Institut Universitaire de Technologie
Aix+Marseille Université

Projet Java

Gestionnaire de mots de passe

Documentation

DUFOUR Mattéo
MUNOZ Matteo

Sommaire

Introduction

Problématique

On a souvent l'habitude de s'inscrire sur des sites ou sur des réseaux sociaux lorsque l'on navigue sur Internet.

Problème : comment retenir tous les mots de passe ?

La solution se trouve dans un **gestionnaire de mots de passe** : protégé par un mot de passe global, il permettrait de stocker une liste entière de mots de passe, de sorte à ce qu'il n'y en ait plus qu'un seul à retenir.

Présentation du projet

L'utilisateur doit pouvoir **stocker** des mots de passe avec différents champs à remplir : le nom du site/réseau social sur lequel est réalisée l'inscription, le lien, le nom d'utilisateur ou l'adresse mail, le mot de passe, une brève description si besoin, et une date d'expiration du mot de passe.

Une fois un mot de passe créé, on doit pouvoir le **supprimer** ou **modifier** l'un de ses champs rempli au préalable. Une des spécificités du gestionnaire est qu'il doit alerter l'utilisateur si l'un de ses mots de passe arrive à expiration.

Toutes ces données doivent être **chiffrées et stockées** dans un fichier pour ne pas laisser les mots de passe en clair.

Analyse

Classes utilisées

Lors de la conception de nos classes, nous avons essayé de respecter une **architecture Modèle-vue-contrôleur** dans la mesure du possible.

- Classes appartenant à la **vue** :

	Description
`Principal.java`	**Initialise** le Gestionnaire ; affiche une fenêtre d'inscription si aucun mot de passe global n'a déjà été créé, ou affiche une fenêtre de connexion si un mot de passe global existe.
`SignUpWindow.java`	Cette fenêtre permet la **création** d'un mot de passe global qui sera utilisé par la suite pour verrouiller/déverrouiller le Gestionnaire.
`LoginWindow.java`	Cette fenêtre permet de **se connecter** au gestionnaire. Est accessible uniquement si un mot de passe global a été créé au préalable.
`ManagerWindow.java`	La **fenêtre principale** du Gestionnaire, où diverses actions sont possibles : ajouter/modifier/supprimer un mot de passe.
`AddPassword.java`	Cette fenêtre permet de **créer** un mot de passe dans le Gestionnaire.
`DialogMessage.java`	Permet de **réutiliser** des fenêtres de dialogue.

- Classes appartenant au **contrôleur** :

	Description
`ProcessingAndHashing.java`	Contient **2 fonctions** : - `passwordProcessing()` : traite le mot de passe global de `SignUpWindow.java` (vérifie que les critères de sécurité soient respectés) - `mainPasswordHashing()` : chiffre le mot de passe global et le stocke dans un fichier `hashed.dat` créé
`LoginController.java`	Contient **2 fonctions** : - `passwordReset()` : affiche une option qui lance une procédure pour réinitialiser le mot de passe global - `passwordComparison()` : compare le mot de passe saisi pour se connecter avec le mot de passe global
`FileEncrypterDecrypter.java`	Chiffre/déchiffre les données.
`ManagePassword.java`	Permet d'ajouter et de supprimer un mot de passe. Il y a une méthode qui permet d'enregistrer les données dans le fichier, et une autre qui permet de les lire.

`Singleton.java`	Garantit une instance pour récupérer une donnée.

- Classe appartenant au **modèle** :

	Description
`Password.java`	Contient la structure d'un mot de passe.
`ModeleTableObjet.java`	Contient la structure d'un tableau.

