# Camunda BPMN

Sahar Khmissi

August 25, 2023

## Contents

# 1 Introduction

In today's fast-paced business environment, organizations strive to optimize their operations, improve efficiency, and enhance productivity. To achieve these goals, they need effective tools and technologies to automate and streamline their business processes. This is where workflow engines come into play.

# 2 Workflow and workflow engines

## 2.1 Workflow

A workflow process is a set of steps or tasks that are followed to complete a specific process or job within an organization. Workflows serve as the foundation for streamlining and automating business processes, allowing organizations to improve operational efficiency, enhance productivity, and achieve higher levels of quality and consistency. Here's an example of what a workflow might look like:

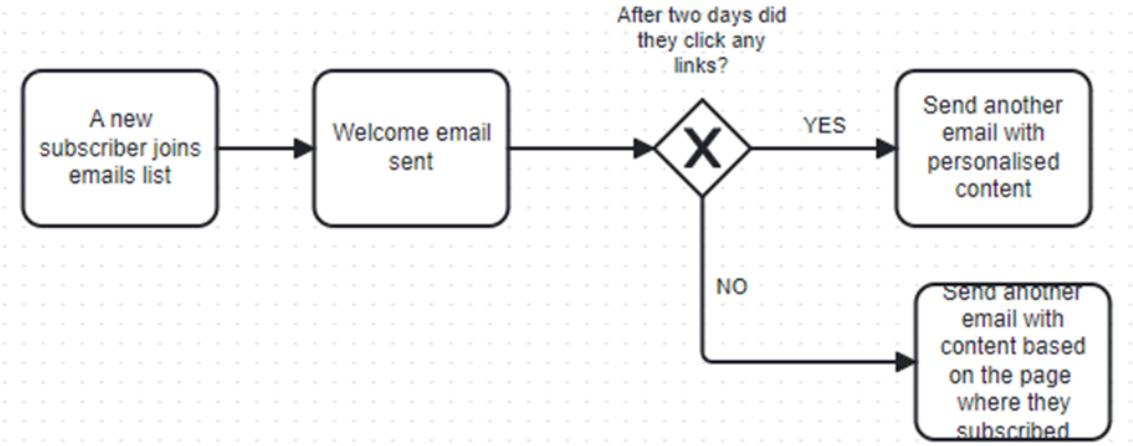


Figure 1: A Workflow Example

1-A new subscriber joins your email list.
2- A pre-written welcome email is sent to that subscriber.
3- After two days, check to see if any of the links in your welcome email were clicked.
4- Send another email with content personalized to their interests based on the links they clicked.

## 2.2 Workflow engine

A workflow engine is a software component or platform that enables organizations to design, automate, and manage their workflows. It acts as the backbone of business process management (BPM), providing a structured and systematic approach to executing tasks, managing resources, and coordinating activities within an organization.

### 2.3 Comparative analysis of workflow engines

| Workflow engines | Advantages | Disadvantages |
|---|---|---|
| **Elsa** | • Support for both short and long workflows<br>• Correlation (possibility of integrating external systems) | No support for BPMN |
| **Windows Workflow Foundation (WF)** | • Messaging activities<br>• Support for long workflows | Support only for Microsoft products |
| **Camunda** | • Cross-platform<br>• Real-time collaboration (comments, presentations, sharing,)<br>• Quick implementation of forms with Forms Builder | Basic Forms |

Table 1: Comparative analysis of workflow engines

## 3 BPMN (Business Process Modeling and Notation)

BPMN is a visual modeling language for business analysis and enterprise process workflows. It is an XML documentation that has a visual presentation. This duality makes BPMN very powerful. The XML document contains all the necessary information to be interpreted by workflow engines and modeling tools like Zeebe. At the same time, the visual representation contains just enough information to be quickly understood by humans, even when they are non-technical people. The BPMN model is source code and documentation in one artifact.

### 3.1 Tasks: Units of work

Tasks represent the individual steps or activities that need to be performed within a business process.

### 3.1.1 Service task

A service task represents a work item in the process with a specific type. When a service task is entered, a corresponding job is created. The process instance stops here and waits until the job is complete.


Figure 2: Service Task Representation

Here is an example of using a service task :

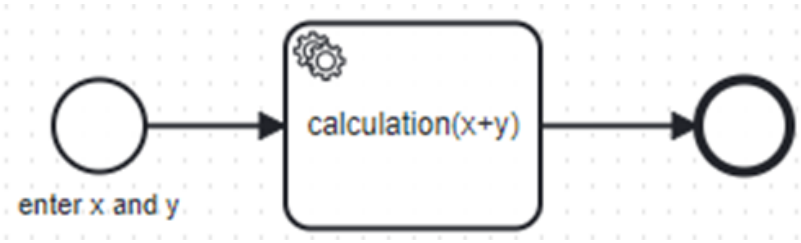


Figure 3: Calculation Service Task

The process will begin by requesting two numbers x and y. These numbers will be transferred to a job service (external system, java class, expression,) to calculate their sum.

