**Chapitre 3: Conception**

**Introduction**Après avoir déjà défini les besoins fonctionnels et non fonctionnels et après détailler les différents cas d’utilisations du projet, au cours du chapitre précédent, il est temps maintenant de faire la phase de modélisation. Cette étape représente une des plus importantes phases dans un projet informatique puisqu’elle définit l’architecture du logiciel et permet d’analyser les besoins fonctionnels afin de les traduire à des diagrammes.

Dans ce chapitre nous commençant par exposer l'architecture utilisée en définissant les différentes couches et composantes ainsi que l’architecture de flux de travail et les patrons de conception utilisé puis nous passons à décrire et exposer les diagrammes les plus utiles pour une description rigoureuse et complète pour notre solution.

1. **Conception architecturale**
   1. **Architecture des flux de travail**
   2. **Courrier arrivé**

Pour notre premier flux de travail qui est le courrier arrivé :

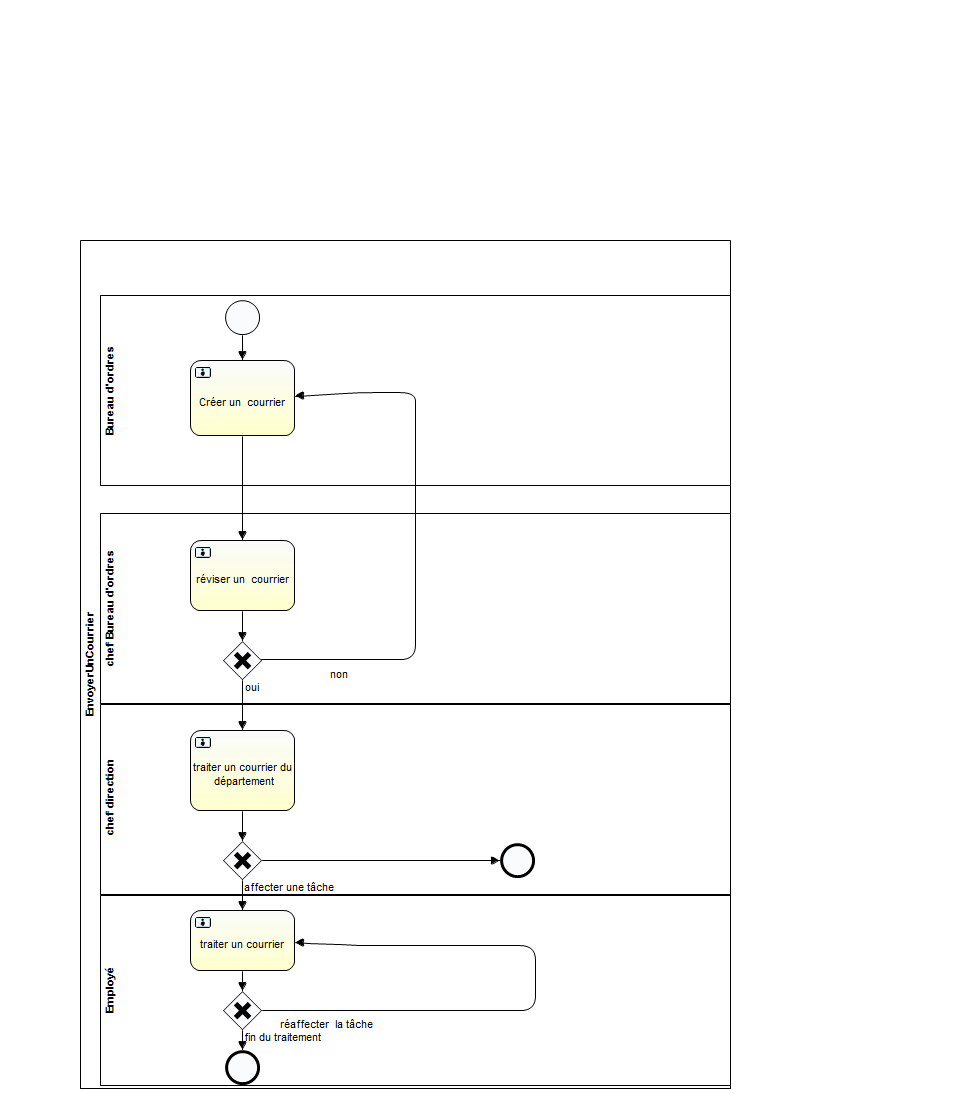
Un document (facture, demande, …) sera reçu par le bureau d’ordre1, un agent du bureau d’ordre va donc créer un courrier arrivé avec toute les informations nécessaire en attachant le(s)les document(s) comme étant une/des pièce(s) jointe(s) du courrier.

Après sa création le courrier sera donc révisé par le chef du bureau d’ordre qui a deux choix :

* Refuser le courrier : dans ce cas le courrier sera renvoyé vers les agents du bureau d’ordre pour le modifier ou le supprimer.
* Valider le courrier : dans ce cas le scénario normal se poursuit.

Le courrier sera envoyé vers le chef de la direction où le courrier est destiné qui va traiter ce courrier, il a le choix donc parmi deux situations :

* Soit archiver le courrier et dans ce cas le flux se termine
* Soit transférer le courrier vers un employé de la direction qui va effectuer d’autre traitements et ce transfère va boucler jusqu’à ce que un employé archive le courrier et met fin à ce flux.



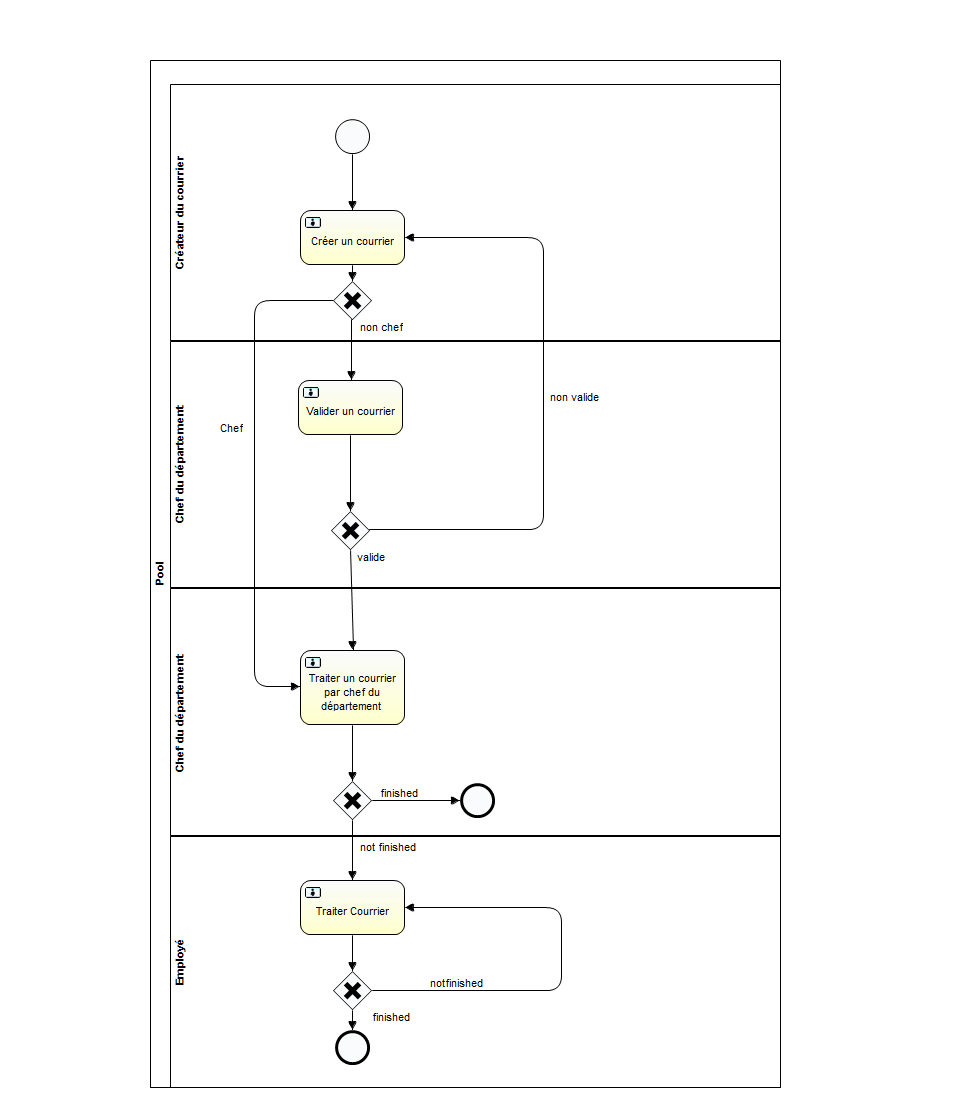
**Figure 13 :** *flux de travail courrier arrivé*

* 1. **Courrier interne**

Notre deuxième flux de travail est un courrier interne :

Ce courrier peut être généralement (demande de congé, demande d’avance, demande de sortie, …). Ce courrier donc peut être créer donc par n’importe quel employé dans la société, à la création il y a vérification du statut du créateur :

* S’il est un chef de département ou bureau d’ordre il va être envoyé directement vers le chef la direction destinatrice pour être traiter.
* Sinon le courrier doit être révisé par le chef de la direction source avant d’être traité par le chef de la direction destinatrice.

Le chef de la direction destinatrice a donc le choix entre archiver le courrier et finir le flux ou bien transférer le document vers l’un des employés de la direction et ainsi de suite.

**Figure 13 :** *Flux de travail courrier interne*

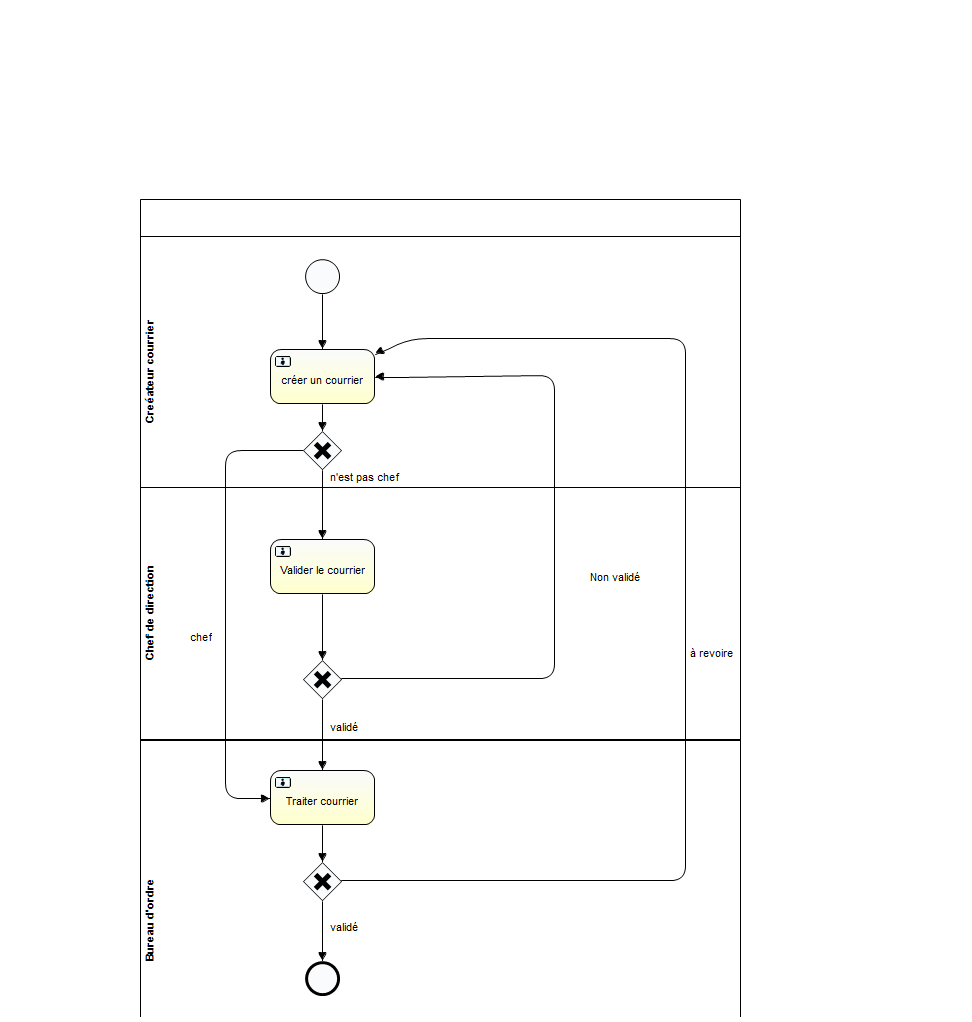
* 1. **Courrier sortant**

Notre troisième flux de travail est un courrier sortant :

Ce courrier peut être généralement (paiement de facture de congé, demande d’informations sur un produit, …). Ce courrier donc peut être créer donc par n’importe quel employé dans la société, à la création il y a vérification du statut du créateur :

* S’il est un chef de département il va être envoyé directement vers le chef du bureau d’ordre pour être traiter.
* Sinon le courrier doit être révisé par le chef de la direction source avant d’être traité par le chef du bureau d’ordre.

