UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Departamento de sistemas informáticos y computación

Universitat Politècnica de València

Desarrollo de un cliente utilizando Restful de un motor de procesos de código abierto

Trabajo final de máster

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

Autor: Carlos López Mayor

Tutor: Juan Sánchez Díaz

Curso 2019 - 2020

1. Planteamiento del problema 3
1.1 Motivación 3
1.2 REST Api Camunda 4
1.2.1 Qué es un REST API? 4
1.2.2 ¿Cómo funciona REST? 5
1.2.3 Atacar a la REST Api 6
1.3 Aplicación Web C# 6
2. Modelado de procesos de negocio con BPMN 7
2.1 Notación BPMN 7
2.2 Ventajas de la notación BPMN 7
2.3 Símbolos de la notación BPMN 8
2.3.1 Símbolos comunes 8
2.3.2 Tipos de evento BPMN 8
2.3.3 Símbolos de evento de BPMN 8
2.3.4 Símbolos de actividades de BPMN 10
2.3.5 Símbolos de puertas de enlace BPMN 10
3. CAMUNDA 12
3.1 Qué es Camunda? 12
3.2 Servidor web 12
3.3 Api REST 15
4. Diseño del cliente 18
4.2 Blazor 18
4.2 Descripción aplicación web 18
4.3 Llamadas Api REST Camunda utilizadas 19
4.3.1 Llamadas a Identity 19
4.3.1.1 Verify User 19
4.3.2 Llamadas a Process Definition 20
4.3.2.1 Get Process Diagram 20
4.3.2.2 Get List 21
4.3.2.3 Get Start Form variables 21
4.3.2.4 Get Rendered Start Form 22
4.3.2.5 Start Instance 22
4.3.3 Llamadas a Process Instance 23
4.3.3.1 Get List 23

4.3.4 Llamadas a Task 23
4.3.4.1 Get List 23
4.3.4.2 Get Task Form Variables 23
4.3.4.3 Get Rendered Form 24
4.3.4.4 Resolve 24
4.3.5 Llamadas a Authorization 25
4.3.5.1 Check 25
5. Manual de usuario 26
5.1 Creación de un proyecto Maven 28
5.2 Insertar imagen del modelo en el proyecto 31
5.3 Generar .WAR 32
5.4 Levantar servicio Camunda 33
5.5 Insertar .WAR en el servicio de Camunda 35
5.5.1 Insertar modelo desde Camunda Modeler 35
5.5.2 Insertar .WAR en Camunda 36
5.6 Login aplicación desarrollada 38
5.7 Process Definitions 39
5.8 Process Instances 40
6. Conclusión 47
6.2 Planes a futuro 47
7. Bibliografía 48
7.1 Referencias Web 48
7.2 Referencias Bibliográficas 49

1. Planteamiento del problema

1.1 Motivación

Los motores de procesos de negocio de código abierto basados en Java suelen proporcionar un conjunto de librerías en Java y un conjunto de archivos war (Web Application Archive) que se despliegan habitualmente en un servidor de aplicaciones como Apache Tomcat o WildFly. En el caso del motor Camunda, explicado en detalle más adelante, los archivos war permiten acceder mediante una aplicación web llamada explorador al contenido del motor de procesos, en particular permiten desplegar e instanciar y ejecutar procesos.

La interfaz es muy limitada y no es habitual ni cómodo acceder a las funcionalidades del motor mediante el explorador de Camunda.

Con interfaz limitada se quiere decir que el explorador de Camunda permite solo un número limitado de tipos de variables a elegir para poder ejecutar los modelos.

Sin embargo el motor de Camunda permite todo tipo de objeto que se le pase en el formato correcto.

Por ello, una de las principales motivaciones para la realización de este proyecto, es la creación de una interfaz amplia y flexible para poder ejecutar todo tipo de modelos con todo tipo de variables.

Por tanto, en el proyecto se pretende crear un cliente web en C# que acceda a la funcionalidad de Camunda mediante la implementación de un servicio Http personalizado que acceda a la api Rest que proporciona el motor de procesos. Esto permitirá instanciar procesos y ejecutarlos proporcionando la información que necesitan, es decir rellenando las variables que espera el proceso en cada una de sus tareas.

Además, al ser el servicio http desarrollado en C#, un lenguaje en el que Camunda no ofrece implementación, se puede implementar en todos los proyecto C# que quieran disponer de dicho servicio para gestionar y ejecutar procesos bpmn, es decir, este servicio se puede utilizar como librería C# y agilizar el desarrollo de las aplicaciones que quieran utilizar el motor de Camunda.

1.2 REST Api Camunda

1.2.1 Qué es un REST API?

La tecnología REST cambió por completo el mundo del desarrollo software a partir del 2000. Este nuevo enfoque de desarrollo de proyectos y servicios web fue definido por Roy Fielding, el padre de la especificación HTTP y uno de los referentes internacionales en todo lo relacionado con la Arquitectura de Redes. En el campo de las APIs, REST (Representational State Transfer- Transferencia de Estado Representacional) es, a día de hoy, el protocolo más utilizado en el desarrollo software.

REST se define como cualquier interfaz que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST.

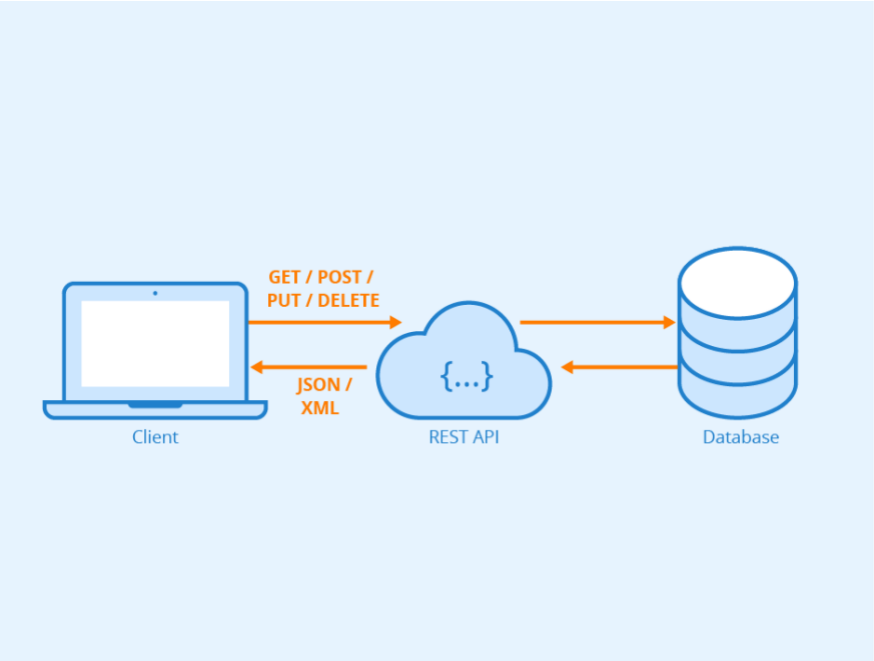


Figura 1. Comunicación API REST

Algunas de las principales características de las Api Rest son:

- Protocolo cliente/servidor sin estado

- Las operaciones más importantes en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar)

Las reglas de una arquitectura REST son:

- Interfaz Uniforme:
  - La interfaz de basa en recursos o clases
  - Transparencia de datos por parte del cliente de las llamadas y respuestas del servidor
  - Los datos recibidos a través de la consulta que le llegan al cliente, permitirán la edición o el borrado de esos datos o recursos:
    - Suponiendo que tenga permisos
    - Por eso en el recurso solicitado se suele enviar un parámetro Id
  - Mensajes descriptivos:
    - Usar las características del protocolo http para mejorar la semántica:
      - HTTP Verbs
      - HTTP Status Codes
      - HTTP Authentication

1.2.2 ¿Cómo funciona REST?

Tomando como ejemplo al servicio que consumimos que es la API REST de Camunda vamos a explicar como funciona REST. REST funciona a través de llamadas que representan peticiones HTTP en las que:

- La URL representa el recurso:
  "http://localhost:8080/engine-rest/tasks"

- El método (GET, POST, PUT, DELETE) representa la operación:
  "GET http://localhost:8080/engine-rest/tasks"

- El código de estado HTTP representa el resultado (200, 204, 400, 404, 500)
  "200 OK HTTP/1.1"
  "404 NOT FOUND HTTP/1.1"

1.2.3 Atacar a la REST Api

La aplicación web creada utiliza una serie de llamadas HTTP para la gestión de ejecuciones de modelos BPMN. Estas llamadas van dirigidas a la Api REST pública de camunda que dispone de una serie de endpoints para dicha gestión.
En cuanto al login de la aplicación también se utiliza el sistema de gestión de usuarios que proporciona camunda, con la que se hacen una serie de llamadas para realizar dicho login.
Camunda tiene la guía de su Api Rest en la web "https://docs.camunda.org/" donde se muestran los distintos endpoints a los que puede atacar.

1.3 Aplicación Web C#

La aplicación desarrollada se podría definir como un Saas (Software as a service) que es un modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía de tecnologías de información y comunicación (TIC), a los que se accede vía Internet desde un cliente. La empresa proveedora TIC se ocupa del servicio de mantenimiento, de la operación diaria y del soporte del software usado por el cliente. Regularmente el software puede ser consultado en cualquier computador, se encuentre presente en la empresa o no. Se deduce que la información, el procesamiento, los insumos, y los resultados de la lógica de negocio del software, están hospedados en la compañía de TIC.

Esta aplicación se encarga de solucionar el problema mencionado anteriormente, es decir, ofrece una interfaz entendible y flexible para la ejecución de modelos BPMN.
Esta aplicación se ha desarrollado con el nuevo framework de microsoft Blazor que utiliza el lenguaje C#. Esta aplicación utiliza una serie de servicios que atacan a la REST Api de Camunda para la ejecución y gestión de modelos BPMN.
Estos "ataques" o consultas a la REST Api se realizan con llamadas HTTP implementadas en un servicio dentro de la aplicación.

2. Modelado de procesos de negocio con BPMN

La aplicación permite como se ha comentado anteriormente, la ejecución de modelos BPMN, pero qué es BPMN?

2.1 Notación BPMN

La notación BPMN 2.0 o ***Business Process Model and Notation 2.0,*** significa en español Notación de Modelado de Procesos de Negocios 2.0 y como su nombre indica, es una evolución en el uso de iconos y símbolos estándar para determinar los flujos y procesos de negocio diseñados en un **diagrama de procesos**.

BPMN 2.0 es la notación más utilizada para diseñar y comunicarse visualmente en el contexto de BPM. Esta notación la creó y perfeccionó la Business Process Management Initiative, que más tarde se unió a la Object Management Group, una asociación internacional abierta y sin fines de lucro, fundada en 1989.

La notación BPMN 2.0 fue desarrollada con el objetivo específico de crear un estándar, un lenguaje común para el **modelado de procesos de negocio**.

Por lo tanto, al diseñar un diagrama de proceso, se pueden utilizar símbolos universales que los profesionales que tienen acceso a esta notación estándar internacional entenderán fácilmente.

Estas son algunas de las razones que hicieron que la notación BPMN 2.0 sea tan difundida y aceptada por los profesionales como una solución excelente para la representación de los procesos.

2.2 Ventajas de la notación BPMN

Algunas de las ventajas que dispone la notación BPMN son:

- Da apoyo a la gestión de procesos de negocio
- Notación intuitiva y entendible para los usuarios poco experimentados en BPM
- Representa de una forma más sencilla la semántica de procesos complejos
- Reduce el ruido que hay entre las etapas de diseño e implementación del proceso
- Al ser una notación intuitiva y fácil de entender, facilita la integración de integrantes en cualquier proyecto.

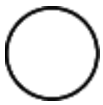
2.3 Símbolos de la notación BPMN

2.3.1 Símbolos comunes

Los símbolos representados en los diagramas BPMN se clasifican en cuatro grupos principales: objetos de flujo, objetos de conexión, carriles y artefactos.

2.3.2 Tipos de evento BPMN

Los eventos representan un suceso que ocurre en el entorno del negocio y que es de interés para un proceso de negocio.



Símbolo de evento de inicio - Indica el primer paso de un proceso.

Símbolo de evento intermedio - Representa cualquier evento que ocurre entre un evento de inicio y uno de finalización.

Símbolo de evento de finalización - Indica el último paso en un proceso

2.3.3 Símbolos de evento de BPMN

Existen diferentes tipos de símbolos de evento que sirven para representar los detalles de los procesos. Los siguientes ejemplos se encuentran dentro de los símbolos de eventos de inicio, pero se pueden combinar con cualquier tipo de

evento. Los símbolos más comunes de eventos representan las siguientes circunstancias:

Símbolo de mensaje - Activa un proceso a partir de un mensaje.

Símbolo de temporizador - Activa un proceso a partir de una fecha u hora.

Símbolo de escalación - Este evento solo se usa en un subproceso de evento. Una escalación ocurre cuando alguien con un nivel más alto de responsabilidad dentro de la organización se involucra en un proceso.

Símbolo condicional - Un proceso se inicia o prosigue dependiendo si se cumple una condición de negocio o regla de negocio.

Símbolo de enlace - Un subproceso que es parte de un proceso más extenso.

Símbolo de error - Evento que interrumpe el proceso cuando detecta un error en la ejecución del modelo.

Símbolo de cancelación - Reacciona a una transacción que se canceló dentro de un subproceso. En un evento de finalización, el símbolo de cancelación indica que se activó la cancelación de un proceso

Símbolo de compensación - Evento que se activa cuando un proceso falla de forma parcial.

Símbolo de señal - Una señal que se comunica en distintos procesos. Un símbolo de señal puede iniciar un proceso, facilitarlo o completarlo

Símbolo múltiple - Activadores múltiples que inician un proceso.

Símbolo de paralelas múltiples - Evento que no permite la continuación de un proceso hasta que no se terminen de ejecutar todos los eventos o procesos activos.

Símbolo de finalización - Evento que al activarse finaliza el proceso actual.

2.3.4 Símbolos de actividades de BPMN

Las actividades describen el tipo de trabajo realizado en una instancia concreta de un proceso. Hay cuatro tipos de actividades de BPMN: tareas, subprocesos, transacciones y actividades de llamada.

Símbolo de tarea - Actividad básica que no puede dividirse en subtareas.

Símbolo de subproceso - Es un grupo de tareas que se integran particularmente bien.

Símbolo de transacción - Es un subproceso especializado que involucra un pago.

Símbolo de llamada - Es un subproceso global que se reutiliza en diversos puntos en el flujo de negocios

2.3.5 Símbolos de puertas de enlace BPMN

Las puertas de enlace son símbolos que se encargan de distribuir o reconducir flujos en un diagrama BPMN. Hay distintos tipos de puertas de enlace:

Símbolo de exclusión - Dependiendo de una condición, la puerta de exclusión redirigirá el flujo del proceso por la ruta que marque dicha condición.

