# Predgovor

**Laravel – priručnik za izradu veb-aplikacija** koncipiran je kao vodič kroz razvoj savremenih veb-aplikacija zasnovanih na Laravelu, jednom od najrasprostranjenijih i najsveobuhvatnijih PHP razvojnih okvira u savremenom veb programiranju. Laravel se izdvojio kao vodeće rešenje zahvaljujući preglednoj arhitekturi, izraženoj fleksibilnosti i bogatom ekosistemu koji omogućava izgradnju aplikacija različitog obima i složenosti, uz očuvanje visokih standarda stabilnosti i efikasnosti.

Priručnik je namenjen širokom spektru čitalaca – od onih koji se prvi put susreću sa Laravelom do onih koji žele da sistematizuju i prodube postojeća znanja. Tekst je strukturiran tako da omogući postepeno savladavanje gradiva: od osnovnih pojmova i koncepata, preko sadržaja srednjeg nivoa složenosti, do naprednih tehnika koje omogućavaju implementaciju kompleksnih rešenja u profesionalnom okruženju.

U okviru sadržaja obrađeni su fundamentalni aspekti razvoja veb-aplikacija: uvod u MVC arhitekturu, Blade sistem, osnovni principi rutiranja, kontroleri, modeli, migracije i početna konfiguracija aplikacije. Poseban segment posvećen je radu sa graditeljem upita, Eloquent ORM-om, middlewareom, komponentama i izgradnji REST API-ja. Detaljno se razmatraju i napredni mehanizmi poput servisnih klasa, FormRequest klasa, testiranja i redova čekanja, kao i koncepti provajdera servisa, injektovanja zavisnosti i arhitekture zasnovane na događajima. Time se čitaocu omogućava sveobuhvatno sagledavanje Laravel ekosistema, uz jasno razlikovanje teorijske osnove i praktične primene.

Poglavlja su oblikovana tako da kombinuju teorijska razmatranja sa primerima iz prakse, čime se olakšava razumevanje i primena obrađenih pojmova. Na taj način uspostavlja se čvrsta veza između teorijskog okvira i konkretnih razvojnih scenarija, što je od posebnog značaja za profesionalni rad.

Prilikom izrade priručnika korišćena su i savremena rešenja zasnovana na veštačkoj inteligenciji – ChatGPT, Claude AI i DeepSeek– kao pomoćni alati u procesu strukturiranja gradiva, razjašnjavanja tehničkih nedoumica i konsultovanja stručne literature. Njihova primena doprinela je preciznosti objašnjenja i omogućila dodatno produbljivanje složenijih aspekata Laravela.

Očekuje se da ovaj priručnik doprinese jasnijem sagledavanju i lakšem savladavanju Laravela, te da posluži kao pouzdana literatura i podsticaj za dalji razvoj u oblasti veb programiranja.

### Napomena o terminologiji

U tekstu je primenjena kombinacija srpskih i engleskih stručnih izraza, u skladu sa uobičajenom praksom u Laravel zajednici i šire u IT industriji:

- Kada postoji opšteprihvaćen i jasno razumljiv prevod, koristi se srpski izraz (npr. *pogled* za *view*, *ruta* za *route*, *kontroler* za *controller*).
- Kada je termin prepoznatljiviji u izvornom obliku ili bi prevod mogao izazvati zabunu, zadržava se engleski izraz (npr. *middleware*, *seeder*, *factory*).
- Prva pojava termina u tekstu po pravilu je propraćena prevodom ili objašnjenjem u zagradi radi jasnoće.

Na ovaj način obezbeđena je doslednost, stručna preciznost i čitljivost teksta, uz očuvanje prirodnog toka jezika.

U trenutku pripreme ovog priručnika aktuelna verzija Laravela bila je 13, te je sadržaj prilagođen toj verziji.

