Université Paris Dauphine

Projet de modélisation.

Modélisation de la gestion de stage.

Axel Acuna et Fils-Yonga Jacques
04/01/2019

Table des matières

Le projet suivant vise à mettre en place un système automatisée pour la gestion des stages. La plateforme utilisée est Activiti-explorer. Elle nous a permis la modélisation, création et la conception du workflow de l'université. La gestion de stage actuelle est gérée par deux entités de l'université les responsables pédagogiques et le service stage. Nous allons dans un prémier temps vous présenter la modélisation en diagramme BPMN2.0 du processus, puis nous allons vous présenter la version automatisée du processus avec les différents détails d'implémentation puis nous conclurons avec les problèmes rencontrés au cours du projet.

Processus BPMN2.0 de la gestion des stages.

Le workflow actuel se subdivise en deux grandes étapes la prémière qui concerne la soumission, validation et enregistrement de l'offre de stage. Puis un deuxième grand processus qui a lieu durant la réalisation du stage qui concerne. En amont à la création du digramme nous avons créé des groupes et des utilisateurs dans activiti, « respo : Responsable », « etu : Etudiant » , « admin : Admin(Administration) », « tuteur1 : Tuteur ».

SELECT * FROM ACT_ID_GROUP;

ID_	REV_	NAME_	TYPE_
management	1	Management	assignment
sales	1	Sales	assignment
marketing	1	Marketing	assignment
engineering	1	Engineering	assignment
user	1	User	security-role
admin	1	Admin	security-role
respo	1	Responsable	security-role
etu	1	Etudiant	assignment
tuteur1	2	Tuteur	assignment

(9 rows, 14 ms)

Edit

Rf : voir partie problème qui explique le pourquoi ça été fait sous H2.

La soumission, validation et enregistrement de l'offre de stage.

Voici la prémière première étape modélisé en processus BPMN2.0 qui explicite le processus de soumission validation

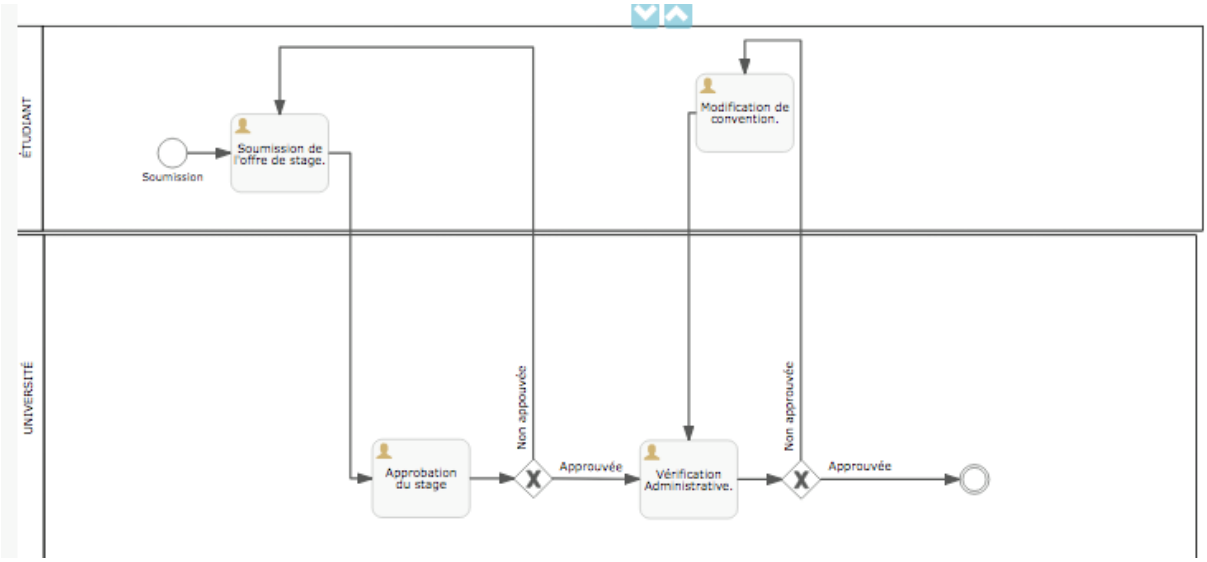


Figure 1 Processus convocation

Dans ce processus l'étudiant fait la soumission de son stage qui est par la suite vérifié par le responsable pédagogique.