### 3.1.2 User Task

A user task is a task performed by a human participant whose the task is assigned to. When the process instance arrives at such a user task, a new job similar to service task is created.
The process instance stops at this point and waits until the job is completed.


Figure 4: User Task Representation

### 3.1.3 Send Task

A send task is used to model the publication of a message to an external system; for example, to a Kafka topic or a mail server. When a process instance enters a send task, it creates a corresponding job and waits for its completion. When the job is complete, the process instance continues.


Figure 5: Send Task Representation

### 3.1.4 Receive Task

A receive task references a message. It is used to wait until a proper message is received. When a receive task is entered, a corresponding message subscription is created. The process instance stops at this point and waits until the message is correlated.


Figure 6: Receive Task Representation

### 3.1.5 Script Task

A script task is an automated activity. When a process execution arrives at the Script task, the corresponding script is executed.

### 3.1.6 Manual Task

A manual task that is external to the BPM engine. It is used to inform the process that another task is being executed in an external system.
This is used to model work that is done by somebody who the engine does not need to know of and there is no known system or UI interface.


Figure 7: Script Task Representation

For the engine, a manual task is handled as a pass-through activity, automatically continuing the process at the moment the process instance arrives.


Figure 8: Manual Task Representation

#### 3.1.7 Business Rule Task

A business rule task synchronously execute one or more rules.
When the process instance arrives at a business rule task a decision is evaluated using the internal DMN (Decision Model and Notation) engine. Once the decision is made, the process instance continues.


Figure 9: Business Rule Task Representation

## 3.2 Gateways: Steering flow

Gateways are elements that route tokens in more complex patterns than plain sequence flow.

### 3.2.1 Exclusive gateways

An exclusive gateway (or XOR-gateway) allows to make a decision based on data (i.e. on process instance variables).
If an exclusive gateway has multiple outgoing sequence flows, all sequence flows except one must have a condition Expression to define when the flow is taken. The gateway can have one sequence flow without condition Expression, which must be defined as the default flow.

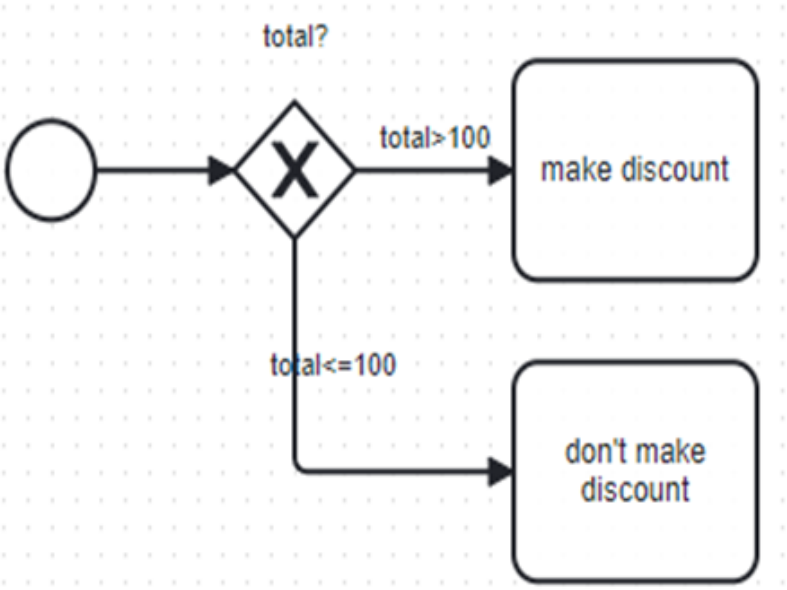


Figure 10: Exclusive Gateway Representation

In this example, a decision is made is based on the "total". If the total is greater than "100", a discount will be made. This flow (total greater than 100) has the following expression "$total > 100$".

### 3.2.2 Parallel gateways

Parallel gateway (or AND-gateway) allows you to split the flow into concurrent paths.

When a parallel gateway with multiple outgoing sequence flows is entered, all flows are taken. The paths are executed concurrently and independently. In this example, the two paths "1" and "2" will be taken.

