Ostvariti sustav za upravljanjem Web sadržajem

Petrač, Antun

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1266

OSTVARITI SUSTAV ZA UPRAVLJANJEM WEB SADRŽAJEM

Antun Petrač

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1266

OSTVARITI SUSTAV ZA UPRAVLJANJEM WEB SADRŽAJEM

Antun Petrač

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1266

Pristupnik:	**Antun Petrač (0036526877)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Leonardo Jelenković

Zadatak: **Ostvariti sustav za upravljanjem Web sadržajem**

Opis zadatka:

Istražiti mogućnosti za izgradnju sustava za upravljanjem sadržaja (Content management system - CMS) preko Weba korištenjem programskog jezika PHP. Osmisliti i ostvariti takav sustav koji će omogućavati administratoru prilagođavanje izgleda i razmještaj elemenata na web-stranici, a korisnicima dodavanje i uređivanje sadržaja. Sadržaj može biti u oblicima web-stranica, poveznica u izbornicima te informacija koje sadržavaju formatirani tekst i slike. Napraviti kratke upute za administratore i korisnike ostvarenog sustava.

Rok za predaju rada: 14. lipnja 2024.

SADRŽAJ

1. Uvod **1**

2. Tehnologije **2**

2.1. Kratki opis tehnologija 2

2.2. Razlozi odabira tehničkog stoga 3

2.2.1. Mogućnosti prilagodbe i fleksibilnosti 3

2.2.2. Dizajn i stil 3

2.2.3. Performanse 3

2.2.4. Sigurnost 4

2.2.5. Zajedničko razvojno iskustvo i ekosustav 4

3. Dizajn Stranice **5**

3.1. Određivanje web ruta 5

3.2. Određivanje komponenti 5

3.3. Određivanje teme 6

3.4. Cjepkanje dizajna na dijelove 6

3.4.1. Dodavanje dodatno potrebnih komponenti 6

3.4.2. Određivanje najkompliciranijih točaka dizajna 6

3.4.3. Određivanje modela 7

4. Struktura i implementacija aplikacije **10**

4.1. Dijagram klasa i pregled sustava 10

4.2. Implementacija modela i glavnih modela Stranica i Blok 11

4.2.1. Model Stranica 14

4.2.2. Model Blok 14

4.3. Povezivanje Backend dijela aplikacije s Frontend-om 16

5. Zaključak **18**

Literatura **19**

Sažetak **20**

Summary **20**

PRIVITAK A: UPUTE ZA UPRAVLJANJE CMS SUSTAVOM **21**

1. UVOD

Motivacija ovoga rada temelji se na istraživanju mogućnosti za izgradnju sustava za upravljanje sadržajem (engl. Content management system - CMS) preko Weba korištenjem programskog jezika PHP. Sama ideja rada bila je osmisliti i ostvariti sustav koji će omogućavati administratoru prilagođavanje izgleda i razmještaj elemenata na web-stranici, a korisnicima dodavanje i uređivanje sadržaja. Sadržaj koji je potrebno omogućiti može biti u oblicima web-stranica, poveznica u izbornicima te informacija koje sadržavaju formatirani tekst i slike. Inspiracija dolazi od kreiranja web stranica korištenjem gotovih platformi kao što su WordPress i WIX. Zbog nezadovoljstva kompleksnosti upoznavanja sustava, nemogućnosti rješavanja postojećih problema, korištenja vanjskih dodataka (engl. plugins) koji udaljavaju developera od koda i raznih drugih problema, zaključak je naći drugo rješenje za sve navedene probleme. Cilj do kojega je u radu planirano doći je kreirati alat u kojemu se nacrtani dizajn lagano može spojiti s modernim i pristupačnim CMS sustavom jer u modernom razvoju Weba statične stranice ne zadovoljavaju potrebu kao što je često dodavanje i promjena aktualnog sadržaja, bila to nekakva novost, galerija, tekst ili slično.

Tijekom korištenja platforme WordPress programer se mora upoznati s programskim jezikom PHP. Za programera programiranje nije korištenje gotovih sučelja, nego je pisanje koda, što je lakše i intuitivnije za realizaciju svih potrebnih funkcija koje se trebaju implementirati. Nakon istraživanja mogućnosti koje PHP jezik nudi i načinom na koji radi, otkriva se okvir (engl. Framework) po imenu Laravel koji uvelike olakšava pisanje koda u PHP jeziku. Sve je jednostavnije i jasnije, od MVC strukture do web ruta, migracija, povezivanja s bazom i mnogog drugog. Problemu je potrebno pristupiti koristeći principe programiranja KISS („Keep it simple, stupid“), DRY („Don't repeat yourself“) i YAGNI („You aren't gonna need it“). KISS princip pomogao je u smislu ne odlaska u veliku širinu i kompliciranju stvari. DRY princip osigurao je manju količinu koda za provjeru i ispravke i naravno, manju težinu aplikacije. YAGNI princip smanjio je vrijeme koje se nije potrošilo na implementaciju stvari koje nisu krucijalne za dostizanje cilja.

U prvome poglavlju rada „Tehnologije“ ukratko ću navesti tehnički stog (engl. tech stack) i objasniti zašto je odabran. Drugo poglavlje prikazat će cjepkanje kompleksnosti traženog dizajna i podjelu sadržaja u modele za postavljanje temelja za implementaciju CMS sustava. Treće poglavlje će pokazati kako sustav radi, odnosno kako sprema i predaje podatke komponentama Frontenda. U zadnjem poglavlju „Zaključak“ sumirat ću napravljeno i navest ću moguća poboljšanja.

2. TEHNOLOGIJE

Za početak svakog projekta bitno je odabrati alat s kojim će se projekt najbolje, najjednostavnije i najtočnije moći odraditi pa iz tog razloga kažemo da se odabir tehnologije bazira na samoj potrebi projekta, a ne nužno na našemu poznavanju određene tehnologije. Ovaj projekt zahtjeva okolinu koja je sigurna, brza i jednostavna. Nakon nezadovoljstva s gotovim platformama i proučavanja novih mogućih odabran je slijedeći tehnički stog:

- PHP okvir Laravel
- Alatni set za CMS otvorenog koda za Laravel TwillCMS
- CSS okvir Tailwind CSS
- Baza podataka MySQL

2.1. Kratki opis tehnologija

Laravel je PHP okvir za web aplikacije poznat po svojoj jednostavnoj sintaksi, moćnim značajkama i širokom ekosustavu. Olakšava posao prilikom odrađivanja uobičajenih zadataka kao što su usmjeravanje (engl routing), autentifikacija, sesija (engl. session) i predmemorija (engl. cache).

TwillCMS je sustav za upravljanje sadržajem otvorenog koda izgrađen na Laravel-u. Omogućuje fleksibilno i intuitivno administratorsko sučelje za upravljanje sadržajem. Kroz svoje sučelje nudi moderne alate za uređivanje sadržaja kao što su prilagodljiva polja obrazaca i upravljanje medijskom bibliotekom.

