|  |  |  |
| --- | --- | --- |
| Projet : **AMUBook** | | |
| **Rapport technique** | | |
| Ref doc : RT | Version : 1.0 | Statut : approuvé |
| Description : Rapport technique pour le projet AMUBook. | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Liste des participants au groupe projet** | | | | | |
| Nom | Initiales | Email | Appartenance | Qualité/Rôle | Présence |
| Jean-Luc Massat | JLM | jean-luc.massat@univ-amu.fr | Laboratoire d’informatique de Marseille | Chef de projet MOA | P |
| Pierre Vincent | PV | pierre.vincent.1@etu.univ-amu.fr | Etudiant | Chef de projet MOE | P |
| Lucas Loignon | LL | Lucas.loignon@etu.univ-amu.fr | Etudiant | Développeur | P |
| Rémi Deutsch | RD | Remi.deutsch@etu.univ-amu.fr | Etudiant | Développeur | P |

*P = présent, A = absent, E = excusé*

|  |  |  |
| --- | --- | --- |
| **Liste de diffusion du document** | | |
| Destinataire | Version(s) diffusée(s) | Date de diffusion de la dernière version |
| Participants | 1.0 | 10/02/2019 |
| Jury projet | 1.0 | 10/02/2019 |
| **Restriction de diffusion** | Ce document ne doit pas être copié ou diffusé à un tiers hors de la liste de diffusion sans l’accord du chef de projet MOA | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Historique des révisions du document** | | | | | | |
| Version | Date révision | Page/sections concernées | Description de la modification | Auteur (initiales) | Date d’approbation | Approuvé par |
| 1.0 | 31/01/2019 | Toutes | Création | PV | 5/02/2019 | Pierre Vincent  Lucas Loignon  Rémi Deutsch |

Table des matières

[I. Objet du document 3](#_Toc536712947)

[II. Rappel des fonctionnalités 3](#_Toc536712948)

[III. Technologies utilisées 3](#_Toc536712949)

[IV. Architecture et Conception 3](#_Toc536712950)

[V. Tests 3](#_Toc536712951)

[VI. Problèmes rencontrés 3](#_Toc536712952)

# Objet du document

Ce document est le rapport technique concernant le projet AMUBook réalisé dans le cadre de l’UE « architecture des applications » sous la direction de M. Massat.

# Rappel des fonctionnalités

Voici les fonctionnalités définies dans le cahier des charges de l’application :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | **Besoin** |
| **Référence** | **Enoncé des fonctions de service** | **Priorité** | **B1** |
| FS1 | Permettre à l’utilisateur de se connecter à son compte | 0 | X |
| FS2 | Permettre à l’utilisateur de se déconnecter de sa session. | 0 | X |
| FS3 | Permettre à l’utilisateur de rechercher un CV, par le nom, le prénom du propriétaire ou le titre d’une activité présente sur celui-ci. | 0 | X |
| FS4 | Permettre à l’utilisateur de consulter un CV. | 0 | X |
| FS5 | Permettre à l’utilisateur de créer un compte (cooptation). | 0 | X |
| FS6 | Permettre à l’utilisateur d’éditer ses informations personnelles. | 1 | X |
| FS7 | Permettre à l’utilisateur d’ajouter une activité à son CV. | 0 | X |
| FS8 | Permettre à l’utilisateur de supprimer une activité de son CV. | 0 | X |
| FS9 | Permettre à l’utilisateur d’éditer une activité de son CV. | 1 | X |
| FS10 | Permettre à l’utilisateur de renseigner le titre et la description de son CV. | 2 | X |
| FS11 | Permettre à l’utilisateur d’éditer le titre et la description de son CV. | 2 | X |
| FS12 | Accueillir l’utilisateur sur une page dédié. | 2 | X |

# Technologies utilisées

**Environnement de développement**

Java version 8

**Serveur d’application**

TomEE-Plus version 7.1.0

Ce package fournit entre autres un serveur Tomcat ainsi que le framework OpenEJB, tous deux indispensables à l’exécution et au fonctionnement de notre application.

Le serveur Tomcat permet de déployer notre archive .war et de lancer l’application web.

**Implementation JSF**

Mojara version 2.2.1

Il s’agit d’une des implémentations les plus populaires pour les spécifications JSF 2.1 et supérieures.

Elle était une des solutions intégrées dans notre outil de développement (IntelliJ).

**Persistance des données**

Hibernate version 4.3.11

Nous avons choisi ce framework en tant que provider pour la persistance des données. Il s’agit de l’implémentation JPA la plus utilisé et elle était parfaitement adapté à nos besoins, notamment pour les fonctionnalités CRUD.

**Bases de données**

HSQLDB in memory

Utilisé principalement pour la phase de développement. Il est cependant impossible de déployer l’application finale de cette façon car elle ne supportera pas les 100.000 utilisateurs comme prévu dans les spécifications. Si l’on désire continuer à utiliser HSQLDB il faudra donc l’utiliser en mode File.

MySQL Server

Utilisé pour le déploiement « réel » de l’application. Un serveur MySQL local a été mis en place et connecté avec notre application afin de permettre une plus grande robustesse (100.000 utilisateurs et plus).

Attention : l’utilisation d’un serveur MySQL distant nécessite une certaine configuration, notamment l’ajout d’un driver dans le serveur Tomcat. Cette procédure sera décrite dans la section [Problèmes rencontrés](#_Problèmes_rencontrés).

**Tests**

JUnit version 4.11

Ce framework nous permet de réaliser les tests unitaires des différentes fonctionnalités de notre application.