Símbolo basado en eventos - A diferencia de la puerta de exclusión, la puerta basada en eventos, como su nombre indica evalúa un evento y redirige el flujo de proceso.

Símbolo de paralelismo - Las puertas de enlace paralelas se emplean para representar dos tareas simultáneas en un flujo de negocio.

Símbolo de inclusión - Separa el flujo de procesos en uno o más flujos.

Símbolo de exclusión basado en eventos - Inicia una instancia nueva del proceso con cada suceso de un evento subsiguiente.

Símbolo de complejidad:- Estas puertas de enlace solo se usan para los flujos más complejos en un proceso de negocio. Un caso de uso ideal para una puerta de enlace compleja se da cuando necesitas puertas de enlace múltiples para describir el flujo de negocio.

Símbolo de paralelismo basado en eventos - Permite que múltiples procesos ocurran al mismo tiempo, pero a diferencia de la puerta de enlace paralela, los procesos dependen de eventos

3. CAMUNDA

3.1 Qué es Camunda?

Camunda BPM es una plataforma open-source de flujo de trabajo y decisiones automáticas. Camunda posee varias herramientas para la creación de flujos de trabajo y modelos de decisión, para la ejecución de modelos desplegados en producción y otorga el permiso a los usuarios para la ejecución de tareas pertenecientes a los flujos de trabajos asociados a dichos usuarios.
Camunda está desarrollada en Java y está lanzada como un software open-source bajo los términos y condiciones de Apache License.

Camunda ofrece el estándar BPMN asociado al motor del flujo de trabajo y el estándar DMN asociado al motor de decisiones que puede ser implementado por las aplicaciones java u otros lenguajes de programación vía REST.

3.2 Servidor web

Camunda ofrece un servidor web con diversas aplicaciones web en las que se encuentran:

- Admin es una aplicación que permite la gestión de usuarios y grupos vía motor del servicio de identidad (Identity Service engine) y permite la gestión de permisos vía motor del servicio de autorizaciones (Authorization Service engine).

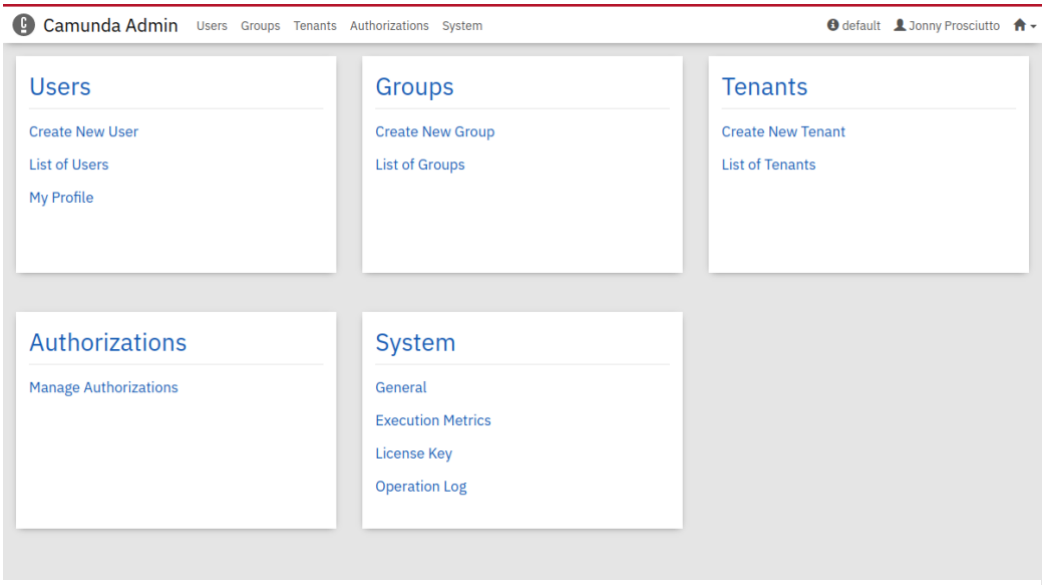


Figura 2. Web Application Camunda Admin

- Cockpit es una aplicación para la monitorizar y para la gestión de las operaciones. Permite el acceso a proceso BPMN desplegados y a decisiones DMN. También permite la búsqueda de instancias, ya sean en ejecución o estén finalizadas y las operaciones realizadas dentro de las instancias.

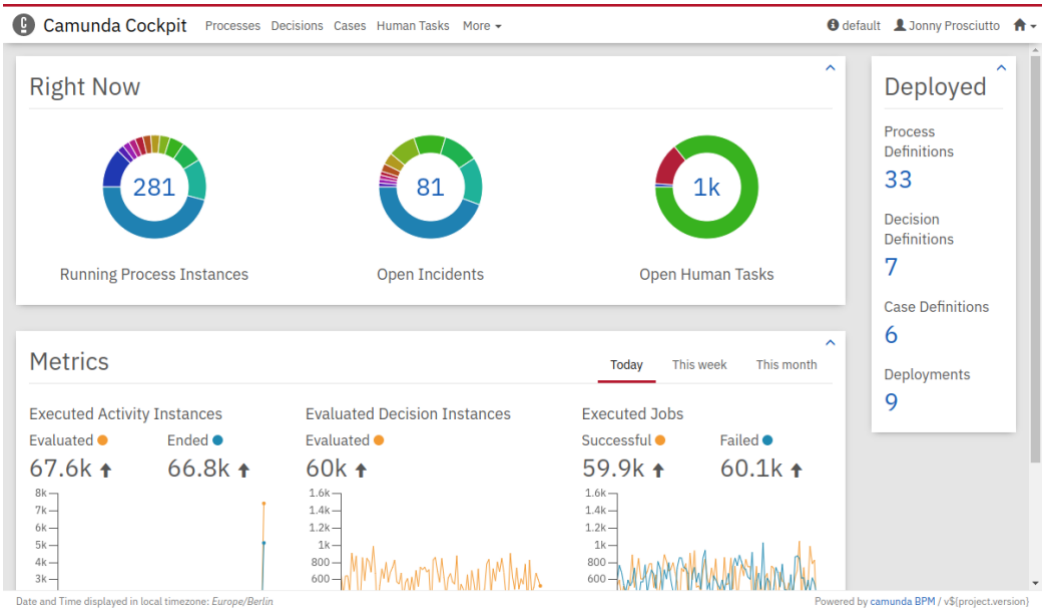


Figura 3. Web Application Camunda Cockpit

- Tasklist es una aplicación web que permite trabajar con *"user tasks"* o tareas asignadas a los distintos usuarios.

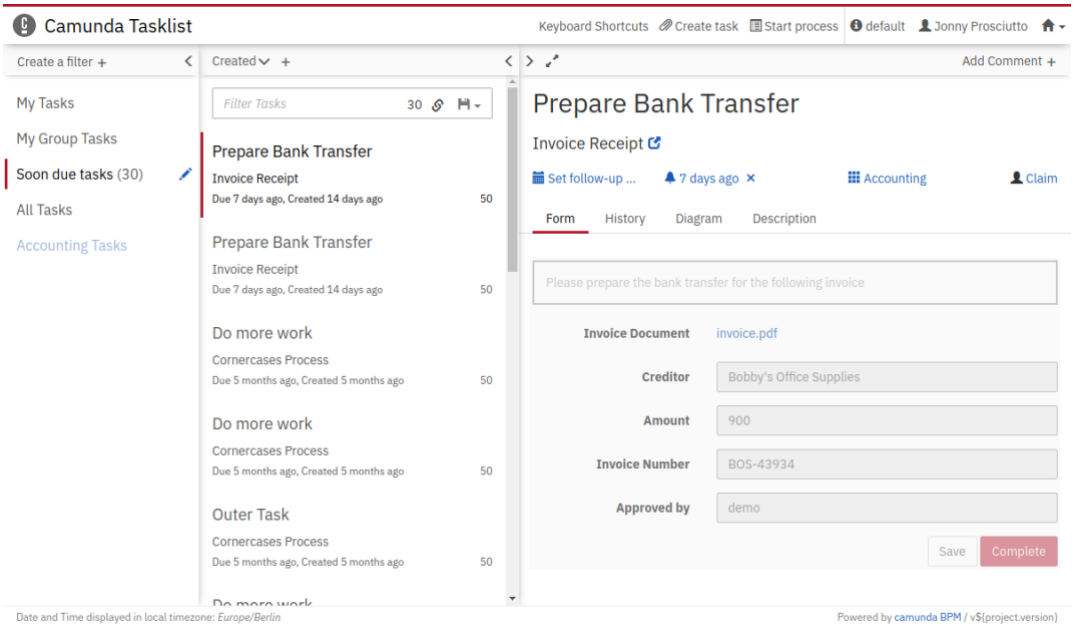


Figura 4. Web Application Camunda Tasklist

- Welcome. La aplicación de bienvenida presenta una serie de links a los que el usuario tiene acceso. Provee de una interfaz para que el usuario pueda modificar sus datos personales así como la dirección de correo, la contraseña, etc. La aplicación también tiene la funcionalidad de añadir custom links o enlaces personalizados a otras intranets o distintos recursos.

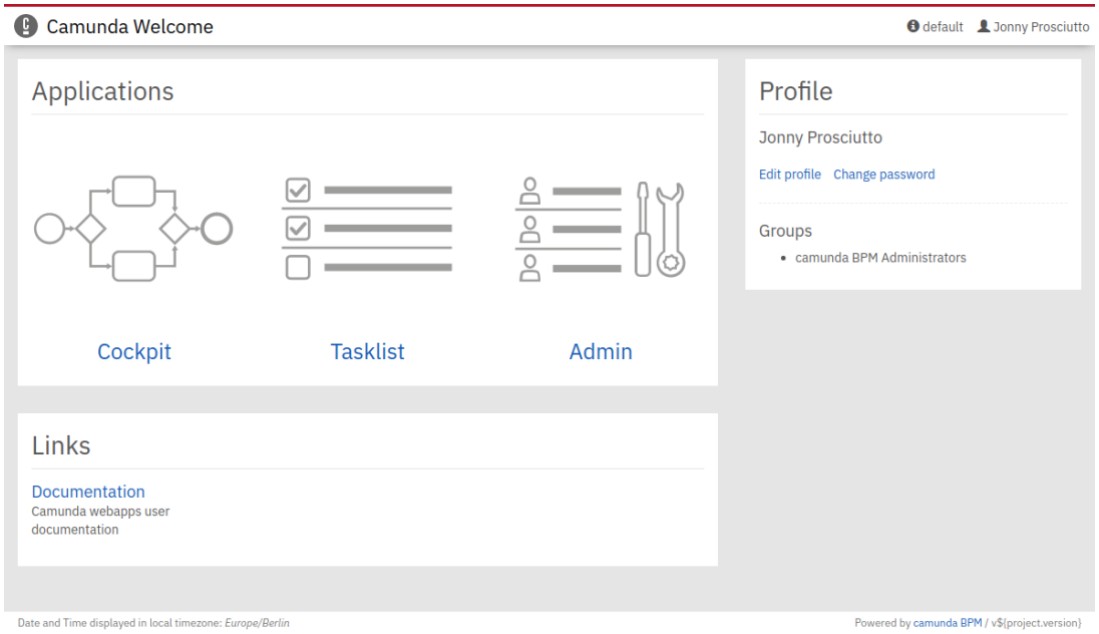


Figura 5. Web Application Camunda Welcome

3.3 Api REST

Camunda REST Api tiene una OpenAPI o api pública que sigue las especificaciones del estándar OpenAPI Specification 3.0.2. OpenAPI es un estándar con una interfaz agnóstica del lenguaje para Apis RESTfull que permite tanto a personas físicas como a ordenadores investigar y entender las capacidades del servicio sin tener acceso al código fuente, documentación o a través de un inspector de tráfico en la red.

El estándar de OpenAPI ofrece opciones para:

- La generación de un cliente (opción elegida para este proyecto) : Que provee de flexibilidad para la adopción del motor de procesos en cualquier lenguaje y la posibilidad de desarrollar implementaciones personalizadas basadas en dicho motor.
- Getting Started Experience o tutorial para los usuarios con opción de probar o testear la REST Api a partir de la documentación y ejemplos que provee.

La finalidad de la REST Api es proveer el acceso a todas las interfaces relevantes del motor.

En la documentación de la REST Api se explican todos los métodos desarrollados. Para cada método proveen de:

- Una descripción informal

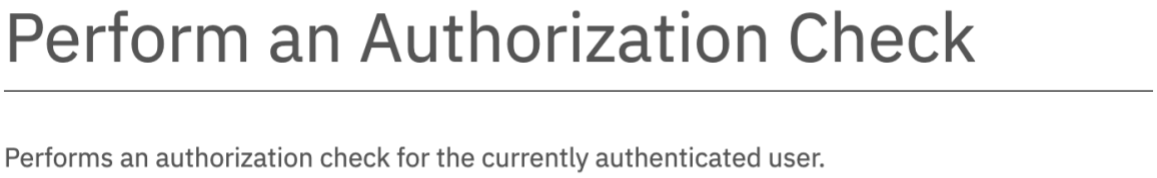


Figura 6. Descripción formal Camunda Endpoint

- Verbo HTTP y su URL

Figura 7. Verbo HTTP y ruta Camunda Endpoint

- Parámetros, caminos (*paths)* y consultas que se pueden realizar.

Parameters

Query Parameters

Name	Description	Required?
permissionName	String value representing the permission name to check for.	Yes
resourceName	String value for the name of the resource to check permissions for.	Yes
resourceType	An integer representing the resource type to check permissions for. See the User Guide for a list of integer representations of resource types.	Yes
resourceId	The id of the resource to check permissions for. If left blank, a check for global permissions on the resource is performed.	No
userId	The id of the user to check permissions for. The currently authenticated user must have a READ permission for the Authorization resource. If `userId` is blank, a check for the currently authenticated user is performed.	No

Figura 8. Parámetros Camunda Endpoint

- Una descripción detallada del contenido de la respuesta

Result

A JSON array with the following properties:

Name	Value	Description
permissionName	String	Name of the permission which was checked.
resourceName	String	The name of the resource for which the permission check was performed.
resourceId	String	The id of the resource for which the permission check was performed.

Figura 9. Resultado Camunda Endpoint

- Posibles códigos de respuesta (200 OK, 204, 400, etc)

Response Codes

Code	Media type	Description
200	application/json	Request successful.
400	application/json	Returned if some of the query parameters are invalid, for example if a permission parameterName is not valid for the provided resourceType. See the Introduction for the error response format.
401	application/json	The user is not authenticated. See the Introduction for the error response format.
403	application/json	When a `userId` is passed and the user does not possess a READ permission for the Authorization resource. See the Introduction for the error response format.
404	application/json	Authorization with given id does not exist. See the Introduction for the error response format.

