Gestion des circuits

Esup-signature permet de créer des circuits de signatures puissants dont les paramètres pour chaque l'étape sont :

- la liste des signataires,
- le type de signature,
- si tous les signataires doivent signer à une étape donnée.

Un utilisateur peut se construire son propre circuit à l'aide de l'assistant (voir : https://www.esup-portail.org/wiki/display/SIGN/Documentation+utilisateur#Documentationutilisateur-Assistantdecr%C3%A9ationd'unedemande)

Les administrateurs peuvent, eux, créer des circuits dont les paramètres sont plus poussés :

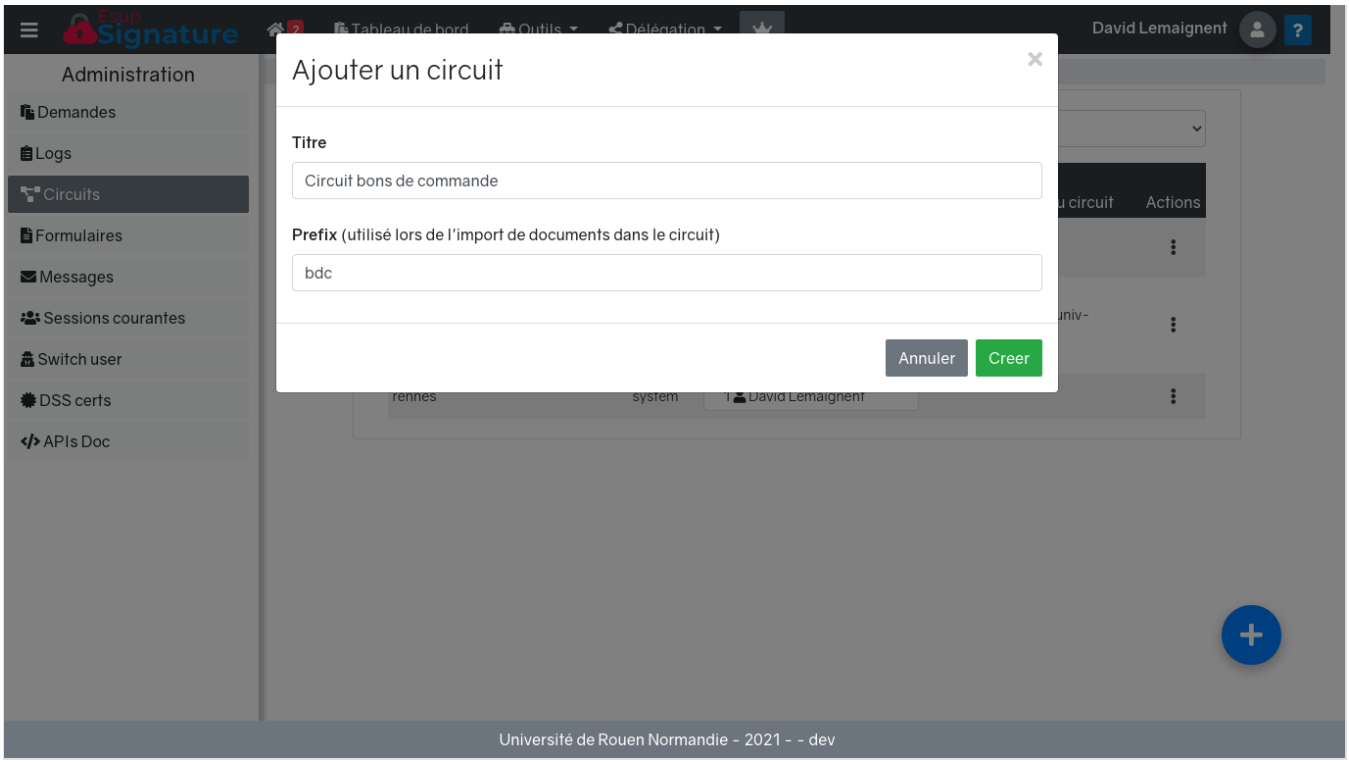
- Configuration des autorisations sur les circuits (qui peut démarrer un circuit) à l'aide des rôles (voir : Configuration de la sécurité)
- Configuration des entrées/sorties pour une récupération et/ou un dépôts automatique des documents
- Activation ou non de la fonction de scan des documents PDF (dans ce cas la description du circuit se trouve dans les métas-données du PDF)

