



**Study and be informed about the following key terms:**

Term	Definition
Event-driven system	<i>see 00_IE-Intro.pptx</i>
Messaging system	<i>see 01_IE-Messaging.pptx</i>
Hyperautomation	<i>see 00_IE-Intro.pptx</i>
Streaming	<i>see 01_IE-Messaging.pptx</i>
Kafka Message	<i>see 01_IE-Messaging.pptx</i>
Kafka Partition	<i>see 01_IE-Messaging.pptx</i>
Kafka cluster	<i>see 01_IE-Messaging.pptx</i>
Kafka broker	<i>see 01_IE-Messaging.pptx</i>
Kafka Producer	<i>see 01_IE-Messaging.pptx</i>
Kafka Consumer	<i>see 01_IE-Messaging.pptx</i>
Kafka Consumer Group	<i>see 01_IE-Messaging.pptx</i>
Microservice	<i>see 01_IE-Messaging.pptx</i>
REST API	REpresentational State Transfer – Application Programming Interface  <i>see 05_IE-APIManagement_Other</i>
ESB	Enterprise Service Bus  <i>see 02_IE-Microservices</i>
Kafka Topic	<i>see 05_IE-APIManagement_Other</i>
SOA	Service-oriented architecture  <i>see 02_IE-Microservices</i>
Middleware	On the one hand, both a software and a DevOps engineer would describe middleware as the layer that “glues” together software by different system components; on the other hand, a network engineer would state that middleware is the fault-tolerant and error-checking integration of network connections.
Enterprise Integration	Is the process of ensuring the interaction between enterprise entities necessary to achieve domain objectives
Enterprise Interoperability	<i>see 00_IE-Intro.pptx</i>
MOM	Message Oriented Middleware  <i>see 01_IE-Messaging.pptx</i>
AMQP	Advanced Message Queuing Protocol  <i>see 01_IE-Messaging.pptx</i>
Publish/Subscribe messaging pattern	is a pattern that is characterized by the sender (publisher) of a piece of data (message) not specifically directing it to a receiver. Instead, the publisher classifies the message somehow, and that receiver (subscriber) subscribes to receive certain classes of messages.  <i>see 01_IE-Messaging.pptx</i>
Reactor pattern	Give the possibility to handle multiple concurrent requests or messages with a single thread. The reactor pattern allows associating I/O events with event handlers. Invokes the event handlers when the expected event is received. Avoiding the creation of a thread for each message, request and connection
Proactor pattern	can be seen as an asynchronous version of the reactor, it is useful when long-running event handlers invoke a continuation when they complete. Such mechanisms allow mixing nonblocking and blocking I/O

### What is it and explain in detail how does it work?

Terraform state persistency
Zookeeper and Kafka integration
The role of zookeeper
Kafka Partition
Kafka Consumer group
Kafka message key
Kafka message offset
Kafka Commit log
REST API
Processing rate between Kafka Producer and Kafka Consumer, which one has more computational consumption?
Kafka Consumer Group rebalance
Kafka replication of partitions in a cluster
How to size the number of Kafka brokers of a Kafka cluster?
How to size the number of Kafka partitions of a Kafka topic?
Kafka commit log
What are the four properties required to a system to be considered a reactive system?
Quarkus Non-blocking database with pipelining
Quarkus Mutiny
Quarkus unification of reactive and imperative models? From a programmatically point of view, is it possible to use both models in Quarkus? How to do it in the JAVA code? (annotations)
Reactor pattern
Proactor pattern

### Is it <QUESTION>? Why or why not?

<Possible to delete or update the Kafka Commit log>
<Possible to include any data type in the Kafka message key>
<Possible to store any data type in the Kafka message>
<Relevant to have a timestamp for stream processing>
<In a kafka cluster, is it possible to consume an unsynchronized message>
<In a kafka in-sync cluster, is it possible to consume an unsynchronized message from one broker>

### What are the advantages and pitfalls of?

Distributed System
File integration
Web service integration
REST APIs integration
ESB integration
Messaging Streaming versus Consumer/Producer
A file connector
Screen scraping
Kafka: Fire-and-forget versus Synchronous messages versus asynchronous messages

### What are the differences between the concepts of?

MOM: Time decoupling vs. asynchronous communications
Kafka: Fire-and-forget vs. Synchronous messages vs. asynchronous messages
Kafka broker vs. Kafka cluster
Blocking Network I/O vs. Multithread Blocking Network I/O vs. Non-Blocking Network I/O

## Reactor pattern vs. Proactor pattern

### How do you? Specify the most relevant steps to perform the following task:

Create an AWS EC2 instance?
Create an AWS EC2 image?
Change an AWS EC2 instance type?
Can you change the AWS EC2 image after launching it?
Access your AWS EC2 instance remotely? (ssh, putty, key pair, other, and network inbound)
Upload files to your AWS EC2 instance from your local computer? (scp, filezilla, key pair, other, and network inbound)
Configure Kafka to be accessed remotely? What are you changing?
Upload a file to a new EC2 instance, using terraform?
Execute a script file into a new EC2 instance, using terraform?
Specify an inbound port into a new EC2 instance, using terraform?
Specify an outbound port into a new EC2 instance, using terraform?
Specify a tag name into a new EC2 instance, using terraform?
Create multiple new EC2 instances, using terraform?

