h e g

Haute école de gestion
Genève

Test de l'automatisation d'un processus sur une plateforme BPMN

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

Par :

François Nobile

Conseiller au travail de Bachelor :

Patrick JOSET, professeur HES

Genève, le 10 octobre 2021

Haute École de Gestion de Genève (HEG-GE)

Filière IG

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre « Bachelor of Science HES-SO en informatique de gestion ».

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : https://www.urkund.com .

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Corsier, le 4 septembre 2021

François Nobile

Remerciements

Je remercie Monsieur Joset pour son idée de projet, son accompagnement et ses retours concernant mon travail de Bachelor.

Résumé

Ce projet a pour but d'essayer d'automatiser un processus en utilisant une plateforme BPMN. Dans le cas de ce travail, le processus à automatiser est un « service-desk ».

Dans ce document, j'explique le BPM et le BPMN. Je présente aussi la plateforme BPMN que j'utilise et le processus que je vais automatiser.

J'explique également pas à pas comment je m'y suis pris pour réaliser ce travail et les difficultés que j'ai rencontré pendant le développement de mon travail.

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

Depuis que la programmation existe, un rêve habite le monde de beaucoup d'informaticiens. Pouvoir développer des applications sans avoir besoin de les programmer. Bien sûr, il faudrait toujours leur dire ce qu'elles doivent faire, mais sans utiliser de code. Tout le monde pourrait ainsi développer des applications qui répondent à leurs besoins sans avoir besoin de grandes connaissances en la matière.

Cela ne veut pas dire que les développeurs disparaîtraient, puisqu'il faudrait développer les logiciels qui permettent de construire des applications.

De nos jours, ce type de logiciels existe et ils portent un nom. Ce sont les logiciels *low-code* ou *no-code*, « peu de code » et « pas de code » en français.

Les logiciels *low-code* et *no-code* sont faciles à prendre en main et simples d'utilisation. Leurs interfaces permettent à l'utilisateur de définir de manières visuelle et interactive l'affichage, l'organisation des données et leurs traitements. Évidemment, du code est généré quelque part, mais il ne sera pas montré à l'utilisateur. D'ailleurs, presque tout le monde a déjà utilisé au moins un logiciel *low-code* puisque le tableur Excel en est un[1].

Les applications *low-code* permettent aussi de briser la barrière entre les informaticiens et les autres employés dans une entreprise. Chacun peut numériser ses activités en ayant simplement suivi une formation à l'utilisation du logiciel.

C'est pareil pour les logiciels *no-code* puisqu'il est possible de développer des applications mobiles sans code[2].

Dans le cadre de mon travail, c'est une plateforme BPMN, *low-code*, que je vais utiliser pour essayer de modéliser un processus métier et en utilisant peu, voire pas du tout de programmation.

[1] https://blog.octo.com/le-low-code-comment-ca-marche/
[2] https://fr.goodbarber.com/create/apps/?gclid=CjwKCAjwtfqKBhBoEiwAZuesiB9n0xDOVEZE6odgsOyGeT0wBRbVnSRsIH-tVfk4SzuMbxoq1ZQCXhoCqeAQAvD_BwE

2. Le Business Process Management

Le *Business Process Management* (BPM) est une activité organisationnelle qui permet d'analyser et répertorier continuellement tous les processus métiers de l'entreprise afin de les automatiser et les optimiser pour les rendre les plus efficients et efficaces possibles[3].

Le plus souvent, ces processus sont analysés puis modélisés informatiquement en utilisant la méthode de modélisation *Business Process Model and Notation* (BPMN).

Un processus métier est caractérisé par un évènement déclencheur en entrée, une suite d'activités constituant la chaîne des valeurs ajoutées et une fin qui se matérialise par un résultat pour le bénéficiaire du processus.

Le business process management est important, car des processus non supervisés ou désordonnés peuvent amener à des erreurs plus fréquentes et de la perte de temps et cela se répercute ensuite sur les employés, qui seront démotivés et travailleront moins bien[4].

2.1 Les étapes du BPM[5]

2.1.1 Diagnostic de l'organisation

Il faut commencer par choisir quels processus modélisés, en alignement avec la stratégie de l'entreprise.

2.1.2 Analyse

L'analyse d'un processus passe par l'observation directe des différentes opérations qui forment ce processus, étudier la documentation s'il y en a et poser des questions aux acteurs concernés.

Cela permet de voir les différentes étapes du processus, ce qui y entre et ce qui en sort, les intervenants, qu'ils soient internes ou externes à l'entreprise, humains ou machines, les dysfonctionnements qui causent des erreurs et des retards et les risques que ce processus encoure. Ainsi, cela permettra aussi de proposer des améliorations.

[3] https://fr.wikipedia.org/wiki/Business_Process_Management
[4] https://kissflow.com/workflow/bpm/business-process-management-overview/
[5] https://www.manager-go.com/organisation-entreprise/bpm.htm

2.1.3 Conception

Cette étape est la conception ou la reconception du processus. C'est à cette étape que le processus va être modélisé.

Pour ce faire, le BPMN fournit tous les éléments graphiques de modélisation requis. Les entreprises utilisent généralement des logiciels BPM *low-code* développés spécifiquement pour ça.

2.1.4 Exécution

Le logiciel BPM va exécuter le processus qui fonctionnera en permanence.

2.1.5 Surveillance

L'utilisation d'outils de surveillances est importante pour s'assurer que les processus fonctionnent correctement et en accord avec les objectifs définis par l'entreprise.

Cela permet aussi d'anticiper les dysfonctionnements, la non-performance et voir ce qui pourrait être amélioré par la suite.

2.1.6 Optimisation

Il est rare que le processus soit parfait du premier coup. C'est dans cette étape, grâce aux données des outils de surveillance, que le processus pourra être optimisé.

3. Le Business Process Model and Notation

Le BPMN est une méthode graphique de modélisation de processus métiers pour représenter les chaînes d'activités métier d'une entreprise.

Cette norme a été officiellement publiée par l'Organisation Internationale de Normalisation en 2013. Elle constitue la norme ISO/IEC 19510[6].

Aujourd'hui, nous utilisons le BPMN 2.0.2 qui est la dernière version développée par l'Object Management Group (OMG).

3.1 L'objectif de BPMN

L'objectif de BPMN est de fournir un langage simple d'utilisation et commun pour la modélisation des processus métiers et de proposer un ensemble de modèles pour favoriser le passage de la modélisation ou conception des processus, vers l'implantation et l'exécution des processus en langage Business Process Exécution Language (BPEL).

3.2 Comment ça fonctionne

3.2.1 Les objets d'étapes

3.2.1.1 Activité

Les activités sont une action ou un ensemble d'actions.

3.2.1.1.1 La tâche

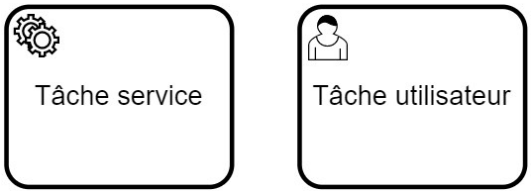
Une tâche est une activité à accomplir. Elle est représentée par un rectangle avec un symbole qui précise le type d'activité.

Ici, seules la « tâche service » et la « tâche utilisateur » nous intéressent.

La « tâche service » sert à invoquer un service. Sur Camunda, c'est fait en utilisant un *worker* codé en Java qui s'occupera de cette tâche.

La « tâche utilisateur » veut dire qu'un travail doit être fait par un acteur humain.

