IZRADA WEB APLIKACIJA TEMELJEM KORISNIČKIH DOŽIVLJAJA

Radić, Kristian

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

KRISTIAN RADIĆ

Z A V R Š N I R A D

IZRADA WEB APLIKACIJA TEMELJEM KORISNIČKIH DOŽIVLJAJA

Split, rujan 2023.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

Predmet: Operativni sustavi

Z A V R Š N I R A D

Kandidat: Kristian Radić

Naslov rada: Izrada web aplikacija temeljem korisničkih doživljaja

Mentor: Ljiljana Despalatović, viši predavač

Split, rujan 2023

SAŽETAK

Cilj ovog završnog rada je izrada web aplikacije temeljene na korisničkim iskustvima, prikupljenim putem ankete. Proces je proveden kroz tri faze. Prva faza uključuje anketiranje ciljane skupine korisnika kako bi se prikupili podaci o korisničkim iskustvima, navikama i sigurnosnim zahtjevima. U drugoj fazi će se izraditi prototip dizajna koji će biti prilagođen korisničkim povratnim informacijama. Konačna faza obuhvaća izradu web aplikacije koristeći tehnologije kao što su React, JavaScript, Webpack i SCSS. Serverska aplikacija će se izraditi koristeći Express framework za Node.js, a podaci će se pohraniti u bazu MongoDB na oblak servisu Atlas uz korištenje biblioteke Mongoose za upravljanje podacima.

Ključne riječi: *dizajn, korisničko iskustvo, web aplikacija*

ABSTRACT

Creating web applications based on user experiences

The aim of this final thesis is to develop a web application based on user experiences gathered through a survey. The process is conducted in three phases. The first phase involves surveying a targeted user group to collect data on user experiences, web habits, and security requirements. In the second phase, a design prototype will be created, tailored to user feedback. The final phase encompasses the development of the web application using technologies such as React, JavaScript, Webpack, and SCSS. The backend will be built using the Express framework for Node.js, and data will be stored in a MongoDB database on the Atlas cloud service, utilizing the Mongoose library for data management.

Keywords: *user experience, user interface, web application*

1. UVOD

Kupovina modnih proizvoda preko interneta postaje svakodnevnica. Kako tehnologija napreduje, tako i očekivanja korisnika postaju sve kompleksnija. Kako bi web trgovine postigle ciljani uspjeh i ostale konkurentne na tržištu, potrebna je njihova prilagodba specifičnim potrebama i očekivanjima korisnika. U središtu ovog pristupa nalazi se korisničko iskustvo, poznatije kao UX (engl. *User experience*). Ovaj aspekt, koji se sve više ističe kao ključna komponenta, ne samo da utječe na uspješnost web trgovine, već i na lojalnost korisnika i njihovu volju za ponovnim posjetom.

Motivacija za ovaj rad proizašla je iz potrebe za dubljim razumijevanjem ponašanja korisnika koji redovito koriste internet za kupovinu modnih proizvoda. Kroz anketu koja je bila usmjerena prema razumijevanju ponašanja i navikama korisnika, težilo se dokazivanju postojanja idealnog dizajna koji može biti postignut prateći povratne informacije korisnika. Prednosti takvog pristupa, u izradi web aplikacija, naglašene su kroz cijeli rad, a posebno se ističe važnost kontinuirane interakcije s korisnicima kako bi se osigurala optimalna funkcionalnost i zadovoljstvo.

Proces izrade aplikacije podijeljen je u pet faza: istraživanje, anketiranje, dizajniranje, razvoj i dokumentiranje. Svaka faza je pažljivo provedena kako bi se osiguralo da konačni proizvod najbolje odgovara potrebama korisnika. Ovaj detaljni pristup osigurava da se svaki aspekt korisničkog iskustva uzima u obzir, od prvog kontakta s web aplikacijom pa sve do postupka kupnje.

U nastavku rada, detaljno je opisana metodologija istraživanja, tehnički aspekti izrade web aplikacije te analiza prikupljenih podataka i njihov utjecaj na konačni dizajn i funkcionalnost aplikacije. Poseban naglasak bit će stavljen na važnost korisničkog iskustva u svakoj fazi razvoja i kako ono oblikuje konačni proizvod.

2. KORISNIČKO ISKUSTVO

2.1. Usporedba korisničkog iskustva s korisničkim sučeljem

Korisničko sučelje, poznatije kao UI (engl. *User Interface*) predstavlja dizajn sučelja kroz koji korisnici imaju interakciju s aplikacijom ili web stranicom. To uključuje estetske elemente poput fontova, rasporeda elemenata i boja. Osim toga, UI se bavi operativnosti menija i drugim interaktivnim elementima koji korisnicima omogućuju navigaciju i interakciju s aplikacijom. Ukratko, UI je ono što korisnik vidi i s čime izravno komunicira.

Korisničko iskustvo, s druge strane, odnosi se na cjelokupno iskustvo korisnika prilikom korištenja aplikacije ili web stranice. To ne uključuje samo dizajn, već i kako se korisnik osjeća prilikom korištenja aplikacije. UX se bavi pitanjima poput: Je li aplikacija intuitivna? Je li lako pronaći informacije? Pruža li aplikacija pozitivno iskustvo? Cilj UX-a je osigurati da aplikacija ne samo da izgleda dobro, već i da zadovoljava potrebe korisnika i pruža im ugodno iskustvo [1].

2.2. Važnost korisničkog iskustva

Korisničko iskustvo odnosi se na percepcije i reakcije korisnika prilikom interakcije s proizvodom ili uslugom, u ovom slučaju, web stranicom. Kvalitetan UX dizajn ključan je za uspjeh svake web stranice jer direktno utječe na zadovoljstvo korisnika. Web stranice s lošim UX-om mogu frustrirati korisnike, što može rezultirati smanjenim angažmanom ili napuštanjem stranice. S druge strane, intuitivne i korisnički prijateljske stranice potiču korisnike da se dulje zadrže i češće vraćaju. Brzina učitavanja stranice, jasna navigacija i pristupačnost ključni su elementi dobrog korisničkog iskustva. Kvalitetan UX dizajn obuhvaća organizaciju sadržaja, čitljivost, navigaciju, interakciju i efikasnost web stranice. Poboljšanja u dizajnu sučelja mogu pružiti rješenja za probleme s kojima se korisnici suočavaju prilikom pristupa i interakcije s web stranicama [2].

2.3. Koraci u implementiranju korisničkog iskustva

Implementacija korisničkog iskustva obuhvaća niz koraka koji osiguravaju da proizvod ili usluga zadovoljava potrebe i očekivanja korisnika. Evo osnovnih koraka u implementaciji korisničkog iskustva:

1. **Definiranje projekta:** U prvoj fazi potrebno je točno odrediti što treba kreirati i zašto. Zašto ovaj proizvod mora postojati? Za koga ovo stvarate? Koje će poslovne probleme to riješiti? U ovom koraku dizajneri proizvoda stvaraju temeljni pristup koji je u skladu s poslovnom strategijom. U konačnici nastaje osnovni nacrt koji se zatim može koristiti u sljedećem koraku.
2. **Istraživanje korisnika:** Ovo je najvažniji korak u implementaciji korisničkog iskustva. Prikupljaju se informacije o korisnicima, njihovim potrebama i željama kako bi se stvorio dizajn koji ispunjava njihova očekivanja. Ovaj proces može uključivati intervjue, ankete, promatranje korisnika, analizu konkurencije i druge metode istraživanja.
3. **Analiza i planiranje:** Uz pomoć informacija sakupljenih u fazi istraživanja, dizajneri počinju transformirati te podatke u strategiju koja će biti nit vodilja cijelog procesa dizajna. U ovoj fazi dizajneri također počinju razmišljati o tome kako će proizvod biti izgrađen i koje će tehnologije biti potrebne.
4. **Dizajn:** Nakon što je strategija postavljena, UX dizajner može početi stvarati stvarni dizajn koji je jednostavan za korištenje. U ovoj fazi dizajna radi se na stvarima kao što su: informacijska arhitektura, navigacija, izgled, upotrebljivost i pristupačnost.
5. **Izrada prototipova:** Ovaj korak uključuje stvaranje osnovnih vizualnih prikaza (engl. *wireframe*) i interaktivnih modela (prototipova) kako bi se vizualiziralo kako će proizvod izgledati i funkcionirati.
6. **Testiranje:** Prije lansiranja, važno je testirati sučelje sa stvarnim korisnicima. Testiranje pomaže identificirati sva područja koja trebaju poboljšanja prije nego što finalni proizvod postane aktivan te daje povratne informacije s gledišta korisnika.
7. **Lansiranje**: Nakon što su svi testovi završeni i problemi riješeni, proizvod je spreman za predaju razvojnom timu na implementaciju.

8. **Iteracije:** Nakon lansiranja, proizvod – bilo da se radi o web stranici, aplikaciji ili drugom digitalnom proizvodu – nije gotov. Proces dizajna je kontinuirani ciklus koji se treba ponavljati dok korisnici komuniciraju s proizvodom i daju povratne informacije o proizvodu. Cilj je kontinuirano poboljšavati korisničko iskustvo uvođenjem malih promjena i poboljšanja tijekom vremena [3].

2.4. Alati i tehnike za UX dizajn

Alati za projektiranje korisničkog sučelja dizajnerima pomažu da u kratkom vremenskom periodu izrade prototipove bez korištenja većeg broja alata. Dizajnerski alati koji pomažu u stvaranju skica sučelja igraju ključnu ulogu u razvoju aplikacije, a istraživanja u ovom području usmjerena su na usporedbu i evaluaciju različitih alata s obzirom na korisničko iskustvo [4]. U nastavku su opisana tri najčešće korištena alata za UX dizajn: Figma, Adobe XD i Axure RP.

2.4.1. Figma

Figma pruža dizajnerima alate za kreiranje interaktivnih prototipova i vizualnih koncepta, omogućujući evaluaciju njihove funkcionalnosti i praćenje razvoja. Osim toga, Figma omogućuje da više suradnika može simultano raditi na istom projektu uz mogućnost praćenja suradnje u stvarnom vremenu [5]. Figma se ističe kao jedan od vodećih alata za dizajn, nudeći rješenje temeljeno na oblaku (engl. *cloud-based*) za kreiranje i prototipiranje s bogatim dizajnerskim funkcionalnostima. Njegovo intuitivno sučelje podržava višestruke verzije projekta, olakšavajući usporedbu različitih dizajnerskih pristupa. Jedna od ključnih prednosti Figme je njezina kolaborativna značajka koja omogućuje simultane izmjene više korisnika, eliminirajući potrebu za preuzimanjem datoteka. Ovaj alat također nudi mogućnost kreiranja različitih vizualnih materijala te detaljnih skica za web stranice ili mobilne aplikacije. Integracija s drugim alatima poput Mazea, Zeplina i Confluencea čini Figmu sveobuhvatnim rješenjem u svijetu dizajnerskih alata. Figma je kompatibilna s različitim platformama, uključujući web preglednike, macOS i Windows, čime se proširuje njezina primjenjivost. Dodatno, razvijen je i dodatak pod nazivom FigJam, specijalizirani alat za digitalno skiciranje, koji timovima pruža interaktivno okruženje za suradnju i planiranje dizajnerskih koncepta [4].

2.4.2. Adobe XD

Adobe XD je prepoznat kao inovativan alat za dizajn koji omogućuje dizajnerima da kreiraju i testiraju prototipove korisničkih sučelja. Ovaj alat pruža integrirano rješenje za dizajniranje korisničkih sučelja, prototipiranje i suradnju među timovima. Jedna od ključnih prednosti Adobe XD-a je njegova sposobnost omogućavanja dizajnerima da brzo prelaze između dizajniranja i prototipiranja, čime se olakšava iterativni proces dizajna. Kao i Figma, Adobe XD također podržava suradničke funkcionalnosti, omogućujući više članova tima da istovremeno rade na projektu i dijele povratne informacije u stvarnom vremenu. Još jedna prednost Adobe XD-a je integracija s drugim Adobeovim proizvodima, što olakšava prelazak između različitih faza dizajnerskog procesa. Uz sve ove funkcionalnosti, Adobe XD nastavlja se razvijati i prilagođavati potrebama modernih dizajnera, čineći ga ključnim alatom u industriji dizajna korisničkih sučelja [6].

2.4.3. Axure RP

Axure RP predstavlja alat za brzo prototipiranje s naglaskom na izradu realističnih, funkcionalnih prototipova putem okidača za događaje miša, dodira i tipkovnice. To je izvrsna platforma za dizajniranje vizualnih prikaza i ostale UX dokumentacije. Jedna od ključnih značajki Axure RP-a su *widgeti* koji omogućuju korisnicima da kreiraju i prilagode dijagram toka s vizualnim prikazima i interaktivnim prototipovima. Dodatno, postoji mogućnost integracije prototipova s postojećim alatima kao što su Jira, Microsoft Teams, Slack, Figma i Sketch [4].

3. KORIŠTENE TEHNOLOGIJE

U početnoj fazi projekta koristi se Google obrasci (engl. *Google Forms*) za prikupljanje informacija radi boljeg razumijevanja korisničkih potreba i očekivanja. Nakon toga, prelazi se na dizajnersku fazu u Figmi, kako bi se stvorili vizualni prototipovi i sučelja koja će služiti kao vodič za programere. Kada je dizajn spreman, prelazi se na razvoj aplikacije. Za razvoj korisničkog sučelja koriste se tehnologija React za izgradnju dinamičkih komponenti, Webpack za optimizaciju i pakiranje kôda, NPM (engl. *Node Package Manager*) kao upravitelj paketa te SCSS (engl. *Sassy Cascading Style Sheets*) za stiliziranje i kreiranje responzivnih dizajna. S druge strane, poslužiteljska strana projekta izrađena je na Node.js kao okruženju za izvršavanje JavaScripta na poslužitelju. Express framework pruža snažne alate za izgradnju web aplikacija, dok MongoDB služi kao baza podataka za pohranu informacija. Kako bi se osigurala sigurnost i dostupnost, koristi se Atlas Cloud. Tijekom cijelog razvojnog procesa, koristio se Git kao sustav za upravljanje verzijama.

3.1. Google obrasci

Google obrasci koriste se za jednostavno i brzo kreiranje anketa budući da omogućuju prikupljanje različitih vrsta informacija na jednostavan i učinkovit način. Prednosti korištenja Google obrazaca su što je to besplatni online alat koji omogućuje jednostavno i učinkovito prikupljanje informacija. Pomoću Google obrazaca mogu se izraditi ankete u svega nekoliko minuta. Sučelje je vrlo jednostavno za korištenje. Svaki korisnik s prosječnim poznavanjem interneta može kreirati obrasce koristeći ovaj alat. Pomoćnik je jednostavan za korištenje, a sučelje olakšava povlačenje i ispuštanje elemenata obrasca i njihovo organiziranje na temelju akcija ili događaja. Na razini dizajna moguće je birati između palete boja, kao i vlastitih slika kao pozadine. Google obrasci pohranjuju primljene povratne informacije kako bi ih se moglo detaljno analizirati. S ovim alatom može se dobiti neograničen broj pitanja i odgovora bez ikakvih troškova, dok drugi alati za ankete zahtijevaju plaćanje ovisno o broju pitanja i primatelja [7].

