Marseille, 02/05/2024 - Achref El Mouelhi

Cas pratique : Python

Ce projet comporte quatre parties :

- Création d'un algorithme de chiffrement personnalisé
- Création d'interfaces graphiques Tkinter permettant la gestion de comptes utilisateurs
- Création d'une interface graphique Tkinter permettant la gestion de fichiers
- Journalisation

1 Algorithme de chiffrement personnalisé

L'objectif de cette partie est de créer les fonctions et classes nécessaires pour le chiffrement d'un mot de passe. Pour chiffrer un mot de passe, on recommande d'utiliser une clé et un sel (salt).

- **sel** : une chaîne de caractères aléatoire, contenant le même nombre de caractère que le mot de passe, ajoutée au mot de passe avant le chiffrement.
- **clé** : un tuple contenant des nombres aléatoires compris entre 0 et 127.
  `len(clé) = len(mot_de_passe) + len(sel)`.

Pour chiffrer le mot de passe, il faut suivre les étapes suivantes :

1. générer un sel aléatoirement,
2. concaténer le mot de passe et le sel,
3. générer la clé aléatoirement,
4. chiffrer le mot de passe revient à rajouter au code ASCII (`https://www.ascii-code.com/fr`) de chaque caractère du résultat de l'étape 2 une valeur de la clé (en respectant l'ordre).

Exemple : pour chiffrer les caractères `'az'` avec la clé `(1, 60)` :

- (code ASCII (a) + 1) % 128 ⇒ (97 + 1) % 128 ⇒ 98 ⇒ caractère correspondant `'b'`
- (code ASCII (z) + 60) % 128 ⇒ (122 + 60) % 128 ⇒ 54 ⇒ caractère correspondant `'6'`

La version chiffrée de `'az'` avec la clé `(1, 60)` est `'b6'`.

Pour chaque utilisateur, nous souhaitons stocker dans une table `utilisateur` les informations suivantes :

- son nom,
- son prénom,
- son email (unique et à utiliser pour l'authentification),
- son genre (une énumération)),
- son mot de passe actuel chiffré et
- son mot de passe précédent chiffré (qui ne lui permet pas de s'authentifier).

Pour chaque mot de passe généré, nous souhaitons également stocker dans une table `compte` les informations suivantes :

- la clé,
- le sel,
- le type : une énumération qui indique si le sel et la clé permettent la génération du mot de passe actuel ou le précédent,
- l'identifiant de l'utilisateur.

Les deux tables `compte` et `utilisateur` ont une clé primaire auto-incrémentale.

Voici quelques indices pour la réalisation de cette partie :

- La fonction `ord(char)` retourne le code ASCII du caractère passé en paramètre.
- La fonction `chr(code)` retourne le caractère associé au code ASCII passé en paramètre.
- Le module `string` fournit des collections de constantes de chaînes de caractères.
- Créer une classe `CustomCrypt` avec une méthode `crypt()` pour le cryptage des mots de passe.
- Placer les fichiers de conversion et de génération de données aléatoire dans des fichiers utilitaires.

2 Gestion de compte utilisateur

L'application, via des interfaces Tkinter, doit permettre à l'utilisateur de réaliser les opérations suivantes

- créer un compte utilisateur
- s'authentifier
- gérer son compte une fois connecté. Supprimer son compte ou le modifier : le nouveau mot de passe saisi doit être différent du précédent

3 Gestion de fichiers Windows

L'application, via une interface Tkinter, doit permettre à l'utilisateur de

- choisir un répertoire Windows (utiliser `filedialog`),
- lister son contenu (fichiers et sous-répertoires) dans une `Listbox` ou `Treeview`,
- supprimer ou renommer les fichiers et les sous-répertoires affichés même s'ils ne sont pas vides.

4 Journalisation (logging)

Dès sa connexion et jusqu'à la déconnexion, toute opération réalisée par l'utilisateur doit être stockée dans un fichier de log (un fichier par jour). Le nom du fichier doit contenir la date du jour. Chaque ligne dans le fichier de journalisation doit contenir dans l'ordre :

- La date et l'heure de l'opération
- L'email de l'utilisateur
- Le type d'opération (connexion, modification de compte, suppression de compte, déconnexion, chargement de contenu d'un répertoire, suppression de fichier ou répertoire...)

5 Règles à respecter

- Application structurée et fichiers répartis en package
- Un seul fichier (`main`) à la racine du projet
- Un environnement virtuel pour le projet placé à la racine
- Un fichier `requirements.txt` à jour pour les dépendances
- Un dépôt **GitHub** pour le projet et un commit pour chaque tâche développée
- Pas de génération de code avec **ChatGPT**
- Code indenté et documenté
- Utilisation maximale de typage et validation par `MyPy`
- Respect des règles décrites dans les documents PEP
- Utilisation des requêtes préparées pour éviter les injections SQL