TailwindCSS je CSS okvir koji se temelji na korisničkim pomoćnim klasama i omogućuje kreiranje prilagođenog dizajna izravno u HTML-u. Omogućuje brz razvoj sučelja s naglaskom na fleksibilnost i mogućnost održavanja.

MySQL je široko poznat sustav za upravljanje relacijskim bazama podataka otvorenog koda. Temelji se na svojoj pouzdanosti, učinkovitosti i jednostavnosti korištenja. Prikladan je izbor za aplikacije koje zahtijevaju strukturirano skladištenje podataka, velike mogućnosti postavljanja upita i učinkovito baratanje podacima.

2.2. Razlozi odabira tehničkog stoga

Kriteriji odabira tehnologije sastoje se od:

1. Mogućnosti prilagodbe i fleksibilnosti
2. Dizajna i stila
3. Performanse
4. Sigurnosti
5. Zajedničko razvojno iskustvo i ekosustav

2.2.1. Mogućnosti prilagodbe i fleksibilnosti

Laravel nudi široki spektar mogućnosti prilagodbe što olakšava prilagođavanje CMS-a specifičnim zahtjevima projekta. Za razliku od WordPressa, koji može zahtijevati značajne izmjene kako bi se postigle određene prilagođene funkcionalnosti, Laravel pruža precizniju kontrolu nad arhitekturom i ponašanjem aplikacije.

TwillCMS omogućuje fleksibilno i intuitivno administratorsko sučelje, koje se može lako prilagoditi i proširiti kako bi odgovaralo jedinstvenim potrebama projekta, nudeći više kontrole u usporedbi s WordPress-ovim unaprijed definiranim administratorskim pločama. Zbog mogućnosti proširenja i potpunoj kontroli, fleksibilnije je rješenje u usporedbi s WordPress-om.

2.2.2. Dizajn i stil

Koristeći TailwindCSS omogućuno je kreiranje prilagođenih dizajna koristeći njegove klase, pružajući veću kontrolu nad kreiranjem dizajna u usporedbi s tradicionalnim WordPress temama, koje mogu biti ograničavajuće i pretrpane.

Laravel-ov Blade sustav za izradu predložaka nudi moćan i praktičan način za stvaranje dinamičnih i višekratno upotrebljivih komponenti. Blade omogućuje integraciju složene logike i komponenti izravno unutar HTML-a, što olakšava pisanje čistog i održivog koda i što se potpuno paše s logikom TailwindCSS alata. Ovaj sustav za izradu predložaka pruža veću fleksibilnost i učinkovitost u usporedbi s tradicionalnim sustavom za izradu predložaka u WordPress-u.

2.2.3. Performanse

Laravel, u kombinaciji s učinkovitim rukovanjem bazom podataka pomoću MySQL-a, može rezultirati visokoučinkovitim aplikacijama prilagođenim specifičnim slučajevima upotrebe, potencijalno izbjegavajući dodatno opterećenje performansama koje može doći s WordPress-ovim značajkama i dodacima opće namjene. Naravno potrebno je držati se tehnika za optimiziranje performansa kao što je predaja poslova u red (engl. queue), podizanja PHP verzije na najnoviju, smanjivanje korištenja vanjskih paketa, određivanje željenog tereta (izbjegavanje N+1 upita) i mnoge druge.

2.2.4. Sigurnost

Laravel uključuje već ugrađene sigurnosne značajke kao što su zaštita od ubacivanja SQL-a, skriptiranja na različitim mjestima (XSS) i krivotvorenja zahtjeva na različitim mjestima (CSRF). Ovo može ponuditi višu razinu sigurnosti u usporedbi s WordPressom, koji može biti ranjiv ako se njime ne upravlja ispravno i ako se ne ažurira zbog široke upotrebe i velikog ekosustava dodataka.

2.2.5. Zajedničko razvojno iskustvo i ekosustav

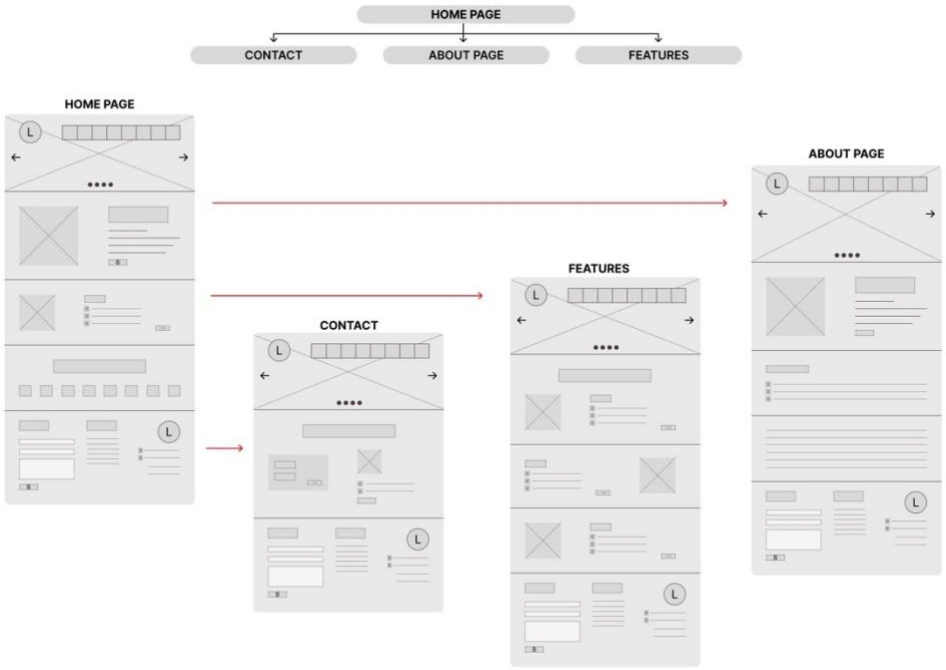
Zbog svog moćnog ekosustava i velike i jako dobro objašnjene dokumentacije, široke aktivne zajednice i velikog broja paketa i alata (kao što su Laravel Livewire, Jetstream i Sail), Laravel poboljšava produktivnost i iskustvo programera.

3. DIZAJN STRANICE

Dizajn stranice kreiran je u web aplikaciji Figma. Dizajn se sastoji od kostura stranice (engl. Wireframe), iscrtanih komponenti koje se koriste na više mjesta (gumb, izbornici, naslovi…), određene teme (boje, tipografije i zadanih razmaka) i svih stranica i njihovih izgleda na kojima se nalazi cjelokupni sadržaj.

3.1. Određivanje web ruta

Pristup dizajnu započinje od kostura čiji se primjer nalazi na slici Sl. 3.1 Kostur web stranice. U kosturu programer vidi dogovoreno ponašanje stranice (engl. user experience). Prema tome dobiva sliku o web rutama i samom pristupu postavljanja istih. Iz početka se može iščitati model Stranica, kojemu se za početak pridodaje atribut naziv i od tog naziva kreiran jedinstveni identifikator (engl. slug) za mogućnost kreiranja web ruta za svaki primjer modela Stranica (sa slike možemo vidjeti „home“, „contact“, „about“ i „features“).