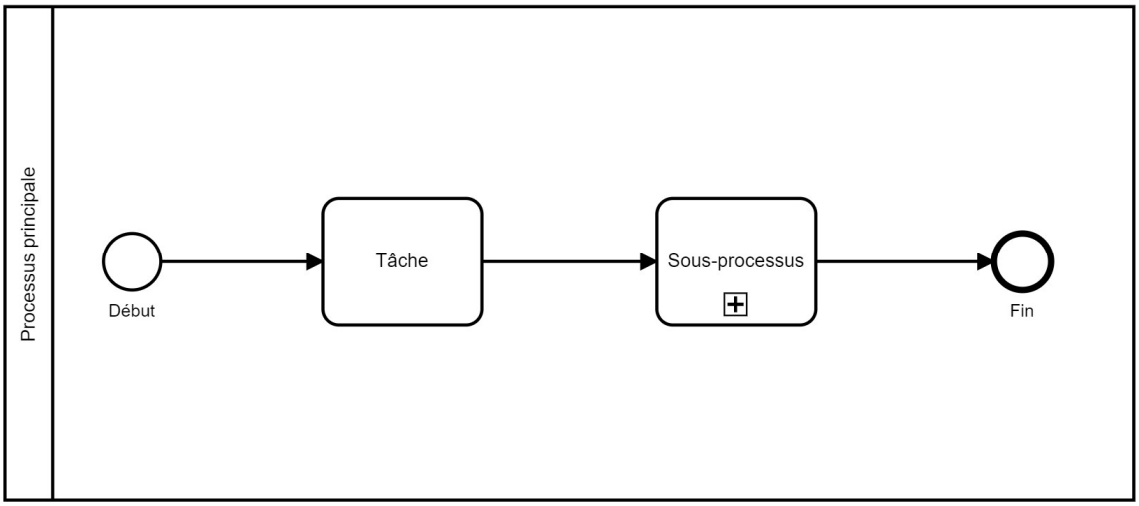
Figure 1 : Tache service et utilisateur



[6]https://fr.wikipedia.org/wiki/Business_process_model_and_notation

(Camunda Cloud)

Figure 2 : Exemple de processus pour l'objet activité

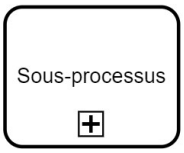


(Camunda Cloud)

3.2.1.1.2 Le sous-processus

Le sous-processus est une activité qui contient, comme son nom l'indique, un sous-processus. C'est l'appel d'un processus dans un autre. Il est représenté par un rectangle avec un petit « + » encadré.

Figure 3 : Sous-processus



(Camunda Cloud)

3.2.1.2 Évènement

Un évènement va déclencher, interrompre ou influencer le déroulement du processus.

Figure 4 : Évènements

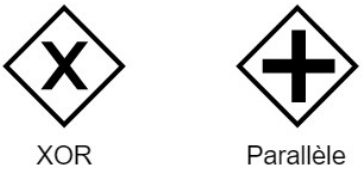
(Camunda Cloud)

Dans le cadre de mon travail, je me suis servi de l'évènement intermédiaire de type timer pour faire que, si une certaine durée est dépassée, le flux continu.

3.2.1.3 Passerelle

Une passerelle influence la séquence d'activités.

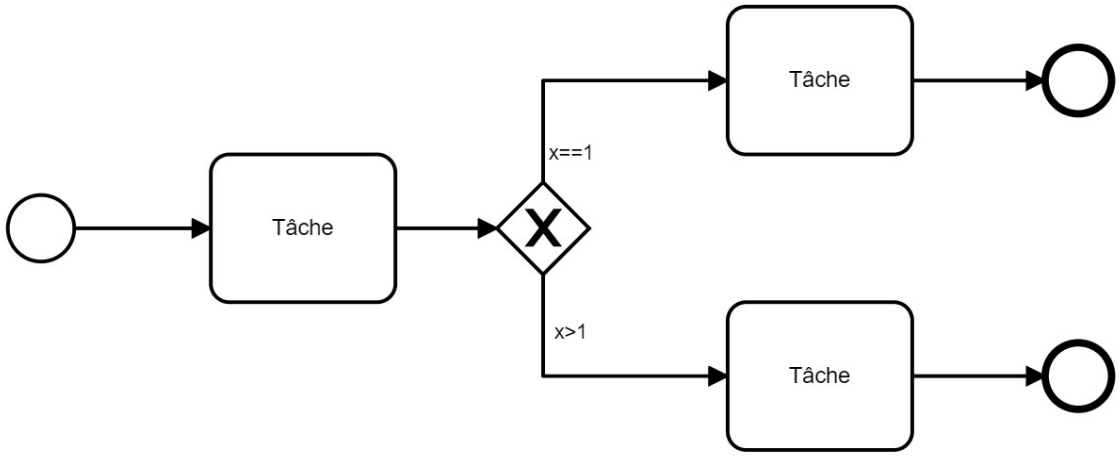
Figure 5 : Passerelles



(Camunda Cloud)

La passerelle XOR (ou exclusif) sert à modéliser une décision.

Figure 6 : Exemple passerelle XOR



(Camunda Cloud)

3.2.2 Les objets de connexion

3.2.2.1 Les flux de séquence

Les flux de séquences servent à connecter les activités entre elles et indiquent l'ordre d'exécution. ⟶

3.2.2.2 Les flux de message

Les flux de message représentent les échanges entre différents processus. Ils indiquent aussi les messages avec les acteurs externes au processus. ⇢

3.2.3 Les objets de collaboration

3.2.3.1 Les piscines et couloirs

Les piscines et couloirs définissent les responsabilités des différents acteurs au sein d'un processus métier.

Figure 7 : Piscine et couloirs

(Camunda Cloud)

4. Camunda

Figure 8 : Logo de Camunda

(Camunda, camunda.com)

Camunda est une plateforme BPMN open source d'automatisation de processus.

Il existe une version gratuite téléchargeable et installable sur son ordinateur personnel qui se nomme « Camunda Modeler », et une version payante en ligne très récente qui se nomme « Camunda Cloud ». La version téléchargeable est plus complète que la version en ligne, mais il faut tout configurer soi-même. La version en ligne est souvent mise à jour.

Pour ce travail, j'ai d'abord utilisé « Camunda Modeler », puis j'ai utilisé la version en ligne « Camunda Cloud » qui dispose d'un essai gratuit de 30 jours.

Je me sers de Camunda, car c'est le logiciel que j'ai appris à utiliser à l'école.

5. Service-Desk

Le Service-Desk est le point de contact unique entre l'utilisateur et le service informatique.

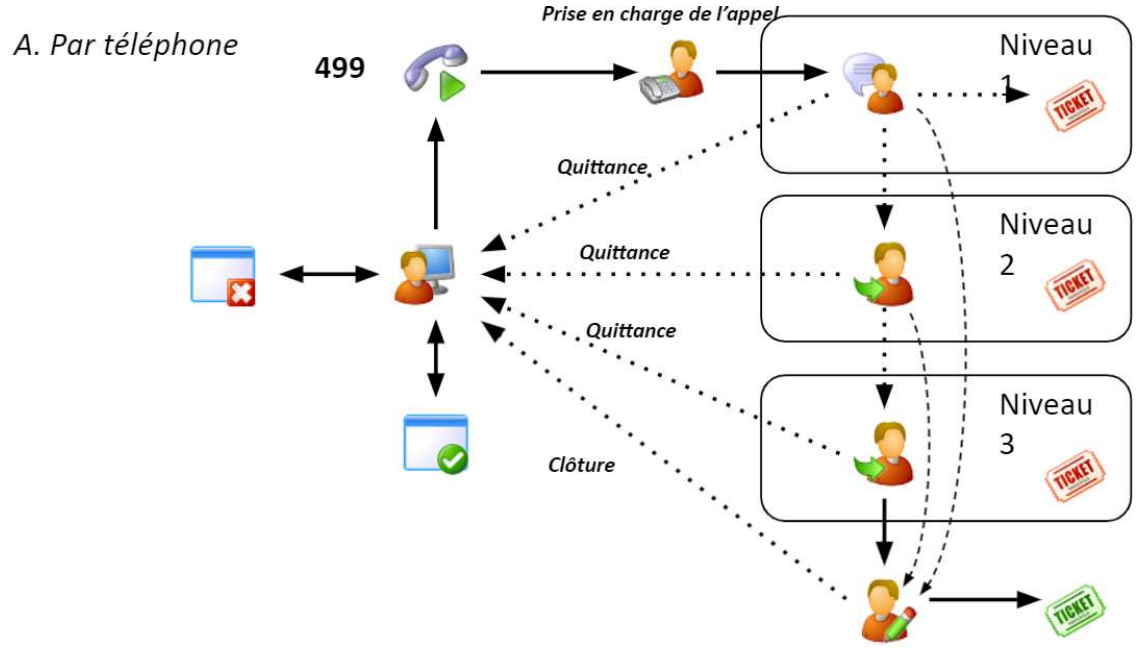
Il s'occupe :

- Des incidents (helpdesk)
- Des demandes de nouveaux services (nouvelle fonctionnalité dans un logiciel)
- Des configurations informatiques
- Des demandes de changements d'environnement
- Des questions informatiques (comment fonctionne cela)

Étant donné que le service-desk est le point de contact unique entre les utilisateurs et le service informatique, c'est lui qui renvoie l'image de la qualité de l'informatique dans l'entreprise. Il est nécessaire que les utilisateurs aient une bonne première impression[7].