Le chef du bureau d’ordre a donc le choix entre archiver le courrier et finir le flux ou bien transférer le document vers l’un des employés de la direction et ainsi de suite.



**Figure 13 :** *flux de travail courrier sortant*

* 1. **Choix des patrons de conceptions**

Les patrons sont des solutions éprouvées à des problèmes spécifiques et récurrents.

Ils utilisent les bonnes pratiques de conception pour définir des solutions standard à différents niveaux d’abstraction.

L’aspect générique de ces solutions permet aux concepteurs de les considérer comme des micro-architectures qui a comme but la réduction de la complexité, assurer la réutilisation du code ainsi que fournir un vocabulaire commun.

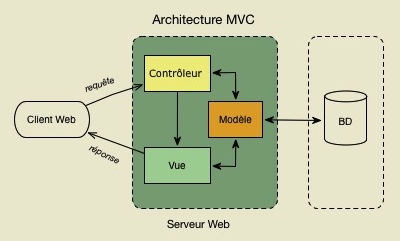
* 1. **Intérêt des patrons de conception**

L’intérêt des patrons de conception réside dans les points suivants :

* Construire des systèmes plus extensibles, plus robustes au changement
* Capitaliser l’expérience collective des informaticiens
* Réutiliser les solutions qui ont fait leur preuve
* Identifier les avantages / inconvénients / limites de ces solutions
  1. **Le patron, de conception MVC (http://prof.bpesquet.fr/cours/modele-mvc/)**

Le modèle MVC1 est une conception logicielle qui sépare l’application en 3 différentes parties :

* Le modèle (les données)
* La vue (la présentation)
* Le Contrôleur (le traitement)



**Figure 13 :** *Le patron de conception MVC*

* + 1. **Le modèle**

Le modèle est la partie qui représente les données de l’application donc encapsule la logique métier ainsi que l’accès aux données.

* + 1. **La vue**

Cette partie s’occupe des interactions avec l’utilisateur. Elle n’effectue aucun traitement et elle est chargée simplement d’afficher les données que lui fournit le modèle et recevoir tous les actions de l’utilisateur.

* + 1. **Le contrôleur**

Le contrôleur gère la dynamique de l’application, il prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser.

* + 1. **Intérêt du patron MVC**

Le bénéfice principal de cette architecture est d’avoir une dépendance minimale entre les différentes couches de l'application ce qui permet de diviser l’équipe de développement selon leur spécialité (intégrateur, développeur, …) et créer des applications plus modulaires et maintenable. De ce fait, ce modèle offre une multitude d’avantages, dont nous citons

quelques-unes :

* La fiabilité
* La productivité
* Les coûts réduits de développement.
  1. **Le patron DAO**

Le pattern DAO2 est une technique utilisé pourisoler la couche de persistance de la reste de l’application, il permet essentiellement le découplage entre la couche métier et la couche de persistance.

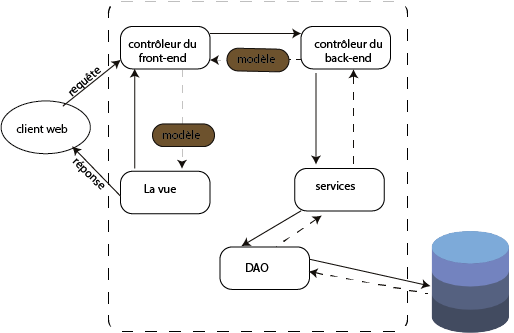
* + 1. **Les avantages du modèle DAO**

Plus facile de modifier le code qui gère la persistance (changement de SGBD ou même de modèle de données)

Plus facile pour les accès.

Sans doute le modèle de conception le plus utilisé dans le monde de la persistance.

* + 1. **Architecture MVC+DAO de l’application**



***Figure****: Architecture MVC + DAO de l’application*

1. **Conception détaillée**

Dans cette partie nous allons mieux détailler l’architecture de la solution proposé en décrivant en premier temps l’outil de conception utilisé dans la modélisation.

Dans un second temps nous allons élaborer les différentes conceptions des bases de données utilisées.

Dans un dernier temps on va décrire les interactions entre les acteurs et le système avec le diagramme de cas d’utilisation et la structure des entités avec le diagramme de classe.

1. **L’outil de conception**

Une des AGL (Atelier Génie Logiciel) les plus utilisées qui nous permet de tracer plusieurs types de diagrammes c’est le logiciel de conception PowerAMC.

**POWERAMC**est l'un des**outils majeurs de modélisation des données et des processus** qui permet d’élaborer des modèles de données que cela soit merise1, UML ou autre de manière, graphique et de les implémenter quel que soit le SGBD et ce de manière automatique.



***Figure****: Logo PowerAMC*

1. **Conception de la base de données**

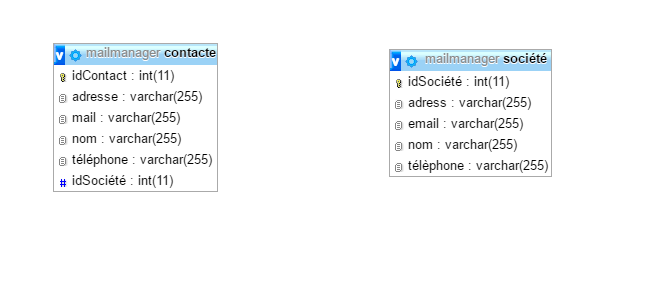
On a choisi la base de données MySQL comme SGBD2 pour stocker les données.

Dans notre application on a disposé essentiellement de deux bases de données :

* Une base de données pour stocker les informations propriétaire à activiti3, la base de données H2 est intégrée par défaut à l’outil activiti mais pour des raisons de persistance de données, il est possible de modifier cette base h2 par une base MySQL par exemple.Cette base est générer automatiquement par le moteur de workfow4 activiti.
* Une base de données pour stocker les informations relatives aux contacts et sociétés avec lesquelles on va échanger les courriers.

Les tableaux qui constituent cette base sont :

* Société : une table qui contient tous les sociétés
* Contact : une table qui contient tous contacts de ces sociétés.

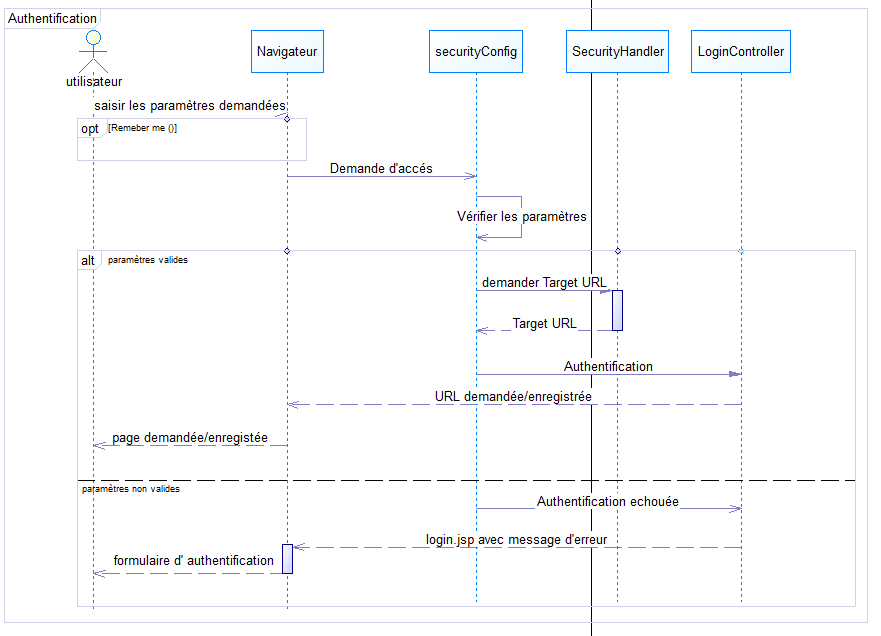


1. **Diagrammes de séquences**

Ces diagrammes indiquent les objets que l’acteur va manipuler et les opérations invoquées sur les différentes classes. Les diagrammes de séquences nous expliquent l’enchainement de l’exécution des différentes étapes pendant l’utilisation de notre application.

* 1. **Authentification**

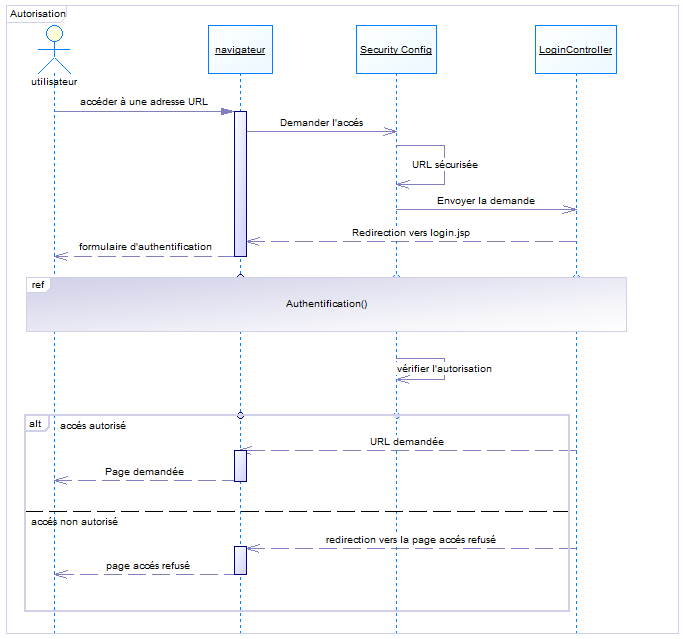
La figure 16 illustre le digramme de séquence « Authentification ». L'utilisateur saisit les paramètres de connexion (la fonctionnalité *Remember-me* est optionnelle). Spring Security vérifie la validité des paramètres saisis par l'utilisateur et refuse l’accès s’ils sont invalides. Sinon une redirection est effectuée vers une page déjà définie dans la classe *SecurityHandler* (TargetURL) ou une URL déjà demandée avant la connexion.

