**🚀 Laravel Socialite Tutorial – Word een Auth Jedi in 4 Dagen**

**Amine Ben Slimene, Didier Vanassche, Robin Caulier**

Laatste update: 05-05-2025



## **🎓 *Een epische reis van composer install tot intergalactische loginmagie***

“De Force is sterk in deze app… vooral met Socialite geïnstalleerd.”

Welkom, jonge Padawan! 👾  
 Ben jij klaar om je saaie loginformulier om te toveren tot een hypersnelle, Google-gevoede, GitHub-geconnecteerde loginervaring?

In deze handleiding, geboren uit de vurige vlammen van Hell Week 🔥, leer je stap voor stap hoe je gebruikers laat inloggen met Google én GitHub, hoe je Laravel overtuigt om netjes om te gaan met wachtwoorden, én hoe jij het licht zwaarder maakt dan de duisternis van foutmeldingen.

✨ Code je mee naar het sterrenstelsel van perfecte authenticatie? Dan is dit jouw weg.

## **🗓️ Dag 1 – De Start van het Avontuur: Project Setup**

## **🎯 Wat is het doel?**

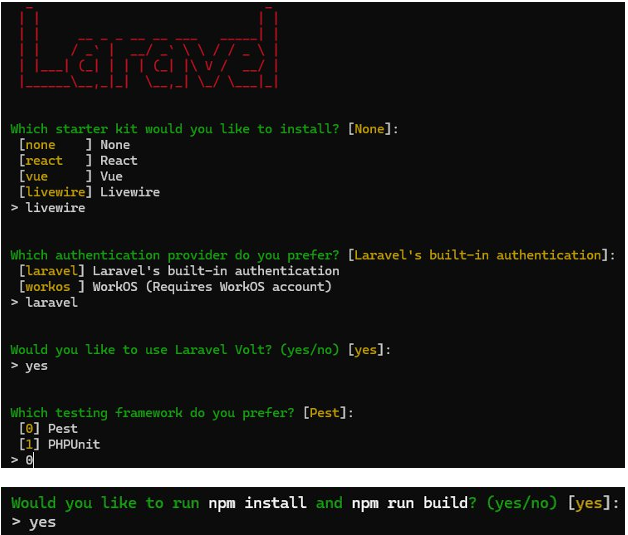
Elke Jedi heeft zijn eigen lichtzwaard nodig... en jij hebt jouw eigen Laravel-project nodig. In deze eerste stap leer je hoe je je eigen Laravel-sterrenstelsel creëert met Livewire, Laravel Volt en ingebouwde authenticatie. Geen kloons of oude projecten deze keer — jij bouwt het helemaal zelf, vers uit de oven van de Artisan Forge. Klaar? Bouw de basis van je Force!

## **✅ Actie 1: Een Nieuw Laravel-Project Aanmaken**

Gebruik deze krachtlijn in je terminal:

livewire new laravelApp

Je krijgt dan een **interactieve wizard** zoals op de afbeelding . Kies **exact** deze instellingen:



1. Which starter kit would you like to install?  
    👉 **Livewire**
2. Which authentication provider do you prefer?  
    👉 **Laravel's built-in authentication**
3. Would you like to use Laravel Volt?  
    👉 yes
4. Which testing framework do you prefer?  
    👉 0 (**Pest**)
5. Would you like to run npm install and npm run build?  
    👉 yes

⚠️ Deze keuzes zorgen ervoor dat je project **volledig klaar staat** met Livewire, Tailwind CSS, auth, Laravel Volt en tooling.

## **✅ Actie 2: Open je project in PhpStorm**

Open je project in PhpStorm met deze stappen:

1. Start PhpStorm
2. Kies **"Open"**
3. Navigeer naar je map laravelApp
4. Klik op **Open**

💡 *Tip: stel in PhpStorm Laravel & Blade support in voor betere auto-completion en highlighting.*

### **⚡ Open een nieuw terminalvenster en voer uit:**

composer run dev

## **✅ Actie 3: Installeer Laravel Socialite**

composer require laravel/socialite

**🧱 Actie 4: Pas je user-migratie aan**

Schema::create('users', function (Blueprint $table) {  
 $table->id();  
 $table->string('name');  
 $table->string('email')->unique();  
 $table->timestamp('email\_verified\_at')->nullable();  
 $table->string('password');  
 $table->rememberToken();  
 $table->timestamps();  
 $table->softDeletes();  
 $table->string('google\_id')->nullable()->unique();  
 $table->string('github\_id')->nullable()->unique();  
});

