API Gestion Matériels Informatiques

API Backend FastAPI pour la gestion et le suivi des équipements informatiques avec import Excel automatisé.

🚀 Fonctionnalités

- **Import Excel automatique** : Importation de fichiers Excel avec gestion automatique des valeurs vides
- **Suivi temporel** : Système de snapshots pour suivre l'évolution des matériels
- **Statistiques avancées** : Tableaux de bord complets avec analyses multi-critères
- **Authentification JWT** : Sécurisation complète de l'API
- **Pagination intelligente** : Optimisation des requêtes avec skip/limit
- **Gestion d'incidents** : Suivi des pannes et problèmes matériels

📋 Prérequis

- Python 3.8+
- MySQL/MariaDB
- pip

🔧 Installation

1. Créer la base de données

```sql
CREATE DATABASE trait8 CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

Puis exécuter le script `database_corrections.sql` pour créer les tables.

2. Installer les dépendances

```bash
pip install -r requirements.txt
```

3. Configuration

Modifier les paramètres de connexion dans `config/database.py` si nécessaire :

```python
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",  # Votre mot de passe MySQL
    database="trait8"
)
```

Modifier la clé secrète JWT dans `utils/security.py` :

```python
SECRET_KEY = "votre_cle_secrete_tres_securisee"
```

4. Lancer l'application

```bash
# Mode développement
uvicorn main:app --reload --host 0.0.0.0 --port 8000

# Ou directement
python main.py
```

L'API sera accessible sur : http://localhost:8000

Documentation interactive : http://localhost:8000/docs

📁 Structure du Projet

```
projet_backend/
│
├── main.py                 # Point d'entrée
├── requirements.txt        # Dépendances
├── database_corrections.sql  # Script SQL
│
├── config/
│   └── database.py         # Configuration DB
│
├── models/
│   └── schemas.py          # Schémas Pydantic
│
```

```
├── services/
│   ├── excel_service.py       # Import Excel
│   ├── statistics_service.py  # Statistiques
│   └── auth_service.py        # Authentification
│
├── routes/
│   ├── auth.py                # Routes auth
│   ├── upload.py              # Routes upload
│   ├── statistics.py          # Routes stats
│   └── materiels.py           # Routes matériels
│
└── utils/
    └── security.py            # Sécurité JWT
```

🔐 Authentification

1. Créer un compte

bash

```
POST /auth/register
{
  "mail": "user@example.com",
  "mot_de_passe": "motdepasse123"
}
```

2. Se connecter

bash

```
POST /auth/login
{
  "mail": "user@example.com",
  "mot_de_passe": "motdepasse123"
}
```

Réponse :

json

```
{
  "access_token": "eyJhbGc...",
  "token_type": "bearer",
  "user": {
    "id": 1,
    "mail": "user@example.com"
  }
}
```

3. Utiliser le token

Ajouter le header à toutes les requêtes protégées :

```
Authorization: Bearer eyJhbGc...
```

📤 Upload de Fichiers Excel

Format du fichier Excel attendu

code	region	district	commune	nom_materiel	etat_materiel	type_materiel	motif	achat_
630601	ATSIMO ANDREFANA	MOROMBE	A	Imprimante 1	Fonctionnel	Imprimante	ER P03	ENY
				Imprimante 2	Fonctionnel	Imprimante		ENY

Notes importantes :

- Les cellules vides dans les colonnes `code`, `region`, `district`, `commune` héritent automatiquement de la dernière valeur non vide
- Les autres cellules vides sont converties en `NULL`

Uploader un fichier

```bash
```

```
POST /upload/excel
Content-Type: multipart/form-data
Authorization: Bearer <token>

file: fichier.xlsx
```

📊 Endpoints Principaux

Statistiques

```bash
# Statistiques complètes pour une date
GET /statistics/?id_date_import=1&skip_type=0&limit_type=10&skip_region=0&limit_region=10

# Dashboard (dernière importation)
GET /statistics/dashboard
```

Matériels

```bash
# Liste tous les matériels
GET /materiels/all?id_date_import=1&skip=0&limit=10

