SAE 3.02 :

Développement d'Applications Communicantes

I. Introduction

Date limite de soumission : Lundi 23 octobre 23:59

Auteurs : GRONDIN Dany ARHIMAN Ludovic CAYAMBO Pierre

Notre compte rendu vise à explorer en profondeur notre projet de développement d'applications communicantes, en mettant l'accent sur notre gestionnaire de mots de passe sécurisé, un composant clé de notre solution. Nous allons également passer en revue les technologies que nous avons exploitées et les étapes cruciales du développement.

Le besoin de sécuriser les informations sensibles, telles que les mots de passe, est devenu incontournable dans le monde connecté d'aujourd'hui. C'est dans ce contexte que notre projet prend tout son sens. Nous avons conçu une solution complète qui permet aux utilisateurs de gérer leurs identifiants et mots de passe de manière robuste et sécurisée.

Motivations pour la réalisation du projet

Plusieurs motivations clés ont influencé notre décision de choisir ce projet :

Renforcement de la sécurité des informations : Dans un environnement numérique de plus en plus exposé aux menaces, la sécurité des données est devenue une préoccupation majeure. La réalisation de ce projet est motivée par le désir de fournir aux utilisateurs un moyen fiable et sécurisé de stocker et gérer leurs informations de connexion. En offrant un environnement de stockage de mots de passe crypté, nous visons à aider les individus à protéger leurs données sensibles contre les cyberattaques.

Simplification de la gestion des mots de passe : De nombreux utilisateurs sont confrontés au défi de gérer un grand nombre de mots de passe pour leurs comptes. La complexité croissante des exigences de sécurité (mots de passe complexes, expiration régulière, etc.) a rendu cette tâche encore plus ardue. Notre projet vise à simplifier la vie des utilisateurs en centralisant la gestion de leurs mots de passe. En stockant de manière sécurisée ces informations sensibles, notre solution permet aux utilisateurs de se concentrer sur l'utilisation de leurs applications sans les tracas liés à la gestion des mots de passe.

Développement de compétences techniques avancées : En plus de répondre aux préoccupations liées à la sécurité, ce projet offre une opportunité inestimable de développer des compétences techniques. Notre équipe à l'occasion d'explorer des domaines complexes tels que le développement d'applications, la gestion de bases de données, la sécurité informatique et la gestion de projet. La poursuite de ces compétences est cruciale pour l'avenir, que ce soit pour relever des défis similaires ou pour évoluer dans un environnement technologique en constante évolution.

L'équipe qui a collaboré sur ce projet est composée de GRONDIN Dany, ARHIMAN Ludovic, CAYAMBO Pierre, chacun apportant des compétences pour garantir le succès du projet. Voici les différents profils des membres de l'équipe :

ARHIMAN Ludovic	A joué un rôle essentiel dans le développement du backend du projet. Il a travaillé sur le développement de la base de données, la gestion de la sécurité des données et d'autres aspects techniques du projet liés à la sécurité des données, y compris le chiffrement des mots de passe du côté serveur, mais aussi sur la création des pages du font end

CAYAMBO Pierre 	Est un autre membre clé de l'équipe qui a également été impliqué dans le développement du backend, en se concentrant sur les aspects techniques liés à la sécurité des données et le chiffrement des mots de passe du côté serveur
GRONDIN Dany 	S'est concentré sur la partie du serveur qui gère l'interaction entre le client et le serveur ainsi que la récupération des données dans la base de donnée

Chaque membre de l'équipe a apporté ses compétences et ses connaissances spécifiques au projet, contribuant ainsi à son succès.

II. Code Source

Description Sommaire du Code Source

Notre code source est l'épine dorsale de notre projet de développement d'applications communicantes. Il s'agit d'une solution logicielle qui offre aux utilisateurs un environnement sûr pour gérer leurs données d'identification. Notre code source est principalement écrit en Python, avec l'utilisation de la bibliothèque Flask pour créer une API robuste qui facilite la communication avec les clients.

Organisation du Code

La structure de notre code source est conçue de manière modulaire pour permettre une meilleure maintenabilité et une extensibilité aisée. Nous avons divisé notre code en plusieurs modules distincts, chacun ayant une responsabilité spécifique :

- **HTML (dossier templates/)**
    - **Les modules d'affichage:** Ces pages servent à modifier les pages principales via l'intermédiaire d'Ajax. Nous retrouvons les fichiers suivants :

      `delete_pass.html` : modifie la page index.html en effaçant les mots de passes des utilisateurs via l'intermédiaire d'Ajax

      `fill_infos.html` : Permet d'afficher sur la div droite les informations relatives au mot de passe (nom du mot de passe, login et mot de passe stocké, etc.)