 **Figure 16 :** *Diagramme de séquence « Authentification »*

* 1. **Autorisation**

La figure ci-dessous illustre le digramme de séquence « Autorisation ». L'utilisateur accède à une URL, le contrôle sera redirigé vers la classe *SpringConfig* qui constate que les toutes les URL sont sécurisées. La demande sera transmise à LoginController, qui est un contrôleur Spring MVC. Ce dernier assure la redirection vers la page d’authentification ‘*Login.jsp’*.

Le processus d’authentification représenté ci-dessus se déclenche. Une fois l’utilisateur veut accéder à une page dont l’accès n’est pas autorisé, une page 403.jsp sera affichée.

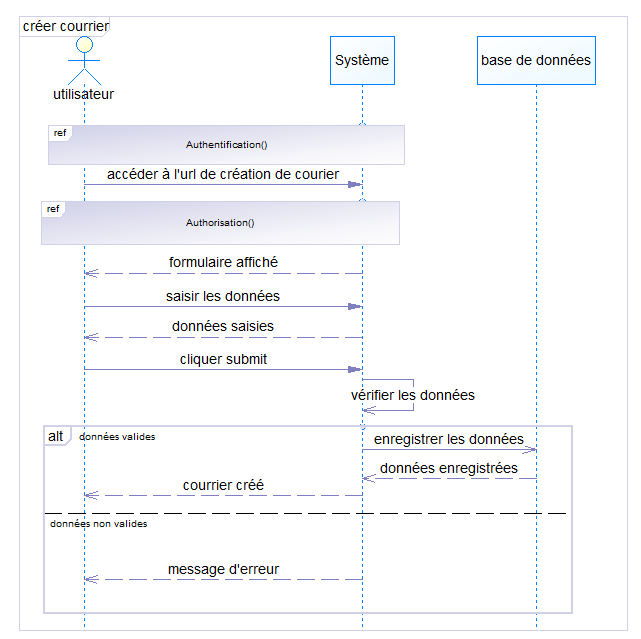


**Figure 17 :** *Diagramme de séquence « Autorisation»*

* 1. **Créer un courrier**

La figure ci-dessous est le diagramme de séquences de création d’un courrier .

Pour créer un courrier l’utilisateur doit s’authentifier, accéder à l’URL de création de courrier s’il est autorisé puis saisie les données demandées dans le formulaire et valide les informations saisies.

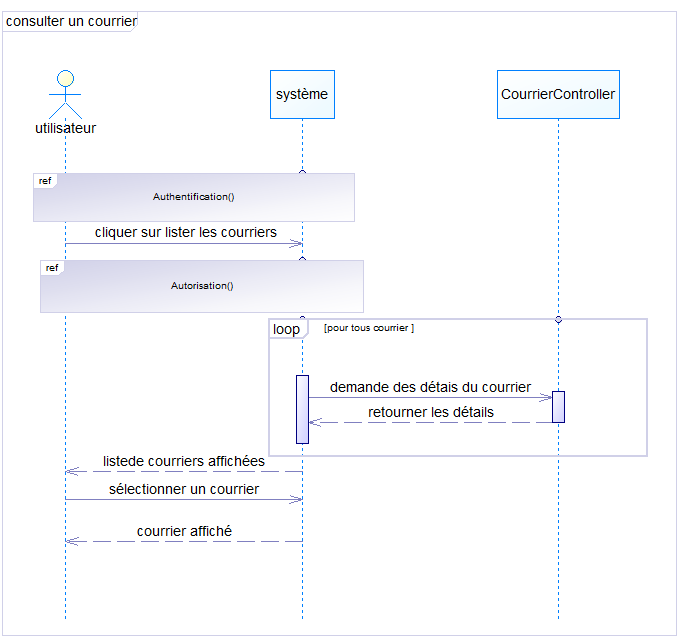


**Figure 18 :** *Diagramme de séquence «créer un courrier»*

* 1. **Consulter les courriers**

Le diagramme de la figure ?? décrit l’opération de consultation d’un courrier.

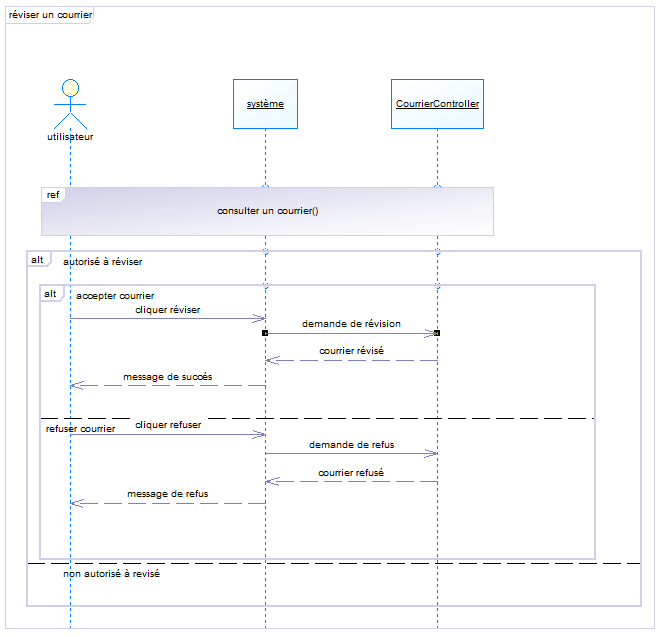
Après l’authentification, l’utilisateur doit cliquer sur lister tous les courriers puis sélectionner le courrier pour qu’il consulte les détails du courrier.



**Figure :** *Diagramme de séquence «Consulter un courrier»*

* 1. **Réviser un courrier**

Comme c’est illustré par la figure ??, pour réviser un courrier l’utilisateur doit

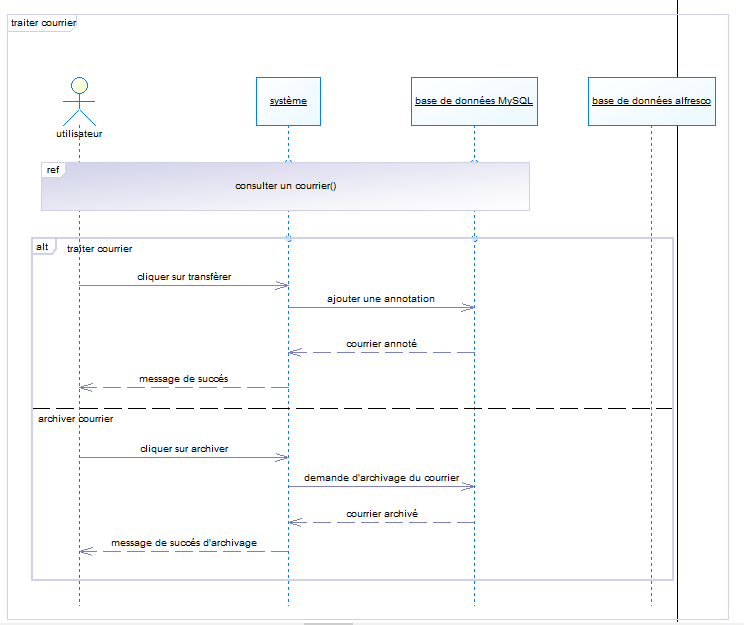
consulter ce courrier puis s’il est autorisé à réviser ce courrier alors il a le choix entre valider ou refuser ce courrier.

**Figure :** *Diagramme de séquence «Valider un courrier»*

* 1. **Traiter un courrier**

Comme c’est illustré par la figure ??, pour traiter un courrier l’utilisateur doit

consulter ce courrier puis s’il est autorisé à traiter ce courrier alors il a le choix entre transférer ou archiver ce courrier.



**Figure :** *Diagramme de séquence «traiter un courrier»*

1. **Diagramme de classe**

Le diagramme de classes est le point central dans un développement orienté objet. En

analyse, il a pour objectif de décrire la structure des entités manipulées par les utilisateurs. En

conception le diagramme de classes représente la structure d’un code orienté objet ou, à un

niveau de détail plus important, les modules du langage de développement.

**///à compléter**

1. **Apache directory studio** **(http://www.active-directory.info/apache-directory-un-projet-libre-de-la-fondation-apache/)**

Apache Directory est un projet libre conçu par la fondation Apache. Ce serveur d’annuaire LDAP fonctionne sur toute plate-forme.

Notons qu’il existe un sous-projet basé sur Eclipse. Ce programme permet de gérer un annuaire intitulé Apache Directory Studio. Ce serveur possède un navigateur d’éditeur LDAP1, un utilitaire LDIF et DSML, un navigateur de schéma ainsi que d’autres fonctionnalités.

On a utilisé la dernière version de son principal composant Apache Directory Server est actuellement le [2.0.0-M12](http://directory.apache.org/studio/downloads.html)

****

**Figure 13:** *Logo Apache Directory Studio*

1. **Maven(https://www.jmdoudoux.fr/java/dej/chap-maven.htm)**

Maven est un outil de construction de projets open source développé par la fondation Apache. Il permet de faciliter et d’automatiser certaines tâches de la gestion d'un projet Java.  
On a utilisé la dernière version 3.3.9

**Figure 13:** *Logo maven*

1. **Bootstrap**

****Notre application consiste à utiliser le Framework Twitter Bootstrap qui est une collection d’outils utiles à la création des sites web et des applications web. C’est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option.

**Figure 13:** *Logo Bootstrap*

1. **JSTL**

JSTL4 est un composant de la plate-forme JavaEE qui décrit plusieurs actions basiques pour les applications web. Cet ensemble standard d'actions personnalisées (Tags personnalisés) propose des actions souvent rencontrées dans les JSP.

Elle propose de développer une page JSP sans y utiliser de code [Java](https://fr.wikipedia.org/wiki/Java_(langage)) directement.

1. **CSS3(http://www.futura-sciences.com/tech/definitions/internet-css-4050/)**

Les feuilles de styles CSS sont un langage qui permet de gérer la présentation d'une page Web. Le langage CSS est une recommandation du World Wide Web Consortium (W3C), au même titre que HTML ou XML.

Le but de CSS est de séparer la structure d'un document HTML et sa présentation.



**Figure 13:** *Logo CSS3*

1. **HTML5**

HTML5 est la dernière révision majeure de HTML.

Le [HTML](http://dico.developpez.com/html/1696-Langages-HTML-HyperText-Markup-Language.php) (HyperText Markup Language) est un format de présentation de données permettant de créer des [pages web](http://dico.developpez.com/html/218-Internet-page-web.php) pouvant être lues dans des [navigateurs](http://dico.developpez.com/html/224-Internet-navigateur.php).

Il s'agit d'un langage [côté client](http://www.chiny.me/glossaire.php#cote-client) (tout comme [CSS](http://www.chiny.me/css-cascading-style-sheets-5.php) et [Javascript](http://www.chiny.me/javascript-pour-rendre-vos-pages-interactives-6.php)). Il est supporté et développé par [W3C](http://www.chiny.me/glossaire.php#w3c).

**Figure 13:** *Logo HTML5*

1. **Javascript**

Javascript est principalement un langage de script coté client, c'est à dire qu'il est exécuté sur le navigateur.

C'est un **langage orienté objet qui permet** d'exécuter des actions en fonction d'événements déclenchés par l'utilisateur.

1. **JQuery**

**JQuery** est une bibliothèque logicielle, rapide et concise conçue pour les programmeurs, JavaScriptopen-source inventée par John Resig en 2006, elle permet de se simplifier la vie à moindre coût pour tout ce qui est manipulation d'une page HTML.

