



# Why I Still Choose Laravel in 2026

For developers choosing a stack for real work.  
A practical case for productivity, flexibility,  
and an ecosystem that saves effort.

**Choose for the work.  
Then build something.**

TIME

**Expressive tools**



SUPPORT

**A connected ecosystem**



RESULT

**Applications you can ship**

# Time spent fighting tools is time lost.

Expressive APIs, clear documentation, and consistent patterns make everyday work easier.

## 01 Understand the next step

A familiar pattern reduces the decisions between an idea and a working feature.

## 02 Find the answer

Good documentation helps you solve a concrete problem without guessing.

## 03 Keep the work pleasant

Enjoying the daily tools matters when you maintain an app for years.

Developer experience is part of the decision. Performance, reliability, and fit still matter.

# Use tools that cover recurring work.

Choose what the application needs. A large toolbox is useful without installing all of it.

## Livewire

### Reactive interfaces

Server-driven interactions for many ordinary application screens.

## Filament

### Administration

Forms, tables, filters, and actions for the app's internal workflows.

## Pest

### Testing

Readable tests that describe what the application should do.

The article also points to deployment, queues, auth, search, billing, and community packages.

# Let the structure grow with the problem.

Laravel's defaults are a starting point. Add structure when the work gives it a purpose.

01

## Start simple

Controllers, models,  
and Blade templates.

02

## → Add useful boundaries

Actions, Services, or DTOs  
when responsibilities need a home.

03

## → Go deeper when needed

More domain structure  
for actual domain complexity.

Make the next change clearer. Do not add layers just to satisfy a diagram.

# Judge the language you can use now.

The article's preference includes modern PHP, not the reputation of its oldest versions.

Enums · readonly · match · named arguments

## Explicit choices

Enums give fixed options a name and keep those choices visible.

## Clear values

Types and readonly declarations help describe a data contract.

## Readable decisions

Modern syntax can make intent easier to follow in ordinary code.

Language features help. Good design, tests, and maintenance still depend on the application.

# A stack includes the people around it.

Documentation is one part of learning. Maintainers, peers, and teaching resources fill other gaps.

---

## Learn

Courses, articles, podcasts, and conference talks.

---

## Ask

An active community to turn to when you hit an unfamiliar problem.

---

## Contribute

Share a fix, a lesson, or an example that helps the next developer.

---

Jeffrey points to Laracon, Laracasts, and helpful package maintainers from his own experience.

# Choose for the constraints you actually have.

A productive default is useful. It is not an answer to every workload or team situation.

---

## Scale

Measure the bottleneck first.  
Extract specific work if the evidence calls for it.

---

## Architecture

A well-structured monolith can simplify delivery.  
Distribution introduces its own operating costs.

---

## Interfaces

Use a frontend approach that fits the interaction.  
Keep JavaScript where the product needs it.

---

The article makes a case for Laravel. It does not replace workload, team, or integration checks.

# Choose the stack that helps you ship.

Use the article as one perspective, then test the fit against your own application.

## 01 The work

What must the app do, and what are its constraints?

## 02 The team

Can the team build and maintain it with this stack?

## 03 The toolbox

Do the ecosystem's tools solve recurring needs?

## 04 The proof

Can you build one representative workflow well?

## 05 The next step

Ship something small and learn from it.

### A PRACTICAL FIT CHECK

Pick one real workflow.

Build it with the tools you expect to use in production.

Review the result, maintenance cost, and the gaps you found.

[Read the full article: Why I Still Choose Laravel in 2026](#)