



# Camunda

A Workflow and Decision automation platform

Ousmane SOW

Technical Lead Java/Devops @ Orange OWF/DSI

**AMBIENT°IT**  
FORMATION - CERTIFICATION - EXPERTISE

# Pape Ousmane SOW

Software Engineer | IT Consultant

- + 8 ans d'experience
- Java Ecosystem (Java, Spring, Quarkus)
- DevOps & Cloud (GCP/AWS, k8s, ci/cd, ...)
- E-Commerce and Telecom Industry



# Organisme de Formation

- **+500 Formations en informatique**
- **3 certification éligibles CPF**
  - DevOps
  - CyberSécurité
  - IA
- **Linux Foundation Partner**
- **Offensive Security Partner**
- **Microsoft Partner**



# Objectifs

- Modéliser et Automatiser des processus BPMN avec Camunda
- Maîtriser les concepts de base du moteur Camunda
- Manipuler le moteur via son API REST
- Manipuler le moteur en Java via le framework Spring
- Connaitre les bonnes pratiques pour un environnement de production

# Pré-requis

- 14h (2 x 7h)
- Pause 1h
- Niveau: Intermédiaire
- APIs REST
- Connaissance en Java (Spring est un plus)
- Postman, Camunda Modeler, IDE java, JDK 11 & Maven, Docker

# Faisons Connaissance

Nom, Poste, Expérience, Objectifs ...

# Agenda

# Jour 1

- Introduction à Camunda
  - Norme BPMN
  - Moteur BPMN Camunda
  - Installation Camunda Platform et Camunda Modeler
  - Les symboles de modélisations
- Modéliser et exécuter un processus métier
  - Camunda Modeler
  - Webapps : Cockpit, Task List, Admin
  - Déploiement d'un process
  - Formulaire Camunda
- API REST de Camunda
  - Fonctionnement et OpenAPI
  - Avantages et Limites
  - Administrer la plateforme
  - Manipulation de process via l'API REST

# Jour 2

- Développer une application Spring Boot basée sur Camunda
  - Quoi, Pourquoi et comment ?
  - Configuration et personnalisation
  - Les interfaces Java du moteur
  - Query API Java
  - Test Unitaire
- DMN avec Camunda
  - Norme DMN
  - Intégrer des tables de décision
- Best Practices Camunda
  - Modélisation
  - Architecture distribuée, Messaging & Multitenancy
  - Performances et Big Data

# Introduction

# Intro

## BPM/BPMN

- BPM: Business Process Management
- BPMN: Business Process Management Notation
  - Standard de modélisation de processus métier (BPM)
- Approche systématique pour gérer des processus métiers
- Améliore l'efficacité des processus métier
- Représente des étapes, flux de données, événements, activités, gateway, acteurs impliqués
- Alignement entre métier et technique

# Intro Camunda

- Plateforme open-source d'automatisation de processus métier (BPM)
- Basé sur le standard BPMN 2.0
- Intégration au SI de l'entreprise
- Améliore l'efficacité des processus métier
- Réduire coût opérationnel
- Satisfaction client

# Intro

## Pourquoi un moteur BPM

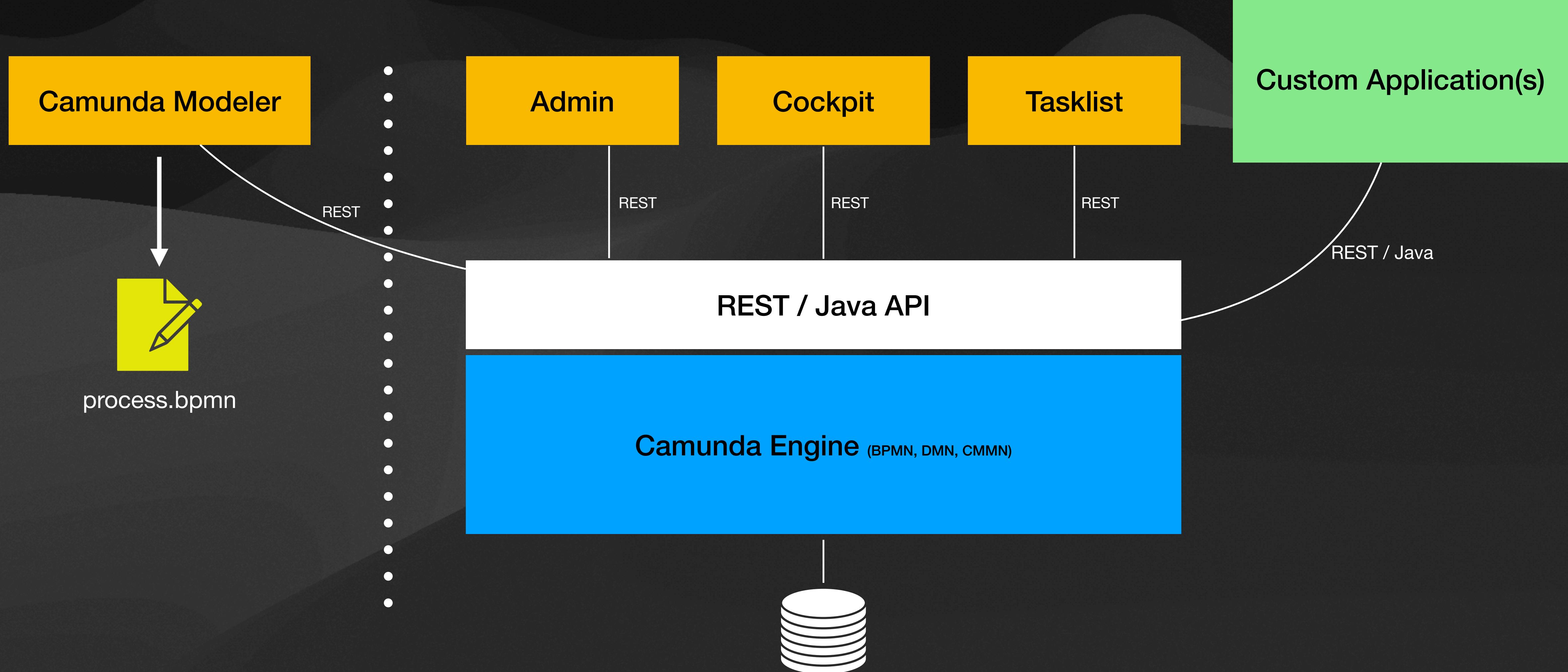
- Orchestration des systèmes et acteurs humain
- Stateful
  - Un process peut prendre du temps (des secondes, ... des années)
- Intégration facile avec SI
- Flexibilité
- Moins de code complexe
- Open source

# Intro

## Installation Modeler et Platform

- Modeler: Site Camunda
  - <https://camunda.com/download/modeler/>
- Platform - <https://docs.camunda.org/manual/latest/installation/>
  - Standalone
  - Docker
  - Embedded in spring (\*\*\*)
  - Karaf / OSGi

# Intro Architecture



# Explorons Camunda Platform Camunda Modeler

TP 1

# BPMN dans Camunda

# Symbols BPMN de Camunda

- 4 Grandes Catégories :
  1. Participants
  2. Gateways
  3. Activités
  4. Événements
  5. Commentaire

