1v0 Ship v1.0 Faster

Laravel Fortify: The Authentication Backend Starter Kit

I had to implement **Laravel Fortify** in a project and the front-end team insisted on designing the entire UI themselves with Vue, so I couldn't rely on Breeze or Jetstream, which already come with their own scaffolding. At first, I wasn't sure how Fortify would fit, because unlike other starter kits, it doesn't give you ready-made views or layouts. But once I dug into it, I realized that was actually a huge advantage. I could enable all the authentication features we needed (registration, password reset, email verification, and even two-factor authentication) while letting the front-end developers craft the UI however they wanted.

It was flexible, backend-focused, and kept the project clean. Later, when we switched to a SPA setup with Sanctum, Fortify continued to power the backend without any issues. That experience convinced me that Fortify is the best option whenever you want complete freedom in building your application's interface, without sacrificing Laravel's authentication power. Since then, I've used it for mobile apps, Vue SPAs, and projects where design freedom was critical.

Laravel Fortify is a backend authentication implementation for Laravel applications. Unlike Breeze or Jetstream, it doesn't provide front-end scaffolding. Instead, it registers routes and controllers for login, registration, password reset, email verification, and two-factor authentication. This makes it perfect for developers who want full control over the user interface while leveraging Laravel's secure backend authentication system.

Installing Laravel Fortify

```
composer require laravel/fortify

php artisan vendor:publish --
provider="Laravel\Fortify\FortifyServiceProvider"
php artisan migrateCode language: JavaScript (javascript)
```

This installs Fortify, publishes its configuration file, and runs database migrations needed for user authentication and password resets.

Enabling Features

```
// config/fortify.php

'features' => [
    Features::registration(),
    Features::resetPasswords(),
    Features::emailVerification(),
    Features::updateProfileInformation(),
    Features::updatePasswords(),
    Features::twoFactorAuthentication(),
],Code language: PHP (php)
```

In `config/fortify.php` you can enable or disable features depending on your project. For example, you might disable registration if you want an invite-only system.

Customizing Views

```
// app/Providers/FortifyServiceProvider.php

use Laravel\Fortify\Fortify;

public function boot()
{
    Fortify::loginView(fn () => view('auth.login'));
    Fortify::registerView(fn () => view('auth.register'));
}Code language: PHP (php)
```

Here you connect Fortify's backend routes to your own Blade templates. That way you can design the login and registration forms exactly as you want.

Blade Example for Login

```
<!-- resources/views/auth/login.blade.php -->

<form method="POST" action="{{ route('login') }}">
    @csrf
    <label>Email</label>
    <input type="email" name="email" required autofocus />

    <label>Password</label>
    <input type="password" name="password" required />

    <button type="submit">Login</button>
</form>Code language: HTML, XML (xml)
```

This Blade form connects directly to Fortify's `login` route. No controller or route setup is required — Fortify handles the backend logic.

Vue SPA Login with Fortify + Sanctum

```
// resources/js/components/Login.vue

<template>
  <form @submit.prevent="login">
    <input v-model="form.email" type="email" placeholder="Email"
required />
    <input v-model="form.password" type="password"
placeholder="Password" required />
    <button type="submit">Login</button>
  </form>
</template>

<script setup>
import { reactive } from 'vue'
import axios from 'axios'

const form = reactive({
  email: '',
  password: ''
})

const login = async () => {
  await axios.get('/sanctum/csrf-cookie')
  await axios.post('/login', form)
  window.location.href = '/dashboard'
}
</script>Code language: HTML, XML (xml)
```

This Vue component integrates directly with Fortify's `/login` endpoint. Sanctum manages the CSRF protection and session, so the SPA can authenticate users securely.

1v0 Ship v1.0 Faster

Tips and Tricks

- Use `auth` and `verified` middleware to secure sensitive routes.
- Override default controllers if you want custom redirects after login.
- Enable `twoFactorAuthentication()` for additional security.
- Combine Fortify with Sanctum when building SPAs or mobile APIs.

Comparison: Fortify vs Breeze vs Jetstream

Feature	Fortify	Breeze	Jetstream
Purpose	Backend-only authentication	Minimal auth starter kit with Blade views	Advanced starter kit with teams and APIs
Front-end	None, fully custom	Blade + Tailwind (Inertia optional)	Livewire + Blade or Inertia + Vue/React
Teams	No	No	Yes
2FA	Yes	No	Yes
Best Use Case	SPAs, mobile apps, custom UIs	MVPs, small to medium apps	SaaS and team-based apps

See comparison of Laravel Starter Kits

Laravel Fortify gives you all the backend power of authentication without dictating how

the front end should look. Whether you're building a Blade-based UI, a Vue SPA, or even a mobile app, Fortify keeps the authentication logic consistent and secure while giving you total design freedom. If flexibility and control are priorities in your project, Fortify is the starter kit to choose.