File Flow Backend

Description

Authentification File Flow Backend est une API RESTful construite avec Spring Boot pour gérer l'authentification des utilisateurs, la gestion des profils, la vérification d'email et la récupération de mot de passe. Elle utilise l'authentification JWT avec des tokens de rafraîchissement et intègre l'API REST Gmail pour l'envoi et la vérification des emails.

Prérequis

- Java 17 ou supérieur
- Maven 3 ou supérieur
- Serveur MySQL en local ou distant (docker recommender)
- Compte Google Cloud pour la configuration de l'API Gmail
- Postman ou tout client REST pour les tests

Installation et configuration

1. Configurer la base de données MySQL

Créer une base de données nommée `fileflow`

Modifier `src/main/resources/application.properties` avec vos identifiants MySQL :

```
spring.datasource.url=jdbc:mysql://localhost:3306/fileflow
spring.datasource.username=root
spring.datasource.password=yourpassword
```

2. Configurer le secret JWT et l'email expéditeur

Dans `application.properties`, ajouter :

```
app.jwt.secret=VOTRE_CLE_SECRETE_JWT
from.google.email=votre-email@gmail.com
```

3. Configuration API Gmail (Google Cloud Console)

- Aller sur Google Cloud Console
- Créer ou sélectionner un projet existant
- Activer l'API Gmail via la bibliothèque d'API
- Créer des identifiants OAuth 2.0 (Client ID et Secret) :

    - Type d'application : Web Application
    - Ajouter les URI de redirection autorisés (ex : `http://localhost:8080/oauth2callback` pour local)
- Télécharger le fichier `credentials.json`
- Placer ce fichier dans le dossier `resources/credentials` du projet (ou configurer son chemin dans le service email)
- Au premier lancement de l'application et lors d'un envoi d'email, un lien d'autorisation sera affiché
- Ouvrir ce lien, autoriser l'accès, les tokens seront sauvegardés pour usages futurs

---

Lancer le projet

Compiler et démarrer avec Maven :

```
mvn clean install
mvn spring-boot:run
```

Le serveur backend sera accessible sur `http://localhost:8080`

API disponibles & utilisation

Méthode	Endpoint	Description	Auth requise
POST	/api/auth/register	Inscription d'un nouvel utilisateur	Non
POST	/api/auth/login	Authentification	Non
POST	/api/auth/refresh-token	Rafraîchir le token JWT	Non
POST	/api/auth/logout	Déconnexion (invalidate refresh token)	Non
POST	/api/auth/forget-password	Demande de réinitialisation mot de passe	Non
POST	/api/auth/reset-password	Réinitialiser mot de passe via token	Non
POST	/api/auth/send-code	Envoyer code de vérification email	Non
POST	/api/auth/verify-code	Vérifier code de vérification email	Non
GET	/api/user/me	Obtenir info utilisateur connecté	Oui
PATCH	/api/user/update	Mettre à jour profil utilisateur	Oui

Tester les APIs

Utilisez Postman ou un client REST.

1. Inscription

- URL : `POST /api/auth/register`
- Body JSON :

```
{
  "firstName": "Jean",
  "lastName": "Dupont",
  "email": "jean.dupont@example.com",
  "password": "MotDePasse1!",
  "confirmPassword": "MotDePasse1!"
}
```

- Réponse type :

```
{
  "success": true,
  "message": "Utilisateur enregistré avec succès",
  "data": {
    "token": "jwt-token",
    "type": "Bearer",
    "user": {
      "id": 1,
      "email": "jean.dupont@example.com",
      "firstName": "Jean",
      "lastName": "Dupont"
    }
  },
  "timestamp": "2025-07-23T20:00:00"
}
```

2. Connexion

- URL : `POST /api/auth/login`
- Body JSON :

```
{
  "email": "jean.dupont@example.com",
  "password": "MotDePasse1!"
}
```

- Réponse similaire à l'inscription

3. Rafraîchir token

- URL : `POST /api/auth/refresh-token`
- Body JSON :

```
{
  "refreshToken": "votre-refresh-token"
}
```

- Réponse avec nouveau token

4. Déconnexion

- URL : `POST /api/auth/logout`
- Body JSON :

```
{
  "refreshToken": "votre-refresh-token"
}
```

- Réponse de confirmation

5. Mot de passe oublié

- URL : `POST /api/auth/forget-password`
- Body JSON :

```
{
  "email": "jean.dupont@example.com"
}
```

- Réponse de confirmation d'envoi d'email

6. Réinitialiser mot de passe

- URL : `POST /api/auth/reset-password?token=token-de-reset`
- Body JSON :

```
{
  "password": "NouveauMotDePasse1!",
  "confirmPassword": "NouveauMotDePasse1!"
}
```

- Réponse de confirmation

7. Envoyer code vérification email

- URL : `POST /api/auth/send-code`
- Body JSON :

```
{
  "email": "jean.dupont@example.com"
}
```

- Réponse de confirmation

8. Vérifier code email

- URL : `POST /api/auth/verify-code`
- Body JSON :

```
{
  "email": "jean.dupont@example.com",
  "code": "123456"
}
```

- Réponse succès ou échec

9. Obtenir utilisateur connecté

- URL : `GET /api/user/me`
- Header : `Authorization: Bearer <access-token>`
- Réponse : infos utilisateur JSON

10. Mettre à jour profil

- URL : `PATCH /api/user/update`
- Header : `Authorization: Bearer <access-token>`
- Body JSON (exemple) :

```
{
  "firstName": "Jean-Pierre",
  "lastName": "Dupont"
}
```

- Réponse : infos utilisateur mises à jour

Notes importantes

- Les mots de passe doivent contenir au moins 8 caractères, avec majuscule, minuscule, chiffre et caractère spécial.
- Le secret JWT doit être fort et bien protégé.
- L'envoi d'email utilise l'API Gmail, assurez-vous que les identifiants et tokens sont bien configurés.
- Les refresh tokens expirent après 7 jours.
- Toutes les réponses API suivent cette structure :

```
{
  "success": true,
  "message": "Message descriptif",
  "data": {},
  "timestamp": "2025-07-23T20:00:00"
}
```