**Chapitre 1 : Présentation de la société et étude de l’existant**

1. **Introduction**

Pour placer notre projet dans son cadre, nous commençons dans ce premier chapitre par une présentation de l’organisme d’accueil PicoSoft, ensuite nous poserons la problématique qui a engendré ce travail, et nous abordons l'étude de l'existant puis on va finir par une présentation de la solution proposée.

1. **Présentation de l’organisme d’accueil**
   1. **Présentation générale de PicoSoft**



**Figure 1: Logo PicoSoft**

PicoSoft est une société d'ingénierie et de conseil en informatique aux services des organisations engagées dans une démarche d'échanges dynamiques d'informations en favorisant le travail de groupe et la mobilité géographique.

Les solutions proposées par PicoSoft sont spécialisées dans le travail collaboratif et les collecticiels, le e-business et les technologies Intranet / Internet / Extranet.

* 1. **Services**

PicoSoft propose les services du consulting de développement spécifique et de formation. La mission de PicoSoft est d'assurer alors une prise en charge globale du cycle de vie des projets de sa clientèle. Elle offre en effet à ses clients une étude approfondie leurs permettant de mettre en place l'architecture la plus adéquate à leurs entreprises. Cette étude comprend, entre autres, l'architecture physique, la topologie d'implantation et la structure logique des serveurs, ainsi que l'organisation et la répartition des rôles et des responsabilités de l'administration du système.

D'un autre côté, PicoSoft met à la disposition de ses clients des solutions répondant à leurs besoins spécifiques et assiste leurs personnels lors du développement des applications de workflow et de travail collaboratif.

Finalement, et pour apporter son savoir-faire auprès de ses clients, PicoSoft offre des formations adaptées dans différents domaines. Citons à titre d'exemples le développement WEB et le travail collaboratif.

* 1. **Produit**

La société PicoSoft est spécialisée dans les technologies IBM Lotus au titre d'Advanced IBM Business Partner. Son expertise dans ces technologies est largement reconnue. Elle développe avec Lotus Notes Domino des solutions génériques.

* + 1. **Présentation du Mail Manager**



**Figure 2: Logo Mail Manager**

Mail Manager qui est un outil simple permettant la gestion du courrierS, des notes de services, des demandes de congés, des PV ainsi que tous formulaires internes circulant au sein d'une entreprise, quel que soit sa forme (lettre, télécopie, courrier électronique). S’appuyant sur une base de données documentaire. Mail Manager permet l'enregistrement, l’affectation, la diffusion et le suivi du courrier des correspondants externes de l'organisation et le classement des documents échangés de façon logique (par expéditeur, par date, par référence, par dossier, etc.).

Mail Manager s'adresse aux administrations publiques, et en général, à toute organisation pour laquelle la gestion du courrier est une fonction stratégique.

* + 1. **Présentation du Quality Manager**

Quality Manager est un progiciel destiné aux entreprises recherchant une solution simple, conviviale et efficace pour les fonctions de gestion de la documentation et de management de la qualité. Il constitue une base de connaissances indispensable à tout processus d'amélioration continue de la qualité grâce à ses fonctions avancées en matière de gestion documentaires et de planification des actions qualités. Parmi ses fonctions, nous pouvons citer la planification, le traitement et le suivi des audits et des non-conformités, la gestion et le suivi des actions correctives et préventives, la gestion des versions des documents et le référencement paramétrable.

* + 1. **Présentation du Leave Manager**

https://lh5.googleusercontent.com/T8eq_KtcXj43ZOD4wqulB-JoCQqf0KWctcZGS9-aSqNpX76rmSygAqmnKyDbW_ki2a2XMqvryw0dhR7lA5aHF1I3axLDEfkxdp-P08IiMWpuih2EsZfAXhmlT23auEcBHwVRH5Y

**Figure 3: Logo Leave Manager**

Leave Manager est une application développée au sein de PicoSoft qui permet de gérer les congés des employés d’une entreprise.

1. **Cadre du projet**

Dans le cadre de ses activités de développement logiciel orientées vers le domaine de

la gestion de contenu d'entreprise (ECM), la société PicoSoft cherche toujours à offrir à ses clients des produits qui soient les plus performants et les plus fiables. Ainsi, elle cherche à fournir des services innovants qui répondent aux différents besoins des clients en se basant sur les nouvelles technologies.

C’est dans ce contexte que la société PicoSoft compte à concevoir et à développer une application Java/JEE de gestion de courriers dans la cadre de notre projet de fin d’études en vue de l’obtention du diplôme national d’ingénieur en informatique.

1. **Problématique**

Un bon nombre d'entreprises décident d'opter pour une solution de gestion de contenu d’entreprise stratégique une fois qu'un enjeu métier important se présente. Généralement, il s'agit de la gestion d’un grand nombre de documents-papiers ou d’informations non structurées créant des goulots d’étranglement à chaque tentative de traitement, d'accès, ou de stockage de ces informations et documents. Prenons l’exemple d’un dossier de demande-prêt : beaucoup d’éléments doivent être incorporés et suivis dans ce dossier. Si l'un des formulaires du dossier venait à manquer, les délais devront être prolongés, et ce manque ne sera pas sans conséquences sur l'issue et la clôture du dossier. Or, avec une solution de gestion de contenu (ECM), tout serait tellement plus simple.

1. **Etude de l'existant**
   1. **Mail manager** ([http://www.PicoSoft.biz/mail\_manager.htm](http://www.picosoft.biz/mail_manager.htm))

La gestion des courriers est une tâche aussi nécessaire que quotidienne dans le bureau d’ordre. Il s’agit d’une tâche critique puisque plusieurs risques peuvent être envisagés et dont les employés du bureau d’ordre souffrent tel que: perte d’un courrier, courrier accumulé sans traitement, difficulté de suivi des courriers… Cela pousse les entreprises à installer un système de gestion de courrier afin de minimiser les problèmes liés à une telle tâche fastidieuse.

C’est dans ce cadre que PicoSoft a développé une application de gestion électronique des courriers, sous le nom de Mail manager utilisé au sein d’elle ainsi que  chez plusieurs clients afin de bien gérer leurs courriers et échapper aux problèmes dont souffre le bureau d’ordre. Cette solution est basée sur le GED Alfresco et le moteur de workflow picoEngine : un moteur de workflow interne développé au sein de PicoSoft. Il permet à une entreprise de gérer différents types de courriers : arrivés, départs et internes.



**Figure 4: Page d'accueil du Mail Manager**

* + 1. **Fonctions**
* Gestion et suivi des courriers en arrivée, départ et interne.
* Supervision et suivi des documents et des tâches.
* Fonction avancée de recherche.
* Classement des courriers.
* Gestions des dossiers.
* Enregistrement et affectation des fax et des e-mails.
* Gestion d'annuaire de contacts internes et externes.

* + 1. **Caractéristiques Techniques**
* Fonctionnement multi-sites et 100 % web.
* Recherche documentaire en texte intégral.
* Signature électronique du courrier.
* Saisie rapide des documents-papiers et électroniques.
* Gestion des droits d'accès à plusieurs niveaux.
* Génération de rapports d'activités.
* Intégration de messagerie : envoi de messages avec liens dynamiques

* 1. **Autres solutions sur le marché**



**Figure 5: Logo maarch**

En plus du mail manager on a trouvé d’autres solutions qui répondent aux besoins de l’entreprise parmi lesquelles on cite: maarch courrier,bitrix24,Epistolaire...Ces solutions sont généralement en mode saas ce qui pousse l’entreprise à penser à la sécurité de ces documents. En plus un coût cher avec limite de l’espace du stockage.

1. **Critique de l’existant**

 Après avoir testé la solution existante et discuté avec ces utilisateurs, nous avons pu dégager plusieurs problèmes fonctionnels, ergonomiques et technologiques. Parmi lesquelles on cite :

* 1. **Problèmes d’ergonomie**
* L’interface du mail-manager est peu chargée.
* Un design non esthétique.
* Une interface non responsive.
* Manque d’interactions.
  1. **Problèmes fonctionnels**
* Problème de lenteur.
* De préférence elle soit ouverte sur le navigateur internet explorer.
  1. **Problème technologique**
* Les technologies utilisées sont dépassées.

1. **Solution proposée**

Afin de corriger les lacunes et les insuffisances citées précédemment et résoudre les problèmes résidents dans l’application existante, l’entreprise d’accueil a proposé de développer une application de gestion de documents avec des flux de travail qui permet à l’entreprise consommant ce produit de manipuler un courrier (entrée, départ ou interne) quel que soit sa forme (lettre, télécopie, facture ...) en attachant à chaque courrier un flux de travail bien déterminé afin de suivre ,enregistrer, affecter et diffuser le courrier.

1. **Conclusion**

Dans ce chapitre nous avons présenté l’organisme d’accueil ainsi que ces produits. Nous avons aussi mis le projet dans son cadre en plus d’une étude détaillé de l’existant accompagné d’une critique. Enfin nous avons présenté la solution qu’on a proposé. Dans le chapitre qui suit, nous allons détailler les besoins fonctionnels et non fonctionnels de notre application.

**Chapitre 2 : Analyse et spécification des besoins**

1. **Introduction**

Après avoir présenté le cadre général de notre projet, nous présentons dans ce chapitre la spécification et l’analyse des besoins. On va commencer par une première partie qui concerne la spécification des besoins fonctionnels et non fonctionnels, puis on a une autre partie qui concerne l’analyse détaillé.

1. **Objectif de l’application**

Notre application a comme but la dématérialisation des courriers au sein de PicoSoft quelques soient leurs types : courriers arrivés, courriers départs, ou courrier internes.

**Courriers arrivés** : Ce sont les courriers qui arrivent à l’entreprise.

**Courriers départs** : Ce sont les courriers rédigés au sein de l’entreprise et qui vont être envoyés à l’un de ces contacts.

**Courriers internes** : Ce sont les courriers qui circulent entre les différentes directions de l’entreprise.

1. **Spécification des exigences**

Afin de clarifier d’avantage le besoin de l’utilisateur, on va présenter dans cette partie les différents besoins fonctionnels et non fonctionnels.

* 1. **Besoins fonctionnels**
     1. **Authentification**

Afin de pouvoir accéder à l’application, l’utilisateur doit procéder à une authentification sécurisée afin d’assurer la traçabilité de leurs actions, ainsi que l’intégrité et la non-falsification des informations.

* + 1. **Gestion des courriers arrivés**

Un employé du bureau d’ordre à la possibilité de gérer les courriers arrivés de l’entreprise : Il peut créer, modifier et supprimer un courrier arrivé.

* + - 1. **Créer un courrier arrivé**

Après avoir scanner le document physique, un employé du bureau d’ordre peut entrer le courrier dans le système en saisissant les différentes informations du courrier (destinataire, expéditeur, date d’arrivée…) et en joignant les pièces jointes.

* + - 1. **Modifier un courrier arrivé**

Avant qu’il passe à la révision, l’employé du bureau d’ordre a la possibilité de modifier le contenu du courrier arrivé.

* + - 1. **Supprimer un courrier arrivé**

L’employé du bureau d’ordre a la possibilité de supprimer un courrier arrivé.

* + 1. **Gestion des courriers départs**

La gestion des courriers départs peuvent être faites par le créateur du courrier lui-même. Il peut être n’importe quel employé de la société.

* + - 1. **Créer un courrier départ**

Après avoir scanné le document physique, n’importe quel employé de la société peut entrer le courrier dans le système en saisissant les différentes informations du courrier (destinataire, date de création…) et en joignant les pièces jointes.

* + - 1. **Modifier un courrier départ**

Avant qu’il passe à la révision, le créateur du courrier a la possibilité de modifier son contenu.

* + - 1. **Supprimer un courrier départ**

Le créateur du courrier a la possibilité de supprimer un courrier départ, initié par lui-même.

* + 1. **Gestion des courriers internes**

Notre application offre aussi à ces utilisateurs la possibilité de gérer leurs courriers internes qui peuvent être lancés par n’importe quel employé de l’entreprise.

* + - 1. **Créer un courrier interne**

Après avoir scanné le document physique, n’importe quel employé de la société peut entrer le courrier dans le système en saisissant les différentes informations du courrier (destinataire, date de création…) et en joignant les pièces jointes.

* + - 1. **Modifier un courrier interne**

Avant qu’il passe à la révision, le créateur du courrier a la possibilité de modifier son contenu.

* + - 1. **Supprimer un courrier interne**

Le créateur du courrier a la possibilité de supprimer un courrier interne initié par lui-même.

* + 1. **Réviser un courrier**

Avant que le courrier passe à sa destination, le courrier doit être révisé. On a alors deux choix :

* + - 1. **Valider un courrier**

Si le courrier est validé, cela signifie que les pièces jointes sont lisibles et par la suite le courrier peut passer à l’étape suivante.

* + - 1. **Refuser un courrier**

Si le courrier est refusé, cela signifie que le supérieur chef du bureau d’ordre ou le chef de direction n’est pas d’accord sur le contenu, d’où le courrier revient à son initiateur.

* + 1. **Consulter un courrier**

N’importe quel utilisateur peut consulter un courrier dans sa boîte, quoi qu’il soit son type : arrivé, départ, ou interne. Il peut visualiser le contenu des pièces jointes ainsi que les différentes informations à propos du courrier.

* + 1. **Transférer un courrier**

N’importe quel utilisateur peut transférer un courrier au sein de sa direction.

* + 1. **Archiver un courrier**

N’importe quel utilisateur peut classer un courrier et par la suite mettre fin à son processus.

* + 1. **Consulter les statistiques**

Les employés du bureau d’ordre ont la possibilité de visualiser des chartes montrant le nombre de courriers actifs et finis pour chaque directions ainsi que le nombre de courriers actifs et finis pour toutes la société.

* + 1. **Gestion des contacts**

Notre application offre aussi la possibilité de gérer les contacts de l’entreprise.

La gestion des contacts renferme : la création, la modification et la suppression du contact.

* + - 1. **Créer un contact**

N’importe quel utilisateur à la possibilité d’ajouter un contact à la société.

* + - 1. **Modifier un contact**

N’importe quel utilisateur à la possibilité de modifier les informations d’un contact.

* + - 1. **Supprimer un contact**

Seuls les employés du bureau d’ordre ont la possibilité de supprimer un contact.

* + 1. **Gestion des sociétés**

Notre application permet aussi de gérer les différentes sociétés avec lesquels l’entreprise communique.

* + - 1. **Créer une société**

N’importe quel utilisateur à la possibilité d’ajouter une société.

* + - 1. **Modifier une société**

N’importe quel utilisateur à la possibilité de modifier les informations d’une société.

* + - 1. **Supprimer une société**

Seuls les employés du bureau d’ordre ont la possibilité de supprimer une société.

* 1. **Besoins non fonctionnels**

Les besoins non fonctionnels présentent les contraintes qui jugent un service du système et définissent le degré de satisfaction du client. Les besoins non fonctionnels de notre plate-forme se résument comme suit :

* + 1. **Sécurité**
* Le processus d’authentification exige que chaque utilisateur dispose d’un login et d’un mot de passe qui garantit que la personne connectée est bien celle qu'elle prétend être.
* Le processus des autorisations permet de limiter les accès à une url à un groupe prédéfinie de personnes qui ont un rôle d’agir sur cette page.
* L’utilisation d’une base de données LDAP permet la sécurisation des mots de passe des utilisateurs puisque LDAP permet de les hacher.
* La sécurisation de la session est assurée par une fonction de protection contre les attaques CSRF : L’objet de cette attaque est de transmettre à un utilisateur authentifié une requête HTTP falsifiée qui pointe sur une action interne au site, afin qu'il l'exécute sans en avoir conscience et en utilisant ses propres droits. L’utilisateur devient donc complice d’une attaque sans même s'en rendre compte. L'attaque étant actionnée par l'utilisateur, un grand nombre de systèmes d'authentification sont contournés, et par l’utilisation de la fonction

« Remember me » qui permet de se rappeler l'identité d'un principal entre les sessions.

* + 1. **Ergonomie et souplesse**

Notre application doit avoir une structure générale claire et simple et doit être compréhensible et facilement utilisée pour le public ciblé avec des interfaces conviviales et lisibles et responsives.

* + 1. **Compatibilité**

L’un des points les plus importants lors du développement d’une application est d’assurer sa compatibilité avec n’importe quelle version du système et n’importe quelle version et type de navigateur.

* + 1. **Aptitude à la maintenance**

Le code de l’application implémentée doit être facile à maintenir et à étendre tout en respectant une structure orientée objets, pour des raisons de réutilisation.

* + 1. **Portabilité**

Cette application peut tourner sur n’importe quel système d’application desktop (Windows, linux).

1. **Spécification détaillée**

Dans cette partie nous allons définir tout d’abords les acteurs. Puis nous allons présenter l’analyse détaillée des besoins fonctionnels de notre application. Pour cela, on va utiliser les diagrammes de cas d’utilisation qui servent à donner une vision globale du comportement fonctionnel du système et montrant l’interaction avec les acteurs.

* 1. **Présentation des acteurs**

Un acteur est l'idéalisation d'un rôle joué par une personne externe, un processus ou une chose qui interagit avec un système.

Notre application est dédiée principalement aux employés de PicoSoft.

En analysant les besoins fonctionnels on distingue 4 types d’acteurs :

**Employé :**

Cet acteur présente un simple employé de l’entreprise qui a des fonctionnalités basiques dans notre application, il peut alors :

* Gérer les courriers départs et internes qui sont lancés par lui-même.
* Transférer les courriers.
* Archiver les courriers.
* Créer un contact.
* Modifier un contact.
* Créer une société.
* Modifier une société.

**Chef de direction :**

Un chef de direction a les mêmes privilèges qu’un employé, mais il peut en plus faire la révision du courrier départ de leurs directions.

**Employé du bureau d’ordre :**

Ces employés ont les mêmes privilèges qu’un simple employé, mais ils possèdent en plus les fonctionnalités suivantes :

* Gérer les courriers arrivés : création, modification et suppression.
* Visualiser des statistiques.
* Gestion de contacts : création, modification et suppression.
* Gestion des sociétés : création, modification et suppression.

