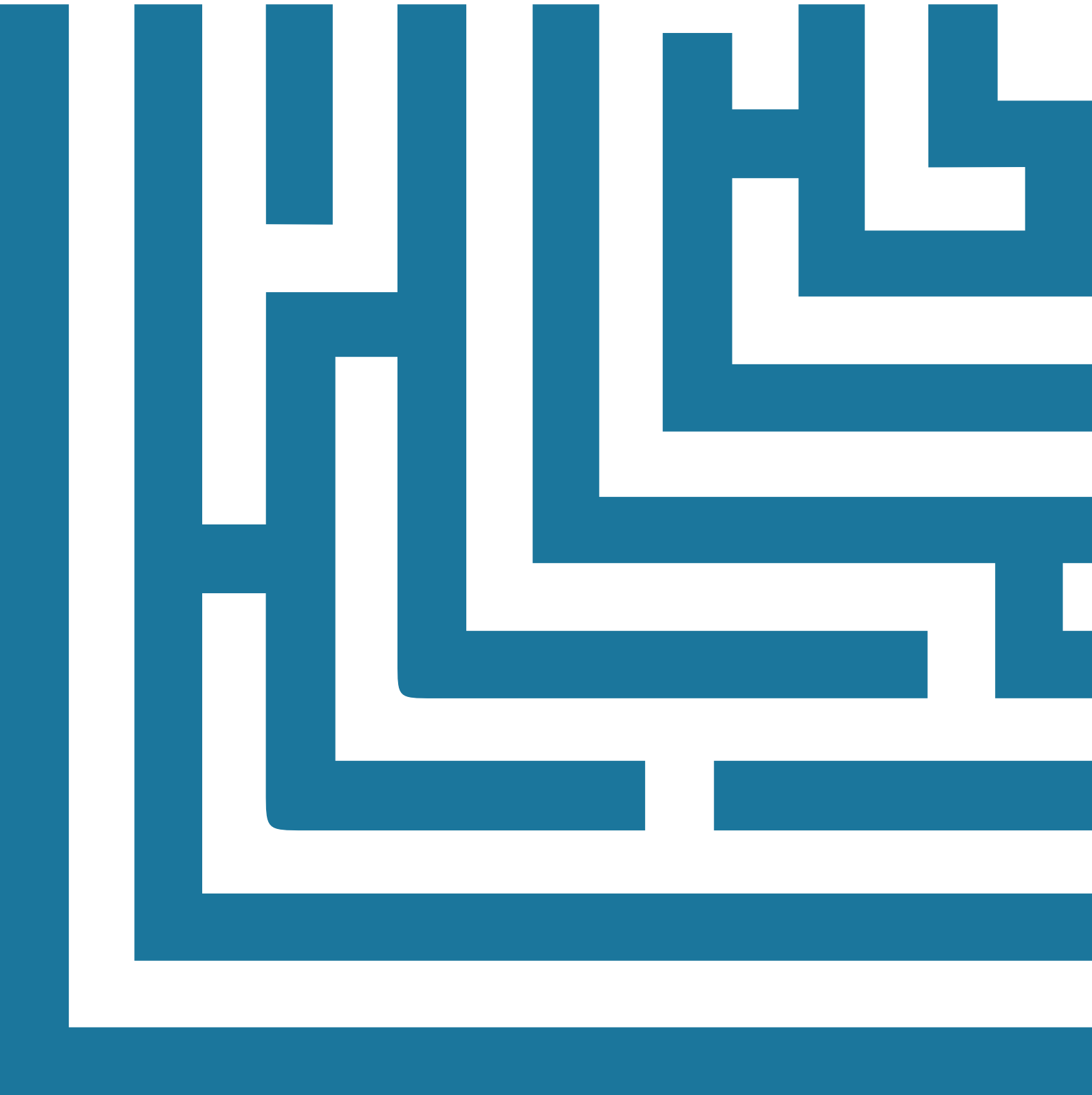
Univerzitetna založba
Univerze v Mariboru

Uvod v WordPress

Saša Brdnik, Tjaša Heričko, Špela Čučko, Sašo Karakatič



Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Uvod v WordPress

Avtorji

dr. Saša Brdnik,

dr. Tjaša Heričko,

Špela Čučko, mag. inž. inf. in teh. kom.,

dr. Sašo Karakatič

26. januar 2026

Naslov *Title*	Uvod v WordPress *Introduction to WordPress*
Avtorij *Authors*	Saša Brdnik (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko) Tjaša Heričko (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko) Špela Čučko (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko) Sašo Karakatič (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko)
Recenzija *Review*	Dejan Lavbič (Univerza v Ljubljani, Fakulteta za računalništvo in informatiko) Niko Lukač (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko)
Jezikovni pregled *Language editing*	Špela Vidmar
Tehnična urednika *Technical editors*	Saša Brdnik (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko) Jan Perša (Univerza v Mariboru, Univerzitetna založba)
Oblikovanje ovitka *Cover designer*	Saša Brdnik (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko)
Grafika na ovitku *Cover graphics*	Adobe Stock
Grafične priloge *Graphics material*	Vsi viri so lastni, razen če ni navedeno drugače. Brdnik, Heričko, Čučko, Karakatič (avtorji), 2026
Založnik *Published by*	Univerza v Mariboru Univerzitetna založba Slomškov trg 15, 2000 Maribor, Slovenija https://press.um.si, zalozba@um.si
Izdajatelj *Issued by*	Univerza v Mariboru Fakulteta za elektrotehniko, računalništvo in informatiko Koroška cesta 46, 2000 Maribor, Slovenija https://feri.um.si, feri@um.si
Izdaja *Edition*	Prva izdaja
Izdano *Published at*	Maribor, Slovenija, januar 2026
Vrsta publikacije *Publication type*	E-knjiga
Dostopno na *Available at*	https://press.um.si/index.php/ump/catalog/book/1074

Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije

Izdajo učbenika sofinancira Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije preko programske skupine P2-0057.

CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

004.9(035)(0.034.2)

UVOD v WordPress [Elektronski vir] / avtorji Saša Brdnik ... [et al.]. - 1. izd. - E-knjiga. - Maribor : Univerza v Mariboru, Univerzitetna založba, 2026

Način dostopa (URL): https://press.um.si/index.php/ump/catalog/book/1074

ISBN 978-961-299-106-7 (PDF)
doi: 10.18690/um.feri.3.2026
COBISS.SI-ID 265762051

ISBN	**978-961-299-106-7 (pdf)**
DOI	**https://doi.org/10.18690/um.feri.3.2026**
Cena *Price*	**Brezplačni izvod**
Odgovorna oseba založnika *For publisher*	**Prof. dr. Zdravko Kačič, rektor Univerze v Mariboru**
Citiranje *Attribution*	**Brdnik, S., Heričko, T., Čučko, Š., Karakatič, S. (2026). *Uvod v WordPress.* Univerza v Mariboru, Univerzitetna založba. doi: 10.18690/um.feri.3.2026**

Kazalo

Kazalo slik

Predgovor

Učno gradivo *Uvod v WordPress* je zasnovano tako, da bralca postopno vodi skozi razumevanje osnov in uporabo sistema za upravljanje vsebin WordPress ter ga uvaja v razvoj spletnih strani, ki temeljijo na WordPressu. Vsako poglavje obravnava eno ključno temo, ki jo najprej predstavimo teoretično, nato pa podkrepimo s praktičnimi primeri. Pri tem bralcu predstavimo preproste primere za ponazoritev osnovnih konceptov, nato pa še zahtevnejše primere, ki prikazujejo naprednejše možnosti uporabe. Namen takšne organizacije učnega gradiva je zagotoviti uvod v razvoj spletnih strani s sistemom WordPress za začetnike ter hkrati služiti kot priročnik za podporo pri njegovi naprednejši uporabi.

Teoretično pomembni koncepti in razlike med termini so posebej izpostavljeni, kar bralcu olajša hitro prepoznavanje ključnih informacij v besedilu, in sicer na naslednji način:

Kaj je WordPress? WordPress je najpopularnejši sistem za upravljanje vsebin, ki poganja več kot 43 % vseh spletnih mest. Nastal je leta 2003 kot preprosta blogerska platforma, danes pa se uporablja za najrazličnejše namene – od preprostih predstavitvenih spletnih strani do velikih medijskih portalov in spletnih trgovin. Njegova moč je v odprtokodni naravi, veliki skupnosti uporabnikov in razvijalcev ter modularni zasnovi, ki med drugim omogoča prilagoditve s temami in vtičniki.

Pri delu z WordPressom je treba upoštevati številne posebnosti in praktične nasvete, ki so izpostavljeni na naslednji način:

Učno gradivo temelji na sistemu WordPress različice 6.8.3 (september 2025). Čeprav se WordPress nenehno razvija, je gradivo zasnovano na temeljnih konceptih, ki jih je po večini mogoče brez večjih težav preizkusiti tudi na starejših ali novejših različicah. Če ob delu naletiš na odstopanja, to izkoristi kot priložnost za praktično raziskovanje in spoznavanje, kako se WordPress prilagaja sodobnim potrebam.

Primeri so izpostavljeni v sivih blokih, ki ponujajo jasen in pregleden vpogled v izseke programske kode skriptnega jezika PHP, slogovne predloge CSS, zapise HTML, nastavitvene datoteke, izvajanje ukazov v ukazni vrstici ter strukturo direktorijev na naslednji način:

Primer 0.0.1 (Skripta PHP s preprostim izpisom.)

```
<?php
  echo 'Uvod v WordPress';
?>
```

Uvod

To učno gradivo je namenjeno vsem, ki se želijo seznaniti z osnovami razvoja spletnih strani z uporabo sistema za upravljanje vsebin WordPress. Nastal je kot zbirka znanj, izkušenj in praktičnih napotkov, ki smo jih pripravili izvajalci predmeta *Praktikum: spletni sistemi in vsebine* na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Namenjen je študentom, začetnikom in vsem, ki se želijo na strukturiran način naučiti vzpostavitve in prilagoditve lastne spletne strani. WordPress je najvidnejši predstavnik sistemov za upravljanje vsebin, znan po svoji odprtokodnosti, prilagodljivosti in razširljivosti. Zaradi nizke vstopne ovire, široke podpore skupnosti ter bogatega ekosistema tem in vtičnikov je še posebej primeren za učenje sodobnih pristopov k razvoju spletnih sistemov in vsebin. Priročnik je razdeljen na več sklopov, ki obravnavajo predstavitev WordPressa, namestitev WordPressa, osnovno delo z WordPressom, pripravo in prilagoditev vsebin za objavo, datotečno strukturo v WordPressu, vizualno urejanje in prilagajanje s temami, vtičniki in dogodki ter uporabo WordPressa kot brezglavega sistema za upravljanje vsebin. Poudarek je na praktičnem pristopu in postopnem nadgrajevanju znanja, s čimer želimo bralcem omogočiti samostojno ustvarjanje, razumevanje, razvoj in upravljanje spletnih strani v okolju WordPress.

POGLAVJE 1

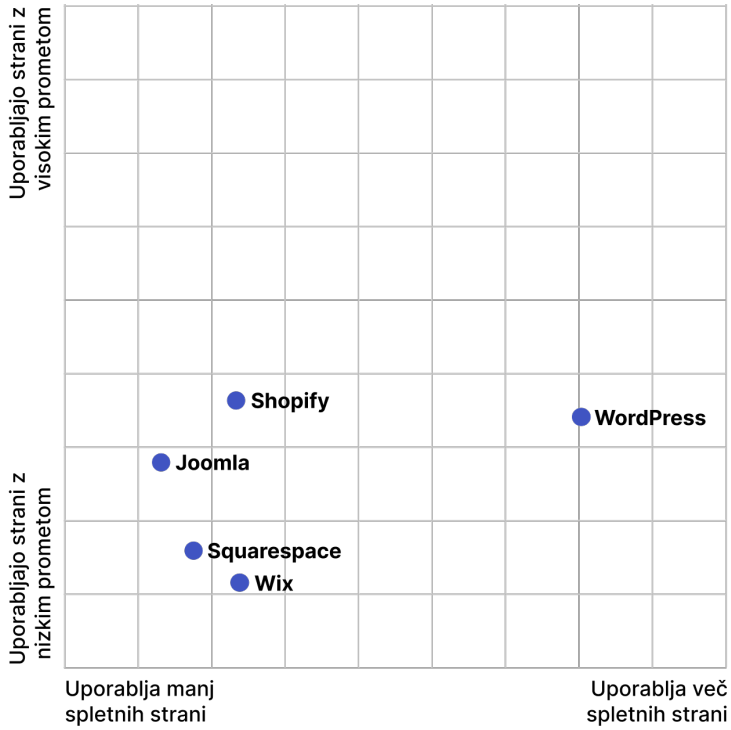
Kaj je WordPress?

WordPress je odprtokodna spletna programska oprema, ki je bila prvotno razvita kot orodje za pisanje blogov, vendar se je sčasoma razvila v sistem za upravljanje vsebin (angl. Content Management System ali CMS), ki lahko poganja spletna mesta, omrežja in skupnosti. Začetki WordPressa segajo v leto 2003, ko sta Matt Mullenweg in Mike Little ustvarila svojo vejitev *b2/cafelog* [1]. WordPress je licenciran pod Splošno javno licenco GNU (angl. GNU General Public License) [2], kar pomeni, da je njegovo izvorno kodo pod določenimi pogoji dovoljeno prosto spreminjati in razširjati. Kot odprtokodna platforma omogoča svobodo uporabe in ustvarja močno skupnost, ki aktivno prispeva k njenemu razvoju. WordPress razvijalcem omogoča hitro vzpostavitev različnih vrst spletnih mest, od osebnih blogov in predstavitvenih strani podjetij do kompleksnih aplikacij. Njegove prednosti vključujejo enostavno objavljanje različnih vsebin, upravljanje uporabnikov, podporo za večjezičnost, optimizacijo za iskalnike ter veliko zbirko tem in vtičnikov. Razvijalcem ponuja obsežen aplikacijski programski vmesnik (angl. Application Programming Interface ali API), podporo za lastne tipe prispevkov (angl. Cutom Post Type ali CPT) in možnost popolne prilagoditve [3].

WordPress.org ≠ WordPress.com Domeno *WordPress.com* upravlja podjetje Automattic Inc., ki ponuja gostovanje in upravljanje spletnih strani z uporabo WordPressa kot storitve. Osnovni paket njihovega gostovanja je brezplačen, dodatne funkcionalnosti pa so na voljo v plačljivih paketih. Gre za t. i. »gostovani WordPress«, kjer sta upravljanje s hitrostjo spletnega mesta in varnostjo ter vzdrževanje večinoma avtomatizirana. Nasprotno je domena *WordPress!WordPress.org* namenjena dejanski programski opremi WordPress, kjer je na voljo tudi vsa dokumentacija, in omogoča prenos odprtokodne različice WordPressa, ki jo nato razvijalec sam namesti na izbrani strežnik. Gre za t.

i. »samogostovani WordPress«, kjer mora razvijalec sam poskrbeti za gostovanje, varnost in vzdrževanje, v zameno pa ima večjo svobodo in nadzor nad spletnim mestom.

Danes je WordPress eden najpogosteje uporabljenih sistemov za razvoj in upravljanje spletnih strani. Po podatkih W3Techs poganja več kot 43,4 % vseh spletnih mest, kar ga uvršča daleč pred druge sisteme za upravljanje vsebin. Več kot 60,8 % spletnih mest, za katera je poznan uporabljen sistem za upravljanje spletnih strani, uporablja WordPress. Za primerjavo sistem Shopify, drugi najpopularnejši sistem za upravljanje vsebin, uporablja zgolj 6,7 % spletnih mest [4]. Primerjava tržnega deleža WordPressa v primerjavi z drugimi sistemi je prikazana na sliki 1.1. Njegova priljubljenost temelji na enostavnosti uporabe, razširljivosti prek tem in vtičnikov ter močni skupnosti razvijalcev in uporabnikov, ki prispevajo k neprestanemu razvoju sistema. Poleg posameznikov WordPress uporabljajo tudi številna podjetja, izobraževalne ustanove in medijske hiše, kar dokazuje njegovo vsestranskost in zanesljivost.



Slika 1.1. Tržni delež WordPressa v primerjavi z drugim sistemom za upravljanje vsebin (september 2025, povzeto po [4]).

WordPress temelji na skriptnem programskem jeziku PHP, ki se izvaja na strežniški strani in omogoča dinamično generiranje spletnih vsebin. Kot privzeti sistem za upravljanje s podatkovnimi bazami (angl. Database Management System ali DBMS) WordPress uporablja podatkovno bazo MySQL oziroma njeno odprtokodno različico MariaDB [5]. Vsi podatki o vsebinah, uporabnikih, nastavitvah in strukturi spletnega mesta so shranjeni v tej podatkovni bazi, medtem ko skripte PHP poskrbijo za njihovo pridobivanje, obdelavo in prikaz uporabnikom na spletnem uporabniškem vmesniku.

POGLAVJE 2

Namestitev sistema WordPress

WordPress je znan po svoji enostavni namestitvi – zaradi svoje enostavnosti je namestitev promovirana kot opravilo, ki traja manj kot pet minut. Zaradi popularnosti WordPressa številni ponudniki spletnega gostovanja ponujajo orodja za samodejno namestitev WordPressa v nekaj klikih. V tem učnem gradivu bo v ospredju vzpostavitev lokalnega razvojnega okolja z orodjem Local in vsebniki Docker ter namestitev WordPressa pri ponudniku gostovanja.

Priporočena praksa pri razvoju spletnih mest vključuje vzpostavitev lokalnega razvojnega okolja, kjer se izvajata razvoj in testiranje. Šele po zaključku razvoja spletno mesto nato namestimo na produkcijski strežnik in z ustrezno konfiguracijo povežemo s končno domeno, kar zagotavlja bolj nadzorovano uvedbo ter zmanjšuje tveganje za napake v produkcijskem okolju.

2.1 Lokalno razvojno okolje s sistemom WordPress

Vzpostavitev lokalnega razvojnega okolja s sistemom WordPress omogoča varen razvoj, prilagajanje in testiranje spletnega mesta pred njegovo namestitvijo na produkcijskem strežniku, pri čemer so na voljo tako specializirana orodja za lokalno namestitev WordPressa, kot je Local, kot tudi splošne rešitve za vzpostavitev lokalnega strežniškega okolja, kot so XAMPP, WAMP, MAMP in LAMP. V preteklosti je vzpostavitev razvojnega okolja temeljila predvsem na splošnih rešitvah, ki pa jih v praksi v zadnjih letih zamenjujejo specializirana orodja. Kljub temu jih v tem učnem gradivu omenjamo, saj je na spletu zaradi pretekle razširjene uporabe na voljo veliko virov in vodičev o tem.

Namesto tega je danes za naprednejše razvijalce priporočljiva namestitev WordPressa s pomočjo orodja Docker. Vzpostavitev lokalnega razvojnega okolja z navedenimi orodji se razlikuje glede na kompleksnost namestitve, prilagodljivost okolja, uporabniško izkušnjo, čas vzpostavitve, namestitev v produkcijsko okolje in platformsko združljivost, kar prikazuje primerjava v preglednici 2.1. V nadaljevanju tega učnega gradiva bo predstavljena lokalna namestitev WordPressa s programsko opremo Local in z orodjem Docker.

Preglednica 2.1. Primerjava pristopov k vzpostavitvi lokalnega razvojnega okolja s sistemom WordPress.

Značilnost	XAMPP / WAMP / MAMP / LAMP	Local	Docker
Postopek namestitve	Ročna namestitev in nastavitev PHP, Apache in MySQL; ročna namestitev WordPressa in nastavitev WordPressa (z namestitvenim čarovnikom v brskalniku)	Namestitev z nekaj kliki; avtomatska konfiguracija okolja z možnostjo nastavitev z uporabniškim vmesnikom	Konfiguracija in namestitev z Docker Compose in ukazno vrstico; ročna nastavitev WordPressa (z namestitvenim čarovnikom v brskalniku)
Nadzor nad okoljem	Visoka prilagodljivost (različne verzije PHP, ročna konfiguracija)	Omejena prilagoditev, vendar zadostuje za večino potreb	Zelo visoka prilagodljivost (poljubne slike in konfiguracije); možno reproducirati identično okolje na različnih napravah
Uporabniška izkušnja	Namenjeno tehnično bolj podkovanim uporabnikom	Intuitiven uporabniški vmesnik, primeren tudi za začetnike	Namenjeno tehnično zelo podkovanim uporabnikom
Čas vzpostavitve	Daljši; zahteva več korakov	Hiter; avtomatiziran postopek	Daljši; relativno hiter po prvi nastavitvi, saj je možno ponovno uporabiti konfiguracije

Nadaljevanje na naslednji strani

Nadaljevanje preglednice 2.1

Značilnost	XAMPP / WAMP / MAMP / LAMP	Local	Docker
Namestitev v produkcijsko okolje	Ročna migracija; pogosto zahteva veliko časa z veliko izzivi	Ročna migracija, vendar obstaja vgrajena podpora za migracijo za določene ponudnike (v plačljivi različici orodja)	Ročna ali avtomatizirana migracija; pogosto poteka hitro in brez večjih izzivov
Platformska združljivost	Na voljo za Windows, macOS, Linux (odvisno od paketa)	Na voljo za Windows in macOS	Na voljo za Windows, macOS in Linux (potreben Docker Engine/Docker Desktop)

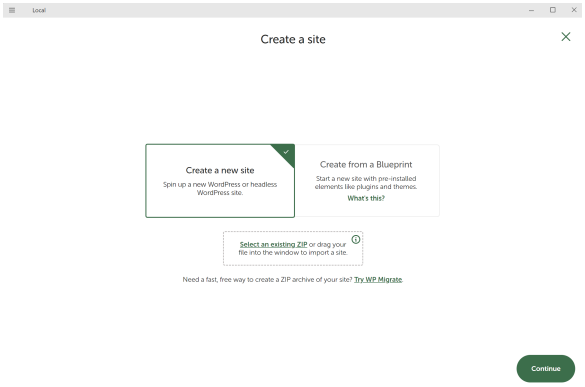
Za začetnike, ki se prvič srečujejo z razvojem spletnih strani, priporočamo lokalno vzpostavitev razvojnega okolja s sistemom WordPress s pomočjo orodja Local. Za naprednejše bralce priporočamo vzpostavitev z vsebniki Docker.

Za delovanje trenutne različice WordPressa (januar 2026) so priporočljive naslednje strežniške zahteve: PHP različice 8.3 ali novejše, MySQL različice 8.0 ali novejše oziroma MariaDB različice 10.6 ali novejše ter podpora za HTTPS. Priporočeni spletni strežniki so Apache (s podprtimi moduli, npr. mod_rewrite) ali Nginx, vendar WordPress lahko deluje tudi na drugih strežnikih, ki omogočajo izvajanje kode PHP in povezavo s podatkovno bazo MySQL/MariaDB. Za okolja, kjer novejše različice še niso na voljo, WordPress omogoča delovanje tudi s PHP različico 7.2.24 ali novejšo in MySQL različico 5.5.5 ali novejšo, vendar so te že izven uradne podpore in lahko predstavljajo varnostno tveganje [5].

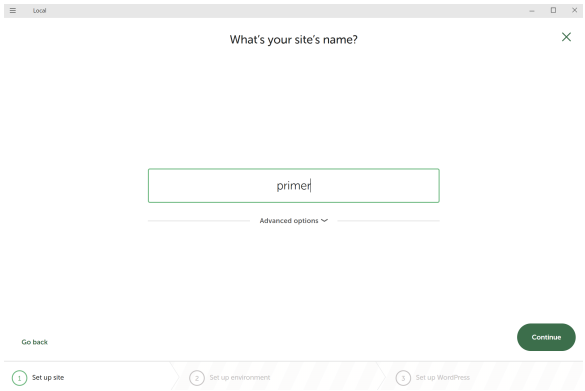
2.1.1 Vzpostavitev lokalnega razvojnega okolja z orodjem Local

Za vzpostavitev lokalnega okolja za razvoj in upravljanje spletnih mest z WordPressom z orodjem Local je najprej treba prenesti programsko opremo Local, ki je brezplačno na voljo za operacijska sistema Windows in macOS na https://localwp.com/. Koraki ustvarjanja novega projekta so prikazani na slikah 2.1 (a-d), in sicer z uporabo orodja

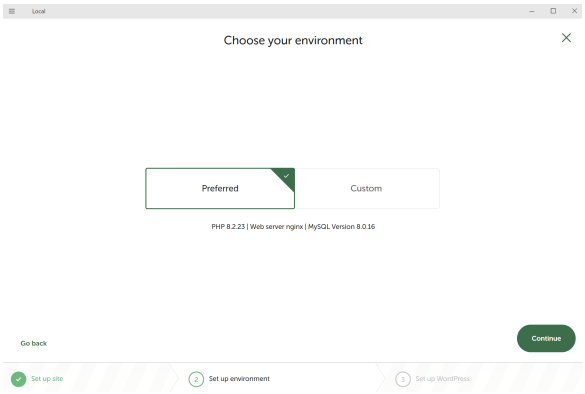
Local različice 9.2.8. Po izvedbi namestitve po navodilih je ob zagonu programa omogočeno ustvarjanje novega spletnega mesta z izbiro možnosti Create a new site (slika 2.1a). V nadaljevanju je treba vnesti ime novega projekta oziroma spletnega mesta (slika 2.1b). Sledi konfiguracija nastavitev okolja (PHP, strežnik, podatkovna baza), pri čemer za večino projektov zadostuje izbira privzetih strežniških nastavitev (slika 2.1c). Sledi še določanje prijavnih podatkov (uporabniško ime in geslo) za privzetega uporabnika s skrbniškimi pravicami za dostop do nadzorne plošče WordPressa (slika 2.1d). Slednje potrebujemo za dostop, zato si jih je treba dobro zapomniti ali zabeležiti.



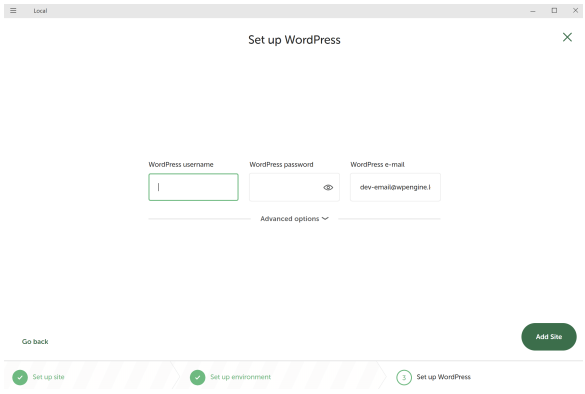
a) Prenos in namestitev orodja Local.



b) Poimenovanje spletnega mesta.



c) Konfiguracija okolja.



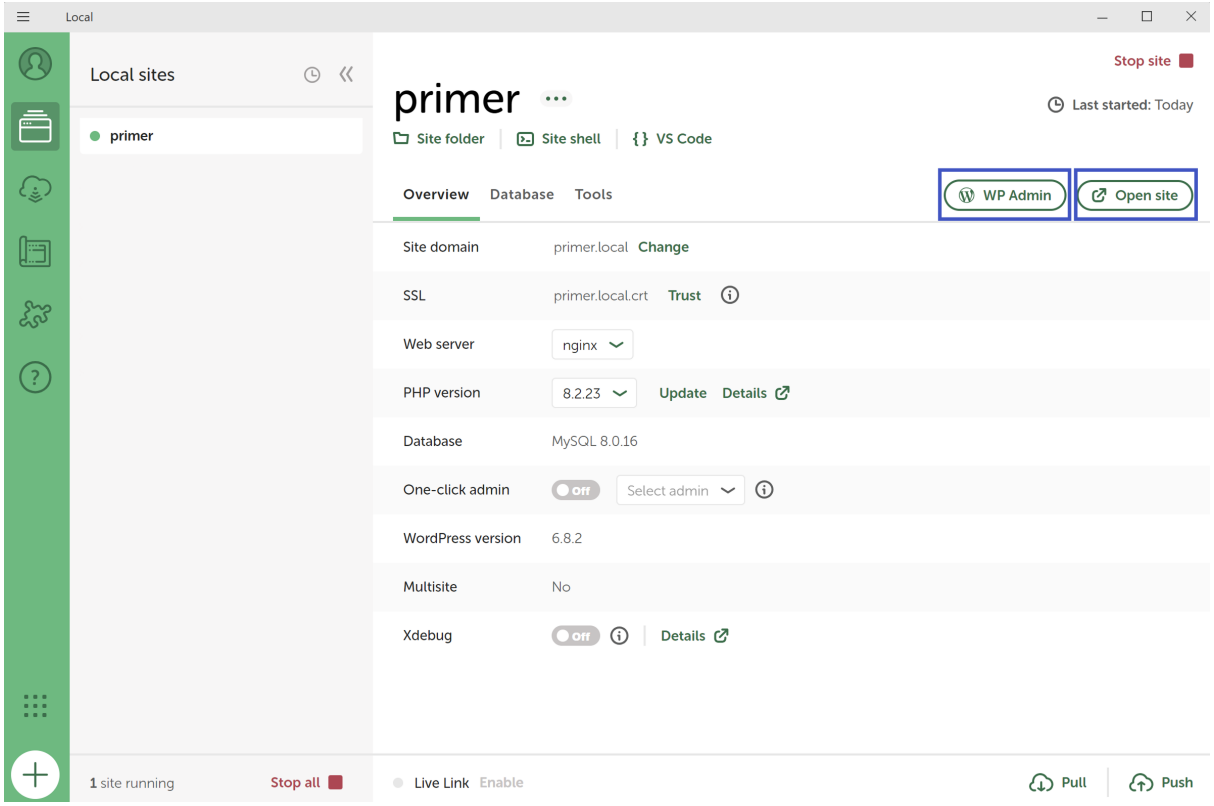
d) Ustvarjanje skrbnika.

Slika 2.1. Lokalna vzpostavitev spletnega mesta, ki temelji na sistemu WordPress, z orodjem Local.

Zagon in dostopanje do strani

Po uspešni vzpostavitvi okolja, ki je prikazana na sliki 2.2, je projekt dodan na seznam lokalnih strani v levem stranskem meniju. Do lokalno vzpostavljenega spletnega mesta je mogoče dostopati z naslednjima dvema gumboma uporabniškega vmesnika orodja:

- Gumb WP Admin, ki preusmeri na prijavno stran nadzorne plošče WordPress, v katero se je mogoče prijaviti s prej določenimi vpisnimi podatki skrbnika.
- Gumb Open site, ki preusmeri na ogled spletne strani kot obiskovalec, tj. končni uporabnik, spletne strani (odpre vstopno stran spletnega mesta – `index.php`).



Slika 2.2. Vzpostavljeno lokalno spletno mesto z uporabo orodja Local.

Gumba Start site in Stop site v zgornjem desnem kotu sta namenjena upravljanju zagnanih okolij.

- Start site zažene vse potrebno za delovanje izbrane WordPress strani: spletni strežnik (Apache/Nginx), interpreter PHP in podatkovno bazo (MySQL/MariaDB). Ko je stran zagnana, lahko do nje lokalno dostopamo v brskalniku preko povezave URL, na primer `primer.local` ali `localhost:9000`, odvisno od nastavitev.

- Stop site ustavi delovanje teh procesov. S tem se strežnik in baza ugasneta, stran pa v brskalniku lokalno ni več dostopna.

- Če izbrana spletna stran po kliku na gumb WP Admin / Open site še ni zagnana, jo orodje Local zažene samodejno.
- Po zagonu strani je do nadzorne plošče WordPress privzeto mogoče dostopati tudi neposredno preko povezave URL `/wp-admin`.
- Ko je stran v orodju Local zagnana, lahko alternativno dostopamo do začetne strani in nadzorne plošče tudi v levem meniju, z desnim klikom na ime strani in izborom Open site ali Admin dashboard.

Datotečni sistem WordPress

Do datotek WordPress spletne strani, vzpostavljene z orodjem Local, lahko dostopamo tako, da v orodju z desnim klikom na ime strani izberemo možnost Site folder. Enak rezultat dosežemo tudi s klikom na ime strani, kjer se gumb Site folder nahaja neposredno pod imenom. V odprti datotečni strukturi se v direktoriju `app/public` nahajajo vse datoteke WordPressa.

Za razvoj in konfiguracijo je osrednjega pomena direktorij `wp-content`, ki vsebuje poddirektorije `themes` (nameščene in razvite teme), `plugins` (nameščeni in razviti vtičniki) ter `uploads` (naložene datoteke, kot so slike in dokumenti PDF). Prav ta direktorij združuje vse ključne komponente, ki omogočajo prilagajanje in nadgradnjo WordPress spletne strani.

Podatkovna baza

Orodje Local omogoča preprost dostop do podatkovne baze z vgrajenim spletnim orodjem Adminer. Ko odpremo posamezno stran in kliknemo na zavihek Database, se prikažejo vsi potrebni podatki za povezavo. Od tam lahko z enim klikom na gumb Open Adminer dostopamo do zbirke podatkovne baze izbranega spletnega mesta neposredno v brskalniku.

Alternativno lahko do podatkovne zbirke dostopamo tudi z drugimi orodji, kot so phpMyAdmin, DBeaver, HeidiSQL ali MySQL Workbenc. V tem primeru je treba uporabiti podatke za ustvarjanje povezave, ki jih Local prikaže v zavihku Database (ime zbirke, uporabniško ime, geslo, gostitelj in vrata).

2.1.2 Vzpostavitev lokalnega razvojnega okolja z vsebniki Docker

V sodobnem razvoju je zaradi uporabe ponovljivega in izoliranega okolja za razvoj ter poznejšo namestitev v produkcijsko okolje vse pogostejša uporaba orodja Docker. Docker omogoča opis celotnega zagonskega okolja v slikah (angl. Images), ki predstavljajo nespremenljive predloge datotečnega sistema s pripadajočo konfiguracijo. Na podlagi takšne slike nato zaženemo vsebnik (angl. Container), tj. izvajalni primerek slike, ki teče v izolaciji od gostiteljskega sistema in drugih vsebnikov.

Predpogoj za vzpostavitev lokalnega razvojnega okolja WordPress z vsebniki Docker je nameščeno orodje Docker Desktop oziroma Docker Engine (vključno z Docker Compose v2, ki bo omogočil zagon in upravljanje več vsebnikov hkrati). Pri tem je treba z uporabo ustreznih datotek Dockerfile izgraditi lastne slike ali pa uporabiti obstoječe slike, ki jih lahko najdemo v registru Docker, kot je Docker Hub.

Zaradi razširjenosti zagonskih okolij, ki jih potrebujemo za vzpostavitev lokalnega razvojnega okolja za delo z WordPressom, najpogosteje uporabljamo obstoječe slike, kot so uradna Docker slika za WordPress (https://hub.docker.com/_/wordpress), uradna Docker slika za MySQL (http://hub.docker.com/_/mysql) in uradna Docker slika za phpMyAdmin (https://hub.docker.com/_/phpmyadmin).

Za vzpostavitev okolja si je najprej treba pripraviti ustrezno datotečno strukturo projekta, kot je prikazano na primeru 2.1.1. Datoteka `compose.yaml` služi kot nastavitvena datoteka, medtem ko bo direktorij `wp-content` služil kot trajna shramba v lokalnem datotečnem sistemu za sinhronizacijo istoimenskega direktorija med vsebnikom in lokalnim datotečnim sistemom.

Primer 2.1.1 (Datotečna struktura projekta za vzpostavitev lokalnega razvojnega okolja z vsebniki Docker.)

```
├── compose.yaml
└── wp-content
```

V nastavitveni datoteki `compose.yaml` definiramo storitve (WordPress, podatkovna baza in po želji orodje za delo s podatkovno bazo, kot je phpMyAdmin), trajne volumne in osnovne omrežne nastavitve, kot je prikazano na primeru 2.1.2. Za definicijo storitev WordPress in podatkovne baze v tem primeru uporabimo dve uradni Docker sliki, ki sta bili predhodno omenjeni. Takšna konfiguracija omogoča vzpostavitev WordPressa na strežniku Apache s podporo jezika PHP in ločeno vzpostavitev podatkovne zbirke v podatkovni bazi MySQL. Z namenom poenostavitve v primer nismo vključili vzpostavitve orodja za delo s podatkovno bazo, kar bi storili na podoben način kot za preostali dve storitvi. V podanih nastavitvah je s pomočjo trajnih volumnov vključeno tudi sinhroniziranje lokalnega direktorija `wp-content` z istoimenskim direktorijem v vsebniku `wordpress`, kar omogoča, da so lokalno izvedene spremembe v direktoriju vidne takoj.

Primer 2.1.2 (Vsebina nastavitvene datoteke `compose.yaml` za vzpostavitev lokalnega razvojnega okolja z vsebniki Docker.)

```
services:
  wordpress:
    image: wordpress:6.8.3-php8.1-apache
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: uporabnik
      WORDPRESS_DB_PASSWORD: geslo
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_TABLE_PREFIX: wp_
    volumes:
      - ./wp-content:/var/www/html/wp-content
    restart: unless-stopped
    ports:
      - 8080:80
```

```
  db:
    image: mysql:8.0
    environment:
      MYSQL_DATABASE: wordpress
      MYSQL_USER: uporabnik
      MYSQL_PASSWORD: geslo
      MYSQL_RANDOM_ROOT_PASSWORD: '1'
    volumes:
      - db_data:/var/lib/mysql
    restart: unless-stopped

volumes:
  db_data:
```

Vzpostavitev razvojnega okolja z orodjem Docker prinaša visoko stopnjo prilagodljivosti, saj ga lahko enostavno prilagodimo posebnim zahtevam in željam. Predstavljeni primeri vzorčne vzpostavitve predstavljajo dobro izhodišče za učinkovit razvoj spletnih rešitev, temelječih na WordPressu. Seveda so možne (in smiselne) nadaljnje prilagoditve.

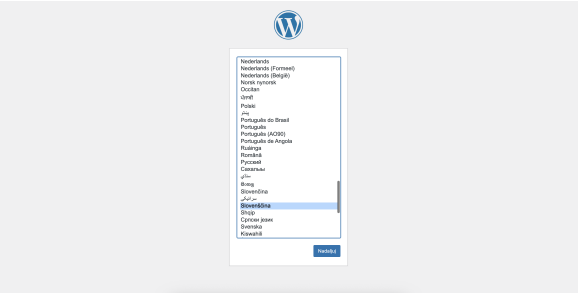
Za uporabo vsebnikov v lokalnem okolju je vedno treba imeti zagnano orodje Docker Desktop oziroma Docker Engine. Za prvi zagon vsebnikov je treba v ukazni vrstici izvesti ukaz iz primera 2.1.3. Naveden ukaz bo po potrebi prenesel vse slike, če še niso bile prenesene, in v odklopljenem načinu ustvaril ter zagnal vse vsebnike, definirane v datoteki `compose.yaml`. Dostop do WordPress spletnega mesta bo nato glede na nastavitve možen v brskalniku na povezavi URL `localhost:8080`.

Primer 2.1.3 (Ukaz v ukazni vrstici za prvi zagon vsebnikov Docker za vzpostavitev lokalnega razvojnega okolja WordPress.)

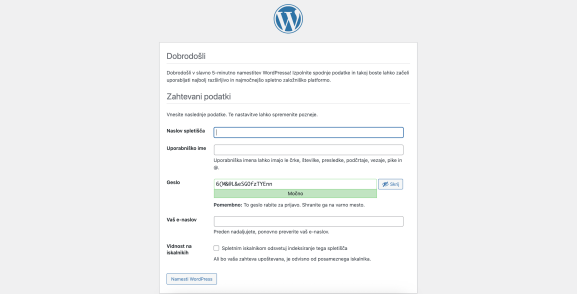
```
$ docker compose up -d
```

Ob prvi vzpostavitvi WordPressa z vsebniki Docker je treba ročno nastaviti WordPress z namestitvenim čarovnikom v brskalniku, kot je prikazano na slikah 2.3 (a-d). Ta zajema nekaj kratkih korakov, v katerih je treba nastaviti jezik vmesnika, naslov spletnega mesta in ustvariti uporabnika s skrbniškimi pravicami za dostop do nadzorne plošče WordPress.