5.1 Déroulement

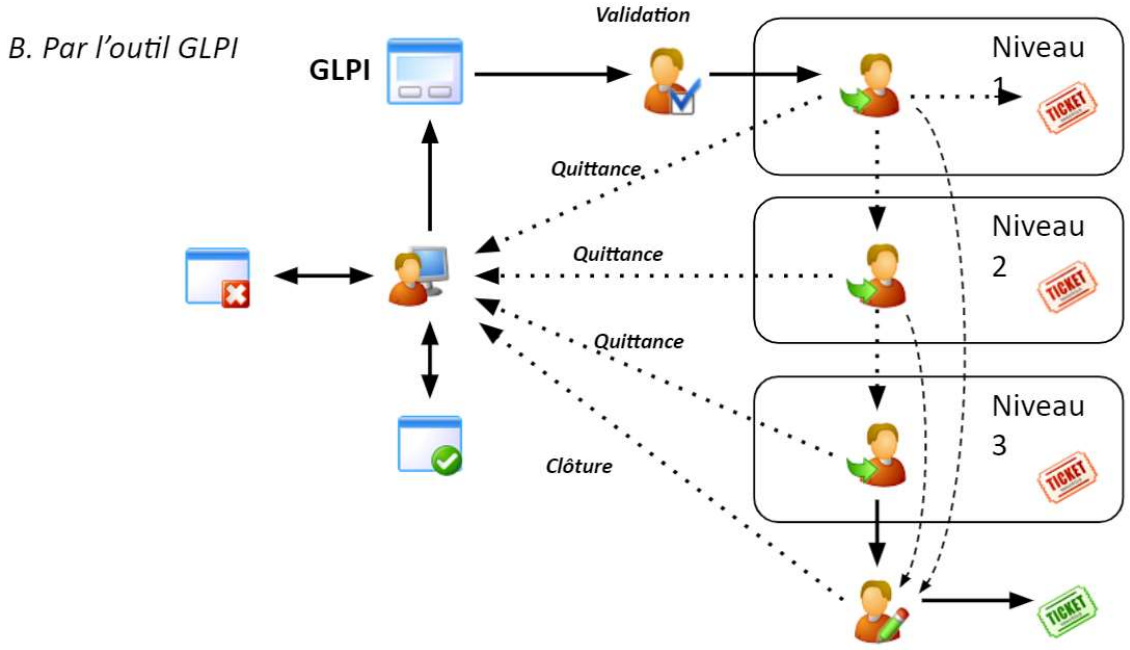
Figure 9 : Procédure d'un Service-Desk par téléphone



(Patrick Joset, Service-Desk.pptx slide 7)

[7] Patrick Joset, Service-Desk.pptx

Figure 10 : Procédure d'un Service-Desk par outil GLPI[8]



(Patrick Joset, Service-Desk.pptx slide 8)

Si l'utilisateur contacte le service-desk par téléphone, une personne prendra l'appel en charge et va créer un ticket avec les coordonnées de l'appelant et la raison de son appel.

Si l'utilisateur fait une demande par GLPI, c'est lui qui entre ses coordonnées et la raison de sa demande.

Que le contacte soit fait par téléphone ou par GLPI, la suite est la même.

5.1.1 Niveau 1

Au niveau 1, en principe, le traitement se fait à distance, par prise de contrôle du poste de l'appelant s'il le faut et pendant une durée limitée et définie qui est de 15 minutes. Si l'incident a pu être résolu, le ticket est clôturé, sinon, il passe au niveau 2. Si le traitement de l'incident prend plus de temps, le ticket passe automatiquement au niveau 2.

5.1.2 Niveau 2

Au niveau 2, le traitement se fait généralement directement sur place, à la place de travail de l'appelant. Ce niveau a aussi une durée limitée qui est de 1 heure. Comme le

[8] Gestionnaire Libre de Parc Informatique est un logiciel libre de gestion des services informatiques et de gestion des services d'assistance (Service-Desk). *Wikipédia*

niveau 1, si l'incident a pu être résolu, le ticket est clôturé, sinon, il passe au niveau 2. Si le traitement de l'incident prend plus de temps, le ticket passe automatiquement au niveau 3.

5.1.3 Niveau 3

Au niveau 3, le traitement se fait par accès à de l'expertise externe. L'utilisation d'une base de connaissances, de forums spécialisés ou du site web du fournisseur est recommandée. L'appel à un spécialiste externe est envisageable. La durée du traitement est généralement longue, de plusieurs jours voire semaine, et ne doit pas être limitée dans le temps.

À la fin, le ticket est clôturé.

Tableau 1 : Résumé des niveaux d'intervention

Niveau	**Lieu**	**Durée**	**Informations complémentaires**
1	À distance	15 minutes (limitée)	
2	Sur place	1 heure (limitée)	
3	Sur place	1 jour à plusieurs semaines (pas limitée dans le temps)	Utilisation d'expertise externe (base de connaissances, forum spécialisé, site web des fournisseurs, spécialiste externe…)

(Patrick Joset, Service-Desk.pptx slide 9)

6. Le travail

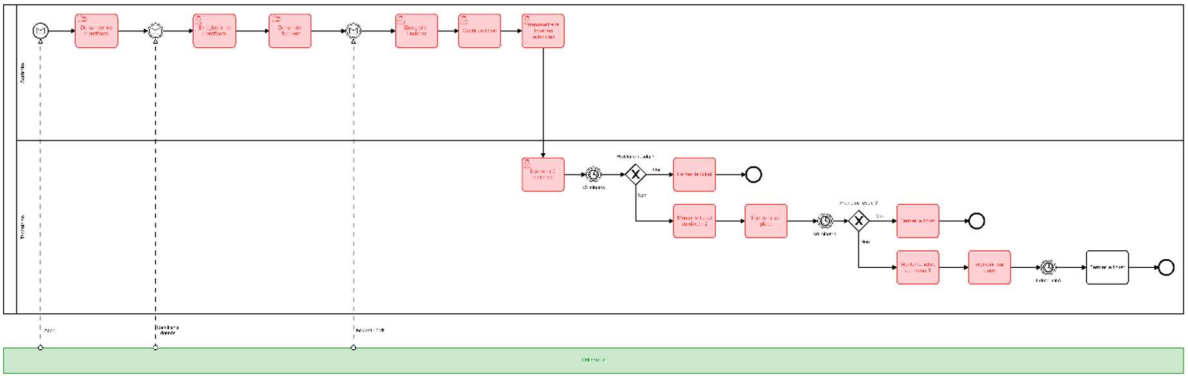
Pour ce travail, j'ai commencé par me renseigner sur le Service-Desk pour savoir ce que c'est et comment cela fonctionne, les différentes étapes du déroulement.

6.1 Camunda Modeler

Je l'ai ensuite modélisé en me servant de ce que j'avais appris en cours d'urbanisation des systèmes d'information.

Pour cette première partie, j'ai utilisé le logiciel Camunda Modeler.

Figure 11 : Première modélisation du Service-Desk



(Camunda Modeler, servicedesk_v1.png)

Ce diagramme de processus métier a deux couloirs, celui du secrétaire et celui du technicien et un acteur externe en vert qui est l'utilisateur.

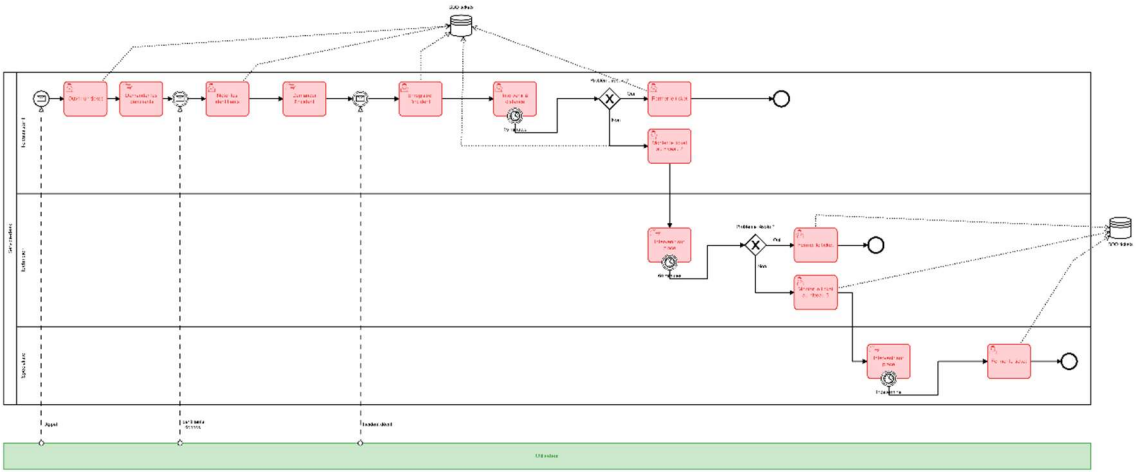
Le processus commence par un appel de l'utilisateur. Le secrétaire répond et demande les identifiants de l'utilisateur. Il attend qu'il les lui donne. Il les note puis demande l'incident. Ensuite, le secrétaire enregistre l'incident, ouvre un ticket et le transmet au technicien.