Sl. 3.1 Kostur web stranice

3.2. Određivanje komponenti

Nakon upoznavanja s kosturom web stranice, idući korak je odabir komponenti. Komponente su elementi koji se koriste na puno različitih mjesta. Primjeri komponenti su samostalna slika, tekst, naslov, kartica s naslovom i tekstom, zaglavlje, podnožje, gumb, iskočni prozor, video i još mnogih drugih. Sve određene komponente potrebno je napraviti

samo jednom, da se izbjegne korištenje istog koda na više različitih mjesta. Osim prethodno označenih komponenti, u prolasku po svim stranicama i njihovim izgledima, pojavit će se još elemenata od kojih je potrebno kreirati nove komponente (primjer: Samostalna slika, kartica (slika s naslovom i kratkim tekstom), itd.). Trenutno određene komponente su: gumb, zaglavlje (engl. header), podnožje (engl. footer), naslovi i tekst.

3.3. Određivanje teme

Boja, razmaci i tipografija čine temu koja se koristi unutar cijelog projekta. U svakome projektu ti zahtjevi mogu biti drugačiji pa se iz tog razloga temu određuje na jednome mjestu kako bi se mogla jednostavno promijeniti. Promjena se treba omogućiti i kroz sučelje CMS sustava, a mjesto predviđeno za to najčešće su postavke (engl. settings).

3.4. Cjepkanje dizajna na dijelove

Završni korak obrade dobivenog dizajna stranice je cjepkanje dizajna na dijelove. Ono se sastoji od dodavanja dodatno potrebnih komponenti, određivanja najkompliciranijih točaka dizajna i određivanja modela.

3.4.1. Dodavanje dodatno potrebnih komponenti

Za početak je potrebno proći po dizajnu i uhvatiti sve preostale elemente koji se koriste na puno različitih mjesta, a nisu do sada ubrojene u komponente. Nova komponenta se može sastojati od više postojećih komponenti (npr. Kartica se sastoji od slike, naslova i teksta). Na taj način moguće je odrediti sve preostale komponente i nastaviti se držati prethodno spomenutih principa programiranja. Neke od komponenti koje se još dodaju su:

- Slika i tekst
- Paket
- Sobe
- Harmonika (engl. accordion)
- Traka ikona
- Sekcija (engl. section)
- WYSIWYG (engl. What you see is what you get)

3.4.2. Određivanje najkompliciranijih točaka dizajna

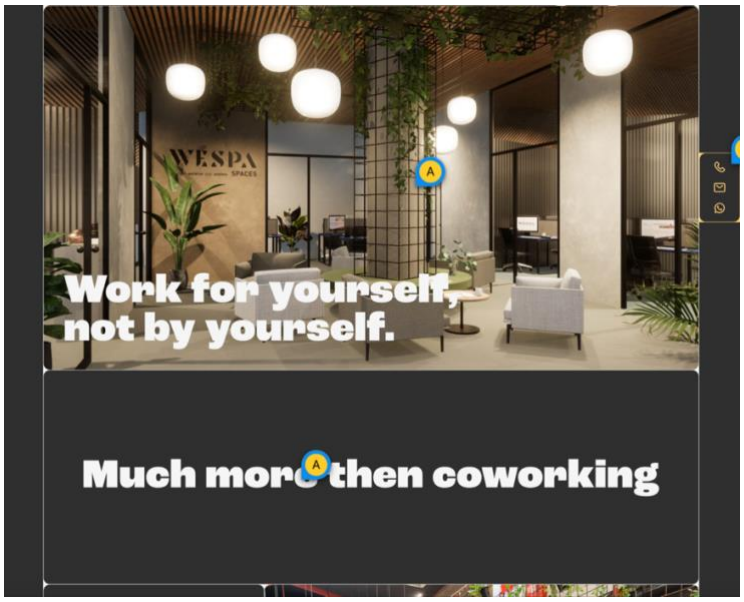
Cilj ovog međukoraka nije vezan za određivanje logike postavljanja za CMS, nego za moguću procjenu trajanja izrade web stranice. Programer na temelju svog iskustva i znanja može bolje organizirati svoje vrijeme i dati točniju vremensku procjenu klijentu ako unaprijed skalira elemente nacrtane u dizajnu. Najkompliciranija točka dizajna je netipično ručno kreiran okvir debljine 1px koji u svakom kutu ima zaobljenje s unutarnje strane što se može vidjeti na slici Sl. 3.2 Okvir. Na prvu postavljanje okvira nije komplicirano, no njegova

komplikacija javlja se kada više komponenti koje se dodiruju sadrže okvir. Potrebno je obratiti pažnju na sve moguće slučajeve dodirivanja svih komponenti što komplicira implementaciju okvira. Procjena za trajanje programiranja zadanog izgleda određena je na 3 tjedna.

Sl. 3.2 Okvir

3.4.3. Određivanje modela

Najbitniji međukorak postavljanja logike CMS sustava je određivanje modela. Prva jasno vidljiva stvar je da je dizajn kompletno određen nekakvim sekcijama (dijelovi okruženi okvirom). Svaku tu sekciju moguće je gledati zasebno kao što se može vidjeti na slici Sl. 3.3 Dvije sekcije u dizajnu. TwillCMS sustav u sebi sadrži model Blok (engl. block) kojeg možemo iskoristiti za sekcije dizajna. U primjeru prvi Blok bio bi blok klizač (engl. slider) koji se sastoji od više stavki klizača (engl. slider item). Svaka stavka klizača sastoji se od komponenti naslov i slika. Drugi blok bio bi blok citat (engl. quote) koji se sastoji od komponente tekst i više stavki riječi (engl. words item). Svaka stavka riječi sastoji se od komponente tekst. Ideja bloka citat je da se jedna riječ zamjenjuje svake 4 sekunde. „Much more then coworking“ za 4 sekunde postaje „Much more then community“.