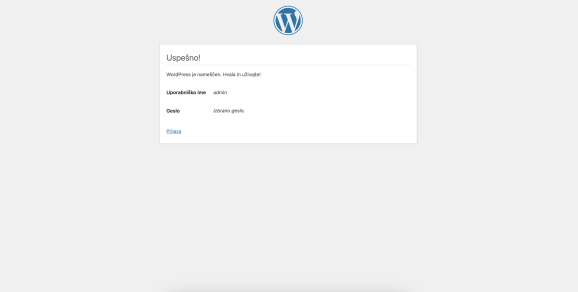
Po uspešni namestitvi je z ustvarjenim uporabnikom možna prijava v nadzorno ploščo WordPress.



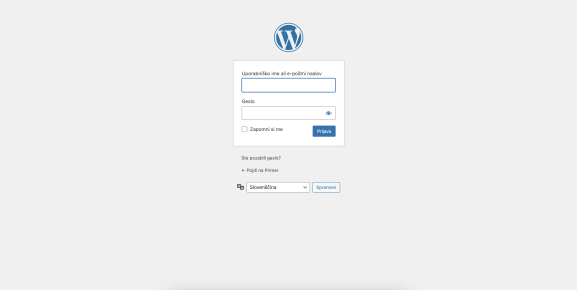
a) Izbira jezika vmesnika.



b) Nastavitve spletnega mesta.



c) Namestitev WordPressa.



d) Prijava v nadzorno ploščo WordPress.

Slika 2.3. Nastavitev WordPressa z namestitvenim čarovnikom v brskalniku.

Za nadaljnje delo z vsebniki Docker so v primeru 2.1.4 navedeni pogosto uporabljeni in koristni ukazi. Najpogosteje se za vsakodnevno rabo uporabljata prva dva ukaza, ki omogočata zagon in zaustavitev vsebnikov.

Primer 2.1.4 (Ukazi v ukazni vrstici za nadaljnje delo z vsebniki Docker v vzpostavljenem lokalnem razvojnem okolju WordPress.)

```
# Zagon/zaustavitev vsebnikov
$ docker compose start
$ docker compose stop

# Dostop v vsebnik (npr. wordpress)
$ docker compose exec wordpress bash

# Zaustavitev in odstranjevanje vsebnikov
$ docker compose down
```

```
# Zaustavitev in odstranjevanje vsebnikov, vključno z volumni
$ docker compose down -v
```

2.2 Namestitev sistema WordPress pri ponudniku gostovanja

Namestitev WordPressa pri ponudniku gostovanja lahko izvedemo na več načinov, najpogosteje pa se uporabljata avtomatizirana namestitev ali ročna namestitev znotraj nadzorne plošče ponudnika gostovanja. Pri tem lahko uporabimo različne načine dostopa do datotečnega sistema spletnega mesta in podatkovne zbirke na oddaljenem strežniku, kar je podrobneje predstavljeno v poglavju 5.5. Za začetnike je vsekakor primernejša avtomatizirana namestitev, saj je hitra in enostavna, medtem ko je ročna namestitev namenjena bolj naprednim razvijalcem, ki želijo popoln nadzor nad postopkom in nastavitvami.

Pri avtomatizirani namestitvi, ki jo ponuja večina ponudnikov, so v nadzorni plošči, kot je cPanel, Plesk ali DirectAdmin, na voljo aplikacije (npr. Softaculous, Installatron ali lastna orodja), ki omogočajo enostavno, enoklikno namestitev WordPressa. Razvijalec običajno v nadzorni plošči poišče WordPress, klikne gumb Install Now in vnese osnovne podatke. V postopku je treba izbrati različico WordPressa, protokol za prenos informacij (priporočeno je HTTPS, če je nameščen certifikat SSL), domeno, naslov in slogan spletne strani, uporabniško ime in geslo za skrbniški dostop, e-naslov, jezik ter direktorij, če želimo, da je WordPress nameščen v poddirektoriju. S klikom na gumb Install sistem samodejno ustvari podatkovno zbirko in namesti WordPress.

Druga možnost je ročna namestitev, ki jo uporabimo, kadar ponudnik avtomatizirane namestitve ne omogoča ali želimo večji nadzor nad postopkom in nastavitvami. Najprej je treba prenesti WordPress z uradne strani in prenesene datoteke naložiti na strežnik v direktorij `public_html`. To lahko storimo bodisi prek odjemalca FTP/SFTP, kot je FileZilla, bodisi prek nadzorne plošče orodja ponudnika gostovanja, kot je cPanel, z orodjem (Online) File Manager. Nato je prek nadzorne plošče ponudnika gostovanja, kot je cPanel, treba ustvariti podatkovno zbirko ter uporabnika z uporabniškim imenom in geslom, mu dodeliti ustrezne pravice ter povezati uporabnika z novo ustvarjeno podatkovno zbirko. Podatke o imenu podatkovne zbirke, uporabniškem imenu in geslu

si je treba zabeležiti, saj jih je treba vnesti v konfiguracijsko datoteko `wp-config.php`. To datoteko lahko uredimo prek FTP/SFTP ali neposredno prek nadzorne plošče, kot je cPanel, z orodjem (Online) File Manager. Ko so nastavitve urejene, v brskalniku odpremo domeno in zažene se namestitveni čarovnik (podoben tistemu, ki je predstavljen na slikah 2.3 (a-d) v predhodnem podpoglavju). V njem izberemo jezik in vnesemo zahtevane podatke, kot so naslov spletne strani, uporabniško ime in geslo za skrbniški dostop ter e-naslov. Postopek zaključimo s klikom na gumb Install WordPress.

V obeh primerih se po uspešni namestitvi v nadzorno ploščo WordPress prijavimo na povezavi URL `https://ime-domene.si/wp-admin`, kjer za dostop uporabimo uporabniško ime in geslo, določeno med namestitvijo.

- Pred namestitvijo WordPressa pri ponudniku gostovanja je treba najprej izbrati ustrezen paket gostovanja in zakupiti domeno.
- Na slovenskem trgu je na voljo več ponudnikov gostovanja, med bolj znanimi so na primer NEOSERV, Domenca, Domovanje in Hostko, poleg njih pa lahko uporabimo tudi tuje ponudnike, kot so na primer SiteGround, Bluehost, Hostinger ali DreamHost.
- Pred nakupom paketa gostovanja je priporočljivo preveriti, ali ta podpira vse potrebne tehnične zahteve za delovanje WordPressa. Ključne so podpora za PHP, podatkovna baza MySQL oziroma MariaDB ter certifikat SSL, ki omogoča varno povezavo prek protokola HTTPS.

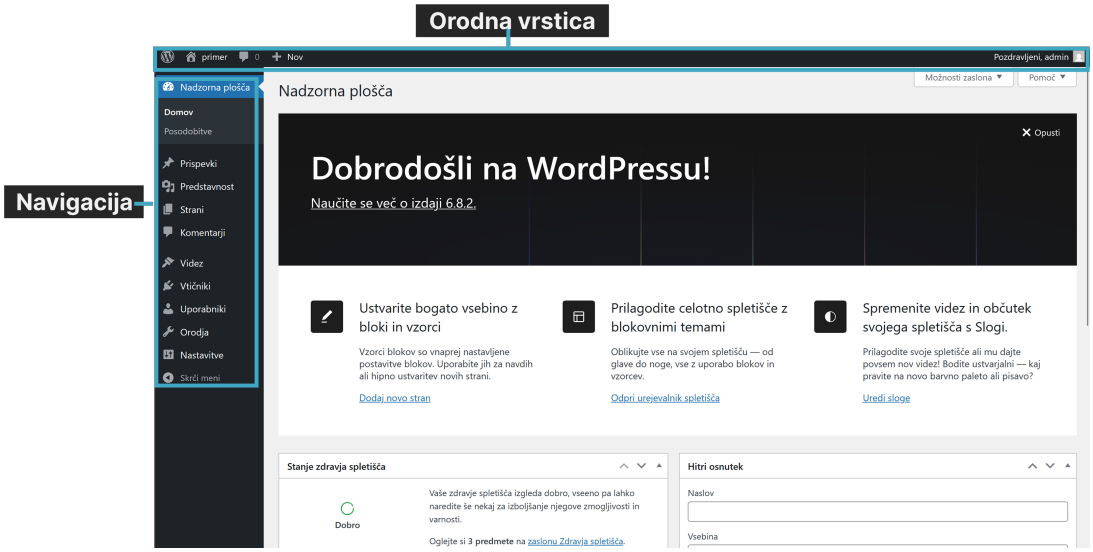
POGLAVJE 3

Osnovno delo s sistemom WordPress

Skrbniški vmesnik WordPress je sestavljen iz več funkcionalnih sklopov, ki omogočajo učinkovito upravljanje spletnega mesta. Skrbniški vmesnik (včasih poimenovan tudi nadzorna plošča kot splošen izraz vmesnika za vse uporabnike, ne glede na vlogo in dovoljenja) je oblikovan po enakem principu ne glede na velikost in kompleksnost vzpostavljenega WordPress projekta. V levem stranskem meniju je glavna navigacija, ki omogoča dostop do vsebinskih modulov, kot so prispevki, strani, medijske vsebine, komentarji, izgled, vtičniki, uporabniki in nastavitve. Zgornji del vmesnika zavzema orodna vrstica, ki ponuja bližnjice do najpogosteje uporabljenih funkcionalnosti, kot so dodajanje nove vsebine, dostop do profila uporabnika ter hiter preklop med nadzorno ploščo in uporabniškim pogledom strani. Takšna struktura omogoča jasno ločitev med različnimi upravljavskimi nalogami in prispeva k preglednosti sistema. Pregled osnovnih delov vmesnika je prikazan na sliki 3.1.

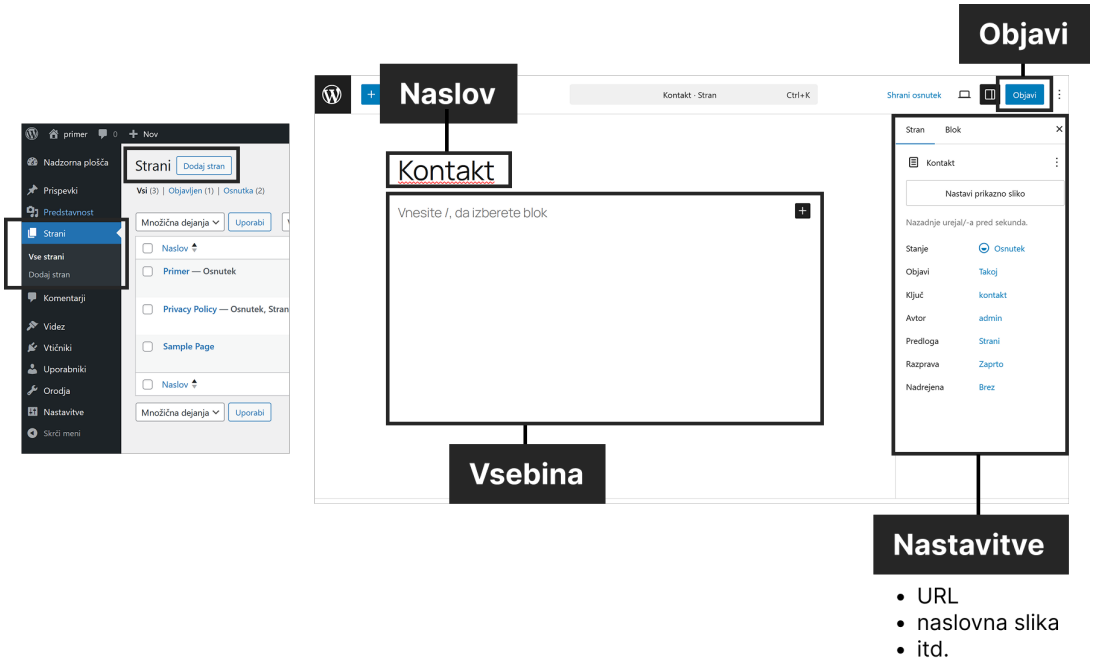
3.1 Strani

V sistemu WordPress so statične vsebine običajno organizirane v obliki strani (angl. Pages), ki se uporabljajo za prikaz informacij, kot so kontaktni podatki, predstavitev podjetja ali organizacije, pogoji uporabe, politika zasebnosti ter druge vsebine, ki se redkeje posodabljajo. Uporabniki lahko ustvarijo poljubno število strani, ki so nato vključene v navigacijo spletnega mesta. Stran je sestavljena iz naslova, vsebine ter dodatnih nastavitev, kot so naslov URL, vidnost in naslovna slika. Vizualno in vsebinsko



Slika 3.1. Pregled osnovnih delov skrbniškega vmesnika WordPress.

urejanje strani privzeto poteka z blokovnim urejevalnikom, kjer je vsaka vsebinska enota predstavljena kot samostojen blok (več o tem v poglavju 6.2). Postopek dodajanja nove strani in osnovni elementi urejevalnika so prikazani na sliki 3.2. Po zaključku dodajanja vsebine je treba stran objaviti (gumb Objavi). Če urejanje vsebine še ni zaključeno, je stran mogoče shraniti tudi kot osnutek. Prav tako je mogoče vnaprej določiti datum in uro objave strani.



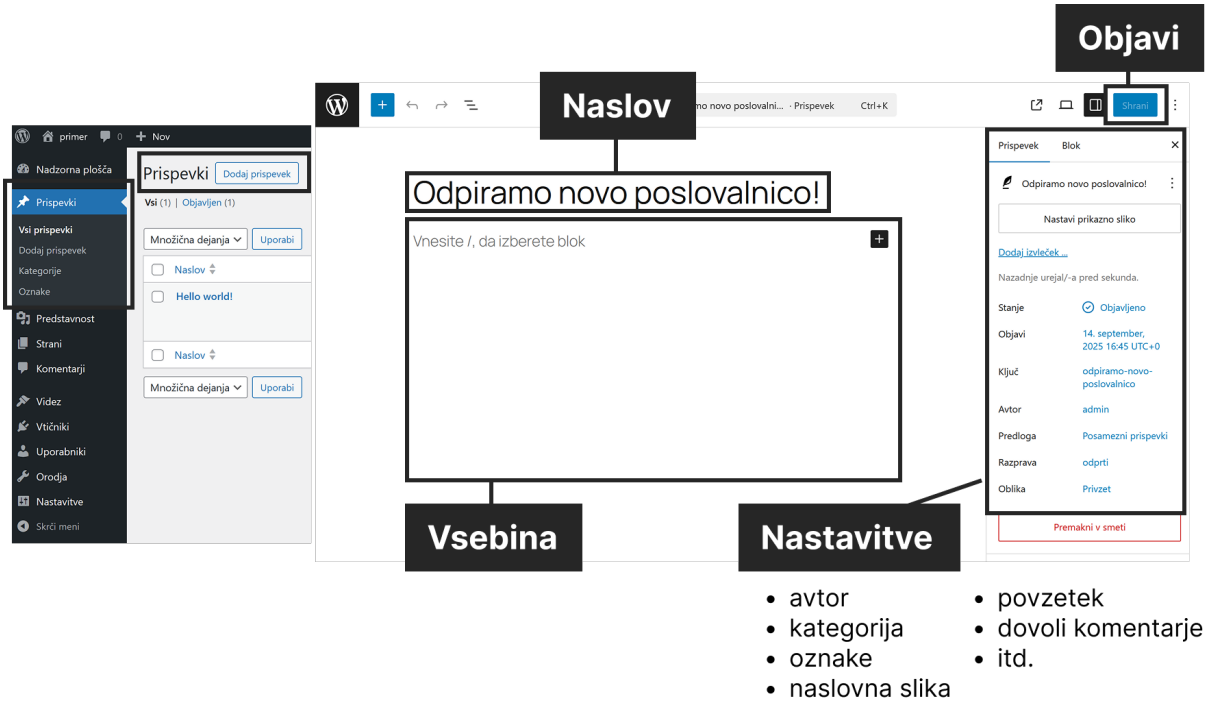
Slika 3.2. Dodajanje in urejanje strani.

3.2 Prispevki

Prispevki (angl. Posts) se v WordPressu uporabaljajo za dodajanje dinamičnih vsebin, kot so novice, obvestila, članki ali druge objave, ki se običajno prikazujejo v kronološkem zaporedju na domači strani ali v blog sekciji spletnega mesta. Vsak prispevek vključuje naslov, vsebino in nabor nastavitev, kot so avtor, kategorija, oznake, naslovna slika, povzetek in možnosti komentiranja. Prispevke je mogoče organizirati z uporabo ustreznih taksonomij (angl. Taxonomies), in sicer privzeto s pomočjo kategorij (angl. Categories) in s pomočjo oznak (angl. Tags), kar olajša razvrščanje ter omogoča boljšo strukturo vsebin. Podobno kot pri urejanju strani uporabniški vmesnik omogoča preprosto urejanje novih prispevkov z blokovnim urejevalnikom (več o tem v poglavju 6.2). Postopek ustvarjanja prispevka in ključne komponente vmesnika so prikazani na sliki 3.3. Postopek je podoben dodajanju strani.

Stran ali prispevek? *Prispevki* se uporabljajo za vsebine, ki se redno posodabljajo in so v kronološkem zaporedju – na primer novice, blog objave ali obvestila. Omogočajo kategorizacijo, označevanje in vključujejo možnost komentiranja. *Strani* pa so namenjene statičnim vsebinam, kot so »O nas«, »Kontakt«, »Pogoji uporabe«, in običajno ne vključujejo datumov, taksonomij ali komentarjev. Uporaba ustreznega tipa prispevka pripomore k boljši organizaciji in uporabniški izkušnji spletnega mesta.

Kategorija ali oznaka? *Kategorijo* izberemo, ko želimo prispevek uvrstiti v eno izmed glavnih tem na naši strani (npr. »Recepti«, »Novice«, »Poletna ponudba«). Vsak prispevek mora imeti vsaj eno kategorijo. Če te ne izberemo, WordPress vsebine privzeto uvrsti v kategorijo z imenom »Nekategorizirano«. Kategorije je mogoče organizirati hierarhično – vsaka kategorija ima lahko nadrejene in podrejene kategorije. To omogoča gradnjo logične strukture, kjer so širše teme na vrhu hierarhije, natančnejše ali bolj specifične pa razporejene pod njimi (npr. kategorija »Recepti« ima lahko podrejene kategorije »Zajtrki«, »Juhe«, »Glavne jedi« in »Sladice«, vsaka od njih pa ima lahko še podrobnejše podkategorije, npr. kategorija »Juhe« ima lahko podkategorije »Zelenjavne juhe«, »Mesne juhe«, »Kremne juhe«). *Oznake* pa dodamo, ko želimo podrobneje opisati vsebino (npr. »poletna ohladitev«, »vegansko«, »hitra priprava«). Oznake niso hierarhične in



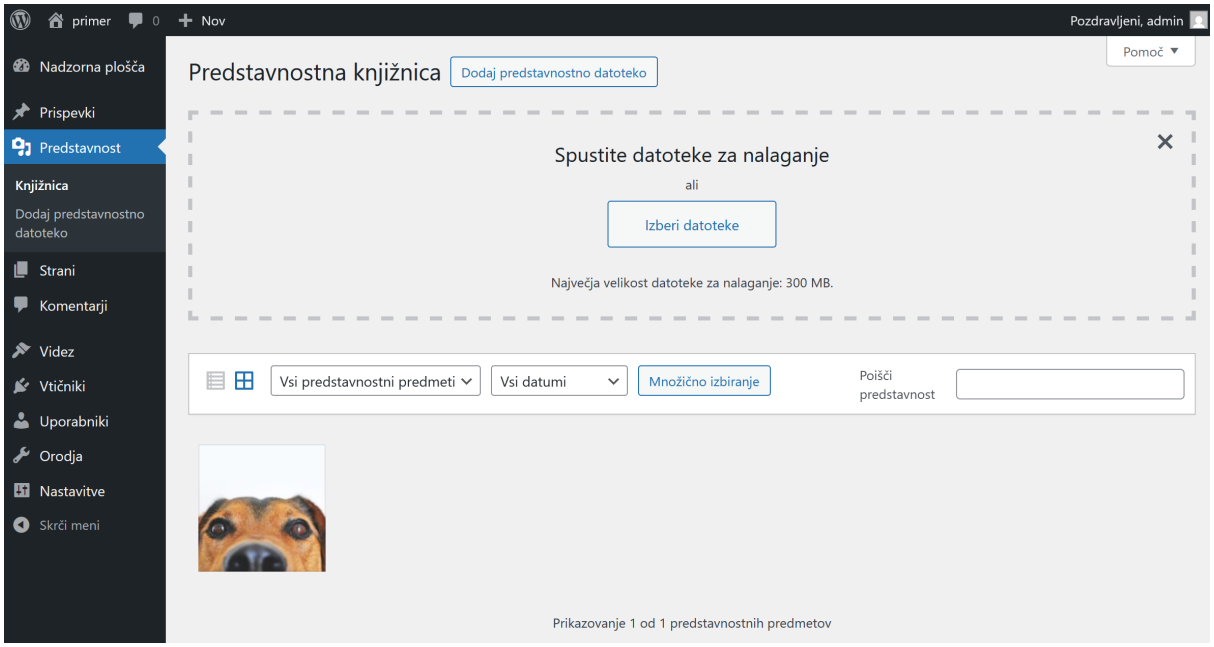
Slika 3.3. Dodajanje in urejanje prispevkov.

služijo kot dodatni opisni elementi, ki bralcem (in iskalnikom) pomagajo lažje najti sorodne prispevke. En prispevek ima lahko poljubno število oznak, pri čemer naj bodo te čim bolj specifične in smiselne. Na primer prispevek v kategoriji »Juhe« s podkategorijo »Zelenjavne juhe« lahko vsebuje oznake »buča«, »jesensko«, »vegansko« in »preprosto«.

3.3 Medijske vsebine

WordPress omogoča enostavno dodajanje in upravljanje medijskih vsebin z zavihkom Predstavnost (angl. Media). Uporabniki lahko naložijo slike (npr. .JPEG, .PNG, .GIF, .WEBP), dokumente (npr. .PDF), zvočne posnetke (npr. .MP3) in videoposnetke (npr. .MP4), ki jih nato vključujejo v prispevke, strani ali druge vsebine. Nalaganje poteka prek grafičnega vmesnika, ki omogoča funkcionalnost *povleci-in-spusti* (angl. Drag-and-Drop) ali klasičen izbor datotek s pomočjo raziskovalca na računalniku. WordPress privzeto omejuje velikost posamezne datoteke pri nalaganju, pri čemer je največja dovoljena velikost običajno določena na strežniški ravni in se najpogosteje giblje od 2 MB do 64 MB, odvisno od nastavitev gostovanja. Osnovni pregled nalaganja in upravljanja z medijskimi

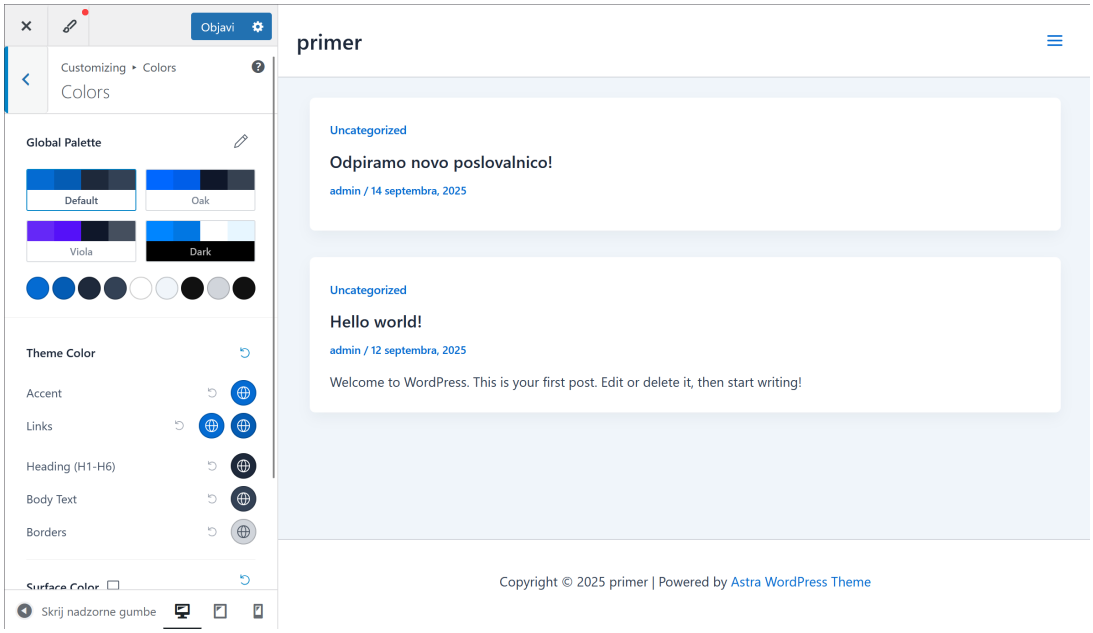
vsebinami je prikazan na sliki 3.4.



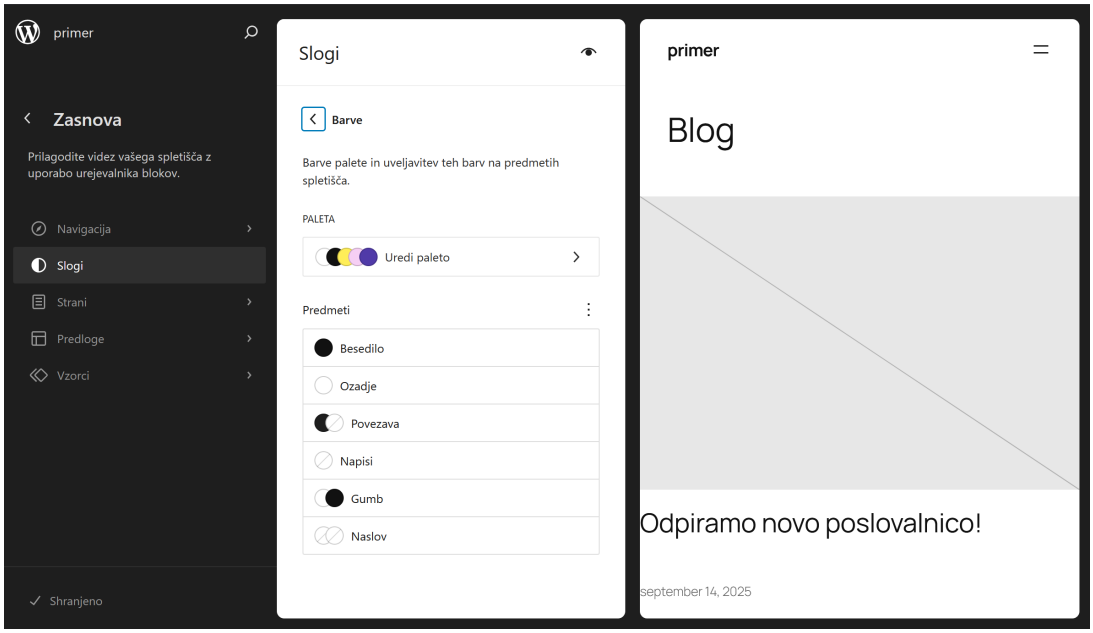
Slika 3.4. Dodajanje in urejanje medijskih vsebin.

3.4 Videz

V zavihku Videz (angl. Appearance) v nadzorni plošči WordPressa lahko upravljamo z vizualnim izgledom spletnega mesta. Ključni element tega zavihka so teme (angl. Themes), ki določajo splošen celosten izgled spletnega mesta – od postavitve in tipografije do barv in slogov posameznih elementov. Teme so zbirke datotek, ki skupaj določajo videz spletnega mesta za končne uporabnike. Obstajata dva tipa tem, klasične teme (angl. Classic Themes) in blokovne teme (angl. Block Themes), ki se razlikujejo glede na način urejanja teme, strukturo datotek ter obseg vizualnih prilagoditev brez kode. Lastnosti in razlike obeh tipov tem so podrobneje predstavljene v poglavju 5.3. V zavihku Videz lahko teme iščemo, nameščamo, aktiviramo in prilagajamo, pri čemer razpoložljive možnosti prilagoditev določa posamezna tema. Primer upravljanja z nastavitvami klasične teme je prikazan na sliki 3.5, medtem ko je upravljanje z nastavitvami blokovne teme prikazano na sliki 3.6. Pri sodobnih blokovnih temah je na voljo tudi urejevalnik spletnega mesta (angl. Site Editor), ki omogoča neposredno urejanje predlog in slogov brez poseganja v programsko kodo.



Slika 3.5. Nastavitve klasične teme (tema Astra).

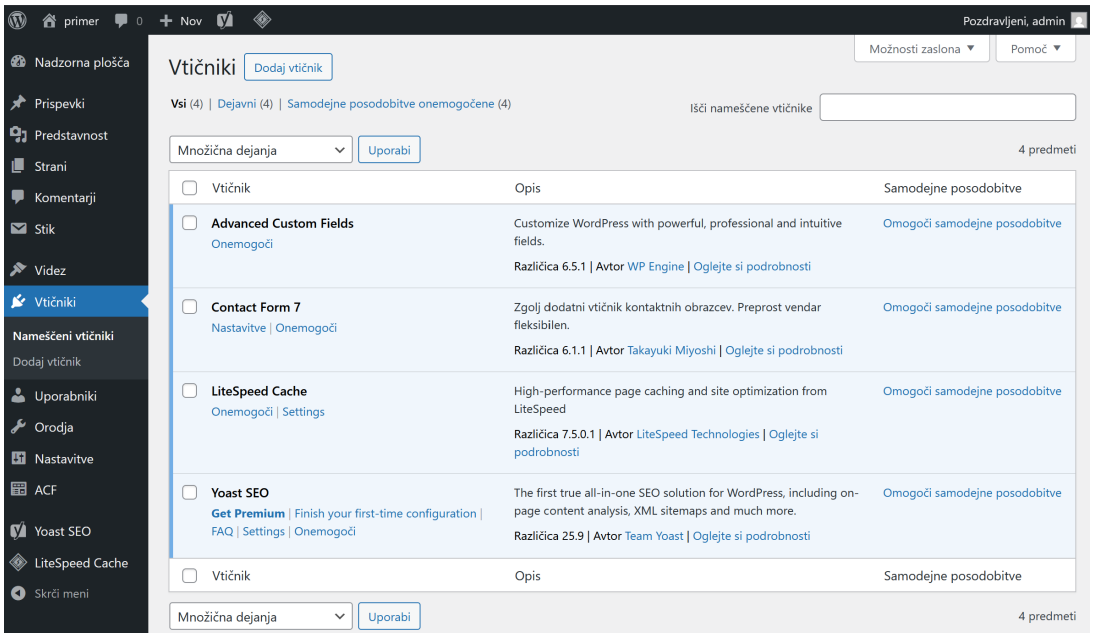


Slika 3.6. Nastavitve blokovne teme (tema Twenty Twenty-Five).

3.5 Vtičniki

V zavihku Vtičniki (angl. Plugins) v nadzorni plošči WordPress lahko iščemo, nameščamo, aktiviramo in upravljamo z vtičniki. Vtičniki v WordPressu so razširitve, ki dodajajo ali spreminjajo funkcionalnosti spletnega mesta brez poseganja v izvorno kodo

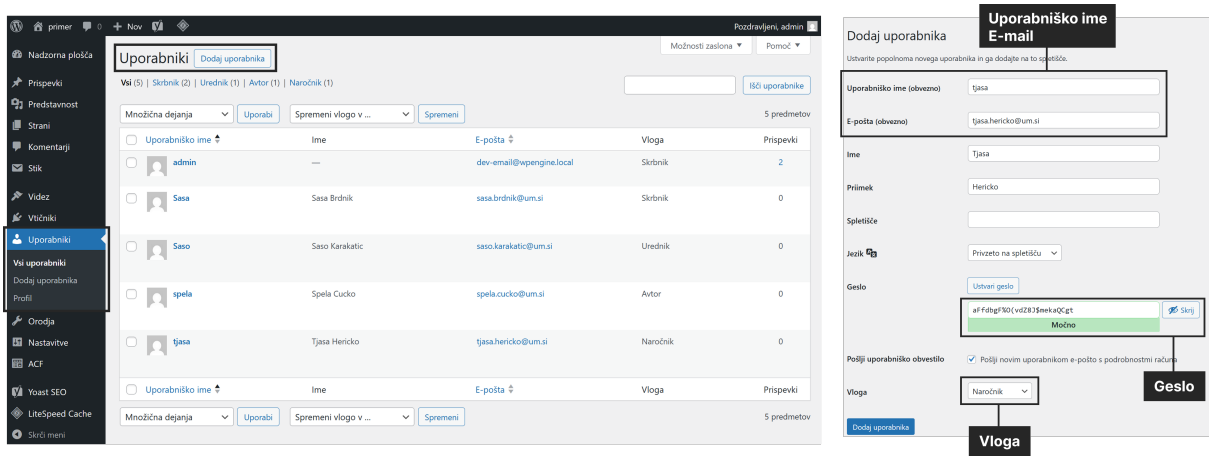
jedra WordPressa. Razvijalcem omogočajo, da po potrebi razširijo osnovne zmogljivosti sistema, na primer z dodajanjem obrazcev, varnostnih mehanizmov, analitike ali integracij z zunanjimi storitvami. Vtičniki temeljijo na uporabi WordPress razredov, funkcij in dogodkov, s katerimi se vključujejo v delovanje WordPressa in drugih vtičnikov. Zaradi modularne zasnove omogočajo popolno prilagodljivost spletnega mesta. Pregled upravljanja z vtičniki v nadzorni plošči WordPress je predstavljen na sliki 3.7, medtem ko so vtičniki podrobneje predstavljeni v poglavju 7.



Slika 3.7. Upravljanje z vtičniki.

3.6 Uporabniki

V WordPressu je upravljanje uporabnikov ključno za upravljanje nadzora nad dostopom in vsebinami na spletnem mestu. Sistem omogoča dodeljevanje različnih uporabniških vlog, odvisno od njihove odgovornosti in nalog. Vsaka vloga ima vnaprej določena dovoljenja, ki omejujejo ali omogočajo določene funkcionalnosti, kot so objavljanje prispevkov, urejanje strani, upravljanje tem in vtičnikov ali spreminjanje nastavitev. Na ta način je mogoče vzpostaviti varno sodelovanje več uporabnikov brez tveganja nenamernih sprememb ali zlorab. Dodajanje uporabnikov je prikazano na sliki 3.8.



Slika 3.8. Dodajanje uporabnikov.

V preglednici 3.1 so predstavljene osnovne vloge uporabnikov, ki jih WordPress ponuja privzeto, skupaj z njihovimi osrednjimi dovoljenji.

Preglednica 3.1. Osnovne vloge uporabnikov z osrednjimi pripadajočimi dovoljenji [6].

Vloga	Opis in dovoljenja
Skrbnik (angl. Administrator)	Ima popoln nadzor nad spletnim mestom, vključno z upravljanjem tem, vtičnikov in uporabnikov. Lahko dodaja, ureja, objavlja in briše prispevke ter strani. Primerna le za zaupanja vredne osebe z najvišjimi pravicami.
Urednik (angl. Editor)	Lahko dodaja, ureja, objavlja in briše vse prispevke in strani – tudi od drugih uporabnikov. Primeren za odgovorne osebe, ki skrbijo za celotno vsebinsko strukturo.
Avtor (angl. Author)	Lahko dodaja, ureja, objavlja in briše le lastne prispevke. Nima dostopa do strani ali nastavitev. Primeren za redne pisce vsebin.
Sodelavec (angl. Contributor)	Lahko dodaja, ureja in briše lastne osnutke prispevkov, vendar ne more objavljati prispevkov. Objavo mora potrditi uporabnik z višjimi pravicami.
Naročnik (angl. Subscriber)	Ima dostop do osebnega profila, ki ga lahko upravlja, in lahko komentira prispevke (če je omogočeno), vendar nima uredniških pravic. Uporaben za registrirane bralce ali stranke.

Skladno z dobrimi praksami informacijske varnosti je priporočljivo, da se vloge dodeljujejo po načelu najmanjših privilegijev (angl. Principal Least Privilege). To pomeni, da nobena oseba ne sme imeti več dovoljenj, kot je najnujnejše za opravljanje njenega dela [7].

3.7 Nastavitve

Nastavitve WordPressa so osrednje izhodišče za prilagoditev osnovnih in naprednih parametrov spletnega mesta. Do njih dostopamo prek zavihka Nastavitve (angl. Settings) v skrbniškem vmesniku. V nadaljevanju so predstavljeni posamezni sklopi nastavitev, ki uporabniku omogočajo nadzor nad delovanjem in prikazom strani:

- Splošne nastavitve (angl. General) so prikazane na sliki 3.9 in vključujejo ključne informacije o spletnem mestu. Vključujejo določitev naslova spletne strani (angl. Site Title) in slogana (angl. Tagline), ki se pogosto prikazujeta v naslovni vrstici brskalnika in kot metapodatka za iskalnike. Naslov URL za WordPress (angl. WordPress Address (URL)) in naslov URL spletišča (angl. Site Address (URL)) določata, kje je nameščeno jedro WordPressa in katera domena predstavlja dostopno spletno mesto. Kontaktni e-naslov skrbnika je uporabljen za pošiljanje sistemskih obvestil. Možno je omogočiti registracijo uporabnikov in določiti privzeto uporabniško vlogo za nove registrirane uporabnike. Na voljo so tudi jezikovne nastavitve, časovni pas in format prikaza datuma ter ure. Nadzorno ploščo je med drugim mogoče uporabljati tudi v slovenskem jeziku.

- Nastavitve pisanja (angl. Writing) omogočajo nastavitev privzete kategorije in oblike novih prispevkov. Na voljo so možnosti za obveščanje o objavi prispevka prek elektronske pošte in nastavitev storitev, ki obveščajo ob novi objavi vsebin.