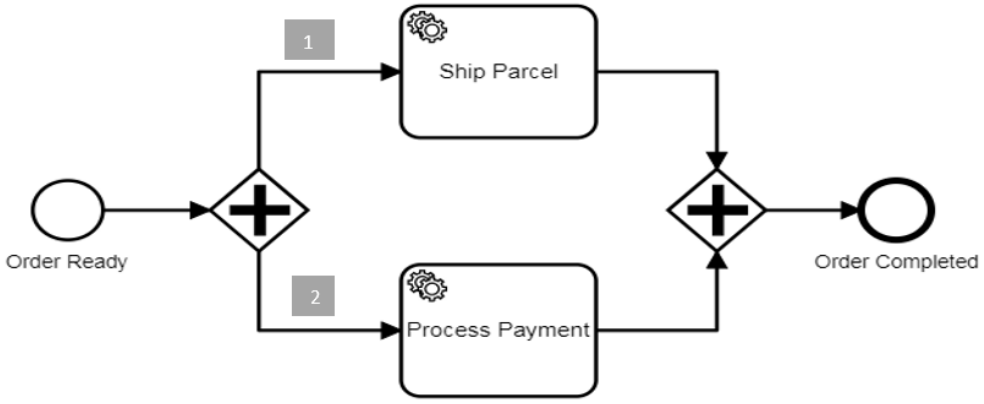


Figure 11: Parallel Gateway Representation

### 3.2.3 Inclusive gateways

An inclusive gateway will activate one or more branches based on the formula within each specific flow.

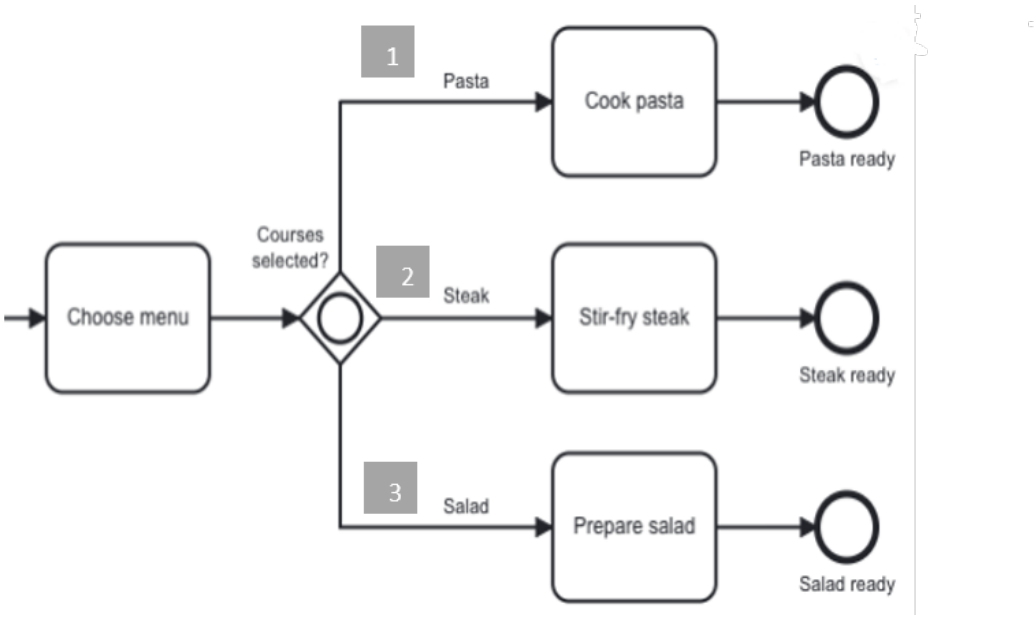


Figure 12: Inclusive Gateway Representation

For example: if the selected courses include pasta and salad, the process instance with take both the two paths “1” and “3”.

## 3.3 Events: Waiting for something to happen

An event is something that happens during the course of a process.

### 3.3.1 Start Events

Start events are used to denote the beginning of a process or subprocess.

### 3.3.2 End Events

End events are used to denote the end of a particular sequence flow.


Figure 13: Start/End Event Representation

### 3.3.3 Message Events

Message events are events which reference a message; they are used to wait until a proper message is received.
There is four types of message events:

- Message Start Event : Message event can be used to start process instances.
- Message Intermediate Catch Event: This event "catches" a specific message during the execution of a process. It waits for the specified message to be received.
- Message Intermediate Throw Event: This event is used to send a specific message during the execution of a process.
- Message End Event: This event indicates the end of a process instance and sends a message as a result.