Ainsi l'étudiant aura l'affichage suivante.

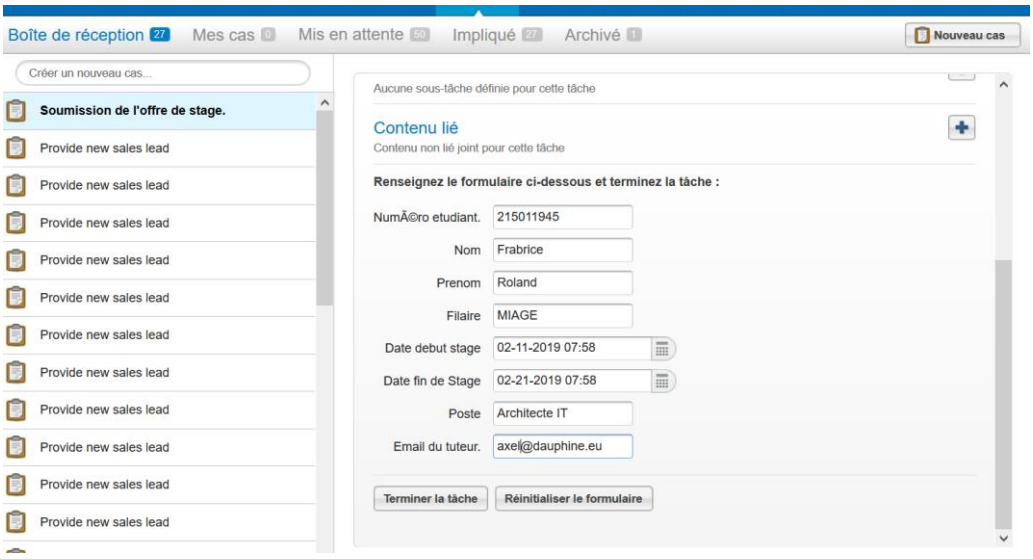


Figure 2 Tache soumission de l'étudiant.

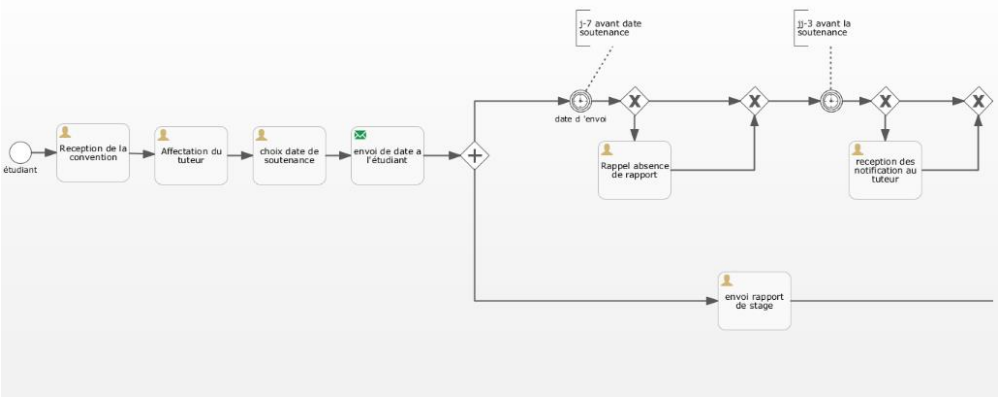
Propriété de la tache soumission offre de stage.

Id	Name	Type
numIdEtu	Numéro etudiant.	string
nomEtu	Nom	string
prenom	Prenom	string
filaire	Filaire	string
dateDebut	Date debut stage	date
dateFin	Date fin de Stage	date
poste	Poste	string
mailTutor	Email du tuteur.	string

Figure 3 Propriétés de la tache soumission. (sert pour la BD)

La personne qui réçoit la tache de verifier le contenu est les responsable pédagogique (Assigne : respo) En cas de non-approbation l'étudiant est dans l'obligation de réeffectuer la tache correctement.