### Some examples of “What is the correct answer?” for Kafka

Integration using files and Message Oriented Middleware have one common feature: a) The same performance b) Broker based on the content of the file/message c) Publish and subscribe pattern of communication <b>d) The receiver may crash, but the communication may still occur later</b>
The main characteristics of a Message Oriented Middleware system is: a) Both the client and the servers must be running simultaneously b) The interaction is always request-reply c) The integration tolerates server crash failures <b>d) The system tolerates client failures but not cluster failures</b>
The main characteristics of a synchronous message system is: <b>a) Both the client and the servers must be running simultaneously</b> b) The interaction requires a callback c) The integration tolerates server crash failures d) The system tolerates client failures but not cluster failures
Regarding Message Oriented Middleware Publish and Subscribe: a) A Consumer subscribes a topic then it may send messages b) A Producer sends a message to a topic and all the subscriber servers are notified c) A Producer sends a message to a static list of servers defined in the MOM and all servers are notified. <b>d) A Producer sends a message to a queue associated with a topic, where all published messages can be read by the Consumers</b>
Regarding Kafka: <b>a) As in a Message Oriented Middleware, in Kafka you can send a message to a particular queue or instead to a topic</b> b) Partitions allow parallelism for the senders of events c) Topic and partitions are alternative concepts; one can send message to a topic or to a partition independently d) The reading offset is incremented by Kafka whenever a message is consumed
Regarding Kafka:

- a) As in a Message Oriented Middleware, in Kafka you can send a message to a particular consumer group
- b) Partitions allow parallelism for the senders of events
- c) Topic and partitions are alternative concepts; one can send message to a topic or to a partition independently
- d) The writing offset is incremented by Kafka whenever a message is produced

Regarding Kafka and fault tolerance mechanisms:

- a) Having multiple partitions on a Topic improves availability
- b) Brokers replicate the entire Topic to improve reliability
- c) A Partition may be replicated and updated with a primary backup protocol that assures availability
- d) The replication protocol assures that all replicas are always coherent

Regarding Kafka and consumer group:

- a) A consumer group is a set of consumers, and all those consumers can read events from all partitions
- b) Kafka management is unaware of the number of consumers in a consumer group
- c) When a consumer fails, the rebalance may create inconsistency in the stream of events consumed
- d) When a consumer fails, Kafka detects the occurrence and synchronizes the offset in all consumers

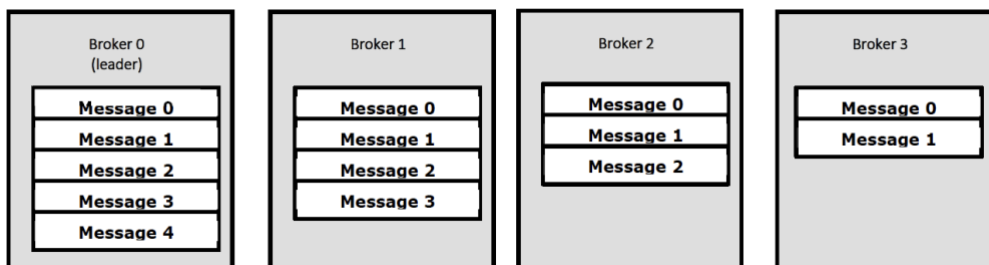
Regarding Kafka and consumer group:

- a) A consumer group is a set of consumers, and all those consumers can read events from all partitions
- b) Kafka management is aware of the number of consumers in a consumer group
- c) When a consumer fails, the rebalance may create inconsistency in the stream of events consumed
- d) A consumer within a consumer group can produce new messages

Consider that a Kafka topic has been created with 5 partitions:

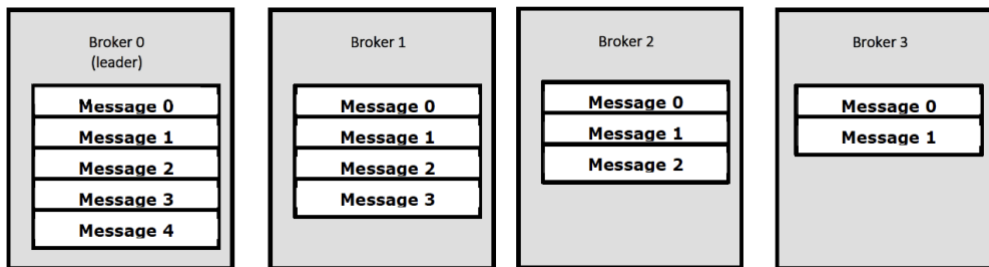
- a) A producer must know the number of partitions to send a message to the topic
- b) A consumer reading from all the partition cannot be assured of the strict order of the messages it reads
- c) The system must have 5 brokers one for each partition
- d) A consumer group reading the topic must have 5 consumers

Consider that a KAFKA cluster is configured to be all in-sync replicas mode. And a topic has one partition replicated on 4 brokers, accordingly to the figure bellow:



- a) Message 3 can be read by a consumer as it is replicated in 50% of the brokers
- b) Message 2 can be read by a consumer has it is replicated in a majority of brokers
- c) Only message 0 and 1 can be consumed
- d) For improving load distribution, a consumer can read from Broker 1

Consider that a KAFKA cluster is configured to be all in-sync replicas mode. And a topic has one partition replicated on 4 brokers, accordingly to the figure bellow:



- a) Any message can be consumed if propagated to all brokers
- b) Message 3 can be read by a consumer as it is replicated in 50% of the brokers
- c) Message 2 can be read by a consumer as it is replicated in a majority of brokers
- d) For improving load distribution, a consumer can read from Broker 1

Consider the following phrase: *"In a consumer group, the members can consume different partitions to create redundancy"*.