### **👤 Actie 5: Werk het User-model bij**

➕ Voeg toe aan $fillable:

protected $fillable = [  
 'name', 'email', 'password', 'google\_id', 'github\_id',  
];

### **🧠 Voeg hulpfuncties toe:**

public function registeredWithGoogle(): bool {  
 return !is\_null($this->google\_id);  
}  
  
public function registeredWithGithub(): bool {  
 return !is\_null($this->github\_id);  
}

**🧼 Laat de Force opnieuw stromen:**

php artisan migrate:fresh --seed

🗓**️ Dag 2 – Google Login: Contact met het Licht**

### **🎯 Wat is het doel?**

Gebruikers kunnen zich aanmelden of inloggen met hun Google-account via Laravel Socialite.

### **☁️ 1. Maak een project aan op Google Cloud Console**

➡️ Ga naar: <https://console.cloud.google.com/>

* Klik bovenaan op **“Project selecteren”** → **“Nieuw project”**
* Geef het een naam zoals LaravelAuthApp
* Klik op **“Aanmaken”**
* Zorg ervoor dat je in je nieuwe project werkt. Zo niet, selecteer het dan.

### **🔓 2. Activeer de Google People API**

* In de zoekbalk: zoek naar **People API**
* Klik op **Google People API**
* Druk op **Activeren**

### **🔑 3. Maak OAuth 2.0-client aan**

* Ga naar: **API's en services > Inloggegevens**
* Klik op **+ Inloggegevens maken > OAuth-client-ID**
* Stel eerst een toestemmingsscherm in als dat gevraagd wordt:
  + Geef je app een naam (bv. LaravelApp)
  + Kies **Extern**
  + Vul de verplichte velden in (je kunt de rest overslaan)
* Klik op **Doorgaan** tot je op het dashboard terugkomt
* Klik opnieuw op **+ Inloggegevens maken > OAuth-client-ID**
* Kies: **Webapplicatie**
* Geef een naam (bv. Laravel Localhost)
* Voeg volgende redirect-URI toe: <http://127.0.0.1:8000/auth/callback/google>Klik op **Aanmaken**  
   ➡️ Je krijgt een **Client ID** en **Client Secret**

### **⚙️ 4. Voeg de gegevens toe aan je .env**

GOOGLE\_CLIENT\_ID=je-client-id  
GOOGLE\_CLIENT\_SECRET=je-secret  
GOOGLE\_REDIRECT\_URI=http://127.0.0.1:8000/auth/callback/google

### **🛠️ 5. Voeg configuratie toe in config/services.php**

'google' => [  
 'client\_id' => env('GOOGLE\_CLIENT\_ID'),  
 'client\_secret' => env('GOOGLE\_CLIENT\_SECRET'),  
 'redirect' => env('GOOGLE\_REDIRECT\_URI'),  
],

### **🧹 6. Cache legen**

php artisan config:clear  
php artisan cache:clear

### **🎮 7. Controller aanmaken**

Met dit commando maak je een nieuwe controller aan genaamd GoogleController in de map Auth. Deze controller zal de logica bevatten voor het inloggen via Google.

php artisan make:controller Auth/GoogleController

In GoogleController.php:

public function redirect()  
{  
 return Socialite::driver('google')->redirect();  
}  
  
public function callback()  
{  
 $googleUser = Socialite::driver('google')->stateless()->setHttpClient(  
 new \GuzzleHttp\Client(['verify' => false])  
)->user();

$user = \App\Models\User::where('email', $googleUser->getEmail())->first();  
  
 if (! $user) {  
 $user = \App\Models\User::create([  
 'name' => $googleUser->getName(),  
 'email' => $googleUser->getEmail(),  
 'password' => bcrypt(Str::random(16)),  
 'google\_id' => $googleUser->getId(),  
 ]);  
 }  
  