Figura 11. Posibles códigos de respuesta Camunda Endpoint

- Ejemplos de uso

Example

Request

GET `/authorization/check?`
`permissionName=READ,resourceName=USER,resourceType=1,resourceId=jonny`

Response

Status 200.

```
{
  "permissionName": "READ",
  "resourceName": "USER",
  "resourceId": "jonny",
  "isAuthorized": true
}
```

Figura 10. Ejemplo de uso Camunda Endpoint

4. Diseño del cliente

Como se ha comentado anteriormente, la aplicación web se ha desarrollado con el nuevo framework de Microsoft, Blazor. Antes de adentrarnos en el diseño del cliente y su funcionalidad vamos a definir qué es Blazor.

4.2 Blazor

Blazor es un framework para el desarrollo de aplicaciones web SPA, como lo pueden ser Angular, React o Vue. Dispone de todo lo que podemos necesitar para crear aplicaciones web profesionales de calidad, como puede ser un sistema de *bindings*, *routing*, componentes, ciclo de vida, validaciones, plantillas, gestión de errores, inyección de dependencias, etc.

El client-side de Blazor es un runtime .NET que corre sobre WebAssembly (WASM), que es una especificación abierta que define un “lenguaje ensamblador” común a todas las máquinas virtuales presentes en los navegadores modernos.

Por tanto, la primera vez que se accede a una aplicación Blazor WebAssembly, el navegador descarga el runtime de .NET como los ensamblados propios de la aplicación y sus dependencias. Tras la carga inicial, gracias a la caché de los navegadores, ya no se vuelve a realizar la descarga de dichos ensamblados.

Además, una de las características más relevantes de Blazor es que permite el desarrollo de interfaces web dinámicas sin la utilización de JavaScript, es decir, únicamente los servicios de .NET.

4.2 Descripción aplicación web

La aplicación consiste en un cliente que permite la gestión y la ejecución de modelos BPMN en C#, mediante llamadas HTTP a una Api REST pública que proporciona Camunda.

Dentro de la aplicación se ha implementado un servicio que realiza una conexión con un cliente HTTP a la url del servicio levantado de Camunda, que si se levanta en local corresponde a la dirección: "*http://localhost:8080/*" o se puede asociar a un dominio comprado.
En la aplicación se ha implementado un login que toma como base el sistema de gestión de usuarios de Camunda.
Posteriormente nos encontramos con una gestión de Process Definitions o WARs desplegados en el servidor de camunda donde se pueden acceder para ver sus distintas ejecuciones.
Dentro de cada process definition, tenemos las Process Instances o ejecuciones del WAR y el estado de dichas ejecuciones, es decir, en qué tarea se encuentra la ejecución.
Y finalmente, una interfaz que permite la ejecución de dichas tareas con una plantilla de modelo de datos simple y flexible.
También hay que decir, que esta aplicación sustituye al explorador de Camunda explicado anteriormente en las web applicactions de Camunda.
También puede utilizarse de base o complemento para cualquier tipo de aplicación que implique la gestión o ejecución de modelos BPMN en Camunda, ya que en esta aplicación se implementa un servicio que permite la utilización de los endpoints de la API de Camunda en el lenguaje C#.

4.3 Llamadas Api REST Camunda utilizadas

Las llamadas que de momento realiza la aplicación desarrollada a la API REST de Camunda son las siguientes:

4.3.1 Llamadas a Identity

Solo se realiza una llamada a este conjunto de endpoints para la identificación de usuarios en la aplicación.

4.3.1.1 Verify User

Verify user es una llamada que devuelve si las credenciales introducidas por parte del usuario son válidas.

Figura 12. Verbo HTTP y ruta Verify User

4.3.2 Llamadas a Process Definition

En este grupo de endpoints se realizan distintas llamadas para la gestión de estos procesos.

4.3.2.1 Get Process Diagram

Esta llamada devuelve la imagen del modelo bpmn del proyecto. Se utiliza para mostrar dicha imagen en la pantalla de detalle del Process Definition.

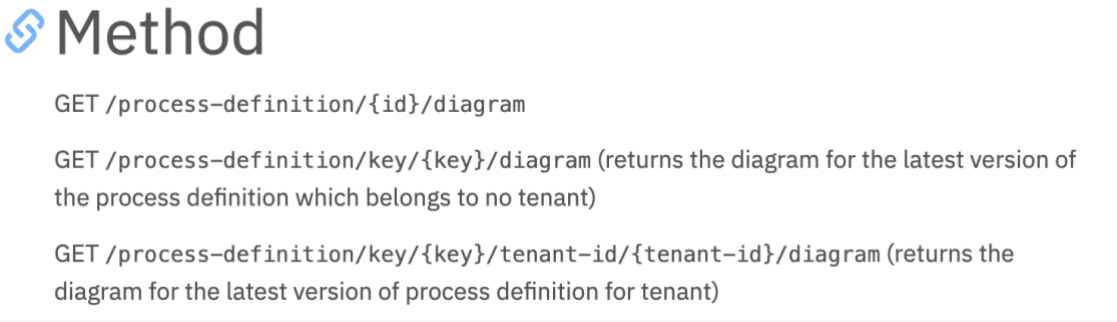


Figura 13. Verbo HTTP y rutas Get Process Diagram

Esta llamada como se observa en la Figura 13, se puede realizar de distintas formas dependiendo de los datos que tengamos a nuestra disposición en el momento que se realiza la llamada.
Se puede hacer teniendo la ID, la key o la key y el tenant-id del process definition.
El resultado de la llamada es la respuesta HTTP 204 en el caso de no tener el proyecto desplegado una imagen en algún directorio del proyecto.

4.3.2.2 Get List

Esta llamada devuelve la lista de process definitions que cumplen los parámetros que se le pasa en la ruta.

Figura 14. Verbo HTTP y ruta Get List

4.3.2.3 Get Start Form variables

Esta llamada devuelve las variables o parámetros necesarios para poder crear una instancia del process Definition.
La respuesta es un JSON con la id de la variable y el tipo.
Y la llamada se puede realizar de distintas formas dependiendo de las variables disponibles en el momento de la llamada.

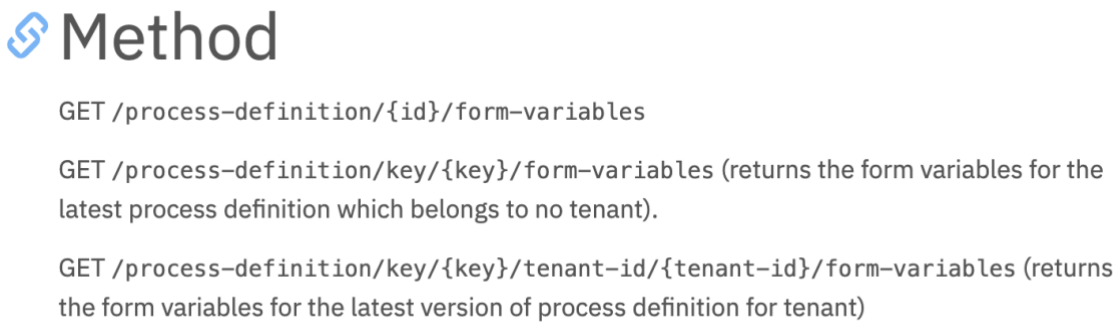


Figura 15. Verbo HTTP y ruta Get Start Form Variables

4.3.2.4 Get Rendered Start Form

Esta llamada devuelve un HTML con el formulario básico de camunda para la creación de una instancia del Process Definition.
En el caso de la aplicación hace una intersección entre la llamada anterior de Get Form Variables y está para realizar su formulario personalizado.

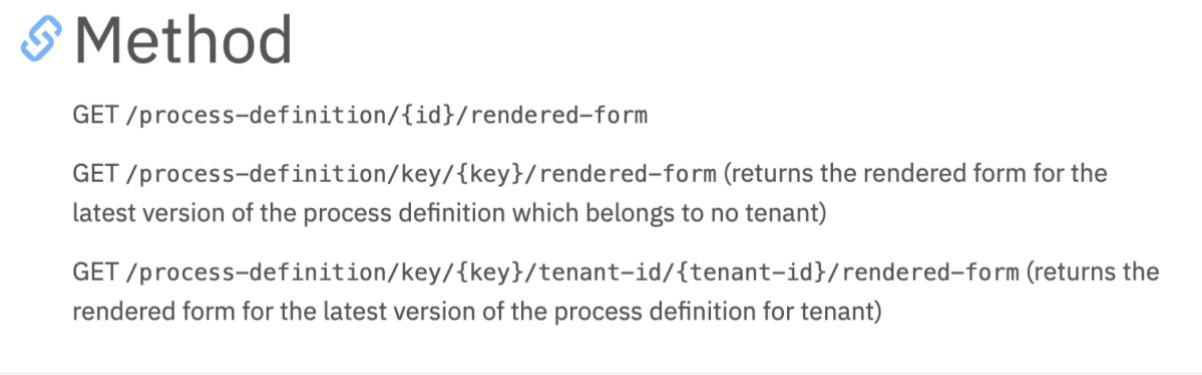


Figura 16. Verbo HTTP y ruta Get Rendered Start Form

4.3.2.5 Start Instance

Esta llamada inicia una instancia del Process Definition que corresponde a los parámetros que se pasan en la llamada HTTP.

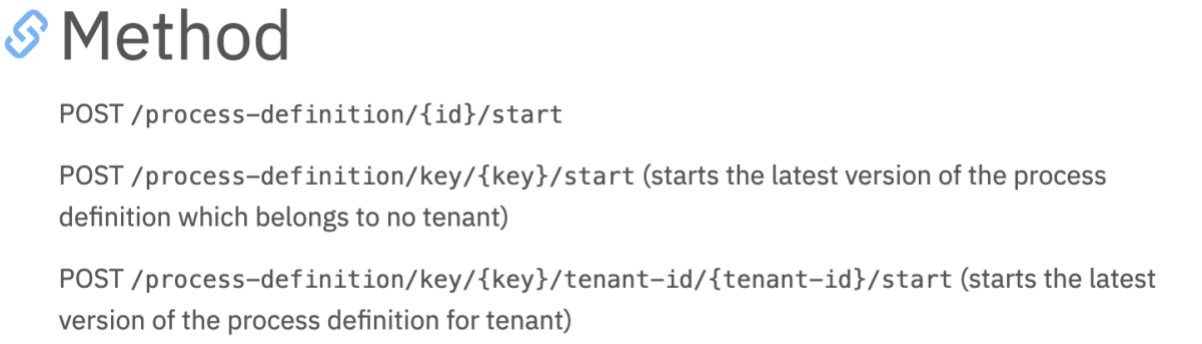


Figura 17. Verbo HTTP y ruta Start Instance

4.3.3 Llamadas a Process Instance

4.3.3.1 Get List

Esta llamada devuelve la lista de process instances que cumplen los parámetros que se le pasa en la ruta y que pertenezcan al process definition especificado.

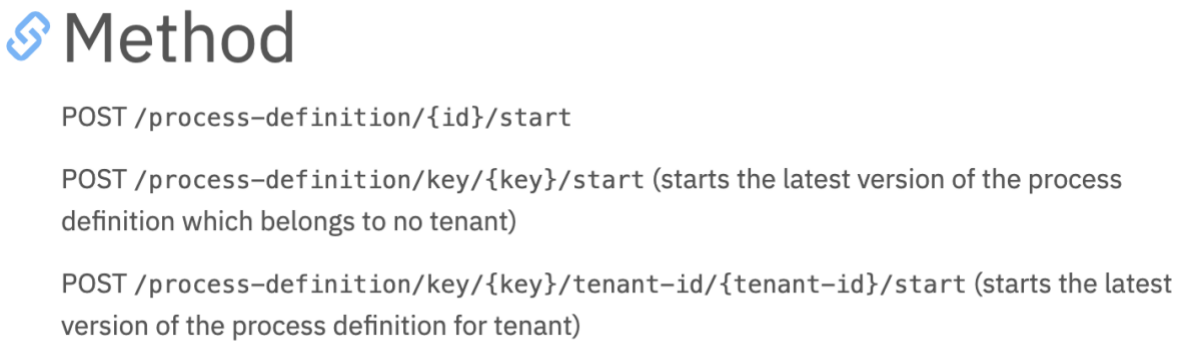


Figura 18. Verbo HTTP y ruta Get List

4.3.4 Llamadas a Task

4.3.4.1 Get List

Esta llamada devuelve la lista de las tasks que cumplen los parámetros que se le pasa en la ruta y que pertenecen al process instance que se especifica.

Figura 19. Verbo HTTP y ruta Get List Task

4.3.4.2 Get Task Form Variables

Esta llamada devuelve las variables o parámetros necesarios para poder ejecutar o iniciar el proceso de ejecución de la task especificada en la ruta.
La respuesta es un JSON con la id de la variable y el tipo.

Figura 20. Verbo HTTP y ruta Get Task Form Variables

4.3.4.3 Get Rendered Form

Esta llamada devuelve un HTML con el formulario básico de camunda para la ejecución o inicio del proceso de ejecución de la task especificada en la ruta.
En el caso de la aplicación hace una intersección entre la llamada anterior de Get Form Variables y está para realizar su formulario personalizado.

Figura 21. Verbo HTTP y ruta Get Rendered Form Task

4.3.4.4 Resolve

Esta llamada ejecuta la task especificada con los parámetros que necesita para poder ejecutarse y devuelve el resultado de la ejecución, es decir, si se ha ejecutado correctamente o no.

Figura 22. Verbo HTTP y ruta Resolve

4.3.5 Llamadas a Authorization

4.3.5.1 Check

A esta llamada se le pasa como parámetros los permisos y el usuario que se quieran comprobar.
La llamada devuelve si el usuario posee los permisos especificados.

Figura 23. Verbo HTTP y ruta Check

5. Manual de usuario

En este apartado se detalla el manual de usuario acompañado de un ejemplo de uso para mejorar la entendibilidad del funcionamiento de la aplicación.

Partimos del proyecto creado con Eclipse que contiene un archivo war y dentro de este el modelo bpmn que representa al proceso

El proyecto que se va a utilizar como ejemplo tiene como nombre: CamundaEjemplo.