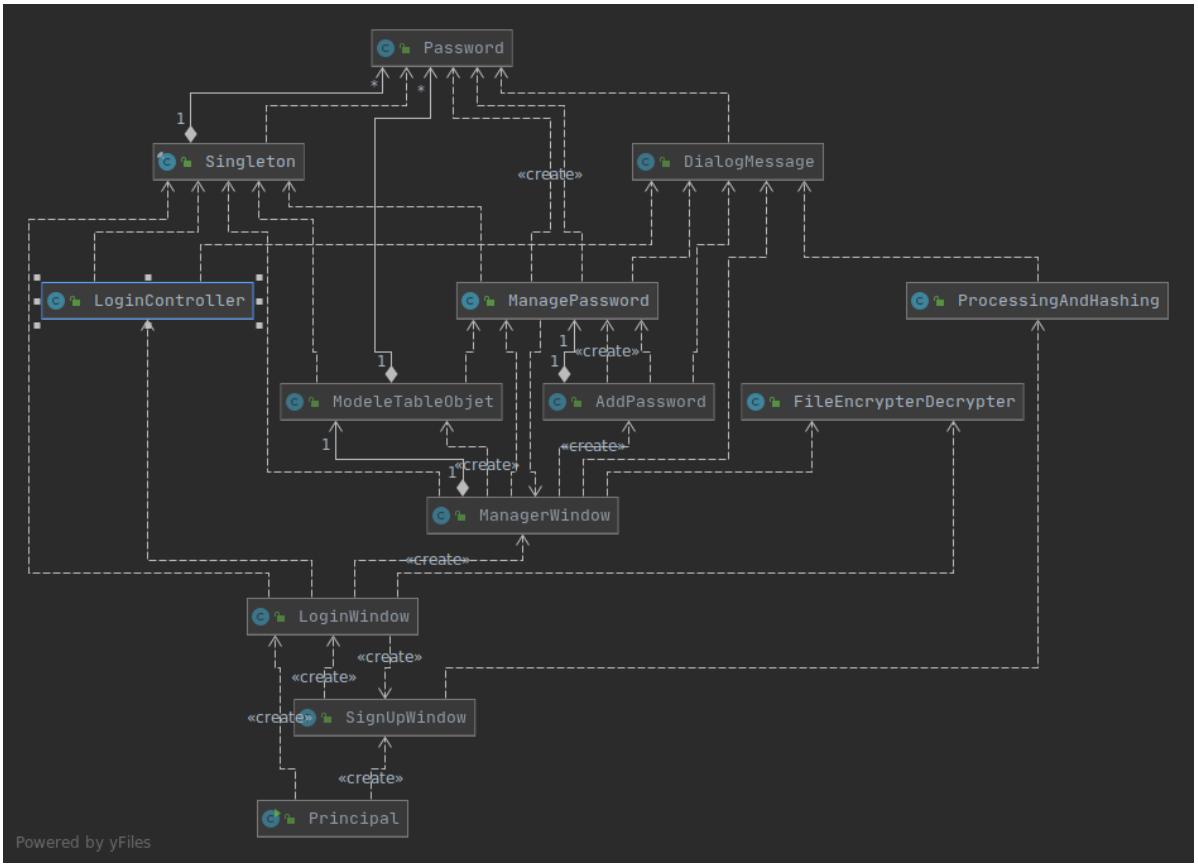
Librairies utilisées

Pour que le Gestionnaire soit une interface graphique, nous utilisons la bibliothèque **Swing**. Cela nous permet d'utiliser des composants tels que des `JLabel`, `JPasswordField`, `JTextField` entre autres.

Autres bibliothèques utilisées :

- *Guava*, pour le hachage en sha 256
- *Cipher*, pour le chiffrement des données des mots de passe
- *Jackson*, pour la gestion des fichiers json
- *JDatePicker*, pour choisir une date dans un calendrier

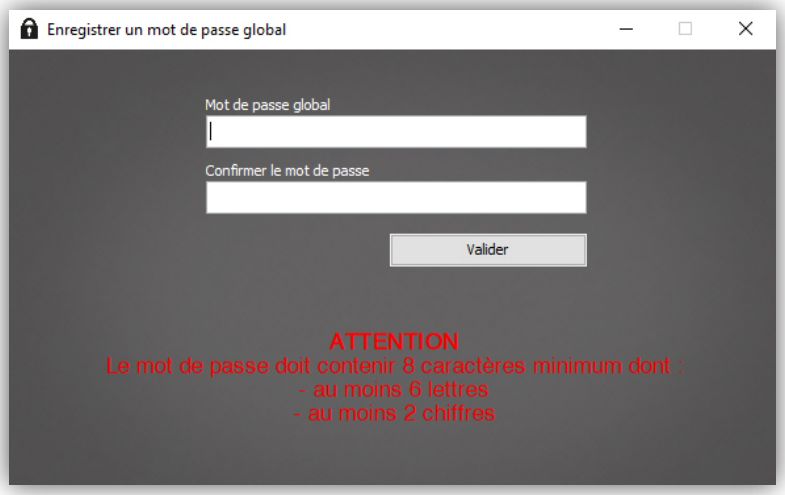
Relation entre les classes



Fonctionnement global

Étape 1 : création du mot de passe global

Au lancement du Gestionnaire, l'utilisateur doit créer un mot de passe global qui servira à le verrouiller/déverrouiller.