      `search.html` : Permet de faire la recherche de mot de passe en fonction de ce que l'on recherche.

- **Les pages principales:** Comme leur nom l'indique, ces pages sont des pages à part entière qui permettent de réaliser des actions précises et sont pour la grande majorité accessibles depuis un bouton de la page principale.

  `index.html` : Cette page est la page principale, elle permet via l'intermédiaire des pages énoncé plus haut d'afficher tous les mots de passes de l'utilisateur, de faire la recherche avec **`search.html`** , de pouvoir afficher les informations du mot de passe en cliquant sur l'icône prévu à cet effet ("i"), de supprimer les mots de passe avec l'icône de corbeille, mais cette page permet aussi de faire d'autre action que nous allons voir un peu plus bas.

  `passwords_gen.html` : Cette page est l'une des fonctionnalités affichées dans la barre latérale de la page **`index.html`**. Elle permet de générer des mots de passe et de les copier

  `add_password.html` : Cette page permet d'ajouter des mots de passes

  `register.html` : Permet d'enregistrer un nouvel utilisateur pour qu'il puisse utiliser le gestionnaire de mot de passes

  `edit_password.html` : Permet de modifier les mots de passe déjà enregistrer dans la base de données

  `login.html` : Permet à un utilisateur de se connecter au gestionnaire de mot de passe en entrant son mot de passe et son nom d'utilisateur

  `404.html` : Cette page est la page d'erreur du serveur qui est retourné si l'on souhaite à céder à une ressource inexistante du serveur

- **Python (dossier `package/`)**
  - **`run.py` :** Ce script python permet de lancer le serveur, de gérer les routes et de définir les fonctions et variable que l'on utilise sur les différentes pages via **`jinja2`**
  - **`db_config.py` :** Ce script permet de gérer la base de données SQL, de faire toutes les requêtes SQL nécessaires
  - **`encryption` :** Ce script permet comme son nom l'indique de chiffrer en AES-256-CBC les mots de passes des utilisateurs afin qu'il soit illisible dans la base de données. Il permet aussi de faire de la dérivation afin de générer une clé de chiffrement symétrique pour le chiffrement et déchiffrement

Choix d'Architecture

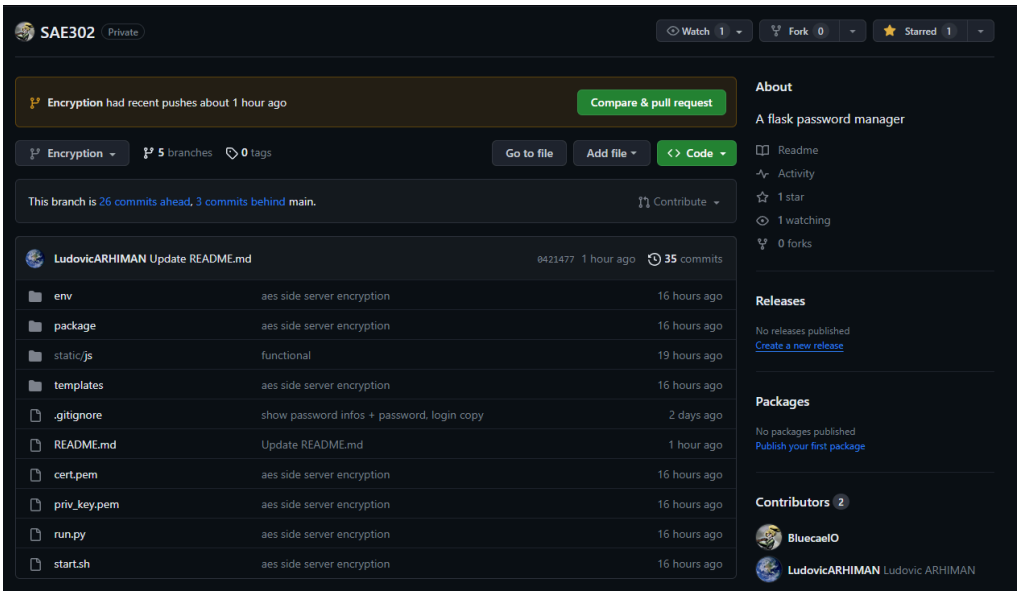
Le choix de l'architecture pour le développement de l'application de gestion de mots de passe repose sur plusieurs considérations essentielles. Flask, en tant que framework Python, est privilégié pour sa simplicité et sa flexibilité, ce qui facilite le développement et la maintenance du code. L'architecture modulaire est adoptée pour une organisation claire du code, favorisant l'extensibilité de l'application.