Y su modelo bpmn es el siguiente:

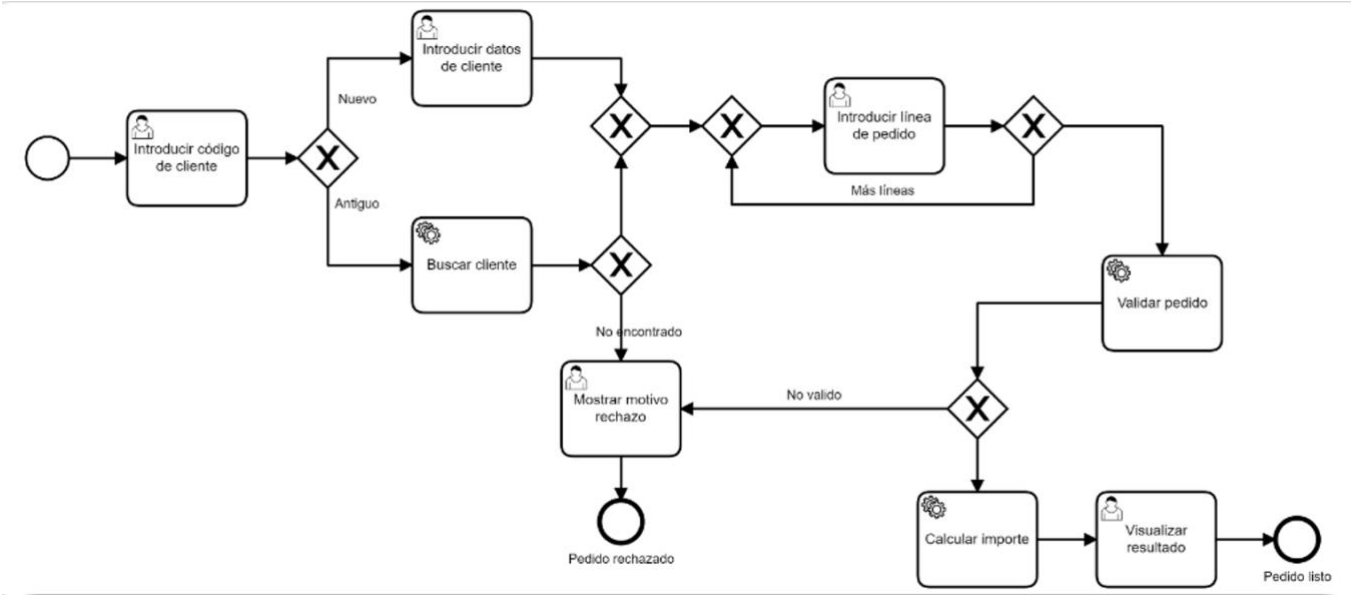


Figura 24. Modelo BPMN del ejemplo de uso.

El modelo de la Figura 24 trata de un sistema de pedido de pizzas que sigue los siguientes pasos:

- Se introduce el código de cliente y dependiendo de si el código es 0 o no tomará un flujo de ejecución diferente.
- En el caso de que el código de cliente sea 0, se pasaría a la tarea manual “Introducir datos de cliente” donde se rellenan los datos del nuevo cliente para insertarlo en la base de datos
- En el caso de que el código de cliente sea diferente a 0, se pasa a la tarea automática “Buscar cliente” donde se recorre la base de datos para encontrar el cliente.

- Si no se encuentra el cliente en la base de datos se pasaría a la tarea manual “Mostrar motivo rechazo” donde se muestra el motivo de que no siga el proceso y termina.
- Si se encuentra el cliente o los datos del nuevo cliente ya han sido rellenados, se pasa a la tarea manual “Introducir línea de pedido” donde se especifica las pizzas que se quiere y su cantidad. También hay un checkbox donde se especifica si se desea añadir más pizzas, si es así, se volvería atrás en el flujo volviendo a la misma tarea para rellenarla otra vez, en el caso de que no, continuaría con el flujo.
- Una vez introducidas todas las pizzas deseadas, se pasa a la tarea automática de validar pedido donde se comprueba que la pizza existe y se valida que el pedido es válido.
- En el caso que no sea válido se pasa a la tarea mencionada anteriormente “Mostrar motivo rechazo” donde se indica el motivo de que no se siga con el proceso.
- En el caso de que el pedido sea válido, se pasa a la tarea automática de calcular el importe donde se suman todos los importes de las pizzas.
- Después de calcular el importe se pasa a la última tarea “Visualizar resultado” donde se muestra el importe total del pedido, es decir, lo que se tiene que pagar.
- Una vez finalizada esta tarea, termina el flujo y por tanto la ejecución del modelo BPMN.

5.1 Creación de un proyecto Maven

En el caso de no tener desarrollado un proyecto para poder seguir los pasos que siguen, los pasos para la creación de un proyecto son:

- Se abre Eclipse y se selecciona en File -> New -> Other.

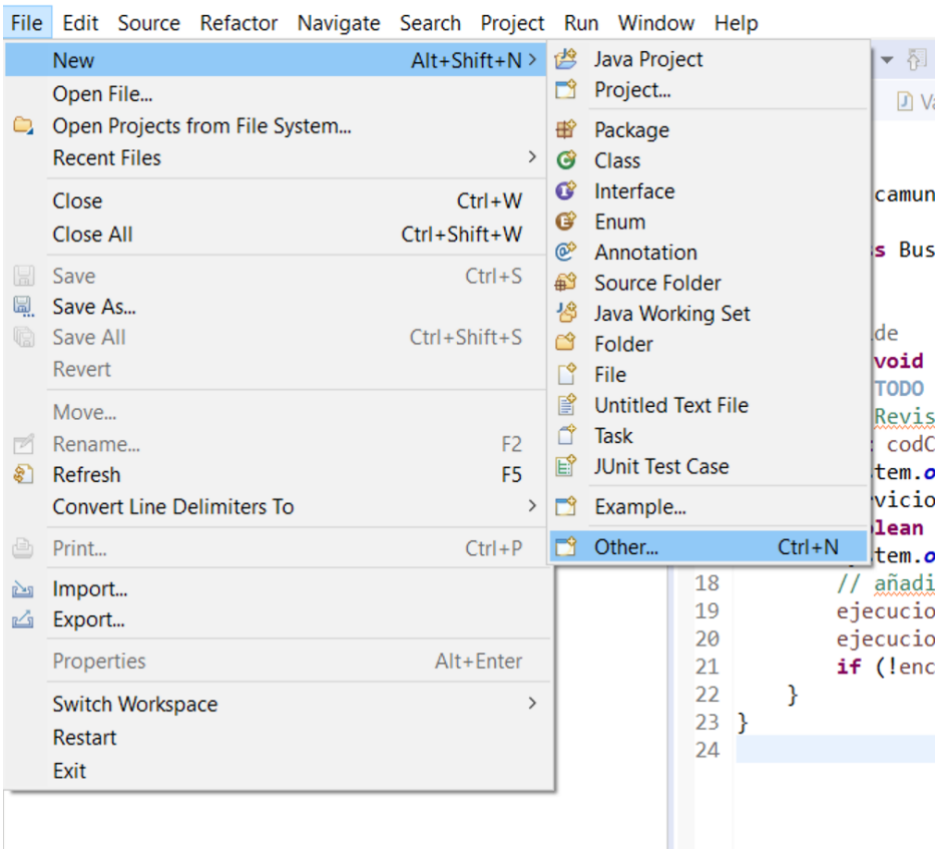


Figura 25. Creación proyecto Maven

- Una vez seleccionada dicha opción se abre una ventana emergente para especificar todos los datos necesarios para su creación. En dicha ventana se debe de especificar el tipo de proyecto que se desea crear. En nuestro caso es un maven project, es decir, seleccionamos la opción de “Maven Project” y pulsamos Next.

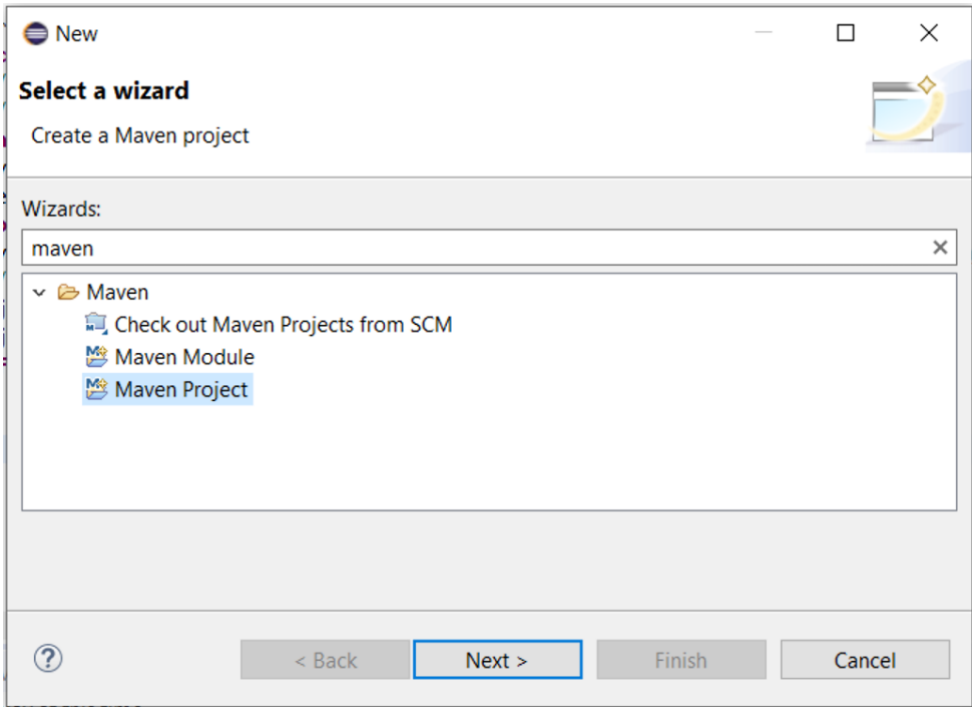


Figura 26. Selección plantilla proyecto

- Seleccionado el tipo de proyecto que se desea crear, aparecen una serie de parámetros necesarios para la creación como por ejemplo el directorio del proyecto. Una vez se especifican, se pulsa Next.

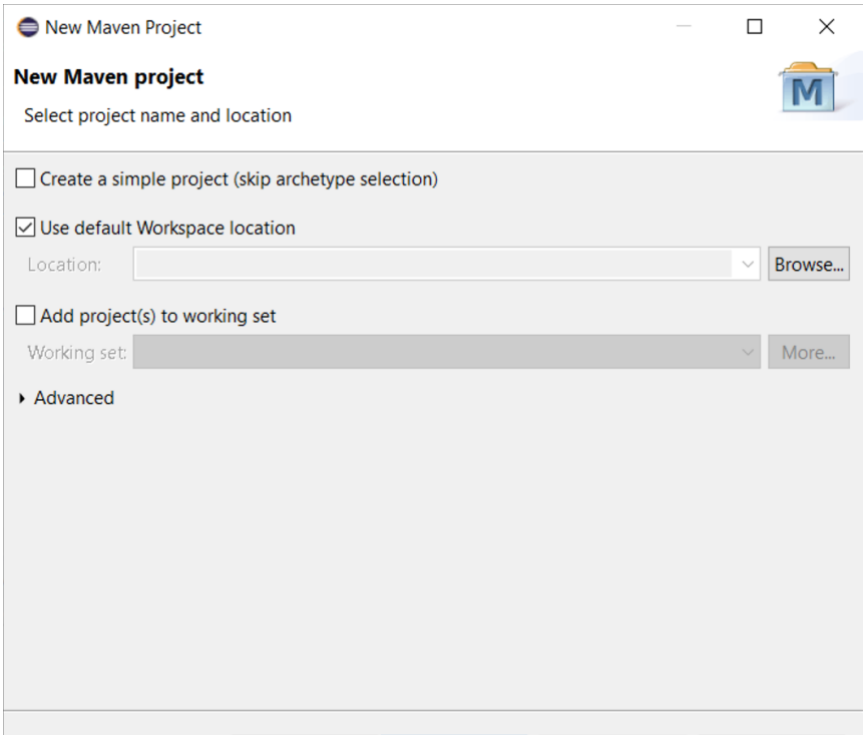


Figura 27. Parámetros para la creación de un proyecto

- Rellenados todos los parámetros, se pasa a la selección de la arquitectura que se desea en la construcción del proyecto.

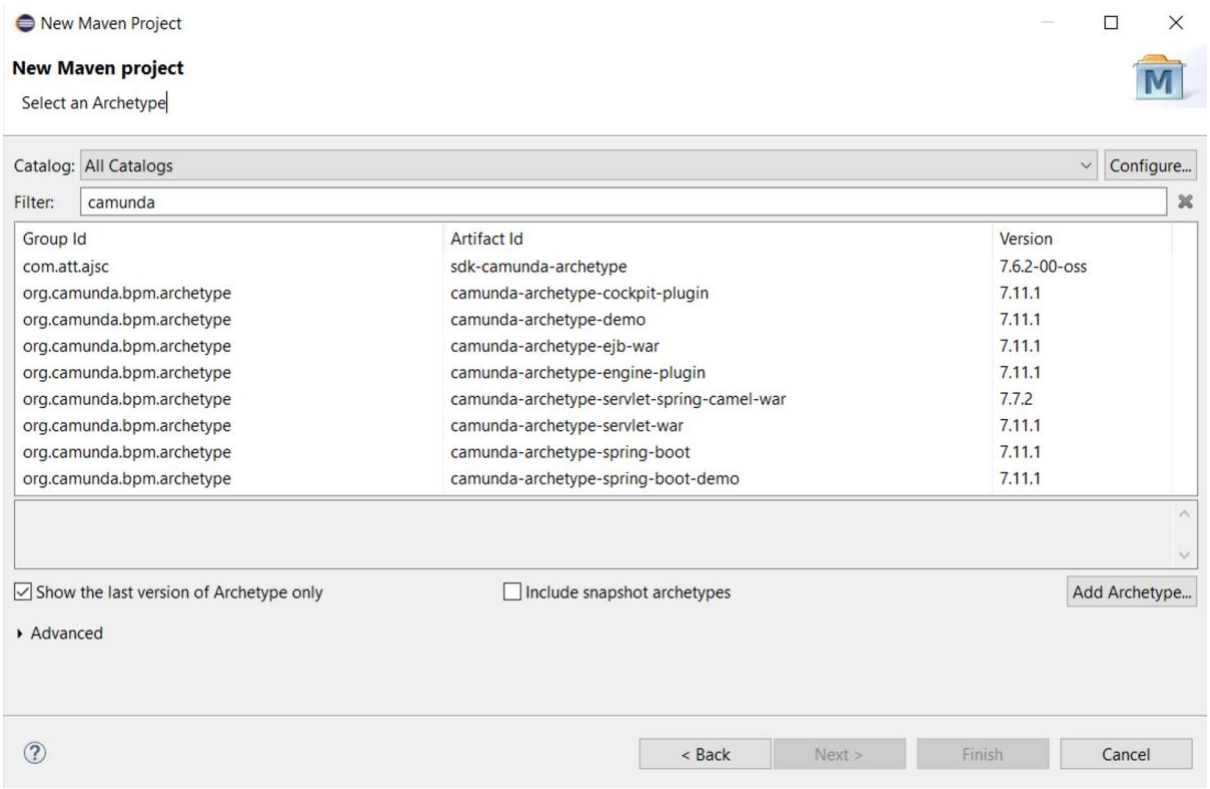


Figura 28. Selección arquitectura del proyecto Maven

- Se selecciona la arquitectura que se desea y se pulsa sobre Finish para finalizar la creación del proyecto Maven.
- Una vez creado el proyecto Maven, se pueden introducir tanto el modelo bpmn que se desea que tenga el proyecto, como las clases con el código de cada tarea del modelo bpmn que se ha añadido al proyecto.