 Auth::login($user);  
 return redirect()->route('dashboard');  
}

#### **💡 Opmerking: waarom stateless() gebruiken?**

De methode stateless() schakelt het gebruik van sessies uit tijdens de OAuth-authenticatie. Dit is bijzonder nuttig in de volgende situaties:

* Tijdens **lokale ontwikkeling**, wanneer sessies niet betrouwbaar worden opgeslagen of doorgestuurd
* Bij gebruik van **externe tools zoals Postman**, of bij testomgevingen zonder sessiebeheer
* Wanneer je foutmeldingen krijgt zoals:

InvalidStateException of State parameter missing

Daarnaast komt het soms voor dat je tijdens lokale tests een foutmelding krijgt zoals:

cURL error 60: SSL certificate problem: unable to get local issuer certificate

Dit heeft te maken met SSL-certificaten op je systeem. Hoewel stateless() deze fout niet direct oplost, helpt het **de authenticatie alsnog te laten werken zonder afhankelijk te zijn van sessietokens**. Het is dus een handige tijdelijke oplossing om vlot verder te kunnen werken!

#### **💡 Opmerking: waarom setHttpClient(new \GuzzleHttp\Client(['verify' => false])) gebruiken?**

Deze code overschrijft de standaard HTTP-client die door Socialite (onderliggend via Guzzle) wordt gebruikt en schakelt SSL-certificaatverificatie uit door 'verify' => false in te stellen. Dit wordt vaak tijdelijk toegepast om SSL-gerelateerde fouten tijdens het ophalen van gebruikersgegevens via Google op te vangen.

### **Wanneer is dit nuttig?**

* **Tijdens lokale ontwikkeling**, vooral op Windows-machines of ontwikkelomgevingen zonder correct geïnstalleerde root-certificaten.
* Wanneer je fouten tegenkomt zoals:

cURL error 60: SSL certificate problem: unable to get local issuer certificate

Dit wijst erop dat de PHP-installatie de vereiste certificaatketen niet kan verifiëren.

### **Wat doet 'verify' => false precies?**

Het voorkomt dat Guzzle (de HTTP-client) probeert om SSL-certificaten te valideren. Daardoor kan de aanvraag alsnog doorgaan, ook al is het certificaat onbetrouwbaar of ontbreekt het.

### **Belangrijk:**

Hoewel handig voor **lokale ontwikkeling of tests**, **is dit absoluut niet aanbevolen voor productieomgevingen**. Het uitschakelen van certificaatverificatie maakt je applicatie kwetsbaar voor man-in-the-middle-aanvallen.  
  
**🌐 8. Routes toevoegen**

In je web.php:

use App\Http\Controllers\Auth\GoogleController;

Route::get('/auth/redirect/google', [GoogleController::class, 'redirect'])->name('google.redirect');  
Route::get('/auth/callback/google', [GoogleController::class, 'callback'])->name('google.callback');

* google.redirect: stuurt de gebruiker door naar Google om in te loggen.
* google.callback: ontvangt de gebruiker terug van Google en verwerkt de login.

### **⚡ 9. Livewire component aanmaken**

Je maakt een Livewire-component aan met de naam auth.google-login.  
 In de methode redirectToGoogle() stuur je de gebruiker door naar de Google-login route.

php artisan make:livewire auth.google-login

In GoogleLogin.php:

<?php

namespace App\Livewire\Auth;

use Livewire\Component;

class GoogleLogin extends Component

{

public function redirectToGoogle()

{

return redirect()->route('google.redirect');

}

public function render()

{

return view('livewire.auth.google-login');

}

}

### **🎨 10. HTML voor knop (in google-login.blade.php)**

<div>

<button

wire:click="redirectToGoogle"

class="w-full flex items-center justify-center gap-2 bg-white text-zinc-700 border border-gray-300 px-4 py-2 rounded-lg hover:bg-gray-100 transition"

>

<svg class="w-5 h-5" viewBox="0 0 48 48">

<path fill="#FFC107" d="M43.6 20.5H42V20H24v8h11.3C33.5 32.5 29.2 36 24 36c-6.6 0-12-5.4-12-12s5.4-12 12-12c3.1 0 5.9 1.2 8 3.1l5.7-5.7C34.5 6.2 29.5 4 24 4 12.9 4 4 12.9 4 24s8.9 20 20 20 20-8.9 20-20c0-1.2-.1-2.3-.4-3.5z"/>