U Beogradu,
2026. godine

**Priredio**
Zoran Bojanić

# SADRŽAJ

**Predgovor** . . . . . **5**

**1. UVOD U LARAVEL** . . . . . **21**
Šta je Laravel? . . . . . 21
Prednosti korišćenja Laravela . . . . . 23
Kada koristiti Laravel? . . . . . 24
Instalacija Laravela . . . . . 24
Koraci za instalaciju novog projekta . . . . . 25
Osnovna struktura aplikacije . . . . . 27
Pregled direktorijuma . . . . . 27
Ključne datoteke . . . . . 29
Kako Laravel funkcioniše . . . . . 29
Praktičan primer osnovne funkcionalnosti . . . . . 31
Šta je novo u Laravelu 13 . . . . . 35
Zaključak . . . . . 39

**2. MVC ARHITEKTURA** . . . . . **41**
Uvod . . . . . 41
Istorijat MVC arhitekture . . . . . 42
Osnovne komponente MVC obrasca . . . . . 43
Model . . . . . 43
Prikaz . . . . . 44
Kontroler . . . . . 45
Ruta - most ka MVC-u . . . . . 46

Tok zahteva u MVC-u . . . . . 46
Sažeti pregled MVC komponenti . . . . . 48
Prednosti MVC arhitekture . . . . . 48
Proširenja MVC obrasca . . . . . 48
MVC i njegove varijante . . . . . 49
Zaključak . . . . . 52

**3. RUTE . . . . . 53**
Uvod u rutiranje . . . . . 53
Pristup obradi ruta . . . . . 54
Datoteke za rute . . . . . 57
API rute . . . . . 57
Dostupne HTTP metode ruta . . . . . 59
Parametri rute . . . . . 60
Obavezni parametri . . . . . 60
Opcioni parametri . . . . . 60
Parametri sa regularnim izrazima . . . . . 61
Injektovanje zavisnosti u rutama . . . . . 62
Organizacija ruta . . . . . 63
Imenovane rute . . . . . 63
Grupe ruta . . . . . 64
Kontroler i grupa ruta . . . . . 65
Prefiksi ruta . . . . . 65
Prefiksi imena ruta . . . . . 65
Primena middleware na rutama . . . . . 66
Pregled registrovanih ruta . . . . . 66
Napredne mogućnosti ruta . . . . . 67
Vezivanje modela za rutu . . . . . 67
Implicitno vezivanje modela za rutu . . . . . 68
Prilagođeni ključ u ruti . . . . . 70
Eksplicitno vezivanje modela za rutu . . . . . 71

Potpisane rute . . . . . 72
Ograničavanje broja zahteva . . . . . 76
Rezervne rute . . . . . 78
Preusmeravanje ruta . . . . . 79
Implementacija prikrivanja metoda u HTML formama . . . . . 81
Keširanje ruta . . . . . 82
HTTP status kodovi i njihovo značenje . . . . . 83
Predlozi za optimizaciju ruta . . . . . 84
Ukratko . . . . . 85

**4. MIDDLEWARE . . . . . 87**
Uloga middleware u obradi zahteva i odgovora . . . . . 89
Kako middleware funkcioniše . . . . . 89
Kreiranje i registracija middleware komponente . . . . . 92
Kreiranje middleware . . . . . 92
Registracija middleware . . . . . 94
Globalni middleware . . . . . 94
Dodeljivanje middleware rutama . . . . . 94
Imenovani middleware . . . . . 95
Ugrađeni middleware . . . . . 98
Grupe middleware . . . . . 99
Prosleđivanje parametara . . . . . 100
Terminabilni middleware . . . . . 100
Bezbednosni aspekti . . . . . 102
Obrada grešaka . . . . . 104
Uticaj na performanse . . . . . 106
Dobre prakse . . . . . 108
Ukratko . . . . . 108

