



Projet Java

**Gestionnaire de mots de passe**

*Documentation*

DUFOUR Mattéo

MUNOZ Matteo

Sommaire

[Introduction 3](#_Toc57148735)

[Problématique 3](#_Toc57148736)

[Présentation du projet 3](#_Toc57148737)

[Analyse 4](#_Toc57148738)

[Classes utilisées 4](#_Toc57148739)

[Relation entre les classes 5](#_Toc57148740)

[Fonctionnement global 5](#_Toc57148741)

[Étape 1 : création du mot de passe global 5](#_Toc57148742)

[Étape 2 : connexion au Gestionnaire 6](#_Toc57148743)

[Étape 3 : utilisation du Gestionnaire 7](#_Toc57148744)

[Réalisation 8](#_Toc57148745)

[Choix techniques 8](#_Toc57148746)

[Hashage et chiffrement des données 8](#_Toc57148747)

[Fichiers créés 8](#_Toc57148748)

[Utilisation 9](#_Toc57148749)

[Mode d’emploi 9](#_Toc57148750)

[Configuration requise 9](#_Toc57148751)

[Conclusion 10](#_Toc57148752)

[Bilan 10](#_Toc57148753)

[Optimisations possibles 10](#_Toc57148754)

[Extensions possibles 10](#_Toc57148755)

# 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 spécificité 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 **cryptées et stockées** dans 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. |
|  |  |

* 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 |
|  |  |
|  |  |
|  |  |

* Classe appartenant au **modèle** :

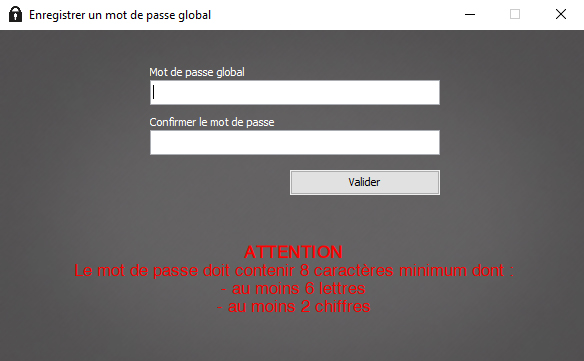
|  |  |
| --- | --- |
|  | **Description** |
| Password.java | Contient la structure d’un mot de passe. |

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.

## 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 :

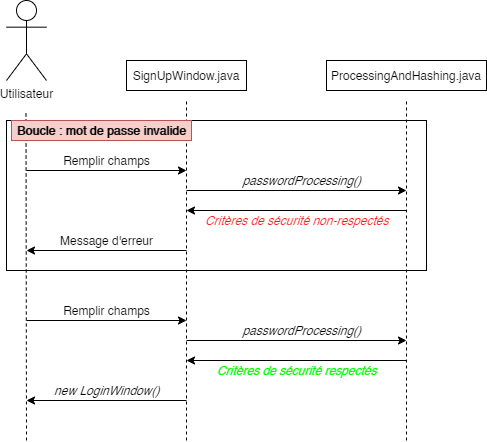
8 caractères mini dont :

au moins 6 lettres

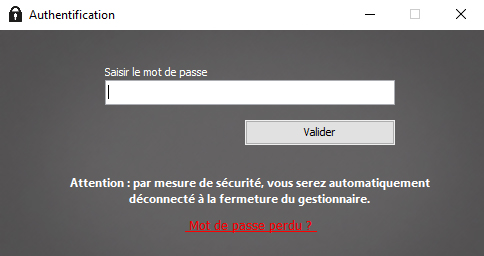
au moins 2 chiffres

*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 :**



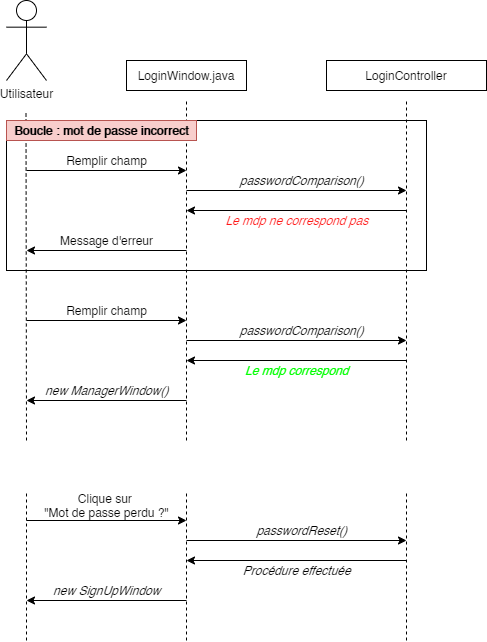
### Étape 2 : connexion au Gestionnaire



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

# Réalisation

## Choix techniques

## Hashage et chiffrement des données

## Fichiers créés

# Utilisation

## Mode d’emploi

## Configuration requise

# Conclusion

## Bilan

## Optimisations possibles

* Définir un parent pour les fenêtres de dialogues, afin qu’elle s’affichent centrées avec la fenêtre parent.

## Extensions possibles

Nous aurions pu implanter les fonctionnalités suivantes :

* 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