Web aplikacija za promociju brandova

Topić, Kristijan

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

„Dr. Mijo Mirković“

Kristijan Topić

Web aplikacija za promociju brendova

Diplomski rad

Pula, 2022.

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

„Dr. Mijo Mirković“

Web aplikacija za promociju brendova

Diplomski rad

Kristijan Topić

JMBAG: 0303064652, redovan student

Studijski smjer: Informatički menadžment

Kolegij: Web tehnologije

Mentor: dr. sc. Goran Matošević

Pula, rujan 2022.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika _____ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom _____ _____ _____

koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

SADRŽAJ

1. UVOD

U ovom diplomskom radu predstavljen je proces razvoja web aplikacije nazvane „Promotezone“ korištenjem Vue JS i Firebase tehnologija. Svrha aplikacije je omogućiti brendovima i promotorima lakše spajanje i pronalaženje jedni drugih prema raznim kategorijama i parametrima te međusobnu komunikaciju preko live chata.

Unutar ovog rada pobliže će biti objašnjena aplikacija „Promotezone“, njene funkcionalnosti i arhitektura te korištene tehnologije. Bit će objašnjeno što su to uopće web aplikacije i kako one funkcioniraju te koji je proces razvoja web aplikacija potpomognut sa nekim primjerima iz procesa razvoja aplikacije Promotezone. Također, bit će objašnjen koji je to značaj marketinga za poslovanje, bit će riječi o procesu stvaranja brenda i o promociji preko društvenih mreža koja je danas postala vrlo popularna.

Promotezone aplikacija je zamišljena kao alat za marketing preko društvenih mreža koji je danas veoma popularan i nekim brendovima dobar dio poslovanja dolazi upravo zavaljujući njemu. Pojavom društvenih mreža, prije svega Instagrama otvorile su se nove prilike za marketing brendova i došlo je do pojave tzv. influencera, ljudi koji imaju moć utjecati na tuđe odluke o kupnji.

2. ZNAČAJ MARKETINGA ZA POSLOVANJE

2.1. BITNE DEFINICIJE

Prema definicijama od strane AMA-e (American Marketing Association) marketing je aktivnost, skup institucija i procesa za stvaranje, komuniciranje, isporuku i razmjenu ponuda koje imaju vrijednost za kupce, klijente, partnere i društvo u cjelini. (Odobreno 2017.).[1]

Robna marka ili brend je naziv, pojam, dizajn, simbol ili bilo koje drugo obilježje koje identificira robu ili uslugu jednog prodavača kao različite od onih drugih prodavača.[2]

Prema ISO standardima robna marka ili brend je nematerijalna imovina koja je namijenjena stvaranju prepoznatljivih slika i asocijacija u umovima dionika, čime se generiraju ekonomske koristi/vrijednosti."[3]

U svojoj suštini marketing je disciplina koja uključuje sve radnje koje neka kompanija poduzima kako bi privukla nove klijente i održavala odnos s njima. Marketing dakle nastoji povezati proizvode i usluge koje neka kompanija nudi, sa potencijalnim kupcima tih proizvoda ili usluga što na kraju osigurava profit za tu kompaniju.

[1] AMA: „What is Marketing? – The Definition of Marketing“ 2022. https://www.ama.org/the-definition-of-marketing-what-is-marketing/#:~:text=Marketing%20is%20the%20activity%2C%20set,Approved%202017) -13.8.2022.
[2] AMA: „What is Marketing? – The Definition of Marketing“ 2022. – 13.8.2022.
[3] ISO: „What 's in a Brand? Quite a bit, actually“ 2020. https://www.iso.org/news/ref2486.html - 13.8.2022.

2.2. ČETIRI BITNE ZNAČAJKE MARKETINGA

Profesor Niel H. Borden je 1950-ih godina popularizirao ideju marketing miksa koja govori da je za sveobuhvatni marketinški plan potrebno obratiti pažnju na nekoliko bitnih područja. Njegova ideja se uobičajeno prevodi u tzv. 4P marketinga (engl. *product*, *price*, *place* i *promotion*).

2.2.1. Proizvod

Proizvod se odnosi na dobra ili usluge koje kompanija planira nuditi svojim kupcima. Proizvod treba nastojati ispuniti određenu potrebu ili manjak na tržištu. Prije nego se proizvod plasira na tržište kompanija, i ljudi unutar kompanije zaduženi za marketing, moraju istražiti tržište, razumjeti svoj proizvod i na koji se on način ističe od konkurencije kako bi mogli stvoriti efektivnu marketinšku strategiju.

2.2.2. Cijena

Cijena određuje za koliko će se novca proizvod ili usluga prodavati kupcu. Prilikom određivanja cijene, kompanije moraju uzeti u obzir cijenu potrebnu za proizvodnju proizvoda, utrošak energije, cijenu radne snage, skladištenje proizvoda te marketinške, distribucijske i mnoge druge troškove kako bi ostvarili profit. Također, kompanije moraju uzeti u obzir i cijene konkurentnih proizvoda kako bi se njihov proizvod ili usluga etablirali kao razumna alternativa na tržištu.

2.2.3. Mjesto

Potrebno je odrediti distribucijsko mjesto proizvoda. Hoće li to biti fizička trgovina, online ili oboje? Koja će strategija donijeti najviše ovisi o mnogim čimbenicima poput fizičke

lokacije trgovine. Da li je na određenom mjestu moguće doći do dovoljnog broja ciljanih kupaca? Potrebno je proanalizirati sve čimbenike i donijeti odluku.

2.2.4. Promocija

Zadnji P se odnosi na promociju. Potrebno je odrediti marketinšku strategiju kojom se želi upoznati kupce sa proizvodom ili uslugom. Ta će strategija uvelike ovisiti o vrsti proizvoda i usluga i shodno tome kompanije će donositi odluke.

2.3. VRSTE MARKETINGA

Vrste marketinga možemo podijeliti na tradicionalne marketinške strategije te moderni pristup u vidu digitalnog marketinga.

2.3.1. Tradicionalne marketinške strategije

Prije postojanja interneta, kompanije su se služile tradicionalnim načinima promoviranja svojih proizvoda i usluga, a to su:

1. Vanjski marketing – izlaganje promotivnog materijala u javnom prostoru, poput plakata, reklama na klupama ili javnom prijevozu.
2. Print marketing – strategija printanja i dijeljenja velikog broja promotivnog materijala kao što su letci brošure i sl.
3. Direktni marketing – davanje promotivnog materijala ciljanim specifičnim kupcima poput kupona, bonova te pamfleta.
4. Elektronički marketing – reklamiranje putem televizije ili radija koje može biti puno učinkovitije od primjerice print marketinga jer lakše obuzima pažnju gledatelja ili slušatelja.

5. Marketing događaja – okupljanje potencijalnih kupaca na nekom organiziranom događaju u svrhu demonstracije proizvoda, to mogu biti konferencije, sajmovi, seminari i sl.

2.3.2. Digitalni marketing

Pojavom interneta otvorile su se brojne nove mogućnosti za promoviranje proizvoda i usluga. Tako je nastao pojam digitalnog marketinga ili online marketinga. Neki od inovativnih strategija marketinga koje su nastale pojavom interneta su:

1. Marketing na tražilicama – kompanije nastoje poboljšati svoje rangiranje na tražilicama tako što će im plaćati za bolji položaj ili će se poslužiti tzv. SEO tehnikom (*Search Engine Optimization*) kako bi njihove stranice organski bile bolje rangirane. SEO tehnika podrazumijeva načine koncipiranja stranice na optimalan način kako bi bila razumljivija algoritmima koji je pretražuju i rangiraju.
2. E-mail marketing – kompanije na razne načine sakupljaju e-mailove svojih kupaca i potencijalnih kupaca te putem e-mail kampanja šalju razne promotivne materijale poput newslettera, kupona, informacija o popustima i budućim rasprodajama i sl.
3. Marketing preko društvenih mreža – kompanije nastoje izgraditi snažnu prisutnost na društvenim mrežama ne bi li povećale svijest o svom brendu te plaćaju oglase koji će se prikazivati na tim društvenim mrežama određenim ciljanim korisnicima.
4. Influencer marketing - danas vrlo popularan tip marketinga i u kontekstu ovog rada najznačajniji. Stavlja se naglasak na sakupljanje ljudi koji imaju veliki broj pratitelja na društvenim mrežama putem kojih oni promoviraju odnosno utječu na ljude da kupuju određene proizvode. Ti se ljudi nazivaju „influenceri“ i oni od promotivnih radnji mogu zaraditi dobre svote novaca bilo preko provizije od prodaje ili preko ugovorene mjesečne naknade, ovisno o tome koliku praćenost imaju. To brendovima omogućuje da se prošire na šire tržište. Influenceri mogu biti bilo tko, slavne osobe, stvaratelji sadržaja, obični kupci i sl.
5. Marketing sadržaja – sve više kompanija se oslanja na stvaranje sadržaja kako bi se povezali sa svojim kupcima i podijelili priču o svom brendu. Taj sadržaj može

biti bilo koje besplatno digitalno dobro poput video seminara, blogova, e-knjiga, infografika i sl. Cilj je napraviti proizvod, najčešće besplatni koji će upoznati ljude sa brendom i pretvoriti ih u kupce.