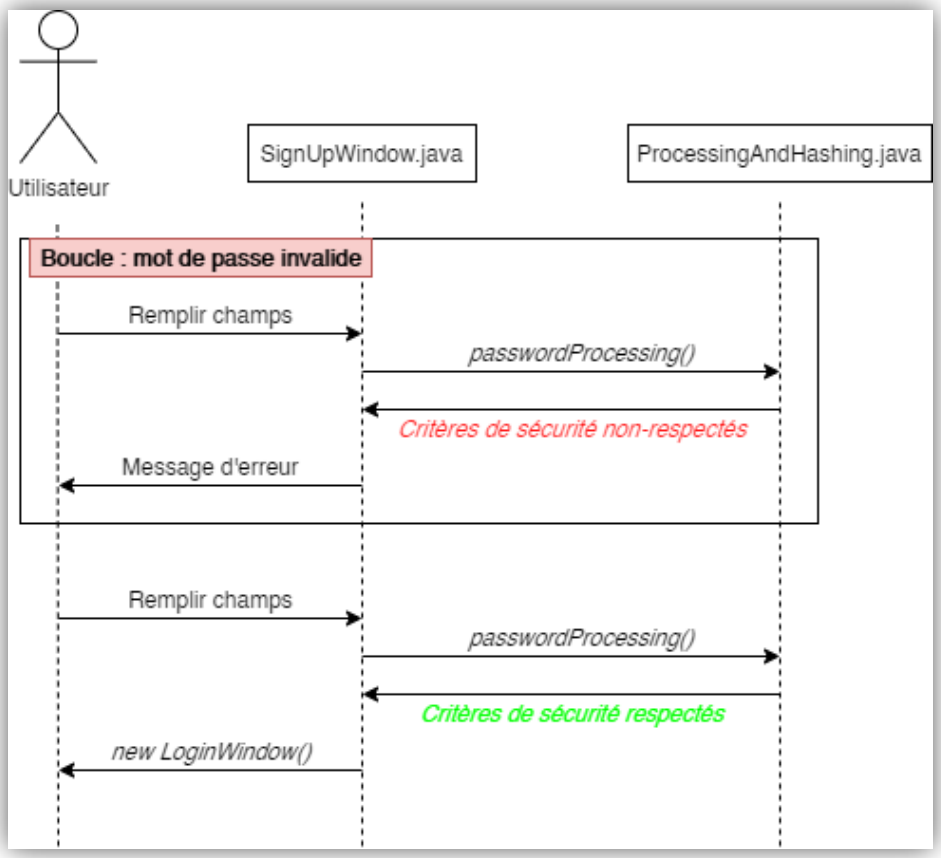


Critères de sécurité à respecter :

> 8 caractères mini dont :
> au moins 6 lettres
> au moins 2 chiffres

Le programme envoie un message d'erreur si ces critères ne sont pas respectés et si le mot de passe saisi dans le premier champ ne correspond pas à celui saisi dans le deuxième champ.

Diagramme de séquence pour l'étape 1 :



Étape 2 : connexion au Gestionnaire

Une fois le mot de passe global créé, l'utilisateur doit se connecter au Gestionnaire.

Le programme envoie un message d'erreur si le mot de passe saisi ne correspond pas au mot de passe enregistré lors de la création du mot de passe global.