**Chef du bureau d’ordre :**

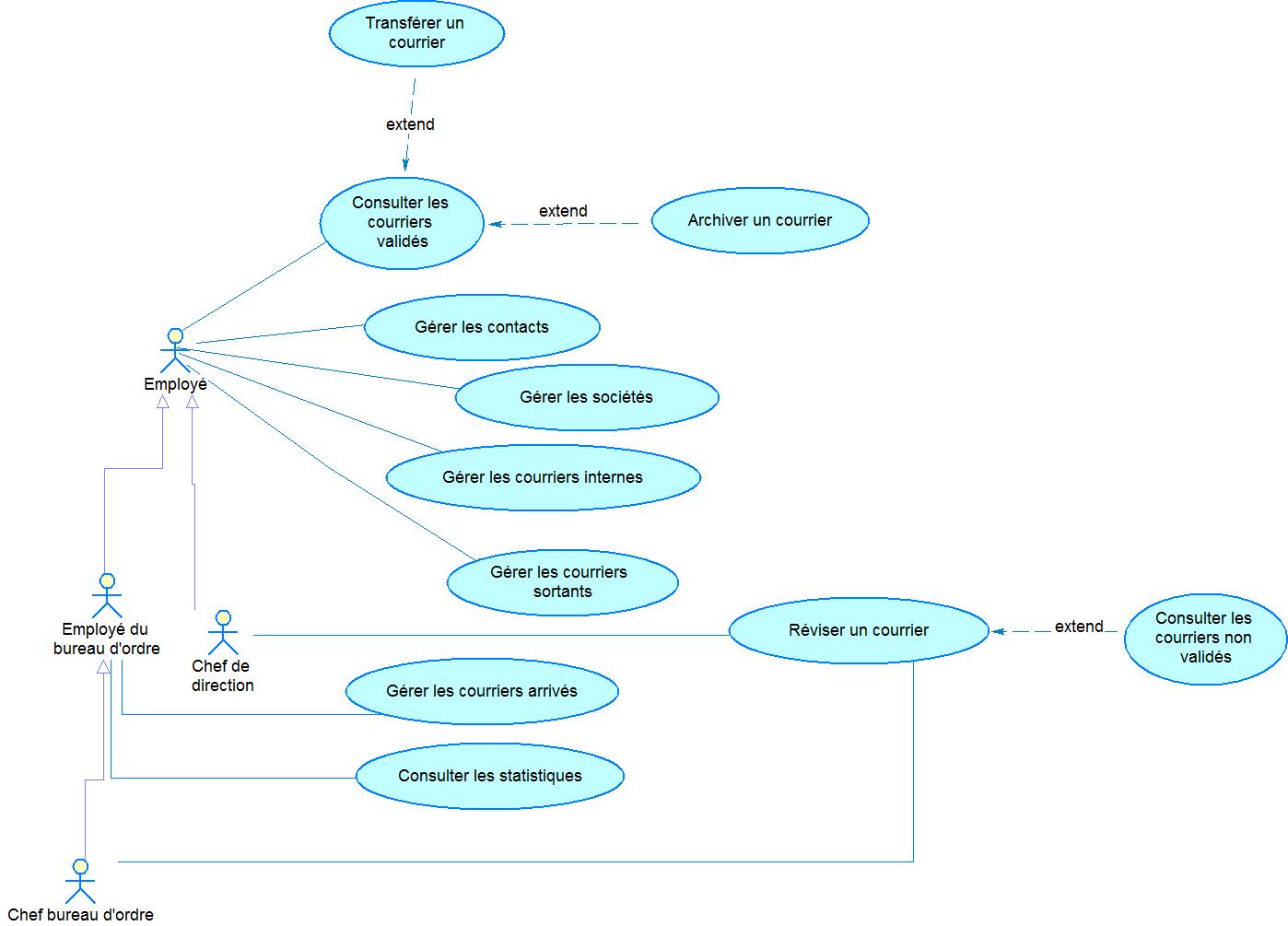
Le chef du bureau d’ordre a les mêmes privilèges qu’un employé du bureau d’ordre, il a comme mission principale la révision des courriers arrivés.

* 1. **Diagrammes de cas d’utilisation**

Les diagrammes de cas d'utilisation sont des [diagrammes UML](https://fr.wikipedia.org/wiki/Unified_Modeling_Language#Les_diagrammes) utilisés pour donner une vision globale du comportement fonctionnel d'un système [logiciel](https://fr.wikipedia.org/wiki/Logiciel).

* + 1. **Diagramme de cas d’utilisation global**

La description des fonctionnalités de notre système nous permet de déduire le diagramme de cas d’utilisation suivant :



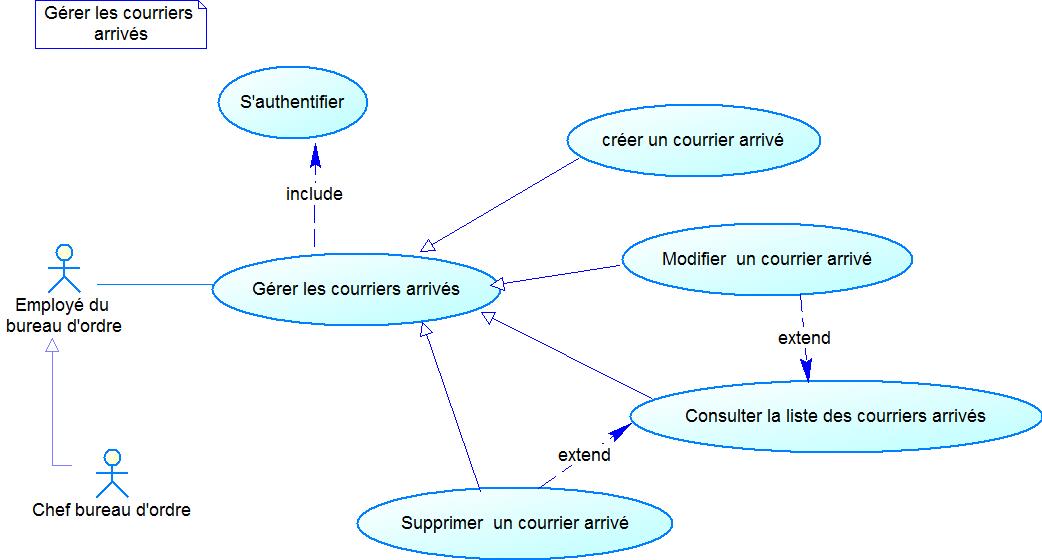
**Figure 6: Diagramme de cas d'utilisation globale**

Ce diagramme montre les différents cas d’utilisation qu’offre notre système ainsi que les acteurs et les tâches dont ils sont éligibles de les faire. Il montre aussi les relations entre les différents cas d’utilisations grâce aux relations « extend » et « include ».

**Remarque :** Afin de ne pas encombrer le diagramme de cas d'utilisation, nous allons présenter en détails le diagramme des cas d’utilisation qui sont liés par tout ce qui est gestion de courriers, sociétés et contacts.

|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Authentification |
| **Acteur** | Employé |
| **Résumé** | Permet à l’employé d’accéder à l’application de façon sécurisée |
| **Description des enchaînements** | |
| **Précondition** | Rien |
| **Post-condition** | Employé authentifié |
| **Scenario nominal** | 1-L’employé accède à l’application.  2-Un formulaire d‘authentification s’affiche.  3-Saisie d’identifiant.  4-Saisie de mot de passe.  5-Cliquer « s’authentifier ».  6-Le système vérifie les données d’authentification.  7-Afficher un message d’authentification avec succès et redirection vers la page d’accueil. |
| **Scénario alternatif** | A6 : Données d’authentification invalide.  7-Afficher un message d’erreur.  8- Le scénario reprend de 2. |

* + 1. **Diagramme du cas d’utilisation de la gérance des courriers arrivés**



**Figure 7: Diagramme du cas d’utilisation de la gérance des courriers arrivés**

Ce cas d’utilisation permet aux employés du bureau d’ordre de gérer les courriers entrant à l’entreprise. Il renferme quatre sous cas d’utilisation : créer, modifier, supprimer un courrier arrivé et consulter la liste des courriers arrivés. Ce diagramme montre aussi que la gestion des courriers arrivés doit être précédées par l’authentification.

|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Créer un courrier arrivé |
| **Acteur** | -Employé bureau d’ordre  -Chef bureau d’ordre |
| **Résumé** | Ce cas d’utilisation permet de faire entrer un courrier arrivé au système. |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | Courrier arrivé crée |
| **Scenario nominal** | 1-L’utilisateur demande l’interface de création d’un courrier arrivé.  2-Un formulaire s’affiche.  3-L’utilisateur entre les informations nécessaires.  4-L’utilisateur clique sur « créer courrier ».  5-Le système vérifie les données saisies.  6-Le système crée un courrier arrivé.  7-Un message de succès de création de courrier s’affiche. |
| **Scénario alternatif** | A5 : Données invalides ou champs obligatoires absents.  1-Afficher un message d’erreur.  2-Le scénario reprend de 2. |

**Tableau 1: Créer un courrier arrivé**

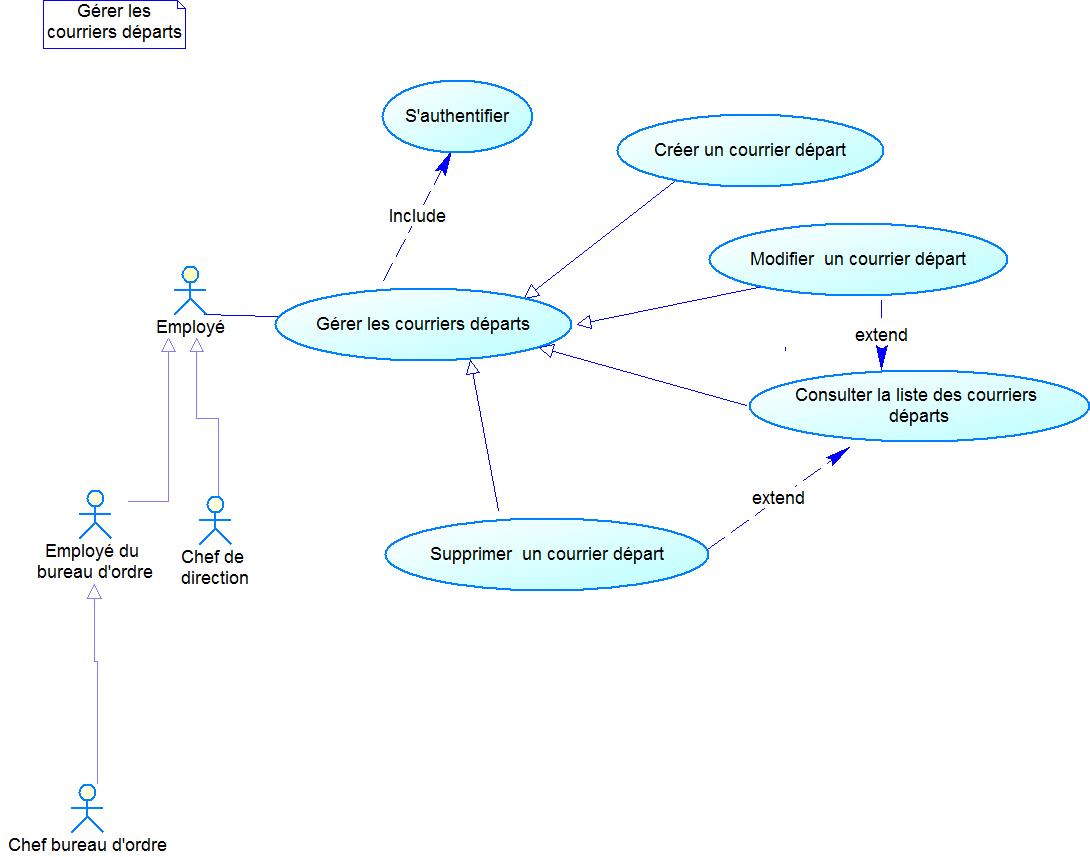
|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Modifier un courrier arrivé |
| **Acteur** | - Employé bureau d’ordre  - Chef bureau d’ordre |
| **Résumé** | Ce cas d’utilisation permet de modifier les données d’un courrier arrivé |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | Courrier arrivé modifié |
| **Scenario nominal** | 1-L’utilisateur demande la liste des courriers arrivés.  2-L’utilisateur choisit le courrier à modifier.  3-L’utilisateur demande l’interface de modification d’un courrier arrivé.  2-Un formulaire s’affiche chargé par les données courantes.  3- L’utilisateur modifie les informations nécessaires.  4- L’utilisateur clique sur « modifier courrier ».  5-Le système demande à l’utilisateur s’il veut vraiment modifier les données du courrier.  6-L’utilsateur répond par « OUI ».  7-Le système vérifie les données saisies.  8-Le système modifie un courrier arrivé.  9-Un message de succès de modification du courrier s’affiche. |
| **Scénario alternatif** | A6 : Annuler la modification.  6-L’utilsateur répond par « ANNULER ».  7- Le scénario reprend de 3.  A7 : Données invalides ou champs obligatoires absents.  8-Afficher un message d’erreur  9- Le scénario reprend de 3 |

**Tableau 2: Modifier un courrier arrivé**

|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Supprimer un courrier arrivé |
| **Acteur** | -Employé bureau d’ordre  -Chef bureau d’ordre |
| **Résumé** | Ce cas d’utilisation permet de supprimer un courrier arrivé du système |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | Courrier arrivé supprimé |
| **Scenario nominal** | 1-L’utilisateur demande la liste des courriers arrivés.  2- La liste des courriers arrivés s’affiche.  3-L’utilisateur choisi le courrier à supprimer.  4- L’utilisateur clique sur « supprimer ».  5-Le système demande si l’utilisateur veut vraiment supprimer ce courrier.  6-L’utlisateur répond par « OUI ».  7-Un message de succès de suppression du courrier s’affiche. |
| **Scénario alternatif** | A6 : Annuler la suppression du courrier.  6- L’utilisateur répond par « ANNULER ».  7- Le scénario reprend de 2. |

**Tableau 3: Supprimer un courrier arrivé**

* + 1. **Diagramme du cas d’utilisation de la gérance des courriers départs**

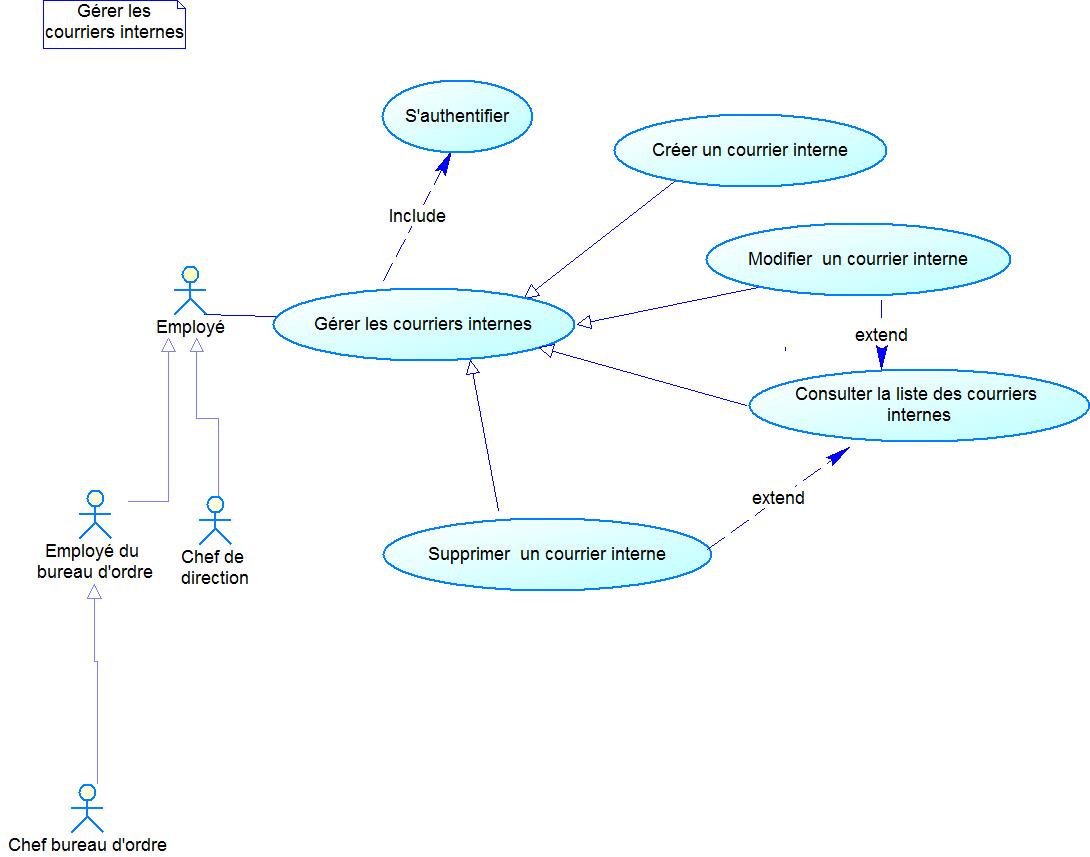


**Figure 8: Diagramme du cas d’utilisation de la gérance des courriers départs**

Cette figure montre le cas d’utilisation gérer les courriers départs, qui permet aux employés de gérer les courriers quittant l’entreprise et qui sont créés par eux-mêmes.

Il renferme quatre sous cas d’utilisation : créer, modifier, supprimer un courrier départ et consulter la liste des courriers départs. Ce diagramme montre aussi que la gestion des courriers départs doivent être précédés par l’authentification.

* + 1. **Diagramme du cas d’utilisation de la gérance des courriers internes**



**Figure 9: Diagramme du cas d’utilisation de la gérance des courriers internes**

Cette figure montre le cas d’utilisation gérer un courrier interne, qui permet aux employés de gérer les courriers circulants entre les différentes directions de l’entreprise et qui sont créés par eux-mêmes.

Il renferme quatre sous cas d’utilisation : créer, modifier, supprimer un courrier interne et consulter la liste des courriers internes. Ce diagramme montre aussi que la gestion des courriers internes doit être précédée par l’authentification.

* + 1. **Diagramme de cas d’utilisation réviser un courrier**

****

**Figure 10: Diagramme de cas d’utilisation réviser un courrier**

Ce cas d’utilisation permet au chef du bureau d’ordre et au chef de direction de valider ou refuser un courrier. Le chef du bureau d’ordre fait la révision de courriers quittant et rentrant au bureau d’ordre et le chef de direction révise les courriers quittant sa direction.