**5. KONTROLERI . . . . . 111**
Tipovi kontrolera . . . . . 111
Kreiranje kontrolera . . . . . 114

Kontroler jedne akcije ....114
Osnovni kontroler ....115
Kontroler resursa ....116
Konfiguracija kontrolera resursa ....119
Kontroler jedinstvenog resursa ....120
Kreiranje API kontrolera resursa ....123
API Singleton resursi ....125
Upotreba middleware u kontroleru ....126
Inline middleware ....128
Middleware i atributi ....129
Middleware i kontroler resursa ....130
Injektovanje zavisnosti u kontroleru ....133
Injektovanje putem konstruktora ....133
Promocija svojstava u konstruktoru ....134
Injektovanja kroz metode kontrolera ....135
Organizacija aplikacije ....137
Smernice za kreiranje efikasnih kontrolera ....138
Ukratko ....140

**6. BLADE SISTEM ....141**
Prikazivanje podataka ....143
Kontrolne strukture ....143
Uslovi ....143
Petlje ....144
Promenljiva $loop ....146
Switch naredba ....147
Uključivanje datoteka ....147
Dodatne strukture ....147
Ugnježdene kontrolne strukture ....148
Nasleđivanje osnovnog šablona ....148
Komponente ....151

Tipovi komponenti . . . . . 151
Klasične komponente . . . . . 152
Prosleđivanje atributa . . . . . 154
Filtriranje atributa . . . . . 157
Imenovani slotovi . . . . . 158
Dinamičko upravljanje klasama komponente . . . . . 158
Inline komponente . . . . . 160
Anonimne komponente . . . . . 161
Organizacija stranice pomoću komponenti . . . . . 163
Forme . . . . . 165
Stekovi . . . . . 167
View composer . . . . . 169
Injektovanje servisa . . . . . 172
Prilagođene Blade direktive . . . . . 173
Upravljanje greškama . . . . . 175
Dobre prakse . . . . . 180
Ukratko . . . . . 183

**7. BAZE PODATAKA . . . . . 185**
Konfiguracija baze podataka . . . . . 185
Migracije . . . . . 186
Dodavanje nove kolone u postojeću tabelu . . . . . 189
Lančano dodavanje atributa kolonama . . . . . 190
Modifikacija kolona . . . . . 191
Upravljanje indeksima kolona . . . . . 193
Rad sa stranim ključevima . . . . . 196
Popunjavanje baze podacima . . . . . 196
Graditelj upita . . . . . 198
Osnovna upotreba SQL upita . . . . . 200
Metode graditelja upita . . . . . 203
Sirovi SQL izrazi u graditelju upita . . . . . 207

Transakcije . . . . . 208
Artisan komande za pregled baze podataka . . . . . 210
Rukovanje greškama . . . . . 211
Hvatanje grešaka . . . . . 211
Prikazivanje grešaka . . . . . 211
Zaštita od SQL injekcionih napada . . . . . 212
Dobre prakse . . . . . 213
Ukratko . . . . . 215

**8. ELOQUENT ORM . . . . . 217**
Uvod u Eloquent . . . . . 217
Kreiranje modela . . . . . 218
Pristup podacima . . . . . 220
Dohvatanje podataka . . . . . 220
Unos, ažuriranje i brisanje podataka . . . . . 222
Unos podataka . . . . . 222
Ažuriranje podataka . . . . . 223
Brisanje podataka . . . . . 225
Eloquent ili graditelj upita . . . . . 228
Opsezi upita . . . . . 228
Globalni opsezi upita . . . . . 228
Lokalni opsezi upita . . . . . 231
Eloquent relacije . . . . . 233
Jedan prema jedan . . . . . 233
Jedan prema više . . . . . 236
Više prema više . . . . . 239
Polimorfne relacije . . . . . 242
Transformacija atributa modela . . . . . 246
Pristupnici . . . . . 246
Mutatori . . . . . 247
Konverzija atributa . . . . . 248

Eloquent kolekcije . . . . . 251
Serijalizacija . . . . . 255
Eloquent događaji . . . . . 257
Rukovanje greškama . . . . . 261
Bezbednosni izazovi i preporuke . . . . . 262
Ukratko . . . . . 263