Dans "Admin" puis "Circuit", vous avez accès à l'outil permet de consulter et de modifier les circuits de signatures. Il est possible de filtrer les circuits présents dans esup-signature :

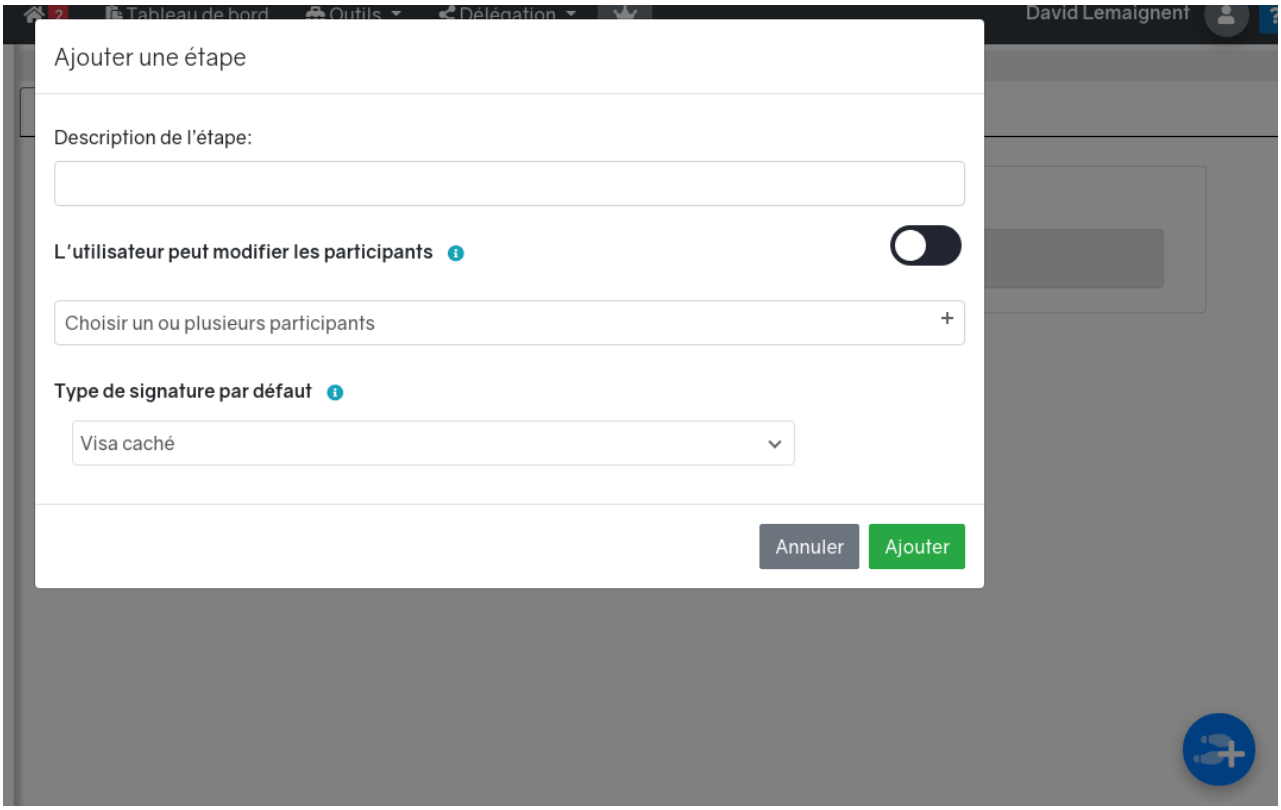
- Workflows globaux (définis pas les administrateurs)
- Classes workflow (définies par les développeurs)
- Workflows Utilisateurs (définis pas les utilisateurs)

Création d'un circuit classique

Pour ajouter un nouveau circuit via l'interface graphique, cliquez sur le bouton bleu "+" puis saisissez un nom et un préfixe (utilisé pour le nommage des documents lors des imports).



