

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
II.	Rappel des fonctionnalités	3
III.	Technologies utilisées.....	4
IV.	Architecture et Conception	6
a)	Package models.....	6
b)	Package services et dao.....	8
V.	Tests.....	9
VI.	Problèmes rencontrés	9
a)	Mise en place de JSF.....	9
b)	Mise en place d'une source de données MySQL.....	9

	Projet	Document	Version	Date	p. 2 / 10
	AMUBook	RT	1.0	31/01/2019	

I. 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.

II. 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

III. 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).

Persistence 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 utilisée et elle était parfaitement adaptée 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

	Projet	Document	Version	Date	p. 4 / 10
	AMUBook	RT	1.0	31/01/2019	

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](#).

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 Bootstrap, 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.

	Projet	Document	Version	Date	p. 5 / 10
	AMUBook	RT	1.0	31/01/2019	

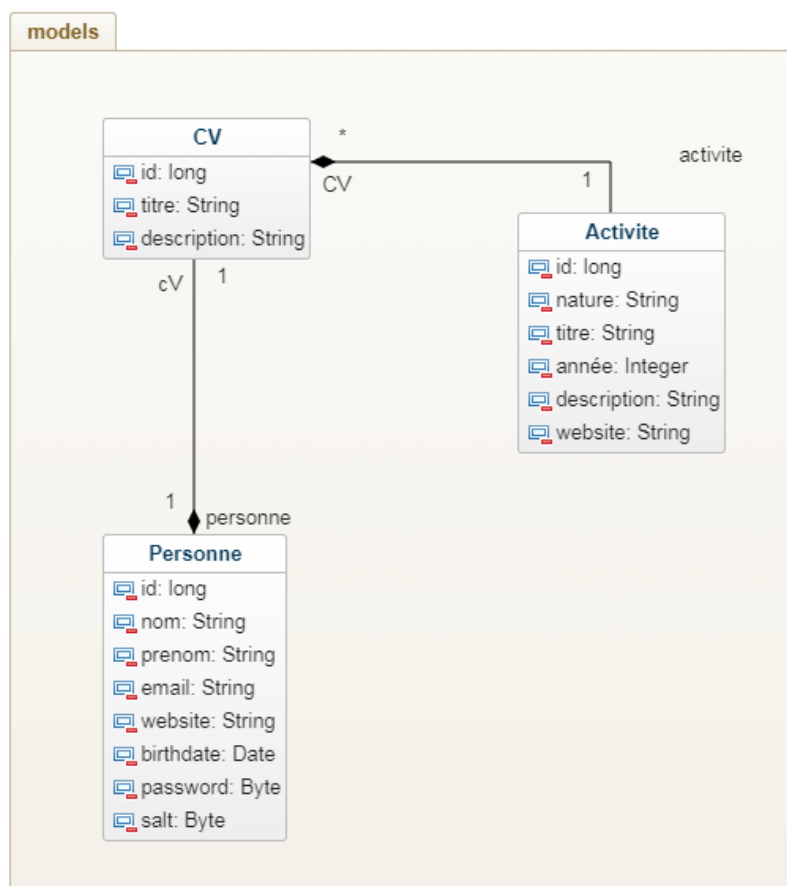
IV. Architecture et Conception


L'architecture de l'application repose sur quatre packages 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 « beans » contient les « backing beans », ces beans sont destinés à être utilisés 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. Ce package ne fera pas l'objet de détail ici.

a) Package models

Voici l'architecture du package models :



	Projet	Document	Version	Date	p. 6 / 10
	AMUBook	RT	1.0	31/01/2019	

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 stockés 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


b) Package beans

Comme dis précédemment, ce package contient des beans gérés par JSF (ManagedBean) qui permettent le stockage d'informations liées aux composants JSF.

Les noms de ces beans se réfèrent directement au nom des composants web auxquels ils sont associés.

