Analiza i optimizacija web mjesta za poboljšanje korisničkog iskustva

Lovrić, Anton

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Anton Lovrić

Analiza i optimizacija web mjesta za poboljšanje korisničkog iskustva

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Anton Lovrić

Matični broj: 0016135356

Studij: Informacijski sustavi

Analiza i optimizacija web mjesta za poboljšanje korisničkog iskustva

ZAVRŠNI RAD

Mentor:

Matija Kaniški, mag. inf.

Varaždin, rujan 2022.

Anton Lovrić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom završnom radu bit će opisani procesi te principi, smjernice i metode dizajna korisničkog sučelja, od grube skice do završnog izgleda web mjesta. Stranice web mjesta trebaju biti responzivne, odnosno dizajnirane na način da se mogu pravilno prikazati na različitim dimenzijama uređaja.

Radi što boljeg dizajna analizirat će se prednosti i nedostaci najpopularnijih web mjesta. Korištenjem alata za optimizaciju i unapređenje rada web stranice kao što su primjerice HTML i CSS validator, GTmetrix moguća je detaljna analiza performansi web mjesta i njihova optimizacija. Osim toga potrebna je i SEO optimizacija kako bi se na web mjesto dovelo što više korisnika.

Naglasak ovog rada je na poboljšanju korisničkog iskustva pa će se u tu svrhu koristiti i alat Google Lighthouse, koji omogućuje evaluaciju korisničkih podataka radi poboljšanja korisničkog iskustva. Također bit će opisana uporaba kolačića, sesije i baze podataka kojim će se poboljšati iskustvo korisnika web stranice.

Nakon istraživanja, proučavanja korisničkih podataka i njihove analize izradit će se sama web aplikacija. U tu svrhu bit će korišteni okviri Laravel i Vue.js. Web aplikacija u navedenim okvirima izradit će se uz primjenu SEO smjernica.

Sadržaj

1. Uvod

Neovisno o tome ako koristimo društvene mreže, web trgovine ili različite web portale možemo izvući isti zaključak - internet je postao neizostavan dio naših života. U vrijeme pisanja ovog završnog rada na internetu trenutno postoji otprilike 1.18 milijardi web mjesta. Ako to nije dovoljno zapanjujuća brojka, trebamo uzeti u obzir da je svakodnevno kreirano 252 tisuće različitih web mjesta [1]. Internet se i dalje razvija vrtoglavom brzinom te je zbog toga potrebno biti u toku s brojnim promjenama.

Web trgovanje je također grana interneta koja ima veliku važnost u poslovanju svake trgovine. Baš kao što je potrebno dobro organizirati fizičku poslovnicu kako bi kupci lako mogli pronaći što im je potrebno, tako je potrebno i pružiti pozitivno korisničko iskustvo za vrijeme korištenja web stranice. Korisničko iskustvo je ključna stavka svake web stranice te može biti razlika između uspjeha i propasti poduzeća.

Korisničko iskustvo (engl. user experience) označava svaku interakciju korisnika s digitalnim proizvodom [2]. Web mjesto ostvaruje pozitivno korisničko iskustvo intuitivnim i oku ugodnim vizualnim dizajnom, brzim performansama i interaktivnošću svojih elemenata. Također jedan iznimno bitan dio korisničkog iskustva koji većina korisnika može zapostaviti je pristupačnost (engl. accessibility). Kako i navodi World Wide Web Consortium (W3C), web mora biti pristupačan korisnicima različitih poteškoća te pristup Internet resursima je osnovno ljudsko pravo [3]. Upravo zbog toga je potrebno pratiti W3C i WCAG (engl. Web Content Accessibility Guidelines) upute kako bi web mjesto bilo dostupno svim korisnicima koji ga žele koristiti.

Kako bi proizvod bio uspješan potrebno je izabrati prikladne alate. U nastavku će biti opisani odabrani alati i njihova uloga u izradi završnog rada. Nadalje će biti izvršena analiza postojećeg web mjesta i usporedba s konkurentnim web mjestima. Kao primjer u ovom završnom radu će biti korištena trgovina glazbom i glazbenom opremom „Free Bird Music Shop“. Nakon toga će biti prikazan proces dizajna, implementacije redizajniranog web mjesta te njegovo održavanje. Na kraju će biti potrebno izvući zaključak iz završnog rada te osvrnuti se na gotov proizvod.

2. Metode i tehnike rada

Pravilan odabir alata je ključni dio izrade svakog projekta. Zbog toga je dosta vremena otišlo u analizu svih dostupnih alata te odabir nekolicine. U nastavku ovog poglavlja će biti opisani odabrani alati, usporedba s konkurentnim alatima i razlog njihovog odabira.

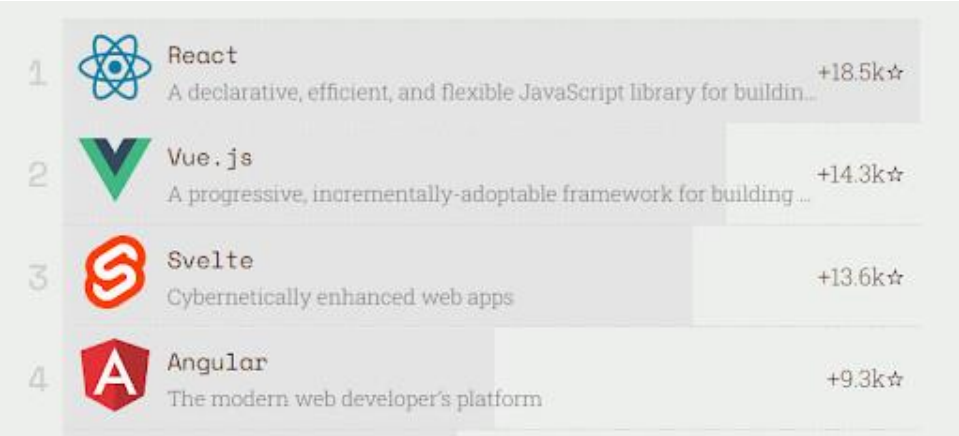
2.1. Karakteristike Vue.js okvira

Nakon nekoliko godina rada u Google-u koristeći Angular, JavaScript okvir (engl. Framework) za rad koji je uglavnom korišten za izradu velikih projekata, Evan You je skupio pregršt znanja o JavaScript okvirima za rad te njihovim prednostima i nedostacima. Unatoč svojoj snazi i velikoj količini značajki, Angular je imao nekoliko problema koji karakteriziraju njegov rad, među kojima su strogo određeni način rada koji programer mora pratiti te glomazan programski kod [4]. Evan You je vidio mjesto za napredak u navedenom okviru za rad i odlučio je uložiti dosta energije i truda u kreiranje svog okvira za rad koji je kasnije dobio naziv Vue.js.

Prva verzija okvira Vue je javnosti prikazana 27. lipnja 2013. godine. Nakon toga Vue se kontinuirano razvija i njegova popularnost je veća iz godine u godinu [5].

Slika 1. Prezentacija Vue-a (https://worldline.github.io/vuejs-training/presentation/#business-model-and-funding)

Za razliku od svojih glavnih konkurenata – okvira za rad Angular i biblioteke React.js, Vue održava njegova bogata zajednica koja volontira i na taj način doprinosi razvoju ovog okvira za rad. Zbog neumornog rada svojih volontera Vue je dostigao veliki uspjeh te se nalazi među najpopularnijim JavaScript okvirima za rad današnjice.



Slika 2. Rang lista najpopularnijih JavaScript okvira za rad (https://www.tecla.io/blog/top-js-frameworks)

Vue je dosegnuo navedenu razinu uspjeha iz različitih razloga, ali samo neki od tih razloga su njegova fleksibilnost, prilagodljivost, brzina i skalabilnost. Evan You je dao zanimljivu oznaku svom okviru za rad nakon što ga je nazvao „Progresivnim JavaScript okvirom za rad“ [6]. Riječ „progresivan“ jako dobro odgovara ovom okviru za rad iz razloga što Vue započinje s iznimno laganom osnovom i raste zajedno s projektom koji se gradi. Iz tog razloga se može reći da je Vue moguće koristiti za male i velike projekte, što ističe njegovu fleksibilnost. Sposobnost rada na projektima različitih veličina ga razlikuje od ostalih okvira za rad kao što su Angular i Svelte, pošto Angular ima veliku količinu alata koji dolaze u paketu dok je Svelte iznimno lagan okvir za rad ali ima svoja ograničenja koja ga sprječavaju u korištenju na većim projektima. Za razliku od ova dva okvira za rad Vue svojom sažetom veličinom od 20kb i mogućnosti brojnih nadogradnji postiže fleksibilnost na projektima [7].

Iznimno bitna stavka svakog digitalnog proizvoda je njegova performansa. Loše performanse mogu upropastiti korisničko iskustvo te zbog toga direktno utjecati na neuspjeh proizvoda. To je više nego dovoljan razlog za ulaganje velike količine resursa radi ostvarivanja dobrih performansi. Vue postiže iznimno visoke performanse zbog svoje veličine od 20kb koja je spomenuta ali i inkrementalnog načina uključivanja komponenti koje su potrebne za rad. Ovo omogućava programeru da uzima samo komponente koje su mu potrebne u radu i time postiže veću brzinu u radu [8].

Uz sve ovo početak rada u Vue-u nije kompliciran za korisnike koji su upoznati s jezicima HTML, CSS i JavaScript. Svojom intuitivnom sintaksom Vue omogućava brz početak

svim korisnicima. Svojim modularnim pristupom i odvajanja HTML, CSS i JavaScript dijelova programskog koda stvara poznato okruženje programerima koji su već radili u sličnim okvirima.

Vue radi na principu komponenti, odnosno svaki prikazani element predstavlja jednu komponentu. Komponente mogu varirati u svojoj složenosti ali njihova velika prednost je u mogućnosti enkapsulacije i odjeljivanju programskog koda. U sljedećem primjeru će biti prikazana iznimno jednostavna komponenta naziva „ElementListe“. Navedena komponenta će primati određeni tekst kao svojstvo i koristeći Vue petlje će biti prikazana na ekranu.

App.vue ElementListe.vue

```
<script setup>
const props = defineProps({
  element: Object
})
</script>

<template>
  <li>{{ element.text }}</li>
</template>
```

Slika 3. Komponenta ElementListe (vlastita izrada)

Unutar <script> oznake je potrebno napisati JavaScript kod koji definira svojstva koja će komponenta primiti. Nadalje, unutar <template> oznaka je definiran element koji će se prikazati i njegove osobine. Vidimo kako će biti prikazan tekst koji je proslijeđen u sklopu objekta element te će on biti umotan u <li> oznaku, koja označava element liste.

App.vue ElementListe.vue

```vue
<script setup>
import ElementListe from './ElementListe.vue'

const listaPrednosti = [
  { id: 0, text: 'Fleksibilnost' },
  { id: 1, text: 'Skalabilnost' },
  { id: 2, text: 'Pristupačnost' },
  { id: 3, text: 'Brzina' }
]
</script>

<template>
  <ol>
    <ElementListe
      v-for="prednost in listaPrednosti"
      :element="prednost"
      :key="prednost.id"
    ></ElementListe>
  </ol>
</template>
```

Slika 4. Prikaz korištenja komponente ElementListe (vlastita izrada)

Pregledom glavne komponente naziva „App.vue“ je moguće bolje razumjeti što se događa s prethodno opisanom komponentom. Unutar <script> oznake je potrebno uključiti prethodno napravljenu komponentu i kreirati polje koje će biti prikazano. Za ovaj primjer je napravljena lista prednosti koje Vue ima u usporedbi sa svojim konkurentima. Dalje u <template> sekciji se prikazuje <ol> oznaka koja označava sređenu listu te putem atributa v-for se iterira kroz polje listaPrednosti. Iteriranjem kroz navedeno polje se stvara sljedeći ispis.

PREVIEW JS CSS SSR

1. Fleksibilnost
2. Skalabilnost
3. Pristupačnost
4. Brzina

Slika 5. Ispis testne komponente (vlastita izrada)

Prethodni primjer pokazuje tek osnovni način rada ovog okvira za rad i njime smo tek zagrebli njegovu površinu. Kako bismo iskoristili punu moć Vue-a bit će korištena njegova nadogradnja – Nuxt 3.

2.1. Karakteristike okvira za rad Nuxt.js

Unatoč tome što Vue ima brojne pogodnosti u svijetu Web programiranja uvijek postoji težnja za daljnjim razvojem. Upravo iz tog razloga je nastao još jedan okvir za rad pod nazivom Nuxt.js. Nuxt je okvir za rad koji je izgrađen na temelju Vue okvira za rad uz iznimno važne značajke koje uvelike olakšavaju rad i poboljšavaju gotovi proizvod [9].

Nuxt 3 je najnovija inačica ovog okvira za rad koja predstavlja možda i najveći korak u svom razvoju. Za razliku od svog prethodnika, Nuxt 2, nova inačica dolazi u potpunosti opremljena s TypeScript podrškom. TypeScript je programski jezik koji proizlazi iz JavaScripta uz dodatne prednosti kao što je strogo određivanje tipova podataka. Strogo određivanje tipova podataka je iznimno bitno kako bi se pogreške otkrile i odstranile prilikom razvoja same aplikacije za razliku od otkrivanja pogrešaka tijekom rada [9].

Osim TypeScript podrške, Nuxt 3 je izgrađen na temelju novog Nitro stroja za renderiranje (engl. Rendering Engine), Vite alatom i podrškom za novi Composition API (engl. Application Programming Interface) koji dolazi zajedno s trećom inačicom okvira za rad Vue. Nitro je novi stroj za renderiranje koji je Nuxt tim izgradio iz temelja kako bi ga mogli koristiti u novoj verziji Nuxt-a. Ovaj stroj za renderiranje dopušta Nuxt-u da dosegne nove visine i postane jedan od najboljih okvira za rad današnjice. Zahvaljujući Nitro stroju za renderiranje Nuxt postiže iznimno visoke brzine rada i dodatno ojačava Nuxt poslužitelj (engl. Server) [9]. Osim performansi, Nuxt donosi različite prednosti i u samom razvoju. Uz TypeScript-ovo osiguravanje tipova podataka, Nuxt omogućava i revolucionarni sustav automatskog uključivanja vanjskih modula. Ovaj sustav znatno ubrzava razvoj aplikacije i pridonosi urednosti koda. Osim automatskog uključivanja modula i dalje su zadržane opcije dinamičkog i eksplicitnog uključivanja modula [10].

Ranije je navedeno kako Vue.js sebe naziva „progresivnim JavaScript okvirom za rad“ te je u potpunosti dostojan tog naziva. Za razliku od Vue-a, Nuxt.js je dobio naziv „hibridni JavaScript okvir za rad“. Riječ „hibridni“ govori da Nuxt.js ima dodatnu dimenziju za razliku od Vue-a, a to je izvršavanje na strani poslužitelja [9]. Ovo je iznimno bitan faktor pošto omogućava nove dimenzije razvoja web aplikacija te se približava statusu skupa programskih okvira (engl. Fullstack) za rad. Uz pomoć rada na poslužitelju moguće je dohvaćati podatke iz baze podataka i odmah ih prikazati korisniku što znatno poboljšava korisničko iskustvo. Osim toga, Domagoj Svjetličić iz zagrebačke firme Bornfight u svom članku vezanom za Nuxt.js ističe

prednosti renderiranja na strani poslužitelja, poboljšano upravljanje meta oznakama ali ponajviše SEO optimizaciju (engl. search engine optimization) koju Nuxt donosi [11].

Upravo prednosti vezane za SEO su veliki razlog zašto je Nuxt.js odličan izbor za izradu ovog projekta. Web dućan ne može ispuniti svoju misiju ako ju korisnici ne mogu pronaći te zbog toga je SEO iznimno bitna stavka prilikom razvoja ovakvih web aplikacija. Osim toga uz alate kao što su Vite i novi Nitro stroj za renderiranje aplikacija će postići znatno više brzine nego prije te korisniku omogućiti pozitivno korisničko iskustvo prilikom korištenja stranice.

2.2. Dizajniranje web mjesta pomoću alata Figma

Estetska privlačnost web mjesta je neophodna za njegovu uspješnost. S obzirom na to koliko web stranica postoji na internetu izrazito je bitno ostvariti pozitivan prvi dojam kod korisnika kako bismo dobili njegovo povjerenje i zadržali ga na web stranici.

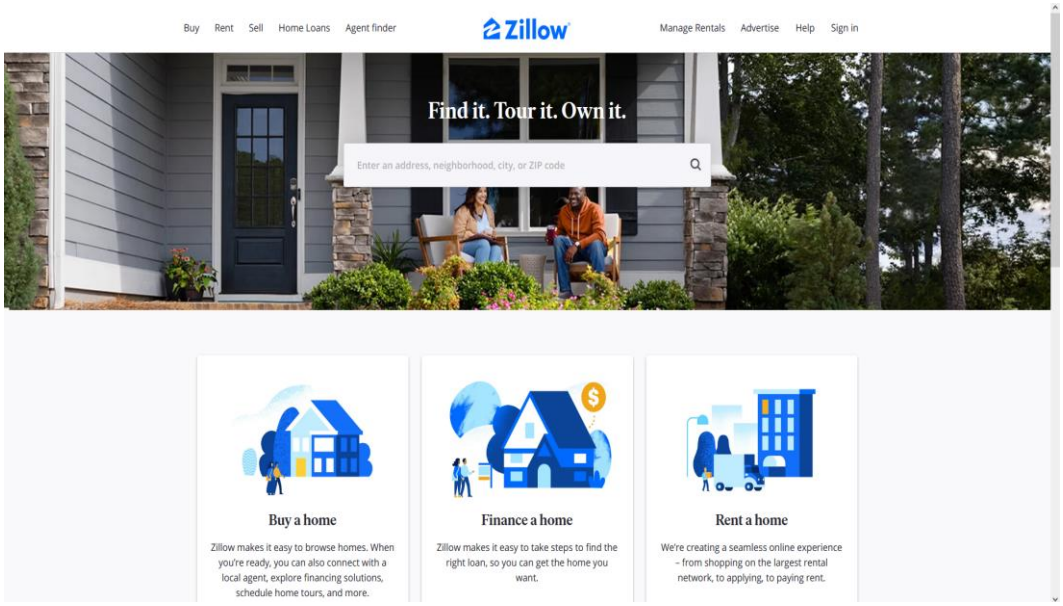
Web dizajn je proces planiranja te izrade izgleda web stranice. Ključni elementi web dizajna su struktura web stranica, raspored elemenata, uporaba slika i fontova. Web dizajn se sastoji od različitih manjih elemenata, među kojima su grafički dizajn, dizajn sučelja te SEO optimizacija [12]. Prilikom implementacije prethodno navedenih elemenata izrazito je bitno obratiti pažnju na dvije stvari: vizualni dojam web stranice i korisničko iskustvo prilikom korištenja web stranice. Oba područja su izrazito bitna te je ključno ispuniti njihove zahtjeve.

Važnost vizualnog dojma je razumljiva sama po sebi ali kreiranje estetski ugodne web stranice možda nije tako intuitivno. Srećom postoje određene smjernice koje mogu pozitivno utjecati na izgled web stranice.

Pravilan odabir i organizacija elemenata koji se nalaze na web stranicu su ključni za funkcionalnost i preglednost web stranice ali i poboljšavanje korisničkog iskustva. Prije kreiranja samog dizajna potrebno je analizirati web mjesto, njegovu svrhu i odrediti ciljanu publiku za koju se ono kreira. Nakon samog planiranja je moguće odrediti koji su elementi potrebni i na koji način ih je potrebno rasporediti kako bi maksimizirali svoju korisnost [13].

Korištenje boja je također izrazito bitna stavka dizajniranja web stranice. Boje ostavljaju snažan dojam i korištenjem određenih boja mogu se probuditi osjećaji sigurnosti, povjerenja ili uzbuđenja kod korisnika. Mnoge web stranice upravo iz tih razloga koriste boje u svom dizajnu [13]. Iako su boje izrazito moćan alat upravo zbog ranije navedenih razloga, jednako je bitno ne pretjerati s bojama pošto je jako lagano zbuniti korisnika te kreirati neurednu stranicu koja ne može jasno iskazati svoje glavne značajke. Radi toga je najbolje koristiti uz glavnu boju stranice i boju teksta koristiti samo jednu ili dvije boje za naglašavanje sadržaja [13]. Odabrane boje treba ujednačeno koristiti kroz cijelo web mjesto kako bi web mjesto bila jedna cjelina.

Zillow je organizacija koja se bavi nekretninama i posjeduje web stranicu koja je dobar primjer pravilnog korištenja prostora te ugodnog i korisnog web dizajna [13].



Slika 6. Primjer pravilno dizajnirane web stranice (https://www.zillow.com/)

Pozicioniranjem logotipa u sredinu zaglavlja stavlja se naglasak na brend te se omogućava lakša organizacija ostalih poveznica koje se nalaze uz logotip. Kratkim sloganom na glavnoj sekciji naslovne stranice se iskazuje jasna poruka koja opisuje misiju organizacije kojoj pripada ova web stranica. Fotografija u pozadini probija monotonost te dodaje boju i samim tim novu dimenziju ostatku dizajna koji je poprilično jednostavan.

Ispod glavnog dijela stranice vidimo usluge koje Zillow pruža svojim korisnicima zgodno prikazane u obliku kartica. Kartice omogućavaju lako organiziranje sadržaja i naglašavanje željenih poruka koju organizacija upućuje svojim korisnicima. Svaka kartica se sastoji od ilustracije, podnaslova, kratkog sažetka te gumba koji omogućava korištenje opisane usluge. Ilustracije koriste plavu boju brenda što pozitivno doprinosi standardizaciji dizajna i time stvara identitet koji organizacija koristi na svojoj web stranici. Ostali elementi kartice omogućavaju korisniku da brzo dođe do potrebnih informacija.

Nakon upoznavanja s pojmom web dizajna i glavnim smjernicama za izradu dobrog dizajna potrebno je analizirati dostupne alate i odabrati alat u kojem će biti izrađen dizajn za web stranicu korištenu u ovom završnom radu.

Tržište web dizajn softvera je bogato različitim, kvalitetnim alatima koji omogućavaju dizajniranje kvalitetnih web stranica. Tri alata koji su najbolji u svojoj konkurenciji su Adobe XD, Sketch i Figma. Sketch je vrhunski alat koji je izrazito popularan u području web dizajna ali glavni nedostatak mu je to što je dostupan jedino na Mac OS operacijskom sustavu. To je također razlog zašto nije korišten u ovom završnom radu. Nadalje Adobe XD je još jedan alat

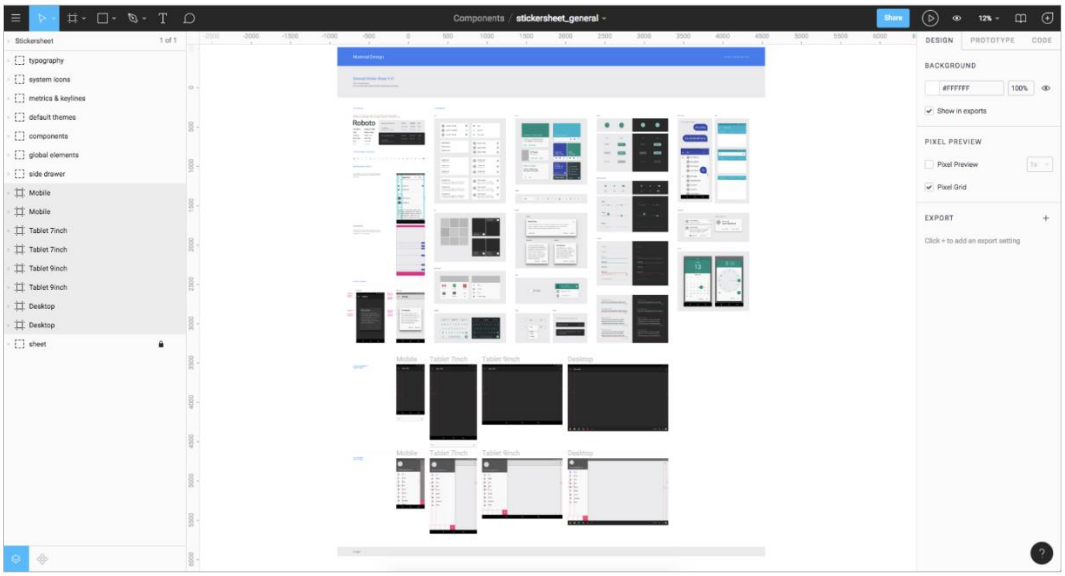
koji dopušta korisniku ugodan i produktivan rad vezan za dizajniranje web stranica ali posjeduje plaćenu licencu što predstavlja prepreku u njegovom korištenju. Svi ovi faktori su utjecali na odluku da Figma postane alat koji će biti korišten u ovom završnom radu [14].

