Rapport de Projet : Application Flask Auth App

Introduction

Dans le cadre de notre formation en licence, nous avons développé une application web utilisant le framework Flask. Ce projet, intitulé **Flask Auth App**, est une application d'authentification qui permet aux utilisateurs de s'inscrire, de se connecter, de gérer leur profil et de se déconnecter. Ce rapport mettra en lumière les aspects techniques du projet, en mettant un accent particulier sur les concepts de **session** et de **cache**, ainsi que leur rôle dans l'application.

Objectifs du Projet

- Implémenter un système d'authentification sécurisé.
- Gérer les sessions utilisateur pour maintenir l'état de connexion.
- Optimiser les performances grâce à l'utilisation du cache.
- Appliquer les bonnes pratiques de développement web avec Flask.

Structure du Projet

Le projet est organisé en plusieurs modules pour une meilleure maintenabilité :

1. app.py : Point d'entrée principal de l'application.
2. models : Contient les modèles de base de données (ex. : User).
3. controllers : Contient la logique métier (ex. : AuthController).
4. views : Définit les routes et les vues (ex. : auth_views.py).
5. templates : Contient les fichiers HTML pour le rendu des pages.
6. static : Contient les fichiers CSS et JavaScript.
7. utils : Fournit des utilitaires comme le cache et les décorateurs.

Fonctionnalités Clés

1. Authentification

- **Inscription** : Les utilisateurs peuvent créer un compte avec un email, un nom d'utilisateur et un mot de passe.
- **Connexion** : Les utilisateurs peuvent se connecter avec leur email et mot de passe.
- **Déconnexion** : Les utilisateurs peuvent se déconnecter, ce qui efface leur session.

2. Gestion de Profil

- Les utilisateurs peuvent mettre à jour leurs informations personnelles (email, nom d'utilisateur, mot de passe).

3. Sécurité

- Les mots de passe sont hashés avec Werkzeug pour garantir leur sécurité.
- Les sessions sont protégées par une clé secrète définie dans config.py.

Concepts Techniques : Session et Cache

1. Session

La session est utilisée pour maintenir l'état de connexion d'un utilisateur entre les requêtes HTTP. Flask gère les sessions en stockant des données côté serveur ou côté client (par exemple, dans des cookies).

Implémentation dans le Projet

- **Stockage des Sessions** : Nous utilisons le type filesystem pour stocker les sessions dans des fichiers locaux. Cela est configuré dans config.py :

```
SESSION_TYPE = 'filesystem'
```

- **Utilisation** :
    - Lors de la connexion, l'ID utilisateur et le nom d'utilisateur sont stockés dans la session :

```
session['user_id'] = user.id

session['username'] = user.username
```

    - Lors de la déconnexion, la session est effacée :

```
session.clear()
```

Avantages

- Permet de maintenir l'état de connexion d'un utilisateur.

- Simplifie la gestion des utilisateurs connectés.

Limites

- Le stockage filesystem peut ne pas être adapté pour des applications à grande échelle.

---

2. Cache

Le cache est utilisé pour stocker temporairement des données fréquemment utilisées afin de réduire les requêtes répétées à la base de données et d'améliorer les performances.

Implémentation dans le Projet

- **Initialisation** : Le cache est configuré avec le type simple dans cache.py :

```
cache = Cache()

cache.init_app(app)
```

- **Utilisation** :
    - Lors de la connexion, les données utilisateur sont mises en cache :

```
cache.set(f'user_{user.id}', user.to_dict(), timeout=300)
```

    - Lors de la récupération des données utilisateur, le cache est consulté en premier

```
cached_user = cache.get(f'user_{user_id}')

if cached_user:

    return cached_user
```

Avantages

- Réduit la charge sur la base de données.
- Améliore les temps de réponse pour les requêtes fréquentes.

Limites

- Les données en cache peuvent devenir obsolètes si elles ne sont pas mises à jour correctement.

Résultats et Analyse

- **Sessions** : Les sessions permettent une gestion fluide des utilisateurs connectés. Cependant, pour une application en production, il serait préférable d'utiliser un stockage de session plus robuste comme Redis.

- **Cache** : L'utilisation du cache a considérablement amélioré les performances, notamment pour les données utilisateur fréquemment consultées.

Améliorations Futures

- **Stockage des Sessions** : Migrer vers un système de stockage distribué comme Redis pour une meilleure scalabilité.
- **Cache Avancé** : Implémenter un cache distribué pour gérer les données à grande échelle.
- **Sécurité** : Ajouter des fonctionnalités comme la validation des tokens CSRF et l'authentification à deux facteurs.

Conclusion

Ce projet a permis de mettre en pratique les concepts de sessions et de cache dans une application Flask. Ces outils sont essentiels pour garantir une expérience utilisateur fluide et des performances optimales. En tant qu'étudiant en licence, ce projet m'a permis de renforcer mes compétences en développement web et en gestion des données.

Annexes

- **Code Source** : https://github.com/farhaouiayoub/Mini_Projet_Flask_Ayoub-Farhaoui-IDAI.git
- **Documentation Flask** : https://flask.palletsprojects.com/