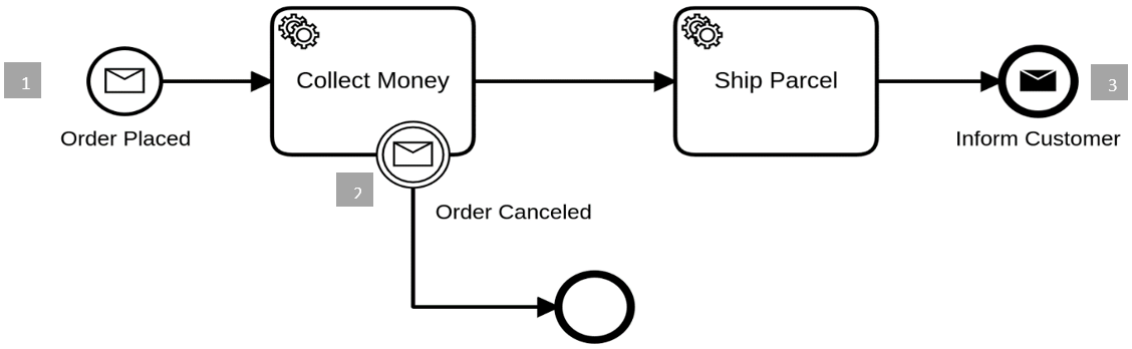


Figure 14: Message Event Representation

The above process will begin after receiving a message informing the system that a new order is placed. The client can stop the process by canceling the order.
In this example, we have three types of message events:
1- Message Start Events
2- Message Intermediate Catch Event
3- Message End Event

### 3.3.4 Signal Events

A signal event in BPMN represents the receipt of a specific signal that triggers a process or a process instance. It is used to indicate a communication or notification between different processes or participants. There is four types of signal events:

- Signal Start Event :This event indicates the start of a process instance upon receiving a specific signal.
- Signal Intermediate Catch Event: This event "catches" a specific signal during the execution of a process. It waits for the specified signal to be received and then triggers the subsequent activities in the process.
- Signal Intermediate Throw Event: This event is used to send a specific signal during the execution of a process.
- Signal End Event: This event indicates the end of a process instance and sends a signal as a result.

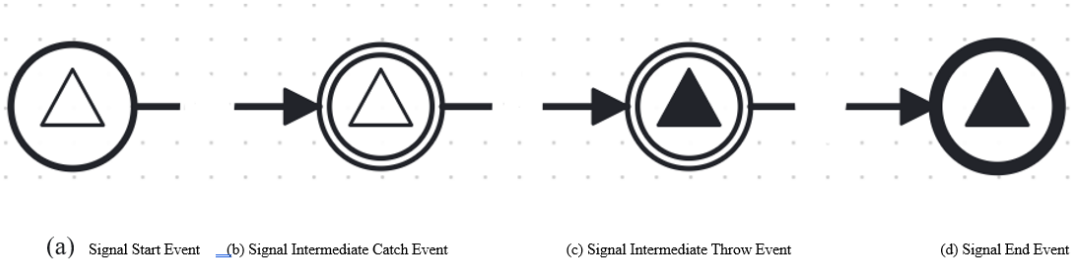


Figure 15: Signal Event Representation

### 3.3.5 Timer Event

A timer event in BPMN represents the occurrence of an event at a specific time or after a specific duration. It is used to model time-based triggers within a process flow.
There are three types of timer events:

- Timer Start Event: It triggers the start of a process instance after a specified amount of time has elapsed. This type of timer event is used to delay the start of a process.
- Timer Intermediate Catch Event: It waits for a specified amount of time during the execution of a process instance before triggering an associated activity or event. It can be used to introduce time-based waiting or timeouts within a process.
- Timer Boundary Event: It is attached to an activity and defines a time limit for the completion of that activity. If the time limit is exceeded, the boundary event is triggered, allowing for escalation or interruption of the activity.

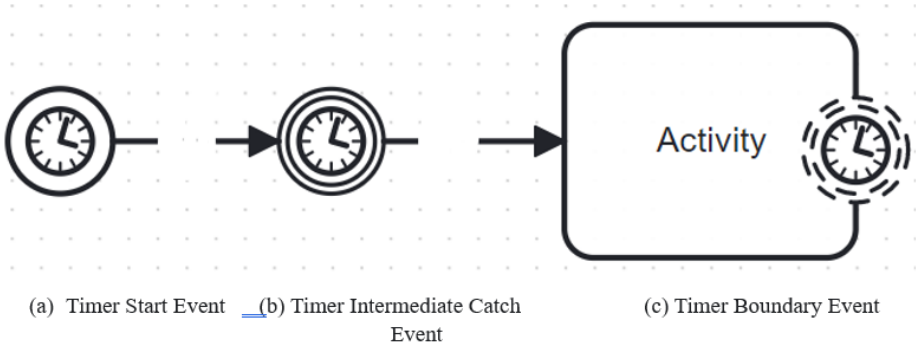


Figure 16: Timer Event Representation

Here is an example of a process using timer events :