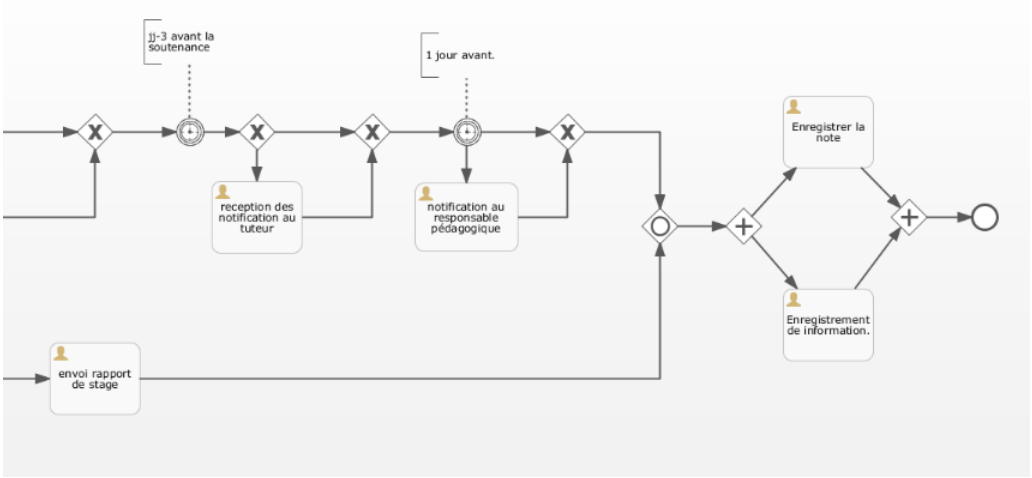
Affectation



Nous avons ici la réalisation finale du processus qui se fait en plusieurs étapes :

L'étudiant fait une soumission de stage, après il y a la réception par le service de stage, si toutes les informations sont bien renseignées, elle est validée.

Ensuite on lui affecte un tuteur s'il ne l'a pas déjà choisi lui-même. A la fin de son stage, le choix de la date de soutenance se fait par le service de stage. Si l'étudiant n'envoie pas son rapport de stage avant une semaine des notifications de rappels lui sont envoyés.

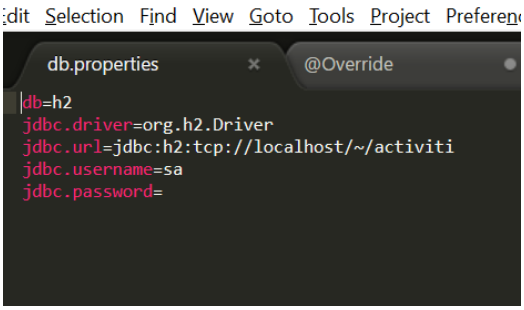


Dans la version finale, nous avons décidé que les notifications se font par messages. A la fin nous avons 2 taches qui se ferons par la suite en automatiques, celle de l'enregistrement de la note. Les informations concernant le stage seront enregistrées dans la BD.

Choix technique et automatisation du processus.

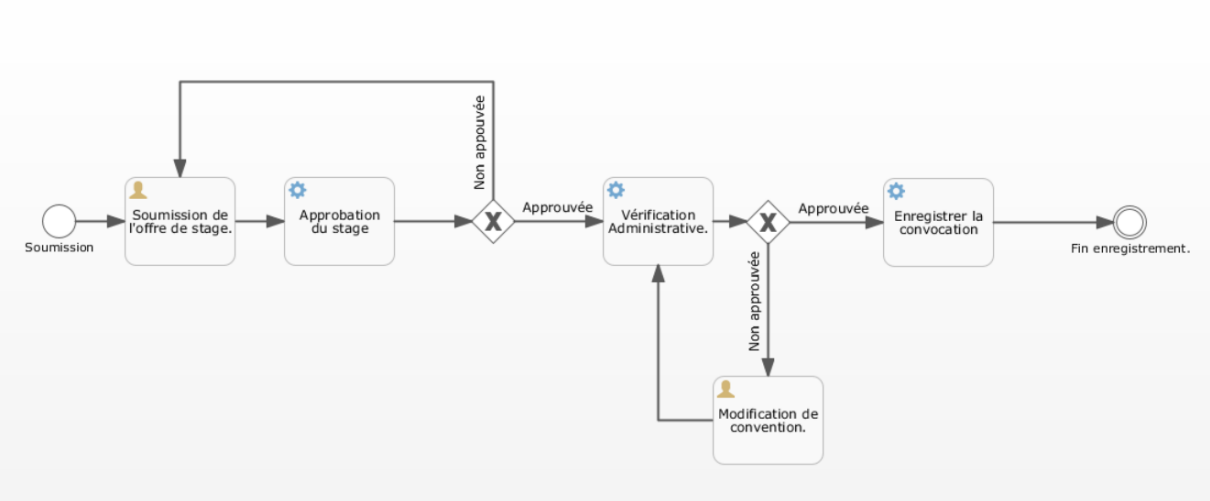
L'objectif de cette partie est d'expliquer et des présenter comment nous avons procédé pour automatiser le processus via les services offerts par activiti-explorer.