2.4. VAŽNOST MARKETINGA

Vrlo je važno konstruirati dobru marketinšku strategiju jer ona kompaniji odnosno njenom brendu može donijeti mnoge dobre stvari. To nije lak zadatak i iziskuje mnogo rada i iskušavanja brojnih pristupa kako bi se strategija dovela do optimizacije. Dobro isplanirana marketinška strategija kompaniji može donijeti:

1. Stvaranje publike – kompanije marketingom nastoje povezati i predstaviti svoj proizvod i poruku svog brenda određenoj ciljanoj publici.
2. Unutarnje obrazovanje – iskušavanjem brojnih marketinških strategija kompanije prikupljaju podatke te dolaze do zaključaka koje su strategije za koje proizvode najučinkovitije.
3. Vanjsko obrazovanje – dobra marketinška kampanja će nastojati obrazovati svoju publiku o svom proizvodu ili usluzi kao i o samoj kompaniji te vrijednostima kojima se ona vodi te prenijeti poruku o svom brendu i objasniti zašto će baš njihovi proizvodi ili usluge obogatiti živote ciljanih potrošača.
4. Stvaranje brenda – nastoji se stvoriti jedinstvena slika i obilježja specifična za proizvode ili usluge, stvaranje jedinstvene priče oko brenda i nastojanje izazivanja specifičnih osjećaja i reakcija potrošača na taj brend. Na neki način nastoji se oblikovati mišljenje javnosti o tom brendu prije nego što javnost uopće ima priliku stupiti u interakciju sa tim brendom i stvoriti svoju sliku.
5. Dugotrajnost – dobra marketinška kampanja nastoji ostaviti dugotrajni utisak u mišljenjima javnosti. Nastoji biti upečatljiva te ostati u sjećanjima potrošača što je duže moguće.
6. Financijska izvedba – krajnji cilj i mjerilo uspješnosti marketinške kampanje je njen financijski rezultat, odnosno koliko je ona donijela novih kupaca i generirala prihoda. Kada se kompanija uspješnom marketinškom kampanjom poveže s

kupcima i ostvari dobar odnos s njima, veća je vjerojatnost da će se ti isti kupci vraćati i ostati vjerni njenom brendu.

3. STVARANJE BRENDA

Pojam brenda se odnosi na poslovni i marketinški koncept koji pomaže ljudima identificirati određenu kompaniju, proizvode ili individuu. Brend nije nešto materijalno i opipljivo već je to cjelokupna priča vrijednosti kojima se kompanija vodi i nešto što pomaže ljudima stvoriti mišljenje o toj kompaniji i njenim proizvodima ili uslugama. Ljudi često miješaju pojam brenda sa logotipima, sloganima i ostalim prepoznatljivim oznakama neke kompanije. Ti su pojmovi usko povezani, no ipak se razlikuju. Logotipi, slogani i slične oznake su marketinški alati kojima se kompanije koriste za promociju svojih proizvoda. Svi ti pojmovi zajedno stvaraju jedinstveni identitet brenda.

Brendovi se često služe jedinstvenim oznakama kako bi se razlikovali od konkurencije i kako bi se stvorio jedinstveni identitet tog brenda. To brendovima može donijeti prednost u odnosu na konkurentne proizvode te zato brendovi nastoje zakonski zaštititi svoje jedinstvene oznake putem tzv. trademarka kako ih drugi ne bi mogli kopirati.

Trademark ili na hrvatskom zaštitni znak se odnosi na neku prepoznatljivu oznaku, frazu, riječ ili simbol koji označava specifični brend ili proizvod i zakonski ga razlikuje od konkurencije. Putem trademarka kompanije zakonski identificiraju svoje brendove i proizvode kao intelektualno vlasništvo. Često možemo vidjeti na određenim logotipima ili sloganima nekih brendova malu oznaku TM koja se odnosi upravo na trademark.

Trademarkovi pomažu razlikovati određene proizvode u pravnim i poslovnim okvirima kao i u očima potrošača. Vrlo su značajni jer se njima zaštićuju određene riječi i grafički elementi koji identificiraju brend. To mogu biti logotipi, slogani ili imena brendova i proizvoda. Korištenjem trademarka kompanije osiguravaju da ostale kompanije ne mogu koristiti njihovo intelektualno vlasništvo bez njihovog dopuštenja. Također sprječavaju korištenje oznaka koje imaju određenu sličnost kao i riječi koje imaju sličan prizvuk ili

značenje sa njihovim trademarkom, kako bi se izbjegla bilo kakva konfuzija potrošača, bilo namjerna ili nenamjerna.

3.1. BREND KAO KAPITAL

Kada kompanija želi stvoriti svoj jedinstveni brend prvo mora odrediti na koji način ona želi biti percipirana od strane okoline. U taj proces ulazi stvaranje jedinstvenog izgleda i osjećaja brenda, jedinstvenih logotipa i slogana kompanije putem koji se želi poručiti poruka kompanije ili vrijednosti kompanije koje ona zastupa. Cilj je napraviti da brend bude što privlačniji i upečatljiviji za potrošača kako bi se on istaknuo od konkurencije, te ultimativno, kako bi se povećala njegova prodaja.

Za osmišljavanje vizualne reprezentacije svog brenda kompanije uobičajeno unajmljuju dizajnerske firme. Dobra vizualna reprezentacija brenda rezultirat će o svjesnosti potrošača o tom brendu i njegovoj poruci.

Velika svijenost potrošača o brendu dovest će do njegove prepoznatljivosti. Stvaranje svjesnosti potrošača o brendu je ključan korak za kreiranje marketinške kampanje. Proizvodi i usluge koji održavaju veliku razinu svijenosti potrošača o svom brendu imaju veću vjerojatnost generiranja prodaje. Potrošači će prije kupiti proizvod za koji znaju, proizvod za koji su čuli u odnosu na onaj koji im je nepoznat.

Kada kompanija uspješno stvori pozitivan utisak kod ciljanih potrošača tada možemo reći kako ona ima brend. Taj brend služi kompaniji kao kapital i rezultirat će povećanjem prodaje ne samo jednog određenog proizvoda, već i za sve druge proizvode koje prodaje ista kompanija. Dobar brend stvara povjerenje potrošača, a nakon dobrog iskustva s jednim proizvodom, vjerojatnije je da će potrošač probati drugi proizvod koji se odnosi na isti brend. Ovaj se fenomen često naziva lojalnost brendu.

Kupci voljno plaćaju visoku cijenu za proizvode nekih brendova, iako bi istu stvar mogli dobiti od nekog drugog za manje novaca. Kupci tom brendu pridaju određenu razinu

kvalitete ili prestiža te proizvode tog brenda smatraju vrjednijima od proizvoda konkurencije, pa su spremni platiti više. Kupci zapravo plaćaju premiju za poslovanje s tvrtkom koju poznaju i kojoj se dive. Budući da tvrtka s uspješno stvorenim brendom ne snosi veće troškove od svojih konkurenata za proizvodnju proizvoda i njihovo plasiranje na tržište, razlika u cijeni odlazi na njihovu stranu. Vrijednost brenda kompanije omogućuje joj da ostvari veći profit pri svakoj prodaji.

Kapital brenda ima tri osnovne komponente: percepciju potrošača, negativne ili pozitivne učinke i rezultirajuću vrijednost. Prije svega, percepcija potrošača, koja uključuje i znanje i iskustvo s brendom i njegovim proizvodima, gradi vrijednost tog brenda. Percepcija koju potrošači imaju o robnoj marki izravno rezultira pozitivnim ili negativnim učincima. Ako je kapital brenda pozitivan, kompanija, njezini proizvodi i financije mogu imati koristi. Ako je kapital brenda negativan, vrijedi suprotno.

Konačno, ti se učinci mogu pretvoriti u materijalnu ili nematerijalnu vrijednost. Ako je učinak pozitivan, opipljiva vrijednost se ostvaruje kao povećanje prihoda ili dobiti. Nematerijalna vrijednost se u marketingu ostvaruje kao svijest potrošača o brendu. Ako su učinci negativni, materijalna ili nematerijalna vrijednost je također negativna. Na primjer, ako su potrošači spremni platiti više za generički proizvod nego za brendirani proizvod onda brend ima negativan kapital. To se može dogoditi ako tvrtka ima veliki opoziv proizvoda ili primjerice izazove ekološku katastrofu o kojoj se naširoko govori.

Kapital brenda ima izravan učinak na obujam prodaje jer potrošači gravitiraju prema proizvodima s velikom reputacijom. Primjerice, kada Apple izdaje novi proizvod, kupci stoje u dugim redovima ispred trgovine kako bi ga kupili iako je njegova cijena obično viša od sličnih proizvoda konkurenata. Jedan od primarnih razloga zašto se Appleovi proizvodi prodaju u tako velikom broju je taj što je tvrtka prikupila zapanjujuću količinu pozitivnog kapitala marke. Budući da je određeni postotak troškova tvrtke za prodaju proizvoda fiksan, veći obujam prodaje znači veće profitne marže.

Zadržavanje kupaca treće je područje u kojem kapital brenda utječe na profitne marže. Vraćajući se na primjer Applea, većina kupaca tvrtke ne posjeduje samo jedan Appleov proizvod, oni ih posjeduju nekoliko. Osim toga, željno iščekuju sljedeće izdanje. Appleova baza kupaca je vrlo lojalna, ponekad na granici fanatizma. Apple uživa visoku stopu

zadržavanja kupaca, što je još jedan rezultat vrijednosti kapitala njegovog brenda. Zadržavanje postojećih kupaca povećava profitne marže smanjenjem iznosa koji tvrtka mora potrošiti na marketing kako bi postigla isti obujam prodaje. Manje košta zadržavanje postojećeg kupca nego stjecanje novog.