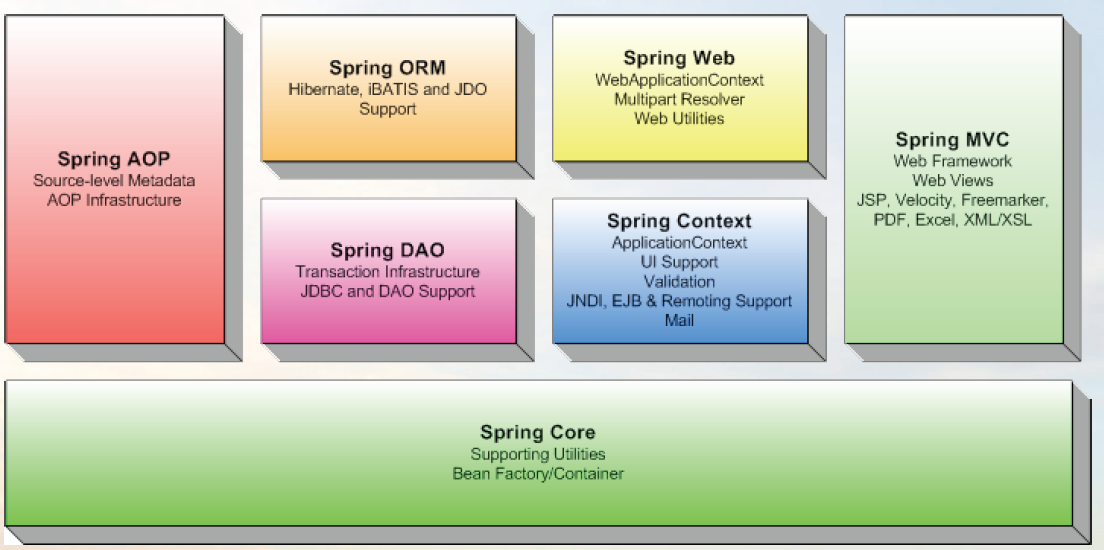
**Figure 13:** *Logo HTML5*

1. **Framework pour la couche service : Spring(https://www.projet-plume.org/fiche/spring)**

Spring est un Framework créé et supporté par l'entreprise SpringSource (qui fait maintenant partie de VMware). Il permet de simplifier le développement d'applications Java et est devenu un standard dans l'industrie du développement logiciel basé sur la plate-forme Java, surtout dans le développement Java EE.

Spring est ainsi un des Frameworks les plus répandus dans le monde Java : sa popularité a grandi au profit de la complexité de Java EE notamment pour ses versions antérieures à la version 5, mais aussi grâce à la qualité et la richesse des fonctionnalités qu'il propose.

Il propose de nombreuses fonctionnalités de base pour le développement d'applications :

* La notion d'IoC (Inversion of Control) qui permet de minimiser l'adhérence du code au framework (le fameux "Don't call us, we will call you").
* L'injection de dépendance (Dependency Injection) qui offre une grande souplesse dans l'architecture en limitant les dépendances codées en dur, et donc idéal pour élaborer facilement toute sorte de tests unitaires.
* Faciliter le développement des DAO de la couche de persistance en utilisant JDBC,JPA ou une solution open source comme Hibernate.
*  Faciliter la testabilité de l'application.

**Figure 33:** *Architecture de Spring*

1. **Framework pour la couche sécurité : Spring Security**

[Spring Security](http://projects.spring.io/spring-security/) a été créé en 2003 et fait partie de l’écosystème [Spring](https://spring.io/). Il fournit des services permettant de sécuriser des applications server JavaEE. Il permet de gérer l'authentification et d'autorisation des utilisateurs dans l'application.

* ***Principe de Spring Security***

Les demandes d’accès sont interceptées par un filtre de servlet qui délègue à un Bean

Spring Security les traitements de vérification d'accès aux pages web. Ce Bean met en œuvre une chaîne de filtres. Chacun des filtres est un Bean auquel est attribuée une tâche précise :

* Intégration dans la session HTTP des informations de sécurité contenues dans la requête (Login, mot de passe, …).
* Vérification de l'identité de l'appelant et affichage d'une invite de connexion si nécessaire.
* Vérification des droits d'accès à la ressource sollicitée.

1. **Framework pour la couche présentation : Spring MVC**

Le framework Spring Web MVC fournit l'architecture Modèle-Vue-Contrôleur (MVC) et les composants prêts à être utilisés pour développer des applications Web flexibles et peu couplées.

Le module Web de Spring MVC fournit une richesse de fonctionnalités, y compris :

* Configuration simple et intégration native avec Spring Framework, tirant parti des fonctionnalités puissantes de Spring et d'autres bibliothèques open source
* Construit sur des technologies Web Java telles que Servlets, JSP et JSTL et peut être déployé dans n'importe quel conteneur Servlet standard, tel que Tomcat
* Implémentée en fonction du modèle d’architecture MVC avec une séparation claire des préoccupations en utilisant des annotations simples et des balises XML d'espace de noms.

1. ***Service web : REST***

"Représentationnel **S**tate **T**ransfer" (REST) n'est pas un protocole, un standard ou encore un format. Il s'agit d'un style d'architecture pour les systèmes distribués.

Un des concepts importants de REST est la notion de ressource. Chaque ressource est accessible par une URI (Uniform Resource Identifier). La ressource étant une notion abstraite, le client et le serveur communiquent en s'échangeant une représentation de cette ressource. Le format de cette représentation peut être XML, JSON...

Les ressources sont accessibles via un ensemble uniforme de commandes fournies par HTTP (essentiellement GET, POST, PUT et DELETE) qui permettent de spécifier l'opération à effectuer sur une ressource [17]

**2. Architecture logique :**

Dans cette section, nous présentons l’architecture n-tiers de notre système.

Architecture JEE n-tiers : le système proposé repose sur une architecture n-tiers. C’est un modèle logique d’architecture applicative qui vise à séparer nettement  n couches logicielles au sein d’un même système (généralement 3 ou 4 couches). Elle sert à modéliser et à présenter ce système comme un empilement de "n" couches dont le rôle de chacune est clairement défini comme suit :

* **Couche Services**

C’est la couche métier de notre application, c’est la couche qui gère la logique de l'application et les traitements à effectuer sur les données. Cette couche permet de séparer clairement les opérations effectuées par le contrôleur et celles qui concernent le modèle des données (la logique business de l'application). Toute action devrait donc passer par cette couche.

Les implémentations des couches services et DAO sont implémentées via des interfaces Java, ce qui assure l’indépendance des couches entre elles.

* **Couche de sécurité**

C’est une configuration qui permet de gérer l’accès aux différentes pages de l’application.

Dans notre application, avec l’utilisation du framework Spring Security, on va gérer principalement deux aspects :

* L’authentification : un mécanisme de login /mot de passe qui permet à l’utilisateur de manipuler ses informations, ses droits, éventuellement ses groupes, etc. Cette authentification se fait à partir d'informations stockées dans un annuaire LDAP.
* L’autorisation : il s’agit de limiter ou restreindre l’accès à une URL aux utilisateurs d’une certaine catégorie ou groupes disposant de certains droits
* **Couche de présentation**

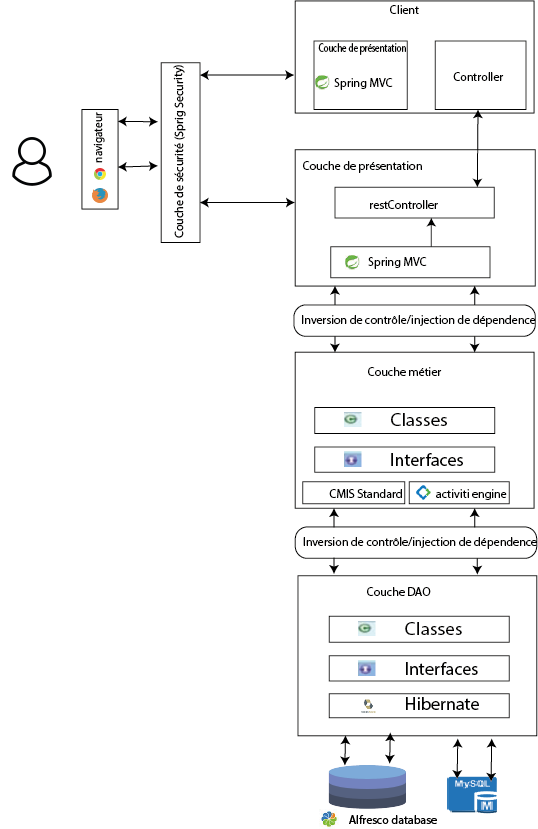
Elle correspond à la partie de l'application visible et interactive avec les utilisateurs.

Nous parlons d'Interface Homme Machine.

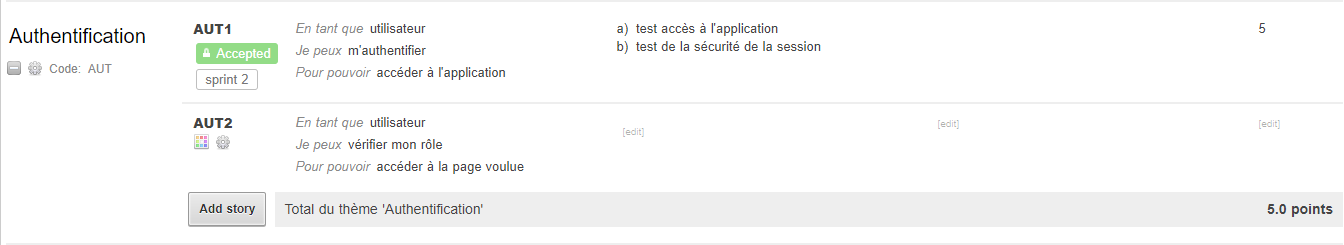
L’utilisateur va interagir avec les vues qui lui sont présentées dans le navigateur. Ses actions vont parfois nécessiter l'interrogation du serveur Spring. Celui-ci traitera la demande et rendra une réponse JSON (JavaScript Object Notation). Celle-ci sera utilisée pour mettre à jour la vue présentée à l'utilisateur.

* **Couche de persistance**

Cette couche est gérée par le patron de conception DAO. Elle utilise le framework Hibernate pour la recherche de données dans une base de données en réalisant lui-même la création des objets et les traitements de remplissage de ceux-ci en accédant à la base de données.