**9. VALIDACIJA . . . . . 265**
Uvod . . . . . 265
Važnost validacije . . . . . 265
Osnovni pristupi validaciji . . . . . 266
Pravila validacije . . . . . 267
Validacija ugnježdenih podataka . . . . . 267
Najčešće korišćena pravila validacije . . . . . 268
Kreiranje prilagođenih pravila validacije . . . . . 269
Validacija pomoću metode validate() . . . . . 270
Ručna validacija pomoću fasade Validator . . . . . 270
Rad sa validiranim podacima . . . . . 271
Validacija u kontroleru . . . . . 274
Prilagođene poruke o greškama u kontroleru . . . . . 275
Validacija specifičnih polja . . . . . 276
Primena prilagođenih validacionih pravila u kontroleru . . . . . 276
Organizovanje validacije u metodama kontrolera . . . . . 277
FormRequest klasa za validaciju . . . . . 278
Kreiranje FormRequest klase . . . . . 278
Autorizacija zahteva . . . . . 279
Pravila validacije u FormRequest klasi . . . . . 279
Primena FormReques klasa u kontroleru . . . . . 280
PHP atributi za konfiguraciju FormRequest klasa . . . . . 281
Prilagođavanje poruka o greškama . . . . . 283
Primer upotrebe FormRequest klase . . . . . 284

Uslovna validacija . . . . . 285
Validacija datoteka . . . . . 287
Validacija nizova . . . . . 288
Validacija sa relacionim podacima . . . . . 289
Unapređenje performansi validacije . . . . . 290
Validacione greške u prikazima . . . . . 291
Definisanje i prilagođavanje poruka o greškama . . . . . 293
Prijava i registracija na istoj stranici . . . . . 295
Dobre prakse . . . . . 297
Ukratko . . . . . 298

**10. AUTENTIFIKACIJA I AUTORIZACIJA . . . . . 301**
Autentifikacija . . . . . 302
Ručna implementacija . . . . . 302
Izmena modela User i tabele users . . . . . 302
Kreiranje kontrolera . . . . . 304
Definisanje ruta . . . . . 307
Blade šablon i prikazi . . . . . 308
Prikaz forme za registraciju . . . . . 309
Prijava korisnika . . . . . 312
Komponenta za prikaz sadržaja u kartici . . . . . 314
Komponenta za privremene poruke . . . . . 314
Autorizacija . . . . . 315
Gate – funkcionalna autorizacija . . . . . 316
Policy klase – pravila pristupa po modelima . . . . . 320
Metode i mehanizmi autorizacije . . . . . 323
Najčešće metode Policy klasa . . . . . 325
Primer: Autorizacija sa Policy klasom . . . . . 325
Otklanjanje grešaka autorizacije . . . . . 332
Dobre prakse . . . . . 334
Ukratko . . . . . 335

**11. SERVISNE KLASE I SERVISNI KONTEJNER ..... 337**
Servisne klase ..... 337
Struktura i organizacija servisnih klasa ..... 339
Struktura direktorijuma za servisne klase ..... 339
Kreiranje i primena servisne klase ..... 339
Postepeni razvoj servisne klase ..... 343
Kada servisne klase nisu neophodne ..... 345
Servisni kontejner ..... 346
Osnovno korišćenje servisnog kontejnera ..... 347
Injektovanje zavisnosti u metode kontrolera ..... 348
Vezivanje servisa ..... 350
Struktura servisnog provajdera ..... 350
Vezivanje singletona ..... 350
Scoped vezivanje ..... 351
Vezivanje instance ..... 352
Kontekstualno vezivanje ..... 353
Razrešavanje servisa ..... 354
Proces razrešavanja servisa i zavisnosti ..... 355
Servisni kontejner u realnim scenarijima ..... 360
Primer upotrebe servisnog kontejnera ..... 363
Smernice za održavanje i skalabilnosti ..... 365
Ukratko ..... 366