<path fill="#FF3D00" d="M6.3 14.7l6.6 4.8C14.1 16.1 18.7 14 24 14c3.1 0 5.9 1.2 8 3.1l5.7-5.7C34.5 6.2 29.5 4 24 4c-7.3 0-13.4 4.1-16.7 10.2z"/>

<path fill="#4CAF50" d="M24 44c5.2 0 10.2-2 13.9-5.3l-6.4-5.3c-2 1.4-4.5 2.2-7.5 2.2-5.2 0-9.5-3.5-11.1-8.2l-6.6 5.1C10.6 39.5 16.8 44 24 44z"/>

<path fill="#1976D2" d="M43.6 20.5H42V20H24v8h11.3c-1.1 3.2-3.4 5.8-6.4 7.4l6.4 5.3C38.8 37.5 44 31.4 44 24c0-1.2-.1-2.3-.4-3.5z"/>

</svg>

<span>Continue with Google</span>

</button>

</div>

### **📥 11. Voeg de knop toe aan je loginpagina**

<livewire:auth.google-login />

### **🧪 12. Test het resultaat**

php artisan serve

Ga naar:

<http://127.0.0.1:8000/login>

Druk op de Google-knop en test de login!

✅ Gefeliciteerd! Je hebt nu een **volledig werkende Google login**. Morgen roepen we de krachten van **GitHub** op.

**🗓️ Dag 3 – GitHub Login: De Schaduwzijde van de Auth**

### **🎯 Wat is het doel?**

Gebruikers kunnen zich aanmelden of inloggen via hun GitHub-profiel, net zoals met Google. Dit biedt hen meer keuze, en jou een robuustere login flow.

### **🧭 1. GitHub OAuth App aanmaken**

➡️ Ga naar: <https://github.com/settings/developers>

* Klik op **“New OAuth App”**
* Vul het formulier in:
  + **Application name**: Laravel App
  + **Homepage URL**: <http://127.0.0.1:8000>
  + **Authorization callback URL**: <http://127.0.0.1:8000/auth/callback/github>
* Klik op **“Register application”**
* Kopieer je **Client ID** en klik op “Generate a new client secret”

### **🔐 2. Voeg gegevens toe aan .env**

GITHUB\_CLIENT\_ID=je-client-id  
GITHUB\_CLIENT\_SECRET=je-client-secret  
GITHUB\_REDIRECT\_URI=http://127.0.0.1:8000/auth/callback/github

**🧠 3. Voeg GitHub-config toe in config/services.php**

'github' => [  
 'client\_id' => env('GITHUB\_CLIENT\_ID'),  
 'client\_secret' => env('GITHUB\_CLIENT\_SECRET'),  
 'redirect' => env('GITHUB\_REDIRECT\_URI'),  
],

### **🔧 4. Controller aanmaken**

php artisan make:controller Auth/GitHubController

In GitHubController.php:

public function redirect()  
{  
 return Socialite::driver('github')->redirect();  
}  
  
public function callback()  
{  
 $githubUser = Socialite::driver('github')->stateless()->setHttpClient(  
 new \GuzzleHttp\Client(['verify' => false])  
)->user();  
  
 $user = \App\Models\User::where('email', $githubUser->getEmail())->first();  
  
 if (! $user) {  
 $user = \App\Models\User::create([  
 'name' => $githubUser->getName() ?? $githubUser->getNickname(),  
 'email' => $githubUser->getEmail(),  
 'password' => bcrypt(Str::random(16)),  
 'github\_id' => $githubUser->getId(),  
 'avatar' => $githubUser->getAvatar(),  
 ]);  
 }  
  
 Auth::login($user);  
 return redirect()->route('dashboard');  
}

### **🌐 5. Routes toevoegen**

In routes/web.php:

use App\Http\Controllers\Auth\GitHubController;  
Route::get('/auth/redirect/github', [GitHubController::class, 'redirect'])->name('github.redirect');  
Route::get('/auth/callback/github', [GitHubController::class, 'callback'])->name('github.callback');

### **⚡ 6. Livewire-component maken**

php artisan make:livewire auth.github-login

In GitHubLogin.php:

public function redirectToGitHub()  
{  
 return redirect()->route('github.redirect');  
}