Do you consider this sentence correct? Justify your answer

**No, its to increase performance**

Consider the following phrase: *"Although it's possible to increase the number of partitions over time, one has to be careful if messages are produced with keys"*.

What is the reason to use a key when sending a message to Kafka? **To meta-identify the messages and to allow balancing partitions**

Why does one have to be careful if the number of partitions is increased over time.

Explain your answer. **Due to the non-sequence that is offered within the topic. Each partition key increase monotonically, but not in the overall of the partition**

Consider the following phrase: *"All partition in a topic must be read by a consumer"*.

What happens when a consumer crashes? Explain clearly what does Kafka in this situation. **The reading offset is kept to resume the consuming after a crash. This offset can be stored in kafka cluster explicitly or implicitly**

In Kafka you can have different semantics for sending a message: Fire-and-forget, synchronous send and asynchronous send. Consider the following code corresponding to sending a message to Kafka in JAVA:

```
ProducerRecord<String, String> record =
    new ProducerRecord<>("CustomerCountry", "Precision Products", "France");
try {
    producer.send(record).get();
} catch (Exception e) {e.printStackTrace();}
}
```

The example corresponds to which semantic? Justify based on the methods used and their behavior in JAVA.

**synchronous send, due to the get method that is called after sending the message**

In Kafka you can have different semantics for sending a message: Fire-and-forget, synchronous send and asynchronous send. Consider the following code corresponding to sending a message to Kafka in JAVA:

```
ProducerRecord<String, String> record =
    new ProducerRecord<>("CustomerCountry", "Precision Products", "France");
try {
    producer.send(record);
} catch (Exception e) {e.printStackTrace();}
}
```

The example corresponds to which semantic? Justify based on the methods used and their behavior in JAVA.

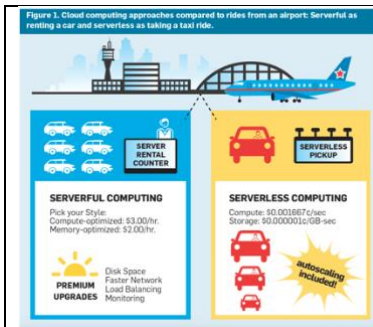
Fire-and-forget send, due to the get method that is not called after sending the message

Being informed, and an expert, about the following main key terms:

Term	Definition
PaaS	Platform As a Service
SaaS	Software As a Service
IaaS	Infrastructure As a Service
FaaS	Function As a Service
laC	Infrastructure as Code
DMN	Decision Model and Notation
BPMN	Business Process Model and notation
BPM	Business Process Management
API Gateway	Application Programming Interface gateway
Serverless computing	See 02_IE-Containers_Cloud.pptx
Microservice	See 02_IE-Microservices.pptx
REST API	REpresentational State Transfer – Application Programming Interface
Workflow systems	See 04_IE-BusinessProcesses.pptx
BPMN Activity	See 04_IE-BusinessProcesses.pptx
BPMN Gateway	See 04_IE-BusinessProcesses.pptx
BPMN Event	See 04_IE-BusinessProcesses.pptx
BPMN Pool	See 04_IE-BusinessProcesses.pptx
BPMN Lane	See 04_IE-BusinessProcesses.pptx
Camunda business key	See 04_IE-BusinessProcesses.pptx
Kubernetes	See 03_IE-Containers_Cloud.pptx
Docker	See 03_IE-Containers_Cloud.pptx
Docker image	See 03_IE-Containers_Cloud.pptx
Docker container	See 03_IE-Containers_Cloud.pptx
Kubernetes Pod	See 03_IE-Containers_Cloud.pptx

What is it and explain in detail how does it work?

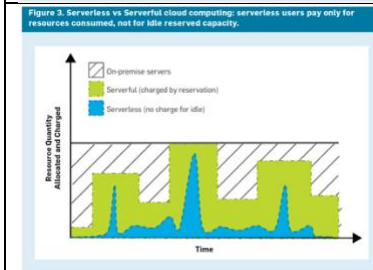
Terraform state persistency schema
REST API
AWS S3 storage service
Business process engine
Authorization of a business process engine
Access Authorization
Identity management
Business process
Camunda form
Camunda http-connector
Camunda user interaction task
Camunda service task
Correlation ID of a process using Camunda business key
Kubernetes
Serverless API invocation process. What is cold start and warm execution?



What type of computational offer from cloud providers is depicted when referring to serverful computing?

What does autoscaling means?

Compare autoscaling and the “premium upgrades” in serverful computing what the main difference between them for the end user is?



Considering the technological dependencies imposed by the following three distinct environments: on-premises servers, serverful and serverless, which were the changes that you need to do if you want to migrate JAVA code:

- From on-premise to serverful? **Adapt to the container operating system and environment + network dependencies**
- From serverful to serverless? **Comply to serverless libraries and methods + change to short living processing**

### What are the advantages and pitfalls of?

DMN versus using BPMN gateways
Pull and push for Camunda task execution
Serverless function
Using Kubernetes instead of docker compose

### What are the differences between the concepts of?

Docker image and docker container
Virtualization and containerization
Microservices framework and serverless functions
Containers and serverless functions
BPMN descriptive process and BPMN executable process
Pull and push for Camunda task execution
BPMN pool and BPMN lane

### How do you? Specify the most relevant steps to perform the following task:

What are the requirements that need to be met to expose an AWS Lambda function to the internet?
---