Le technicien intervient à distance pendant 15 minutes. Si le problème est résolu, le ticket est fermé, sinon, le ticket monte au niveau 2. Le technicien intervient sur place pendant 60 minutes. Si le problème est résolu, le ticket est fermé, sinon, le ticket monte au niveau 3. Le technicien continue d'intervenir jusqu'à ce qu'il résolve le problème.

J'ai envoyé ce diagramme à mon professeur. Il m'a fait remarquer plusieurs problèmes.

Le technicien ne s'occupe pas du premier, du deuxième et du troisième niveau. Le ticket doit être enregistré quelque part. Le technicien ne doit pas faire la tâche d'intervention puis attendre une certaine durée. Il faut que la tâche ait une certaine durée maximum.

Figure 12 : Deuxième modélisation du Service-Desk



(Camunda Modeler, servicedesk_v2.png)

Dans cette deuxième version du processus, j'ai ajouté une base de données « BDD tickets » pour stocker les tickets d'incidents.

J'ai aussi modifié les évènements de durée. En lisant la documentation sur internet, j'ai appris qu'il est possible d'attacher un évènement à une tâche, ce qui veut dire que la tâche se terminera si l'évènement survient. En faisant ainsi, si la durée définie est atteinte, la tâche se termine.

J'ai ensuite importé le diagramme sur Camunda Cloud.

6.2 Camunda Cloud

6.2.1 Échec du déploiement

Après avoir importé le diagramme, j'ai essayé de le déployer.

Le déploiement a échoué.

Figure 13 : Erreur de déploiement sur Camunda Cloud

(Camunda Cloud)

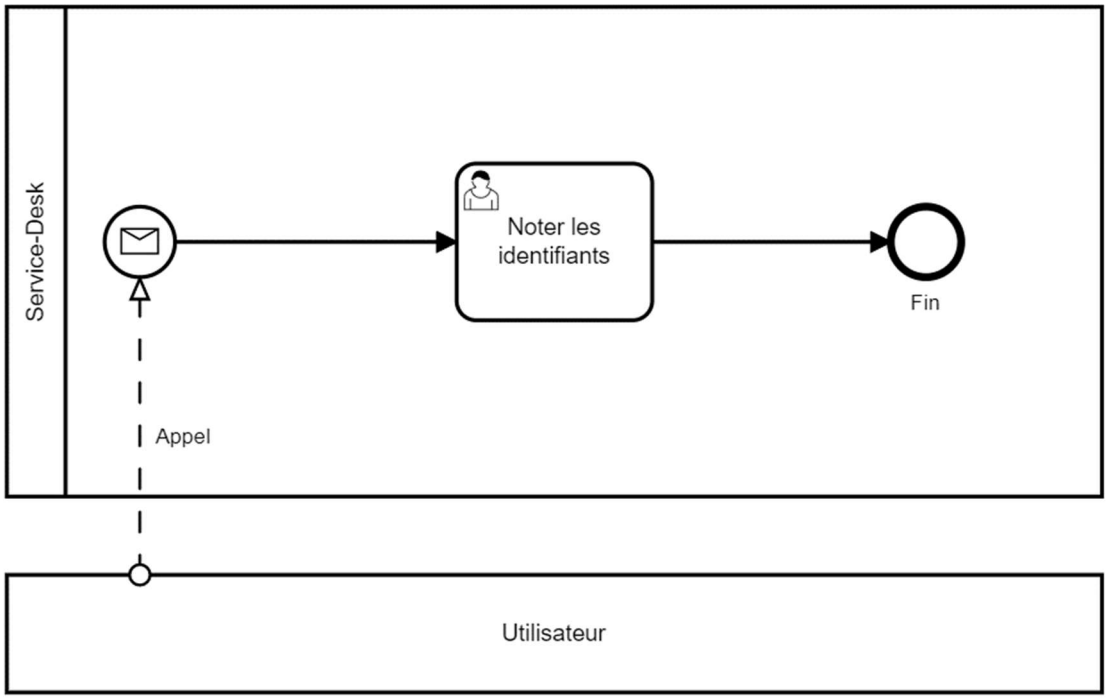
La recherche de l'erreur sur internet n'a rien donné.

Je n'avais jamais déployé de processus. J'imaginais bien que ce n'était pas aussi simple que ça.

Au lieu de rechercher l'erreur dans mon diagramme, j'ai décidé de recommencer depuis le début et d'essayer petit à petit pour être sûr que tout fonctionne.

6.2.2 Recommencer depuis le début

Figure 14 : Premier diagramme



(Camunda Cloud)

Ce processus commence par un appel de l'utilisateur. Le téléassistant note les identifiants et c'est la fin.

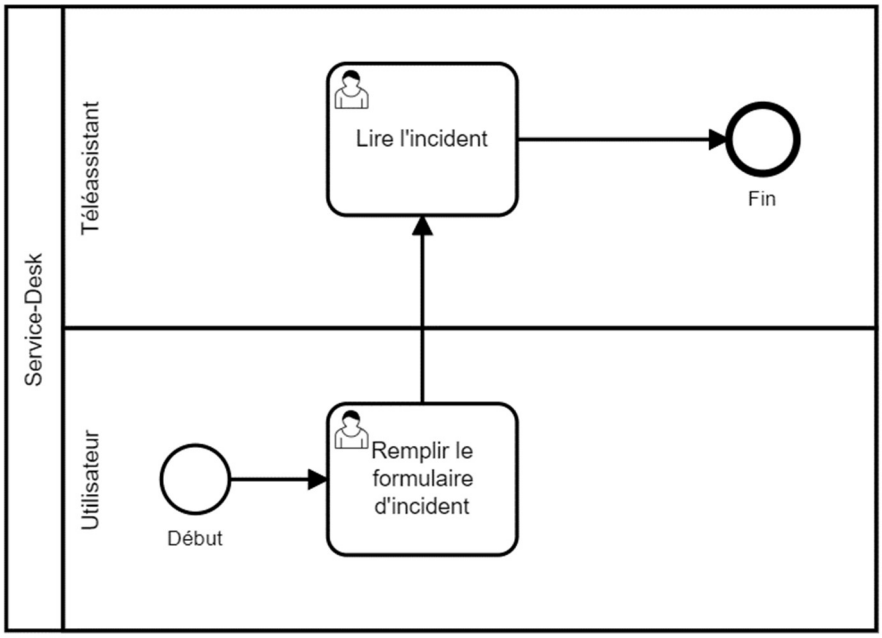
Mon problème ici est de réussir à simuler un appel pour débuter le processus.

Après de longues recherches, je me rends compte que simuler un appel n'est pas possible et simuler un message est bien trop compliqué à mettre en place.

Comme vu dans le chapitre sur le Service-Desk, la procédure peut commencer soit par un appel, soit par le remplissage d'un formulaire sur un GLPI. Je décide donc de changer le début du processus.