Nous avons dans un prémier temps essayer de nous connecter au sein de la base donnée d'activiti. Pour s'y faire nous avons utilisé H2 data base. Nous avons dans un prémier temps changé le fichier *db.properties* WEB-INF dans webapp.



Ceci nous a permis d'avoir accès aux tables via H2.

Nous avons après avoir fait cela modifié le processus bpmn fait précédemment en mettant *approbation du stage* et *vérification administrative* en automatique (en tant que tache service). Et nous avons ajouté un service automatique qui enregistre les informations de la convocation dans un table convocation où sont enregistrer les données qui concerne la convocation saisie par l'étudiant.



Vérification pédagogique :

Cette méthode nous retourne un Booléen. Dans ce cas on vérifie si les toutes informations sont correctes si c 'est le cas elle retourne vrai au cas contraire elle retourne faux.

Si faux est renvoyé alors la validation pédagogique du stage n'a pas été approuvé. Sinon elles renvoient vrai et passe à la tache suivante.

N.B : Fait appelle à la Class org.activiti.MyDelegate.

```
public class MyDelegate implements JavaDelegate {
    @Override
    public void execute(DelegateExecution execution) throws Exception {
        execution.setVariable( s: "approbation", verificationPedagogique(execution));
    }

    private Boolean verificationPedagogique(DelegateExecution execution){
        Map<String,Object> mapVar = execution.getVariables();
        for(String key : mapVar.keySet()){
            Object val = mapVar.get(key);
            if(val instanceof String){
                String sVal = (String) val;
                if(sVal.equals("")){
                    return false;                }
            }
        }
        String post = (String) execution.getVariable( s: "poste");

        return post.length() >= 5;
    }
```

Vérification administrative :

Fait appel à une méthode de type booléen, elle vérifie si le mail du tuteur lus en input contient le caractère « @ » si oui elle retourne vrai au cas contraire elle retourne faux.

Si faux est renvoyé alors la validation administrative du stage n'a pas été approuvé. Sinon elle renvoie vrai et passe à la tache suivante.

```
public class MyDelegate2 implements JavaDelegate {
    @Override
    public void execute(DelegateExecution delegateExecution) throws Exception {

        delegateExecution.setVariable( s: "approbation",verificationAdministratif(delegateExecution));
    }

    private Boolean verificationAdministratif(DelegateExecution execution ){
        String email  =(String)execution.getVariable( s: "mailTutor");

        if(email.contains("@")) {
            return true;
        }
        return  false;
    }
```

N.B : Fait appelle à la Class org.activiti.MyDelegate2.

Enregistrement de la convention :

La convention de stage est enregistrée dans la base de données via une requete SQL. Qui enregistre dans la base de donnée jdbc :h2 :tcp://localhost/-/activiti de H2 DB, toutes les informations saisies par l'étudiant.

Voici le code chargé de d'enregistrer les informations.