3.2. Figma

Kao što je prethodno opisano, Figma je jedan od vodećih alata za dizajn zbog svoje dostupnosti, suradničkih mogućnosti i jednostavne integracije s drugim alatima i

platformama. Figma nudi napredne alate za prototipiranje, omogućujući dizajnerima da brzo stvaraju interaktivne demonstracije svojih dizajna.

3.3. React

React je JavaScript knjižnica otvorenog kôda, čija je glavna namjena kreiranje korisničkih sučelja, a može se koristiti i kao osnova za izradu mobilnih aplikacija. Koristi se jer omogućava izgradnju složenih korisničkih sučelja od manjih dijelova kôda koji upravljaju određenim dijelom korisničkog sučelja [8]. Jedna od glavnih prednosti Reacta je što programeru daje virtualni DOM (engl. *Document Object Model*) umjesto stvarnog DOM-a. Time obrađuje minimalan broj DOM operacija koje su potrebne za postizanje novog stanja te smanjuje broj grešaka i ponavljanje kôda. React koristi JSX (engl. *JavaScript eXtensible Markup Language*), sintaksu koja kombinira JavaScript s HTML-om (engl. *HyperText Markup Language*), omogućavajući razvoj komponenata s deklarativnom strukturom sličnom HTML-u. React je postao ključna tehnologija u razvoju web aplikacija te je pridobio veliku popularnost među programerima zbog svoje fleksibilnosti i mogućnosti prilagodbe [9]. Uz to, React se često koristi u kombinaciji s drugim tehnologijama, kao što su Redux ili GraphQL, kako bi se olakšao razvoj i upravljanje aplikacijom. React i dalje nastavlja biti ključna komponenta u razvoju web tehnologija, pružajući alate koji omogućavaju brži razvoj i bolje korisničko iskustvo. React je postao ne samo osnova za web aplikacije, već i osnova za izradu mobilnih aplikacija putem React Native, što dodatno proširuje njegovu primjenu. S obzirom na njegovu široku primjenu i zajednicu koja ga podržava, React će vjerojatno ostati ključna tehnologija u predstojećim godinama [8].

3.3.1. Webpack

Webpack je alat za pakiranje modula koji omogućuje programerima da razviju složene web aplikacije sastavljene od mnogo različitih resursa. Pomoću Webpacka, programeri mogu transformirati, minimizirati i pakirati JavaScript, CSS, slike i druge resurse u optimizirane pakete spremne za produkciju. Način na koji se danas piše JavaScript razlikuje se od kôda koji preglednik može izvršiti. Webpack može premostiti ovaj jaz i proizvoditi kôd kompatibilan s više preglednika [10].

3.3.2. Npm

Npm predstavlja ključni sustav za upravljanje paketima unutar sustava Node.js. Njegova osnovna svrha je omogućiti programerima jednostavnu instalaciju, ažuriranje i upravljanje softverskim paketima koji su neophodni za razvoj aplikacija. U praksi se često kombinira s Reactom kako bi se omogućilo efikasno upravljanje skriptama tijekom razvojnog procesa. Kombinacija npm-a i Reacta omogućuje programerima da maksimalno iskoriste pakete dostupne na npm registru. To uključuje različite alate, biblioteke i komponente koje mogu značajno poboljšati i ubrzati razvoj aplikacija temeljenih na Reactu. Dodatno, kroz npm skripte, programeri mogu automatizirati niz zadataka koji se javljaju tijekom razvoja React aplikacija. To obuhvaća aktivnosti poput pokretanja lokalnog razvojnog poslužitelja, kreiranja produkcijske verzije aplikacije ili izvođenja testnih skripti [11].

3.3.3. SCSS

Oblikovanje predstavlja jedan od raznih procesa tijekom izrade web stranica. Kako se razvoj web stranica širi, tako njihovi stilski listovi (engl. *stylesheets*) postaju sve veći i složeniji, što ih čini težima za održavanje s vremenom. Upravo u ovakvim situacijama SCSS pre-procesor postaje izuzetno koristan alat. SCSS pomaže da se jednostavnije napišu CSS kôdovi dopuštajući upotrebu petlji, funkcija, varijabli i matematičkih operacija. Proširuje standardne CSS karakteristike jednostavnim uvođenjem prednosti osnovnog programskog jezika. Uzimajući u obzir sve navedene prednosti, jasno je zašto mnogi razvijatelji preferiraju korištenje SCSS-a u kombinaciji s modernim *frontend* tehnologijama poput Reacta [12].

3.4. Node.js

Kao otvorena platforma, Node.js pruža slobodu za razvoj mrežnih aplikacija u stvarnom vremenu. Karakterizira ga postupno, događajno vođeno I/O sučelje. Budući da se temelji na jednoj niti s događajnom petljom, nijedna izvršna operacija se ne može blokirati. Node.js je postao popularna platforma za izradu kompletnih web aplikacija. Omogućuje kreiranje web aplikacija, hibridnih aplikacija i desktop aplikacija. Jedna od ključnih prednosti Node.js-a je visoka performansa. Zahvaljujući V8 pogonskom alatu iz Google Chromea, koji je napisan u C++, Node.js brzo i učinkovito pretvara JavaScript u strojni kôd. Ovaj motor je posebno važan jer Google ulaže u njegovo poboljšanje, pružajući

podršku za napredne JavaScript mogućnosti. Node.js se lako skalira, omogućujući razvojnim inženjerima da prošire aplikacije horizontalno dodavanjem dodatnih čvorova ili vertikalno dodavanjem resursa pojedinim čvorovima. Za frontend razvojne inženjere, koji su već upoznati s JavaScriptom, rad s Node.js na poslužiteljskoj strani postaje intuitivan. Node.js također nudi niz prednosti poput smanjenja vremena učitavanja korištenjem predmemoriranja, poboljšanja vremena odziva i povećanja performansi. Kroz integraciju s raznim alatima i platformama, poput Electrona i NW.js, Node.js omogućuje izradu kros-platformnih aplikacija bez potrebe za pisanjem zasebnog kôda za različite desktop verzije [13].

3.4.1. Express framework

Express framework omogućuje brz razvoj web aplikacija baziranih na Node.js platformi. Omogućuje postavljanje posredničkih softvera, poznatih kao *middlewares*, kako bi se odgovorilo na zahtjeve HTTP-a. Unutar Express frameworka, postoji definirana tablica usmjeravanja koja se koristi za izvođenje različitih radnji, a te radnje ovise o metodi HTTP i konkretnom URL-u (engl. *Uniform Resource Locator*). Osim toga, Express pruža mogućnost dinamičkog prikaza stranica HTML, a prikaz se temelji na argumentima koji su proslijeđeni određenim predlošcima [14].

3.5. MongoDB

MongoDB predstavlja najpopularniju nerelacijsku (NoSQL) bazu podataka, otvorenog je kôda s orijentacijom na dokumente. Za razliku od relacijskih baza podataka koje se temelje na tablicama, baze NoSQL se horizontalno skaliraju što ih čini pravim izborom kada je riječ o obradi velikih količina podataka. Ako baza podataka radi na jednom poslužitelju, doseći će svoj limit skaliranja. Umjesto toga, baze NoSQL nude potpuno drugačiji mehanizam za pohranu i dohvat podataka. MongoDB je primjer takve baze podataka koja se skalira dodavanjem sve više poslužitelja te povećava produktivnost svojim fleksibilnim modelom dokumenata. MongoDB pohranjuje glavne podatke u minimalnom broju dokumenata, a ne tako da ih razbija na više relacijskih struktura, što omogućuje efikasniju i organiziraniju pohranu informacija. Zahvaljujući indeksiranju, MongoDB može brzo obraditi ogromne količine podataka, čime se postiže učinkovito pretraživanje bez potrebe za pregledom svakog pojedinog dokumenta [15].

3.5.1. Atlas cloud

Atlas Cloud je usluga koju nudi MongoDB i predstavlja potpuno upravljani globalni oblak za baze podataka. Ova usluga omogućuje korisnicima da lako postave, nadgledaju i održavaju svoje instance MongoDB-a u oblaku. Jedna od ključnih prednosti Atlas Clouda je njegova sposobnost automatskog skaliranja, što znači da se resursi mogu prilagoditi prema potrebama aplikacije bez ručne intervencije. Osim toga, Atlas Cloud pruža visoku razinu sigurnosti, uključujući šifriranje podataka u mirovanju i prilikom prijenosa, kao i napredne značajke poput automatiziranih sigurnosnih kopija i vraćanja podataka. Uz to, korisnicima omogućuje postavljanje baza podataka u više regija radi optimizacije performansi i usklađenosti S obzirom na sve navedene karakteristike, nije iznenađujuće da se ovakav tip baze podataka koristi u različitim industrijama, od bankarstva do web razvoja [16].

3.6. Git

Git je alat koji programerima omogućuje praćenje i upravljanje promjenama u izvornom kôdu tijekom razvoja softverskih projekata. Funkcionira kao sustav za kontrolu verzija, gdje svaka promjena kôda ima svoju jedinstvenu oznaku, omogućujući timovima da surađuju na istom projektu bez međusobnog prepisivanja rada. Jedna od ključnih prednosti Gita je njegova sposobnost da radi distribuirano, što znači da svaki programer ima potpunu lokalnu kopiju repozitorija, što omogućuje rad čak i kada nema internetske veze [17].

3.6.1. Github

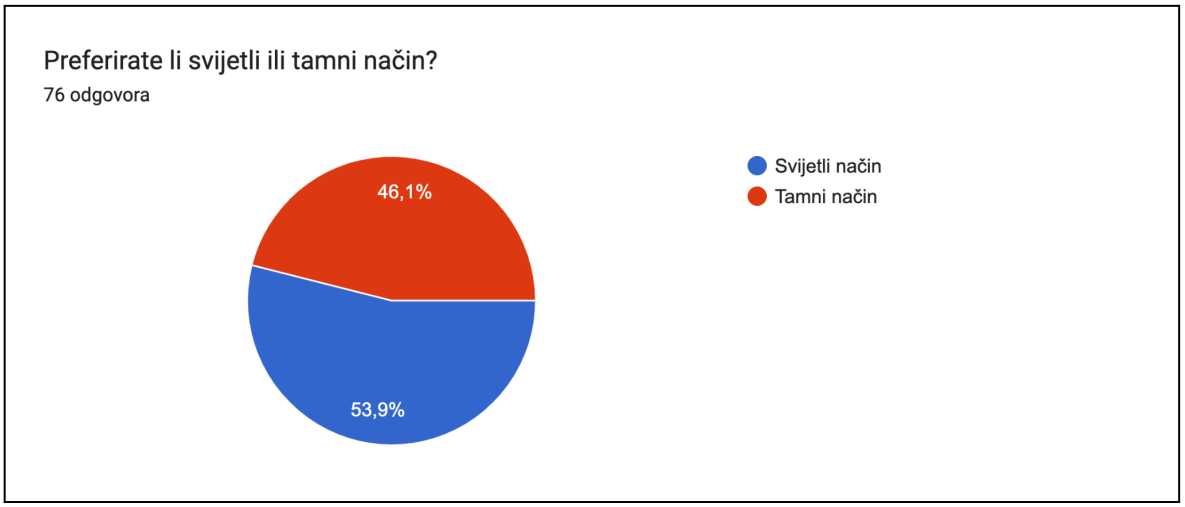
GitHub je web platforma koja koristi Git kao osnovu i omogućuje programerima da pohrane, dijele i surađuju na kôdu putem interneta. Osim što pruža centralizirano mjesto za pohranu repozitorija, GitHub nudi niz alata koji olakšavaju suradnju, poput sustava za praćenje problema, upravljanje projektima i mogućnost pregleda kôda. Mnogi otvoreni izvorni projekti koriste GitHub kao mjesto gdje zajednice mogu surađivati i doprinositi razvoju projekta [18].

4. OPIS PRAKTIČNOG RADA

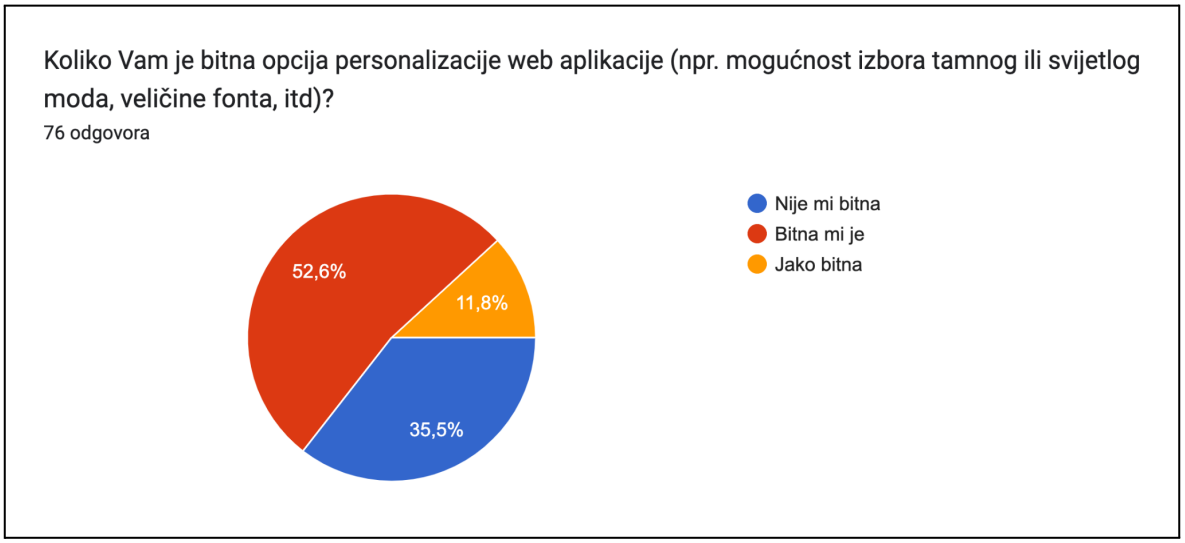
U sljedećem poglavlju napravljena je detaljna analiza i usporedba rezultata ankete ispitivanja afiniteta korisnika sa funkcionalnostima web aplikacije. U istraživanju je sudjelovalo 76 ispitanika, pružajući uvid u različite korisničke profile. Ispitanici su bili različitih dobnih skupina, krećući se u rasponu od 15 do 50 godina te su sudjelovala oba spola.

