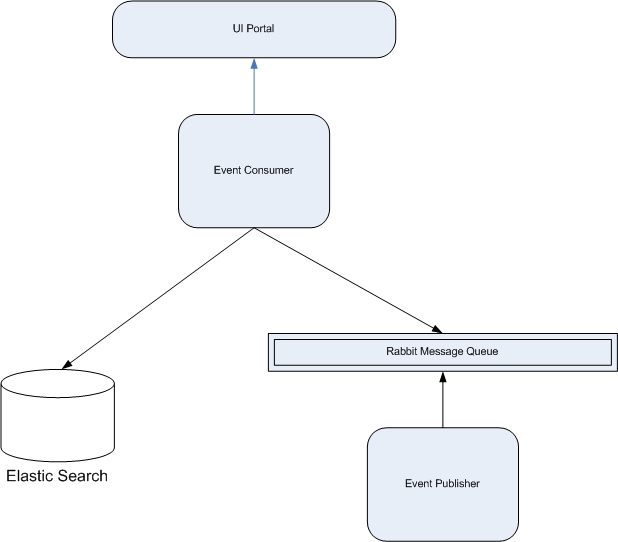
# Objective

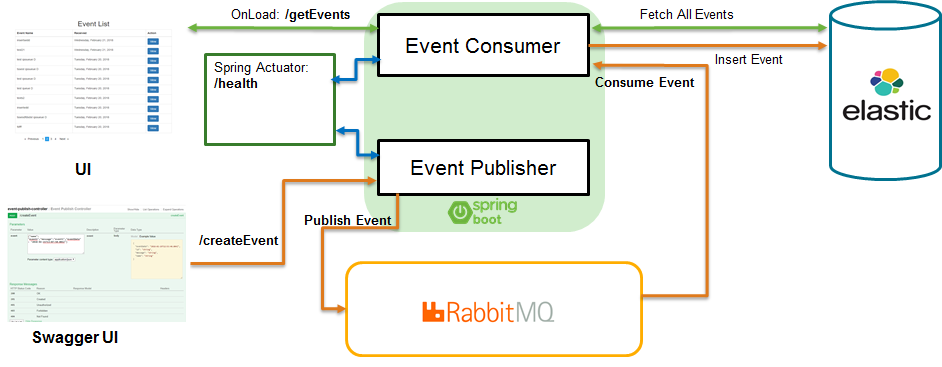
The objective of this coding exercise is to develop an event logging and real-time notification system. An Event Publisher service will represent a back-end system that will create messages and push them to a message queue. An Event Consumer will listen to the queue, store all events into database, and then notify and end-user in real-time that an event was received. It is assumed the end-user will be logged into a UI portal where notification messages will be received.

For this exercise, please develop Event Publisher, and Event Consumer components needed for this application and containerize the application as Docker Images. Standard Docker images for Rabbit MQ and Elastic Search/Mongo should be used. Finally, please provide a Docker Compose file that will bring up the entire application stack to demo and testing purposes.

The following is an architecture diagram of this system, with some requirement details and notes of each component.

# Architecture





## Event Publisher Micro Service

Spring Boot micro service

* Expose a REST API endpoint (with a Swagger UI) that will take some parameters or a request body as the contents of a message, and generate an event
* The service should be configured to connect to Rabbit MQ and bind to an exchange and routing key on startup
* The service should publish the event to Rabbit MQ
* Build and deployed as a Docker image

## Rabbit MQ

* Message queue bus handling all queue of messages between components
* Configure an Exchange with Routing Key (Publisher), and Queue (Consumer) to be used

## Event Consumer Micro Service

* Consumer will create a Rabbit MQ queue to the specified exchange and routing key listening for events
* For events received, define an Elastic Search index and store the event
* Develop a mechanism that this consumer service can connect to the UI and publish an event to the client
* Provide a REST endpoint to also list all events in a paginated way, such that the UI can display all events stored

# Requirements

* Latest release versions of software:
  + Java
  + Spring Boot
  + Spring Data
  + Rabbit
  + Elastic Search / Mongo
  + Swagger
* All services should be built as Docker images
* A Docker Compose file should be provided to bring up the stack for testing
* Swagger UI should be provided for the REST APIs
* Unit tests should be provided for all developed components, using Mockito for mocking objects.
* Version end point
* Health check end point

# Solution

|  |  |  |
| --- | --- | --- |
| * Customer   + AddCustomer   + RemoveCustomer * AutoInsuranceQuote   + AddVehicle   + RemoveVehicle   + AddDriver   + RemoveDriver | EventPublisher | ElasticSearch  Event Store |
|  | EventConsumer | MongoDB  Collections Customer & Quote |

|  |  |  |
| --- | --- | --- |
| Event Name | Action |  |
| AddCustomer | A Customer is added in Mongo DB |  |
| RemoveCustomer | A Customer is removed from Mongo DB |  |
| AddVehicle | A Vehicle is added to the Customer account | An auto insurance quote has been created |
| AddDriver | A Driver is added to the Customer account | An auto insurance quote has been updated |