Nous avons choisi Python en raison de sa lisibilité et de sa richesse en bibliothèques, offrant ainsi une base solide pour le développement web. La sécurité étant une priorité, nous avons mis en place le chiffrement AES-256-CBC pour protéger les données sensibles. Pour la gestion des données, nous avons opté pour une base de données SQL en raison de sa robustesse et de ses performances.

Notre architecture a été pensée pour évoluer facilement, permettant l'ajout de nouvelles fonctionnalités sans compromettre la stabilité de l'application. Flask facilite cette adaptabilité. Enfin, nous utilisons des fichiers HTML pour l'interface utilisateur, ce qui s'intègre harmonieusement avec Flask, créant ainsi des pages web dynamiques pour offrir la meilleure expérience utilisateur possible. En résumé, notre choix d'architecture vise à privilégier la simplicité, la sécurité, la modularité et l'extensibilité, garantissant ainsi la robustesse et la maintenabilité de notre application.

Liens vers le Référentiel Git

Vous pouvez consulter notre référentiel Git pour accéder au code source complet de l'application.



III. Comment déployer l’application

Dans la section III de notre compte rendu, nous nous penchons sur la phase de déploiement de l'application. Pour mettre en service notre projet à partir du code source. Nous expliquons également comment déployer la base de données PostgreSQL et fournissons une liste des dépendances indispensables pour exécuter le projet.

Nous présentons deux approches distinctes pour le déploiement de l'application, chacune avec ses avantages et inconvénients :

1. **Déploiement sur la machine virtuelle fournie** :
    - Avantages :
        - Cette option est prête à l'emploi, nécessitant essentiellement une adresse IP fixe pour fonctionner (fonctionne quand même avec une adresse IP par DHCP).
        - Le déploiement est rapide et direct.
    - Inconvénients :
        - Elle n'est pas compatible avec tous les virtualiseurs, se limitant principalement à VMWare dans ce contexte.
        - De plus, elle n'est pas compatible avec toutes les architectures de processeur, se limitant aux processeurs x86, avec l'incapacité de prendre en charge l'architecture ARM en raison des limitations de la machine utilisée pour créer la machine virtuelle.
2. **Déploiement sur sa propre machine** :
    - Avantages :
        - Cette option offre une flexibilité maximale, permettant l'installation sur la machine de votre choix, adaptée à vos besoins.
    - Inconvénients :
        - Cependant, elle nécessite un déploiement manuel plus complexe, impliquant l'installation de la base de données, des dépendances et de l'application.
        - De plus, cela exige que pip3 et Python soient préalablement installés et correctement configurés sur la machine.
        - Il est vivement recommandé d'utiliser des environnements virtuels pour gérer les dépendances, ce qui peut représenter une étape supplémentaire.

Installation Manuelle

Avant de commencer le processus d'installation manuelle, nous devons nous assurer de satisfaire aux prérequis suivants :

Pré-requis :

- Nous devons disposer de Python 3.10 ou d'une version ultérieure installée sur nos systèmes.
- Nous devons avoir PIP3, l'outil de gestion de packages Python, pour l'installation de modules Python.
- Il est essentiel que Python soit configuré pour l'utilisation d'environnements virtuels, une bonne pratique pour isoler les dépendances du projet.

Comment déployer la base de données PostgreSQL

L'application pour fonctionner a besoin d'une base de données SQL pour fonctionner, pour notre projet, nous avons décidé d'utiliser la base de données PostgreSQL pour les raisons énoncées précédement.

Pour ce faire, nous devons installer PostgreSQL à partir de la commande suivante :

```
# apt install postgressql
```

Puis, on se connecte avec l'utilisateur postgres

```
# su postgres
```

Nous accédons à la base de données postgres

```
psql
```

Puis, nous modifions son mot de passe

```
 postgres=# \password postgres
```

La base de données est maintenant prête à l'emploi

À noter que cette machine va uniquement accepter la connexion en localhost, l'application doit donc être installée sur la machine qui héberge la base de données

Récupérer le code source

Pour récupérer le code source, nous allons utiliser l'outil **git** que nous avons utilisé plus tôt pour la programmation de l'application.