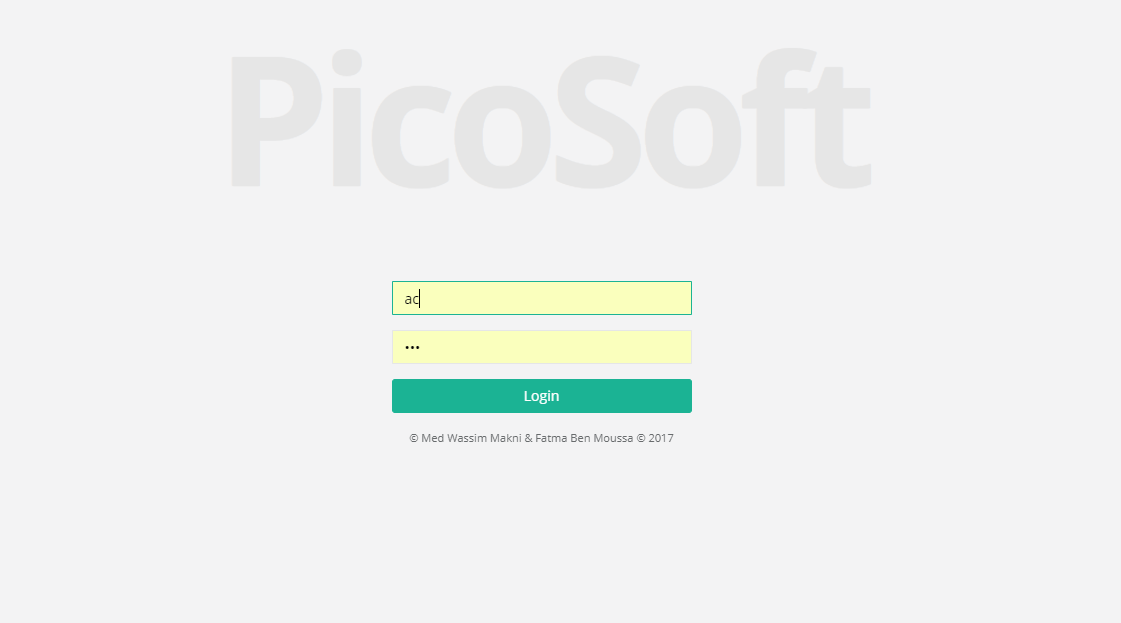


* 1. ***Authentification et contrôle d’accès***

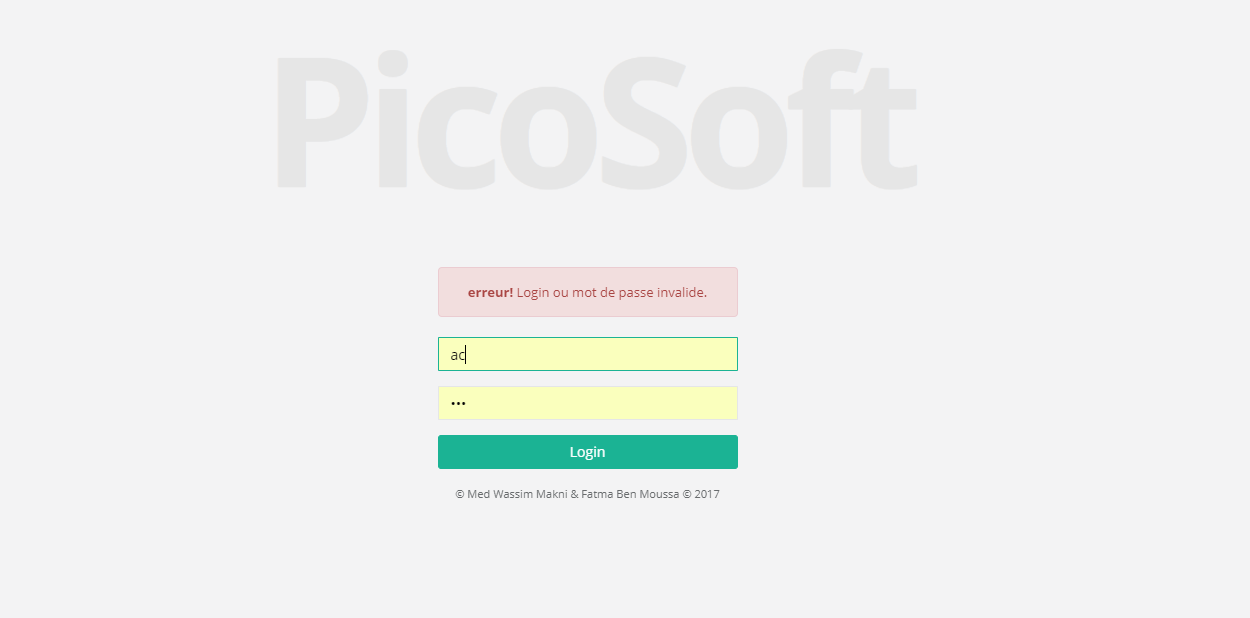


Le processus d’authentification et de contrôle d’accès est un processus majeur dans notre application puisqu’en fonction du rôle de la personne authentifiée, chaque page a ses spécifications et ses données à afficher. Donc à chaque fois que l’utilisateur ouvre une page, le système de sécurité vérifie qu’il est authentifié et vérifie son rôle (la direction où il appartient) dans l’annuaire LDAP.

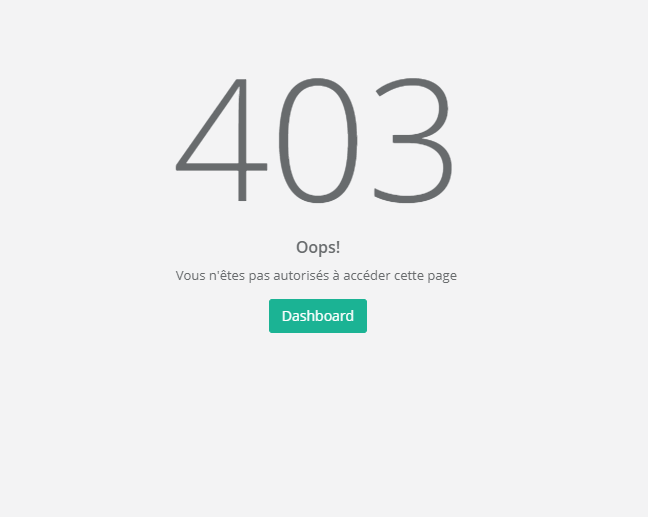
La figure 38 représente l'interface de l'authentification où l'utilisateur doit saisir son login et mot de passe puis il doit valider en cliquant sur le bouton « Se connecter ».



Ainsi le système va vérifier la correspondance des données avec celles dans la base. En cas de réussite, l’utilisateur sera redirigé vers son interface d’accueil sinon un message d'erreur s'affichera comme l’indique la figure 39.



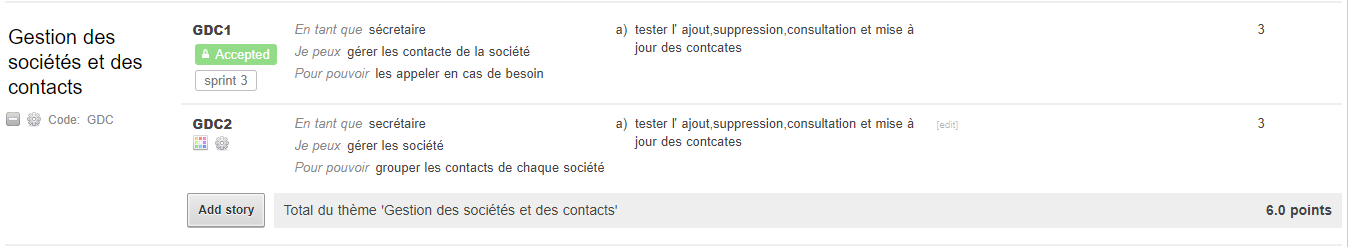
Si l’utilisateur n’a pas le droit d’accéder à une page, dans ce cas, une page 403 sera affichée. (Figure 42). La page 403 représente le code de réponse du protocole http et signifie que le serveur a compris la requête du client, mais refuse de l’exécuter.



3. Release 3

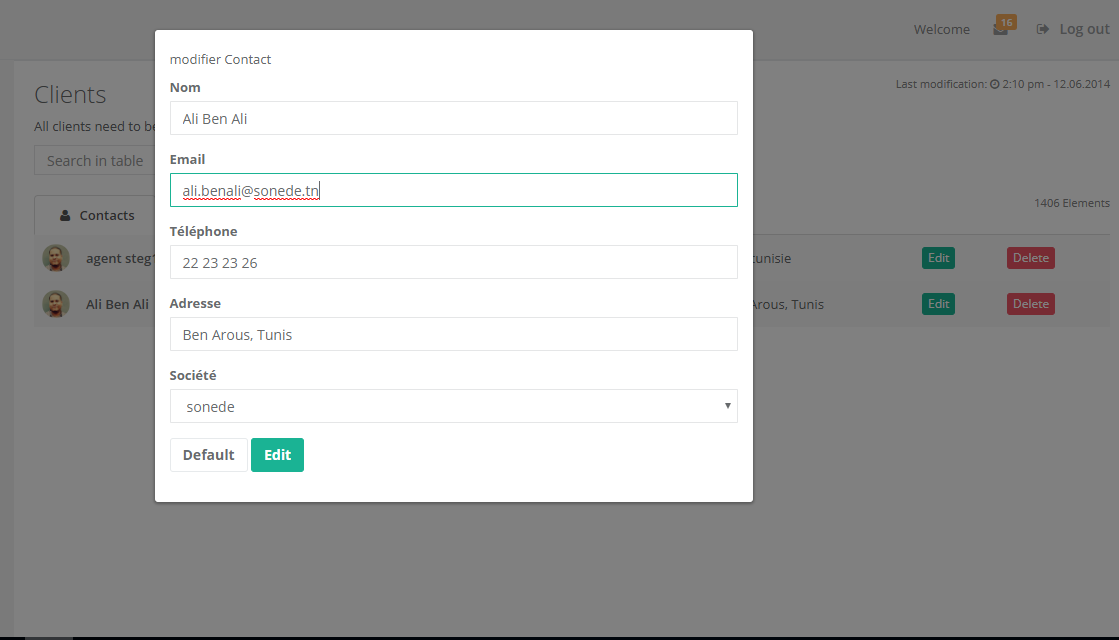
3.1 Sprint 1 : Gestion des contacts et des sociétés

Le tableau 16 suivant illustre le Backlog de l’administrateur pour le premier sprint du troisième « release ».



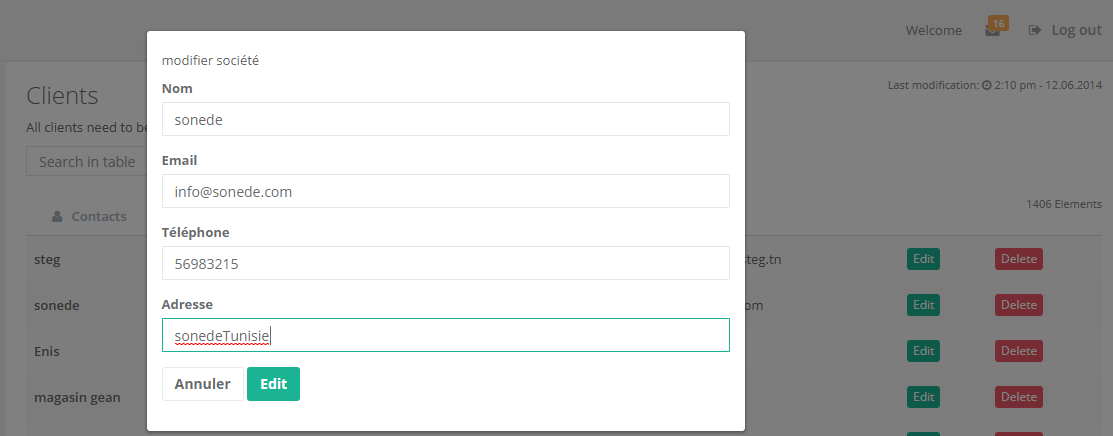
La figure 44 décrit l’interface de consultation des contacts et des sociétés. Un tableau contient à la fois la liste des contacts et sociétés est affiché ainsi que les actions permettant la suppression et la modification d’un utilisateur.

* ***Interface d’édition d’un contact*** :

L’appui sur le bouton « modifier » de l’interface de consultation de contacts de la figure 44 permet d’ouvrir un modèle qui renvoie un formulaire remplit par les informations relatives au contact sélectionné.