Stvaranje brenda pruža brojne prednosti, bilo da se radi o korporaciji ili pojedincu. Uspješno brendiranje dovodi do stvaranja određenih dojmova kod potrošača. Tvrtka koja može prenijeti svoju poruku sposobna je inducirati emocije kod svojih kupaca. Oni razvijaju jedinstvene odnose s tim kompanijama, omogućujući im da kapitaliziraju njihovu lojalnost. Kompanije se također oslanjaju na te kupce kako bi im pomogli privući druge, nove kupce.

4.

4. DRUŠTVENE MREŽE I NJIHOVA ULOGA U PROMOCIJI BRENDOVA

Nastajanjem društvenih mreža pojavile su se nove mogućnosti i kanali putem kojih se kompanije i njihovi brendovi mogu promovirati. Više od 80% potrošača izjavili su da društvene mreže, pogotovo sadržaj tzv. „influencera“, značajno utječu na odluke o kupnji. Marketinški stručnjaci u svim industrijama pokreću evoluciju marketinga na društvenim mrežama koji postaje sve važniji kanal, usmjeren na sve važniju i sve više rastuću publiku.

Od pojave MySpace-a, prve društvene mreže koja je dosegla razinu od milijun korisnika, nastale sada već davne 2004 godine, prošlo je 18 godina. Dramatičan rast interaktivnih digitalnih kanala odveo je društvene medije do razina koje izazivaju čak i doseg televizije i radija. Do prvog tromjesečja 2022. globalno je bilo 4,6 milijardi korisnika društvenih mreža što je preko 58% svjetske populacije. Dakle više od pola svjetske populacije se koristi nekom društvenom mrežom. Godišnje se taj broj u prosjeku povećava za 5.1% odnosno za oko 227 milijuna ljudi. Prosječni korisnik dnevno provede 2 sata i 29 minuta koristeći se nekom društvenom mrežom.

Nakon svega izrečenog, nije ni čudo da kompanije sve više usmjeravaju svoje marketinške strategije prema društvenim mrežama. To je ogroman kanal koji kompanije nastoje iskoristiti za promociju svog brenda i stjecanje prednosti u odnosu na konrurenciju. Društvene mreže to koriste te na neki način zapravo prodaju pažnju i vrijeme svojih korisnika kompanijama koje se žele promovirati na njihovom kanalu.

4.1. SNAGA MARKETINGA NA DRUŠTVENIM MREŽAMA

Velikim korištenjem društvenih mreža javlja se ogromni potencijal za marketing i za izgradnju brenda kompanija, povećanje prodaje i povećanje prometa na njihovim web stranicama. Osim što kompanijama pruža način da stupe u kontakt s postojećim kupcima i dođu do novih, marketing na društvenim mrežama ima namjenski izrađenu analitiku podataka koja marketinškim stručnjacima omogućuje praćenje uspjeha njihovih napora i identificiranje još više načina za uključivanje novih korisnika.

Snaga marketinga na društvenim mrežama potaknuta je njegovim velikim potencijalom u tri ključna marketinška područja:

1. Povezivanje - društvene mreže ne samo da omogućuju kompanijama da se povežu s kupcima na načine koji su prije bili nemogući, već postoji i mnogobrojan niz načina za povezivanje s ciljanom publikom.
2. Interakcija - dinamična priroda interakcije na društvenim mrežama, bilo da se radi o izravnoj komunikaciji ili pasivnom “lajkanju”, omogućuje tvrtkama da iskoriste mogućnosti besplatnog oglašavanja interakcijom postojećih i potencijalnih kupaca. Vrlo je pozitivno za kompaniju kada se priča o njenom brendu što može biti vrijedan pokretač odluka potrošača, te isto tako činjenica da se te interakcije događaju na društvenoj mreži čini ih mjerljivima. Kompanije mogu mjeriti uspješnost njihovih marketinških kampanja te koliko im se ulaganja vratilo u obliku povećanja prodaje.
3. Podaci o kupcima - dobro osmišljen plan marketinga preko društvenih mreža pruža još jedan neprocjenjivi resurs za poboljšanje marketinških rezultata: podatke o kupcima. Umjesto da budu preplavljeni velikom količinom podataka, kompanije imaju mogućnost ne samo za izdvajanje podataka o kupcima, već i za pretvaranje istih u djelotvornu analizu tržišta kako bi mogli stvarati nove strategije za prikupljanje novih kupaca

4.2. SPECIFIČNA VRSTA MARKETINGA NA DRUŠTVENIM MREŽAMA

Pojavom društvenih mreža, u prvom redu Instagrama, postala je popularna nova metoda marketinga na društvenim mrežama, a riječ je o influencer marketingu. Influencer marketing nije novost. Postoji već mnogo godina. U zadnje vrijeme je postao vruća tema za marketinške stručnjake, posebno kao alternativa tradicionalnom oglašavanju koje može biti skupo i neučinkovito. Što je još važnije, potrošači ne vole da im se reklamiraju proizvodi i marketinške poruke svakim danom imaju sve manje vjerodostojnosti.

U suštini, influencer marketing je vrsta marketinga na društvenim mrežama koja koristi preporuke i spominjanje proizvoda od strane utjecajnih osoba tzv. influencera, pojedinaca koji imaju posvećene sljedbenike na društvenim mrežama i koji se smatraju stručnjacima unutar svog područja. Influencer marketing funkcionira zbog velike količine povjerenja koje su influenceri izgradili sa svojim sljedbenicima, a njihove preporuke služe kao oblik društvenog dokaza potencijalnim kupcima nekog brenda.

Jedna od najvećih zabluda o influencerima je da oni moraju bit netko s velikim brojem pratitelja na društvenim mrežama. Na taj se način brka popularnost sa utjecajem. Dakako, jedno ne isključuje drugo, no čin utjecanja zahtijeva određeni rezultat: promjenu mišljenja ili ponašanja potrošača. Influencer je, dakle, netko tko ima moć utjecati na percepciju drugih ili ih navesti da učine nešto drugačije poput toga da kupe određeni proizvod ili uslugu nekog brenda.

U takvoj dinamici odnosa influenceri zapravo služe kao ambasadori određenog brenda gdje oni promoviraju njihove proizvode na društvenim mrežama odnosno utječu na svoje sljedbenike da kupe proizvod tog brenda i za to dobivaju određenu naknadu.

Osoba mora imati kombinaciju tri ključna čimbenika kako bi je se moglo smatrati influencerom: doseg, kontekstualni kredibilitet i prodajno umijeće. Što su ova tri čimbenika veća, to je veći potencijal utjecaja veći.

Doseg označava sposobnost prenašanja poruke velikom broju ljudi. To se najčešće manifestira u obliku velike praćenosti na društvenim mrežama. Doseg je važan, ali sam po sebi nije dovoljan. Osoba koja nema veliki doseg, ali ima veliki kredibilitet kod pratitelja i veliko prodajno umijeće zove se mikro-influencer.

Također je važno da osoba ima veliko povjerenje i autoritet od strane svojih sljedbenika, odnosno da bude prepoznata kao stručno lice o određenoj temi što joj daje kredibilitet. Primjerice slavni sportaš će imati veliki kredibilitet kod preporuke sportske prehrane, a manji u npr. politici.

Umijeće prodaje također je važan faktor za influencere jer moraju imati sposobnost uvjeriti pratitelje u svoje gledište. Moraju imati sposobnost objasniti stvari na uvjerljiv način, prenesti jasnu poruku i jasno iznesti svoja stajališta.

Influencer marketing treba biti pošten i autentičan. Influencer govori o nekom proizvodu ne zato što je za to plaćen, već zato što to želi, jer smatra da je neki brend zanimljiv, a informacije korisne njegovim pratiteljima.

Stoga influencer marketing zahtijeva vrijeme, predanost i fokus te mora biti transparentan i pošten. Influencer će govoriti o dobrim i lošim stranama nekog proizvoda. Ali učinit će to na način koji je daleko vjerodostojniji i korisniji nego što bi to mogao učiniti bilo koji oglas.

Kada se pravilno izvede, influencer marketing kombinira doseg, kredibilitet i prodajno umijeće grupe influencera koji će zagovarati proizvod nekog brenda potrošačima, što će u konačnici rezultirati povećanjem svijesti, poboljšanom percepcijom i povećanjem prodaje proizvoda tog brenda.

5. WEB APLIKACIJE

Što je to web aplikacija? Web aplikacija je web mjesto koje sadrži stranice koje nemaju u potpunosti utvrđen sadržaj već se on određuje dinamički, ovisno o djelovanju korisnika te se zbog toga one zovu dinamičke web aplikacije. Dinamičke web aplikacije koriste web preglednik i web tehnologije kako bi obavljale određene zadatke preko interneta. One su različite u odnosu na statične web aplikacije koje su u suštini obične web stranice gdje je sav sadržaj statičan i unaprijed određen.

5.1. PREDNOSTI DINAMIČKIH WEB APLIKACIJA

Dinamičke web aplikacije nude mogućnost za puno više funkcionalnosti u odnosu na statične web stranice jer je njihov sadržaj podoban manipulaciji korisnika. To znači da korisnik ima mogućnost interakcije s tim sadržajem na način na koji to ne može imati u statičnoj web stranici što povećava njegovo iskustvo. Primjerice, korisnik neke web trgovine može filtrirati i pretraživati neke proizvode po određenim parametrima i pronaći što ga zanima. Pronađeni proizvod može staviti u svoju košaricu i unijeti podatke za dostavu. Na kraju procesa možda želimo prikazati podatke koje je korisnik unio. To ne možemo napraviti unutar statičnih web stranica jer ne možemo predvidjeti sadržaj koji će korisnik unijeti. Dakle, sadržaj stranice će se prikazivati dinamički, ovisno o raznim interakcijama korisnika sa tom stranicom.