### Some examples of “What is the correct answer?”

<p>An application built with a microServices framework:</p> <ol style="list-style-type: none"> <li>Can communicate with other microServices using REST which is very well suited to a time decoupled pattern of communication</li> <li><b>Can communicate with other microServices using events which are very well suited to a time decoupled pattern of communication</b></li> <li>Can communicate with other microServices using shared databases</li> <li>The programming model using service synchronous invocation is more complex than event invocation</li> </ol>
<p>From the point of view of an API that is exposing services:</p> <ol style="list-style-type: none"> <li>API's are focused are focused on the governance of services</li> <li>They must be defined in REST</li> <li>SOA has exactly the same objective</li> <li><b>API's expose a business asset that has value for the owner</b></li> </ol>
<p>Serverless Functions:</p> <ol style="list-style-type: none"> <li>There is no server executing the function</li> </ol>

- b) The main objective is to reduce function administration to a minimum
- c) It is a model of execution that existed on premises and migrated to the cloud
- d) The main objective is speed of execution

Pitfalls related with Serverless functions:

- a) More difficult to have high availability
- b) Latency on cold starts
- c) Can only be used for synchronous invocations
- d) Long time execution functions are allowed

Benefits of a Business Process oriented approach:

- a) Automatically builds a service hierarchy
- b) It produces applications which are more performant than with traditional development
- c) Business analysts model the Business Processes in BPMN and then IT people need to create the executable counterparts
- d) It creates a complete application which can be integrated with legacy systems and external services

A microservice framework scope:

- a) Microservices are based on the principle that they can be orchestrated by a controller service
- b) Microservices in opposite to SOA do not have to be high granularity services
- c) An application based on microservices defines a choreography using its interfaces
- d) Microservices are highly dependent on an Enterprise Service Bus

Considerer a business service of "Selling a product" on an E-commerce application. The interface of the service is specified in JSON and uses the REST protocol. The first time a user purchases something the services requires authentication and register that internally. After that, on a second invocation, the service already has the user identity and follows for the purchase. From this simple description chose the best answer:

- a) Loosely coupled and connection oriented
- b) Tightly coupled and connection oriented
- c) Loosely coupled and connectionless
- d) Tightly coupled and connectionless

Camunda authentication and authorization. Which sentence is false?

- a) Camunda can use an external identity provider
- b) Camunda doesn't need to have an authenticated user
- c) Camunda has an internal authorization service that enforced authorization based on the role defined in BPMN
- d) Camunda can invoke external services providing authentication credentials

## Business Processes and overall integration

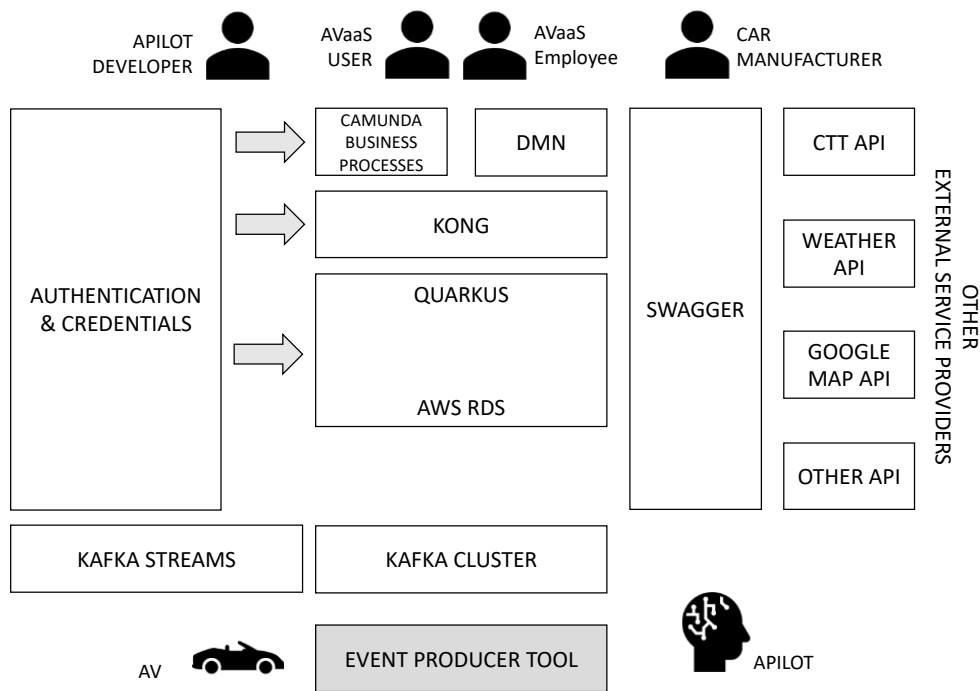


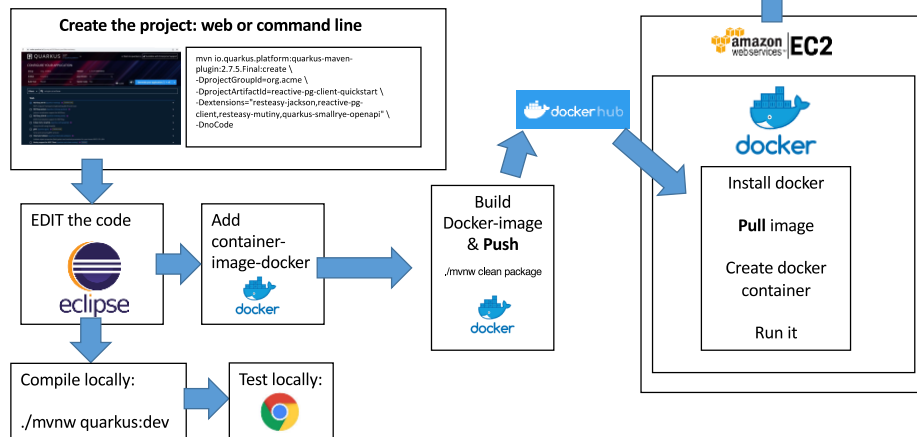
Figure 1

Consider the previous figure representing the components of the architecture of your AVaaS project.