**12. ARTISAN I TINKER ..... 369**
Uvod u Artisan ..... 369
Osnovne Artisan komande ..... 370
Opcije ..... 372
Pisanje prilagođenih komandi ..... 374
Interaktivni unos u Artisan komandama ..... 376
Registrovanje komandi ..... 377
Pozivanje komandi iz koda ..... 378
Komanda zasnovana na anonimnoj funkciji ..... 378

Testiranje Artisan komandi . . . . . 379
Opcije za upravljanje izlazom i interaktivnošću . . . . . 380
Prilagođavanje Artisan izlaza . . . . . 381
Tinker . . . . . 383
Napredna upotreba Tinkera . . . . . 385
Rizici Tinkera u produkciji . . . . . 386
Preporučeni pristup korišćenju Tinkera . . . . . 389
Prilagođavanje datoteka predložaka . . . . . 390
Ukratko . . . . . 392

**13. DOGAĐAJI I REDOVI ČEKANJA . . . . . 393**
Događaji i slušači . . . . . 393
Kreiranje događaja i slušača . . . . . 395
Registracija događaja i slušača . . . . . 395
Automatsko otkrivanje . . . . . 396
Ručna registracija . . . . . 397
Slušači zasnovani na anonimnoj funkciji . . . . . 397
Definicija događaja . . . . . 398
Definicija slušača . . . . . 399
Emitovanje događaja . . . . . 400
Redovi čekanja . . . . . 405
Integracija događaja i redova čekanja . . . . . 409
Slušači u redu čekanja . . . . . 410
Anonimni slušači događaja u redovima čekanja . . . . . 411
Podešavanje veze, naziva i kašnjenja reda . . . . . 411
Slušači koji uslovno čekaju u redu . . . . . 412
Ručna interakcija sa redom . . . . . 413
Slušači reda čekanja i transakcije baze podataka . . . . . 414
Rukovanje neuspelim zadacima . . . . . 415
Kontrola izvršavanja zadataka u redovima čekanja . . . . . 415
Otklanjanje grešaka događaja . . . . . 419
Dobre prakse . . . . . 420

Ukratko . . . . . 421

**14. IZGRADNJA REST API-JA . . . . . 423**
Uvod . . . . . 423
API kontroler i JSON odgovori . . . . . 425
Organizacija kontrolera . . . . . 425
Struktura kontrolera . . . . . 426
Standardizacija JSON odgovora . . . . . 428
REST API rute . . . . . 429
Verzionisanje REST API-ja . . . . . 430
Rad sa CORS zahtevima . . . . . 431
Rad sa HTTP zaglavljima . . . . . 433
Čitanje zaglavlja . . . . . 433
Slanje zaglavlja . . . . . 434
Podešavanje kontrole keširanja zaglavlja . . . . . 434
Rad sa većim skupovima podataka . . . . . 435
Osnovna paginacija . . . . . 435
JSON struktura odgovora . . . . . 436
Dinamički broj zapisa po strani . . . . . 436
Jednostavna paginacija . . . . . 437
Prilagođena struktura JSON odgovora . . . . . 437
Filtriranje . . . . . 438
Sortiranje . . . . . 440
Transformacija rezultata . . . . . 441
Autentifikacija pomoću Sanctuma . . . . . 445
Prilagođavanje 404 odgovora . . . . . 451
Bezbednosni aspekti REST API-ja . . . . . 454
Ukratko . . . . . 455