Programiranje dinamičkih web aplikacija developerima nudi brojne mogućnosti i olakšava im proces razvoja te štedi vrijeme. Ako se vratimo na prethodni primjer web trgovine, pretpostavljajući da će ona sadržavati neke proizvode, u statičnoj stranici ti bi proizvodi morali biti ručno uneseni jedan po jedan te bi morali ponovno pisati isti HTML kod iznova za svaki proizvod. U dinamičkoj web aplikaciji moguće je odrediti statični HTML kod koji će se ponavljati za svaki proizvod (primjerice neka kartica proizvoda) koji će biti nadopunjen sa dinamičkim sadržajem jedinstvenim za svaki proizvod poput njegovog imena, URL - a slike tog proizvoda, opisa proizvoda i sl.

5.2. KAKO FUNKCIONIRA DINAMIČKA WEB APLIKACIJA

Web aplikacije su obično kodirane u jezicima koje podržava web preglednik, a to su HTML, CSS i JavaScript. One se sastoje od statičnih i dinamičkih stranica. Statična stranica je stranica koja se ne mijenja kada je poslužena korisniku, u suprotnosti sa dinamičkom stranicom koja je modificirana prije nego li je poslužena korisniku. Posluživanje stranice korisniku odvija se putem web servera i aplikacijskog servera.

5.2.1. WEB SERVER

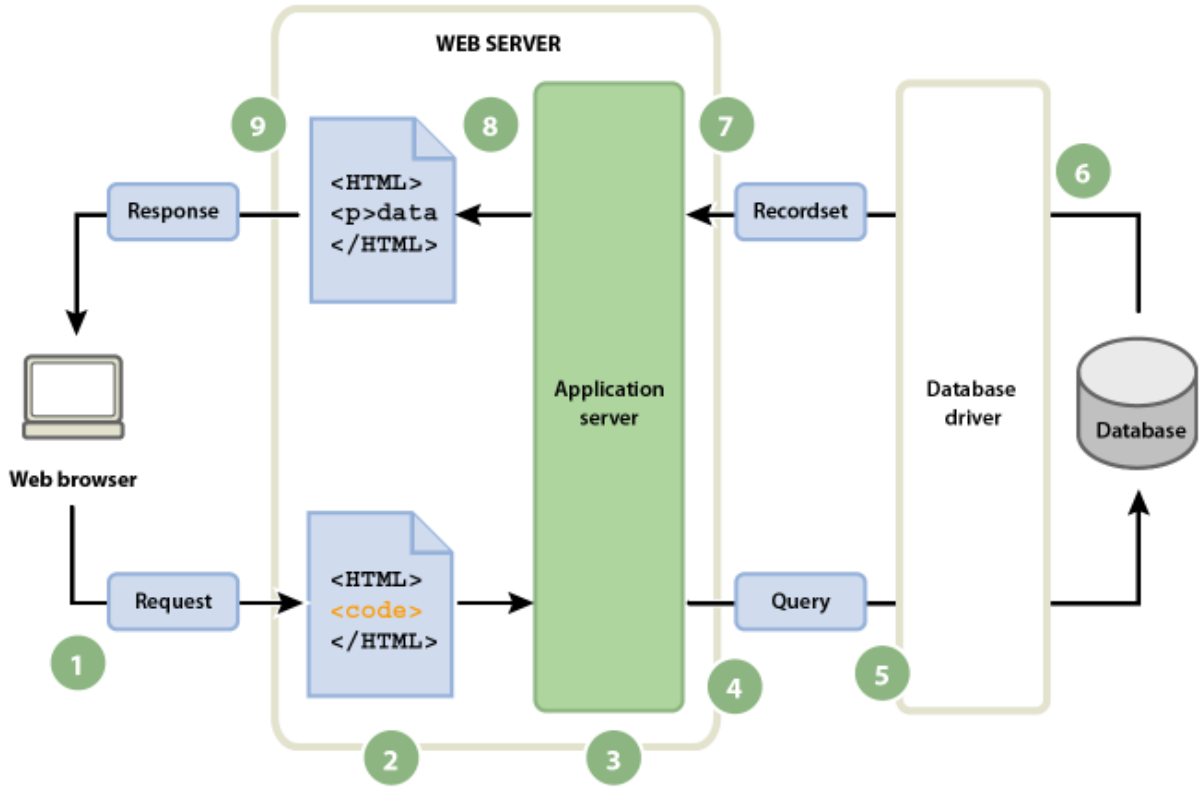
Svaka web stranica i web aplikacija nalazi se na nekom web serveru koji poslužuje njezine stranice klijentu odnosno web pregledniku. Kada korisnik zatraži određenu stranicu tako što klikne određeni link ili unese njenu adresu, web preglednik putem HTTP protokola šalje zahtjev web serveru za tu stranicu. Unutar web servera nalazi se statičan sadržaj stranice (HTML, CSS, slike itd.) te ukoliko je poslani zahtjev bio za statičnu stranicu tada ju web server poslužuje nazad web pregledniku.

5.2.2. APLIKACIJSKI SERVER

Ukoliko je poslani zahtjev bio za stranicu koja sadrži dinamički sadržaj tada ju web server prosljeđuje posebnom aplikacijskom serveru koji sadrži logiku za procesuiranje dinamičkog sadržaja. Aplikacijski server zatim čita kod na stranici i dovršava njen sadržaj prema određenim instrukcijama, što može zahtijevati i dohvaćanje određenih podataka iz baze podataka. Aplikacijski server ne komunicira direktno sa bazom podataka već putem posredništva tzv. database driver – a koji služi za interpretiranje podataka u bazi koji su inače nečitljivi za aplikacijski server. Kroz database driver, aplikacijski server prosljeđuje tzv. database query koji sadrži instrukcije za izvlačenje podataka iz baze podataka. Baza podataka aplikacijskom serveru zatim vraća tzv. recordset koji sadrži dohvaćene podatke koji će se koristiti za dovršavanje sadržaja stranice. Na kraju, aplikacijski server dovršenu

stranicu poslužuje nazad web serveru koji ju zatim poslužuje web pregledniku. Sve što web preglednik na kraju dobije je čisti HTML.

Slika 1: Kako funkcionira dinamička web aplikacija



Izvor:

Adobe: „Understand web applications“ 2021

https://helpx.adobe.com/dreamweaver/using/web-applications.html, (pristupljeno 18. rujan 2022.)

U gornjoj ilustraciji jasnije se može vidjeti opisani tok komunikacije između web preglednika, web servera, aplikacijskog servera te baze podataka.

5.3. SINGLE PAGE APLIKACIJE

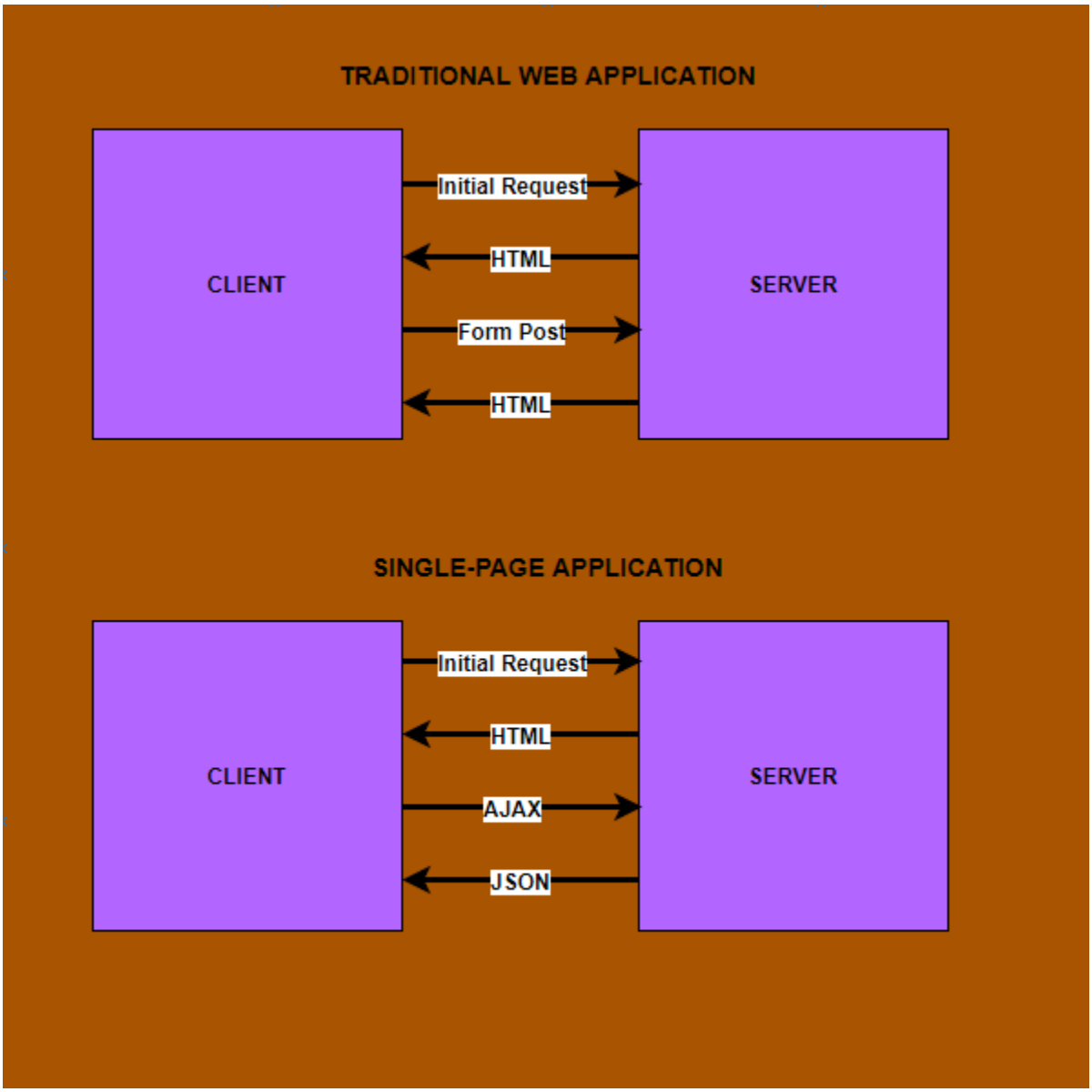
Single page aplikacije su posebna vrsta dinamičkih web aplikacija i najmoderniji pristup razvoja inspiriran mobilnim aplikacijama u kojemu se svaka web aplikacija u suštini sastoji od samo jedne HTML stranice u koju se dinamički učitava sadržaj ovisno o interakciji korisnika. Single page aplikacije su u potpunosti dinamičke i one se za promjenu stranica oslanjaju isključivo na JavaScript. Ta promjena stranica je prividna jer se korisnik zapravo nalazi na istoj stranici na kojoj se samo mijenja sadržaj. Da se stranica zapravo ne mijenja može se primijetiti iz činjenice da se web preglednik ne učitava već je promjena stranice momentalna.