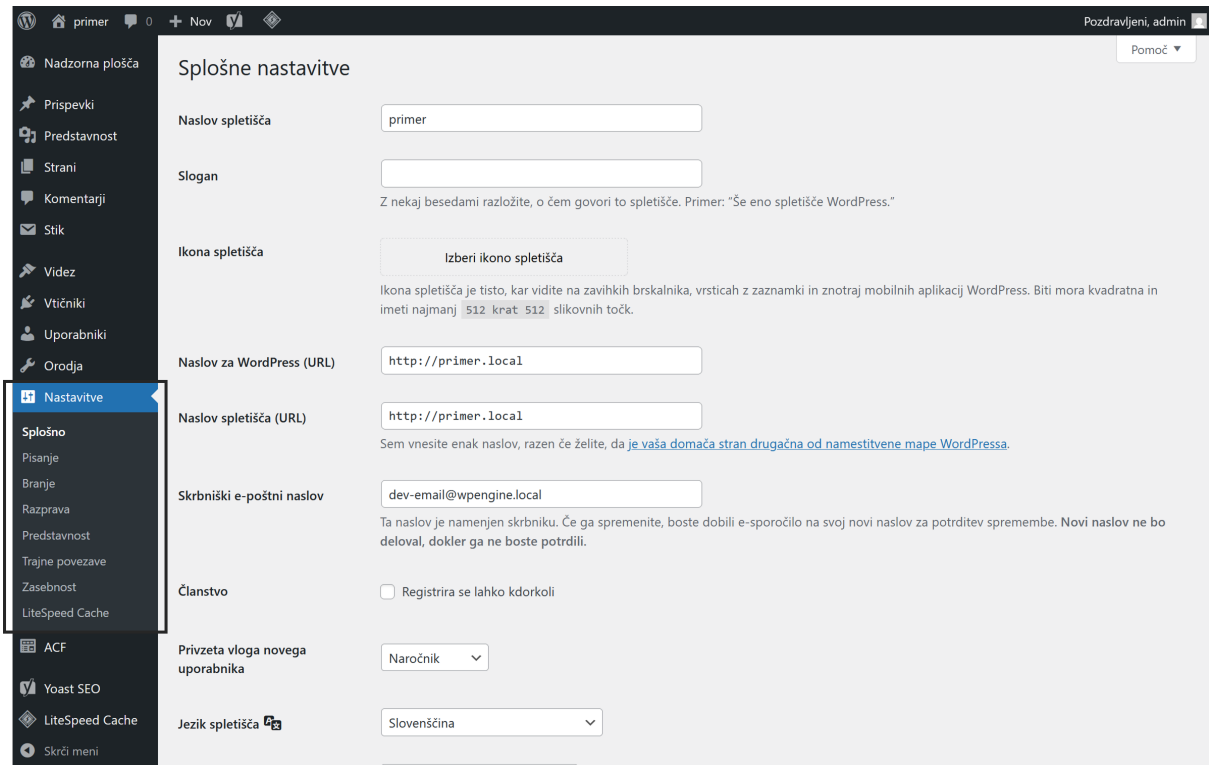
- Nastavitve branja (angl. Reading) vplivajo na prikaz domače strani, pri čemer je možno izbirati med prikazom najnovejših prispevkov ali uporabo statične strani. V tem razdelku se prav tako določa število prikazanih prispevkov na posamezni strani in možnost, da spletnim iskalnikom odsvetujemo indeksiranje spletnega mesta, kar je koristno med razvojem.

- Nastavitve razprave (angl. Discussion) ponujajo upravljanje komentarjev. Vključujejo možnost omogočanja ali onemogočanja komentarjev, nadzor nad obveščanjem, moderiranjem, uporabo avatarjev ter druge nastavitve, povezane z interakcijo uporabnikov.

- Nastavitve medijskih vsebin (angl. Media) omogočajo določitev privzetih dimenzij slik, ki se samodejno ustvarijo ob nalaganju (npr. sličica, srednja in velika velikost). Možno je tudi organizirati medijske datoteke po datumu nalaganja.

- Trajne povezave (angl. Permalinks) določajo strukturo povezav prispevkov. Na voljo je izbira med več vnaprej definiranimi oblikami (npr. na osnovi ID-ja, naslova, datuma) ali določitev prilagojene strukture povezav URL, kar vpliva na berljivost in optimizacijo spletnega mesta za iskalnike.

- Zasebnost (angl. Privacy) omogoča izbiro strani z izjavo o zasebnosti, kar je zlasti pomembno za skladnost s predpisi o varstvu osebnih podatkov (npr. GDPR). WordPress omogoča ustvarjanje nove strani z vnaprej pripravljenim besedilom, ki se lahko dodatno prilagodi glede na potrebe spletnega mesta.



Slika 3.9. Splošne nastavitve WordPress spletnega mesta.

POGLAVJE 4

Priprava in prilagoditev spletnih vsebin za objavo

To poglavje je namenjeno pripravi in prilagoditvi spletnih vsebin v kontekstu WordPressa za objavo s ciljem boljše vidnosti v spletnih iskalnikih. Pri tem upoštevamo smernice s področja optimizacije za spletne iskalnike (angl. Search Engine Optimization ali SEO). Optimizacijo je treba razumeti kot neprestan proces prilagajanja vsebin in tehničnih lastnosti ter nastavitev spletnega mesta, da jih iskalniki lažje preiščejo, indeksirajo in ustrezno uvrstijo v rezultate iskanja. Optimizacijo vsebin za iskalnike v tem poglavju obravnavamo kot nabor konkretnih odločitev pri pisanju, strukturiranju in objavljanju vsebin. Prilagoditve vsebin so osredotočene na jasnost, berljivost in ustreznost, brez poudarka na trikih ali zavajanju iskalnikov.

4.1 Notranji in zunanji dejavniki ter metrike optimizacije vsebin za iskalnike

V praksi optimizacijo za iskalnike pogosto ločimo glede na notranje in zunanje vidike. Pri tem razlikujemo med dejavniki, ki določajo, kaj in kako pripravimo ter prilagodimo, ter metrikami, s katerimi učinek merimo.

Notranje metrike (angl. On-Site) nakazujejo, kako dobro je vsebina na spletni strani tehnično in vsebinsko pripravljena za iskalnike in uporabnike. Iskalniki vsebino obdelujejo prek strukture strani (HTML/DOM) in metapodatkov ter jo po potrebi tudi upodabljajo, zato so ključni jasna informacijska struktura, pravilna raba naslovov, smiselni URL-ji, ustrezno zapisani metapodatki (npr. opisi za izsek v rezultatih iskanja), alternativni opisi

slik (atribut `alt`) ter premišljena uporaba ključnih besed in taksonomij. Vsebina mora biti razumljiva, logično razdeljena in pisana predvsem za uporabnika, ne za algoritem. Pretiravanje s ključnimi besedami ali tehnične napake vodijo v slabše uvrstitve [8].

Zunanje metrike (angl. Off-Site) kažejo, kako je spletna stran umeščena v širši splet. Med najpomembnejše signale sodijo povezave z drugih spletnih mest. Več kot je kakovostnih in relevantnih povezav, večji ugled lahko pripišemo strani in višje se lahko uvrsti v rezultatih iskanja. Vse povezave niso enakovredne – povezave z zaupanja vrednih in uveljavljenih strani praviloma štejejo bistveno več kot množica nepomembnih. Sem sodijo tudi omembe in deljenje vsebin na zunanjih platformah (npr. družbena omrežja) [8].

4.2 Dobre prakse optimizacije vsebin za iskalnike

Optimizacija vsebin na spletnem mestu ne obsega le enkratne prilagoditve, temveč predstavlja neprestan niz majhnih in doslednih dejanj ter odločitev. Spodnji seznam zajema izbrane pogoste dobre prakse, ki posredno ali neposredno vplivajo na indeksiranje vsebin in spletnega mesta v iskalnikih [9, 10].

- `Kratek URL` – URL naj bo kratek, pomenski in brez nepotrebnih parametrov. Iz naslova mora biti vsebina strani takoj jasna. *Primer:* `/recept/cokoladni-piskoti` namesto `/index.php?id=123&lang=sl`.
- `Alternativni opis pri slikah` – vsaka slika mora imeti alternativni opis (atribut `alt`), ki opisuje dejansko vsebino slike. To izboljša razumevanje vsebine s strani iskalnikov in dostopnost za uporabnike z bralniki zaslona. Izjeme so dekorativne slike (npr. slika vodoravne črte, ki predstavlja vizualno razmejitev). *Primer:* `alt="Čokoladni piškot"` namesto `alt="slika1"` .
- `Pravilna uporaba prispevkov ali strani` – kot je bilo omenjeno v poglavju 3.2, so prispevki namenjeni časovno vezanim vsebinam, strani pa trajnim informacijam. Napačna raba vodi v slabšo informacijsko arhitekturo. *Primer:* Novice in recepte objavljamo kot prispevke, opis spletnega mesta kot kuharske platforme z recepti pa kot stran.

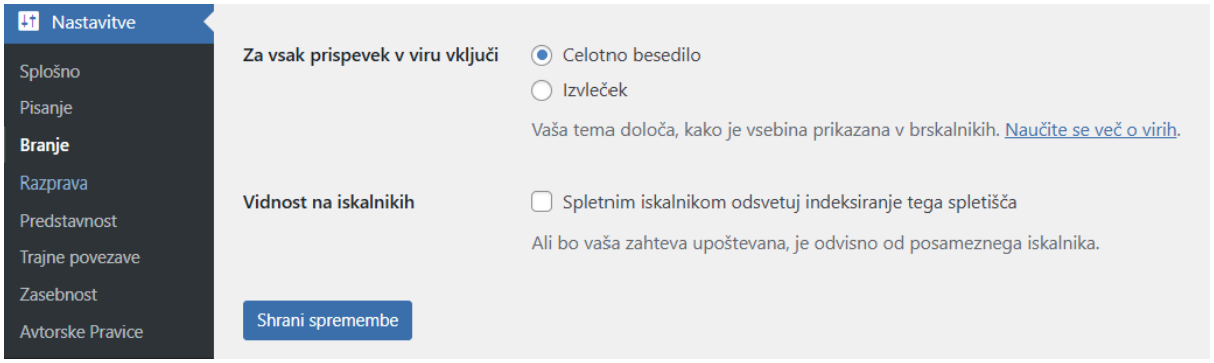
- `Uporaba taksonomij` – kategorije in oznake naj bodo smiselne, omejene in dosledno uporabljene. Njihov namen je strukturiranje vsebine in lažja navigacija. *Primer:* Ena vsebinska kategorija in nekaj natančnih oznak namesto več skoraj identičnih kategorij.
- `Kratki opisi` – kratki opisi kot metapodatki naj povzamejo bistvo vsebine. Naj bodo informativni, jedrnati in vsebinsko skladni z dejansko vsebino strani. *Primer:* 1–2 povedi, ki jasno odgovorita, kaj lahko uporabnik na strani pričakuje.
- `Prelomi »Preberi več«` – prelomi izboljšajo preglednost daljših vsebin in omogočajo hitrejši pregled objav. Uporabnik se sam odloči za nadaljnje branje. *Primer:* Povzetek prispevka na arhivski strani in celotna vsebina na podstrani.
- `Relevantne povezave` – notranje in zunanje povezave morajo biti vsebinsko smiselne in kontekstualno umeščene. Pomagajo uporabniku pri razumevanju teme in iskalniku pri vzpostavljanju povezav med pojmi. *Primer:* Povezava na sorodno vsebino (npr. recept) ali zaupanja vreden zunanji vir, ne generične ali zavajajoče povezave tipa »klikni tukaj«.
- `Optimizacija slik` – slike naj bodo ustreznih dimenzij in stisnjene. *Primer:* Brez nalaganja izvornih fotografij visoke ločljivosti, če to ni nujno potrebno.
- `Upravljanje predpomnjenja` – predpomnjenje se nanaša na mehanizem, pri katerem se statične ali redko spreminjajoče vsebine shranijo v začasni pomnilnik, da se ob ponovnem obisku ne generirajo in ne nalagajo znova. S tem se zmanjša obremenitev strežnika in bistveno pospeši nalaganje strani. *Primer:* predpomnjenje na ravni strežnika ali znotraj WordPressa.

4.3 Analiza optimizacije za iskalnike

Analiza optimizacije za iskalnike temelji na sistematičnem preverjanju tehničnih in vsebinskih dejavnikov, ki vplivajo na odkrivanje, indeksiranje ter vidnost spletnega mesta v iskalnikih. V praksi se za osnovno diagnostično preverjanje pogosto uporabljajo orodja, bodisi namenska orodja za analizo optimizacije za iskalnike bodisi splošna orodja za analizo kakovosti spletnih mest, kot je Google Lighthouse [11].

Lighthouse na izbrani strani izvede nabor analiz in poda ocene ter priporočila za ključna področja, ki predstavljajo tehnične predpogoje in dobre prakse za iskalnike. Med tipičnimi preverjanji so prisotnost in osnovna pravilnost meta oznak (npr. meta opis), raba naslovnih elementov, berljivost URL-jev, dostopnost slik z alternativnimi opisi, osnovna mobilna prilagoditev (npr. pravilna raba `viewport`) ter druge temeljne tehnične nastavitve. Pomembno je poudariti, da Lighthouse ne izvede celovite analize optimizacije za iskalnike in hkrati prav tako ne more nadomestiti vsebinske presoje (npr. ustreznosti, namena iskanja in kakovosti vsebine), temveč služi kot hiter pregled pogostih pomanjkljivosti na strani. Pomemben del analize je tudi povezava z zmogljivostjo strani. Hitrost nalaganja, velikost slik, število zahtevkov in uporaba predpomnjenja vplivajo na zmogljivost ter uporabniško izkušnjo, kar lahko posredno vpliva tudi na razvrstitev v iskalnih rezultatih. Zato se analiza optimizacije za iskalnike ne omejuje le na vsebino, temveč vključuje tudi tehnične vidike delovanja strani. Lighthouse rezultate predstavi v obliki številčnih ocen in konkretnih priporočil. Namen analize ni doseganje popolne ocene, temveč prepoznavanje ključnih pomanjkljivosti in razumevanje, kako posamezni popravki vplivajo na uporabnost in tehnično kakovost strani. Orodje je zato primerno kot diagnostični pripomoček, ki omogoča ponovljivo analizo in primerjavo stanja pred optimizacijo in po njej.

Pri optimizaciji za iskalnike je nujno preveriti tudi sistemske nastavitve v nadzorni plošči WordPressa. V nastavitvah *Branje* mora biti možnost »Vidnost na iskalnikih« izklopljena, kot je prikazano na sliki 4.1. Omogočena nastavitev namreč spletnim iskalnikom posreduje priporočilo, naj spletnega mesta ne indeksirajo, ne glede na sicer ustrezno optimizacijo vsebine.



Slika 4.1. Nastavitve branja v nadzorni plošči WordPress.

POGLAVJE 5

Datotečna struktura v sistemu WordPress

V prejšnjih poglavjih so bile predstavljene osnovne funkcionalnosti jedra WordPressa, ki omogočajo upravljanje spletnih vsebin. A le redko uporaba zgolj osnovnih funkcionalnosti zadostuje za razvoj in upravljanje spletne rešitve. Najpogosteje je treba uporabiti vizualne prilagoditve z uporabo tem, funkcionalne prilagoditve z uporabo vtičnikov in druge prilagoditve, ki zahtevajo dobro poznavanje jedra WordPressa. V nadaljevanju bodo predstavljeni ključni direktoriji in datoteke WordPress, shema podatkovne zbirke, hierarhija predlog in načini dostopa do datotek ter podatkovne zbirke.

5.1 Ključni direktoriji in datoteke

Korenski direktorij WordPress projekta, primer katerega je prikazan na primeru 5.1.1, vsebuje tri ključne sistemske direktorije (`wp-admin`, `wp-content`, `wp-includes`) ter vrsto datotek, ki skrbijo za delovanje sistema za upravljanje vsebin. Med pomembnejšimi izstopajo datoteke `index.php`, `wp-config.php` in `.htaccess`. Te datoteke in direktoriji skupaj tvorijo jedro sistema in omogočajo vse od vzpostavitve povezave z bazo podatkov do nalaganja skrbniškega vmesnika in prikaza spletnih strani končnim uporabnikom [12, 13].

Primer 5.1.1 (Datotečna struktura WordPress projekta.)

```
├── wp-admin
│   ├── themes
│   │   └── ...
│   ├── plugins
│   │   └── ...
│   ├── uploads
│   │   └── ...
│   └── index.php
├── wp-content
│   └── ...
├── wp-includes
│   └── ...
├── .htaccess
├── index.php
├── license.txt
├── local-xdebugInfo.php
├── readme.html
├── wp-activate.php
├── wp-blog-header.php
├── wp-comments-post.php
├── wp-config.php
├── wp-config-sample.php
├── wp-cron.php
├── wp-links-opml.php
├── wp-load.php
├── wp-login.php
├── wp-mail.php
├── wp-settings.php
├── wp-signup.php
├── wp-trackback.php
└── xmlrpc.php
```

Vsak izmed navedenih direktorijev in datotek ima svojo specifično vlogo ter prispeva k delovanju WordPressa:

- Direktorij `wp-admin` je osrednji direktorij za skrbniško upravljanje spletnega mesta WordPress. V njem so poddirektoriji (npr. `css`, `js`, `network`, `user`) in pomembne datoteke, kot so `admin.php`, `plugins.php` in `themes.php`. Te datoteke omogočajo nalaganje vmesnika za urejanje vsebin, upravljanje uporabnikov, nastavitev tem

in vtičnikov. Direktoriji se neposredno povezuje z delovanjem nadzorne plošče in omogoča dostop do vseh skrbniških funkcionalnosti sistema.

- Direktorij `wp-includes` vsebuje temeljno izvorno kodo sistema WordPress, knjižnice in funkcionalnosti, ki jih uporabljajo drugi deli sistema. Zaradi občutljivosti kode je odsvetovano kakršnokoli ročno spreminjanje vsebine tega direktorija. Izjema je lahko datoteka `functions.php`, vendar tudi pri tej velja, da spremembe raje izvajamo v aktivni temi (več o tem v poglavju 6.5).

- Direktorij `wp-content` vključuje vse podatke, ki jih je možno neposredno upravljati, tj. teme, vtičnike in naložene vsebine. Vsaka tema ali vtičnik ima svoj direktorij znotraj direktorija `themes` oziroma `plugins`, ki vsebuje vse potrebne datoteke za delovanje posamezne teme oziroma vtičnika. Pomemben direktorij znotraj direktorija `wp-content` je tudi `uploads`, kjer se hranijo slike, dokumenti in druge medijske datoteke, strukturirane po letih in mesecih njihovega nalaganja na spletno mesto. Redkeje se zgodi, da so določeni vtičniki (npr. požarni zidovi) nameščeni izven direktorija `wp-content`, saj se morajo naložiti pred drugo vsebino.

- Datoteka `wp-config.php` vsebuje ključne podatke za povezavo z bazo podatkov, varnostne ključe in nastavitve, ki vplivajo na delovanje spletnega mesta (npr. vključitev diagnostike, predpona preglednic ipd.). Zaradi občutljivih informacij je pogosto tarča napadalcev, zato jo je treba ustrezno zaščititi. Čeprav jo lahko ročno uredimo, je priporočljivo uporabljati vtičnike za večino sprememb.

- Datoteka `.htaccess` omogoča nadzor nad strukturo povezav (npr. preusmeritve, trajne povezave, blokade naslovov IP). Deluje zgolj na strežnikih Apache in je privzeto skrita, zato jo je treba v odjemalcu FTP/SFTP posebej prikazati. Uporablja se tudi za napredne varnostne ukrepe, kot je zaščita z geslom, vendar je tudi večino teh funkcionalnosti priporočljivo implementirati z vtičniki.

- Datoteka `index.php` je ena najpomembnejših sistemskih datotek v WordPressu, saj skrbi za prikaz spletnega mesta ob obisku končnega uporabnika. Gre za začetno točko nalaganja, ki sproži vključevanje ustreznih jedrnih datotek in inicializacijo sistema. Ko uporabnik pošlje zahtevo za določeno stran, se datoteka `index.php` aktivira in prevzame vlogo usmerjevalnika, ki določi, katere vsebine in funkcionalnosti bodo prikazane. Pomen te datoteke je še posebej razviden, ko ta v

korenskem direktoriju ni prisotna. V takem primeru se ob obisku spletnega mesta uporabniku v brskalniku prikaže celotna vsebina direktorija, kar predstavlja resno varnostno pomanjkljivost in seveda slabo uporabniško izkušnjo. Iz tega razloga je datoteka `index.php` prisotna ne le v korenskem direktoriju, temveč tudi v številnih poddirektorijih, kot je na primer `wp-content`, kjer preprečuje razkritje internih struktur spletnega mesta. V mnogih poddirektorijih ta datoteka ne vsebuje nobene dejanske programske logike, temveč je prazna z izjemo kratkega komentarja, kot je »Silence is golden.«. Takšne prazne datoteke `index.php` služijo kot zaščitni mehanizem za prikrivanje vsebine direktorija pred javnim dostopom, brez potrebe po kompleksnejših varnostnih pristopih.

Izmed treh sistemskih direktorijev so posegi v direktorij `wp-admin` in direktorij `wp-includes` močno odsvetovani. Spreminjanje jedrnih datotek ni dobra praksa, saj morebitne posodobitve sistema WordPress prepišejo vse spremembe. Kljub temu je možno prilagoditi delovanje sistema WordPress, in sicer v direktoriju `wp-content`, kar bomo spoznali v poglavjih 6–8.

Struktura datotek in direktorijev v WordPressu je zasnovana tako, da ločuje jedro sistema od uporabniških nastavitev in vsebine. Za zagotavljanje stabilnosti in varnosti je priporočljivo, da se spremembe izvajajo izključno prek tem in vtičnikov ter da se jedrne datoteke, z izjemo direktorija `wp-content`, nikoli neposredno ne spreminjajo. Vsak poseg v nastavitvene datoteke naj bo premišljen, spremljan z varnostno kopijo in izveden le, če ni na voljo ustrezne alternativne rešitve v obliki vtičnika [12].

5.2 Shema podatkovne zbirke

Podatkovna zbirka je sestavni del vsake WordPress spletne strani. Skupaj z datotekami sestavlja temeljno infrastrukturo sistema. Medtem ko datoteke definirajo logiko, obliko in delovanje spletnega mesta, relacijska podatkovna baza shranjuje vsebino, ki jo ustvarijo uporabniki. To vključuje prispevke, strani, komentarje, uporabniške račune, nastavitve in drugo. WordPress za to uporablja sistem za upravljanje relacijskih podatkovnih baz MySQL, ki shranjuje podatke v strukturiranih preglednicah. Za obdelavo teh podatkov se uporabljajo ukazi poizvedovalnega jezika SQL. Ob prvi namestitvi WordPressa je

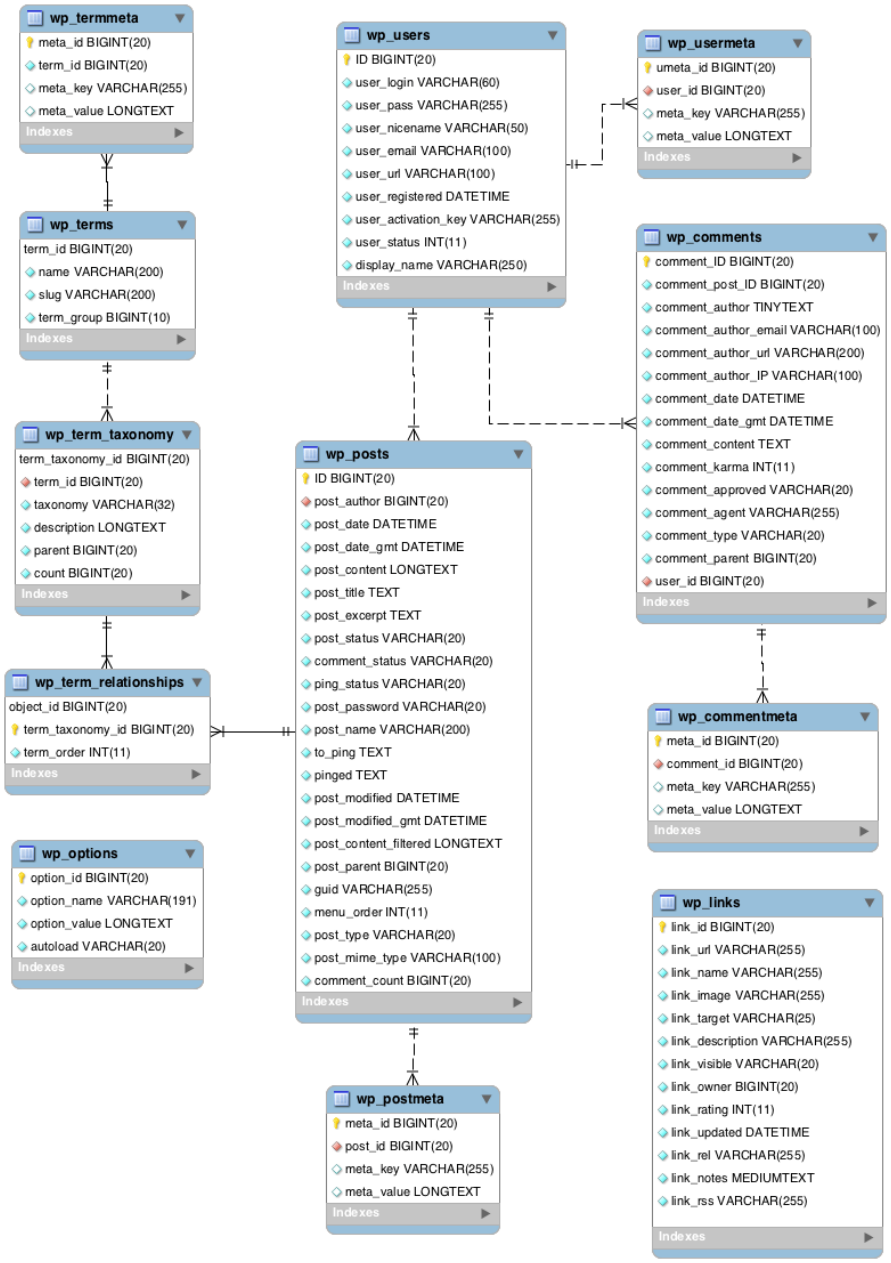
zato treba ustvariti novega uporabnika in podatkovno zbirko, kar omogoča vzpostavitev povezave med spletnim mestom in zbirko podatkov. Vsaka WordPress stran je povezana z eno samo podatkovno zbirko, ki vsebuje več povezanih preglednic [13, 14]. Osnovne (jedrne) preglednice, ki jih WordPress ustvari ob namestitvi, vključujejo [14]:

- `wp_options` – splošne nastavitve strani,
- `wp_users` – uporabniški računi,
- `wp_usermeta` – dodatni podatki o uporabnikih,
- `wp_posts` – prispevki in strani,
- `wp_postmeta` – dodatni podatki o prispevkih,
- `wp_terms`, `wp_term_taxonomy`, `wp_term_relationships` – kategorizacija vsebin s taksonomijami (kategorije, oznake),
- `wp_comments` – komentarji,
- `wp_commentmeta` – dodatni podatki o komentarjih,
- `wp_links` – podatki o zunanjih povezavah (v starejših namestitvah).

Osnovna shema podatkovne zbirke je prikazana na entitetno-relacijskem diagramu na sliki 5.1. Sčasoma se zbirka podatkov razširi, saj lahko vsak nameščen vtičnik doda nove preglednice in vnose. Na primer, spletne trgovine, ki uporabljajo vtičnik WooCommerce, razdelijo podatke o naročilih, kupcih in izdelkih med več preglednic. Pomembno je razumeti, da preglednice pogosto delujejo v soodvisnosti – podatki se povezujejo z relacijami med preglednicami. Na primer: za izpis komentarjev WordPress uporablja tako preglednico `wp_comments` kot `wp_commentmeta` [13, 14].

5.3 Hierarhija predlog klasičnih tem

Videz spletnega mesta v WordPressu temelji na uporabi tem, pri čemer razlikujemo med klasičnimi temami in blokovnimi temami. Klasične teme se zanašajo na t. i. poizvedbeni niz (angl. Query String), ki določa, katera predloga oziroma kombinacija predlog v hierarhiji datotek PHP bo uporabljena za prikaz določene vsebine. Poizvedbeni niz je



Slika 5.1. Entitetno-relacijski diagram podatkovne zbirke osnovne WordPress strani [14].

informacija, vključena v povezavo URL posamezne podstrani, in predstavlja osnovo za prepoznavanje tipa vsebine, ki jo uporabnik zahteva (npr. kategorija, iskanje, prispevek, stran). Osrednji mehanizem, ki upravlja prikaz vsebine v WordPressu, je hierarhija predlog (angl. Template Hierarchy). Sistem deluje tako, da WordPress na osnovi vrste poizvedbe določi, katera predloga naj se uporabi, in nato išče ustrezne datoteke v direktoriju aktivne teme, dokler ne najde prve ustrezne predloge. Če ustrezne predloge

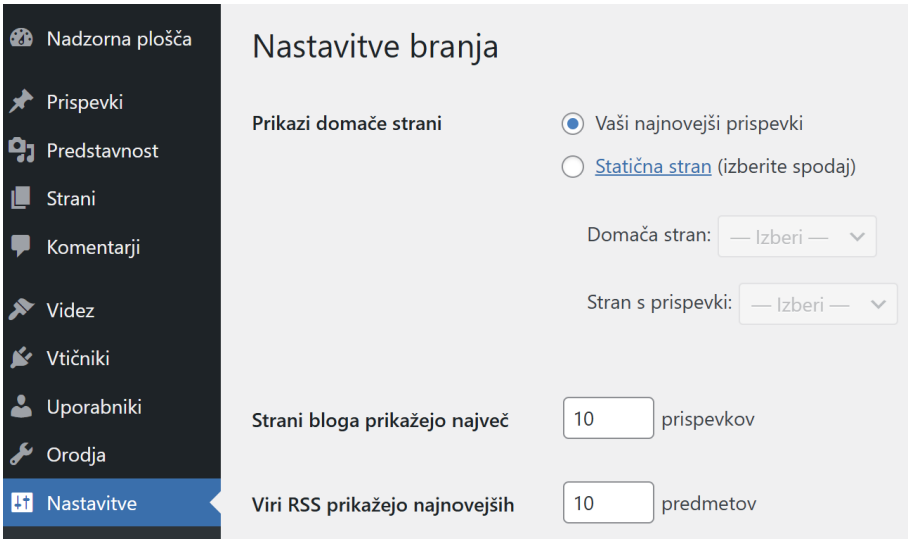
ni mogoče najti, se uporabi osnovna predloga `datoteka!index.php`, ki mora biti prisotna v vsaki temi [15]. Diagram na sliki 5.2 prikazuje, katere predloge se uporabijo za prikaz WordPress strani glede na pravila hierarhije predlog [15].
Primer opisuje uporabo zaporedja predlog, ki so vključene pri prikazu posamezne strani. Če je stran WordPress nameščena na naslovu `http://primer.si/blog/` in obiskovalec klikne povezavo do določene kategorije, npr. `http://primer.com/blog/kategorija /glavne-jedi/`, WordPress začne iskati ustrezno predlogo v aktivni temi. Najprej se preveri, ali obstaja datoteka, ki se ujema s kratkim imenom kategorije (angl. Slug), torej v tem primeru `category-glavne-jedi.php` ali `category-sladice.php`, če je takšno ime kategorije. Če predloge z imenom kategorije ni, WordPress preveri, ali obstaja datoteka, ki se ujema z ID-jem kategorije – npr. `category-4.php`, če je ID kategorije 4. Če tudi ta ne obstaja, sistem nadaljuje iskanje splošne predloge za kategorije `category.php`. Če niti ta datoteka ni na voljo, WordPress uporabi bolj generično predlogo za arhivirane vsebine – `archive.php`. Če še vedno ni mogoče najti ustrezne predloge, WordPress na koncu uporabi osnovno predlogo `index.php`, ki je obvezna v vsaki temi. Tak način iskanja predlog omogoča visoko stopnjo prilagodljivosti pri oblikovanju posameznih tipov strani, hkrati pa zagotavlja, da bo vsebina vedno prikazana, tudi če specifične predloge manjkajo.

5.3.1 Predloge domače strani

Enak pristop hierarhije se upošteva tudi pri izbiri predloge za prikaz najpomembnejše strani – domače strani. Kot je prikazano na sliki 5.2, se v WordPressu za prikaz vstopne strani najprej uporabi predloga `front-page.php`, ne glede na to, ali je za domačo stran uporabljena nastavitev statične strani ali prikaza zadnjih prispevkov. Če ta datoteka ne obstaja, WordPress glede na nastavitve v zavihku Nastavitve → Branje (prikazane na sliki 5.3) uporabi datoteke predlog v naslednjem vrstnem redu [15]:

1. `home.php` – če je nastavljeno prikazovanje zadnjih prispevkov,
2. `page.php` – če je nastavljena statična začetna stran,
3. `index.php` – če prejšnje predloge ne obstajajo.

Slika 5.2. Hierarhija predlog (datotek), uporabljenih za prikaz posamezne strani.



Slika 5.3. Nastavitve domače strani.

5.3.2 Predloge prispevka

Podobno kot v predhodnih primerih se za prikaz posameznega prispevka uporablja datoteka predloge za posamezen prispevek (angl. Single Post Template), pri čemer sledi hierarhični poti od najbolj specifične do najbolj splošne predloge. Sistem predloge upošteva v naslednjem vrstnem redu [15]:

1. `single-post-type-slug.php` – najprej se preveri, ali obstaja predloga prikaza za točno določen prispevek. Na primer, če gre za prispevek tipa izdelek s kratkim imenom `cokoladni-piskoti-12`, bo iskana predloga `single-recept-cokoladni-piskoti-12.php`.

2. `single-post-type.php` – če predloga `single-cokoladni-piskoti-12.php` ne obstaja (kar je pogost scenarij, saj razvijalci običajno ne določajo ločenega oziroma individualnega izgleda strani individualnih prispevkov), WordPress išče predlogo za določen tip prispevka, npr. `single-recept.php`. Ta predloga se pogosto uporablja, tudi kadar so dodani lastni tipi prispevkov. Za dodan lasten tip prispevka recept, bi na primer dodali predlogo `single-recept.php`.

3. `single.php` – če tudi zgornja predloga ni na voljo, WordPress nato preide na splošno predlogo za posamezen prispevek, ne glede na njen tip.

4. `singular.php` – če tudi predloga `single.php` ni na voljo, preveri še nekoliko bolj generično predlogo `singular.php`, ki se uporablja za vse posamezne vsebine

(prispevki, strani ipd.).

5. `index.php` – če nobena od zgornjih predlog ni prisotna, sistem na koncu uporabi osnovno predlogo `index.php`. Uporaba te predloge je sicer zadnja rešitev in vsebino prikazuje v precej omejenem obsegu (npr. ne prikazuje vseh metapodatkov recepta Čokoladni piškoti, kot so na primer zahtevnost, sestavine, čas peke).

Opisana hierarhična logika iskanja predlog ne velja le za opisane primere prikaza posameznega prispevka in domače strani, temveč je na enak način uporabljena tudi pri prikazu vseh drugih vrst vsebin v WordPressu, od arhivov kategorij in oznak do posameznih prispevkov z lastnimi tipi prispevkov ter iskalnih rezultatov ali strani z napakami. Na sliki 5.2 je shematično prikazan celoten potek iskanja predlog glede na vrsto poizvedbe, medtem ko so vsi primeri in posebnosti vrstnega reda uporabe predlog podrobneje opisani v uradni dokumentaciji [15].

5.4 Hierarhija predlog blokovnih tem

V nasprotju s klasičnimi temami, kjer strukturo določajo datoteke PHP in predloge, so blokovne teme zasnovane na sistemu blokov. Vsak element spletnega mesta je predstavljen kot blok ali skupek blokov, kar omogoča izjemno prilagodljivost pri oblikovanju. Urejanje in prilagajanje poteka neposredno v urejevalniku spletnega mesta, ki temelji na konceptu urejanja celotnega spletnega mesta (angl. Full Site Editing ali FSE). Hierarhija blokovne teme je organizirana po velikosti, od posameznih gradnikov, predlog in delov predlog do celotne teme [16, 17]:

- Blok (angl. Block) – osnovni gradniki vsebine, kot so odstavek, slika, gumb ali skupina.
- Vzorec bloka (angl. Block Pattern) – vnaprej oblikovana kombinacija blokov, ki jo lahko znova uporabimo in prilagodimo.
- Deli predloge (angl. Template Parts) – ponovljiv del predloge, namenjen strukturalnim elementom, npr. glavi, nogi ali stranski vrstici.
- Predloga (angl. Template) – določa celotno postavitev za določeno vrsto vsebine (npr. stran, prispevek, arhiv).

- Tema (angl. Theme) – najvišja raven, ki vključuje vse prejšnje elemente in določa celoten videz, slog in vedenje spletnega mesta.

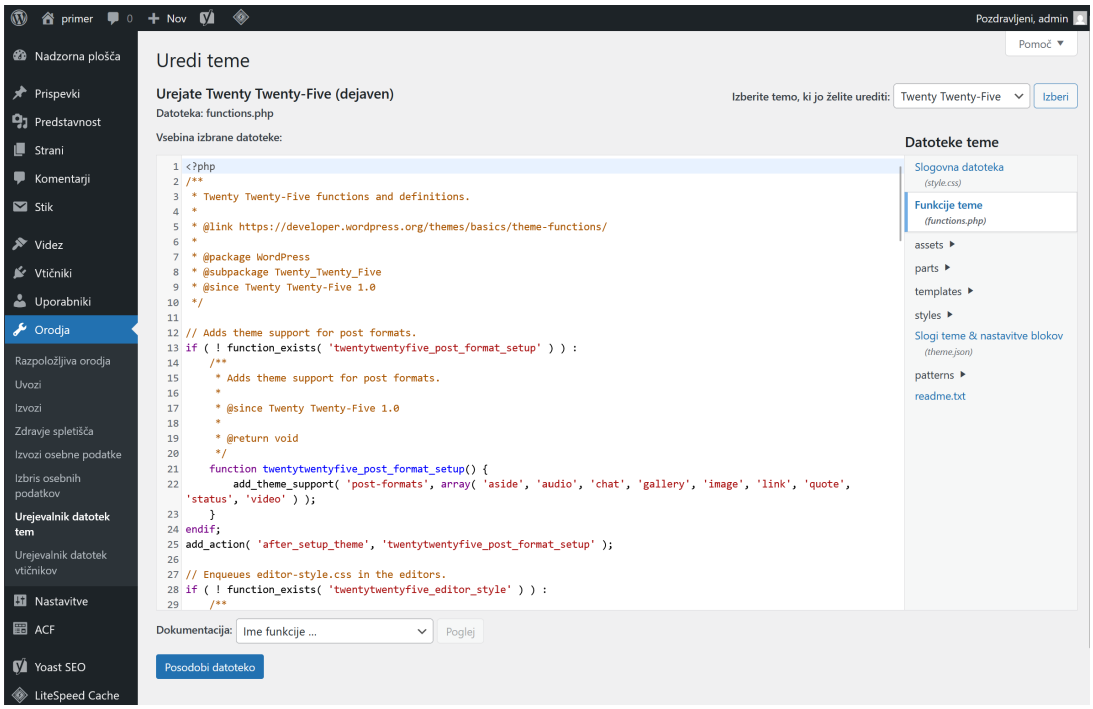
Blokovni pristop je zasnovan na modularnem oblikovanju, katerega osnovna ideja je oblikovanje spletnega mesta iz manjših blokov, ki so pripravljeni za ponovno uporabo. Bloki delujejo kot gradniki, ki jih je mogoče kombinirati v vzorce ali ponovno uporabne komponente, kot so odseki za predstavitev izdelkov, obrazci za prijavo ali komentarji strank. Bloke je mogoče prilagajati z datoteko `theme.json` ali vizualno z orodjem globalnih slogov. Ključna načela sestavljanja blokov vključujejo začetek z osnovnimi elementi (kot so besedilo, slike, gumbi), postopno gradnjo kompleksnejših struktur in odzivno oblikovanje, ki zagotavlja pravilno prikazovanje na vseh napravah. Cilj je ustvarjanje prilagodljivega sistema, ne posamezne strani.
Datoteka `theme.json` ima ključno vlogo pri upravljanju nastavitev in slogov blokovne teme. Omogoča nadzor nad barvami, tipografijo, razmiki, videzom posameznih blokov ter registracijo predlog in delov predlog. V hierarhiji nastavitev imajo pri blokovnih temah prednost uporabniške spremembe, sledijo podrejena tema, glavna tema in privzete WordPress nastavitve [17, 18]. Primer datoteke `theme.json` in njenega prilagajanja je podrobneje predstavljen v poglavju 6.2.

