**UNIDAD ACADÉMICA:** Ingeniería de sistemas.

**CURSO:** Curso de Profundización: Software 1.

**PRÁCTICA Nº XX:** Implementacion Worker

# OBJETIVOS

* + *Implementar un Worker mediante una Service Task*

# CONSULTA PREVIA

* + *Clase teórica de la Docente*

# EQUIPOS, MATERIALES Y REACTIVOS

Se debe listar los equipos, materiales y reactivos requeridos para el desarrollo de la práctica.

|  |  |  |
| --- | --- | --- |
| **Equipos** | **Materiales** | **Sustancias y/o Reactivos** |
| Computadores | Camunda Modeler Camunda Run Node JS  Visual Studio Code |  |

# PROCEDIMIENTO O METODOLOGÍA

Ahora, haciendo uso de la herramienta Camunda Modeler vamos a desarrollar el siguiente ejercicio donde se aprenderá a crear e implementar un **worker.**

**Nial nos propone crear un modelo con fin de ayudar a Marie Joseph Paul Yves Roch Gilbert du Motier, Marques de La Fayette a escapar de Francia.**

**Todo inicia con el interés del Marques de La Fayette de escaparse de Francia, pero pronto se da cuenta que necesita prepararse para la partida. Al darse cuenta de que no puede hacerlo solo, busca la ayuda de su amigo Jean Xavier o de Austria. Una vez que recibe la ayuda, el Marqués de La Fayette cumple un tiempo en el extranjero y esta travesía termina con el Marques de La Fayette sintiéndose seguro.**

Se inicia creando una carpeta “**Guia Lab**”, en ella se crearán 3 carpetas más:

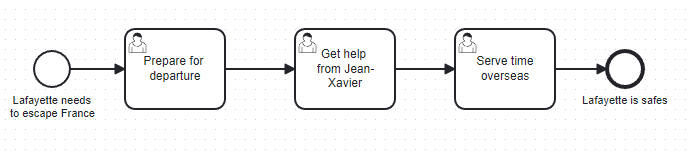
* Forms: donde se guardarán todos los Formularios.
* Models: donde se guardarán todos los Modelos.
* Workers: donde se guardarán todos los Workers.

**Camunda Run:**

Se abre IntelliJ y se despliega la máquina de Camunda integrada con SpringBoot, una vez desplegada y logeado se procede a abrir Camunda Modeler.

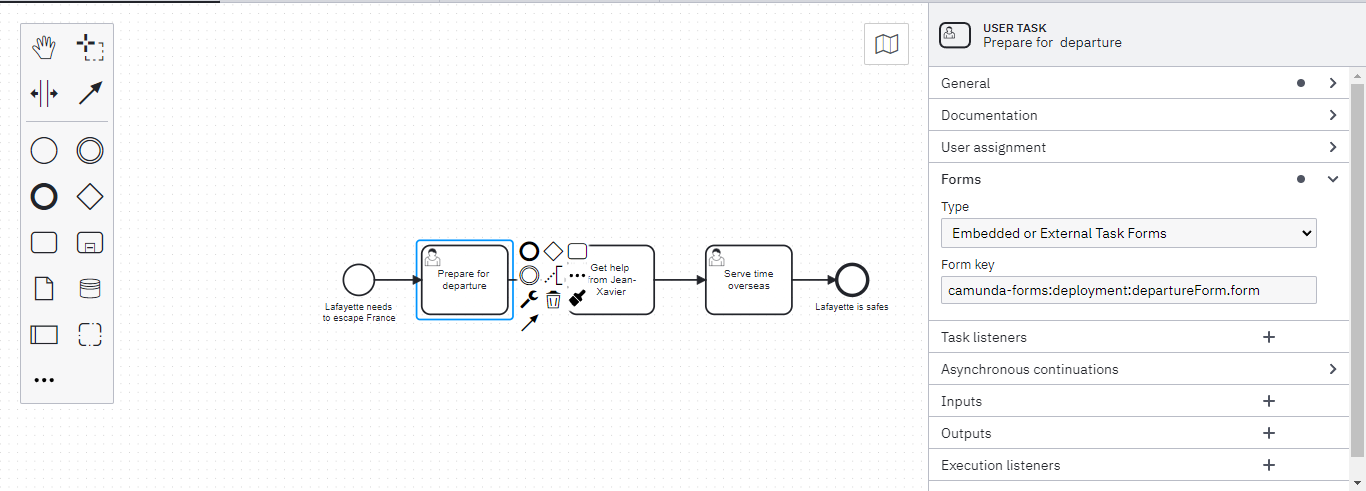
**Modelo:**

Se abre el Camunda Modeler y crea el siguiente modelo:



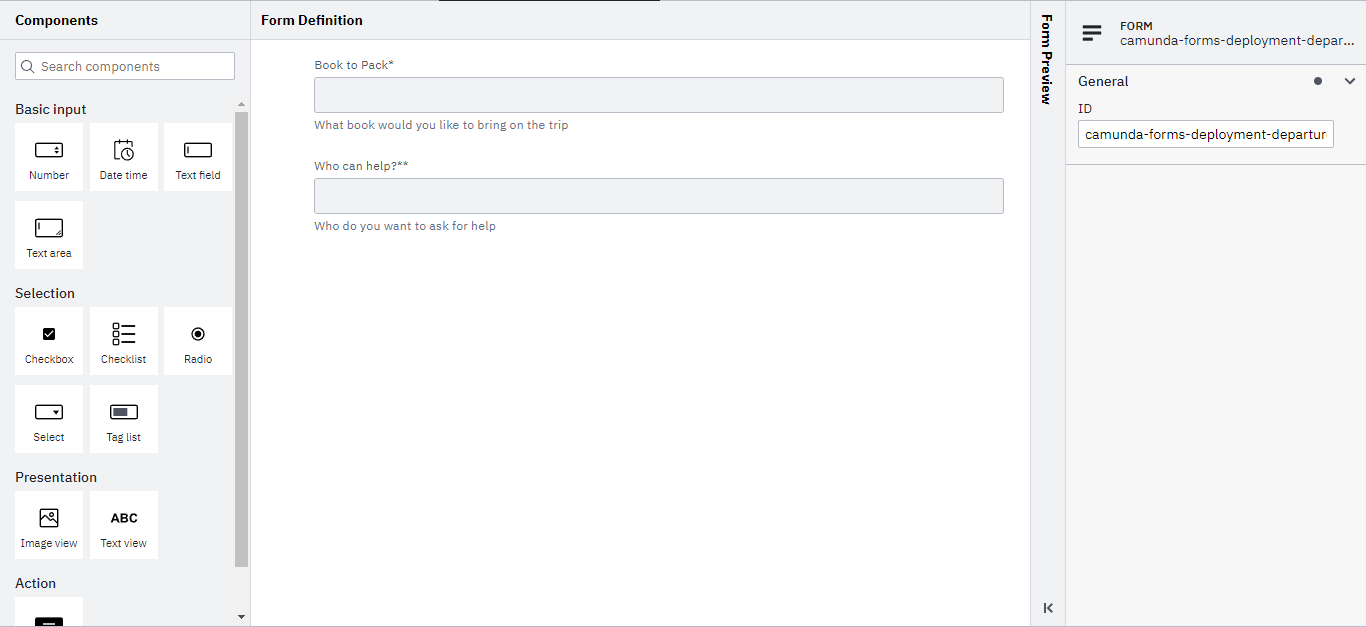
***Figura 01. Modelo Inicial***

Para este ejercicio haremos uso de formularios **Embebidos o Formularios Externos de Camunda (Embedded or External Task Form)**



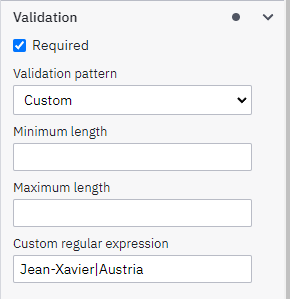
***Figura 02. Implementación Formulario Embebido***

Para el formulario “***departureForm.form***”, se crearán 2 campos de tipo texto y se le asignara una ***key*** a cada uno: ***book*** y ***help*** respectivamente

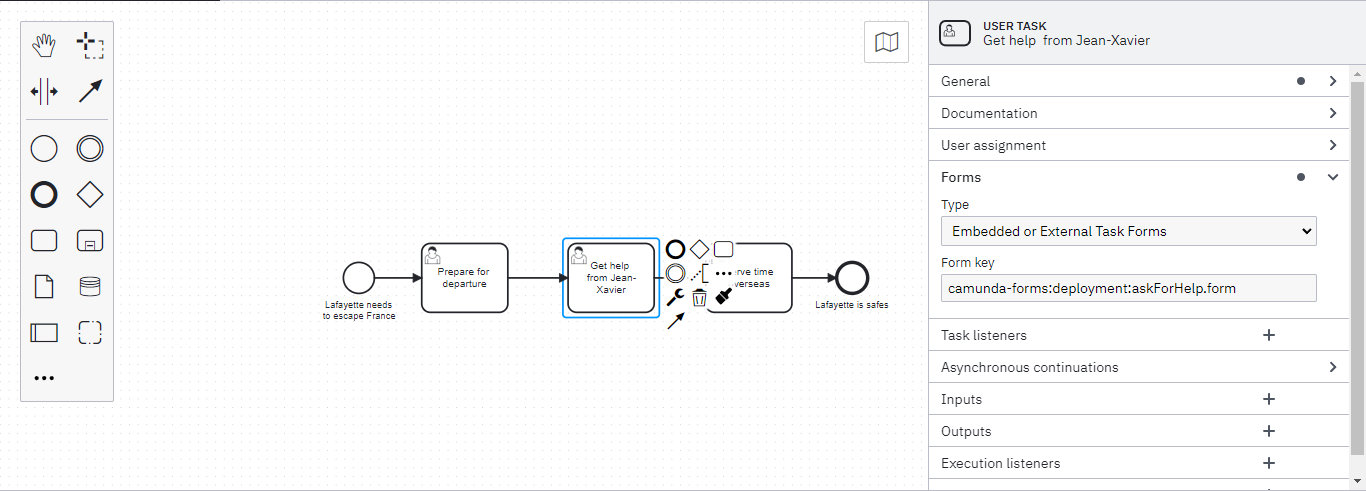


***Figura 03. Creación Formulario***

A este ultimo campo de texto se le configura una validación de tipo personalizada o ***Validation*: *Custom***, se marca la casilla ***Required*** y se asigna la siguiente expresión regular personalizada:

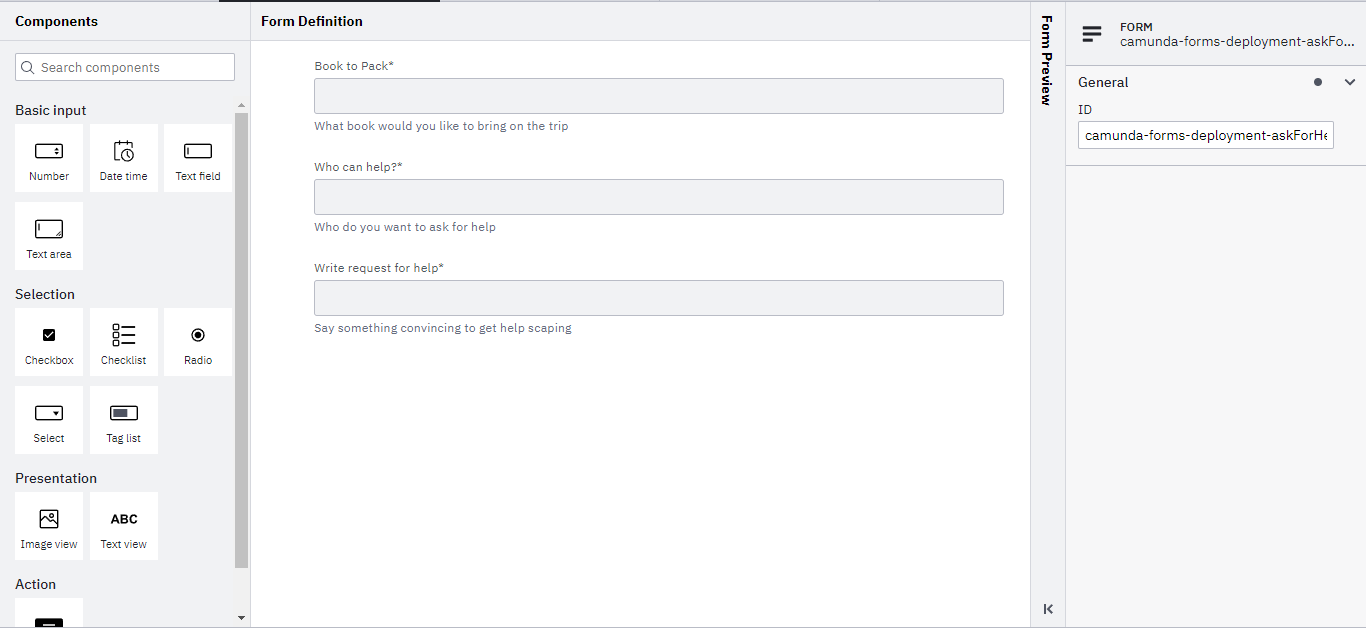


***Figura 04. Asignación de una validación en un campo especifico***



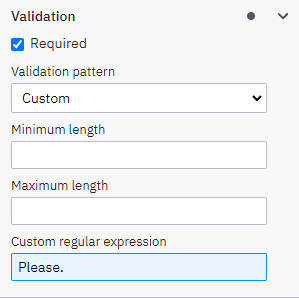
***Figura 05. Implementación Formulario Embebido***

Para el formulario “***askForHelp*.*form***”, se crearán 3 campos de tipo texto y se le asignara una ***key*** a cada uno: ***book***, ***help*** y ***helpNote*** respectivamente



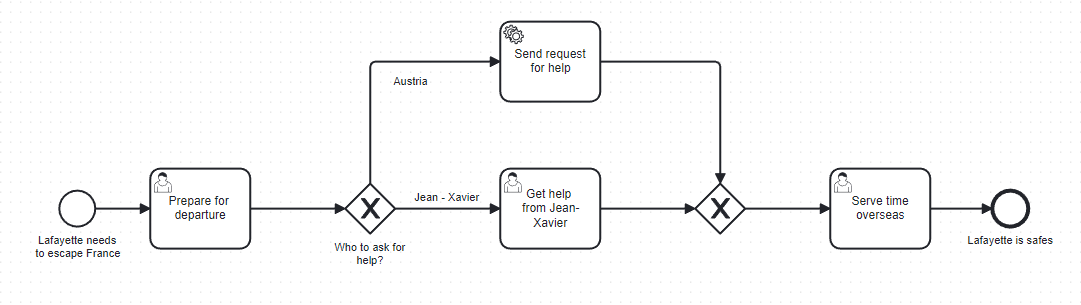
***Figura 06. Creación Formulario***

A este último campo de texto se le configura una validación de tipo personalizada o ***Validation*: *Custom***, se marca la casilla ***Required*** y se asigna la siguiente expresión regular personalizada:



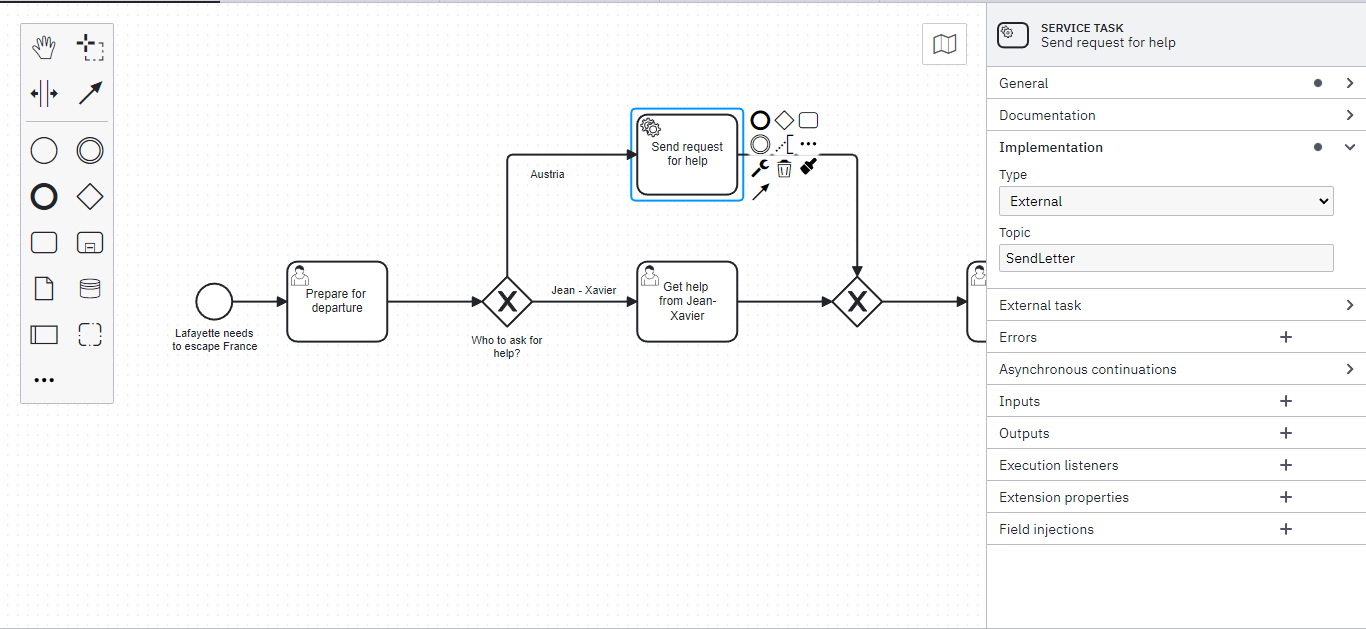
***Figura 07. Asignación de una validación en un campo especifico***

Una vez se termina de crear el modelo anterior se procede a realizar la siguiente modificación a dicho modelo:



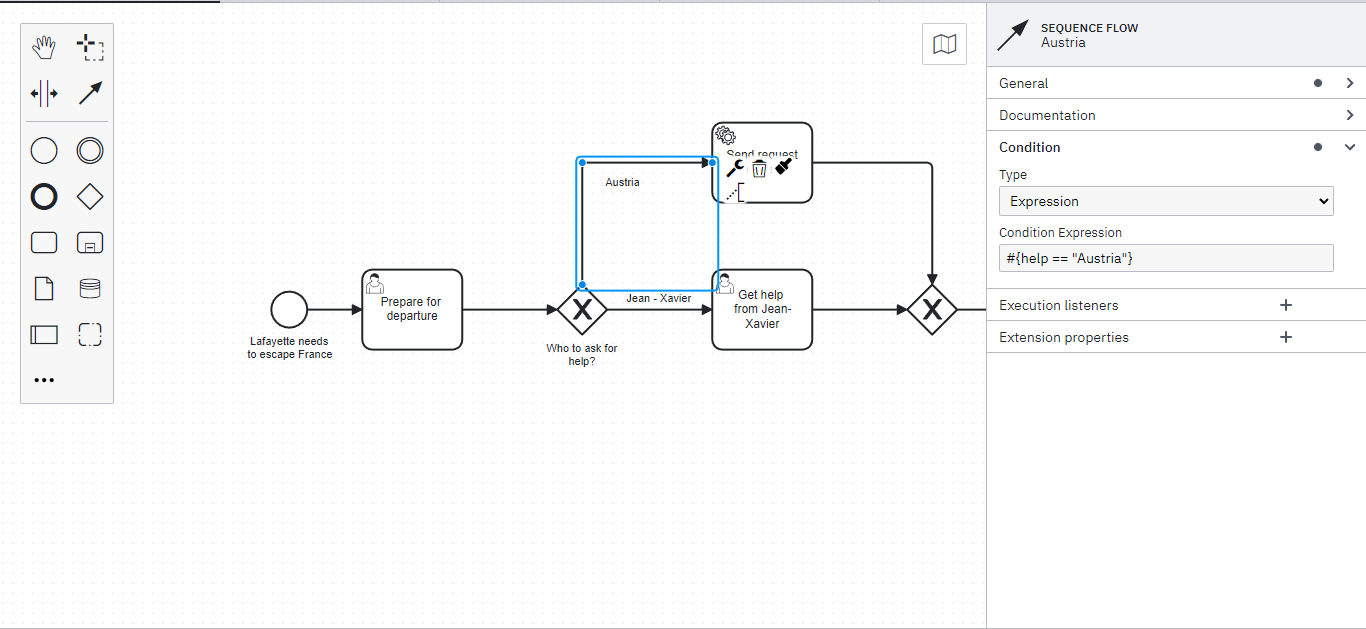
***Figura 08. Modelo Inicial Modificado***

Donde se le ha agregado una compuerta **XOR** y una actividad de tipo ***Service Task*** esta actividad de tipo Service Task se configura con el tipo de implementación Externa o ***Implementation*: *External***, asignando en el campo ***Topic*** la palabra ***SendLetter*.**



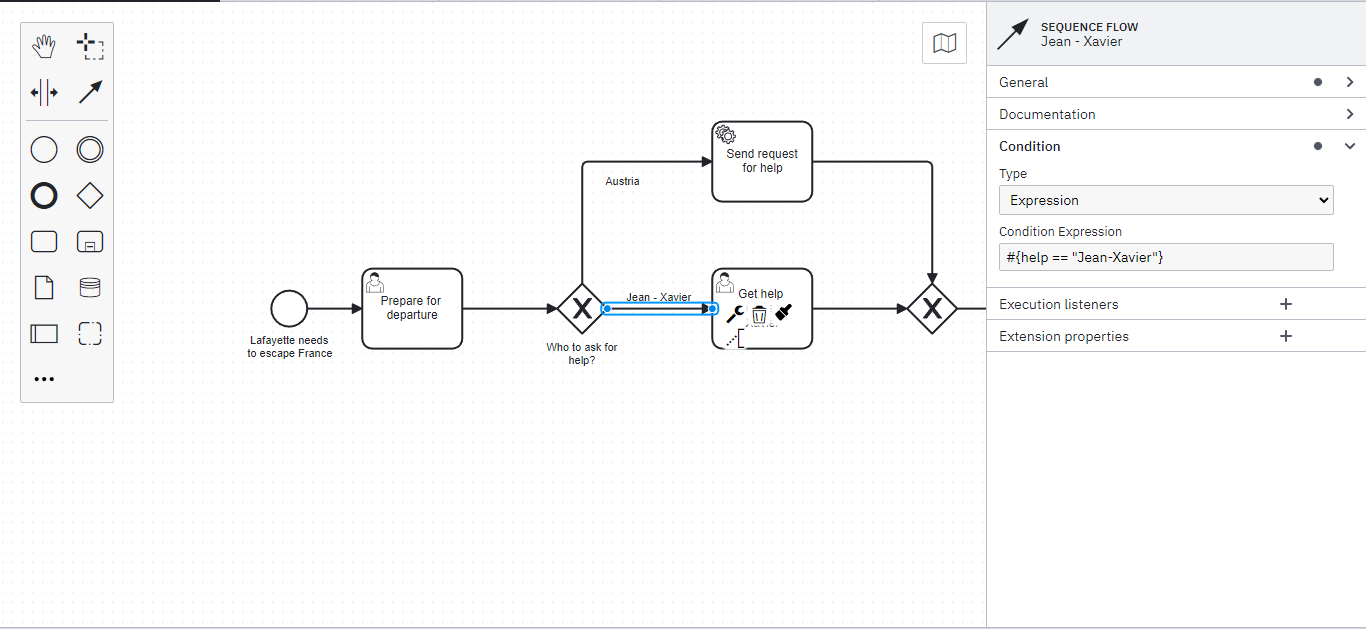
***Figura 09. Asignación de Worker mediante Service Task***

Seguido a esto se configura el flujo de la compuerta XOR, asignando una condición de tipo Expresion o ***Expression*** y colocando la siguiente condición.



***Figura 10. Asignación de Condición para el Flujo de la Compuerta***

De la misma forma se configura el otro flujo restante asignando la siguiente condición:



***Figura 11. Asignación de Condición para el Flujo de la Compuerta***

Una vez terminada la implementación y configuración de la nueva tarea y compuerta, se procede a abrir Visual Studio Code.