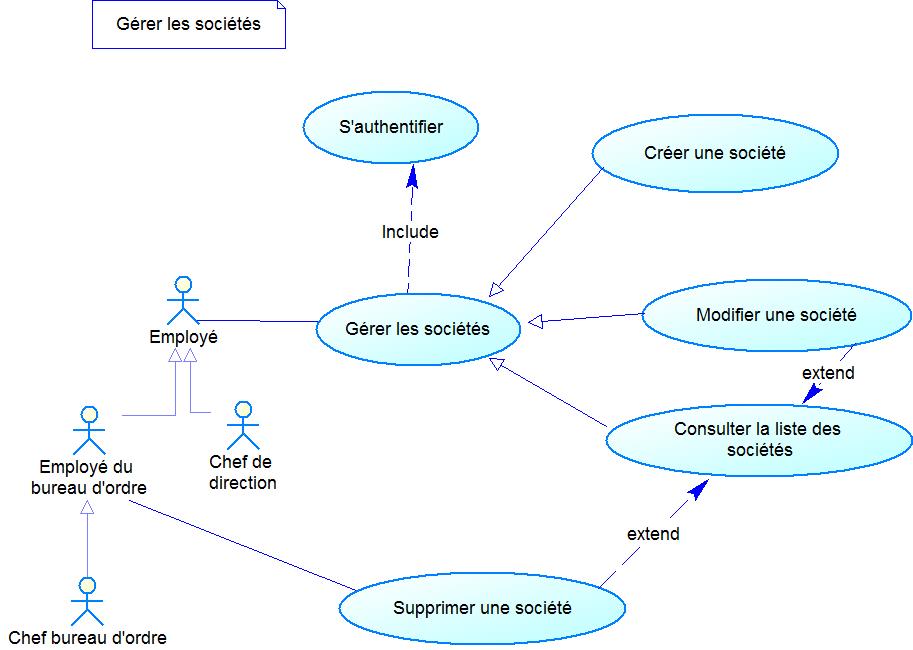
|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Valider un courrier arrivé |
| **Acteur** | Chef bureau d’ordre |
| **Résumé** | Ce cas d’utilisation permet de valider un courrier arrivé |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | Courrier arrivé validé |
| **Scenario nominal** | 1-L’utilisateur demande la liste des courriers arrivés à traiter.  2-La liste des courriers arrivés à traiter s’affiche.  3- L’utilisateur choisi le courrier à traiter.  4- Le courrier s’affiche.  5-L’utilisateur clique sur « valider ».  6-Le système demande à l’utilisateur s’il veut vraiment faire la validation de ce courrier.  7-L’utilisateur répond par « OUI »  8-Un message de succès de validation de courrier s’affiche. |
| **Scénario alternatif** | A7 : Annuler la validation du courrier.  7-L’utilisateur répond par « ANNULER ».  8- Le scénario reprend de 4. |

**Tableau 4: Valider un courrier arrivé**

|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Refuser un courrier arrivé |
| **Acteur** | Chef bureau d’ordre |
| **Résumé** | Ce cas d’utilisation permet de refuser un courrier arrivé |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | Courrier arrivé refusé |
| **Scenario nominal** | 1-L’utilisateur demande la liste des courriers arrivés à traiter.  2-La liste des courriers arrivés à traiter s’affiche.  3-L’utilisateur choisit le courrier à traiter.  4-Le courrier s’affiche.  5-L’utilisateur saisi le critère de refus.  6-L’utilisateur clique sur « refuser ».  7-Le système demande à l’utilisateur s’il veut vraiment faire le refus de ce courrier.  8-L’utilisateur répond par « OUI ».  9-Un message de succès de refus de courrier s’affiche. |
| **Scénario alternatif** | A8 : Annuler le refus du courrier.  8-L’utilisateur répond par « ANNULER ».  9-Le scénario reprend de 4. |

**Tableau 5: Refuser un courrier arrivé**

* + 1. **Diagramme de cas d’utilisation de la gérance des sociétés**



**Figure 11: Diagramme de cas d’utilisation de gérance des sociétés**

Ce cas d’utilisation permet aux employés de gérer les sociétés avec lesquelles l’entreprise échange des courriers. Il renferme quatre sous cas d’utilisation : créer une société, modifier une société, consulter la liste des sociétés, supprimer une société.

Ce diagramme montre aussi que la suppression d’une société ne peut être faite que par un employé du bureau d’ordre ou chef du bureau d’ordre.

Ce cas d’utilisation doit être aussi précédé par une authentification.

|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Créer une société |
| **Acteur** | N’importe quel employé |
| **Résumé** | Ce cas d’utilisation permet de créer une société |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | Société créée |
| **Scenario nominal** | 1-L’utilisateur demande l’interface de création de société.  2-Un formulaire s’affiche.  3-L’utilisateur entre les informations nécessaires.  4-L’utilisateur clique sur « créer société ».  5-Le système vérifie les données saisies.  6-Le système crée une société.  7-Un message de succès de création de société s’affiche. |
| **Scénario alternatif** | A5 : Données invalides ou champs obligatoires absents.  6-Afficher un message d’erreur.  7- Le scénario reprend de 2. |

**Tableau 6: Créer une société**

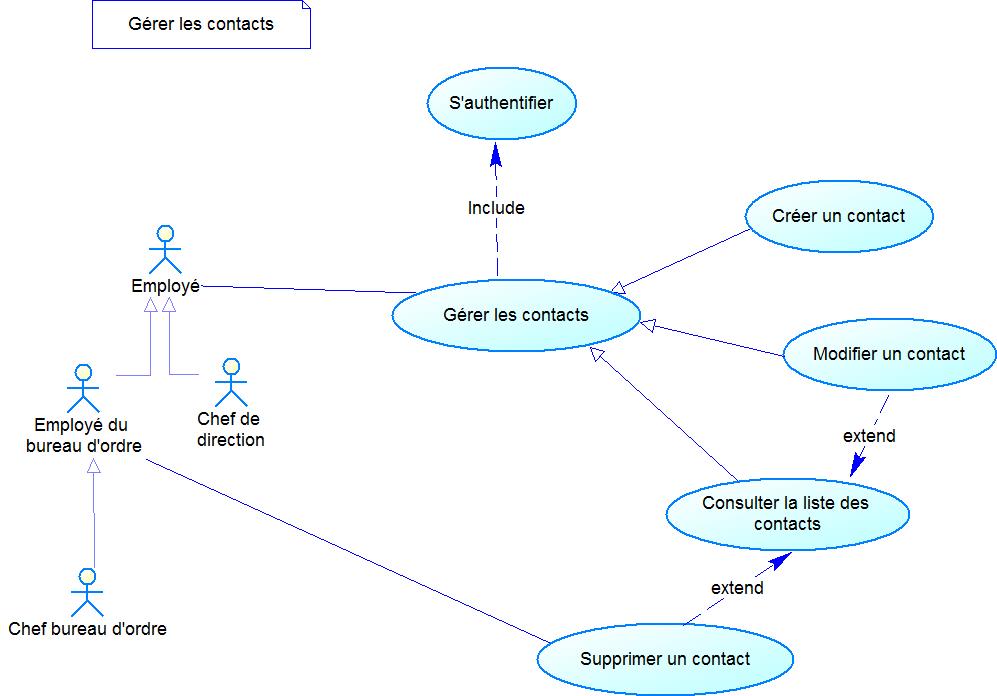
|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Modifier une société |
| **Acteur** | N’importe quel employé |
| **Résumé** | Ce cas d’utilisation permet de modifier les données d’une société |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | société modifiée |
| **Scenario nominal** | 1-L’utulisteur demande la liste des sociétés.  2-La liste des sociétés s’affiche.  3-L’utilisateur choisi la société à modifier.  4-L’utilisateur demande l’interface de modification d’une société.  5-Un formulaire s’affiche chargé par les données courantes.  6-L’utilisateur modifie les informations nécessaires.  7-L’utilisateur clique sur « modifier société ».  8-Le système vérifie les données saisies.  9-Le système modifie la société.  10-Un message de succès de modification de société s’affiche. |
| **Scénario alternatif** | A8 : Données invalides ou champs obligatoires absents.  9-Afficher un message d’erreur.  10-Le scénario reprend de 2. |

**Tableau 7: Modifier une société**

|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Supprimer une société |
| **Acteur** | -Employé bureau d’ordre  -Chef bureau d’ordre |
| **Résumé** | Ce cas d’utilisation permet de supprimer une société du système |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | Société supprimé |
| **Scenario nominal** | 1-L’utilisateur demande la liste des sociétés.  2- La liste des sociétés s’affiche.  3- L’utilisateur choisi la société à supprimer.  4-L’utilisateur clique sur « supprimer ».  5-Le système demande si l’utilisateur veut vraiment supprimer cette société.  6-L’utlisateur répond par « OUI ».  7-Un message de succès de suppression de société s’affiche. |
| **Scénario alternatif** | A5 : Annuler la suppression de la société.  6- L’utilisateur répond par « ANNULER ».  7-Le scénario reprend de 2. |

**Tableau 8: Supprimer une société**

* + 1. **Diagramme de cas d’utilisation de la gérance les contacts**



**Figure 12: Diagramme de cas d’utilisation de la gérance des contacts**

Ce cas d’utilisation permet aux employés de gérer les contacts qui appartiennent aux sociétés avec lesquelles l’entreprise échange des courriers. Il renferme quatre sous cas d’utilisations : créer un contact, modifier un contact, consulter la liste des contacts, et supprimer un contact.

Ce diagramme montre aussi que la suppression d’un contact ne peut être faite que par un employé du bureau d’ordre ou chef du bureau d’ordre.

Ce cas d’utilisation doit être aussi précédé par une authentification.

|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Créer un contact |
| **Acteur** | N’importe quel employé |
| **Résumé** | Ce cas d’utilisation permet de créer un contact |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | Contact créé |
| **Scenario nominal** | 1-L’utilisateur demande l’interface de création de contact.  2-Un formulaire s’affiche.  3-L’utilisateur entre les informations nécessaires.  4-L’utilisateur clique sur « créer contact ».  5-Le système vérifie les données saisies.  6-Le système crée un contact.  7-Un message de succès de création de contact s’affiche. |
| **Scénario alternatif** | A5 : Données invalides ou champs obligatoires absents.  6-Afficher un message d’erreur.  7-Le scénario reprend de 2. |

**Tableau 9: Créer un contact**

|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Modifier un contact |
| **Acteur** | N’importe quel employé |
| **Résumé** | Ce cas d’utilisation permet de modifier les données d’un contact |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | contact modifié |
| **Scenario nominal** | 1-L’utilisateur demande la liste des contacts.  2-La liste des contacts s’affiche.  3-L’utilisateur choisi le contact à modifier.  4-L’utilisateur demande l’interface de modification d’un courrier arrivé.  5-Un formulaire s’affiche chargé par les données courantes.  6- L’utilisateur modifie les informations nécessaires.  7- L’utilisateur clique sur modifier contact.  8-Le système vérifie les données saisies.  9-Le système modifie le contact.  10-Un message de succès de modification de contact s’affiche. |
| **Scénario alternatif** | A8 : Données invalide ou champs obligatoires absents.  9-Afficher un message d’erreur.  10-Le scénario reprend de 2. |

**Tableau 10: Modifier un contact**

|  |  |
| --- | --- |
| **Sommaire** | |
| **Titre** | Supprimer un contact |
| **Acteur** | -Employé bureau d’ordre  -Chef bureau d’ordre |
| **Résumé** | Ce cas d’utilisation permet de supprimer un contact du système |
| **Description des enchaînements** | |
| **Précondition** | Employé authentifié |
| **Post-condition** | contact supprimé |
| **Scenario nominal** | 1-L’utilisateur demande la liste des contacts.  2- La liste des contacts s’affiche.  3-L’utilisateur choisi le contact à supprimer.  4-L’utilisateur clique sur « supprimer ».  5-Le système demande si l’utilisateur veut vraiment supprimer ce contact.  6-L’utlisateur répond par « OUI ».  7-Un message de succès de suppression. de contact s’affiche. |
| **Scénario alternatif** | A5 : Annuler la suppression du contact  6-L’utilisateur répond par « ANNULER »  7-Le scénario reprend de 2 |

**Tableau 11: Supprimer un contact**

1. **Conclusion**

Dans ce chapitre nous avons fixé les besoins fonctionnels et non fonctionnels de notre système. Puis nous avons enchainé avec les diagrammes de cas d'utilisation en présentant les acteurs intervenants du système ainsi que leurs rôles.

Nous prendrons en considération le résultat de cette spécification pour la conception

qui fera l'objet du chapitre suivant. Cette étude préalable couvre l'architecture globale de la solution et la conception des différents niveaux qui composent cette architecture.

**Chapitre 3: Conception**

**Introduction**Après avoir déjà défini les besoins fonctionnels et non fonctionnels et après avoir détaillé les différents cas d’utilisation 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 commençons par exposer l'architecture utilisée en définissant les différentes couches et composantes ainsi que l’architecture du flux de travail et les patrons de conception utilisés 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**

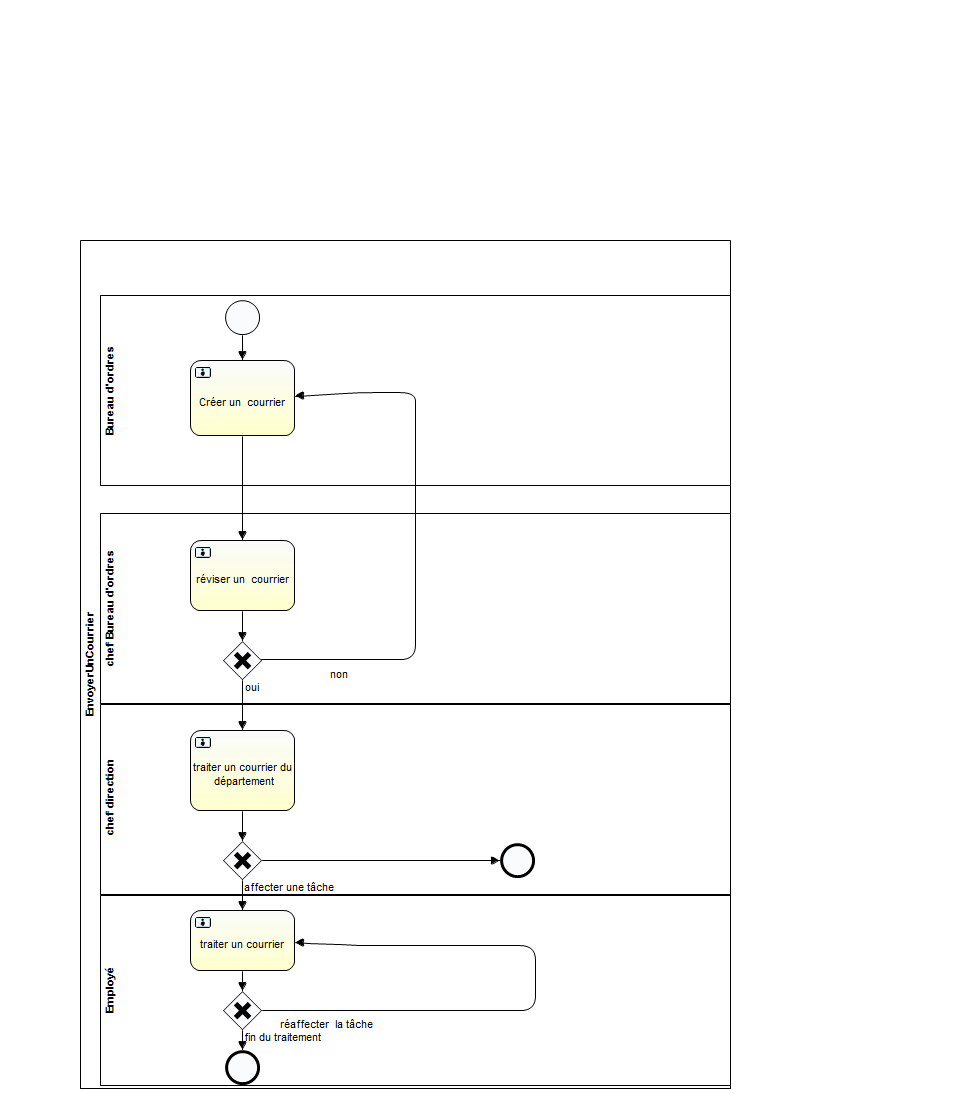
Pour notre premier flux de travail qui est le courrier arrivé :

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 nécessaires en attachant le(s) 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 qui le traitera. Il aura le choix entre 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’autres traitements et ce transfert va boucler jusqu’à ce qu’un employé archive le courrier et met fin à ce flux.

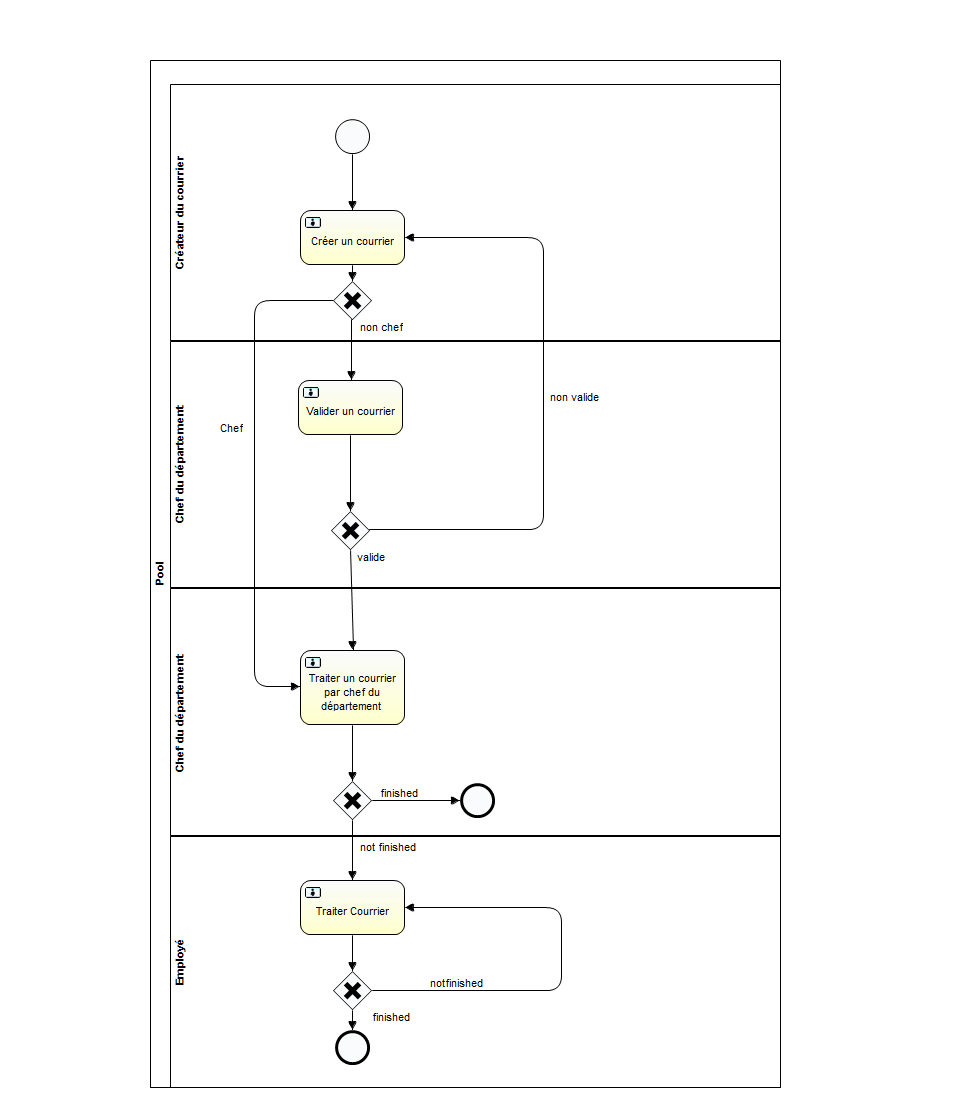
**Figure 13 :** *flux de travail courrier arrivé*

* 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 peut être donc créé par n’importe quel employé dans la société. A la création il y a vérification du statut du créateur :

* S’il est un chef de département ou de bureau d’ordre, il va être envoyé directement vers le chef de la direction destinatrice pour être traité.
* 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 14 :** *Flux de travail courrier interne*

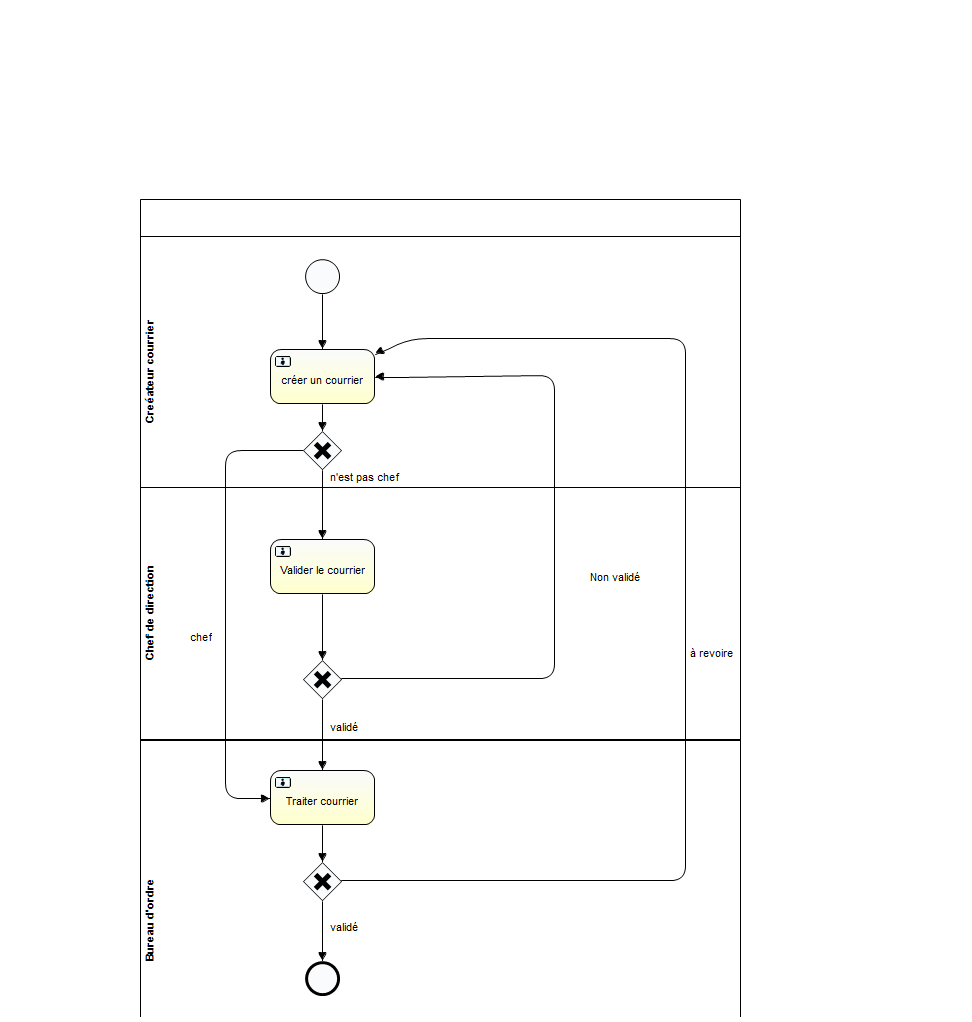
* 1. **Courrier sortant**

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 peut être donc créé par n’importe quel employé dans la société. A 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 traité.
* 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’initiateur du courrier.