- 1) Component characterization and classification:
  - a) What is the core role of the Quarkus component?
  - b) Identify, and explain, two benefits of using a Kafka cluster instead of using a single Kafka node?
  - c) Explain, with one example, the functionality of the Kong component?
- 2) Camunda Business Processes and DMN:
  - a) What is the role of the Camunda Business Processes?
  - b) What is the role of the DMN?
  - c) Explain the dependency existing between the Camunda Business Processes and DMN
- 3) AVaaS microservice deployment & testing:

## Docker instance inside EC2 instance by dockerhub!

Test remotely: **TÉCNICO LISBOA**



Considering the process depicted in the above figure to deploy a new Quarkus microservice in the AWS cloud environment.

- Why is Docker used? Explain the reasons referring to at least 2 advantages.
- Without Docker, what are the other alternatives to achieve the same deployment?
- What is the technology that could be used to automate the Quarkus microservices unit tests (both locally and remotely)?

## Business processes

Consider the following process model description that the **PharmacyComp** company follows when a client submits a drug prescription.

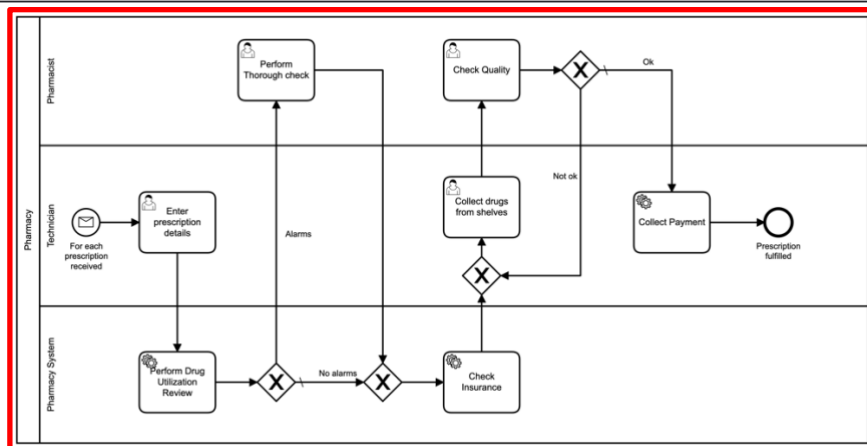
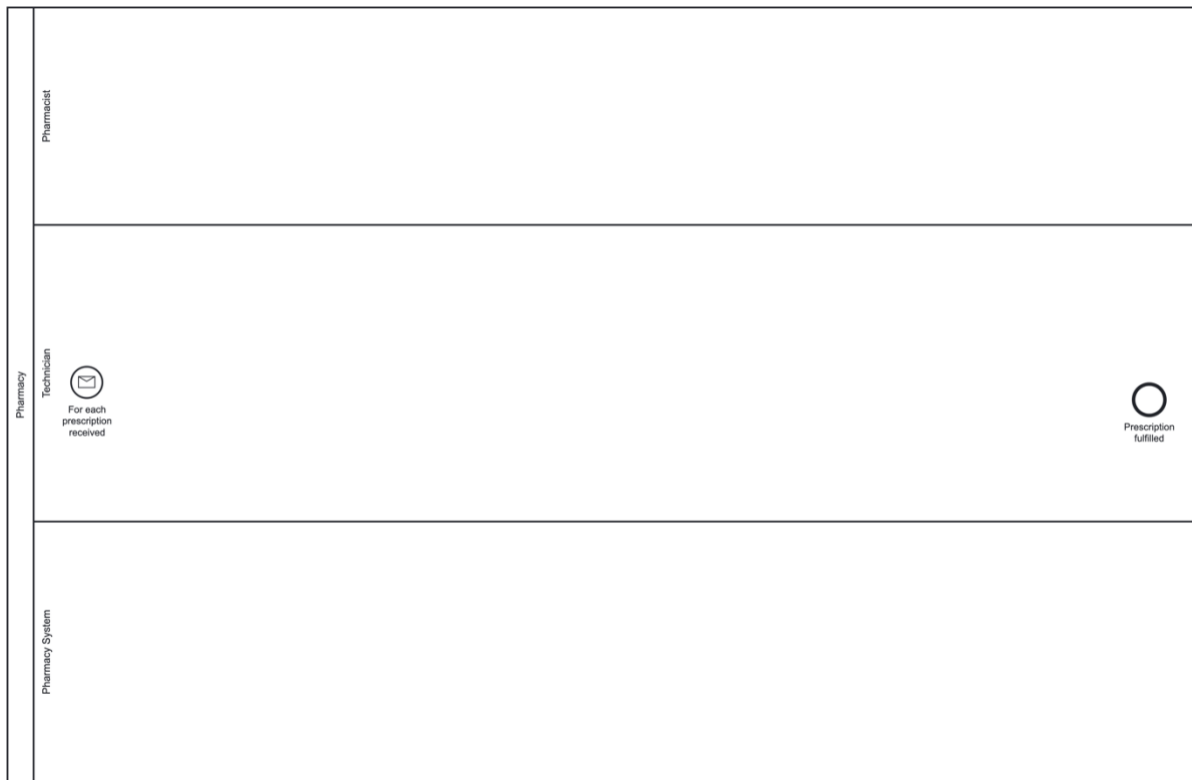
The process starts when a technician receives a new prescription.