6.2.3 Formulaire au lieu d’un appel

Figure 15 : Deuxième diagramme



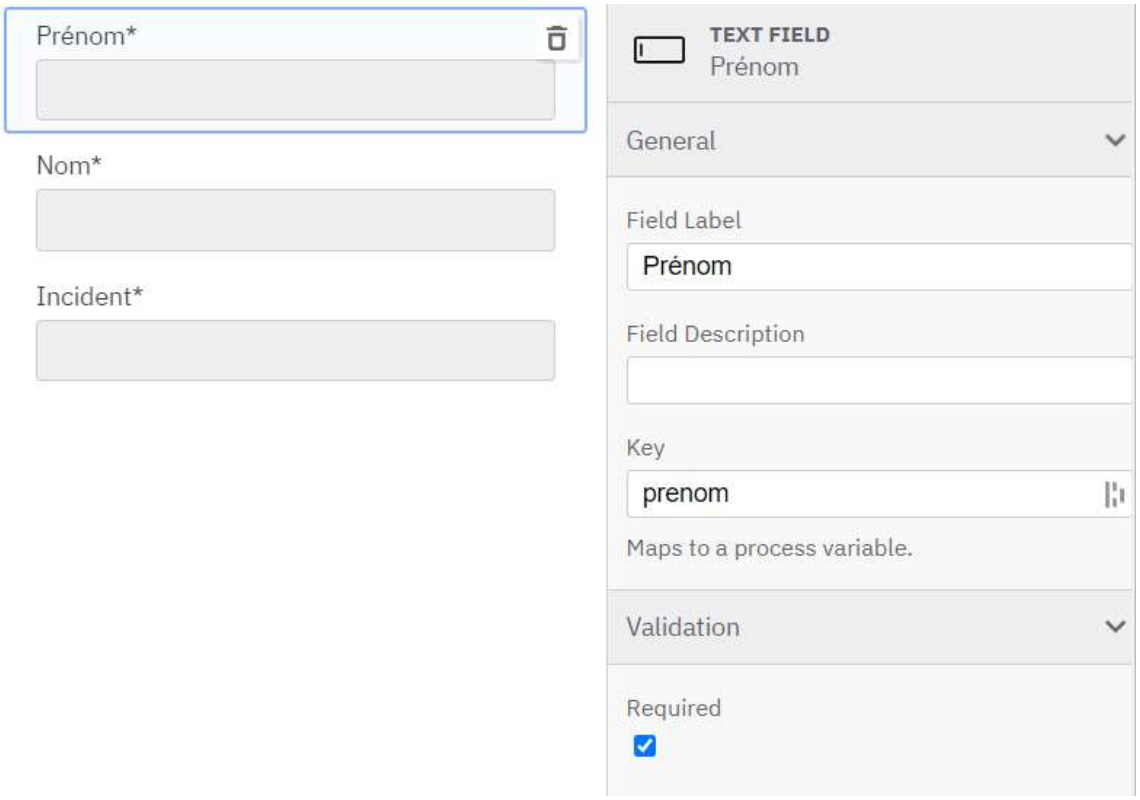
(Camunda Cloud)

L’utilisateur est maintenant un acteur interne au processus. C’est lui qui lance le processus en remplissant le formulaire d’incident.

Sur Camunda, il est possible de créer un formulaire puis de l’ajouter à une tâche. Ce formulaire sera rempli par l’utilisateur au moment voulu.

Je crée deux formulaires. Un formulaire destiné à l’utilisateur et un destiné aux intervenants (téléassistant, technicien, spécialiste).

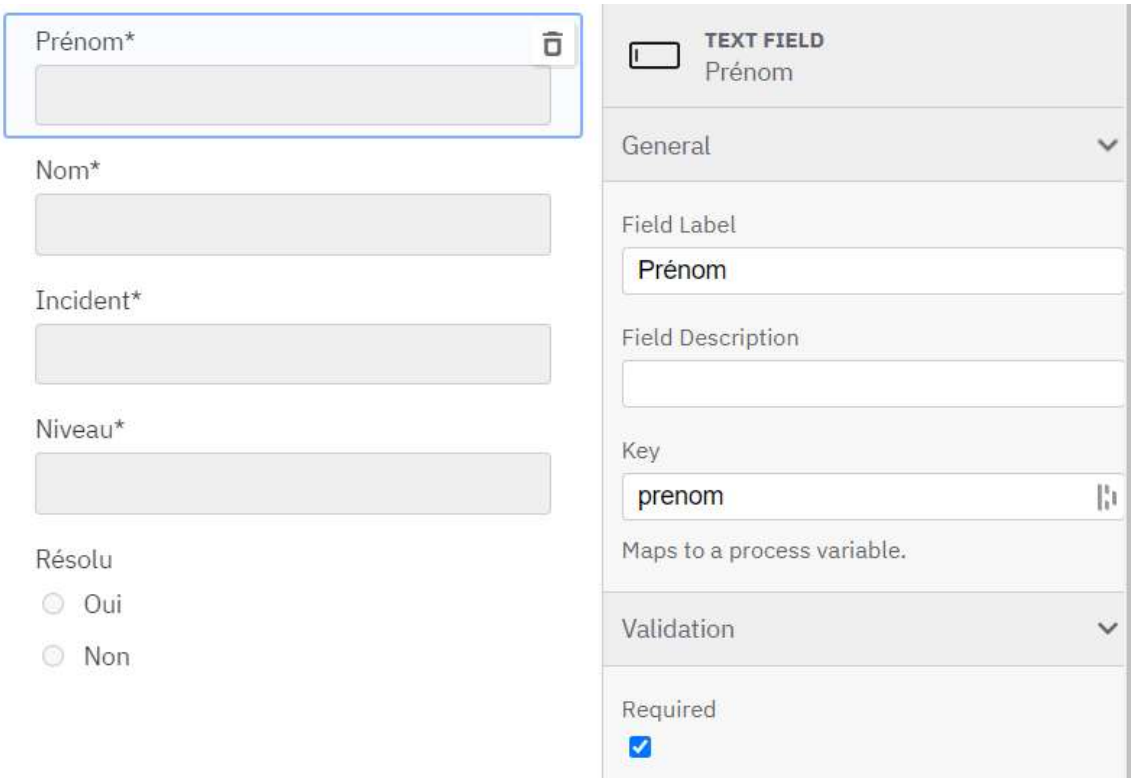
Figure 16 : Formulaire utilisateur



(Camunda Cloud)

Ce formulaire contient le nom, le prénom et l'incident.

Figure 17 : Formulaire intervenants



(Camunda Cloud)

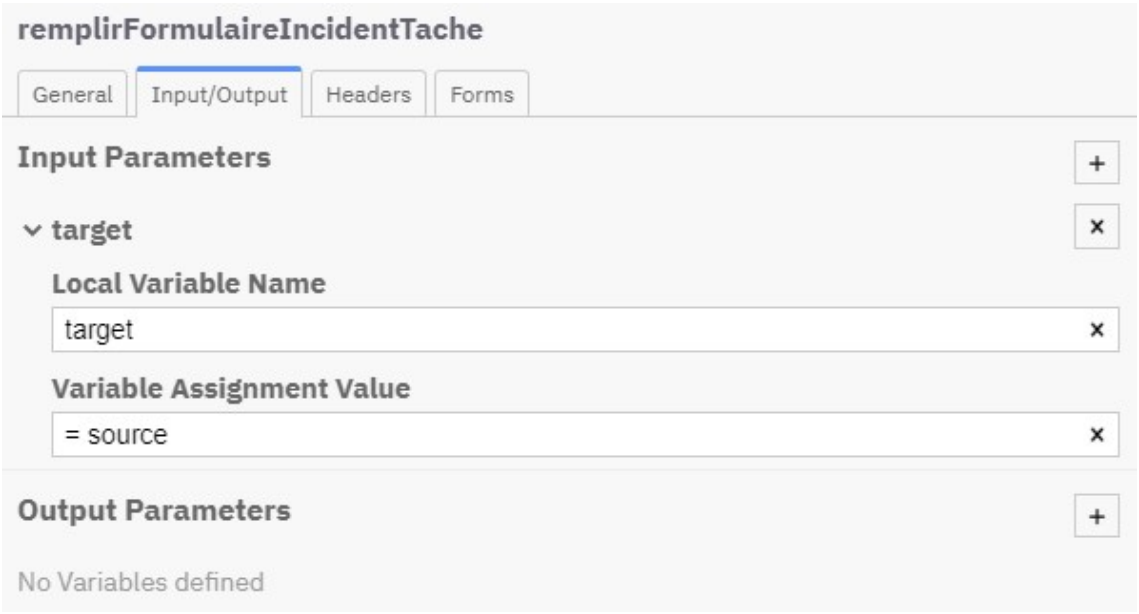
Ce formulaire contient le nom, le prénom, l'incident, le niveau qui va de 1 à 3 et si l'incident a été résolu.

Sur les deux captures d'écran de formulaire, j'ai sélectionné le champ « Prénom ». Dans les propriétés du champ sur la droite de la capture, il y a la propriété « Key ». Cette propriété est comme un nom de variable. Donc si je veux récupérer la valeur du prénom, j'ai juste à me servir de la variable.

Maintenant, j'aimerais faire que, de base, le champ « Résolu » soit à « non » et le champ « Niveau » soit à « 1 ».

