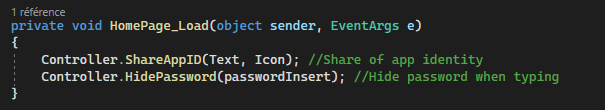
Explication du code

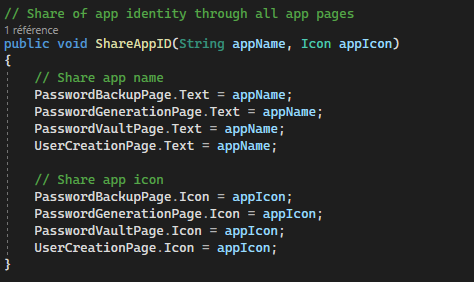
Page d’accueil

Pour que toutes les pages de l’application partagent l’identité de l’application (son nom et son logo), depuis la page d’accueil « HomePage », on fait appel à la méthode « ShareAppID » de la page « Controller » en lui donnant en paramètres le nom du formulaire et l’icône qui lui est associé. Ce partage a lieu au chargement du formulaire comme le prouve le nom de la méthode événementielle : « … \_Load »

[[1]](#footnote-1)

CODE CACHÉ

La page dédiée au contrôleur connaissant toutes les pages de l’application transmet ces informations aux autres pages.



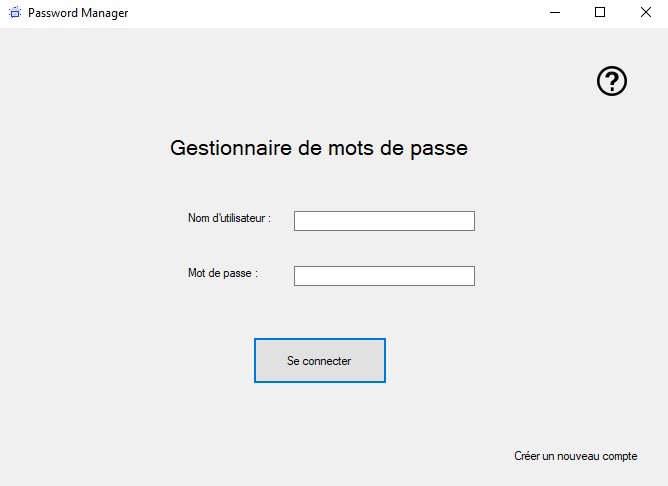
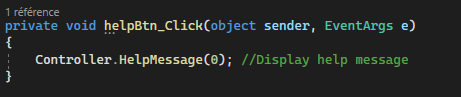
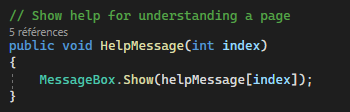


Figure : Identité de l'application

Un autre point commun entre les différentes pages de l’application est la présence d’un bouton d’aide (?). Après un clic sur ce bouton, un message est affiché grâce à la méthode « HelpMessage » du contrôleur.



La méthode « HelpMessage » affiche le message d’aide via la propriété « Show » d’une « MessageBox ». Celle-ci se réfère à une liste de type « String » et affiche un message en fonction de la valeur reçue en paramètre.



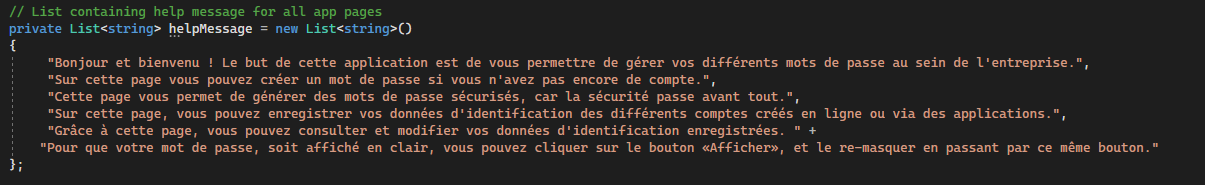
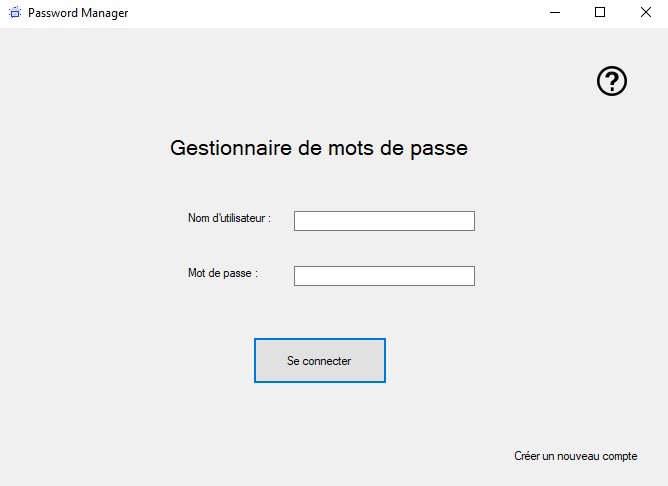
Voici les différents messages contenus dans la liste. Il y en a cinq, pour les cinq pages de formulaire.