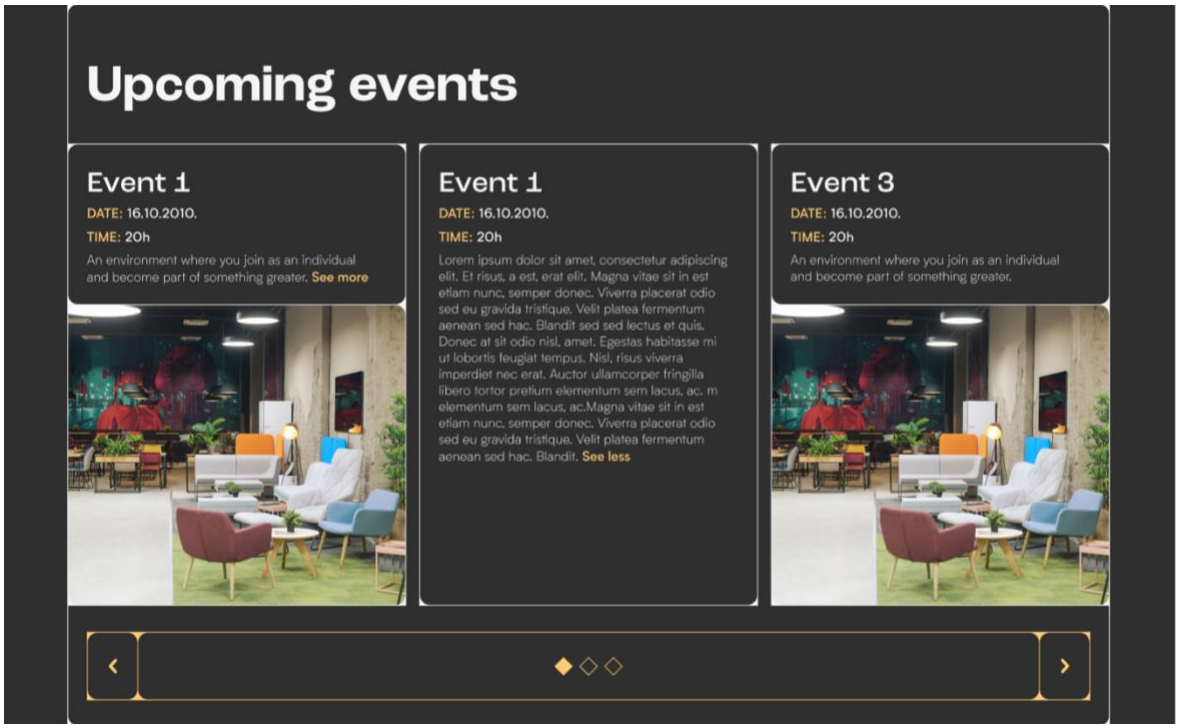


Sl. 3.3 Dvije sekcije u dizajnu

Nakon određivanja svih jednostavnih blokova, na red dolaze blokovi koji se pojavljuju na više različitih mjesta i u sebi sadrže veliku količinu informacija kao što se može vidjeti na slici Sl. 3.4 Ponavljajući blok s više informacija. Na temelju veće količine sadržaja i sekcije koja se nalazi na više različitih mjesta, potrebno je odrediti novi model koji će u ovome primjer biti model Događaj (engl. Event) i sastojat će se od slike, naslova, datuma i vremena početka i kraja, opisa i lokacije na kojoj se provodi. Po prolazu po zahtjevnijim blokovima u skup modela dodajemo slijedeće modele:

- Lokacije
- Događaji
- Navigacija
- Navigacija podnožja
- Članovi
- Sobe

Model Stranica sastojat će se od atributa naslov, jedinstvenog identifikatora i relacije na model Blokova kako bi se moglo na svaku stranicu pridodati sve blokove koje stranica po dizajnu treba imati.



Sl. 3.4 Ponavljajući blok s više informacija

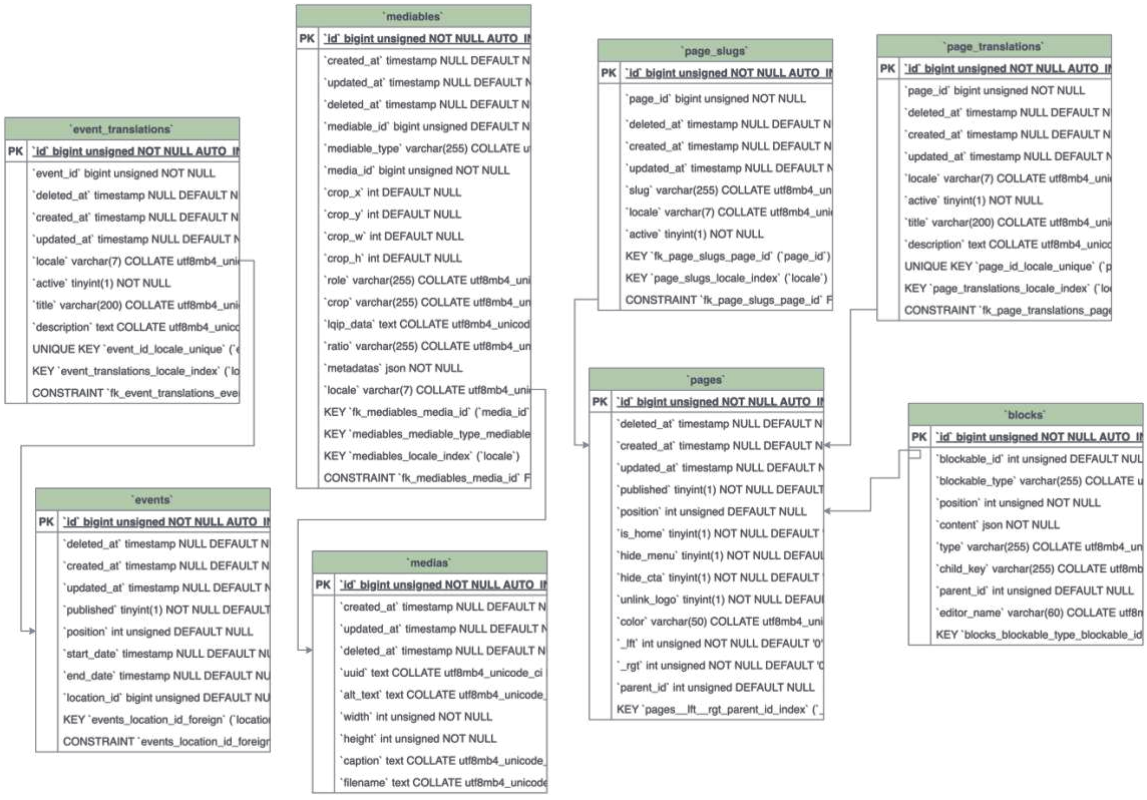
Nakon određivanja svih komponenti i modela, zaključena je priprema za programiranje sustava, njegovih modela i komponenti.

4. STRUKTURA I IMPLEMENTACIJA APLIKACIJE

U ovome poglavlju pojašnjene su strukture podataka, odnosi između postavljenih modela, način implementacije CMS sustava, način povezivanja Frontend dijela aplikacije s podacima dobivenim iz Backend dijela aplikacije, MVC struktura aplikacije i glavni modeli aplikacije: Stranica i Block.