6.2.4 Input/Output Parameters

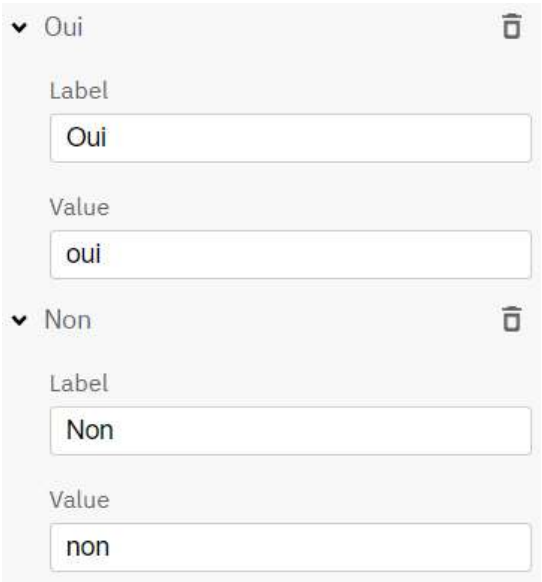
Figure 18 : Input/Output Parameters



(Camunda Cloud)

Sur Camunda, les tâches ont des propriétés. Parmi ces propriétés, il y a les « Input/Output Parameters ». Ce sont des assignations de variables avec « target » qui définit où « source » sera assigné.

Figure 19 : Valeurs de « Résolu »



(Camunda Cloud)

Dans le formulaire du ticket pour les intervenants, j'ai mis les valeurs du bouton radio « Résolu » à « oui » pour le oui et « non » pour le non.

Figure 20 : Formulaire utilisateur rempli

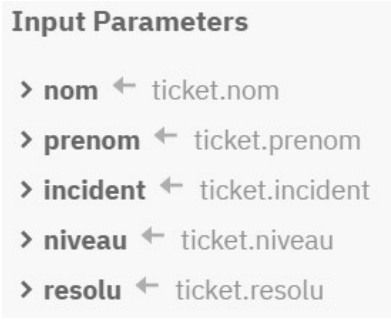
(Camunda Cloud)

Figure 21 : Output parameters

(Camunda Cloud)

En sortie de la tâche « remplir le formulaire d'incident », je crée un objet JSON « ticket » auquel j'assigne les valeurs des champs du formulaire, son niveau à 1 et son état à « non ».

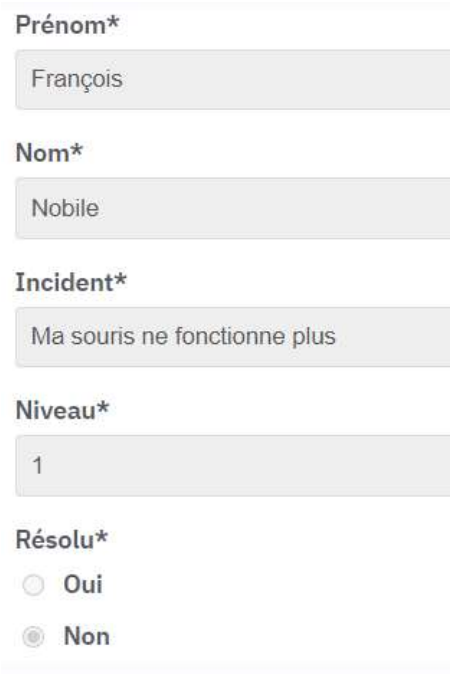
Figure 22 : Input parameters



(Camunda Cloud)

Ensuite, en entrée de la tâche « Lire l'incident », je récupère ces valeurs que j'assigne aux champs du formulaire.

Figure 23 : Formulaire intervenants rempli

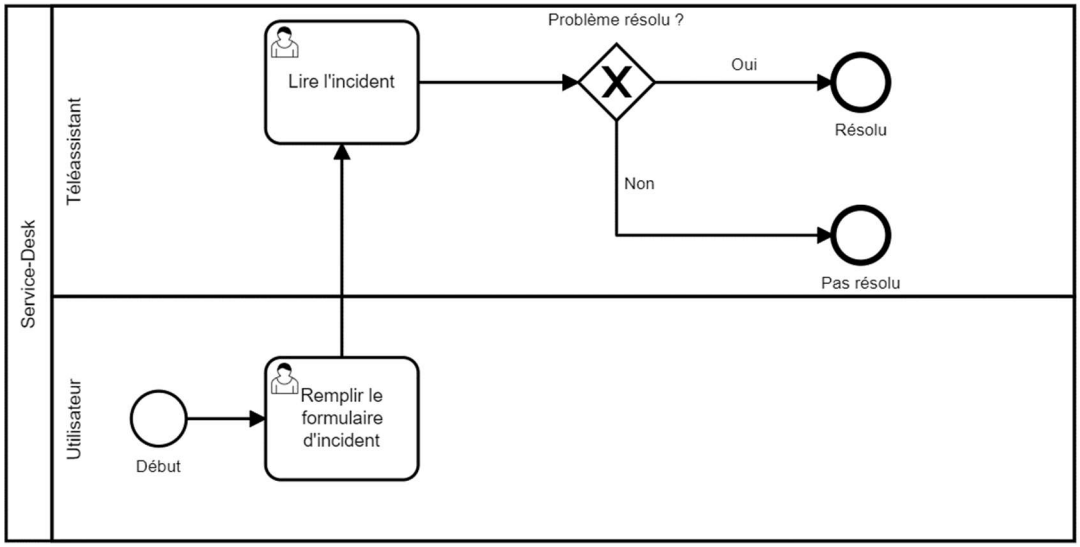


(Camunda Cloud)

Les informations du formulaire utilisateur sont bien gardées, le niveau est automatiquement mis à 1 et le ticket n'est pas résolu.

6.2.5 Résolu ou pas

Figure 24 : Diagramme avec passerelle

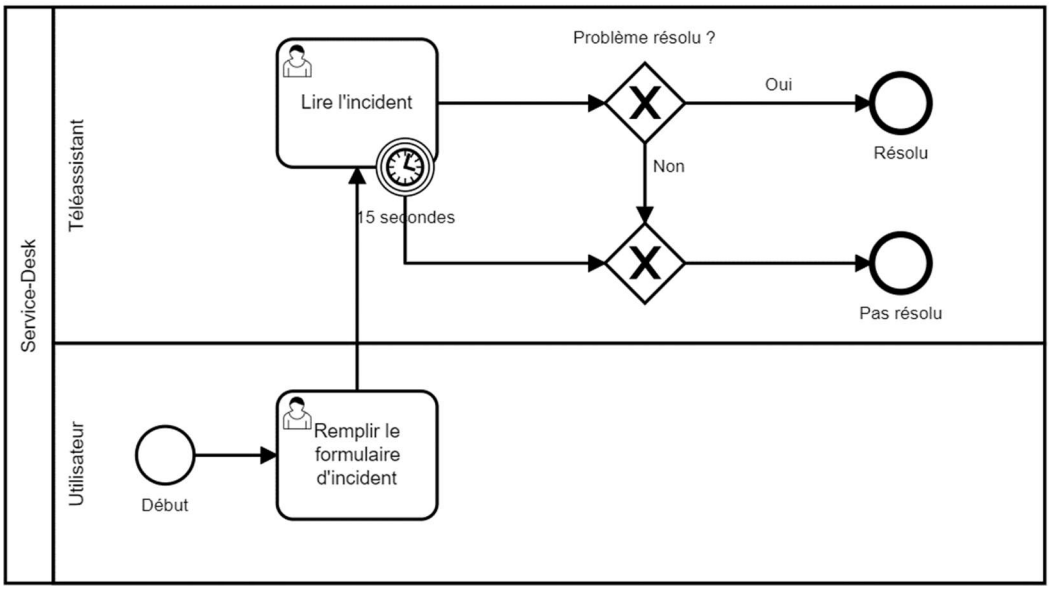


(Camunda Cloud)

À cette étape, j'ai simplement ajouté une passerelle pour décider quel chemin prendre en fonction de la valeur de « ticket.resolu ».