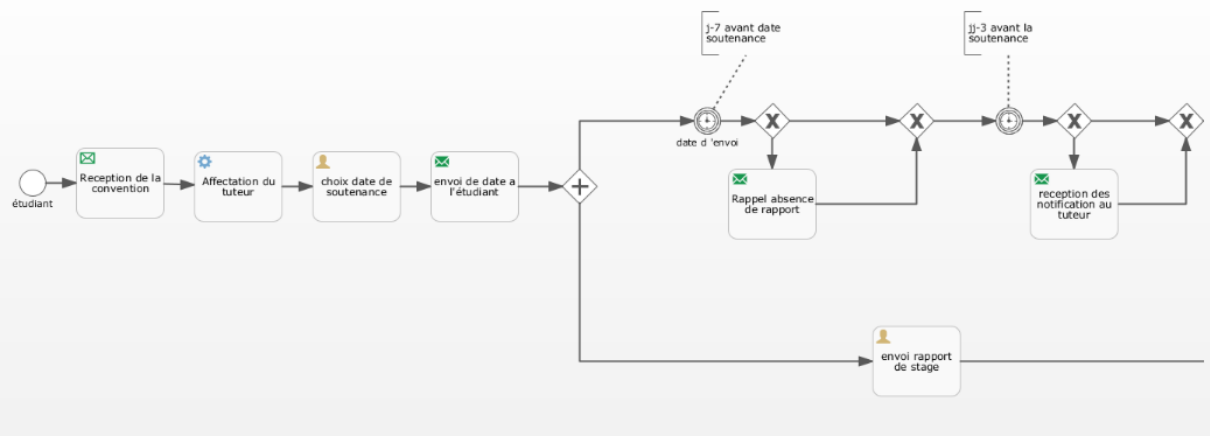
```java
public class MyDelegate3 implements JavaDelegate {
    @Override
    public void execute(DelegateExecution delegateExecution) throws Exception {
        //Create connection.
        Class.forName("org.h2.Driver");
        Connection conn = DriverManager.getConnection( url: "jdbc:h2:tcp://localhost/~/activiti", user: "sa", password: "");
        Statement stmt = conn.createStatement();
        //Create Table
        String requete = "CREATE TABLE IF NOT EXISTS " + "CONVOCATION" + " (" + "NUMETU VARCHAR(255), " +
                "NOM VARCHAR(255), PRENOM VARCHAR(255), " +
                "FILIAIRE VARCHAR(255), "
                + "DATEDEBUT DATE, DATEFIN DATE, POSTE VARCHAR(255), EMAILTUTOR VARCHAR(255), PRIMARY KEY(NUMETU)" + ");";
            //execution
        stmt.execute(requete);

        System.out.println("** Table create **");
        //add convocation in INSERT ROW.
        PreparedStatement pst = conn.prepareStatement( sql: "INSERT INTO \"CONVOCATION\" values (?,?,?,?,?,?,?,?)");
        pst.setString( parameterIndex: 1, (String) delegateExecution.getVariable( s: "numIdEtu"));
        pst.setString( parameterIndex: 2, (String) delegateExecution.getVariable( s: "nomEtu"));
        pst.setString( parameterIndex: 3, (String) delegateExecution.getVariable( s: "prenom"));
        pst.setString( parameterIndex: 4, (String) delegateExecution.getVariable( s: "filaire"));
        pst.setDate( parameterIndex: 5, (Date) delegateExecution.getVariable( s: "dateDebut"));
        pst.setDate( parameterIndex: 6, (Date) delegateExecution.getVariable( s: "dateFin"));
        pst.setString( parameterIndex: 7, (String) delegateExecution.getVariable( s: "poste"));
        pst.setString( parameterIndex: 8, (String) delegateExecution.getVariable( s: "mailTutor"));
        //UPDATE TABLE
        pst.executeUpdate();
        System.out.println("** Row inserted **");
        conn.close();
    }
}
```

Elle crée un table convocation si elle n'est pas présente puis elle rajoute les la ligne correspondant aux informations saisies par l'étudiant.