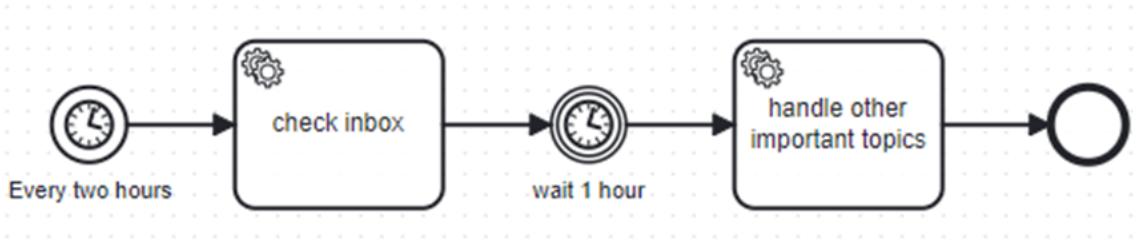


Figure 17: Timer Event Example

In the figure, every two hours a new process instance will begin. After checking the inbox, the process will wait for one hour and resume.

### 3.3.6 Error Event

In process automation, you often encounter deviations from the default scenario. One way to resolve these deviations is using a BPMN error event, which allows a process model to react to errors within a task.
The two types of error events are :

- Error Boundary Event: The Error Boundary Event is used to handle errors within the context of a specific activity. It is attached to a boundary of an activity and represents an alternative path in case of an error occurrence.
- Error End Event: The Error End Event is used to explicitly indicate the end of a process due to an error condition.


Figure 18: Error Event Representation

For example, if an invalid credit card is used in the process below, the process takes a different path than usual and uses the default payment method to collect money.

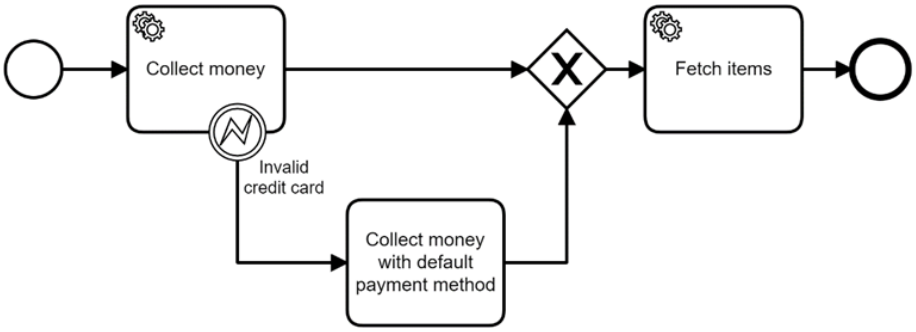


Figure 19: Error Event Example

### 3.3.7 Link Event

Link events are intermediate events that connect two sections of a process.

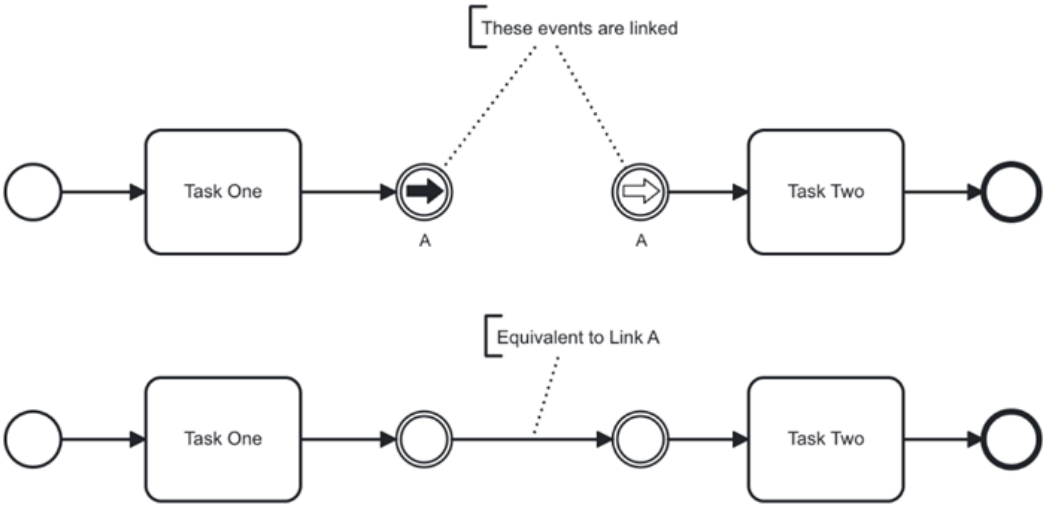


Figure 20: Link Event Representation

Link events have a throwing link event as the "exit point", and a catching link event as the "re-entrance point". They are linked together by their link name.