**15. OSNOVE TESTIRANJA SA PESTOM . . . . . 457**
Uvod . . . . . 457
Instalacija Pesta . . . . . 458

Osnovna struktura testova ....460
Tipovi testova ....461
Jedinični testovi ....461
Funkcionalni testovi ....462
Definisanje testova i očekivanja ....463
Definisanje testova ....463
Provera rezultata u Pestu ....464
Priprema i čišćenje testnog okruženja ....465
Testiranje skupova podataka ....467
Preskakanje testova ....467
Označavanje nedovršenih testova ....468
Testiranje HTTP odgovora ....469
Provera baze podataka ....471
Grupisanje testova ....473
Napredne tehnike testiranja ....474
Mock i spy objekti ....474
Fabrike modela i popunjavanje podataka ....477
Zavisnosti između testova ....480
Otklanjanje grešaka ....481
Dobre prakse ....484
Ukratko ....485

**16. CRUD SISTEM ZA UPRAVLJANJE DOBAVLJAČIMA I NJIHOVIM RAČUNIMA ....487**
Instalacija Laravela ....490
Konfiguracija baze podataka ....490
Definisanje migracija ....490
Implementacija modela i relacija ....493
Validacija korisničkog unosa ....496
Kreiranje kontrolera ....501
Definisanje ruta ....506

Kreiranje probnih podataka . . . . . 506
Blade komponente i prikazi . . . . . 509
Komponente i njihova struktura . . . . . 509
Komponente korišćene u aplikaciji . . . . . 509
Struktura i organizacija prikaza . . . . . 516
Filtriranje i pretraga dobavljača . . . . . 524
Dodavanje i brisanje računa . . . . . 527
Dinamičko upravljanje računima na klijentskoj strani . . . . . 527
Testiranje sistema . . . . . 535
Instalacija Pesta . . . . . 535
Struktura testova . . . . . 536
Funkcionalni testovi . . . . . 536
Supplier CRUD – osnove (prikaz, kreiranje, brisanje) . . . . . 536
Validacija uključujući unique pravila . . . . . 538
AJAX brisanje računa . . . . . 540
Jedinični test . . . . . 540
Pokretanje testova . . . . . 542
Bezbednosni aspekti rada sa podacima . . . . . 542
Ukratko . . . . . 544

**17. APLIKACIJA ZA E-TRGOVINU . . . . . 547**
Uvod . . . . . 547
Instalacija aplikacije . . . . . 548
Podešavanje okruženja . . . . . 548
Specifikacija funkcionalnosti . . . . . 549
Migracije . . . . . 550
Modeli i relacije . . . . . 553
Generisanje i inicijalno popunjavanje baze podataka . . . . . 557
Autentifikacija . . . . . 561
Proširenje tabele users . . . . . 561
Kontroler za autentifikaciju . . . . . 562

Blade datoteke za prijavu i registraciju .....563
Kreiranje servisnih klasa .....567
Kreiranje kontrolera .....570
Kreiranje Blade šablona i prikaza .....575
Ažuriranje količine i cene .....590
Admin panel .....591
Zaštita administratorskih ruta .....597
Organizacija ruta u aplikaciji .....598
Pokretanje aplikacije .....600
Testiranje .....600
Podešavanje test okruženja .....600
Testiranje registracije korisnika .....602
Testiranje kreiranja porudžbine iz korpe .....602
Testiranje filtriranje proizvoda po nazivu .....604
Testiranje zabrane pristupa administratorskom panelu .....605
Zaključak .....606
**18. INDEX .....609**
**19. LITERATURA .....619**