To je moguće zato što se stranica ne konstruira na serveru koji vraća gotovi HTML, kao što je to slučaj u tradicionalnom multi page pristupu, već se ona konstruira u web pregledniku. Web server će pregledniku poslužiti tu jednu stranicu i sav JavaScript kod s kojim preglednik zatim barata. Ukoliko je potrebno pristupiti nekim podacima u bazi, web preglednik šalje HTTP zahtjev serveru koji komunicira s bazom i vraća potrebne podatke. Važno je napomenuti kako aplikacija nikada ne komunicira direktno s bazom iz sigurnosnih razloga (na taj način bi izložili bazu javnom pristupu jer se JavaScript kod izvodi direktno u pregledniku) već u tome posreduje određena backend aplikacija na serveru, primjerice neki API, koji komunicira s bazom i dostavlja zatražene podatke na frontend.

Glavna prednost single page aplikacija je njihova brzina. Stavlja se naglasak na što bolje iskustvo za korisnika. Ne postoji učitavanje stranica niti ekstra vremena čekanja. Postoji samo jedna stranica koju korisnik posjećuje u koju se sav sadržaj učitava pomoću JavaScripta. Također zbog načina na koji funkcioniraju single page aplikacije, još jedna prednost je mogućnost korištenja istog backenda za razvoj mobilne aplikacije.

S druge strane, određeni nedostatak single page aplikacija je izazovna implementacija SEO optimizacije, zbog čega će određene aplikacije koje će stavljati veliki naglasak na sadržaj, poput web trgovina, blogova i sl. i dalje koristiti tradicionalni multi page pristup gdje je svaka stranica posebna HTML datoteka koja se dohvaća sa web servera.

Slika 2: Razlika između tradicionalnih i single page web aplikacija



Izvor:

DZone: „How Single Page Web Applications Actually Work“ 2019,

https://dzone.com/articles/how-single-page-web-applications-actually-work, (pristupljeno 19. rujan 2022.)

6. PROCES RAZVOJA WEB APLIKACIJA

Proces razvoja web aplikacija sastoji se od nekoliko ključnih faza:

1. Stvaranje ideje
2. Planiranje razvoja
3. Osmišljavanje dizajna
4. Programiranje web aplikacije
5. Testiranje i lansiranje
6. Održavanje

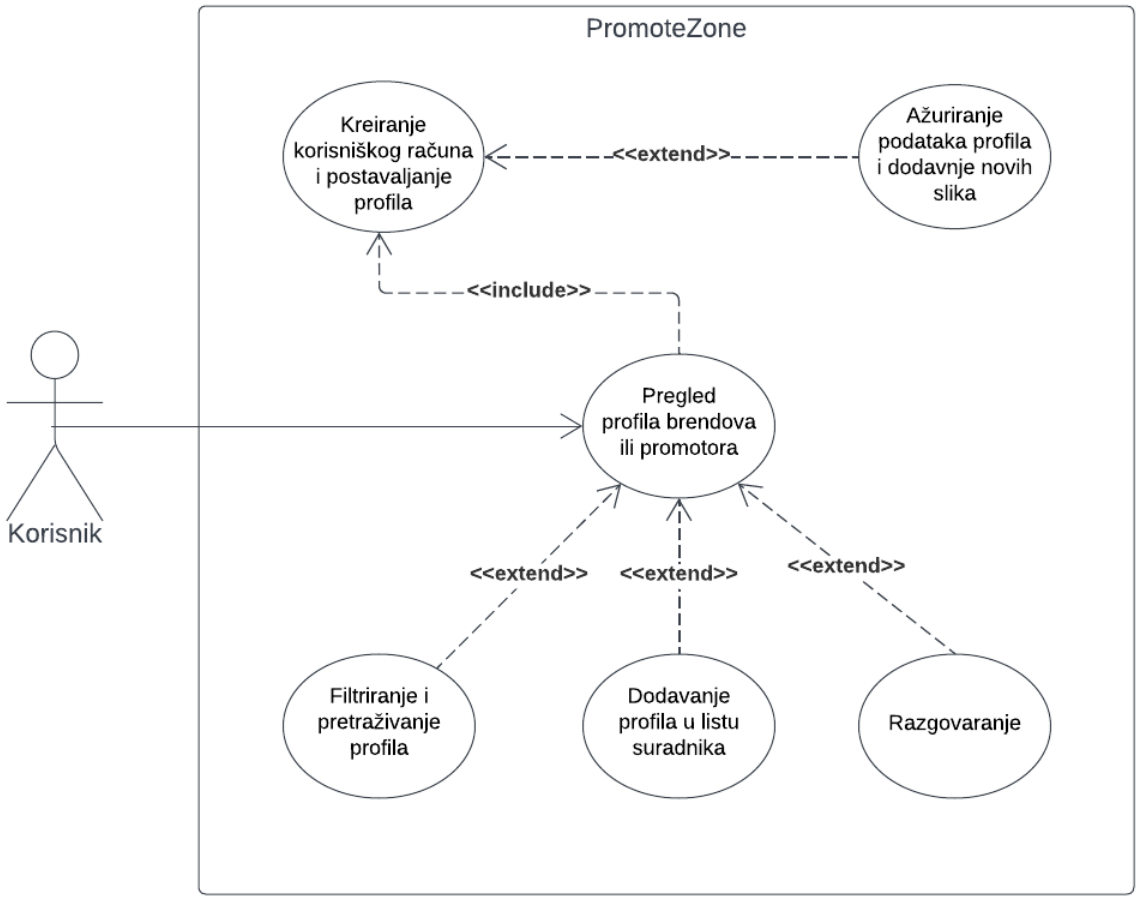
6.1. STVARANJE IDEJE

Prvi korak u razvoju web aplikacija je stvaranja ideje i određivanje tzv. MVP – a (Minimum Viable Product). MVP možemo odrediti kao minimalni proizvod koji će imati vrijednost za korisnika. Važno ga je odrediti zato što će razvoj svih funkcionalnosti odjednom iziskivati mnogo vremena, a u fazi razvoja aplikacija neće donositi prihod za kompaniju. Lansiranjem MVP – a kompanija može dobiti vrijedne povratne informacije kao i steći inicijalne korisnike.

Standardni način u programskom inženjerstvu za komuniciranje ideja kao i za modeliranje sustava je korištenje tzv. UML dijagrama. UML (Unified Modeling Language) je jezik za modeliranje nastao 1994. kako bi standardizirao do tada različite notacijske sustave i pristupe dizajnu softvera.

Postoje razne vrste UML dijagrama koje se koriste za različite primjene. Za komuniciranje ideje softvera koristi se UML Use Case dijagram.

Dijagram 1: UML Use Case dijagram aplikacije Promotezone



U gornjoj slici prikazan je UML Use Case dijagram aplikacije Promotezone. Unutar velikog pravokutnika označavamo granice sustava, dakle sve što se nalazi izvan tog pravokutnika nije dio tog sustava. Grafikom čovječuljka naznačeni su akteri u aplikaciji, u ovom slučaju to je korisnik. Unutar eliptičnih oblika nalaze se slučajevi upotrebe aplikacije. Vezom asocijacije korisnik je povezan sa slučajem upotrebe „pregled profila brendova ili promotora“. Veza asocijacije označava da je korisnik taj koji je inicirao tu radnju. Include vezom označavamo da slučaj upotrebe „pregled profila brendova ili promotora“ uključuje slučaj upotrebe „kreiranje korisničkog računa i postavljanje profila“ što znači da korisnik ne može pregledavati profile bez da je kreirao korisnički račun. Extend vezom označavamo kako slučaj upotrebe „ažuriranje podataka profila i dodavanje novih slika“ proširuje slučaj upotrebe „kreiranje korisničkog računa“. Extend vezom povezujemo one

slučajeve upotrebe koji nisu obavezni za korisnika. Također, extend vezom povezujemo slučajeve upotrebe „filtriranje i pretraživanje profila“, „dodavanje profila u listu suradnika“ i „razgovaranje“ sa slučajem upotrebe „pregled profila brendova ili promotora“ te tako označavamo kako proces pregledavanja profila sadrži i te opcije za korisnika.