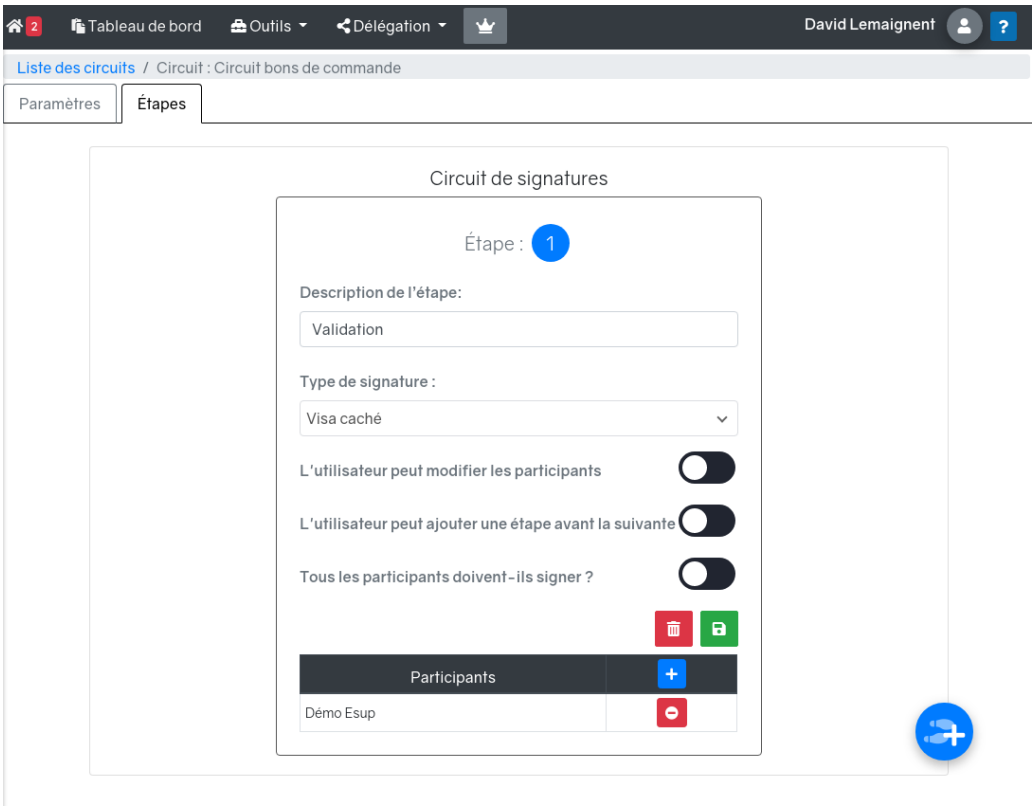
Vous serez redirigé vers la page permettant d'ajout des étapes pour ce circuit. Utilisez le bouton pour ajouter une étape :



Le formulaire d'ajout est simplifié. Il permet juste de configurer les paramètres suivants pour l'étape :

- la description
- laisser à l'utilisateur, qui démarre le circuit, la possibilité de modifier les participants de l'étape
- les participants (si plusieurs participants, la possibilité d'imposer toutes les signatures est proposée)
- le type de signature

Chaque étape ajoutée est représentée par un pavé reprenant les paramètres et donnant la possibilité des les modifier :



Chaque étape est indépendante. De ce fait, il faudra enregistrer vos modifications étape par étape.

De plus, attention à l'ordre des étapes, il n'est pas possible de le modifier. Il faudra supprimer puis recréer les étapes si besoin.

Il existe un utilisateur particulier nommé "**Créateur de la demande**". Lors de la création du circuit, il sera substitué par l'utilisateur courant. On retrouve cet utilisateur au niveau du module de recherche des utilisateurs en tapant **creator** ou **Créateur de la demande**