4.1. Dijagram klasa i pregled sustava

Pregledom dijagrama klasa aplikacije vidljivog na slici Sl. 4.1 Dijagram klasa, prikazane su klase korištene u CMS sustavu. Na slici nisu uključene sve klase, nego one na temelju kojih je potrebno objasniti strukturu spremljenih klasa. Za jasnije objašnjenje promatranje će se bazirati na modelu Događaji određenom prema slici Sl. 3.4 Ponavljajući blok s više informacija. U lijevom kutu slike Sl. 4.1 Dijagram klasa predstavljene su dvije klase, klasa „events“ koja u odnosu s klasom „event_translations“ čini model Događaji. Klasa „event_translations“ služi kao pomoćna tablica za lakše omogućavanje višejezičnosti u CMS sustav pa se iz tog razloga u tu tablicu postavljaju svi atributi koje će biti potrebno prevesti na više jezika: title (hrv. naslov) i description (hrv. opis). Atribut locale određuje o kojem se jeziku prijevoda radi, a atribut active određuje je li taj prijevod aktivan ili nije. Na istome principu svaki Model koji ima atribute s potrebnim prevođenjem podijeljen je na glavnu klasu i na klasu prijevoda. Model Stranica ima dodatnu klasu „page_slugs“ u kojoj se na temelju određenog glavnog ključa (u ovome slučaju to je naslov), postavlja jedinstveni identifikator.



Sl. 4.1 Dijagram klasa

Model Stranica u odnosu je s modelom Blok na način da se na svaku dodanu stranicu može pridodati neki blok. Blok je s ostalim povezan klasom „Related“, na taj način je moguće u blokove uključivati druge modele i dobiti blok prikaza na slici Sl. 3.4 Ponavljajući blok s više informacija, odnosno prikazati 3 različite vrijednosti modela Događaj. Klasa „medias“ služi za dodavanje medija u medijsku biblioteku (engl. Media Library). Odnos s klasom „mediables“ određuje gdje će određeni medij biti korišten. Naprimjer slika se pridoda u blok kartica, na taj način dodat će se nova vrijednost klase „mediables“ u kojoj će biti zapisano da se novo kreirana vrijednost klase „medias“ koristi u novo kreiranom bloku kartica. U trenutku spremanja bloka kartica, u bazu podataka sprema se nova vrijednost klase „medias“, „mediables“ i novi model Blok.

4.2. Implementacija modela i glavnih modela Stranica i Blok

TwillCMS nudi komandu za kreiranje modela:

```
php artisan twill:make:module moduleName
```

Nakon što se komanda pokrene, nude se dodatne opcije:

- hasBlocks – dodavanje relacija na Blok model
- hasTranslation – dodavanje dodatne tablice „moduleName_translations“
- hasSlug – dodavanje dodatne tablice „moduleName_slugs“ u koju se zapisuju jedinstveni identifikatori
- hasMedias – relacija na tablicu „mediables“
- hasFiles – relacija na „files“
- hasPosition – dodavanje atributa „position“ koji služi za određivanje redoslijeda vrijednosti modela
- hasRevisions – dodavanje dodatne tablice „moduleName_revisions“ u koju se zapisuju stanja i promjene modela
- hasNesting – dodavanje gniježđenja (engl. nesting) za mogućnost odnosa roditelj – djeca

U slučaju modela Stranica pridodano mu je:

- hasBlocks
- hasTranslation
- hasSlug
- hasMedias
- hasRevisions
- hasPosition
- hasNesting

Nakon što se odaberu sve opcije, stvara se file za migraciju, model, repozitorij, kontroler i forma za kreiranje.

U migraciju i model se dodaju svi potrebni atributi. Repozitorij modela središnje je mjesto za upravljanje podatkovnim operacijama koje se odnose na specifične modele. Sažima logiku za dohvaćanje, pohranjivanje i manipuliranje podacima, pružajući čist i organiziran način interakcije s bazom podataka. Kontroler za model unutar CMS sustava definira glavna svojstva za kreiranje modela i tamo se mijenja tablični prikaz vrijednosti modela kojem ćemo pristupati nakon što dodamo rutu. U formu za kreiranje potrebno je dodati sva polja za ispunu povezana na dodane atribute.

Potrebno je dodati rutu za pristupanje novokreiranom modelu u `routes/twill.php`

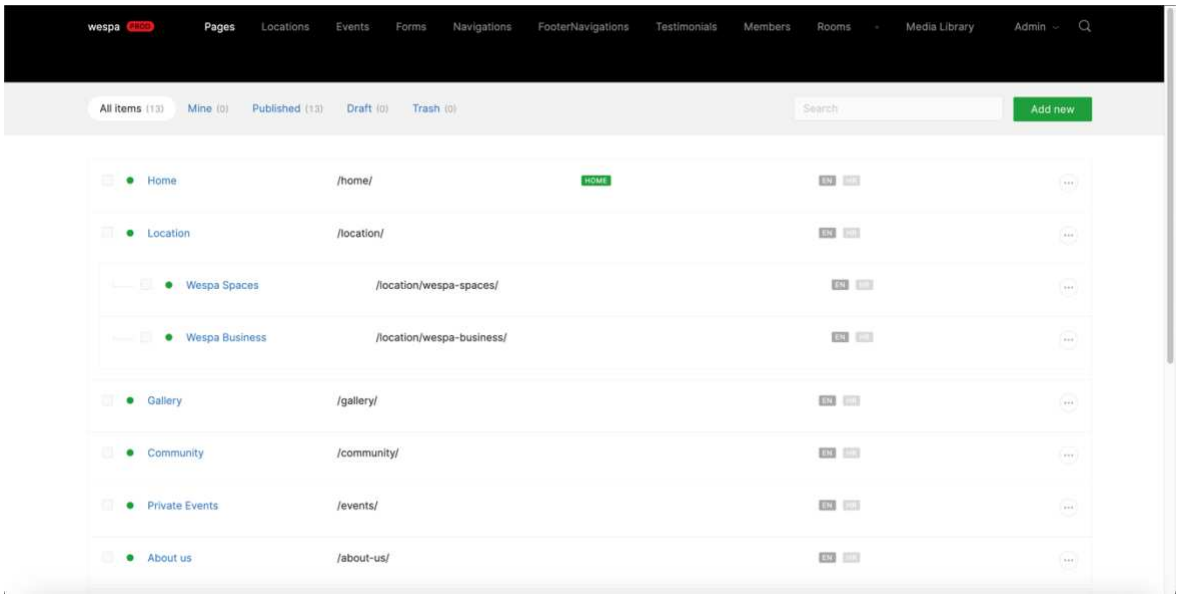
Za slučaj modela Stranica ruta bi glasila: `TwillRoutes::module('pages');`

Stranici modela Stranica u adminu se sada može pristupiti putem linka `/admin/pages`.