6.2. PLANIRANJE RAZVOJA

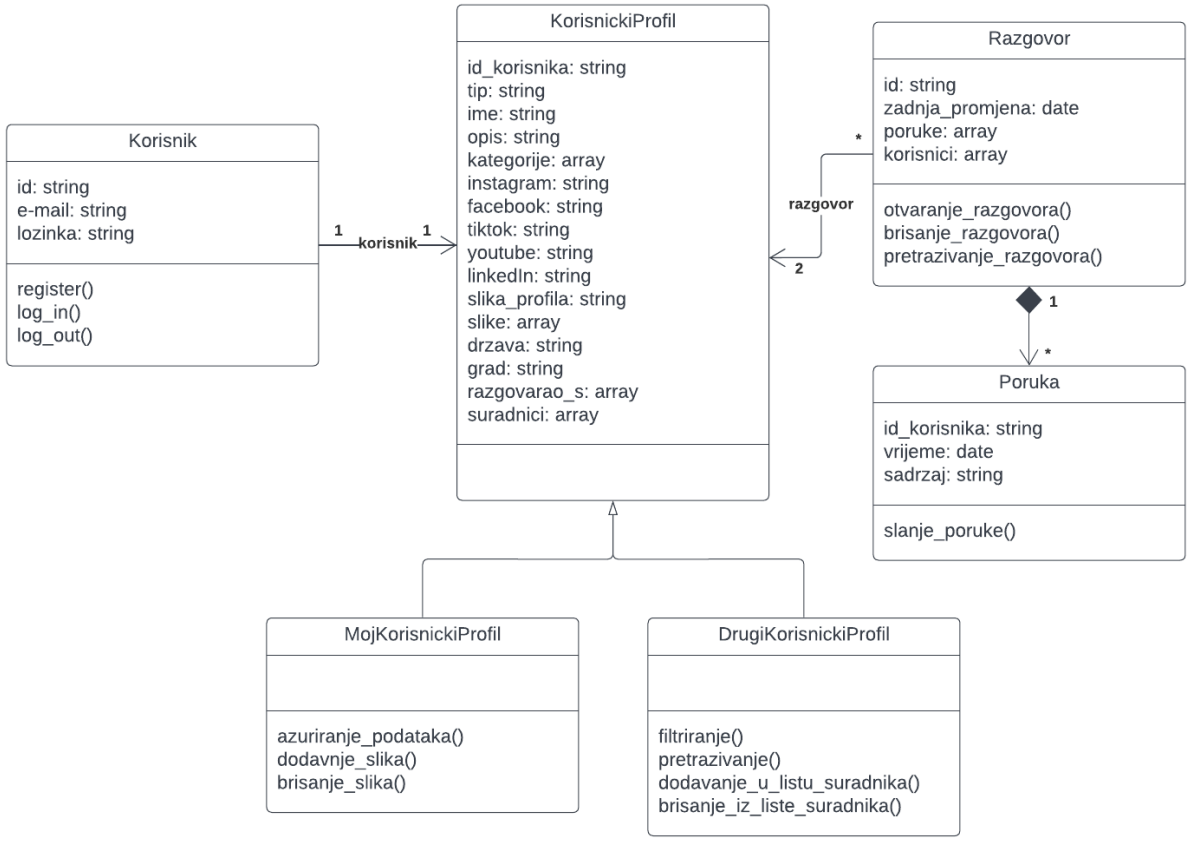
U fazi planiranja razvoja odlučuje se o korištenim tehnologijama i pravi se plan razvoja. Budući da je u prethodnom koraku utvrđena ideja aplikacije imamo jasniju sliku o tome što je potrebno napraviti. U slučaju aplikacije Promotezone odlučeno je kako će se ona razvijati korištenjem Vue JS razvojnog okvira koji nudi brojne alate koji će olakšati razvojni proces te će se u svrhu backenda koristiti Firebase koji sadrži pakete koji savršeno odgovaraju zahtjevima aplikacije Promotezone poput autentifikacije, Firebase Storage – a u kojeg će se spremati slike, baze podataka u realnom vremenu koja je potrebna za chat funkcionalnost te hostinga same aplikacije.

Također, u planiranju procesa razvoja možemo se poslužiti UML dijagramima kako bi bolje dočarali potrebne korake u razvoju te napravili okvirnu skicu sustava.

Najčešći dijagram koji se koristi u planiranju i modeliranju sustava je UML klasni dijagram. On se koristi u različitim primjenama, primjerice njime možemo skicirati sve potrebne klase i interakcije između njih koje je sadržavati određeni objektno orijentirani projekt.

Također, njime je moguće prikazati domenski model aplikacije odnosno koje su to informacije koje će ta aplikacija obrađivati i kojim će se konceptima ona voditi.

Dijagram 2: UML klasni dijagram aplikacije Promotezone



U gornjem dijagramu prikazan je domenski model aplikacije Promotezone. On je sadržan od nekoliko klasa od kojih će svaka imati neke atribute (podatke) i metode (funkcije koje će se izvršavati nad tom klasom). Klasa „Korisnik“ koja ima atribute id koji će biti tipa string, e-mail koji će biti tipa string i lozinka koja će biti tipa string. Također, ona sadrži metode „register“, „log_in“ te „log_out“. Klasa „Korisnik“ vezom asocijacije povezana je sa klasom „KorisnickiProfil“ što će značiti da korisnik ima korisnički profil. Kardinalnostima veza (mali brojevi na početku i na kraju veze) označavamo da korisnik može imati samo jedan korisnički profil te također, korisnički profil može imati samo jednog korisnika. Klasa „KorisnickiProfil“ sadrži atribute id korisnika koji će biti tipa string, tip koji će biti tipa string, ime koje će biti tipa string, opis koji će biti tipa string, kategorije koje će biti tipa array,

instagram koji će biti tipa string, facebook koji će biti tipa string, tiktok koji će biti tipa string, youtube koji će biti tipa string, linkedIn koji će biti tipa string, slika profila koja će biti tipa string, slike koje će biti tipa array, drzava koja će biti tipa string, grad koji će biti tipa string, razgovarao s koji će biti tipa array te suradnici koji će biti tipa array. Vezom nasljeđivanja klasa „KorisnickiProfil“ povezana je sa klasama „MojKorisnickiProfil“ i „DrugiKorisnickiProfil“ te je time označeno kako su one podklase klase „KorisnickiProfil“ te one nasljeđuju sve atribute i metode klase roditelja. Klasa „MojKorisnickiProfil“ sadži metode „azuriranje_podataka“, „dodavanje_slika“ i „brisanje_slika“. Klasa „DrugiKorisnickiProfil“ sadrži metode „filtriranje“, „pretrazivanje“, „dodavanje_u_listu_suradnika“ i „brisanje_iz_liste_suradnika“. Klasa „Razgovor“ sadrži atribute id koji će biti tipa string, zadnja promjena koja će biti tipa string, poruke koje će biti tipa array i korisnici koji će biti tipa array. Također, sadržavat će metode „otvaranje_razgovora“, „brisanje_razgovora“ i „pretrazivanje_razgovora“. Klasa „Razgovor“ vezom asocijacije povezana je sa klasom „KorisnickiProfil“ te je time označeno kako razgovor ima korisnički profil. Kardinalnostima veze označeno je kako razgovor može imati 2 i samo 2 korisnička profila, a korisnički profil može imati nula ili više razgovora, što označavamo zvjezdicom. Klasa „Poruka“ sadrži atribute id korisnika koji će biti tipa string, vrijeme koje će biti tipa date i sadržaj koji će biti tipa string te će također sadržavati metodu „slanje_poruke“. Vezom kompozicije povezujemo klase „Razgovor“ i „Poruka“ te tako označavamo kako razgovor može imati nula ili više poruka, a poruka mora imati jedan i samo jedan razgovor. Crnim rombom veze kompozicije označavamo da klasa „Poruka“ ne može postojati bez klase „Razgovor“ odnosno ukoliko bi izbrisali neki razgovor brišu se i sve poruke koje su povezane s tim razgovorom.