### **🎨 7. HTML voor GitHub-knop (in github-login.blade.php)**

<div>

<button

wire:click="redirectToGitHub"

class="w-full flex items-center justify-center gap-2 bg-white text-zinc-800 border border-gray-300 px-4 py-2 rounded-lg hover:bg-gray-100 transition"

>

<svg class="w-5 h-5" fill="currentColor" viewBox="0 0 24 24">

<path d="M12 .5C5.37.5 0 5.87 0 12.47c0 5.28 3.438 9.75 8.207 11.325.6.112.82-.262.82-.582 0-.288-.01-1.05-.016-2.063-3.338.738-4.042-1.61-4.042-1.61-.546-1.388-1.332-1.757-1.332-1.757-1.09-.757.083-.742.083-.742 1.204.085 1.837 1.257 1.837 1.257 1.07 1.835 2.807 1.304 3.492.997.107-.79.418-1.304.762-1.603-2.665-.305-5.466-1.354-5.466-6.025 0-1.33.475-2.417 1.256-3.27-.125-.307-.543-1.54.118-3.212 0 0 1.01-.327 3.3 1.25a11.51 11.51 0 0 1 3.003-.403c1.02.005 2.047.138 3.003.403 2.29-1.577 3.298-1.25 3.298-1.25.662 1.672.245 2.905.12 3.212.783.853 1.255 1.94 1.255 3.27 0 4.682-2.803 5.717-5.475 6.014.43.373.823 1.108.823 2.233 0 1.615-.014 2.92-.014 3.32 0 .322.217.698.825.58C20.565 22.22 24 17.752 24 12.47 24 5.87 18.627.5 12 .5z"/>

</svg>

<span>Continue with GitHub</span>

</button>

</div>

### **📥 8. Voeg de knop toe aan je loginpagina**

<livewire:auth.github-login />

🧠 Tip: je kunt beide knoppen samen onder je formulier zetten, bijvoorbeeld zo:

<div class="relative text-center">  
 <div class="my-4 flex items-center justify-center space-x-2 text-sm text-zinc-500">  
 <span class="w-1/5 border-t"></span>  
 <span>of</span>  
 <span class="w-1/5 border-t"></span>  
 </div>  
 <div class="space-y-4">  
 <livewire:auth.google-login />  
 <livewire:auth.github-login />  
 </div>  
</div>

### **🚀 9. Test het resultaat**

Start je server opnieuw (indien nodig):

php artisan serve

Ga naar:

<http://127.0.0.1:8000/login>

Test de GitHub-login met een geldig account!

✅ Proficiat! Je beheerst nu zowel de licht- als schaduwzijde van de loginkracht 😄  
 Morgen tackle je de ultieme uitdaging: **hybride login + wachtwoordbeheer**.

## **🗓️ Dag 4 – De Ultieme Test: Hybride Login en Wachtwoordbeheer**

### **🎯 Wat is het doel?**

Zorgen dat gebruikers die zich aanmelden via Google of GitHub later ook kunnen inloggen met e-mail en wachtwoord – zonder dubbele accounts of foutmeldingen.

### **⚔️ Het probleem**

* 🧑‍🚀 Een gebruiker meldt zich aan met Google → er is geen wachtwoord ingesteld.
* 🧑‍💻 Later probeert hij in te loggen met e-mail + wachtwoord → dat werkt niet.
* 📛 Als hij probeert zich te registreren → “E-mail bestaat al.”
* 🤯 Frustratie en verwarring!

### **✅ De oplossing: can\_set\_password**

We gaan Laravel slim maken:

* We voegen een boolean toe aan de users tabel: can\_set\_password
* Die bepaalt of een gebruiker een klassiek wachtwoord mag/moet instellen

### **🛠️ Stappen**

#### **1. Kolom toevoegen aan users migratie**

$table->boolean('can\_set\_password')->default(true);

Voer de migratie uit:

php artisan migrate:fresh --seed

#### **2. In User.php: $fillable bijwerken**

protected $fillable = [  
 'name',  
 'email',  
 'password',  
 'google\_id',  
 'github\_id',  
 'can\_set\_password',  
];

#### **3. In je GoogleController en GitHubController:**

Wanneer een gebruiker via Socialite wordt aangemaakt, voeg toe:

'can\_set\_password' => false,

#### **4. Instellen van een wachtwoord**

In de instellingenpagina (bijv. update-password component):

* Als can\_set\_password = false, toon geen veld voor huidig wachtwoord
* Valideer zonder current\_password in dat geval
* Na succesvol instellen van wachtwoord, update can\_set\_password naar true

Auth::user()->update([  
 'password' => Hash::make($this->password),  
 'can\_set\_password' => true,  
]);

**Open** resources\views\livewire\settings\password.blade.php  
 **Pas** de functie updatePassword() aan

public function updatePassword(): void { $rules = [ 'password' => ['required', 'string', Password::defaults(), 'confirmed'], ];

if ($this->can\_set\_password) {  
 $rules['current\_password'] = ['required', 'string', 'current\_password'];  
 }  
  
 try {  
 $validated = $this->validate($rules);  
 } catch (ValidationException $e) {  
 $this->reset('current\_password', 'password', 'password\_confirmation');  
 throw $e;  
 }  
  
 Auth::user()->update([  
 'password' => Hash::make($validated['password']),  
 'can\_set\_password' => true,  
 ]);  
  
 $this->reset('current\_password', 'password', 'password\_confirmation');  
  
 $this->dispatch('password-updated');  
}

### **🔐 Wat doet can\_set\_password?**

De eigenschap **can\_set\_password** wordt gebruikt om te bepalen **of de gebruiker verplicht is om zijn huidige wachtwoord in te geven** wanneer hij een nieuw wachtwoord instelt.

### **🧠 Waarom is dat belangrijk?**

Sommige gebruikers hebben zich misschien **geregistreerd via Google of GitHub**, en hebben dus **geen wachtwoord ingesteld**.  
 Bij hen is er geen bestaand wachtwoord om te controleren.  
 Zodra ze een wachtwoord aanmaken, willen we voortaan **extra veiligheid**: bij toekomstige wijzigingen moeten ze wél hun huidige wachtwoord invullen.

### **📘 Gedrag van de functie:**

if ($this->can\_set\_password) {  
 $rules['current\_password'] = ['required', 'string', 'current\_password'];  
}

Als can\_set\_password = true, dan:

* moet de gebruiker zijn **huidige wachtwoord** invullen en
* wordt het gevalideerd via de regel 'current\_password'.

Als can\_set\_password = false, dan:

* hoeft de gebruiker **geen huidig wachtwoord** in te vullen.
* dit is bijvoorbeeld het geval bij eerste keer wachtwoord instellen.

### **✅ Na succesvolle wijziging:**

Auth::user()->update([  
 'password' => Hash::make($validated['password']),  
 'can\_set\_password' => true,  
]);

→ Hiermee wordt het nieuwe wachtwoord opgeslagen  
 → én can\_set\_password wordt op true gezet: **de volgende keer is verificatie verplicht.**

### **💡 Bonus: informatieve melding tonen**

“Je account is aangemaakt via Google. Stel een wachtwoord in om ook klassiek te kunnen inloggen.”

—

|  |  |
| --- | --- |
| **🚀 Klaar!** Vanaf nu kunnen gebruikers:   * Met Google of GitHub inloggen * Later een wachtwoord instellen * Dan ook inloggen met e-mail + wachtwoord   ✅ Geen dubbele accounts  ✅ Geen foutmeldingen  ✅ Volledige flexibiliteit voor je gebruikers  🎉 *Gefeliciteerd! Je hebt alle krachten van de moderne login onder de knie.* |  |

## **🧘‍♂️ Tot slot – De Auth Saga is Voltooid**

En zo eindigt jouw queeste, jonge Jedi van de code.

Je hebt:

✅ een project opgezet uit het niets  
 ✅ Socialite getemd zoals een wilde Rancor  
 ✅ Google en GitHub leren praten met je database  
 ✅ de balans hersteld tussen klassiek inloggen en moderne magie  
 ✅ én de gebruikerservaring gered van de vergetelheid

En dat allemaal zonder je muis ooit als lightsaber te gebruiken (hoewel… misschien toch een beetje 🤺).

Maar vergeet niet… een echte developer is nooit klaar... hij compileert gewoon door naar het volgende avontuur!