Figure : Bouton d'aide



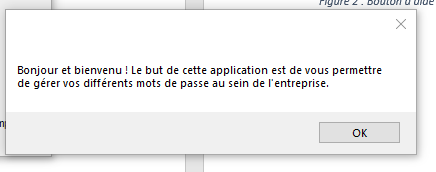
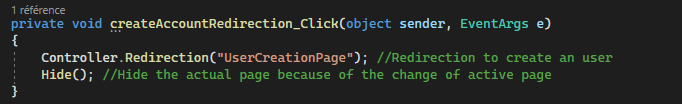
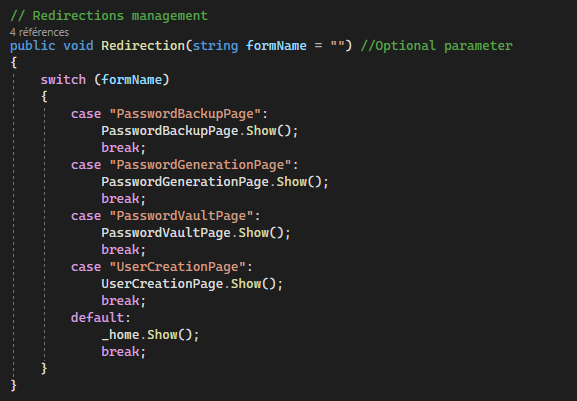


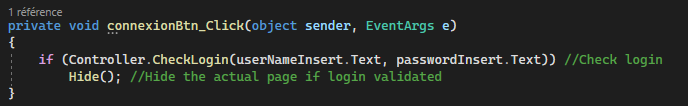
Figure : Message bouton d'aide

Gestion de la navigation

A chaque changement de page, la page précédente est fermée.

Ce changement de page a toujours lieu après une manœuvre de l’utilisateur (clic sur un bouton ou un label). Dans les méthodes évènementielles chargées de gérer ces redirections, on appelle le contrôleur en spécifiant le nom de la page vers laquelle on aimerait se diriger.

Dans certains cas, la redirection est gérée directement dans le contrôleur. Mais la fermeture d’une page se faisant toujours dans le code qui lui est propre, certaines méthodes du contrôleur retournent des valeurs booléennes aux formulaires qui font appel à elles. C’est le cas par exemple pour « CheckLogin » :



Si et uniquement si la valeur retournée est positive, la page se ferme.

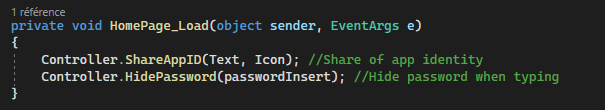
L’utilisateur a ainsi la possibilité de retenter pour la page d’authentification, la connexion à l’application.

Sécurité

Les différents éléments sécuritaires implémentés dans cette application,

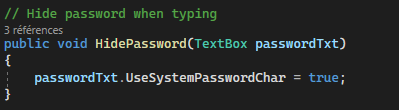
Mot de passe caché

Au moment de s’authentifier et de se connecter, l’utilisateur doit entrer un mot de passe ou en définir un. Durant cette phase de l’utilisation de l’application, il a été jugé utile de cacher son mot de passe de la vue de tous, afin de garantir la confidentialité de ses données.