***Worker*:**

Se abre el Visual Studio Code y se crea un archivo JavaScript **(*Los* *Worker* *no* *están* *sujetos* *a* *un* *lenguaje* *en* *específico*)**.

Importamos la siguiente librería:

“***import { Client, Variables, logger } from "camunda-external-task-client-js";***”

Esta librería lo que proporciona es la posibilidad de conectarnos a la máquina de Camunda (Camunda Run), se está importando específicamente “Client”, “Variables” y “logger”.

* Client: con ese objeto se accede al proceso mediante una instancia y se suscribe a una lista de pendientes.
* Variables: con ese objeto se logra crear variables para luego mandarlas a la instancia del proceso desplegado.
* logger: con ese objeto se le da permiso al worker para acceder a la máquina de Camunda (Camunda Run).

Importada la librería se procede a crear la instancia del proceso:

***// configuration for the Client:***

***//  - 'baseUrl': url to the Process Engine***

***//  - 'logger': utility to automatically log important events***

***const config = { baseUrl: "http://localhost:8080/engine-rest", use: logger };***

***// create a Client instance with custom configuration***

***const client = new Client(config);***

Una vez creada la instancia del worker, se le asigna a que topic estará atendiendo mediante la siguiente línea de código:

***client.subscribe("SendLetter", async function({ task, taskService }) { … });***

Una vez suscrito al topic “***SendLetter***”, se procede a realizar la lógica con la que se manejaran las instancias que lleguen al Worker con dicho topic.

Dentro de la función asíncrona, colocar el siguiente código:

Se obtiene la variable “book” que corresponde al titulo del libro y se imprime por consola:

***const bookTitle = task.variables.get("book");***

***console.log("\*\* Reminder to Read: "+ bookTitle + "\*\*");***

Luego se procede a crear una variable que se enviara a la instancia del proceso:

***const austriaResponse = "Sorry Laff can't help, BTW. Prussia wants a quick word with you.."***

***const processVariables = new Variables();***

***processVariables.set("austriaResponse", austriaResponse);***

Y por ultimo se completa la actividad en la instancia del proceso de manera asíncrona:

***await taskService.complete(task, processVariables);***

Terminada la lógica del Worker, se procede a guardar el archivo y abrir una consola de comandos donde se ejecutará el siguiente comando, con el fin de instalar la librería “camunda-external-task-client-js”.

***“npm install -s camunda-external-task-client-js”***

Instalada la librería se procede a abrir el archivo “***Package.json***” y se coloca la siguiente configuración, con la que se le permitirá trabajar con los módulos de la librería:

***{***

***"type": "module",***

***"dependencies": {***

***"camunda-external-task-client-js": "^3.0.1"***

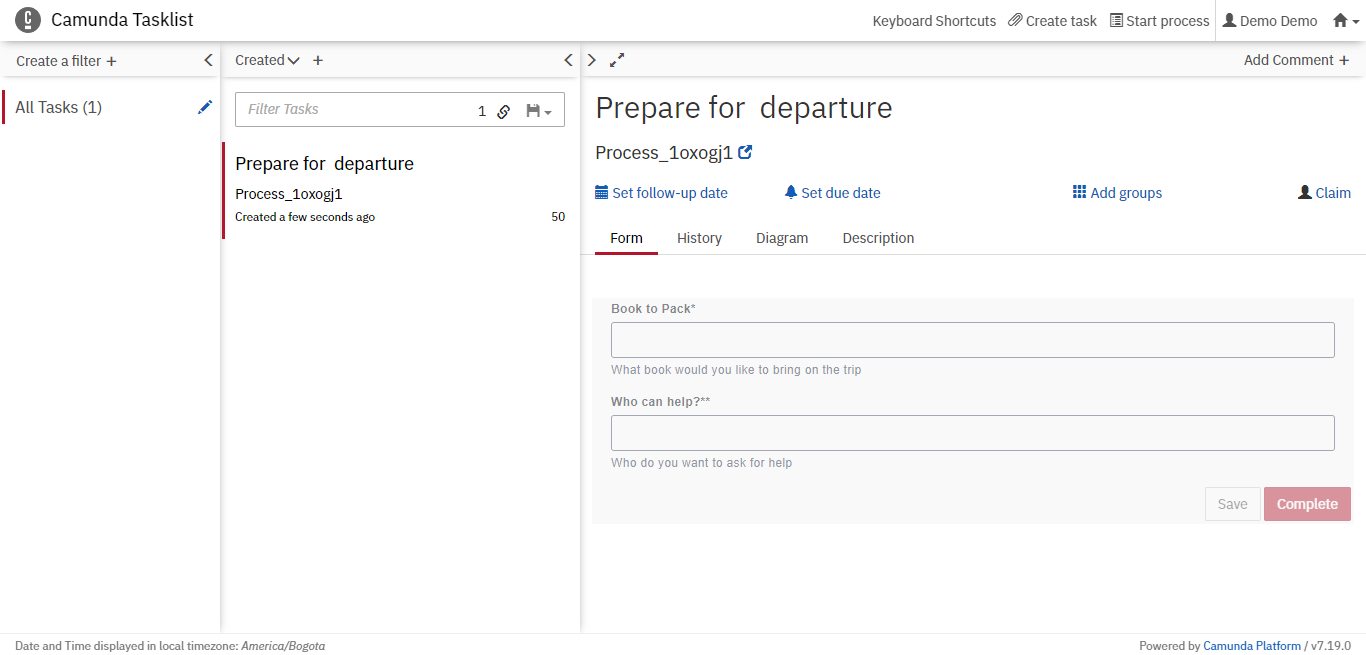
***}***

***}***

Realizada la configuración en el archivo “***Package.json***”:

* Se despliega la máquina de Camunda integrada con SpringBoot.
* Se despliega el modelo hecho en Camunda Modeler, incluyendo ambos formularios.
* se procede a ingresar a la carpeta Workers donde se guardó el archivo JS y se corre el siguiente comando **“*node .\nombreArchivoJS.js*”**.