5.5 Dostop do datotečnega sistema WordPress in podatkovne zbirke

Kadar spletno stran WordPress upravljamo v lokalnem razvojnem okolju, je dostop do datotečnega sistema WordPress možen preprosto prek raziskovalca računalnika. Pogosto pa je spletna stran WordPress nameščena na strežniku spletnega gostitelja, kar pomeni, da so vse datoteke in podatkovna zbirka fizično shranjene na oddaljenem računalniku. Za upravljanje, vzdrževanje ali odpravljanje napak na strani je pogosto potreben neposreden dostop do teh virov. Obstaja več načinov, kako do njih dostopati, vsak s svojimi prednostmi in slabostmi. Izbira metode je odvisna od tehničnega znanja, vrste gostovanja in specifičnih potreb uporabnika.

5.5.1 Dostop prek nadzorne plošče WordPress

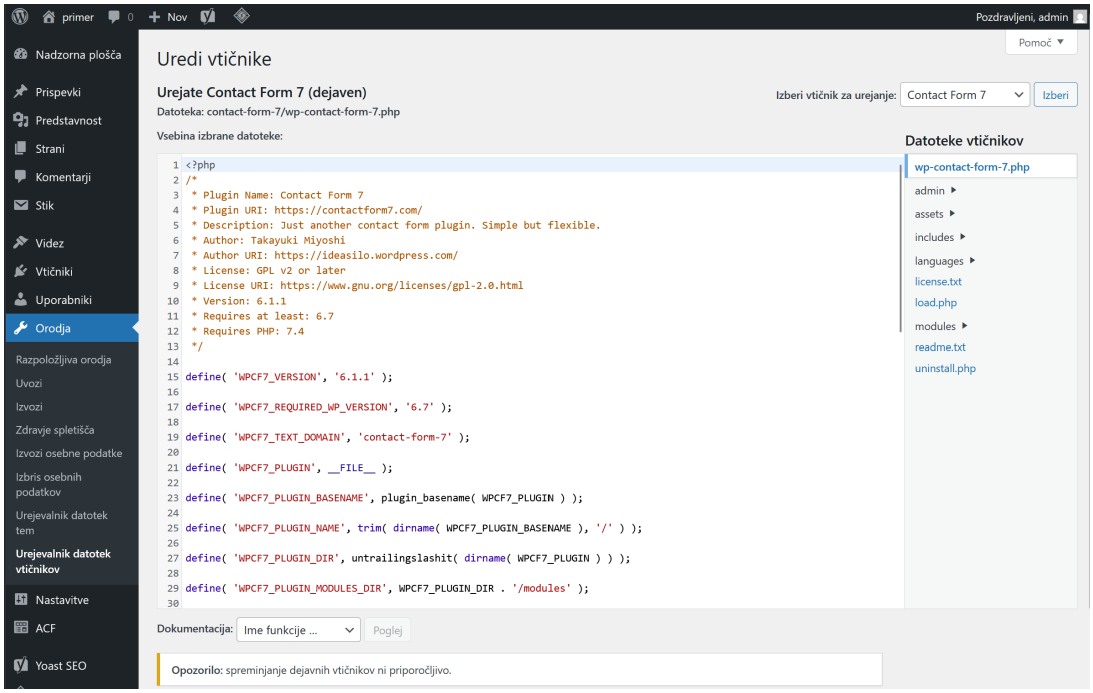
Do določenih datotek je možno privzeto dostopati s skrbniškim vmesnikom WordPress. Slednje omogočata urejevalnik datotek tem (angl. Theme File Editor) in urejevalnik datotek vtičnikov (angl. Plugin File Editor), ki ju je v primeru uporabe blokovnih tem možno najti pod zavihkom Orodja (angl. Tools), kot prikazuje slika 5.4 in slika 5.5. V primeru uporabe klasičnih tem sta pod zavihkom Videz (angl. Appearance) in Vtičniki (angl. Plugins).



Slika 5.4. Urejevalnik datotek tem v nadzorni plošči WordPress.

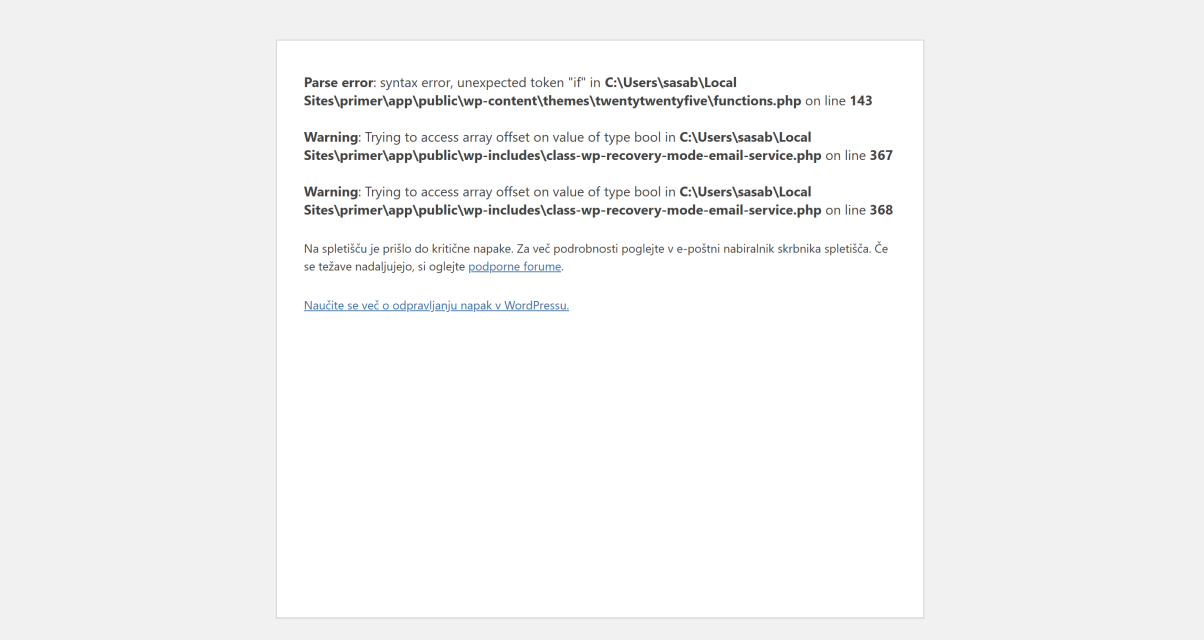
Neposredno urejanje teme ali vtičnika znotraj nadzorne plošče WordPressa ni priporočljiva praksa. Neposredno urejanje lahko namreč okvari spletno mesto, kar onemogoči dostop do nadzorne plošče ter s tem do urejevalnika teme ali vtičnika znotraj nje.

Prek nadzorne plošče WordPress je prav tako možno dostopati do celotnega datotečnega sistema in podatkovne baze, in sicer z uporabo vtičnikov za dostop do datotek WordPress in podatkovne zbirke. Takšen način omogoča neposreden pregled in upravljanje strežniških datotek kar znotraj nadzorne plošče WordPress, brez potrebe po dodatni prijavi ali uporabi zunanjih orodij. Med najbolj priljubljenimi rešitvami je vtičnik



Slika 5.5. Urejevalnik datotek vtičnikov v nadzorni plošči WordPress.

WP File Manager, ki po namestitvi omogoča pregled celotne strukture WordPress datotek v vizualnem vmesniku. Njegova uporaba je preprosta in dostopna tudi manj tehnično podkovanim uporabnikom [13]. Podoben pristop je možen tudi pri delu z bazo podatkov. Vtičniki, kot sta Database Admin in WP Adminer, omogočajo vpogled v strukturo podatkovne zbirke brez ročne prijave v orodja, kot je phpMyAdmin. Čeprav so ti vtičniki funkcionalno nekoliko omejeni in imajo osnovnejši vmesnik, predstavljajo uporabno rešitev za osnovne posege, kot so vpogledi v preglednice ali hitra diagnostika. Glavna prednost takšnega pristopa je preprostost uporabe in dostopnost neposredno iz nadzorne plošče WordPress. Vendar je to hkrati tudi njegova ključna slabost – v primeru resnejših napak in okvar na strani dostop do nadzorne plošče ni možen, kar pomeni, da mora skrbnik uporabiti alternativen način dostopa do datotek (npr. z uporabo protokola FTP/SFTP). Primer preprečenega dostopa do nadzorne plošče zaradi sintaktične napake v datoteki `functions.php` je prikazan na sliki 5.6. Kljub temu so vtičniki za upravljanje datotek in podatkovne zbirke ob precejšnji meri pazljivosti učinkovita rešitev za majhna opravila, zlasti pri upravljanju strani brez dostopa do strežniških orodij [13].



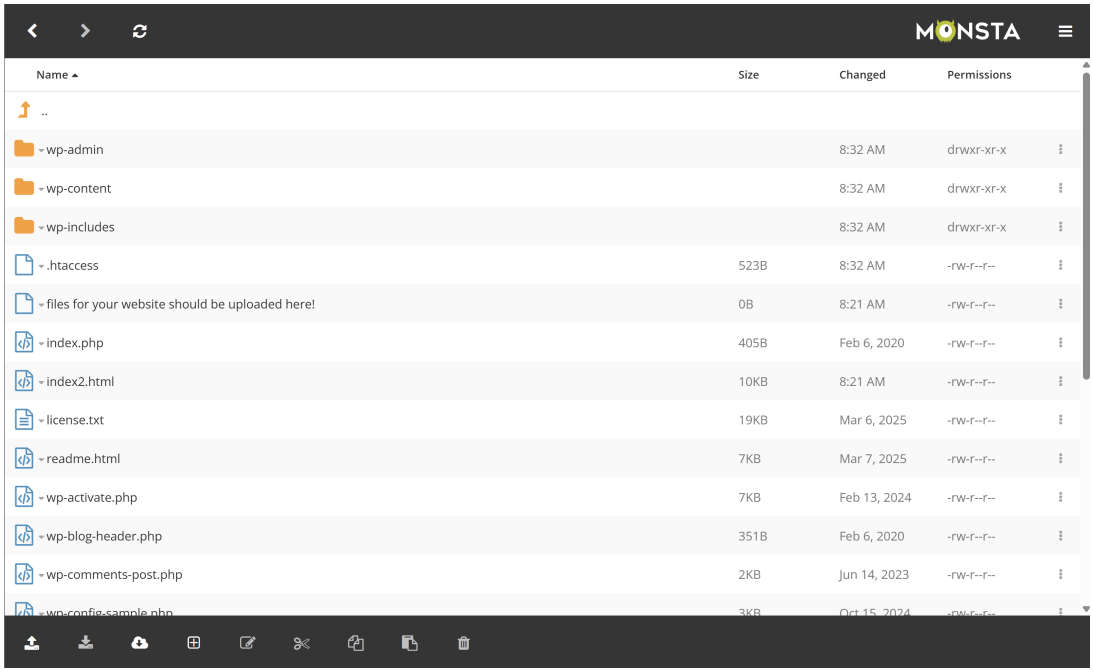
Slika 5.6. Onemogočen dostop do nadzorne plošče (in vtičnika za urejanje) zaradi sintaktične napake.

5.5.2 Dostop prek nadzorne plošče cPanel

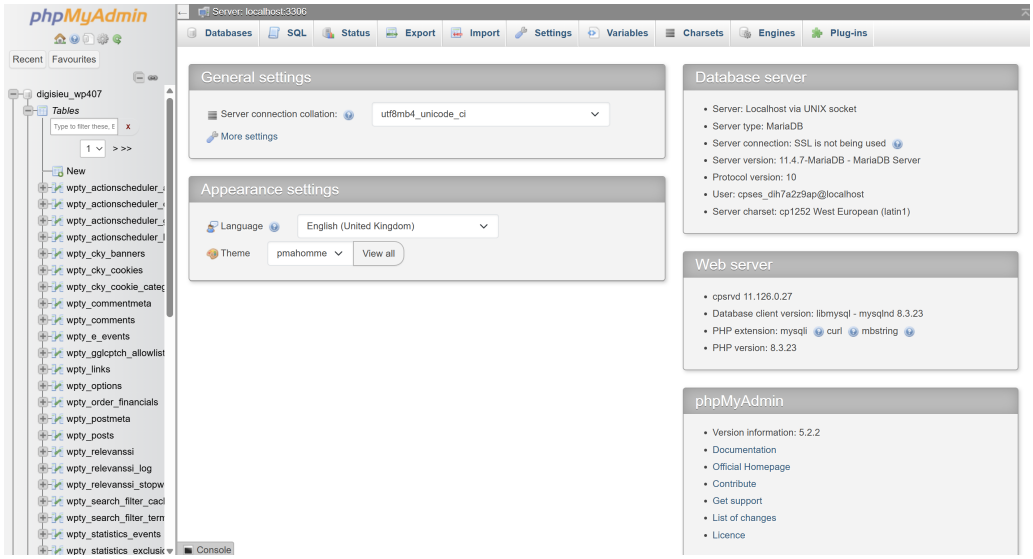
Najpogosteje uporabljeno orodje za dostop do datotek WordPress in baze podatkov je z uporabo cPanel [19] – nadzorne plošče, ki jo ponuja večina ponudnikov gostovanja. V cPanelu sta za ta namen ključni orodji (Online) File Manager za upravljanje datotek in phpMyAdmin za delo z bazo podatkov. Dostop do cPanela je odvisen od ponudnika gostovanja – najpogosteje je možen dostop do cPanela prek naslova URL v obliki `ime-domene.si/cpanel`, kjer se prijavimo z ločenimi cPanel dostopnimi podatki (ne z WordPress dostopnimi podatki). Znotraj cPanela (kot prikazano na sliki 5.7) lahko z (Online) File Managerjem pregledujemo datoteke WordPress strani, kot bi jih v lokalnem raziskovalcu. Datoteke lahko prenesemo, lokalno uredimo in nato ponovno naložimo na strežnik [13].

Za neposredno delo s podatkovno bazo se pogosto uporabi orodje phpMyAdmin, ki omogoča pregled in urejanje preglednic podatkovne zbirke. Če gostitelj ne omogoča samodejne prijave, so podatki za prijavo v datoteki `wp-config.php`. Pogled podatkovne baze v phpMyAdmin je prikazan na sliki 5.8.

Nekateri ponudniki gostovanja uporabljajo alternativo cPanelu, kot sta Plesk in Webmin, ali pa imajo lasten uporabniški vmesnik z gumbom za dostop do podatkovne zbirke. V takšnih primerih se za delo s podatkovno bazo lahko uporablja orodje Adminer, ki opravlja



Slika 5.7. Prikaz datotečne strukture WordPress projekta – cPanel.

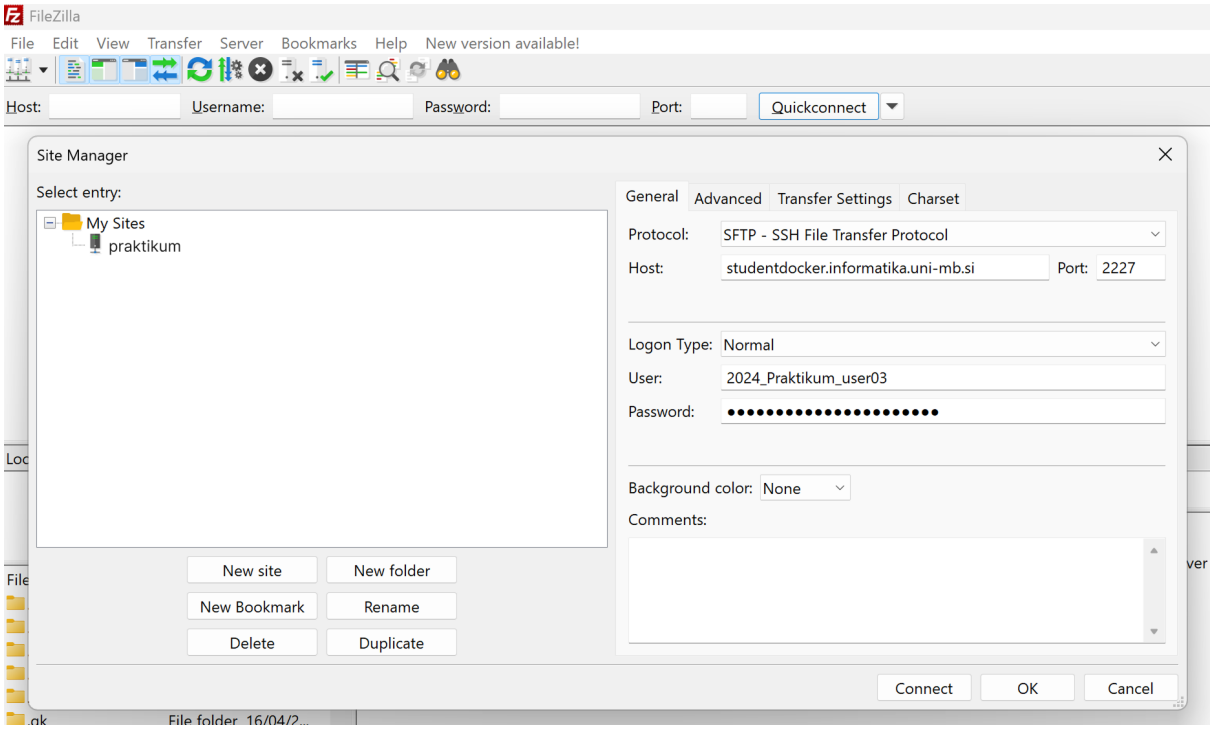


Slika 5.8. Prikaz sheme podatkovne baze WordPress projekta – phpMyAdmin.

podobno funkcijo kot phpMyAdmin [13].

5.5.3 Dostop prek protokola FTP/SFTP

Protokol FTP (angl. File Transfer Protocol) je eden starejših in najpogosteje uporabljenih protokolov za prenos datotek med strežniki in odjemalci. V nasprotju s protokolom HTTP, ki se uporablja za prikaz spletnih vsebin, je protokol FTP namenjen neposrednemu prenosu datotek, pri čemer zahteva avtentikacijo z uporabniškim imenom in geslom. Zaradi varnostnih razlogov se danes pogosteje uporablja varna različica, t. i. protokol SFTP (angl. SSH File Transfer Protocol), ki omogoča šifrirano komunikacijo med odjemalcem in strežnikom [20]. Za uporabo protokola FTP ali protokola SFTP je treba uporabljati poseben program (t. i. odjemalec FTP/SFTP). FileZilla [21] je eden izmed popularnih odprtokodnih odjemalcev, ki omogoča preprost, hiter in zanesljiv dostop do strežnikov. Program je na voljo za večino operacijskih sistemov (Windows, macOS, Linux) in ponuja grafični uporabniški vmesnik, ki je primerljiv z raziskovalcem datotek v operacijskem sistemu. Postopek vzpostavitve povezave z uporabo orodja FileZilla je prikazan na sliki 5.9.



Slika 5.9. Dostop do datotečne strukture WordPress projekta na strežniku z orodjem FileZilla.

5.5.4 Dostop z ukazno vrstico in protokolom SSH

Protokol SSH (angl. Secure Shell) je napreden varnostni protokol, ki omogoča varen in neposreden dostop do oddaljenih strežnikov z ukazno vrstico. V okolju WordPress se uporablja predvsem za skrbniške naloge, kot so nadzor strežniških datotek, nameščanje posodobitev, brisanje vtičnikov in izvajanje diagnostičnih postopkov [22]. Za razliko od grafičnih vmesnikov, kot sta cPanel in FileZilla, dostop prek protokola SSH ne ponuja vizualnega pregleda nad datotekami, temveč zahteva osnovno znanje uporabe ukazne vrstice. Tak pristop poveča zahtevnost uporabe, a omogoča tudi bistveno večjo prilagodljivost in zmogljivost. Ena izmed ključnih prednosti protokola SSH v okolju WordPress je možnost uporabe orodja WP-CLI (angl. WordPress Command Line Interface) [23], ki omogoča izvajanje številnih za WordPress specifičnih ukazov brez potrebe po grafičnem vmesniku. Z njim lahko upravljamo uporabnike, urejamo nastavitve, posodabljamo vtičnike in teme ter izvajamo postopke varnostnega kopiranja ali diagnostike. Takšen pristop je posebej uporaben v okoljih, kjer je učinkovitost pomembnejša od preprostosti uporabe ali kjer spletno gostovanje ne vključuje naprednejših grafičnih orodij. Protokol SSH z uporabo ukazne vrstice tako predstavlja primerno orodje za razvijalce in sistemske skrbnike, ki želijo popoln nadzor nad okoljem WordPress [13].

POGLAVJE 6

Teme in vizualno urejanje

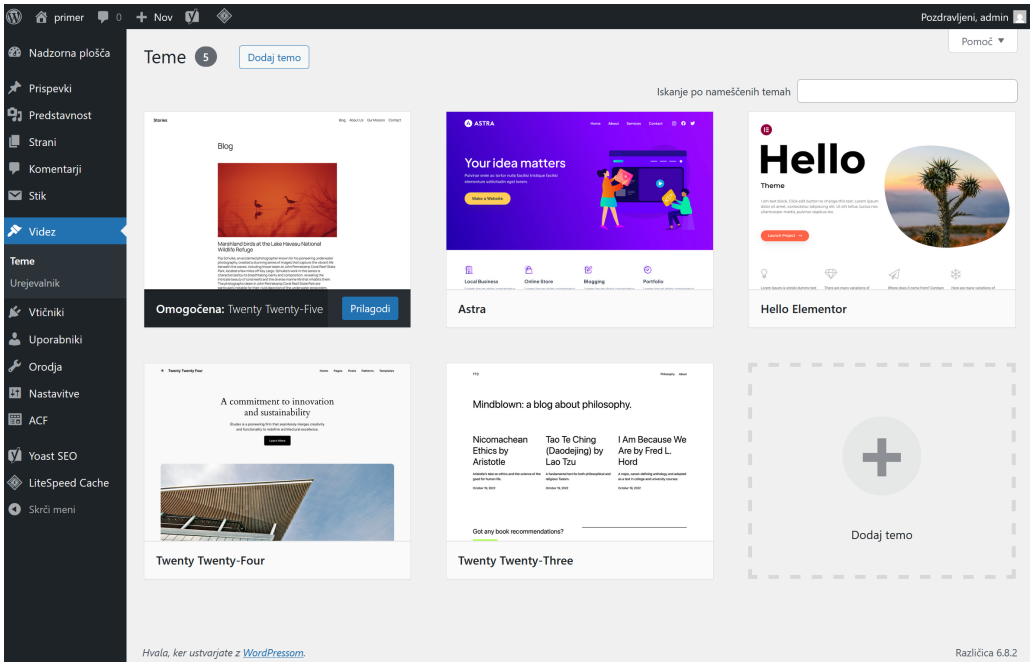
V predhodnih poglavjih so bile teme že večkrat omenjene kot mehanizem, s katerim v WordPressu upravljamo z vizualnim izgledom spletnega mesta. V tem poglavju bomo podrobneje predstavili dva glavna tipa tem, in sicer klasične in blokovne teme. Osredotočili se bomo na predstavitev, kako in med kakšnimi temami lahko izbiramo, predstavitev lastnosti posameznih tipov tem in načinov urejanja videza z njimi. Predstavili bomo tudi vtičnike za vizualno izdelavo strani in koncept podtem.

Kaj je tema? Tema v WordPressu je zbirka datotek, ki določa videz, postavitev in včasih tudi delovanje spletne strani. Danes poznamo predvsem dve vrsti tem, in sicer klasične teme in blokovne teme. Teme vsebujejo številne datoteke, kot so PHP/HTML, JavaScript, CSS, slike in dokumentacija. Uporabimo lahko brezplačne ali plačljive obstoječe teme. Teme lahko razvijemo tudi sami. Videz spletne strani, ki ga določa izbrana klasična ali blokovna tema, lahko dodatno prilagajamo na več načinov – z uporabo vgrajenega urejevalnika Gutenberg, vtičnikov za vizualno oblikovanje strani (angl. Page Builders) in z razvojem lastne podteme.

Pogosto prihaja do prepletanja med funkcionalnostmi tem in vtičnikov, zato velja dobra praksa: tema naj skrbi za videz in predstavitev vsebine, vtičniki pa za nadzorovanje vedenja in funkcionalnosti spletne strani. Tema naj ne vsebuje ključnih funkcionalnosti – če temo zamenjamo, jih namreč v tem primeru izgubimo.

6.1 Izbira teme

Brezplačne teme najdemo v uradnem katalogu WordPressa, ki je na voljo na povezavi https://wordpress.org/themes/ ali znotraj skrbniškega vmesnika v zavihku Videz → Teme. Uporaba brezplačnih tem neposredno iz uradnega kataloga WordPress predstavlja preprost in dostopen način za določitev izgleda spletne strani. Njihova glavna prednost je, da so brezplačne in na voljo v velikem številu, kar omogoča hitro izbiro vizualne podobe brez dodatnih stroškov. Kljub temu imajo brezplačne teme tudi pomanjkljivosti – ni zagotovila, da bodo dolgoročno vzdrževane ali imele redne posodobitve, kar lahko vpliva na varnost in združljivost z novejšimi različicami WordPressa. Velikost tem se močno razlikuje, kar lahko vpliva na hitrost nalaganja strani. Poleg tega lahko ob večjih posegih ali prilagoditvah pride do konfliktov z drugimi vtičniki ali gradniki.
Uporabimo lahko tudi plačljive teme, ki jih najdemo na spletnih tržnicah. Nakup plačljive teme prek platform, kot sta Theme Forest (posamični nakupi, običajno 30–70 USD) ali Envato Elements (mesečna oziroma letna naročnina), omogoča hiter in profesionalen začetek pri postavitvi izgleda spletne strani. Plačljive teme pogosto vključujejo vnaprej pripravljene predloge, sodoben in odziven izgled ter možnost uvoza vzorčnih vsebin, kar olajša predstavo o končnem izgledu strani. Prednosti plačljivih tem so tudi redno vzdrževanje, podpora in pogoste posodobitve, ki prispevajo k boljši varnosti in združljivosti. Vendar so te teme pogosto obremenjene z dodatnimi funkcionalnostmi in vsebinami, ki morda niso potrebne, kar lahko vpliva na zmogljivost in velikost strani. Optimizacija je običajno zahtevnejša kot pri namensko razviti ali minimalistični temi. Ob večjih prilagoditvah se lahko pojavijo konflikti z vtičniki za vizualno izdelavo strani ali drugimi vtičniki, zato je predhodna združljivost ključna.
Na sliki 6.1 je prikazana zbirka WordPress s temami in seznam nameščenih tem, kjer so nekatere klasične, vendar združljive z vtičniki za urejanje strani (npr. Astra), druge pa temeljijo na blokovni arhitekturi (npr. Twenty Twenty-Five).
Povzetek primerjave različnih vrst WordPress tem je predstavljen v preglednici 6.1, in sicer klasičnih tem, »lahkih« klasičnih tem kot posebne podvrste klasičnih tem, namenjene vizualnemu urejanju z uporabo vtičnikov za vizualno izdelavo strani, in blokovnih tem. Zraven značilnosti posamezne vrste teme so navedeni še vidnejši predstavniki.



Slika 6.1. Izbira teme – WordPress knjižnica tem.

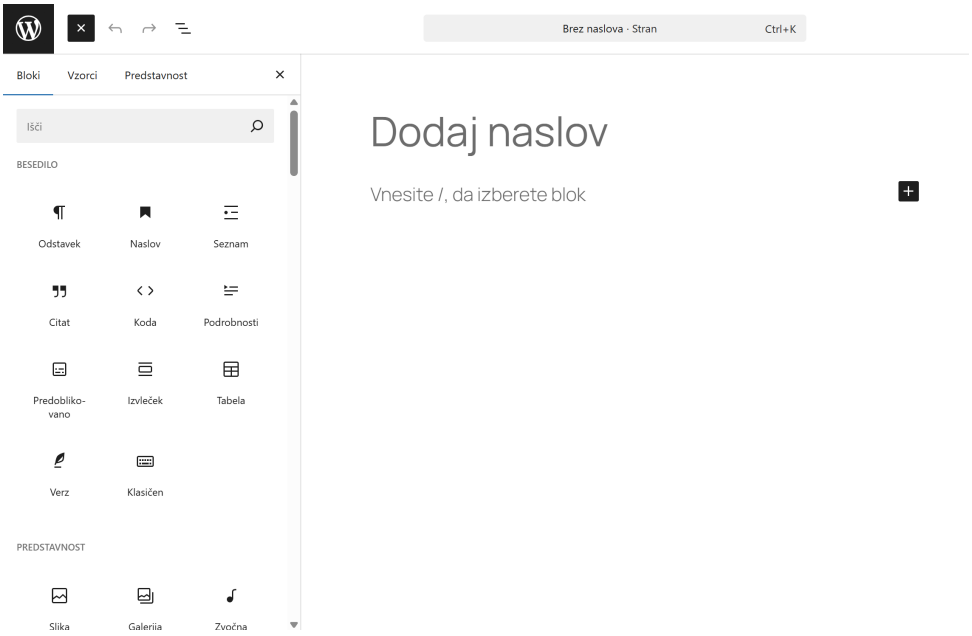
Preglednica 6.1. Vrste WordPress tem in njihove značilnosti.

Značilnost	Klasična tema	»Lahka« klasična tema	Blokovna tema
Opis	Temelji na datotekah predlog, kot so `index.php` in `style.css`	Temelji na datotekah predlog, kot so `index.php` in `style.css`, a z minimalno strukturo, zasnovano za uporabo z vtičniki za vizualno izdelavo strani	Temelji na blokovni arhitekturi in datoteki `theme.json`; omogoča celotno urejanje spletnega mesta
Način urejanja	Klasični urejevalnik, vmesnik brez blokov	Uporaba vtičnikov za vizualno izdelavo strani (npr. Elementor, WPBakery)	Vgrajen urejevalnik Gutenberg
Primeri	Twenty Seventeen, Twenty Twenty-One	Hello Elementor, Astra (v kombinaciji z Elementorjem)	Twenty Twenty-Four, Twenty Twenty-Five

Pred izbiro (ali nakupom) teme je priporočljivo preveriti, ali je tema združljiva z izbranim vtičnikom za vizualno izdelavo strani, če načrtujemo njegovo uporabo.

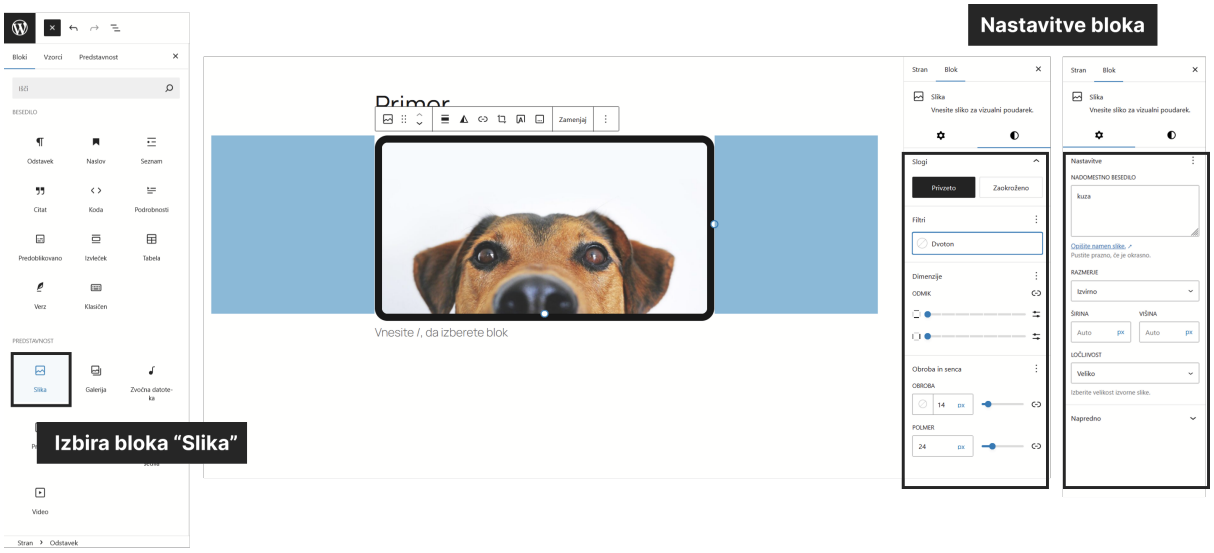
6.2 Urejevalnik Gutenberg in blokovne teme

Gutenberg je privzeti in vgrajeni blokovni vizualni urejevalnik WordPressa (angl. WordPress Block Editor). Predstavlja sodobno uredniško okolje, zasnovano na konceptu blokov. Vsak del vsebine (naslovi, odstavki, slike, galerije, obrazci ipd.) je predstavljen kot blok, kar omogoča modularno in intuitivno gradnjo spletnih strani. Pogled dodajanja blokov na stran je prikazan na sliki 6.2. Bloke je mogoče enostavno vstavljati, premikati, preurejati, podvajati ali pretvoriti v druge tipe blokov. Ta arhitektura spodbuja ponovno uporabo vsebine (angl. Reusable Blocks) ter omogoča uporabo vnaprej pripravljenih vzorcev (angl. Patterns), ki olajšajo sestavljanje vizualno usklajenih vsebin.



Slika 6.2. Struktura blokov, vidna pri urejanju strani.

Primer dodajanja preprostega bloka in upravljanja z njegovimi nastavitvami je prikazan na sliki 6.3. Vsak blok ima določene nastavitve, ki so odvisne od izbranega bloka.
Poleg osnovnih funkcionalnosti urejevalnik podpira tudi gradnjo predlog in delov predlog, s čimer je omogočeno oblikovanje celotne strukture spletnega mesta brez pisanja kode. Gutenberg temelji na serializaciji podatkov z uporabo atributov, ki se zapisujejo v



Slika 6.3. Primer dodajanja bloka »Slika« in upravljanja z njegovimi nastavitvami.

komentarje HTML, kar omogoča tako statične kot dinamične bloke. Z variacijami blokov je mogoče ponujati več vnaprej določenih nastavitev enega bloka, kar razvijalcem olajša ustvarjanje kompleksnejših rešitev. Celoten sistem dopolnjuje možnost oblikovanja in nastavitve slogov prek globalnih slogov (angl. Styles), ki se definirajo v datoteki `theme.json`, kar dodatno krepi usklajenost in prilagodljivost tem. Gutenberg tako poenostavlja urejanje vsebin, hkrati pa omogoča napredne zmožnosti za razvijalce in urejevalce vsebin [24].

Primer datoteke `theme.json`, v kateri je mogoče urejati nastavitve teme, temelječe na blokih, je prikazan na primeru 6.2.1. Osnovno ogrodje datoteke `theme.json` vključuje ključne sklope: nastavitve (`settings`), sloge (`styles`), predloge (`customTemplates`), dele predlog (`templateParts`) in vzorce (`patterns`). Vsak od teh delov ima svojo vlogo pri definiranju vizualne identitete in strukture spletnega mesta. V tej datoteki je mogoče določiti privzete nastavitve za vizualne sloge, kot so barvne palete, tipografija, razmiki in možnosti orodij za urejanje. Struktura datoteke temelji na sintaksi JSON. V nastavitvah, predstavljenih v primeru, so definirane tri osnovne barve s pripadajočimi identifikatorji (npr. `"slug": "base"`), kar omogoča njihovo dosledno uporabo v celotni temi. Poleg tega so možnosti, kot so `defaultDuotone`, `defaultGradients` in `defaultPalette`, onemogočene, kar pomeni, da tema ne uporablja privzetih slogov WordPressa, temveč lastne. Spreminjanje teh nastavitev omogoča večji nadzor nad oblikovanjem in zagotavlja skladno uporabniško izkušnjo v okviru vizualne identitete spletnega mesta.

Primer 6.2.1 (Primer datoteke `theme.json`.)

```
{
  "$schema": "https://schemas.wp.org/wp/6.7/theme.json",
  "version": 3,
  "settings": {
    "appearanceTools": true,
    "color": {
      "defaultDuotone": false,
      "defaultGradients": false,
      "defaultPalette": false,
      "palette": [
        {
          "color": "#FFFFFF",
          "name": "Base",
          "slug": "base"
        },
        {
          "color": "#111111",
          "name": "Contrast",
          "slug": "contrast"
        },
        {
          "color": "#FFEE58",
          "name": "Accent 1",
          "slug": "accent-1"
        },
        ...
      ]
    }
  },
  "styles": {},
  "customTemplates": {},
  "templateParts": {},
  "patterns": []
}
```

Urejevalnik, prikazan na sliki 6.4, omogoča urejanje navigacije (menijev), stilov (globalnih barv, tipografij, nastavitev posameznik blogov ipd.), strani, predlog (prispevka, arhiva prispevkov, rezultatov iskanja, strani 404 ipd.), vzorcev in delov predlog (glava, noga ipd.).

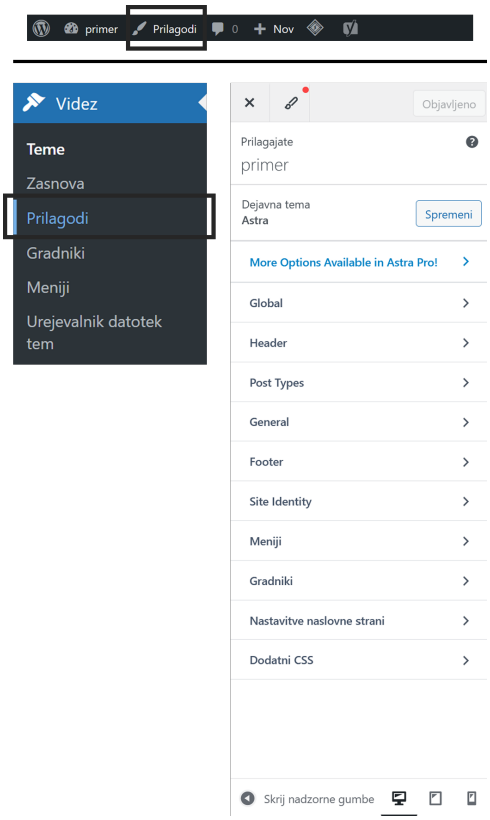
Slika 6.4. Urejevalnik Gutenberg.

Gutenberg je bil prvotno predstavljen v WordPressu različice 5.0 (december 2018) in predstavlja pomemben premik k enostavnejšemu in vizualno usmerjenemu urejanju vsebin. Z uvedbo celostnega urejanja strani v različici 5.9 (januar 2022), ki je uvedel sklop funkcionalnosti, ki omogočajo uporabo blokov vizualnega urejevalnika Gutenberg za vse dele spletne rešitve ter z razširitvijo blokovnih tem, vzorcev in imenika blokov, je postal pomembno orodje za prilagajanje celotne spletne strani brez potrebe po kodi. Prihodnji načrti Gutenberga vključujejo še dve vnaprej določeni fazi – izboljšanje sodelovanja med avtorji in uvajanje večjezične podpore neposredno v jedro sistema [25].

6.3 Klasične teme

Klasične teme v WordPressu temeljijo na tradicionalni strukturi predlog in uporabljajo klasični urejevalnik vsebin. Namesto gradnje strani z bloki se pri teh temah postavitev določa s pomočjo datotek predlog (npr. `header.php`, `index.php`, `footer.php`) in ročnega vnašanja vsebine. Uporabniki lahko strani urejajo znotraj klasičnega besedilnega vmesnika, ki omogoča osnovno oblikovanje, brez možnosti vizualnega urejanja postavitve posameznih elementov. Zaradi svoje stabilnosti in predvidljive strukture ostajajo

primerna izbira za tehnično podkovane uporabnike ali za projekte, kjer ni potrebe po naprednem vizualnem urejanju. Dostop do urejanja nastavitev klasične teme je prikazan na sliki 6.5. Nastavitve klasičnih tem, ki so na voljo, so odvisne od izbrane teme. Kljub temu so v splošnem nastavitve klasičnih tem v primerjavi z nastavitvami blokovnih tem veliko bolj omejene, imajo pa razvijalci veliko bolj odprte roke pri prilagoditvah izvornih datotek.