Tout d'abord, nous allons installer **git** via la commande suivante

```
# apt install git
```

Puis, on récupère le code source

```
git clone https://github.com/BluecaelO/SAE302.git
```

Exécution de l'application

On peut maintenant modifier le code en changeant le chemin des fichiers "cert.pem" et "priv_key.pem" en indiquant leurs **chemins absolut** dans le script **run.py**

```
289    if __name__ == '__main__':
290        app.run(debug=True,host="0.0.0.0",port=8080,ssl_context=('<chemin>/cert.pem', '<chemin>/priv_key.pem'))
```

On modifie aussi le script bash "start.sh" en modifiant le chemin du dossier env/ présent dans le code source ainsi que le chemin du fichier run.py

```
3    source <path_to_env_folder>/env/bin/activate
4    python3 <path_to_run.py>/SAE302/run.py
```

Une fois le code modifié, on peut le lancer avec le script “start.sh” qui va se charger d’utiliser le bon environnement virtuel et lancer le serveur

```
# chmod +x start.sh
./start.sh
```

On peut maintenant se rendre dans un navigateur et entrer l’adresse IP suivit du port 8080 (https://<ip>:8080) pour avoir accès au gestionnaire de mot de passe

IV. Technologie Utilisée

Les technologies choisies pour la réalisation de ce projet comprennent divers langages de programmation, protocoles réseau et méthodes de persistance, chacun ayant un rôle spécifique dans le système.

Description des technologies utilisées

Pour développer notre gestionnaire de mots de passe sécurisé, nous avons opté pour un ensemble de technologies bien adaptées à nos besoins. Ces technologies comprennent Python, Flask, PostgreSQL, ainsi que des protocoles réseau standard tels que HTTPS.

Justification du choix de ces technologies

Le choix de ces technologies repose sur divers facteurs. Python a été choisi pour sa simplicité, sa polyvalence et sa large adoption dans le domaine du développement web. Flask, un framework Python léger, a été retenu pour sa facilité d'utilisation et son support robuste pour la création d'API web. PostgreSQL a été sélectionné en raison de sa fiabilité en tant que système de gestion de base de données relationnelle, et son utilisation est conforme aux normes de sécurité élevées.

Avantages des Technologies Choisies :

Le choix de ces technologies présente plusieurs avantages clés.

- **Python** : Python est largement adopté, ce qui signifie qu'il dispose d'une vaste communauté de développeurs et facilite la résolution de problèmes. De plus, Python permet un développement rapide.
- **Flask** : Flask est simple à prendre en main et offre un environnement de développement flexible pour la création d'applications web.
- **PostgreSQL** : PostgreSQL est un système de gestion de base de données robuste et open source, idéal pour le stockage sécurisé des informations de connexion des utilisateurs.

Inconvénients des Technologies Choisies :

Cependant, il est important de noter que chaque technologie présente des inconvénients potentiels.

- **Python** : Bien que polyvalent, Python peut ne pas être la meilleure option pour des applications nécessitant des performances extrêmes.
- **Flask** : Bien que léger et facile à utiliser, Flask peut manquer de certaines fonctionnalités présentes dans des frameworks plus complexes.
- **PostgreSQL** : Bien que robuste, PostgreSQL peut être complexe à gérer pour les applications à grande échelle.