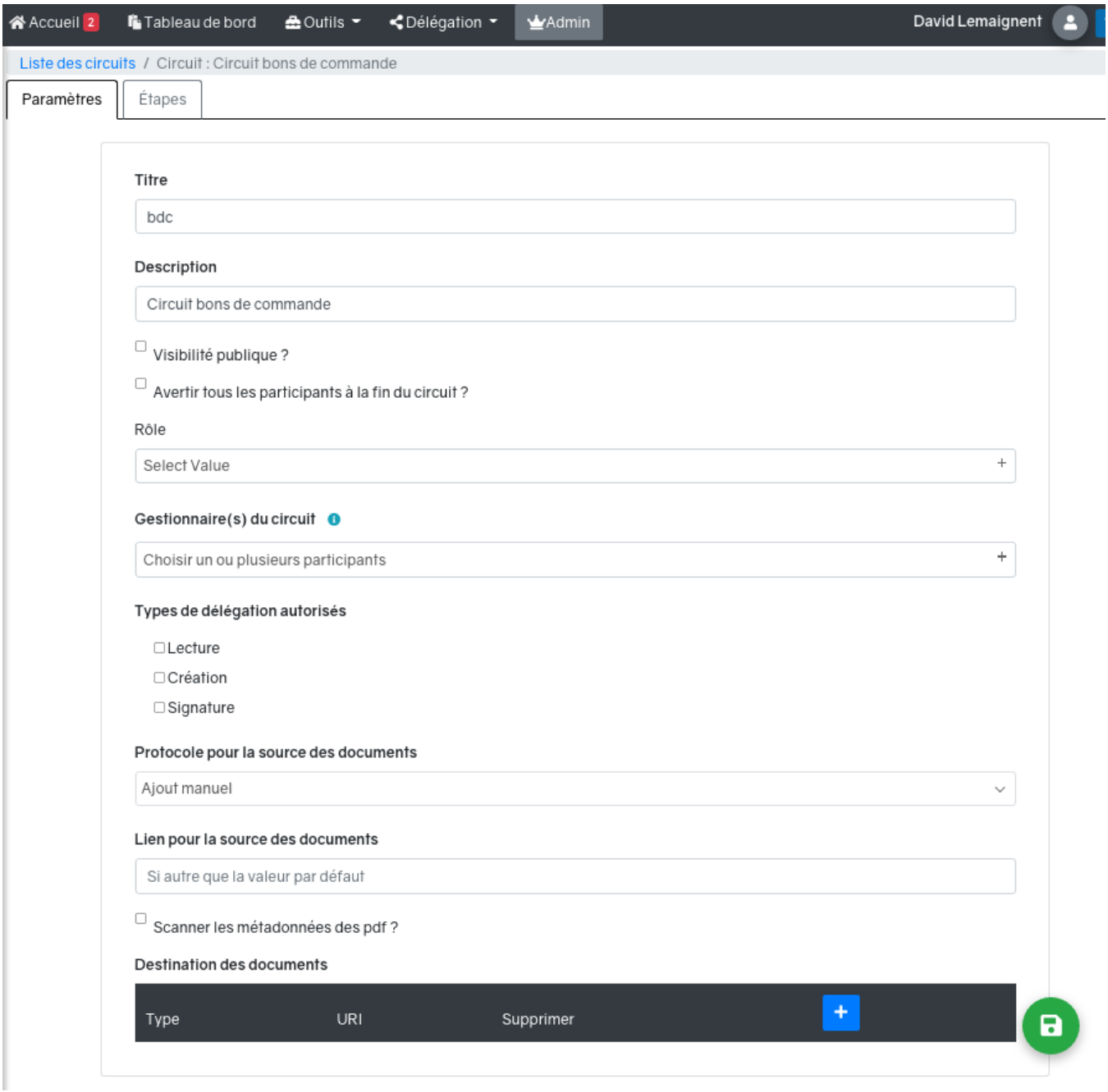
Dans cette vue il est possible d'activer la notion d'étape "infinie" en activant "L'utilisateur peut ajouter une étape avant la suivante". Si cette option est activée, vous donnerez la possibilité aux participants de cette étape, d'ajouter des étapes intermédiaires (non prévues à l'origine).

Chaque étape créée à la suite d'une étape "infinie" sera elle aussi une étape "infini". Cela se traduit, au niveau de l'interface utilisateur, par une question qui lui est posée lorsque qu'il s'apprête à signer : "Signer et passer à l'étape suivante" ou "signer et ajouter une étape"

Paramètres généraux d'un circuit

Dans l'onglet "Paramètre" de votre circuit, vous pourrez modifier sa configuration globale :

- Titre
- Description (celle qui apparaît sur le le bouton permettant de démarrer le circuit)
- Visibilité publique (tout le monde peut démarrer le circuit)
- Avertir tous les participants à la fin du circuit
- Les rôles autorisés à démarrer le circuit. Les rôles sont obtenus en fonction de la configuration voir Configuration de la sécurité. Ce paramètre est surchargé par la "Visibilité publique"
- Les gestionnaires du circuits ; ils peuvent accéder à toutes les demandes correspondant au circuit
- Type de délégations autorisées : Lecture, création et/ou signature. Verrouille les possibilités de déléguer, à d'autres, les actions sur les documents correspondant à ce circuit
- Protocole pour la source des documents parmi : smb (partage réseau, cmis (GES nuxeo, alfresco, ...), vfs (dossier local)
- Lien pour la source des documents
- Le paramètre "Scanner les métadonnées" (valable seulement pour les documents provenant d'une source de donnée)
- Une ou plusieurs destinations parmi : smb, cmis, vfs ou mail



Définir une source pour les documents

Comme vu précédemment, il est possible de définir un emplacement source pour alimenter un circuit. Esup signature possède une tache planifiée qui "scan" régulièrement toutes les sources de documents définies dans les différents circuits.