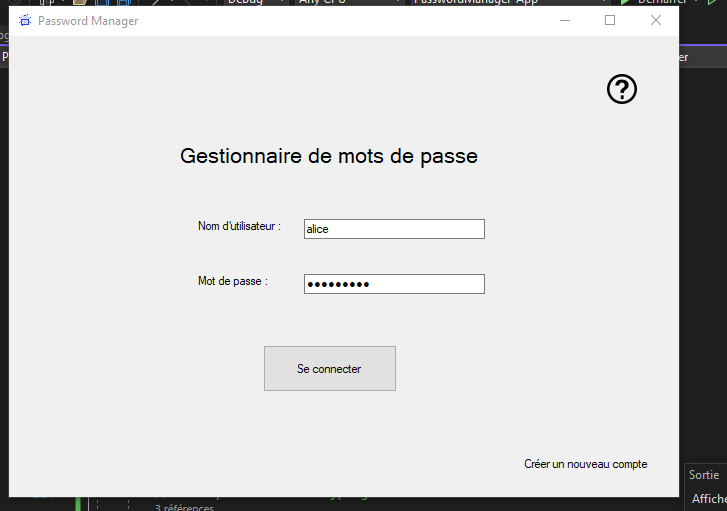


CODE CACHÉ

La méthode « HidePassword » qui prend en paramètre la textbox du mot de passe se charge de ça de la manière suivante :

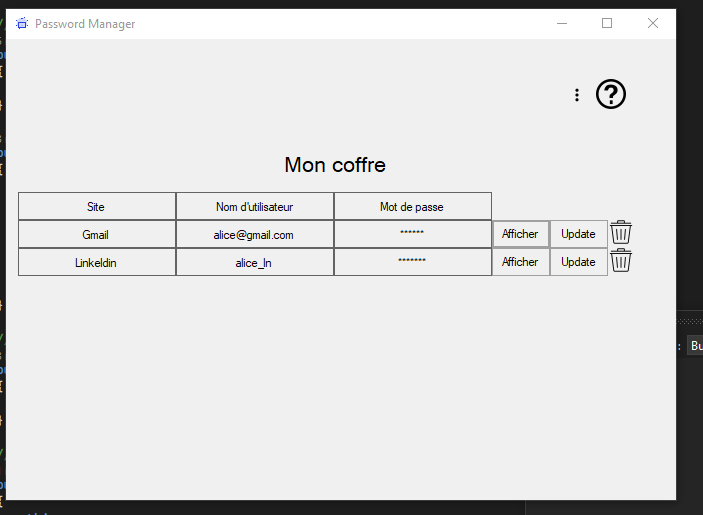


Cette propriété va convertir chaque entrée de l’utilisateur en un caractère codé :



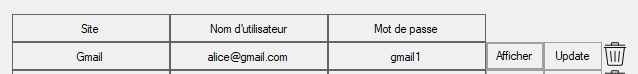
Et bien qu’accessible lors de la création du formulaire, il a été choisi, en prévision de l’évolution de l’application (gestion de la visibilité par l’utilisateur), de déjà implémenter une méthode pour gérer ce besoin.

Ce même souci de cacher le mot de passe, se retrouve dans la page « Mon coffre » :



Ici l’utilisateur pour voir son mot de passe, doit cliquer sur le bouton « Afficher ».

Celui-ci est ensuite affiché en clair comme suit :



Le bouton permet également de retourner à l’état d’origine du mot de passe ; c’est-à-dire caché.