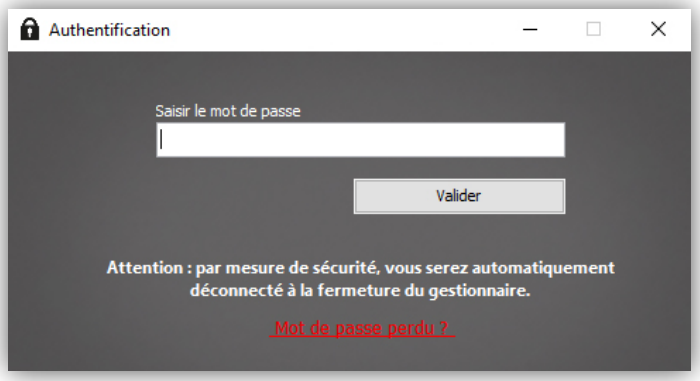
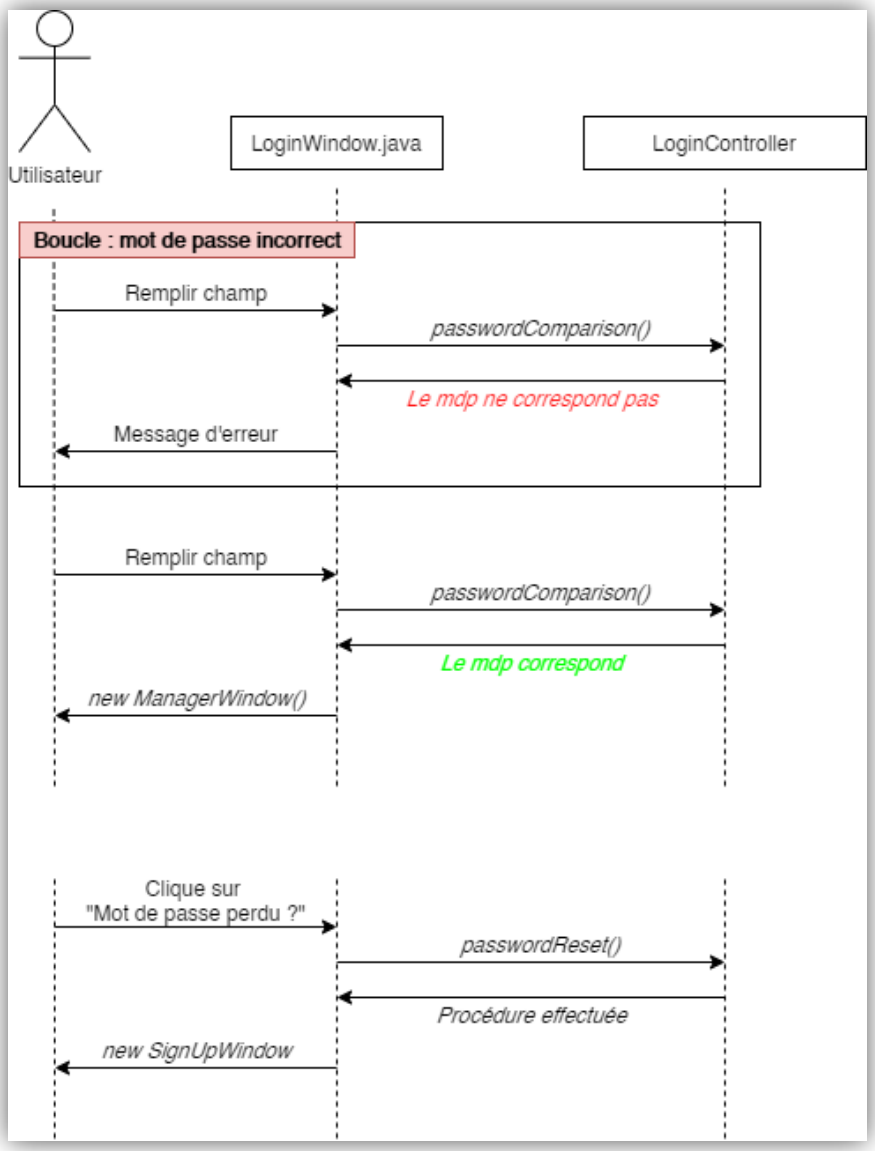
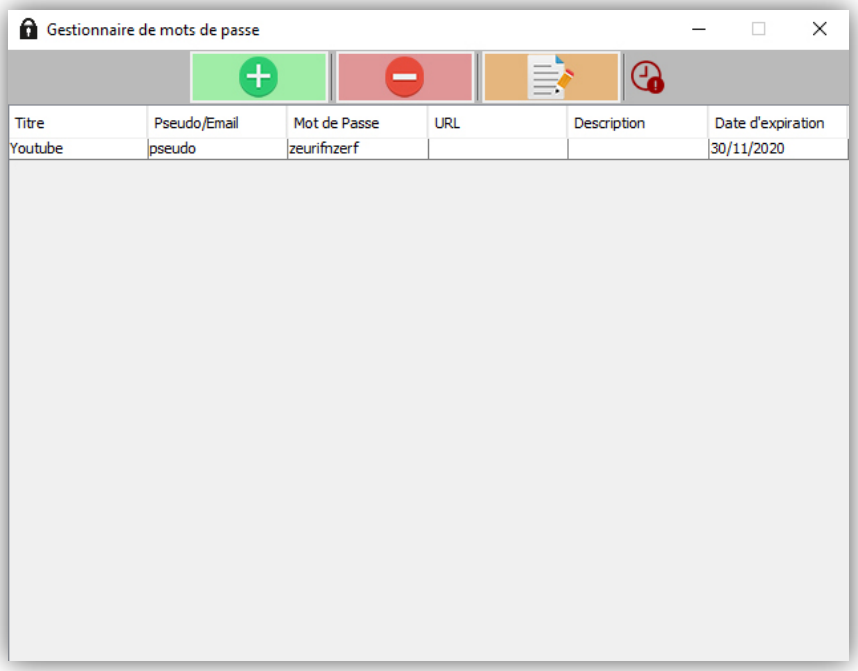