**Figure 15 :** *flux de travail courrier sortant*

* 1. **Choix des patrons de conception**

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.

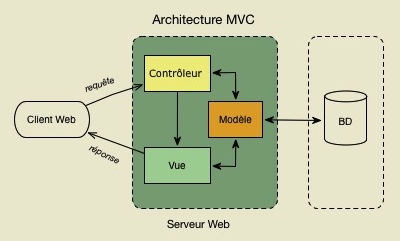
* 1. **Intérêt des patrons de conception**

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 trois différentes parties :

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



**Figure 16 :** *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 toutes 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.

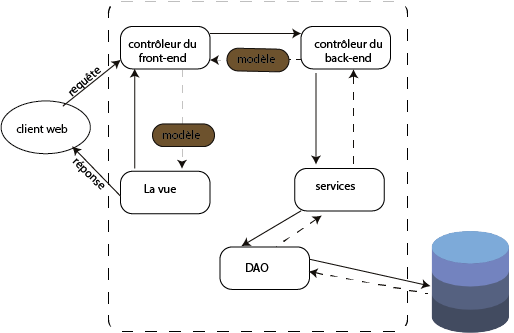
* + 1. **Intérêt du patron MVC**

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 maintenables. 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 utilisée pourisoler la couche de persistance du reste de l’application, il permet essentiellement le découplage entre la couche métier et la couche de persistance.

* + 1. **Les avantages du modèle DAO**
* Il est plus facile de modifier le code qui gère la persistance (changement de SGBD ou même de modèle de données).
* Il est plus facile pour les accès.
* Il est sans doute le modèle de conception le plus utilisé dans le monde de la persistance.
  + 1. **Architecture MVC+DAO de l’application**



***Figure 17****: Architecture MVC + DAO de l’application*

1. **Conception détaillée**

Dans cette partie nous allons mieux détailler l’architecture de la solution proposée 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. **Conception de la base de données**

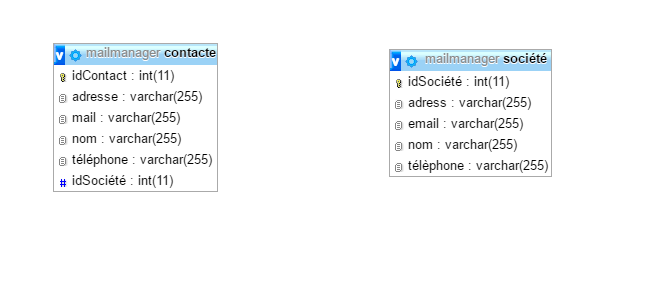
On a choisi la base de données MySQL comme SGBD2 pour stocker les données.

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

* Une base de données pour stocker les informations propriétaires 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 générée 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 toutes les sociétés
* Contact : une table qui contient tous les contacts de ces sociétés.



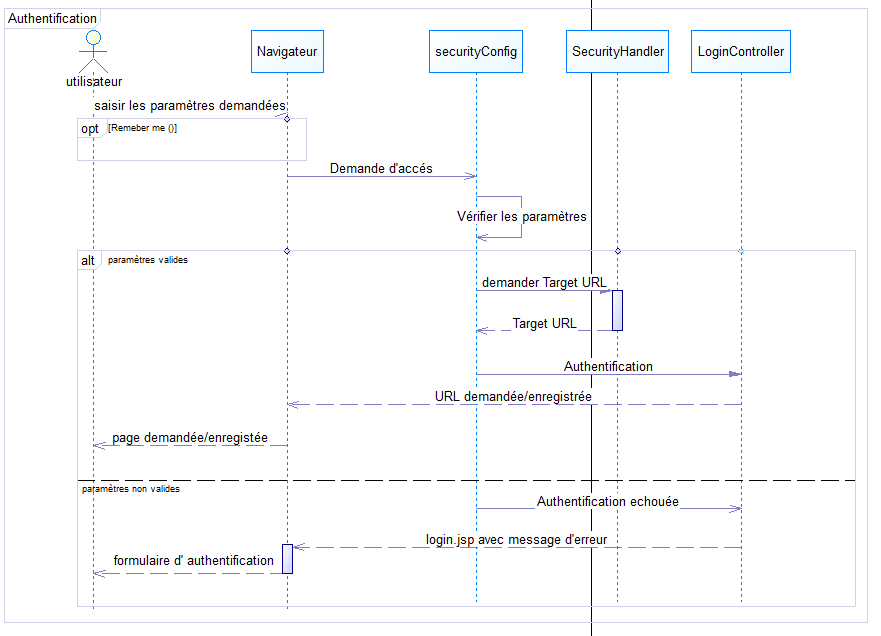
***Figure 18*** *: conception de la base de données MySQL*

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**

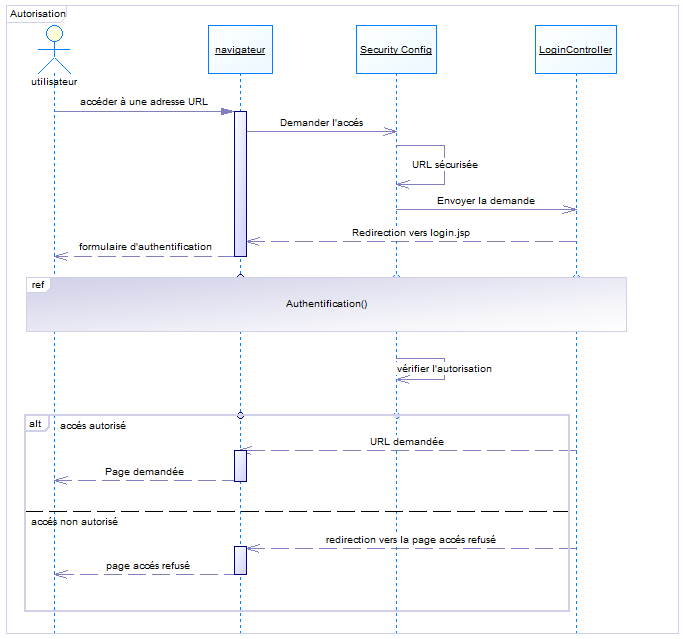
La figure 16 illustre le digramme de séquence « Authentification ». L'utilisateur saisit les paramètres de connexion (la fonctionnalité *Remember-me* est optionnelle). Spring Security vérifie la validité des paramètres saisis par l'utilisateur et refuse l’accès s’ils sont invalides. Sinon une redirection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

 **Figure 19 :** *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 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.

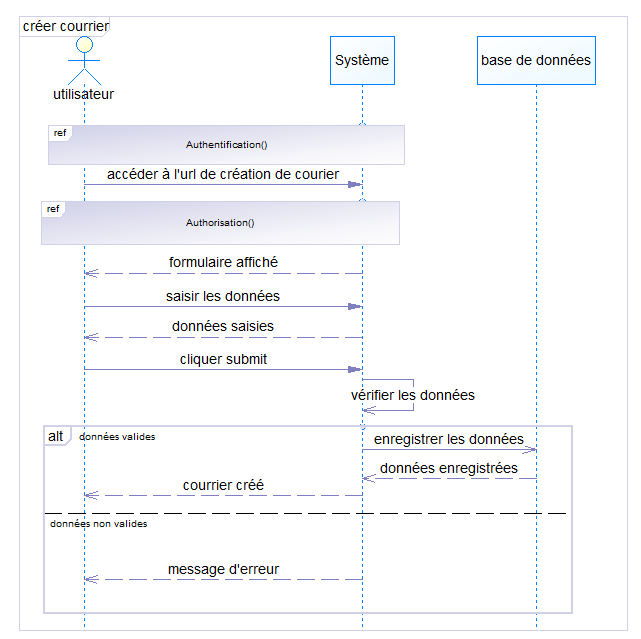


**Figure 20 :** *Diagramme de séquence « Autorisation»*

* 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 saisit les données demandées dans le formulaire et valide les informations saisies.

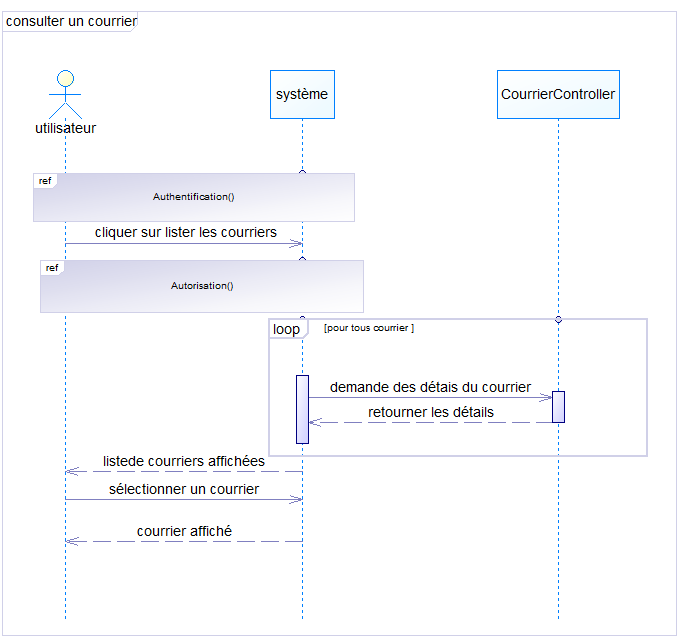


**Figure 21 :** *Diagramme de séquence « créer un courrier »*

* 1. **Consulter les courriers**

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

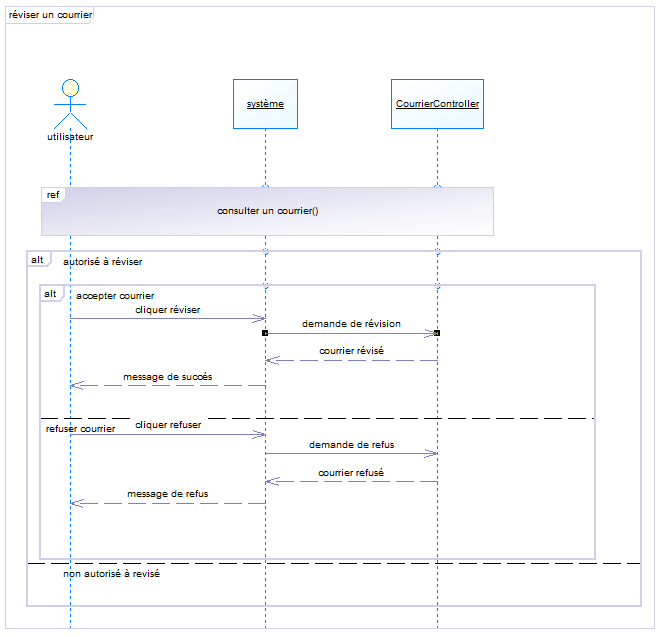
Après l’authentification, l’utilisateur doit cliquer sur « lister tous les courriers » puis sélectionner le courrier pour qu’il consulte ses détails.



**Figure 22 :** *Diagramme de séquence « Consulter un courrier »*

* 1. **Réviser un courrier**

Comme il est illustré dans la figure 24, pour réviser un courrier l’utilisateur doit

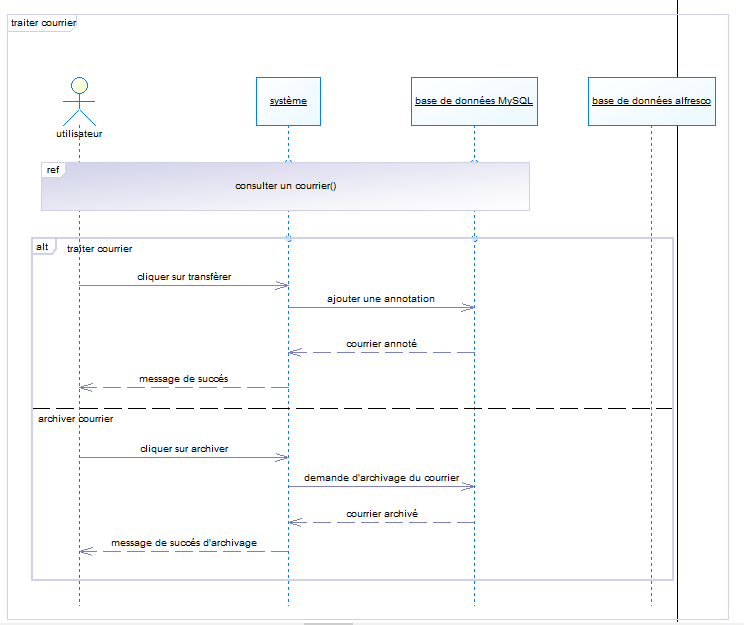
le consulter puis s’il est autorisé à le réviser, alors il a le choix entre valider ou refuser.

**Figure 23 :** *Diagramme de séquence «Valider un courrier»*

* 1. **Traiter un courrier**

Comme c’est illustré par la figure 25, pour traiter un courrier l’utilisateur doit

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



**Figure 24 :** *Diagramme de séquence «traiter un courrier»*

1. **Diagramme de classe**

Le diagramme de classe 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**

**Figure 25 :** *Diagramme de Classe*

**Conclusion**

Dans ce chapitre, nous avons présenté la conception générale et détaillée de notre solution pour faciliter la transition vers la phase de l’implémentation. Dans le chapitre suivant, nous allons exposer le travail réalisé tout au long de ce projet ainsi que l’environnement que nous allons utiliser pour le développer.

**Chapitre 4 : Réalisation**

1. **Introduction**

Tout au long de ce chapitre, et après avoir finalisé la partie conceptuelle de notre application, nous présentons quelques détails qui concernent la réalisation.

Tout d’abord, on va commencer par une présentation de l’environnement matériel et logiciel utilisé. Ensuite on va détailler l’architecture logique et physique. Par la suite, nous finissons par présenter certaines interfaces développées tout en définissant leurs fonctionnalités.

1. **Méthodologie Scrum**
   1. **Présentation de la méthodologie Scrum**

http://www.agiliste.fr/introduction-methodes-agiles/#Approche-Agile-plut-t-que-m-thode-Agile

Lors de notre projet, on a choisi scrum comme méthode agile, il est considéré comme un cadre ou « framework » de gestion de projet. Ce cadre est constitué d'une définition des rôles, de réunions et d'artefacts.

Scrum définit 3 rôles :​

* Le « **Product Owner** » qui porte la vision du produit à réaliser (représentant généralement le client).
* Le « **Scrum Master** » garant de l'application de la méthodologie Scrum.
* **L'équipe de développement** qui réalise le produit.

Lors d’un projet utilisant comme méthode scrum, l’équipe est sensée d’effectuer des réunions fréquentées afin de s’assurer de la continuité du projet et de contrôler la productivité de l’équipe. Ces réunions sont strictement limitées dans le temps (timeboxing).

Différentes réunions dans un projet Scrum :

* **Planification du Sprint**

Au cours de cette réunion, l’équipe sélectionne les éléments prioritaires du « Product Backlog » qu’ils vont réaliser durant ce sprint.

* **Revue de Sprint**

Au cours de cette réunion qui a lieu à la fin du sprint, l'équipe de développement présente les fonctionnalités terminées au cours du sprint et recueille les feedbacks du Product Owner et des utilisateurs finaux. C'est également le moment d'anticiper le périmètre des prochains sprints et d'ajuster au besoin la planification de release (nombre de sprints restants).

* **Rétrospective de Sprint**

La rétrospective qui a généralement lieu après la revue de sprint est l'occasion de s'améliorer (productivité, qualité, efficacité, conditions de travail, etc) à la lueur du "vécu" sur le sprint écoulé (principe d'amélioration continue).

* **Mêlée quotidienne**

Il s'agit d'une réunion de synchronisation de l'équipe de développement qui se fait debout (elle est aussi appelée "stand up meeting") en 15 minutes maximum au cours de laquelle chacun répond principalement à 3 questions : « Qu'est-ce que j'ai terminé depuis la dernière mêlée ? Qu'est-ce que j'aurai terminé d'ici la prochaine mêlée ? Quels obstacles me retardent ? »

Parmi les artefacts scrum on cite :

http://www.thierry-pigot.fr/scrum-en-moins-de-10-minutes/

* **Le product backlog**

Le référentiel des exigences initiales est dressé et hiérarchisé avec le client. Il constitue ce que l’on nomme le product backlog. Il ne doit pas nécessairement contenir toutes les fonctionnalités attendues dès le début du projet. Il va évoluer durant le projet en parallèle avec les besoins du client.

* **User Story**

Les fonctionnalités décrites portent le nom de User Stories et sont décrites en employant la terminologie utilisée par le client.