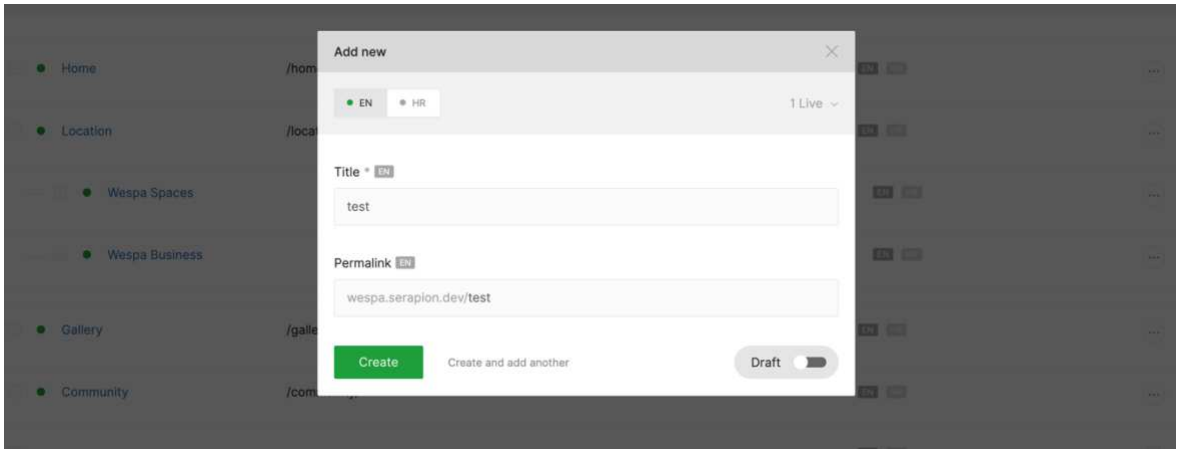
Pristupanjem linku, dolazimo do tabličnog prikaza svih vrijednosti zapisanih u bazu podataka kao što je vidljivo na slici Sl. 4.2 Prikaz modela Stranica u CMS sustavu. Moguće je dodati novu, urediti i obrisati staru (engl. CRUD operations). Kreiranjem nove stranice (klikom na gumb „Add new“) otvara se iskočni prozor (engl. Pop up), vidljivo na slici Sl. 4.3 Iskočni prozor za kreiranje nove stranice, za upis naziva i polje za određivanje jedinstvenog identifikatora, koji se automatski generira ovisno o upisanom nazivu. Klikom na gumb „Create“, kreira se nova stranica i ulazi se na njezinu stranicu uređivanja (slika Sl. 4.4 Stranica uređivanja sadržaja stranice). Moguće opcija dodavanja su:

- odabir glavne boje
- dodavanje blokova
- Use as Home
- Hide Menu
- SEO polja
- Status

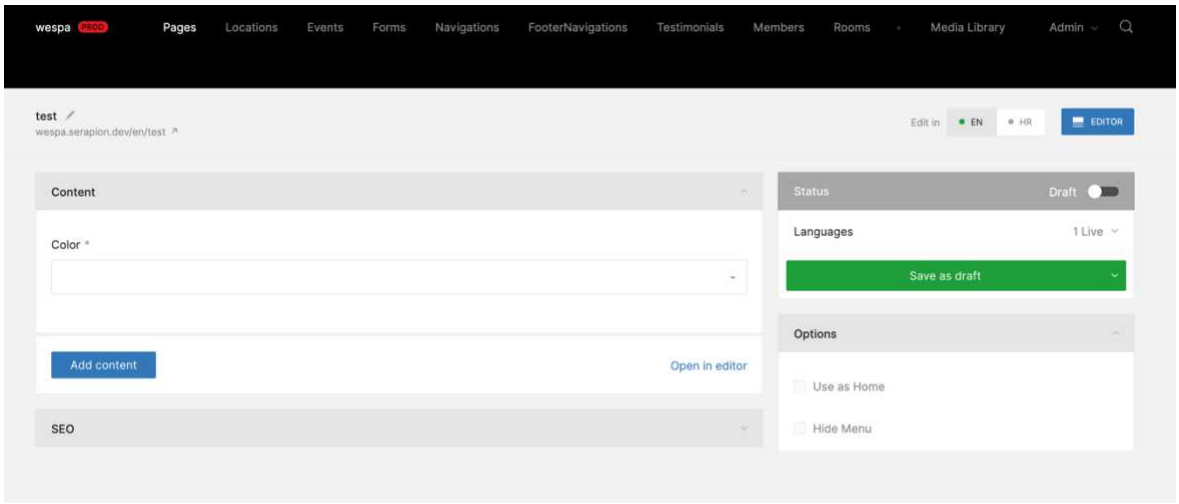
Odabir glavne boje određuje koja boja će prevladavati na stranici (ponuđene narančasta i crvena), dok se dodavanje novih blokova vrši po potrebi. Use as Home opcija određuje da je novo kreirana stranica početna stranica aplikacije. Hide Menu opcija koristi se na stranicama tipa „U izradi“ kada upravitelj stranice još nije zadovoljan sa stranicom, i ne želi da je na njoj vidljiv kreirani izbornik unutar zaglavlja. SEO polja služe za samostalno određivanje SEO parametara i samim time boljom pozicijom na pretraživačima. Status može biti označen kao skica ili uživo. Status skica znači da stranica nije vidljiva na internetu, dok status uživo znači da jest.



Sl. 4.2 Prikaz modela Stranica u CMS sustavu



Sl. 4.3 Iskočni prozor za kreiranje nove stranice



Sl. 4.4 Stranica uređivanja sadržaja stranice

4.2.1. Model Stranica

Model Stranica sastoji se od prevodivih atributa title, active, tablice „Blocks“ koja je uključena osobinom (engl. trait) HasAreaBlocks i tablice slug koja je uključena osobinom HasLocaleSlug koja na temelju određenog atributa u polju „translatedAttributes“ (atribut title) određuje jedinstveni identifikator za kreiranja rute za svaku stranicu (primjer: Stranica koja ima naslov Kontakt, poprima jedinstveni identifikator 'kontakt' i pomoću identifikatora i web rute moguće je pristupiti na linku `/kontakt`.

Web ruta:

```
Route::multilingual('{slug}/', RouteController::class)->where('slug',
'.*')->name('page'); php artisan twill:make:module moduleName
```

U RouteController-u na temelju atributa slug dohvaća se model Stranica koji u sebi sadrži vrijednost atributa slug dobivenog na ruti. Na taj način dohvaćen je model i njegov sadržaj može se u RouteController-u pustiti na frontend. Ako je stranica sa slugom pronađena, poziva se funkcija show u PageController-u

```
return $this->pageController->show($page);
```

u kojemu se svi podaci, zajedno sa skupom blokova šalju na frontend. Ako ne postoji stranica sa vrijednosti slug, RouteController baca error 404, stranica nije pronađena.

Stranice kreirane u ovome projektu su: Home, Locations – Wespa Spaces i Wespa Business, Gallery, Community, Private Events, About us i stranica Contact, koja je izvedena izvan CMS zbog svoje mogućnosti slanja maila.

4.2.2. Model Blok

Kao što je spomenuto u 3.4.3, cijeli dobiveni dizajn podijeljen je na blokove. Iz tog razloga potrebno je napraviti puno blokova. Za primjer odabran je jednostavan blok „naslov“. Blokovi se nalaze na putanji `resources/views/admin/blocks`.