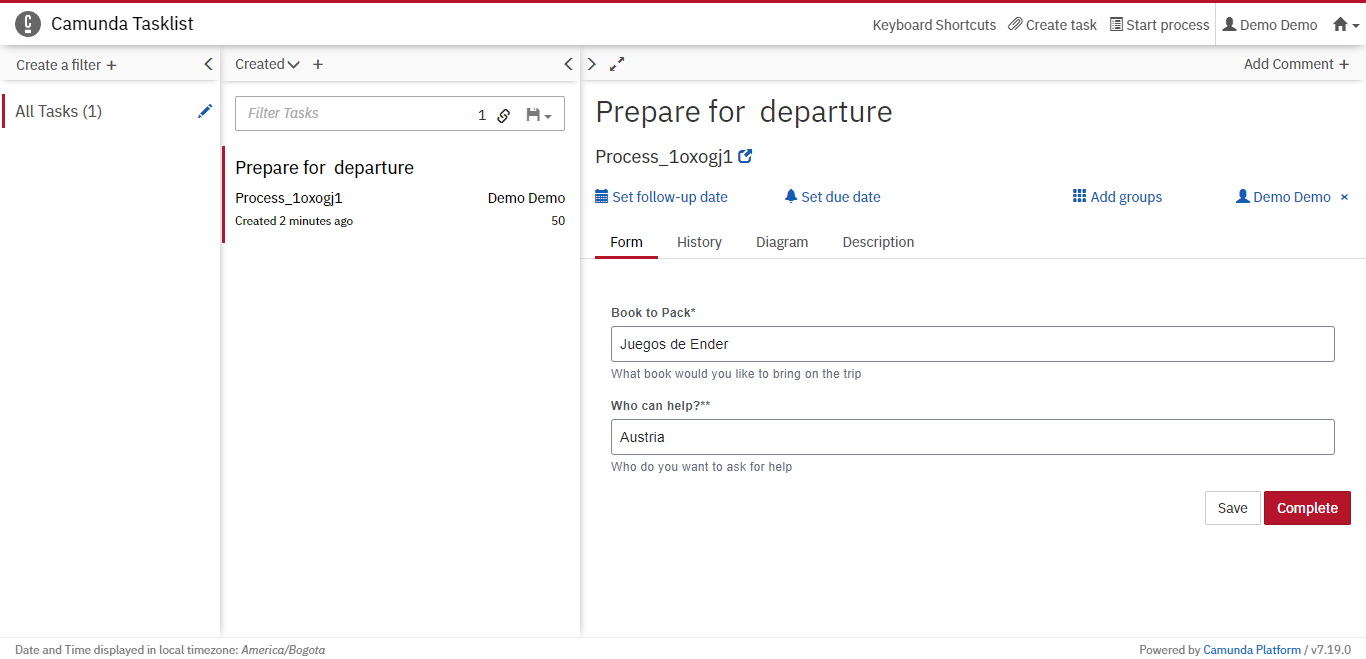
Se procede a crear una instancia del proceso:



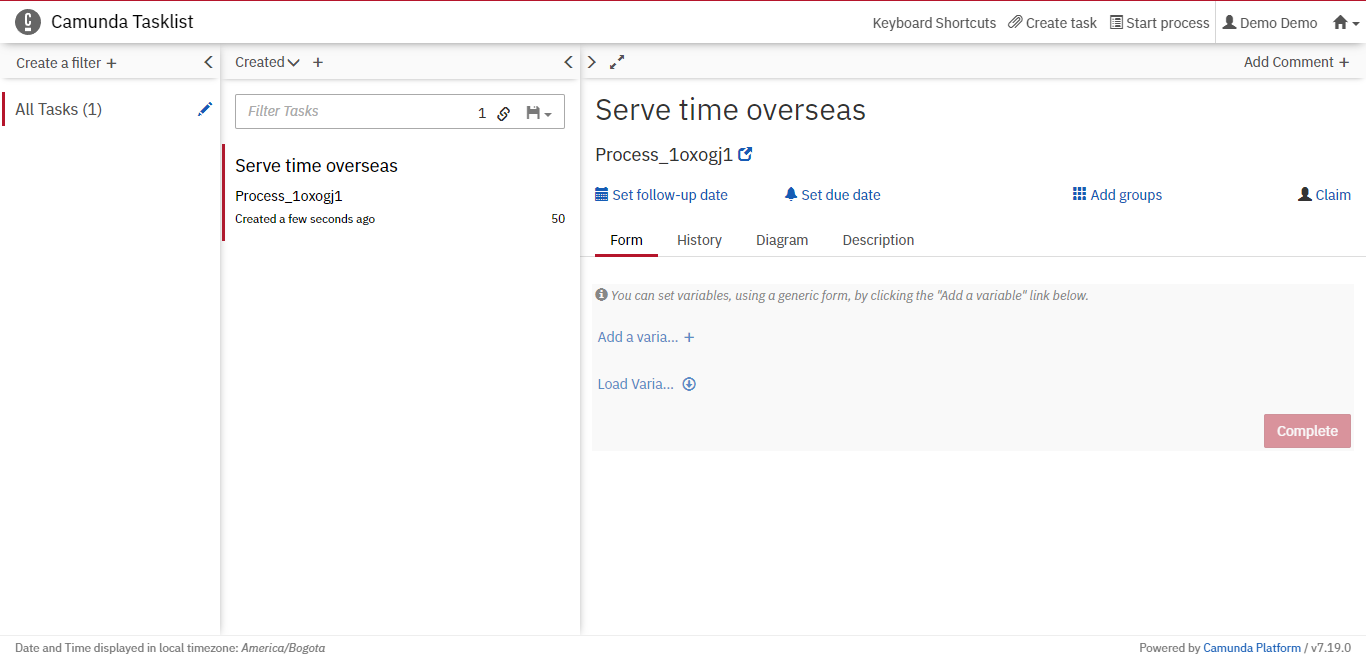
***Figura 12. Creación de una Instancia del Proceso***

En caso de colocar en el segundo campo la palabra “Austria”, se asignará al Worker la actividad de la instancia, en caso contrario, se le mostrará el siguiente formulario.