* 1. **Pourquoi choisir Scrum?**

<https://www.pentalog.fr/notre_demarche/methode-agile-scrum.htm>

Scrum se différencie des autres méthodes de développement par ses avantages qui font de ce procédé une réponse pragmatique aux contraintes actuelles des chefs de produits :

* Méthode itérative et incrémentale
* Maximisation de la coopération
* Augmentation de la communication

1. **Environnement de travail**

Nous présentons dans cette section les environnements matériels et logiciels, utilisés pour le développement de notre application.

* 1. **Environnement matériel**

Notre application a été développée sur nos propres machines, dont leurs caractéristiques techniques et logistiques sont détaillées dans le tableau ci-dessous.

|  |  |  |
| --- | --- | --- |
|  | Machine 1 | Machine 2 |
| Identifiant | Asus 550 XL | Asus |
| Processeur | I5 4200U | I7-4710HQ |
| RAM | 6 Go | 8 Go |
| Disque dur | 500 Go | 1 To |
| Système d’exploitation | Windows 8.1 | Windows 10 |

* 1. **Environnement logiciel**

Dans cette section, on va présenter les logiciels qu’on a utilisé pour la mise en œuvre de notre application.

* + 1. **Outils logiciels**
       1. **Eclipse**



**Figure 26 : Logo eclipse**

Eclipse est un IDE, développé par IBM. Il est gratuit, écrit en Java, extensible par des greffons et multi-plate-formes. C’est un logiciel qui simplifie la tâche de programmation grâce à ses raccourcis et ses aides. Il permet de créer des programmes dans de nombreux langages de programmation tels que :JAVA,C++,PHP…

Toutes les fonctions qu'on peut attendre de ce genre de logiciel sont présentes ou existent sous forme de greffons (coloration syntaxique, complétion, debugger, gestion de projets, intégration aux gestionnaires de versions, …).

https://doc.ubuntu-fr.org/eclipse

http://www.enseignement.polytechnique.fr/informatique/profs/Julien.Cervelle/eclipse/

* + - 1. **PowerAMC**

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 27*** *: Logo PowerAMC*

* + - 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 .

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. **MySQL**



**Figure 28 : Logo MySQL**

MySQL est un SGBD qui permet de stocker les données en masse. Ce système est basé sur SQL et fonctionne sur toutes les plateformes.

Il possède des API pour nombreux langages de programmation notamment c, c++, java, PHP...et bénéficie d’un large public.

http://www.phpsources.org/mysqlsources.html

* + - 1. **Apache Tomcat**



**Figure 29 : Logo Tomcat**

Apache Tomcat est un conteneur web, open source. Il est diffusé sous la licence Apache. Ces qualités lui permettent d’être fréquemment utilisé dans des environnements de production.

https://www.jmdoudoux.fr/java/dej/chap-tomcat.htm#tomcat-12

* + - 1. **GitHub**



**Figure 30 : Logo GitHub**

GitHub est un service en ligne qui permet d'héberger ses repositories de code. GitHub est un outil gratuit pour héberger du code open source, et propose également des plans payants pour les projets de codes privés. C'est le numéro un mondial et il héberge plus d'une dizaine de millions de repositories.

https://openclassrooms.com/courses/gerer-son-code-avec-git-et-github/github-qu-est-ce-que-c-est

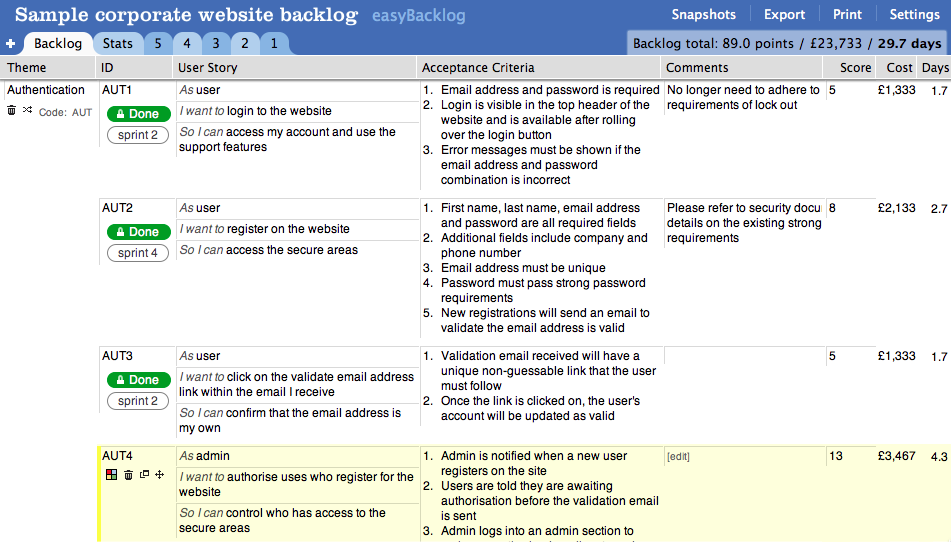
* + - 1. **EasyBacklog**



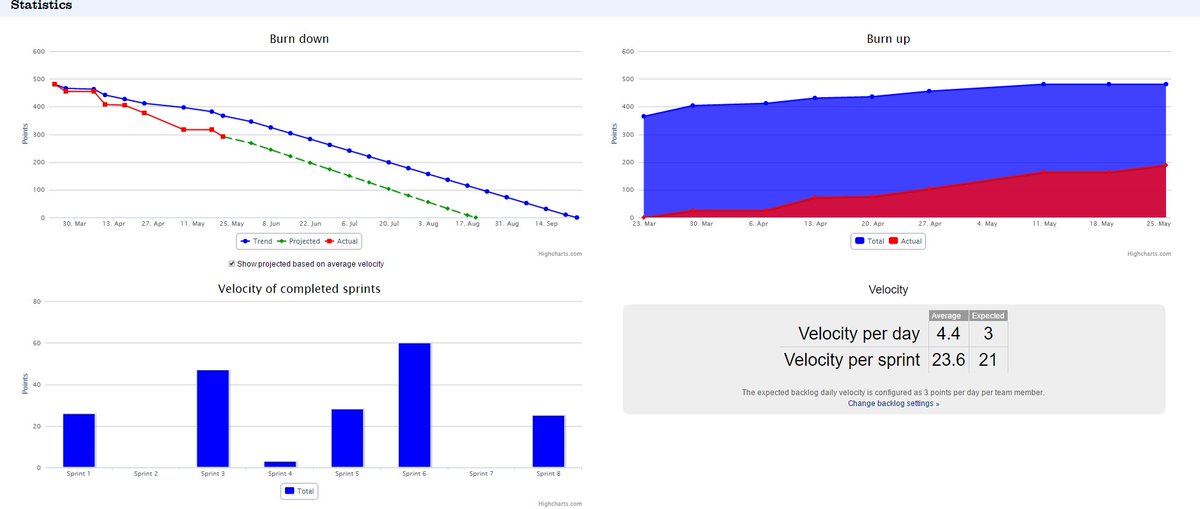
**Figure 31 : Logo EasyBacklog**

EasyBacklog est un outil web, gratuit, utilisé lors la méthodologie scrum. Il permet de créer des backlogs tout en offrant la possibilité de gérer les sprints et de les faire affecter des user stories. Il permet aussi de contrôler le progrès grâce au down chart. Il permet aussi la gestion des utilisateurs et de leurs affecter des tâches.

http://agile-tools.net/agileprojectmanagement/easybacklog.aspx



**Figure 32 : Exemple de backlog**



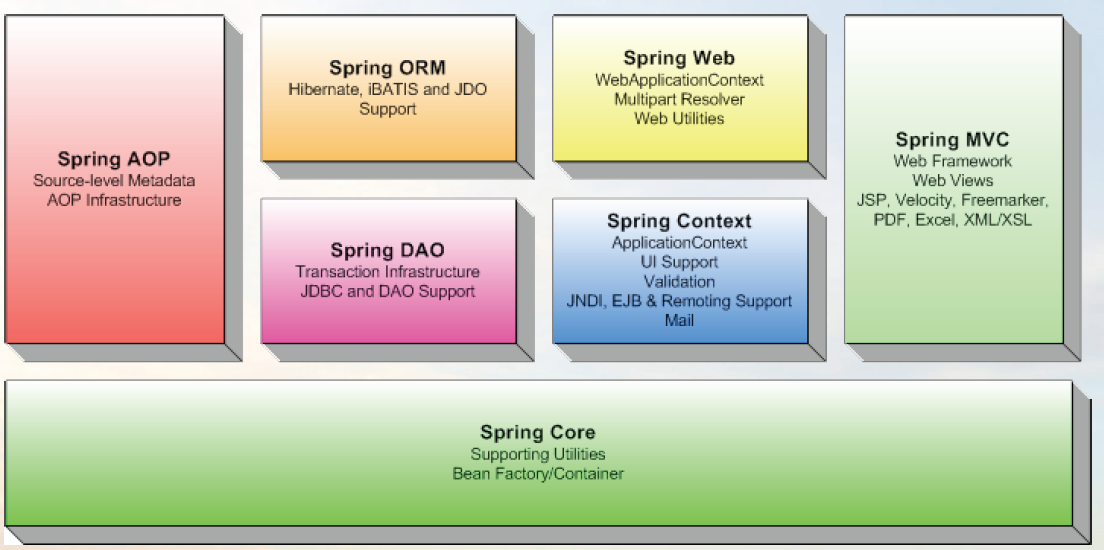
**Figure 33 : Outils statistiques de EasyBacklog**

* + 1. **Framework et choix technique**
       1. **Spring**
          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 plateforme 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 34:** *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**

"Representational **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 du XML, du 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.

* + - 1. **Hibernate**



**Figure 35 : Logo Hibernate**

Hibernate est un framework de persistance, open source, léger et ORM. Il a été lancé en 2001 par Gaving Kim.Il est utilisé pour s’occuper de la persistance des objets java dans la base de données. On peut l’utiliser soit pour des applications web soit pour des clients lourds.<http://www.opentuto.com/quest-ce-que-le-framework-hibernate/>

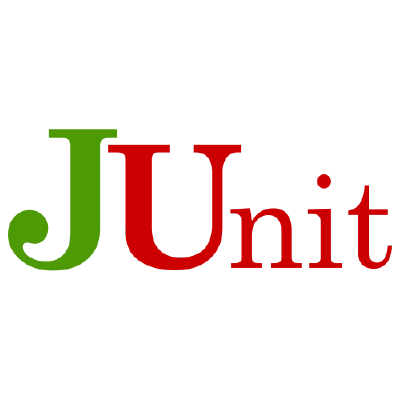
 ***Figure 36 : Architecture d’Hibernate***

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

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

**Figure 37:** *Logo maven*

* + - 1. **JUnit**



**Figure 38 : Logo JUnit**

https://www.jmdoudoux.fr/java/dej/chap-junit.htm

Afin de s’assurer du bon fonctionnement de notre système, on a testé les services de notre application via des tests unitaires assurés par le framework JUnit. Il s’agit d’un framework open-source pour le développement et l’exécution des tests unitaires automatisables.

Il a été développé par Erich Gamma et Kent Beck.

JUnit propose :

* Un framework pour le développement des tests unitaires.
* Des applications pour l’exécution des tests.

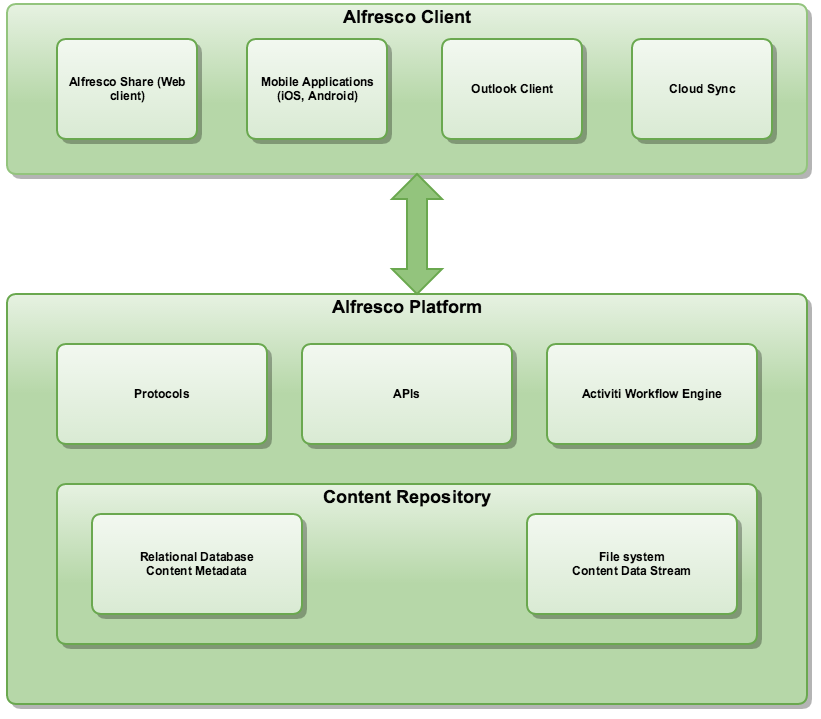
* + - 1. **Alfresco**



**Figure 39: Logo Alfresco**

Alfresco est à la base un dépôt de contenus. Il possède aussi des services pour les applications de contenus. Il assure la persistance des documents et d’autres contenus dont leurs métadonnées seront stockées dans une base de données relationnelles. En plus des fonctionnalités basiques de la gestion des documents, alfresco offre plusieurs services liés au contenus, citons par exemple des services pour les dossiers, services de versionning. Il y a aussi des services de haut niveau comme service Thumb mail(),service des sites pour la création et la gestion des sites dans l’application share,service de tagging ;qui permet de tagger le contenu avec des mots-clés. Ces services sont implémentés en java et exposent des API.

Le serveur alfresco est extensible. On peut faire l’extension en java, javascript, et FreeMaker. En plus, il permet de créer des workflows par Activiti BPMN.



**Figure 40 : Architecture Alfresco**

**http://docs.alfresco.com/5.1/concepts/dev-arch-overview.html**

**http://docs.alfresco.com/5.0/concepts/dev-arch-overview.html**

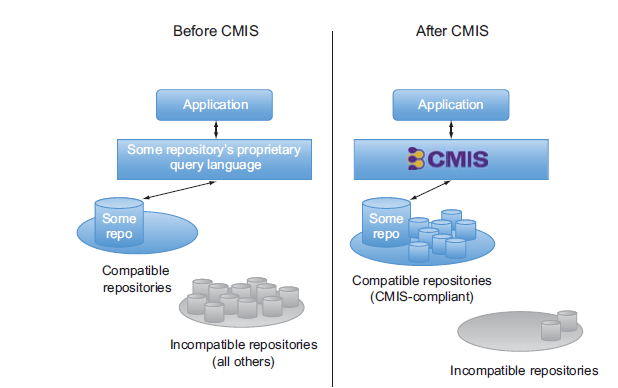
* + - 1. **Cmis**

Le standard cmis assure l’interconnexion d’une GED open source et un CMS. Il permet d’homogénéiser l’accès à une gestion du contenu. Il s’agit d’un modèle extensible basé sur 4 types : document, fichier, relation et politique appliqués au document. Il est basé sur des requêtes SQL.

Plusieurs raisons nous ont poussé à choisir CMIS comme solution pour communiquer avec ALFRESCO,parmi lesquels on cite :

* Pértinité
* Affaiblissement de vendor lock in
* Meilleure intégration

**https://www.pulsar-informatique.com/services/ged-open-source/modalites-d-acces-d-une-ged/interoperabilite-et-ged/standard-cmis**

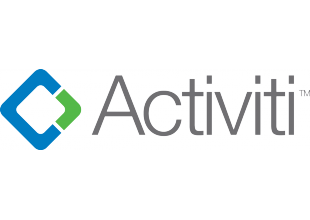


**Figure 41 : Cmis et la compatibilité**

Grâce au CMIS, les développeurs peuvent créer des solutions qui communiquent avec plusieurs dépôts avec la minimisation de vendor lock-in et dont le coût est moins cher.

*CMIS and Apache Chemistry in Action*

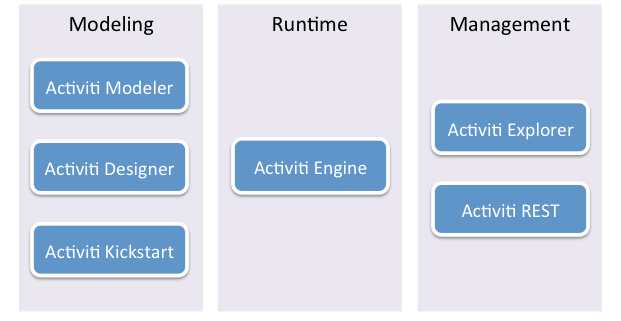
* + - 1. **Activiti**



**Figure 42 : Logo Activiti**

Activiti est une solution open source de gestion des processus métiers (Business Process Management – BPM) et de workflows implémentant la norme BPMN 2.0. Activiti peut être utilisé pour créer des flux de données complexes complètement intégrés à Alfresco. Sa première version est sortie en 2010.Il est facile à utiliser pour les développeurs JAVA.

Voici les différents composants Activiti se combinant pour former une solution complète de BPM dans un contexte complet de développement de logiciels.



**Figure 43 : Composants d'Activiti**

* **Activiti Engine** : c’est un moteur de processus java fonctionnant nativement en BPMN2.
* **Activiti Explorer** : il s’agit d’une application web qui donne accès à l’exécution du moteur Activiti. Elle offre la possibilité de créer et gérer les processus.
* **Activiti Modeler** : il peut être utilisé pour créer des processus BPMN 2.0 graphiquement à l’aide d’un navigateur.
* **Activiti Designer** : il s’agit d’un plugin Eclipse qui permet de modéliser les processus BPMN 2.0 à partir de votre environnement IDE. Il supporte également les extensions spécifiques à une activité pour vous permettre d’utiliser le plein potentiel de vos processus et du moteur.
* http://www.starxpert.fr/bonita/
  + - 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 44:** *Logo Bootstrap*

* + 1. **Technologies**
       1. **JEE**

JEE est l’acronyme de Java Enterprise Edition, il s’agit d’une spécification Java, destinée aux applications d’entreprises. Cette plate-forme possède plusieurs avantages à ces utilisateurs, parmi lesquels on cite : la portabilité, l’indépendance, la multitude de librairies, la sécurité.

* + - 1. **JSON**

JSON (JavaScript Object Notation – Notation Objet issue de JavaScript) est un format léger d'échange de données. Il est facile à lire ou à écrire pour des humains. Il est aisément analysable ou généré par des machines. Il est basé sur un sous-ensemble du langage de programmation JavaScript.

<http://www.json.org/jsonfr.html>

* + - 1. **Ajax**

Ajax est l’acronyme de Asynchronous Javascript And XML.Il ne s’agit pas d’une technologie, mais d’un ensemble de technologies comme : JavaScript, XML, JSON, HTML, et CSS). Il permet la mise à jour rapide du contenu d’une page web sans charger toute la page.