Slika 6.5. Prilagoditve klasične teme.

Poseben primer klasičnih tem so t. i. »lahke« klasične teme, ki so minimalistične in namenjene vizualnemu urejanju spletnega mesta z uporabo vtičnikov za vizualno izdelavo strani (angl. Page Builders), kot sta Elementor in WP Bakery.

6.4 Vtičniki za vizualno izdelavo strani

Vtičniki za vizualno izdelavo strani so namenski vtičniki za WordPress, ki omogočajo vizualno oblikovanje spletnih strani brez neposrednega pisanja kode. Uporabniku ponujajo grafični vmesnik za urejanje postavitve spletnih elementov po principu

povleci-in-spusti, kar bistveno poenostavi proces oblikovanja vsebin. Na prvi pogled takšni vtičniki zelo spominjajo na privzeti urejevalnik Gutenberg, kar ni nenavadno – popularnost in razširjenost vtičnikov za vizualno izdelavo strani je predstavljal povod za nastanek urejevalnika Gutenberg, pri čemer vtičniki za vizualno izdelavo strani predstavljajo navdih za (nadaljnji) razvoj Gutenberga.

Vtičniki za vizualno izdelavo strani omogočajo vstavljanje in prilagajanje različnih gradnikov (npr. besedil, slik, gumbov, obrazcev, postavitev v mreži) ter hkrati zagotavljajo predogled sprememb v realnem času. Zaradi tega so posebej primerni tako za uporabnike brez programerskega predznanja kakor za hitrejše prototipiranje v profesionalnih okoljih. Med najbolj uveljavljenimi predstavniki so na podlagi podatkov o številu aktivnih namestitev iz uradnega kataloga WordPress vtičnikov [26] (povzeto julija 2025) naslednji: Elementor (več kot 10.000.000 aktivnih namestitev), Beaver Builder (več kot 100.000 aktivnih namestitev), Brizy (več kot 80.000 aktivnih namestitev) in Visual Composer (več kot 50.000 aktivnih namestitev). Po podatkih W3Tech je Elementor tako popularen, da ga uporablja kar 12,6 % vseh spletnih strani [27].

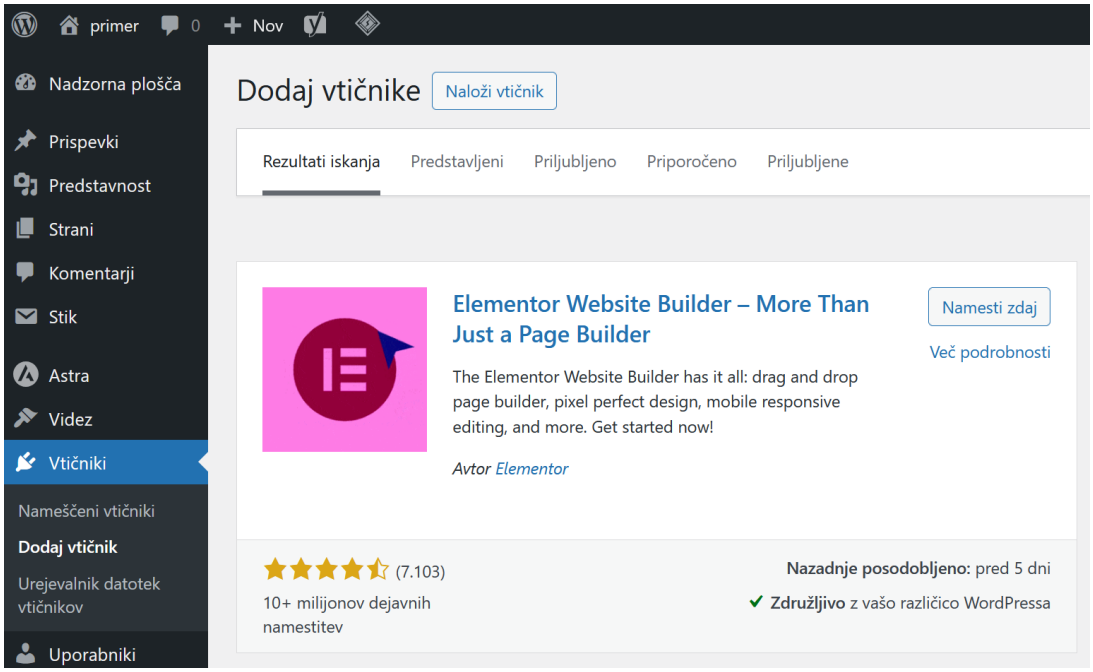
Čeprav vtičniki za vizualno izdelavo strani ponujajo visoko stopnjo prilagodljivosti in prijaznost do uporabnika, lahko v določenih primerih vplivajo na zmogljivost spletne strani ali otežijo poznejše nadgrajevanje, zato je njihova uporaba priporočljiva predvsem tam, kjer sta hitrost izdelave in vizualna prilagodljivost prioriteta.

6.4.1 Elementor

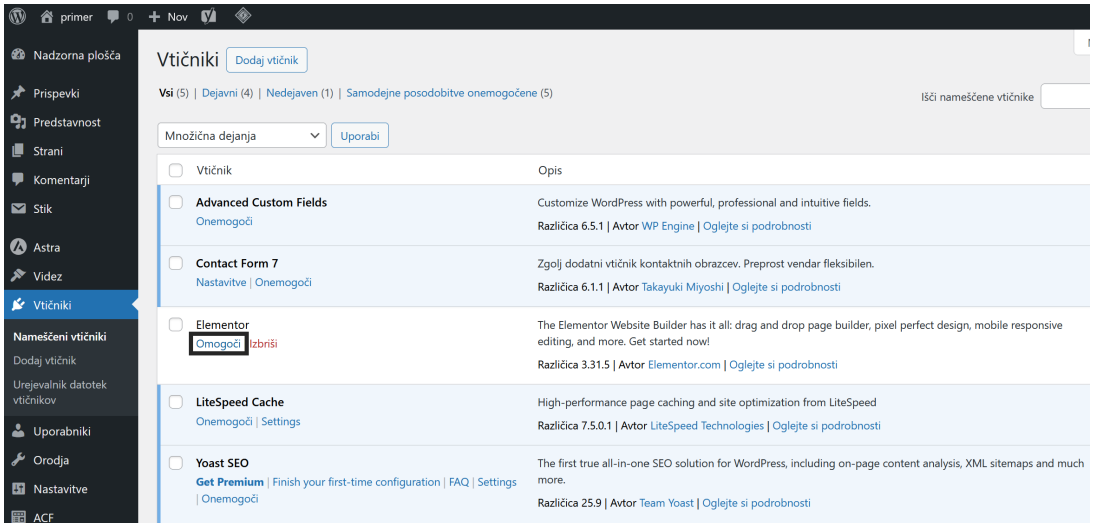
V tem poglavju bo predstavljeno urejanje spletnih strani z uporabo vtičnika Elementor, ki je eden od najpogosteje uporabljenih vizualnih urejevalnikov za WordPress. Na začetku je treba Elementor namestiti, kar je mogoče narediti v zavihku Vtičniki → Dodaj nov vtičnik, kot je prikazano na sliki 6.6. Vtičnik je po namestitvi treba še aktivirati, kot je prikazano na sliki 6.7.

Z Elementorjem je mogoče urejati izgled in strukturo strani ter prispevkov. V urejevalnik Elementor po dodajanju nove strani v WordPressu vstopimo tako, da na obstoječi strani kliknemo gumb Edit with Elementor, kot je to prikazano na sliki 6.8.

Stran je priporočljivo predhodno poimenovati in shraniti, sicer jo vtičnik avtomatično poimenuje z identifikatorjem (npr. Elementor #456 namesto želenega imena).

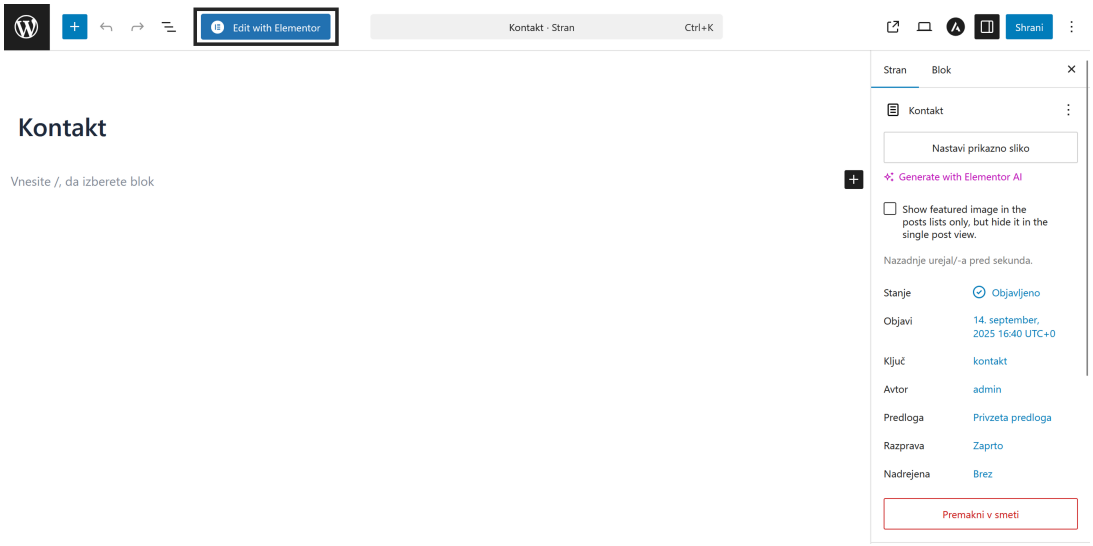


Slika 6.6. Nalaganje vtičnika Elementor.



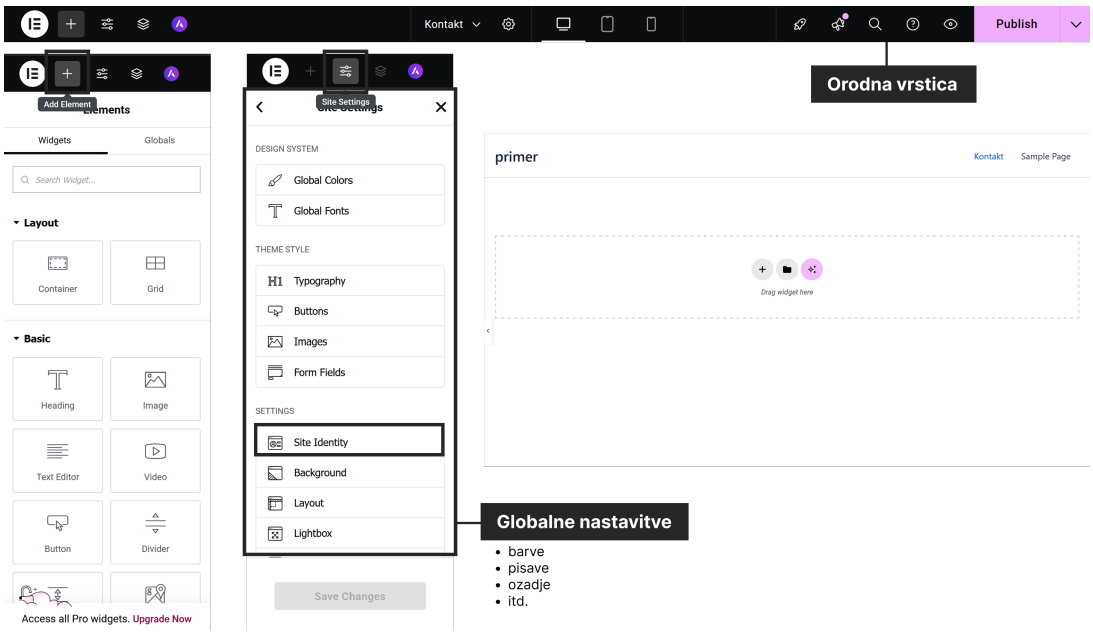
Slika 6.7. Omogočanje vtičnika Elementor.

Na sliki 6.9 je prikazana delovna površina urejevalnika Elementor med oblikovanjem spletne strani. V zgornjem delu zaslona je orodna vrstica, ki omogoča dostop do osnovnih funkcionalnosti, kot so shranjevanje, objava, predogled in preklapljanje med pogledi za različne naprave (namizje, tablica, mobilni telefon). Leva orodna vrstica prikazuje zavihek z gradniki (angl. Widgets), ki jih lahko uporabnik z metodo *povleci-in-spusti* vstavi v strukturo strani. Gradniki vključujejo temeljne vsebinske elemente, kot so naslovi, slike, videoposnetki, gumbi in delilniki. Za lažjo organizacijo elementov sta na voljo

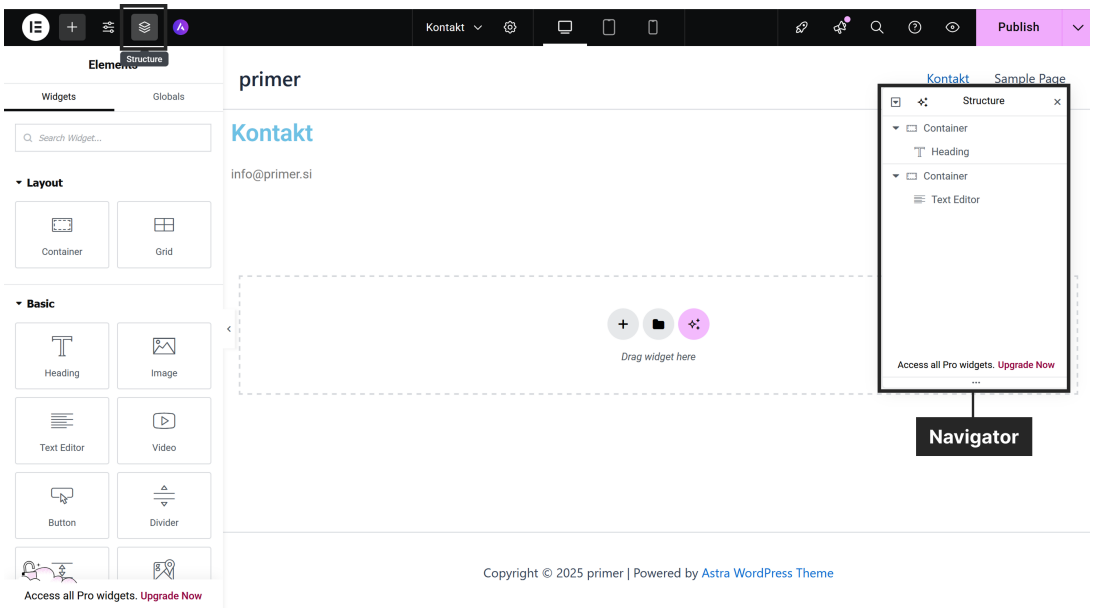


Slika 6.8. Urejanje strani z Elementorjem – izbira.

gradnika vsebnik in mreža. Zgornji gumb [+] oz. [Add Element] odpira opisan izbor vsebinskih komponent. Drugi primer (desno od prvotno opisanega primera) iste leve orodne vrstice prikazuje aktiviran zavihek [Site Settings], ki omogoča dostop do globalnih nastavitev spletnega mesta. Med njimi so na voljo parametri za določanje globalnih barv, pisav, slogov tipografije, oblikovanje gumbov in druge ključne oblikovne komponente. Spremembe, izvedene znotraj teh nastavitev, se nato odražajo na vseh straneh (znotraj iste teme), kar omogoča čim bolj enoten uporabniški vmesnik.



Slika 6.9. Urejanje globalnih nastavitev Elementorja.

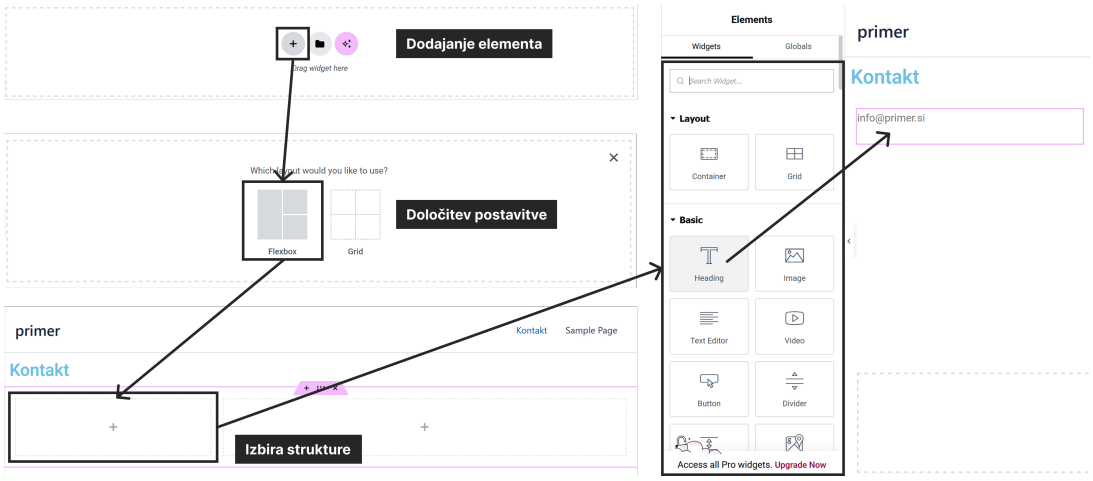


Slika 6.10. Hierarhija gradnikov v Elementorju.

Zaradi velikega števila elementov, ki jih je mogoče uvrstiti na spletno stran, Elementor omogoča tudi strukturni pregled hierarhije in organizacije vseh uporabljenih gradnikov. Ta je prikazan na sliki 6.10.

Postopek oblikovanja strukture spletne strani v urejevalniku Elementor poteka zaporedno, kot je prikazano na sliki 6.11. Najprej je priporočljivo dodati novo vsebinsko območje s klikom na gumb za vstavljanje elementa. Vmesnik nato zahteva določitev osnovne postavitve, pri čemer lahko uporabnik izbira med prilagodljivim vsebnikom ali standardno mrežno razporeditvijo. Sledi izbira strukture znotraj izbrane postavitve, pri čemer so na voljo različne razporeditve stolpcev. Ko je struktura določena, se posameznim stolpcem dodelijo vsebinski gradniki, ki jih uporabnik preprosto povleče iz levega stranskega menija na želeno mesto. Takšen modularni pristop omogoča hitro, pregledno in natančno oblikovanje. Uporabnik ga sicer lahko preskoči in gradnike s seznama vstavlja neposredno v prazno okolje. V tem primeru se uporabi privzeta postavitev, kjer je celotna vsebina predstavljena v enem stolpcu.

Po določitvi osnovne strukture je mogoče nadalje urejati ključne parametre postavitve posameznega vsebinskega območja znotraj strani. Uporabnik lahko v zavihku Layout natančno prilagodi širino (npr. širina vsebine, širina vsebnika), minimalno višino, poravnavo in razporeditev elementov znotraj sekcije, kot je prikazano na sliki 6.12. Poleg tega je mogoče nastaviti smer postavitve (vodoravno ali navpično), razmike med elementi (angl. Gaps), obnašanje pri prelomih vrstic (angl. Wrap) ter določiti način poravnave po



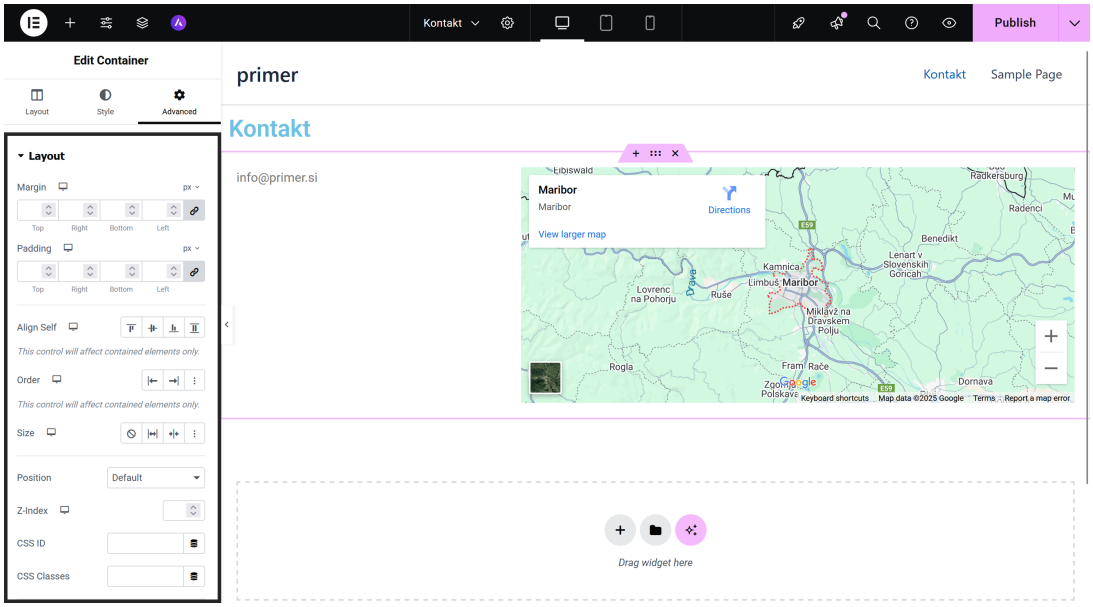
Slika 6.11. Osnovna organizacija strukture strani v Elementorju.

glavni in prečni osi (angl. Justify, Align).

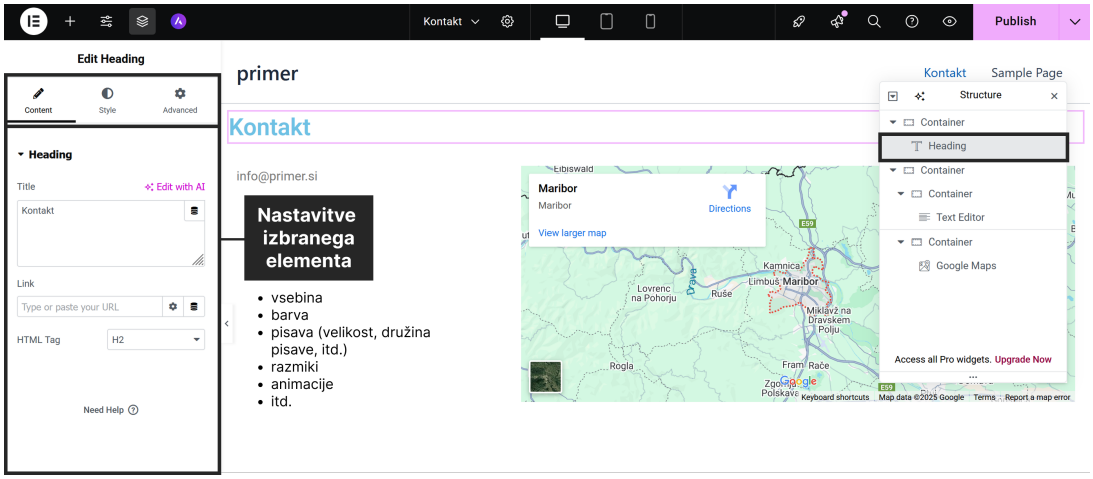
Takoj ko so posamezni elementi dodani v vsebinske bloke, je mogoče urejati lastnosti vsakega posameznega gradnika, kot je prikazano na sliki 6.13. Na sliki je izbran element naslov, kar omogoča dostop do njegovih nastavitev prek treh zavihkov: Content, Style in Advanced. V zavihku Content je mogoče določiti besedilno vsebino, hierarhični nivo naslova (npr. H1, H2) ter po potrebi dodati povezavo. V zavihku Style se nastavljajo tipografske lastnosti, kot so barva besedila, velikost pisave, družina pisave, razmiki med vrsticami in poravnava besedila. Zavihek Advanced omogoča dodatne možnosti, kot so zunanji in notranji robovi (angl. Margin, Padding), odzivne nastavitve in animacije. Organizacija nastavitev je enaka za vse elemente, tudi če se posamezne postavke med gradniki nekoliko razlikujejo.

6.4.2 Razširitve vtičnika Elementor

Opisane funkcionalnosti Elementorja so omejene na brezplačno različico. Med drugim je vtičnik mogoče razširiti z naročnino Elementor Pro [28], ki omogoča celovito vizualno oblikovanje spletnih mest brez potrebe po programiranju. V osnovnem paketu uporabniku ponujajo dostop do 50 dodatnih gradnikov, neposrednega urejevalnika tem za prilagajanje glavnih delov spletne strani, kot so glava, noga, arhivske strani in posamezne predloge, ter osnovna marketinška orodja, kot je gradnik za obrazce. Dodatni paket Advanced Solo uporabnikom ponuja 82 gradnikov, podporo za dinamične vsebine prek vtičnika ACF in lastnega tipa prispevkov, vgrajeni gradnik pojavnih oken ter napredndintegracije za trženje. Poleg tega omogoča popolno integracijo z vtičnikom WooCommerce za postavitev

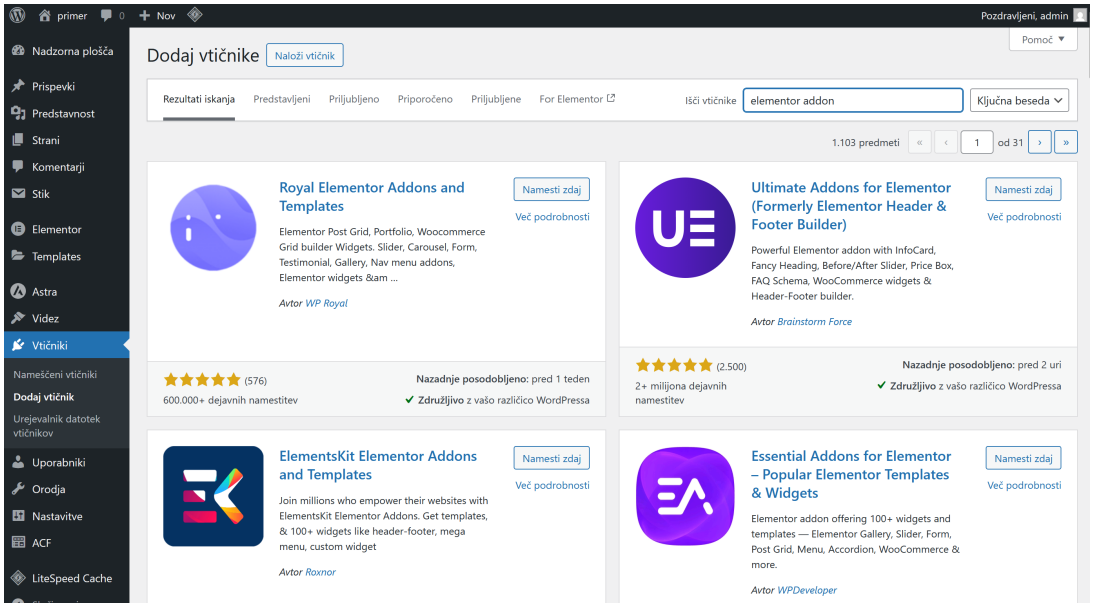


Slika 6.12. Osnovna organizacija strukture strani v Elementorju.



Slika 6.13. Urejanje posameznega gradnika v Elementorju.

spletnih trgovin ter povezave s plačilnimi sistemi, kot sta PayPal in Stripe. Kljub temu je postavitev spletne strani popolnoma izvedljiva tudi brez plačljive različice vtičnika. Funkcionalnosti Elementorja lahko alternativno razširimo z uporabo drugih vtičnikov, ki ponujajo dodatne elemente in dodatne razširitve nastavitev elementov. Primeri takšnih vtičnikov so prikazani na sliki 6.14. Vtičniki, ki razširjajo funkcionalnosti Elementorja dopolnjujejo tudi sicer plačljive funkcionalnosti Elementorja, kot je na primer urejanje glave in noge.



Slika 6.14. Vtičniki, ki razširjajo funkcionalnosti Elementorja.

6.5 Podteme

Vzpostavitev podteme (podrejene teme) ali t. i. otroške teme (angl. Child Theme) je priporočena praksa pri razvoju WordPress tem, saj omogoča varno prilagajanje videza, strukture in funkcionalnosti spletnega mesta brez spreminjanja izvorne (nadrejene) teme. Otroška tema podeduje vse značilnosti nadrejene teme, obenem pa razvijalcu omogoča, da z lastnimi slogovnimi datotekami (`style.css`) in funkcijskimi datotekami (`functions.php`) doda ali prilagodi določene funkcionalnosti. Ključna prednost tega pristopa je v tem, da se s posodobitvijo nadrejene teme ohranijo vse spremembe, narejene v podtemi, kar bistveno poveča vzdržljivost in nadgradljivost spletnega mesta. Vsaka podtema je organizirana v svojem direktoriju in vsebuje vsaj osnovni datoteki za definicijo sloga in funkcionalnosti – ti predstavljata minimalni nabor datotek za pravilno delovanje. Tak pristop je zlasti uporaben pri dolgoročni nadgradnji in prilagajanju spletnih mest po meri.

6.5.1 Ustvarjanje podteme s pomočjo vtičnika

Eden izmed pristopov ustvarjanja podteme vključuje uporabo namenskih vtičnikov, kot sta na primer Child Theme Generator in Child Theme Configurator. Ti vtičniki omogočajo enostavno generiranje podteme prek grafičnega uporabniškega vmesnika, brez

potrebe po ročnem urejanju datotek. Z nekaj kliki se ustvari osnovna struktura podteme, ki vključuje potrebne datoteke. Po uspešnem ustvarjanju podteme se lahko vtičnik s spletne strani tudi odstrani, saj za nadaljnje delovanje podteme ni več potreben. Ta pristop je še posebej primeren za uporabnike brez programerskega znanja.

6.5.2 Ročno ustvarjanje podteme

Naprednejši pristop k ustvarjanju podteme vključuje ročno ustvarjanje podteme v datotečni strukturi. Prvi korak ustvarjanja otroške teme je lociranje direktorija `wp-content`
`/themes`. Znotraj njega je treba ustvariti nov direktorij za podtemo. V direktoriju nato ustvarimo datoteko `style.css`, kamor vpišemo osnovne metapodatke o podtemi, kot so ime, avtor, predloga (`Template`) in različica. Minimalni delujoč primer takšne datoteke je prikazan v primeru 6.5.1. Opisan pristop ponuja večji nadzor nad strukturo in kodo ter je primeren za uporabnike, ki se želijo poglobiti v prilagajanje WordPress tem. Najpomembnejša postavka v tem dokumentu je `Template`, ki določa, na podlagi katere starševske teme je bila ustvarjena otroška tema.

Priporočljivo je, da ime direktorija vsebuje poimenovanje direktorija starševske teme z dodatkom `-child` (npr. `twentytwentyone-child`), kar poveča preglednost in sledi ustaljenim praksam razvoja.

Primer 6.5.1 (Primer stilne predloge `style.css` podteme.)

```
/*
Theme Name: Twenty Twentyone Child Theme
Template: twentytwentyone
*/
```

V direktoriju podteme je treba ustvariti tudi datoteko `functions.php`, ki omogoča vključevanje funkcionalnosti, običajno pa tudi uvoz slogovnih predlog starševske teme. Treba je preveriti, ali starševska tema samodejno poskrbi za nalaganje svojih stilov. Če to drži, dodatni koraki za uvoz slogov niso potrebni, saj WordPress poskrbi za dedovanje slogov prek mehanizmov prednosti stilov. Če pa osnovna tema ne omogoča samodejnega

uvoza slogov, mora podtema to storiti eksplicitno s funkcijo `wp_enqueue_scripts` z uporabo ustreznega mehanizma `enqueue`. V tem primeru je treba dodati eno izmed metod nalaganja, kar je odvisno od načina, kako osnovna tema vključuje lastne stile (npr. prek `get_template_directory_uri()` ali `get_stylesheet_directory_uri()`). Pravilna izbira metode zagotavlja, da se stilne predloge uvozijo v pravilnem zaporedju, kar preprečuje morebitne konflikte med elementi nadrejene in podrejene teme. Primer nalaganja slogov v otroški temi v datoteki `functions.php` je na voljo v primeru 6.5.2.

Primer 6.5.2 (Primer nalaganja slogov v otroški temi (`functions.php`).)

```
<?php
  add_action('wp_enqueue_scripts', 'enqueue_parent_styles');
  function enqueue_parent_styles(){
    wp_enqueue_style('parent-style',
    get_template_directory_uri().'/style.css');
  }
?>
```

V pristopu, ki vključuje ročno ustvarjanje otroške teme, je mogoče v direktorij dodati tudi posnetek zaslona (v formatu .PNG, velikosti 1200 × 900 pikslov). Slika mora biti poimenovana screenshot.png, kot je prikazano na primeru 6.5.3. Ta slika se bo prikazala v nadzorni plošči WordPressa, na seznamu tem. Ko je struktura otroške teme pravilno vzpostavljena (vključno z datotekama `style.css` in `functions.php`), se v nadzorni plošči WordPress v zavihku Izgled → Teme prikaže nova tema, ki jo lahko aktiviramo. Pomembno je, da ob tem starševske teme ne izbrišemo, saj otroška tema funkcionalno in vizualno temelji na njej. Če je vse pravilno nastavljeno, ob aktivaciji otroške teme ne bo zaznati vidnih sprememb – kar pomeni, da je prevzem lastnosti iz starševske teme uspešno izveden.

Primer 6.5.3 (Datotečna struktura otroške teme.)

```
wp-content
├── themes
│   ├── twentytwenty-child
│   │   ├── functions.php
│   │   ├── screenshot.png
│   │   └── style.css
│   └── twentytwenty
│       └── ...
└── ...
```

6.5.3 Ustvarjanje podteme z ukazno vrstico

WordPress omogoča tudi upravljanje celotnega sistema z ukazno vrstico. Orodje WP-CLI (WordPress Command Line Interface) [29] je uradno orodje, namenjeno izvajanju skrbniških opravil, kot so namestitev, posodobitve ter upravljanje tem in vtičnikov, brez uporabe grafičnega vmesnika. S tem razvijalcem omogoča hitrejše delo, večjo avtomatizacijo in učinkovitejše upravljanje okolja WordPress. Celoten nabor ukazov za WP-CLI je objavljen v dokumentaciji orodja [29], v nadaljevanju je prikazan primer uporabe orodja ukazne vrstice za ustvarjanje podteme, imenovane Astra Child (Primer 6.5.4). Primer uporablja ukaz `wp scaffold child-theme` [30] in predpostavlja, da je trenutno že naložena in aktivirana starševska tema Astra.

Primer 6.5.4 (Primer ustvarjanja otroške teme z WP-CLI.)

```
$ wp scaffold child-theme astra-child --parent_theme=astra
```

6.5.4 Prilagoditev predlog in dodajanje funkcionalnosti v podtemi

Podtema se uporablja za urejanje in spreminjanje funkcionalnosti in delovanja spletnega mesta. Dodajanje, razširjanje in urejanje funkcionalnosti poteka z datoteko `functions.php`, ki je del podteme. Po vedenju je primerljiva z WordPress vtičniki, saj omogoča izvajanje kode PHP ob nalaganju strani, vključevanje dodatnih skript, registracijo menijev, vnosnih mest za gradnike (angl. Widgets), dodajanje kratkih kod

(angl. Shortcodes), novih lastnih taksonomij, novih lastnih tipov prispevkov in drugo. Funkcionalno lahko torej z ustreznim programiranjem dosežemo enake učinke kot z razvojem lastnega vtičnika. Kljub temu je treba upoštevati, da ob morebitni menjavi teme vse funkcionalnosti in spremembe iz datoteke `functions.php` izgubimo. Zato je pri izbiri razvoja ali uporabe vtičnika oziroma implementaciji sprememb v `functions.php` treba upoštevati smernice razvoja. Te so povzete v preglednici 6.2.

Preglednica 6.2. Primerjava med WordPress vtičnikom in datoteko `functions.php`.

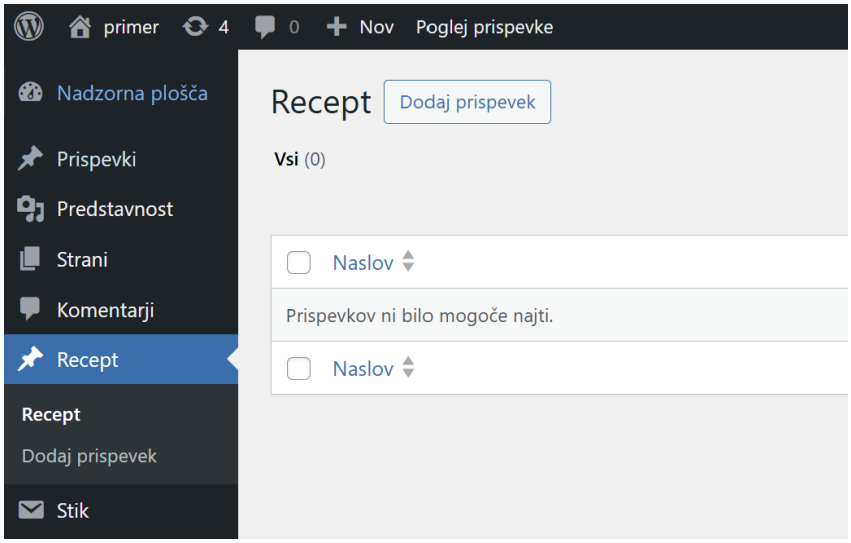
Vtičnik	Datoteka `functions.php`
• Zahteva unikatno glavo (angl. Plugin Header). • Se nahaja kot poddirektorij v `wp-content/plugins`. • Se izvede ob nalaganju strani, če je vtičnik aktiviran. • Je dostopen za vse teme. • Načeloma mora imeti en namen.	• Ne zahteva unikatne glave. • Se nahaja kot poddirektorij teme v `wp-content/themes/ime-teme`. • Se izvede zgolj takrat, kadar je v direktoriju izbrane teme. • Je dostopen samo za temo, v kateri se nahaja (če je tema spremenjena, dodanih in razširjenih funkcionalnosti ne moremo uporabljati). • Ima številne bloke kode z različnimi nameni.

Primer dodajanja lastnega tipa prispevka je eden izmed pogostejših posegov v delovanje WordPressa. Prikaz takšnega primera za dodajanje lastnega tipa prispevka z nazivom `Recepti`, je prikazan v primeru 6.5.5. Pri tem se uporabi v WordPressu vnaprej registrirana funkcija `register_post_type()` [31]. Ta omogoča razvijalcem, da registrirajo lastne tipe prispevkov, s čimer razširijo osnovno funkcionalnost sistema za upravljanje vsebin. S pomočjo te funkcije lahko ustvarimo lastne tipe prispevkov (npr. »dogodki«, »recepti« ali »projekti«), ki se obnašajo podobno kot privzeti prispevek, vendar imajo lastne oznake, povezave URL, upravljalne vmesnike in možnosti prikaza. Funkcija zahteva enoličen ključ vrste prispevka (do 20 znakov) ter nabor parametrov, ki opredeljujejo, ali bo tip prispevka javen, na voljo v vmesniku WordPress REST API, vključen v navigacijo, podprt s funkcijami, kot so urejevalnik, avtor ali komentarji, in ali bo imel arhiv. Pomembno je, da se funkcija vedno kliče v okviru akcije `init`, da se zagotovi pravilno nalaganje.