Dokument bloka „naslov“ kreiran je koristeći Laravel Blade tehnologiju i on se nalazi na putanji `resources/views/admin/blocks/title.blade.php`.

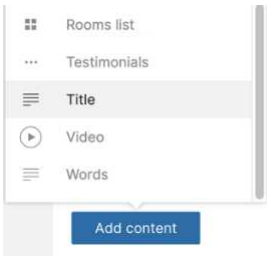
Isječak koda 4.1. Primjer kreiranja bloka „naslov“ u kodu

```
@twillBlockTitle('Title')
@twillBlockIcon('text')
@twillBlockGroup('content')

@include('admin.partials.hide-block')

@formField('input', [
'name' => 'title',
'type' => 'textarea',
'label' => 'Title text',
```

```
'maxlength' => 250,
'rows' => 4,
'required' => true,
])
```

„twillBlockTitle“ određuje naslov prikazan na vrhu slike Sl. 4.5 Primjer dodavanja kreiranog bloka „naslov“ u CMS sustavu. „twillBlockIcon“ određuje ikonu prikazanu na slici Sl. 4.6 Primjer odabira bloka koji se dodaje u CMS sustav. „twillBlockGroup“ određuje grupaciju koju je moguće pridodati određenom Modelu što služi za jednostavnost korištenja sučelja. Model Stranica može imati potrebu za dodavanje svih vrsta blokova, a modelu Novost dovoljan je blok naslov i tekst. U ovome slučaju, za model Stranica određuje se grupacija 'content', za model Novost nova grupacija 'news'. Na taj način na modelu Novost vidljiva su samo dva bloka, dok je na modelu Stranica vidljivo sve ostalo.

Sl. 4.5 Primjer dodavanja kreiranog bloka „naslov“ u CMS sustavu



Sl. 4.6 Primjer odabira bloka koji se dodaje u CMS sustav

Nakon popunjavanja sadržaja u bloku i spremanja sadržaja, novo kreirani blok pridodaje se skupu blokova stranice na kojoj je blok dodan. Vrste kreiranih blokova su: accordion, amenities, button, events, gallery, hero, hero-repeater, hero-vertical-repeater, hero-video, media-text, media-text-repeater, members, numbers, packages, price-plan, quote, rooms, testimonials, title, video i words. Accordion, events, members, rooms i testimonials blokovi koriste odnos između modela. Accordion blok koristi poveznicu na model Sobe, events blok koristi poveznicu na model Događaj, members koristi poveznicu na model Član, rooms koristi poveznicu na model Sobe i testimonials blok koristi poveznicu na model Iskustva. Njihovo kreiranje dodaje id modela i id bloka u tablicu „Related“ pa je na taj način moguće u bloku na stranici prikazati više vrijednosti jednog modela.

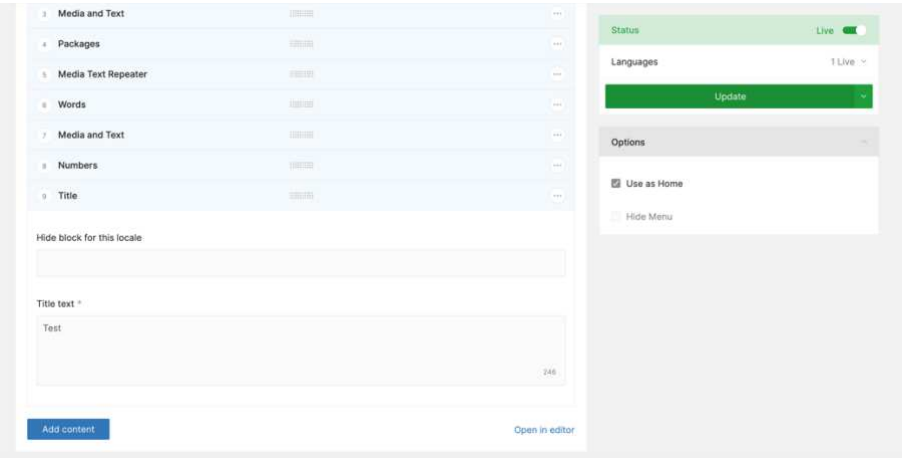
4.3. Povezivanje Backend dijela aplikacije s Frontend-om

Nakon što se u CMS dijelu aplikacije dodao novi blok unutar admina, potrebno je za taj blok napraviti i njegov frontend blok koji je povezan s blokom unutar admina istim nazivom. Frontend blokovi nalaze se na putanji `resources/views/blocks` što znači da bi za slučaj bloka naslov, ta putanja bi bila `resources/views/blocks/title.blade.php`. U datoteci title.blade.php sadržaj iz bloka dohvaćamo koristeći definiranu varijablu „block“ koja u sebi sadrži sva polja dodana u admin bloku. Popunjenom polju „title“ pristupamo streličnom (engl. arrow) funkcijom `$block->input('title')`.

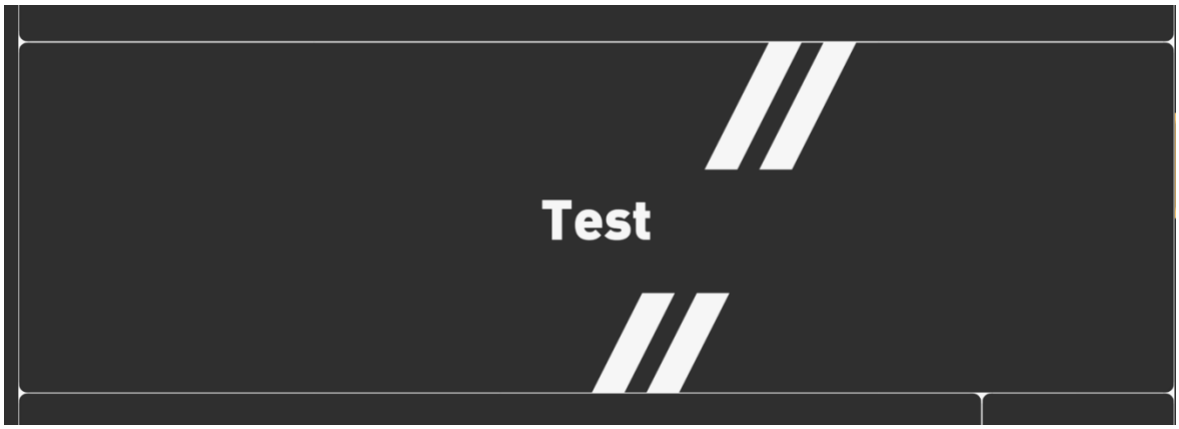
Ispunjavanjem i spremanjem bloka klikom na gumb „Update“ u CMS sustavu situaciju je moguće vidjeti na slici Sl. 4.7 Dodani i spremljeni blok „naslov“ u CMS sustavu. Taj blok je spremljen u skup blokova koji se nalaze na istoj stranici. Kada se pristupi određenoj stranici putem linka `/{slug}`, PageController poslat će sve potrebne informacije (Model Stranicu) u frontend datoteku page.blade.php. Frontend datoteka page jednostavna je i kratka:

```
@extends('layouts.site', ['isHome' => $page->is_home ?? false])

@section('content')
    {!! $page->renderBlocks(false, [] , ['pageColor' => $page-
>color, 'isHome' => $page->is_home]) !!}
@endsection
```

Prvi linija određuje da frontend datoteka page proširuje glavni izgled (engl. Layout) s sekcijom „content“. Unutar sekcije „content“ pristupom modelu Stranice predanog putem kontrolera u varijablu `$page`, funkcijom `renderBlocks` dohvaćaju se svi blokovi kreirani na stranici kojoj se pristupilo. Na taj način Novo dodani blok „naslov“ bit će vidljiv na stranici kao što se može vidjeti na slici Sl. 4.8 Dodani blok „naslov“ vidljiv na stranici.