<http://www.chiny.me/ajax-c-est-quoi-10-1.php>

* + - 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 45:** *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 [coté 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 46 :** *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**

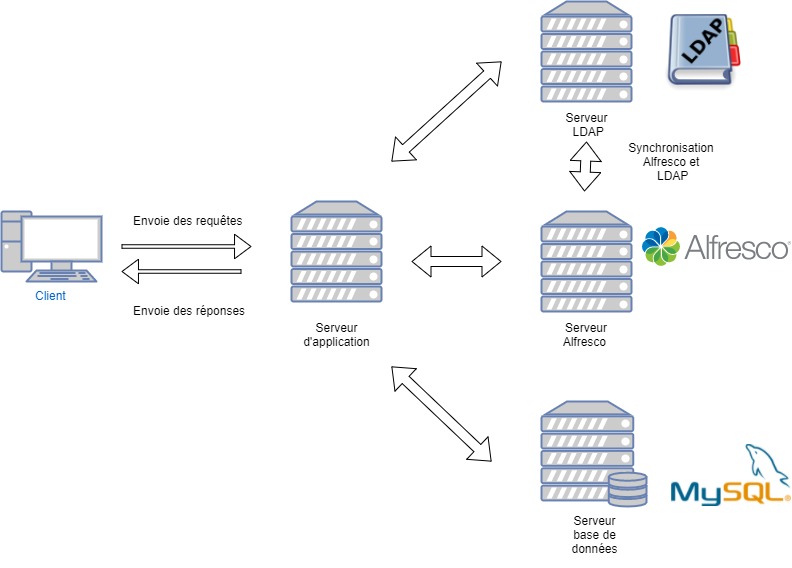
**JQuery** est une bibliothèque logicielle, rapide et concise conçue pour les programmeurs, JavaScriptopen-source inventée par John Resig en 2006, elle permet de se simplifier la vie à moindre coût pour tout ce qui est manipulation d'une page HTML.

**Figure 47 :** *Logo HTML5*

1. **Architecture de l’application**

Dans cette partie on va présenter l’architecture physique et logique de notre application. On va tout d’abord commencer par l’architecture physique qui nous donne une idée sur le partitionnement des matériels, puis on présente l’architecture logique de l’application avec ces différentes couches.

* 1. **Architecture Physique**



**Figure 48 : Architecture physique**

* **Niveau client :** Il s’agit de l’interface entre l’utilisateur et le serveur d’application. L’utilisateur va l’utiliser pour visualiser les résultats des requêtes envoyées à ce dernier.
* **Niveau serveur d’application :** il a comme rôle de traiter les requêtes venant du côté client.il communique aussi avec les serveurs de persistance : comme MySQL, alfresco et LDAP.
* **Serveur base de données :** persiste les données venant du serveur d’application et les fournit en cas de besoin.
* **Serveur Alfresco :** ce serveur a comme rôle de gérer les documents.
* **Serveur LDAP :** fournit les différents utilisateurs qui vont se connecter à l’application.  
  Une synchronisation est établie entre les utilisateurs d’Alfresco et LDAP pour des raisons de sécurité des courriers au niveau d’Alfresco.
  1. **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 implémenté 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, l’utilisation du framework Spring Security nous permet de 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 des informations stockées dans un annuaire LDAP.
* L’autorisation : il s’agit de limiter ou restreindre l’accès à une URL aux utilisateurs d’une certaines catégories ou groupes disposant de certains droits
* **Couche de présentation**

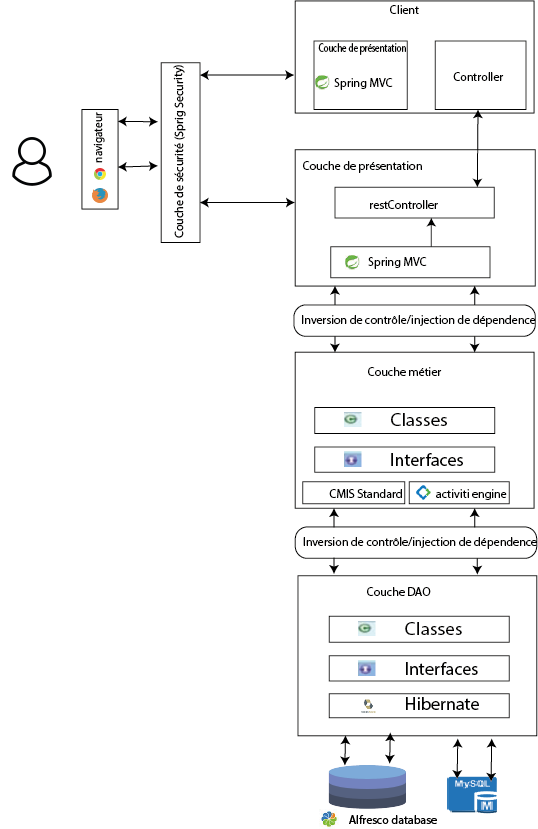
Elle correspond à la partie de l'application visible et interactive avec les utilisateurs.

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.



**Figure 49 : Architecture logique**

1. **Les releases**
   1. **Release 1 : Configuration**

Avant de commencer la phase de développement, on a tout d’abord passé par une phase de configuration durant laquelle on a configuré nos serveurs.

* + 1. **Sprint 1 : Configuration du serveur LDAP**

Durant ce sprint on a installé apache directory studio et on a suivi la même hiérarchie du serveur LDAP de l’organisme d’accueil.

* + 1. **Synchronisation d’Alfresco avec LDAP**

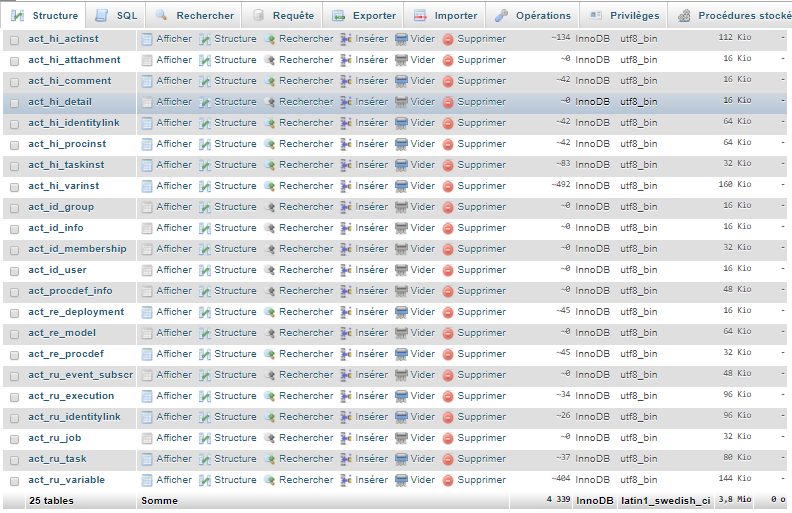
Durant ce sprint on a configuré Alfresco pour qu’il synchronise ses groupes et ses utilisateurs à ceux de LDAP.Pour cela on a créé des scripts de configuration. On a aussi fait des tests pour s’assurer des politiques des documents affectés à des utilisateurs ou des groupes LDAP.

* + 1. **Sprint 3: Synchronisation entre activiti et LDAP**

De même pour activiti, on a fait la configuration du moteur de workflow activiti pour qu’il prenne LDAP comme système d’identification. Puis on a fait les tests nécessaires pour savoir si ça a marché ou non, en lançant des processus activiti et en testant leur déroulement.

* + 1. **Sprint 4 : configuration de la base de données activiti**

Vu que activiti fonctionne par défaut sur une base de données h2 qui est une base de données in memory ce qui signifie la perte d’information en cas de fermeture du serveur. Pour cela on a configuré activiti sur une base de données MySQL dans laquelle le moteur de workflow va générer automatiquement ces tables où il stocke les informations des processus.



**Figure 50: Base de données d'Activiti**

* 1. **Release 2** 
     1. **Sprint 1 : Implémentation du patron DAO pour toutes les entités**

Durant ce sprint, on a fait l’implémentation du patron DAO de toutes les entités et puis on a fait les tests nécessaires.

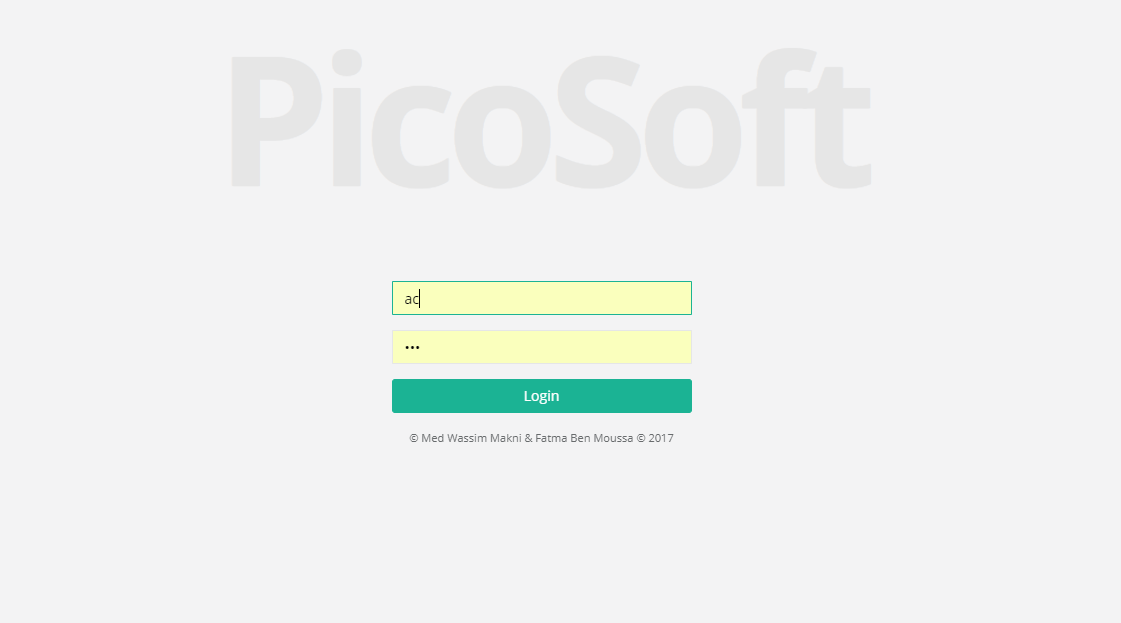
* + 1. **Authentification et contrôle d’accès**

Le processus d’authentification et de contrôle d’accès est un processus majeur dans notre application puisqu’en fonction du rôle du personne authentifié chaque page a ses spécifications et ses données à afficher.

A chaque fois que l’utilisateur ouvre une page, le système de sécurité vérifie son authentification et identifie son rôle (la direction où il appartient) dans l’annuaire LDAP.

La figure 38 représente l'interface de l'authentification.

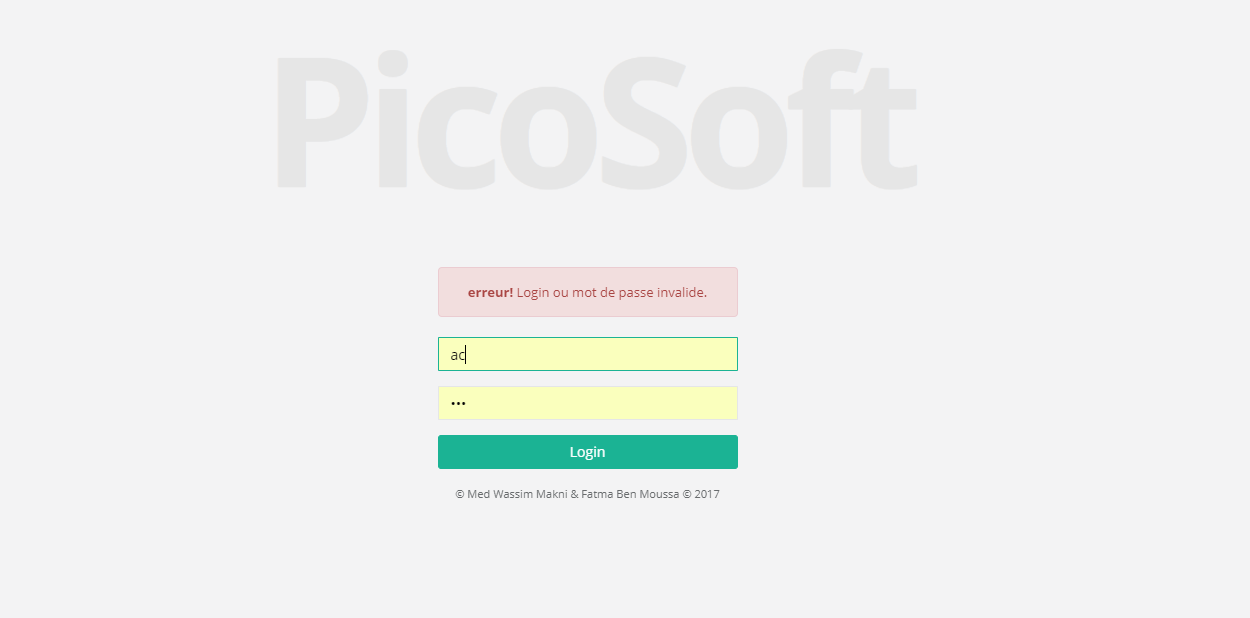
L'utilisateur doit d’abord saisir son login et mot de passe puis il valide en cliquant sur le bouton « Se connecter ».



**Figure 51: interface de Login**

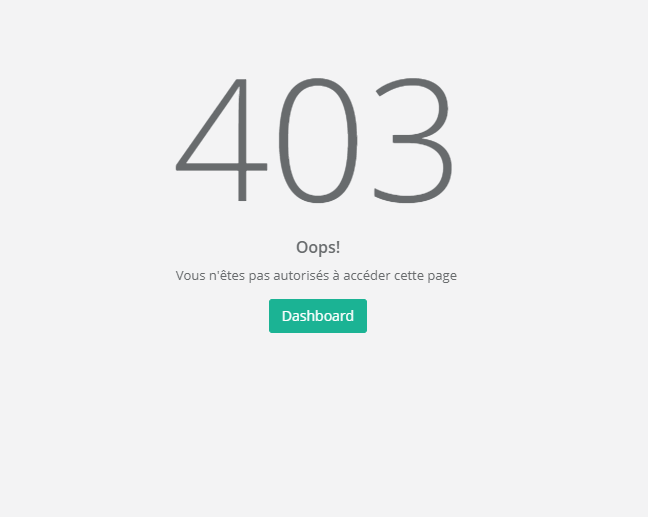
Ainsi le système va vérifier la correspondance des données avec celle dans la base de données :

* En cas de réussite, l’utilisateur sera redirigé vers son interface d’accueil
* En cas d’échec, un message d'erreur s'affichera comme l’indique la figure 39.



**Figure 52: interface de Login en cas d’échec**

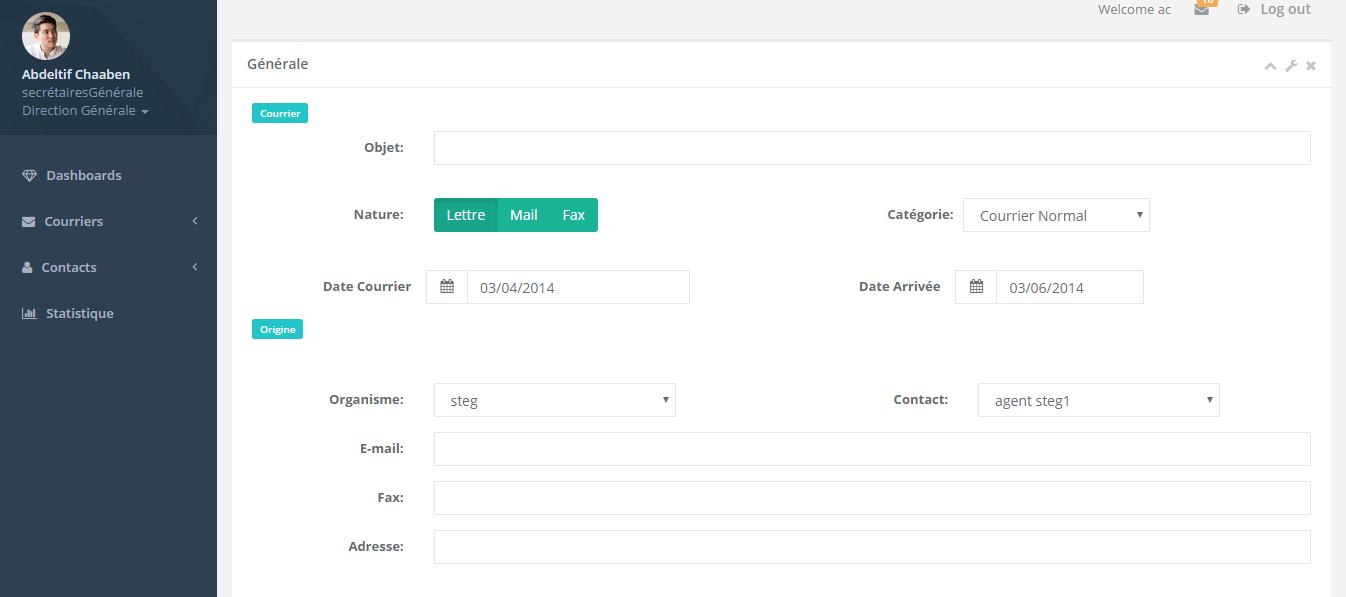
Dans le cas où l’utilisateur n’a pas le droit d’accéder à une page, une page 403 sera affichée. (Figure 42). La page 403 représente le code de réponse du protocole http et signifie que le serveur a compris la requête du client, mais refuse de l’exécuter.

 **Figure 53: Page 403**

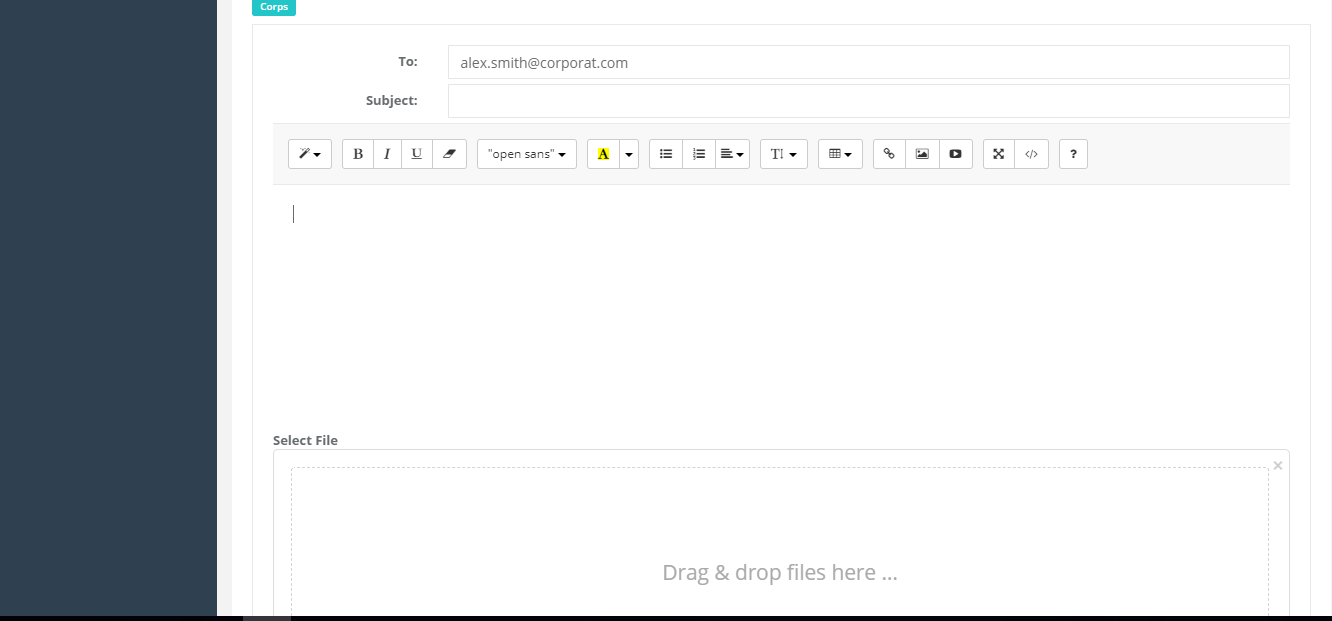
* + 1. **Sprint 3: Gestion des courriers arrivés**

Toute au long de ce sprint, on a fait le design du workflows du processus courrier arrivé et la préparation de sa partie frontale. Puis on a fait l’implémentation de leurs services qui s’est suitée par des tests unitaires.