Primer 6.5.5 (Primeri dodajanja lastnega tipa prispevka (`functions.php`).)

```
<?php
  function dodaj_tip_prispevka_recept() {
    register_post_type( 'recept',
      array(
        'labels' => array(
          'name' =>  'Recept',
          'singular_name' => 'Recept'
        ),
        'public' => true,
        'has_archive' => true,
        'rewrite' => array('slug' => 'Recept'),
        'show_in_rest' => true,
        'supports' => array( 'title', 'thumbnail', 'excerpt' )
      )
    );
  }
  add_action( 'init', 'dodaj_tip_prispevka_recept' );
?>
```

Ustrezna raba funkcije `register_post_type()` omogoča napredno organizacijo vsebin in boljšo uporabniško izkušnjo za avtorje in urednike vsebin. Vključitev lastnega tipa prispevka Recept v nadzorni plošči WordPress je prikazana na sliki 6.15.

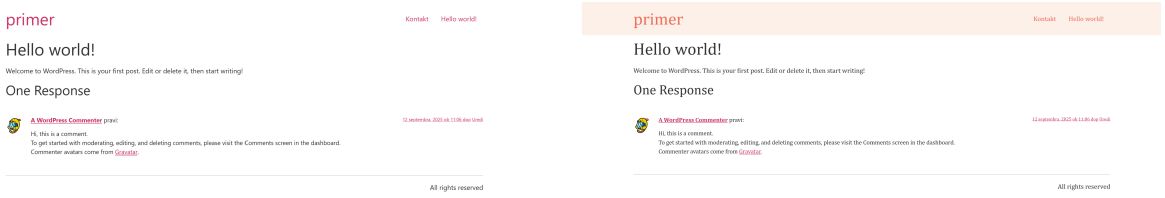


Slika 6.15. Dodan lasten tip prispevka `Recept`.

Na podoben način se odražajo tudi spremembe slogovne datoteke `style.css`. Primer 6.5.6 prikazuje spremembo pisave, barve ozadja telesa in glave strani, naslova ter velikosti pisave naslova H2. Odraz spremembe na dejanskem izgledu strani je prikazan na sliki 6.16.

Primer 6.5.6 (Sprememba slogovne datoteke otroške teme (style.css).)

```
* {
  font-family: "Cambria";
}
body {
  background-color: white;
}
#site-header{
  background-color: #FDF0E9;
}
#site-header a {
  color: #EF6D58;
}
h2.entry-title {
  font-size: 3rem;
}
```



a) Prvotni izgled teme (Hello Elementor). b) Spremenjen izgled teme.

Slika 6.16. Prikaz sprememb, implementiranih v primeru 6.5.6.

POGLAVJE 7

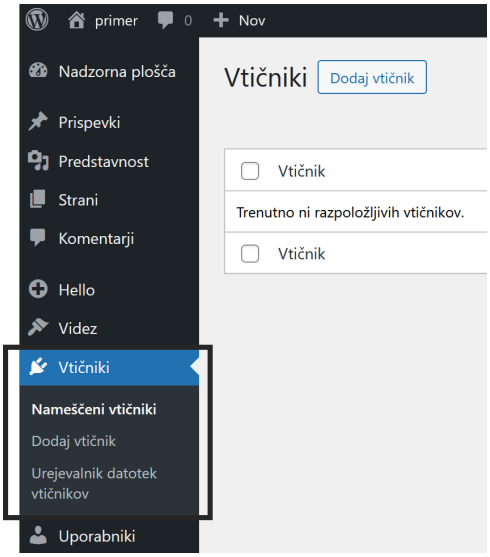
Vtičniki

V predhodnih poglavjih so bili vtičniki večkrat omenjeni kot mehanizem za razširjanje funkcionalnosti spletnega mesta WordPress brez poseganja v jedro sistema. V tem poglavju bo tej tematiki namenjena bolj celostna obravnava. Podrobneje bo predstavljena vloga vtičnikov v arhitekturi WordPressa, najpogostejši načini njihove uporabe ter prikazi primerov in virov, kjer jih je mogoče pridobiti. Ob tem bodo izpostavljeni tudi primeri vtičnikov, ki so privzeto vključeni v WordPress, primeri pogostih vtičnikov, ter razlaga, kako delujejo v ozadju. V nadaljevanju bodo obravnavani pomembni vidiki varnosti, vzdrževanja in posodabljanja vtičnikov. Predstavljene bodo tudi osnove razvoja lastnega vtičnika, kar je še posebej uporabno, kadar obstoječe rešitve ne zadostujejo specifičnim potrebam. Poglavje tako predstavlja ključni korak k razumevanju in učinkoviti uporabi enega najpomembnejših gradnikov WordPress ekosistema.

Kaj je vtičnik? Vtičniki omogočajo razširitev in dopolnitev osnovnih funkcionalnosti, ki jo WordPress privzeto vključuje. Jedro WordPressa je zasnovano tako, da ostane čim bolj enostavno in optimizirano, kar zagotavlja večjo prilagodljivost in zmanjšuje odvečno kodo. Z uporabo vtičnikov lahko razvijalci dodajo specifične funkcionalnosti, s čimer prilagodijo svoje spletno mesto glede na individualne potrebe projekta. Vtičnik zajema skupino datotek, organiziranih v en direktorij, in vključuje datoteke PHP, JS in CSS, grafike, tekstovne datoteke ipd.

WordPress vtičnike je mogoče najti na več načinov – najobsežnejši vir zanje je uradni katalog WordPress vtičnikov, dostopen neposredno iz skrbniškega vmesnika prek zavihka Vtičniki → Dodaj vtičnik, ki je prikazan na sliki 7.1. Do kataloga je mogoče dostopati tudi izven nadzorne plošče WordPress, na naslednji povezavi https://WordPress.org/plugins/. Poleg brezplačnih možnosti so na voljo tudi komercialni vtičniki na platformah, kot sta

CodeCanyon ali Envato Elements, pa tudi pri manjših ponudnikih, ki ponujajo bolj specializirane rešitve. Za napredne razvijalce pa obstaja možnost, da vtičnik razvijejo sami in ga popolnoma prilagodijo specifičnim potrebam svojega spletnega mesta. Prav tako je možno določene obstoječe vtičnike prilagoditi.



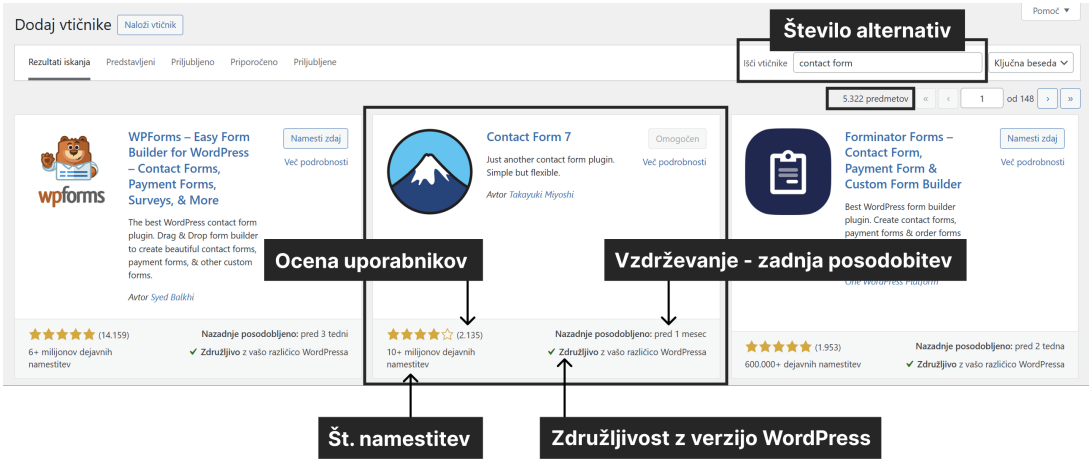
Slika 7.1. Dodajanje vtičnikov preko WordPress nadzorne plošče.

V začetni namestitvi WordPressa sta pogosto privzeto že nameščena dva vtičnika, Aksimet in Hello Dolly. Akismet preverja komentarje, ki jih uporabniki lahko dodajo na WordPress stran, in ugotavlja, ali gre za neželeno vsebino (angl. Spam). Vtičnik Hello Dolly nima globljega tehničnega pomena, temveč simbolično predstavlja optimizem in duh generacije, kot ga je ujel Louis Armstrong v znameniti pesmi Hello, Dolly. Gre za prvi uradni WordPress vtičnik. Ko je aktiviran, se v zgornjem desnem kotu skrbniškega vmesnika naključno prikazujejo verzi iz te pesmi [32]. Uporabniki lahko sicer oba (po lastni presoji) tudi odstranijo.

7.1 Izbira vtičnikov

Uradni katalog vtičnikov WordPress je imel med pripravo tega gradiva več kot 59.000 brezplačnih vtičnikov [26]. Pri tem je za pogoste scenarije, kot je na primer vključitev obrazca, na voljo več kot 5.000 vtičnikov. Pri izbiri ustreznega vtičnika med vsemi obstoječimi alternativami je ključnega pomena, da razvijalec oceni njegovo zanesljivost, skladnost in dolgoročno vzdrževanje. Priporočljivo je preveriti, ali je vtičnik združljiv z

izbrano različico WordPressa ter ali je bil nedavno posodobljen, kar kaže na aktivno vzdrževanje. Pomemben pokazatelj kakovosti je tudi število prenosov in aktivnih namestitev, saj večja razširjenost pogosto pomeni stabilnost in uporabnost. Poleg tega je smiselno upoštevati povprečno oceno uporabnikov in prebrati komentarje, kjer uporabniki delijo konkretne izkušnje glede delovanja, podpore in morebitnih težav. Kombinacija teh kriterijev omogoča premišljeno izbiro vtičnika, ki bo varno in učinkovito služil potrebam spletnega mesta. Primer predstavljenih odločitvenih faktorjev, ki vplivajo na izbiro vtičnika, je prikazan na sliki 7.2.



Slika 7.2. Izbira vtičnika – pregled odločitvenih faktorjev.

Posebno pozornost je treba nameniti tudi t. i. »freemium« vtičnikom, ki so v osnovni različici brezplačni, a ponujajo dodatne funkcije le ob plačilu. Tak model je sicer primeren za večino izbranih vtičnikov, vendar je pri odločitvi treba upoštevati, ali osnovna različica zadostuje potrebam projekta in ali bo nakup naprednih funkcionalnosti nujen.

Pred namestitvijo in aktivacijo vtičnika na spletnem mestu je dobro pregledati dokumentacijo vtičnika in preizkusiti vtičnik v izoliranem okolju, na primer na t. i. igrišču na naslednji povezavi https://playground.wordpress.net/.

7.1.1 Pregled najpogosteje uporabljenih vtičnikov

Kot predhodno omenjeno, WordPress že v uradnem katalogu ponuja na tisoče vtičnikov, ki pokrivajo najrazličnejše potrebe – od osnovne optimizacije strani, varnosti in generiranja rezervnih kopij do analitike, obrazcev in spletne prodaje. Zaradi tako obsežne ponudbe je za uporabnike koristno poznavanje vtičnikov, ki v skupnosti razvijalcev veljajo

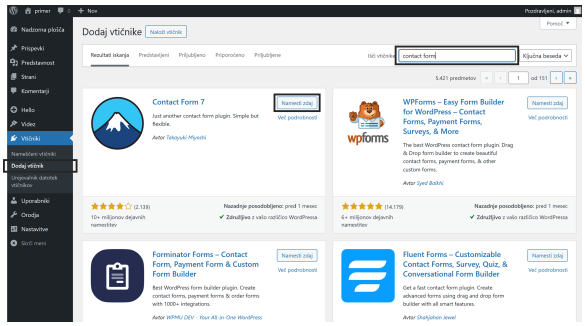
kot zanesljivi, razširjeni in dobro podprti. V preglednici 8.1 je predstavljen pregled nekaj priljubljenih brezplačnih vtičnikov iz uradnega kataloga WordPress [26], ki se uporabljajo na milijonih spletnih mest po vsem svetu. Njihova priljubljenost je običajno posledica kombinacije enostavne uporabe, zanesljive podpore in aktivne skupnosti.

Preglednica 7.1. Primeri najbolj razširjenih brezplačnih vtičnikov WordPress (podatki iz uradnega imenika vtičnikov [26], julij 2025).

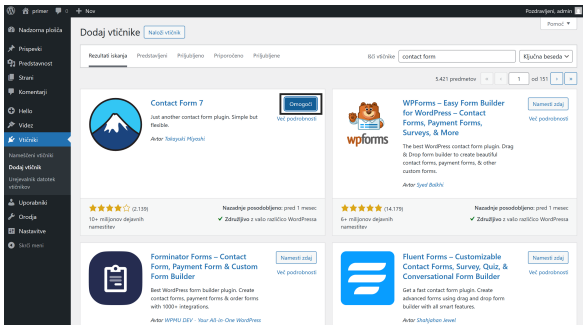
Ime vtičnika	Opis	Aktivne namestitve	Povprečna ocena
Yoast SEO	Optimizacija spletnih strani za iskalnike	10+ milijonov	5 / 5
Contact Form 7	Ustvarjanje kontaktnih obrazcev	10+ milijonov	4 / 5
Elementor	Vizualni urejevalnik strani	10+ milijonov	4.5 / 5
WooCommerce	Vzpostavitev spletne trgovine	7+ milijonov	4.5 / 5
Akismet Anti-Spam	Samodejna zaščita pred nezaželenimi komentarji	6+ milijonov	4.5 / 5
Wordfence Security	Varnostni vtičnik s požarnim zidom	5+ milijonov	4.5 / 5
Jetpack	*Vse-v-enem* orodje za varnost, hitrost in statistiko	4+ milijoni	3.5 / 5
WPForms Lite	Uporabniku prijazen ustvarjalnik obrazcev	6+ milijonov	5 / 5
UpdraftPlus	Varnostno kopiranje in obnova spletne strani	3+ milijone	5 / 5
LiteSpeed Cache	Optimizacija hitrosti in predpomnjenje	7+ milijonov	5 / 5
Advanced Custom Fields (ACF®)	Prilagoditev lastnih tipov prispevkov	2+ milijona	4.5 / 5

7.2 Uporaba vtičnikov

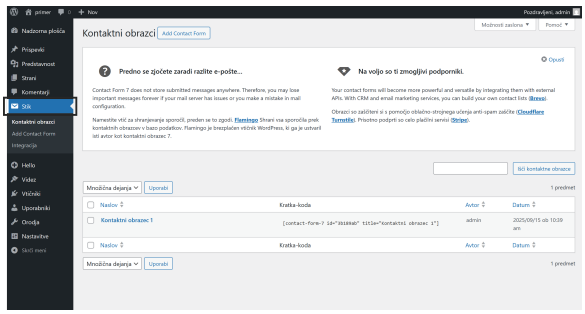
Delo z vtičniki v WordPressu vključuje več faz, od namestitve do dejanske uporabe v vsebini spletnega mesta in vzdrževanja. Prvi korak je namestitev vtičnika, ki jo lahko izvedemo neposredno iz nadzorne plošče (Vtičniki → Dodaj vtičnik), kot je prikazano na sliki 7.3a ali z ročnim prenosom in nalaganjem stisnjene datoteke ZIP. Po namestitvi je treba vtičnik aktivirati (slika 7.3b), da se funkcionalnosti vključijo v delovanje spletne strani. Naslednji korak je nastavitev in prilagoditev. Večina vtičnikov ponuja možnosti nastavljanja z uporabniškim vmesnikom znotraj nadzorne plošče, kot je prikazano na sliki 7.3c. Ob aktivaciji vtičnika ta običajno doda bližnjico do nastavitev v meni – bodisi kot svojo menijsko postavko ali podpostavko obstoječega menija (najpogosteje menija Orodja ali menija Nastavitve). Naprednejši vtičniki omogočajo tudi integracijo z vnaprej določenimi dogodki (več o tem v poglavju 8), s katerimi razvijalci lahko prilagodijo delovanje vtičnika na ravni kode.



a) Iskanje in nameščanje vtičnika.



b) Aktivacija vtičnika.

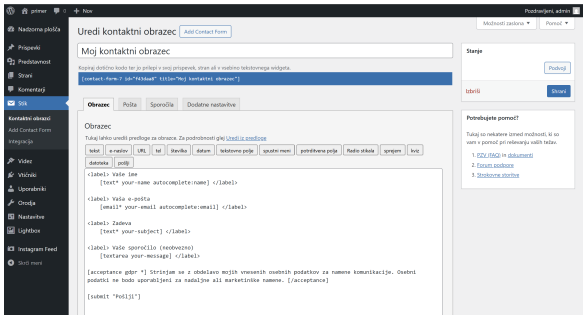


c) Upravljanje nastavitev vtičnika.

Slika 7.3. Primer namestitve in aktivacije vtičnika Contact Form 7.

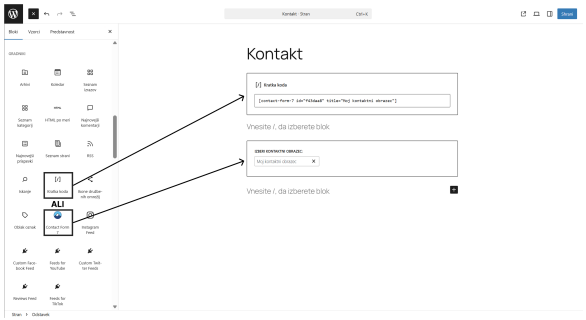
Vtičniki se nato uporabljajo na različne načine, odvisno od njihovega namena. Pogosto omogočajo upravljanje prek nastavitvenih zaslonov v skrbniškem vmesniku, vstavljanje funkcionalnosti prek blokov urejevalnika Gutenberg ali vizualnih urejevalnikov ali pa

vključitev s pomočjo kratkih kod (angl. Shortcode) neposredno v besedilo prispevkov in strani. Na ta način WordPress omogoča tako začetnikom kot naprednim uporabnikom, da učinkovito vključujejo dodatne funkcije brez obsežnega programiranja. Primer vključitve obrazca s kratko kodo, ki je bil zgrajen z vtičnikom Contact Form 7, je predstavljen na sliki 7.4. Na sliki 7.4a je predstavljena nastavitev novega obrazca »Moj kontaktni obrazec«. Na sliki 7.4b je prikazana vključitev vtičnika na stran Kontakt prek bloka urejevalnika Gutenberg »kratka koda«. Contact Form 7 v Gutenberg doda tudi lasten gradnik, ki je prikazan na sliki 7.4c. Vključitev obrazca je tako omogočena s tem namenskim gradnikom ali s splošnim gradnikom »kratka koda«. Na sliki 7.4d je nato prikazan dejanski izgled kontaktnega obrazca na strani »Kontakt«, kot jo vidijo obiskovalci strani (pri tem je bil ohranjen le en gradnik – prikazan na sliki 7.4b, ohranitev obeh gradnikov, prikazana na sliki 7.4c, bi namreč povzročila podvojen prikaz obrazca).

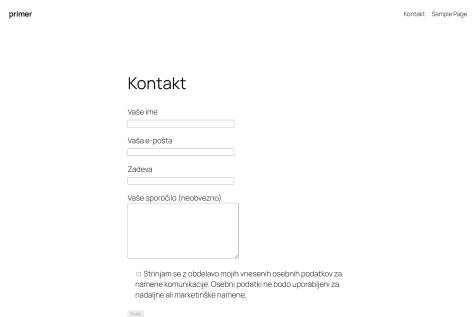


a) Nastavitev primera.

b) Vključitev obrazca s kratko kodo.



c) Gradnik »kratka koda« in namenski gradnik Contact Form.



d) Izgled vdelanega kontaktnega obrazca.

Slika 7.4. Primer uporabe vtičnika Contact Form 7 s kratko kodo.

7.3 Razvoj lastnega vtičnika

Razvoj lastnih vtičnikov v sistemu WordPress omogoča razširitev funkcionalnosti spletnega mesta brez poseganja v jedro sistema. Lasten razvoj omogoča ohranjanje popolnega nadzora nad obsegom in zahtevami funkcionalnosti, testiranjem ter nadzor nad posegi v podatkovno bazo in jedro sistema. Hkrati omogoča popoln nadzor nad upravljanjem z morebitnimi občutljivimi podatki. Gre za uveljavljen pristop, ki omogoča modularno dodajanje novih zmožnosti, s čimer se ohranjajo prilagodljivost, vzdržljivost in nadgradljivost spletnega okolja. Temeljna struktura in dobre prakse razvoja vtičnikov so zbrane v uradnem priročniku *WordPress Plugin Developer Handbook* [33]. Slednji vključuje dokumentacijo o ključnih temah, kot so uporaba akcij in filtrov, varnostna priporočila, obravnava zasebnosti, upravljanje z metapodatki, delo z uporabniki ter definiranje lastnih tipov prispevkov in taksonomij. Priročnik predstavlja standardiziran vir informacij, ki ga razvija in vzdržuje skupnost WordPress, in s tem zagotavlja aktualnost ter usklajenost z razvojem platforme.
Primer 7.3.1 predstavlja razvoj preprostega vtičnika. Osnovni koraki razvoja vtičnika so naslednji:

1. Izbira imena vtičnika (ime mora biti unikatno in se ne sme ponavljati z imeni drugih vtičnikov, ki so že nameščeni na spletnem mestu).

2. Ustvarjanje direktorija vtičnika in datoteke PHP v: `wp-content/plugins/ime-vticnika/{primer-vticnika}.php`.

3. Dodajanje glave datoteki PHP. Oblika glave je pri tem definirana vnaprej in predstavljena v primeru 7.3.1.

4. Programiranje za dodajanje funkcionalnosti. Pri tem so razvijalcem na voljo vse funkcije in razredi PHP, celoten nabor dogodkov, funkcij in razredov WordPress, ki so predstavljeni v poglavju 8.

5. Aktivacija vtičnika na spletni strani.

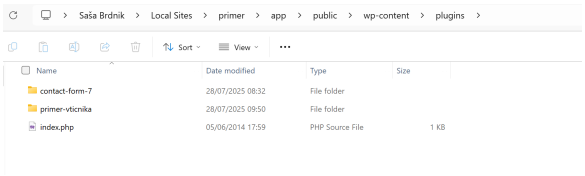
Primer 7.3.1 (Primer razvoja preprostega vtičnika za izpis avtorskih pravic.)

```
<?php
  /*
  Plugin Name: Primer vtičnika
  */

  // Funkcija, ki se izvede, ko uporabimo kratko kodo [avtorske-pravice]
  function izpisi_avtorske_pravice() {
    $leto = date('Y');
    $sporocilo = "<p>Praktikum © $leto</p>";
    return $sporocilo;
  }

  // Registracija kratke kode
  add_shortcode('avtorske-pravice', 'izpisi_avtorske_pravice');
?>
```

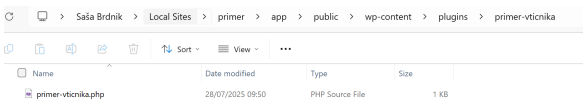
Dodajanje vtičnika iz zgornjega primera je za večjo jasnost prikazano tudi na sliki 7.5. Slika 7.5a prikazuje ustvarjanje direktorija z imenom vtičnika (brez presledkov, brez šumnikov) v `wp-content/plugins`. Sledi dodajanje istoimenske datoteke PHP znotraj tega direktorija, ki je prikazano na sliki 7.5b. Po tem je mogoče delovanje vtičnika preveriti z njegovo aktivacijo prek nadzorne plošče (slike 7.5c). Če na nadzorni plošči ne vidimo izpisa nobene napake, so bili vsi koraki do sedaj uspešno izvedeni. Vtičnik iz primera je definiral kratko kodo – za njegovo uporabo jo je treba umestiti še na poljubno mesto na spletni strani. Na primeru 7.5d je prikazana vključitev kratke kode `[avtorske-pravice]` z gradnikom »kratka-koda« v nogo strani. Morebitne sintaktične napake vtičnika bi bile razvidne takoj po shranjevanju strani z novo dodano kratko kodo. Če teh ni, preverimo še izgled strani z dodanim gradnikom, ki je prikazan na sliki 7.5e.

7.3.1 Upravljanje vtičnika preko nadzorne plošče WordPress

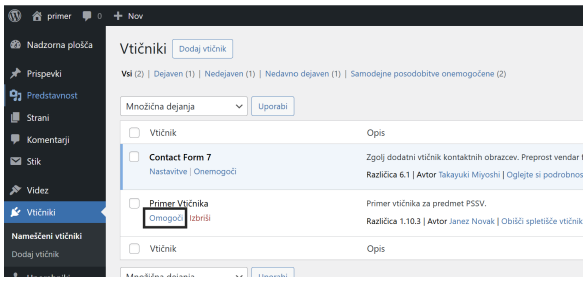
Da bi bilo možno upravljanje vtičnika preko nadzorne plošče WordPress, je treba osnovno strukturo vtičnika ustrezno razširiti. Ključni korak pri tem je dodajanje lastne strani z nastavitvami v nadzorno ploščo, kar se najpogosteje izvaja z uporabo funkcije `add_menu_page()` ali z akcijo `admin_menu`. Na tej strani je nato mogoče implementirati dodatne funkcionalnosti, kot so obrazci za vnos podatkov, preverjanje veljavnosti in



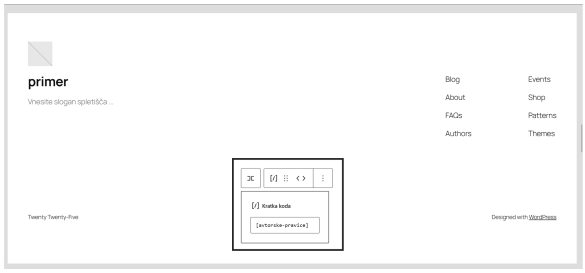
a) Ustvarjanje direktorija »primer-vticnika«.



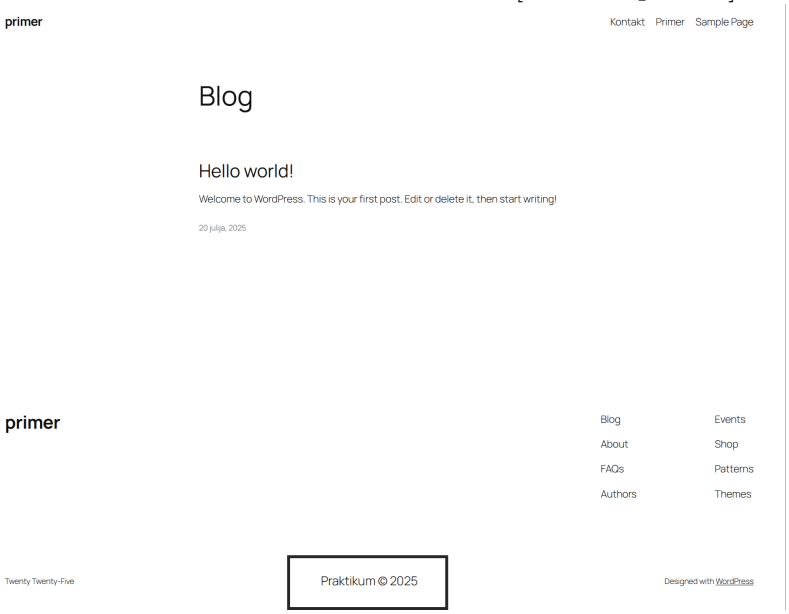
b) Dodajanje datoteke `primer-vticnika.php`.



c) Aktivacija vtičnika.



d) Dodajanje definirane kratke kode [avtorske-pravice] v nogo.



e) Končni izgled noge.

Slika 7.5. Primer razvoja in uporabe preprostega vtičnika iz primera 7.3.1.

obdelava nastavitev. Za shranjevanje in upravljanje podatkov se uporablja uradni vmesnik WordPress Settings API [34], ki omogoča varno in standardizirano upravljanje nastavitev znotraj vtičnika. Takšna razširitev primera 7.3.1 je prikazana na primeru 7.3.2.

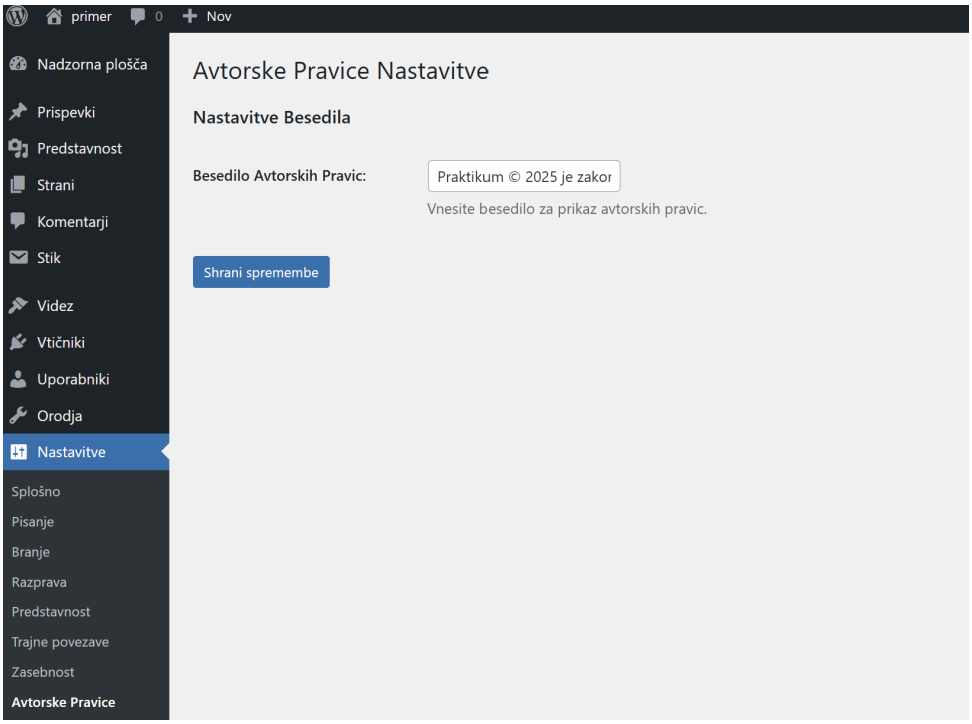
Primer 7.3.2 (Razširitev vtičnika z dodajanjem možnosti upravljanja z nadzorno ploščo.)

```
<?php
  /*
  Plugin Name: Primer vtičnika
  */

  // 1. Kratka koda
  function izpisi_avtorske_pravice() {
    $besedilo = get_option('avtorske_pravice_text',
    'Praktikum © ' . date('Y'));
    return "<p>" . esc_html($besedilo) . "</p>";
  }
  add_shortcode('avtorske-pravice', 'izpisi_avtorske_pravice');

  // 2. Dodajanje nove strani z nastavitvami v meni
  function dodaj_avtorske_pravice_nastavitve() {
    add_options_page(
      'Avtorske Pravice Nastavitve',
      'Avtorske Pravice',
      'manage_options',
      'avtorske_pravice_nastavitve',
      'prikazi_avtorske_pravice_nastavitve'
    );
  }
  add_action('admin_menu', 'dodaj_avtorske_pravice_nastavitve');

  // 3. Izpis obrazca z nastavitvami
  function prikazi_avtorske_pravice_nastavitve() {
    ?>
      <div class="wrap">
        <h2>Avtorske Pravice Nastavitve</h2>
        <form method="post" action="options.php">
        <?php
          settings_fields('avtorske_pravice_nastavitve_skupina');
          do_settings_sections('avtorske_pravice_nastavitve_skupina');
        ?>
```

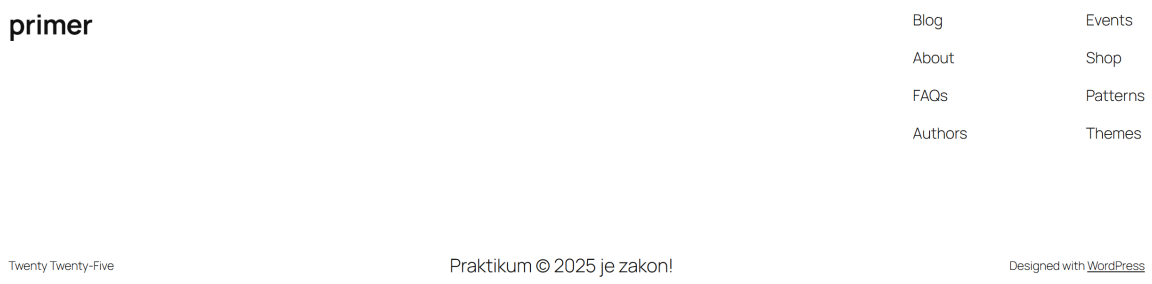
Primer (nadaljevanje)

```
        <table class="form-table">
          <tr valign="top">
            <th scope="row">Besedilo Avtorskih Pravic:</th>
            <td>
              <input type="text" name="avtorske_pravice_text" value="
                <?php
                  echo esc_attr(get_option('avtorske_pravice_text',
                  'Praktikum © ' . date('Y')));
                ?>" />
              <p class="description">
                Vnesite besedilo za prikaz avtorskih pravic.
              </p>
            </td>
          </tr>
        </table>
        <?php submit_button(); ?>
      </form>
    </div>
  <?php
  }

  // 4. Registracija nastavitev
  function avtorske_pravice_nastavitve_register() {
    register_setting(
      'avtorske_pravice_nastavitve_skupina',
      'avtorske_pravice_text');
    add_settings_section(
      'avtorske_pravice_nastavitve_sekcija',
      'Nastavitve Besedila',
      '',
      'avtorske_pravice_nastavitve_skupina');
  }
  add_action('admin_init', 'avtorske_pravice_nastavitve_register');

?>
```

Razširjen primer 7.3.2 je prikazan na sliki 7.6. Po shranjevanju posodobljene kode vtičnika lahko na nadzorni plošči opazimo dodane nastavitve za upravljanje z vsebino polja za avtorske pravice (slika 7.6a). Po shranjevanju posodobljene vrednosti polja se ta posodobi tudi v nogi strani, kjer je kratka koda še vedno uporabljena (slika 7.6b).



a) Dodajanje nastavitev vtičnika v meni Nastavitve.



b) Posodobljen izgled v nogi strani.

Slika 7.6. Razširitev vtičnika z dodajanjem možnosti upravljanja prek nadzorne plošče.

POGLAVJE 8

Dogodki, funkcije in razredi Wordpress

V sistemu WordPress predstavljajo dogodki (angl. Hooks) mehanizem, ki omogoča razširjanje in spreminjanje obnašanja sistema ali vtičnikov brez poseganja v izvorno kodo osrednjih datotek. Gre za vnaprej definirane točke v izvajanju WordPressa, tj. dogodke, ki so navedeni v dokumentaciji WordPressa ali vtičnika. Ta pristop prispeva k boljši modularnosti, združljivosti med razširitvami in večji varnosti pri posodabljanju jedra. WordPress podpira dve osnovni vrsti dogodkov: akcije (angl. Actions), ki omogočajo izvajanje dodatne funkcionalnosti ob določenem dogodku (npr. po shranjevanju prispevka), ter filtre (angl. Filters), ki omogočajo spremembo podatkov pred prikazom ali obdelavo (npr. sprememba vsebine naslova). V tem poglavju predstavljamo tudi ključne funkcije WordPress (angl. WordPress Functions) in razrede WordPress (angl. WordPress Classes), ki so pogosto uporabljeni v kombinaciji z dogodki z namenom razširjanja in spreminjanja obnašanja sistema ali vtičnikov.

8.1 Akcije in filtri

Uporaba dogodkov je temeljna za razvoj vtičnikov in tem, saj omogoča elegantno integracijo v sistem brez spreminjanja izvornih datotek. Seznam vseh razpoložljivih akcij [35] in filtrov [36] je na voljo v uradni dokumentaciji WordPress. Za uporabo dogodkov v WordPress okolju je treba upoštevati dva ključna koraka:

1. Najprej je treba definirati lastno funkcijo, ki bo služila kot povratni klic (angl. Callback) in določala, kaj naj se izvede ali spremeni ob določenem dogodku.

Ta funkcija mora biti strukturirana tako, da sprejme ustrezne parametre, ki jih WordPress posreduje v okviru izbranega dogodka.

2. Nato je treba to funkcijo registrirati z ustreznim dogodkom, kar se izvede z uporabo funkcije `add_action()` za akcije ali `add_filter()` za filtre. S tem WordPress poveže dogodek s funkcijo, ki naj se izvede ob nastopu tega dogodka. Tak način povezovanja omogoča dinamično razširjanje funkcionalnosti brez posegov v jedro sistema ali v obstoječe vtičnike [37].

Akcija ali filter? *Akcije* [35] omogočajo dodajanje podatkov ali spreminjanje delovanja. Akcije ne vračajo podatkov. Na primer, če želimo po uspešni prijavi uporabnika poslati obvestilo skrbniku po e-pošti, bi uporabili akcijo, saj gre za dodatno funkcionalnost, ki ne vpliva na izhodne podatke. *Filtri* [36] omogočajo preoblikovanje podatkov in hkrati vrnejo prilagojeno vrednost. Tipičen primer bi bil, če želimo spremeniti prikaz naslova prispevka tako, da se mu doda oznaka »Novo!« v primeru sveže vsebine. V tem primeru uporabimo filter, saj želimo obstoječo vrednost naslovnega niza pred prikazom spremeniti in posredovati nazaj v sistem. Ključna razlika med njima je torej v tem, da akcije sprožajo dodatna dejanja brez vračanja vrednosti, medtem ko filtri vračajo spremenjene podatke in vplivajo na končni izpis ali obnašanje spletnega mesta.

Primer 8.1.1 prikazuje uporabo akcije za uvoz stilnih predlog vtičnika iz datoteke CSS.

Primer 8.1.1 (Uporaba akcije za uvoz stilnih predlog.)

```
<?php
  function uvozi_stilne_predloge_vticnika() {
    wp_enqueue_style('primer-vticnika-css', plugin_dir_url(__FILE__) .
    '/stili.css');
  }
  add_action('wp_enqueue_scripts','uvozi_stilne_predloge_vticnika');
?>
```

Primer prikazuje uporabo filtra za prilagoditev naslova bloga in vračanje preoblikovane vrednosti nazaj v sistem.

Primer (nadaljevanje)

```
<?php
  function spremeni_naslov_prispevka($title) {
    if(get_post_type() === 'post'){
      return 'Blog: '. $title;
    }
    return $title;
  }
  add_filter('the_title', 'spremeni_naslov_prispevka');
?>
```

8.2 Funkcije

V WordPressu funkcije predstavljajo osnovno enoto, s katero se izvaja komunikacija med jedrom, temami in vtičniki. Vsaka funkcija izvaja specifično nalogo, kot so delo z bazo podatkov, registracija tipov prispevkov, dodajanje filtrov in akcij ter povezovanje zunanjih orodij. Gre za vgrajene ukaze PHP, ki omogočajo prilagoditev videza, delovanja in strukture spletnega mesta brez poseganja v jedro. Razumevanje teh funkcij omogoča učinkovitejše programiranje, večjo prilagodljivost in nadzor nad delovanjem WordPress okolja.

Primeri nekaj najpogosteje uporabljenih jedrnih funkcij v WordPressu so predstavljeni v preglednici 8.1. Celoten nabor funkcij je objavljen v središču za razvijalce WordPress Developer Resources [38].

Preglednica 8.1. Pogosto uporabljene WordPress funkcije in njihov namen [39].

Funkcija	Namen / cilj
`register_post_type()`	Registrira lasten tip prispevka, kar omogoča ustvarjanje strukturiranih vsebin poleg privzetih prispevkov in strani.
`add_action()`	Poveže uporabniško funkcijo z določenim dogodkom v WordPress jedru, s čimer omogoča razširjanje obnašanja sistema.

