📘 Authentification dans Django REST Framework

🔐 Types d'authentification disponibles

1. Basic Authentication

- Envoie email/username + password à chaque requête.

Inconvénients :

- Stockage du mot de passe côté client (risqué).
- Envoi en clair dans chaque requête.

---

2. Token Authentication ✅ (celui utilisé dans le cours)

- Envoie un token d'authentification avec chaque requête après login.
- Le token est généré une fois, et utilisé jusqu'à expiration/suppression.

```
# Générer un token (endpoint typique)
POST /api/user/token/
{
  "email": "user@example.com",
  "password": "securepassword"
}

# Exemple de header à inclure ensuite
Authorization: Token abcd1234tokenvalue
```

Avantages :

- Simplicité (inclus par défaut dans DRF).
- Séparation login/authentification.
- Meilleure sécurité que Basic Auth (pas d'envoi du mot de passe).

Inconvénients :

- Le token doit être sécurisé côté client.
- Si compromis, il permet un accès total.
- Vérification via base de données à chaque requête.

---

3. JSON Web Token (JWT)

- Utilise des access + refresh tokens.
- Plus complexe, mais utile pour les apps à grande échelle.
- Réduit la charge sur la base de données.

---

4. Session Authentication

- Utilise des cookies pour stocker la session utilisateur.
- Courant pour les applications web.
- Moins pratique pour une API REST sans interface web.

📲 Fonctionnement du Token Auth

- L'utilisateur envoie ses identifiants à un endpoint d'authentification.
- Le serveur renvoie un token.
- Le client stocke ce token (localStorage, sessionStorage, cookie…).
- Toutes les requêtes authentifiées incluent le token dans le header HTTP.

🚪 Déconnexion (Logout)

- Pas de vraie API de logout nécessaire :
- Il suffit de supprimer le token localement.

Pourquoi ne pas créer une API logout ?

- Le client peut ne jamais l'appeler (ex. : app supprimée, déconnexion brutale).
- Données peu fiables côté serveur.

```
# Si besoin, créer un endpoint pour logout :
# Supprimer le token du serveur
DELETE /api/user/token/
```

✅ Pourquoi choisir le Token Auth ici ?

- Bon compromis entre sécurité et simplicité.
- Fonctionne out-of-the-box avec Django REST Framework.
- Pas besoin de bibliothèques externes.
- Facile à implémenter dans n'importe quel client (mobile, web, etc.).