5.2 Insertar imagen del modelo en el proyecto

El primer paso que se realiza es abrir el modelo BPMN del proyecto con el programa Camunda Modeler.

Una vez abierto el modelo BPMN, se guarda dicho modelo como .png dentro del proyecto maven.

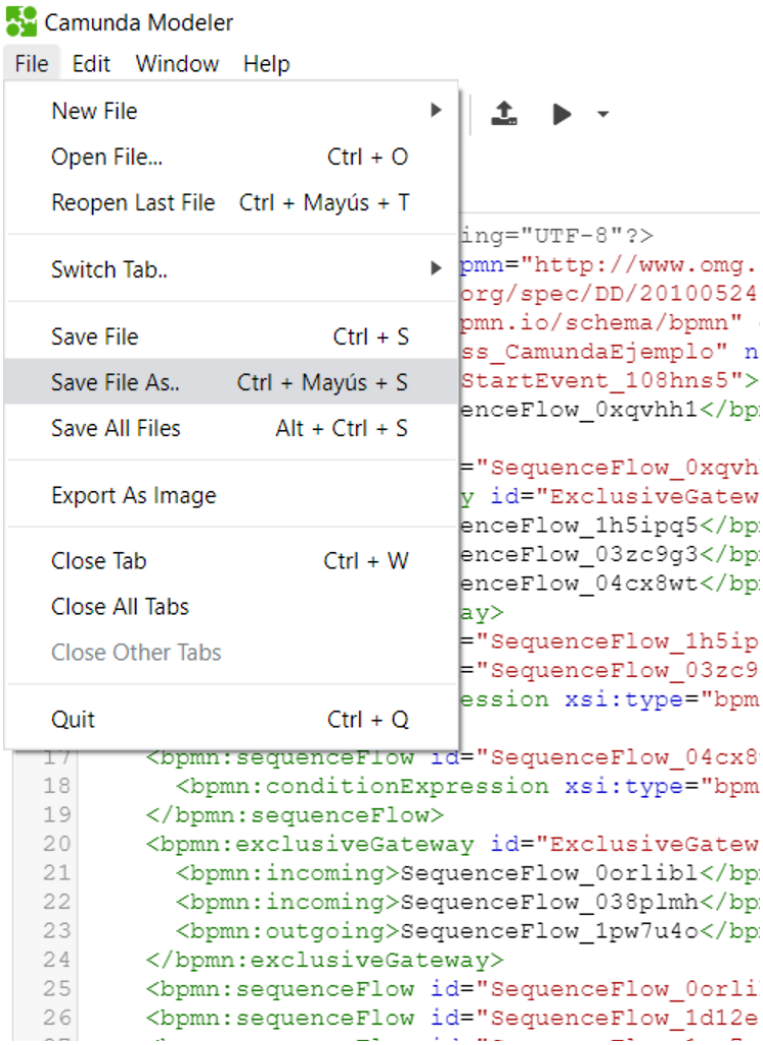


Figura 29. Guardar como Camunda Modeler

El archivo .png se guarda en el mismo directorio que el archivo .bpmn que contiene el modelo.

Este paso se realiza para que la aplicación pueda mostrar el modelo a la hora de su gestión y ejecución de tareas.

5.3 Generar .WAR

Una vez generado el archivo .png a partir del modelo, se pasa a generar el archivo .war del proyecto desarrollado.

Para ello se abre el proyecto maven en eclipse. Una vez abierto, nos dirigimos al archivo "*pom.xml",* que es un archivo XML que contiene la información relativa al proyecto y la configuración usada por Maven para poder compilar el proyecto y poderlo ejecutar, y pulsamos botón derecho, nos dirigimos a la opción del menú *RunAs -> Maven install*

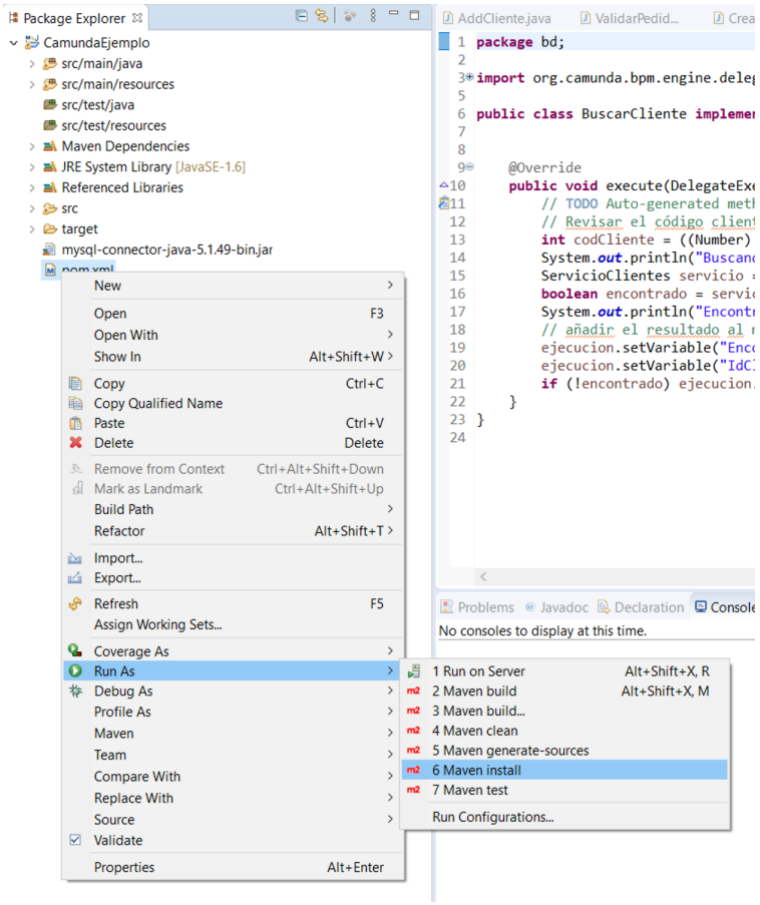


Figura 30. Generar WAR en Eclipse

Una vez pulsada la opción comentada anteriormente, se empezará la ejecución y en la consola emergente se podrá ver el progreso de dicha ejecución.

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.144 s
[INFO] Finished at: 2020-08-12T19:38:43+02:00
[INFO] ------------------------------------------------------------------------
```

Figura 31. Salida consola tras la generación del WAR

5.4 Levantar servicio Camunda

Después de generar el .WAR nos disponemos a levantar el servicio de Camunda para poder acceder a los servicios que nos ofrece Camunda y que la aplicación utiliza para la gestión y ejecución de modelos BPMN.

Para ello, abrimos el terminal y nos dirigimos a la ruta donde tenemos descargado el servicio de camunda. Una vez allí, insertamos el comando *".\start-camunda.bat"* para iniciar el servicio de camunda.