The process interest involves three participants: Technician, Pharmacist and Pharmacy System. First, the technician enters the prescription details. Afterwards, the pharmacy system performs a drug utilization review. If an alarm is triggered, then the Pharmacist Perform a Thorough check. Otherwise, the pharmacy system checks the insurance coverage (3<sup>rd</sup> party company) to verify credit.

If drugs are covered by insurance, then the technician collect the drugs from shelves and the Pharmacist check its quality.

If drugs quality is OK the technician collects the payment and the process ends. Otherwise, technician repeats the drugs collection from shelves and the Pharmacist check its quality again.

- Design the **PharmacyComp** process using BPMN notation, for a non-executable process, and reusing the existing elements in the following pool:



2) Now, consider that the company wants to automate this process to deal with drug prescription in a more efficient manner.

a) Considering the CAMUNDA engine capabilities, explain how this part of the previous universe of discourse could be implemented?

“...the technician enters the prescription details ...”

Explain, in detail, the required steps to implement this part of the process.

b) Considering the CAMUNDA engine capabilities, explain how this part of the previous universe of discourse could be implemented?

“...the pharmacy system checks the insurance coverage (3<sup>rd</sup> party company) to verify credit ...”

Explain, in detail, the required steps to implement this part of the process.

## Business processes

Consider the following process model description that the **InsuranceComp** company follows when a client submits an insurance claim.

The process starts when a customer submits a new insurance claim.  
Each insurance claim goes through a two-stage evaluation process.

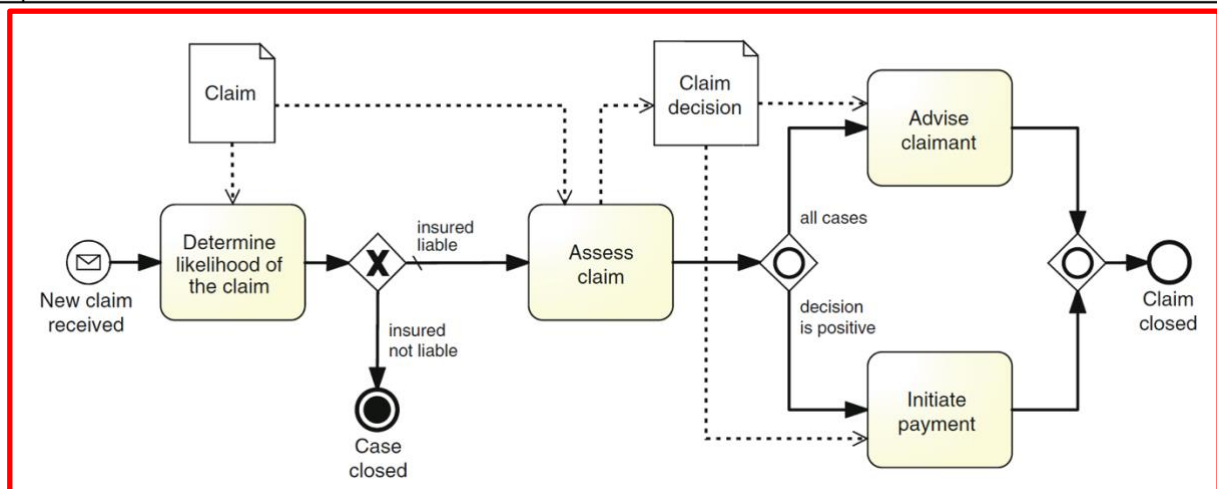
First of all, the liability of the customer is determined. Secondly, the claim is assessed in order to determine if the insurance company has to cover this liability and to what extent.

If the claim is accepted, payment is initiated, and the customer is advised of the amount to be paid. All tasks except "Initiate Payment" are performed by claim handlers. There are three claim handlers.

The task "Initiate Payment" is performed by a financial officer.

There are two financial officers.

- 3) Design the **InsuranceComp** process using BPMN notation, for a non-executable process, and reusing the existing elements in the following pool:



- 4) Now, consider that the company wants to automate this process to deal with claims in a more efficient manner.

- 1) Considering the CAMUNDA engine capabilities, explain how this part of the previous universe of discourse could be implemented?

"...There are three claim handlers..."

Explain, in detail, the required steps to implement this part of the process.

- 2) Considering the CAMUNDA engine capabilities, explain how can the

"...the liability of the customer is determined ..."

