Documentation du Projet de Gestion de l'Authentification à Deux Facteurs (2FA)

Ce projet met en œuvre un système d'authentification à deux facteurs (2FA) utilisant des codes basés sur le temps (TOTP) pour renforcer la sécurité des comptes d'utilisateur. Il utilise FastAPI comme framework web et SQLAlchemy pour la gestion de la base de données.

I. Dépendances:

Le projet utilise plusieurs bibliothèques :

base64 : Pour encoder/décoder les clés secrètes.

io : Pour gérer les flux de bytes (génération de code QR).

secrets : Pour générer des clés secrètes aléatoires.

typing : Pour les annotations de type (optionnel).

qrcode : Pour générer des images de code QR.

fastapi : Framework web pour construire l'API.

pyotp : Bibliothèque pour la génération et la vérification de mots de passe à usage unique basés sur le temps (TOTP).

sqlalchemy : ORM pour interagir avec une base de données SQLite.

II. Modèle de base de données:

Une table de base de données nommée utilisateurs est définie à l'aide de SQLAlchemy.

La table stocke deux colonnes :

user_id : Identifiant unique de l'utilisateur (clé primaire).

secret_key : Clé secrète encodée en base64 utilisée pour générer des codes TOTP.

III. Classe TwoFactorAuth

Cette classe encapsule les fonctionnalités liées à la 2FA pour un utilisateur spécifique.

Elle prend user_id et secret_key comme arguments lors de l'initialisation.

Propriétés :

totp : Instance de pyotp.TOTP pour générer et vérifier les codes.

secret_key : Méthode getter pour accéder à la clé secrète.

qr_code : Propriété qui génère et met en cache l'image de code QR de l'utilisateur au format PNG.

Méthodes :

_generate_secret_key : Génère une clé secrète aléatoire encodée en base64.

get_or_create_secret_key : Récupère ou crée une clé secrète pour un utilisateur à partir de la base de données.

_create_qr_code : Génère une image de code QR basée sur l'URI d'approvisionnement de l'utilisateur.

verify_totp_code : Vérifie un code TOTP fourni par rapport à la clé secrète de l'utilisateur.

IV. Application FastAPI

L'application est définie à l'aide de FastAPI.

Une fonction de dépendance get_db crée et gère une session de base de données en utilisant SQLAlchemy.

Une autre dépendance get_two_factor_auth récupère ou crée une instance TwoFactorAuth pour un utilisateur donné.

V. Points de terminaison de l'API

/add-user : Accepte un objet JSON avec un champ secret_key. Crée une nouvelle entrée d'utilisateur dans la base de données et renvoie le user_id généré.

/get-user/{user_id} : Récupère les informations de l'utilisateur (ID utilisateur et clé secrète) en fonction du user_id fourni. Renvoie une erreur si l'utilisateur est introuvable.

/enable-2fa/{user_id} : Renvoie la clé secrète de l'utilisateur (utile pour activer la 2FA sur des applications externes).

/generate-qr/{user_id} : Génère et renvoie l'image de code QR de l'utilisateur sous forme de flux PNG. Renvoie une erreur si l'utilisateur est introuvable.

/verify-totp/{user_id} : Accepte un code TOTP en paramètre et le vérifie par rapport à la clé secrète de l'utilisateur. Renvoie un objet JSON indiquant la validité.

VI. Gestion des erreurs

L'application utilise la gestion des exceptions de FastAPI pour lever des exceptions HTTP avec des codes d'état appropriés (par exemple, 400 pour les requêtes incorrectes, 404 pour introuvable) en cas d'erreurs.

VII. Fonctionnalité globale

Cette API permet de gérer les clés secrètes des utilisateurs pour la 2FA. Elle fournit des fonctionnalités pour ajouter des utilisateurs, récupérer des informations utilisateur, activer la 2FA avec la clé secrète, générer des codes QR pour l'inscription des utilisateurs et vérifier les codes TOTP pendant le processus d'authentification.

Cette documentation fournit un aperçu des fonctionnalités et de l'utilisation du projet de gestion de l'authentification à deux facteurs (2FA) basé sur FastAPI et SQLAlchemy.