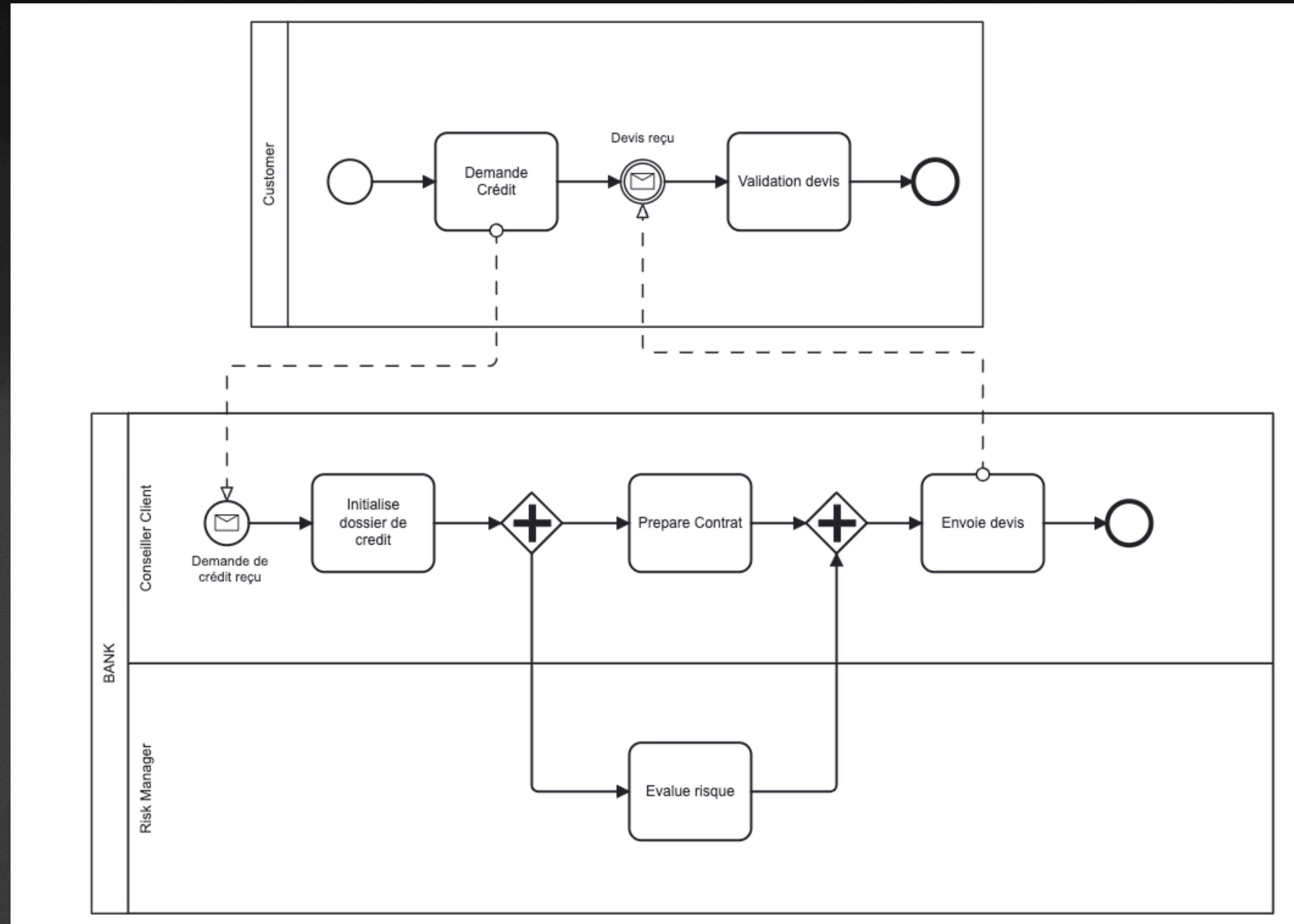
# Symbols BPMN de Camunda

## Participants (Pool & Lane)

- Un pool permet de représenter un acteur ou une organisation
  - Regroupe les activités
- Lane est une subdivision d'un pool - représente une partie du pool
  - Décrire rôle (activité) des acteurs/groupe d'acteurs au sein du pool
- Chaque pool a son début et fin de process

# Symbols BPMN de Camunda

## Participants (Pool & Lane)



# Symbols BPMN de Camunda

## Activités



Unité de travail à accomplir

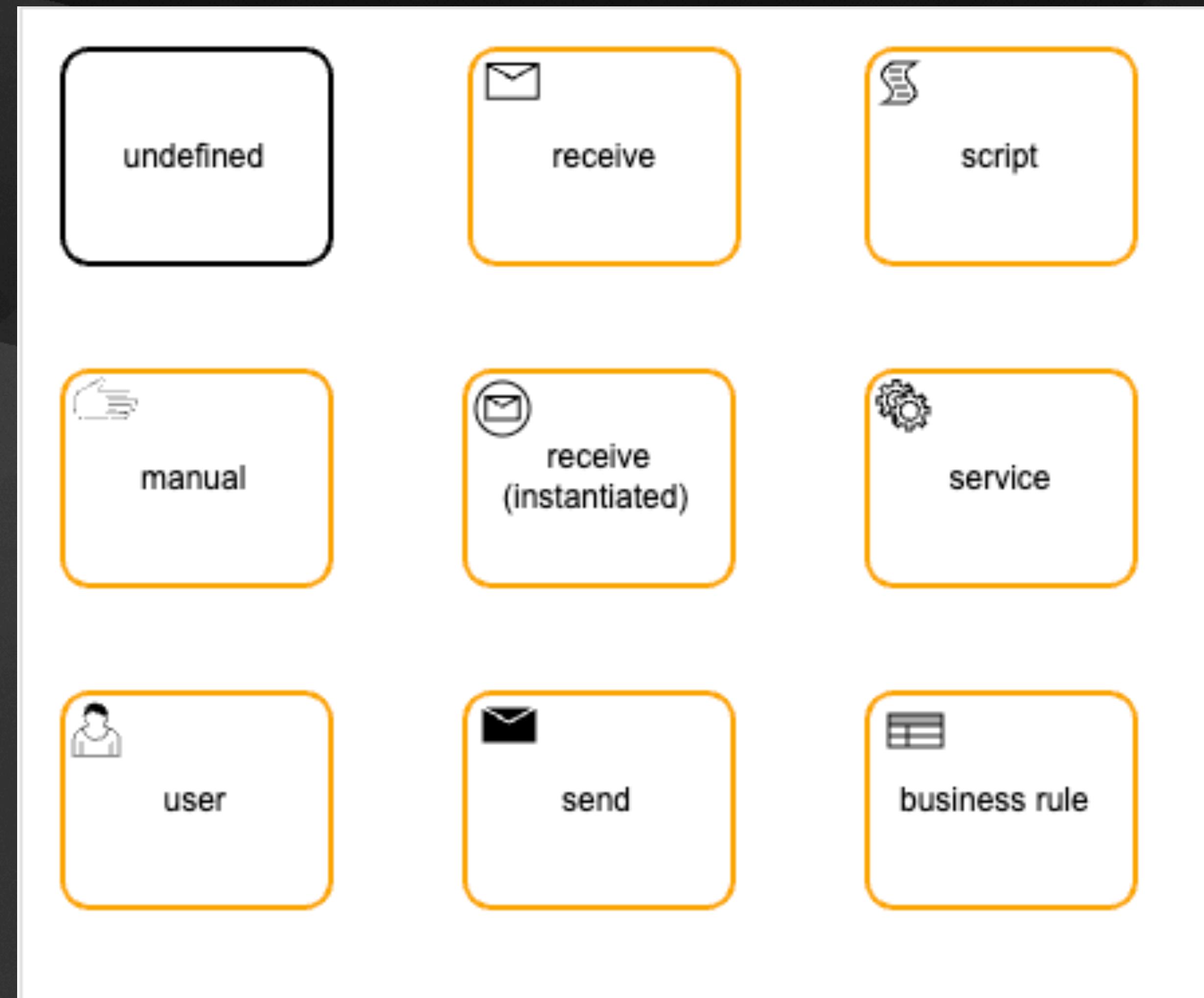
Un processus autonome incorporé dans un processus parent