```
PS C:\                          \Documents\TFM\camunda-bpm-tomcat-7.12.0> .\start-camunda.bat
'starting camunda BPM platform 7.12.0 on Apache Tomcat 9.0.24"
```

Figura 32. Comando para levantar el servicio de Camunda

Una vez levantado el servicio de camunda se abrirá automáticamente el terminal del Apache Tomcat donde podremos todas las acciones y el progreso de servicio de camunda.

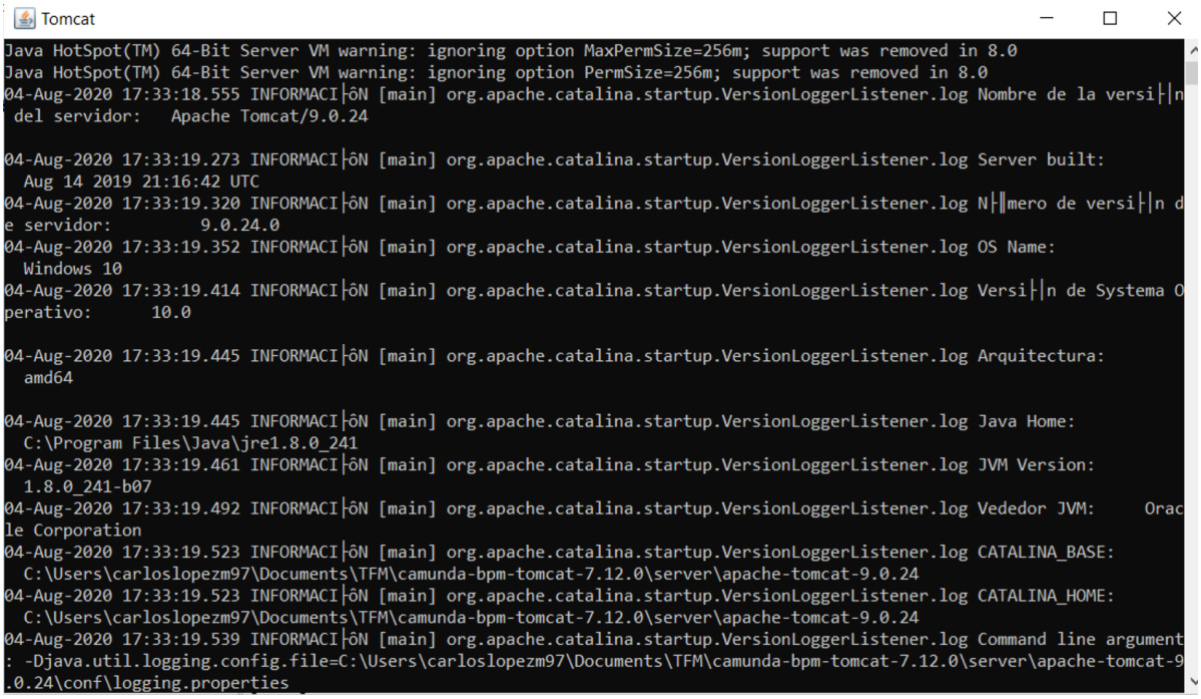


Figura 33. Salida consola al levantar el servicio de Camunda

Para corroborar que el servicio de camunda está levantado, como en este caso se ha levantado en local, abrimos el navegador e insertamos la url *"http://localhost:8080/camunda".*

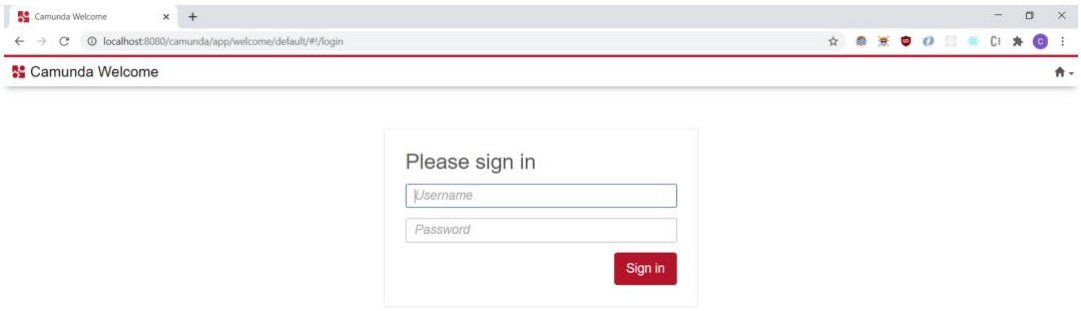


Figura 34. Log In Camunda Web

Al no estar logueados, nos redirige a la pantalla de login donde podemos identificarnos y entrar a la página de inicio de camunda.

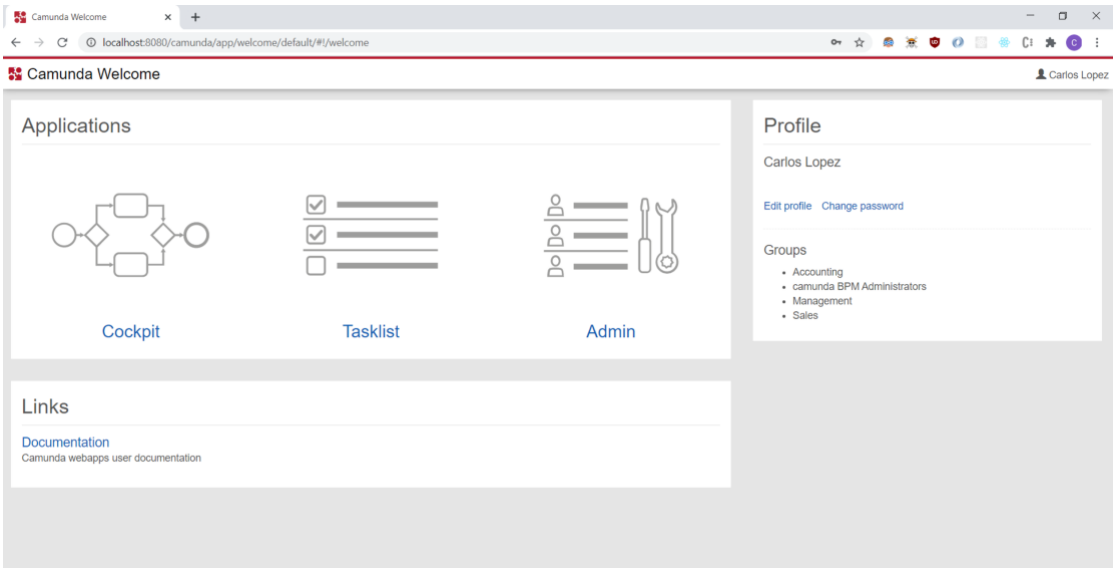


Figura 35. Camunda Welcome

Como se puede observar en la Figura 35, la pantalla de inicio de camunda nos proporciona links a las distintas aplicaciones web disponibles de camunda dependiendo del rol del usuario. En esta caso, como el usuario Carlos López es administrador, nos da la posibilidad de ir al servicio web de Admin donde se administran y gestionan los usuarios, grupos y permisos que poseen cada uno de ellos, Tasklist donde se pueden ver y ejecutar las tasks asignadas al usuario y por

último al Cockpit donde se pueden ver y gestionar los procesos BPMN desplegados en Camunda.

5.5 Insertar .WAR en el servicio de Camunda

Una vez generado el .WAR y haber levantado el servicio de Camunda, nos disponemos a desplegar dicho .WAR dentro de Camunda.
Hay que decir, que Camunda Modeler posee una opción para desplegar desde el programa un modelo BPMN, pero solo despliega el modelo y no todo el proyecto.

5.5.1 Insertar modelo desde Camunda Modeler

Figura 36. Deploy current diagram Camunda Modeler

En esta opción, nos dirigimos a la barra de herramientas del Camunda Modeler y seleccionamos la opción Deploy Current Diagram.Una vez seleccionada esa opción, se abre una ventana emergente donde especificamos la url donde queremos desplegar el modelo y el nombre que se le quiera dar.

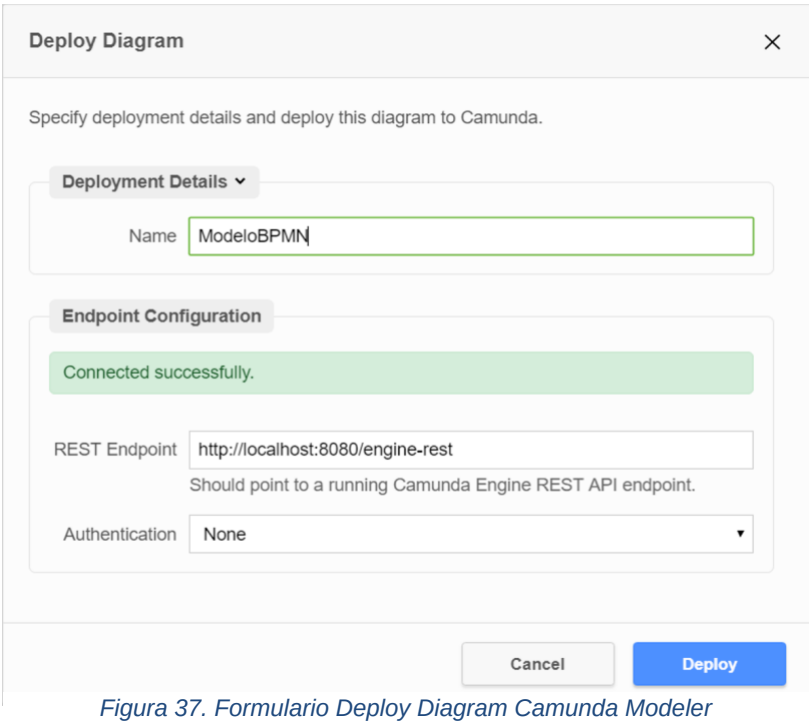


Figura 37. Formulario Deploy Diagram Camunda Modeler

5.5.2 Insertar .WAR en Camunda

Esta opción es la que permite desplegar proyecto maven en camunda a través de archivos .WAR generados a partir de dichos proyecto.

Recogemos el archivo .WAR creado anteriormente y lo colocamos en el directorio "*path_to_camunda\server\apache-tomcat-9.0.24\webapps"* donde el parámetro "path_to_camunda" es la ruta hasta donde se tiene ubicado el servicio de camunda.

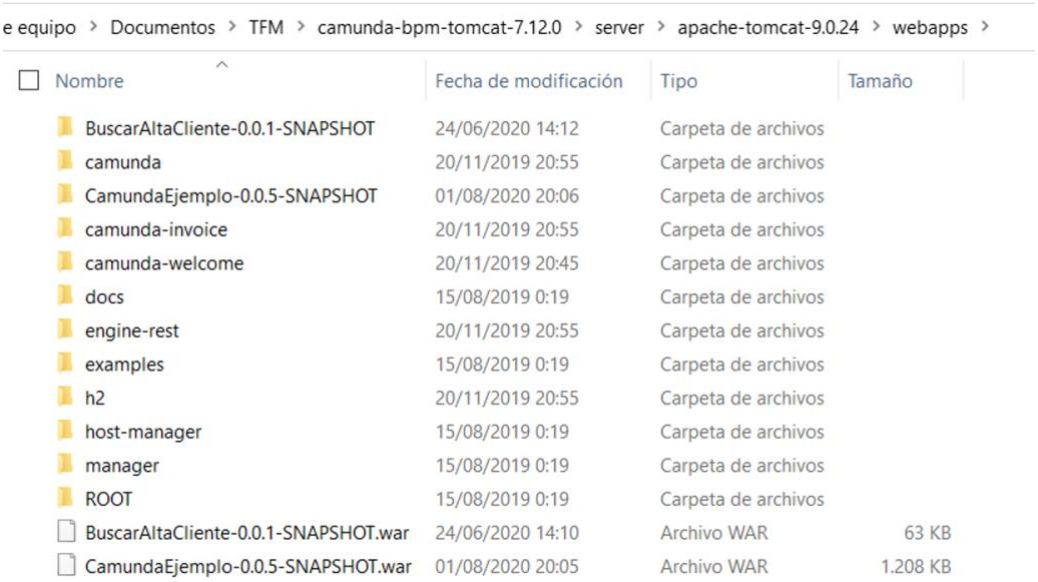


Figura 38. Directorio Camunda donde se encuentran las webapps

Como se observa en la Figura 38, ya hay colocados varios archivos .WAR que una vez introducidos en el directorio con el servicio de Camunda levantado, el propio servicio despliega el proyecto y genera una carpeta con el mismo nombre del archivo .WAR.

Hay dos formas de corroborar que .WAR ha sido desplegado correctamente, la primera nos dirigimos a la consola de Apache Tomcat abierta al levantar el servicio

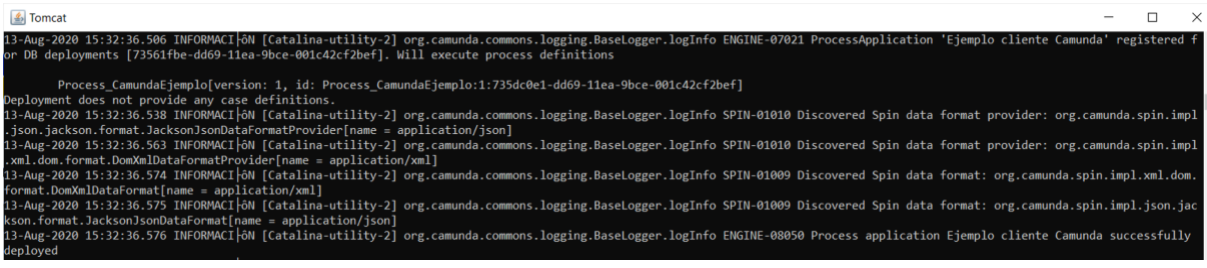


Figura 39. Salida Camunda al desplegar un WAR

Camunda y comprobamos que el archivo .WAR se ha desplegado correctamente.

La segunda, nos dirigimos a la página web de Camunda y pulsamos la opción “Cockpit”.

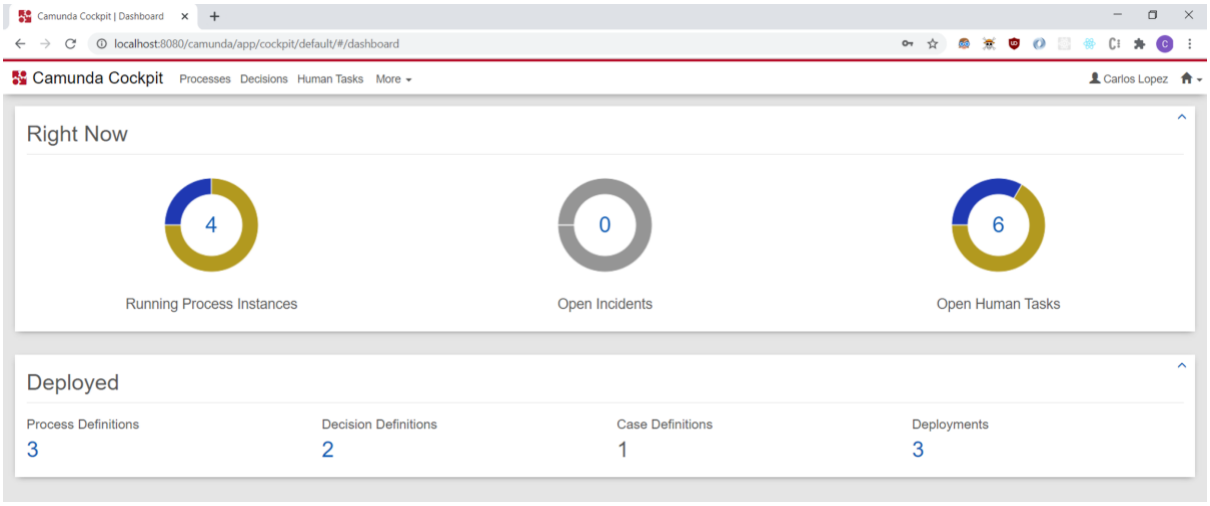


Figura 40. Camunda Cockpit

Una vez hemos seleccionado la opción “Cockpit” podemos ver en la sección “Deployed” la cantidad de “Process Definitions” o Proyectos desplegado, “Decision Definitions”, “Case Definitions” y “Deployments”.

En este caso, el dato que nos interesa es “Process Definitions”. Pulsamos sobre el número debajo de “Process Definitions” para que nos dirija a la vista más detallada de cada uno de ellos.

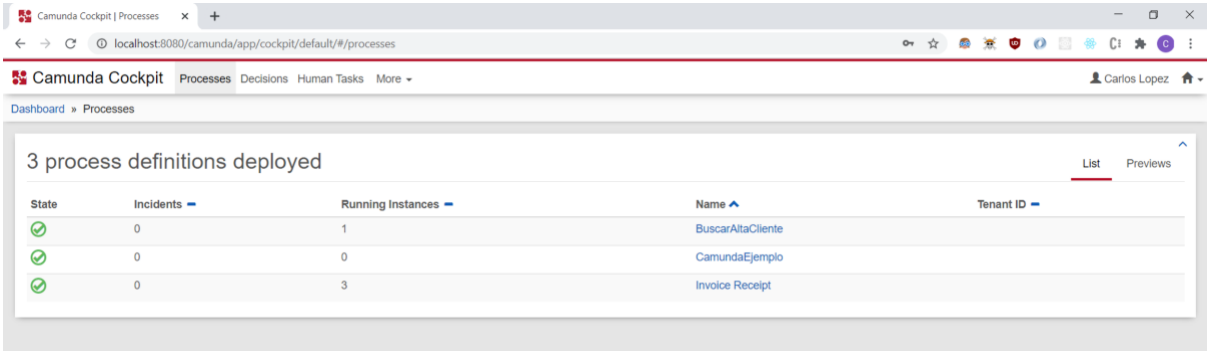


Figura 41. Camunda Cockpit - Procesos

Una vez en la vista más detallada de los “Process Definitions” comprobamos que el proyecto está desplegado.

5.6 Login aplicación desarrollada

Una vez realizados todos los pasos anteriores nos disponemos a utilizar la aplicación para la gestión del proyecto desplegado en camunda.
La primera pantalla que nos encontramos al abrir la aplicación en el navegador es la pantalla de identificación o login.

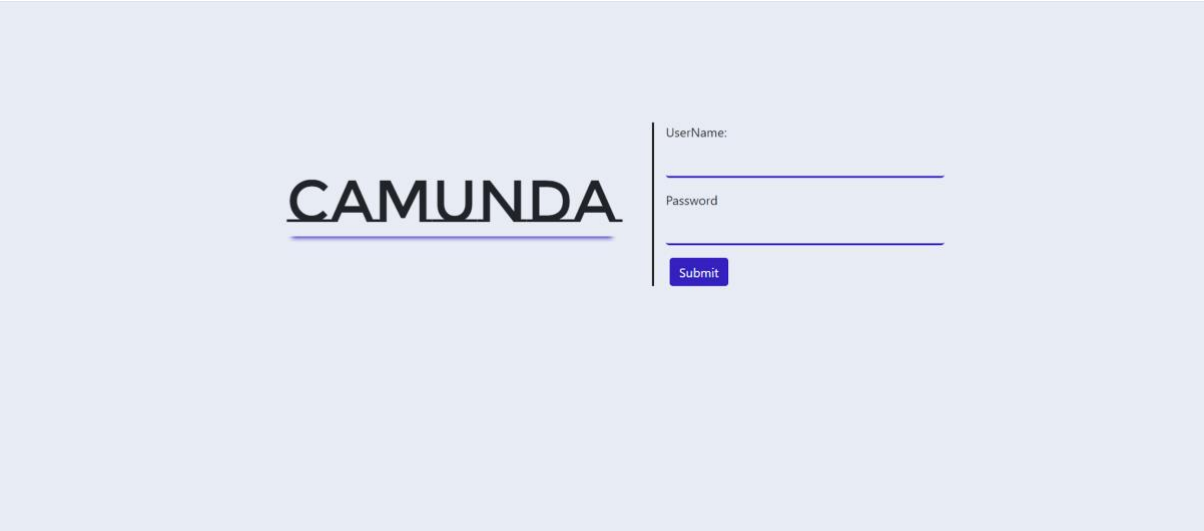


Figura 42. Log In aplicación desarrollada

La aplicación tiene un sistema de rutas que antes de mostrar cualquier pantalla se realiza una comprobación de si existe un usuario logueado y también una comprobación de permisos del usuario, es decir, no se puede acceder a ninguna pantalla a no ser que se esté identificado.
Esto se consigue con las Claims que disponemos en el proyecto y que nos llegan del HttpContext de la aplicación.
En esta pantalla se introducen el userName y la password para identificarse. Este proceso de identificación realiza una llamada al sistema de gestión de usuarios de camunda para comprobar que las credenciales que se le pasan como parámetros corresponden a un usuario registrado en dicho servicio.

5.7 Process Definitions

En la pantalla principal nos encontramos una lista con los Process Definitions.
Los Process Definitions son todos los proyectos desplegados en el servicio de Camunda.
En esta pantalla se hace una petición a la REST API para recuperar dicha lista donde se recogen y se tratan todos los datos de la manera más sensible posible y que aparezcan en la pantalla aquellos datos más relevantes y que puedan mejorar la entendibilidad de la pantalla.
Los datos seleccionados para aparecer en la lista son el Nombre del Process Definition, la id la cual es autogenerada por Camunda y la key del proyecto creada cuando se despliega el mismo en el servicio.

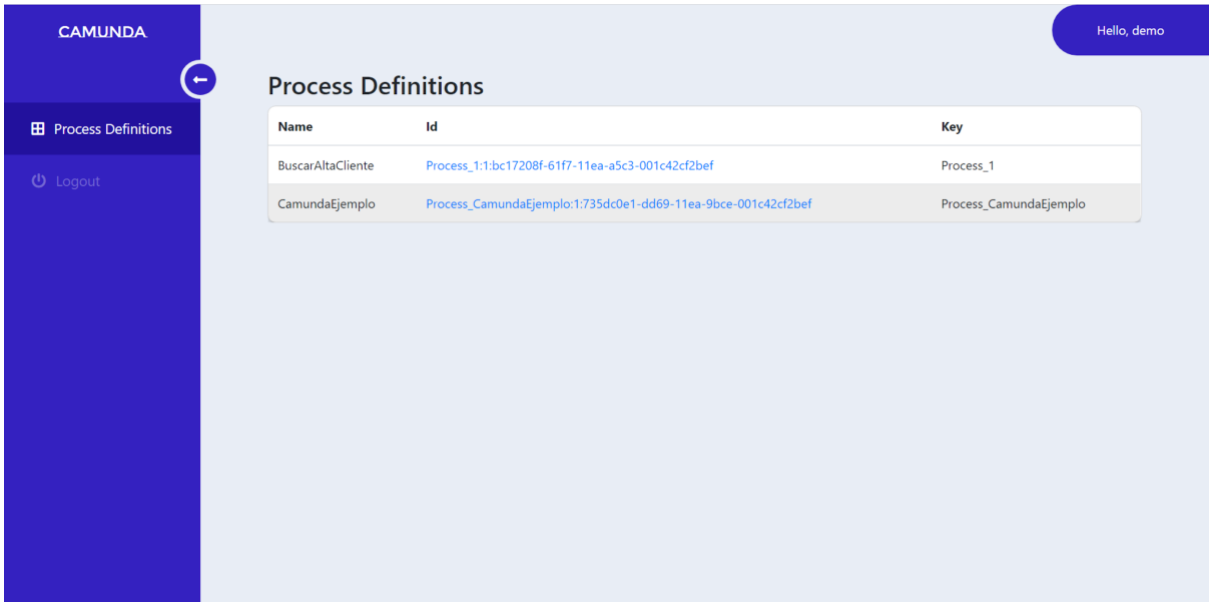


Figura 43. Pantalla Principal aplicación desarrollada