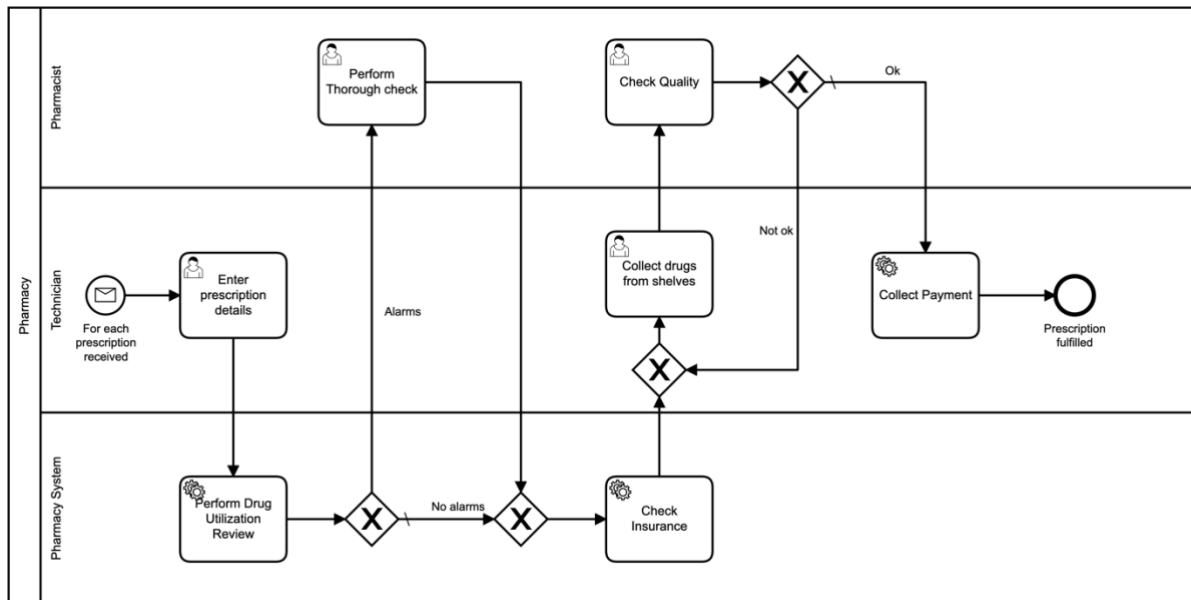
be implemented?

Explain, in detail, the required steps to implement this part of the process.

- 3) Considering the CAMUNDA engine capabilities, propose two different implementation options to inform the client about the **InsuranceComp'** decision?  
Explain, in detail, the required steps to implement both options.

### Executable BPMN in Camunda

Consider the following process model description that the **PharmacyComp** company follows when a client submits a drug prescription.



Consider that this company wants to automate this process to deal with prescriptions in a more efficient manner using the CAMUNDA engine capabilities.

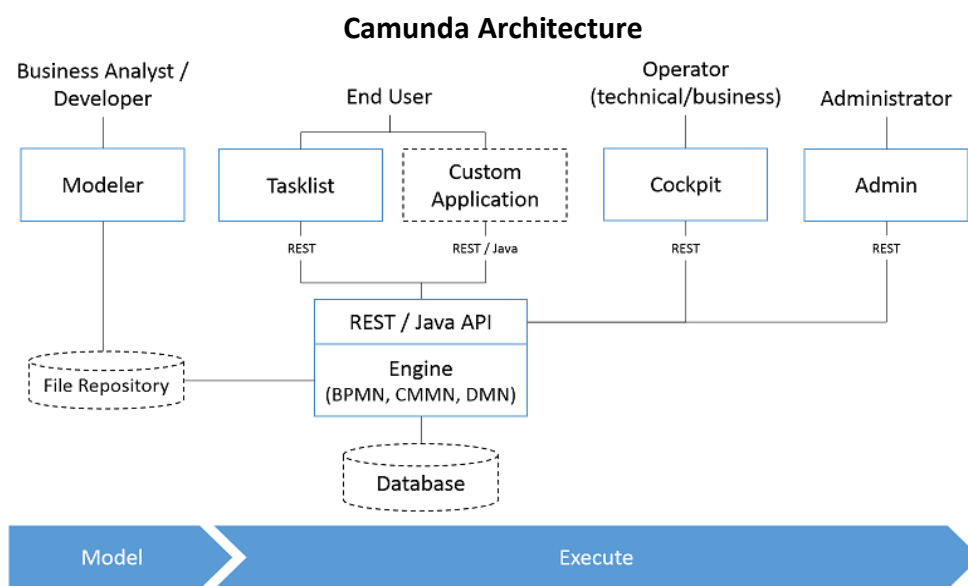
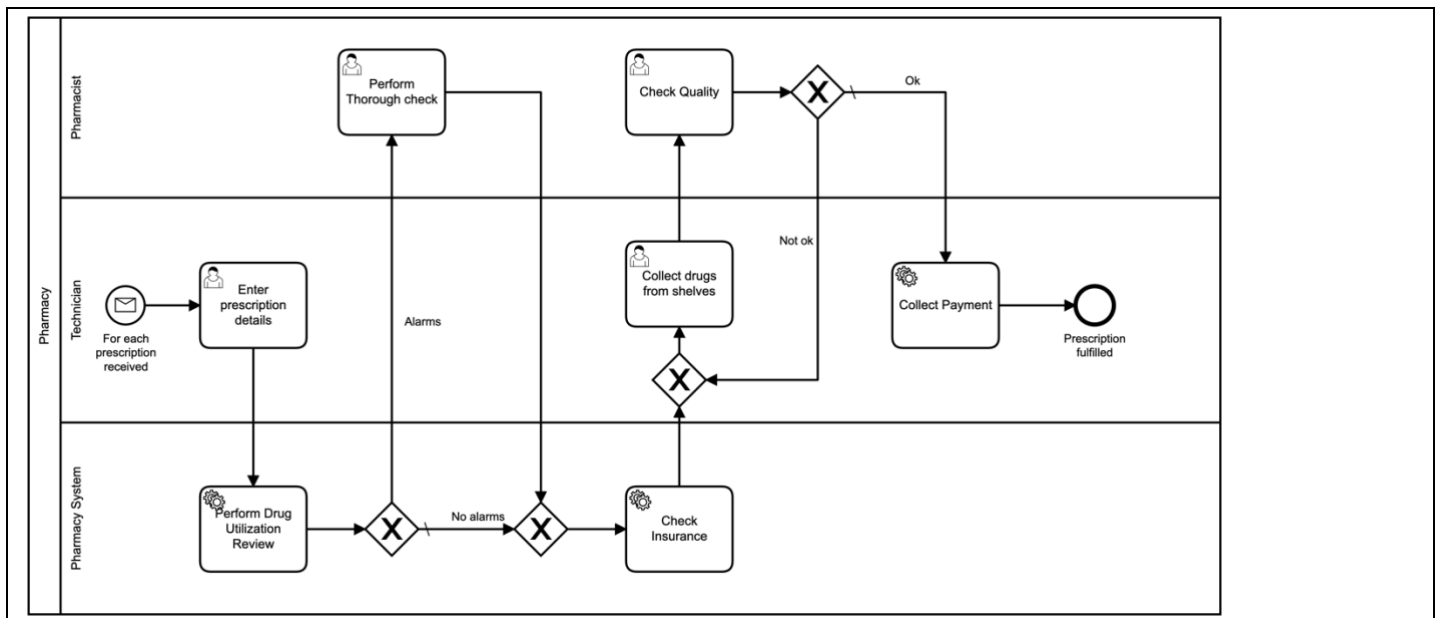
- 4) Explain and discuss, in detail, the two alternative solutions (Push and Pull) to implement the “Collect Payment” Task?