6.2.6 Timer Event

Figure 25 : Diagramme avec Timer Event



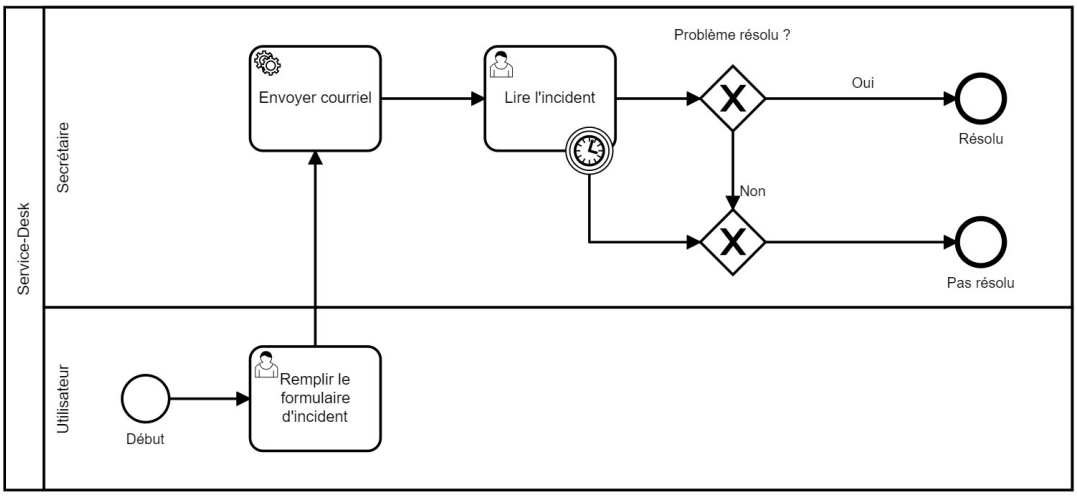
(Camunda Cloud)

J'ai ajouté un évènement timer à la tâche « Lire l'incident » pour qu'elle se termine automatiquement après 15 secondes et passe directement à « Pas résolu ». La durée est définie en utilisant le format ISO 8601[9].

6.2.7 Envoyer un courriel

Le dernier élément qui manque à mon diagramme est l'envoi d'un courriel pour notifier le téléassistant d'un nouvel incident.

Figure 26 : Diagramme avec envoie d'un courriel



(Camunda Cloud)

L'ajout d'une tâche service permet cela. Il faut simplement lui donner un type et un nombre de tentatives.

Figure 27 : Propriétés tâche service

(Camunda Cloud)

Je lui ai donné le type « email » et une tentative.

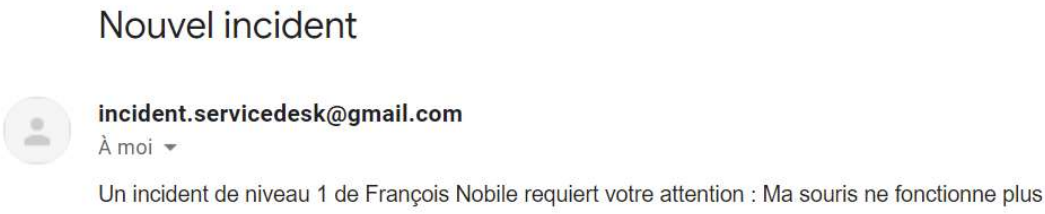
[9] https://docs.camunda.io/docs/reference/bpmn-processes/timer-events/timer-events/#timers

Pour exécuter cette tâche, il faut programmer un « worker ». Camunda met à disposition, sur GitHub, le code d'un « worker » en Java[10]. Ensuite, j'ai cherché comment envoyer un courriel en Java également[11]. Le code est en annexe.

Mon « worker » se connecte d'abord à mon « Cluster » Camunda Cloud. Quand il détecte un « job » de type « email », il récupère les variables nom, prénom, incident et niveau pour former le contenu du message, il prépare le courriel (source, destination, objet, contenu) puis l'envoi.

J'ai utilisé une adresse Gmail comme serveur SMTP pour envoyer le courriel.

Figure 28 : Exemple de courriel



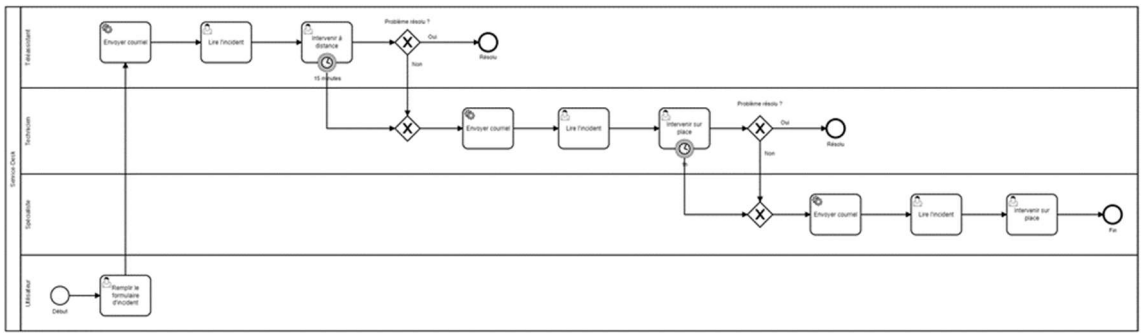
(François Nobile, mail.google.com)

Maintenant que j'ai mis tous les éléments en place, il ne me reste plus qu'à modéliser le processus en entier.

[10] https://github.com/camunda-cloud/camunda-cloud-get-started/blob/master/java/src/main/java/io/camunda/getstarted/EmailWorker.java
[11] https://netcorecloud.com/tutorials/send-email-in-java-using-gmail-smtp/

6.2.8 Diagramme final

Figure 29 : Diagramme final



(Camunda Cloud)

Il y a maintenant quatre acteurs.

- L’utilisateur : débute le processus en ouvrant un ticket.
- Le téléassistant : s’occupe du premier niveau et intervient à distance.
- Le technicien : s’occupe du deuxième niveau et intervient sur place.
- Le spécialiste : s’occupe du troisième niveau et intervient sur place.

Le niveau change tout seul lorsqu’il le faut et les acteurs sont notifiés par courriel.

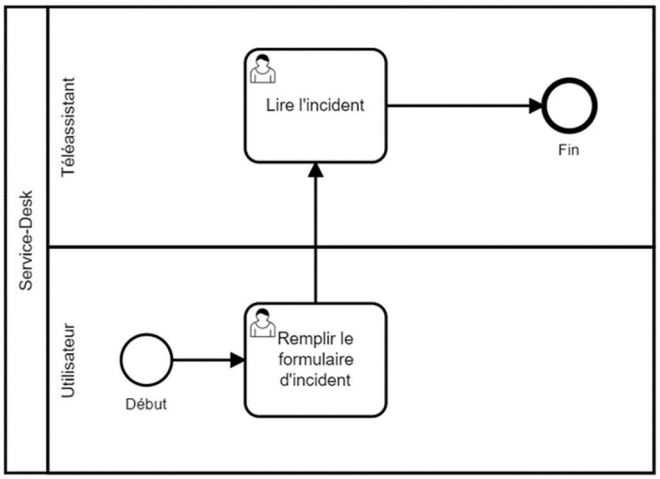
7. Problèmes et difficultés rencontrées

Durant la modélisation du processus, j'ai rencontré quelques problèmes et difficultés.

Le premier problème que j'ai rencontré est celui du chapitre « Échec du déploiement ». Rechercher sur internet l'erreur que j'avais ne donnait rien, alors c'était plus simple de recommencer depuis le début.

Le deuxième gros problème que j'ai eu était pour mettre une valeur par défaut à « niveau » et « resolu » (cf. input/output parameters).

Figure 30 : Processus



(Camunda Cloud)

J'avais mis les variables en paramètre de sortie de la première tâche, et quand je regardais dans le formulaire des intervenants à la deuxième tâche, toutes les variables des champs du formulaire qui avaient été remplis (« nom », « prenom » et « incident ») étaient vides, mais « niveau » était bien à 1 et « resolu » à non.

J'ai demandé de l'aide sur le forum de Camunda, mais malheureusement, personne n'a pu m'aider.

En essayant plusieurs manières de faire, j'ai fini par trouver celle que j'ai utilisée et qui fonctionne, à savoir, créer un objet JSON « ticket » et mettre les variables en entrée et sortie.

Ma dernière difficulté était d'envoyer un courriel avec une tâche. Sur Camunda Cloud, il y a un processus d'exemple ou un courriel est envoyé, mais ils n'expliquent pas comment le faire. En cherchant sur internet, j'ai trouvé qu'un *worker* est requis pour s'occuper d'une tâche et j'ai suivi une documentation trouvée sur « Github » qui explique

comment faire. Il me restait plus que l'envoie du courriel. Pour ça, je ne sais pas pourquoi, mais je cherchais à envoyer un courriel avec Camunda Cloud, ce qui m'a fait perdre énormément de temps, alors que je devais simplement rechercher comment envoyer un courriel en Java.

Plus généralement, le problème avec Camunda est que, quand j'ai une erreur, une question ou un problème, la taper dans le navigateur ne suffit pas. Ce n'est pas comme pour les langages de programmation, ou, la plupart du temps, la réponse est trouvée facilement.

Avec Camunda, il faut soit demander de l'aide dans le forum, soit lire la documentation. Le forum est bien, mais il repose sur la communauté. Si personne ne peut ou ne veut vous répondre, vous n'aurez pas de solution.