4.1 Dizajn i korisničko sučelje

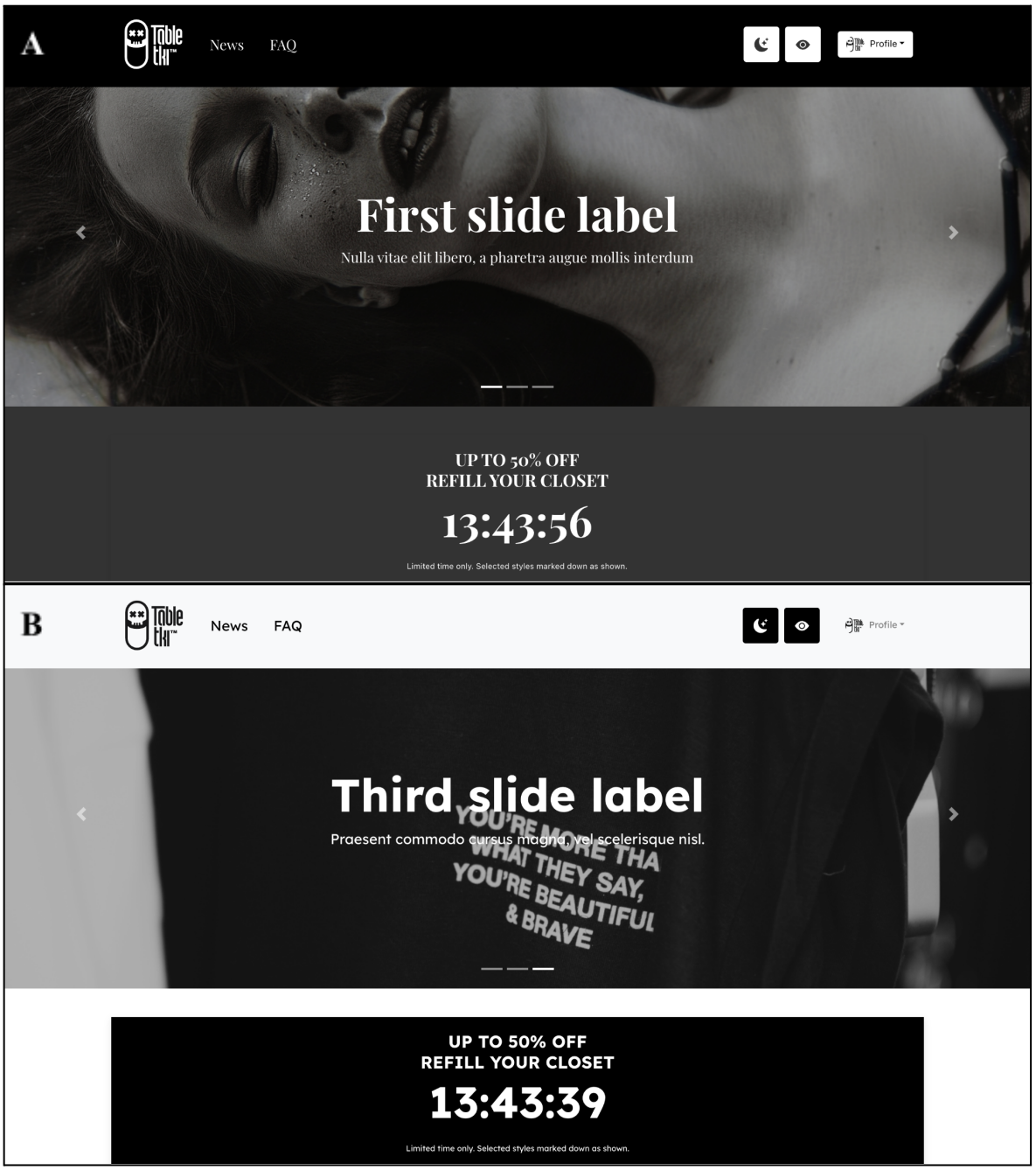
4.1.1. Personalizacija web aplikacije



Slika 1. Grafički prikaz rezultata ankete za pitanje: Preferirate li svijetli ili tamni način? Rezultati pokazuju da 46,1% ispitanika preferira tamni način rada dok 53,9% ispitanika preferira svijetli način rada.



Slika 2. Grafički prikaz rezultata ankete na pitanje: Koliko Vam je bitna opcija personalizacija web aplikacije? Rezultati pokazuju da 64,2% ispitanika smatra personalizaciju bitnom ili jako bitnom, dok 35,5% ispitanika ne smatra bitnom.

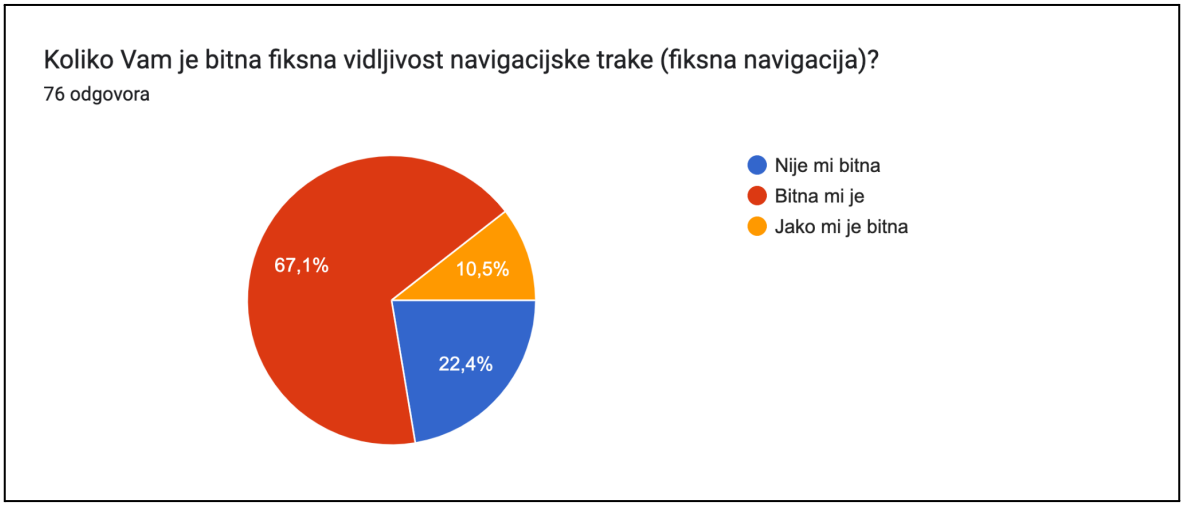


Slika 3. Dizajn na temelju korisničkog iskustva. **A.** Tamni način rada. **B.** Svijetli način rada s fontom koji pomaže kod disleksije.

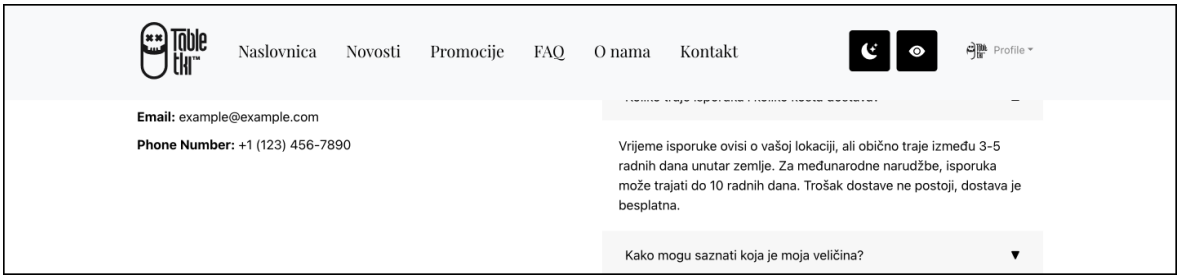
Na temelju rezultata ankete vidljivih na slici 1 može se zaključiti da je većina ispitanika sklonija svijetlom načinu prikaza, dok je nešto manji postotak pokazao tendenciju prema tamnom načinu. Iako je svijetli način nešto popularniji među ispitanicima, razlika u postotcima nije značajno velika, što ukazuje na podijeljeno mišljenje među korisnicima. Na slici 2 vidljivo je da je većini bitna mogućnost odabira svijetlog i tamnog načina, kao i izbora veličine fonta i slično.

Slika 3A prikazuje tamni način rada. Ova opcija može biti aktivirana klikom na odgovarajući botun u navigacijskoj traci. Kada korisnik odabere ovu postavku, njegov odabir se sprema u lokalnu memoriju preglednika. Time je osigurano da se odabrani način prikaza zadržava čak i ako korisnik pređe na drugu stranicu ili napusti aplikaciju. Slika 3B ilustrira svijetli način rada te posebnu opciju za korisnike s disleksijom. Aktivacija ovih opcija postiže se klikom na botun s ikonom oka, koji se nalazi u navigacijskoj traci. Kao i u prethodnom slučaju, korisnički odabir se sprema u lokalnu memoriju preglednika. Nijedan botun nema tekstualno objašnjenje uza sebe. Umjesto toga koriste se samo ikone kako bi se potaklo korisnika na interakciju i otkrivanje novih funkcionalnosti. Vjeruje se da ovakav pristup može pobuditi znatiželju korisnika i pružiti mu ugodno iznenađenje kada otkrije koju funkciju zapravo imaju ikone.

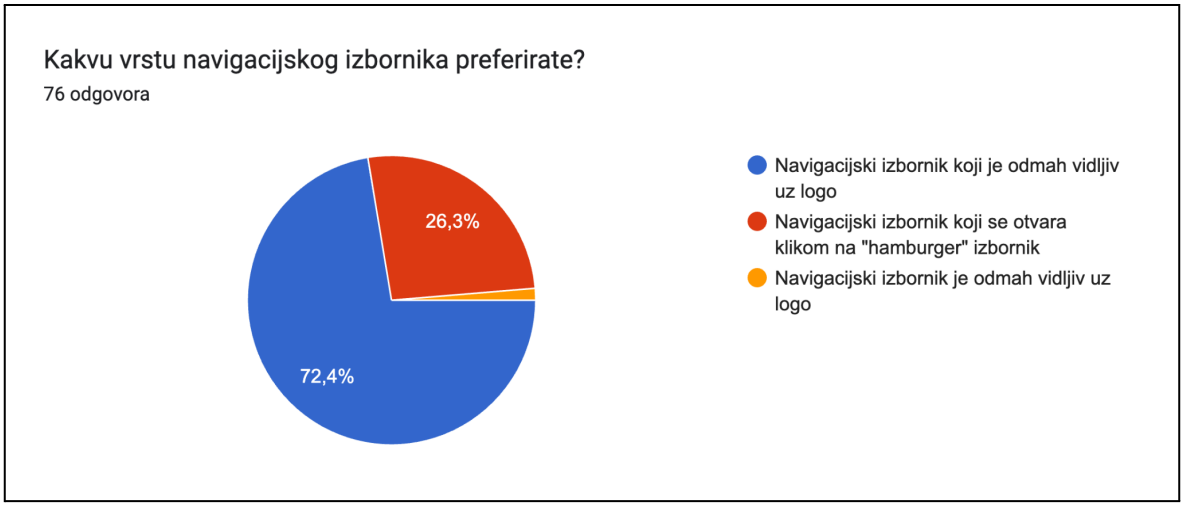
4.1.2. Navigacijski izbornik



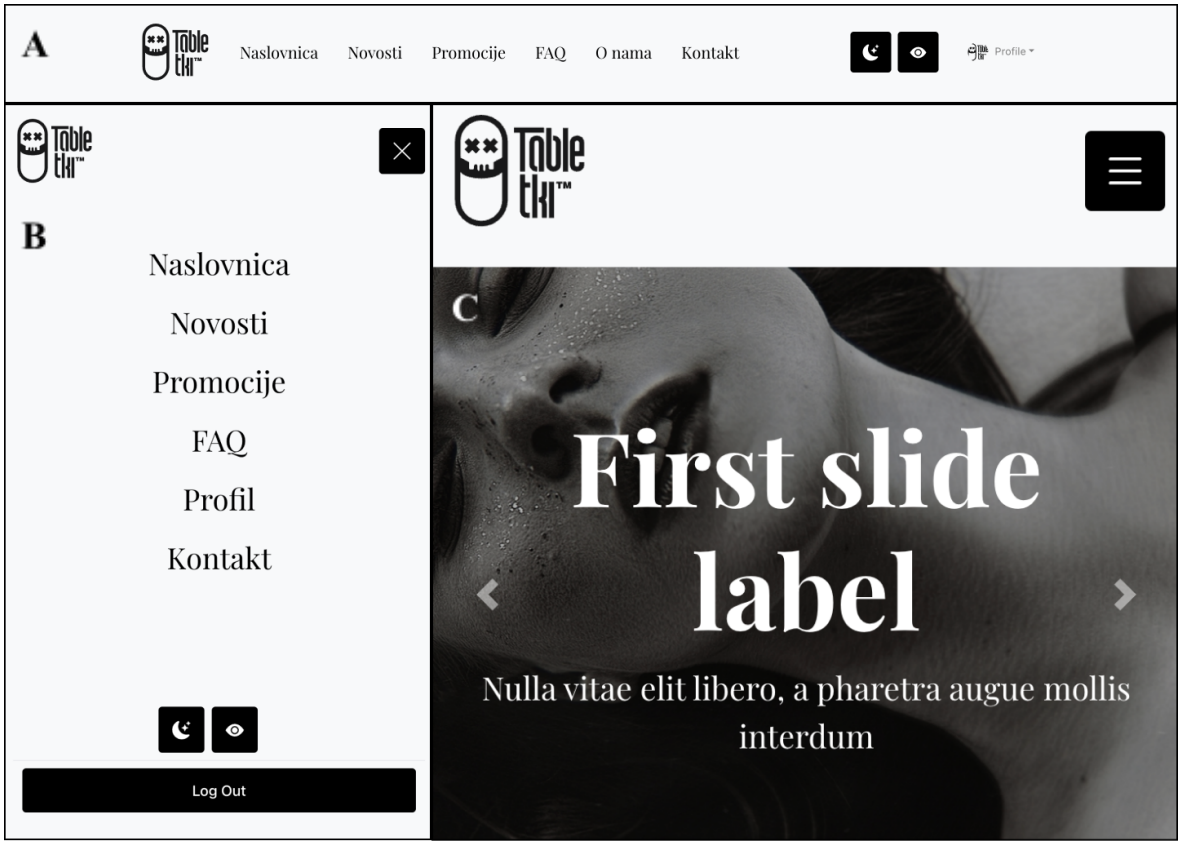
Slika 4. Grafički prikaz rezultata ankete za pitanje: Koliko Vam je bitna fiksna navigacija? Rezultati pokazuju da 77,6% ispitanika preferira fiksnu navigaciju bitnom ili jako bitnom, dok 22,4% njih ne smatra fiksnu navigaciju bitnom.



Slika 5. Fiksna navigacija.



Slika 6. Grafički prikaz rezultata ankete za pitanje: Kakvu vrstu navigacijskog izbornika preferirate?