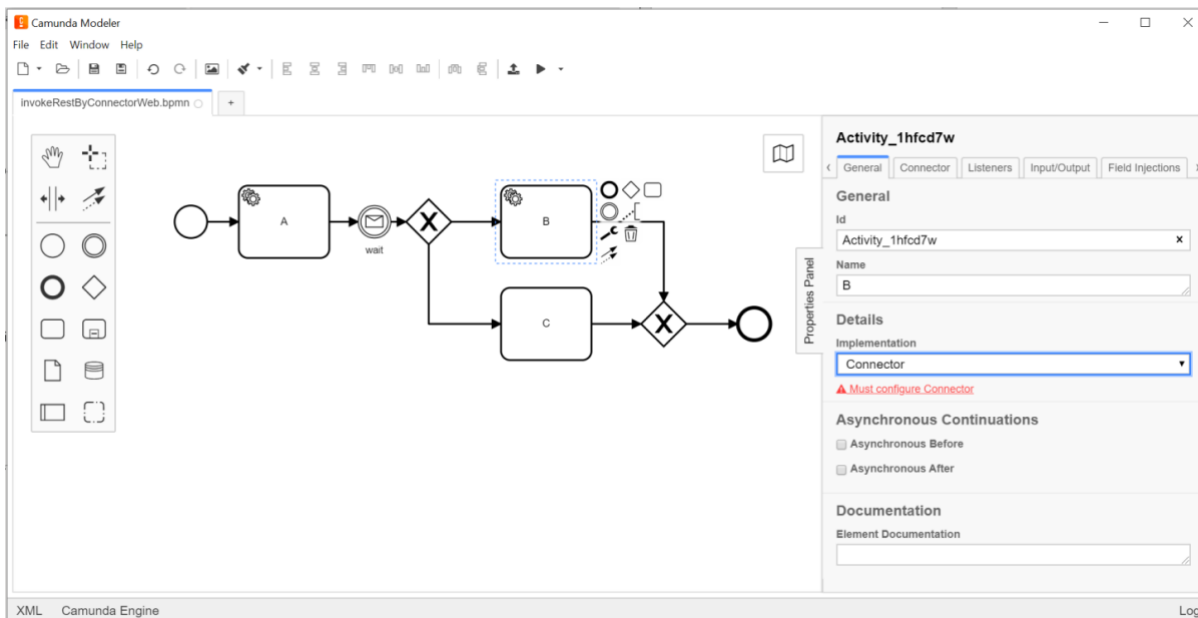
- 5) Describe, in detail, two technological alternatives that are available to trigger the instantiation of this process model, as depicted in the next figure.



- 6) Propose, in detail, a change in the process model to inform the client about the **PharmacyComp'** process result execution. You can design the change directly in the business process model and/or explain it textually.



- 1) Considering the CAMUNDA architecture depicted in the previous Figure, identify the component (or the components) that is (or are) responsible to execute the instances of a business process?
- 2) Consider that multiple instances of a given business process, at the same time, are required to be executed. This is a functionality supported by CAMUNDA. What is the CAMUNDA mechanism that can differentiate each instance of the given business process? Explain with an example how it works.
- 3) Considering the Camunda architecture, what is the difference between an *“assignee”* and a *“candidate groups”* defined within the context of a BPMN task? In which situation should you use one or the other?
- 4) Considering the following figure, where task B is being implemented as a http-connector, what are the mandatory configurations to enable such connectivity?



## BPMN versus DMN

Consider the following DMN decision table:

Meal				
Hit Policy: Unique				
	When	And	Then	
	Season	Number of guests	Meal	Annotations
	string	string	string	
1	"Spring"	<= 4	"Green asparagus"	
2	"Spring"	5	"White asparagus"	
3	"Spring"	[6..8]	"Spinach"	
4	"Spring"	>= 9	"Pasta"	
5	not("Spring")	-	"Lasagne"	
+	-	-		

- 1) Design an equivalent BPMN model using gateways, that is able to perform exactly the same behavior.
- 2) Discuss advantages and disadvantages of using a DMN decision table instead of using a BPMN model with gateways.
- 3) How can a DMN decision table be linked with a BPMN process using CAMUNDA? Give an example.