	Figma	Sketch	Adobe XD
Platforma	Aplikacija temeljena na pregledniku	Aplikacija za radnu površinu i preglednik	Aplikacija za radnu površinu i mobilna aplikacija
Operacijski sustav	MacOS, Windows, Linux	MacOS	MacOS, Windows, iOS, Android
Suradnja	Suradnja u stvarnom vremenu	Suradnja u stvarnom vremenu za MacOS Sketch pretplatnike	Suradnja u stvarnom vremenu na projektima sinkroniziranim s oblakom
Početak rada	Lekcije i vježbe dizajna	Dokumentacija	Video upute i vodiči korak po korak
Dodaci	Sve veća biblioteka dostupna u aplikaciji	Velika biblioteka, preuzeta izvana	Sve veća biblioteka dostupna u aplikaciji
Vektorska manipulacija	Vektorske mreže	Vektorske staze	Vektorske staze
***Cijena**	Besplatna početna verzija ili 12 USD mjesečno po uredniku (SAD)	30-dnevno besplatno probno razdoblje, zatim 9 USD mjesečno po uredniku (SAD)	7-dnevno besplatno probno razdoblje, zatim 9,99 USD mjesečno (SAD)

Slika 7. Usporedba alata Figma, Sketch i Adobe XD (https://www.coursera.org/articles/figma-vs-sketch-vs-adobe-xd)

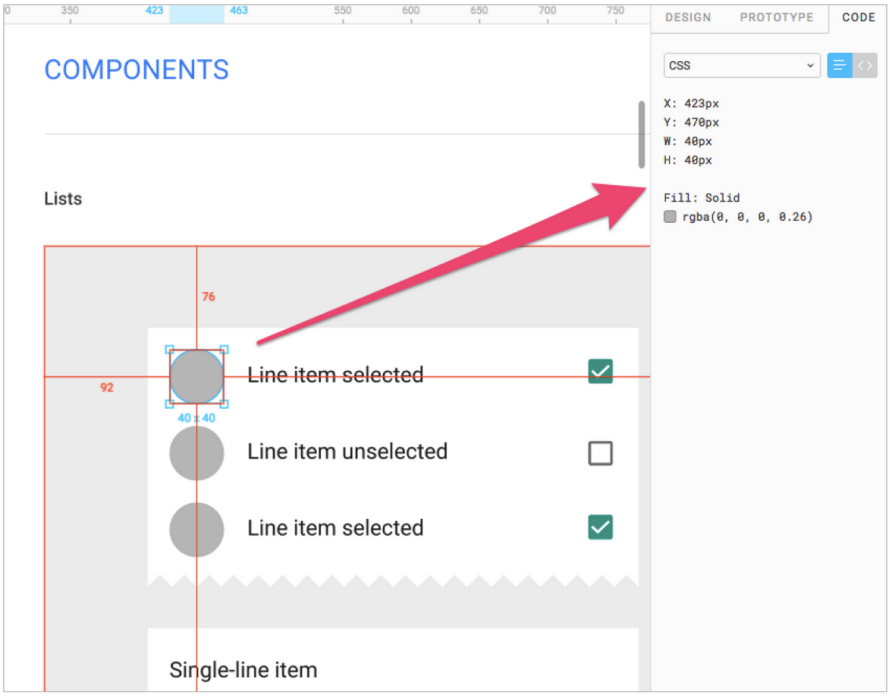
Figma je web design alat koji je temeljen na tehnologiji u oblaku (engl. Cloud) , ali postoji i besplatna stolna(engl. Desktop) inačica aplikacije koja se može koristiti na isti način. Arhitektura u oblaku omogućava korisnicima da koriste Figmu na bilo kojem uređaju. Ovo pogotovo dolazi do izražaja u timskom okruženju te ne zahtijeva od svakog korisnika identičnu radnu okolinu kako bi radili zajedno na projektu [15].

Figma je iznimno intuitivan alat i svojim jednostavnim dizajnom omogućava korisniku da izrazito brzo započne s izradom žičanih modela, dizajna te prototipa svoje web stranice. Sve glavne funkcije se nalaze u alatnoj traci na vrhu aplikacije, s lijeve strane je moguće pronaći sve elemente u trenutnom projektu te klikom na svaki element je moguće vidjeti njegove detalje.



Slika 8. Alat Figma (https://www.toptal.com/designers/ui/figma-design-tool)

Širok raspon mogućnosti omogućava Figmi da bude kompletan alat svakog web dizajnera. Kreiranje žičanih modela (engl. Wireframe) je jednostavno i brzo te omogućava dizajneru da brzo iznese svoje ideje na platno. Nakon toga je potrebno dodati određene detalje, boje i fotografije žičanim modelima i time kreirati potpuni dizajn web mjesta. Figma nije od koristi samo web dizajnerima nego i programerima. Osim tog što predstavlja izgled web mjesta, također i pomaže programeru da implementira dizajn time što pruža i CSS kod za uređivanje svakog elementa. Navedeni CSS isječci izrazito pomažu programeru da točno procijeni visinu i širinu elementa, debljinu rubova, točne specifikacije sjene elementa i sl.



Slika 9. CSS Isječci u alatu Figma (https://www.toptal.com/designers/ui/figma-design-tool)

Osim CSS isječaka koji prikazuju kako implementirati izgled pojedinog elementa, Figma omogućava i kreiranje interaktivnih prototipa koji prikazuju planirano ponašanje web stranice i pojedinih elemenata na njoj. Korištenje prototipa ne služi samo dizajneru da lakše iznese svoje ideje i programeru da ih lakše shvati, nego i klijentima kako bi mogli procijeniti ako su njihove želje ispunjene. Tu se ponovno vidi moć toga što je Figma alat izgrađen na tehnologiji u oblaku, pošto je moguće lako podijeliti prototip i dizajn drugoj osobi bez ikakvih poteškoća. Sve opisane karakteristike čine Figmu savršenim alatom za slučaj korištenja potrebnim u izradi ovog završnog rada.

2.3. Biblioteka za korisničko sučelje

Kreiranje dizajna i interaktivnog prototipa je samo jedan korak prema stvaranju ugodnog korisničkog sučelja. Idući korak je naravno implementacija prethodno napravljenog dizajna koristeći Nuxt.js okvir za rad. Pošto moderna web mjesta imaju velik broj različitih komponenti u svojim korisničkim sučeljima, kreiranje svih tih komponenti od nule bi predstavljao brojne poteškoće u općem razvoju aplikacije. Korištenje biblioteke za korisničko sučelje (engl. UI) može znatno ubrzati razvoj cijele web stranice pošto na početku omogućava isprobavanje pojedinih funkcionalnosti web mjesta i rješavanje novootkrivenih problema. Također gotove komponente omogućavaju da se više vremena posveti ostalim, bitnim funkcionalnostima web stranice koja imaju znatan utjecaj na korisničko iskustvo. Naposljetku, UI biblioteke pružaju komponente koje provjereno rade i pružaju čvrste temelje za rad. Ove komponente su testirane od strane više iskusnih programera u svrhu kreiranja što boljeg finalnog proizvoda. Iz navedenih razloga će UI biblioteka biti korištena u ovom završnom radu.

Uslijed rasta popularnosti okvira za rad Vue.js jednako tako se razvija i cijeli njegov ekosustav. Ekosustav okvira za rad predstavlja sve pakete, biblioteke i ostale dodatke koji omogućuju brži i bolji razvoj aplikacija u navedenom okviru za rad. Postoje brojne biblioteke UI komponenti koje imaju funkcionalne i vizualno ugodne komponente kao što su Vuetify, Naive UI, Vuestic UI i ostali. Odabir ispravne biblioteke za ovu svrhu je složena i bitna odluka te ću u nastavku predstaviti neke od tih biblioteka i objasniti konačni izbor.

2.3.1. Naive UI

Prva inačica UI biblioteke Naive UI je izdana u ožujku 2020. godine ali tek nakon godinu dana je dobila podršku mnogih Vue programera koji su ju koristili. Gledajući napravljene promjene (engl. Changelog) očito je kako sve više programera doprinosi ovom projektu te da se svakom novom inačicom ispravljaju prethodno napravljene pogreške i dodaju se nove značajke [16]. Čitajući kroz opis ove biblioteke u službenoj dokumentaciji lako je dobiti dojam o tome što Naive UI predstavlja. Naive UI je kompletna, brza ali i jednostavna biblioteka [17].

Osim toga što posjeduje čak 80 komponenti, većina komponenti izgledaju iznimno dobro te pružaju pozitivno iskustvo. Uz to Naive UI bilježi dobre performanse zbog mogućnosti „tree-shakeanja“ biblioteke te učitavanja točno onih komponenti koje su potrebne za rad. Na ovaj način se izbjegava nepotrebno usporavanje web mjesta [17].

Nažalost, unatoč svojim prednostima Naive UI ima svoje poteškoće zbog kojih ipak nije izabran za ovaj završni rad. Dokumentacija je poprilično komplicirana, nije jasno opisan proces korištenja biblioteke u Nuxt okruženju te zbog toga je instalacija poprilično nezgodna. Osim toga komponente nisu napravljene uzimajući u obzir pristupačnost, stoga korisnici s vizualnim poteškoćama neće moći koristiti komponente na pravilan način. Ovo predstavlja veliku ulogu u poboljšavanju općenitog korisničkog iskustva te je razlog zašto je odabrana druga UI biblioteka.

2.3.2. Vuetify

Pregledavajući statistiku na Github repozitoriju ove biblioteke lako je uočiti kako je Vuetify jedna od, ako ne i najpopularnija Vue UI biblioteka. Uz 721 osobu koja je doprinijela projektu, preko 35.000 pozitivnih ocjena te čak 200.000 korisnika, Vuetify je zaista gigant u Vue ekosustavu [18]. Vuetify se čak predstavlja kao poseban UI okvir za rad, a ne samo UI biblioteka što donosi brojne pogodnosti i govori puno o sposobnostima Vuetify-a [19].

Kao što je navedeno u dokumentaciji, Vuetify je strogo izrađen po Material Design specifikaciji koja je napravljena od strane Google-a 2014. godine te predstavlja zlatni standard u dizajnerskom svijetu. Sve komponente su izrađene uz „mobile first“ pristup koji prioritizira izgled i performanse na mobilnim ekranima [19]. Ovakav pristup je koristan u modernom dobu pošto sve veći broj korisnika koristi svoje mobilne uređaje kako bi posjećivali razna web mjesta.

Zbog velike količine stručnjaka koji doprinose ovom projektu te velikoj zajednici koja koristi Vuetify u osobnim projektima moguć je brz ciklus razvoja te čak tjedna ažuriranja [19]. Osim toga, velika zajednica omogućava pomoć svim novim korisnicima Vuetify-a u slučaju da naiđu na poteškoće dok koriste ovaj UI okvir.

Vuetify također pruža vodiče kroz sve komponente koje pruža kako bi novi korisnici mogli što prije započeti s radom. Dokumentacija je opsežna ali jasna te na raspolaganju je veliki broj komponenti koje pokrivaju razne slučajeve korištenja. Komponente su naravno vrhunske u vizualnom smislu, ali također pružaju izvrsnu potporu za korisnike s poteškoćama [20]. Osim toga što ispunjavaju WCAG standarde za pristupačnost, Vuetify također pruža potporu za lako korištenje više jezika u svojim komponentama [21].

Slika 10. Usporedba Vuetify-a s ostalim UI okvirima za rad (https://vuetifyjs.com/en/introduction/why-vuetify/)

Unatoč brojnim prednostima Vuetify-a, ovaj okvir za rad nije korišten iz jednog ključnog razloga. Nažalost Vuetify za vrijeme izrade ovog završnog rada i dalje ne podržava Vue 3, nego radi isključivo na Vue 2 inačici. Podrška za Vue 3 bi trebala uskoro doći, ali do tad Vuetify nažalost nije opcija za Vue 3 [22]. Vue 3 predstavlja budućnost ovog okvira za rad te ovo nije vrijeme kada bi se trebale izrađivati Vue 2 aplikacije, nego vrijeme kada bi se Vue 2 aplikacije trebali migrirati na Vue 3 verziju. Kreiranjem Vue 2 aplikacije riskiralo bi se brzo zastarijevanje aplikacije što predstavlja dodatni trošak i gubljenje vrijednih resursa, povrh toga što Vue 3 pruža broje mogućnosti koje poboljšavaju rad u ovom okviru za rad.

2.3.3. Vuestic UI

Nakon duge potrage koja se sastojala od čitanja puno dokumentacije, isprobavanja brojnih komponenata te dubinsko istraživanje svojstava različitih biblioteka naišao sam na Vuestic UI. Prema službenoj dokumentaciji, Vuestic je opisan kao UI okvir za rad koji omogućava implementaciju iznimno sofisticiranih dizajnerskih ideja [23]. Za razliku od Vuetify-a, Vuestic u potpunosti podržava Vue 3 i Nuxt 3 okvire za rad što je iznimno bitno za izradu ovog završnog rada te sadrži preko 50 komponenti koje se mogu koristiti u raznim situacijama. Također, Vuestic pruža potpunu „tree-shaking“ podršku koja omogućava optimizaciju web mjesta koje koristi ovaj UI okvir za rad [23]. Nadalje, sve komponente se vrhunski prilagođavaju ekranima raznih veličine, što je izrazito bitno za kreiranje modernog web mjesta.

U usporedbi s Vuetify okvirom za rad, Vuestic ima puno manji broj korisnika, stručnjaka koji doprinose ovom projektu te samim tim je smanjena pouzdanost rada svih komponenti [24]. Unatoč tome, tim koji trenutno razvija ovaj okvir za rad je iznimno predan projektu te osim toga što članovi tima posjeduju izvrsno znanje također su i posvećeni cijeloj zajednici koja koristi

njihov okvir za rad. Iz osobnog iskustva mogu reći kako su voditelji projekta jako brzo izlazili u susret kad god bi se javile poteškoće u redu te su iznimno brzo popravljali te probleme. Ovakav transparentni i prijateljski odnos prema zajednici govori dosta o posvećenosti i predanosti ovom projektu.

Za razliku od Naive UI biblioteke, Vuestic ima iznimno jasnu i konkretnu dokumentaciju koja olakšava njegovo korištenje. Instalacija i početak rada s Vuestic-om je iznimno jednostavan i brz, te je moguće započeti rad s njim u roku od svega nekoliko minuta. Svaka komponenta je detaljno opisana u dokumentaciju, navedena su sva svojstva komponenti koja se mogu modificirati i prilagođavati potrebama pojedinog projekta. Navedena modifikacija je iznimno bitna jer omogućava fleksibilnost u radu te mogućnost prilagođavanja situacijama koje se pojavljuju za vrijeme izrade projekta. Također, komponente je lako uređivati koristeći Tailwind CSS, što je još jedan alat koji je korišten za stiliziranje komponenti te je njegova integracija u ovaj projekt bila iznimno glatka i nije predstavljala nikakve poteškoće.

Izbor komponenti koji pruža Vuestic UI je izrazito raznolik te pruža neke komponente koje su izrazito bitne za web dućan. Nadalje, te komponente su jako lijepo dizajniranje i svojom pristupačnošću omogućavaju pozitivno korisničko iskustvo svim svojim korisnicima. Ako povrh toga navedemo kompletnu, konkretnu i korisnu dokumentaciju te pozitivnu zajednicu programera koji koriste ovaj okvir za rad dobijemo jedan iznimno dobar okvir za rad koji je na usponu te bi jako lako mogao postati glavni izbor za brojne Vue programere. Iz tih razloga je Vuestic odabran kao UI okvir za rad koji će biti korišten za izradu ovog završnog rada.

2.4. Karakteristike Laravel okvira za rad

Unatoč tome što je korisničko sučelje neophodno za pozitivno korisničko iskustvo i za uspješno, moderno web mjesto izrazito je bitno imati i čvrste temelje u pozadini, odnosno dobru izvedbu na strani poslužitelja (engl. backend). Strana poslužitelja označava dio web mjesta koji korisnik ne vidi i ne može mu pristupiti, a odgovoran je za brojne procese kao što su:

- Procesiranje zahtjeva poslanih na poslužitelj
- Pristupanje podacima u bazi podataka
- Kreiranje, ažuriranje i brisanje podataka iz baze podataka

I mnogim drugim [25].

Zajedničkim radom, strana poslužitelja (engl. Frontend) i strana poslužitelja surađuju kako bi stvorili ugodno korisničko iskustvo. Strana klijenta prosljeđuje zahtjev strani poslužitelja, dok servis strane poslužitelja obrađuje rezultate, komunicira direktno s bazom podataka i vraća odgovor klijentskoj strani [25]. Ako je ovaj kanal komunikacije uspješno

implementiran, onda je ostvaren iznimno produktivan način rada koji osigurava ugodno korištenje web mjesta.

Na tržištu je moguće pronaći pregršt različitih jezika, alata i okvira za rad koji omogućavaju izradu kvalitetnog servisa na strani poslužitelja, stoga je potrebno proučiti sve mogućnosti i na temelju istraživanja donijeti finalnu odluku. PHP je najpopularniji programski jezik koji se koristi u ovu svrhu, što potvrđuje činjenica da ga koristi 77.4% web mjesta kojima je poznat programski jezik na strani poslužitelja. Osim toga koriste ga neke ogromne stranice kao što su Wikipedia, Facebook, Zoom i Instagram [26].

PHP je iznimno moćan jezik ali posjeduje pojedina ograničenja. Kreiranje većeg projekta koristeći čisti PHP je izrazito izazovan zadatak te je zbog toga potrebno koristiti brojne alate. Jedan od tih alata je Laravel okvir za rad koji je najpopularniji okvir za rad temeljen na programskom jeziku PHP.

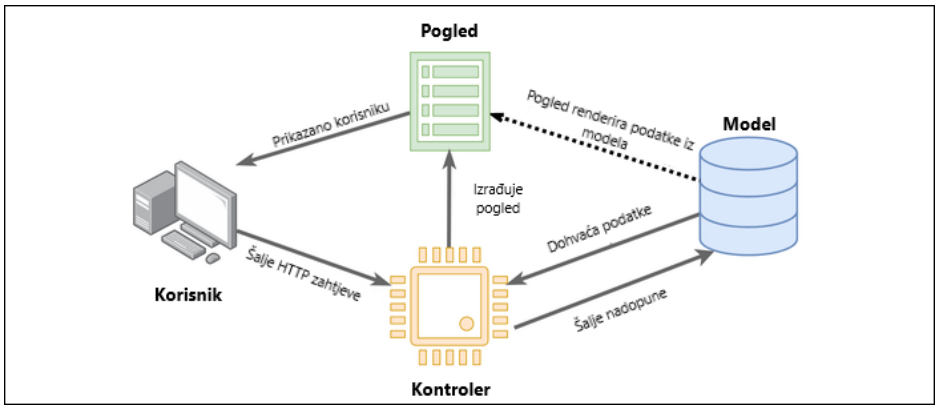
Laravel je PHP okvir za razvoj modernih web aplikacija. Ovaj okvir pruža stabilnu strukturu i početnu točku za kreiranje aplikacije te preuzima veliki dio zadataka na sebe dok omogućava korisniku da se posveti drugim područjima aplikacije [27]. Službena Laravel dokumentacija navodi nekoliko razloga zašto koristiti ovaj okvir za rad. Jedan od njih je to što je Laravel progresivan okvir za rad u smislu da raste zajedno s programerom. To znači da je Laravel iznimno prilagodljiv alat i mogu ga koristiti programeri različitih razina znanja. Početnici imaju pristup dokumentaciji i raznim vodičima, dok iskusniji programeri imaju pristup naprednim alatima koji im omogućuju izradu profesionalnih, optimiziranih aplikacija [27]. Nadalje Laravel je iznimno skalabilan te sadrži široku zajednicu koja svakodnevno nadograđuje Laravel ekosustav novim paketima i pomaže ostalim programerima u razvoju aplikacija [27].

Laravel omogućava dva glavna načina rada. Jedan od njih je izrada aplikacije skupom programskih okvira, odnosno izrada strane klijenta i strane poslužitelja u istom alatu. Dokumentacija tvrdi kako je ovo najčešći način izrade aplikacija te ujedno i njihova preporuka za izradu aplikacija. Za izradu sučelja klijentske strane Laravel omogućava korištenje Blade predložaka ili hibridnu tehnologiju kao što je Inertia. Nadalje, Laravel je moguće koristiti za izradu RESTful API servisa koji će posluživati aplikaciju izrađenu nekim drugim okvirom za rad.

U ovom završnom radu strana poslužitelja će biti implementirana u obliku RESTful API servisa. Ovaj način rada sam odabrao zbog toga što je Nuxt iznimno jak okvir za rad sam po sebi te pruža brojne pogodnosti koje su izrazito bitne za izradu sučelja klijentske strane. Također, na ovaj način je moguće odvojiti repozitorij za servis na strani poslužitelja i sučelja

klijentske strane. Ovo omogućava povećanu urednost projekta i olakšava kontroliranje projekata za vrijeme njihovog rasta.

Laravel radi u potpunosti na strani poslužitelja koristeći MVC (engl. Model-View-Controller) uzorak dizajna [28]. Model predstavlja oblik podataka koji koristi aplikacija. Primjerice model proizvoda u dućanu sadrži sve informacije o proizvodu. Kontroler sadrži metode koje služe za interakciju s modelom. Metode mogu biti korištene za dohvaćanje, kreiranje, ažuriranje te brisanje proizvoda ali isto tako i za druge operacije kao što je ocjenjivanje proizvoda. Kontroler je u direktnoj komunikaciji s modelom te ga ažurira shodno zahtjevu korisnika. Samim tim kontroler sadrži većinu logike koja je korištena u aplikaciji. Pogled (engl. View), predstavlja predložak koji kontroler izrađuje koristeći informacije koje dobije iz modela. Pogled sadrži sve HTML komponente aplikacije [28].



Slika 11. MVC način rada (https://www.howtogeek.com/devops/what-is-laravel-and-how-do-you-get-started-with-it/)

MVC uzorak je nešto što je jako rano objašnjeno u ciklusu učenja Laravela te njegovo razumijevanje omogućava puno brži početak razvoja aplikacija. Pravilno razumijevanje MVC uzorka također omogućava praćenje normi prilikom razvoja aplikacije što doprinosi izradi stabilne aplikacije i pisanje urednog, razumljivog i održivog koda.

Laravel također posjeduje sposobnost kreiranja migracija, koje omogućuju povezivanje modela sa stvarnom bazom podataka i kreiraju komunikacijski kanal između kontrolera i baze podataka. Osim toga migracije predstavljaju i sustav verzioniranja baze podataka te omogućava pregled strukture baze podataka kroz cjelokupni životni ciklus aplikacije. Migracije je tada moguće vraćati, osvježavati i poništavati ovisno o potrebama [29].

Kreiranje modela, kontrolera i migracija je izrazito jednostavno koristeći naredbeni redak (engl. Terminal). U slučaju da želimo kreirati model proizvoda, potrebno je samo upisati sljedeću naredbu:

```
php artisan make:model Proizvod --migration
```

Make:model naredba upućuje na kreiranje modela, dok *--migration* zastavica omogućava automatsko kreiranje migracije iz novog modela. Migraciju je potrebno kreirati kako bi se model mogao implementirati u bazi podataka. Implementacija modela u bazu podataka se ostvaruje naredbom:

```
php artisan migrate
```

Nakon uspješne migracije, potrebno je kreirati kontroler koji će sadržavati svu logiku za rukovanje podacima novokreiranog modela. Kontroler je moguće kreirati sljedećom naredbom:

```
php artisan make:controller ProizvodController --api
```

Kontroler je kreiran samom naredbom *make:controller*, dok –api zastavica omogućava automatsko kreiranje metoda za sve CRUD (create, read, update, delete) operacije , koje je naknadno moguće urediti. Naravno, moguće je brisanje navedenih metoda i dodavanje novih, ovisno o potrebama [30].

Kreatori Laravela su u više navrata naglasili kako inzistiraju na pisanju lijepog koda i olakšavanju života programerima koji koriste ovaj okvir [31]. Ovo upućuje na glavnu svrhu Laravela – omogućiti programerima da kreiraju odlične aplikacije bez brige na pojedine ključne detalje. Tijekom razvoja Laravel aplikacije, programer ne mora trošiti vrijeme na brojne detalje na koje bi morao paziti da razvija aplikaciju u čistom PHP-u. Laravel će se pobrinuti za te detalje o pozadini, dok se programer može posvetiti općem dizajnu svojih modela, kontrolera i pogleda.