En la parte izquierda de la Figura 43 nos encontramos un menú lateral donde aparecen dos opciones, por el momento, que son Process definitions que redirecciona a la pantalla principal y el Logout que cierra la sesión y nos redirecciona al login.
En un futuro, se podrán añadir más funcionalidades de manera más sencilla, ya que se ha desarrollado la aplicación con vista a futuro, es decir, se ha desarrollado un servicio bastante genérico para facilitar el código reutilizable, aumentar la velocidad y eficiencia del tiempo de programación y mejorar la entendibilidad.

Además, se ha añadido el detalle de poder esconder el menú lateral para aumentar el tamaño de la ventana principal y otorgar más espacio a la aplicación.

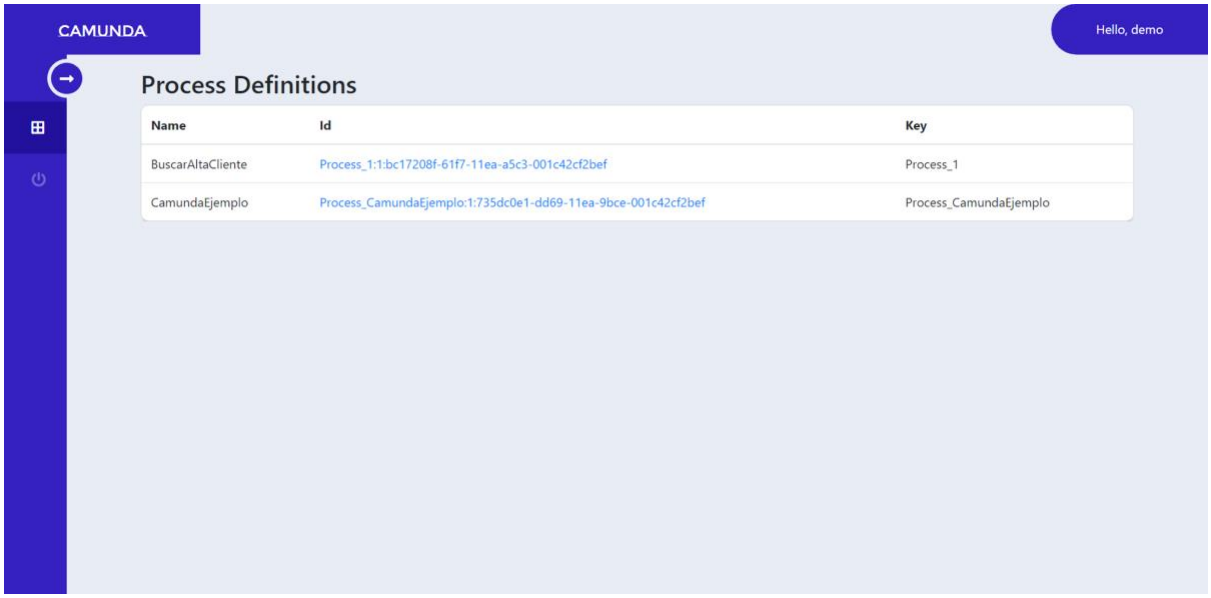


Figura 44. Menú lateral replegado

Posteriormente, podemos acceder a ver con más detalle el Process definition pulsando sobre el campo ID del Process definition deseado. En este caso, se elige la opción CamundaEjemplo que es el proyecto con el que se está realizando el ejemplo.

5.8 Process Instances

Al seleccionar el Process Definition deseado, la aplicación nos redirige a la pantalla del detalle de dicho Process Definition.

Lo primero que se observa en la pantalla, es una imagen del modelo .bpmn del proyecto que mejora tanto el aspecto visual de la aplicación como mejora la entendibilidad a la hora de ejecutar dicho Process Definition.

Posteriormente nos encontramos una lista de Process Instances.

Los Process Instances son instancias / ejecuciones del Process Definition que sirven para poder probar el modelo y su comportamiento.

Además se ha implementado la funcionalidad de poder crear instancias. Esta funcionalidad está disponible para aquellos que tienen permisos para crear instancias.

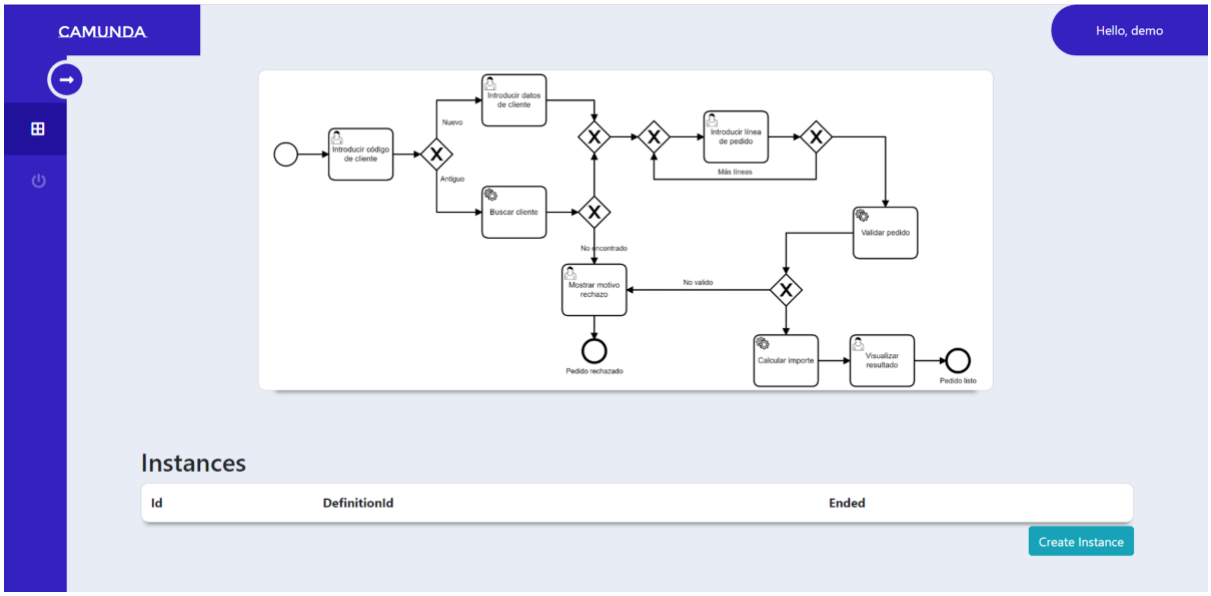


Figura 45. Pantall de detalle de un Process Definition de la aplicación desarrollada

Con la lista de process instances se realiza el mismo procedimiento que con los Process Definitions, es decir, se realiza una llamada HTTP a la REST API de Camunda para conseguir aquellos Process Instances que estén relacionados o asignados al Process Definition que se pasa como parámetro.
Los parámetros seleccionados son la id autogenerada por el servicio de camunda a la hora de la creación, el DefinitionId que hace referencia al Process Definition al que pertenece y por último Ended que indica si ha finalizado o no el Process Instance.

Para poder crear un Process instance, lo primero que debemos de asegurarnos es que el usuario identificado tenga los permisos para ejecutar dicha acción.
Si el usuario no tiene los permisos necesarios dicho botón se mostrará deshabilitado.
Esta comprobación se realiza con una llamada HTTP donde se le pasa el usuario identificado con el permiso de creación de instancias.
Posteriormente se trata la información sensible de la respuesta y se filtra para poder mostrar el botón deshabilitado o no dependiendo del usuario.

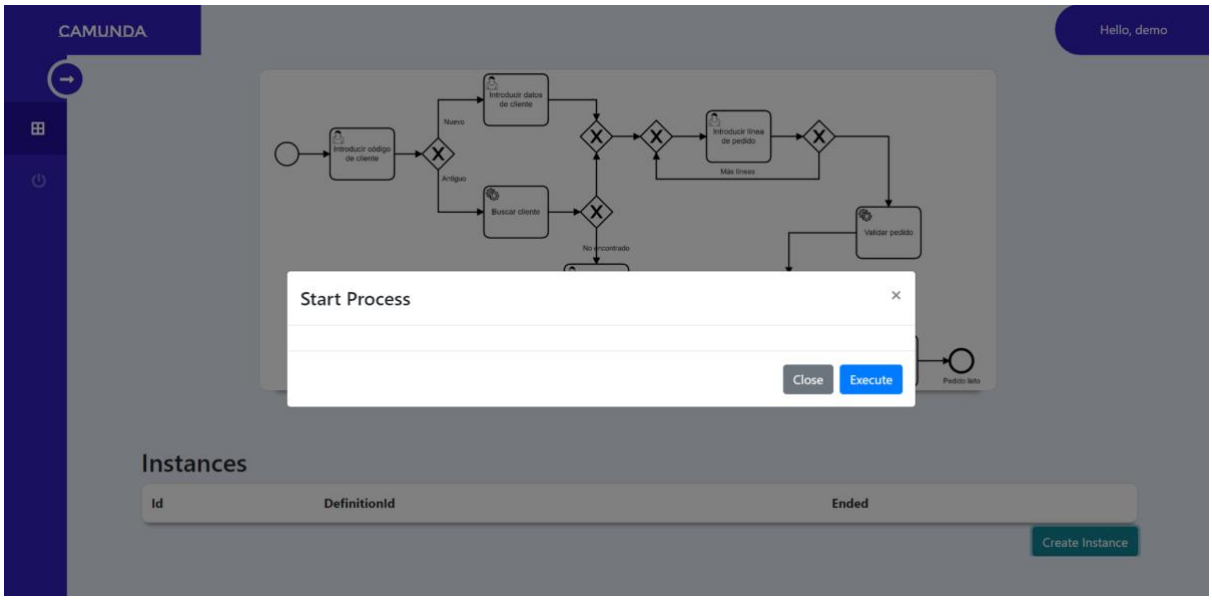


Figura 46. Formulario de creación de una instancia

Pulsando el botón designado para crear las instancias, se abre una ventana emergente donde aparecerán automáticamente todos los parámetros necesarios que se necesitan rellenar para poder crear dicha instancia, en el caso de que tuviera.

Para mostrar los parámetros se realizan dos llamadas HTTP, una que devuelve el identificador del parámetro con el tipo y la otra devuelve la estructura html que debe seguir el formulario. Con estas dos llamadas se realiza un formulario personalizado y adaptado a la tecnología Blazor.

En el ejemplo se observa que para crear una instancia no es necesario ningún parámetro.

Una vez rellenados todos los parámetros, en el caso de que los hubiere, para acabar la creación, se pulsa sobre el botón situado a la parte inferior derecha “Execute” que cierra la ventana emergente y empieza el proceso de creación de la instancia.

El botón Execute se encarga de realizar una petición al servicio de Camunda en el que se pasan los parámetros con su identificador y su valor en el formato json correcto que acepta Camunda.

Posteriormente la respuesta es tratada para filtrar primero el código de respuesta de la llamada, es decir, si ha sido un 200 OK donde se indica que la petición se ha realizado correctamente o un 400 Bad Request o 500 Internal server error, que vienen acompañados de un mensaje que especifica el error. Dicho mensaje se filtra y se inserta en las notificaciones posteriores al finalizar la ejecución de la instancia.

Si la creación ha sido correcta y el proceso ha terminado correctamente al terminar el proceso saldrá una notificación emergente indicando si ha sido correcta o no.
La notificación será de color verde cuando la ejecución ha sido correcta o roja si ha habido algún error.

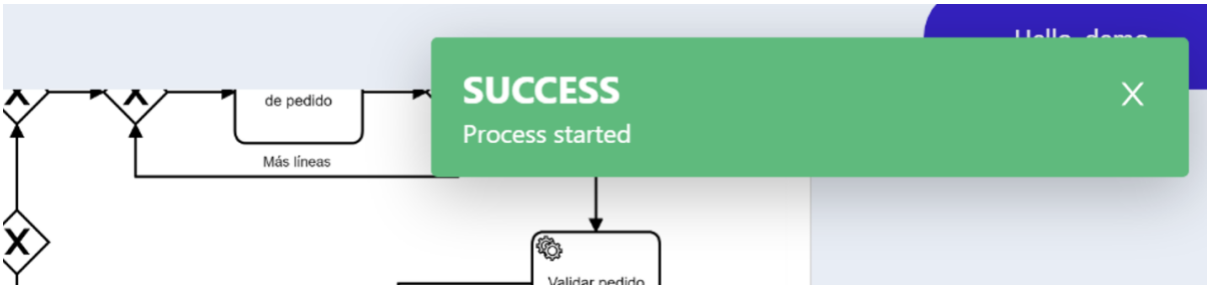


Figura 47. Notificación resultado del proceso

Una vez creada la instancia del proyecto, se hace una recarga de la pantalla para volver a hacer las llamadas y conseguir los datos actualizados. Para ver el estado o en la tarea en la que se encuentra, se pulsa sobre la id de la instancia.

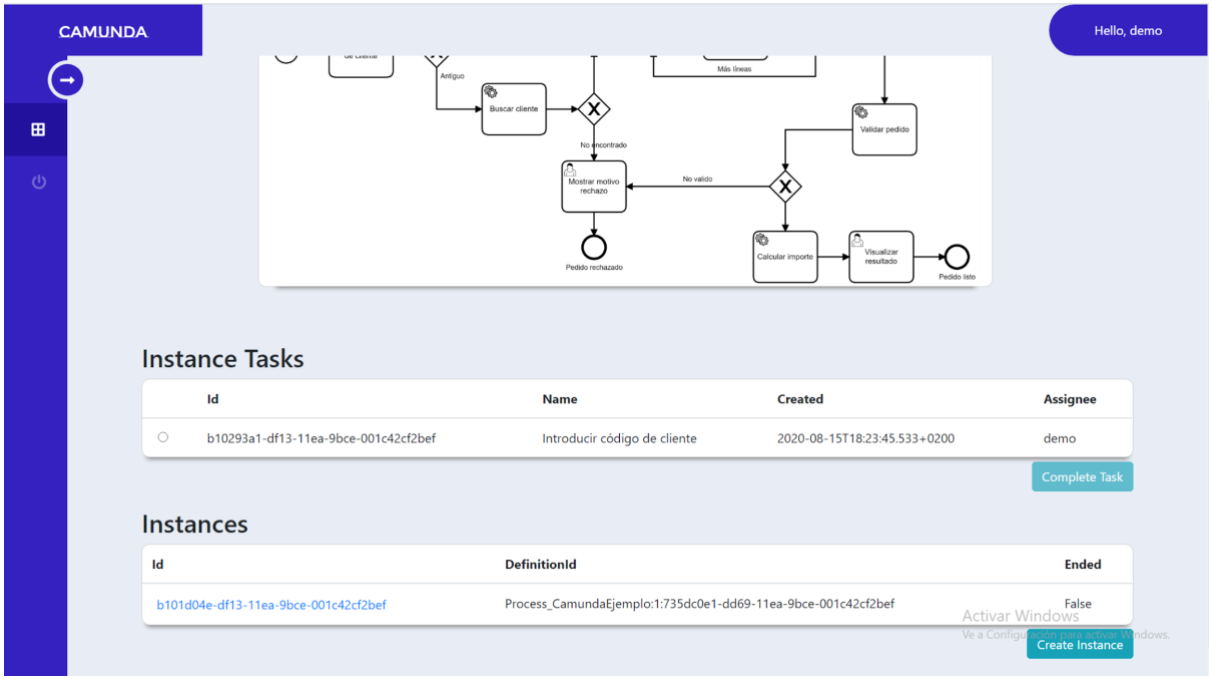


Figura 48. Lista tareas para una instancia

Pulsando el campo ID de la instancia deseada, aparece una lista donde especifica la tarea o tareas que debe ejecutar el usuario para poder continuar con la ejecución de la instancia. Puede darse el caso en el que la tarea de la instancia no esté asignada

al usuario identificado y no pueda continuar la ejecución hasta que el usuario asignado a la tarea la complete.

Dichas tareas se recogen de una petición al servicio de Camunda donde se especifica la instancia y el usuario al que tienen que estar asignadas.

La respuesta del servicio se analiza y se recogen los datos importante y que deben aparecer en la lista para ayudar a la entendibilidad de la aplicación.

Dicha lista de tareas muestra la ID autogenerada por Camunda, el Nombre, la fecha de creación de la tarea y el usuario a la que está asignada la tarea.