# Matériels par commune
GET /materiels/by-commune?id_date_import=1&commune=Ambahita&skip=0&limit=10

# Nouveaux matériels entre 2 dates
GET /materiels/nouveaux?date_ancienne=1&date_nouvelle=2&skip=0&limit=10

# Détails d'un matériel
GET /materiels/{id_snapshot}

# Recherche par code
GET /materiels/search/by-code?code=630601&id_date_import=1
```

Upload

```bash
```

```
# Historique des uploads
GET /upload/history?skip=0&limit=10

# Liste des dates d'importation
GET /upload/dates
```

Authentification

bash

```
# Changer le mot de passe
POST /auth/change-password
{
  "ancien_mot_de_passe": "ancien",
  "nouveau_mot_de_passe": "nouveau"
}

# Changer l'email
POST /auth/change-mail
{
  "nouveau_mail": "nouveau@example.com",
  "mot_de_passe": "motdepasse"
}

# Infos utilisateur
GET /auth/me
```

🎯 Exemple Complet d'Utilisation

python

```python
import requests

BASE_URL = "http://localhost:8000"

# 1. Créer un compte
response = requests.post(f"{BASE_URL}/auth/register", json={
    "mail": "admin@example.com",
    "mot_de_passe": "admin123"
})
print(response.json())

# 2. Se connecter
response = requests.post(f"{BASE_URL}/auth/login", json={
    "mail": "admin@example.com",
    "mot_de_passe": "admin123"
})
token = response.json()["access_token"]
headers = {"Authorization": f"Bearer {token}"}

# 3. Uploader un fichier Excel
with open("materiels.xlsx", "rb") as f:
    files = {"file": f}
    response = requests.post(
        f"{BASE_URL}/upload/excel",
        files=files,
        headers=headers
    )
    print(response.json())

# 4. Récupérer les statistiques
response = requests.get(
    f"{BASE_URL}/statistics/dashboard",
    headers=headers
)
stats = response.json()
print(f"Total matériels: {stats['statistics']['resume_global']['total_materiels']}")

# 5. Lister les matériels
response = requests.get(
    f"{BASE_URL}/materiels/all?id_date_import=1&skip=0&limit=10",
    headers=headers
)
```

```
materiels = response.json()
print(f"Nombre de matériels: {materiels['total']}")
```

🔍 Détails Techniques

Gestion des Snapshots

Le système utilise une approche de **snapshots temporels** :

- Chaque import crée une nouvelle date dans `date_import`
- Les matériels physiques (`materiel_physique`) sont des références persistantes
- Les états (`materiel_informatique`) sont des snapshots liés à une date d'import

Cela permet de :

- Suivre l'évolution d'un matériel dans le temps
- Comparer les états entre différentes dates
- Calculer les nouveaux matériels et matériels perdus

Optimisation des Requêtes

- **Pagination** : Tous les endpoints de liste supportent `skip` et `limit`
- **Index** : Les colonnes fréquemment utilisées sont indexées
- **Context Manager** : Gestion automatique des connexions DB

Sécurité

- **JWT** : Tokens avec expiration (24h par défaut)
- **Bcrypt** : Hachage sécurisé des mots de passe
- **Validation** : Pydantic pour valider toutes les entrées
- **CORS** : Configurable pour la production

🐛 Débogage

Problème de connexion à la base de données

Vérifier dans `config/database.py` :

- Host, user, password corrects

- Base de données existe
- Utilisateur a les permissions nécessaires

Erreur lors de l'import Excel

- Vérifier le format du fichier (xlsx ou xls)
- Vérifier les noms des colonnes (doivent correspondre)
- Créer le dossier `uploads/` à la racine du projet

Token expiré

Le token expire après 24h. Il faut se reconnecter.

📝 Notes de Production

Pour déployer en production :

1. Changer `SECRET_KEY` dans `utils/security.py`
2. Configurer CORS dans `main.py` avec les origines autorisées
3. Utiliser un serveur de production (gunicorn + nginx)
4. Activer HTTPS
5. Configurer les variables d'environnement pour les secrets
6. Mettre en place des sauvegardes régulières de la BD

📄 Licence

Ce projet est développé pour un usage interne.

👨‍💻 Support

Pour toute question ou problème, consultez la documentation interactive sur `/docs`.