Laravel Livewire Project Structure

Full-stack Laravel with Livewire. Reactive components without JavaScript.

#laravel #php #livewire #full-stack #spa #backend

PNG PDF Copy Prompt

Project Directory

```
myproject/
    artisan
    composer.json
    package.json
    tailwind.config.js
    vite.config.js
    .env
    .env.example
  app/
    Livewire/ Livewire compon…
      Counter.php Example compone…
      Users/ User domain
        UserTable.php
        CreateUser.php
        EditUser.php
      Forms/
        ContactForm.php
    Http/
      Controllers/ Minimal page co…
        Controller.php
        DashboardController.php
      Middleware/
    Models/
      User.php
    Actions/ Extracted busin…
      CreateUser.php
      UpdateUser.php
    Providers/
  bootstrap/
  config/
  database/
  resources/
    views/
      layouts/
        app.blade.php @livewireStyles…
      livewire/ Livewire compon…
        counter.blade.php
        users/
          user-table.blade.php
          create-user.blade.php
      components/ Blade components
        button.blade.php
        input.blade.php
      dashboard.blade.php
    css/
    js/
  routes/
    web.php
  tests/
  public/
  storage/
```

Why This Structure?

Livewire lets you build reactive UIs in pure PHP. Component classes handle state and actions, Blade templates render HTML. No JavaScript needed for forms, modals, or data tables—Livewire handles reactivity via AJAX.

Key Directories

app/Livewire/ - PHP component classes with properties and methods

resources/views/livewire/ - Blade templates for each component

app/Actions/ - Business logic extracted from components

resources/views/components/ - Reusable Blade UI components

Livewire Component

```
// app/Livewire/Users/UserTable.php
class UserTable extends Component
{
    public string $search = '';

    public function render()
    {
        return view('livewire.users.user-table', [
            'users' => User::where('name', 'like', "%{$thi
        ]);
    }
}
```

Getting Started

1. `composer create-project laravel/laravel myproject`
2. `composer require livewire/livewire`
3. `php artisan livewire:publish --config`
4. `php artisan make:livewire Counter`
5. Add `@livewireStyles` and `@livewireScripts` to layout

When To Use This

- Traditional apps wanting SPA-like reactivity
- Teams with strong PHP skills, weak JS skills
- CRUD apps with forms and data tables
- Admin panels and dashboards
- When you want to avoid frontend complexity

Trade-offs

Network round-trips - Every interaction hits the server

Not for real-time - Use WebSockets for live updates

Component state - Large states serialize on every request

Naming Conventions

Components - PascalCase: UserTable, CreateUser

Views - kebab-case: user-table.blade.php

Folders - Group by domain: Users/, Forms/