Laravel uz to pruža i ostale brojne prednosti. Razvoj aplikacija je iznimno brz te omogućava da se u kratkom rasponu vremena pređe od početka razvoja do izdavanja aplikacije na tržište [31]. Veliki broj eksternih paketa nadopunjuje pojedine nedostatke Laravela i pruže dodatne metode koje olakšavaju rad. Jedan od njih je Laravel Sanctum koji pruža olakšanu autentifikaciju i autorizaciju korisnika [31]. Laravel Forge je također izvrstan servis koji omogućava lako izdavanje aplikacije i njezino javno korištenje uz kreiranje besplatnog SSL (engl. Secure Socket Layer) certifikata. Također, Laravel ima izvrsnu potporu za servise koji služe za pohranu datoteka na poslužitelju, kao što je AWS S3 Bucket. Nadalje omogućeno je automatsko testiranje koda koje osigurava njegov rad u različitim situacijama [31].

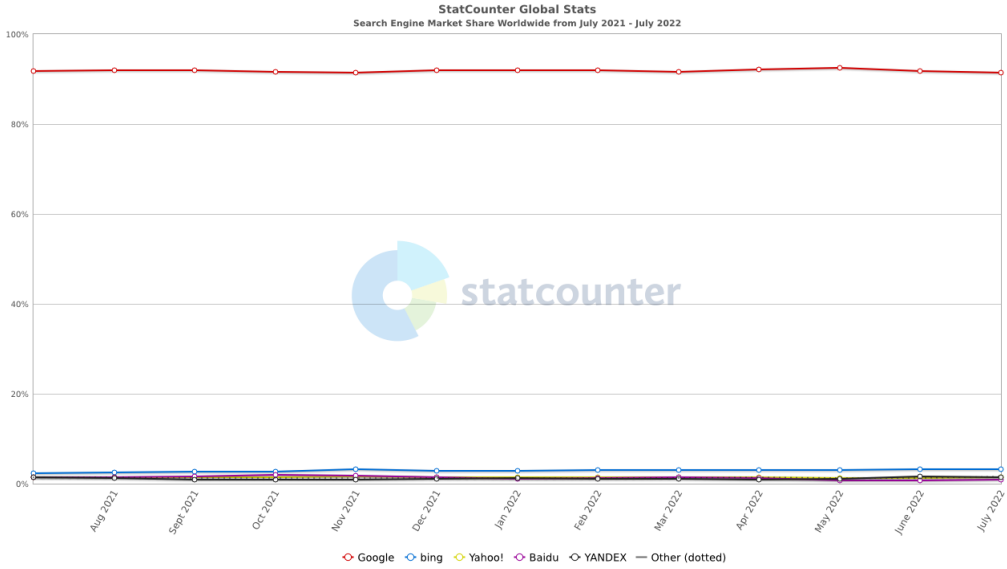
2.5. Alati za optimizaciju

Izrada samog web mjesta je samo jedan dio u cjelokupnom procesu izrade ovakvog projekta. Nakon uspješne izrade, navedeno web mjesto je potrebno testirati i vidjeti gdje još ima mjesta za napredak kako bi gotovi proizvod bio što bolji. Potrebno je testirati razna područja, to su pouzdanost funkcionalnosti, ako sve funkcionalnosti rade na pravilan način,

performanse web mjesta, testiranje pristupačnosti te SEO testiranje. Srećom postoje brojni alati koji su napravljeni baš u tu svrhu, te ću sada kratko opisati neke od njih koji će biti korišteni za vrijeme izrade ovog završnog rada.

2.5.1. Lighthouse

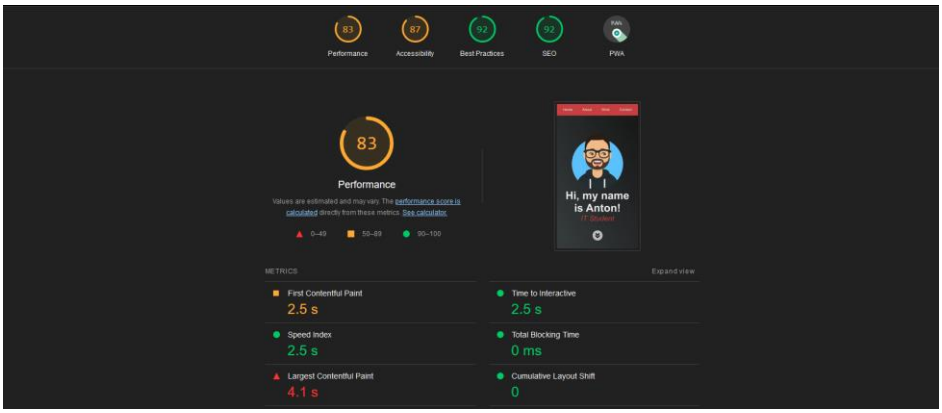
Činjenica da je Google najpopularnija web tražilica je općepoznato. Unatoč tome što postoji veliki broj web tražilica, Google zauzima preko 92% svih pretraga na internetu [32]. Svojim preporukama i kvalitetom je prošao svoju konkurenciju i postao preferirana tražilica velikoj količini korisnika.



Slika 12. Udio internet pretraga podijeljeno po tražilici (https://gs.statcounter.com/search-engine-market-share)

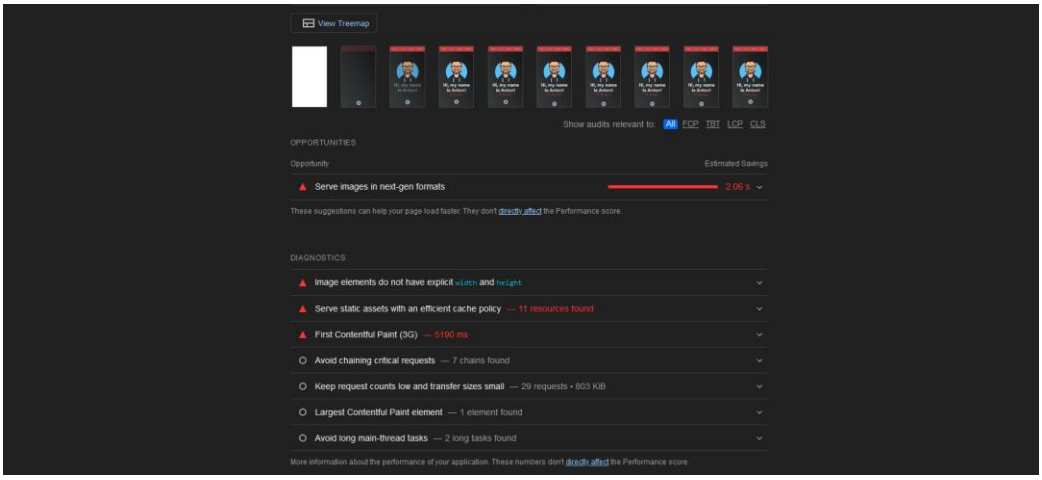
S obzirom da većina korisnika koristi Google, imalo bi smisla koristiti Google-ov alat za analiziranje svojstava kao što su SEO, performanse i slično. Upravo to predstavlja Google Lighthouse, jedan vrhunski alat koji analizira sve ove stvari koje su bitne za uspjeh web mjesta.

Lighthouse radi na principu da web mjestu dodijeli ocjenu 0-100 u 5 kategorija, performanse, pristupačnost, najbolja praksa, SEO i progresivna web aplikacija [33]. Pošto su ovo ključna područja u kojima web mjesto treba ispuniti minimalne uvjete za uspješno poslovanje, Lighthouse daje ključne informacije za analizu web mjesta.



Slika 13. Primjer rezultata Lighthouse analize (vlastita izrada)

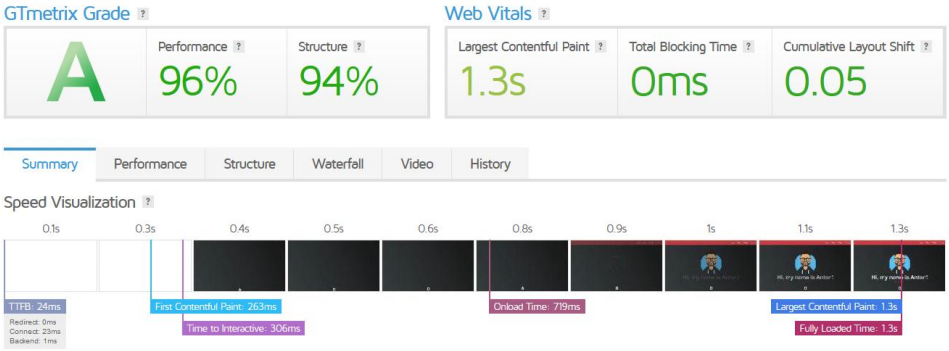
Osim toga, Lighthouse pruža različite metrike koje nam daju bolji dojam o tome što točno treba unaprijediti kod web mjesta. To su vrijeme do pojavljivanja prve slike, vrijeme dok stranica ne postane interaktivna, indeks brzine itd. Nadalje, Lighthouse pruža razne savjete kako bismo ubrzali svoje web mjesto. Ovako precizna analiza točno navodi problematične elemente, razlog zašto su oni problematični te koliko vremena je moguće uštediti rješavanjem navedenih problema.



Slika 14. Detaljna analiza korištenjem alata Google Lighthouse (vlastita izrada)

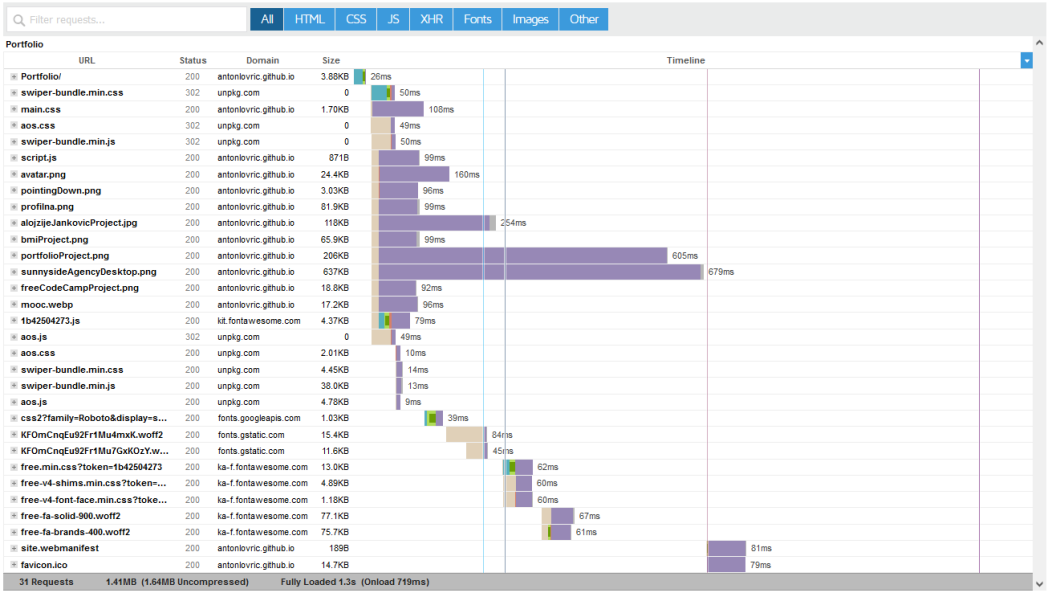
2.5.2. GTMetrix

GTMetrix je još jedan alat koji služi za analizu brzine i davanje povratnih informacija vezane za unaprjeđivanje web mjesta [34]. Nakon pokretanja analize, GTMetrix dodjeljuje web mjestu ocjenu od F – A. Nadalje, GTMetrix vraća dodatne podatke u podijeljene u više kartica među kojima su: sažetak, performanse, struktura, vodopad, video te povijest.



Slika 15. Primjer GTMetrix analize (vlastita izrada)

Većina metrika koje pruža GTMetrix su jednake onima koje pruža i Lighthouse, kao što su vrijeme do pojavljivanja prve slike, vrijeme dok stranica ne postane interaktivna i tako dalje. Važnost GTMetrix-a je u tome što pruža različite grafove koji omogućuju lakše unaprjeđivanje stranice. Pregledom videa učitavanja stranice moguće je točno odrediti koji dio stranice usporava proces te pregledavanjem vodopada učitavanja moguće je analitički odrediti koji nedostaci se nalaze na web mjestu.



Slika 16 GTMetrix vodopad učitavanja (vlastita izrada)

3. Razrada teme

Pravilni odabir alata koji će biti korišteni prilikom izrade ovog završnog rada je neophodan za kreiranje novog, uspješnog proizvoda. Nakon uspješne potrage za pravim alatima, njihovim uspoređivanjem i odlučivanjem o optimalnim alatima potrebno je početi s izradom novog proizvoda.

Proces izrade novog web mjesta je izrazito složen i sastoji se od više koraka. Prvi među njima je analiza trenutnog proizvoda koji je potrebno unaprijediti te usporedba s konkurentnim web mjestima. Ovaj korak je izuzetno bitan jer nam omogućava pregled trenutno situacije i brže određivanje ciljeva novog proizvoda. Također je potrebno odrediti ciljanu publiku te svrhu proizvoda kojeg je potrebno izraditi. Nakon toga je potrebno odrediti konkurentne organizacije te vidjeti način na koje one rješavaju različite probleme. Tek onda je moguće implementirati novo rješenje. Implementacija se sastoji od 4 glavna koraka:

1. Dizajn web mjesta
2. Izrada baze podataka
3. Izrada RESTful API servisa na strani poslužitelja
4. Izrada sučelja klijentske strane

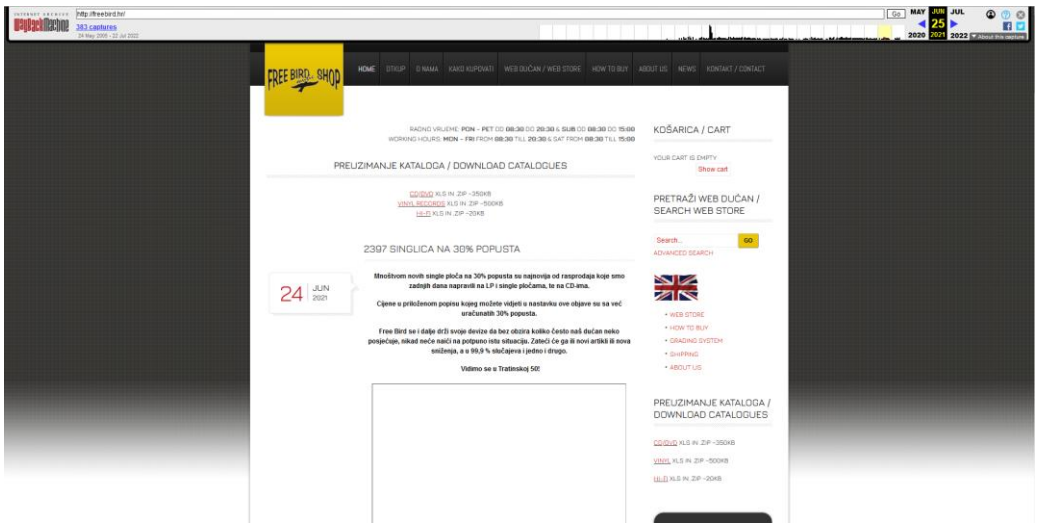
Svaki od ova 4 koraka je izrazito važan u procesu izrade novog web mjesta. Dizajn web mjesta omogućava pravilno planiranje izgleda web mjesta kako bi se ostvarilo pozitivno korisničko iskustvo u smislu estetskog izgleda te ugodnog iskustva za vrijeme korištenja funkcionalnosti web mjesta. Baza podataka mora biti pravilno izrađena kako bi podaci bili organizirani na način da se mogu postići visoke brzine rukovanja podacima. Servis na strani poslužitelja osigurava sigurnost podataka i komuniciranje s bazom podataka te samim tim predstavlja ključan komunikacijski kanal između korisnika i brojnih podataka. Nakon toga potrebno je izraditi sučelje klijentske strane koje će predstavljati vanjski izgled cijelog stroja koji predstavlja ovo web mjesto. Sučelje klijentske strane ima iznimno veliku važnost jer omogućava korisniku da koristi sve usluge koje su implementirane na servisu strane poslužitelja. Bez dobrog sučelja klijentske strane mnoge značajke stranice neće doći do izražaja te je zbog toga neizmjerno bitno.

3.1. Analiza postojećeg web mjesta

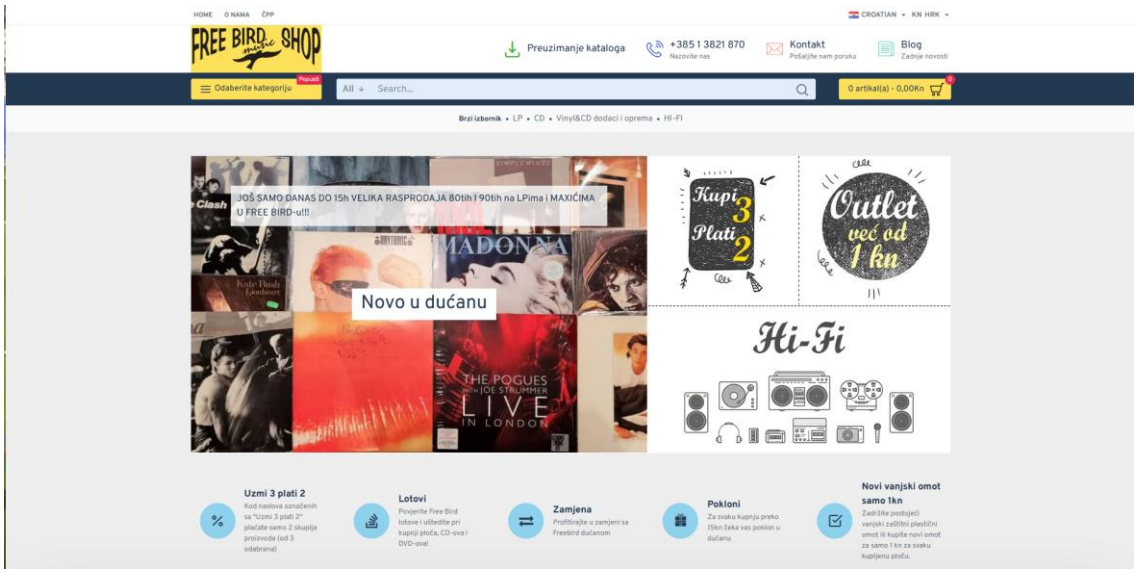
Free Bird music shop je dućan nastao 1994. godine te specijalizira u trgovini gramofonskim pločama i drugim nosačima zvuka u Hrvatskoj [35]. Svojim kontinuiranim i neumornim radom kroz dugi niz godina Free Bird se uspostavio kao jedan od ako ne i najvećom trgovinom svog tipa na ovim područjima. Osim prodaje glazbenih uređaja i medija,

Free Bird također pruža usluge popravljanja i otkupljivanja navedenih predmeta [35]. Free Bird uvijek postavlja svoje kupce na prvo mjesto, čestim sniženjima i pružanja vrhunskih usluga svojim kupcima.

Trenutno web mjesto Free Bird dućana je veliki napredak u usporedbi s prethodnom inačicom. Implementiran je novi, oku ugodniji dizajn koji je ima i pojedine nove značajke koje unapređuju cijelo web mjesto. Stara verzija web mjesta nije imala online katalog, nego je bio priložen veći broj excel datoteka koje je korisnik morao ručno skidati te ih, opet ručno, pregledati. Ovaj način rada očito ne odgovara jednom web dućanu, nego više web mjestu koje služi kao oglasna ploča za informacije o dućane te svakakve novosti vezane za dućan. Izradom novog web mjesta je implementiran i online katalog koji omogućava bržu pretragu i pregled te online kupovinu željenih artikala, što je za pohvalu. Također je preuređen dio s novostima vezanih za dućan, prikazan je veći broj informacija te su predstavljene na ljepši način. U svakom slučaju napravljeni su veliki koraci prema modernom web mjestu te su implementirani brojni napreci.



Slika 17. Izgled naslovne stranice Free Bird web dućana 2021. godine (https://web.archive.org/web/20210625024623/http://freebird.hr/)



Slika 18. Izgled naslovne stranice Free Bird web dućana u vrijeme pisanja ovog rada (https://freebird.hr/)

Unatoč tome što je Free Bird Music uzeo velike korake prema naprijed u smislu razvoja svog web dućana, ostali su još neki problemi u radi i poneka područja gdje ima dosta mjesta za napredak.

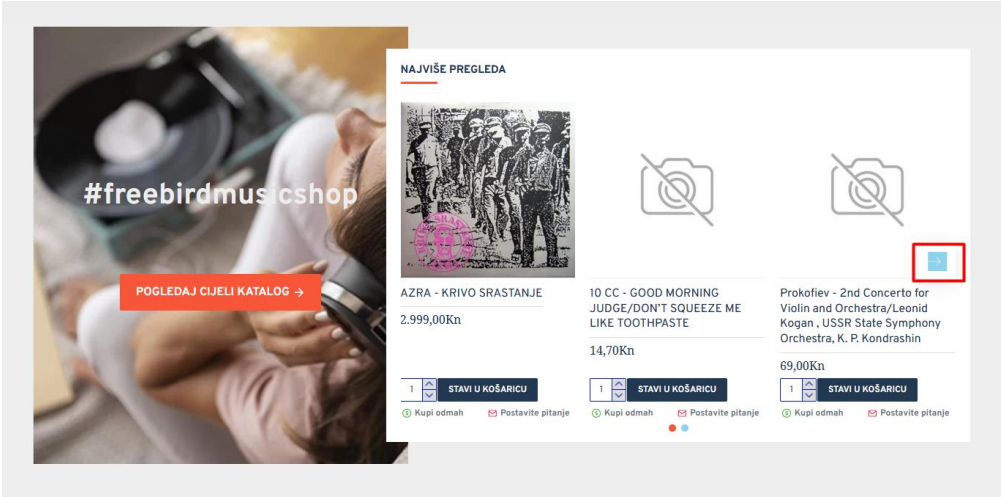
Sam dizajn web mjesta je puno bolji nego prije, ali i dalje postoje neki elementi čija funkcionalnost može predstavljati problem. Prvi primjer toga su klizači koji su korišteni na više mjesta. Klizače možemo vidjeti na naslovnoj stranici kako su korišteni za prikaz novosti i prikaz popularnih proizvoda. U svijetu web razvoja klizači su postali izrazito popularni zadnjih nekoliko godina. Postoje brojni razlozi zašto brojna web mjesta koriste klizače [36]:

- Ugodan izgled
- Mogućnost isticanja važnih informacija
- Prikaz većeg broja značajki vezano za proizvode
- Prikaz većeg broja fotografija vezano za proizvod
- Uzimanje pažnje korisniku
- Stvaranje dojma modernog web mjesta

Iako su klizači iznimno popularni na današnjim web mjestima oni nisu dobar izbor za prikaz informacija. Korištenje klizača može postati neophodno u nekim slučajevima, kao što je ograničena veličina ekrana. Ako veličina ekrana nije dovoljno velika da se prikažu svi elementi, korištenje klizača je potpuno legitimna i razumna odluka. Usprkos tomu, potrebno je izbjegavati klizače u slučajevima u kojima nisu prijeko potrebni te pronaći različite alternative za njih.

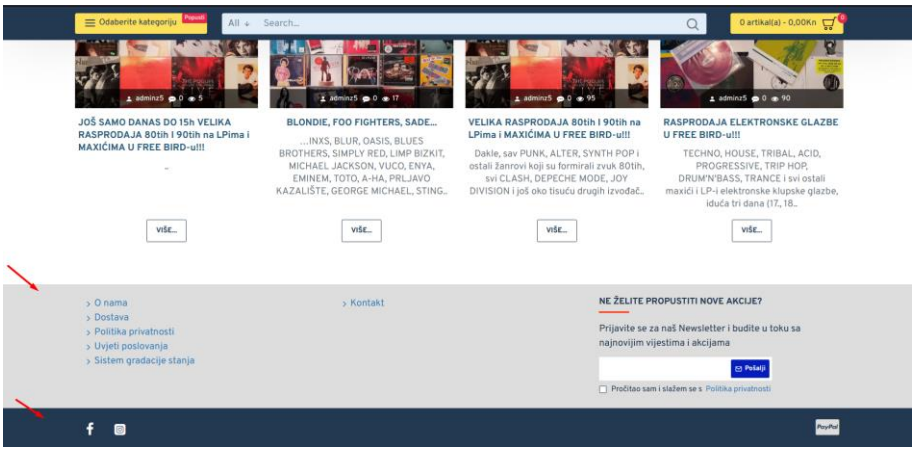
Postoje brojni razlozi zašto klizači nisu dobar izbor na web mjestima. Jedan od njih se naziva „Oglasna sljepoća“ [37]. Oglasna sljepoća (engl. banner-blindness) označava pojavu gdje korisnici podsvjesno ignoriraju sve što nalikuje reklamama. Ignoriraju takav sadržaj pošto

ih on ne zanima i pažnju im odvlači glavni sadržaj stranice [38]. Pošto brojne organizacije reklamiraju svoj sadržaj u obliku klizača jako je lako asocirati klizače s reklamama. Adam Fellows je čak naveo da je za vrijeme svog testiranja uočio da velika količina sadržaja koja se nalazi u klizačima prođe nezapaženo uslijed gubitka koncentracije korisnika [39]. Preopterećivanje korisnika može predstavljati iznimno velik problem je osjećaj preopterećenosti može predstavljati negativan osjećaj. U dosta slučajeva je bolja opcija predstaviti korisniku samo glavne proizvode kako bi mu privukli pažnju [39]. Nadalje, klizači predstavljaju velike prepreke korisnicima s poteškoćama. Pošto klizači u većini slučajeva koriste strelice ili točkice za navigaciju, javljaju se problemi vezani za vidljivost navedenih navigacijskih elemenata te u dosta slučajeva i manjak kontrasta između pozadine i tih elemenata [37]. Pristupačnost je problem koji se javlja konkretno na primjeru klizača Free Bird dućana. Klizač na naslovnoj stranici koja je prikazana slikom 18 svoje elemente za navigaciju dobije tek nakon prijelaza miša preko klizača te u potpunosti onemogućava korisnicima s poteškoćama korištenje klizača. Slika 19 ističe element koji je korišten za navigaciju te je očito kako navigacija između pozadine gumba i same strelice nije dovoljna. Osobi s poteškoćama ovaj gumb neće puno značiti te će joj samo predstavljati smetnju. Među ostalom do navedenog gumba nije moguće doći tipkovnicom, što je još jedan nedostatak.