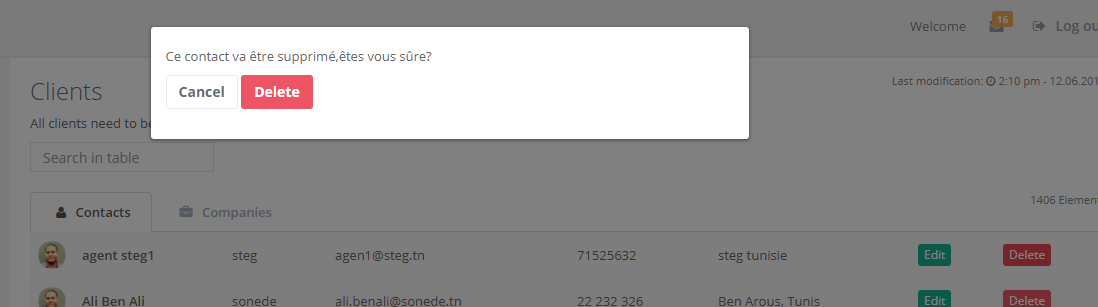
* ***Interface d’édition d’une société*** :

L’appui sur le bouton « modifier » de l’interface de consultation des sociétés de la figure 44 permet d’ouvrir un modèle qui renvoie un formulaire remplit par les informations relatives à la société sélectionnée.



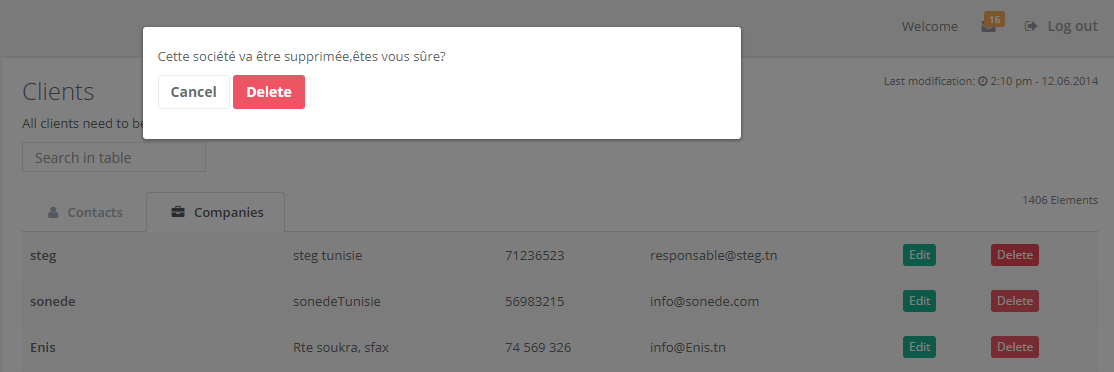
* ***Interface de suppression d’un contact*** :

Avant de supprimer un contact, il faut bien vérifier qu’il est sûr de cette action.

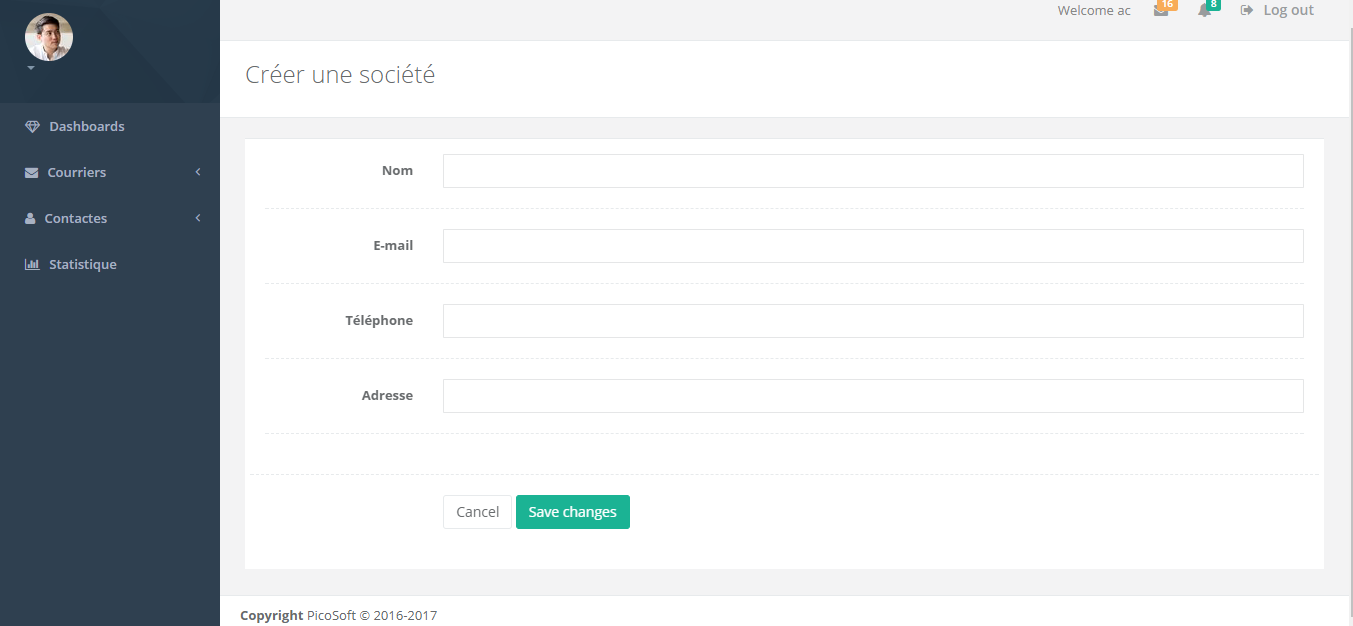


* ***Interface de suppression d’une société*** :

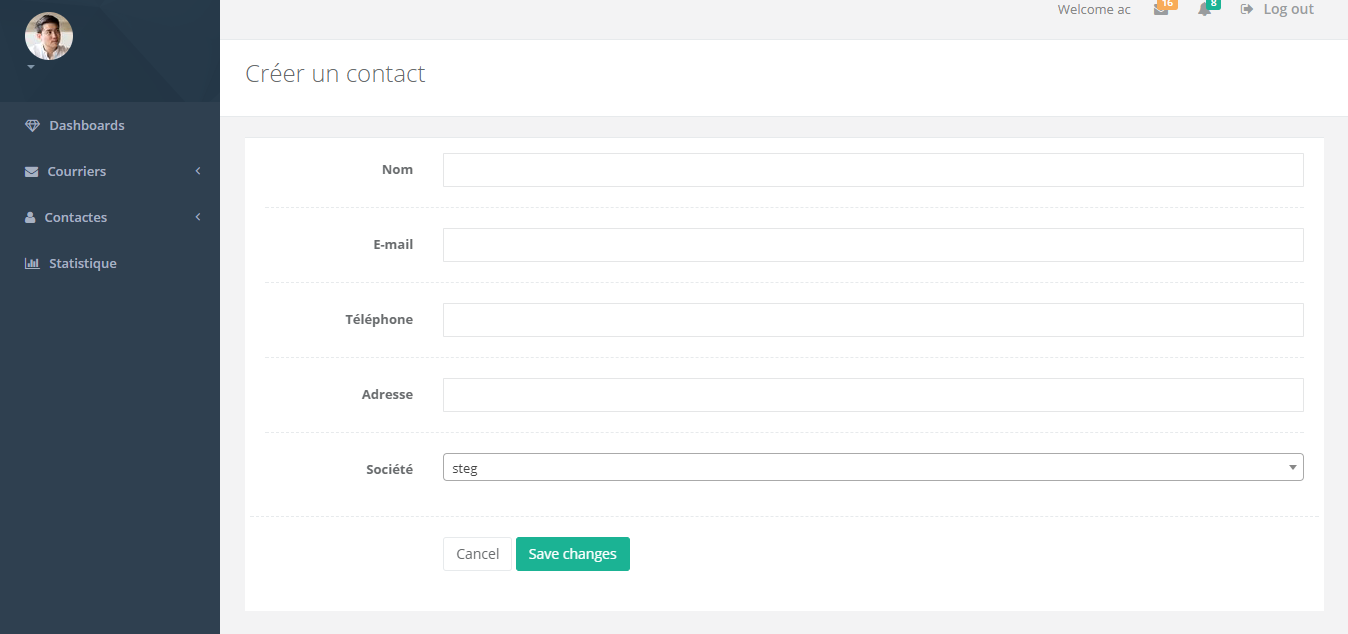
Avant de supprimer un contact il faut bien vérifier qu’il est sûr de cette action.



La figure 44 décrit l’interface de création des sociétés.



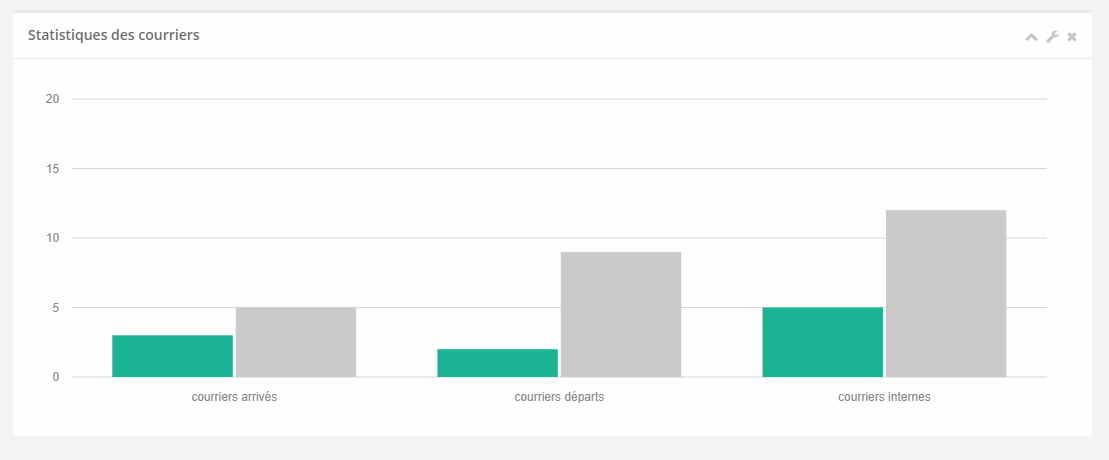
La figure 44 décrit l’interface de création des contacts. Chaque contact doit appartenir à une société donc une liste de sélection est présente dans le formulaire contenant la liste de toutes les sociétés.



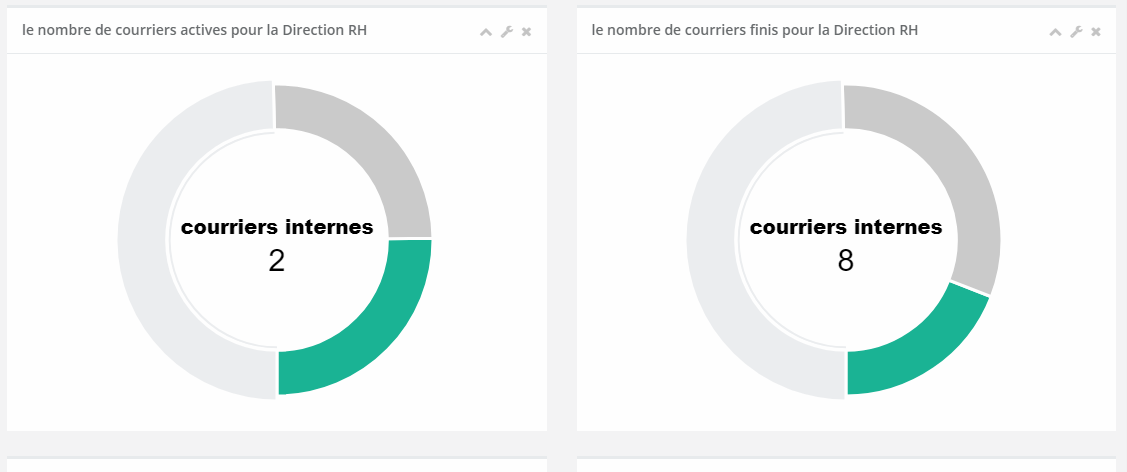
3.4 Sprint 4 : Statistiques

Pour mieux améliorer la visibilité de la société et savoir tous les détails sur l’échange de courriers, l’ajout d’une interface qui donne les statistiques devient très important

La figure 44 décrit un graphe qui représente le nombre de courriers actifs et finis pour chaque type de courrier : arrivés, départs et internes.



