PROJET JEE - RAPPORT

(Document de 18 pages)

Résumé

Ce document à pour fonction d'expliquer comment les besoins du cahier des charges ont été implémentés. Comment les composants principaux du site fonctionnent et de visualiser les résultats obtenus.

Mots Clés

JEE, Annuaire, Rapport

Université d'Aix-Marseille
Département Informatique et Interactions

3 Place Victor Hugo
13331 MARSEILLE CEDEX 3

SECTION DES RÉDACTEURS

Nom	Prénom	Contribution
MAURICE	Gautier	Rédacteur

CONTACTS

Nom	Prénom	Email	Fonction
MAURICE	Gautier	gautier.maurice@etu.univ-amu.fr	Développeur

TABLE DES MATIÈRES

1. LES SOURCES REMISENT

Les sources remisent sont :

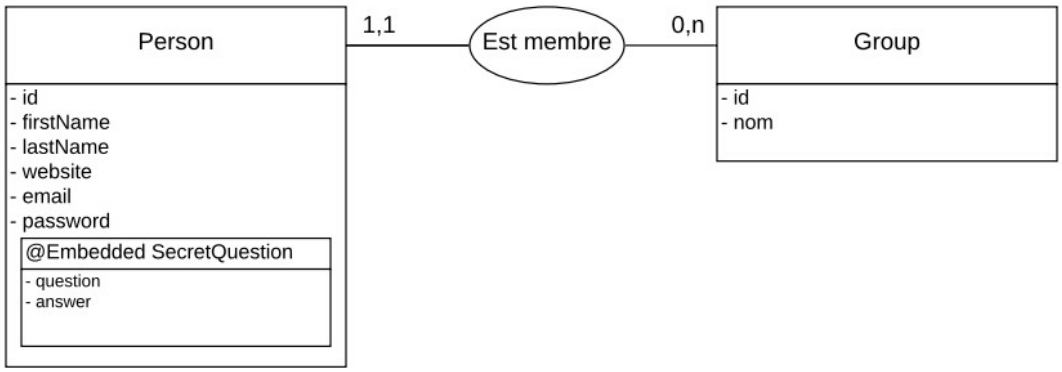
- CDC.pdf : Cahier des charges
- Rapport.pdf : Rapport
- src/ : Sources, tests et ressources de l'application web.
- pom.xml : Fichier maven
- annuaire-0.0.1-SNAPSHOT.war : Exécutable de l'application web sur un serveur Tomcat

2. LES OUTILS UTILISÉS

L'application web a été réalisé avec JEE, Hibernate, Spring-boot, Spring-mvc, . Les tests ont étés réalisés avec les Mocks fournis par Spring-boot et JUnit. L'application est déployée sur un serveur Tomcat. Bootstrap, FontAwesome et JSTL ont été utilisé pour l'édition des vues.

3. L'ACCÈS AUX DONNÉES

3.1. Modèle Entités/Relations



Pour des questions de maintenabilités, une classe à part a été créée pour le tuple : (question, réponse secrète).

3.2. La couche DAO

3.2.1. L'interface

L'accès aux données doit implémenter les méthodes pour :

- Récupérer les entités en fonction de leur identifiant,
- Récupérer toutes les entités d'un certain type,

- Récupérer toutes les personnes d'un certain groupe,
- Proposer un argument de filtre de sélection pour toute les méthodes retournant une liste de résultats.

3.2.2. L'implémentation

La couche DAO accède à la base de données avec le JDBC, et le l'ORM Hibernate a été choisi pour construire les requêtes.

3.3. Le service DirectoryManager

En fonction de l'utilisateur, les accès aux données ne sont pas les mêmes, ce service gère la logique des accès aux données et, par mesure de sécurité supplémentaire, nettoie les données qui ne devrait pas être accessible à l'utilisateur courant.

Pour fonctionner le service doit savoir qui est actuellement authentifié dans la session courante.

4. LA RÉINITIALISATION DU MOT DE PASSE

4.1. Principe général

Il y a trois étapes pour la réinitialisation d'un mot de passe, identifier la personne qui a perdu son mot de passe, lui poser sa question personnelle et saisir sa réponse secrète, et finalement saisir le nouveau mot de passe.