Diagramme de séquence pour l'étape 2 :

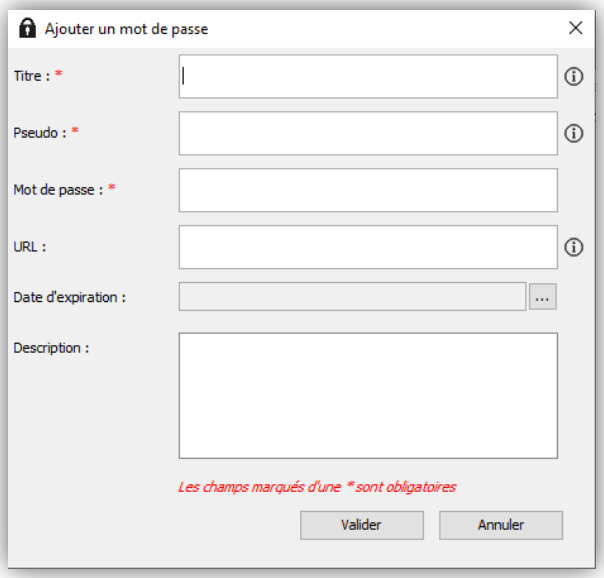


Étape 3 : utilisation du Gestionnaire



Liste des fonctionnalités :

- **Ajouter** un mot de passe
- **Supprimer** un mot de passe
- **Modifier** un mot de passe (non-aboutie)
- Afficher la liste des mots de passe qui **expirent** dans une période inférieure à 5 jours.



Réalisation

Choix techniques

- Un mot de passe est stocké dans un fichier JSON pour plusieurs raisons : le traitement des propriétés du mot de passe est plus facile, et une bibliothèque existe déjà, ce qui nous facilite la tâche.
- Fonctionnement hors-ligne.

Hachage et chiffrement des données

- Le mot de passe global est haché en sha256 avec la bibliothèque *Guava* de Google, c'est une méthode sûre et elle permet de comparer plus facilement le mot de passe saisi avec le mot de passe global enregistré lors de la connexion.
- Les autres données sont chiffrées en AES (Advenced Encryption Standard).

Fichiers créés

- `hashed.dat` : contient le mot de passe global haché.
- `data.json` : contient les données (propriétés des mots de passe créés dans le Gestionnaire) chiffrées.

Utilisation

Configuration requise

Prérequis :

- Avoir Maven
- Avoir le jdk, version Java 11

Mode d'emploi

Pour compiler :

- Se placer à la racine
- Commande `mvn package`

Pour exécuter :

- Se placer dans **/target**
- Exécuter `java -jar Password-Manager-Beta_1.1-jar-with-dependencies.jar`

Conclusion

Bilan

Notre Gestionnaire de mots de passe est bien abouti, malgré une fonctionnalité manquante (modifier un mot de passe) ; il est totalement fonctionnel et convient à une utilisation régulière.

Malgré tout, il peut encore être amélioré : voir les sections suivantes.

Optimisations possibles

Liste non-exhaustive d'optimisations possibles :

- **Définir un parent** pour les fenêtres de dialogues, afin qu'elles s'affichent centrées avec la fenêtre parent.
- Créations de « **helpers** » pour définir et utiliser des éléments graphiques en dehors des constructeurs des vues.
- Icônes de **meilleure qualité**.
- Utiliser un **layout**, plutôt que le définir sur *null* et utiliser des coordonnées à la place. Cela aurait permis de centrer les éléments graphiques proprement.
- Faire en sorte que les mots de passe soient **directement chiffrés** à leur création.
- Permettre à l'utilisateur de **définir le délai** pour avertir des mots de passe expirant bientôt (moins de 5 jours par défaut actuellement).

Extensions possibles

Nous aurions pu implanter les fonctionnalités suivantes :

- Possibilité de **modifier** un mot de passe (fonctionnalité non-aboutie).
- Possibilité de **générer** un mot de passe aléatoire lors de la création d'un mot de passe dans le Gestionnaire.
- Possibilité de **dupliquer** une entrée.
- Possibilité d'**importer** un fichier contenant des mots de passe pour les stocker dans le Gestionnaire.
- Possibilité d'**exporter** le fichier.
- Possibilité de **masquer** et d'**afficher** un mot de passe dans le tableau à volonté.
- Permettre de **rechercher** un mot de passe.
- Permettre de **trier** les mots de passe par propriété.