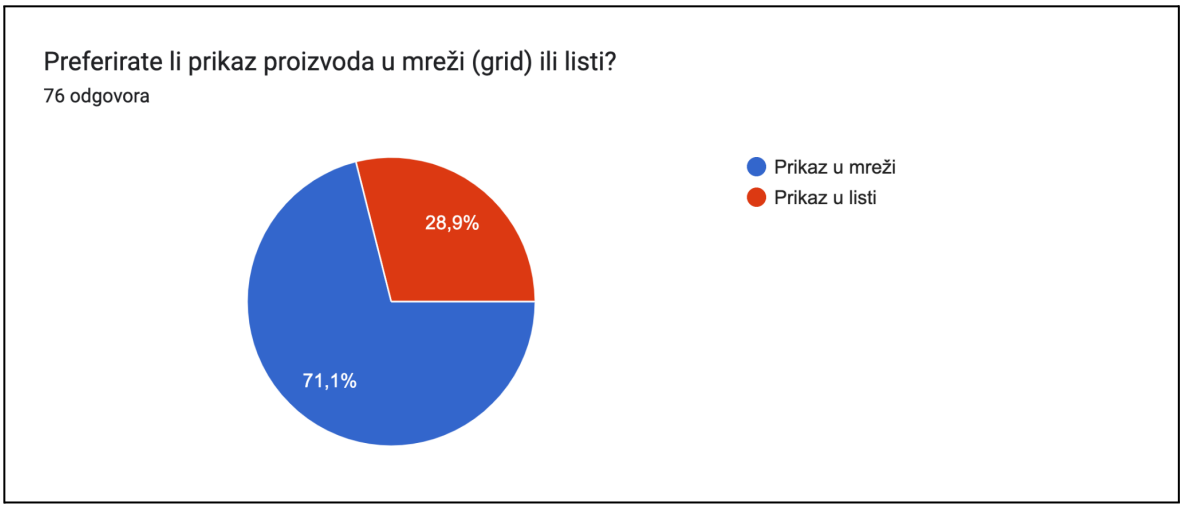


Slika 7. Oblik navigacije na temelju korisničkog iskustva. **A.** Navigacijski traka gdje je navigacijski izbornik odmah vidljiv uz logo. **B.** Otvoreni navigacijski izbornik na mobilnim uređajima. **C.** Zatvoreni navigacijski meni na mobilnim uređajima.

Na temelju rezultata ankete vidljivih na slici 4 može se zaključiti da većina ispitanika preferira da je navigacijska traka fiksna. Fiksna navigacijska traka korisnicima daje pristup glavnim dijelovima web stranice, bez obzira gdje se nalaze na stranici. Primjer

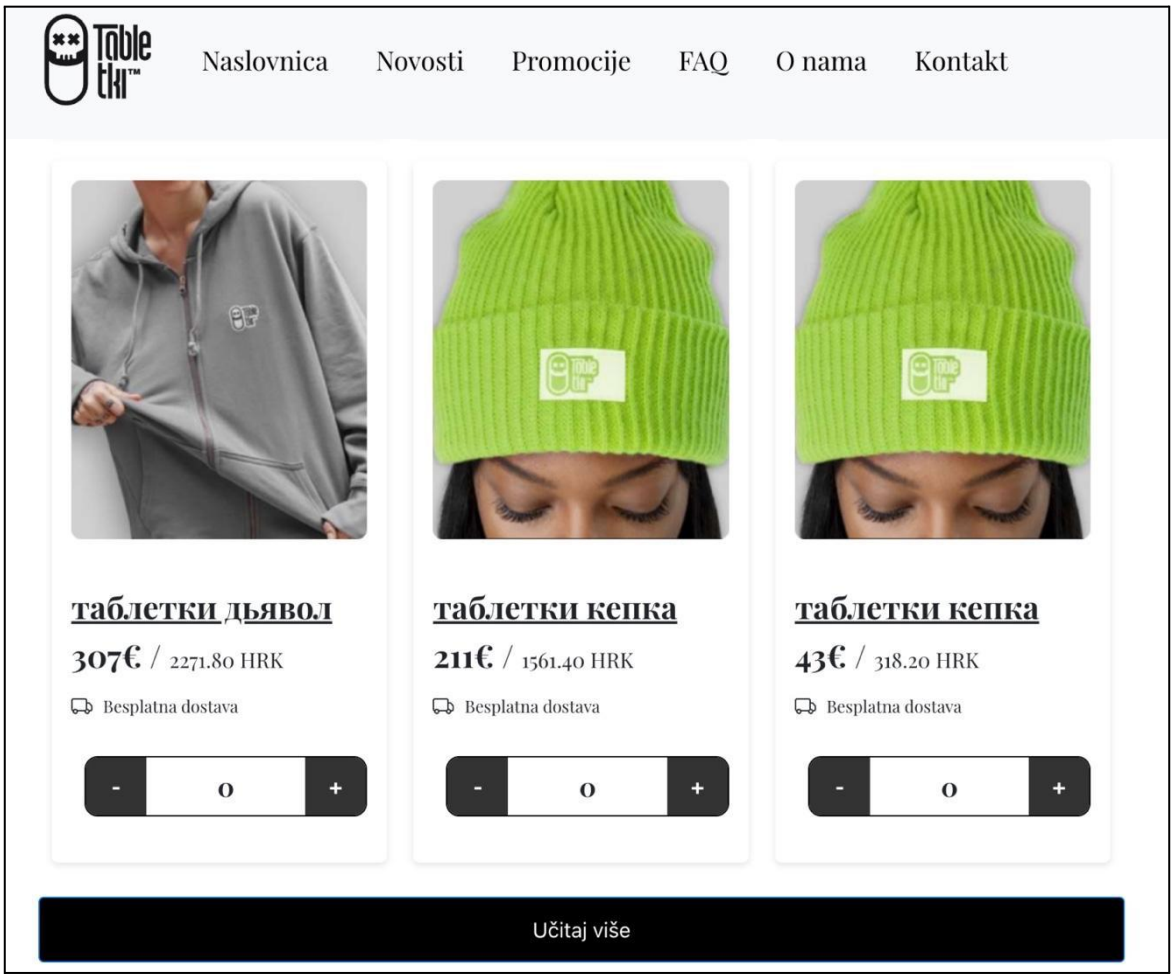
fiksne navigacije vidljiv je na slici 5 Fiksna navigacija može smanjiti konfuziju, pridonijeti na kohezivnosti dizajna i poboljšati korisničko iskustvo. Nadalje, većina ispitanika (72,4%) preferira navigacijski izbornik koji je odmah vidljiv uz logo, kao što je vidljivo na grafičkom prikazu na slici 6. Slika 7A prikazuje navigacijski izbornik koji je odmah vidljiv uz logo. Unatoč mišljenjima korisnika koje su prikupljene putem ankete, postoji određena pravila dizajna koja se moraju poštovati, posebno kada je riječ o mobilnim uređajima. Na mobilnim uređajima, prostor je ograničen, stoga navigacijski izbornik ne može biti smješten odmah uz logo. Umjesto toga, koristi se 'hamburger' botun kako bi se osiguralo da korisničko iskustvo ostane intuitivno i učinkovito. Primjer takvih izbornika vidljiv je na slici 7B i slici 7C.

4.1.3. Prikaz proizvoda



Slika 8. Grafički prikaz rezultata ankete za pitanje: Preferirate li prikaz proizvoda u mreži ili listi? Rezultati pokazuju da 71,1% ispitanika preferira prikaz u mreži naspram prikaza u listi.

Slika 9. Grafički prikaz rezultata ankete za pitanje: Koliko Vam je važno imate opciju da su svi proizvodi prikazani na jednoj stranici s “učitaj više” umjesto kroz višestraničnu navigaciju?



Slika 10. Prikaz proizvoda na temelju korisničkog iskustva.

Na temelju rezultata ankete vidljivih na slici 8, može se uočiti da većina ispitanika preferira prikaz proizvoda u mreži. Mrežni prikaz omogućava veći fokus na vizualnim elementima, poput slika proizvoda, što može pomoći u donošenju odluke o kupnji. Takav prikaz optimizira korištenje prostora omogućavajući korisnicima da brzo i lako pregledavaju i uspoređuju proizvode. Uz sve to, intuitivnost i preglednost mrežnog prikaza često rezultiraju boljim korisničkim iskustvom. Što se tiče rezultata ankete prikazanih na slici 9, na pitanje o važnosti opcije prikaza svih proizvoda na jednoj stranici s funkcijom „učitaj više“ u odnosu na višestraničnu navigaciju, može se zaključiti da većina ispitanika smatra tu opciju bitnom. S ukupno 43,4% ispitanika koji su odgovorili da im je to bitno i dodatnih 21,1% koji smatraju da im je to jako bitno, jasno je da postoji snažna tendencija prema toj funkcionalnosti. Učitavanje proizvoda putem „učitaj više“ botuna omogućava kontinuirano pregledavanje bez potrebe za prelaskom na novu stranicu. Paginacija može prekinuti korisnikov tok pregledavanja, obeshrabriti korisnika i otežati mu kupovinu. Slika 10 prikazuje primjer prikaza proizvoda u mreži s „učitaj više“ opcijom.

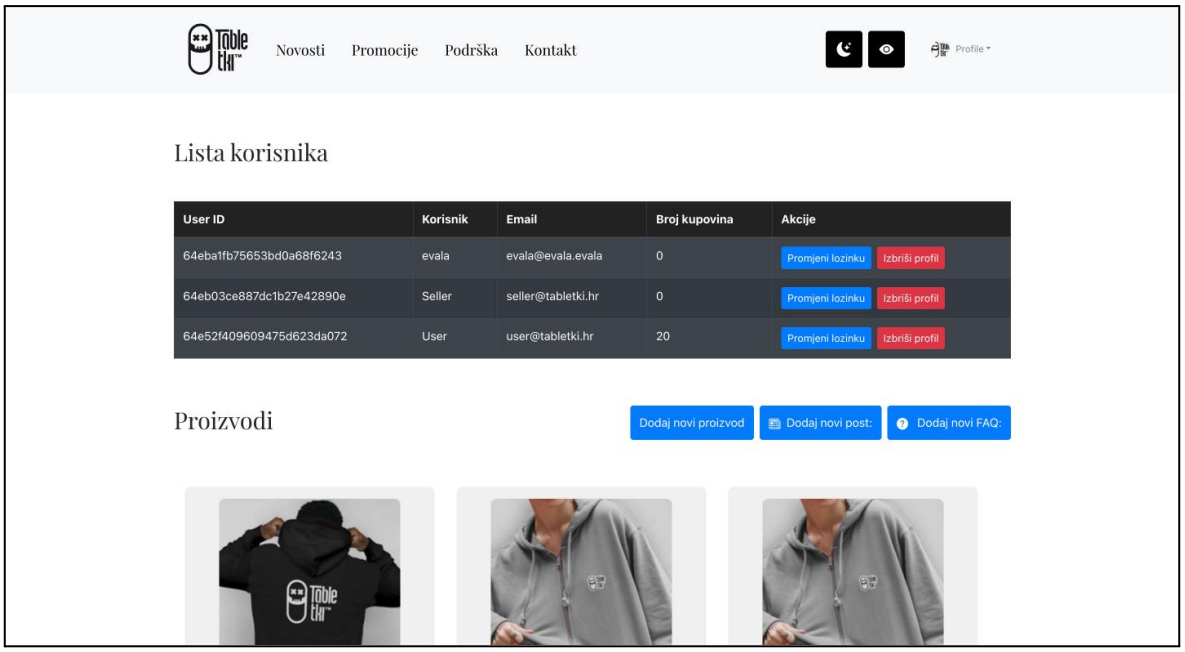
4.2. Funkcionalnosti i sadržaj

4.2.1. Prijava

Slika 11. Grafički prikaz rezultata ankete za pitanje: Preferirate li prijavu odmah ili prijavu tek kod kupovine?

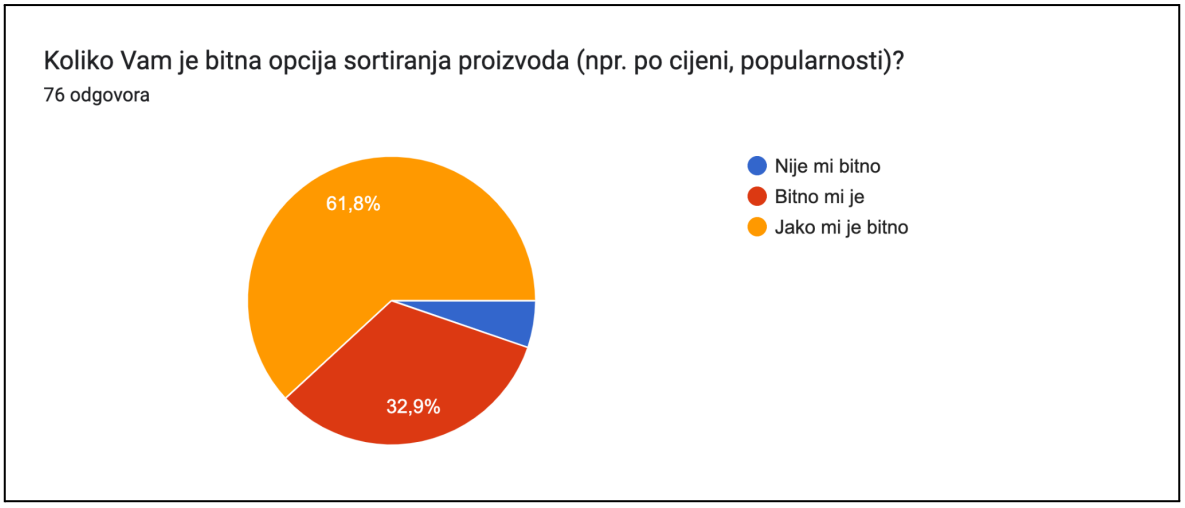
Na temelju rezultata ankete prikazani na slici 11 vidljivo je da većina ispitanika preferira prijavu tek pri kupnji, umjesto prijave odmah. Obavezna prijava prije kupnje može imati negativan utjecaj na korisnike iz različitih razloga. Brzina, jednostavnost i sloboda su aspekti koje korisnici često cijene prilikom online kupovine. Proces koji

uključuje obaveznu prijavu može usporiti korisničko iskustvo. Postoji zabrinutost korisnika u pogledu privatnosti i sigurnosti njihovih osobnih podataka. Osim toga, postoji otpor prema otvaranju novih internet računa i potrebi za pamćenjem dodatnih lozinki. Osjećaj obveze koji se stvara obaveznom prijavom može negativno utjecati na ukupno korisničko iskustvo i odluku o kupnji. Aplikacija nudi tri različite vrste računa, odnosno uloge. Uloga gost omogućuje pregled proizvoda, vijesti i najčešćih pitanja. Uloga kupac pruža mogućnost pristupa postavkama profila i funkcionalnosti kupnje. Dok uloga prodavač pruža dodatne opcije, uključujući kreaciju proizvoda, objavu vijesti i upravljanje najčešćim pitanjima što pokazuje slika 12.

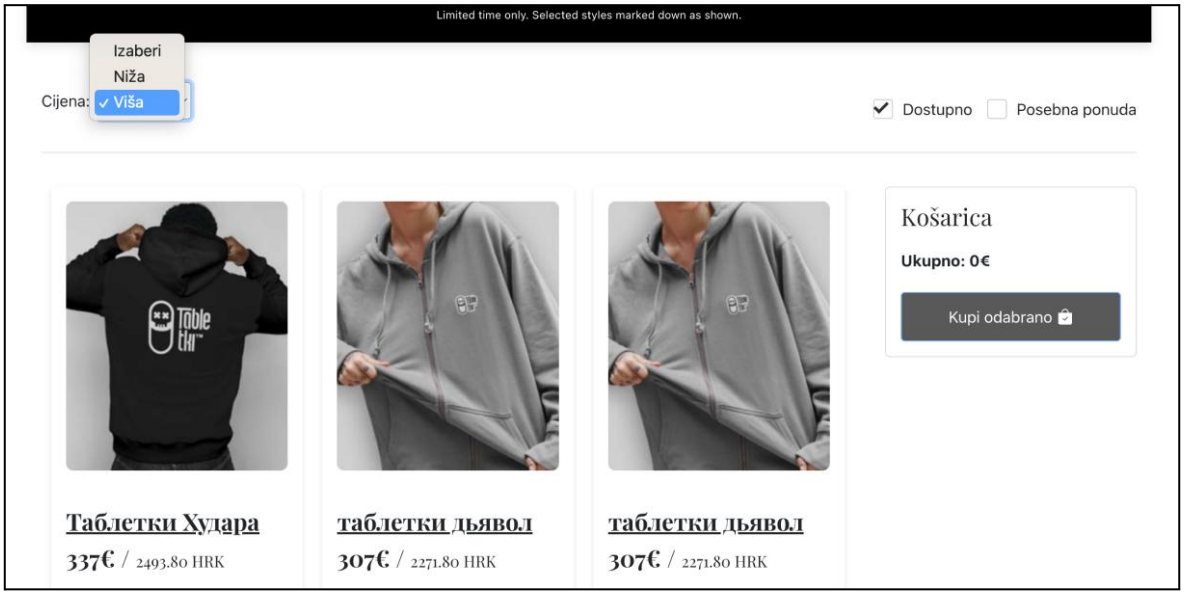


Slika 12. Prikaz naslovne stranice za ulogu prodavač.

