Contexte précis

- **Projet A (SaaS)** :
    - Gère l'authentification centrale via Passport (OAuth2).
    - Délivre des tokens aux applications clientes.
- **Projet B (application propre)** :
    - Possède sa propre base utilisateurs / données.
    - Veut s'authentifier auprès du SaaS Projet A pour vérifier la validité des tokens, permissions, etc.

⚙️ Comment procéder ?

1. Projet B obtient un token via `client_credentials`

- Projet B s'identifie auprès de Projet A avec son `client_id` / `client_secret` OAuth.
- Reçoit un `access_token` valide.

2. Projet B gère ses propres utilisateurs localement

- L'utilisateur se connecte sur Projet B (avec sa propre base et système).
- Projet B peut, selon besoin, vérifier la validité d'un token émis par Projet A (ex: token de session ou accès API).
- Ou Projet B utilise Projet A uniquement comme fournisseur d'identité externe (ex: SSO).

3. Communication sécurisée

- Les échanges entre Projet B et Projet A utilisent le token `client_credentials` (machine-to-machine).

- Projet B peut aussi transmettre des tokens utilisateurs à Projet A pour vérification via endpoints d'authentification (ex: `/oauth/token/info` si tu exposes ce genre d'API).

🧠 Résumé

Projet	Rôle
Projet A (SaaS)	Fournisseur OAuth centralisé (authentification)
Projet B	Application autonome avec sa propre DB + client OAuth

- **Token machine-to-machine** protège les échanges entre Projet B et Projet A.
- **Gestion utilisateurs séparée** dans Projet B.
- Projet B peut utiliser Projet A comme fournisseur OAuth ou pour valider des tokens.

Super ! Voici comment tu peux gérer la validation des tokens OAuth émis par Projet A dans Projet B, ainsi que des pistes pour synchroniser les utilisateurs si besoin.

---

1. Validation du token OAuth côté Projet B

Projet B reçoit un token OAuth (par exemple un access_token JWT ou opaque) généré par Projet A et doit vérifier qu'il est valide avant d'autoriser l'accès.

Options de validation :

a) Validation via introspection OAuth (recommandé)

- Projet A doit exposer un **endpoint d'introspection** (ex : `/oauth/introspect`) qui prend un token et retourne son statut (valide, périmé, scopes, utilisateur lié, etc).

- Depuis Projet B, tu envoies le token reçu à cet endpoint et tu vérifies la réponse :

php
CopierModifier

```php
$response =
Http::asForm()->post('https://api-projet-a.com/oauth/introspect', [
    'token' => $accessToken,
    'client_id' => 'PROJET_B_CLIENT_ID',
    'client_secret' => 'PROJET_B_CLIENT_SECRET',
]);

$data = $response->json();

if ($data['active']) {
    // Token valide, continuer
} else {
    // Token invalide ou expiré, rejeter
}
```

Note: Laravel Passport ne fournit pas nativement un endpoint d'introspection. Il faudra éventuellement l'implémenter toi-même dans Projet A, en vérifiant le token via la base ou JWT.

---

b) Validation locale du token JWT (si token JWT)

- Si Projet A émet des tokens JWT signés, Projet B peut vérifier localement la signature, la validité de la date, et extraire les infos.
- Pour cela, tu dois partager la clé publique utilisée par Projet A.
- Cela évite un appel réseau mais demande de gérer la rotation des clés.

---

2. Synchronisation / gestion des utilisateurs entre Projet A et Projet B

Si Projet B a sa propre base utilisateurs, mais que tu veux centraliser la gestion d'identité dans Projet A, voici quelques méthodes :

a) Synchronisation à la connexion

- Lorsqu'un utilisateur s'authentifie sur Projet B, tu peux vérifier son identité via Projet A (par exemple via un token OAuth).
- Si l'utilisateur n'existe pas localement, tu le crées (ou mets à jour) à partir des données reçues de Projet A.

b) Synchronisation périodique

- Projet B récupère régulièrement (via API sécurisée) la liste des utilisateurs ou modifications de Projet A.
- Permet d'avoir les données à jour sans devoir re-authentifier chaque fois.

c) Utilisation d'un système SSO (Single Sign-On)

- Projet B redirige l'utilisateur vers Projet A pour s'authentifier.
- Une fois authentifié, Projet A renvoie un token que Projet B peut utiliser.
- Projet B crée ou met à jour localement l'utilisateur.

---

3. Exemple simple de synchronisation à la connexion

php
CopierModifier

```
// Sur Projet B, après réception du token Projet A (ex: access_token)
$userInfo = Http::withToken($accessToken)->get('https://api-projet-a.com/api/user');

if ($userInfo->successful()) {
    $data = $userInfo->json();

    // Chercher l'utilisateur local
    $user = User::where('email', $data['email'])->first();

    if (!$user) {
        // Créer nouvel utilisateur local
```

```
        $user = User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            // autres champs...
        ]);
    } else {
        // Mettre à jour si besoin
        $user->update([
            'name' => $data['name'],
            // ...
        ]);
    }

    Auth::login($user);
}
```

En résumé

Étape	Description
Validation token OAuth	Via introspection ou validation JWT
Synchronisation utilisateurs	À la connexion ou périodiquement
SSO possible	Redirection vers Projet A pour login