Sl. 4.7 Dodani i spremljeni blok „naslov“ u CMS sustavu



Sl. 4.8 Dodani blok „naslov“ vidljiv na stranici

Tim procesom objašnjeno je osmišljeno ponašanje CMS sustava i logika njegovog korištenja. Proces komunikacije CMS sustava i Frontend dijela aplikacije objašnjen je na najjednostavnijem primjeru radi jednostavnijeg shvaćanja, ali isti proces vrijedi i za one kompliciranije.

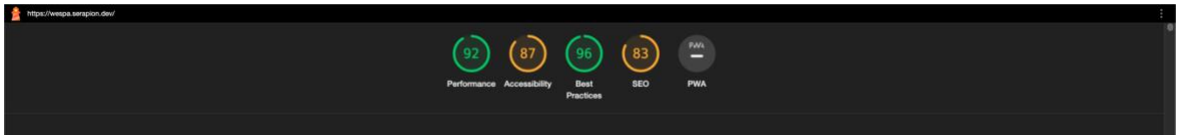
Isječak koda 4.2. Body dio Glavnog izgleda site.blade.php

```
            <x-icon-bar/>
            @if($svgSprite ?? false)
                <div class="absolute w-0 h-0 invisible">
                    <svg data-url="{{ asset('assets/svg/sprite.svg')}}"/>
                </div>
            @endif
            <x-header :page-title="$pageTitle"
                      :hide-menu="$hideMenu ?? false"
                      :is-error-page="$isErrorPage ?? false"
                      :language-slugs="$language_slugs ?? null">
            </x-header>

            <main>
                <div class="px-4 md:px-8 lg:px-20 content transition-all
duration-500 ease-in-out">
                    @yield('content')
                </div>

            </main>

            <x-footer></x-footer>
```

5. ZAKLJUČAK

U okviru ovog rada ostvaren je CMS sustav. Jednostavnost kreiranja projekta, jednostavnost korištenja CMS sustava, određenost koda (bez dodatnih viškova) neke su od prednosti kreiranja vlastitog sustava. Određenost koda dobivena korištenjem principa YAGNI znatno je smanjila težinu aplikacije. Principom DRY dijelovi blokova su podijeljeni u komponente i na taj način se smanjila količina koda za otprilike 30 %, a isto tako je i povećana jednostavnost promijene komponenti jer je se komponenta naslov mijenja samo na jednome mjestu. Za razliku od korištenja gotovih platformi, poznatiji su svi dijelovi aplikacije i samo poznavanje srži aplikacije dovodi do boljeg ostvarenja projekta.

Rezultati Google testa Lighthouse su prikazani na slici Sl. 5.1 Google Lighthouse test.



Sl. 5.1 Google Lighthouse test

Za dostizanje ocjene 100 učinkovitosti bilo bi potrebno koristiti modernije formate slika, poraditi na upravljanju predmemorije, postaviti projekt na brži/bolji hosting i poraditi na dohvaćanju vanjskih biblioteka (npr. ne sve odjednom nego po potrebi). Za približavanje ocjeni 100 pristupačnosti potrebno je nadodati nazive na linkove koji objašnjavaju kamo vode. Za približavanje ocjene 100 SEO dijela potrebno je postaviti projekt na vlastitu u CMS sustavu popuniti SEO polja stranice.

Što se CMS sustava tiče, nije odrađena podjela poslova po ulogama korisnika, odnosno nije otvorena rola korisnik koja može samo mijenjati sadržaj, ali ne i njime upravljati. Osim toga bilo bi potrebno u CMS sustavu postaviti Model Postavke u kojemu bi se mogle promijeniti sve glavne postavke kako bi CMS sustav bio što prilagodljiviji (npr. dodati odabir fonta ili glavnih boja, uređivanja stranica s greškama i slično).

LITERATURA

1. Taylor Otwell, Laravel dokumentacija, https://laravel.com/docs/10.x, 14. 2. 2023.
2. Quentin Renard, Twill dokumentacija, https://twillcms.com/docs/2.x/, 6. 3. 2020.
3. Adam Wathan, Tailwindcss dokumentacija, https://tailwindcss.com/docs, 9. 12. 2021.
4. Repozitorij predmeta *Programsko inženjerstvo*, https://fer.unizg.hr/predmet/proinz

SAŽETAK

Naslov: Ostvariti sustav za upravljanjem Web sadržajem

Sažetak

Kompleksnost upoznavanja sustava, nemogućnost rješavanja postojećih problema i korištenje vanjskih dodataka temeljni su problemi gotovih CMS sučelja. Cilj ovoga rada bio je realizirati sustav koji će biti jednostavniji za programera i otvoreniji za upravitelja sustava. Sustav je odrađen u PHP okviru Laravel te je nakon toga provedeno testiranje kvalitete ponašanja sustava. Implementacija je uspješna jer je korisnicima omogućeno lagano upravljanje sadržajem, programski kod je čitljiv, a po rezultatima testova sustav je postigao visoke ocjene u svojoj učinkovitosti.

Ključne riječi: PHP, Laravel, CMS sustav, razvoj weba

SUMMARY

Title: Implement custom content management system

Summary

The complexity of getting to know the system, the inability to solve existing problems and the use of external plugins are the fundamental problems of ready-made CMS interfaces. The goal of this work was to realize a system that will be simpler for the programmer and more open for the system manager. The system was developed in the Laravel PHP framework, after which quality testing of the system's behavior was carried out. The implementation is successful because users are enabled to easily manage the content, the program code is readable, and according to the test results, the system achieved high marks in its efficiency.

Keywords: PHP, Laravel, CMS system, web development,

PRIVITAK A: UPUTE ZA UPRAVLJANJE CMS SUSTAVOM

Za korištenje kreiranog CMS sustava u privitku dostavljam upute u obliku pdf datoteke upute_za_upravljanje_cms_sustavom.pdf koja je priložena uz rad. Za pristupanje aplikaciji obavezno otvoriti i proučiti privitak A.