# 4 Project Setup Guide

Before you start working on this project, please ensure you have the following prerequisites installed and configured:

## 4.1 Camunda Modeler V5.12.0

You can download the Camunda Modeler from the official website: https://camunda.com/download/modeler/
Open the BPMN file in the API "CamundaModels/PostingProcess.bpmn" in the desktop modeler.

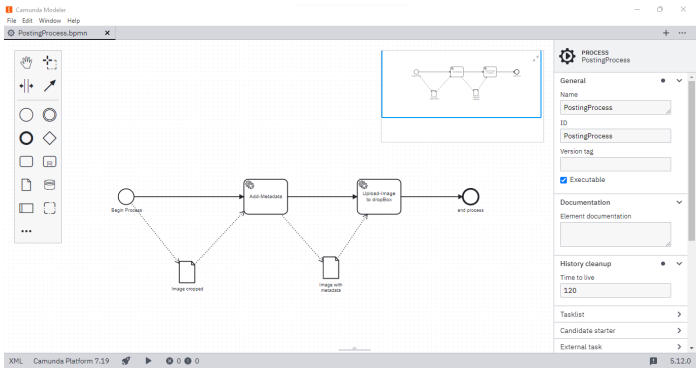

Figure 21: Camunda Modeler

## 4.2 Camunda Platform Run V7.19.0

You need to have the Camunda BPM Engine installed and running. For installation and execution instructions, please visit https://docs.camunda.org/manual/7.19/installation/camunda-bpm-run/
Once the engine is installed and running you can access the Camunda platform run interface via http://localhost:8080/camunda/app/.

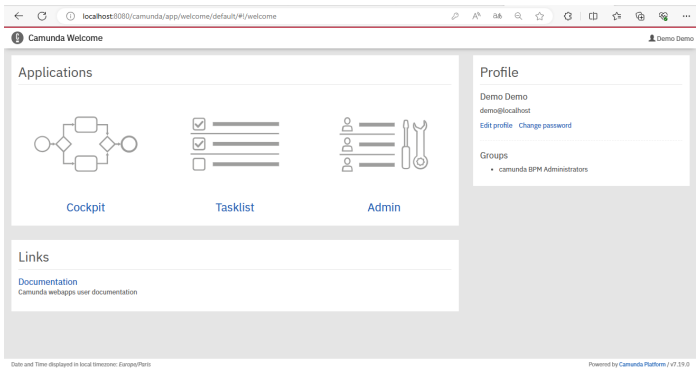


Figure 22: Camunda Platform Run

## 4.3 Workflow Process Deployment

You need to deploy your process model opened in the camunda modeler on the camunda platform run.

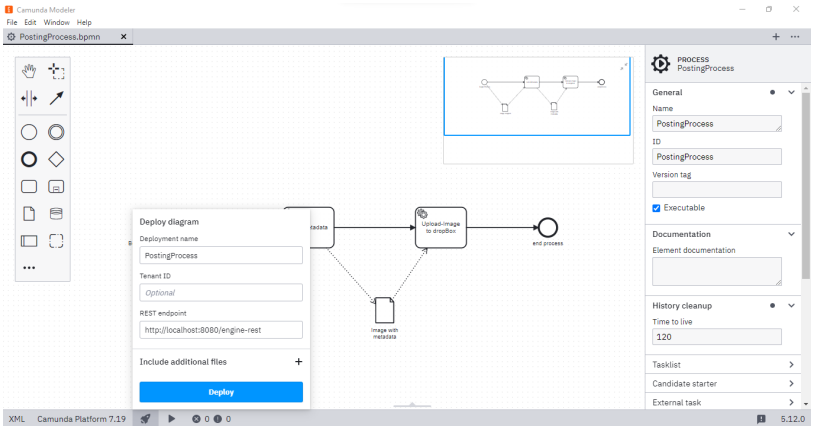


Figure 23: Deployment

Click the "Deploy" button to initiate the deployment process. The platform will process the BPMN file, create the necessary database entries, and make the process available for execution.

## 4.4 ReactJs V18.2.0

Install React and the required packages for the ReactJs.