Appel d'un sous process externe dans un process

Comme un subprocess mais déclencher par un event

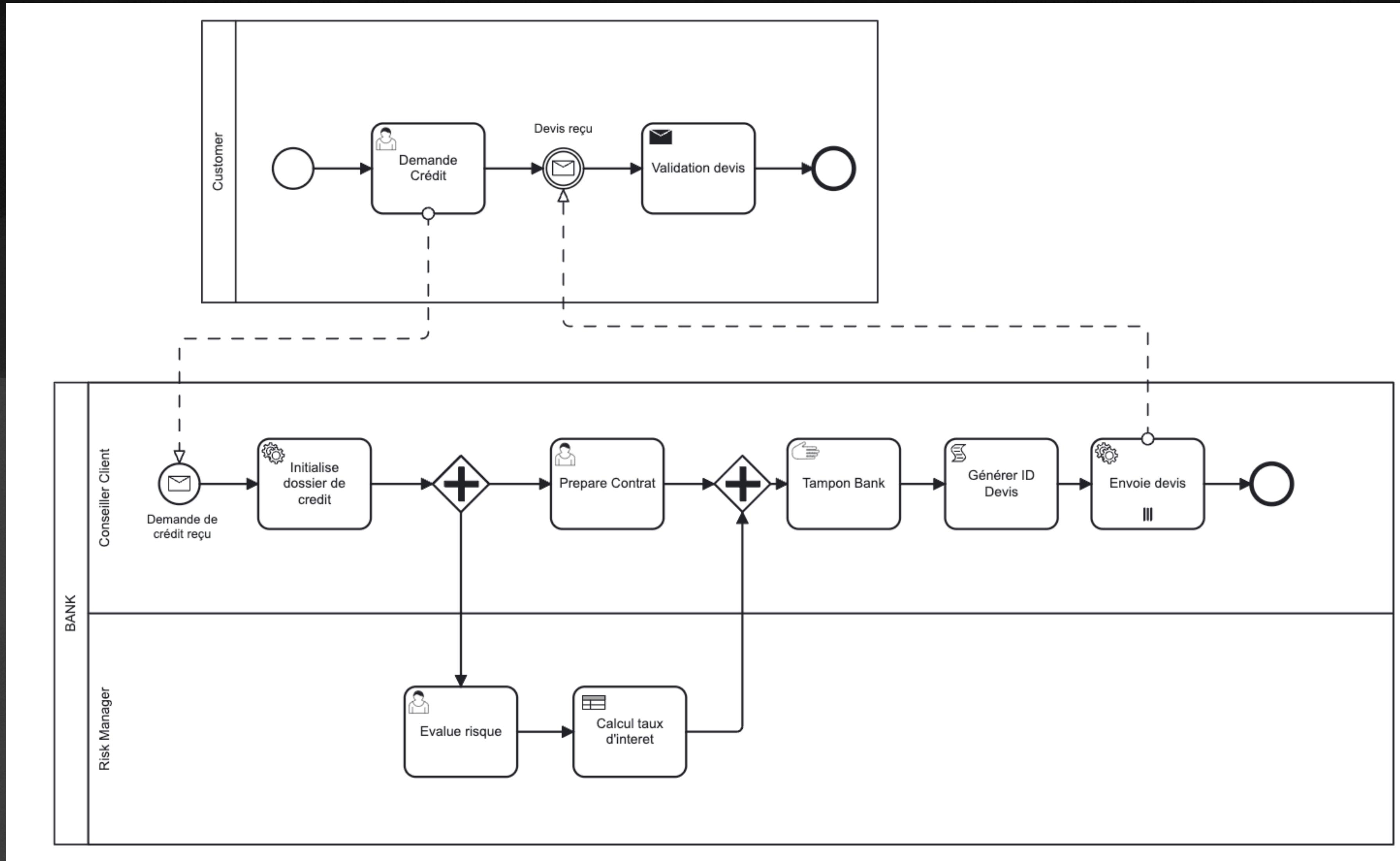
# Symbols BPMN de Camunda

## Activités - Task



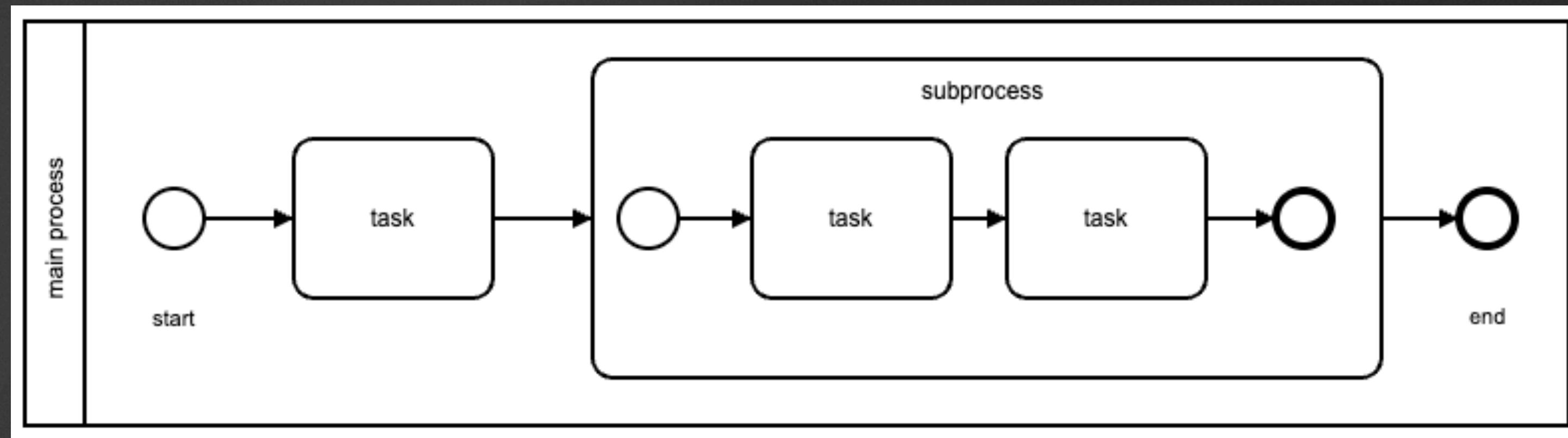
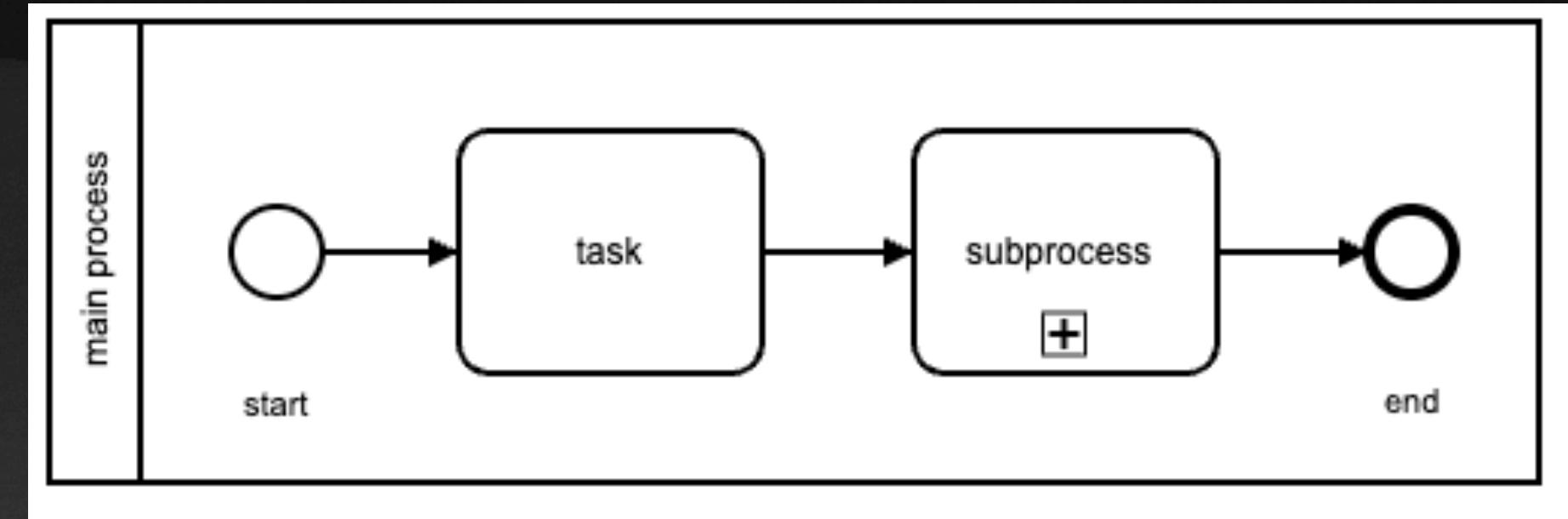
# Symbols BPMN de Camunda

## Activités - Task



# Symbols BPMN de Camunda

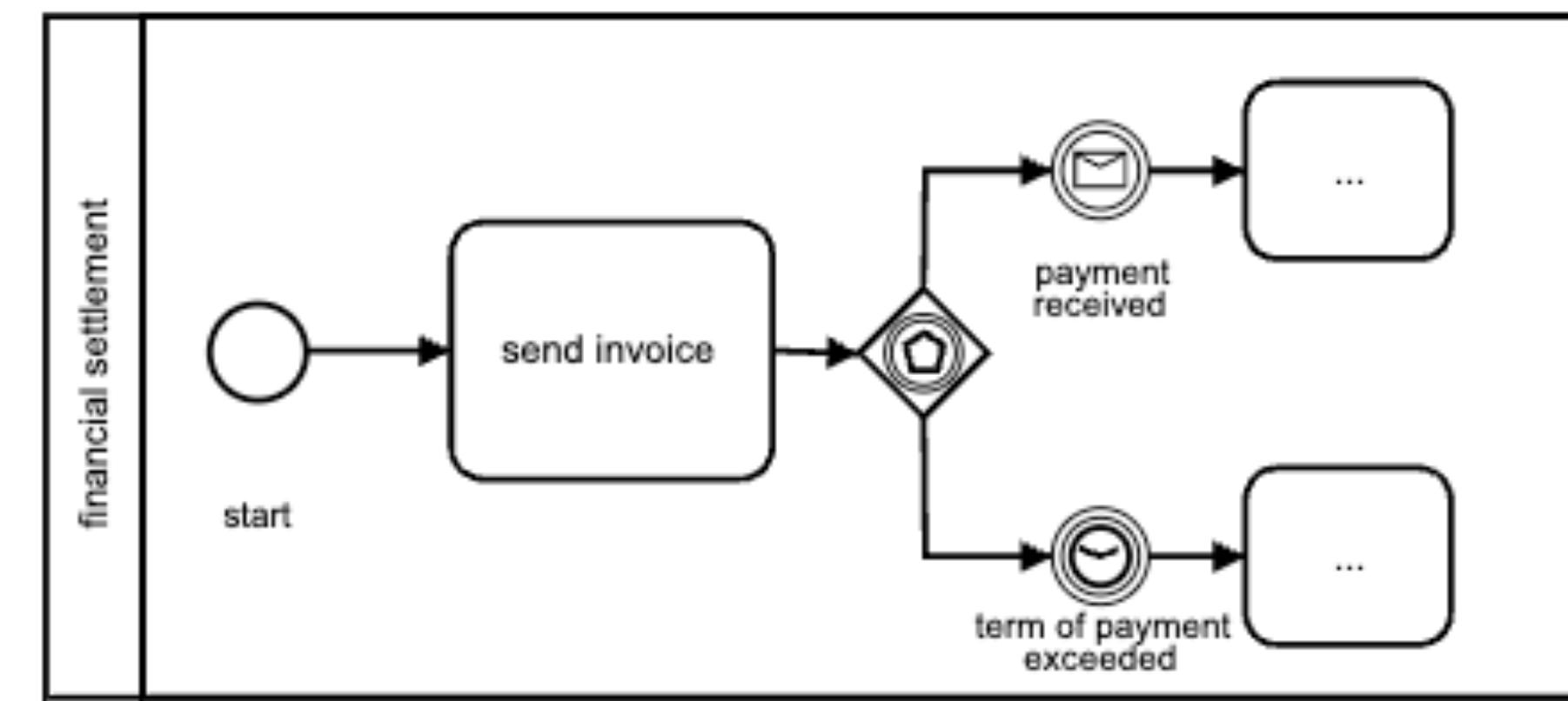
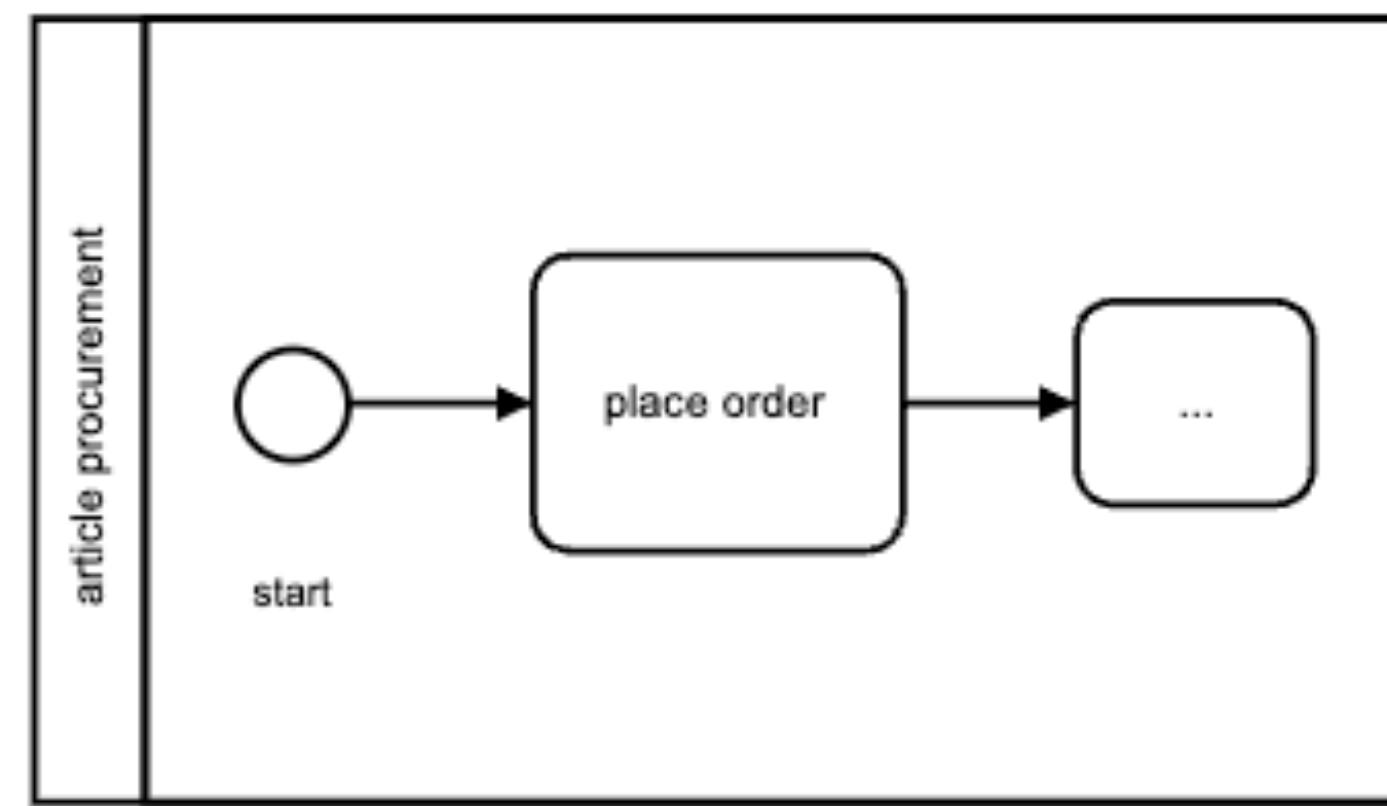
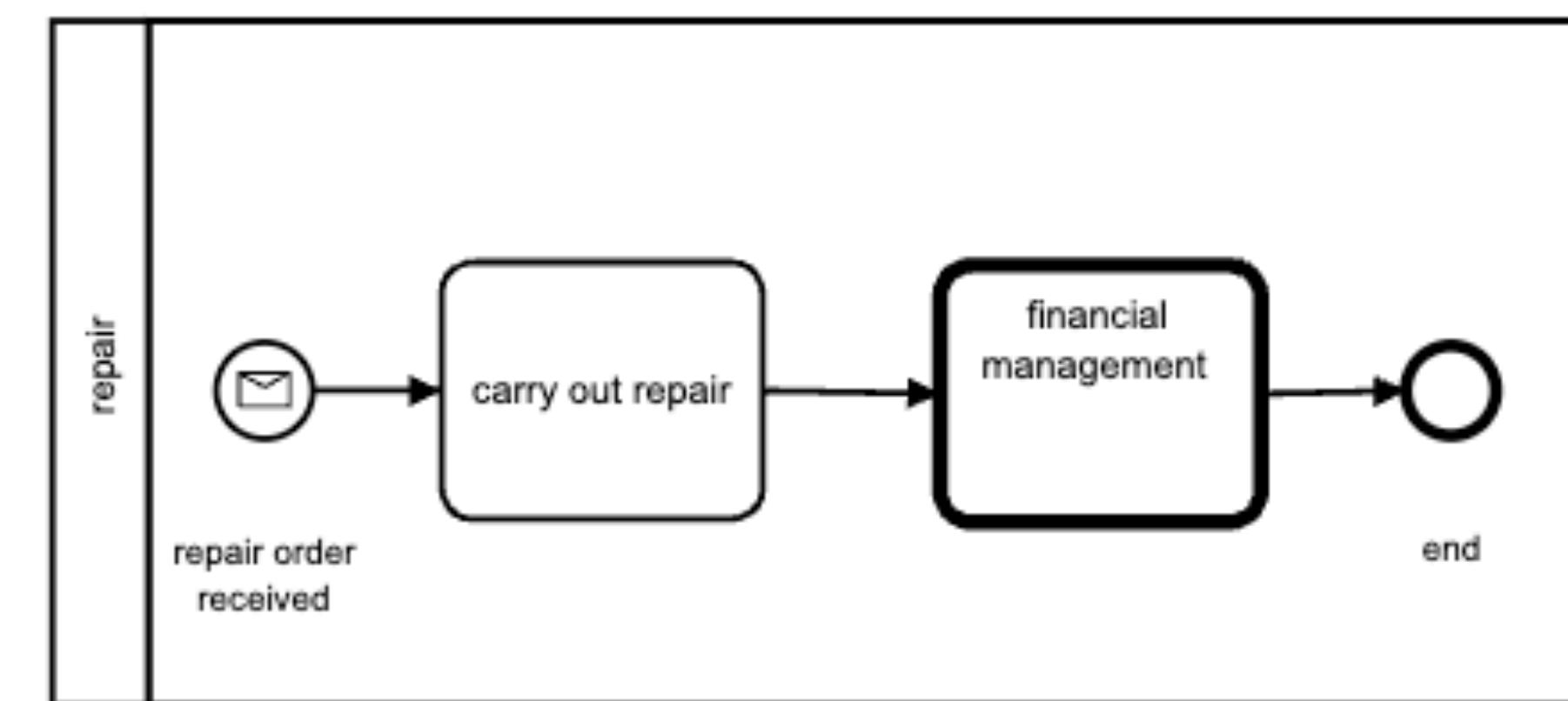
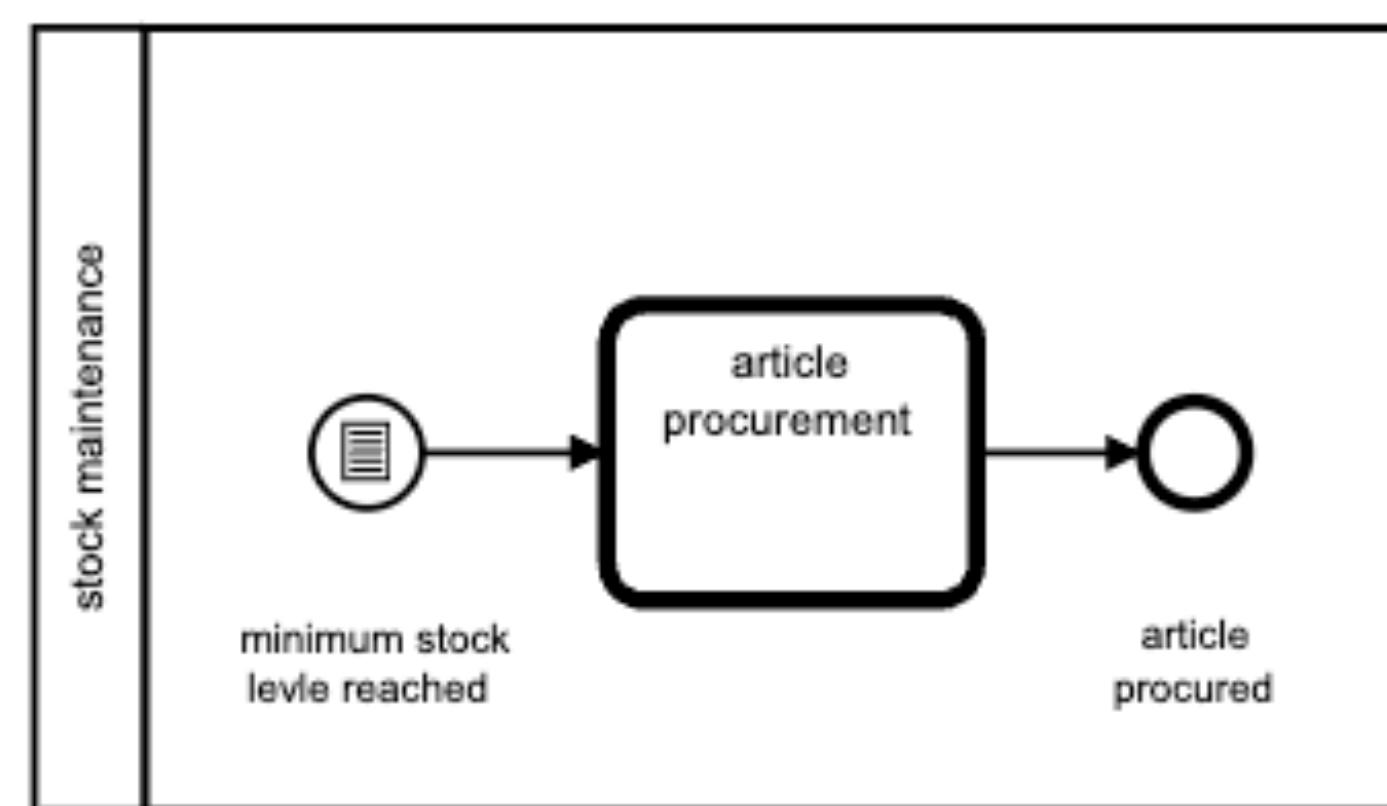
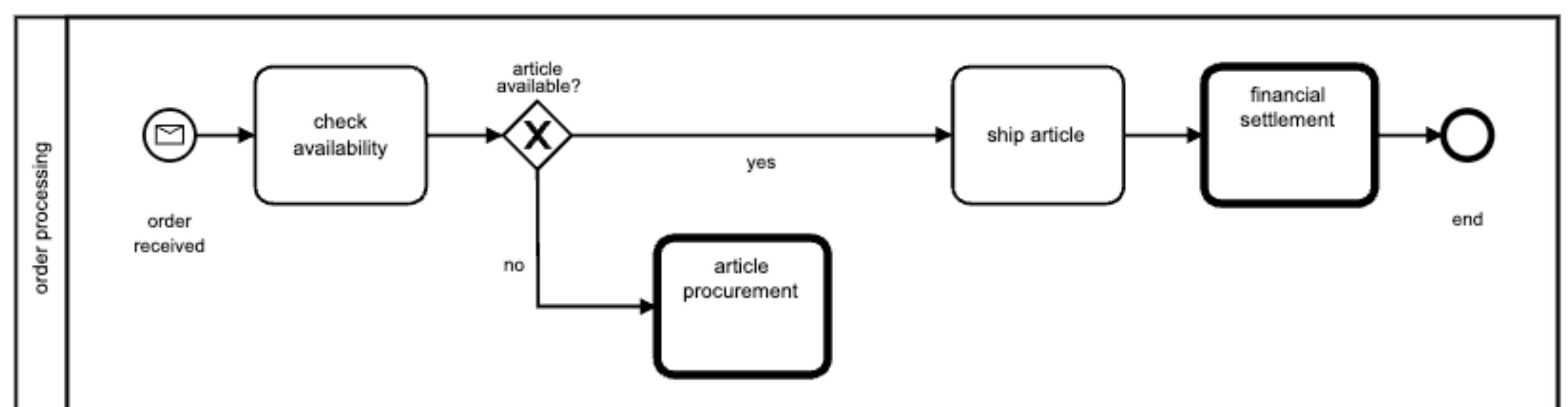
## Activités - Subprocess



# Symbols BPMN de Camunda

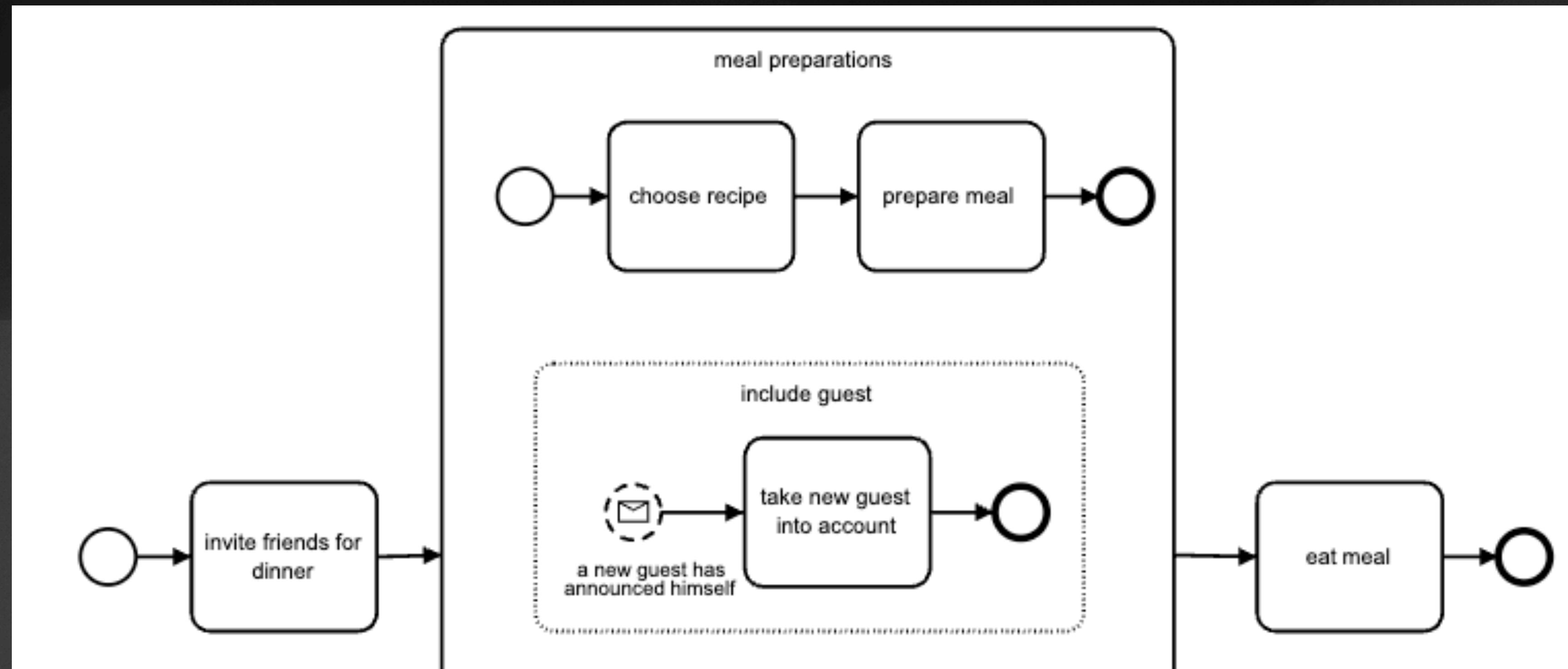
## Activités - Call Activity

- Modularité et réutilisabilité
- Pas embarqué dans le process appelant



# Symbols BPMN de Camunda

## Activités - Event Subprocess

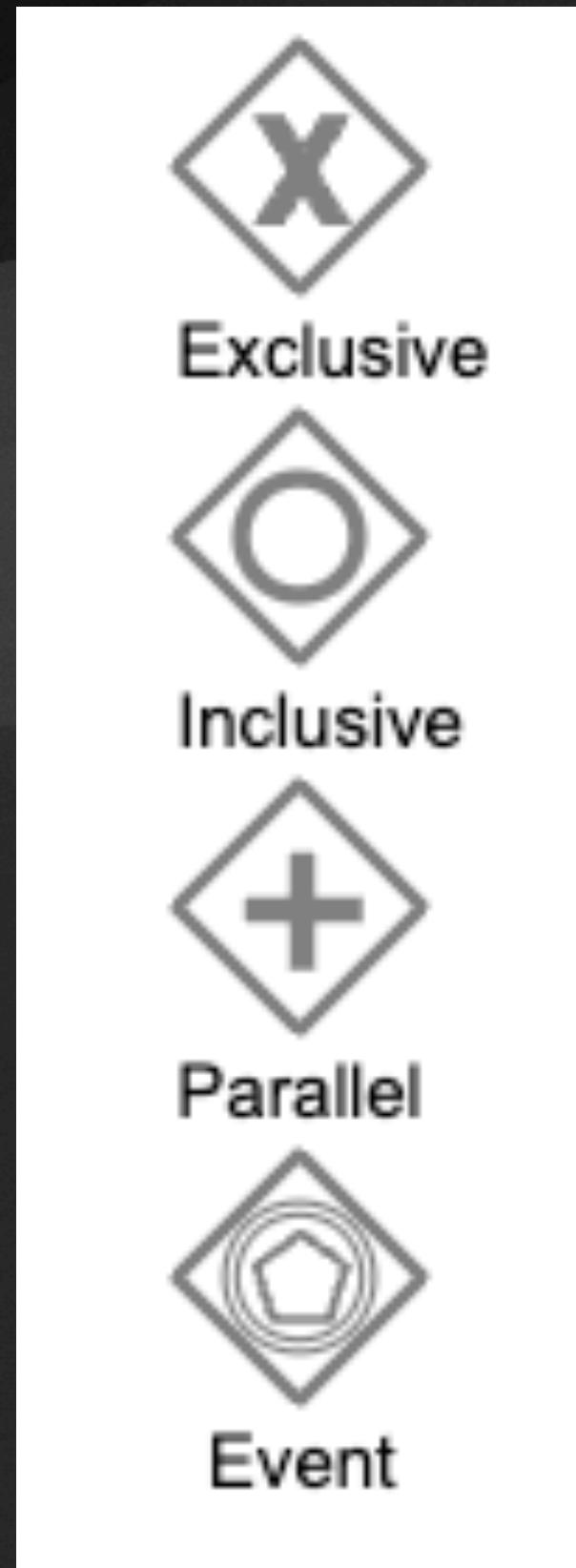


- Subprocess démarré par un event

# Symbols BPMN de Camunda

## Activités - Gateway

- Permet de contrôler la sequence du flux
- 4 Gateway :
  - Exclusive Gateway
  - Inclusive Gateway
  - Parallel Gateway
  - Event Gateway



# Symbols BPMN de Camunda

## Activités - Event

- Déclenchement d'action dans le process
  - Traitement async
  - Cas exceptionnelles
    - 3 Types d'événements :
      1. Debut
      2. Intermédiaires
      3. Fin
    - Catching Events
    - Throwing Events
    - Boundary
    - Subprocess Events

# Symbols BPMN de Camunda

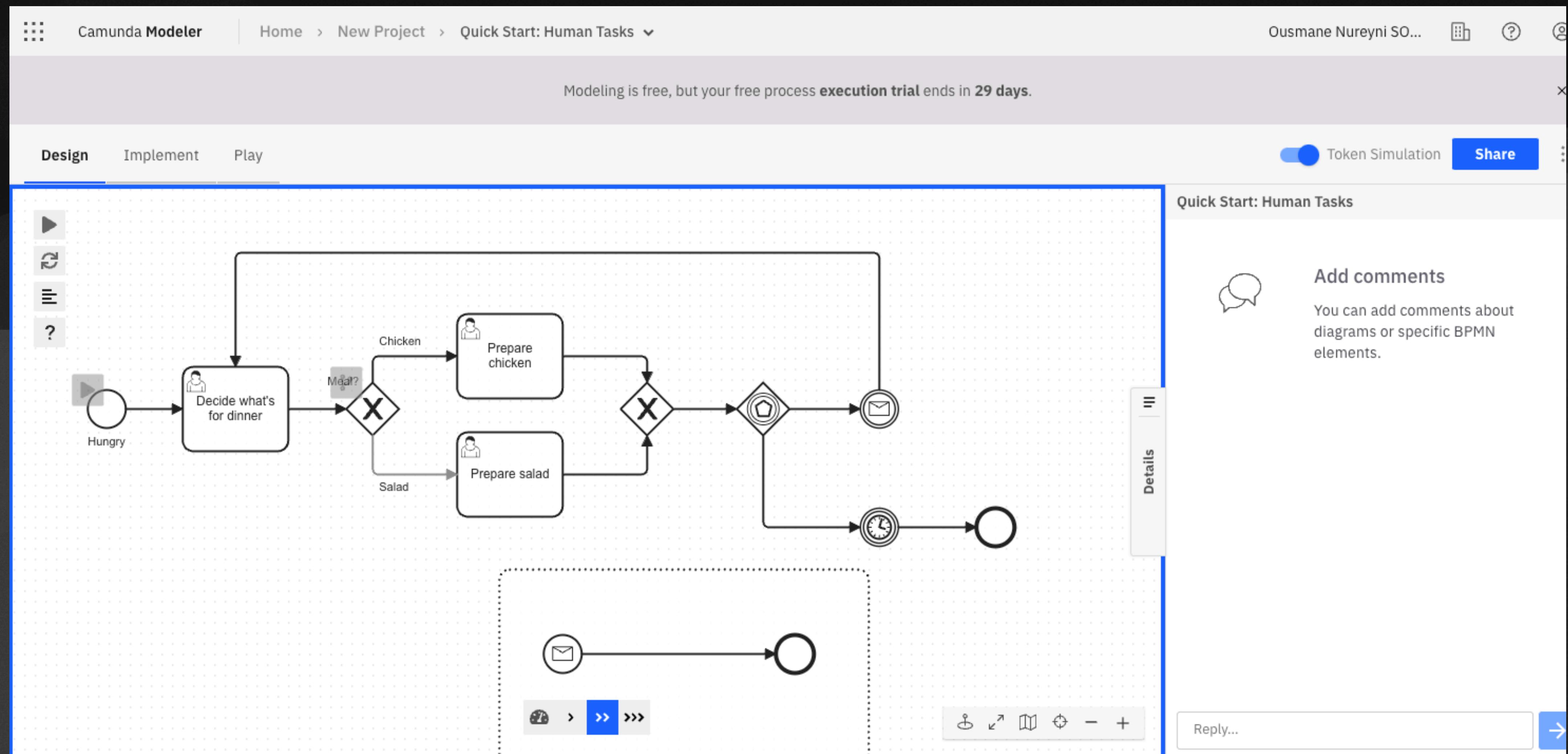
## Activités - Event

<https://camunda.com/bpmn/reference/>

# Les variables

- Données stockées dans le contexte d'un processus
- Utiliser pour des calculs, prise de décisions, contrôle de flux
- Cycle de vie
  - Creation en début de process ou en milieu de process
  - Types: String, Number, Date, Boolean, List, File, Object
  - Scope: Process, Task, Loop
  - Défini par le système ou l'utilisateur

# Fonctionnement de Camunda



# Hello World

TP 2

Faire un BPMN qui va afficher “Hello World” sur la console du moteur  
Camunda

L'instruction groovy pour afficher sur la console est **println**

# Modéliser et Manipuler le moteur

# Camunda Modeler

- Outil de modélisation graphique pour les processus métier basé sur BPMN, DMN et CMMN
- Intuitive et facile
- Génère le XML décrivant le process pour le moteur Camunda
- Permet de déployer et tester le fonctionnement de son modèle
- Possibilité de customiser avec des plugins

# Camunda WebApps

- Cockpit
  - Supervision, Contrôle état temps réel, Statistiques, ...
- Tasklist
  - Visualisation taches, Traitement, Assignation, ...
- Admin
  - Groupes, Utilisateurs, Autorisations, ...

Applications Web:  
Spring Boot, AngularJS

# Déploiement

- Cockpit
- Camunda Modeler
- API REST
- Via Java / Spring Boot

# Manipulation

- Cockpit/Tasklist
- Camunda Modeler (Lancement)
- API REST
- Via Java / Spring Boot

# Intégration des User Tasks

- Tache BPMN où un utilisateur doit :
  - Interagir avec le moteur
  - Eventuellement lui donner des informations
- Pour faire avancer le process
- Exemples:
  - Approuver/Rejet une demande
  - Completer la demande

# Intégration des User Tasks

## Les formulaire Camunda

- UI pour saisir des données et compléter un User Task
- 3 Possibilités: GeneratedForm, Camunda Form, External/Embedded Form
- Page de completion généré dans l'app TaskList
- Possibilité de rajouter de la validation

PS: Pas la seule option pour les user tasks

# Déploiement HelloWorld

TP 3

Déployer le modèle Hello World et Lancer une instance

Modifier le process pour demander le nom de la personne  
et afficher “Hi <given\_name>” ou “Hello World” par défaut

# Hello World v2

TP 4

Faire un Generated Form puis un Camunda Form pour demander le nom

# On mange où ce midi?

TP 5

Process pour choisir le lieu (cantine, resto) puis **aller manger**  
Rentrer après 2h dans tous les cas

Modéliser et déployer le process

# Camunda REST API

# Fonctionnement API REST

- Expose toutes les interfaces pour gérer et manipuler le moteur
- Suis la norme OpenAPI Specification
- Bien documentée avec des exemples
- Intégration facile avec d'autres systèmes (CRM, ERP, ...)
- Permet de créer des applications personnalisé

# Avantages

- Flexibilité pour intégration
- Standardisé
- Documentation complète
- Sécurisé
- Extensibilité

# Limites

- Complexe à architecturer
- Performance et Réactivité
- Fonctionnalités Limitées
- Gestion des erreurs complexe

# Rest API Reference

<https://docs.camunda.org/manual/7.13/reference/rest/>

<https://docs.camunda.org/rest/camunda-bpm-platform/7.20-SNAPSHOT/>

# Administre le moteur Camunda

- User
- Group
- Deployment
- Process Definition
- History
- Incidents

# Manipulation de process

- Process Definition
- Execution
- Process Instance
- Task
- Message
- Signal
- Event Subscription

On va manger  
par API REST

TP 6



Développer une application Spring Boot  
basée sur Camunda

# Quoi ?

- Spring Boot: framework Java qui simplifie et accélère le développement
- Fortement utilisé dans l'écosystème Java
- Manipulation programmatique avec Java directement sur le moteur
- Intégration de Camunda dans un projet Spring Boot
  - Autoconfiguration via les starters
  - Bean à disposition pour opérer la platform et executer ces process
  - Possibilité de modifier les bean d'autoconfiguration

# Pourquoi ?

- Simplification intégration
- Rapidité de développement
- Flexibilité et extensibilité
- Intégration logique métier plus simple
- Gestion des erreurs
- Ajout d'autres technologies (BigData, Messaging, Monitoring...)

# Comment ?

- Ajout des dépendances starters
- Moteur est embarqué dans l'application
- Configuration via fichier properties Spring Boot
- Chargement des BPMN au démarrage
- 12 Interfaces/Bean Spring pour faire tous ce que l'on veut 😊

# Configuration

- Datasource spring par default
- Possibilité de configurer un second data source via  
`@Bean(name="camundaBpmDataSource"),`  
`@Bean(name="camundaBpmTransactionManager")`
- DefaultHistoryConfiguration
- JPA
- DefaultAuthorizationConfiguration

# Configuration

- `org.camunda.bpm.spring.boot.starter.configuration.CamundaProcessEngineConfiguration`
- `org.camunda.bpm.spring.boot.starter.configuration.CamundaDatasourceConfiguration`
- `org.camunda.bpm.spring.boot.starter.configuration.CamundaHistoryConfiguration`
- `org.camunda.bpm.spring.boot.starter.configuration.CamundaHistoryLevelAutoHandlingConfiguration`
- `org.camunda.bpm.spring.boot.starter.configuration.CamundaJobConfiguration`
- `org.camunda.bpm.spring.boot.starter.configuration.CamundaDeploymentConfiguration`
- `org.camunda.bpm.spring.boot.starter.configuration.CamundaJpaConfiguration`
- `org.camunda.bpm.spring.boot.starter.configuration.CamundaAuthorizationConfiguration`
- `org.camunda.bpm.spring.boot.starter.configuration.CamundaFailedJobConfiguration`
- `org.camunda.bpm.spring.boot.starter.configuration.CamundaMetricsConfiguration`

# Configuration

camunda.bpm

- .enabled

- .history-level

- .auto-deployment-enabled

- .license-file

- .deployment-resource-pattern

# Configuration

camunda.bpm.job-execution

- .core-pool-size

camunda.bpm.database

- .schema-update

- .type

- .schema-name

- .jdbc-batch-processing

# Configuration

camunda.bpm.authorization

- .enabled

- .tenant-check-enabled

camunda.bpm.admin-user

- .id

- .password

- .firstName, .lastName

# Déployer nos process via Spring Boot

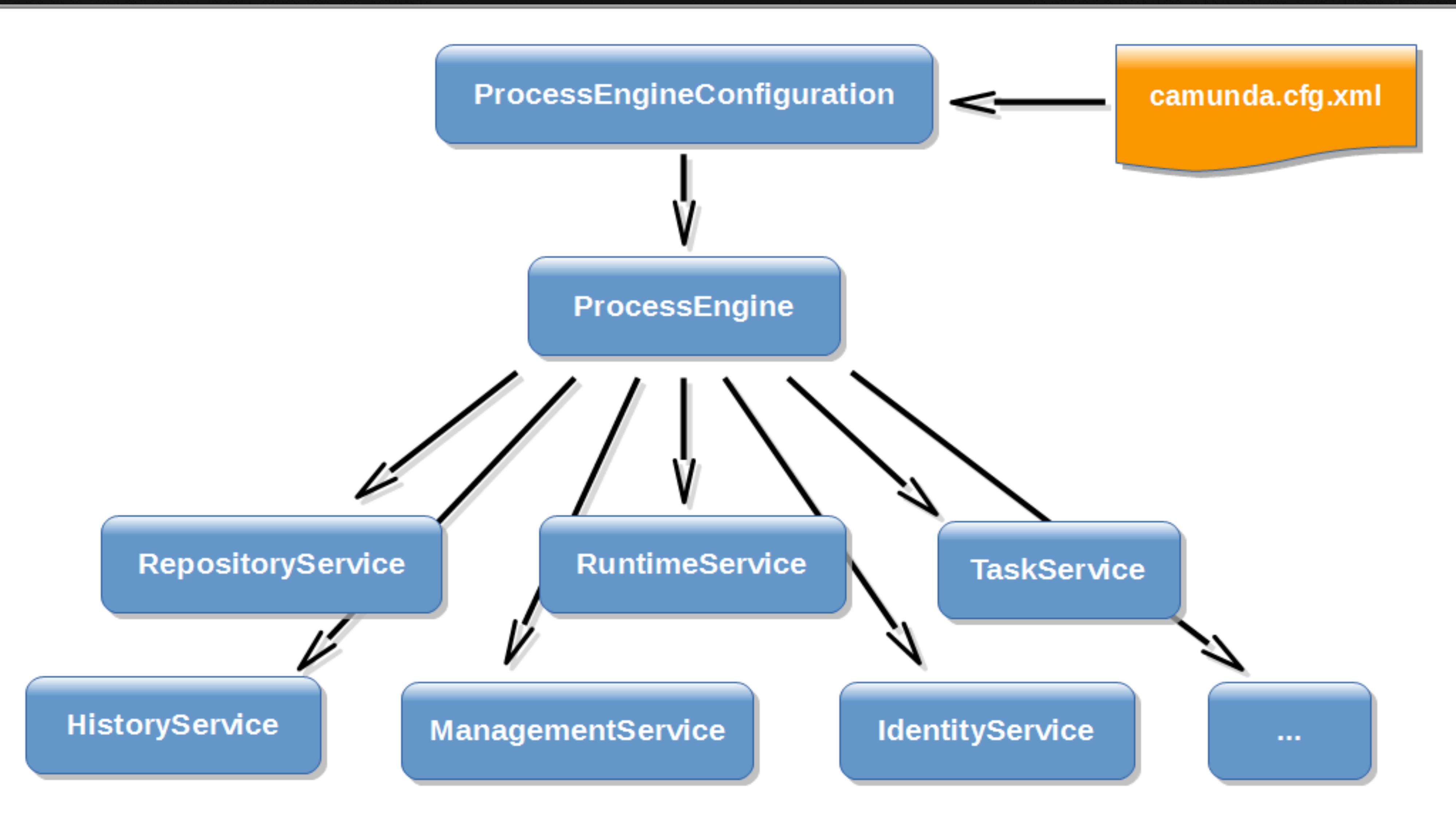
TP 6

Utilisez le Camunda starter pour bootstrapper une application embarquant Camunda  
Recupérer les fichier de la branche java\_part sur le github

# Bean Spring Camunda

- ProcessEngine
- RepositoryService
- RuntimeService
- TaskService
- IdentityService
- FormService
- HistoryService
- ManagementService
- FilterService
- ExternalTaskService
- CaseService
- DecisionService

# Bean Spring Camunda



# Query API Camunda

- Sur la plus part des interfaces de composants
- API pour interroger les données en BDD
- Programmatique: Requête préparé et prédéfinis
- Facile à utiliser

# Query API Camunda

```
List<Task> tasks = taskService.createTaskQuery()
    .taskAssignee("kermit")
    .processVariableValueEquals("orderId", "0815")
    .orderByDueDate().asc()
    .listPage(20, 50);
```

# Test Unitaire

- Camunda Platform Assert
- Bibliothèques d'assertions
- Facile à utiliser
- Assertions prédéfinies
- Assertions personnalisées
- Intégration avec JUnit

# Aller au resto via springboot

TP 7

Completer le code pour faire fonctionner le controller PauseDejController

# DMN avec Camunda

# DMN

- Standard OMG pour modéliser des tables de décision
- Facilement compréhensible par les acteurs du SI
- Formalisation de règle complexe d'un processus de décision
- Automatisation et intégration dans Camunda
- Efficace et Précis

# Fonctionnement des DMN

- Entité déployable du moteur
- Consomme des variables en entrée
- Execute la table de decision
  - Politique de decision
- Resultats dans une ou plusieurs variable

# Les hit policy

- First
- Any
- Unique
- Priority (not yet supported in Camunda v8.2)
- Rule Order
- Collect & Collect with aggregator

# Fonctionnement des DMN

Decision 1		Hit Policy: Unique					
	When Input "standard","premnium"	And Solde Moyen double	And Mauvais Payeur boolean	Then taux integer	And message string		Annotations
1	"standard"	>= 100	-	5			
2	"premnium"	>= 1000	-	1	<span>Input Values</span>		
3	"standard"	>= 1000	-	2			
4	"premnium"	< 100	-	4	<span>edit</span>		
5	-	-	true	-1	"Pas droit au credit, Engagements non respectés"		
+/-	-	-	-	-			

# Ajout Decision Table pour calcul du taux d'intérêt

TP 8

# Les Best Practices

# Best Practice

## Modélisation

- BPMN simple et consis
- Modèles réutilisable avec séparation des responsabilités : call activity
- Encapsuler les parties complexe dans des sous process
- Eviter les boucles infinis
- Gérer les traitements async via les events (signal, message, ...)
- Modéliser les tache utilisateur en utilisant des events
- Ne pas mettre les logiques metiers sur Camunda

# Best Practice

## Architecture

- Dans le cas d'une application complexe mettre en place une surcouche
- Utiliser des systèmes de replica du moteurs et de la BDD
- Microservices et communication via un MoM du SI
- Reserver une BDD dédié à Camunda et faire des backups/restore
- Remonter des metrics exploitable par des outils de monitoring
- Multiclient : Multitenancy
- Utiliser Camunda embarqué dans une app spring

# Best Practice

## Performance et Big Data

- Politique purge process inactif
- Ne pas abuser sur des Query API complexe
- Big Data: Travailler sur les tables de History pour éviter des deadlocks
- Surveiller les performance du moteur et de la BDD (JMX, Grafana, ...)

# Evaluation

# Des Questions

# Merci!