4.2. Le service de réinitialisation du mot de passe

Le service de réinitialisation du mot de passe est lié à la session pour être sûr que personne ne puisse brûler les étapes et changer un mot de passe sans avoir répondu à nos mesures de sécurité (Annexe 1 : Diagramme états / transitions réinitialisation du mot de passe).

Ce service propose :

- Une méthode pour préciser quel est la prochaine étape pour la réinitialisation du mot de passe (récupérer l'email, la réponse secrète, ou le nouveau mot de passe),
- Les méthodes permettant de faire avancer la récupération du mot de passe d'une étape. Ces méthodes retournent vrai ou faux pour indiquer si la récupération du mot de passe a pu avancer d'une étape.

5. LES ROUTES

5.1. Le Mapper

Toutes les routes sont éditées au même endroit, les autres composants de l'application doivent s'en servir pour l'affectation de toutes variables se rapportant à une route.

Ainsi les routes peuvent être modifier très facilement sans affecter le fonctionnement du site.

6. LA GESTION DES TRIS

Pour que tout ce qui ce qui se rapporte à un tri soit accessible par tous les contrôleurs, les tris sont gérés par un service.

6.1. Le service de tri

Pour chaque entité propose une fonction qui prend en paramètre la liste à trier et le type de tri a effectuer.

Les type de trie sont généralement composés de l'attribut sur lequel trier et de l'ordre (ascendant ou descendant).

7. L'INTERNATIONALISATION

7.1. Gérée par Spring

L'internationalisation est gérée automatiquement par Spring, la gestion de la variable de session 'Locale' est automatiquement changé avec la valeur affectée au paramètre 'language' pour n'importe quel route.

7.2. L'accessibilité depuis les vues

Le menu principal possède un menu déroulant composé des traductions disponible, chaque item de ce menu est un lien vers l'URL actuelle avec les même paramètres sauf le paramètre 'language' qui varie en fonction du lien.

8. CHARGEMENT DE DONNÉES FACTICES

Pour tester le site web plus facilement des données factices sont générées lors de la configuration du DAO.

8.1. La classe DataLoader

La classe DataLoader créée des données factices.

Elle a deux paramètres : le nombre de personne a créer et le nombre de groupe à créer.

Ensuite les personnes créées ont des noms / prénoms aléatoires, un mail égal à ‘mail@<id>’ où id est l’identifiant de la personne dans la base de données, un mot de passe égal à : ‘pass<id>’. Elles sont affectées aléatoirement dans un groupe.

9. LES CONTRÔLEURS

9.1. Association des fonctions aux routes

Pour être sûr d’associer leurs fonctions aux bonnes routes, ils utilisent le Mapper.

9.2. Les paramètres globaux

Certains paramètres peuvent potentiellement être traités par plusieurs routes comme :

- ‘id’ : Identifiant de l’entité concernée par la route,
- ‘sort’ : Type de tri à utiliser pour les listes retournées dans la page,
- ‘pattern’ : Filtre de recherche pour les listes retournées dans la page.

9.3. Revue des contrôleurs

9.3.1. BaseController

Il permet de factoriser le code commun à tous les autres contrôleurs. Pour l’instant, il ne fait que rajouter l’attribut : ‘isLogged’ au modèle, qui précise si la session courante est utilisée par un utilisateur authentifié.

9.3.2. AuthController

Il traite les routes qui concerne la connexion et la déconnexion.

9.3.3. HelloController

Il traite les routes qui font office de page d’accueil.

9.3.4. PersonController

Il traite les actions liées aux personnes, à savoir :

- L’édition d’une personne,
- La visualisation d’une personne,
- Le listing de toutes les personnes de l’annuaire.

9.3.5. GroupController

Il traite les actions liées aux groupes, à savoir :

- La visualisation d'un groupe,
- Le listing de tout les groupes de l'annuaire.

9.3.6. LostPasswordController

Il traite les actions à la réinitialisation du mot de passe c'est à dire qu'une seule route. Lorsque la requête est de type GET alors il demande au service la prochaine étape et retourne la vue correspondante.

Lorsque la méthode est de type POST, il appelle la fonction du service qui correspond au paramètre, redirige avec une requête de type GET en précisant s'il y eu une erreur.

10. LES VUES

Les vues sont construites par les pages Jsp dans le dossier 'src/main/webapp/WEB-INF/'.