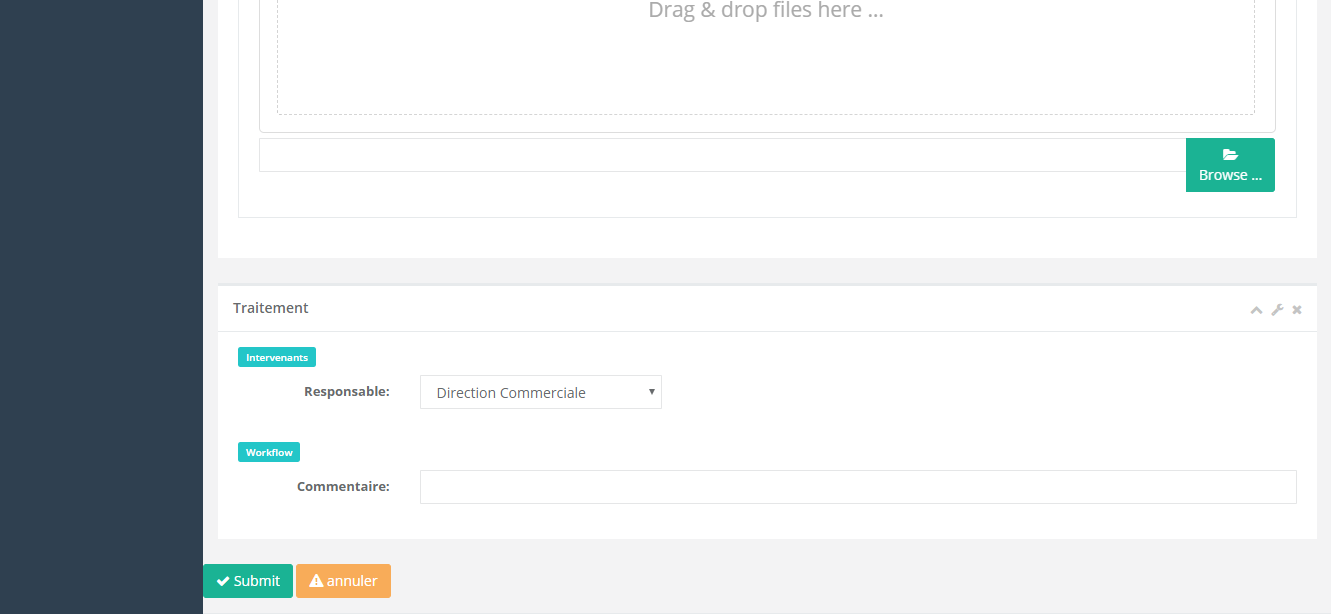
Voici quelques interfaces montrant la gestion des courriers arrivés dans notre application.



**Figure 54: Interface de création d'un courrier arrivé**



**Figure 55 : Interface de création d'un courrier arrivé**



**Figure 56 : Interface de création d'un courrier arrivé**

 En cliquant sur « Créer un courrier arrivé », l’interface ci-dessus s’affiche. Elle est divisée en 3 parties :

* Une partie qui concerne des informations du courrier : dans cette partie, l’utilisateur doit obligatoirement préciser : l’objet du courrier, sa nature, sa catégorie, sa date, sa date d’arrivé.
* Une partie qui concerne l’origine du courrier : dans cette partie, l’utilisateur entre les informations qui concerne l’origine du courrier : nom de société, contact.

L’Email, l’adresse et le fax seront automatiquement remplis selon le contact.

* Une partie qui concerne le traitement du courrier : l’utilisateur doit préciser à quelle direction le courrier va être envoyé. Il peut aussi ajouter un commentaire.

En cliquant sur « Enregistrer » le système vérifie les champs saisis. En cas d’erreur un message d’erreur s’affiche, sinon un message de succès de création du courrier s’affiche et le courrier s’ajoute.

Interface succès

Interface erreur

 En cliquant sur « Courriers arrivés », la liste des courriers arrivés s’affiche.

L’utilisateur peut modifier,supprimer ou filtrer les courriers.

En cliquant sur modifier, l’interface de modification s’affiche.

Interface modification

L’utilisateur change les informations du courrier et clique sur « Enregistrer ».

En cliquant sur « Enregistrer » le système vérifie les champs saisis. En cas d’erreur, un message d’erreur s’affiche, sinon un message de succès de création du courrier s’affiche en et le courrier sera modifié.

 Interface succès

Interface erreur

Si l’utilisateur clique sur « Supprimer », un message s’affiche, demandant, s’il veut vraiment supprimer ce courrier.

Interface du message demande de confirmation

 S’il clique sur « OUI », le message sera supprimé et un message de succès de suppression s’affiche.

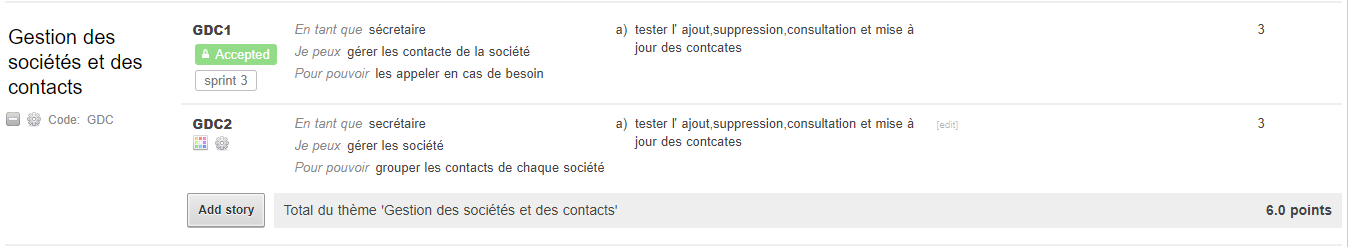
Interface du message succès suppression

S’il clique sur « ANNULER », un message d’annulation de suppression de courrier s’affiche.

Interface du message annulation de suppression

* 1. **Release 3** 
     1. **Sprint 1 : Gestion des contacts et des sociétés**

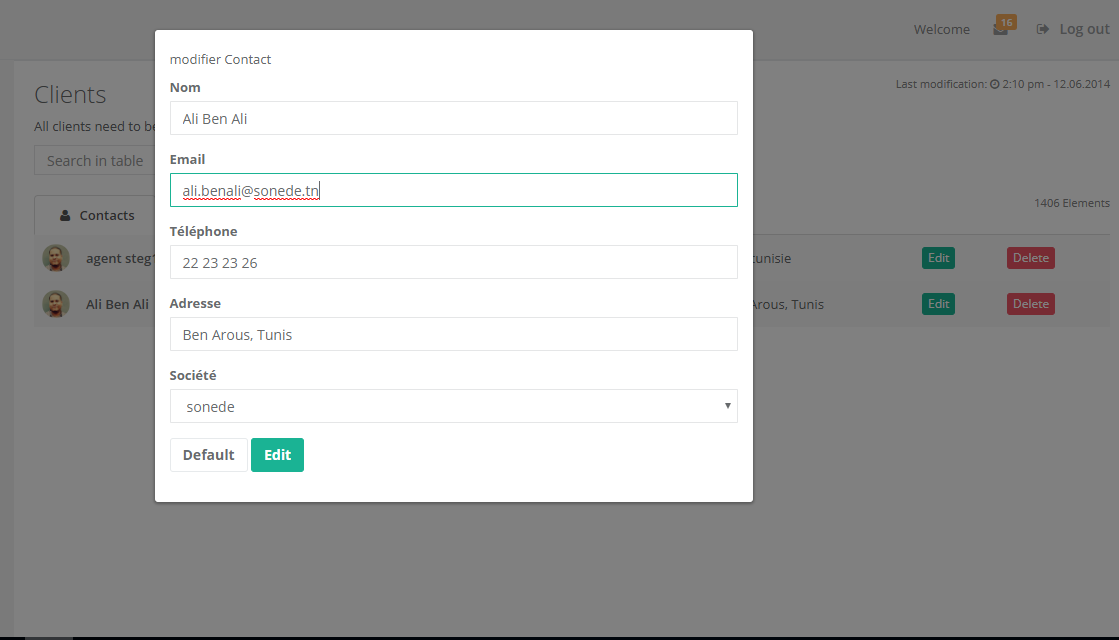
La figure suivant illustre le Backlog de l’administrateur pour le premier sprint du troisième « release ».



La figure 44 décrit l’interface de consultation des contacts et des sociétés. Un tableau contenant deux tabulations (l’une pour la liste des contacts et l’autre pour la liste des sociétés) sera affichée.

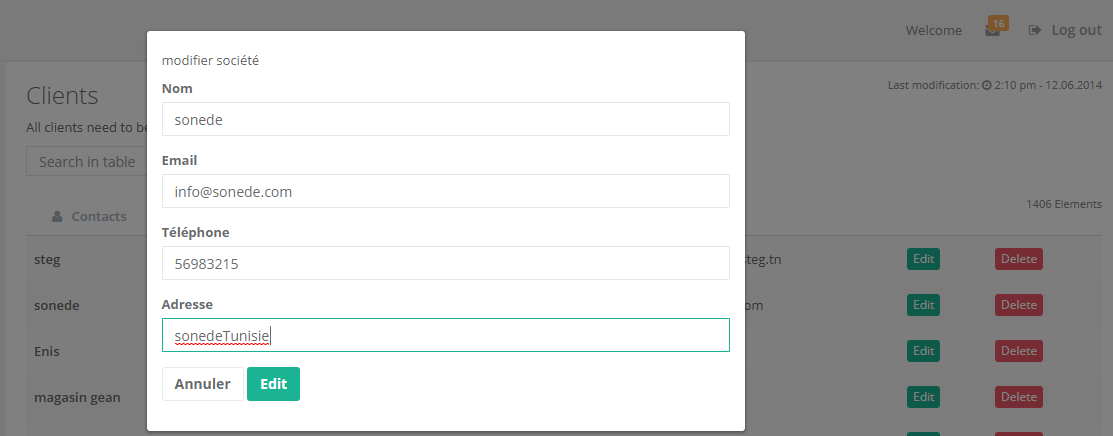
// à compléter

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

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

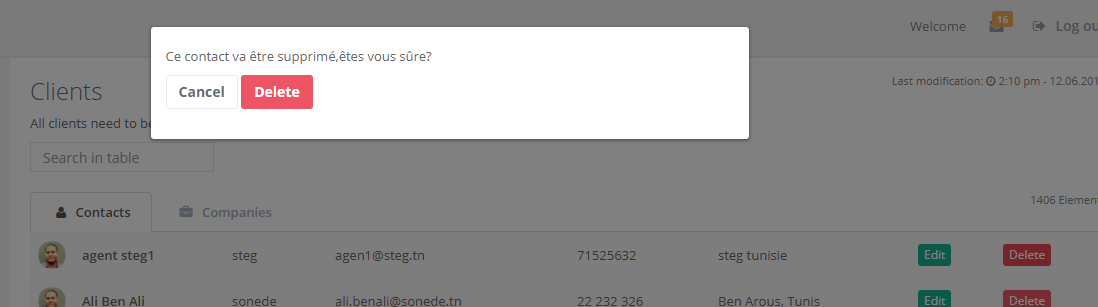
* ***Interface d’édition d’une société*** :

L’appui sur le bouton modifier de l’interface de consultation des sociétés de la figure 44 permettent d’ouvrir un modèle. Ce modèle renvoi un formulaire remplit par les informations relatives à la société sélectionnée.



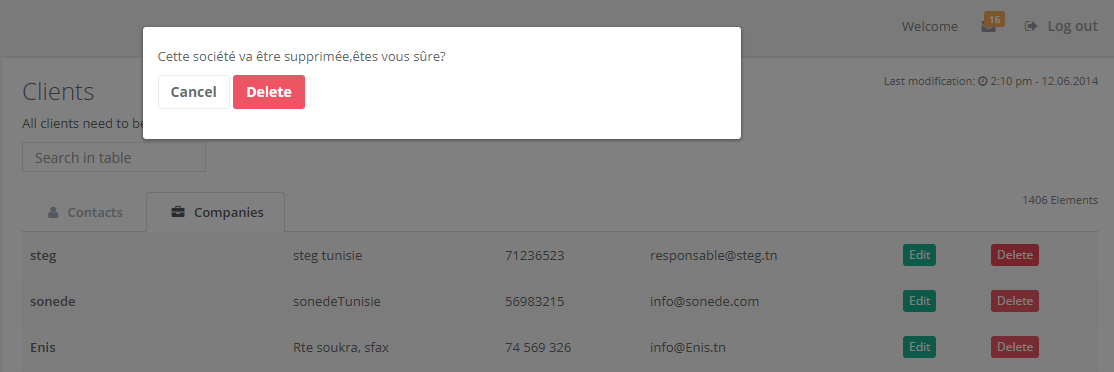
* ***Interface de suppression d’un contact*** :

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

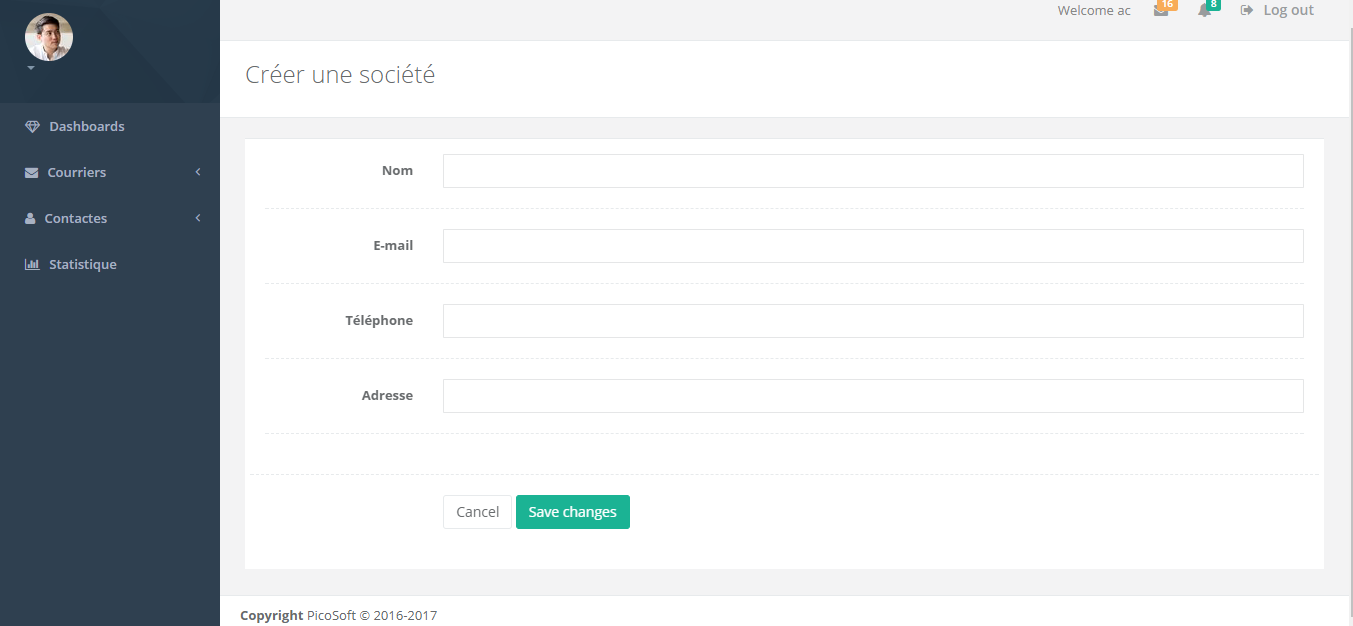


* ***Interface de suppression d’une société*** :

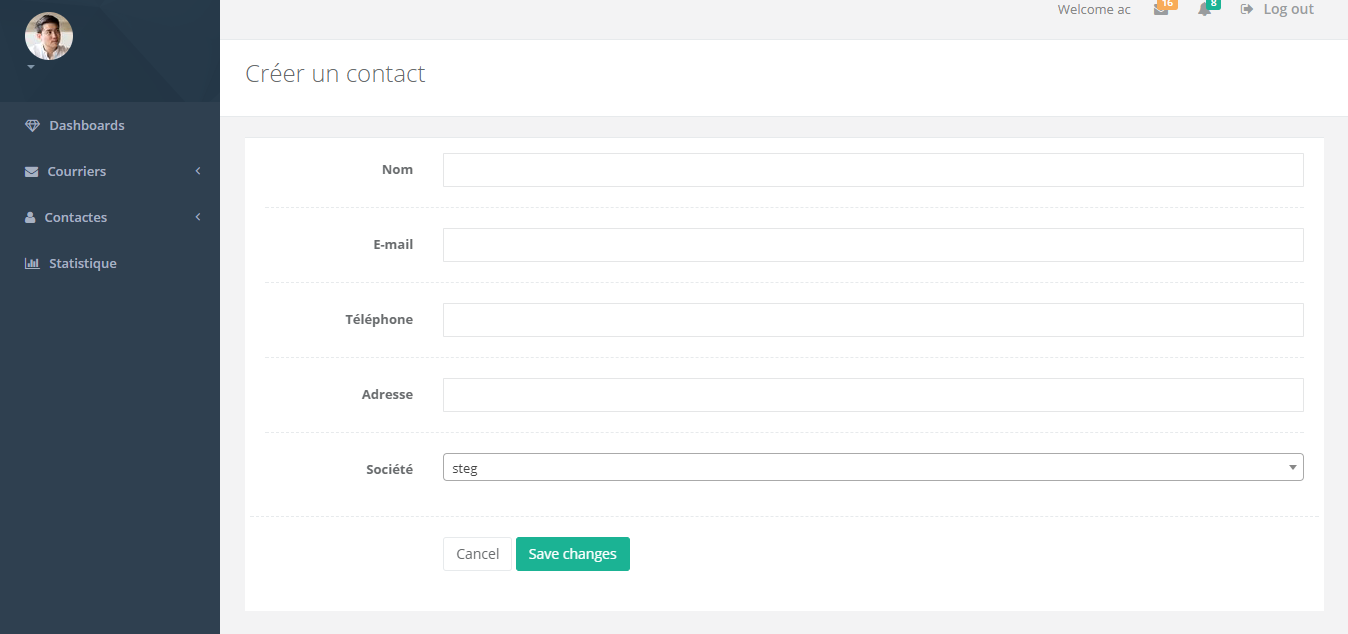
Avant de supprimer un contact il faut bien vérifier qu’il est sûre de cette action .



