En Laravel, il existe **plusieurs solutions d'authentification** selon le type d'application (monolithique, SPA, API, etc.) et le niveau de personnalisation souhaité. Voici un panorama complet des **principales approches d'authentification en Laravel**, avec leurs avantages, cas d'usage et exemples.

---

1. Authentification intégrée : `laravel/ui` (Blade + Auth classique)

Description

C'est la solution d'authentification "classique" basée sur Blade et les sessions. Utilisée dans les projets monolithiques où le front et le back sont sur le même serveur.

Installation

```
composer require laravel/ui
php artisan ui bootstrap --auth
npm install && npm run dev
```

Avantages

- Simple à mettre en place
- Basée sur les sessions et cookies
- S'intègre parfaitement avec Blade
- Bon point de départ pour une application monolithique

Inconvénients

- Peu adaptée aux API ou front-end séparés (React, Vue, etc.)

---

2. Laravel Breeze (auth minimaliste moderne)

Description

Breeze est la version moderne et légère de `laravel/ui`. Il propose une base d'authentification simple, propre et compatible avec TailwindCSS.

Installation

```
composer require laravel/breeze --dev
php artisan breeze:install
npm install && npm run dev
php artisan migrate
```

Avantages

- Code clair et minimaliste
- Utilise les dernières pratiques Laravel
- Version SPA disponible avec Inertia.js ou Vue/React

Inconvénients

- Pas de fonctionnalités avancées (2FA, gestion de sessions, etc.)

---

3. Laravel Jetstream (auth avancée)

Description

Jetstream est la version plus complète de Breeze. Il inclut des fonctionnalités avancées prêtes à l'emploi :

- Authentification
- Vérification d'email
- Gestion de profil
- Sessions multiples
- 2FA (Two-Factor Authentication)
- API tokens (via Sanctum)

Installation

```
composer require laravel/jetstream
php artisan jetstream:install livewire
# ou
php artisan jetstream:install inertia
npm install && npm run dev
php artisan migrate
```

Avantages

- Très complet
- Compatible Livewire ou Inertia.js
- Gestion complète de compte utilisateur

Inconvénients

- Plus complexe à personnaliser
- Surdimensionné pour de petits projets

---

4. Laravel Sanctum (auth pour SPA / API)

Description

Sanctum fournit une authentification légère basée sur des **tokens** pour :

- Les SPA (Single Page Applications)
- Les API
- Les apps mobiles

Il gère les **cookies sécurisés (CSRF)** pour les SPA et **API tokens personnels** pour les autres clients.

Installation

```
composer require laravel/sanctum
php artisan migrate
```

Dans `app/Http/Kernel.php` :

```
'api' => [
    \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
    'throttle:api',
    \Illuminate\Routing\Middleware\SubstituteBindings::class,
],
```

Avantages

- Idéal pour API + SPA (Vue, React, Angular, etc.)
- Simple et léger
- S'intègre parfaitement avec Jetstream

Inconvénients

- Pas de OAuth2 (il gère des tokens simples seulement)

---

5. Laravel Passport (auth OAuth2 complète)

Description

Passport est une implémentation complète du protocole **OAuth2**, adaptée aux API REST sécurisées. Il gère :

- Les clients OAuth (client credentials, password grant, etc.)
- Les tokens d'accès et de rafraîchissement
- Les scopes et autorisations

Installation

```
composer require laravel/passport
php artisan migrate
php artisan passport:install
```

Avantages

- Sécurisé et robuste
- Conforme OAuth2 standard
- Idéal pour API publiques ou multi-clients

Inconvénients

- Plus lourd et plus complexe à configurer
- Pas nécessaire pour une simple SPA

---

6. Socialite (auth via réseaux sociaux)

Description

Socialite permet de gérer l'authentification via des services externes :

- Google
- Facebook
- GitHub
- Twitter, etc.

Installation

```
composer require laravel/socialite
```

Exemple d'utilisation :

```php
Route::get('/auth/redirect', [SocialAuthController::class, 'redirect']);
Route::get('/auth/callback', [SocialAuthController::class, 'callback']);
```

```php
use Laravel\Socialite\Facades\Socialite;

public function redirect()
{
    return Socialite::driver('github')->redirect();
}

public function callback()
{
    $user = Socialite::driver('github')->user();
}
```

Avantages

- Auth rapide via fournisseurs externes
- Intégration simple avec Jetstream ou Breeze

Inconvénients

- Dépendance aux API externes
- Gestion additionnelle requise pour lier comptes locaux et externes

---

7. Auth personnalisée (Custom Guards & Providers)

Description

Pour les cas très spécifiques (API non standard, SSO, LDAP, etc.), Laravel permet de créer ses propres :

- **Guards** → définissent comment l'utilisateur est authentifié
- **User Providers** → définissent où et comment les utilisateurs sont stockés

Exemple :

Dans `config/auth.php` :

```
'guards' => [
    'api-custom' => [
        'driver' => 'token',
        'provider' => 'users',
    ],
],
```

Choix rapide selon le besoin

Type d'application	Solution recommandée
Application monolithique (Blade)	Laravel Breeze ou Laravel UI
Application moderne avec Inertia/Livewire	Laravel Jetstream
SPA (Vue, React, Angular)	Laravel Sanctum
API publique / mobile	Laravel Passport
Auth via réseaux sociaux	Laravel Socialite
Auth très spécifique (LDAP, SSO, etc.)	Custom Guard

Souhaitez-vous que je te fasse un **tableau comparatif détaillé** (avec exemples de flux d'authentification et structures de tables) pour chaque solution ? Cela te donnerait une vue encore plus concrète pour choisir la bonne.