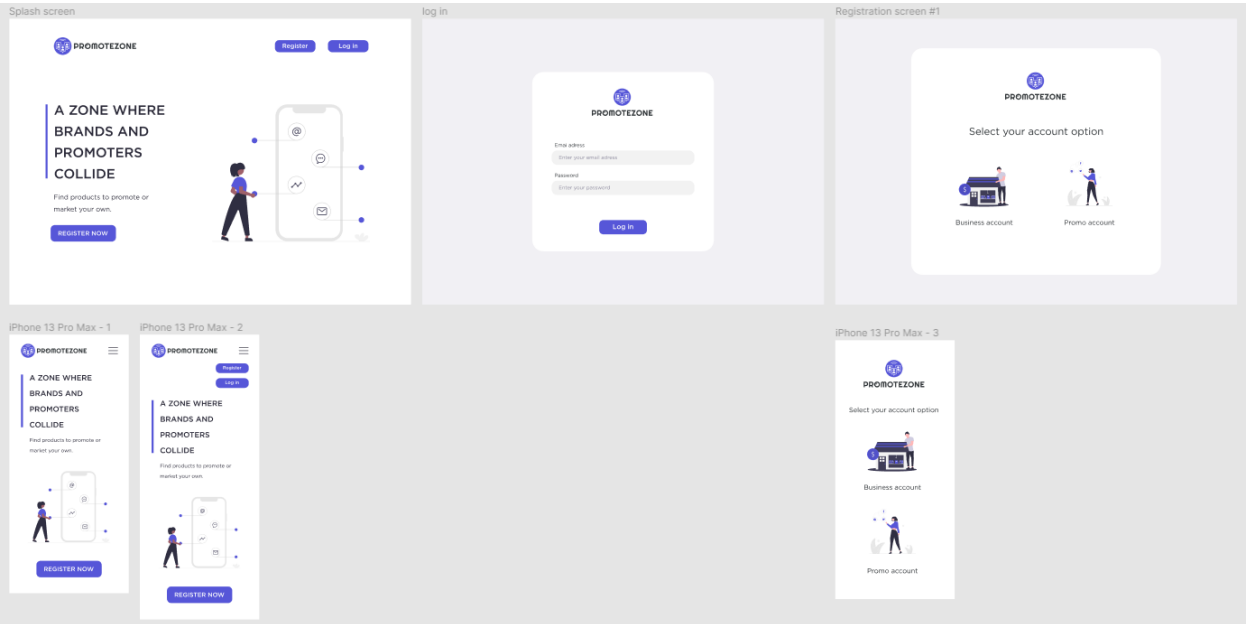
6.3. OSMIŠLJAVANJE DIZAJNA

Nakon što imamo ideju i jasni model aplikacije možemo krenuti u izradu prototipa aplikacije i osmisliti njen dizajn. Za to se koriste alati kao što su Adobe XD i Figma namijenjeni za UI (User Interface ili korisničko sučelje) i UX (User Experience ili korisničko iskustvo) dizajn. U tom procesu crtamo svaki zasebni ekran aplikacije i povezujemo ga s povezanim ekranima kako bi simulirali gotovu aplikaciju. To može biti pogotovo korisno u komuniciranju ideje s klijentom kojemu možemo jasno predstaviti viziju aplikacije.

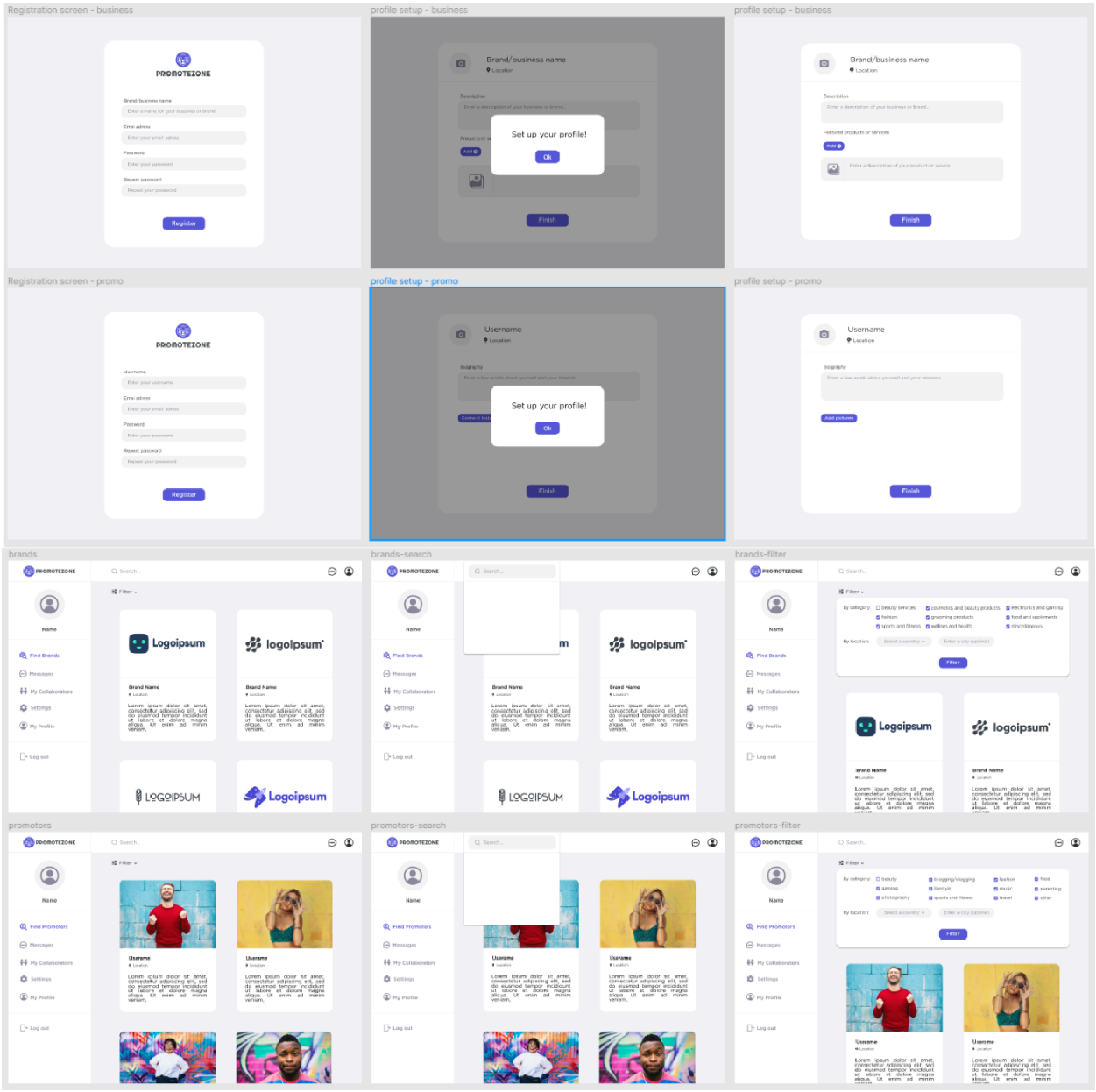
U procesu dizajna aplikacije finaliziraju se svi detalji vezani za dizajn i stvara sa jasna slika aplikacije što će u dugom roku uštediti vrijeme i olakšati posao developerima koji će imati jasno definirane potrebe razvoja aplikacije.

U procesu dizajna aplikacije Promotezone korištena je Figma. U tom je procesu skiciran njen vizualni identitet i početna zamisao.