Mockito version 2.23.0

Ce framework est un outil qui facilite l’implémentation de nos tests unitaires. Il fournit des instances virtuelles de classes dont dépendent nos méthodes de test.

**Front-end**

JQuery version 3.3.1

Il s’agit un framework qui offre une surcouche au javascript natif. Il est souvent requis par les framework front-end tels que Boostrap, Chartjs, Particles etc…

Bootstrap version 4.1.3

Il s’agit du framework CSS le plus populaire, il permet de mettre en place rapidement et facilement une interface graphique moderne, adaptable (« responsive ») et user-friendly.

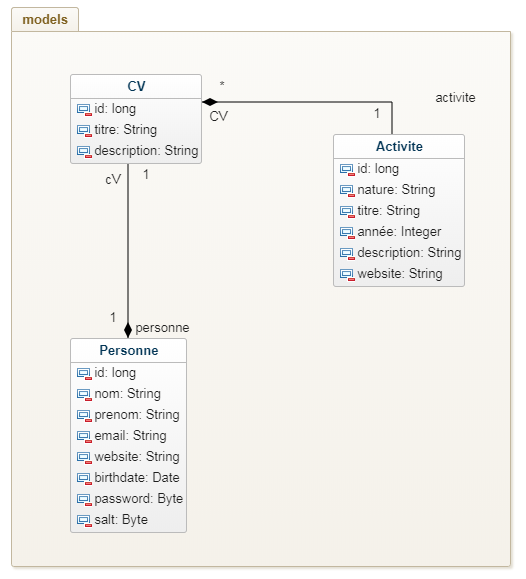
# Architecture et Conception

L’architecture de l’application repose sur quatre package qui ont des buts distincts.

* Le package « models » contient tous les beans destinés à être persisté en base de données grâce à JPA.
* Le package « services » contient tous les beans de la couche service de l’application.
* Le package « bean » contient les « backing beans », ces beans sont destiné à être utilisé avec des composants web de JSF.
* Le package « dao » contient tous les objets se chargeant de faire des accès à la base de données.
* Le package « controllers » ne contient qu’une seule classe : le contrôleur web de JSF.

## Package models

Voici l’architecture du package models :



Nous avons trois classes destinées à être persistées : CV, Personne et Activité. Une personne est composée d’un CV qui lui-même est composé de plusieurs activités.

Le seul point technique pouvant être abordé ici est la gestion de mots de passes. On remarquera un attribut « salt » en plus du mot de passe dans le bean Personne. En effet les mots de passe ne sont pas stocké en clair dans la base de données ils sont cryptés.

L’algorithme de cryptage est le suivant lors de la création du mot de passe :

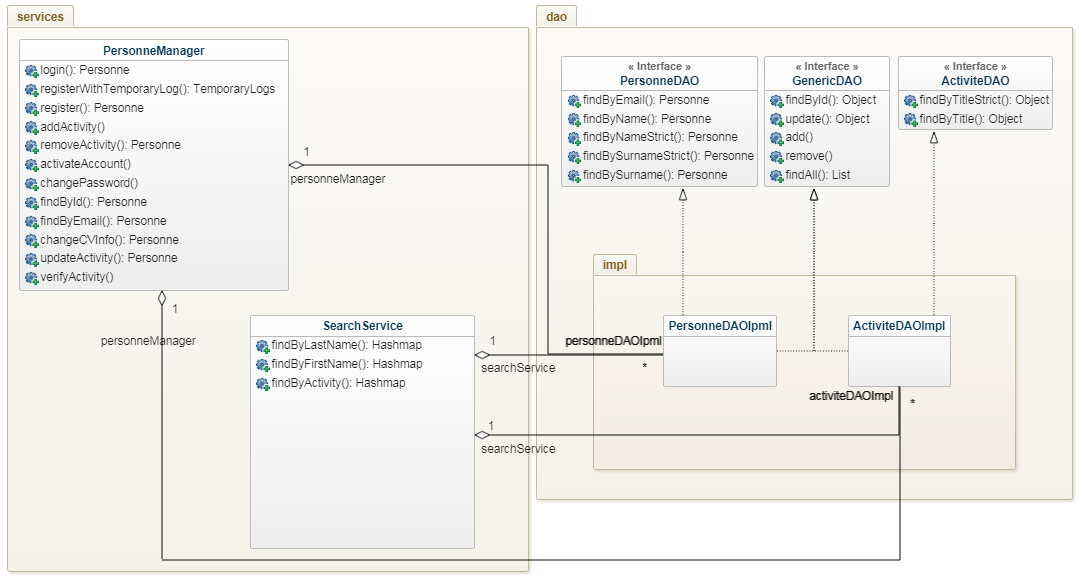
* Générer un sel aléatoire de 128 octets
* Ajouter ce sel au début du mot de passe
* Crypter le mot de passe obtenu avec l’algorithme SHA-512
* Stocker le mot de passe crypté et le sel dans le bean Personne

La procédure est quasiment la même lors de la connexion :

* Ajouter le sel au mot de passe proposé
* Crypter le mot de passe salé avec l’algorithme SHA-512
* Vérifier s’il correspond au mot de passe stocké

## Package services et dao

Voici l’architecture des relations entre la couche service et dao :



On peut voir que nous avons deux classes de services, PersonneManager regroupe les services liés à la gestion des personnes tandis que SearchService regroupe les services liés à la recherche.

# Tests

# Problèmes rencontrés