Nadaljevanje na naslednji strani

Nadaljevanje tabele 8.1

Funkcija	Namen / cilj
`add_filter()`	Omogoča manipulacijo z vrednostmi (npr. vsebino, HTML) pred njihovim izpisom, tako da filtrira rezultat funkcije.
`wp_enqueue_script()`	Uporablja se za vključevanje datotek JavaScript na pravilen način s podporo za odvisnosti in verzije.
`wp_enqueue_style()`	Omogoča pravilno vključevanje datotek CSS v temo ali vtičnik z nadzorom nad nalaganjem.
`get_post()`	Pridobi podatke o prispevku (ali strani) glede na ID prispevka; pogosto se uporablja pri prilagojenih poizvedbah.
`the_content()`	Izpiše vsebino trenutnega prispevka, filtrirano z vsemi registriranimi filtri (npr. za kratke kode ali oblikovanje).
`get_template_part()`	Naloži del predloge (npr. `content.php`) in omogoča ponovno uporabo strukturiranih vsebinskih blokov v temah.
`add_theme_support()`	Vključi podporo za dodatne funkcije teme (npr. sličice, menije, HTML5), kar omogoča boljše integracije.
`register_taxonomy()`	Registrira taksonomijo (kategorijo ali oznako), ki jo lahko povežemo s privzetimi prispevki ali lastnimi tipi prispevkov za boljšo kategorizacijo.
`have_posts()`, `the_post()`	Zanka predstavlja osnovni mehanizem WordPressa za iteracijo skozi prispevke. Funkcija `have_posts()` preverja, ali obstaja še kakšen prispevek, medtem ko funkcija `the_post()` pripravi podatke trenutnega prispevka za izpis z drugimi funkcijami (npr. `the_title()`, `the_content()`).
`add_shortcode()`	Doda novo kratko kodo (npr. `[prikazi_recepte]`), jo poveže z uporabniško definirano funkcijo, ki vrne vsebino za zamenjavo na uporabljenem mestu.

8.3 Razredi

Razred `WP_Query` je eden izmed ključnih gradnikov sistema WordPress. Uporablja se za poizvedovanje po vsebinah, kot so prispevki, strani ali lastni tipi prispevkov. Omogoča natančno določanje pogojev iskanja, filtriranja in razvrščanja rezultatov. V ozadju ga

WordPress uporablja tudi za t. i. glavno poizvedbo (angl. Main Query), ki določa, katera vsebina se prikaže na posamezni strani. Razvijalci ga pogosto uporabljajo za ustvarjanje dodatnih, prilagojenih poizvedb v temah in vtičnikih [40]. Primer 8.3.1 prikazuje uporabo razreda `WP_Query` v kombinaciji z uporabo zanke (angl. The Loop) za pridobitev objav. Pri tem ukaz `wp_reset_postdata()` poskrbi, da se globalni kontekst prispevkov po zaključeni zanki povrne na prvotno stanje, kar prepreči neželeno vplivanje na druge dele strani.

Primer 8.3.1 (Osnovna uporaba WP_Query in zanke.)

```
<?php
  $args = array(
    'post_type' => 'post',
    'posts_per_page' => 5
  );

  $query = new WP_Query($args);

  if ($query->have_posts()) :
    while ($query->have_posts()) :
      $query->the_post(); ?>
      <h2><?php the_title(); ?></h2>
      <?php the_excerpt(); ?>
    <?php endwhile;
    wp_reset_postdata();
  else :
    echo 'Ni objav.';
  endif;
?>
```

Razred `WP_Query` podpira širok nabor parametrov, ki omogočajo filtriranje vsebin. Najpogosteje uporabljeni so:

- `post_type` – določa tip vsebine (npr. post, recept, product),
- `posts_per_page` – število objav, prikazanih v poizvedbi,
- `category_name` – filtriranje po imenu kategorije,
- `tag` – filtriranje po oznakah,

- `author` – prikaz prispevkov določenega avtorja,
- `orderby` in `order` – določata vrstni red (npr. po datumu, naslovu ali poljih po meri),
- `meta_query` – omogoča iskanje po vrednostih v poljih po meri,
- `tax_query` – združuje pogoje po različnih taksonomijah,
- `date_query` – filtrira objave glede na datum (npr. zadnji teden),
- `paged` – omogoča paginacijo rezultatov.

Poleg praktične uporabe ima razred `WP_Query` pomembno vlogo v notranji arhitekturi sistema WordPress. Predstavlja vmesni sloj med podatkovno zbirko in logiko prikaza vsebine. Namesto neposrednih poizvedb SQL, ki bi jih prožili neposredno nad podatkovno bazo, omogoča standardiziran način pridobivanja podatkov, s čimer zagotavlja združljivost med različnimi tipi vsebin in razširitvami. Vsaka poizvedba, ki jo izvede WordPress, je pravzaprav instanca razreda `WP_Query` [40].

En WP_Query ali več? Pomembno je razumeti razliko med glavno poizvedbo in sekundarnimi poizvedbami. Glavna poizvedba se ustvari samodejno ob nalaganju strani in določa, katera vsebina bo prikazana na trenutnem naslovu URL. Sekundarne poizvedbe so tiste, ki jih razvijalec ustvari sam, običajno znotraj predlog ali vtičnikov, kadar želi prikazati dodatne sklope vsebin.

Razred `WP_Query` sledi objektno orientirani zasnovi, kar omogoča enotno uporabo funkcij, kot so `have_posts()`, `the_post()` in `get_posts()`. Poleg tega WordPress z uporabo filtra `pre_get_posts` omogoča tudi filtriranje in spreminjanje poizvedb še preden se te izvedejo. To razvijalcem daje možnost vplivanja na rezultate glavne poizvedbe brez potrebe po neposrednem poseganju v predloge. Z vidika učinkovitosti je `WP_Query` optimiziran za delo s podatkovno zbirko WordPress, vendar lahko prekomerna uporaba kompleksnih parametrov (zlasti pri večnivojskih `meta_query` ali `tax_query`) bistveno poveča čas izvedbe poizvedbe. Pri večjih spletnih mestih je zato priporočljivo uporabljati predpomnjenje rezultatov in indeksiranje podatkovnih preglednic.

- Pozabljena uporaba `wp_reset_postdata()` lahko povzroči napačne naslove in povezave v nadaljevanju strani.
- Napačna uporaba parametra `paged` v `WP_Query` lahko onemogoči delovanje paginacije.

V preglednici 8.2 je prikazan pregled pogosto uporabljenih razredov v okolju WordPress, ki tvorijo temelj njegovega objektno orientiranega sistema. Vsak razred ima določeno vlogo pri upravljanju podatkov, uporabnikov, datotek, vmesnika ali API-ja [41].

Preglednica 8.2. Pogosto uporabljeni WordPress razredi in njihov namen [41].

Razred	Namen / cilj
`WP_Query`	Uporablja se za izvajanje poizvedb po prispevkih, straneh ali lastnih tipih prispevkov; omogoča natančno filtriranje in razvrščanje ter tvori osnovo glavne poizvedbe.
`WP_User`	Predstavlja uporabnika in omogoča dostop do podatkov, vlog in zmožnosti; pogosto za preverjanje pravic.
`WP_Role`	Določa vloge in njihove zmožnosti; omogoča dodajanje ali odstranjevanje pravic.
`WP_Error`	Standardizirano vračanje napak in njihova obravnava brez prekinjanja izvajanja.
`WP_Widget`	Osnovni razred za gradnike; razvijalci ga razširijo za nove funkcionalnosti v stranskih vrsticah.
`WP_Filesystem`	Abstrakcijski sloj za datotečne operacije (branje, pisanje, kopiranje) z združljivostjo med okolji.
`WP_REST_Server`	Obravnava REST API zahteve; registracija poti in izvajanje metod HTTP (GET, POST, DELETE).
`WP_REST_Request`	Predstavlja posamezno REST zahtevo in omogoča dostop do parametrov, poti in metod.
`WP_REST_Response`	Oblikovanje in pošiljanje odzivov v REST API, vključno s statusno kodo in podatki.
`WP_Comment_Query`	Poizvedovanje po komentarjih s filtriranjem po avtorju, statusu ali tipu objave.
`wpdb`	Globalni razred za delo z bazo; varno izvajanje SQL poizvedb in pridobivanje rezultatov.

Nadaljevanje na naslednji strani

Nadaljevanje tabele 8.2

Razred	Namen / cilj
`WP_Theme`	Predstavlja aktivno temo in omogoča dostop do informacij ter preverjanje združljivosti.
`WP_Filesystem_Direct`	Podrazred `WP_Filesystem` za neposredne datotečne operacije brez FTP.

8.4 Praktični primeri uporabe dogodkov, funkcij in razredov

8.4.1 Registracija lastnega tipa prispevka in definiranje polj po meri

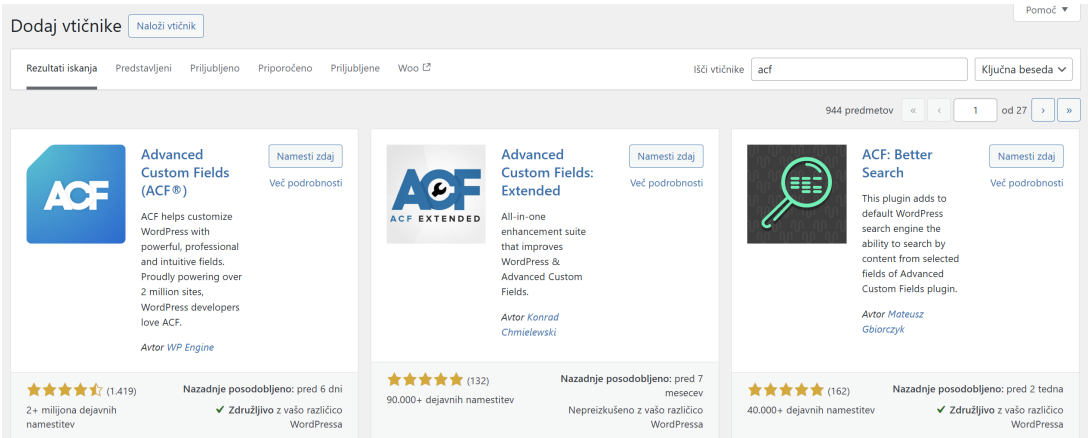
V primeru 6.5.5 je bilo prikazano dodajanje lastnega tipa prispevka, registriranega v datoteki `functions.php`. Pri odločitvi, ali bomo posamezni poseg v WordPress implementirali v obliki novega vtičnika ali zgolj z dodajanjem kode v datoteko `functions.php` v podtemi, je treba upoštevati obseg posega, njegovo kompleksnost in njegovo dolgoročnost.

Nov vtičnik ali poseg v `functions.php`? *Vnos sprememb v datoteko* `functions.php` je primeren predvsem za manjše prilagoditve, ki so specifične za trenutno aktivno temo, kot so dodatne funkcije, registracija menijev ali nalaganje slogov. Takšne spremembe se nanašajo na prikaz in vedenje strani, vezano na temo, in ne potrebujejo večje modularnosti ali ponovne uporabe. Po drugi strani pa se svetuje razvoj *ločenega vtičnika*, kadar dodajamo funkcionalnost, ki ni vezana izključno na temo, jo lahko večkrat uporabimo, jo enostavno prenašamo med spletnimi mesti ali želimo ohraniti prilagoditve ob spremembi teme. Vtičniki so zato primernejši za samostojne funkcije, ki širijo osnovno delovanje WordPressa ali drugih vtičnikov, kot je WooCommerce.

Definicijo lastnega tipa prispevka iz primera 6.5.5 lahko torej izvedemo znotraj datoteke `functions.php` aktivne podteme ali kot samostojen vtičnik. Kadar je lasten tip prispevka tesno povezan z določeno temo ali gre za enostavno spletno mesto, kjer tip prispevka ne potrebuje dolgotrajne prenosljivosti, je smiselno uporabo definirati v

datoteki `functions.php`. Če pa želimo, da tip prispevka ostane delujoč neodvisno od uporabljene teme, ga ponovno uporabimo na več projektih ali ga razvijamo ločeno, je priporočljivo ustvariti lasten vtičnik. S tem zagotovimo večjo modularnost, prenosljivost in preglednost kode. V obeh primerih se uporablja funkcija `register_post_type()`, ključna razlika pa je predvsem v organizaciji in trajnosti rešitve.

Ne glede na mesto vpeljave lastnega tipa prispevka, je njihove nastavitve mogoče bolj napredno urejati z enim izmed priljubljenih vtičnikov – Advanced Custom Fields (ACF) [42], prikazanim na sliki 8.1. Ta omogoča razširjeno upravljanje vsebin v WordPressu s pomočjo po meri definiranih vnosnih polj. Uporabnikom ponuja enostaven urejevalnik polj, ki jih je mogoče dodajati kjerkoli – na prispevke, strani, uporabnike, taksonomije, komentarje in celo v lastne nastavitvene strani. Vrednosti teh polj je nato mogoče enostavno prikazati v datotekah posameznih predlog. ACF tako omogoča strukturiranje podatkov brez poseganja v kodo ter s tem WordPress spremeni v zmogljiv sistem za upravljanje vsebin. ACF od različice 6 naprej omogoča tudi registracijo lastnih tipov prispevkov in taksonomij neposredno z uporabniškim vmesnikom. Komercialna različica (ACF PRO) dodatno vključuje še nabor dodatnih funkcionalnosti, kot so npr. ponavljajoča se polja, polja za fleksibilno vsebino ter možnosti za izdelavo lastnih blokov v vmesniku Gutenberg.

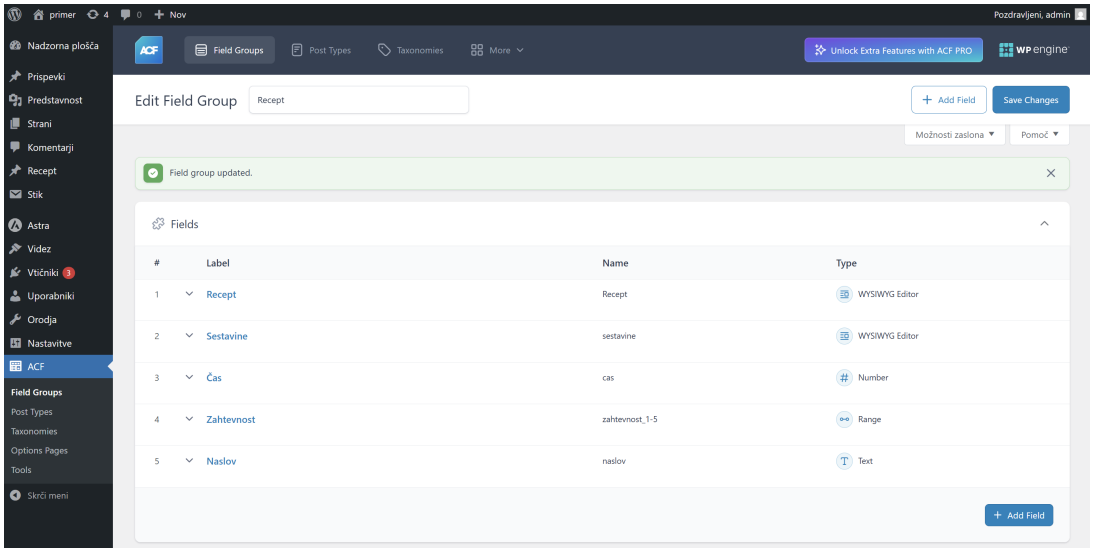


Slika 8.1. Advanced Custom Fields (ACF®) vtičnik.

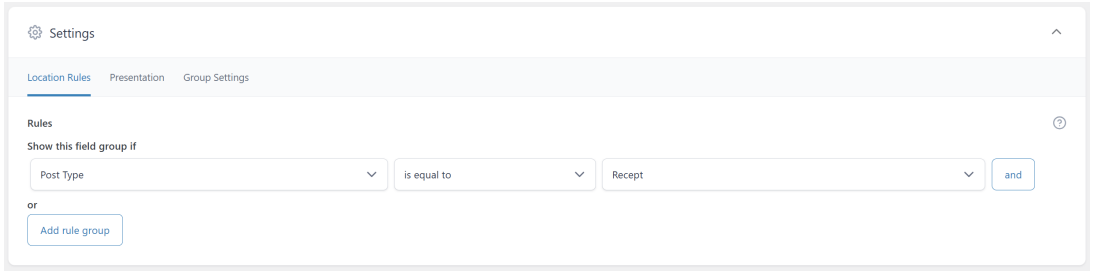
Po aktivaciji vtičnika je podrobnosti glede dodatnih polj določenega tipa prispevkov mogoče urediti v nadzorni plošči, v postavki menija ACF. Na sliki 8.2a je v skrbniškem vmesniku prikazan urejevalnik skupin polj ACF, v katerem sta definirani dve polji (naslov in opis naloge), pri čemer sta za vsako določena oznaka in tip (besedilo oziroma večvrstični

vnos). Poljem je mogoče določati tudi omejitve, obveznost izpolnitve in validacijska pravila. Na sliki 8.2b je vidna nastavitev pogojev prikaza, kjer je določeno, da naj se ta skupina polj prikaže zgolj pri prispevkih tipa `Recept`. Brez te nastavitve pravila predhodno definiranih polj ne bodo uporabljena. Na sliki 8.2c je nato prikazano vnosno okolje lastnega prispevka tipa `Recept`, kjer so vidna polja, dodana z ACF.
Vneseni metapodatki, ki so bili testni vsebini dodani z urejevalnikom (slika 8.2c), so po shranjevanju uspešno dodani v podatkovno bazo. Vendar njihov prikaz še ni določen. Ob ogledu vsebine tega prispevka tako pridemo na prazno stran. Za definiranje prikaza na novo dodanih polj je treba dodati in urediti predlogo `single-recept.php`. Za hiter opomnik glede delovanja hierarhije predlog v WordPressu se vrnite na poglavje 5.3. Primer v nadaljevanju temelji na temi Astra, ki ji je bila dodana otroška tema (kot je bilo predstavljeno v poglavju 6.5). Skladno s pravili hierarhije predlog je bila v direktorij otroške teme `astra-child` dodana datoteka `single-recept.php`. V njeno vsebino je bila za večjo skladnost prekopirana vsebina datoteke `single.php` iz starševske teme Astra. S tem smo ohranili prikaz glave, drobtinic in stranskega menija. Umestitev datoteke je prikazana na sliki 8.3a. Izgled prispevka brez sprememb predloge pa je prikazan na sliki 8.3b.
V kopirani datoteki `single.php` teme Astra (zdaj `single-recept.php`) je bila zamenjana privzeta vrstica za prikaz vsebine `astra_content_loop` z vsebino, predstavljeno v primeru 8.4.1. Preostali del predloge je ostal nespremenjen. Iz primera je razvidno, da je bila za prikaz metapodatkov, ki so bili dodani z novimi polji ACF, uporabljena funkcija `get_field()` [43]. Funkcija je del vtičnika ACF in omogoča pridobivanje vrednosti polj po meri iz določene vsebine v WordPressu. Njena uporaba je ključna pri izpisovanju vsebin, ki so bile dodane prek vmesnika ACF. Definirana je kot `get_field( $selector, [$post_id = false], [$format_value = true], [$escape_html = false] );`. Pri tem funkcija sprejme naslednje parametre [43]:

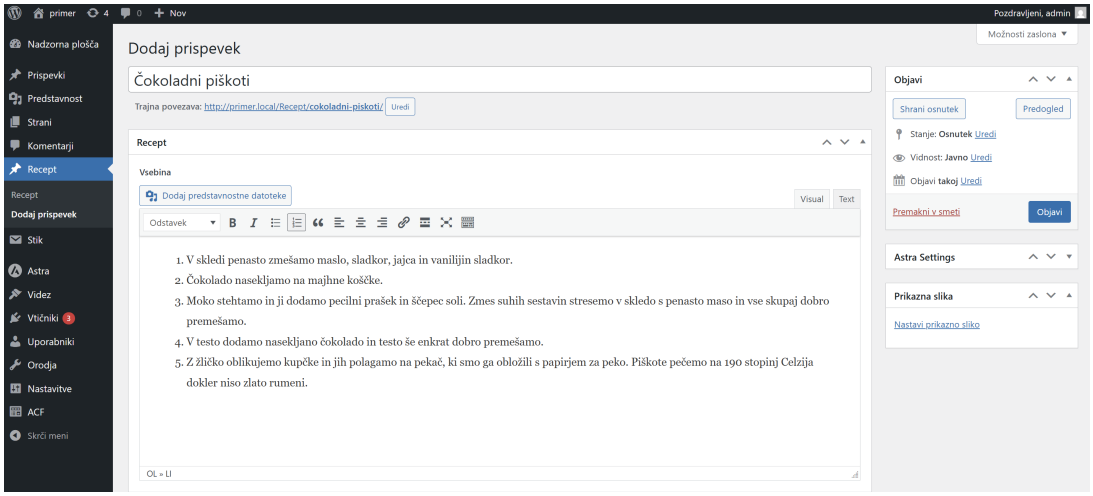
- `$selector` (niz, obvezno) – ime ali ključ polja, ki ga želimo pridobiti.
- `$post_id` (opcijsko) – ID prispevka (ali uporabnika, termina itd.), iz katerega se polje bere. Če je vrednost parametra *false*, se uporabi trenutni prispevek znotraj zanke (`the_loop`).
- `$format_value` (logična vrednost, opcijsko) – določa, ali naj se uporabi logika formatiranja (npr. za izbire ali datume). Privzeta vrednost je *true*.



a) Dodajanje nastavitev vtičnika v meni Nastavitve.



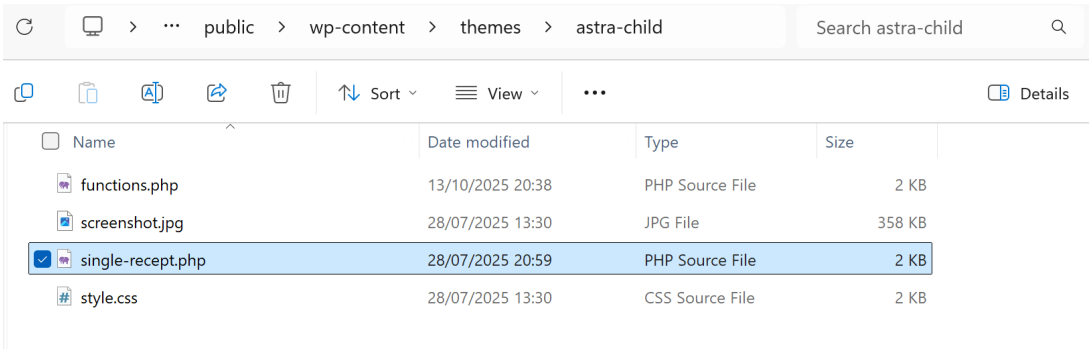
b) Dodelitev dodanih polj lastnemu tipu prispevka `Recept`.



c) Dodajanje prve vsebine tipa prispevka `Recept`.

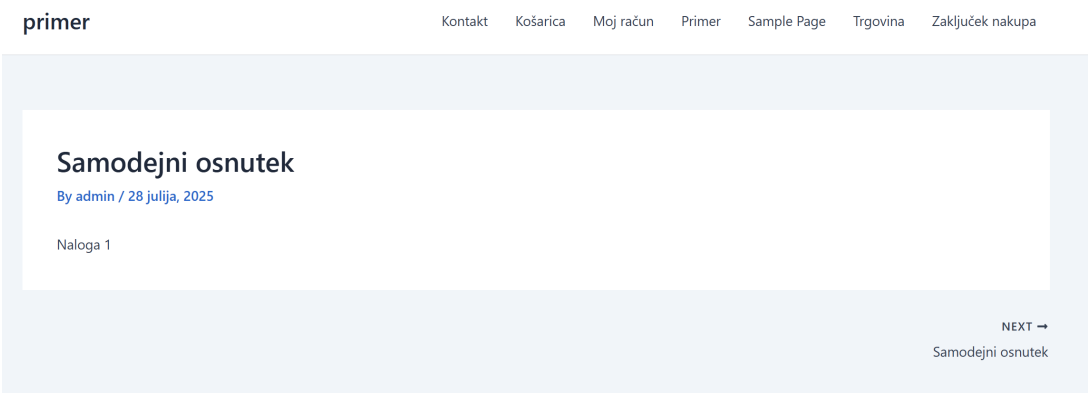
Slika 8.2. Uporaba ACF za definiranje vnosnih polj tipa prispevka `Recept`.

- `$escape_html` (logična vrednost, opcijsko) – če je omogočeno (`format_value` je *true*), vrne HTML-varno različico vrednosti. Privzeta vrednost tega parametra je *false*.



a)

Dodajanje predloge za lasten tip prispevka `Recept`.



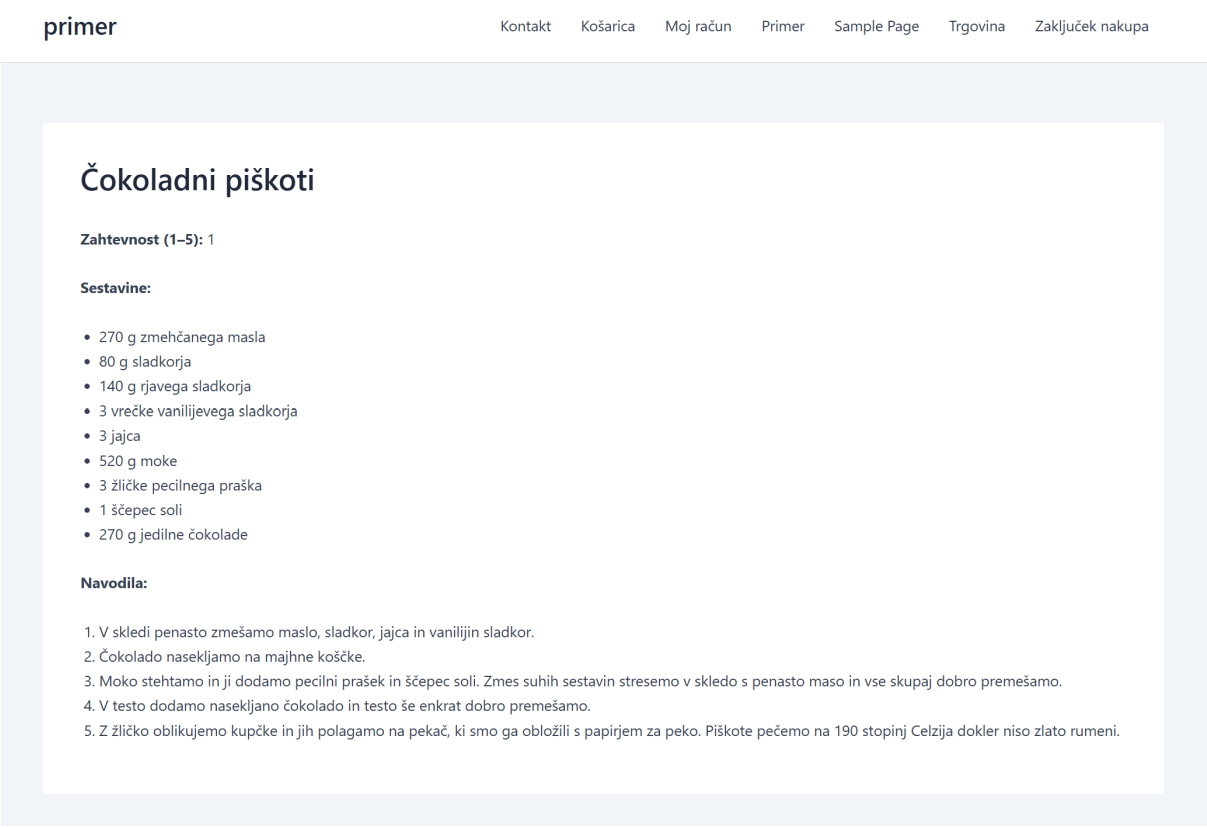
b) Izgled strani brez dodatne prilagoditve predloge.

Slika 8.3. Dodajanje osnovne predloge za lasten tip prispevka `Recept`.

Primer 8.4.1 (Prikaz dodanih polj ACF pri lastnem tipu prispevka `Recept`.)

```
<?php
  if ( have_posts() ) :
    while ( have_posts() ) :
      the_post();
      $naslov = get_field( 'naslov' );
      $sestavine  = get_field( 'sestavine' );
      $recept  = get_field( 'recept' );
      $zahtevnost   = get_field( 'zahtevnost_1-5' );
      ?>

      <?php astra_primary_content_top(); ?>

      <article <?php post_class(); ?>>
        <header class="entry-header">
          <h1 class="entry-title"><?php echo esc_html( $naslov ); ?></h1>
        </header>

        <div class="entry-content">
          <?php if ( $zahtevnost ) : ?>
            <p><strong>Zahtevnost (1-5):</strong>
            <?php echo esc_html( $zahtevnost ); ?></p>
          <?php endif; ?>
          <?php if ( $sestavine ) : ?>
            <p><strong>Sestavine:</strong>
            <?php echo ( $sestavine ); ?></p>
          <?php endif; ?>
          <?php if ( $recept ) : ?>
            <p><strong>Navodila:</strong>
            <?php echo ( $recept ); ?></p>
          <?php endif; ?>
        </div>
      </article>

      <?php
    endwhile;
  endif;
?>
```

Izgled testne vsebine, katere vnos je bil predstavljen na sliki 8.3, je po posodobitvi predloge `single-recept.php` prikazan na sliki 8.4.



Slika 8.4. Posodobljen prikaz lastnega tipa prispevka `Recept`.

Če ob dodajanju predloge za lasten tip prispevka ta ni takoj uporabljena za prikaz, je mogoče WordPress prisiliti k osvežitvi pravil prikaza. To je mogoče storiti v zavihku Nastavitve → Trajne povezave. Ob odprtju strani z nastavitvami je preprosto potrebno klikniti Shrani, brez da bi vnesli kakršnekoli spremembe. Naprednejši uporabniki lahko za to uporabijo funkcijo `flush_rewrite_rules();` ali katerega od vtičnikov za izbris predpomnilnika.

8.4.2 Registracija lastne taksonomije

Dodajanje nove taksonomije je ključen korak pri organizaciji vsebin na strani WordPress, zlasti kadar uporabljamo lastne tipe prispevkov. S taksonomijami omogočimo dodatno strukturiranje vsebin, ki presega osnovni sistem kategorij in oznak. Na primer, pri tipu prispevkov `Recept` lahko dodamo taksonomijo »Tema« ali »Sezona«, s čimer zagotovimo

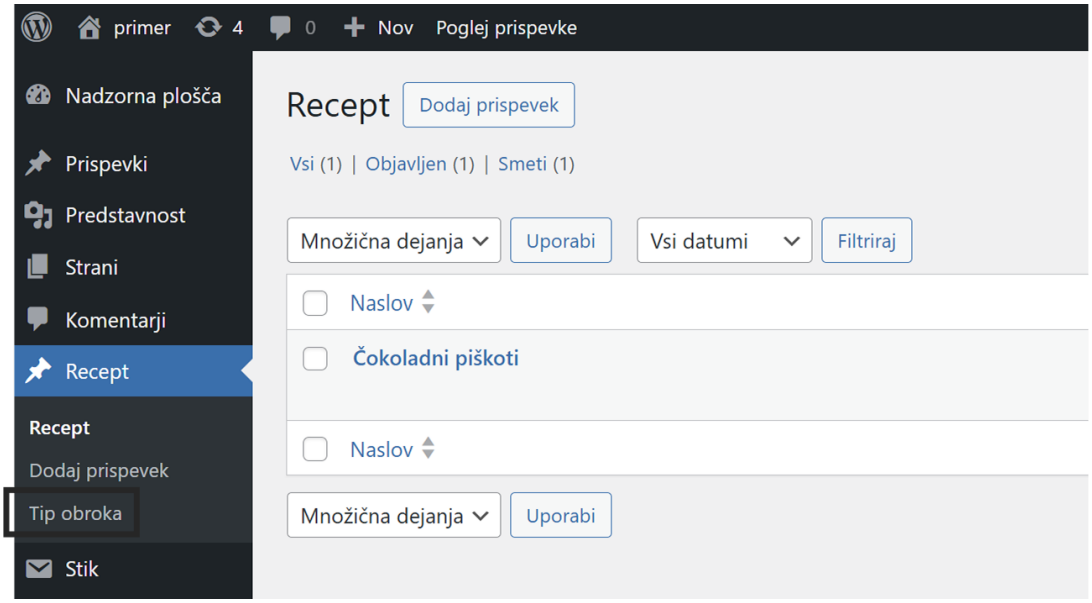
boljši pregled, filtriranje in uporabniško navigacijo po vsebini. WordPress omogoča registracijo taksonomij s funkcijo `register_taxonomy()` [44], kar omogoča popolno prilagoditev konvencij imenovanja, hierarhične strukture in povezave z določenimi tipi prispevkov. Nove taksonomije je enako kot nove tipe prispevkov mogoče dodati v kodi ali z vtičnikom ACF. Primer 8.4.2 prikazuje dodajanje nove taksonomije v kodi.

Primer 8.4.2 (Dodajanje taksonomije `Tip obroka` lastnemu tipu prispevkov `Recept`.)

```
<?php
  function dodaj_taksonomijo_tip_obroka() {
    register_taxonomy( 'tip_obroka', 'recept',
      array(
        'labels' => array(
          'name' =>  'Tip obroka',
          'singular_name' => 'Tip obroka'
        ),
        'public' => true,
        'hierarchical' => true,
        'show_in_rest' => true,
      )
    );
  }
  add_action( 'init', 'dodaj_taksonomijo_tip_obroka' );
?>
```

Po dodajanju taksonomije *Tip obroka* (in po povezavi taksonomije s tipom prispevka `Recepti`) je ta na voljo v nadzorni plošči v postavki menija Recepti → Tip obroka, kot je prikazano na sliki 8.5.

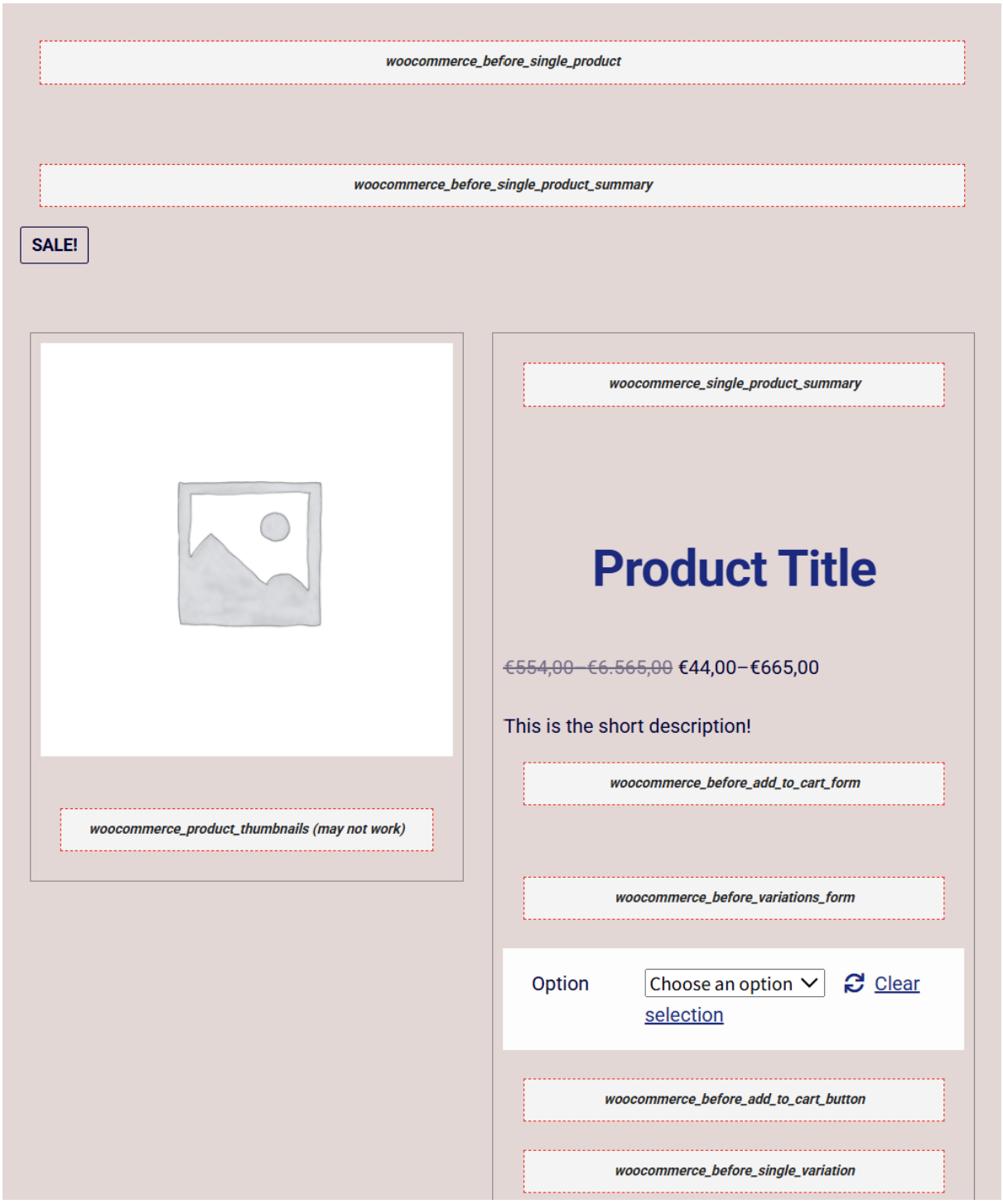
Lasten tip prispevka ali taksonomija? *Lasten tip prispevka* v WordPressu je smiselno uporabiti, kadar se struktura vsebine razlikuje od obstoječih prispevkov, kadar potrebujemo dodatno funkcionalnost (npr. povezavo z e-trgovino), želimo drugačen prikaz ali lažjo administracijo vsebin ter različna dovoljenja za uporabnike. Po drugi strani je smiselno ustvariti novo *taksonomijo*, kadar želimo vsebine s podobno strukturo le bolje organizirati, izboljšati SEO, omogočiti označevanje z oznakami ali hierarhično klasifikacijo ter s tem izboljšati uporabniško izkušnjo in upravljanje vsebine.



Slika 8.5. Prikaz dodane taksonomije pri tipu prispevka `Recept`.

8.4.3 Uporaba dogodkov za posege v delovanje vtičnikov

Nekateri naprednejši vtičniki omogočajo razvijalcem posege v vnaprej določena mesta vtičnika na podoben način, kot je to omogočeno na jedru sistema WordPress. Primer takšnega vtičnika je WooCommerce, vtičnik, ki funkcionalnosti WordPressa razširja z dodajanjem celovite spletne trgovine. Zaradi svoje kompleksnosti razvijalcem omogoča, da sami prilagodijo nekatera mesta dodanih funkcionalnosti spletne trgovine [45]. Celoten seznam dogodkov, ki jih razvijalci lahko uporabijo, je naveden v dokumentaciji vtičnika WooCommerce [46]. Na sliki 8.6 [46] je prikazan vizualni vodič po dogodkih, ki jih WooCommerce omogoča na strani posameznega produkta. Podobni vodiči so na različnih blogih na voljo tudi za podstrani arhiva vseh izdelkov, za strani košarica, zaključek nakupa ipd. [47].



Slika 8.6. Dogodki WooCommerce vtičnika na strani posameznega izdelka [47].