Prvo poglavlje

# 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.

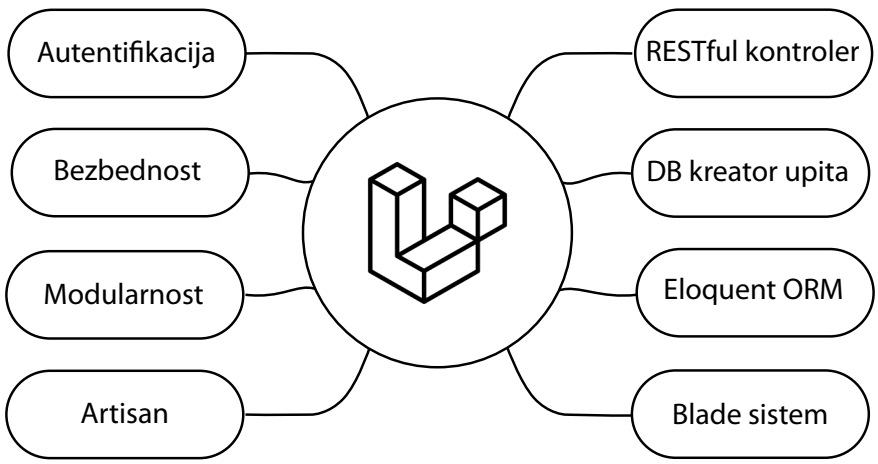
Priručnik je usklađen sa verzijom 13 Laravel razvojnog okvira i obuhvata njegove ključne funkcionalnosti.

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 nalozima 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 lozinki, 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.

**Sistemski zahtevi:**

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** `Composera`

`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 razvojnom 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 `Composera`. Laravel je ovde smešten.

## *Ključne datoteke*

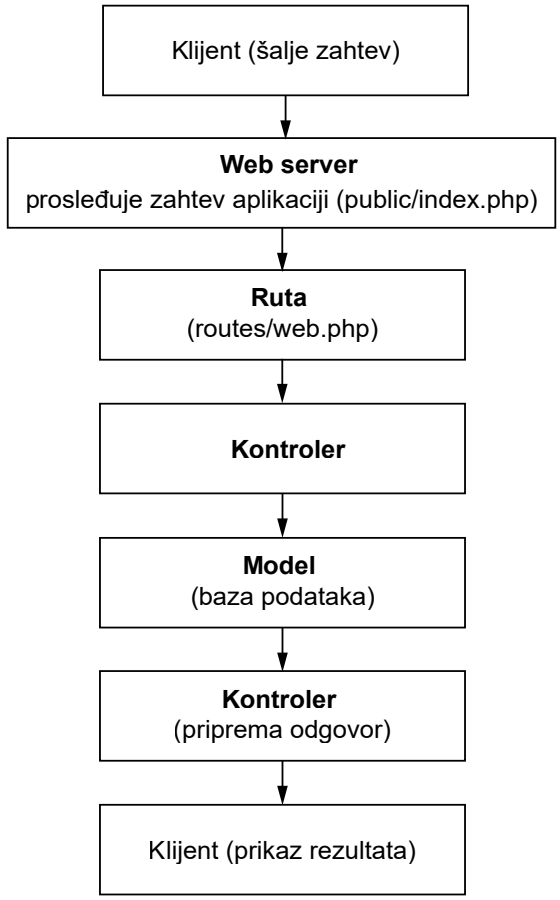
| **Naziv datoteke** | **Funkcija** | **Napomena** |
|---|---|---|
| `.env` | Glavna konfiguracija aplikacije (baza, API ključevi, servisi) | Ne čuvati u Git repozitorijumu, podesiti lokalno ili preko sistema za distribuciju |
| `artisan` | CLI alat za Laravel zadatke (generisanje kontrolera, migracija, testiranje) | Središnji alat za razvoj i administraciju |
| `composer.json` | Definiše PHP zavisnosti i Laravel pakete | Nakon izmene koristiti `composer install` ili `composer update` |
| `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 kompajliranje frontend resursa | Definiše ulazne tačke, HMR, integraciju sa Tailwind/Vue/React |
| `server.php` | Alternativni development server | Obično se koristi samo u specifičnim 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 definicijom 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 obrađuju 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 razvojnom iskustvu jeste proširena podrška za **PHP atribute**. Ovakav pristup omogućava definisanje konfiguracije neposredno iznad klase ili metode, čime se smanjuje potreba za dodatnim javnim svojstvima i poboljšava preglednost koda.

Umesto ranijeg pristupa:

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

sada je moguće koristiti atribute:

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

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

Podrška za atribute 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.