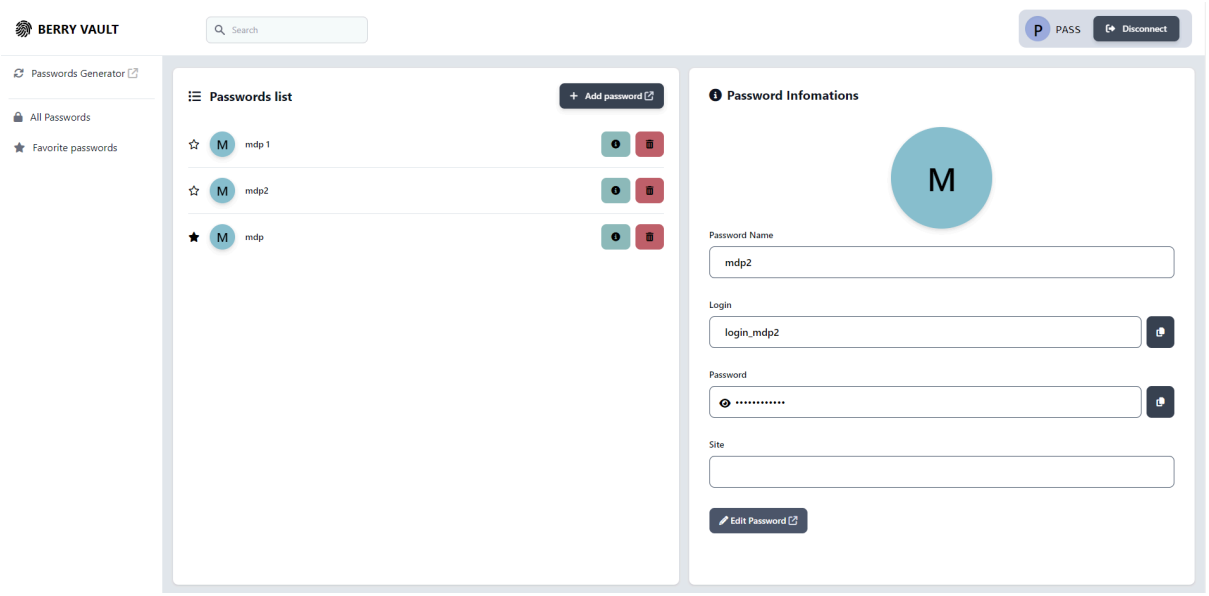
Dans l'ensemble, le choix de ces technologies pour la réalisation de notre gestionnaire de mots de passe a été motivé par une approche équilibrée entre la sécurité des données et la simplicité de développement.

V. Fonctionnalités de l'Application

- **Description des fonctionnalités principales**:

Le gestionnaire de mot de passe comprend l'ajout et la suppression de mot de passe, la modification de ces derniers par l'intermédiaire d'une interface WEB afin de permettre à l'utilisateur de pouvoir confier la gestion de ses mots de passes à l'application et non plus de les stocker sur des support physique ou numérique peut sécurisé.

- **Affichage de l'application** : L'application se présente comme ceci avec une barre latérale qui permet d'aller vers le générateur de mot de passe, d'afficher tous les mots de passe, d'afficher les mots de passe en favoris. Nous pouvons aussi retrouver deux sections au centre de l'écran, une sur la gauche qui permet d'afficher les mots de passe stockés par l'utilisateur, et une partie à droite qui permet de pouvoir voir et copier les logins et mots de passe sélectionnés. Nous pouvons aussi voir une barre de recherche qui permet de rechercher les mots de passe dans la base de données à partir de leurs noms. *Il convient de noter que le choix de développer l'application en anglais visait à toucher un public plus large, à simplifier l'expérience des utilisateurs et à rendre l'application accessible à une communauté mondiale. La simplicité de la langue anglaise et sa large adoption dans le domaine de la technologie ont joué un rôle clé dans cette décision, facilitant ainsi l'interaction avec un grand nombre d'utilisateurs.*



VI. Réussites et Challenges

Ce qui a Fonctionné :

Plusieurs éléments ont bien fonctionné pendant le développement de notre gestionnaire de mots de passe :

1. **Choix des Technologies** : Le choix des technologies, y compris Python, Flask, PostgreSQL, et le chiffrement AES-256-CBC, s'est avéré judicieux. Ces technologies ont contribué à créer un environnement sécurisé et convivial.
2. **Gestion des Mots de Passe** : La mise en œuvre efficace de la gestion des mots de passe a permis aux utilisateurs de stocker leurs informations sensibles en toute sécurité.
3. **Chiffrement** : Le chiffrement AES-256-CBC a été un succès pour garantir la sécurité des données stockées sur le serveur.

Difficultés Rencontrées :

Le projet nous a exposés à des difficultés techniques, en particulier en ce qui concerne la sécurité des données :

1. **Sécurité des Données** : La principale difficulté résidait dans la garantie de la sécurité des données stockées sur le serveur, notamment les mots de passe des utilisateurs.
2. **Chiffrement** : L'implémentation du chiffrement AES-256-CBC a nécessité une attention particulière pour minimiser son impact sur les performances tout en maintenant un haut niveau de sécurité.

Solutions Apportées aux Problèmes :

Pour surmonter ces défis, nous avons mis en place les solutions suivantes :

1. **Audit de Sécurité** : Un audit de sécurité complet (dans la limite de nos compétences en pentesting d'application web) a été effectué pour identifier et corriger les vulnérabilités potentielles. Cela a permis d'améliorer la protection des données.
2. **Optimisation du Chiffrement** : L'optimisation de l'implémentation du chiffrement AES-256-CBC a permis de minimiser son impact sur les performances tout en maintenant un haut niveau de sécurité.