La figure 44 décrit un graphe qui représente le nombre de courriers actifs et finis pour chaque direction qui se trouve dans la société pour chaque type de courrier.

Dans cet exemple on donne le nombre de courriers actifs et finis pour la direction ressource humaine.

3.2 Sprint 2 : gestion des courriers internes

* ***Interface de création d’un courrier interne*** :

Suite à une authentification réussite, l’administrateur peut accéder au menu de création du nouveau courrier interne.

Après avoir rempli toutes les informations nécessaires correctement, il va y avoir création d’un nouveau courrier interne. Une fois l’utilisateur clique sur le bouton « envoyer », le système vérifie la validité des champs qui vont être les paramètres du processus courrier interne et vont être stockés dans la base de données du moteur de workflow activiti.

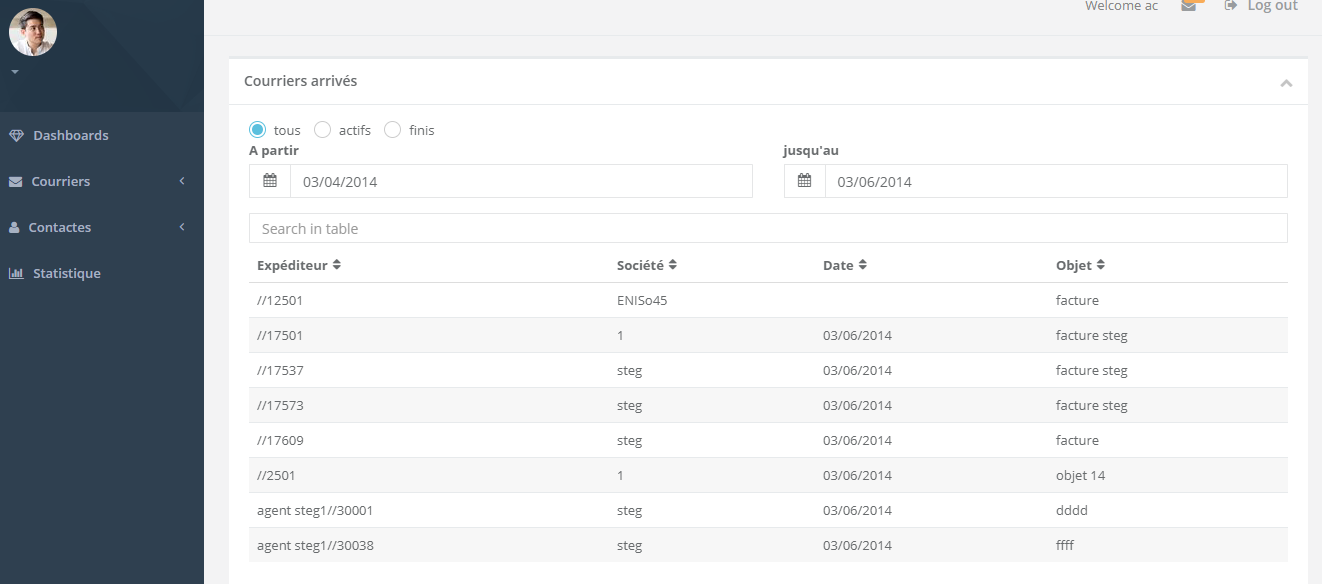
Si tous les paramètres sont valides il va y avoir création d’un nouveau courrier interne.

Dans le cas inverse et en cas de d’une erreur dans n’importe qu’elle phase de création (insertion des variables de processus dans la base MySQL, insertion de pièces jointes dans la base de données Alfresco, …) une page d’erreur va être envoyée.



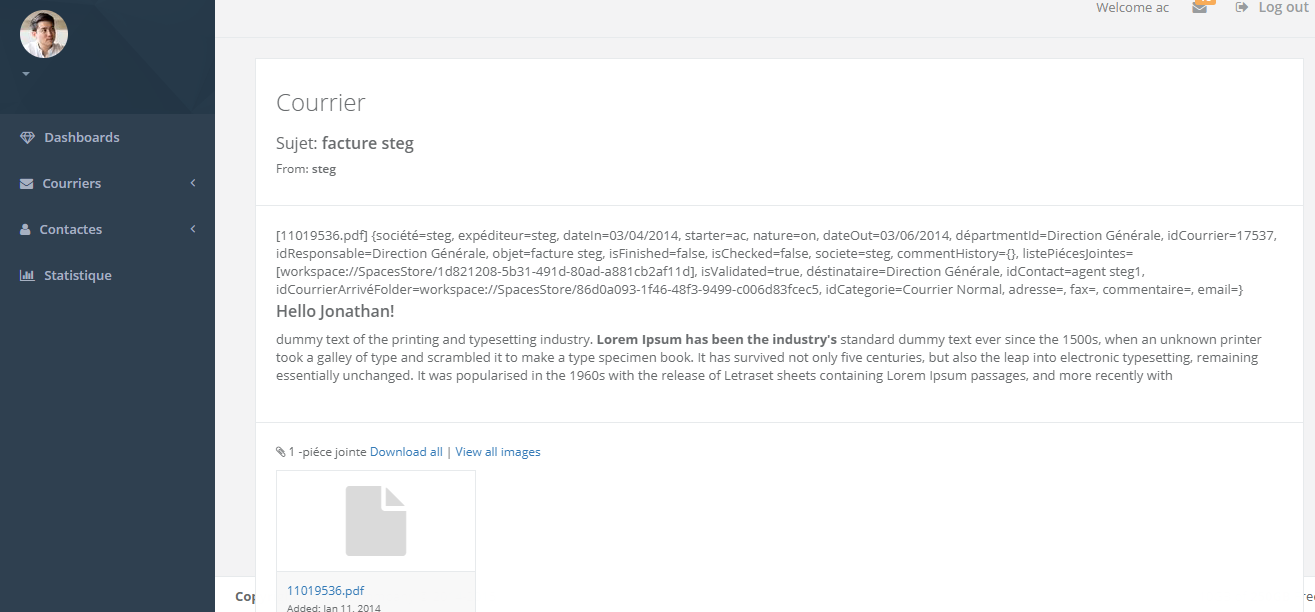
* ***Interface d’consultation des courrier interne*** :

En appuyant sur le menu « courrier interne » dans la barre de navigation, on peut afficher la liste des courriers internes

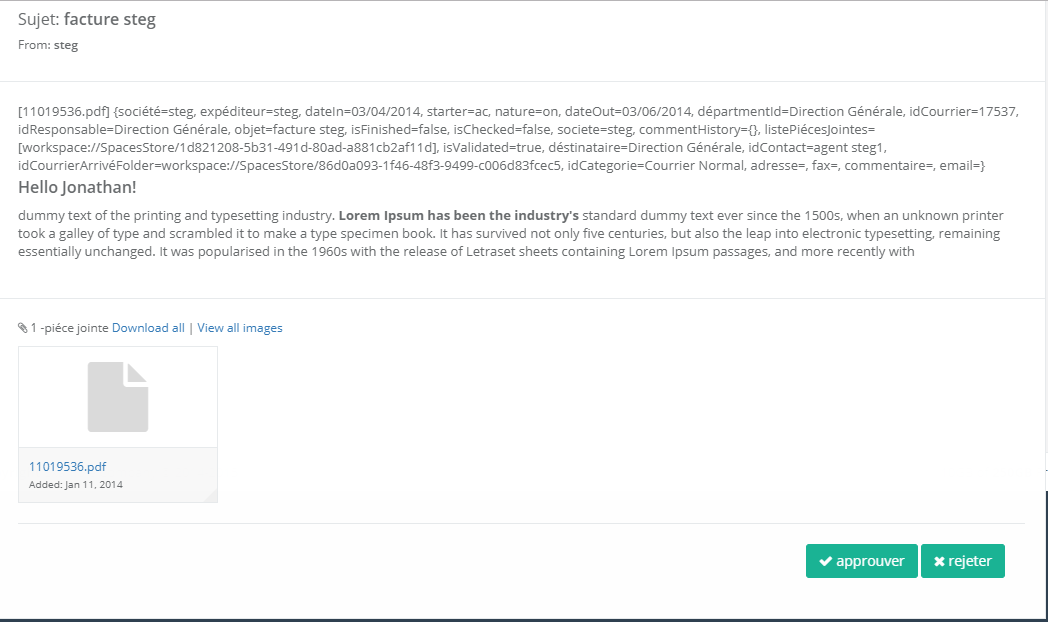


En appuyant sur l’un des courriers, on peut consulter tous les détails de ce courrier.

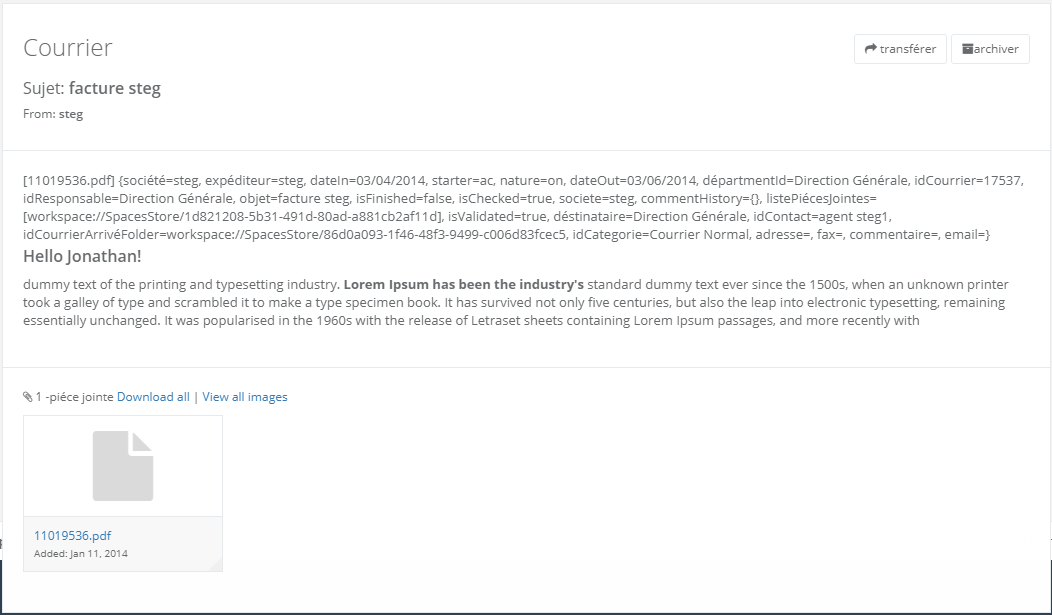
Cette interface peut changer selon l’état de processus :



Si par exemple, un courrier est bloqué dans la phase révision du courrier par le bureau d’ordre, quand un membre du bureau d’ordre ouvre ce courrier, il trouve l’interface de consultation du courrier avec deux boutons pour qu’il valide ou refuse le courrier



Dans un autre cas, et si par exemple le courrier est en phase de traitement et le chef de direction concerné ouvre le courrier, il va trouver deux boutons de traitement de courrier. Donc soit archiver ce courrier et finir le processus, soit transférer le courrier à un inférieur hiérarchique.