4.2.2. Sortiranje



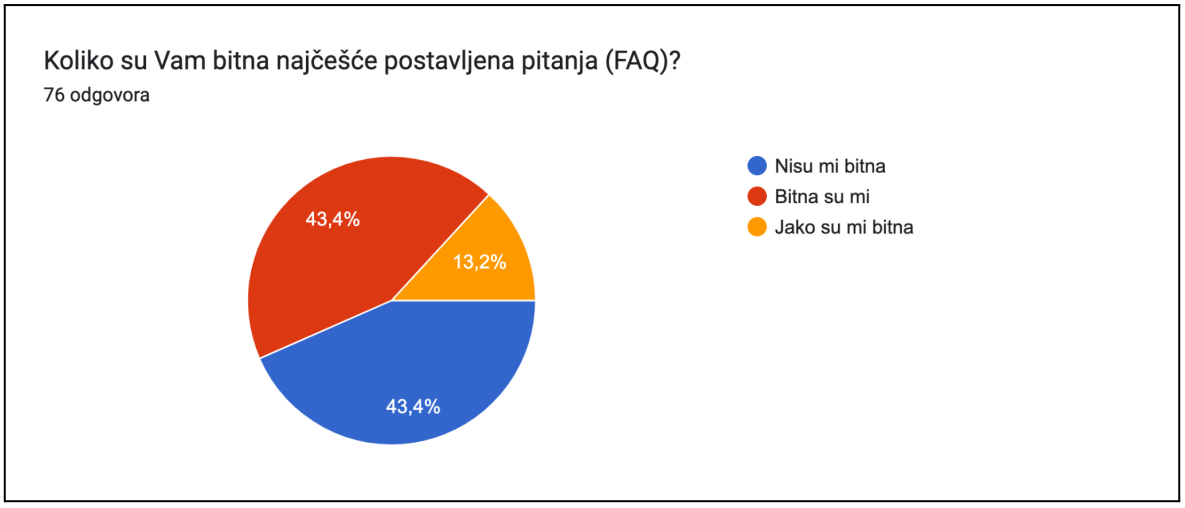
Slika 13. Grafički prikaz rezultata ankete za pitanje: Koliko Vam je bitna opcija sortiranja proizvoda?



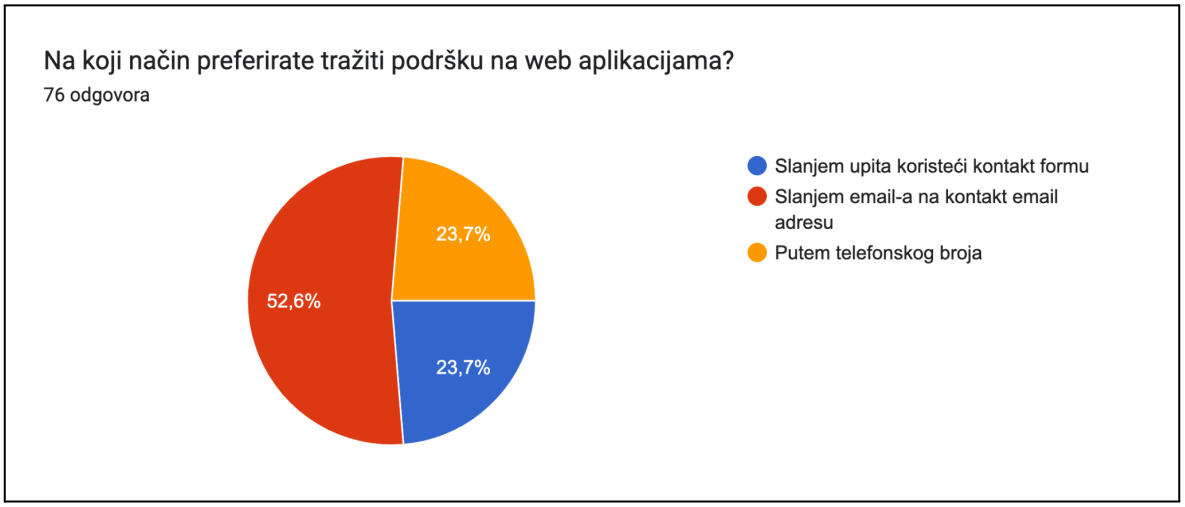
Slika 14. Prikaz sortiranja proizvoda.

Slika 13 pokazuje da je velikoj većini bitna ili jako bitna mogućnost sortiranja proizvoda. Sortiranje u web trgovinama ključan je aspekt web trgovine. Ako proizvod nije kompleksan, nema potrebe za naprednim filterima. Filteri omogućuju korisnicima da brzo pronađu proizvode koji najbolje odgovaraju njihovim potrebama. Na slici 14 je prikazan primjer sortiranja proizvoda po cijeni.

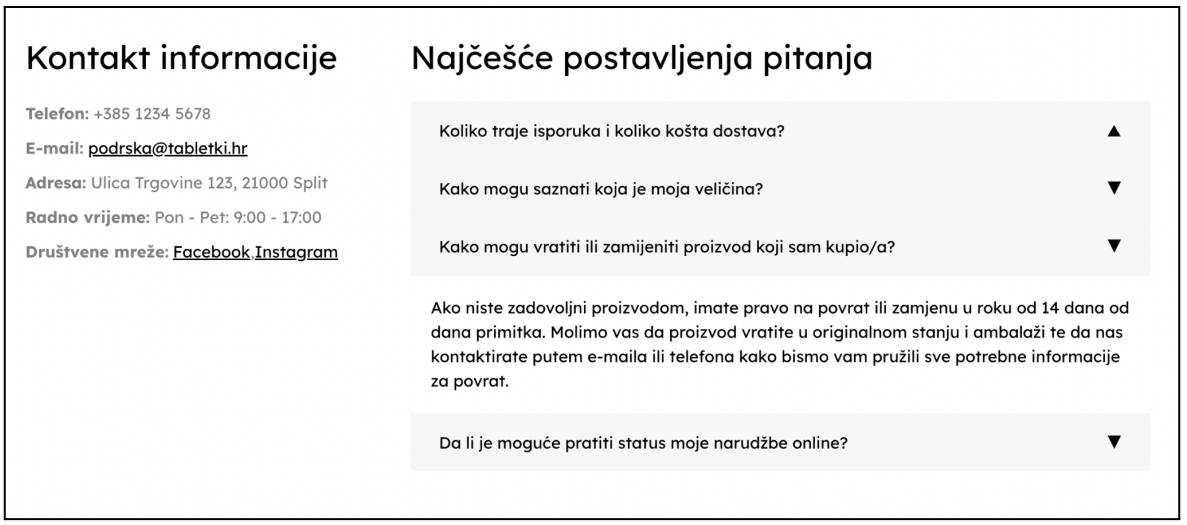
4.2.3. Najčešće postavljena pitanja i kontakt



Slika 15. Grafički prikaz rezultata ankete za pitanje: Koliko su Vam bitna najčešće postavljena pitanja?



Slika 16. Grafički prikaz rezultata ankete za pitanje: Na koji način preferirate tražiti podršku na web aplikacijama?



Slika 17. Primjer kontakt informacija i najčešće postavljenih pitanja.

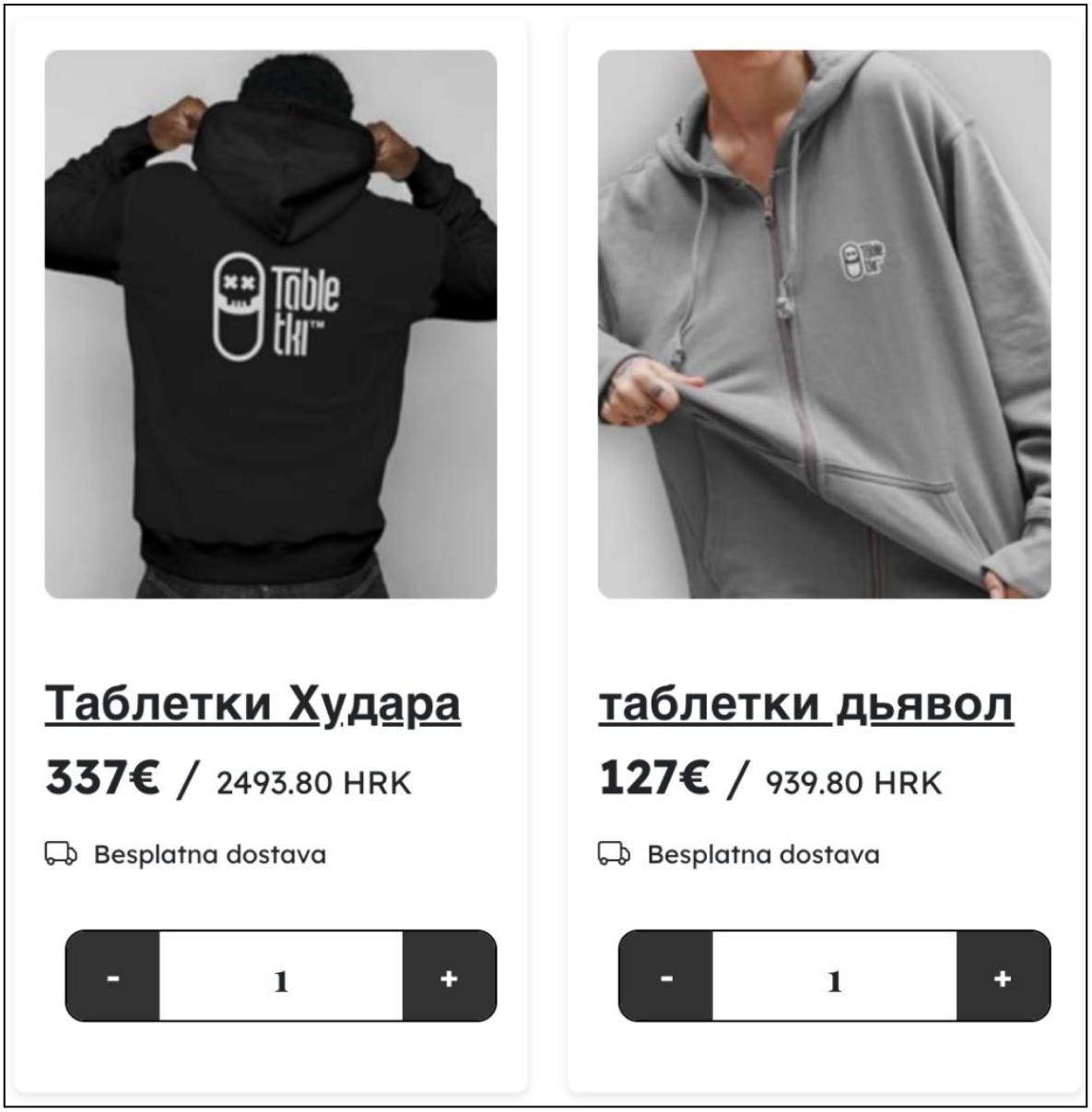
Na slici 15 vidljivo je da jednak postotak korisnika percipira FAQ (engl. *Frequently Asked Questions*) bitnima i nebitnima. Međutim, 13,2% korisnika ističe da su FAQ izuzetno bitna, zbog čega su integrirana u značajke web aplikacije. FAQ sekcija olakšava proces odlučivanja o kupnji. Transparentnost informacija u FAQ sekciji često povećava povjerenje korisnika u trgovinu i može poboljšati njihovo ukupno iskustvo. Osim toga, pružanjem jasnih odgovora smanjuje se potreba za kontaktiranjem službe za korisnike, što optimizira operativnu efikasnost trgovine. Dodatno, dobro strukturirana FAQ sekcija može pridonijeti boljoj vidljivosti trgovine na tražilicama. Na temelju rezultata ankete, jasno je da korisnici web aplikacija preferiraju tradicionalne metode komunikacije kada traže podršku.

Na slici 16 vidljivo je da većina, odnosno preko 50% ispitanika, preferira slanje e-maila na kontakt adresu kao metodu traženja podrške. U usporedbi s tim, samo petina korisnika preferira korištenje kontakt forme. Ovo može ukazivati na to da korisnici cijene jednostavnost i izravnost e-mail komunikacije. Slanje e-maila je radnja s kojom su mnogi već upoznati i osjećaju se ugodno koristeći je. S druge strane, kontakt forme mogu imati različite interakcije, validacije i korake koji mogu biti konfuzni ili zahtjevni za korisnike. Stoga, prilikom razvoja web aplikacija, važno je uzeti u obzir odabir korisnika kako bi se osigurala optimalna korisnička podrška i zadovoljstvo. Slika 17 prikazuje primjer najčešće postavljenih pitanja i kontakt informacija.

4.2.4. Cijene i troškovi dostave

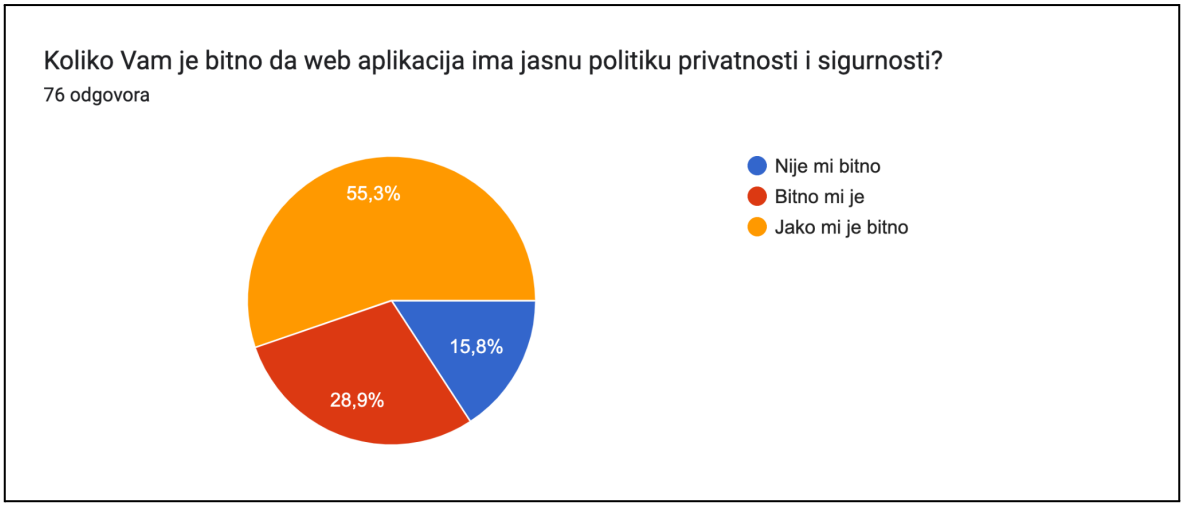
Slika 18. Grafički prikaz rezultata ankete za pitanje: Koliko Vam je bitno da web aplikacija ima jasno označene cijene i troškove dostave?