10.1. Framework CSS

Pour avoir un prototype présentable le framework css Bootsrap 4.1 à été utilisé, ainsi que l'outil d'icônes Font Awesome 4.7 (Annexe 2 : Page d'un groupe).

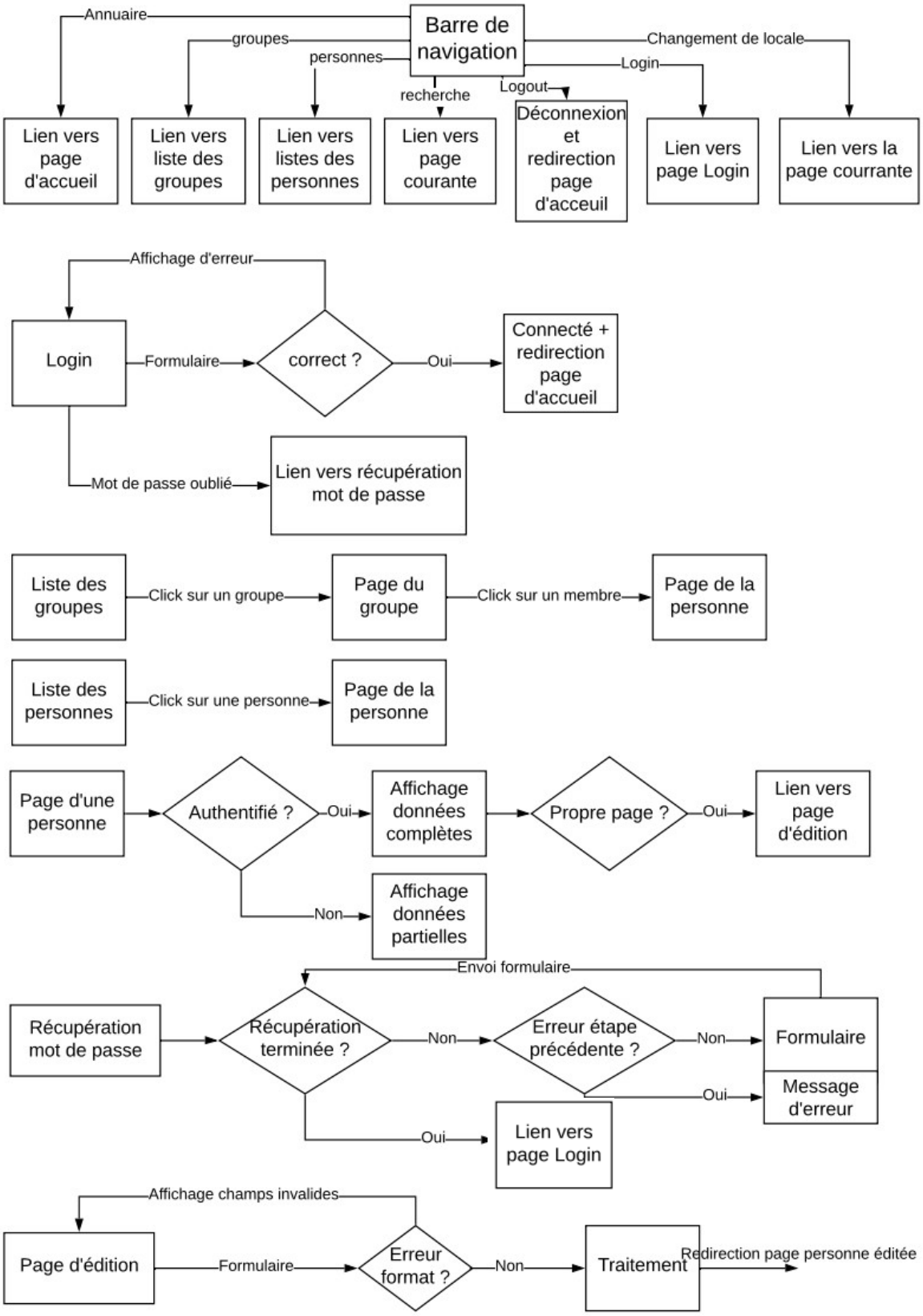
10.2. La fabrique d'URLs

Pour être sûr que tous nos liens soient valide, on une librairie de tag 'url-builer' a été mise en place, elle utilise des tags codés en Java pour être en lien avec le Mapper.

