Université du Québec à Rimouski

Campus de Lévis

RAPPORT

TRAVAIL PRATIQUE #2

Par

Bastien Goulet et Cédrick Larrivée

Travail présenté à M. Martin Arsenault.

Dans le cadre du cours Sécurité informatique.

INF36207-MS

25 février 2023

Table des matières

[Algorithme MD5 3](#_Toc128245293)

[Application dictionnaire 3](#_Toc128245294)

[Application hachage 3](#_Toc128245295)

[Solution 3](#_Toc128245296)

[Résultats 3](#_Toc128245297)

[Conclusion 3](#_Toc128245298)

[Annexes 3](#_Toc128245299)

[Références bibliographiques 3](#_Toc128245300)

# Algorithme MD5

# Application dictionnaire

L’application dictionnaire est relativement simple d’utilisation. Entrez dans les bons champs la longueur minimale et maximale. Par la suite, sélectionner les différents caractères que vous voulez dans la génération de vos mots de passe. Pour ce faire,e il vous suffit de cocher les options qui vous intéressent, notez que vous pouvez aussi entrer des caractères au choix dans le champ à cet effet. Vous devrez ensuite sélectionner un emplacement pour sauvegarder votre fichier ainsi que le nommer. Finalement, il ne vous reste plus qu’a cliqué sur générer et de patienté avec votre café pour que votre dictionnaire de mots de passe de génère à l’endroit désiré. Évidemment si l’utilisateur ne remplit pas une de ces conditions, un message d’erreur apparaitra à l’écran.

# Application hachage

L’application de hachage est elle aussi très simple d’utilisation. Premièrement, sélectionner un dictionnaire de mots de passe dans la première fenêtre. Quand cela sera fait, une autre fenêtre s’ouvrira. Vous pourrez voir en haut à droite le nombre de tentatives ainsi que le temps écoulé. Au centre de la fenêtre se trouve une zone pour entrer votre hache qui sera décryptée quand vous appuierez sur le bouton validé. Enfin, vous trouverez en bas à droit le nombre de mots dans le dictionnaire.

## Solution

|  |  |
| --- | --- |
| 937557fac5cffc250ccf72031474078b | G2Ma1N |
| 2414766fb5121dfebdf220d4b8a550a0 | Ma1s0N |
| 973d2d342378f637aeeb9ec96f5a4b46 | B0nB0n |
| 6e63c6bf94b66e04ff2f48be546f0110 | ababbababb |
| 90b2360c704e0a805cee6dd6fd71eeaa | zyxyzxyxxz |
| 154e5cae63af27012426896ab9da2ac0 | divergances |
| 139230ff1ef21bb767bb9b475cf99873 | donnnaient |
| 7746fcc0b507887866f8bd227b37122c | Donetsks |
| db50a77debe1141e236bfefb468440ab | Eau-de-viiie |

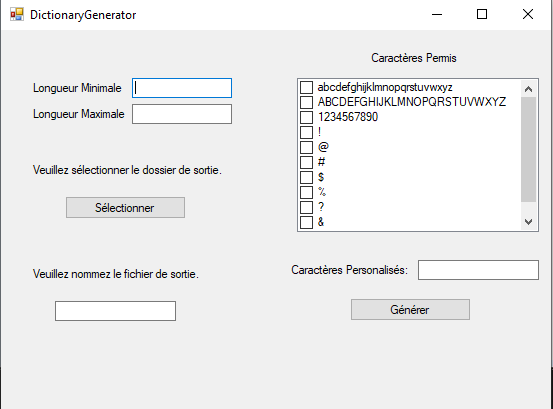
# Résultats

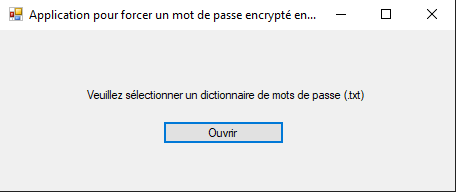
L’utilisation de l’architecture en couches n’était pas un bon choix sans en être un mauvais. Par contre, cela reste une bonne pratique, car cette architecture permet d’avoir un projet qui est bien structuré. Nous avons rencontré un problème lors de la génération des mots de passe pour le dictionnaire. Une approche avec fonction récursive a été utilisée. Cela causait des problèmes de débordement de mémoire quand l’application devait générer des mots à 5 et plus. Après recherche, nous avons trouvé une solution. L’utilisation de yeild lors du retour dans des fonctions de type IEnumerable permet de pallier à ce problème, car il permet de donner la prochaine valeur d’une itération quand il est utilisé.

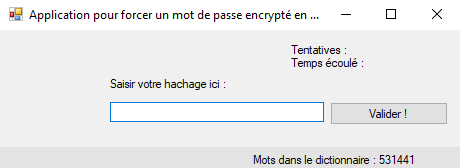
# Conclusion

En conclusion, nos applications répondent bien aux divers critères qui étaient exigés. Notre application dictionnaire permet la création de dictionnaires volumineux. Aucun problème n’a été soulevé pour générer un dictionnaire de plus 56 gigaoctets. En ce qui concerne notre application de hachage, elle aussi fonctionne bien en plus d’être plus simple d’utilisation que notre application de génération de dictionnaire.

# Annexes







# Références bibliographiques

Acamar. “Openfiledialog Does Not Close.” *Openfiledialog Does Not Close*, 20 Nov. 2013, https://social.msdn.microsoft.com/Forums/vstudio/en-US/e32fd547-c9ac-4287-a0a8-cb777fe67da3/openfiledialog-does-not-close?forum=vbgeneral.

Adegeo. “Make Thread-Safe Calls to Controls - Windows Forms .NET Framework.” *Make Thread-Safe Calls to Controls - Windows Forms .NET Framework | Microsoft Learn*, https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/how-to-make-thread-safe-calls-to-windows-forms-controls?view=netframeworkdesktop-4.8.

BillWagner, and pkulikov. “Yield Statement - Provide the next Element in an Iterator.” *In an Iterator | Microsoft Learn*, 12 Feb. 2022, https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/statements/yield.

Dotnet-Bot. “Filedialog Class (System.windows.forms).” *(System.Windows.Forms) | Microsoft Learn*, https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.filedialog?redirectedfrom=MSDN&view=windowsdesktop-7.0.