Pour cela il faut, en premier lieu, avoir défini un compte d'accès pour chaque types de source que l'on voudra utiliser. Esup-signature propose nativement les protocoles SMB, CMIS et VFS repris de l'application esup-filemanager (Esup File Manager).

La configuration globale se fait ici : Configuration#fs(filesystem).

Pour chaque circuit on peur saisir un lien vers lequel le compte configuré à un accès complet, par exemple : smb://<host>/bdc/a_signer par exemple.

Voici la liste des protocoles pris en charge par esup-signature ainsi que la manière de les configurer :

Type	Chemin	Adresse à saisir	Remarques

SMB	Absolu	smb://<adresse du serveur>/<chemin du dossier>	smb-test-uri à configurer obligatoirement pour activer la fonctionnalité. L'utilisateur est configuré dans application.yml (smb-login, smb-password)
CMIS	Relatif	cmis://<chemin du dossier> ex: cmis://default-domain/workspaces/test	Il faut configurer l'adresse du server nuxeo dans application.yml au niveau de (cmis-test-uri ex: http://mon-nuxeo.univ-ville.fr:8081/nuxeo, cmis-login, cmis-password) Cette partie est amenée à changer prochainement pour permettre les chemins absolus
FILE	Absolu	/<chemin du dossier>	vfs-test-uri à configurer obligatoirement pour activer la fonctionnalité.
FTP / SFTP	Absolu	ftp://<user>:<password>@<adresse du serveur>/<chemin du dossier>	vfs-test-uri à configurer obligatoirement pour activer la fonctionnalité.

⚠ Lorsqu'un document est intégré de cette façon, il est supprimé du dossier source.

Définir une destination pour les documents

Il est possible de définir un/des emplacement(s) de destination pour stocker les documents en fin de circuit.

La configuration globale doit être faite comme vu pour les sources de documents.

Pour chaque circuit on peut saisir un ou plusieurs lien vers lesquels les documents seront envoyés.

Voici la liste des protocoles pris en charge par esup-signature pour les destinations:

- smb, cmis, file, sftp / ftp, pour ces types, il faut suive les explications du tableau ci-dessus
- http:// https:// (fera une requete, type REST, vers l'url avec en paramètre l'ID de la demande de signature, son statut signé ou refusé et le numéro d'étape concerné
- mailto (envoi du document en pièce jointe au destinataire)

Voici un exemple de code à mettre coté application métier pour permettre à esup-signature de transmettre les informations du circuit via REST. Ici il s'agir d'un controleur Spring. La requête est envoyée en GET, esup-signature doit avoir un accès direct à cette API.

```
    @GetMapping(value = "/return-test")
    @ResponseBody
    public ResponseEntity<Void> returnTest(@RequestParam("signRequestId") String signRequestId, @RequestParam
("status") String status, @RequestParam("step") String step) {
        logger.info(signRequestId + ", " + status + ", " + step);
        //ici, le code à executer coté application métier en fonction du statut
        return ResponseEntity.ok().build();
    }
```

Définir les participants à un circuit à posteriori

Dans certains cas, les participants à une étape ne peuvent pas être prédéfinis dans esup-signature.

Dans le cas concret du circuit des bons de commande à l'université de Rouen, les signataires sont détermines en fonction de l'unité budgétaire (donc par l'application métier, SIFAC).

Le cas d'usage à Rouen est que les utilisateurs génèrent des bons de commande "à signer" au format PDF et les déposent dans leur dossier de travail.