Na slici 18 vidljivo je da sveukupno 98,7% ispitanika smatra jasno označene cijene i troškove dostave bitnima ili jako bitnima. Transparentnost u prikazu cijena i troškova dostave na web aplikaciji značajno utječe na korisničko povjerenje i zadovoljstvo. Integriranjem jasnih informacija o cijenama uz proizvode, kao i detaljnom stranicom koja opisuje sve troškove, stvara se potpuna jasnoća prilikom online kupovine. Takva transparentnost smanjuje mogućnost nesporazuma i neugodnih iznenađenja prilikom finalizacije kupnje, čime se potiče korisnikova odluka o kupnji. Jasno označavanje svih troškova pridonosi osjećaju sigurnosti i pouzdanosti web aplikacije, što rezultira većom vjerojatnošću ponovne kupnje i preporuke drugima. Transparentnost u cijenama i troškovima dostave, prema rezultatima ankete, ključna je za izgradnju dugoročnog odnosa povjerenja između korisnika i trgovine. Na slici 19 je primjer jasno vidljivih cijena i troškova dostave.

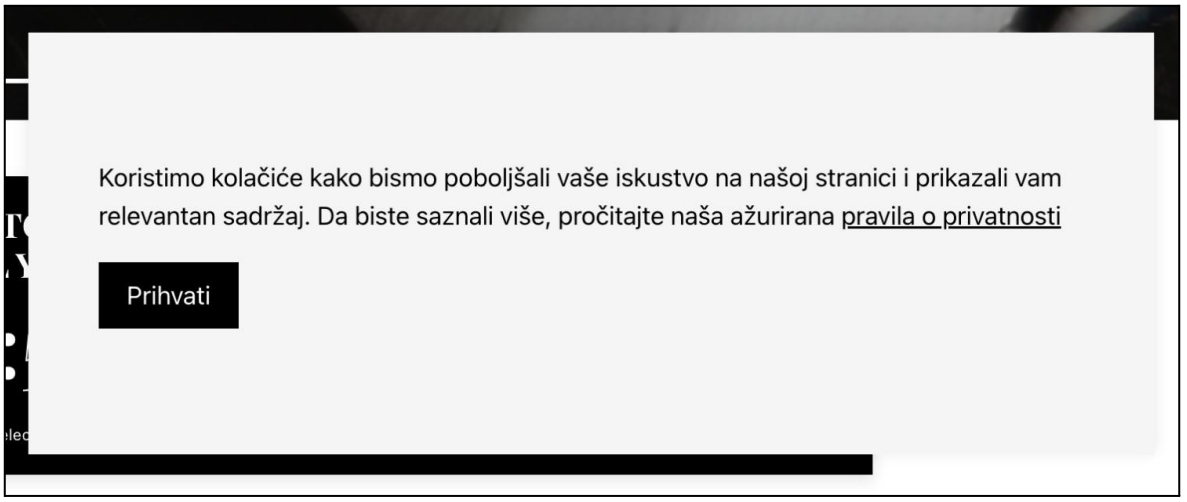


Slika 19. Primjer jasno vidljivih cijena i troškova dostave.

4.2.5. Politika privatnosti i sigurnosti



Slika 20. Grafički prikaz rezultata ankete za pitanje: Koliko Vam je bitno da web aplikacija ima jasnu politiku privatnosti i sigurnosti?

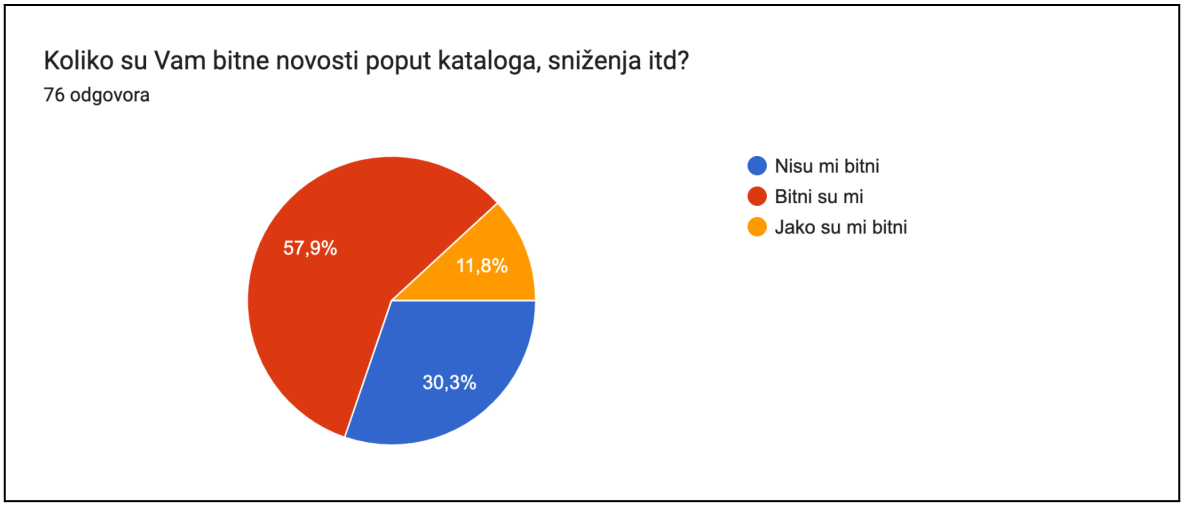


Slika 21. Skočni prozor koji informira korisnike o pravilima korištenja kolačića na web aplikaciji.

Slika 20 pokazuje kako više od pola ispitanika smatra jasnu politiku privatnosti i sigurnosti jako bitnom. Jasna politika privatnosti i sigurnosti je standard koji se očekuje od suvremenih web aplikacija i ključna komponenta koja gradi povjerenje korisnika. Kreiranjem stranica Politika i privatnost i Uvjeti korištenja, korisnicima se pruža uvid u to kako se njihovi podaci koriste, pohranjuju i štite. Transparentnost može rezultirati bržim donošenjem odluka od strane korisnika jer kada korisnik točno zna kako će njegovi podaci biti korišteni, manje je vjerojatno da će napustiti web stranicu ili aplikaciju. Usklađenost s međunarodnim standardima i regulativama, poput Opće uredbe o zaštiti podataka (GDPR, od engl. *General Data Protection Regulation*), postaje sve važnija jer web aplikacije koje

jasno komuniciraju svoje prakse u vezi s privatnošću i sigurnošću pokazuju da poštuju prava svojih korisnika i zakonske obveze. Pružanjem mogućnosti korisnicima da prihvate uvjete korištenja i politike kolačića, kao što je prikazano na slici 21, web aplikacija poboljšava korisničko iskustvo i potiče povjerenje i lojalnost korisnika.

4.2.6. Novosti

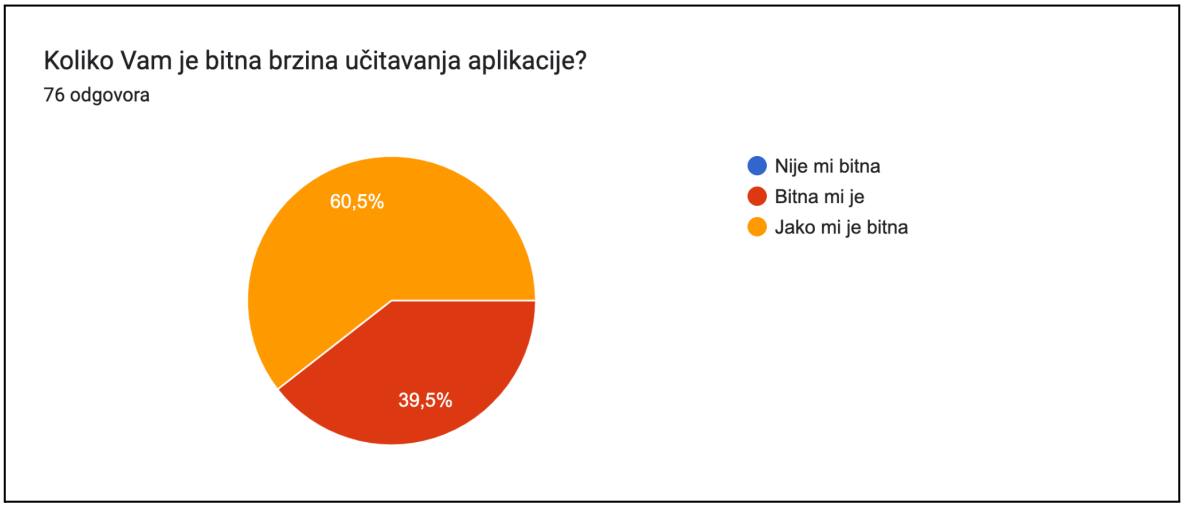


Slika 22. Grafički prikaz rezultata ankete za pitanje: Koliko su Vam bitne novosti poput kataloga, sniženja itd.?

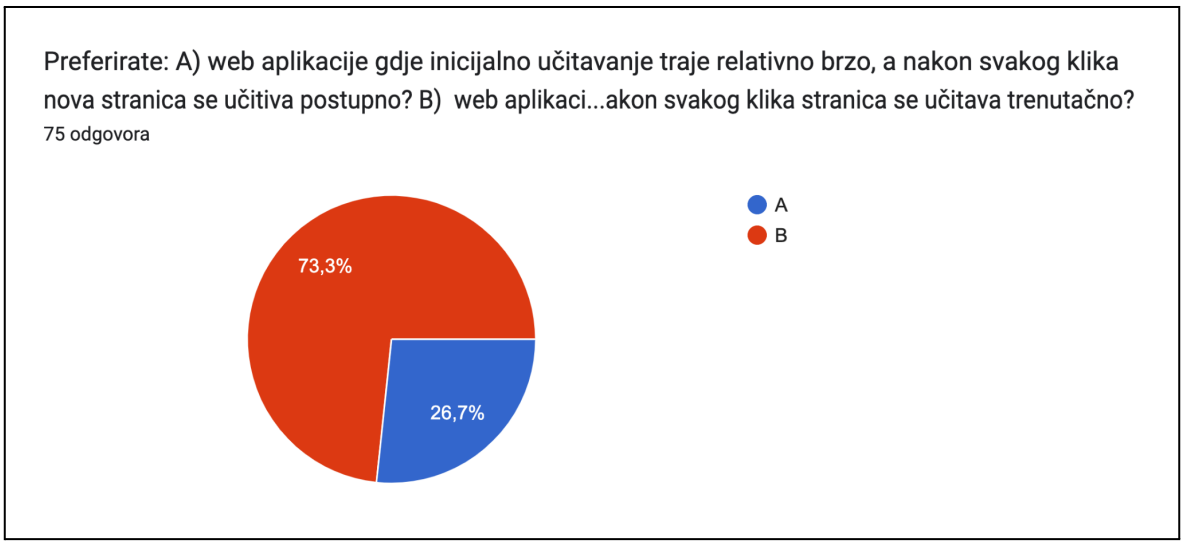
Slika 23. Primjer promocije.

Slika 22 pokazuje da većina korisnika preferira dobivati novosti u vezi sniženja, kao i kataloge. Promocije su ključne u web trgovinama jer motiviraju kupce da provode više vremena na web stranici. Osim pregledavanja proizvoda, korisnici često čitaju prateće tekstove i koriste dostupne alate na stranici. Ovo ne samo da povećava mogućnost prodaje, već i produbljuje odnos između korisnika i brenda. Kroz ovu interakciju, brend ima priliku dodatno se povezati s korisnicima, što može rezultirati povećanom lojalnošću i, na kraju, većom profitabilnošću. Kupci koji su lojalni brendu često postaju ambasadori brenda, preporučujući proizvode i usluge svojim prijateljima i obitelji, što može rezultirati organskim rastom baze kupaca. Promocije poput sniženja i posebnih ponuda, igraju ključnu ulogu u izgradnji i održavanju lojalnosti kupca. One pružaju osjećaj vrijednosti i ekskluzivnosti, sugerirajući kupcima da brend cijeni njihovu odanost i želi im pružiti najbolje moguće ponude. Primjer promocije modnog brenda prikazan je na slici 23.

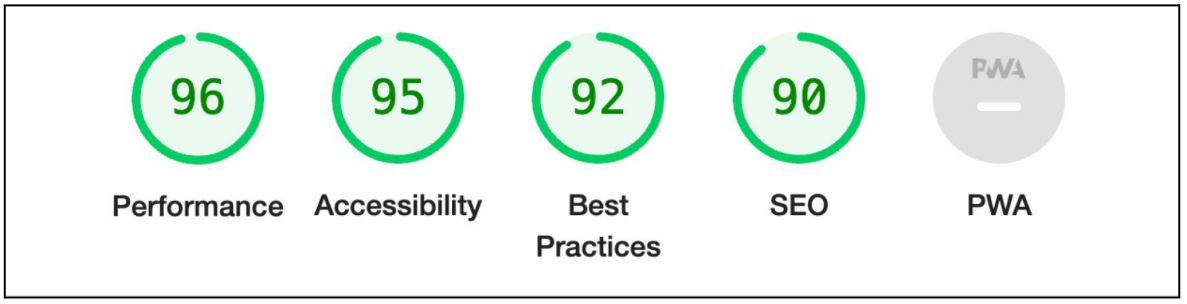
4.3. Performanse i brzina aplikacije



Slika 24. Grafički prikaz rezultata ankete za pitanje: Koliko Vam je bitna brzina učitavanja proizvoda?



Slika 25. Grafički prikaz rezultata ankete za pitanje: Preferirate li web aplikacije gdje inicijalno učitavanje traje relativno brzo, a nakon svakog klika nova stranica se učitava postupno? ili web aplikacije gdje inicijalno učitavanje traje sporije, a nakon svakog klika stranica se učitava trenutačno?



Slika 26. Prikaz rezultata Google Lighthouse za performanse, pristupačnost, najbolje prakse i optimizaciju za tražilice (SEO).

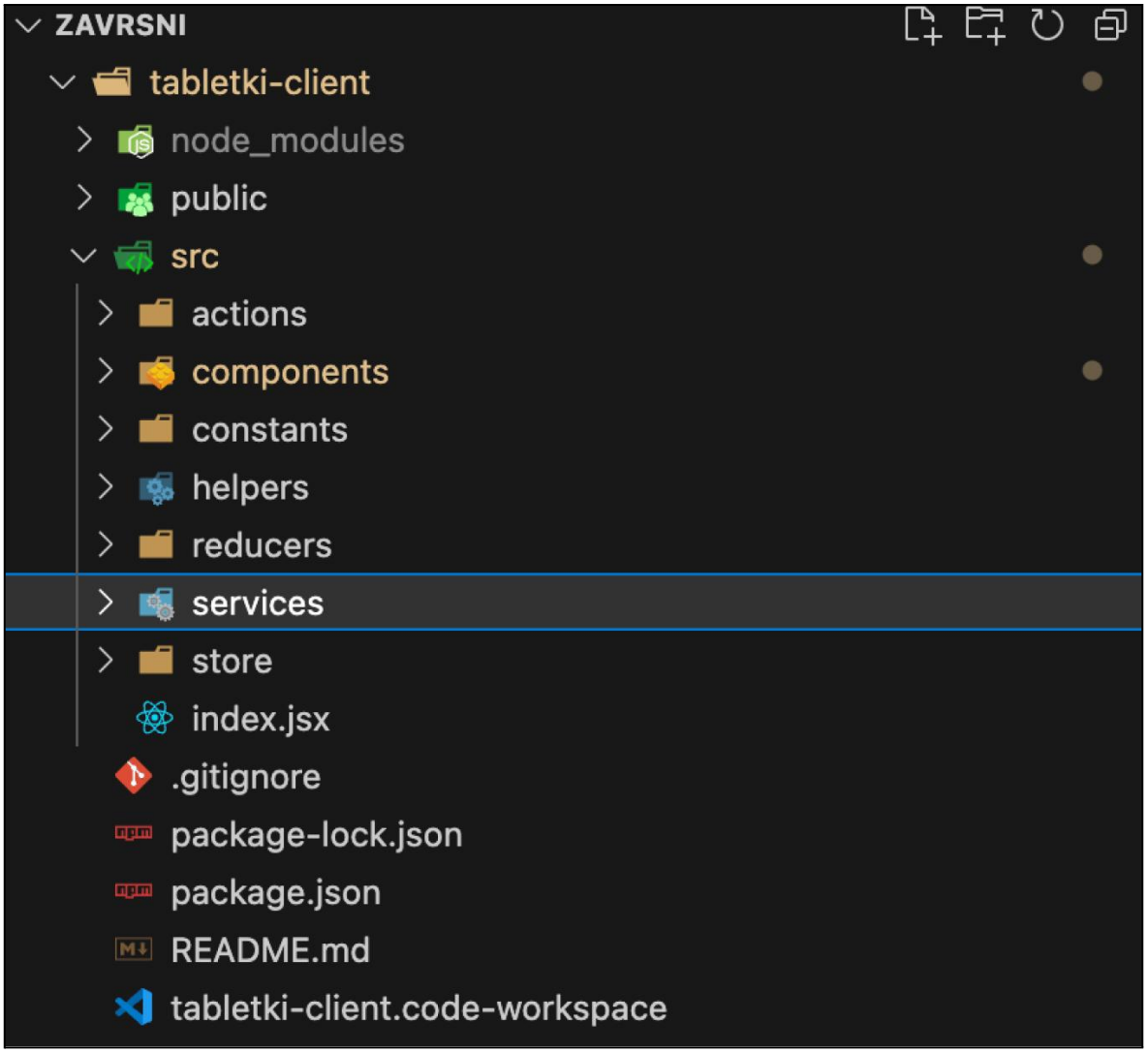
Svi ispitanici koji su sudjelovali u anketi smatraju brzinu učitavanja aplikacije bitnom ili jako bitnom, kao što je prikazano na slici 24. Slika 25 pokazuje da većina ispitanika preferira da inicijalno učitavanje web aplikacije traje sporije, a da se nakon svakog klika stranica učitava trenutačno. Korištenje tehnologija poput Reacta i Node.js omogućava aplikacijama da postignu iznimne performanse, posebno u pogledu brzine učitavanja (slika 26) Asinkrono učitavanje, koje je karakteristično za JavaScript, omogućava aplikacijama brzu obradu podataka i interakciju, čime se korisnicima pruža gotovo trenutačan odaziv. U svijetu web trgovina, brzina učitavanja stranica može biti ključna za zadržavanje korisnika i poticanje prodaje. Jednostranične aplikacije, koje su često izrađene pomoću Reacta, eliminiraju potrebu za učitavanjem novih stranica prilikom prelaska na drugi dio aplikacije, čime se dodatno poboljšava fluidnost korisničkog iskustva. Osim toga, optimizacija servera kroz Node.js smanjuje vremensko kašnjenje i

osigurava brzu obradu zahtjeva. Ukratko, kombinacija ovih tehnologija pruža korisnicima glatko i učinkovito korisničko iskustvo, dok istovremeno optimizira performanse aplikacije.

4.4. Aplikacija

U sljedećem paragrafu je opisan rad aplikacije kroz opis klijenta i poslužitelja aplikacije.

4.4.1. Klijent aplikacije



Slika 27. Prikaz strukture klijent direktorija.

Klijent aplikacije predstavlja korisničko sučelje aplikacije i odgovoran je za interakciju s korisnicima. Sastoji se od različitih komponenata i elemenata koji se iscrtavaju u pregledniku te komunicira s poslužiteljem kako bi dohvatio, prikazao ili poslao podatke. Kroz organiziranu strukturu mapa i datoteka, klijent upravlja različitim aspektima aplikacije, od korisničke autentifikacije do prikaza informacija i upravljanja sadržajem (slika 27.).

Glavna ulazna točka aplikacije na kojoj React iscrtava svoj sadržaj unutar `root` sekcije elementa je `Index.js`.

Mapa `Actions` sadrži metode koje šalju događaje koji će biti uhvaćeni u `redux reduceru`. `Reducer` je čista funkcija koja uzima trenutno stanje aplikacije i akciju kao argumente te vraća novo stanje. Glavna svrha `reducera` je da specificira kako će se stanje aplikacije promijeniti u odnosu na određenu akciju. U ovom slučaju koristi se za dohvaćanje podataka korisnika s poslužitelja i pohranu u globalno *`redux`* stanje kojem se može pristupiti iz komponenata.

Mapa `Constants` sadrži nepromjenjive objekte koje se koriste u aplikaciji.

Mapa `Reducers` sadrži globalni dokument koji je glavni držač stanja - u ovom slučaju stanja korisnika. Definira kako će određeni događaji utjecati na stanje i kako će podaci, preuzeti s poslužitelja, biti mapirani na globalno stanje.

Mapa `Auth` sadrži komponentu za usmjeravanje koja provjerava ima li trenutni korisnik, ako je prijavljen, specifična prava za pristup zadanoj ruti i posjeduje li korisnik validan token.

Mapa *`Components`* sadrži više mapa koji specificiraju odakle se komponenta koristi i za koju svrhu. Unutar mape se nalaze sve datoteke jsx (javascript XML). Na ispisu 1 prikazana je komponenta `Banner.jsx`, funkcionalna komponenta koja služi za prikaz niza slika u klizaču. Svaka slika u klizaču može imati poveznicu, naslov i opis. Ova komponenta je modularna i može se koristiti na različitim mjestima unutar aplikacije, samo joj je potrebno proslijediti odgovarajuće podatke. Funkcionalne komponente, poput ove, pružaju jasan i modularan način za izgradnju sučelja, čineći aplikaciju održivom i lako proširivom.

```
import React from "react";
import { Carousel } from "react-bootstrap";
import { Link } from "react-router-dom";
import "./Banner.scss";
const Banner = ({ id, className, data }) => {
  return (
    <Carousel id={id} className={className}>
      {data.map((item, index) => (
        <Carousel.Item key={index}>
          <Link to={item.to}>
            <img className="d-block w-100 h-100" src={item.img} alt={item.title} />
          </Link>
          {(item.title || item.description) && (
            <Carousel.Caption className="d-none d-md-flex">
              <h3>{item.title}</h3>
              <p>{item.description}</p>
            </Carousel.Caption>
          )}
        </Carousel.Item>
      ))}
    </Carousel> )};
export default Banner;
```

Ispis 1. Primjer funkcionalne komponente.

Mapa `Routing` sadrži komponentu usmjerivača koja definira sve komponente koje se iscrtavaju pod određenom putanjom i ulogom. Kada korisnik posjeti određenu putanju, metoda odlučuje koju komponentu treba prikazati na temelju definiranih ruta. Javne rute, koje su definirane pomoću komponente `Route` (ispis 2), dostupne su svim korisnicima i ne zahtijevaju posebne uvjete za pristup. Na primjer, stranice za prijavu ili registraciju obično su javne rute jer ih svaki posjetitelj može vidjeti. S druge strane, privatne rute, koje su definirane pomoću komponente `PrivateRoute`, zahtijevaju određene uvjete kako bi korisnik mogao pristupiti, poput provjere je li korisnik prijavljen. Ako korisnik ne ispunjava uvjete, obično je preusmjeren na drugu stranicu, poput stranice za prijavu. Privatne rute često se koriste za zaštitu osjetljivih dijelova aplikacije, poput korisničkog profila ili postavki.

```
function Routes() {
  return (
    <Router>
      <Switch>
        <Route exact path={routes.LOGIN} component={LoginPage} />
<Route exact path={routes.REGISTER} component={RegisterPage} />
        <Route path={routes.DASHBOARD} component={Dashboard} />
        <Route path={routes.NEWS} component={NewsPage} />
        <Route path={routes.SINGLE} component={NewsSingle} />
        <Route path={routes.SINGLE} component={NewsSingle} />
        <Route path={routes.TERMS} component={TermsSingle} />
<PrivateRoute path={routes.PROFILE} component={ProfilePage} />
      <PrivateRoute path={routes.CHECKOUT} component={CheckoutPage} />
      </Switch>
    </Router>
  );}
```

Ispis 2. Metoda koja definira sve komponente koje se iscrtavaju pod određenom putanjom i ulogom.

Mapa `Services` sadrži servise koji služe kao sredstvo za izolaciju i centralizaciju logike koja se odnosi na komunikaciju s vanjskim izvorima, poput API-ja (engl. *Application Programming Interface*). Kroz servise, aplikacije mogu komunicirati s bazama podataka, postavljati ili preuzimati podatke. Korištenjem servisa, aplikacije održavaju čistu i organiziranu strukturu kôda, gdje je svaka funkcionalnost jasno definirana i odvojena od ostalih dijelova aplikacije. Ispis 3 prikazuje servis za učitavanje slika proizvoda.

```
import { serviceHelper } from '../helpers';
const uploadProductImage = (imageFormData) => {
  const requestOptions = {
    method: 'POST',
    body: imageFormData,
  };
return
serviceHelper.authFetch(`${'http://localhost:9999'}/upload/upload-product-image`,
requestOptions).then(serviceHelper.handleResponse);
};

export const uploadService = {
  uploadProductImage
};
```

Ispis 3. Servisi za učitavanje slika proizvoda i korisnika.

Mapa `Helpers` sadrži sve pomoćne metode koje se ponovno koriste u aplikaciji ili su prevelike da bi bile unutar komponenata. Ispis 4 prikazuje pomoćne metode za rad na tokenu. Ovakav princip drži kôd čistim i upotrebljivim.

```
const getUserIdFromToken = () => {
  const token = getDecodedToken();
  if (!token) return null;
  const { user } = token;

  if (!user) return null;
  return user.id;
};

export const tokenHelper = {
  isTokenExpired,
  removeTokenFromLocalStorage,
  getTokenFromLocalStorage,
  setTokenToLocalStorage,
  getDecodedToken,
  getUserIdFromToken,
};
```

Ispis 4. Primjer metode i poziva svih metoda u TokenHelper.jsx datoteci.

Mapa `Store` u React aplikacijama s Reduxom igra ključnu ulogu u upravljanju i pohrani stanja aplikacije. Ispis 5 prikazuje `store.jx` dokument, centralno mjesto gdje se čuva cjelokupno stanje, omogućujući komponentama koordiniran i predvidljiv pristup i promjene. Umetanjem `loggera` omogućuje se praćenje i analiza akcija koje se šalju skladištu i njihov utjecaj na stanje.

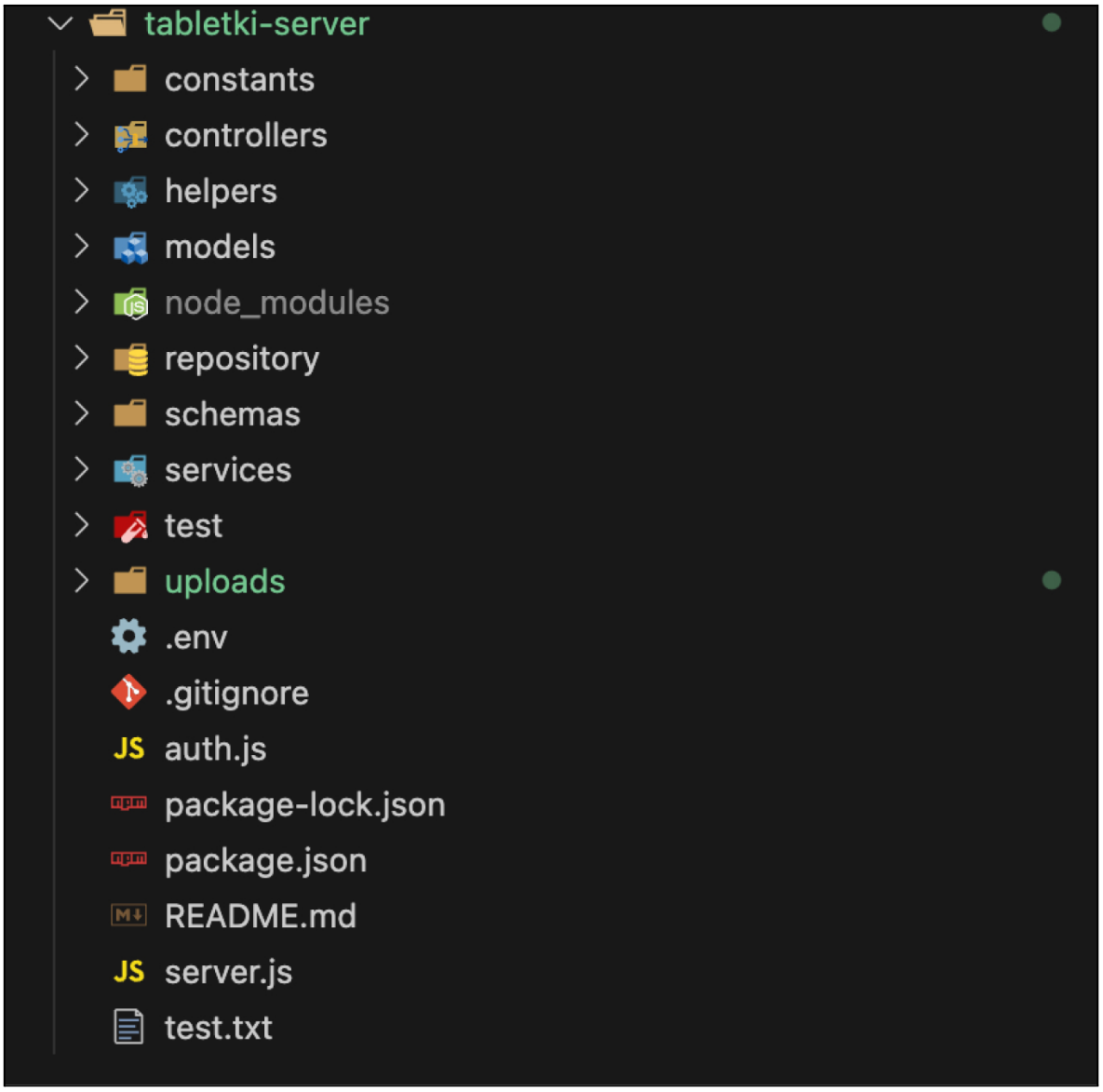
```
import thunkMiddleware from 'redux-thunk';
import logger from 'redux-logger';
import { configureStore } from '@reduxjs/toolkit';
import rootReducer from '../reducers';

export const store = configureStore({
  reducer: rootReducer,
  middleware: [thunkMiddleware, logger],
});
```

Ispis 5 - Primjer skladišta u Tabletki store.js datoteci.

4.4.2. Poslužiteljska aplikacija

Poslužiteljska aplikacija upravlja svim zahtjevima i podacima. Poslužitelj sluša dolazne zahtjeve s klijentske strane, obrađuje ih kroz definirane kontrolere, komunicira s bazom podataka te vraća odgovarajuće odgovore natrag klijentu (slika 28).



Slika 28. Prikaz strukture poslužitelj direktorija.

Ispis 6 prikazuje dokument `Server.js` predstavlja osnovnu konfiguraciju i inicijalizaciju poslužitelja u aplikaciji. Kao glavna ulazna točka poslužitelja, ona otvara vezu i sluša na određenom portu, spremna za obradu dolaznih zahtjeva. Kroz ovu datoteku,

aplikacija se povezuje s bazom podataka, definira `middleware`, postavlja rute i uključuje sve kontrolere koji upravljaju logikom obrade zahtjeva.

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const mongoose = require('mongoose');
const controllers = require('./controllers/controllers');
const routes = require('./constants/routes');
require('dotenv').config();
require('dotenv').config();
const server = express();
const port = process.env.PORT || 9999;
server.use(cors({
  origin: process.env.VM_APP_CLIENT_URL.toString(),
}));
mongoose.connect(process.env.VM_APP_ATLAS_SERVER_CONNECTION_STRING, {
  useNewUrlParser: true, useUnifiedTopology: true,
});
server.use(bodyParser.urlencoded({ extended: true }));
server.use(bodyParser.json());
server.use(routes.UPLOADS, express.static('uploads'));
server.use(routes.USER, controllers.userController);
server.use(routes.PRODUCT, controllers.productController);
server.use(routes.TEST, controllers.testController);
server.use(routes.UPLOAD, controllers.uploadController);
server.use(routes.FAQ, controllers.faqController);
server.get('/', (req, res) => {
  res.send('Tabletki app server');
});
server.listen(port, () => {
  console.log(`Express app has run on port: ${port}`);
});
```

Ispis 6. Server.js

Kontroleri predstavljaju središnji dio aplikacije koja koristi Node.js i Express okvir, odgovorni za obradu dolaznih zahtjeva i slanje odgovora Svaki kontroler je specijaliziran za određeni skup funkcionalnosti. Ispis 7 prikazuje metodu `FaqController` koja upravlja često postavljanim pitanjima, omogućujući kreiranje, čitanje, ažuriranje i brisanje informacija vezanih za ta pitanja. Metoda `UserController` upravlja svim funkcionalnostima vezanim za korisnike, uključujući registraciju, prijavu i autentifikaciju korisnika. Metoda `ProductController` omogućuje upravljanje informacijama o proizvodima, podržavajući osnovne operacije poput kreiranja, čitanja, ažuriranja i brisanja proizvoda. Metoda `UploadController` pruža funkcionalnost za prijenos medija, poput slika ili videozapisa, omogućujući korisnicima da dodaju i pristupe bogatim medijskim sadržajima. Metoda `TestController` koristi se za provjeru ispravnosti i funkcionalnosti poslužitelja, osiguravajući da sve radi kako je očekivano. Kroz ove kontrolere, aplikacija

može efikasno upravljati različitim aspektima poslovanja i pružiti korisnicima sve potrebne funkcionalnosti.

```
const express = require('express');
const protectedRouteAuth = require('../auth');
const roles = require('../constants/roles');
const faqService = require('../services/faqService');

const router = express.Router();

router.post('/add-faq', protectedRouteAuth[roles.SELLER], async (req, res) => {
  const { faq, err } = await faqService.addFaq(req.body);

  if (err) {
    return res.status(err.status).json({ error: err.message });
  }

  return res.status(200).json({
    faq,
  });
});

router.get('/get-faqs', async (req, res) => {
  const { faqs, err } = await faqService.getFaqs();

  if (err) {
    return res.status(err.status).json({ error: err.message });
  }

  return res.status(200).json({
    faqs,
  });
});

module.exports = router;
```

Ispis 7. Kontroler za kreiranje i dohvaćanje najčešćih pitanja.

Mapa `Helpers` sadrži metode koje se koriste za segregaciju logike iz kontrolera i servisa kako bi se postigao čišći kôd i metode koje se često ponavljaju kroz aplikaciju.

Mapa `Models` sadrži definicije modela koji se koriste za validaciju podataka koji dolaze na API (engl. *Application Programming Interface*) s klijentske strane. Kroz ove modele osigurava se da podaci koji se šalju na API odgovaraju očekivanoj strukturi i ispunjavaju određene uvjete. Ispis 8, prikazuje model podataka za proizvod koristeći AJV (engl. *Another JSON Validator*). Ovaj model specificira kako trebaju izgledati podaci za proizvod, definirajući tipove podataka i druge atribute za svako svojstvo proizvoda. Na primjer, svojstvo `productName` mora biti tekstualnog tipa s minimalnom duljinom od 3 znaka. Također, neka svojstva, poput `amount`, `cost` i `productName`, označena su kao obavezna, što znači da moraju biti prisutna kada se šalju podaci na API. Kroz ovakvu

validaciju osigurava se integritet podataka i sprječavaju potencijalne pogreške prilikom obrade podataka.

```
const productModel = {
  type: 'object',
  properties: {
    amount: { type: 'number', minimum: 1 },
    cost: { type: 'number', minimum: 0.01 },
    productName: { type: 'string', minLength: 3 },
    imageUrl: { type: 'string' },
    description: { type: 'string' },
    specialOffer: { type: 'boolean' },
  },
  required: ['amount', 'cost', 'productName'],
};

const productModels = {
  productModel
};

module.exports = { productModels };
```

Ispis 8. Model proizvoda.

Mapa `Repository` sadrži generičke CRUD (*engl. create, read, update, and delete*) Mongoose metode kako bi ažurirale bazu podataka MongoDB na Atlasu.

Mapa `Schemes` u kontekstu MongoDB-a predstavlja definicije struktura podataka koje aplikacija koristi. Ispis 9 sadrži shemu koja specificira izgled proizvoda kada se pohrani u bazi podataka MongoDB. Svako svojstvo proizvoda, poput `productName` ili `amount`, ima određeni tip podataka i druge atribute koji specificiraju njegove karakteristike, poput obaveznosti ili zadane vrijednosti. Korištenjem ove sheme, osigurava se da svaki proizvod koji se sprema u bazu podataka odgovara očekivanoj strukturi i sadrži sve potrebne informacije.

```
const mongoose = require('mongoose');

const productSchema = mongoose.Schema({
  productName: {
    type: String,
    required: true,
  },
  amount: {
    type: Number,
    required: true,
  },
  cost: {
    type: Number,
    required: true,
  },
  sellerId: {
    type: String,
    required: true,
  },
  specialOffer: {
    type: Boolean,
    required: false,
    default: false,
  },
  description: {
    type: String,
    required: false,
    default: "",
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
  imageUrl: {
    type: String,
    required: true,
  },});
const Product = mongoose.model('Products', productSchema);
module.exports = Product;
```

Ispis 9. Shema proizvoda.

Mapa `Uploads` služi kao spremište za medije koji su preneseni putem aplikacije. Kada korisnici prenose datoteke, poput slika ili dokumenata, putem metoda aplikacije te datoteke se spremaju lokalno na poslužitelju unutar mape poput `Uploads`. Ova mapa omogućuje centraliziranu pohranu svih prenesenih medija, olakšavajući njihovo upravljanje, pristup i distribuciju. Ispis 10 prikazuje `.env` dokument koji se koristi se za pohranu konfiguracijskih varijabli okruženja koje su specifične za aplikaciju. Ove varijable omogućuju konfiguraciju različitih aspekata aplikacije bez potrebe za izmjenom izvornog kôda. Pomoću datoteke `.env`, osjetljive informacije, poput tajnih ključeva ili lozinki, mogu se držati izvan izvornog kôda, što pridonosi sigurnosti aplikacije.

```
VM_APP_CLIENT_URL=http://localhost:3000
VM_APP_ATLAS_SERVER_CONNECTION_STRING=mongodb+srv://admin:admin@clustertabletki.nimpc
vt.mongodb.net/?retryWrites=true&w=majority
VM_APP_JWT_SECRET=sifra
PORT=9999
```

Ispis 10. Konfiguracijske varijable u `.env` dokumentu

6. ZAKLJUČAK

U ovom završnom radu, centralna tema bila je izrada web aplikacija s posebnim naglaskom na korisničko iskustvo. Detaljnom analizom i usporedbom rezultata ankete, pružen je duboki uvid u korisničke želje i očekivanja kada je riječ o dizajnu i funkcionalnostima web aplikacija. Istraživanje je obuhvatilo različite dobne skupine, što je omogućilo razumijevanje širokog spektra korisničkih potreba i želja. Jedan od ključnih zahtjeva istraživanja je identifikacija idealnog obrasca za izradu web trgovina. Ovaj obrazac, koji se može interpretirati kao trenutačni trend ili čak kao refleksiju ljudske prirode u kontekstu online kupovine, ukazuje na to da korisnici teže platformama koje su jednostavne za korištenje, intuitivne i efikasne. Osim toga, postoji jasna potreba za personalizacijom iskustva, što dodatno potvrđuje važnost dobro osmišljenog korisničkog iskustva. S tehnološke strane, rad se oslanjao na suvremene alate i tehnologije koji omogućuju kreiranje optimalnog UX-a, poput Reacta, JavaScripta i SCSS-a. Korištenje ovih tehnologija omogućava brži razvoj, veću prilagodljivost i bolju interakciju s korisnicima. U konačnici, ovaj rad ne samo da ističe važnost korisničkog iskustva kao ključnog faktora u dizajnu i razvoju web aplikacija, već i naglašava potrebu za stalnim prilagođavanjem i inovacijama. Tehnologija i korisničke ambicije brzo se mijenjaju stoga pružanje izvanrednog korisničkog iskustva postaje imperativ za uspjeh svake web aplikacije.

7. LITERATURA

[1] V. Sharma i A. K. Tiwari, „A study on user interface and user experience designs and its tools“. https://www.wjrr.org/download_data/WJRR1206016.pdf (pristupljeno 26. kolovoz 2023.).

[2] L. Marques *i ostali*, „Understanding UX Better: A New Technique to Go beyond Emotion Assessment“, *Sensors* , sv. 21, izd. 21, lis. 2021, doi: 10.3390/s21217183.

[3] M. Myre, „The UX Design process: The ultimate 8-step guide“, 24. ožujak 2019. https://designlab.com/blog/what-is-the-ux-design-process/ (pristupljeno 26. kolovoz 2023.).

[4] „19 Best UI/UX tools to empower your product design team“, *Maze*, 19. kolovoz 2021. https://maze.co/collections/ux-ui-design/tools/ (pristupljeno 26. kolovoz 2023.).

[5] J. Cardello, „17 best UI/UX design tools for the modern day designer in 2023“, *Webflow*, 26. listopad 2022. https://webflow.com/blog/ui-ux-design-tools (pristupljeno 26. kolovoz 2023.).

[6] C. Machado i J. C. Campos, „Towards the integration of user interface prototyping and model-based development“, u *2021 International Conference on Graphics and Interaction (ICGI)*, stu. 2021, str. 1–8.

[7] S. Melo, „Advantages and disadvantages of Google forms“, *DataScope*, 15. lipanj 2018. https://datascope.io/en/blog/advantages-and-disadvantages-of-google-forms/ (pristupljeno 26. kolovoz 2023.).

[8] S. B. Uzayr, N. Cloud, i T. Ambler, *JavaScript Frameworks for Modern Web Development: The Essential Frameworks, Libraries, and Tools to Learn Right Now*. Apress, 2019.

[9] N. G. Obbink, I. Malavolta, G. L. Scoccia, i P. Lago, „An extensible approach for taming the challenges of JavaScript dead code elimination“, u *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, ožu. 2018, str. 291–401.

[10] M. Pontus, „Maintain control: A Webpack/React tutorial, pt. 1“, *Toptal Engineering Blog*, 05. studeni 2019. https://www.toptal.com/react/webpack-react-tutorial-pt-1 (pristupljeno 26. kolovoz 2023.).

[11] „About npm“. https://docs.npmjs.com/about-npm (pristupljeno 26. kolovoz 2023.).

[12] A. P. Paramitha, „ReactJS with SASS/SCSS“, 30. lipanj 2022. https://www.mitrais.com/news-updates/reactjs-with-sass-scss/ (pristupljeno 26. kolovoz 2023.).

[13] „Coding ninjas studio“. https://www.codingninjas.com/studio/library/advantages-and-disadvantages-of-nodejs (pristupljeno 26. kolovoz 2023.).

[14] „ExpressJS - Overview“. https://www.tutorialspoint.com/expressjs/expressjs_overview.htm (pristupljeno 27. kolovoz 2023.).

[15] „MongoDB: An introduction“, *GeeksforGeeks*, 04. studeni 2015. https://www.geeksforgeeks.org/mongodb-an-introduction/ (pristupljeno 27. kolovoz 2023.).

[16] F. U. Zaman, M. A. Khuhro, K. Kumar, N. Mirbahar, M. Z. Khan, i A. Kalhoro, „Comparative case study difference between Azure cloud SQL and Atlas MongoDB NoSQL database citefactor.Org-journal“. https://www.citefactor.org/article/index/192855/comparative-case-study-difference-bet

ween-azure-cloud-sql-and-atlas-mongodb-nosql-database (pristupljeno 27. kolovoz 2023.).
[17] M. Vuorre i J. P. Curley, „Curating Research Assets: A Tutorial on the Git Version 2 Control System“.
[18] R. Crystal-Ornelas *i ostali*, „A guide to using GitHub for developing and versioning data standards and reporting formats“, *Earth Space Sci.*, sv. 8, izd. 8, kol. 2021, doi: 10.1029/2021ea001797.

Kristian Radić
Hercegovačka 32, 21000 Split
OIB: 45459290528

1. IZJAVA O AUTENTIČNOSTI ZAVRŠNOG RADA

Izjavljujem da sam završni rad pod nazivom

IZRADA WEB APLIKACIJA TEMELJEM KORISNIČKIH DOŽIVLJAJA

izradio/la samostalno.

Svi dijelovi rada rezultat su isključivo mojega vlastitog rada i temelje se na mojim istraživanjima. Dijelovi rada koji su citirani iz različitih izvora jasno su označeni kao takvi te navedeni u fusnoti i literaturi.

Split,

Ime i Prezime