- axios: npm install axios
- react-bootstrap : npm install react-bootstrap -
- react-easy-crop : npm install react-easy-crop

## 4.5 .NET V6.0

These are the required libraries in the API

- CamundaClient Library : allows communication with the Camunda BPM Engine.
- DropBox.Api Library : used for interacting with the Dropbox API.
- System.Drawing.Common Library : provides image processing capabilities.
- Microsoft.EntityFrameworkCore Library

## 4.6 Local Storage

Ensure you have the folowing folders on your local storage :

- C:/automate-posting/ProductsCropped/Facebook
- C:/automate-posting/ProductsCropped/Instagram
- C:/automate-posting/ProductsCropped/Twitter
- C:/automate-posting/ProductsMD/Facebook
- C:/automate-posting/ProductsMD/Instagarm
- C:/automate-posting/ProductsMD/Twitter

# 5 Get Started

## 5.1 Project Flow Representation

This is the flow diagram of the project :

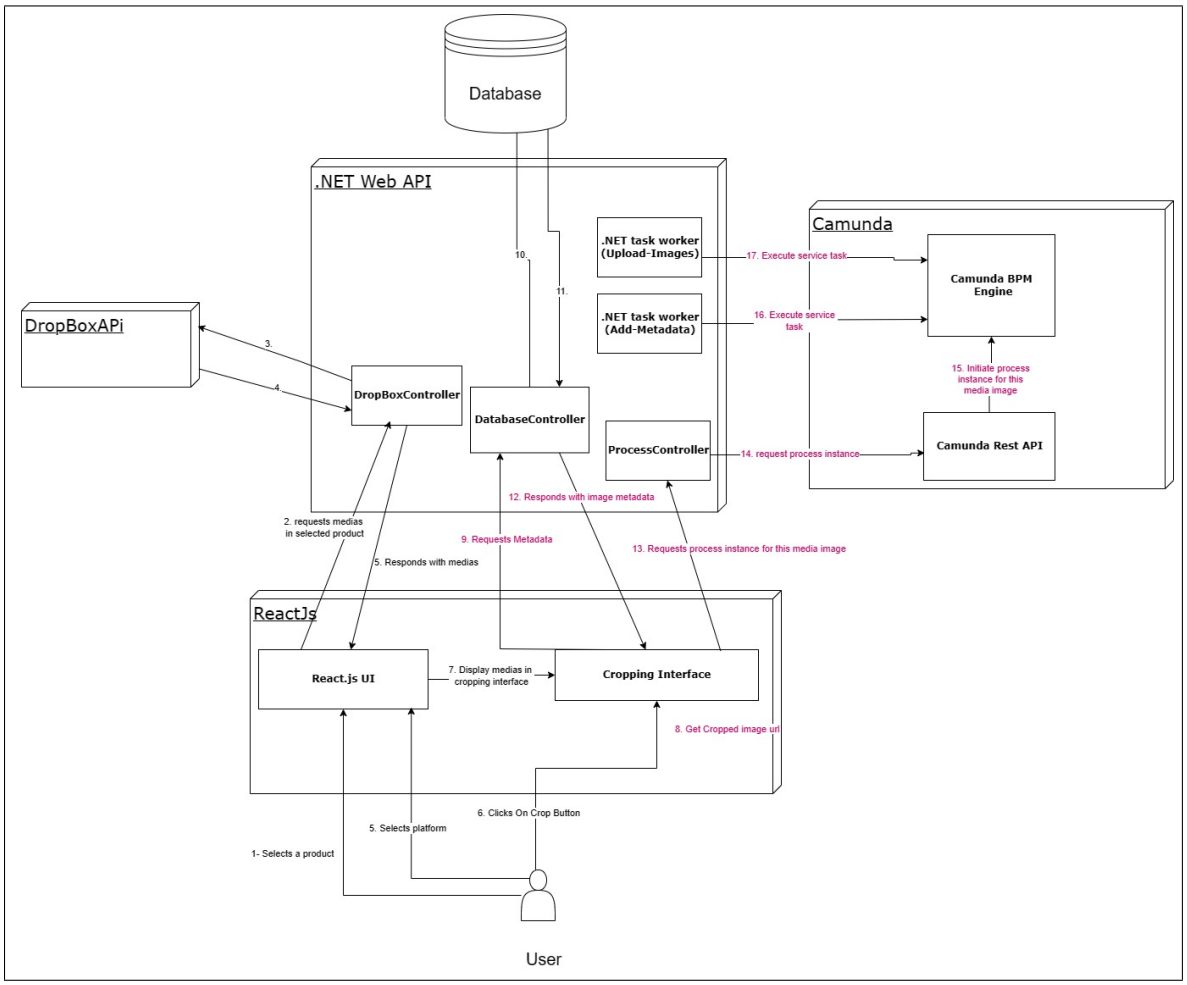


Figure 24: Project Flow Representation

Nb: This flow is after project selection.

- Within the "Products" section, the user selects a specific product (1).
- Reactjs initiates a request to the API by sending a request to the DropBoxController for media associated with the selected product (2) which communicates with the Dropbox API (3) to retrieve media files related to the selected product (4).
- The user selects a target platform (5). Then all medias associated with the product will be displayed in croppers inside the cropping interface (6).