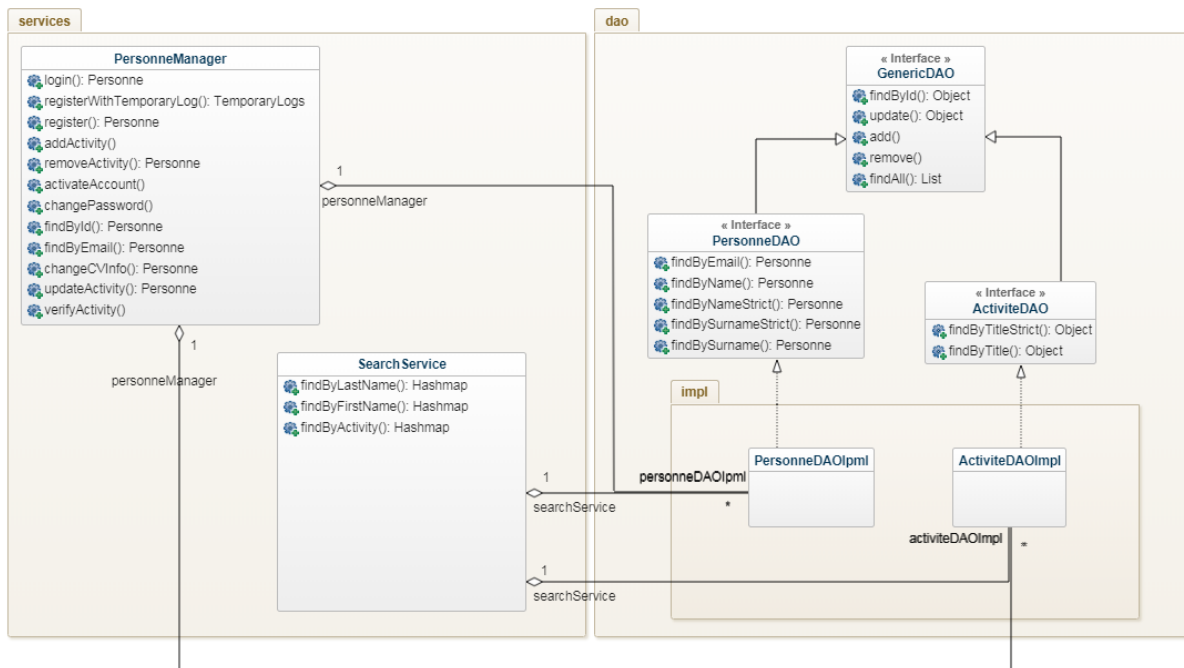
Par exemple : le bean LoginPageBean se réfère au composant web « LoginPage » qui permet à un utilisateur de fournir son login et son mot de passe. Ce bean stocke donc le login et le mot de passe d'un utilisateur qui sera utilisé dans le contrôleur.

Cette méthode permet de déplacer les différents attributs dont les composants web peuvent avoir besoin dans une classe propre afin de ne pas saturer le contrôleur.

	Projet	Document	Version	Date	p. 7 / 10
	AMUBook	RT	1.0	31/01/2019	

c) 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
- **SearchService** regroupe les services liés à la recherche.

Ces services utilisent la couche dao afin d'interagir avec la base de données.

Les DAOs sont au nombre de deux, il existe un DAO pour les beans **Personne** et un autre pour les beans **Active**.

Ces DAOs étendent l'interface **GenericDAO** qui définit les actions CRUD basique possible sur la base de données. L'implémentation des DAOs est faites dans les classes **PersonneDAOImpl** et **ActiveDAOImpl**.

V. Tests

Afin de développer ce logiciel nous nous sommes inspiré des méthodes agiles et en particulier de la méthode XP.

Lors du développement nous avons donc pu mettre en place une série de tests unitaires pour chaque méthode implémentée. Ces tests ont été écrit en même temps que la méthode elle-même et couvre une bonne partie du code.

Les tests unitaires peuvent être trouvé dans [/src/test/java/unit](#).

VI. Problèmes rencontrés

Lors du développement de ce projet, nous nous sommes heurtés à une série de problèmes que nous essaierons de détailler dans cette partie.

a) Mise en place de JSF


Nous travaillons sur l'IDE IntelliJ et lors de la mise en place du framework JSF la version de Mojarra utilisé par défaut est la 1.1.2. Il s'avère qu'avec cette version le projet ne fonctionnais pas sous TomEE. Après de plus ample recherche nous avons décidé de télécharger et d'utiliser Mojarra en version 2.2.1 ce qui a résolu notre problème.

b) Mise en place d'une source de données MySQL

Lors du développement nous avons utilisé HSQLDB en tant que source de données, cette solution est suffisante pour un contexte de développement, mais ne suffit plus lors d'une production. Nous avons donc décider de tirer avantage de l'abstraction fournie par JPA pour changer notre source de données.


Pour cela nous avons modifier le fichier « persistence.xml » afin d'y intégrer une configuration MySQL et avons modifier nos DAO en conséquence. Cette modification n'était pas prise en compte par notre serveur applicatif TomEE. Le serveur essayait d'exécuter des requêtes MySQL sur un serveur HSQLDB ce qui engendrais le plantage de l'application.

Afin de résoudre notre problème il a fallu ajouter la datasource MySQL définie dans le fichier « /conf/openejb.xml » dans le fichier « /conf/tomee.xml » de notre serveur TomEE.

	Projet	Document	Version	Date	p. 9 / 10
	AMUBook	RT	1.0	31/01/2019	

En plus de cet ajout dans le fichier de configuration il a fallu rajouter le driver MySQL dans le répertoire « /lib » de TomEE.

Ces manipulations nous ont permis de faire tourner notre application avec une base de données MySQL.

	Projet	Document	Version	Date	p. 10 / 10
	AMUBook	RT	1.0	31/01/2019	