La figure 44 décrit l’interface de création d’une société.



La figure 44 décrit l’interface de création d’un contact.

Dans notre approche,Chaque contact doit appartenir à une société 

* + 1. **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 de nouveau courrier interne.

Après avoir remplir toutes les informations nécessaires, l’utilisateur clique sur le bouton « envoyer », le système vérifie ainsi la validité des champs

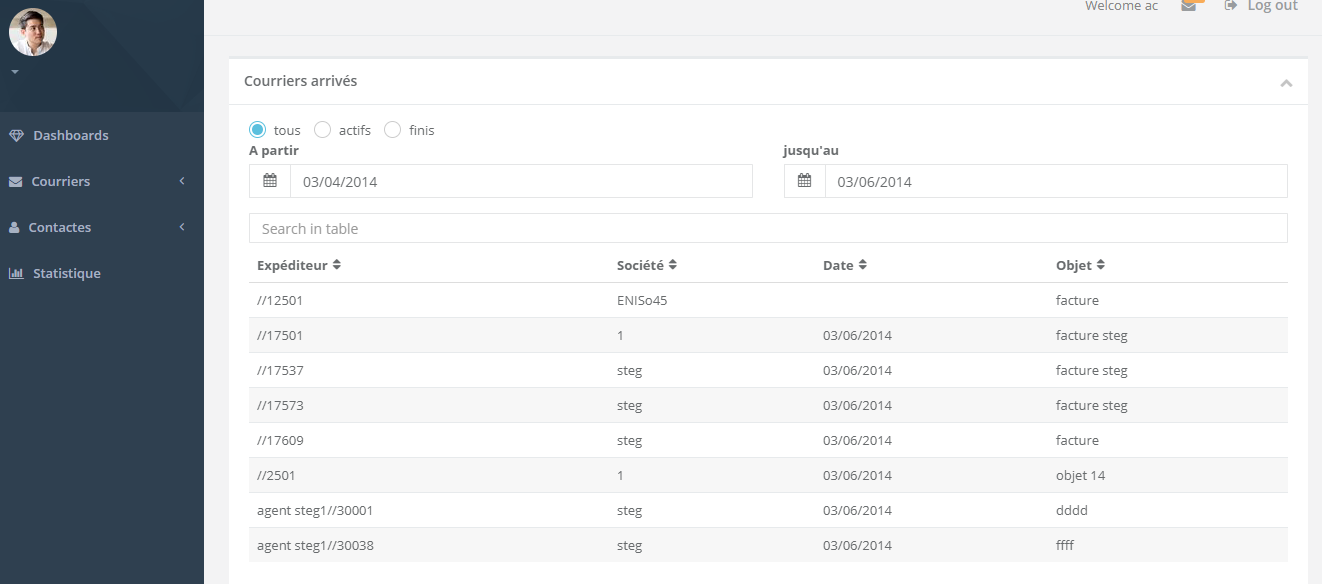
Si tous les paramètres sont valides il va y avoir création d’un nouveau courrier interne et tous les paramètres saisis deviennent alors des paramètres de processus stockés dans la base de données du moteur de workflow activiti.

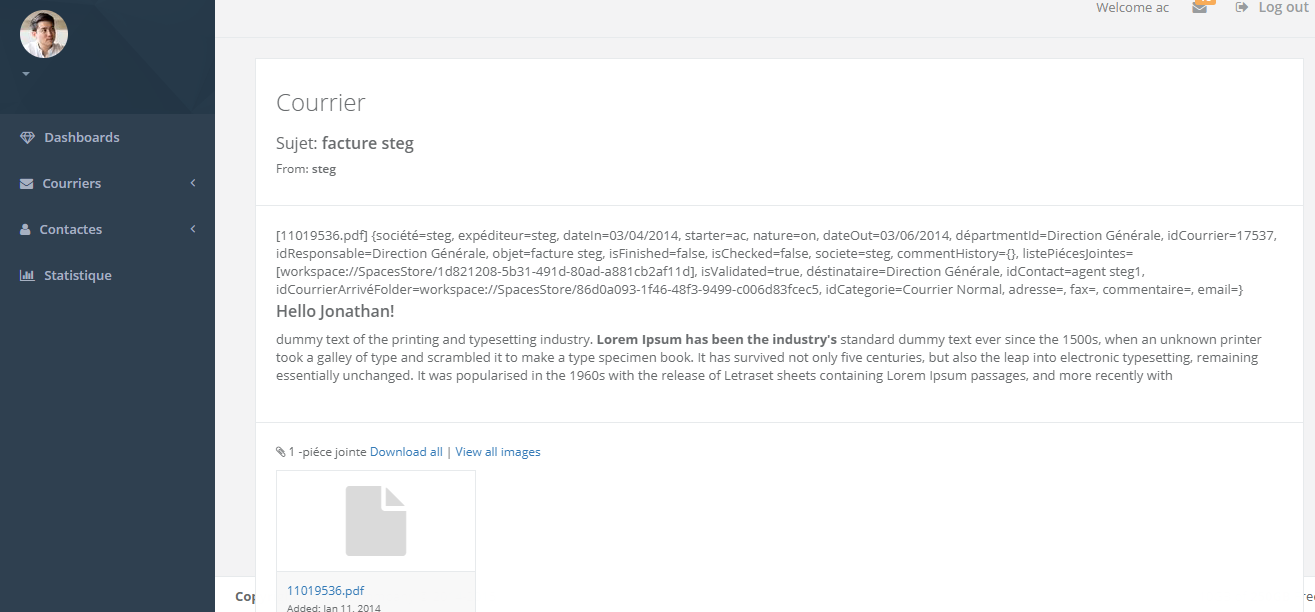
En cas d’une erreur dans n’importe qu’elle phase de création (insertion des variables de processus dans la base MySQL, insertion du/des pièce(s) jointe(s) dans la base de données Alfresco, …) une page d’erreur va être renvoyée.



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

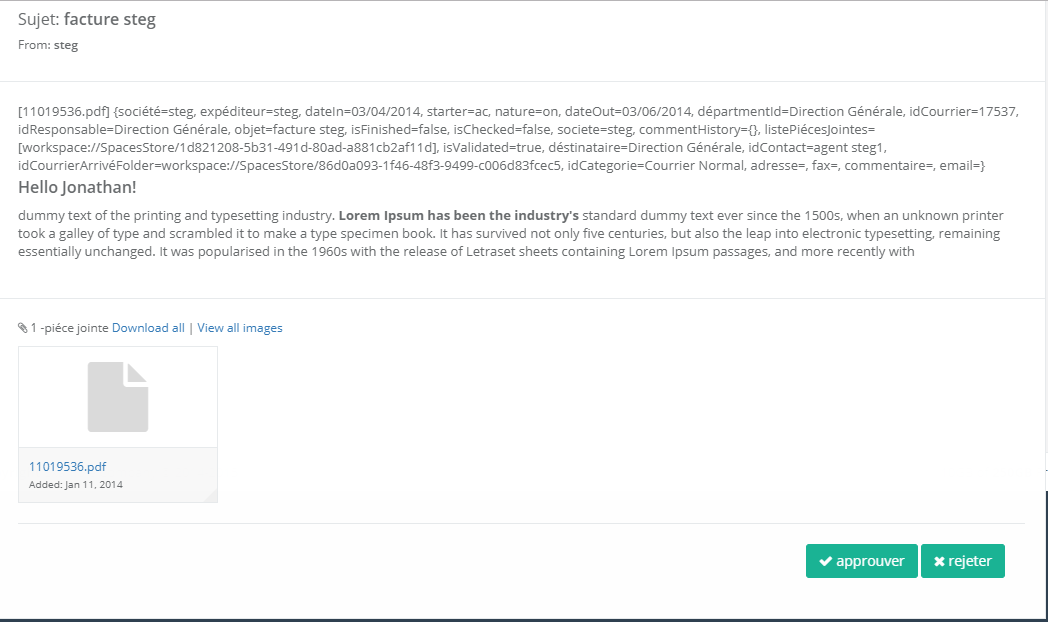
En appuyant sur le menu courrier interne dans la barre de navigation, on peut afficher la liste des courriers internes



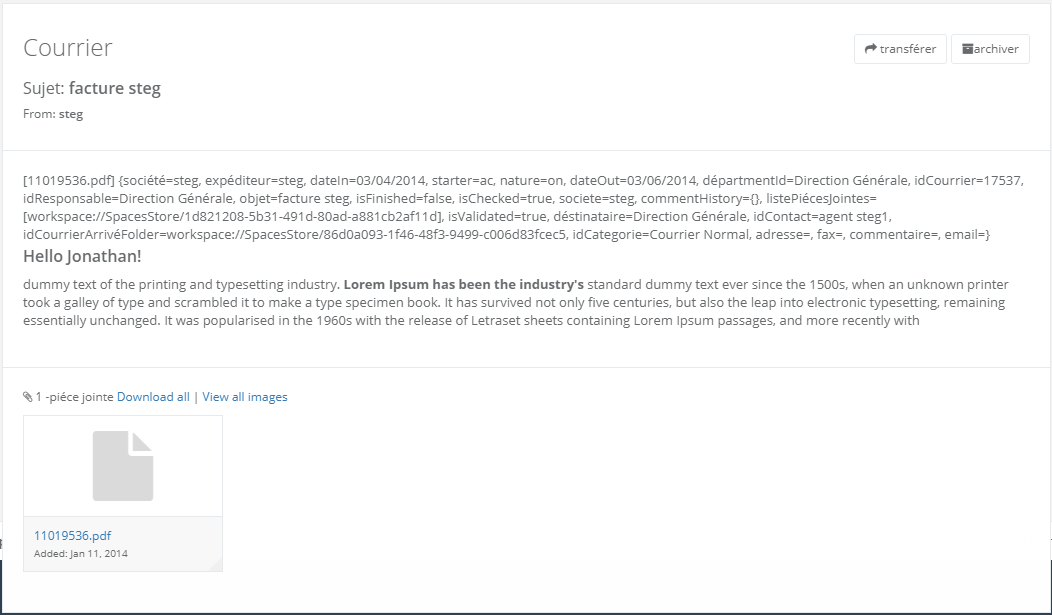
En appuyant sur l’un des courriers, on peut consulter tous les détails de celui-ci.

Cette interface peut changer selon l’état de processus :

Si par exemple un courrier est bloqué dans la phase révision par le bureau d’ordre, à l’entrée d’un membre du bureau d’ordre à cette interface, il constate la présence de deux boutons supplémentaires pour la validation ou le refus du courrier.



Dans un autre cas,si le courrier est en phase de traitement et le chef de direction concerné ouvre ce courrier, il va constater la présence de deux boutons aidant pour le traitement.



* ***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.

///

* + 1. **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 de départ.

Après avoir remplir toutes les informations nécessaires, l’utilisateur clique sur le bouton « envoyer », le système vérifie ainsi la validité des champs

Si tous les paramètres sont valides il va y avoir création d’un nouveau courrier de départ et tous les paramètres saisis deviennent alors des paramètres de processus stockés dans la base de données du moteur de workflow activiti.

En cas d’une erreur dans n’importe qu’elle phase de création (insertion des variables de processus dans la base MySQL, insertion du/des pièce(s) jointe(s) dans la base de données Alfresco, …) une page d’erreur va être renvoyée.



* ***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 celui-ci

Cette interface peut changer selon l’état de processus :

Si par exemple un courrier est bloqué dans la phase révision par le bureau d’ordre, à l’entrée d’un membre du bureau d’ordre à cette interface, il constate la présence de deux boutons supplémentaires pour la validation ou le refus du courrier.

////

Dans un autre cas, si le courrier est en phase de traitement et le chef de direction concerné ouvre ce courrier, il va constater la présence de deux boutons aidant pour le traitement.

///

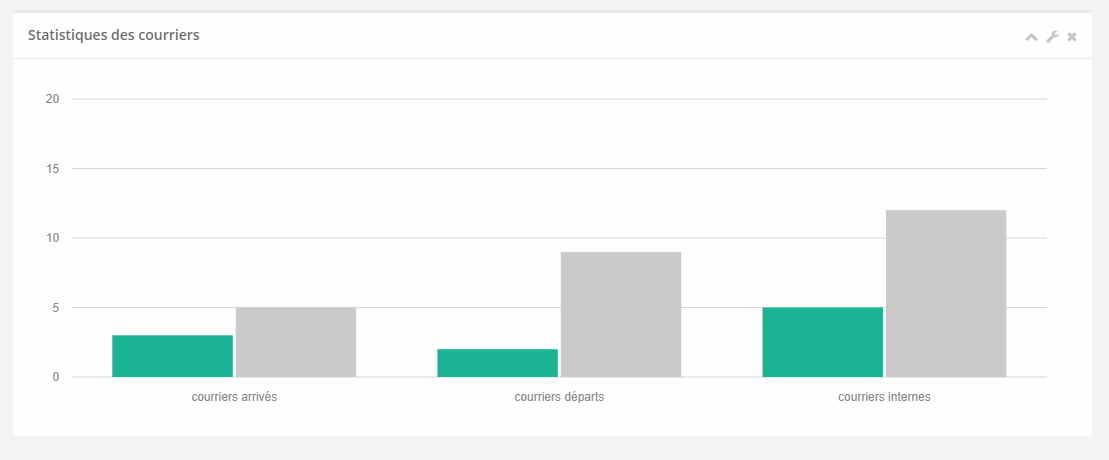
* ***Interface de modification des courrier départ*** :

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

///

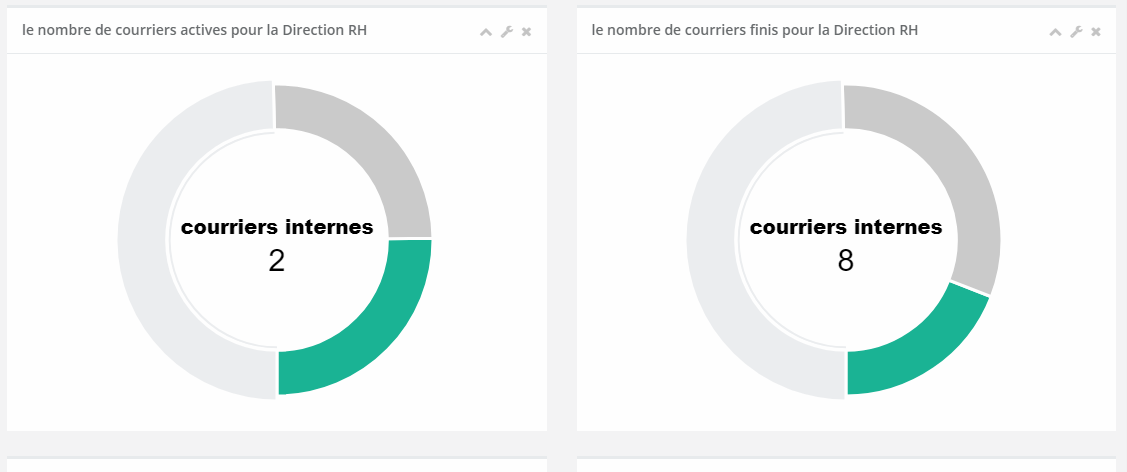
* + 1. **Sprint 4 : Statistiques**

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



La figure 44 décrit un graphe qui représente le nombre de courriers actives et finies pour chaque direction de la société.

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



**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.

1.7 Outils collaboratifs utilisés

1.7.1  Outils pour la gestion de tâches: github(https://fr.wikipedia.org/wiki/GitHub)

GitHub (exploité sous le nom de GitHub, Inc.) est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git(Git est un logiciel de contrôle de version, ce qui signifie qu’il gère les modifications d’un projet sans écraser n’importe quelle partie du projet).

GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres. Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet.

1.7.2 Outils pour la gestion de projet : easybacklog (http://doeswhat.com/2012/03/18/interview-with-matthew-oriordan-easybacklog/)

EasyBacklog est un outil de gestion de backlog Agile et Scrum facile à utiliser, conçu pour les agences et les équipes agiles travaillant avec «Agile». Il a des capacités pour aider à la versionnement et faire des rapport sur les changements tout au long du processus de livraison.

Cet outil permet de créer un backlog initial afin que on peut commencer à estimer la portée, les horaires et les coûts du projet donc c’est  un outil rapide  qui simplifie les  estimations des coûts et du temps

# Bibliographie

http://www.PicoSoft.biz/mail\_manager.htm. (s.d.).

1.8  Choix de l'approche Objet

Pour la conception de notre système, nous avons adopté une méthode orientée objet du fait des avantages qu'elle offre

* La rapidité et l'efficacité
* Les possibilités d'extension des besoins du système
* La réutilisation des objets
* La souplesse de conception
* La facilité de la maintenance des applications

L'adoption d'une approche objet pour la conception s'appuie sur une méthode ou un langage efficace pour modéliser le système d'information. Dans notre cas nous avons utilisé uml/rup

CHAPITRE 2 : Analyse et spécification des besoins

Introduction

I. Spécification des exigences

1. Besoins fonctionnels

2. Besoins non fonctionnels

Les besoins non fonctionnels présentent les contraintes qui jugent un service du

système et définissent le degré de satisfaction du client. Les besoins non fonctionnels de notre

plate-forme se résument comme suit :

1)      Sécurité :

* Le processus d’authentification : exige que chaque utilisateur dispose d’un login et d’un mot de passe qui garantissent que la personne connectée est bien celle qu'elle prétend être.
* Le processus des autorisations : permet de limiter les accès à une url à un groupe prédéfinie de personnes qui ont un rôle d’agir sur cette page.

L’utilisation d’une base de données LDAP permet la sécurisation des mots de passes des utilisateurs puisque LDAP permet de les hacher.

La sécurisation de la session est assurée par une fonction de protection contre les attaques CSRF(L’objet de cette attaque est de transmettre à un utilisateur authentifié une requête HTTP falsifiée qui pointe sur une action interne au site, afin qu'il l'exécute sans en avoir conscience et en utilisant ses propres droits. L’utilisateur devient donc complice d’une attaque sans même s'en rendre compte. L'attaque étant actionnée par l'utilisateur, un grand nombre de systèmes d'authentification sont contournés.) et par l’utilisation de la fonction « Remember me » qui permet de se rappeler l'identité d'un principal entre les sessions

2)   Ergonomie et souplesse

Notre application doit avoir une structure générale claire et simple et doit être compréhensible et facilement utilisé pour le public cible avec des interfaces conviviale et lisible et responsive.

3)      Compatibilité

L’un des points les plus importants lors du développement d’une application est d’assurer sa compatibilité avec n’importe quelle version du système et n’importe quelle version et type de navigateur.

4)      Aptitude à la maintenance

Le code de l’application implémentée doit être facile à maintenir et étendre tout en respectant une structure orientée objets pour des raisons de réutilisation.

5)portabilité

Cette application peut tourner sur n’importe qu’elle système d’application desktop(windows,linux ) .