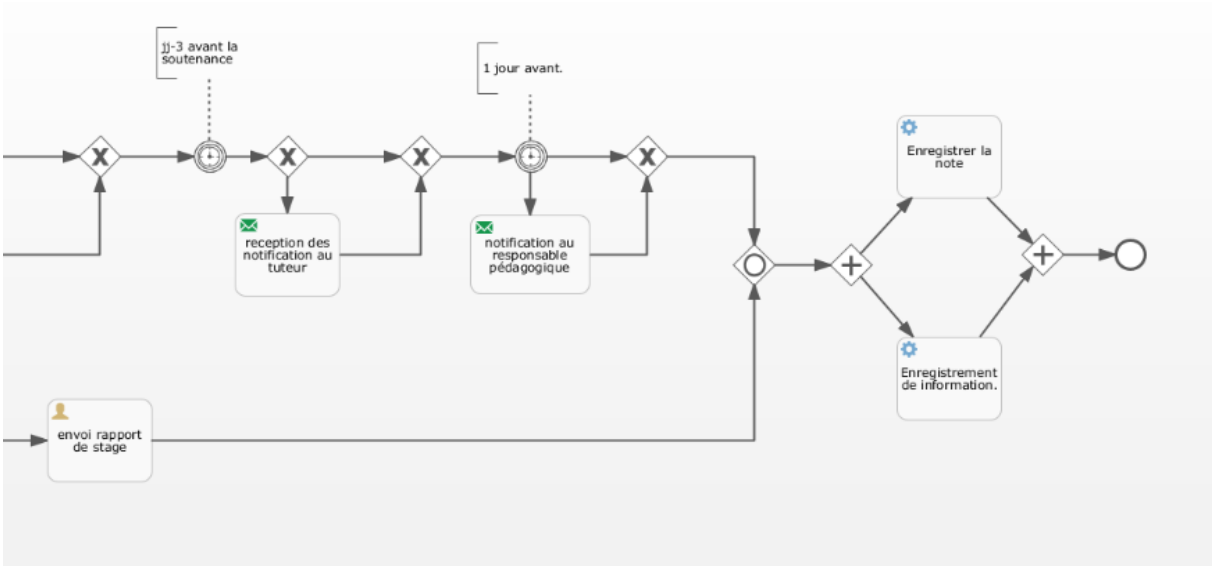
Cette tache permet ainsi une sécurité supplémentaire en enregistrant dans une base donnée les convocations saisies par l'étudiant.

Processus incomplet.

Pour la deuxième partie nous n'avons pas pu crée et automatiser le deuxième processus. Toute fois nous avons eu des idées le concernant. Voici comment on aurait procédé :



On aurait ainsi automatisé l'affectation du tuteur en faisant appel à la base de données et en attribuant un tuteur de l'université à l'étudiant. Les actions des rappels lors de problème de delais auraient été effectué avec un envoie de mail pour les personnes concernées.



L'enregistrement se serait effectué au sein de la base données de façon automatique.

On aurait créé deux classes qui effectue les taches suivantes.

Affectation Tuteur :

Nous aurions implémenté la méthode comme suit :

On aurait vérifié si l'étudiant a déjà un tuteur attribué si non on lui aurait cherché dans notre base de données un tuteur.

Envoi de messages d'Alertes :

Ici cette méthode prendrait comme paramètre une date et calcule la différence entre notre date enregistre dans la base de données si la différence est inférieure ou égale à 7 la fonction envoi des messages (notification de type String)

Nous allons dans partie suivant expliciter le problème qui nous ont empêché d'avancer dans l'automatisation du deuxième processus.

Problème rencontrés et modifications apportées :

Dans cette rubrique nous allons mettre en évidence les problèmes rencontrés au sein du projet et les différentes solutions qu'on a pu dans certaine situation mettre en place.

- Nous n'avons pas pu faire des sub-processus dans activiti ce qui nous a bloqué pour faire un diagramme cohérent e propre.
- Nous avons dû utiliser un système alternatif pour la création des groupes et des utilisateurs. En effet le bouton « crée groupe » ou « crée utilisateur » n'était pas disponible durant tout le projet ainsi nous avons ajouté directement via les tables de H2 BD les groupes et les utilisateurs.
- En ce qui concerne l 'impression de la convention de stage et l'envoi à l'entreprise, ainsi nous avons créé des processus pour chaque étape. Néanmoins nous n'avons pas su activer la fonctionnalité envoie de mail ni de document, ainsi l'étudiant ne reçoit aucun document qui concerne la convocation à imprime.

Nous avons donc séparé le processus principal en deux grands processus en supposant qu'il y existe la possibilité de crée de sous processus.

Pour revenir a l'envoie de mail nous avons fait la chose suivante au sein de notre fichier. En configurant *engine.properties.*

email.enabled=true
email.host=smtp.gmail.com
email.port=587

email.tls=true
email.useCredentials=true
email.username=mon_mail@dauphine.eu
email.password=mon_mdp

Toutefois cela n'a pas marché.

ANNEXE :

Git du projet : https://github.com/axelacu/Modelisation-M1