* ***Interface de modification des courrier interne*** :

C’est la même interface de création d’un courrier interne mais avec tous les champs remplis et en attente de modification.

///

3.3 Sprint 3 : gestion des courriers de départs

* ***Interface de création d’un courrier départ*** :

Suite à une authentification réussite, l’administrateur peut accéder au menu de création de nouveau courrier départ.

Après avoir remplir toutes les informations nécessaires correctement, il va y avoir création d’un nouveau courrier départ. Une fois l’utilisateur clique sur le bouton « envoyer » le système vérifie la validité des champs qui vont être les paramètres du processus courrier départ et vont être stockés dans la base de données du moteur de workflow activiti.

Si tous les paramètres sont valides, il va y avoir création d’un nouveau courrier départ.

Dans le cas inverse et en cas d’une erreur dans n’importe qu’elle phase de création (insertion des variables de processus dans la base MySQL, insertion du pièce jointe dans la base de données Alfresco, …) une page d’erreur va être envoyée.

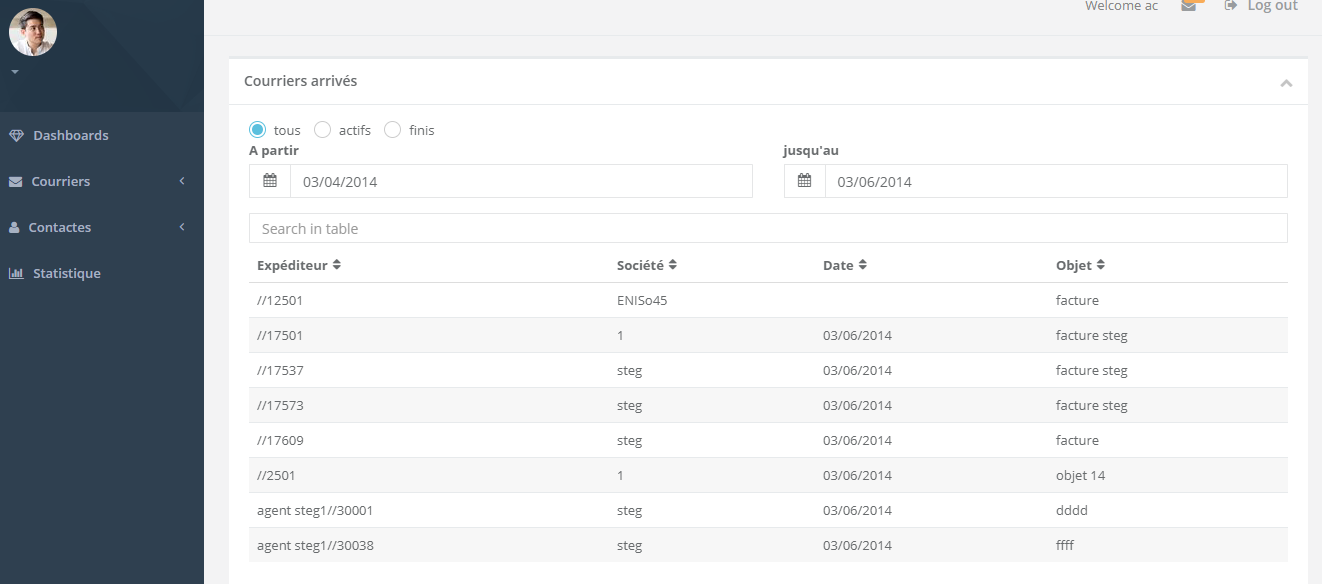


* ***Interface d’consultation des courrier départ*** :

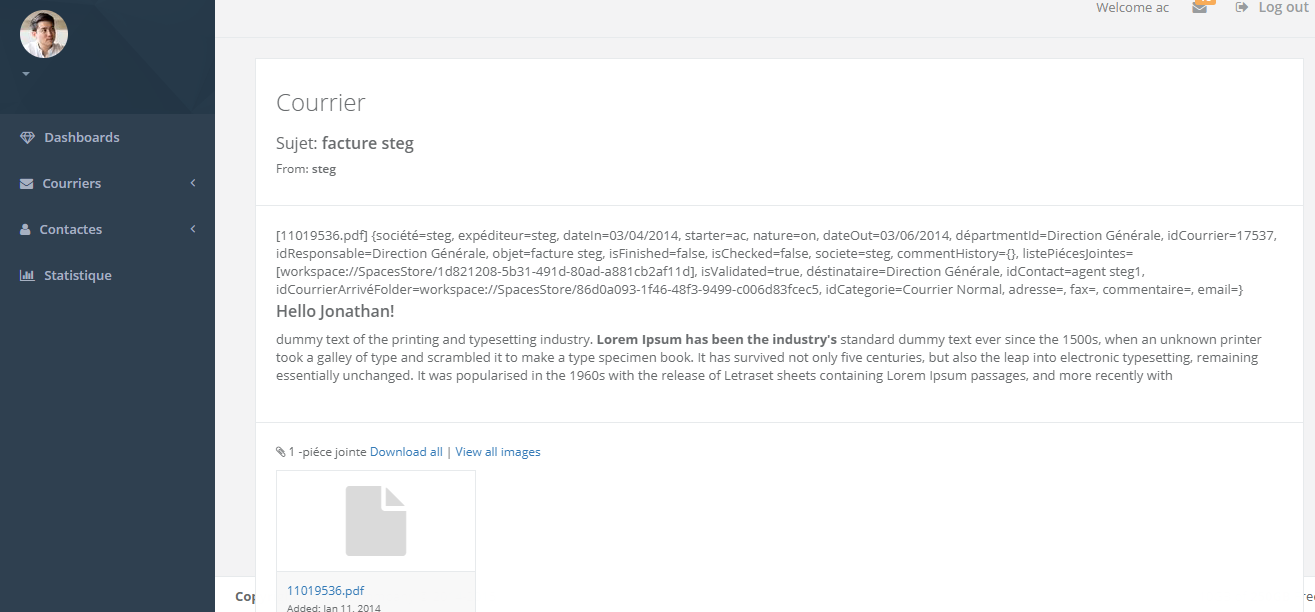
En appuyant sur le menu courrier départ dans la barre de navigation, on peut afficher la liste des courriers départs

En appuyant sur l’un des courriers on peut consulter tous les détails de ce courrier.

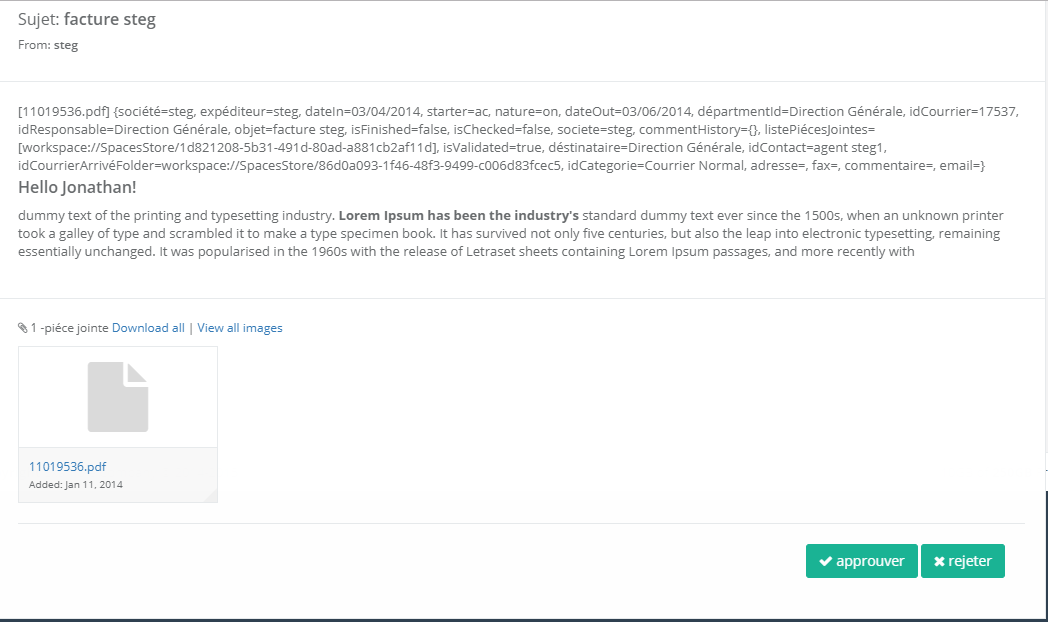
Cette interface peut changer selon l’état de processus :



Si par exemple un courrier est bloqué dans la phase révision du courrier par le bureau d’ordre, quand un membre du bureau d’ordre ouvre ce courrier, il trouve l’interface de consultation du courrier avec deux boutons pour qu’il valide ou refuse le courrier

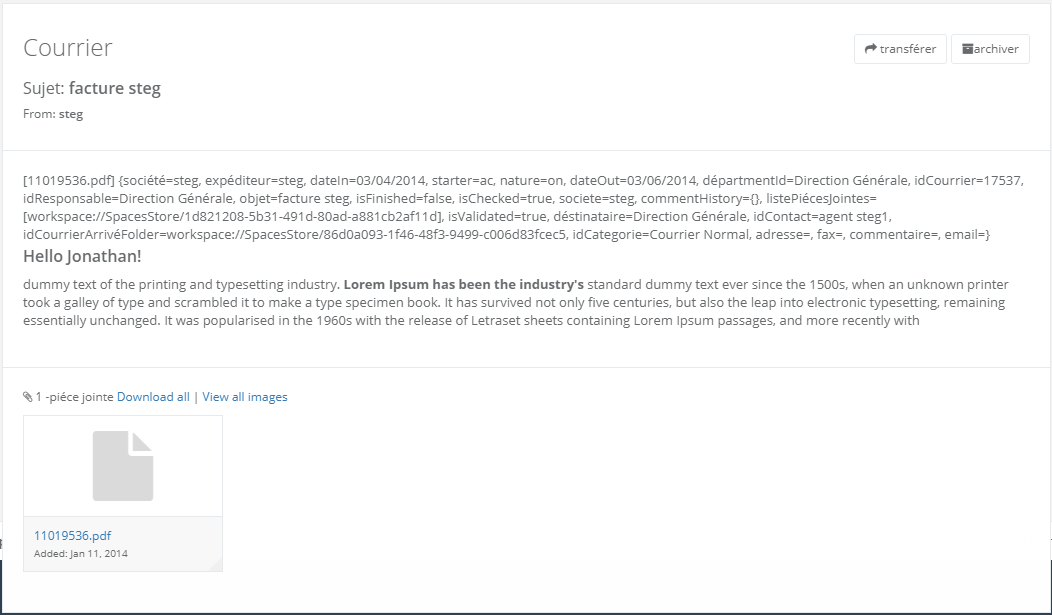


Dans un autre cas, et si par exemple le courrier est en phase de traitement et le chef de direction concerné ouvre le courrier, il va trouver deux boutons de traitement de courrier : soit archiver ce courrier et finir le processus, soit transférer le courrier à un inférieur hiérarchique.



* ***Interface de modification des courrier départ*** :

C’est la même interface de création d’un courrier départ mais avec tous les champs remplis et en attente de modification.



**Conclusion**

Le premier volet de ce chapitre a abouti à fixer le choix technique et l’environnement de développement matériel et logiciel. Ensuite, nous avons présenté une liste des interfaces de l’application pour expliquer le fonctionnement de notre solution.