Debajo de la lista aparece un botón desactivado para la ejecución de las tareas.

Para activar dicho botón se tiene que seleccionar la tarea que se quiera ejecutar. Una vez seleccionada, el botón se activa.

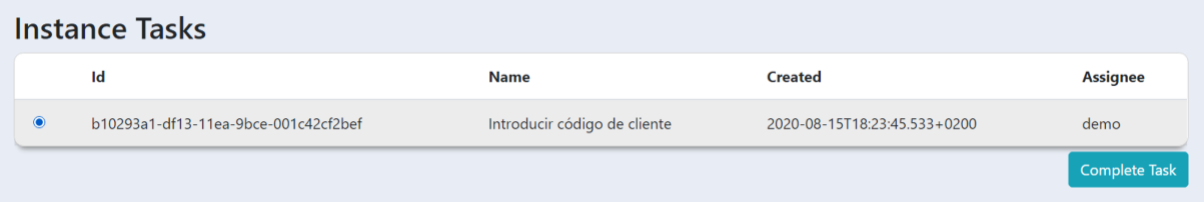


Figura 49. Selección tarea para su ejecución

Al seleccionar la tarea deseada a ejecutar y activarse el botón “Complete task”, se pulsa sobre dicho botón para empezar el proceso de ejecución y como en la ejecución de los Process Instances se abre una ventana emergente con los parámetros necesarios para poder ejecutar esa tarea.

Figura 50. Formulario de ejecución de tareas

Para poder mostrar automáticamente los parámetros que necesita la tarea seleccionada, se realizan dos llamadas HTTP igual que con los process Instances pero a diferente dominio, es decir, para conseguir los parámetros de la instancia se llama al dominio process Definition y para las tareas se llama al dominio Task.
Dichas respuestas son utilizadas para la construcción de un formulario personalizado y adaptado para el framework de Blazor.
Una vez rellenados dichos parámetros con los datos deseados, se pulsa sobre el botón situado en la parte inferior derecha “Execute” el cual realiza la llamada a la Api REST de Camunda y cierra la ventana emergente.
Si el resultado de la ejecución es correcto o no, se le notifica al usuario mediante una notificación en la parte superior derecha. Si el resultado es correcto el color de la notificación será verde, si es incorrecto el color será rojo.

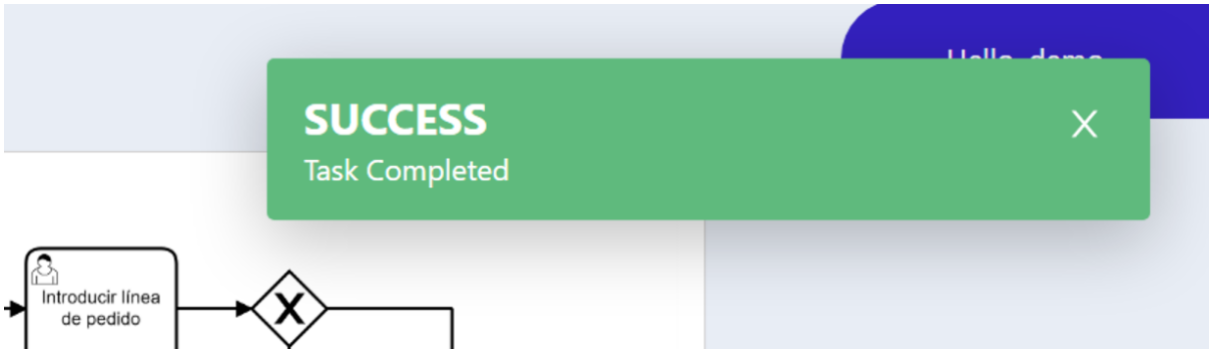


Figura 52. Notificación de éxito en el proceso

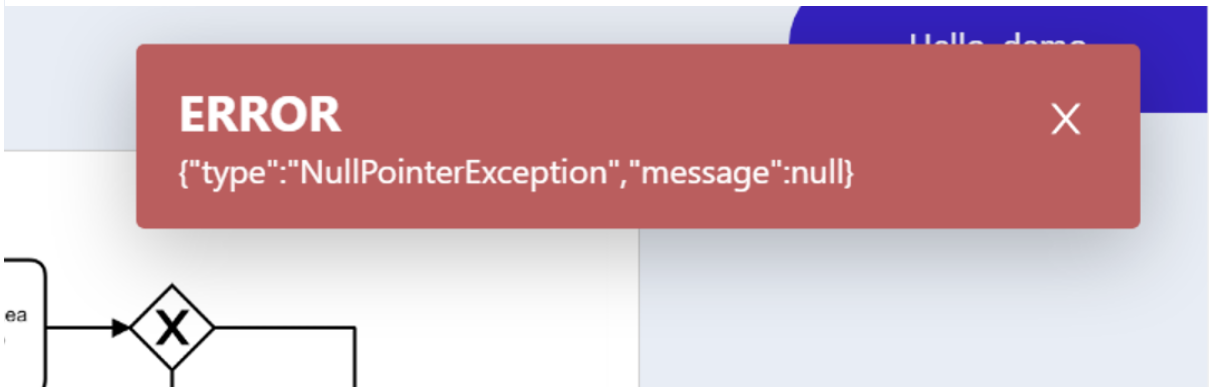


Figura 51. Notificación de error en el proceso

El mensaje de error de la imagen es debido a que un parámetro se ha pasado como nulo. Este mensaje proviene de la respuesta dada por el servicio de Camunda a través de una petición HTTP a dicho servicio.

Los pasos a realizar para cada tarea son los mismos, excepto los parámetros que necesita cada tarea, dichos parámetros varían dependiendo de las necesidades de la tarea.

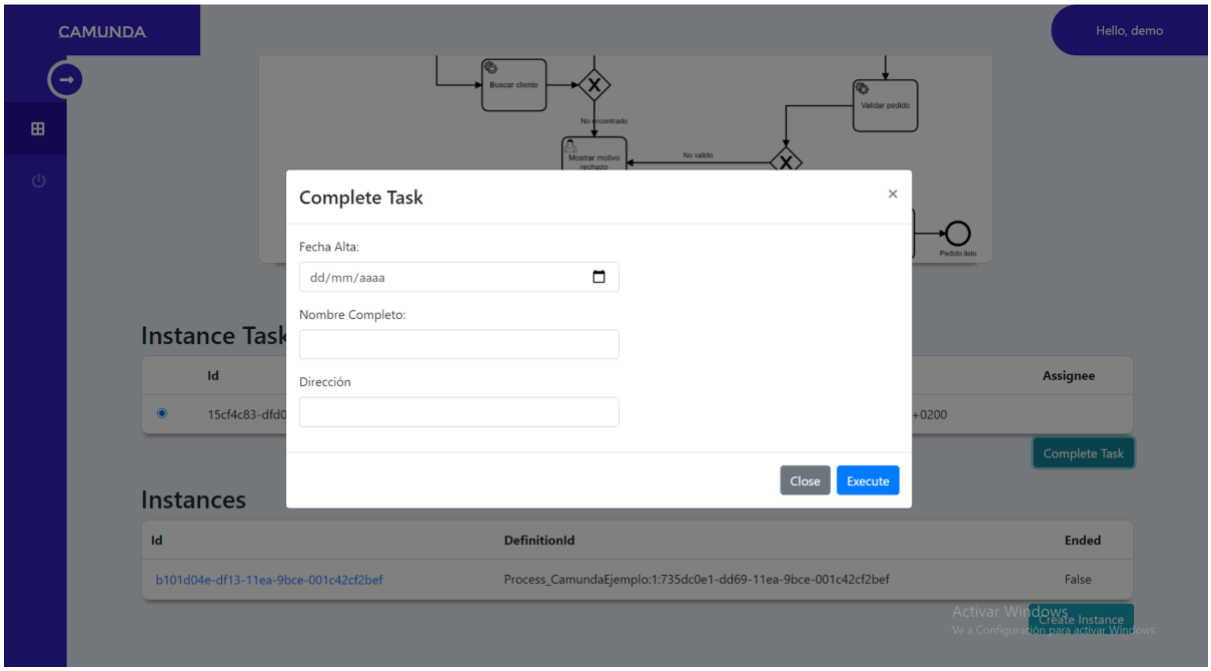


Figura 53. Formulario de ejecución de tareas (Ejemplo)

Una vez se han ejecutado todas las tareas de la instancia, dicha instancia se cierra y desaparece de la lista de instancias del proyecto.

6. Conclusión

Con el desarrollo de este proyecto C#, se dispone de una herramienta personalizada para la gestión y ejecución de modelos .bpmn a través de Camunda.
Esta aplicación ofrece una interfaz más entendible, es decir, es una interfaz que conlleva poco tiempo de aprendizaje.
Además, ofrece una interfaz personalizada para la ejecución e instanciación de procesos, es decir, ofrece una amplia gama de tipos de variables para su ejecución, ya que como se ha comentado anteriormente, el motor de Camunda acepta todo tipo de objeto siempre que se le pase con el formato correcto, pero por el contrario su explorador ofrece un número limitado de tipos de variables para su ejecución.

6.2 Planes a futuro

La aplicación está desarrollada de forma que permite la implementación de nuevos servicios de Camunda de manera sencilla y rápida, ahorrando tiempo y mejorando la efectividad.
De momento, la aplicación ofrece una interfaz de gestión y ejecución de procesos BPMN, pero en un futuro se puede desarrollar un sistema de gestión de usuarios, es decir, una interfaz capaz de gestionar los usuarios creados en Camunda. Con ello se podrían crear usuarios, eliminarlos, además de la creación de grupos de usuarios.
También se podría implementar un servicio de gestión de permisos y autorizaciones, es decir, asignar o quitar permisos a los distintos usuarios. Otra funcionalidad sería en lugar de asignar permisos a los usuarios, asignarlos a los grupos de usuarios, es decir, dependiendo del rol que vayan a desempeñar los usuarios de un mismo grupo, asignar a dicho grupo una serie de autorizaciones.

En conclusión, esta aplicación permite la gestión y ejecución de procesos bpmn en C#, lenguaje que Camunda no ofrece implementación, aparte de ofrecer un servicio para la comunicación con la API REST. Es una aplicación útil, flexible y fácil de utilizar que permite a los usuarios que la utilicen en un futuro comprobar de manera más sencilla sus procesos y la gestión de los mismos.

7. Bibliografía

7.1 Referencias Web

Las referencias utilizadas para la realización de este proyecto son:

- Blazor Microsoft (27 de julio de 2020) Microsoft. https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor
  Esta página web se ha tomado como referencia para el aprendizaje del lenguaje y para las dudas durante el desarrollo de la aplicación.
- Camunda Documentación (30 de julio de 2020) Camunda. https://docs.camunda.org/manual/7.13/
  Esta página web se ha tomado como referencia para el aprendizaje de las web applications disponibles en Camunda, así como fuente de información para la utilización de los distintos endpoints utilizados en la aplicación.
  Además, también se ha utilizado como guía para el desarrollo del servicio de conexión que utiliza la aplicación.
- API REST (15 de junio de 2020) GitBooks. https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html
  Esta página web se ha utilizado como referencia para el aprendizaje de los estándares de API REST y la comunicación entre las API REST y las aplicaciones web.
- Blazorise Documentación (10 de julio de 2020) Blazorise. https://blazorise.com/docs/
  Esta página web se ha utilizado como fuente de estilos, es decir, que esta librería se utiliza como guía base de estilos de los componentes.
- BPMN Documentación (20 de julio de 2020) BPMN. http://www.bpmn.org/
  Esta página web se ha utilizado para el aprendizaje y como fuente de información para consultar dudas sobre la notación.

7.2 Referencias Bibliográficas

Las referencias bibliográficas utilizadas en este proyecto son:

- Morales R., (2016). *Fundamentos de BPMN: Una guía básica para el diseño de procesos,* Ediciones Reiner.
- Hitpass, B, (2011). *BPMN 2.0 Manual de Referencia y Guía Práctica,* BPM Center.