Slika 19. Nepristupačan navigacijski gumb klizača (https://freebird.hr/index.php?route=common/home)

Također jedna zbunjujuća značajka vezana za dizajn je manjak konzistentnosti u pojedinim područjima. Primjerice web mjesto ima dva pretinca koji imaju ulogu podnožja, odnosno sadrži pojedine poveznice koji služe za navigaciju na web stranice koje posjeduju općenite informacije vezane za web mjesto, kao što su informacije o web dućanu, politika privatnosti, uvjeti poslovanja itd.



Slika 20. Podjela podnožja u dva dijela (https://freebird.hr/index.php?route=common/home)

U usporedbi s prethodnom inačicom ovog web mjesta, dodavanje kataloga je bio veliki korak prema naprijed. Katalog omogućava web mjestu da se može nazvati web dućanom te ima veliku korist za sve korisnika web mjesta. Iako je jako dobar dodatak, implementirani katalog ima pojedine poteškoće u svom radu koji narušavaju općenito korisničko iskustvo.

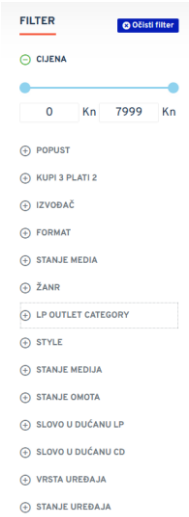
Kategorija koja sadrži više problema u svom radu su filteri proizvoda. Filteri su bitan svake kolekcije podataka koja se prikazuje korisniku. Implementacijom filtriranja proizvoda korisniku se omogućava brža navigacija kroz veliku količinu proizvoda te sužavanjem izbora korisnik može pronaći ono što traži.

Jedan nedostatak kod filtera koji se trenutno nalaze na web mjestu je redundantnost pojedinih filtera. Jedna od kategorija za koju se može reći da je nepotrebna naziva se „Kupi 3 plati 2“. „Kupi 3 plati 2“ akcija koju pruža Free Bird dućan je odlična prilika za pronalaskom dobrih akcija ali to ne bi trebala biti posebna kategorija. Ova akcija bi se logički mogla grupirati s nekim drugim kategorijama kao što su ostali popusti. Na taj način bi korisničko sučelje bilo pojednostavljeno te bi korisničko iskustvo bilo poboljšano.

Još jedna kategorija za koju se može reći da je suvisla je kategorija „Izvođači“. Pregled proizvoda po izvođačima može biti jako koristan krajnjem korisniku ali implementacija ove pretrage putem filtera je kriva iz više razloga. Jedan od tih razloga je prevelik broj filtera. Na svojoj web stranici vlasnici Free Bird Music dućana tvrde kako u svojoj kolekciji uvijek posjeduju oko 80.000 artikala [35]. Uzimajući u obzir ovu brojku može se procijeniti da je to veliki broj različitih izvođača svih tih albuma. Zbog velikog broja izvođača iznimno je teško korisniku snaći se među svim tim opcijama kako bi napokon filtrirali proizvode prema željenom izvođaču. Osim što je iznimno teško snaći se među svim tim opcijama, učitavanje toliko opcija također ima negativan utjecaj na performanse web mjesta te uzrokuje zastajkivanje i sporiji rad web stranice. Također, filter izvođača je jedini filter gdje nije moguće filtrirati na temelju više kriterija. Onemogućavanje korisniku aktiviranje više kriterija negativno utječe na korisničko

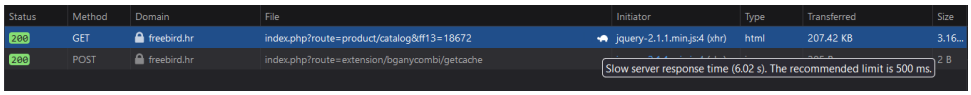
iskustvo te pojava takve situacije jasno govori kako ovaj kriterij nije prikladan korištenju filtera nego bi trebao biti implementiran na drukčiji način, kao što je primjerice tekstualno pretraživanje. Ako pogledamo problem s gledišta dizajna baze podataka, to znači da je potrebno normalizirati tablicu na način da će biti potrebno kreirati novi unos za svakog izvođača te će se referencirati njegov vanjski ključ. Svi ovi razlozi negativno utječu na performanse, korisničko iskustvo ali i programersko iskustvo jer znatno komplicira cijeli zadatak. Osim što nije korisno svojstvo, ono negativno utječe na web stranicu te bi čak bilo korisno u potpunosti izbrisati ovaj filter. Kada uzmemo u obzir da se gotovi svi ovi nedostaci mogu reći i za filter „Style“ onda to već postaju ozbiljni problemi u području filtera.

Još jedan problem je sam broj dostupnih filtera koji se nalaze na web stranici. Filtera je definitivno previše te je ovo još jedna značajka koja može preopteretiti korisnika, a loš utjecaj toga je opisan ranije u ovom radu. Broj filtera se znatno može smanjiti uklanjanjem opcija filtriranja kao što su „Izvođač“, „Style“, „slovo u dućan“ za kategorije CD i LP te duplicirani filter „Stanje medija“. Duplicirani filter „Stanje medija“ se pogotovo ističe kao velika greška te ju je potrebno ispraviti u svakom slučaju. Filtriranje po slovu nije korisno većini korisnika te ne pridonosi veliku važnost sustavu filtriranja.



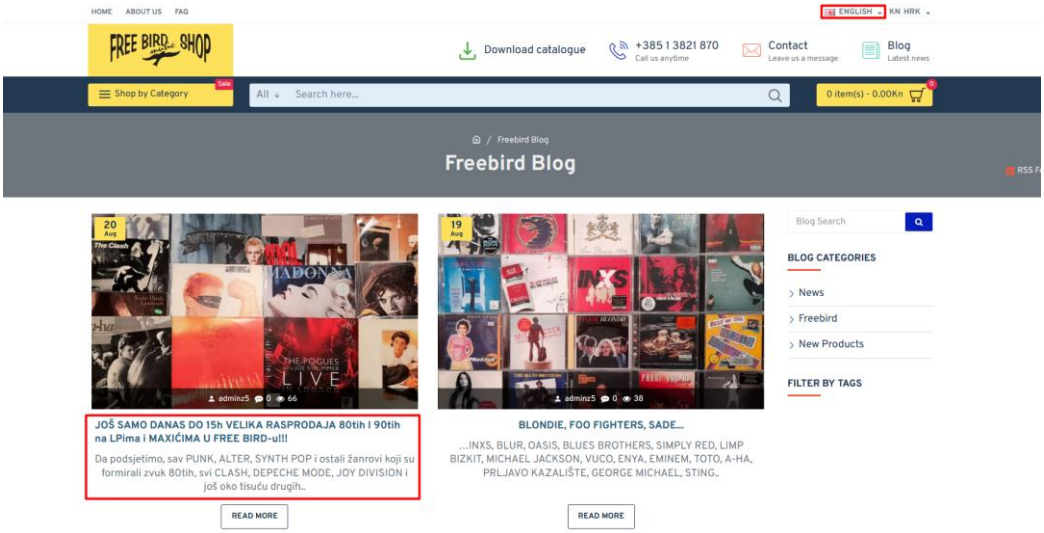
Slika 21. Svi dostupni filteri na web mjestu (https://freebird.hr/index.php?route=product/catalog)

Performanse dohvaćanja filtriranih rezultata su također poprilično loše. Analizom brzine izvršavanja zahtjeva dolazimo do rezultata koji upućuju na to da neki zahtjevi znaju trajati i preko 6 sekundi, što je predugo i može negativno utjecati na korisničko iskustvo. Uzimajući u obzir i to da se filtrirani artikli automatski dohvaćaju u trenutku aktiviranja filtera, odabiranje većeg broja filtera predstavlja jako dug proces koji će gotovo sigurno negativno utjecati na korisničko iskustvo.



Slika 22. Brzina dohvaćanja filtriranih proizvoda (vlastita izrada)

Korištenje dva jezika je jako korisno i zanimljivo svojstvo web mjesta te omogućava znatno većem korisnika da pristupe web mjestu i da ga koriste. Nažalost ovo svojstvo nije pravilno implementirano te dolazi do miješanja jezika. Ranije je opisan „Style“ filter koji je prikazan na hrvatskoj verziji web mjesta unatoč toga što predstavlja riječ na engleskom jeziku. Također postoje i različiti primjeri hrvatskog teksta na stranim verzijama web mjestima. Ovo je vjerojatno uzrokovano time što ručni unos nije preveden, te je ručni unos implementiran jedino na hrvatskom jeziku. Ovo jako lako može prouzročiti pomutnju kod korisnika te je problem koji svakako treba riješiti. U slučaju da nije moguće prevesti ručni unos korisnika, bolje je opredijeliti se za web mjesto na jednom jeziku koje će biti konzistentno u svim situacijama.



Slika 23. Hrvatski tekst na stranoj verziji stranice (https://freebird.hr/index.php?route=journal3/blog)

3.2. Implementacija novog rješenja

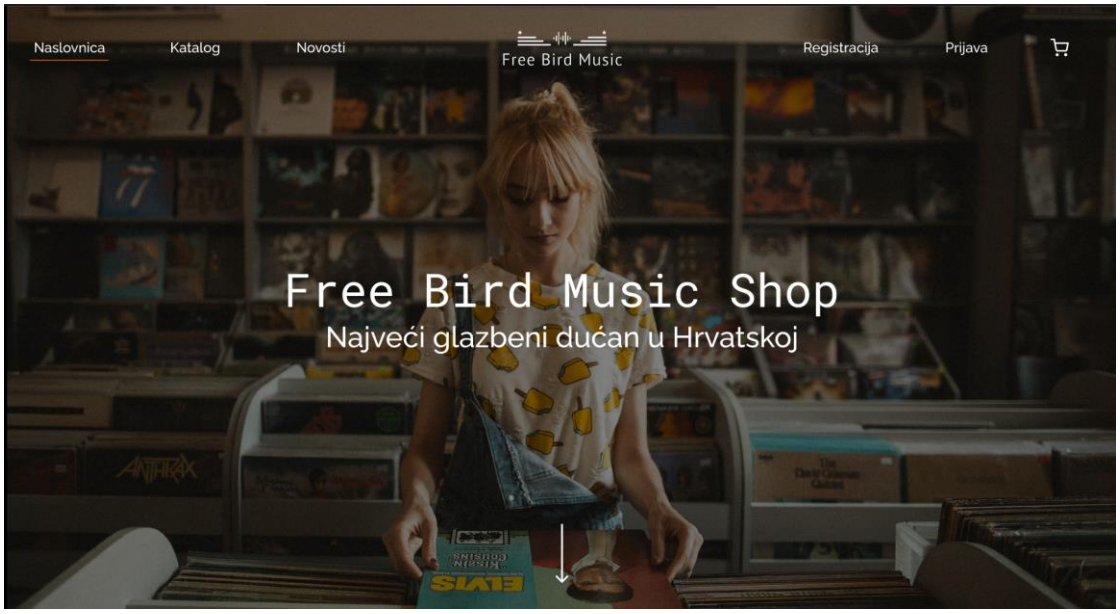
Nakon analize web mjesta koja obuhvaća trenutno stanje web mjesta, njegove nedostatke i prednosti, potrebno je smisliti novo rješenje i implementirati ga. Proces implementacija će biti podijeljen u 4 poglavlja koja su bila opisana ranije: dizajn web mjesta, izrada baze podataka, izrada API-a na strani poslužitelja te izrada sučelja klijentske strane. Svaki od ova 4 koraka je izuzetno važan te su zahtijevali najviše vremena i truda za vrijeme izrade ovog završnog rada.

3.2.1. Dizajn web mjesta

Za pravilno dizajniranje web mjesta potrebno je točno odrediti ciljanu publiku koja će koristiti web mjesto. Skupljanje glazbene memorabilije u obliku gramofonskih ploča, CD-ova i ostalih nositelja medija je bio hobi brojnih obožavatelja glazbe unazad preko 20 godina. Fizički mediji su jedno vrijeme izgubili veliki dio popularnosti dolaskom digitalnog oblika glazbe pošto je glazba bila nadohvat ruke svima koji su imali pristup internetu. Unatoč velikom utjecaju digitalne glazbe, fizički mediji poput vinila se ponovno vraćaju u modu. Steven John u svom članku „Zašto vinili imaju povratak u 2022.“ ističe kako vinili imaju određenu romantičnu energiju te potiču slušanje glazbe kao aktivnost, a ne kao pozadinsku buku [40]. U istom članku se navodi kako preko 60% mladih osoba u dobi od 18 do 29 godina posjeduju gramofon, te mnogi ostali planiraju kupiti gramofon [40]. Ove informacije nam daju zanimljiv dojam o tome kakvo web mjesto bi trebalo dizajnirati. Pomiješana demografija ciljane publike nam govori kako je potrebno implementirati moderan dizajn koji će se svidjeti mlađoj populaciji, ali isto tako dizajn mora biti direktan i konkretan kako bi bio lako razumljiv starijoj populaciji. Nadalje zbog romantične aure koja je usko povezana uz pojam vinila, potrebno je stvoriti ugodan i opušteni ambijent na web mjestu koji će omogućiti korisnicima da uživaju u korištenju web mjesta.

3.2.1.1. Naslovna stranica

Prva web stranica koja će biti dizajnirana je naslovna stranica. Dizajniranje naslovne stranice će postaviti ton za izgled i osjećaj ostatak web mjesta.

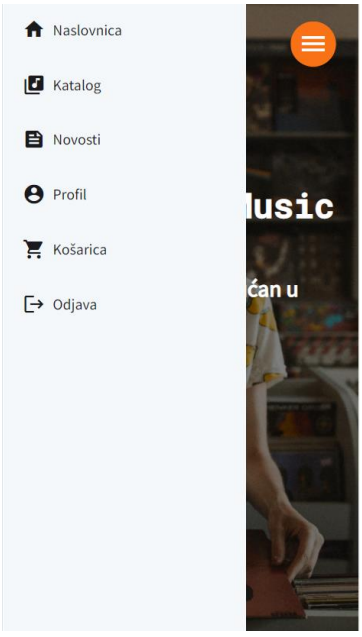


Slika 24. Dizajn hero sekcije naslovne stranice (https://freebird-music.vercel.app/)

Poanta ove hero sekcije je da se odmah prilikom posjećivanja web mjesta ostvari traženi dojam na korisnika. Fotografija u pozadini je ugodna za gledanje te daje poruku da je Free Bird dućan za prodaju glazbene opreme i memorabilije. Ta poruka je dodatno naglašena naslovom i podnaslovom koji djelomično prekrivaju fotografiju. Fotografija je ručno zatamnjena kako bi se implementirao općenito tamniji dizajn web mjesta, radi povećavanja osjećaja ugode, opuštenosti i kako bi se stvorio odgovarajući ambijent.

Na slici 24 je također vidljivo zaglavlje koje je jednako na svim stranicama ovog web mjesta. U sredini sadrži svoj logo te s obje strane je moguće pronaći poveznice koje omogućavaju korištenje osnovnih funkcija web mjesta kao što su pristup katalogu, objavama, prijava i registracija te pregled košarice. Ovakav način dizajniranja zaglavlja je sličan dizajnu web mjesta Zillow, čiji dizajn je opisan ranije u ovom radu i prikazan slikom 6. Zaglavlje je fiksirano na vrh ekrana kako bi bio dostupno korisniku cijelo vrijeme tijekom listanja prema dnu stranice.

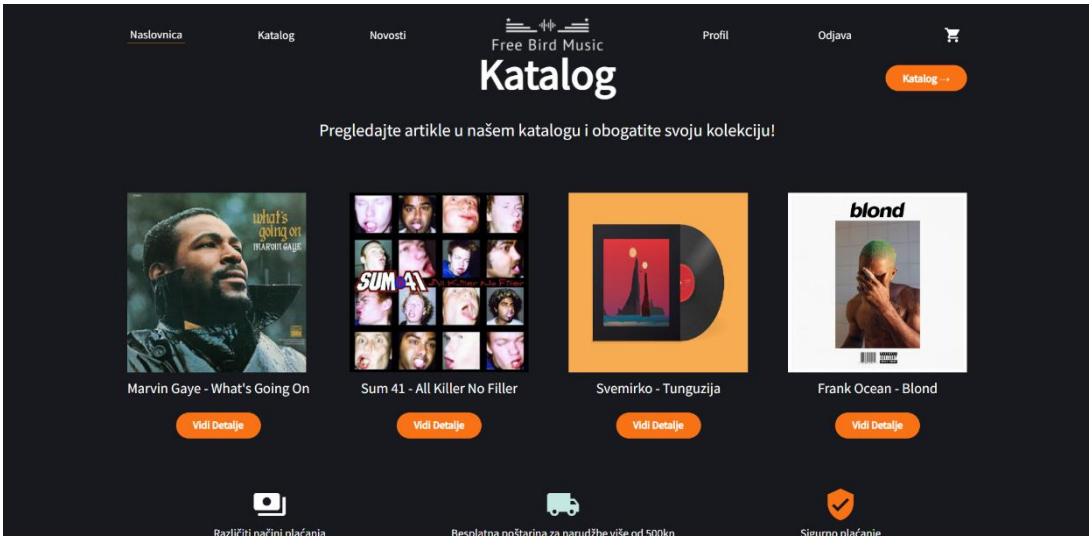
Ovakav dizajn zaglavlja zbog svoje širine, količine sadržaja i organizacije nije prikladan manjim ekranima te je na mobilnim uređajima drukčije implementiran. Na mobilnim uređajima je dodan gumb kojim se otkriva navigacija s lijeve strane ekrana i omogućeno je normalno korištenje navigacije. Ova funkcionalnost ostvaruje urednost ekrana te maksimizaciju njegovog prostora.



Slika 25. Navigacija na mobilnim uređajima (https://freebird-music.vercel.app/)

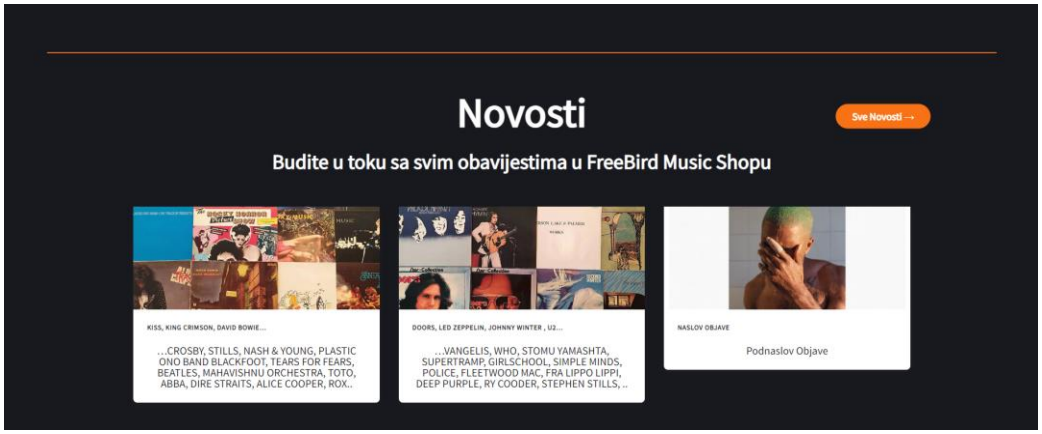
Nakon hero sekcije potrebno je prikazati sadržaj koji može biti dostupan korisnicima. Listanjem prema dnu dolazimo do sekcije s naslovom „Katalog“ te ona služi za prikaz istaknutih proizvoda. Proizvode je moguće istaknuti ručnom manipulacijom podataka u bazi koristeći alat poput Heidi SQL ili MySQL Workbench. Za svaki proizvod je prikazana fotografija i naziv

proizvoda te gumb koji vodi na prikaz detalja proizvoda. Osim prikaza proizvoda, prikazane su i pojedine prednosti kupovine u Free Bird dućanu kao što su sigurno plaćanje, različiti načini plaćanja te besplatna poštarina ukoliko se ispune željeni uvjeti.



Slika 26. Prikaz sekcije Katalog na naslovnoj stranici (https://freebird-music.vercel.app/)

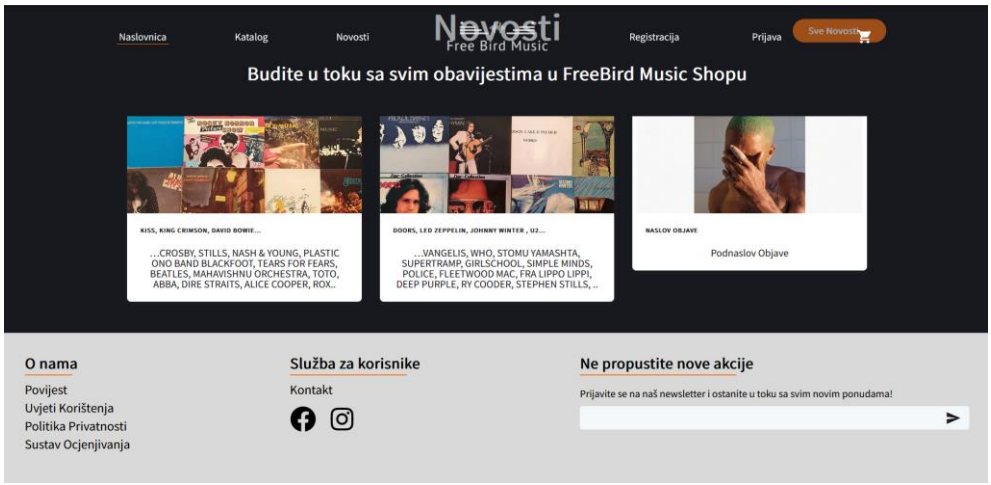
Nakon prikaza kataloga na naslovnoj stranici dolazimo do prikaza sekcije „Novosti“. Ova sekcija služi za prikaz 3 posljednje objavljenje novosti, njihove naslove, naslovne fotografije te podnaslove. Također je moguće otići do stranice koja sadrži sve novosti. Iznad ove sekcije je također prikazan tematski prekid koji naglašava početak i završetak pojedine sekcije.



Slika 27. Sekcija Novosti na naslovnoj stranici (https://freebird-music.vercel.app/)

Na samom kraju naslovne stranice dolazimo do podnožja. Podnožje više nije podijeljeno u dva dijela kao što smo prikazali na trenutnoj verziji web mjesta, nego su sve informacije postavljene u istu sekciju. Na ovaj način je lakše odrediti što je točno podnožje ali nije narušena njegova urednost. Također su poveznice podijeljene u 3 odvojene grupe: „O

nama“, „Kontakt“ i „Ne propustite nove akcije“. Sve grupe su točno naznačene svojim naslovom. Prve dvije grupe su razumljive same po sebi, dok grupa „Ne propustite nove akcije“ sadrži kratki obrazac za prijavu na newsletter koji je moguće automatski kontrolirati putem vanjskog servisa MailChimp.

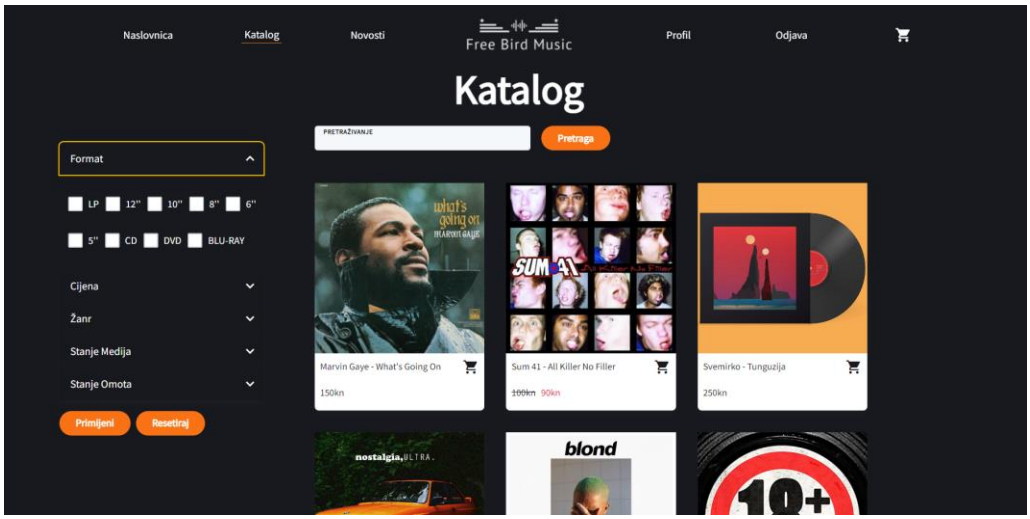


Slika 28. Prikaz podnožja na naslovnoj stranici (https://freebird-music.vercel.app/)

Nakon prikaza kataloga na naslovnoj stranici dolazimo do prikaza sekcije „Novosti“. Ova sekcija služi za prikaz 3 posljednje objavljenje novosti, njihove naslove, naslovne fotografije te podnaslove. Također je moguće otići do stranice koja sadrži sve novosti. Iznad ove sekcije je također prikazan tematski prekid koji naglašava početak i završetak pojedine sekcije.

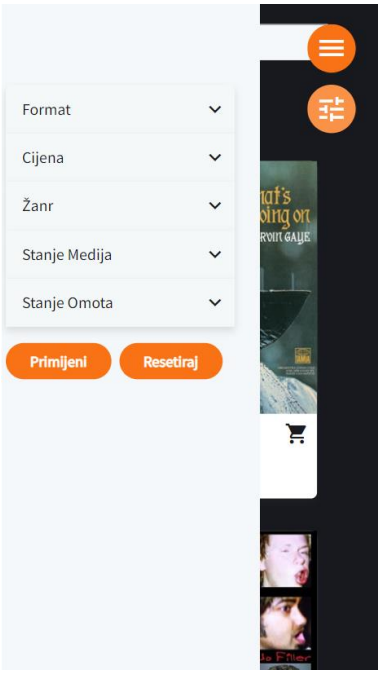
3.2.1.2. Katalog

Stranica katalog je jedna od najvažnijih web stranica na ovom web mjestu pošto sadrži sve proizvode web dućana. Na ovoj stranici korisnik može pregledati sve proizvode, pretraživati ih po naslovu, filtrirati po raznim kategorijama itd. Uz filtre se nalaze dva gumba, jedan za primjenu filtera a drugi za čišćenje svih filtera. Na ovaj način je riješen prethodno opisan problem kod dohvaćanja novih proizvod kod svakog odabira filtera. Proizvodi su prikazani u obliku kartica, gdje svaka kartica sadrži naziv proizvoda, cijenu, popust (u slučaju da je proizvod na akciji) te gumb za dodavanje proizvoda u košaricu. Klikom na svaku karticu je moguće posjetiti detalje proizvoda i provjeriti dodatna svojstva proizvoda kao što su stanje omota i medija, izdanje, žanr itd.



Slika 29. Prikaz stranice katalog (https://freebird-music.vercel.app/catalogue)

Filteri su implementirani u obliku tzv. accordion izbornika. Također su izbačeni redundantni filteri koji su ranije spomenuti i omogućena je pretraga po izvođači na način da je ime izvođača uključeno u naslov. Na ovaj način su filteri pregledniji, uredniji i imaju bolje performanse. Na manjim ekranima su filteri premješteni u pretinac koji se otkriva s lijeve strane na način sličan kao i navigacijskom meniju. Naravno dodan je i još jedan gumb koji omogućava ovu funkcionalnost.

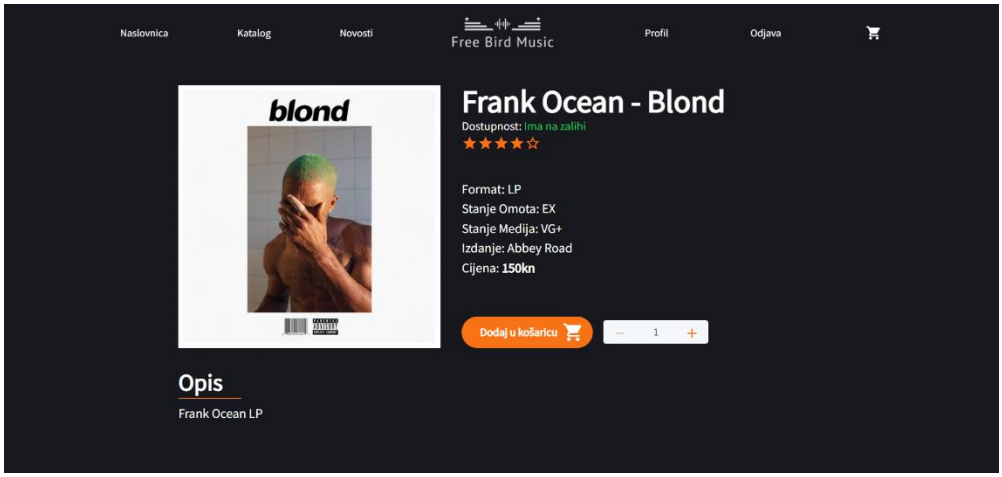


Slika 30. Prikaz filtera kataloga na mobilnim ekranima (https://freebird-music.vercel.app/catalogue)

3.2.1.3. Detalji proizvoda

Proizvodi su neophodan dio svakog web dućana pa je stoga potrebno prikazati što veći broj informacija o tom proizvodu. Prikaz svih informacija nije moguć na karticama proizvoda pošto nema dovoljnog prostora te je iz tog razloga napravljena stranica za prikaz detalja proizvoda. Tu su dostupne sve važne informacije o proizvodima, kao što su:

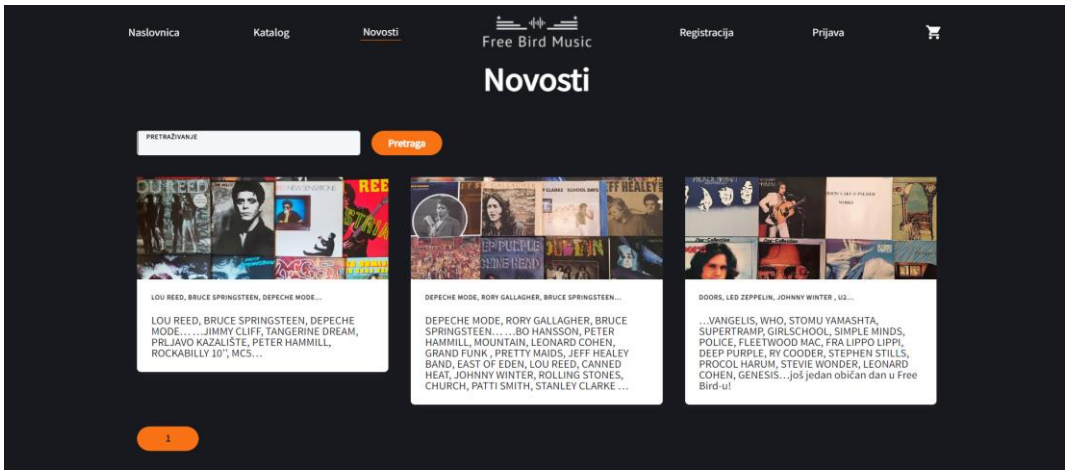
- Naslov
- Dostupnost
- Ocjena
- Format
- Stanje omota i medija
- Izdanje
- Cijena
- Opis



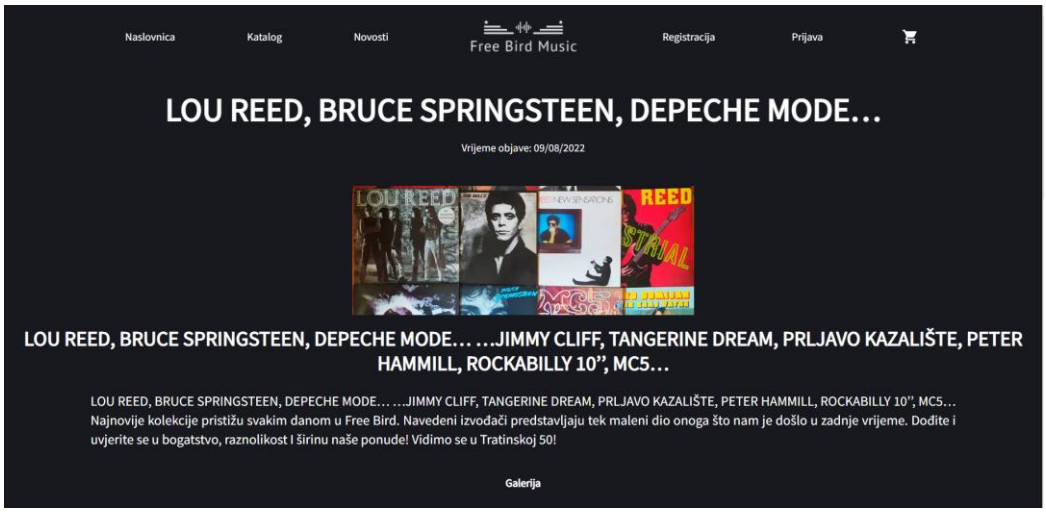
Slika 31. Prikaz stranice detalja proizvoda (vlastita izrada)

3.2.1.4. Novosti

Free Bird dućan je oduvijek imao mogućnost kreiranja novosti na svojim web mjestima. Ovaj sustav je nešto nalik blogu i omogućava vlasnicima naglašavanje novih proizvoda te najavljivanje svakakvih novosti vezanih za dućan. Svojom strukturom stranica za pregled novosti te detalja novosti je simetrično strukturi stranica za pregled proizvoda i detalja proizvoda. Web stranica novosti omogućava pretraživanje novosti prema naslovu (slika 32), dok detalji novosti prikazuju naslov, podnaslov, tekst, datum objave te galeriju fotografija (slika 33).



Slika 32. Web stranica Novosti (https://freebird-music.vercel.app/blogPosts)



Slika 33. Web stranica detalji novosti (https://freebird-music.vercel.app/posts/1)

3.2.1.5. Kupnja proizvoda

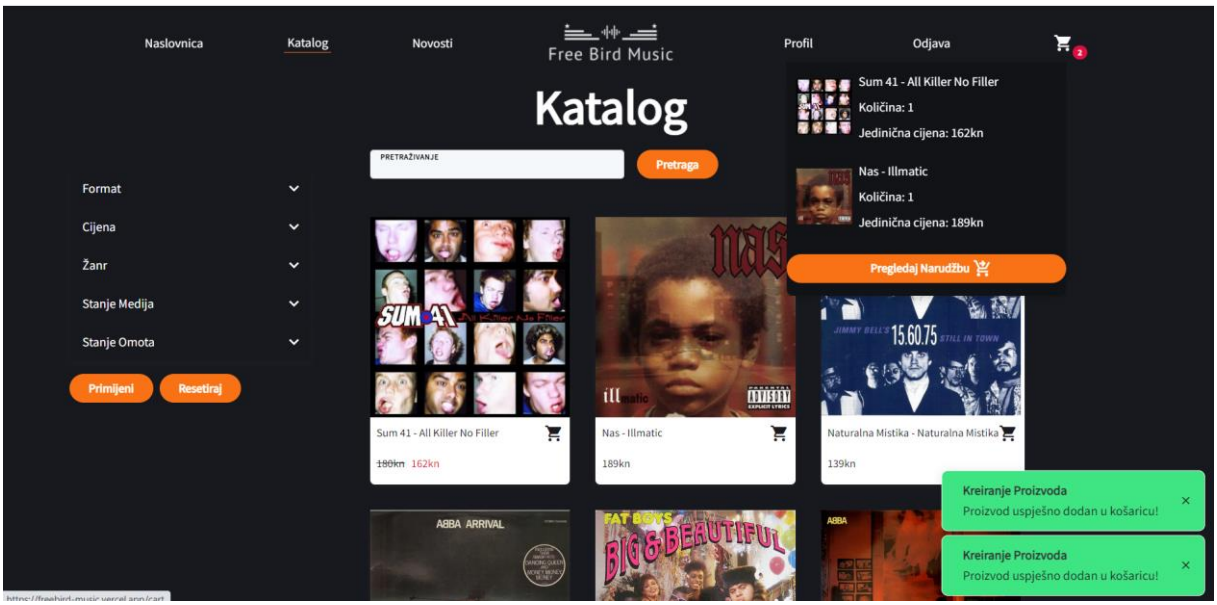
Kupnja proizvoda je osnovna funkcionalnost svakog web dućana te je iznimno bitno ostvariti izvrsno korisničko iskustvo. Cijeli proces kupnje se sastoji od više koraka:

1. Pregled proizvoda
2. Dodavanje u košaricu
3. Pregled košarice
4. Ispunjavanje informacija za naplatu

Pregled proizvoda smo pokrili u prethodnim točkama te opisali kako je moguće pregledati proizvode putem stranice kataloga te putem pregleda detalja proizvoda. Dodavanje u košaricu je također nakratko pokriveno, proizvod je moguće dodati u košaricu prilikom pripadajućih gumba putem pripadajuće ikone na kartici proizvoda ili klikom na gumb „Dodaj u košaricu“ na stranici detalja proizvoda.

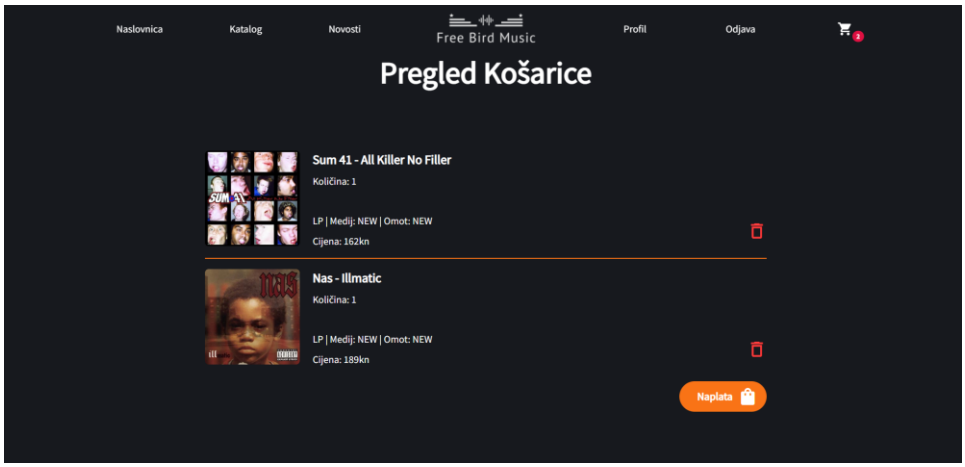
Pregled košarice omogućava korisniku pregled elemenata koje korisnik trenutno planira kupiti. Prilikom pregleda kupovine nije potrebno uklopiti sve moguće podatke pošto ih je korisnik već sam pregledao prilikom dodavanja u košaricu, ali je bitno uklopiti informacije vezane za samu kupovinu, odnosno naziv proizvoda, jedinična cijena i odabrana količina pojedinog proizvoda. Nakon što je proizvod dodan u košaricu ažurira se i ikona košarice u gornjem desnom kutu. Dodan je mali crveni indikator koliko elemenata trenutno ima u košarici te prijelazom miša preko ikone otvara se meni s osnovnim informacijama o navedenim elementima. Na dnu navedenog izbornika se nalazi gumb za pregled narudžbe koji omogućava korisniku da još jednom pregleda i potvrdi detalje svoje narudžbe te nastavi s kupovinom.

Slika 34 također obuhvaća poruke uspjeha dodavanja proizvoda u košaricu. Ovi tipovi poruka se nalaze u brojnim situacijama kroz cijelo web mjesto i daju korisniku korisne povratne informacije o akcijama koje pokušava izvršiti.



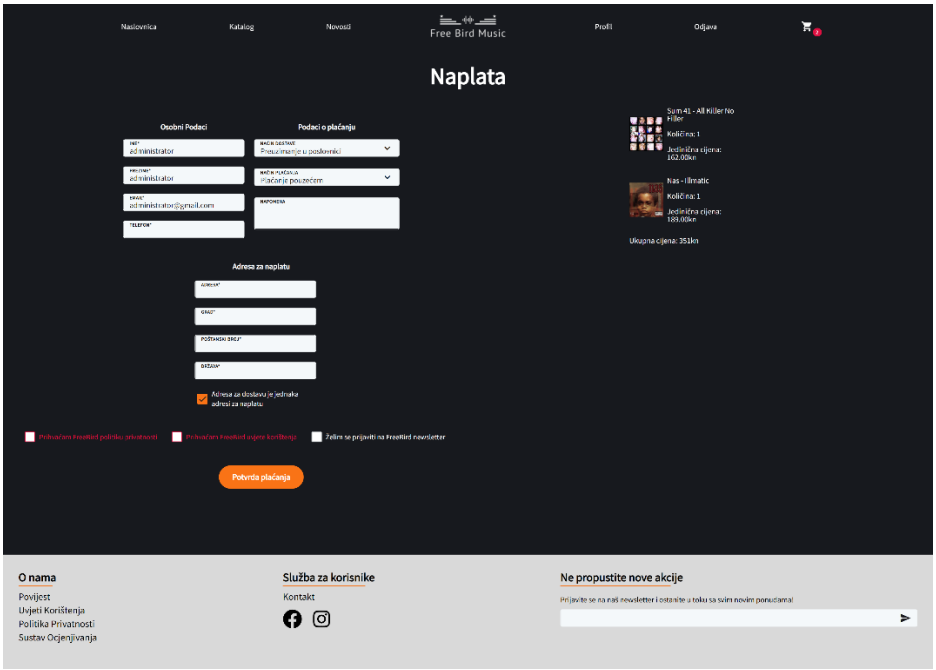
Slika 34. Dodavanje proizvoda u košaricu (vlastita izrada)

Pregled narudžbe je jednostavan ali iznimno bitna stranica jer omogućava korisniku da još jednom pregleda sve informacije o odabranim proizvodima prije nego što nastavi s narudžbom. Pošto elementi sada nisu ograničeni malim menijem koji se otvara na prijelaz miša preko ikone košarice, moguće je priložiti puno više podatak o odabranim proizvodima. Osim elemenata koji su prikazani u prethodno navedenom meniju, prikazani su elementi poput tipa proizvoda, stanja omota i medija te ukupna cijena proizvoda. Uz to je kod svakog proizvoda prikazana ikonica za uklanjanje proizvoda iz košarice. Svaki proizvod je odvojen narančastom linijom radi preglednosti i urednosti te se na kraju nalazi gumb za prijelaz na naplatu.



Slika 35. Pregled narudžbe (https://freebird-music.vercel.app/cart)

Web stranica za naplatu predstavlja posljednji korak u procesu kupovine. Na ovoj stranici se traži od korisnika unos osobnih podataka bitnih za izvršavanje naplate i isporuke. Svi elementi za unos podataka se nalaze na lijevoj strani ekrana dok desna strana ekrana sadrži kratak pregled proizvoda koji se nalaze u narudžbi. Na slici 36 je vidljiv potvrdni okvir koji označava da je adresa za dostavu jednaka adresi za naplatu. U slučaju da ovaj okvir ostane neoznačen stvaraju se još 4 polja za unos koji se odnose na adresu za dostavu. Na kraju obrasca se nalaze 3 okvira za potvrdu od koja su dva obavezna i narudžba neće moći biti izvršena dok se oni ne označe. Obavezni okviri za potvrdu se odnose na prihvaćanje politike privatnosti i uvjeta korištenja, dok se treći, neobavezni, odnosi na prijavu za Free Bird newsletter. Također je implementiran vizualni indikator ako uvjeti nisu ispunjeni u obliku crvenog teksta koji označuje grešku.



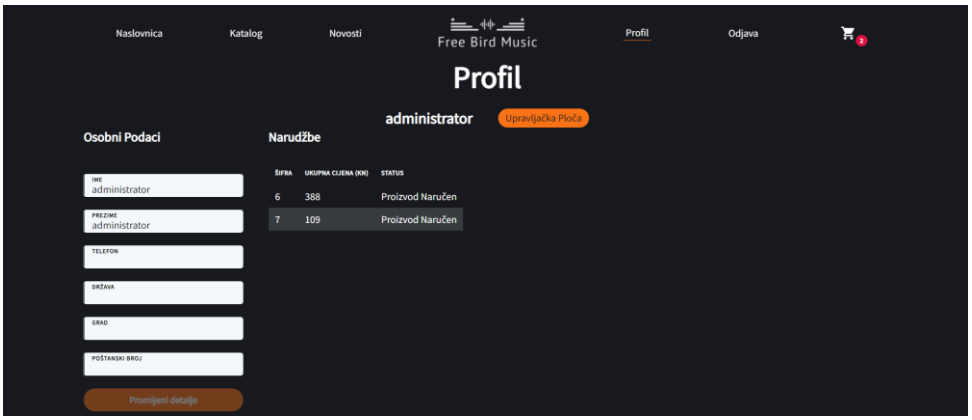
Slika 36. Web stranica za naplatu (https://freebird-music.vercel.app/checkout)

3.2.1.6. Profil

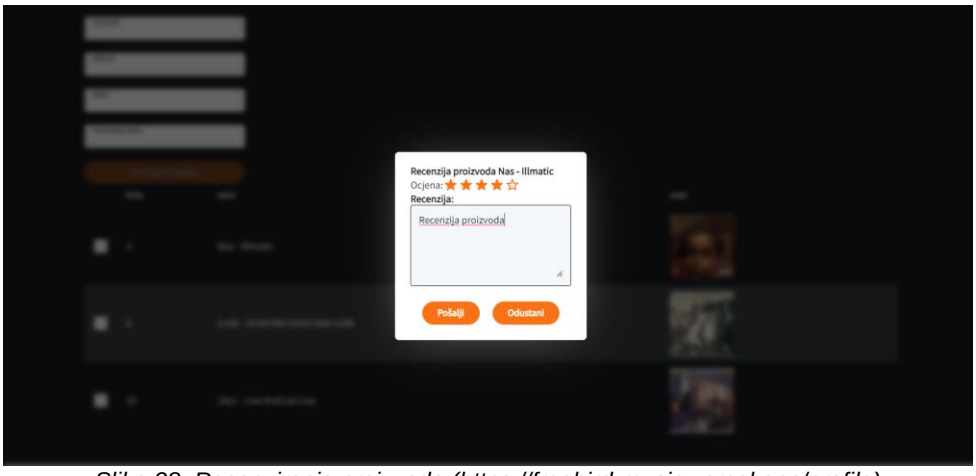
Na slici 36 je vidljivo da su već ispunjena polja vezana za ime, prezime i email adresu. Ova polja nisu ručno ispunjena nego su unaprijed ispunjena povlačenjem podataka unesenih na profilu registriranog korisnika. Ovo označava potpuno novu značajku web mjesta koja nije dostupna na trenutnoj verziji web mjesta. Nakon uspješne registracije, potvrde računa i prijave moguće je koristiti račun u kupovini proizvoda. Naravno, kupovina je dostupna i neregistriranim korisnicima ali oni neće imati neke prednosti koje imaju registrirani korisnici. Neke od tih prednosti su:

- Pamćenje sadržaja košarice na različitim uređajima
- Pamćenje sadržaja košarice nakon gašenja preglednika
- Pregled svih narudžbi
- Ocjenjivanje proizvoda

Pregled narudžbi i ocjenjivanje proizvoda je dostupno preko stranice profila koja je dostupna svim prijavljenim korisnicima.



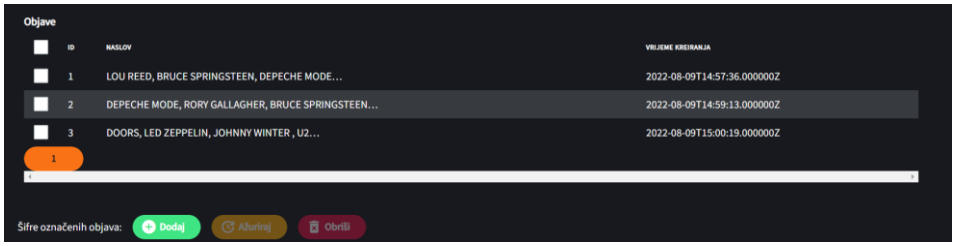
Slika 37. Pregled detalja profila (https://freebird-music.vercel.app/profile)



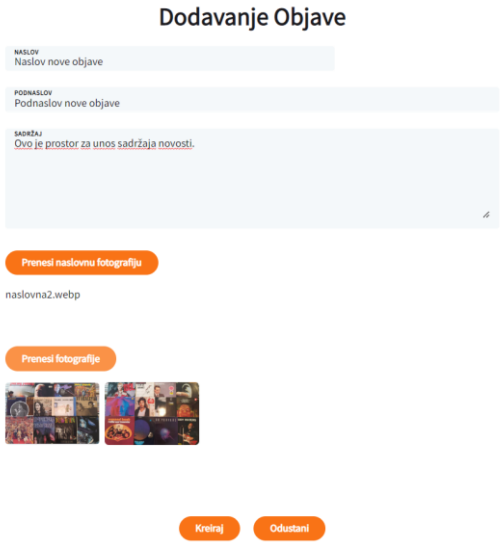
Slika 38. Recenziranje proizvoda (https://freebird-music.vercel.app/profile)

3.2.1.7. Upravljačka ploča

Pokraj korisničkom imena koje je prikazano na profilu, administratorima je prikazan gumb koji vodi na upravljačku ploču. Primjer toga je vidljiv na slici 37. Administratorska ploča služi za pregled podataka kao što su proizvodi, objave i pregled korisnika. Također su implementirane operacije brisanja korisničkih računa te kreiranje, ažuriranje i brisanje novosti i proizvoda. Na ovaj način je administratorima omogućeno lakše kontroliranje sadržaja koji je dostupan na web mjestu. Ostale sadržaje je moguće kontrolirati i putem eksternih alata kao što su Heidi SQL i MySQL Workbench.



Slika 39. Pregled dostupnih objava putem upravljačke ploče (https://freebird-music.vercel.app/dashboard)



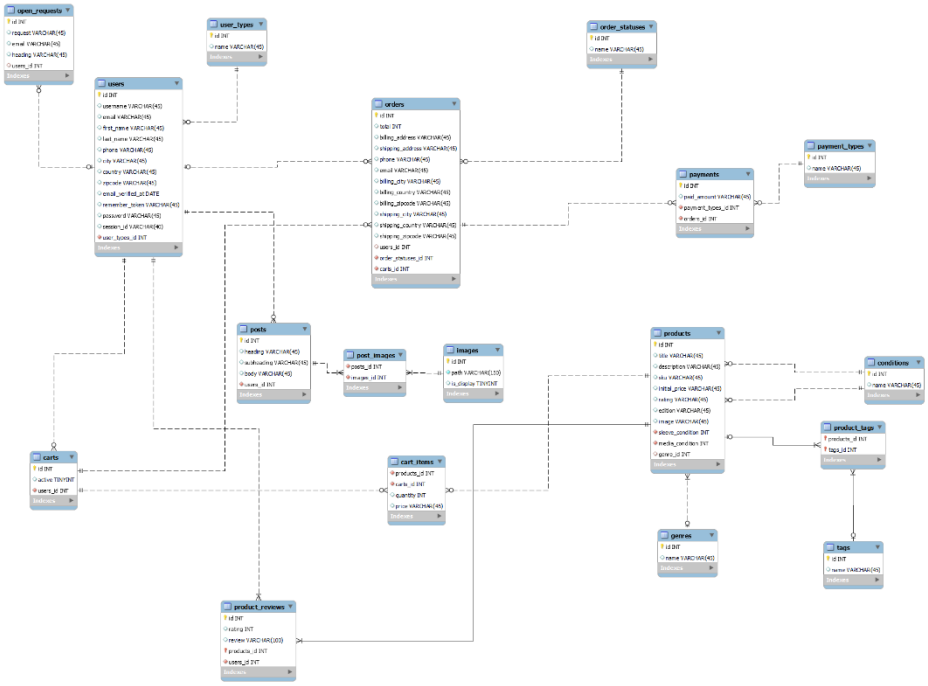
Slika 40. Obrazac za kreiranje objave (https://freebird-music.vercel.app/dashboard)

3.2.2. Izrada baze podataka

Baza podataka je ključan dio većine aplikacija pošto omogućavaju trajno spremanje podataka. Pojam baze podataka označava organiziranu kolekciju strukturiranih informacija, odnosno podataka [41]. Postoje brojni tipovi baza podataka, kao što su relacijske baze podataka, objektno orijentirane baze podataka, distribuirane baze podataka, NoSQL baze podataka i tako dalje [41]. Za izradu baze podataka u ovom završnom radu bit će korišten MySQL jezik te relacijska baza podataka. Relacijske baze podataka se ističu po konzistentnosti podataka, fleksibilnosti podataka i tome što su podaci međusobno povezani

ključevima [42]. Ova svojstva omogućavaju izradu baze podataka koja ima čisto definiranu strukturu, stabilna je te posjeduje visoke brzine dohvaćanja podataka.

Same tablice u bazi podataka će biti izrađene korištenjem Laravel migracija te će taj proces biti opisan u poglavlju 3.2.3., ali prije toga je potrebno izraditi pravilan model po kojem će biti moguće. Bit će potrebno izraditi ERA model, gdje naziv ERA označava entitete (engl. Entities), veze (engl. Relationships), i atribute (engl. Attributes). ERA model prikazuje sve entitete u obliku tablica te njihove atribute u obliku stupaca u tablici. Veze su ostvarene putem vanjskih ključeva te određuju kardinalnost odnosa između entiteta.



Slika 41. ERA model baze podataka (vlastita izrada)

Slika 41 prikazuje ERA model koji će biti korišten za izradu baze podataka u ovom završnom radu. ERA model se sastoji od 18 entiteta koji sadrže razni broj atributa. Entiteti su imenovani na engleskom jeziku jer je to konvencija koju bi trebalo pratiti prilikom korištenja alata Laravel. Radi boljeg shvaćanja zadatka sada ću opisati neke entitete i njihovu važnost.

Entitet koji je prikladan početku ovog opisa se nalazi u gornjem lijevom uglu modela te se naziva *Users,* odnosno *korisnici*. Entitet *Users* je dobar entitet za početak opisa jer ima vezu s čak 5 entiteta te samim tim predstavlja bitan dio ovog modela. Entitet *Users* sadrži veliki broj atributa, kao što su: korisničko ime, ime, prezime, broj telefona, adresa i tako dalje. Većina atributa predstavlja osobne podatke o korisniku koje će biti korišteni u procesu naplate, što je slušaj koji je ranije opisan. Također se nalaze neki drugi atributi *session_id*, *remember_token i email_verified_at* koji služe za općenito funkcioniranje web mjesta. Session_id služi za

identifikaciju korisnika po trenutnoj sesiji, remember_token je korišten za funkcionalnost pamćenja korisnika te email_verified_at služi za prepoznavanje potvrđenih računa.

User_type_id je vanjski ključ koji referencira tablicu *user_types,* odnosno *tipovi korisnika*. Vanjski ključ označava stupac ili više stupaca koji ostvaruju vezu između dvije tablica. Veza između tablica *user_types* i *users* je tipa jedan više naprema 0 ili više, to nam govori da svaki korisnik ima jedan tip, dok svaki tip korisnika može imati 0 ili više korisnika. Tablica *user_types* služi za identificiranje uloga korisnika te ima dva unosa, obični korisnik i administrator. Važnost ovoga je u tome što administrator ima ovlasti manipulirati sadržajem na web mjestu, dok obični korisnik to naravno ne može.

Tablica *users* također ima vezu i sa tablicom *open_requests*, odnosno tablicom odgovornom za otvorene upite. Otvoreni upitni su bitni za korisnike kako bi mogli slati zahtjeve za različite proizvode te nudili proizvode za otkup. Svakom zahtjevu je potrebno priložiti zaglavlje, tekst i email. Veza između tablica *users* i *open_requests* je nula ili jedan naprema 0 ili više. Razlog zašto upit ima 0 ili jedan korisnika je kako bi se omogućilo neregistriranim korisnicima slanje upita.

Još jednu vezu s tablicom *users* posjeduje tablica *posts*. Ova tablica sadrži sve novosti na web mjestu. Svaki unos se sastoji od naslova, podnaslova i teksta. Fotografije koje se nalaze u svakoj objavi nisu spremljene u bazu podataka nego su spremljene u vanjskom AWS S3 servisu i njihova poveznica je spremljena u bazu podataka. Poveznice su spremljene u entitetu *images,* te pošto svaka objava može imati više fotografija kreiran je slabi entitet *post_images* koji služi za povezivanje te dvije tablice.

Iduća tablica koja je bitna za rad je tablica *carts* koja predstavlja košarice korisnika. Ova tablica služi za spremanje košarica koje korisnik koristi kako bi se one mogle dohvatiti u slučaju da se korisnik odjavi tijekom kupovine, promijeni uređaj ili slično. Sastoji se od indikatora *active* koji nam govori ako je pojedina košarica aktivna te *users_id* što predstavlja vanjski ključ koji referencira tablicu *users*. Veza između tablica *carts* i *users* je jedan naprema 0 ili više, pošto svaka košarica mora imati jednog korisnika a svaki korisnik može ali ne mora imati više košarica. Razlog zašto korisnik može imati više košarica je u tome što se nakon završavanja narudžbe kreira nova košarica i označuje se kao „aktivna košarica.

Ključan entitet bez kojeg funkcioniranje ijednog web dućana ne bi bilo moguće je tablica *products*, odnosno tablica proizvoda. Sami atributi ovog entiteta se razlikuju od dućana do dućana ali njegova uloga i veze se mogu primijeniti na brojne vrste web dućana. Na ovom konkretnom primjeru atributi entiteta *products* su naslov, opis, SKU, početna cijena, stanje medija, stanje omota, ocjena, izdanje, putanja fotografije, te identifikacijski ključ žanra. Identifikacijski ključ žanra služi kao vanjski ključ za referenciranje žanra, dok stanje medija i

stanje omota služe kao vanjski ključ za referenciranje tablice *conditions*, odnosno stanja. Stanja medija i omota su implementirani kao ista tablica radi standardizacije oznaka stanja što uvelike pojednostavljuje cijeli sustav ocjenjivanja. Osim toga kreirani su entiteti *tags* za dodavanje oznaka proizvodima te slabi entitet *product_tags* kako bi se omogućilo povezivanje više proizvoda s više oznaka.

Nakon opisa tablice *products* moguće je stvoriti slabi entitet *cart_items* koji označava proizvode u košarici. Ovo je ključni entitet koji omogućava dodavanje proizvoda u košaricu te samim tim osnovnu funkcionalnost web dućana.

Uz entitet *carts* usko je povezan entitet *orders* koji predstavlja narudžbe u dućanu. Ovaj entitet je opisan pred kraj je povezuje sve ostale ključne entitete koji su do sada opisani u ovom završnom radu. Entitet *orders* predstavlja klimaks cijelog procesa korištenja web dućana koji se sastoji od prijave korisnika, pregleda proizvoda, dodavanje proizvoda košarice te na kraju dovršavanje narudžbe. Iz tog razloga je neizmjerno bitno pravilno dizajnirati ovaj entitet kako bi imao dobre performanse te bio koristan korisnicima, ali i administratorima, web mjesta. *Orders* sadrži brojne atribute, među kojima su ukupna cijena, broj telefona, email te podaci kao adresa, grad, država i poštanski broj koji imaju svoje inačice za naplatu i dostavu. Osim toga moguće je pronaći vanjske ključeve za korisnika, košaricu i status narudžbe koji povezuju narudžbu s ostalim ključnim dijelovima baze podataka. Vanjski ključ za korisnika je jedini vanjski ključ ovog entiteta koji nije obavezan, pošto postoji uvjet da neregistrirani korisnici mogu kupovati proizvode.

Nakon toga je potrebno kreirati entitet *payments* koji predstavlja plaćanja pojedinih narudžbi. Povezuje se na tablicu *orders* pripadajućim vanjskim ključem te mu entitet *payment_types* opisuje tip uplate, pošto postoje različiti načini plaćanja. Osim toga dostupan je atribut paid_amount koji daje informaciju o plaćenom iznosu pojedine narudžbe.

3.2.3. Izrada servisa na strani poslužitelja

Uslijed dizajniranja baze podataka potrebno je izraditi servis na strani poslužitelja koji će web dućan koristiti. Uz bazu podataka strana poslužitelja postavlja temelj web aplikacije koji će omogućiti dizajnu i sučelju klijentske strane da zablistaju i omoguće web aplikaciji da ostvari svoj potencijal. Servis na strani poslužitelja ove web aplikacije biti zadužen biti odgovoran za rukovanje zahtjevima, obradom podataka i isporukom potrebnih podataka web aplikaciji na brz, efikasan i pouzdan način.

3.2.3.1. RESTful API

Za vrijeme opisa alata Laravel naveden je isječak iz dokumentacije koji govori da je moguće Laravel koristiti na dva načina: u svrhu izrade aplikacije skupom programskih okvira ili RESTful API-a [27]. Također je navedeno kako će u radu strana poslužitelja biti izrađena u obliku API-a te su opisani razlozi iza te odluke. Sada je potrebno opisati što točno je API, što nam omogućava te što je to RESTful arhitektura.

API predstavlja pravila koja omogućavaju komunikaciju s određenim softverskim sustavom [43]. API služi kao poveznica između korisnika i resursa kojima korisnici žele pristupiti. Prilikom izrade API-ja otkrivaju se određene putanje koje korisniku pružaju resurse koje je zatražio [43]. Uslijed primanja resursa korisnik može koristiti te podatke u različite svrhe.

RESTful ili REST označava softversku arhitekturu koja predstavlja strukturu i određuje kako bi API trebao funkcionirati. REST arhitektura se ističe svojom pouzdanošću i visokim performansama te je iznimno popularan način omogućavanja komunikacije. Osim toga REST aplikacije je lako modificirati te su zbog toga iznimno fleksibilne i skalabilne [43].

REST se sastoji od više smjernica koje pomažu programerima da izradi brz, pouzdan i skalabilan API koji će biti korišten u određenoj web aplikaciji. Spomenute smjernice predstavljaju principe rada unutar REST arhitekture [43].

Jedan od tih principa je kreiranje jednoobraznog sučelja (engl. uniform interface). Ovo je fundamentalni pristup i omogućava standardiziran način dohvaćanja podataka. Jednoobrazno sučelje nameće 4 ograničenja [43] :

1. Zahtjevi identificiraju resurse pomoću jednoobraznih identifikatora resursa (URI)
2. Klijenti imaju dovoljno informacija u slučaju da žele modificirati ili izbrisati resurs
3. Klijenti u odgovoru primaju dovoljno informacija za daljnje procesiranje podataka
4. Klijenti dobivaju sve ostale informacije o povezanim resursima koji su potrebni za dovršavanje zadatka

Također je bitno ostvariti „statelessness“, odnosno zahtjevi ne smiju imati stanje. To znači da bi klijent u pravilu trebao moći slati zahtjeve kojim god redoslijedom želi ali će rezultat uvijek biti isti. Svaki zahtjev treba biti izoliran jedan od drugog te bi poslužitelj trebao razumjeti i ispuniti zahtjev svaki put [43].

Još jedna bitna stavka koja karakterizira REST način rada je mogućnosti predmemoriranja podataka. Predmemoriranje podataka omogućava brz rad i bolje korisničko iskustvo pošto bitno skraćuje vrijeme obrade zahtjeva. Dohvaćanje predmemoriranih podataka

je iznimno moćno svojstvo zbog visoke brzine predmemorije i pravilno rukovanje predmemorijom može imati veliki utjecaj na rad i performanse aplikacije.

U narednim poglavljima bit će opisano korištenje modela, kontrolera i migracija u alatu Laravel. Zbog velike količine koda neće biti pokriveni svi modeli, kontroleri i migracije nego ću za primjer uzeti sve što je bilo potrebno za izradu novosti koje su korištene na web mjestu. Cijeli ostali programski kod je dostupan na Github repozitoriju: https://github.com/antonlovric/freebird-music

3.2.3.2. Modeli

U poglavlju 2.5.1. je već pojašnjen pojam modela i njegova važnost u MVC arhitekturi. Korištenje modela u radu aplikacije je omogućeno korištenjem alata Eloquent. Eloquent je ORM (engl. object-relational mapper) koji omogućava laku interakciju s bazom podataka [44]. Eloquent modeli omogućavaju razne operacije koje pomažu prilikom obrada podataka u bazi podataka te su se prilikom izrade ovog završnog rada prikazali kao neizmjerno koristan alat.

Izrada samog modela je poprilično jednostavna. Potrebno je kreirati polje naziva *fillable* koje sadrži sve atribute koje je moguće ručno ispuniti te metode koje su korištene za dohvaćanje svih elemenata koji su povezani vanjskim ključevima.

```php
class Post extends Model
{
    use HasFactory;
    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'heading',
        'subheading',
        'body',
        'user_id',
    ];

    public function images() {
        return $this->hasMany(PostImage::class, "post_id", "id");
    }

    public function user() {
        return $this->belongsTo(User::class, "user_id", "id");
    }
}
```

Slika 42. Isječak koda korišten za kreiranje modela Post (vlastita izrada)

U polje naziva *fillable* je potrebno staviti atribute koji su već navedeni u poglavlju 3.2.2., odnosno naslov, podnaslov, sadržaj te identifikator korisnika. Osim toga slika 42 prikazuje i metode naziva *images* i *user* koje služe za dohvaćanje fotografija objave i korisnika koji je kreirao objavu.

Veze između Eloquent modela su prikazane ranije spomenutim metodama. Ranije je spomenuto kako svaka objava ima više fotografija te je zbog toga potrebno kreirati metodu za to i pratiti traženu Laravel konvenciju imenovanja. Pošto objava ima više fotografija, naziv metode treba biti u množini, odnosno *images*. Unutar ove metode poziva se *hasMany* metoda koja govori da ovaj model „ima više“ instanci nekog resursa. Metodi *hasMany* je potrebno proslijediti više argumenata. Prvo se prosljeđuje klasa modela koji je referenciran u ovoj vezi, zatim naziv vanjskog ključa u referenciranoj tablici te naziv identifikacijskog ključa modela *Post*. Na sličan način je kreirana i metoda *user*, samo što ona sadrži jednu instancu resursa *User* te se osim imenovanja metode u jednini koristi metoda *belongsTo* umjesto metode *hasMany*.

3.2.3.3. Kontroleri

Nakon kreiranja modela moguće je nastaviti s kreiranjem kontrolera. Kontroleri sadrže sve metode koje omogućavaju izravan rad s bazom podataka. Kontroler za objave je kreiran idućom naredbom:

```
php artisan make:controller PostController --api
```

Zastavica –api automatski generira metode *index, store, show, update* i *destroy*. Metoda *index* dohvaća sve instance određenog resursa, *store* služi za kreiranje novog resursa, *show* služi za prikaz traženog resursa, *update* služi za ažuriranje traženog resursa te *destroy* služi za brisanje traženog resursa. Ovo su osnovne metode za CRUD operacije te služe kao dobra nit vodilja za implementiranje ostatka kontrolera. Uz ove metode kreirane su dodatne metode *recentPosts* koja služi za dohvaćanje posljednje 3 objave te *destroyPosts* koje omogućava brisanje više objava.

```php
/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function index(Request $request)
{
    $pageSize = $request->page_size ?? 10;
    $requestHeading = $request->query("heading");

    return Post::with("images")->when($requestHeading, function($query, $heading) {
        $query->where("heading", "LIKE", "%" . $heading . "%");
    })->paginate($pageSize);
}
```

Slika 43. Index metoda u kontroleru objava (vlastita izrada)

Slika 43 prikazuje *index* metodu u kontroleru koji služi za upravljanje objavama. Na strani poslužitelja je implementirano straničenje radi efikasnosti rada. Zbog toga je na početku potrebno odrediti broj elemenata na svakoj stranici, što označava varijabla *$pageSize.* Ova varijabla poprima vrijednost *page_size* parametra ili, ako on nije proslijeđen, poprima

vrijednost 10. Također je implementirana mogućnost pretraživanja po naslovu objave. Vrijednost po kojoj se vrši pretraga se sprema u *$requestHeading* varijablu. U nastavku se vrši dohvaćanje objave po proslijeđenim parametrima. Kako bismo dohvatili poveznice fotografija objave u upitu je potrebno uključiti i veze modela. Ovo se radi metodom *with* te joj se kao argument prosljeđuje naziv veze koji je potrebno uključiti [45]. Na dobivenom se poziva metoda *when* koja omogućava izvršavanje takozvane *callback* funkcije ako je ispunjen određeni uvjet. Unutar *callback* funkcije je napisan upit koji dohvaća objave koje sadrže traženi naslov. Na samom kraju se poziva *paginate* metoda koja vrši straničenje te joj se kao argument prosljeđuje broj elemenata po stranici [46]. Može se reći da ovo zvuči poprilično komplicirano ali slika 43 ukazuje na to da je rješenje čak poprilično elegantno uzimajući u obzir sve uvjete ispunjene. Ovo je samo jedan primjer kako Laravel omogućava elegantna rješenja svojom MVC arhitekturom.

```php
/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $request->validate([
        "heading" => "required|string",
        "body" => "required|string",
        "session_id" => "required|exists:users,session_id"
    ]);

    $user_id = User::query()->where("session_id", "=", $request["session_id"])->first("id")["id"];
    $request->request->remove("session_id");
    $request->request->add(["user_id" => $user_id]);

    return Post::create($request->all());
}
```

Slika 44. Metoda store u kontroleru objava (vlastita izrada)

Pošto metoda *store* služi za kreiranje novih objava ona ključna metoda za ovaj kontroler. Prije nego što se započne bilo što vezano za kreiranje objave potrebno je validirati sve proslijeđene parametre. Ovo može biti poprilično nezgodan zadatak ali Laravel pruža jednostavan način za validaciju podataka. Metoda *store* prima parametar $*request* koji sadrži sve podatke o zahtjevu za kreiranje objave. Na *$request* parametru poziva se *validate* funkcija koja služi za validaciju. Ona prima asocijativno polje kao argument koje sadrži sva potrebna pravila za validaciju [47]. Slika 44 prikazuje sva pravila, a to su da su parametri *heading* i *body* obavezni i tipa su *string te da je parametar session_id* obavezan te postoji u tablici *users* u stupcu *session_id*.

Nakon toga je potrebno dohvatiti identifikacijski ključ korisnika. Zbog sigurnosnih razloga klijent nikad ne dolazi do identifikacijskog ključa korisnika nego ima pristup samo identifikatoru sesije koji se obnavlja kad god započne nova sesija. Putem identifikatora sesije se dohvaća identifikacijski ključ korisnika i sprema se u varijablu *$user_id*. Potom se identifikator sesije uklanja iz zahtjeva i dodaje se identifikator korisnika. Ovo se može činiti

kontraintuitivno ali omogućava lakše kreiranje objave pozivanjem metode *create* na *Post* model. Ovoj metodi se prosljeđuju svi parametri u zahtjevu kao argument putem pozivanja metode *all* na varijabli *$request.*

```php
/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    return Post::with("images")->find($id);
}
```

Slika 45. Metoda show kontrolera objave (vlastita izrada)

Metoda *show* ima poprilično jednostavnu svrhu te je samim tim jednostavno i napisana. Sastoji se od samo jedne linije koda, poziva se *with* metoda na *Post* model te se na rezultatu te operacije poziva funkcija *find* koja vraća resurs koji odgovara identifikatoru koji joj je proslijeđen kao argument [44].

```php
/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    return Post::where("id", $id)->update($request->all());
}
```

Slika 46. Metoda update kontrolera objave (vlastita izrada)

Metoda *update* je jednostavna kao i metoda *show*, samo što se u metodi *update* na modelu *Post* poziva metoda *where* kako bi se došlo do tražene objave te se metodi *update* prosljeđuju svi parametri u zahtjevu putem metode *all*.

```php
/**
 * Remove the specified resource from storage.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    return Post::destroy($id);
}
```

Slika 47. Metoda destroy kontrolera objava (vlastita izrada)

Brisanje resursa je također iznimno lagano korištenjem metoda Eloquent modela. Slika 47 to savršeno pokazuje, pošto je dovoljno samo proslijediti identifikator objave *destroy* metodi koja se poziva na modelu *Post.*

```php
/**
 * Get last 3 posts
 *
 * @param  string  $title
 * @return \Illuminate\Http\Response
 */
public function recentPosts()
{
    return Post::with("images")->orderBy("id", "desc")->take(3)->get();
}
```

Slika 48. Metoda recentPosts kontrolera objava (vlastita izrada)

Metoda *recentPosts* služi za dohvaćanje 3 najnovije objave. Ova funkcionalnost je potrebna na naslovnoj stranici kako bi se korisnicima mogle prikazati najnovije vijesti vezane za web mjesto. Ova funkcionalnost je implementirana na način da se metodom *with* vrate slike objave, metodom *orderBy* se sortiraju silazno prema identifikatoru objava pošto se identifikatori slijedno dodjeljuju novostima prilikom kreiranja unosa. *Take* metodom se dohvaćaju 3 rezultata te se koristi *get* metoda za vraćanje rezultata u obliku polja [48].

```php
/**
 * Remove Multiple Posts.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function destroyPosts(Request $request)
{
    $ids = $request->ids;
    return ["response" => Post::whereIn("id",$ids)->delete(), "status" => 200];
}
```

Slika 49. Metoda destroyPosts kontrolera objava (vlastita izrada)

Metoda *destroyPosts* je izrazito slična metodi *destroy* samo što služi za istovremeno brisanje većeg broja objava. Ovo je ostvareno pozivanjem metode *whereIn* na modelu *Post* te pozivanjem *delete* metode na dobivenom rezultatu. Osim odgovora vraća se i status kod 200 koji govori klijentu da je operacija uspješno izvršena.

3.2.3.4. Migracije

Migracije su još jedan neizmjerno bitan koncept koji je korišten u izradi ove Laravel aplikacije. Migracije su korištene za izradu same baze podataka. Migracijama je iznimno lako kreirati tablice, dodavati, ažurirati i brisati atribute te dodavati svakakva ograničenja. Pošto su migracije implementirane kao način verzioniranja sustava one pružaju standardizirani način ažuriranja baze podataka, te je ovo jedna od glavnih prednosti migracija koja je spomenuta u službenoj Laravel dokumentaciji [29]. Svaka migracija se sastoji od svoje *up* i *down* metode.

Up metoda se pokreće prilikom pokretanja migracije, dok se *down* metoda pokreće nakon poništavanja migracije [29].

```php
/**
 * Run the migrations.
 *
 * @return void
 */
public function up()
{
    Schema::create('posts', function (Blueprint $table) {
        $table->id();
        $table->string("heading", 50);
        $table->string("subheading", 40);
        $table->string("body", 350);
        $table->foreignId("user_id")->constrained()->onDelete("cascade");
        $table->timestamps();
    });
}
```

Slika 50 „up“ metoda migracije za izradu tablice „posts" (vlastita izrada)

Up metoda prikazana slikom 50 daje jako dobar dojam o tome kako je lako kreirati tablice baze podataka koristeći Laravel migracije. Svaki stupac je moguće kreirati jednom linijom koda. Metoda *id* služi za kreiranje identifikacijskog ključa trenutne tablice. Dodjeljuje mu naziv „id“, postavlja ga kao primarni ključ i omogućava mu automatsko inkrementiranje prilikom kreiranja novog zapisa. *String* metoda služi za kreiranje stupca tipa varchar te prima više argumenata. Prvi argument označava naziv stupca, drugi argument označava duljinu teksta dok treći argument označava zadani tekst [29]. Metoda *foreignId* omogućava stvaranje stupca koji predstavlja vanjski ključ. Argument navedene funkcije predstavlja naziv stupca koji će biti povezan vanjskim ključem, pozivanje metode *constrained* služi za određivanje referencirane tablice. Metoda *onDelete* služi za kaskadno brisanje povezanih elemenata [29]. Naposljetku se vidi metoda *timestamps* koja služi za kreiranje stupaca koji označavaju vrijeme stvaranja i ažuriranja tablice [29].

3.2.3.5. API krajnje točke

Kako bi se omogućilo korištenje metoda kontrolera a samim time i rad s bazom podataka potrebno je otkriti API krajnje točke. Ove krajnje točke predstavljaju URL na koji je potrebno poslati zahtjev koji će poslužitelj kasnije obraditi. Kreiranje API krajnjih točaka se izvršava u datoteci *api.php* koja bude automatski kreirana prilikom inicijalizacije projekta. Krajnje točke u ovoj datoteci su nezaštićene te im svatko može pristupiti. Također je moguće zaštiti rute kreiranjem middleware-a i postavljanjem određenih uvjeta.

```php
49  Route::controller(PostController::class)->group(function () {
50      Route::get("/posts", "index");
51      Route::get("/posts/latest", "recentPosts");
52      Route::get("/posts/{id}", "show");
53  });
```

Slika 51. Nezaštićene API krajnje točke za objave (vlastita izrada)

Slika 51 predstavlja 3 krajnje točke koje primaju GET zahtjeve. Svaka krajnja točka je grupirana po svom kontroleru, odnosno vidljivo je kako su na slici 51 krajnje točke grupirane po kontroleru PostController. Krajnje točke se grupiraju po kontroleru pozivanjem metode *controller* na modelu *Route* koji je kreiran za vrijeme inicijalizacije projekta. Metodi *controller* je potrebno proslijediti klasu kontrolera po kojem želimo grupirati rute i pozvati metodu *group* na rezultatu. Metodi *group* je potrebno proslijediti callback funkciju kao argument koje će sadržavati sve krajnje točke koje želimo grupirati. Same krajnje točke se kreiraju tako da na Route modelu pozovemo metodu koja označava kakvog tipa će biti krajnja točka: *get*, *post*, *put*, *patch*, *delete* i tako dalje. Odabrana metoda se poziva te joj se kao prvi argument se šalje tekst krajnje točke, a kao drugi element navodi funkcija iz kontrolera koja će biti izvršena. Važno je zapamtiti kako se krajnja točka nadovezuje na adresu poslužitelja. Pošto je adresa poslužitelja https://anton-freebird.com/ može se zaključiti kako na slici 51 prva ruta glasi https://anton-freebird.com/api/posts i posjećivanjem te rute se poziva metoda index koja je ranije opisana.

Isto tako je moguće zaštiti željene rute. Štićenje rute se izvodi pozivanjem *group* metode na *Route* modelu i prosljeđivanjem asocijativnog polja koje referencira middleware za štićenje rute i callback funkciju koja grupira željene rute. Primjer štićenja ruta je može vidjeti na slici 52.

```php
//Protected routes
Route::group(["middleware" => ["auth:sanctum"]], function () {

    Route::controller(PostController::class)->group(function() {
        Route::post("/posts", "store");
        Route::put("/posts/{id}", "update");
        Route::delete("/posts/{id}", "destroy");
        Route::post("/posts/deletePosts", "destroyPosts");
    });
```

Slika 52. Zaštićene krajnje točke (vlastita izrada)

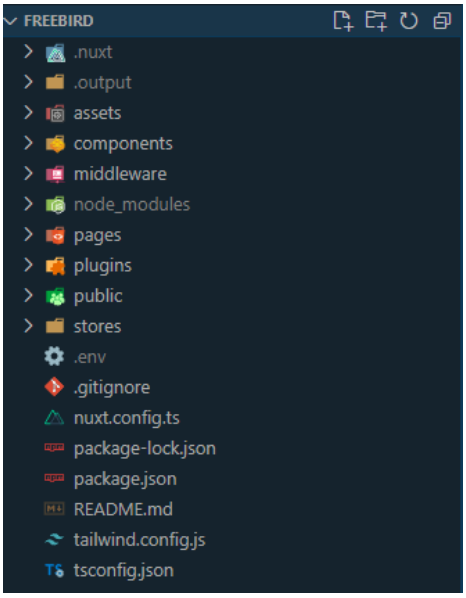
3.2.4. Izrada sučelja klijentske strane

Korištenje JavaScript okvira za rad se pokazalo kao neizmjerno važna odluka u procesu izrade ovog završnog rada. Kreiranje ovakvog projekta uz korištenje čistog JavaScripta bi bio ogroman pothvat i rezultat bi vjerojatno znatno manje kvalitete. Glavne značajke Nuxt-a u izradi ovog rada koje su pripomogle u izradi ovog rada su:

- Podjela programskog koda po komponentama
- Poticanje korištenja pravilne organizacije datoteka
- Lako korištenje Vuestic UI okvira za rad
- Olakšana integracija TailwindCSS okvira za rad

- Unaprijed kreirani servisi za dohvaćanje podataka i manipuliranje kolačićima
- Laka integracija s alatom Pinia za korištenje spremišta
- Renderiranje na strani poslužitelja

Primjer dobre organizacije datoteka koju svaki Nuxt projekt zahtijeva je podjela komponenata, stranca, middleware-a i Pinia spremišta u odvojene mape. Na ovaj način se iznimno lako snaći u projektu te je njegovo održavanje uvelike olakšano. Primjer organizacije datoteka je prikazan na slici 53. Mape nisu otkrivene zbog velikog broja komponenti ali se može dobiti dobar dojam o tome kako je potrebno organizirati Nuxt projekt.



Slika 53. Organizacijska struktura datoteka u Nuxt 3 projektu (vlastita izrada)

Sukladno demonstraciji rada u Laravel okviru za rad, u demonstraciji rada koristeći Nuxt 3 okvir za rad bit će prikazane komponente korištene za prikaz novosti.

Za vrijeme opisa rada u Nuxt 3 i Vue 3 okruženju napomenuo sam kako je potrebno podijeliti kod u dvije veće oznake, *template* i *script*. *Template* oznaka predstavlja sve komponente koje će biti renderirane na ekran. Tu je također moguće prosljeđivanje svojstava komponentama kako bi ih one mogle interno koristiti.

```vue
<template>
    <div>
        <the-header />
        <posts-overview
            :key="posts"
            :pending="pending"
            :currentPage="posts?.current_page"
            :totalPages="posts?.last_page"
            :posts="posts?.data"
            :searchQuery="page?.query"
            @change_page="handlePageChange"
            @search="handleSearch"
        />
        <the-footer />
    </div>
</template>
```

Slika 54. Template kod stranice za novosti (vlastita izrada)

Slika 54 prikazuje od čega se sastoji template oznaka korištena za implementaciju stranice novosti. *Template* oznaka smije imati samo jedno dijete, te su zbog toga sve komponente omotane *div* oznakom. Unutar te oznake se nalaze komponente za zaglavlje, pregled novosti i podnožje. Odvajanje koda u 3 različite komponente stvara mogućnost ponovnog korištenja koda i olakšavanje održivosti koda zbog njegove urednosti. Komponenta za pregled novosti je imenovana *posts-overview* i prosljeđuju joj se brojni atributi kao što su trenutna stranica, ukupni broj stranica, novosti i tako dalje. Stavljanjem dvotočke ispred naziva atributa omogućeno je prosljeđivanje vrijednosti varijable umjesto teksta, dok se stavljanjem znaka @ ispred naziva atributa deklarira slušatelj događaja. Slušatelj događaja označava funkciju koja će se okinuti uslijed izvršavanja pojedine akcije. Tako su u ovom konkretnom primjeru korišteni slušatelji događaja za promjenu stranice i pretragu novosti.

```vue
<script setup>
const page = reactive({ currentPage: 1, query: '' });
const config = useRuntimeConfig();

const {
    data: posts,
    pending,
    refresh,
} = useLazyAsyncData('posts', () =>
    $fetch(`${config.API_BASE_URL}/posts`, {
        params: {
            page: page.currentPage,
            page_size: 10,
            heading: page.query,
        },
    })
);

const handlePageChange = async ({ newPage } = args) => {
    page.currentPage = newPage;
    await refresh();
};

const handleSearch = async ({ query } = args) => {
    page.query = query;
    await refresh();
};

useHead({
    title: 'Novosti',
});
</script>
```

Slika 55. Script oznaka stranice novosti (vlastita izrada)

Script oznaka obuhvaća logiku i funkcionalnost komponente, odnosno sve što je implementirano putem JavaScripta. Na samom početku se kreira *reactive* objekt koji služi za stvaranje objekta s reaktivnom vrijednošću. Odnosno *reactive* sadrži objekt čija će vrijednost biti promijenjena te će ta promjena odmah biti vidljiva [49]. Stvoren je *reactive* objekt naziva *page* te sadrži svojstva *current_page* i *query* koji označavaju trenutnu stranicu prikaza novosti i upit za pretraživanje novosti.

Nakon toga slijedi iznimno bitan isječak koda koji služi za dohvaćanje podataka s REST API-ja koji je ranije kreiran. Ovo je bitan isječak jer napokon vidimo spajanje dijela strane klijenta sa servisom dijela poslužitelja projekta. Dohvaćanje podataka se izvršava pomoću *useLazyAsyncData* kuke koja predstavlja jedan on podržanih načina dohvaćanja podataka. Ključna riječ *lazy* označuje da proces dohvaćanja podataka ne blokira navigaciju na stranicu, odnosno da će se podaci dohvaćati paralelno s učitavanjem stranice. Ako se stranica učita ranije nego što podaci budu dohvaćeni, *pending* svojstvo rezultata ove metode će biti istinit i koristi se za prikaz indikatora za učitavanje. Na ovaj način se poboljšava korisničko iskustvo pošto korisnik zna da se ništa nije pokvarilo s aplikacijom nego samo mora pričekati dohvaćanje podataka. Prvi argument ove kuke predstavlja ključ po kojem se može vršiti ponovno dohvaćanje podataka te samu funkciju za dohvaćanje podataka. Korištenjem *$fetch* funkcije vrši se samo dohvaćanje podataka. Prvi argument ove funkcije predstavlja rutu koju treba pozvati kako bi se okinula željena funkcija na strani poslužitelja. U ovom primjeru je vidljiva uporaba *config* objekta koji služi za pristup varijablama okoline. Varijable okoline služi za korištenje vrijednosti koje ovise o okolini u kojoj se koristi aplikacija. Glavna uporaba varijabli okoline u ovom projektu je promjena rute za pristup API-ju, pošto je drukčija adresa lokalnog poslužitelja i poslužitelja na službenoj verziji aplikacije. Drugi argument ove funkcije predstavlja objekt koji sadrži opcije zahtjeva. Postoji veliki broj opcija kao što su parametri, tijelo zahtjeva, tip metode, zaglavlja i tako dalje [50]. U ovom konkretnom primjeru su postavljeni svi potrebni parametri za modifikaciju zahtjeva, odnosno broj stranice, broj elemenata po stranici i naslov za pretraživanje. Dohvaćeni podaci se nalaze u destrukturiranoj varijabli *posts*, *pending* označava status dohvaćanja podataka i *refresh* označava funkciju za osvježivanje zahtjeva.

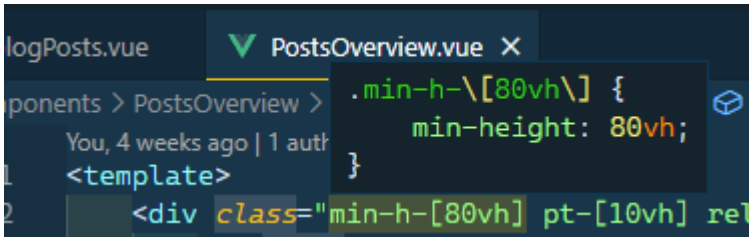
Osim toga dostupne su dvije funkcije: *handlePageChange* i *handleSearch* koje služe kao rukovatelji za događaje koje *posts-overview* komponenta emitira svom roditelju. Funkcija *handlePageChange* se okida na svaku promjenu stranice dok se *handleSearch* okida prilikom pretraživanja novosti. Obje funkcije rade na istom principu, ažuriraju *reactive* objekt i zatim osvježavaju zahtjev s ažuriranim vrijednostima.

Na samom kraju ove *script* oznake korištena je *useHead* kuka koja služi za postavljanje meta podataka vezanih za ovu stranicu. Ova kuka omogućava postavljanje raznih podataka kao što su naslov stranice, set znakova, *viewport* vrijednost i tako dalje [51].

```
<template>
    <div class="min-h-[80vh] pt-[10vh] relative sm:w-9/12 w-10/12 mx-auto">
        <h1 class="text-3xl mb-4 sm:mb-0 sm:text-5xl sm:mt-4 text-center">Novosti</h1>
        <div class="inline-flex mt-7 flex-col">
            <div class="inline-flex gap-4">
                <va-input label="Pretraživanje" class="w-[40ch]" v-model="input.searchQuery" />
                <va-button @click="searchHandler" type="submit" color="#f97316" text-color="#fff"
                    >Pretraga</va-button
                >
            </div>
            <va-inner-loading :loading="props.pending">
                <div class="mt-8 grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-8">
                    <post-component
                        v-for="(post, index) in posts.postCollection"
                        :key="index"
                        :image="getDisplay(post.images)"
                        :postTitle="post.heading"
                        :postDescription="post.subheading"
                        :postId="post.id"
                    />
                </div>
            </va-inner-loading>
            <va-pagination
                :pages="totalPages"
                v-model="input.pageNumber"
                @update:model-value="handlePageChange"
                class="col-span-1 my-8 sm:col-span-2 md:col-span-3"
            />
        </div>
    </div>
</template>
```

Slika 56. Template oznaka komponente posts-overview (vlastita izrada)

Template oznaka komponente *posts-overview* predstavlja nešto složeniji kod. Zbog svih Tailwind CSS klasa koje su primijenjene na ove elemente stvara se osjećaj neurednog koda kojeg je teško čitati. Ovo je nedostatak Tailwind CSS okvira za rad ali se ipak pokazao bitnim alatom u izradi ovog projekta. Na prvu Tailwind CSS klase mogu izgledati zastrašujuće ali prijelazom miša preko njih vidljivo je objašnjenje što točno koja klasa radi, kao što je prikazano na slici 57. Rad s Tailwind CSS okvirom je jako brz i olakšan je iznimno preciznom, urednom i kvalitetnom dokumentacijom koja sadrži sve klase i njihovo objašnjenje.



Slika 57. Prikaz detalja Tailwind CSS klase (vlastita izrada)

Analizom *template* oznake komponente *posts-overview* je vidljivo da imamo *div element* koji služi kao pretinac koji obuhvaća sve ostale elemente od kojih je izrađena ova komponenta. Tekst unutar komponenti znatno olakšava razumijevanje koda, stoga je ne početku vidljiv naslov *Novosti* i pretinac koji sadrži polje za pretraživanje novosti. U pretincu za pretraživanje novosti vidimo prvi uporabu Vuestic UI okvira za rad, komponente *va-input* i *va-button*. Sve Vuestic komponente imaju prefiks „*va*“ pa ih je lakše prepoznati unutar koda. Svojstvo koje ranije nije bilo prikazano je *v-model* i to je ključno svojstvo u Vue sustavu. *V-model* direktiva veže vrijednost komponente unutar *template* oznake i vrijednost varijable iz *script* oznake [52]. Ovo je način praćenja vrijednosti *template* oznake i kreiranja logike na temelju te vrijednosti. U nastavku koda vidljiva je uporaba komponente *va-inner-loading* komponente koja služi za prikaz ikone za učitavanje za vrijeme dohvaćanja elemenata.

Prosljeđuje joj se *pending* vrijednost koju je komponenta primila iz roditelja i ovisno o toj vrijednosti se prikazuje ikona učitavanja ili *post-component* komponenta.

Prilikom prikaza *post-component* komponente vidljiva je još jedna prednost Vue okvira za rad, a to je renderiranje elemenata unutar petlji. *V-for* vrijednost omogućava renderiranje komponenti na temelju polja elemenata, konkretno na ovom primjeru se renderiraju elementi unutar polja *postCollection* [53]. Ovo omogućava prosljeđivanje atributa svakom pojedinačnom elementu, povećanu urednost koda i skalabilnost koda pošto nije potrebno ručno dodavati komponente uslijed kreiranja nove objave. Svakom elementu unutar *v-for* petlje je potrebno dodijeliti jedinstveni ključ atributom *:key* kako bi se mogli međusobno razaznati. Na kraju je vidljiva još *va-pagination* komponenta koja služi za prikazivanje izbornika za straničenje.

```
<script setup>
const config = useRuntimeConfig();
const props = defineProps({
    posts: [],
    currentPage: Number,
    totalPages: Number,
    pending: Boolean,
    searchQuery: String,
});
const input = reactive({ searchQuery: props.searchQuery, pageNumber: props.currentPage });

const emits = defineEmits(['change_page', 'search']);

const posts = reactive({ postCollection: props.posts });

const getDisplay = (images) => images?.find((image) => image.is_display == 1)?.url;

const searchHandler = () => {
    emits('search', { query: input.searchQuery });
};

const handlePageChange = () => {
    emits('change_page', { newPage: input.pageNumber });
};
</script>
```

Slika 58. Script oznaka komponente posts-overview (vlastita izrada)

Slikom 58 je prikazan sadržaj *script* oznaka komponente *posts-overview* i većina stvari koji se nalaze unutar nje su stvari koje su ranije prikazane. Jedna od stvari koja nije ranije prikazana je korištenje *definePropsfunkcije*. Korištenjem funkcije *definePropsmoguće* određivanje atributa koji će biti proslijeđeni komponenti te je moguće i povezati tip podataka atributa [54]. Povezivanje tipova podataka je korisno kako bi se ograničile vrijednosti koje se mogu proslijediti i samim tim je jednostavnije rukovanje tim atributima.

Još jedna funkcija koja ranije nije prikazana je *define*Emits funkcija. Ona omogućava definiranje svih događaja koje će komponenta moći emitirati roditelju [54]. Emitiranje se vrši korištenjem funkcije *emits* koja je korištena unutar funkcija *searchHandler* i *handlePageChange*. Prvi argument funkcije *emits* označava naziv događaja koji je ranije definiran dok drugi element predstavlja argument proslijeđen uz emitirani događaj.

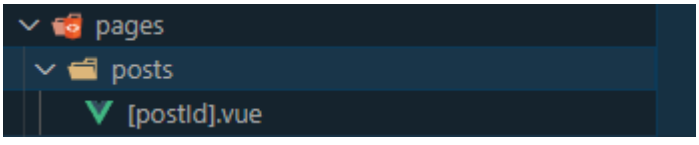
```vue
<template>
    <nuxt-link :to="`/posts/${props.postId}`">
        <va-card>
            <div class="max-w-[500px] max-h-[150px] overflow-hidden">
                <img
                    :src="props.image"
                    :alt="props.postTitle"
                    style="width: 500px; height: 150px; object-fit: cover"
                />
            </div>
            <va-card-title class="text-center">{{ props.postTitle }}</va-card-title>
            <va-card-content>{{ props.postDescription }}</va-card-content>
        </va-card>
    </nuxt-link>
</template>

<script setup>
const props = defineProps({
    image: {
        type: String,
        default: '',
    },
    postTitle: {
        type: String,
    },
    postDescription: {
        type: String,
        default: '',
    },
    postId: {
        type: Number,
        default: null,
    },
});
</script>
```

Slika 59. Post-component komponenta (vlastita izrada)

Post-component komponenta je komponenta koja prikazuje karticu objave. Iznimno je jednostavna komponenta i primjer od kakvih komponenti bi web mjesto trebalo biti izrađeno. *Script* oznaka je korištena samo za definiranje atributa koji mogu biti proslijeđeni komponenti i kasnije se ti atributi koriste unutar *template* oznake kako bi se izradila komponenta.

Novost u ovoj komponenti je uporaba *nuxt-link* komponenta za navigaciju. Ovo je komponenta koja se koristi umjesto <*a*> oznake za brzu navigaciju na interne stranice, dok se <*a*> oznaka koristi za navigaciju na eksterne web stranice [55]. Za uspješnu navigaciju potrebno je priložiti *to* atribut, koji je u ovom slučaju dinamičkog oblika jer njegova vrijednost ovisi o identifikacijskom ključu objave. Pošto *nuxt-link* služi za navigaciju na stranicu prikaza detalja novosti, kreirana je dinamička stranica koja će prikazivati sadržaj s obzirom na proslijeđenu vrijednost [56]. Kreiranje dinamičke stranice je moguće na način koji je prikazan slikom 60. Unutar mape *pages* potrebno je kreirati još jednu mapu unutar koje će se nalaziti *vue* datoteka čije ime je spremljeno unutar uglatih zagrada. Uglate zagrade označavaju vrijednost koja će biti dinamička te se ona može koristiti unutar dinamičke stranice [56].

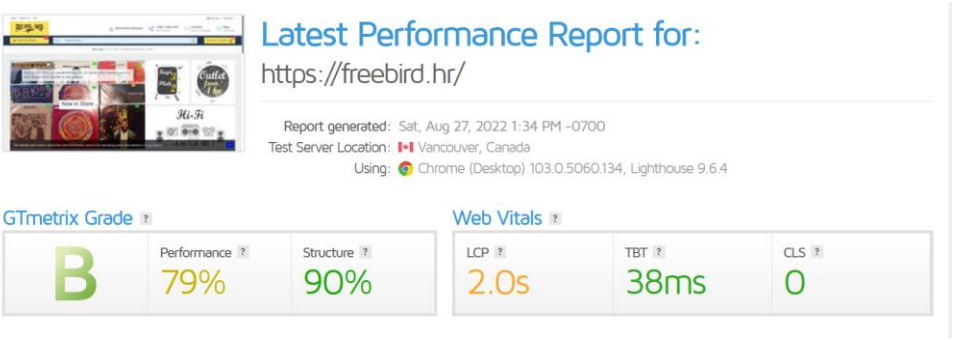


Slika 60. Struktura dinamičke stranice (vlastita izrada)

Ostale komponente i stranice dijele principe koji su prikazani u ovom poglavlju. Zbog sličnosti i velikog broja komponenti neće sve biti prikazane, ali je moguće pogledati sav programski kod na Github repozitoriju: https://github.com/antonlovric/freebird-front

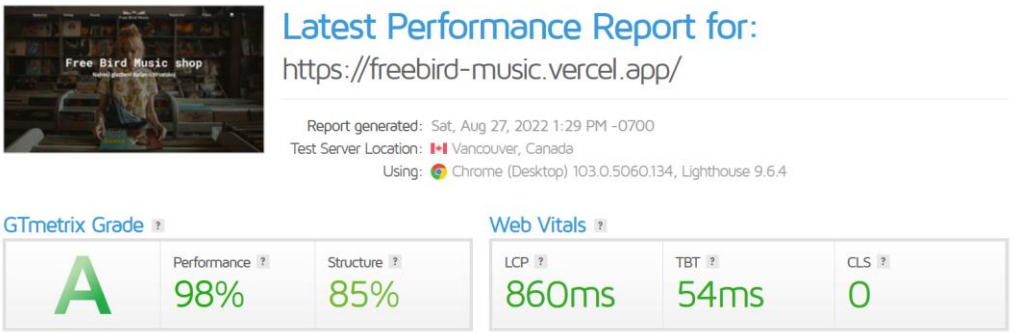
3.2.5. Testiranje

Nakon izrade web mjesta potrebno ga je testirati prethodno opisanim alatima i vidjeti ako je potrebno još nešto unaprijediti oko web mjesta. Prije svega prikazat ću rezultate testiranja trenutnog web mjesta radi usporedbe s novoizrađenim web mjestom.



Slika 61. Testiranje trenutnog web mjesta alatom GTMetrix (https://gtmetrix.com/reports/freebird.hr/t83vD3Mt/)

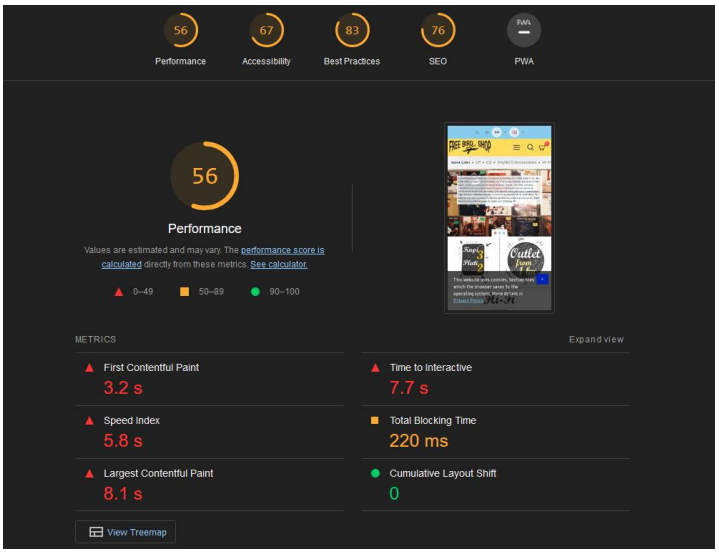
Trenutno web mjesto je dobilo solidne ocjene na GTMetrix testu. 79% za performanse nije idealan rezultat ali 90% za strukturu pomaže tome da cjelokupna ocjena dođe do ocjene B. Glavni nedostatak je 2s trajanje za LCP, odnosno Largest Contentful Paint što označava potrebno vrijeme za učitavanje sadržaja web mjesta [34].



Slika 62. Testiranje novog web mjesta korištenjem alata GTMetrix (https://gtmetrix.com/reports/freebird-music.vercel.app/oy3MjoMO/)

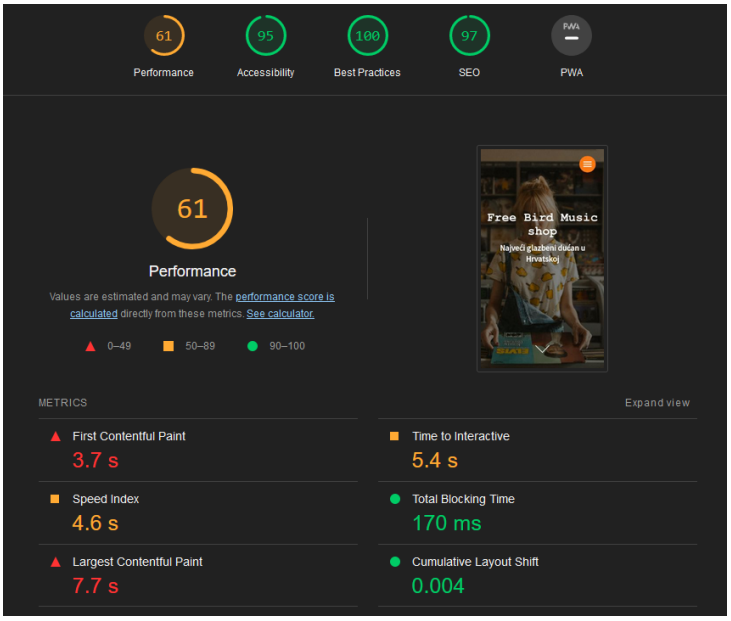
Velikim napretkom u području performansa postignuta je i bolja općenita ocjena web mjesta. Performanse su skočile na čak 98% te je ostvarena ocjena A unatoč tome što je ocjena strukture pala za 5%. Vrijeme LCP-a je također znatno ubrzano, s početnih 2s na 860ms.

Također je potrebno testirati web mjesto pomoću alata Lighthouse, zbog dodatnih kategorija za koje Lighthouse testira web mjesta. Ocjene u kategorijama performanse, pristupačnosti, najboljih praksi i SEO će nam dati bolji dojam o tome ako je web mjesto unaprijeđeno. Lighthouse testovi imaju dvije podjele, testovi na mobilnim i na stolnim uređajima.



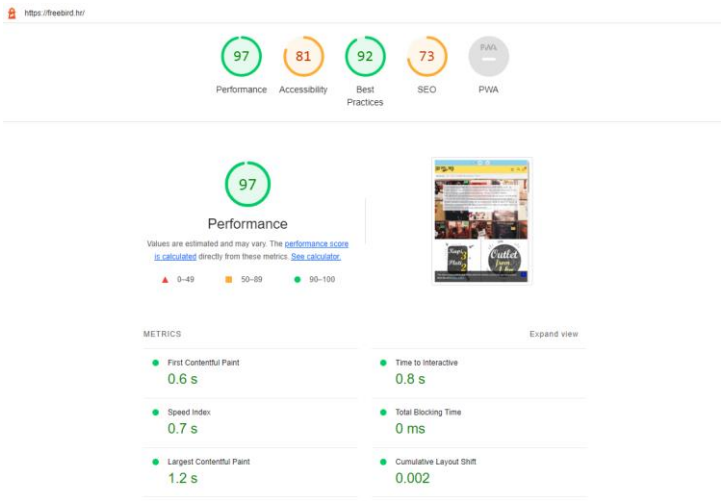
Slika 63. Testiranje trenutnog web mjesta na mobilnim uređajima alatom Lighthouse (vlastita izrada)

Performanse trenutnog mjesta na mobilnim uređajima su u redu, kao i sve ostale kategorije. Ima mjesta za napredak ali su rezultati prolazni. Ocjene za pristupačnost i SEO su ispod očekivanja tako da su to područja na kojima se definitivno može napredovati.



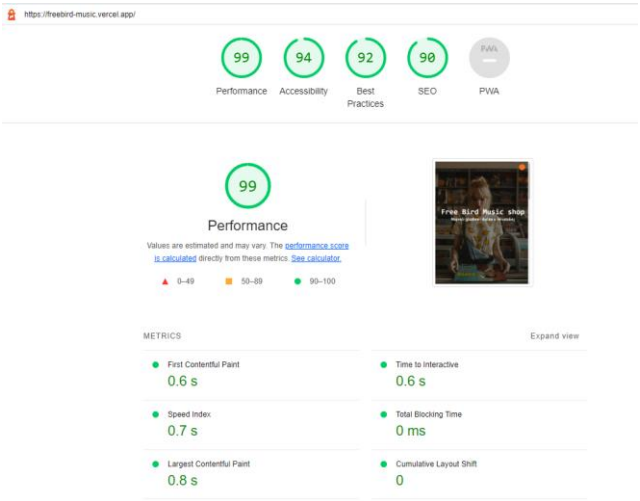
Slika 64. Testiranje novog web mjesta na mobilnim uređajima alatom Lighthouse (vlastita izrada)

Iz rezultata prikazanih na slici 64 je vidljivo da je web mjesto unaprijeđeno u svakoj kategoriji koju mjeri alat Lighthouse. Ocjena za performanse je malo unaprijeđena sa samo 6 bodova, ali su sva ostala područja napravili veliki korak prema naprijed. Pristupačnost je bolja za 28 bodova, najbolje prakse za 17 te SEO za 21 bod. Ovo je značajna razlika te je veliki korak prema naprijed.



Slika 65. Testiranje trenutnog web mjesta na stolnim uređajima putem alata Lighthouse (vlastita izrada)

Performanse i kategorija „najbolje prakse“ su na stolnim uređajima znatno bolje nego na mobilnim uređajima, što je očekivano. Unatoč tome, ocjene za pristupačnost i SEO su i dalje ispod očekivanog, iako je ostvaren napredak i u tim područjima.



Slika 66. Testiranje novog web mjesta na stolnim uređajima alatom Lighthouse (vlastita izrada)

Napredak je ostvaren i na novom web mjestu, gdje sve kategorije imaju ocjenu preko 90. Ovo je znak kvalitetnog web mjesta koje pokriva više područja za provjeru kvalitete. Performanse, pristupačnost i SEO su vjerojatno najbitnija područja kada se vrši analiza web mjesta i činjenica da novo web mjesto ima izvrsne ocjene u svim tim kategorijama govori da je projekt uspješan. Samo web mjesto je moguće posjetiti na poveznici: https://freebird-music.vercel.app/

4. Zaključak

Svijet web programiranja se zaista razvija vrtoglavom brzinom. Lako je dobiti osjećaj da svaki dan nastaje novi okvir za radi koji će postati glavni alat svakog programera kroz nekoliko godina, pronalazi se novi način rješavanja problema koji godinama predstavljaju ogromnu prepreku većini programera ili dolazi novi trend na kojeg cijeli svijet mora paziti. U takvom, ponekad kaotičnom stanju, je bitno ostati u toku sa svim novostima scene web programiranja kako ne bismo ostali iza svoje konkurencije. Biti u toku ne znači zagristi na svaku novinu koja se pojavi, pošto postoji veliki broj alata koji su došli i ubrzo otišli sa scene. Zbog toga je bitno posjedovati vještinu konciznog analiziranja situacije kako bismo donijeli pravu odluku u takvim trenucima.

Za vrijeme izrade ovog završnog rada bilo je brojnih problema koji su zahtijevali pravilnu analizu. Potrebno je analizirati dostupne alate za izradu, načine rješavanja problema te donijeti odluku u kojem smjeru će projekt nastaviti. Odluke koje su donesene prije izrade samog rada su odmah oblikovale strukturu i smjer ovog završnog rada. Da su korišteni neki drugi okviri za rad, drukčiji oblik baze podataka ili da je korišten drukčiji pristup dizajniranju web mjesta, ovaj projekt bi bio neizmjerno drukčiji. Ovaj rad ponovno podsjeća na to koliko različitih vještina je potrebno za izradu web mjesta ove razine. Potrebno je zapamtiti da su programer na strani klijenta, programer na strani poslužitelja, stručnjak za baze podataka te web dizajner različita radna mjesta te s razlogom na tim područjima radi više ljudi. Svako od navedenih područja je ključno za uspješnu izradu web mjesta i pravilno funkcioniranje sustava. Manjak ekspertize u jednom od ovih područja se znatno osjeti na gotovom proizvodu te nakon korištenja stranice ostavlja gorak okus u ustima korisniku.

Ovaj rad je također dobar pokazatelj koliko pažnje je potrebno posvetiti krajnjem korisniku. Iskustvo korisnika treba uvijek biti jedan od glavnih prioriteta prilikom izrade web mjesta. Web mjesto koje nije ugodno za uporabu neće dugo opstati na tržištu jer neće moći akumulirati publiku koja će redovno posjećivati to web mjesto. Kao što je u uvodu rada spomenuto, Internet se iznimno brzo razvija te je zbog toga potrebno izraditi kvalitetan proizvod kojem će se korisnici rado vraćati.

Tijekom izrade ovog rada bilo je puno trenutaka ljutnje, frustracije, stresa i razočarenja, ali to je sastavan dio izrade svakog većeg projekta. Između početne ideje koja je inicirala prijavu ovog završnog rada i posljednje linije koda koja je korištena u projektu se nalazi putovanje neizmjerne važnosti. Ovaj projekt je zahtijevao veliku količinu vremena i energije, ali je zauzvrat donio pregršt znanja i iskustva i to se ne može zanemariti. Izrada ovakvog projekta je ključan korak u razvoju svakog programera te sam iz tog razloga zahvalan na ovom procesu.

Popis literature

[1] N. Huss, “How Many Websites Are There in the World? (2022),” *Siteefy*, Apr. 06, 2022. [Online]. Dostupno: https://siteefy.com/how-many-websites-are-there/. [Pristupljeno: 28.08.2022.]
[2] “What Is UX Design? User Experience Definition | Adobe XD Ideas,” *Ideas*. [Online]. Dostupno: https://xd.adobe.com/ideas/career-tips/what-is-ux-design/. [Pristupljeno: 28.08.2022.]
[3] “Accessibility - W3C.” [Online]. Dostupno: https://www.w3.org/standards/webdesign/accessibility. [Pristupljeno: 28.08.2022.]
[4] “Between the Wires: An interview with Vue.js creator Evan You,” *freeCodeCamp.org*, May 30, 2017. [Online]. Dostupno: https://www.freecodecamp.org/news/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4/. [Pristupljeno: 28.08.2022.]
[5] “Presentation of Vue.js | Introduction to Vue.js.” [Online]. Dostupno: https://worldline.github.io/vuejs-training/presentation/. [Pristupljeno: 28.08.2022.]
[6] N. Jacques, “Jump Start Vue.js, 2nd edition”, SitePoint 19.10.2021.
[7] M. Chaudhari, “Why should you use Vue.js in 2022?,” *Medium*, Jul. 24, 2021. [Online]. Dostupno: https://mayank1513.medium.com/why-should-you-use-vue-js-in-2022-780f2d053919. [Pristupljeno: 28.08.2022.]
[8] “Angular vs. React vs. Vue.js: Comparing performance,” *LogRocket Blog*, Oct. 26, 2021. [Online]. Dostupno: https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/. [Pristupljeno: 28.08.2022.]
[9] “The Hybrid Vue Framework,” *Nuxt 3*. [Online]. Dostupno: https://v3.nuxtjs.org/. [Pristupljeno: 28.08.2022.]
[10] J. Granja, “Will Nuxt 3 Make Vue.js Great Again?,” *Medium*, May 09, 2022. [Online]. Dostupno: https://betterprogramming.pub/will-nuxt-3-make-vue-js-great-again-122672de31ed. [Pristupljeno: 28.08.2022.]
[11] D. Derežić, “Blog | Nuxt.js over Vue.js: when should you use it and why,” *Bornfight*. [Online]. Dostupno: https://www.bornfight.com/blog/nuxt-js-over-vue-js-when-should-you-use-it-and-why/. [Pristupljeno: 28.08.2022.]
[12] “What is web design (and how do I get it right)?,” *99designs*, Oct. 02, 2018. [Online]. Dostupno: https://99designs.com/blog/web-digital/what-is-web-design/. [Pristupljeno: 28.08.2022.]

[13] M. Storm, “Beautiful Websites: 6 Tips for Attractive Website Design,” *WebFX*. [Online]. Dostupno: https://www.webfx.com/blog/web-design/beautiful-websites/. [Pristupljeno: 28.08.2022.]
[14] “Figma vs. Sketch vs. Adobe XD: Which Design Tool Is Better?,” *Coursera*. [Online]. Dostupno: https://www.coursera.org/articles/figma-vs-sketch-vs-adobe-xd. [Pristupljeno: 28.08.2022.]
[15] “The Power of Figma as a Design Tool,” *Toptal Design Blog*. [Online]. Dostupno: https://www.toptal.com/designers/ui/figma-design-tool. [Pristupljeno: 28.08.2022.]
[16] “Naive UI.” [Online]. Dostupno: https://www.naiveui.com/en-US/dark/docs/changelog. [Pristupljeno: 28.08.2022.]
[17] “Naive UI.” [Online]. Dostupno: https://www.naiveui.com/en-US/dark/docs/introduction. [Pristupljeno: 28.08.2022.]
[18] “Vuetify,” *Github*. [Online]. Dostupno: https://github.com/vuetifyjs/vuetify. [Pristupljeno: 28.08.2022.]
[19] “Why you should be using Vuetify - Vuetify .” [Online]. Dostupno: https://vuetifyjs.com/en/introduction/why-vuetify/. [Pristupljeno: 28.08.2022.]
[20] “Accessibility (a11y) - Vuetify” [Online]. Dostupno: https://vuetifyjs.com/en/features/accessibility/. [Pristupljeno: 28.08.2022.]
[21] “Internationalization (i18n) - Vuetify” [Online]. Dostupno: https://vuetifyjs.com/en/features/internationalization/. [Pristupljeno: 28.08.2022.]
[22] “The Vuetify roadmap - Vuetify” [Online]. Dostupno: https://vuetifyjs.com/en/introduction/roadmap/. [Pristupljeno: 28.08.2022.]
[23] “Vuestic UI — Vue.js 3.0 UI Framework,” *Vuestic UI*. [Online]. Dostupno: https://vuestic.dev/en/introduction/overview. [Pristupljeno: 28.08.2022.]
[24] “Vuestic UI,” *Vuestic UI*. [Online]. Dostupno: https://github.com/epicmaxco/vuestic-ui. [Pristupljeno: 28.08.2022.]
[25] “Backend Definition.” [Online]. Dostupno: https://techterms.com/definition/backend. [Pristupljeno: 28.08.2022.]
[26] “Usage Statistics and Market Share of PHP for Websites, August 2022.” [Online]. Dostupno: https://w3techs.com/technologies/details/pl-php. [Pristupljeno: 28.08.2022.]
[27] M. Stauffer, “Laravel: Up & Running: A framework for Building Modern PHP Apps.”, O’Reilly Media, 20.04.2019.
[28] A. Heddings, “What Is Laravel, And How Do You Get Started with It?,” *How-To Geek*. [Online]. Dostupno: https://www.howtogeek.com/devops/what-is-laravel-and-how-do-you-get-started-with-it/. [Pristupljeno: 28.08.2022.]

[29] “Migrations - Laravel - The PHP Framework For Web Artisans.” [Online]. Dostupno: https://laravel.com/docs/9.x/migrations. [Pristupljeno: 28.08.2022.]
[30] “Controllers - Laravel - The PHP Framework For Web Artisans.” [Online]. Dostupno: https://laravel.com/docs/9.x/controllers. [Pristupljeno: 28.08.2022.]
[31] L. Singla, “Why Laravel Framework is the Best Choice for PHP Web Development,” *Insights - Web and Mobile Development Services and Solutions*, Apr. 27, 2022. [Online]. Dostupno: https://www.netsolutions.com/insights/laravel-framework-benefits/. [Pristupljeno: 28.08.2022.]
[32] I. Wibowo, “Search Engine Market Share (2022): Google, Bing, Yahoo etc.,” *Tooltester*, May 27, 2022. [Online]. Dostupno: https://www.tooltester.com/en/blog/search-engine-market-share/. [Pristupljeno: 28.08.2022.]
[33] N. Patel, “What is Google Lighthouse?,” *Neil Patel*, Dec. 22, 2020. [Online]. Dostupno: https://neilpatel.com/blog/google-lighthouse/. [Pristupljeno: 28.08.2022.]
[34] “What is GTMetrix and How Can It Help - Site Speed - 8MS Blog,” *8 Million Stories (8MS)*, Feb. 26, 2019. [Online]. Dostupno: https://8ms.com/blog/what-is-gtmetrix-and-how-can-it-help-your-business-online/. [Pristupljeno: 28.08.2022.]
[35] “About Us.” [Online]. Dostupno: https://freebird.hr/index.php?route=information/information&information_id=4. [Pristupljeno: 28.08.2022.]
[36] click2view, “The benefits of carousels,” *Click2View*, Dec. 16, 2020. [Online]. Dostupno: https://click2view.asia/the-benefits-of-carousels/. [Pristupljeno: 28.08.2022.]
[37] J. Rinaldi, “Why Homepage Carousels Are Bad (& 3 Alternatives to Try Instead).” [Online]. Dostupno: https://www.impactplus.com/blog/why-homepage-carousels-are-bad. [Pristupljeno: 28.08.2022.]
[38] M. Aktas, “What is Banner Blindness? + 11 Clever Methods to Avoid It,” *UserGuiding*, Feb. 26, 2021. [Online]. Dostupno: https://userguiding.com/blog/banner-blindness/. [Pristupljeno: 28.08.2022.]
[39] “Why Image Carousels Are Almost Always A Bad Idea,” *The Good*. [Online]. Dostupno: https://thegood.com/insights/ecommerce-image-carousels/. [Pristupljeno: 28.08.2022.]
[40] “Why Vinyl Records Are Making a Comeback in 2022,” *The Manual*, Jan. 04, 2022. [Online]. Dostupno: https://www.themanual.com/culture/why-vinyl-is-coming-back/. [Pristupljeno: 28.08.2022.]
[41] “What is a database?” [Online]. Dostupno: https://www.oracle.com/database/what-is-database/. [Pristupljeno: 28.08.2022.]

[42] “What is a Relational Database? Features & Uses,” *Salesforce UK Blog*. [Online]. Dostupno: https://www.salesforce.com/uk/blog/2022/01/what-is-a-relational-database.html. [Pristupljeno: 28.08.2022.]
[43] “What is RESTful API? - RESTful API Beginner’s Guide - AWS,” *Amazon Web Services, Inc.* [Online]. Dostupno: https://aws.amazon.com/what-is/restful-api/. [Pristupljeno: 28.08.2022.]
[44] “Eloquent: Getting Started - Laravel - The PHP Framework For Web Artisans.” [Online]. Dostupno: https://laravel.com/docs/9.x/eloquent. [Pristupljeno: 28.08.2022.]
[45] “Eloquent: Relationships - Laravel - The PHP Framework For Web Artisans.” [Online]. Dostupno: https://laravel.com/docs/9.x/eloquent-relationships. [Pristupljeno: 28.08.2022.]
[46] “Database: Pagination - Laravel - The PHP Framework For Web Artisans.” [Online]. Dostupno: https://laravel.com/docs/9.x/pagination. [Pristupljeno: 28.08.2022.]
[47] “Validation - Laravel - The PHP Framework For Web Artisans.” [Online]. Dostupno: https://laravel.com/docs/9.x/validation. [Pristupljeno: 28.08.2022.]
[48] “Collections - Laravel - The PHP Framework For Web Artisans.” [Online]. Dostupno: https://laravel.com/docs/9.x/collections. [Pristupljeno: 28.08.2022.]
[49] “Reactivity API: Core | Vue.js.” [Online]. Dostupno: https://vuejs.org/api/reactivity-core.html#reactive. [Pristupljeno: 28.08.2022.]
[50] “useFetch,” *Nuxt 3*. [Online]. Dostupno: https://v3.nuxtjs.org/api/composables/use-fetch/. [Pristupljeno: 28.08.2022.]
[51] “Head Management,” *Nuxt 3*. [Online]. Dostupno: https://v3.nuxtjs.org/guide/features/head-management/. [Pristupljeno: 28.08.2022.]
[52] “What you should know about Vue v-model.” [Online]. Dostupno: https://learnvue.co/tutorials/v-model-guide. [Pristupljeno: 28.08.2022.]
[53] “List Rendering | Vue.js.” [Online]. Dostupno: https://vuejs.org/guide/essentials/list.html. [Pristupljeno: 28.08.2022.]
[54] “<script setup> | Vue.js.” [Online]. Dostupno: https://vuejs.org/api/sfc-script-setup.html#defineprops-defineemits. [Pristupljeno: 28.08.2022.]
[55] “Nuxt 3 - <NuxtLink>,” *Nuxt 3*. [Online]. Dostupno: https://v3.nuxtjs.org/api/components/nuxt-link/. [Pristupljeno: 28.08.2022.]
[56] “Nuxt 3 – Pages Directory,” *Nuxt 3*. [Online]. Dostupno: https://v3.nuxtjs.org/guide/directory-structure/pages/. [Pristupljeno: 28.08.2022.]

Popis slika