La documentation est bien écrite, mais elle est disponible uniquement en anglais ou en allemand. Aussi, lorsque vous lisez un sujet, ce sujet fait référence à un autre sujet, qui fait référence à un autre sujet et ainsi de suite. Je me suis retrouvé avec une dizaine d'onglets de navigateur ouverts pour essayer de comprendre une seule chose.

Encore un dernier point sur Camunda Cloud. Cette plateforme en ligne est assez récente et les développeurs sortent des mises à jour régulièrement. Il m'est arrivé que ce que j'avais fait fonctionnait, puis deux jours plus tard ça ne fonctionnait plus parce que ce n'était pas la même version du logiciel.

8. Conclusion

En conclusion, je peux dire que ce test d'automatisation d'un processus sur une plateforme BPMN est une réussite.

Ce travail a quand même requis de la programmation, mais pas directement sur Camunda. Si le « worker » avait déjà été configuré et que l'entreprise dans laquelle je travaillais avait établi une norme pour envoyer un courriel, je n'aurais rien eu besoin de faire.

Ce processus n'était pas compliqué ni complexe, donc il est normal qu'il puisse t'être réalisé avec une plateforme BPMN. Un processus d'un niveau plus élevé serait aussi réalisable, mais il faudrait plus d'expérience avec la plateforme.

Ce travail m'a fait réaliser ce que sont les logiciels *low-code* et *no-code* et ce qu'ils proposent, ce qui me fait me demander ce dont ils seront capables de faire dans 10 ans.

9. Bibliographie

FAURE Alain, SOLLIER Laurent, 2019. Le low-code, comment ça marche ?. *octo* [en ligne]. 5 septembre 2019. [Consulté le 29 septembre 2021]. Disponible à l'adresse : https://blog.octo.com/le-low-code-comment-ca-marche/

GRANGER, Laurent, 2021. BPM – Business Process Management : comprendre et appliquer. *Manager GO !* [en ligne]. Dernière mise à jour le 9 septembre 2021. [Consulté le 30 août 2021]. Disponible à l'adresse : https://www.manager-go.com/organisation-entreprise/bpm.htm

2021. A Full Overview of Business Process Management (BPM). *Kissflow* [en ligne]. 15 juin 2021. [Consulté le 30 août 2021]. Disponible à l'adresse : https://kissflow.com/workflow/bpm/business-process-management-overview/

Business Process Management. *Wikipedia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 21 avril 2021 20:01. Consulté le 30 août 2021. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Business_Process_Management

Business Process Model and Notation. *Wikipedia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 16 avril 2020 20:58. Consulté le 30 août 2021. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Business_process_model_and_notation

BPMN 2.0 Implementation Reference. *Camunda Docs* [en ligne]. Consulté le 4 septembre 2021. Disponible à l'adresse : https://docs.camunda.org/manual/7.14/reference/bpmn20/

Berndruecker, 2021. EmailWorker.java. *GitHub* [en ligne]. Dernière mise à jour le 21 juillet 2021. Consulté le 29 août 2021. Disponible à l'adresse : https://github.com/camunda-cloud/camunda-cloud-get-started/blob/master/java/src/main/java/io/camunda/getstarted/EmailWorker.java

Rishabh Mishra, 2020. How To Send Email In Java Using Gmail SMTP ?. *netcore* [en ligne]. Dernière mise à jour de la page le 17 juillet 2020. Consulté le 29 août 2021. Disponible à l'adresse : https://netcorecloud.com/tutorials/send-email-in-java-using-gmail-smtp/

JOSET, Patrick, 2021. *Service-Desk.pptx* [fichier PowerPoint]. Document envoyer par l'auteur lui-même.

AÏDONIDIS-FLÜCKIGER, Christine, 2021. *626-1 Couche Métier 2021 Sem2.ppt* [en ligne]. [Consulté le 10 septembre 2021]. Disponible à l'adresse : https://cyberlearn.hes-so.ch/mod/resource/view.php?id=1446994

JetBrains, 2021. *Intellij IDEA 2021.2.1 (Community Edition)* [logiciel]. Version 11.0.11+9-1504.16. JetBrains s.r.o. [Consulté le 13 juin 2021]. Disponible à l'adresse : https://www.jetbrains.com/fr-fr/idea/download/#section=windows

Camunda, 2021. *Camunda Cloud* [logiciel en ligne]. Camunda. [Consulté le 20 juillet]. Disponible à l'adresse : https://console.cloud.camunda.io/

Camunda, 2021. *Camunda Modeler* [logiciel]. Version 4.6.0. Camunda. [Consulté le 12 mai 2021]. Disponible à l'adresse : https://camunda.com/download/modeler/

Camunda. *camunda.com* [en ligne]. [Consulté le 10 octobre 2021]. Disponible à l'adresse : https://www.google.com/url?sa=i&url=https%3A%2F%2Fcamunda.com%2F&psig=AOvVaw1KYkNVObOUERD11SquuOjv&ust=1633939846201000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCPDJ_Leyv_MCFQAAAAAdAAAAABAD

Annexe 1 : code du worker

La classe « ZeebeClientFactory » qui me permet de me connecter à mon processus.

```
import io.camunda.zeebe.client.ZeebeClient;

public class ZeebeClientFactory {
    public static ZeebeClient getZeebeClient() {
        return ZeebeClient.newCloudClientBuilder()
                .withClusterId("03a42239-ab41-4cd8-a90b-9a99e4e8fe67")
                .withClientId("4gPVVmeSYyps.BoGDkLR9tYpwjGFN~7-")

.withClientSecret("4DFyeUcZh4L0HsK~GzgZQF9MYRBPJw853CoZNoebKGTKNwxG6xNh.myOB8E74iD-")
                .withRegion("bru-2")
                .build();
    }
}
```

La classe « EmailWorker » qui permet d'envoyer un courriel.

```
import io.camunda.zeebe.client.ZeebeClient;
import java.util.Scanner;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class EmailWorker {
    private static final Logger LOG = LogManager.getLogger(EmailWorker.class);

    public static void main(String[] args) {

        // Recipient's email ID needs to be mentioned.
        String to = "francois.nobile@gmail.com";
        // Sender's email ID needs to be mentioned
        String from = "incident.servicedesk@gmail.com";

        // Assuming you are sending email from through gmails smtp
        String host = "smtp.gmail.com";
        // Get system properties
        Properties properties = System.getProperties();
        // Setup mail server
        properties.put("mail.smtp.host", host);
        properties.put("mail.smtp.port", "465");
        properties.put("mail.smtp.ssl.enable", "true");
        properties.put("mail.smtp.auth", "true");

        // Get the Session object.// and pass username and password
        Session session = Session.getInstance(properties, new javax.mail.Authenticator()
{
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication("incident.servicedesk@gmail.com",
"qns#h?58pJGMqTP9");
            }
        });

        // Used to debug SMTP issues
        session.setDebug(true);

        try (ZeebeClient client = ZeebeClientFactory.getZeebeClient()) {

            client.newWorker().jobType("email").handler((jobClient, job) -> {

                // get information about the ticket
                String nom = (String) job.getVariablesAsMap().get("nom");
                String prenom = (String) job.getVariablesAsMap().get("prenom");
```

```
                String incident = (String) job.getVariablesAsMap().get("incident");
                String niveau = Integer.toString((Integer)
job.getVariablesAsMap().get("niveau"));
                final String message_content = "Un incident de niveau " + niveau + " de
" + prenom + " " + nom + " requiert votre attention : " + incident;

                LOG.info("Sending email with message content: {}", message_content);

                try {
                    // Create a default MimeMessage object.
                    MimeMessage message = new MimeMessage(session);

                    // Set From: header field of the header.
                    message.setFrom(new InternetAddress(from));

                    // Set To: header field of the header.
                    message.addRecipient(Message.RecipientType.TO, new
InternetAddress(to));

                    // Set Subject: header field
                    message.setSubject("Nouvel incident");

                    // Now set the actual message
                    message.setText(message_content);

                    System.out.println("sending...");
                    // Send message
                    Transport.send(message);
                    System.out.println("Sent message successfully....");

                jobClient.newCompleteCommand(job.getKey()).send();

                } catch (MessagingException mex) {
                    mex.printStackTrace();
                }

            }).open();

            // run until System.in receives exit command
            waitUntilSystemInput("exit");
        }
    }

    private static void waitUntilSystemInput(final String exitCode) {
        try (final Scanner scanner = new Scanner(System.in)) {
            while (scanner.hasNextLine()) {
                final String nextLine = scanner.nextLine();
                if (nextLine.contains(exitCode)) {
                    return;
                }
            }
        }
    }
}
```