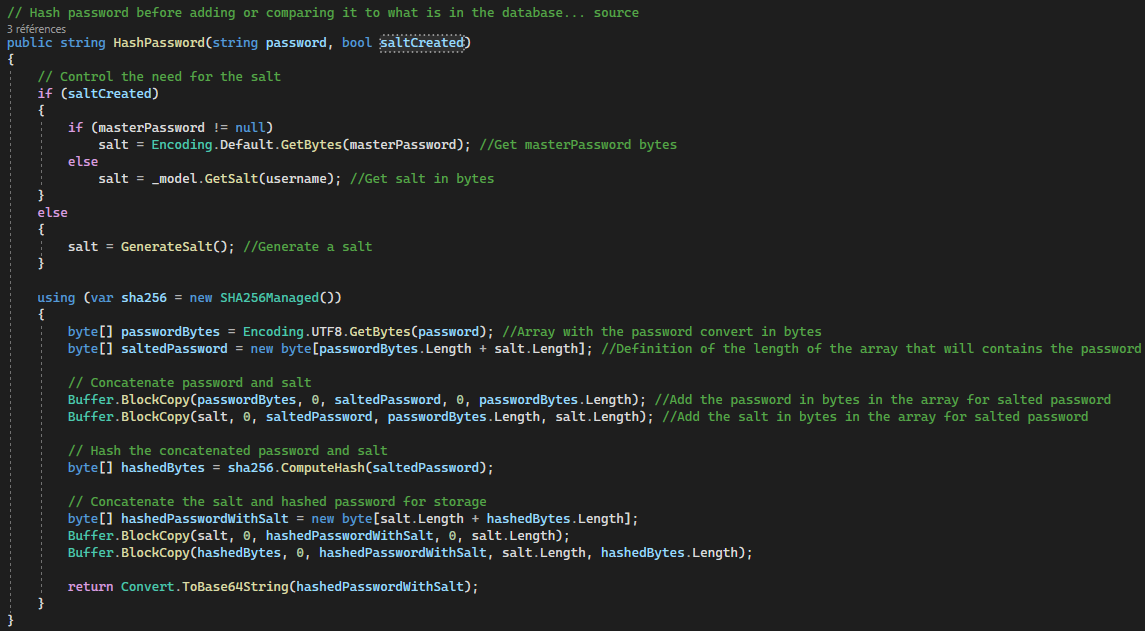
Lors de la sauvegarde du mot de passe, il n’a pas été jugé utile d’intégrer cette même logique, pour permettre à l’utilisateur de facilement revenir sur son mot de passe s’il souhaite le modifier. Ce point sera expliqué en détail plus loin.

Hachage de mot de passe

Le hachage du mot de passe intervient à différents moments dans l’application :

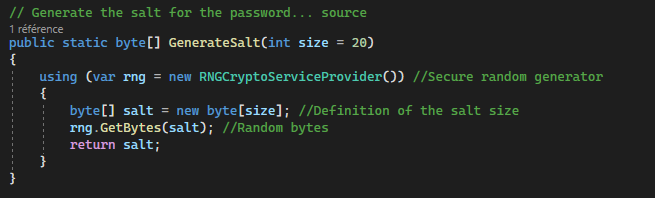
* Lors de l’authentification
* Lors de la création d’un compte
* Lors de la sauvegarde d’un mot de passe

Cependant, pour gérer ce hachage, une seule méthode est utilisée :



D’entrée, grâce au paramètre « saltCreated », on identifie s’il y a besoin d’un sel.

Si tel est le cas, un sel est généré grâce à la méthode « GenerateSalt » :



Seule la création de compte nécessite qu’un sel soit généré.

Autrement, comme la raison de l’appel de méthode n’est pas su, on vérifie si la variable « masterPassword » est défini. Cette variable n’est définie qu’à deux moments dans le programme : suite à l’authentification et après la création de compte. Si cette variable est nulle, aucun utilisateur n’est actuellement connecté et l’utilisateur serait en train de se connecter à l’application. Son sel est donc récupéré dans la base de données. Si au contraire la variable « masterPassword » est défini, sa valeur est transformée en bytes.

Logique pour afficher les mots de passe.

MessagesBox

Les « MessageBox » tout au long du programme, permettent comme expliqué précédemment d’afficher des messages pour guider l’utilisateur.

Celles-ci sont interviennent à différents moments :

Dans la page d’accueil :

* Pour informer l’utilisateur de la non validité de ses identifiants et l’inciter à créer un compte si nécessaire.

Dans la page « Créer un compte » :

* Pour s’assurer que

Avant de

Coffre-fort

1. Le code n’étant pas lié à l’explication donnée, celui-ci a été caché. [↑](#footnote-ref-1)