**Austria:**

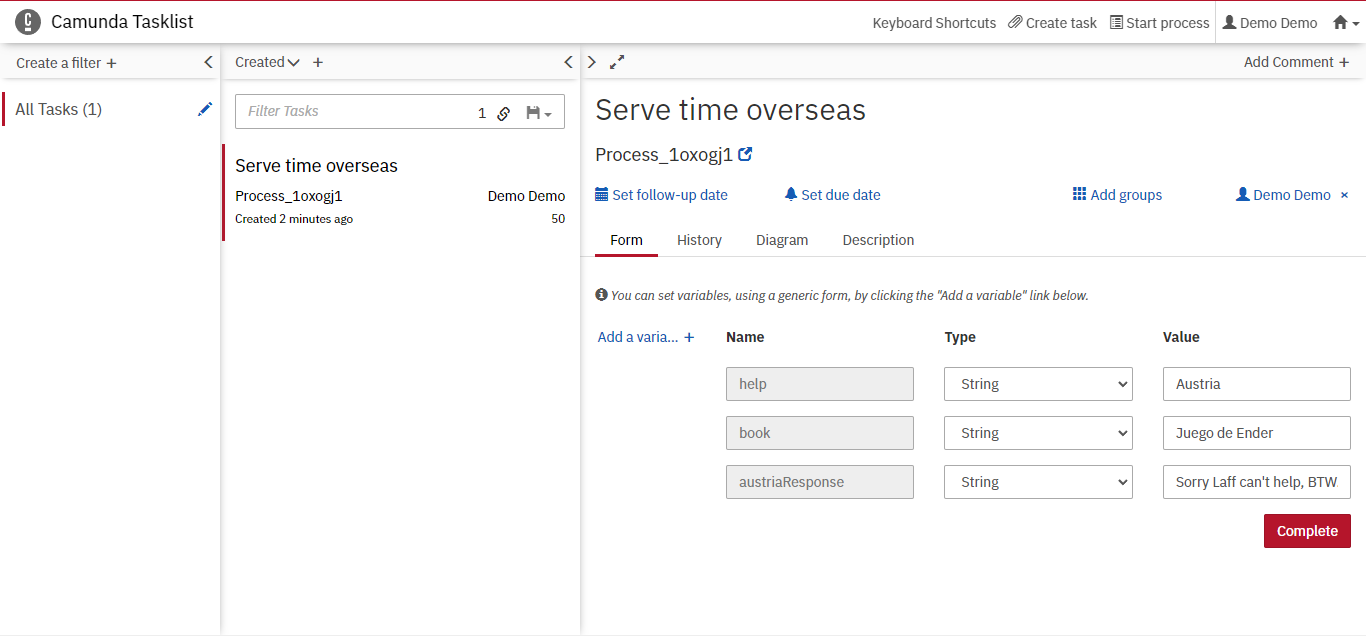


***Figura 13. Primer Flujo de la Compuerta***



***Figura 14. Actividad de Visualización de Variables con Intervención del Worker***

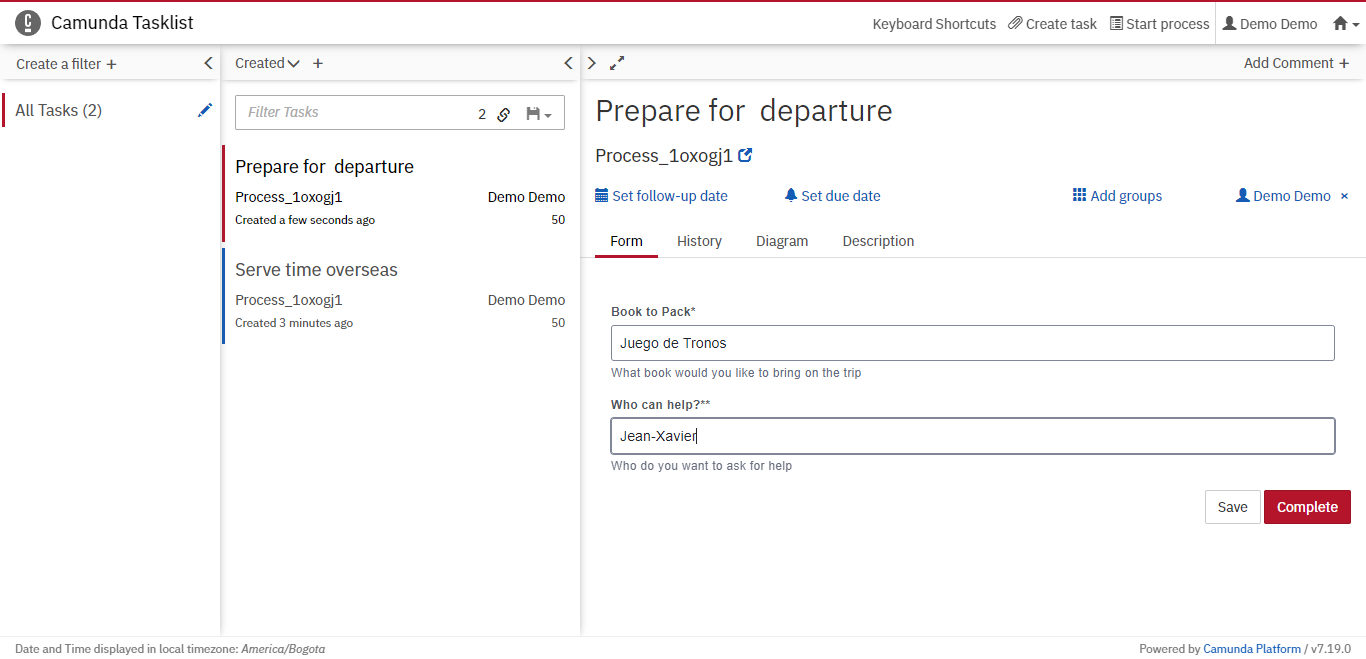
Para este caso, se le dará click en “***claim***” y luego en “***Load Variables*”** esta opción lo que hace es mostrarnos todas las variables del proceso.