- After clicking on "crop" button (7), We will iterate through media images. So the following steps will be repeated for each media :
- Reactjs sends a request to the DatabaseController to fetch references of visible products in this media (9). The Database Controller then establishes communication with the database(10) and retrieves the relevant data (11), which is returned to React.js (12).
- Once the ProcessController receives the request with necessary information from Reactjs (13),it sends a request to the Camunda REST API to start a workflow process instance (14). This request contains essential data such as image path, platform, media name, project name, product name, and references of primary and visible products.

```
var camunda = new CamundaEngineClient(new System.Uri("http://localhost:8080/engine-rest/engine/default/"), null, null);
string processInstanceId = camunda.BpmnWorkflowService.StartProcessInstance("PostingProcess", new Dictionary<string, object>() {
        {"imagePath", imagePath },
        {"platform", platform },
        {"mediaName", mediaName },
        {"productName", productName },
        {"projectName", projectName },
        {"primaryProductReference",primaryProduct},
        {"secondaryProductsReferences",secondaryProducts},
    });
```

Figure 25: Start Process Instance

The CamundaEngineClient, connected to the Camunda REST API, facilitates the initiation of the workflow instance named 'PostingProcess.
Once the workflow instance is started (15), Camunda Workflow Engine takes charge, guiding the flow and execution of tasks as defined in the process model.

- When the process instance reaches the external service task "Add-Metadata", the Camunda engine identifies that this task needs to be executed externally by a worker.

```
[ExternalTaskTopic("Add-Metadata")]
[ExternalTaskVariableRequirements("imagePath", "platform", "primaryProductReference", "secondaryProductsReferences")]
0 références
public class Add_Metadata : IExternalTaskAdapter
{
    0 références
    public void Execute(ExternalTask externalTask, ref Dictionary<string, object> resultVariables)
    {
        // Adding metadata logic


        resultVariables.Add("imagePathToUpload", newImagePath);

    }
```

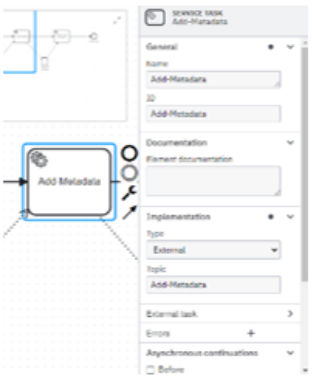


Figure 26: Add Metadata Worker

- The .NET external worker "Add-Metadata" periodically polls the Camunda engine using the Camunda client library. The worker checks for available external tasks with the topic "Add-Metadata" that are waiting to be processed.
  When the external worker finds an available external task that needs to be processed, it executes the logic defined in the Execute method of "Add-Metadata" class (16).
  The worker prepares result variable (imagePathToUpload) to be sent back to the Camunda engine. It is the path of the media after adding metadata to be uploaded on dropBox.
  After successfully executing the external task, the worker reports the completion to the Camunda engine.

- The process instance progresses to the "Upload-Image" service task with the topic "Upload-Image".


Figure 27: Upload Image Worker

The .NET external worker "UploadImage" periodically polls the Camunda engine using the Camunda client library. The worker searches for available external tasks with the topic "Upload-Image" that are waiting to be processed.
Atfer finding an available external task with the topic "Upload-Image" that needs to be processed, it executes the logic defined in the Execute method of "UploadImage" class(17).

- The process instance continues until it reaches an end event, indicating the successful completion of the process.

## 5.2 API Endpoints

Here are some of the key API endpoints provided by the project:

- **POST /api/dropbox/Upload** : Upload an image to Dropbox.
- **GET /api/dropbox/GetMediasInProduct/project/product** : Get image information for a specific project and product.
- **GET /api/dropbox/GetFolders/folderPath** : Get a list of folders in Dropbox giving the path.
- **GET /api/database/getMediaIdByUrl/url** : Get media ID by URL.
- **GET /api/database/getProductIdByReference/reference** : Get product ID by reference.
- **GET /api/database/getProductReferenceBy/id**: Get product reference by ID.
- **GET /api/database/getProjectIdByTitle/title**: Get project ID by title.
- **GET /api/database/getVisibleProducts/mediaId/projectId** : Get references of visible products for a media and project.
- **POST /api/process/begin** : This endpoint takes a JSON payload with information about the media, platform, product and project. It starts a Camunda process.

# 6 Conclusion

Camunda is a workflow engine that helps organizations to automatize their business processes.
It uses the visual language BPMN as a effective tool to model workflows that are quickly understood by humans.
This project aims to create a website that crops images to fit different platforms and enhances them with metadata.
I used Camunda Workflow Engine to organize jobs such as adding metadata to images and uploading them. I achieved this by using specialized .NET workers that handle these tasks efficiently.