10.3. Autres

D'autres tags comme, la création de certains boutons, la création des listes d'entités, d'une barre de recherche, ou la création d'une URL égale à l'URL courante en changeant juste un seul paramètre ont étés mis en place car ils sont fréquemment utilisés dans le code.

11. DIAGRAMME DE NAVIGATION



12. AMÉLIORATIONS ENVISAGEABLES

12.1. L'utilisation de DTOs

Le DirectoryManager 'nettoie' les données pour être sûr qu'aucune information qui ne devait pas être affiché le soit par erreur, certes ce mécanisme présente des avantages mais pour l'instant il est beaucoup trop coûteux en temps de calcul et à chaque fois que des données devront être 'nettoyées' différemment il faudra modifier le service.

L'utilisation de DTO dans la DAO réglerai les deux problèmes.

Il suffirait de passer en argument des fonctions DAO la classe de DTO souhaitée et seul les champs de la classe DTO seront chargés et il n'y aura pas à reparcourir toutes les entités retournées.

Si une nouvelle règle de transfert de données doit être mise en place, il faudra seulement créer la classe DTO associée et la passer en argument.

C'est sans doute l'amélioration la plus importante, mais elle est aussi complexe à mettre en place.

12.2. Conception du service de récupération de mot de passe

Pour l'instant il est possible de rajouter une étape supplémentaire dans la récupération du mot de passe assez facilement. La logique sera gérée dans le service et la méthode POST devra juste être capable de traiter un paramètre supplémentaire.

Il y a quand même un léger problème : c'est qu'un raccourci a été utilisé pour avoir un minimum de code à modifier en cas de changement. Et il y a donc un pseudo lien entre le service et les vues. Un tel couplage ne pose pas de problème à ce stade du projet mais pourrait en poser dans l'avenir.

Une solution générique peut être envisageable, le service avec récupération de mot de passe fonctionne comme un formulaire 'wizard' mais avec des calculs intermédiaire sur le serveur entre chaque étape. Un jour, on pourrait très bien avoir une autre fonctionnalité qui fonctionne avec le même principe et il faudrait recoder une deuxième fois, et ce n'est pas envisageable.

Cette solution reste quand même coûteuse à implémenter, mais elle est probablement rentable pour des raisons évidentes de maintenabilités.

12.3. Conception du mécanisme de tri

Une première amélioration pourrait être la mise en place de champs statique contenant les identifiants des types de trie disponible.

Ainsi, comme pour le Mapper, les vues et les contrôleurs seront sûr de gérer les tries correctement, même si leurs formats changent.

13. TESTS RÉALISÉS

13.1. Tests DAO

Toute les méthodes de l'interface DAO ont été testées. Avant chaque test les données de la base sont nettoyées. Tous les tests sauvegardent le jeu de données avec la DAO et vérifient les résultats en appelant les fonctions de recherches de la DAO (Annexe 3 : Suite de tests DAO).

13.2. Test DataLoader

Pour cette suite, un seul test est nécessaire, il s'assure que la base des données contienne des données au démarrage (Annexe 4 : Suite de tests DataLoader).

13.3. Tests DirectoryManager

Le DirectoryManager utilise essentiellement la couche DAO, un mock a été mis en place pour simuler le comportement d'une DAO valide.