Trois possibilités :

- Mettre en place un script qui va calculer le workflow, utiliser les web-services pour configurer les participants du circuit des bons de commandes et injecter le document. (pour plus de détails sur les web services voir : Web services REST)
- Mettre en place un script qui calcule le workflow, inscrit ce workflow dans les métas-données du document (PDF) et le copie dans un dossier défini comme "source" au niveau d'Esup-signature.
  Pour cette dernière solution il faut donc créer un circuit comme vu précédemment, cocher la case "Scanner les métadonnées des PDF" et définir une source pour la récupération des documents au niveau des paramètres généraux
- Implémenter une classe worflow (voir le chapitre suivant)

Les métas-données qui doivent être inscrite dans les document PDF sont les suivantes:

- sign_type_default_val : contenant le type de signature (visa, pdfImageStamp, certSign ou nexuSign)

- sign_step#<n> : contenant la liste des participants de l'étape n
- sign_target : contenant le chemin de dépôt des documents après signature

Lorsque le scheduler passera pour importer les documents, ceux-ci seront analysés, le circuit sera généré en fonction des informations trouvées dans les métas-données.

Exemple de code java permettant d'ajouter les métas-données à un fichier pdf :

```
PDDocument document = PDDocument.load(in);
PDDocumentInformation info = document.getDocumentInformation();
info.setCustomMetadataValue("sign_type_default_val", "pdfImageStamp");
info.setCustomMetadataValue("sign_step#1", "[machin@univ-ville.fr, truc@univ-ville.fr]");
info.setCustomMetadataValue("sign_target_key","smb://serveur_de_fichiers/la_destination/signed");
```

Utiliser les classes workflow

Pour les cas les plus spécifiques, il est possible d'ajouter, au code source original d'esup-signature, une classe qui décrira précisément un circuit.

> L'intérêt de cette solution est de pouvoir mettre en place des mécanismes complexes de calcul des participants. On peut par exemple imaginer de calculer le n+1 de l'utilisateur courant. Cela nécessite d'avoir des compétences en développement et connaitre notamment le langage Java et le framework Spring

Une nouvelle classe workflow devra implémenter la classe "DefaultWorkflow". Des exemples sont déjà présents dans le code source original d'Esup-signature dans le dossier src/main/java/org/esupportail/esupsignature/service/workflow/impl/

Votre classe doit être construite comme suit :

```
package org.esupportail.esupsignature.service.workflow.impl;

import org.esupportail.esupsignature.entity.Data;
import org.esupportail.esupsignature.entity.User;
import org.esupportail.esupsignature.entity.WorkflowStep;
import org.esupportail.esupsignature.entity.enums.SignType;
import org.esupportail.esupsignature.exception.EsupSignatureUserException;
import org.esupportail.esupsignature.service.workflow.DefaultWorkflow;
import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.List;

@Component
public class BasicWorkflow extends DefaultWorkflow {

   private String name = "BasicWorkflow";
   private String description = "Une signature";
   private List<WorkflowStep> workflowSteps;

   @Override
   public String getName() {
      return name;
   }

   @Override
   public String getDescription() {
      return description;
   }

   @Override
   public List<WorkflowStep> getWorkflowSteps() {
      if(this.workflowSteps == null) {
         try {
            this.workflowSteps = generateWorkflowSteps(userService.getCurrentUser(), null, null);
         } catch (EsupSignatureUserException e) {
            return null;
         }
      }
      return this.workflowSteps;
   }

   public void initWorkflowSteps() {
      this.workflowSteps = new ArrayList<>();
   }

   @Override
   public List<WorkflowStep> generateWorkflowSteps(User user, Data data, List<String> recipentEmailsStep)
throws EsupSignatureUserException {
      List<WorkflowStep> workflowSteps = new ArrayList<>();
/* ici on construit la liste des étapes du circuit */
      WorkflowStep workflowStep = new WorkflowStep();
      workflowStep.setStepNumber(1);
      workflowStep.setSignType(SignType.pdfImageStamp);
      workflowStep.setDescription("Choix du signataire");
      workflowStep.setChangeable(true);
      if(data != null) {
         workflowStep.setRecipients(workflowService.getFavoriteRecipientEmail(1, data.getForm(),
recipentEmailsStep, user));
      } else {
         workflowStep.getRecipients().add(recipientService.createRecipient(null, userService.getGenericUser
("Utilisateur issue des favoris", "")));
      }
      workflowSteps.add(workflowStep);
      return workflowSteps;
   }
}
```

Il faut donc a minima :

- Préciser un nom et une description (dans name et description)
- Implémenter la fonction generateWorkflowSteps() qui retournera une liste de WorkFlowStep (les étapes calculée en fonction de l'utilisateur courant "user")