole newspaper's production process, using Apache Camel (*camel.apache.org*), RabbitMQ (*rabbitmq.com*), and SQLite (*sqlite.org*). For now, we assume that all the texts are automatically approved. Thus your technical tasks are to:

1. Define a channel that takes each submitted article and publishes it onto a queue named *mqpending*. The message must be a JSON object with the same attributes of the input file.

2. Define a channel that listens the *mqpending* queue and that for each message, (i) generates a JSON file with title, author, submission date, content, accepted date, and reviewer name as attributes; and (ii) next, stores the it in the *accepted* directory.

3. Define a channel that every hour connects to Twitter API to collect the top-ten trending topics tweets and stores them into a SQLite database.

3.1 Deliverables

1. The code of your solution written in Java 8 or superior

2. The Unit tests showing that your code works correctly

3. The Maven file (i.e., *pom.xml*) describing the dependencies, and all the configurations to build, test, and run your solution.

3.2 Team and Grading scheme

1. You can work in group of up to two members.
2. Grading will be based on both correctness, architecture, documentation, and organization.