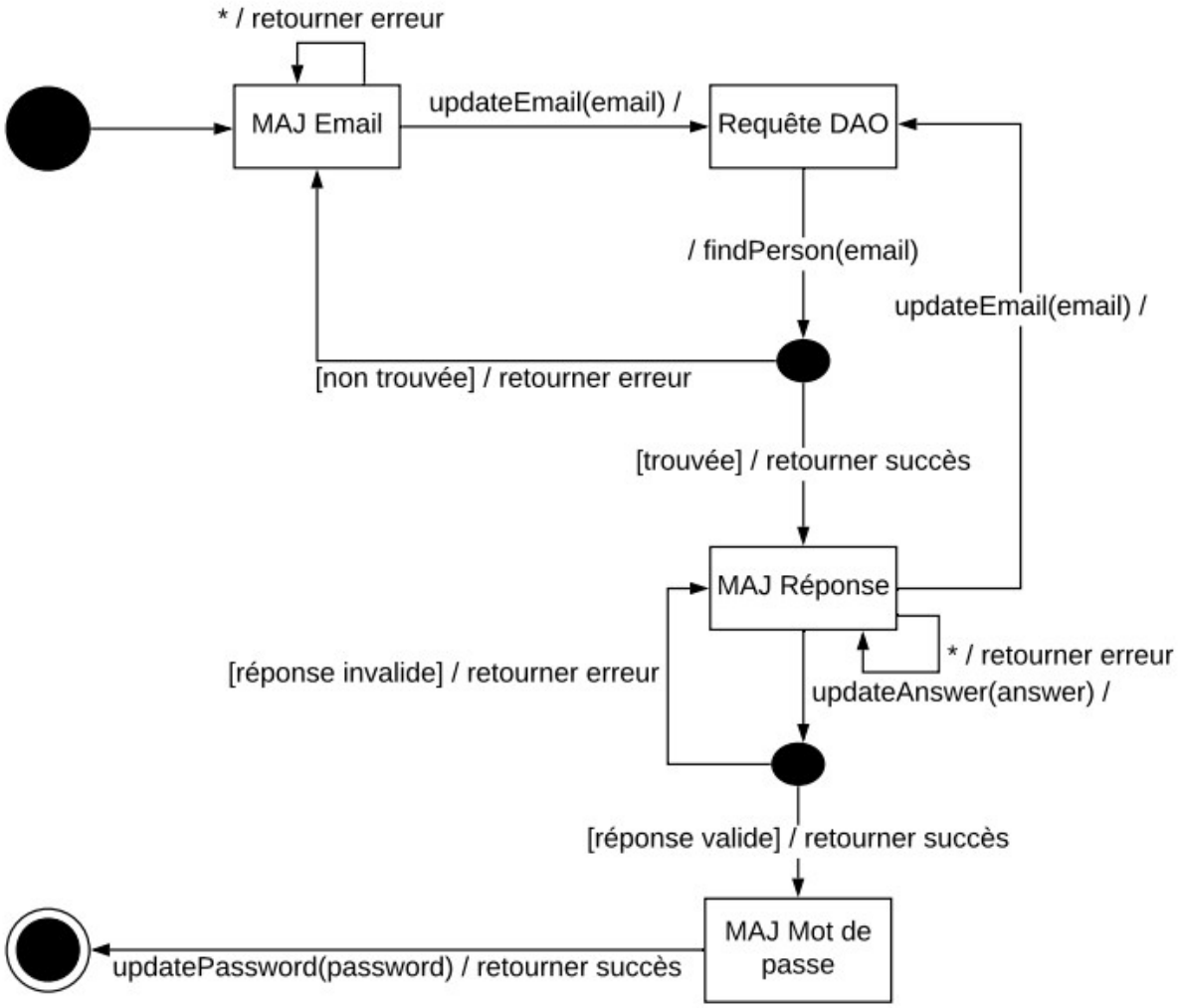
Les tests vérifient principalement que les opérations et accès aux données sont bien respectées en fonction de l'utilisateur dans la session (Annexe 5 : Suite de tests DirectoryManager).

13.4. Tests EntitySorter

Ce service vérifie que les tries sont corrects (Annexe 6 : Suite de tests EntitySorter).

ANNEXES

1 – Diagramme états / transitions récupération mot de passe



2 – Page d'un groupe

TaylorGroupeJones1
Nombre de membres : 8

ID	Nom	Prénom
429	Wilson-Saquet	Oscar-Jack
666	Beth-Taylor	Charlie-Charlie
737	Jones-Evans	James-George
886	Johnson-Brown	Jack-Charlie
1077	Evans-Jones	George-Oscar
1391	Brown-Saquet	Jacob-Jack
1416	Saquet-Johnson	Oscar-Jacob
1492	Smith-Thomas	Harry-Jack

3 – Suite de tests DAO

Nom de la fonction test	Jeu de données	Résultat attendu
testClearDB	2 personnes	La recherche de ces personnes dans la DAO retourne NULL
testAddCorrectPerson	Une personne respectant toute les contraintes sur ses champs	La recherche de cette personne dans la DAO fonctionne correctement
testAddDuplicatedPerson	Deux personnes identiques	L'enregistrement de la deuxième personne lève une exception
testAddTooLongName	Une personne ayant un nom trop long	L'enregistrement de cette personne lève une exception
testAddGroup	Groupe vide	L'enregistrement de ce groupe se passe correctement
testAddDuplicatedGroup	Deux groupes identique	L'enregistrement du deuxième groupe lève une exception