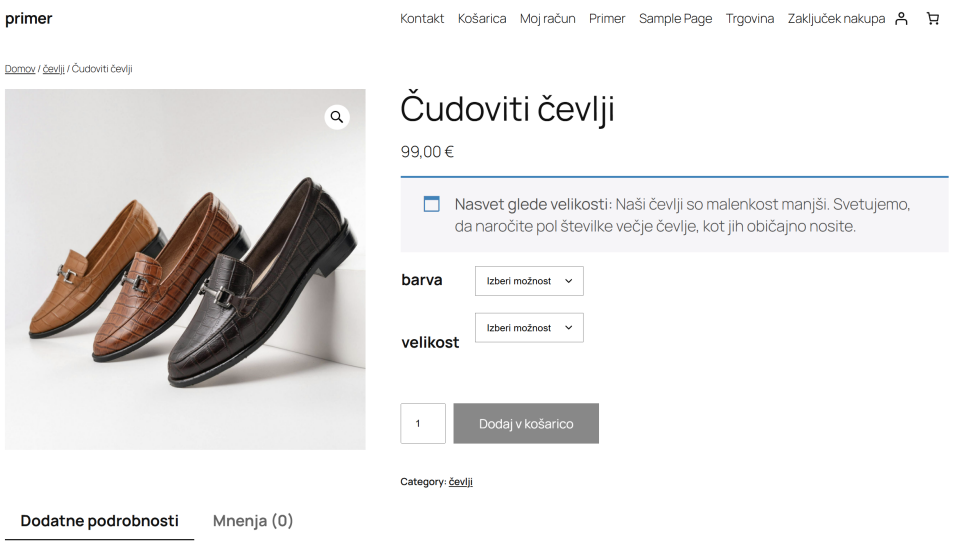
Razumevanje in uporaba dogodkov razvijalcem omogočata dinamično vstavljanje vsebin ali prilagajanje obstoječega vedenja brez potrebe po neposrednih posegih v predloge tem. S tem se povečuje prenosljivost kode in omogoča večja prilagodljivost, ne glede na spremembe tem ali posodobitve vtičnika. Uporabnik lahko s pomočjo teh dogodkov

doda vsebino pred ali po elementih, kot so gumbi za nakup, opisi izdelkov ali elementi na zaključni strani nakupa. Kombinacija dokumentacije [45] in vizualnih prikazov [47] tako omogoča celovit pristop k naprednemu prilagajanju WooCommerce trgovin. Primer 8.4.3 predstavlja uporabo WooCommerce dogodka `woocommerce_before_variations_form` za prikaz izbranega besedila preden uporabnik v spletni trgovini izbere variacijo izdelka (v predstavljenem primeru izbere velikost čevljev). Koda se za preizkus lahko doda v datoteko `functions.php`. Predpogoj za njeno delovanje je aktiven vtičnik WooCommerce, ki uvede dogodek, na katerega se primer navezuje.

Primer 8.4.3 (Primer uporabe dogodka (v `functions.php`).)

```
<?php
  function shoe_size_notice() {
    echo '<div class="woocommerce-info" style="margin-bottom:15px;">
      <strong>Nasvet glede velikosti:</strong>
        Naši čevlji so malenkost manjši.
        Svetujemo, da naročite pol številke večje čevlje,
        kot jih običajno nosite.
       </div>';
  }
  add_action('woocommerce_before_variations_form', 'shoe_size_notice');
?>
```

Sprememba izgleda strani WooCommerce izdelka, ki je bila sprožena na primeru 8.4.3, je prikazana na sliki 8.7.



Slika 8.7. Prilagojen izgled z uporabo dogodka WooCommerce (primer 8.4.3).

8.5 WordPress vsebine za razvijalce

Spletno mesto *WordPress Developer Resources*(https://developer.wordpress.org/) [39] je osrednje razvojno središče za razvijalce, ki želijo graditi teme, vtičnike ali razširitve za WordPress. Ponuja celovito dokumentacijo jedrnih funkcij (kot je npr. `register_post_type()`), razredov in API-jev, varnostne prakse, postopke razvoja z blokovnim urejevalnikom Gutenberg in še mnogo več. Poleg referenc vsebuje tudi priročnike, primere uporabe ter povezave do razvojnega bloga in odprtokodne kode. Zaradi rednih posodobitev in natančne razlage funkcionalnosti je to nepogrešljiv vir za vsak resnejši razvoj v okolju WordPress.

POGLAVJE 9

WordPress kot brezglavi sistem za upravljanje vsebin

V sodobnem spletnem razvoju se WordPress vse pogosteje uporablja kot brezglavi sistem za upravljanje z vsebinami (angl. Headless Content Management System). Pri tem je njegova vloga omejena na shranjevanje in urejanje vsebin, prikaz pa prevzame ločeno razvito obličje. Tak pristop omogoča večjo prilagodljivost, zmogljivejšo uporabniško izkušnjo in boljšo integracijo s sodobnimi tehnologijami, kot so ogrodja JavaScript (npr. React, Next.js ali Vue.js). Pri tem odjemalci najpogosteje dostopajo do podatkov s programskim vmesnikom WordPress REST API ali s programskim vmesnikom GraphQL.

9.1 Dostop do podatkov s programskeim vmesnikom WordPress REST API

WordPress REST API omogoča standardiziran vmesnik za komunikacijo z WordPress spletnimi mesti prek izmenjave podatkov v obliki JSON. Vmesnik zagotavlja t. i. končne točke (angl. Endpoints), ki omogočajo dostop do vgrajenih vsebin, kot so prispevki, strani in taksonomije. Čeprav je zasnovan predvsem za razvijalce, omogoča strukturirano, varno in razširljivo interakcijo z vsebino spletnega mesta brez potrebe po neposredni obdelavi predlog PHP. Javna vsebina je dostopna brez avtentikacije, medtem ko so dostopi do zasebne vsebine, meta podatkov ali uporabniških informacij zaščiteni in zahtevajo ustrezna dovoljenja.

9.1.1 Privzete končne točke

Sistem vsebuje vnaprej definirane končne točke, kot je na primer `/wp-json/wp/v2/posts`, ki omogoča dostop do prispevkov. Pri registraciji lastnih tipov prispevkov je za izpostavitev vsebin treba eksplicitno omogočiti podporo za REST API – običajno z nastavitvijo parametra `'show_in_rest'=> true` – saj se ti tipi sicer ne vključijo samodejno v razpoložljive končne točke API-ja. Pri tem je mogoče definirati tudi obliko končne točne (npr. `'rest_base'=> 'projekti'`). Pridobivanje prispevkov je prikazano na primeru 9.1.1.

Primer 9.1.1 (Izsek odziva za zahtevek `/wp-json/wp/v2/posts` prek vmesnika WordPress REST API.)


```
[
  {
    "id": 1,
    "date": "2025-07-20T12:58:04",
    "date_gmt": "2025-07-20T12:58:04",
    "guid": { "rendered": "http://primer.local/?p=1" },
    "modified": "2025-07-20T12:58:04",
    "modified_gmt": "2025-07-20T12:58:04",
    "slug": "primer-prispevka",
    "status": "publish",
    "type": "post",
    "link": "http://primer.local/primer-prispevka/",
    "title": { "rendered": "Primer prispevka" },
    "content": {
      "rendered": "\n\u003Cp\u003EDobrodošli v svet WordPressa!\u003C/p\u003E\n",
      "protected": false
    },
    "excerpt": {
      "rendered": "\n\u003Cp\u003EDobrodošli v svet WordPressa!\u003C/p\u003E\n",
      "protected": false
    },
    "author": 1,
    "featured_media": 0,
    "comment_status": "open",
    "ping_status": "open",
```

Primer (nadaljevanje)

```
    "sticky": false,
    "template": "",
    "format": "standard",
    ...
  }
]
```

Prav tako je možno dostopati do taksonomij. Za pridobitev seznama vseh razpoložljivih taksonomij uporabimo končno točko `GET /wp/v2/taxonomies`. Odgovor vsebuje podatke o vseh registriranih taksonomijah v sistemu (npr. kategorije, oznake ali lastne taksonomije). Če želimo pridobiti podrobnosti o točno določeni taksonomiji, uporabimo končno točko z identifikatorjem `GET /wp/v2/taxonomies/<taksonomija>`. Primer rezultatov dostopa do končne točke `http://primer.local/wp-json/wp/v2/categories` je prikazan na primeru 9.1.2. Prikazuje podrobnosti prve izmed dodanih kategorij – »Glavna jed«.

Primer 9.1.2 (Odsek odziva za zahtevek `/wp-json/wp/v2/categories` z vmesnikom WordPress REST API.)

```
[
  {
    "id": 23,
    "count": 1,
    "description": "",
    "link": "http://primer.local/category/recepti/glavne-jedi/",
    "name": "Glavne jedi",
    "slug": "glavne-jedi",
    "taxonomy": "category",
    "parent": 21,
    "meta": [],
    "acf": [],
    "_links": {
      "self": [
        {
         "href": "http://primer.local/wp-json/wp/v2/categories/23",
         "targetHints": { "allow": [ "GET" ] }
```

Primer (nadaljevanje)


```
        }
      ],
      "collection": [
        { "href": "http://primer.local/wp-json/wp/v2/categories" }
      ],
      "about": [
        { "href": "http://primer.local/wp-json/wp/v2/taxonomies/category" }
      ],
      "up": [
        {
          "embeddable": true,
          "href": "http://primer.local/wp-json/wp/v2/categories/21"
        }
      ],
      "wp:post_type": [
        {
          "href": "http://primer.local/wp-json/wp/v2/posts?categories=23"
        }
      ]
    }
  }
]
```


Vmesnik WordPress REST API poleg dostopa do prispevkov omogoča tudi dostop do drugih virov, kot so medijske datoteke (na primer slike, dokumenti, zvok in video), ki so v sistemu shranjene kot poseben tip prispevkov. Vsak medijski element vsebuje strukturirane podatke, kot so ID, naslov, povezava, tip datoteke, tip MIME (angl. Multipurpose Internet Mail Extensions), metapodatki (npr. dimenzije slike) ter spremljajoča besedila (alternativen opis, napis). Dostop do medijskih datotek je mogoč prek končne točke `/wp/v2/media` [48]. Z njo lahko:

- pridobimo seznam datotek (`GET /wp/v2/media`), kjer so rezultati omejeni s parametri za filtriranje, paginacijo, avtorja, status, tip ali tip MIME;
- ustvarimo nov medijski element (`POST /wp/v2/media`), pri čemer se poleg datoteke po želji podajo tudi dodatni podatki, kot so naslov, alternativen opis ali povezava na prispevek;

- pridobimo posamezen element (`GET /wp/v2/media/id`), kjer se vrnejo vse informacije o določeni datoteki;
- posodobimo element (`POST /wp/v2/media/id`), npr. za spremembo napisa, alternativnega opisa ali povezanega prispevka;
- izbrišemo element (`DELETE /wp/v2/media/id`), bodisi z uporabo koša bodisi z neposredno trajno odstranitvijo (`force=true`).

Na ta način vmesnik WordPress REST API omogoča, da so medijske datoteke enako dostopne in upravljive kot prispevki in druge vrste vsebin.

9.1.2 Globalni parametri, paginacija in razvrščanje

Pri dostopu do podatkov preko vmesnika WordPress REST API je na voljo veliko globalnih parametrov [49], ki veljajo za vse končne točke in omogočajo nadzor nad načinom pridobivanja in oblikovanja podatkov. Poleg tega je pri delu z večjimi zbirkami virov nujno razumevanje paginacije, ki omogoča razdelitev podatkov na obvladljive dele. Globalni parametri (imenovani tudi metaparametri) delujejo na ravni celotnega odziva in so na voljo pri vseh končnih točkah. Najpogosteje uporabljeni parametri so predstavljeni v preglednici 9.1.

Preglednica 9.1. Pregled najpogostejših globalnih parametrov vmesnika WordPress REST API [49].

Parameter	Opis in primer uporabe
_fields	Omogoča pridobivanje samo določenih polj vira, kar zmanjša obseg vrnjenih podatkov in izboljša učinkovitost. *Primer:* `/wp/v2/posts?_fields=id,title,excerpt,link`
_embed	Poleg osnovnega vira vrne tudi povezane vire (npr. avtor ali taksonomije), kar zmanjša število potrebnih zahtevkov. *Primer:* `/wp/v2/posts?_embed=author,wp:term`
_method	(ali glava `X-HTTP-Method-Override`) Omogoča združljivost kadar strežnik ali odjemalec ne podpira vseh HTTP metod (npr. `DELETE`). *Primer:* `/wp/v2/posts/42?_method=DELETE`
_envelope	Vse podatke odziva (telo, glave, status) vrne v JSON obliki znotraj telesa, kar je uporabno pri omejitvah posrednikov ali odjemalcev.

Pri delu z večjo količino podatkov vmesnik WordPress REST API privzeto vrne omejeno število virov na stran (privzeto 10). To omogoča enostavnejše nalaganje in obdelavo podatkov. Paginacijo nadzorujemo s parametri, predstavljenimi v preglednici 9.2.

Preglednica 9.2. Parametri za paginacijo v vmesniku WordPress REST API [50].

Parameter	Opis in primer uporabe
?page=	Določi stran rezultatov v zbirki. *Primer:* `/wp/v2/posts?page=2`
?per_page=	Določi število vrnjenih zapisov na stran (veljavne vrednosti: 1–100). *Primer:* `/wp/v2/posts?per_page=5`
?offset=	Določi zamik, od katerega naj se začnejo vračati rezultati. *Primer:* `/wp/v2/posts?offset=15`

Za nadzor nad vrstnim redom rezultatov sta na voljo še dva dodatna parametra. Parameter `?order=` določa naraščajoč (`asc`) ali padajoč (`desc`) vrstni red. Parameter `?orderby=` pa določa lastnost, po kateri so rezultati razvrščeni (npr. `date`, `title`, `id`, `slug`). Vsak paginiran odziv vključuje tudi glavi `X-WP-Total` (skupno število zapisov) in `X-WP-TotalPages` (skupno število strani), kar omogoča natančno upravljanje z nadaljnjimi zahtevki [50].

9.1.3 Avtentikacija in dovoljenja

Vmesnik WordPress REST API privzeto omogoča prost dostop do javno objavljenih vsebin, kot so prispevki in strani. V primeru dostopanja do zaščitenih virov ali izvedbe sprememb je potrebna avtentikacija. Standardna metoda avtentikacije v Wordpressu je *avtentikacija s piškotki*, ki se uporablja znotraj okolja WordPress, kadar je uporabnik prijavljen v nadzorno ploščo. Avtentikacija za dostop prek REST API vključuje vrednosti nonce (angl. Number Used Once) [51], ki preprečujejo napade večdomenskega ponarejanja zahtevkov oziroma napade CSRF (angl. Cross-Site Request Forgery). Nonce vrednosti so enkratna kriptografsko števila, ki v Wordpressu delujejo kot varnostni žetoni, ki so vezani na uporabnika in akcijo ter služijo zaščiti naslove URL in obrazcev pred zlorabami. Pri uporabi vtičnikov in tem je podpora temu mehanizmu vgrajena, medtem ko je pri ročnih zahtevkih AJAX nonce treba posredovati v glavi zahteve (`X-WP-Nonce`) ali kot

parameter. Avtentikacija s piškotki je zanesljiva izbira za razvoj rešitev, ki delujejo znotraj WordPressa [52].

Od različice 5.6 WordPress vključuje tudi podporo za *applikacijska gesla*, ki omogočajo osnovno avtentikacijo HTTP prek varne povezave HTTPS. Gesla za aplikacije se generirajo v uporabniškem profilu in so priporočena rešitev za varno povezovanje oddaljenih aplikacij s sistemom.

Za naprednejše integracije so na voljo *vtičniki za avtentikacijo*, kot sta OAuth 1.0 [53] in JSON Web Tokens (JWT) [54], ki omogočajo prilagodljive in standardizirane pristope, zlasti v scenarijih, kjer so potrebne bolj kompleksne oblike avtorizacije.

V profilu uporabnika v nadzorni plošči obstaja tudi možnost dodajanja gesla aplikacij, ki je primerno predvsem za razvoj in testiranje, saj zahteva posredovanje uporabniškega imena in gesla pri vsaki zahtevi [52].

9.1.4 Praktični primer uporabe podatkov na ločenem obličju

Primer 9.1.3 prikazuje preprosto uporabo sistema WordPress kot brezglavega sistema za upravljanje vsebin. Vsebinski prispevki so pridobljeni iz končne točke, ki jih nato s pomočjo jezika JavaScript prikažemo na preprostem obličju.

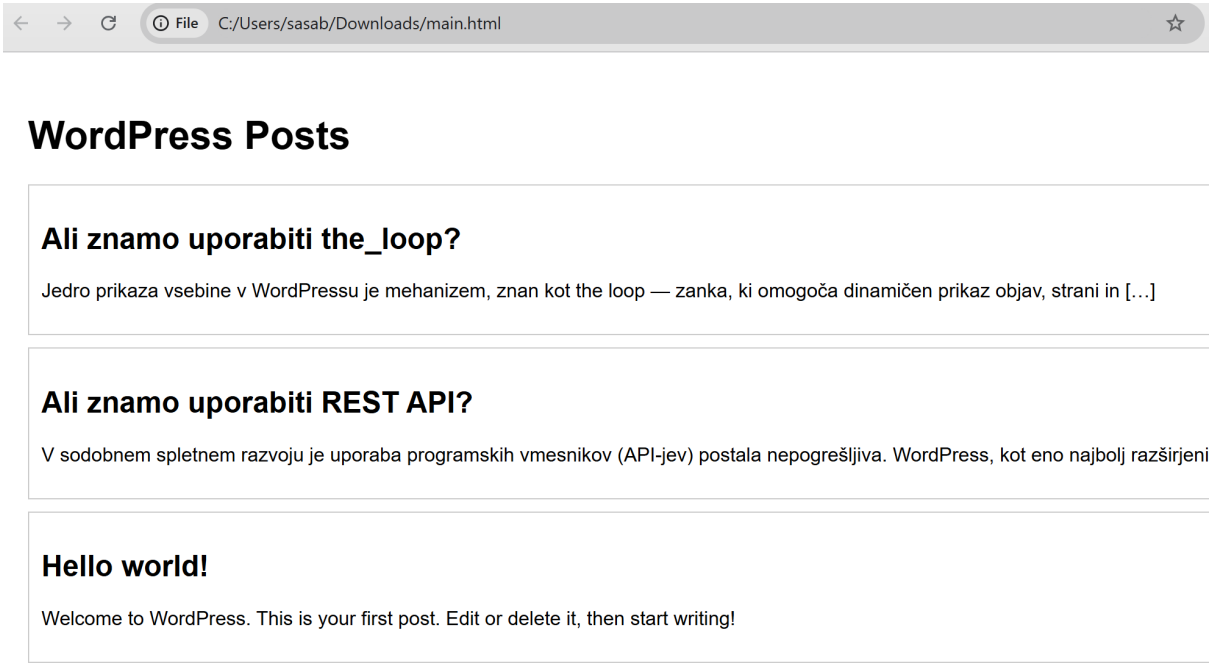
Primer 9.1.3 (Prikaz podatkov iz vmesnika WordPress REST API na preprostem obličju.)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Posts</title>
  <style>
    body {
      font-family: sans-serif;
      padding: 20px;
    }
    .post {
      border: 1px solid #ccc;
      padding: 10px;
      margin-bottom: 10px;
    }
  </style>
</head>

<body>
  <h1>WordPress Posts</h1>
  <div id="posts"></div>

<script>
fetch('http://primer.local/wp-json/wp/v2/posts')
  .then(res => res.json())
  .then(data => {
    const container = document.getElementById('posts');
    data.forEach(post => {
    const div = document.createElement('div');
    div.className = 'post';
     div.innerHTML = `
        <h2>${post.title.rendered}</h2>
        ${post.excerpt.rendered}
      `;
      container.appendChild(div);
    });
   })
```

Primer (nadaljevanje)

```
\end{vr}
      .catch(err => {
        document.body.innerHTML += '<p style="color:red;">
        Error: ' + err.message + '</p>';
      });
  </script>
</body>
</html>
```

Izgled zgoraj predstavljenega preprostega primera podatkov končne točke prispevkov je prikazan na sliki 9.1.

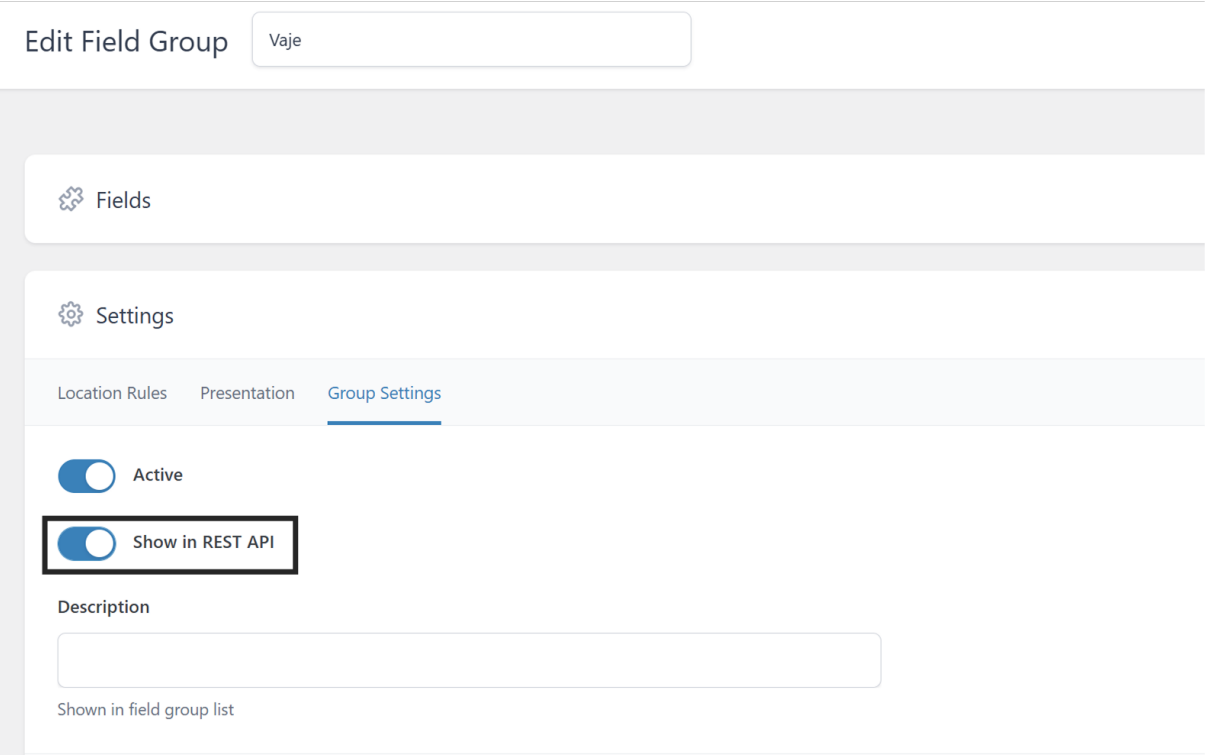


Slika 9.1. Izgled primera uporabe podatkov brezglavega sistema WordPress (primer 9.1.3).

Podobno bi lahko dostopali do vseh omogočenih tipov prispevkov. Za lastne tipe vsebin (npr. `Recepti`), ki so bili dodani v kodi, je treba zagotoviti, da je pri definiciji tipa prispevka v funkciji `register_post_type` omogočena izpostavitev podatkov prek vmesnika WordPress REST API. To je doseženo s `'show_in_rest'=> true`. Za primer kode, ki je bil uporabljen za dodajanje lastnega tipa prispevka `Recepti`, se vrnite k

primeru 8.4.2.

Če je za upravljanje s podrobnostmi tipa prispevka uporabljen vtičnik ACF, je ključno, da je v nastavitvah omogočena izpostavitev podatkov prek vmesnika WordPress REST API, kot je prikazano na sliki 9.2.



Slika 9.2. Omogočanje REST izpostavitve lastnega tipa prispevka `Recept` v ACF.

Zaključek

To učno gradivo predstavlja zgoščeno, a hkrati dostopno izhodišče za razumevanje in uporabo sistema WordPress v sodobnem spletnem razvoju. Njegov namen je bralcu približati WordPress ne le kot orodje za objavljanje spletnih vsebin, temveč kot zmogljivo in prilagodljivo razvojno okolje. S sistematičneim uvajanjem ključnih funkcionalnosti gradivo ponuja celovit vpogled v možnosti, ki jih WordPress kot sistem za upravljanje vsebin ponuja razvijalcem. Vsebina gradiva se po zahtevnosti stopnjuje – od namestitve, osnovne uporabe sistema WordPress, priprave vsebin, izbire teme in vtičnikov do naprednejših primerov uporabe, vključno z ustvarjanjem in uporabo podteme, ustvarjanjem in uporabo lastnih vtičnikov, prilagajanjem delovanja WordPressa z uporabo dogodkov in uporabo WordPressa kot brezglavega sistema za upravljanje vsebin. Na ta način gradivo presega zgolj klasično obravnavo urejevalnika vsebin in uporabnika v vlogi skrbnika spletne strani. Bralce spodbujamo, da WordPress razumejo kot modularni sistem, ki ga je mogoče razširiti, nadgraditi in vključiti v različna digitalna okolja – od preprostih spletnih strani do spletnih trgovin in spletnih aplikacij po meri. Poleg teoretičnih razlag smo se avtorji v gradivo trudili vključiti več praktičnih primerov, izvorno kodo za pogoste prilagoditve in vizualne predstavitve, s čimer upamo, da smo olajšali učenje tudi začetnikom. Čeprav učno gradivo tega ne zajema, bralce spodbujamo, da pridobijo znanja, povezana predvsem z optimizacijo in varnostjo spletnih mest, temelječih na WordPressu – dva vidika, ki sta še posebej pomembna v svetu WordPressa. Čeprav se WordPress kot platforma nenehno razvija z novimi različicami, prenovljenim uporabniškim vmesnikom, izboljšavami varnosti in dodajanjem novih funkcionalnosti, temeljne arhitekturne zasnove ter razvojni pristopi ostajajo razmeroma stabilni. Osnove, kot so razširljivost prek tem in vtičnikov, uporaba funkcij in dogodkov, delo s podatki prek vmesnika REST API ter koncepti, kot so teme, vtičniki, taksonomije, predloge in dogodki tvorijo jedro sistema, ki se skozi leta bistveno ne spreminja. Tudi če se bodo posamezne podrobnosti sistema s časom spremenile, bo znanje iz učnega gradiva ostalo uporabno, saj bralcem podaja temeljna načela.

To učno gradivo obravnava širok nabor tem, povezanih z razvojem v okolju WordPress, vendar kljub temu ostajajo tudi področja, ki so bila zavestno izpuščena ali zgolj nakazana, a si zaslužijo nadaljnjo obdelavo. Mednje spadajo predvsem testiranje in nadzor kakovosti, avtomatizirani varnostni pregledi, statična analiza kode ter osnovni funkcionalni testi uredniških tokov. Prav tako bi bilo smiselno v prihodnje podrobneje obravnavati zmogljivostne in optimizacijske vidike, kot so predpomnjenje, optimizacija obdelave medijskih vsebin in spremljanje odzivnosti sistema. Posebno pozornost si zasluži tudi tematika migracij, kjer bi lahko bolj operativno predstavili procese uvajanja sprememb na produkcijsko okolje, izdelavo varnostnih kopij in vzpostavitev povratnih scenarijev ob morebitnih težavah. Ta področja predstavljajo nadaljevalne vsebine, namenjene naprednejšim uporabnikom in razvijalcem, ki delujejo v kompleksnejših okoljih.

Literatura

[1] WordPress.org. *FAQ About WordPress*. Dostopano: 19. 7. 2025. URL: https://codex.wordpress.org/FAQ_About_WordPress.

[2] Free Software Foundation. *GNU General Public License, Version 3*. Dostopano: 19. 7. 2025. URL: https://www.gnu.org/licenses/gpl-3.0.html.

[3] WordPress.org. *WordPress Features*. Dostopano: 19. 7. 2025. URL: https://WordPress.org/about/features.

[4] W3Techs. *Usage Statistics and Market Share of WordPress*. Dostopano: 9. 9. 2025. URL: https://w3techs.com/technologies/details/cm-WordPress.

[5] WordPress.org. *Server Requirements*. Dostopano: 19. 7. 2025. URL: https://WordPress.org/about/requirements.

[6] WordPress.org. *Roles and Capabilities*. Dostopano: 19. 7. 2025. URL: https://WordPress.org/documentation/article/roles-and-capabilities.

[7] Ivana Boštjančič Pulko in sod. *Priročnik kibernetske varnosti*. Elektronska izdaja. Ljubljana: Urad Vlade Republike Slovenije za informacijsko varnost, 2025. URL: https://www.gov.si/assets/vladne-sluzbe/URSIV/Datoteke/Prirocnik-kibernetske-varnosti.pdf.

[8] Diah Aryani in sod. »Comparative Analysis Of On-Page And Off-Page White Hat Search Engine Optimization (SEO) Techniques On Website Popularity«. V: *International Journal of Science, Technology and Management* 4 (maj 2023), str. 527–533. DOI: 10.46729/ijstm.v4i3.815.

[9] Elementor. *WordPress SEO Best Practices*. 2023. URL: https://elementor.com/blog/wordpress-seo-best-practices/ (pridobljeno 18. 12. 2025).

[10] Google. *SEO Starter Guide*. 2024. URL: https://developers.google.com/search/docs/fundamentals/seo-starter-guide (pridobljeno 18. 12. 2025).

[11] Google. *Lighthouse*. 2024. URL: https://developer.chrome.com/docs/lighthouse/overview/ (pridobljeno 18. 12. 2025).

[12] WordPress.org. *WordPress Files*. Dostopano: 25. 7. 2025. URL: https://codex.wordpress.org/WordPress_Files.

[13] Karishma Sundaram. *Beginner's Guide to Understanding the Structure of a WordPress Site*. Dostopano: 25. 7. 2025. URL: https://www.malcare.com/blog/beginners-guide-to-understanding-the-structure-of-a-WordPress-site.

[14] WordPress.org. *Database Description*. Dostopano: 25. 7. 2025. URL: https://codex.wordpress.org/Database_Description.

[15] WordPress.org. *Template Hierarchy*. Dostopano: 25. 7. 2025. URL: https://developer.wordpress.org/themes/basics/template-hierarchy.

[16] Wordpress.org. *Overview of WordPress Block Theme Terms and Hierarchy*. Dostopano: 11. 10. 2025. URL: https://learn.wordpress.org/lesson/overview-of-wordpress-block-theme-terms-and-hierarchy.

[17] WordPress.org. *Theme Structure*. Dostopano: 13. 10. 2025. URL: https://developer.wordpress.org/themes/core-concepts/theme-structure.

[18] WordPress.org. *Global Settings and Styles*. Dostopano: 13. 10. 2025. URL: https://developer.wordpress.org/themes/core-concepts/global-settings-and-styles.

[19] cPanel, L.L.C. *cPanel – The Hosting Platform of Choice*. Dostopano: 25. 7. 2025. URL: https://www.cpanel.net.

[20] Behrouz Forouzan. *TCP/IP: Protocol Suite*. 1st. New Delhi, India: Tata McGraw-Hill Publishing Company Limited, 2000.

[21] FileZilla Project. *FileZilla – The free FTP solution*. Dostopano: 25. 7. 2025. URL: https://filezilla-project.org.

[22] WordPress.org. *SSH Access – Developer Tools*. Dostopano: 25. 7. 2025. URL: https://developer.WordPress.com/docs/developer-tools/ssh.

[23] WordPress.org. *WP-CLI – Developer Tools*. Dostopano: 25. 7. 2025. URL: https://developer.WordPress.com/docs/developer-tools/wp-cli.

[24] WordPress.org. *Key Concepts – Block Editor Architecture*. Dostopano: 21. 7. 2025. URL: https://developer.wordpress.org/block-editor/explanations/architecture/key-concepts.

[25] WordPress.org. *WordPress Roadmap*. Dostopano: 21. 7. 2025. URL: https://WordPress.org/about/roadmap.

[26] WordPress.org. *Plugins*. Dostopano: 19. 7. 2025. URL: https://WordPress.org/plugins.

[27] W3Techs. *Usage of Elementor*. Dostopano: 19. 7. 2025. URL: https://w3techs.com/technologies/details/cm-elementor.

[28] Elementor Ltd. *Elementor Pricing Plans*. Dostopano: 23. 7. 2025. URL: https://elementor.com/pricing-plugin.

[29] WordPress.org. *WP-CLI Commands*. Dostopano: 13. 10. 2025. URL: https://developer.wordpress.org/cli/commands.

[30] WordPress.org. *WP-CLI – Scaffold Child Theme*. Dostopano: 13. 10. 2025. URL: https://developer.wordpress.org/cli/commands/scaffold/child-theme.

[31] WordPress.org. *register_post_type()*. Dostopano: 25. 7. 2025. URL: https://developer.wordpress.org/reference/functions/register_post_type.

[32] WordPress.org. *Introduction to Plugins*. Dostopano: 25. 7. 2025. URL: https://WordPress.org/documentation/article/introduction-to-plugins.

[33] WordPress.org. *Plugin Developer Handbook*. Dostopano: 25. 7. 2025. URL: https://developer.wordpress.org/plugins.

[34] WordPress.org. *Settings API*. Dostopano: 25. 7. 2025. URL: https://developer.wordpress.org/plugins/settings/settings-api.

[35] WordPress.org. *Actions*. Dostopano: 25. 7. 2025. URL: https://developer.wordpress.org/plugins/hooks/actions.

[36] WordPress.org. *Filters*. Dostopano: 25. 7. 2025. URL: https://developer.wordpress.org/plugins/hooks/filters.

[37] WordPress.org. *Hooks*. Dostopano: 25. 7. 2025. URL: https://developer.wordpress.org/plugins/hooks.

[38] WordPress.org. *Function Reference*. Dostopano: 13. 10. 2025. URL: https://developer.wordpress.org/reference/functions.

[39] WordPress.org. *WordPress Developer Resources*. Dostopano: 25. 7. 2025. URL: https://developer.wordpress.org.

[40] WordPress Developers. *WP Query Class – More Information*. Dostopano: 13. 10. 2025. 2025. URL: https://developer.wordpress.org/reference/classes/wp_query/#more-information.

[41] *Search results for "WP_Theme"*. Dostopano: 13. 10. 2025. WordPress.org — Developer Resources. 2025. URL: https://developer.wordpress.org/?s=WP_Theme.

[42] WordPress.org. *Advanced Custom Fields – WordPress plugin*. Dostopano: 25. 7. 2025. URL: https://WordPress.org/plugins/advanced-custom-fields.

[43] WPEngine, Inc., Advanced Custom Fields. *get_field()*. Dostopano: 25. 7. 2025. URL: https://www.advancedcustomfields.com/resources/get_field.

[44] WordPress.org. *Function Reference – register_taxonomy*. Dostopano: 28. 7. 2025. URL: https://developer.wordpress.org/reference/functions/register_taxonomy.

[45] WooCommerce Documentation Team. *Introduction to Hooks, Actions and Filters*. Dostopano: 25. 7. 2025. URL: https://woocommerce.com/document/introduction-to-hooks-actions-and-filters.

[46] WooCommerce Developers. *WooCommerce Hook Reference*. Dostopano: 25. 7. 2025. URL: https://woocommerce.github.io/code-reference/hooks/hooks.html.

[47] Rodolfo Melogli. *WooCommerce Visual Hook Guide: Single Product Page*. Dostopano: 25. 7. 2025. URL: https://www.businessbloomer.com/woocommerce-visual-hook-guide-single-product-page.

[48] WordPress.org. *REST API Reference – Media*. Dostopano: 28. 7. 2025. URL: https://developer.wordpress.org/rest-api/reference/media.

[49] WordPress.org. *REST API Handbook – Global Parameters*. Dostopano: 28. 7. 2025. URL: https://developer.wordpress.org/rest-api/using-the-rest-api/global-parameters.

[50] WordPress.org. *REST API Handbook – Pagination*. Dostopano: 28. 7. 2025. URL: https://developer.wordpress.org/rest-api/using-the-rest-api/pagination.

[51] WordPress.org. *Common APIs Handbook – Nonces*. Dostopano: 28. 7. 2025. URL: https://developer.wordpress.org/apis/security/nonces.

[52] WordPress.org. *REST API Handbook – Authentication*. Dostopano: 28. 7. 2025. URL: https://developer.wordpress.org/rest-api/using-the-rest-api/authentication.

[53] OAuth Core Workgroup. *OAuth Core 1.0 Revision A*. Dostopano: 28. 7. 2025. URL: https://oauth.net/core/1.0a.

[54] OAuth Core Workgroup. *Auth0 documentation – JSON Web Tokens*. Dostopano: 28. 7. 2025. URL: https://auth0.com/docs/secure/tokens/json-web-tokens.

[55] WordPress.org. *Themes*. Dostopano: 19. 7. 2025. URL: https://WordPress.org/themes.

[56] WordPress.org. *REST API Handbook*. Dostopano: 28. 7. 2025. URL: https://developer.wordpress.org/rest-api.

[57] NEOSERV. *WordPress "child" tema: zakaj in kako jo ustvariti?* Dostopano: 6. 9. 2025. URL: https://www.neoserv.si/blog/wordpress-child-tema.

[58] WordPress Contributors. *Plugin Handbook*. Dostopano: 13. 10. 2025. WordPress.org. 2025. URL: https://developer.wordpress.org/plugins/.

Stvarno kazalo

DOI
https://doi.org/
10.18690/um.feri.3.2026

ISBN
978-961-299-106-7

Uvod v WordPress

Saša Brdnik, Tjaša Heričko, Špela Čučko, Sašo Karakatič

Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, Slovenija
sasa.brdnik@um.si, tjasa.hericko@um.si, spela.cucko@um.si, saso.karakatic@um.si

Učbenik celovito in sistematično predstavlja sodoben sistem za upravljanje vsebin WordPress, namenjen študentom, učiteljem in začetnikom v razvoju spletnih mest. Obravnava temeljne koncepte delovanja WordPressa, od lokalne namestitve in osnovnih nastavitev do upravljanja vsebin, uporabnikov in medijev. Poseben poudarek je namenjen temam, vtičnikom in prilagajanju funkcionalnosti, vključno z osnovami razvoja lastnih vtičnikov, tipov vsebin in taksonomij. Učbenik se dotakne tudi optimizacije za iskalnike ter dobrih praks pri vzdrževanju spletnih mest. Vsebina je podprta s številnimi praktičnimi primeri, posnetki zaslona in razlagami, ki omogočajo postopno razumevanje snovi. Učbenik bralca vodi od osnov do naprednejših konceptov ter omogoča samostojno izdelavo in upravljanje profesionalnega WordPress spletnega mesta. Namenjen je tako pedagoški rabi kot tudi praktični uporabi v realnih projektih.

Ključne besede:
WordPress,
sistem za upravljanje vsebin,
praktični primeri,
CMS,
učbenik

Univerzitetna založba Univerze v Mariboru

DOI
https://doi.org/
10.18690/um.feri.1.2026

ISBN
978-961-299-106-7

Keywords:
WordPress,
content management
system,
pracrtical exmaples,
CMS,
textbook

University of Maribor Press

Introduction to WordPress

Saša Brdnik, Tjaša Heričko, Špela Čučko, Sašo Karakatič

University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia
sasa.brdnik@um.si, tjasa.hericko@um.si, spela.cucko@um.si, saso.karakatic@um.si

The textbook provides a comprehensive and systematic introduction to the modern WordPress content management system, aimed at students, teachers, and beginners in website development. It covers the basic concepts of WordPress, from local installation and basic settings to content, user, and media management. Special emphasis is placed on themes, plugins, and customizing functionality, including the basics of developing your own plugins, content types, and taxonomies. The textbook also touches on search engine optimization and best practices for website maintenance. The content is supported by numerous practical examples, screenshots, and explanations that enable a gradual understanding of the material. The textbook guides the reader from the basics to more advanced concepts and enables the independent creation and management of a professional WordPress website. It is intended for both educational use and practical application in real projects.

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko