



Uvod u Laravel

PREGLED POGLAVLJA

- Šta je Laravel?
- Ključne funkcionalnosti
- Prednosti korišćenja Laravela
- Kada koristiti Laravel?
- Instalacija Laravela
- Koraci za instalaciju
- Osnovna struktura aplikacije
- Glavni direktorijumi
- Ključne datoteke glavnog direktorijuma
- Tok HTTP zahteva kroz aplikaciju
- Šta je novo u Laravelu 13
- Zaključak

Šta je Laravel?

Laravel je moderan **PHP razvojni okvir**¹ otvorenog koda, namenjen razvoju veb-aplikacija i API servisa. Autor Laravela je **Taylor Otwell**, koji ga je kreirao 2011. godine. Laravel koristi **MVC (Model–View–Controller)** arhitekturu kao osnovni organizacioni model, uz dodatne slojeve i servise koji proširuju njene mogućnosti.

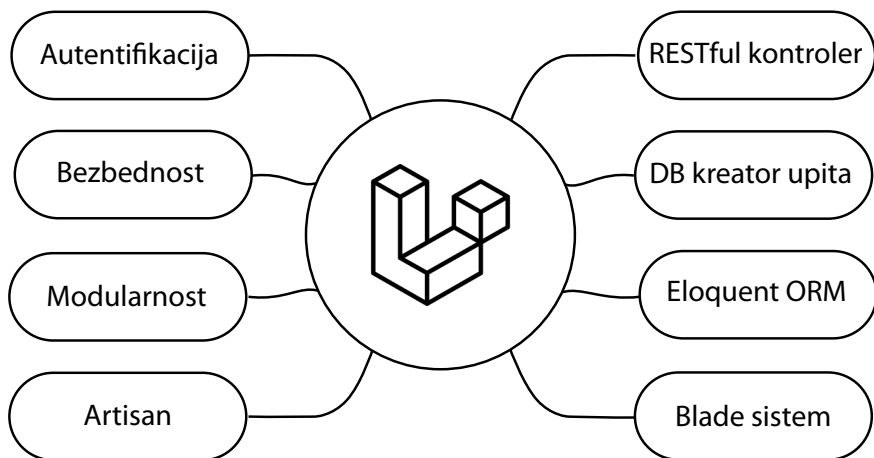
Priručnik je usklađen sa verzijom 13 Laravel razvojnog okvira i obuhvata njegove ključne funkcionalnosti.

Jedna od karakteristika po kojoj se Laravel izdvaja jeste njegovo kontinuirano unapređivanje. Kako ističe osnivač Laravela,

„Mislim da je jedna od stvari koja Laravel čini prilično jedinstvenim to što je to veliki backend veb razvojni okvir koji se neprestano unapređuje kroz našu sopstvenu upotrebu”. Taylor Otwell.

¹ Termin framework prevodi se kao razvojni okvir, jer označava unapred pripremljeni skup alata, pravila i struktura koji olakšavaju i ubrzavaju razvoj veb-aplikacija.

PHP je jedan od najrasprostranjenijih jezika za razvoj veb-aplikacija. Pokreće veliki broj sajtova i platformi, uključujući WordPress i mnoge druge. Uprkos brojnim novim tehnologijama, njegova jednostavnost, velika zajednica i bogat ekosistem biblioteka čine ga i dalje veoma relevantnim izborom za mnoge projekte.



Slika 1.1. Ključne funkcionalnosti Laravela

Slika 1.1. prikazuje ključne komponente i osnovne koncepte Laravela. Svaka prikazana stavka predstavlja jednu od temeljnih funkcionalnosti koje Laravel pruža za razvoj savremenih veb aplikacija.

Autentifikacija – ugrađeni mehanizam za upravljanje korisničkim nalogima koji obuhvata prijavu, odjavu, registraciju i resetovanje lozinke. Laravel obezbeđuje osnovne funkcionalnosti autentifikacije koje se mogu jednostavno prilagoditi potrebama aplikacije, uz podršku za rad sa sesijama i sigurnu proveru identiteta korisnika.

Bezbednost – skup ugrađenih mehanizama za zaštitu aplikacije od najčešćih bezbednosnih pretnji. Laravel obezbeđuje mehanizme za heširanje lozinke, šifrovanje podataka i zaštitu ruta, čime doprinosi bezbednom izvršavanju i pouzdanom funkcionisanju aplikacije.

Modularnost – princip organizacije izvornog koda u logički odvojene celine koje se mogu nezavisno razvijati, održavati i ponovo koristiti. Ovakav pristup povećava preglednost koda, olakšava proširivanje aplikacije i smanjuje međuzavisnost između komponenti.

Artisan – alat komandne linije namenjen automatizaciji razvojnih zadataka u Laravelu. Koristi se za generisanje izvornog koda, uključujući kontrolere, modele i migracije. Omogućava pokretanje migracija i popunjavanje baze podataka testnim podacima. Takođe se koristi za izvršavanje različitih sistemskih i razvojnih komandi koje ubrzavaju i standardizuju proces razvoja.

RESTful kontroler – tip kontrolera zasnovan na REST arhitektonskim principima, namenjen organizaciji logike za rad sa resursima. Sadrži standardne metode (npr. index, show, store, update, destroy), koje predstavljaju operacije nad resursom i mapiraju se na odgovarajuće HTTP metode (GET, POST, PUT/PATCH i DELETE). Time se postiže standardizovan i predvidiv način projektovanja veb API-ja.

DB kreator upita (*Query Builder*) – fluentni interfejs za konstruisanje SQL upita bez potrebe za direktnim pisanjem sirovog SQL koda. Podržava pregledniji, fleksibilniji i sigurniji rad sa bazom podataka, uz automatsku zaštitu od SQL injection napada.

Eloquent ORM – Laravelov objektno-relacioni mapirajući sistem (*Object-Relational Mapping*) koji koristi obrazac Active Record. Tabele baze podataka predstavljene su kroz PHP klase, odnosno modele. Pojedinačni zapisi predstavljeni su kao objekti, čime se obezbeđuje intuitivan i objektno orijentisan pristup radu sa podacima.

Blade sistem – Laravelov sistem za izradu i organizaciju korisničkog interfejsa. Blade omogućava pregledno povezivanje PHP logike i HTML sadržaja kroz šablone i prikaze, uz podršku za nasleđivanje rasporeda, komponente i direktive. Tokom izvršavanja, Blade šabloni i prikazi se kompajliraju u čisti PHP kod, što omogućava bolje performanse i efikasnije generisanje prikaza.

Prednosti korišćenja Laravela

- **Brz razvoj** – Ugrađene alatke i konvencije ubrzavaju razvoj aplikacija bez dodatnih konfiguracija.
- **Čitljiv i održiv kod** – MVC struktura jasno razdvaja poslovnu logiku od prikaza, što olakšava održavanje i proširenje aplikacije.
- **Aktivna zajednica i bogata dokumentacija** – Detaljna zvanična dokumentacija i velika programerska zajednica pružaju podršku kroz forume, tutorijale i pakete otvorenog koda.

- **Modularnost** – Novi moduli mogu se dodavati bez narušavanja postojeće logike.
- **Fleksibilnost** – Laravel je pogodan za različite vrste aplikacija, od jednostavnih veb-stranica do kompleksnih poslovnih sistema.

Kada koristiti Laravel?

Laravel je pogodan za:

- Projekte koji zahtevaju brz razvoj uz visoku skalabilnost.
- Timski rad zahvaljujući jasnoj strukturi.
- Aplikacije sa naprednim funkcijama koje zahtevaju složenu poslovnu logiku, rad sa korisnicima i ulogama, relacione baze podataka, kao i dugoročno održivu i testabilnu arhitekturu.
- Poslovne sisteme: ERP (softver za upravljanje poslovnim procesima), e-trgovina, rezervacije.

Laravel je odličan izbor za projekte gde je važno postići balans između performansi, funkcionalnosti i brzine razvoja.

Instalacija Laravela

Instalacija Laravela podrazumeva pripremu razvojnog okruženja i preuzimanje osnovne strukture aplikacije. Pravilno izvršena instalacija obezbeđuje stabilnu osnovu za dalji razvoj i nesmetan rad sa svim Laravel funkcionalnostima.

Razvojna okruženja za Laravel

Razvojno okruženje je skup alata i konfiguracija koji omogućavaju razvoj, pokretanje i testiranje aplikacije na lokalnom računaru.

U praksi se najčešće koriste tri pristupa konfiguraciji razvojnih okruženja.

1. Integrisana okruženja (Laragon, XAMPP, WAMP)

Ova okruženja kombinuju sve potrebne komponente – PHP, veb-server, bazu podataka i dodatne servise – u jedan paket, što omogućava brzu instalaciju i pokretanje projekta. Njihova mana je ograničena kontrola nad pojedinačnim komponentama i moguća razlika u odnosu na produkciono okruženje.

2. Laravel-orijentisano okruženje (*Herd*)

Laravel Herd je lokalno razvojno okruženje koje omogućava jednostavno pokretanje i upravljanje Laravel aplikacijama bez složene konfiguracije. U besplatnoj verziji prednost je optimizacija za Laravel i brz rad, dok je ograničenje to što je potrebna dodatna instalacija i konfiguracija npr. MySQL baze podataka.

3. Kontejnerska okruženja (*Docker*)

Aplikacija i svi servisi pokreću se u izolovanim kontejnerima, što omogućava identičnu lokalnu i produkcionu konfiguraciju – idealno za timski rad. Nedostatak je složenija konfiguracija i duži period učenja kako bi se efikasno koristilo.

Pre instalacije potrebno je **proveriti** da li okruženje zadovoljava osnovne sistemske zahteve za aktuelnu verziju Laravela.

Sistemske zahteve:

1. **PHP:** verzija 8.3 ili novija.
2. **Composer:** alat za upravljanje PHP zavisnostima (*najnovije izdanje*).
3. **MySQL** (Ukoliko se koristi).
4. **Veb-server:** Apache ili Nginx.
5. **Node.js / npm:** upravljanje frontend zavisnostima.
6. **Baza podataka:** MySQL, PostgreSQL, SQLite i SQL Server.

Koraci za instalaciju novog projekta

1. Instalacija pomoću Composer-a

Composer preuzima potrebne zavisnosti i kreira osnovnu strukturu aplikacije.

a. Preuzimanje Laravel projekta

Instalacija novog projekta:

```
composer create-project --prefer-dist laravel/laravel  
ime_projekta
```

pri čemu je Composer prethodno instaliran **lokalno**.

Napomena: Parametar `--prefer-dist` je opcioni i ubrzava instalaciju preuzimanjem distributivnih paketa umesto kloniranja Git repozitorijuma.

Komanda kreira novi projekat u direktorijumu `ime_projekta` i instalira najnoviju stabilnu verziju Laravela.

b. Instalacija preko Laravel instalera

Ako je Composer **globalno** instaliran Laravel instalater se dodaje samo jednom:

```
composer global require laravel/installer
```

Nakon toga novi projekat kreira se komandom:

```
laravel new ime_projekta
```

Zatim se ulazi u glavni direktorijum projekta.

```
cd ime_projekta
```

Korisnik zatim kreira bazu podataka, na primer MySQL:

```
CREATE DATABASE ime_baze;
```

2. Podešavanje okruženja

Datoteka `.env` čuva konfiguracione vrednosti koje određuju ponašanje aplikacije, odvajajući ih od izvornog koda. Nalazi se u glavnom direktorijumu projekta i sadrži parametre kao što su podešavanja konekcije ka bazi podataka, jedinstveni aplikacioni ključ, konfiguracija e-pošte i druge promenljive okruženja.

Podešavanje osnovnih parametara

Potrebno je otvoriti datoteku `.env` i podesiti sledeće parametre za povezivanje sa bazom:

- **Naziv i tip baze podataka**

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=ime_baze
DB_USERNAME=korisnik
DB_PASSWORD=lozinka
```

- **Jedinstven ključ aplikacije**

Laravel zahteva jedinstven ključ aplikacije `APP_KEY` za enkripciju podataka. Ukoliko ključ nije postavljen Artisan komanda generiše ključ:

```
php artisan key:generate
```

3. Zatim se izvršavaju migracije kako bi se kreirale tabele u bazi:

```
php artisan migrate
```

4. Pokretanje razvojnog servera

Laravel uključuje ugrađeni razvojni server koji se pokreće komandom:

```
php artisan serve
```

Aplikacija postaje dostupna na adresi `http://localhost:8000`. Početna stranica Laravela potvrđuje uspešnu instalaciju.

5. Dodatni koraci

Instalacija i pokretanje Node.js paketa. Ako se koriste Vite, Vue, React ili SCSS, potrebno je instalirati odgovarajuće `npm` pakete komandom:

```
npm install
```

Nakon toga, komanda

```
npm run dev
```

kompajlira i pokreće frontend resurse u razvojnem okruženju aplikacije.

Osnovna struktura aplikacije

Struktura Laravel aplikacije sledi principe MVC arhitekture i podrazumeva jasan raspored poslovne logike, prikaza, ruta i resursa.

Pregled direktorijuma

- `app/` poslovna logika aplikacije.
 - `Http/`: veb sloj aplikacije.
 - `Controllers/`: HTTP kontroleri.
 - `Middleware/`: filteri zahteva.
 - `Requests/`: validaciju ulaza.
 - `Models/`: Eloquent modeli.
- `bootstrap/`
 - Datoteka `app.php` inicijalizuje osnovne Laravel komponente i obezbeđuje automatsko učitavanje klasa.

- Direktorijum `cache` čuva keširane podatke radi bržeg izvođenja aplikacije.
- Datoteka `providers.php` registruje servisne provajdere koje treba učitati.

- `config/`

Direktorijum sadrži konfiguracione datoteke (`app.php`, `database.php`, `mail.php`) koje obezbeđuju jednostavno prilagođavanje aplikacije.

- `database/`

Direktorijum namenjen za rad sa bazom podataka:

- `factories/`: „Fabrika modela” (*Factory*) za generisanje testnih podataka.
- `migrations/`: Skripte za upravljanje bazama podataka.
- `seeders/`: Alati za unošenje osnovnih podataka u bazu.

- `lang/`

Direktorijum za lokalizaciju aplikacije.

- `public/`

Korenski direktorijum koji je javno dostupan. Sadrži `index.php` datoteku, kao i statičke resurse poput CSS-a, JavaScript-a i slike.

- `resources/`

Sadrži resurse za frontend:

- `views/`: Blade šabloni i prikazi podataka.
- `lang/`: Datoteke za lokalizaciju jezika.
- `css/`, `js/`: Izvorne frontend datoteke koje se koriste kada je aktiviran alat kao što je Vite.

- `routes/`

Sve datoteke za definisanja ruta aplikacije.

- `web.php`: Rute za veb-aplikaciju.
- `console.php`: Definicija konzolnih komandi.

- `storage/`

Prostor za privremene datoteke:

- `app/`: Podaci specifični za aplikaciju.
- `framework/`: Privremene datoteke poput sesija, keša i prikaza.
- `logs/`: Logovi aplikacije.

- `tests/`

Direktorijum za testove.

- **Feature/**: Testovi koji obuhvataju kompletne funkcionalnosti.
- **Unit/**: Testovi za pojedinačne klase i metode.
- **vendor/**: Direktorijum za sve biblioteke koje se instaliraju pomoću Compose-
ra. Laravel je ovde smešten.

Ključne datoteke

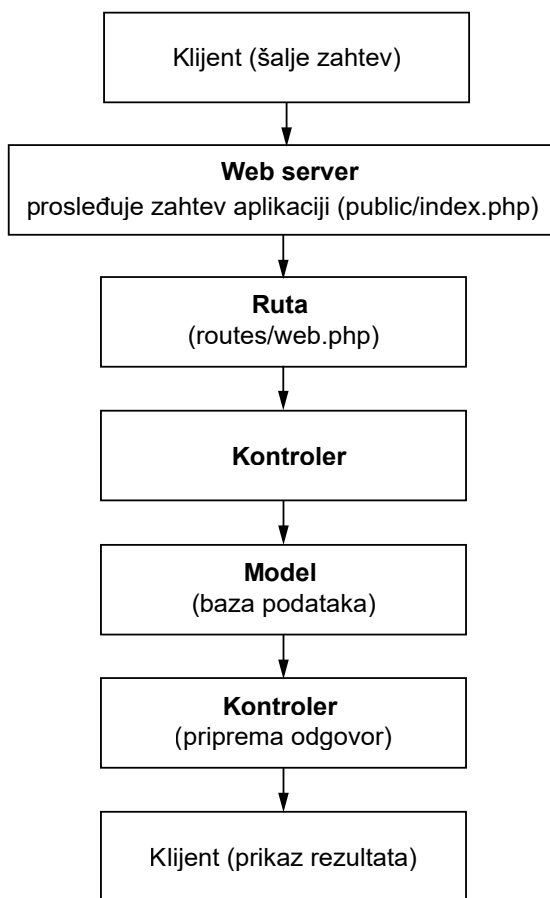
Naziv datoteke	Funkcija	Napomena
.env	Glavna konfiguracija aplikacije (baza, API ključevi, servisi)	Ne čuvati u Git repozitorijumu, podesiti lokalno ili preko sistema za distribuciju
artisan	CLI alat za Laravel zadatke (generisanje kontrolera, migracija, testiranje)	Središnji alat za razvoj i administraciju
composer.json	Definiše PHP zavisnosti i Laravel pakete	Nakon izmene koristiti <code>composer install</code> ili <code>composer update</code>
composer.lock	Zaključava verzije PHP paketa	Osigurava konzistentnu instalaciju
package.json	Definiše JS pakete i skripte za frontend	Laravel koristi Vite; sadrži dev i build skripte
package-lock.json / pnpm-lock.yaml/ yarn. lock	Zaključava verzije JS paketa	Osigurava konzistentan build
vite.config.js	Konfiguracija Vite alata za kom- pajliranje frontend resursa	Definiše ulazne tačke, HMR, integraciju sa Tailwind/Vue/React
server.php	Alternativni development server	Obično se koristi samo u specifič- nim situacijama
README.md	Dokumentacija projekta	Uputstva za instalaciju, razvoj i produkciju
.gitignore	Definiše koje datoteke Git ignoriše	Sprečava uključivanje osetljivih ili generisanih datoteka
phpunit.xml	Konfiguracija testiranja aplikacije	Za PHPUnit testove
.env.example	Primer .env datoteke	Za postavljanje okruženja
tailwind.config.js	Konfiguracija Tailwind CSS-a	Samo ako se Tailwind koristi u projektu
.editorconfig	Standardi formatiranja koda	Opcionalno, za uniformnost u timu

Kako Laravel funkcioniše

Tipičan HTTP zahtev u Laravel aplikaciji prolazi kroz jasan tok: započinje de-
finicijom u `routes/web.php`, potom se prosleđuje odgovarajućem kontroleru u `app/`

Http/Controllers/. Kontroler komunicira sa modelima iz app/Models/, a dobijeni podaci prosleđuju se i prikazuju u odgovarajućim prikazima (resources/views). Takva struktura obezbeđuje jasnu podelu odgovornosti, što olakšava održavanje i proširivanje aplikacije.

Na slici 1.2. prikazan je pojednostavljen tok HTTP zahteva kroz aplikaciju. Kontroler najpre komunicira sa modelom, a zatim priprema i vraća odgovor klijentu. Ovakav prikaz omogućava jasnije sagledavanje uloge ruta, kontrolera, modela i prikaza, kao i njihove međusobne interakcije.



Slika 1.2. Pojednostavljeni dijagram toka procesa u aplikaciji

Na dijagramu nisu eksplicitno prikazani sledeći slojevi i mehanizmi, iako su oni sastavni deo realne Laravel aplikacije:

- **Validacija podataka** – izvršava se pre obrade zahteva, najčešće kroz `FormRequest` klase ili validaciju unutar kontrolera.
- **Autentifikacija** – provera identiteta korisnika putem Auth sistema.
- **Autorizacija** – kontrola pristupa resursima kroz `Policy` i Gate mehanizme.
- **Middleware** – mehanizam kroz koji HTTP zahtev i odgovor prolaze pre i posle obrade u aplikaciji.
- **Servisne klase** – izdvajanje poslovne logike iz kontrolera radi bolje organizacije i testabilnosti koda.

Ovi elementi se u praksi integrišu u tok HTTP zahteva, pre njegovog dolaska u kontroler ili tokom same obrade u kontroleru.

Praktičan primer osnovne funkcionalnosti

Sledeći primer demonstrira kreiranje jednostavne funkcionalnosti za prikaz liste zadataka. Cilj je prikazati URL `/tasks` sa listom zadataka preuzetih iz baze podataka.

Primer obuhvata ključne elemente Laravel aplikacije – **migraciju, model, kontroler, rutu i prikaz podataka** – i demonstrira osnovni tok obrade HTTP zahteva.

Migracija

Prvi korak je kreiranje migracije za tabelu `tasks`, koja omogućava definisanje i izmenu strukture baze podataka pomoću PHP koda.

Komanda za kreiranje migracije:

```
php artisan make:migration create_tasks_table
```

Laravel generiše datoteku sa imenom: `YYYY_MM_DD_HHMMSS_create_tasks_table.php`

U generisanoj datoteci definiše se struktura tabele:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
```

```

{
    public function up(): void
    {
        Schema::create('tasks', function (Blueprint $table) {
            $table->id(); // primarni ključ, auto-increment
            $table->string('title'); // naslov zadatka
            $table->timestamps(); // polja created_at i updated_at
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('tasks'); // brisanje tabele prilikom rollbacka
    }
};

```

Komanda koja pokreće neizvršene migracije:

```
php artisan migrate
```

Laravel kreira tabelu tasks u bazi podataka prema definisanoj strukturi.

Model

Sledeći korak je kreiranje Eloquent modela Task koji predstavlja entitet u aplikaciji i enkapsulira logiku za rad sa tabelom u bazi podataka.

```
php artisan make:model Task
```

Model Task u datoteci app/Models/Task.php koristi atribut #[Fillable] kao novu funkcionalnost za definisanje polja koja se mogu masovno dodeljivati prilikom kreiranja ili ažuriranja zapisa. Svojstvo \$fillable, korišćeno u ranijim verzijama, i dalje je podržano u najnovijoj verziji Laravela.

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Attributes\Fillable;

#[Fillable('title')]
class Task extends Model
{
    //
}

```

Model omogućava jednostavnu i intuitivnu interakciju sa tabelom `tasks` putem Eloquent ORM-a.

Kontroler

Kontroleri primaju HTTP zahteve, obrađuju podatke pomoću modela i vraćaju odgovarajući odgovor.

Komanda kreira kontroler:

```
php artisan make:controller TaskController
```

U kontroleru `TaskController` definisana je metoda `index` koja preuzima sve zadatke iz baze i prosleđuje ih prikazu:

```
<?php

namespace App\Http\Controllers;

use App\Models\Task;
use Illuminate\View\View;

class TaskController extends Controller
{
    // prikazuje listu svih zadataka
    public function index(): View
    {
        // dohvata sve zadatke iz baze
        $tasks = Task::all();

        // prosleđuje podatke u prikaz resources/views/tasks/index.blade.php
        return view('tasks.index', compact('tasks'));
    }
}
```

`Task::all()` preuzima sve zapise iz tabele povezane sa modelom `Task` (podrazumevano `tasks`). Funkcija `compact('tasks')` kreira niz oblika `['tasks' => $tasks]` koji se prosleđuje prikazu.

Ruta

Ruta definiše kako aplikacija prima HTTP zahtev i mapira ga na odgovarajuću metodu kontrolera koja obrađuje zahtev.

U datoteci `routes/web.php` definisana je ruta koja povezuje URL `/tasks` sa metodom `index` kontrolera `TaskController`:

```
use App\Http\Controllers\TaskController;

Route::get('/tasks', [TaskController::class, 'index']);
```

Prikaz podataka

Blade prikaz služi za prikazivanje podataka korisniku.

Datoteka `resources/views/tasks/index.blade.php` sadrži prikaz liste zadataka:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Lista zadataka</title>
</head>
<body>
  <h1>Lista zadataka</h1>

  <ul>
    @foreach($tasks as $task)
      <li>{{ $task->title }}</li>
    @endforeach
  </ul>
</body>
</html>
```

Napomena: Blade direktiva `@foreach` iterira kroz kolekciju `$tasks`. Dvostruke vitičaste zagrade `{{ }}` bezbedno ispisuju podatke jer automatski pružaju zaštitu od XSS napada (*zlonamerni skript se ubacuje i izvršava u pregledaču korisnika*).

Prikazani primer ilustruje osnovnu strukturu Laravel aplikacije i način na koji njeni ključni slojevi međusobno sarađuju:

1. Migracija definiše strukturu baze podataka.
2. Model predstavlja podatke i obezbeđuje rad sa njima.
3. Kontroler upravlja logikom i priprema podatke.
4. Ruta povezuje HTTP zahtev sa metodom kontrolera.
5. Blade prikaz renderuje podatke i vraća HTML odgovor.

Ovaj tok predstavlja osnovu svakog Laravel projekta.

Šta je novo u Laravelu 13

Laravel 13 zvanično je objavljen **17. marta 2026. godine** i donosi niz značajnih unapređenja, uz poseban fokus na **minimalan broj izmena koje mogu narušiti kompatibilnost postojećih aplikacija**. Ova verzija predstavlja evolutivno unapređenje razvojnog okvira, sa naglaskom na veću produktivnost, savremene PHP prakse i integraciju funkcionalnosti zasnovanih na veštačkoj inteligenciji.

1. Priprema: PHP 8.3 kao obavezan uslov

Prvi korak pre korišćenja novih funkcionalnosti jeste ažuriranje PHP okruženja. Laravel 13 zahteva **PHP 8.3** ili noviju verziju, što predstavlja obavezan preduslov za instalaciju i nadogradnju sistema.

2. Korišćenje PHP atributa

Jedna od najznačajnijih novina u razvojnem iskustvu jeste proširena podrška za **PHP attribute**. Ovakav pristup omogućava definisanje konfiguracije neposredno iznad klase ili metode, čime se smanjuje potreba za dodatnim javnim svojstvima i poboljšava preglednost koda.

Uместo ranijeg pristupa:

```
class Post extends Model
{
    protected $fillable = ['title', 'body', 'status'];
}
```

sada je moguće koristiti attribute:

```
use Illuminate\Database\Eloquent\Attributes\Fillable;

#[Fillable('title', 'body', 'status')]
class Post extends Model
{
    //
}
```

Podrška za attribute u Laravelu 13 proširena je na više delova razvojnog okvira, uključujući kontrolere, autorizaciju, redove čekanja, validaciju i Eloquent modele. Ova funkcionalnost je opcionalna, pa su postojeći pristupi i dalje u potpunosti podržani.

3. Integracija AI funkcionalnosti

Laravel 13 uvodi zvanični **Laravel AI SDK**, koji omogućava jedinstvenu integraciju različitih AI provajdera unutar aplikacije. Ovim pristupom razvoj funkcionalnosti zasnovanih na veštačkoj inteligenciji postaje dosledniji, jer se komunikacija sa različitim servisima obavlja kroz jedinstven, Laravel-native (izvorni) API.

SDK obuhvata širok skup mogućnosti:

- generisanje teksta,
- agente sa podrškom za pozivanje alata,
- generisanje embedding vektora,
- obradu audio i vizuelnog sadržaja,
- integraciju sa vektorskim bazama podataka,
- semantičku pretragu.

Embedding vektori predstavljaju numerički zapis podataka u obliku niza brojeva, čime sistem može da proceni značenje i sličnost između pojmova. Na taj način aplikacija ostaje nezavisna od konkretnog AI provajdera, uz zadržavanje konzistentnog razvojnog iskustva.

4. Usmeravanje redova čekanja

Dodata je nova mogućnost **centralizovanog usmeravanja poslova u redove čekanja**. Sada je moguće definisati pravila koja određuju na koji red i konekciju se određeni posao automatski prosleđuje.

Konfiguracija se vrši, na primer, u klasi `AppServiceProvider`:

```
use App\Jobs\ProcessPodcast;  
use Illuminate\Support\Facades\Queue;  
  
Queue::route(ProcessPodcast::class, connection: 'redis', queue: 'podcasts');
```

Na ovaj način svaki posao klase `ProcessPodcast` automatski se usmerava na Redis konekciju i smešta u red `podcasts`, čime se postiže centralizovano i pregledno upravljanje redovima čekanja.

5. Napredna pretraga (JSON:API i vektorska pretraga)

Laravel 13 sada u svom jezgru nudi podršku za **JSON:API kompatibilne resurse**, uključujući:

- serijalizaciju odgovora,

- uključivanje relacionih podataka,
- generisanje linkova,
- odgovarajuća HTTP zaglavlja.

Pored toga, uvedena je podrška za **vektorsku i semantičku pretragu**, naročito uz PostgreSQL i pgvector ekstenziju.

```
$documents = DB::table('documents')
    ->whereVectorSimilarTo('embedding', 'Najbolje vinarije u Napa dolini')
    ->limit(10)
    ->get();
```

Ovakav upit pronalazi dokumente čija je semantička sličnost najbliža zadatoj frazi.

6. Upravljanje kešom

Dodata je nova metoda `Cache::touch()`, koja omogućava produžavanje vremena trajanja već keširane vrednosti bez potrebe za njenim čitanjem i ponovnim upisom.

```
Cache::touch('user_session:123', 3600);

Cache::touch('analytics_data', now()->addHours(6));

Cache::touch('report_cache', null);
```

Ovakav pristup unapređuje performanse, jer se vreme isteka ažurira jednom operacijom, bez dodatnog prenosa same vrednosti kroz aplikaciju. Posebno je koristan pri radu sa velikim količinama podataka u kešu.

7. Autentifikacija i bezbednost

Passkey autentifikacija

Starter kitovi (*početni paketi*) sada imaju ugrađenu podršku za **Passkeys autentifikaciju**. Ova funkcionalnost omogućava prijavu bez upotrebe lozinke, korišćenjem:

- biometrijskih podataka,
- PIN koda,
- hardverskog bezbednosnog ključa.

Na taj način proces autentifikacije postaje brži, jednostavniji i bezbedniji, uz dodatnu zaštitu od phishing napada i krađe lozinki.

8. Reverb podrška za baze podataka

Laravel Reverb dobio je novi drajver za rad sa bazom podataka, čime je pojednostavljeno horizontalno skaliranje aplikacija koje koriste WebSocket komunikaciju u realnom vremenu.

Laravel Reverb je WebSocket server namenjen ostvarivanju stalne dvosmerne komunikacije između klijenta i servera unutar Laravel aplikacija. WebSocket predstavlja protokol koji omogućava razmenu podataka bez potrebe za ponovnim uspostavljanjem veze.

Umesto obavezne upotrebe Redis infrastrukture, sada je moguće koristiti postojeću **MySQL** ili **PostgreSQL** bazu podataka za koordinaciju komunikacije između više instanci aplikacije. Redis je brza baza podataka koja podatke čuva u RAM memoriji.

Ovakav pristup posebno je koristan za male i srednje projekte, jer smanjuje infrastrukturnu složenost i troškove održavanja.

9. Konzolne komande

Artisan komande sada takođe podržavaju pristup zasnovan na atributima.

```
use Illuminate\Console\Command;
use Illuminate\Console\Attributes\Signature;
use Illuminate\Console\Attributes>Description;

#[Signature('reports:generate {user} [--queue]')]
#[Description('Generisanje mesečnog analitičkog izveštaja za određenog korisnika.')]
class GenerateReportCommand extends Command
{
    public function handle()
    {
        $user = $this->argument('user');
    }
}
```

U ovom pristupu definicija komande vrši se putem PHP atributa `Signature` i `Description`, čime se pojednostavljuje struktura klase i eliminiše potreba za eksplicitnim svojstvima `$signature` i `$description`.

Prelazak na Laravel 13

Laravel 13 donosi **evolutivna unapređenja, a ne radikalne promene**. Fokus razvojnog tima usmeren je na povećanje produktivnosti kroz savremene PHP prakse,

posebno kroz upotrebu atributa, kao i na pripremu aplikacija za integraciju AI sistema.

Zahvaljujući minimalnom broju nekompatibilnih izmena, proces nadogradnje na novu verziju Laravela je relativno jednostavan. To ujedno znači da je u velikom broju slučajeva moguće koristiti kod pisan za prethodne verzije, kao i postojeće video tutorijale i druge obrazovne materijale, uz manje ili nikakve izmene.

Zaključak

Laravel je moćan i elegantan PHP razvojni okvir koji značajno olakšava razvoj savremenih veb-aplikacija. Njegova jasna struktura, bogat skup ugrađenih funkcija i pažljivo osmišljena arhitektura čine ga pouzdanim alatom za projekte svih veličina i nivoa složenosti. Modularnost, čitljivost i fleksibilnost koje Laravel nudi ubrzavaju razvoj i olakšavaju kasnije održavanje aplikacije.

Ovo uvodno poglavlje predstavilo je ključne pojmove i osnove Laravela, kao i njegove glavne prednosti. Nakon ovog poglavlja, čitalac treba da

- razume šta je Laravel i koje su njegove glavne prednosti,
- razume tok HTTP zahteva,
- zna kako da instalira Laravel i pokrene novu aplikaciju,
- poznaje osnovnu strukturu direktorijuma i organizaciju Laravela.

Ovim znanjem stečen je solidan temelj za dalji rad u Laravel okruženju.

U ovom delu predstavljene su i osnovne izmene uvedene u Laravelu 13.