testAddPersonToGroup	Un groupe et une personne appartenant à ce groupe	Que le groupe contienne bien un membre, que la personne appartienne bien à ce groupe
testAddMultiplePersonToGroup	Un groupe et plusieurs personne	Que le groupe contienne tous ces membres et que toutes les personnes appartiennent à ce groupe
testPersonQuitGroup	Un groupe et une personne qui appartient à ce groupe	Après avoir supprimer la personne du groupe : le groupe ne contient plus ce membre et la personne n'appartient à aucun groupe
testPersonChangeGroup	Deux groupes (A et B) et une personne appartenant au groupe A.	La personne n'appartient plus au groupe A mais au groupe B.
testFindPersonWithEmail	Une personne	La recherche à partir de l'email se passe correctement
findPersonWithEmailUnexisting	Une personne	La recherche à partir d'une autre email retourne NULL.

4 – Suite de tests DataLoader

Nom de la fonction test	**Jeu de données**	**Résultat attendu**
testGroupArePresent	– Chargement initial –	Que des groupes soient présents
testUsersAreInGroup	– Chargement initial –	Que toutes les personnes soient présentes dans les groupes d'origines

5 – Suite de tests DirectoryManager

Le jeu de données Mock => 2 groupes A et B, 2 personnes A et B.

Nom de la fonction test	Jeu de données	Résultat attendu
testGetAndSetConfig	Ø	Que la configuration du service fonctionne correctement
testMockWorking	Ø	Que l'injection du Mock pour la couche DAO fonctionne
successAuth	Mot de passe correct	Retourne vrai et l'id de la personne soit enregistrée dans le service
failAuth	Mot de passe incorrect	Retourne faux et l'id de la personne ne soit pas enregistrée dans le service
anonymousCanAccessGroup	Ø (Mock)	Qu'une personne non authentifiée puisse accéder aux groupes
anonymousCannotAccessConfidentialPersonInfo	Ø (Mock)	Qu'une personne non authentifiée ne puisse pas accéder aux emails et date de naissance.
anonymousCannotAccessConfidentialPersonInfoViaGroup	Ø (Mock)	Qu'une personne non authentifiée ne puisse pas accéder aux emails et date de naissance en passant par les getters des entités groupes.
anonymousCanAccessPublicPersonInfo	Ø (Mock)	Qu'une personne non authentifiée puisse accéder aux données publique des personnes

loggedCanAccessPublicPersonInfo	Ø (Mock)	Qu'une personne authentifiée puisse accéder aux données publique des personnes
loggedCanAccessConfidentialPersonInfo	Ø (Mock)	Qu'une personne authentifiée puisse accéder aux emails et date de naissance des personnes
loggedCanAccessConfidentialPersonInfoViaGroup	Ø (Mock)	Qu'une personne authentifiée puisse accéder à toutes les données des personnes en passant par les groupes
loggedCannotUpdateAnotherPerson	Ø (Mock)	La mise à jour de B authentifié en tant que A n'a aucun effet
anonymousCannotUpdateAnyPerson	Ø (Mock)	La mise à jour de B en tant qu'utilisateur non authentifiée n'a aucun effet
loggedCanUpdateHimself	Ø (Mock)	La mise à jour de A authentifié en tant que A fonctionne

6 – Suite de tests EntitySorter

Nom de la fonction test	**Jeu de données**	**Résultat attendu**
testIdAscSort	5 personnes avec pour id : 12, 8, 18, 18 et 5	Tri ascendant en fonction des id correct
testIdDescSort	5 personnes avec pour id : 12, 8, 18, 18 et 5	Tri descendant en fonction des id correct
testLastNameAscSort	5 personnes avec pour 'lastName': name1, name4, name2, name5 et	Tri ascendant en fonction des 'lastName' correct

	name1	
testLastNameDescSort	5 personnes avec pour ‘lastName’: name1, name4, name2, name5 et name1	Tri descendant en fonction des ‘lastName’ correct
testFirstNameAscSort	5 personnes avec pour ‘firstName’: name1, name4, name2, name5 et name1	Tri ascendant en fonction des ‘firstName’ correct
testFirstNameDescSort	5 personnes avec pour ‘firstName’: name1, name4, name2, name5 et name1	Tri descendant en fonction des ‘firstName’ correct