***Figura 15. Actividad de Visualización de Variables***

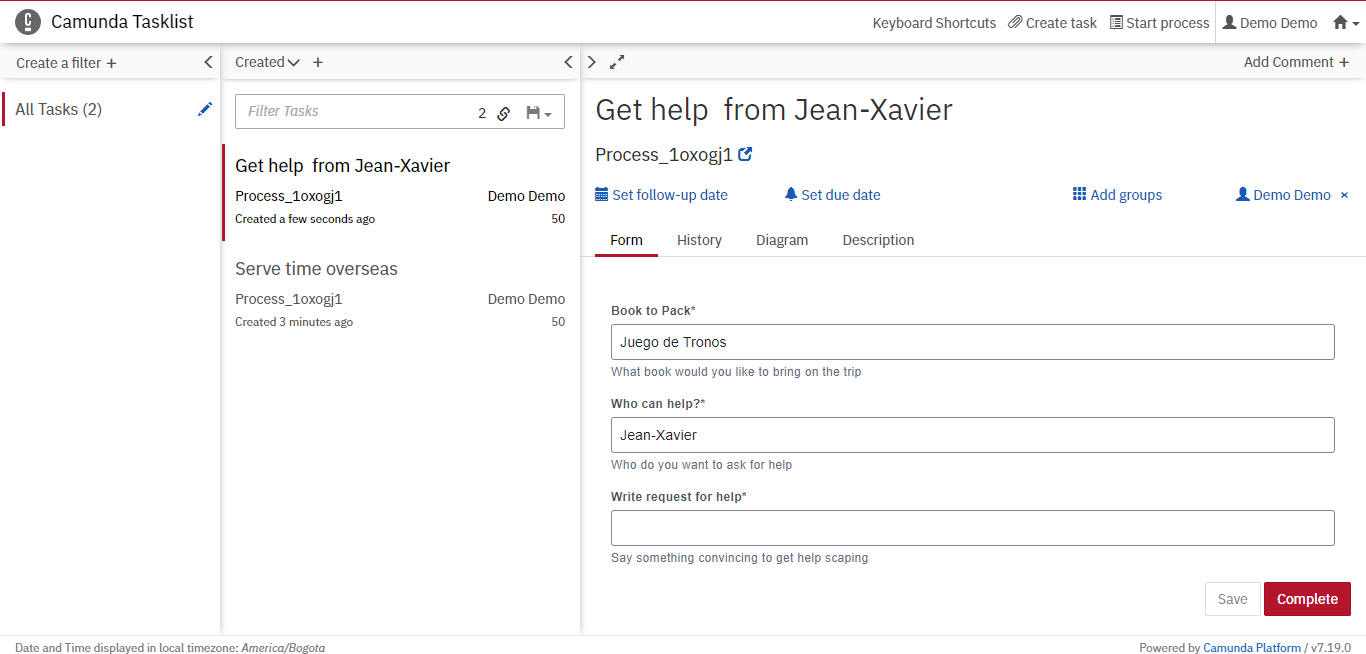
En ese apartado se logra observar la variable que únicamente se crea mediante el worker junto a su mensaje.

**Jean-Xavier:**

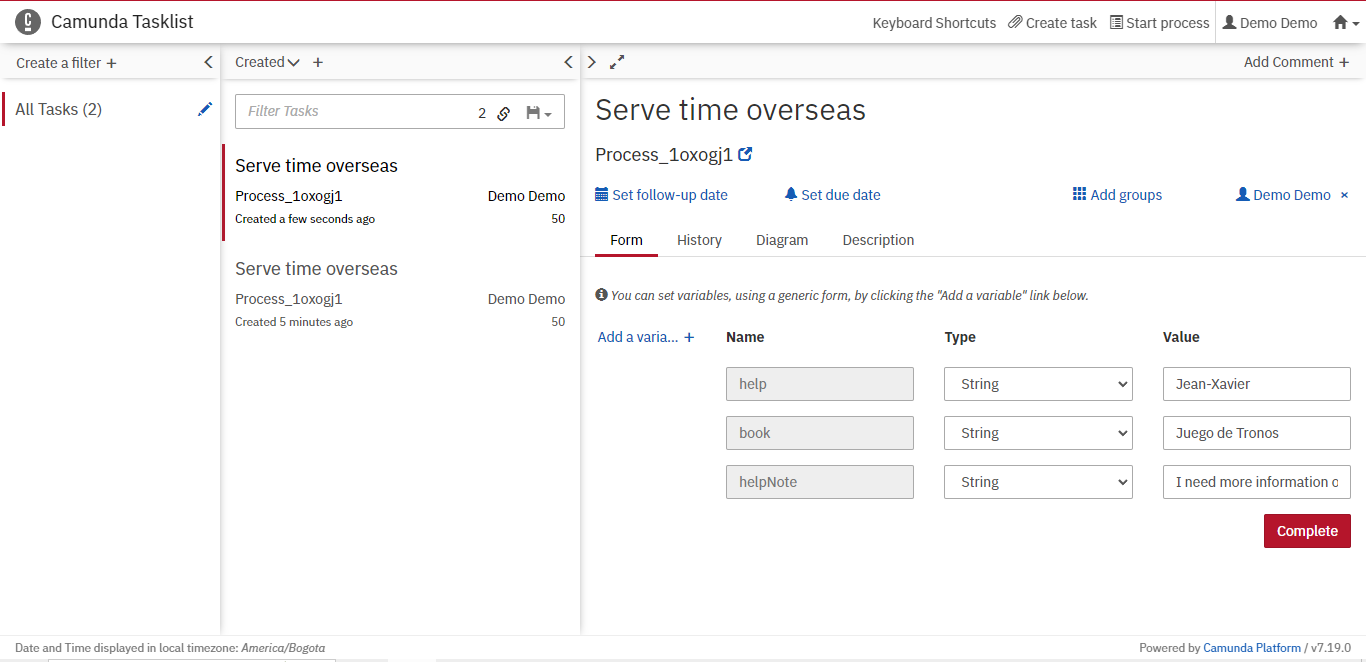
****

***Figura 16. Segundo Flujo de la Compuerta***

Con esta opción se nos muestra un formulario que no indica colocar la solicitud de ayuda:



***Figura 17. Flujo Donde no Interviene el Worker***



***Figura 18. Actividad de Visualización de Variables***

## Desarrollo del taller.

Cree y despliegue el modelo implementando la service task con el topic SendLetter.

Cree y ejecute el archivo JS con la configuracion del Worker.

Cree una instancia del proceso y visualice la operación del Worker.

# RESULTADOS

* + *Subir a la plataforma unillanos.edu-labs.co un archivo empaquetado con el modelo final obtenido en Camunda Modeler.*