Apprentissages Tirés des Challenges :

Ce projet nous a enseigné des leçons précieuses en matière de développement d'applications sécurisées :

1. **Sécurité prioritaire** : La sécurité des données est une priorité absolue. Investir du temps et des ressources dans la mise en place de mesures de sécurité solides est essentiel.
2. **Équilibrage des Performances** : Trouver un équilibre entre la sécurité et les performances est crucial. Le chiffrement doit être efficace, mais ne doit pas entraver de manière significative les opérations de l'application.
3. **Respect de la Vie Privée des Utilisateurs** : Le projet nous a rappelé l'importance de respecter la vie privée des utilisateurs. Nous devons être responsables de la manière dont nous gérons et stockons les données sensibles.
4. **Collaboration efficace** : La résolution des défis techniques, en particulier en matière de sécurité, a renforcé notre capacité à travailler en équipe. La communication et la collaboration sont devenues essentielles pour résoudre ces problèmes complexes.

VII. Perspectives d'Amélioration

Ce qui pourrait être ajouté ou amélioré dans le projet :

1. **Chiffrement Client-Side** : Une amélioration significative serait d'implémenter un chiffrement client-side, ce qui signifie que les données sensibles sont chiffrées localement sur l'appareil de l'utilisateur avant d'être envoyées au serveur. Cela renforcerait considérablement la sécurité en évitant que le serveur ait accès aux données en clair.
2. **Authentification à Deux Facteurs (2FA)** : L'ajout de l'authentification à deux facteurs offrirait une couche de sécurité supplémentaire. Les utilisateurs pourraient choisir d'utiliser des méthodes telles que des codes SMS, des applications d'authentification ou des clés matérielles pour renforcer leur accès.
3. **Audit des Mots de Passe** : Intégrer un outil d'audit des mots de passe permettra aux utilisateurs de vérifier la robustesse de leurs mots de passe existants et de recevoir des recommandations pour les renforcer.

Idées pour le Développement Futur :

1. **Intégration de Biométrie** : Permettre aux utilisateurs d'utiliser la biométrie, telle que la reconnaissance d'empreintes digitales ou faciales, pour déverrouiller leur gestionnaire de mots de passe, renforçant ainsi la convivialité et la sécurité.
2. **Prise en charge de Navigateurs et d'Applications Mobiles** : Élargir la portée de l'application en développant des extensions de navigateur et des applications mobiles pour un accès encore plus facile aux mots de passe stockés.
3. **Analyse des Fuites de Données** : Intégrer une fonctionnalité qui surveille les fuites de données sur le web et avertit les utilisateurs si leurs informations d'identification sont compromises.
4. **Gestion d'Identité** : Évoluer vers une solution de gestion d'identité plus complète qui permettrait aux utilisateurs de gérer non seulement leurs mots de passe, mais aussi d'autres informations d'identification telles que les cartes de crédit et les informations de compte.

VII. Conclusion

En conclusion, le développement de notre gestionnaire de mots de passe communicant représente une étape significative dans la création d'une solution logicielle sécurisée pour la gestion des informations d'identification. Ce projet a été guidé par un engagement envers la sécurité, la simplicité et l'accessibilité.

Nous avons choisi des technologies bien adaptées à nos objectifs, notamment Python, Flask et PostgreSQL, pour garantir une expérience utilisateur fluide et sécurisée. Le choix de développer l'application en anglais visait à toucher un public plus global et à simplifier l'interaction avec l'application.

Les fonctionnalités de l'application, y compris la gestion des mots de passe, le générateur de mots de passe et la recherche, offrent une expérience intuitive . Cependant, des perspectives d'amélioration notables ont été identifiées, telles que le chiffrement client-side, l'authentification à deux facteurs et l'audit des mots de passe.

Ce projet nous a confronté à des défis techniques, en particulier en ce qui concerne la sécurité des données, et nous a incités à mettre en place des pratiques de sécurité rigoureuses. Les leçons tirées de ces défis renforcent notre capacité à créer des solutions logicielles sécurisées à l'avenir.

Finalement, notre gestionnaire de mots de passe communicant représente une étape importante dans notre parcours de développement. Il a été conçu avec une attention particulière à la sécurité des données, offrant ainsi aux utilisateurs un moyen fiable de gérer leurs informations d'identification en toute sécurité. Toutefois, il reste un potentiel d'amélioration pour offrir une sécurité renforcée et une expérience utilisateur encore plus conviviale.