Slika 3: Dizajn aplikacije Promotezone u Figmi, 1. dio

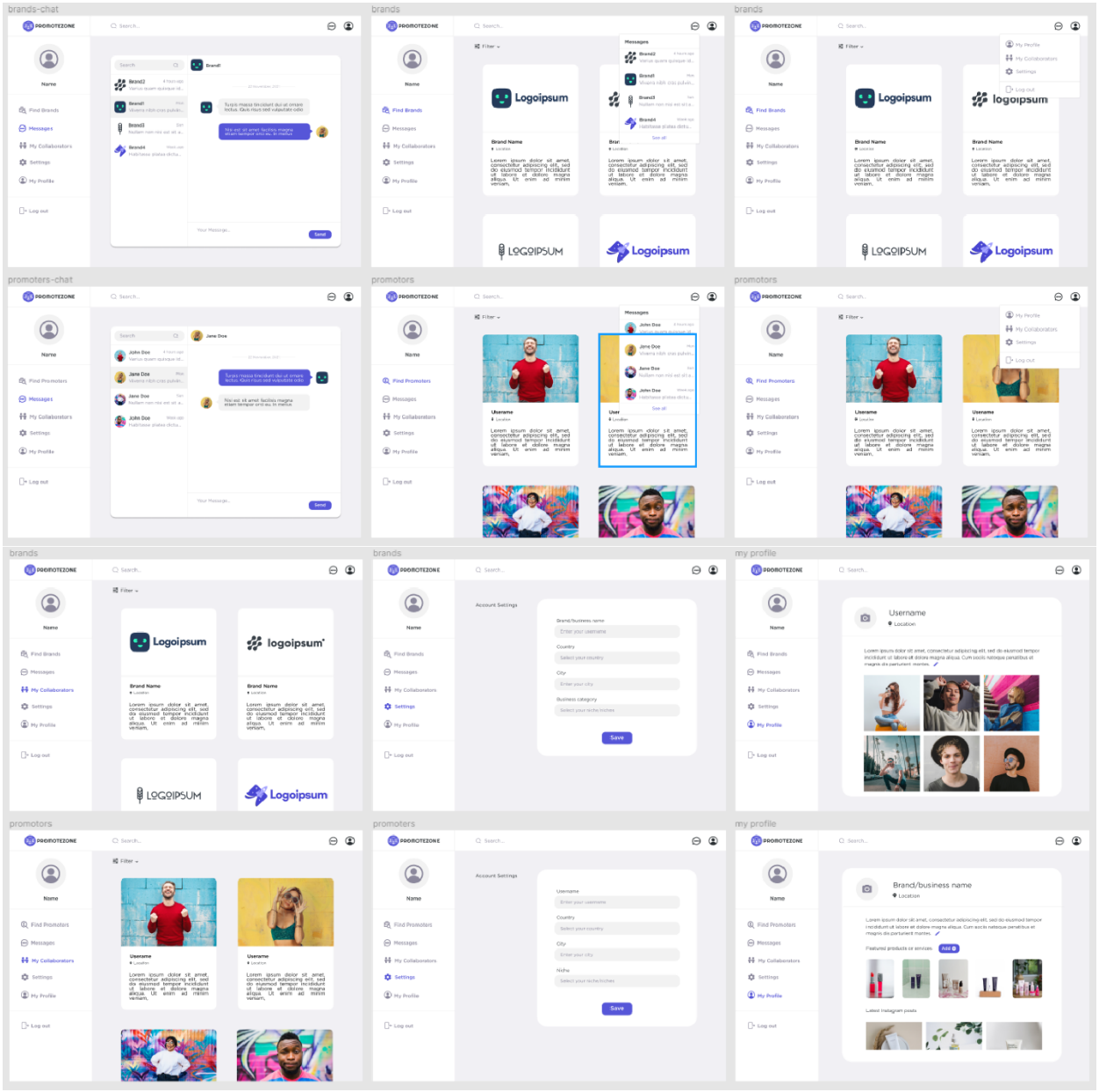


Slika 4: Dizajn aplikacije Promotezone u Figmi, 2. dio



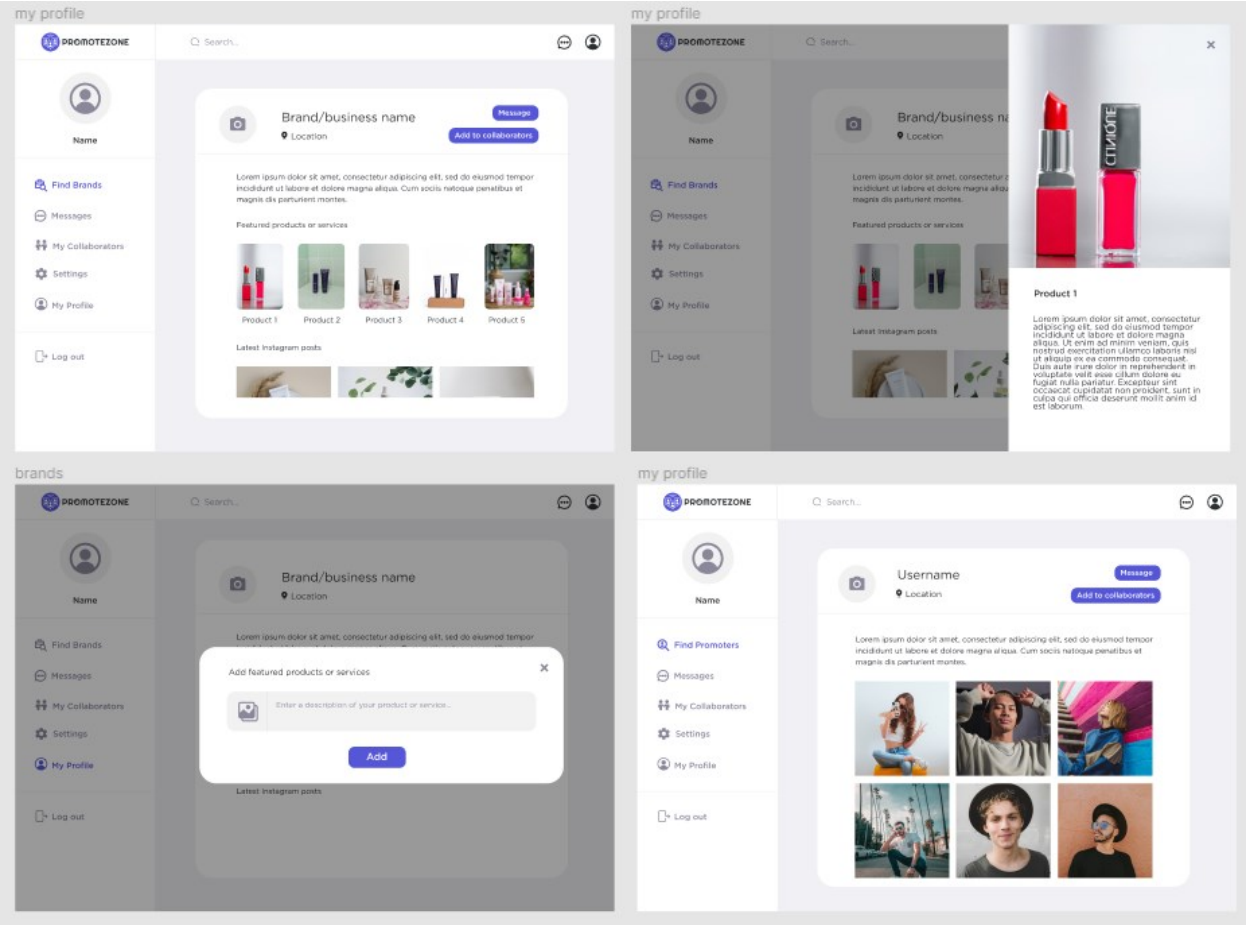
Prikaz registracijskog procesa aplikacije Promotezone te home stranice sa karticama profila u Figmi.

Slika 5: Dizajn aplikacije Promotezone u Figmi, 3. dio



Prikaz ekrana za razgovaranje te postavke profila i sami profil.

Slika 6: Dizajn aplikacije Promotezone u Figmi, 3. dio



Prikaz profila i dodavanja slika na profil.

6.4. PROGRAMIRANJE APLIKACIJE

Programiranje aplikacije omogućuje funkcioniranje predviđenih značajki i gradi aplikaciju koja će imati neku vrijednost za korisnika. Svaka aplikacija sastojat će se od frontend i backend dijela. Frontend dio obuhvaćat će sav onaj dio koji korisnik vidi, dakle korisničko sučelje i njegova logika, što će biti posao za frontend developera. Backend dio aplikacije je onaj dio koji korisnik ne vidi, ali je zaslužan za funkcioniranje aplikacije poput baze podataka, servera, API – a i sl. To će biti posao za backend developera.

6.5. TESTIRANJE I LANSIRANJE APLIKACIJE

Završna faza razvoja aplikacije je njeno testiranje i otklanjanje mogućih bugova. Ovisno o veličini same aplikacije taj proces može biti dugotrajan jer je potrebno testirati svaku funkcionalnost do zadnjeg detalja kako bi se osigurala potrebna razina kvalitete i kako bi se aplikaciju moglo lansirati. Uobičajeno, to će biti posao za softver testera koji će prijaviti pronađene bugove developerima koji će ih zatim popraviti.

6.6. ODRŽAVANJE

Nakon što se aplikacija lansira tu posao ne staje, već je aplikaciju potrebno konstantno održavati i ažurirati po potrebi ukoliko se dogodi neki problem ili otkrije neki novi bug. Kako će se povećavati broj korisnika u aplikaciji otkrivat će se novi problemi koji se do tada nisu mogli otkriti te će se također ponuditi izvor informacija koje će se moći analizirati i utvrditi koje bi se stvari na aplikaciji eventualno mogle poboljšati.

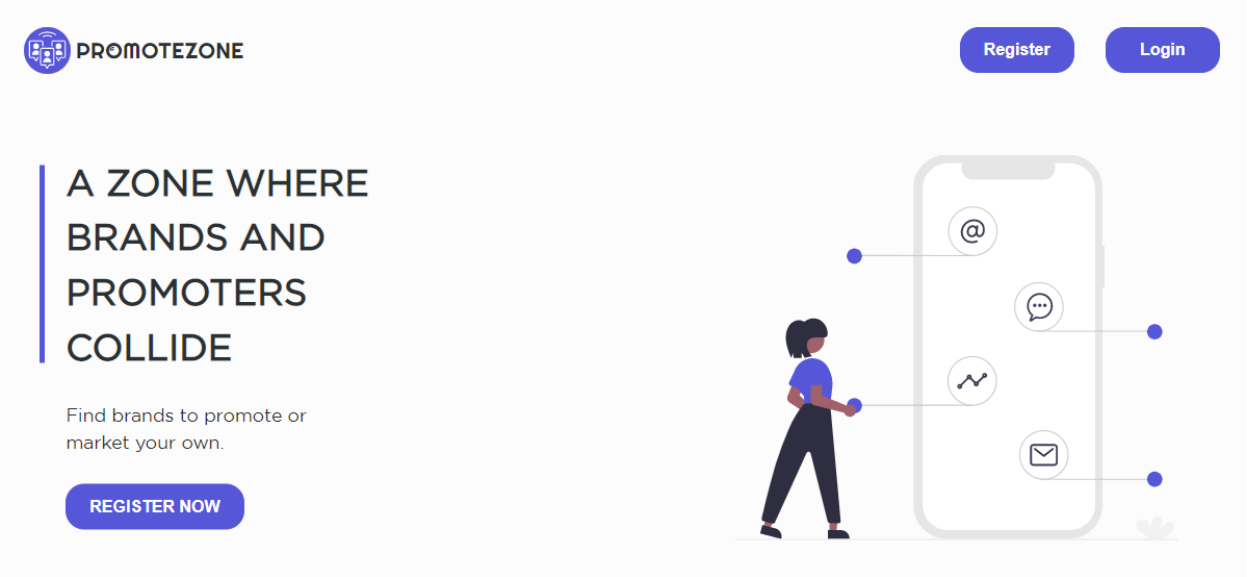
7. APLIKACIJA PROMOTEZONE

Ideja iza aplikacije „Promotezone“ je povezati brendove sa njihovim možebitnim promotorima. Na društvenim mrežama nije uvijek lako pronaći influencere u određenim kategorijama i geografskim lokacijama. Stoga je ideja aplikacije na neki način spojiti regionalne influencere i lokalne mikro-influencere i pružiti im priliku za monetiziranje svog društvenog utjecaja sa brendovima koji također traže prilike i načine za proširivanje svoje poruke. To može biti bilo kakav slučaj, od primjerice lokalne trgovine koja planira rasprodaju i želi proširiti riječ u gradu ili lokane slastičarnice koja ima najbolji kolač i nastoji na taj način predstaviti svoj brend do primjerice nekog velikog brenda sportske prehrane koji traži ambasadore koji će biti vjerni zagovarači njihovog brenda.

7.1. FUNKCIONALNOSTI

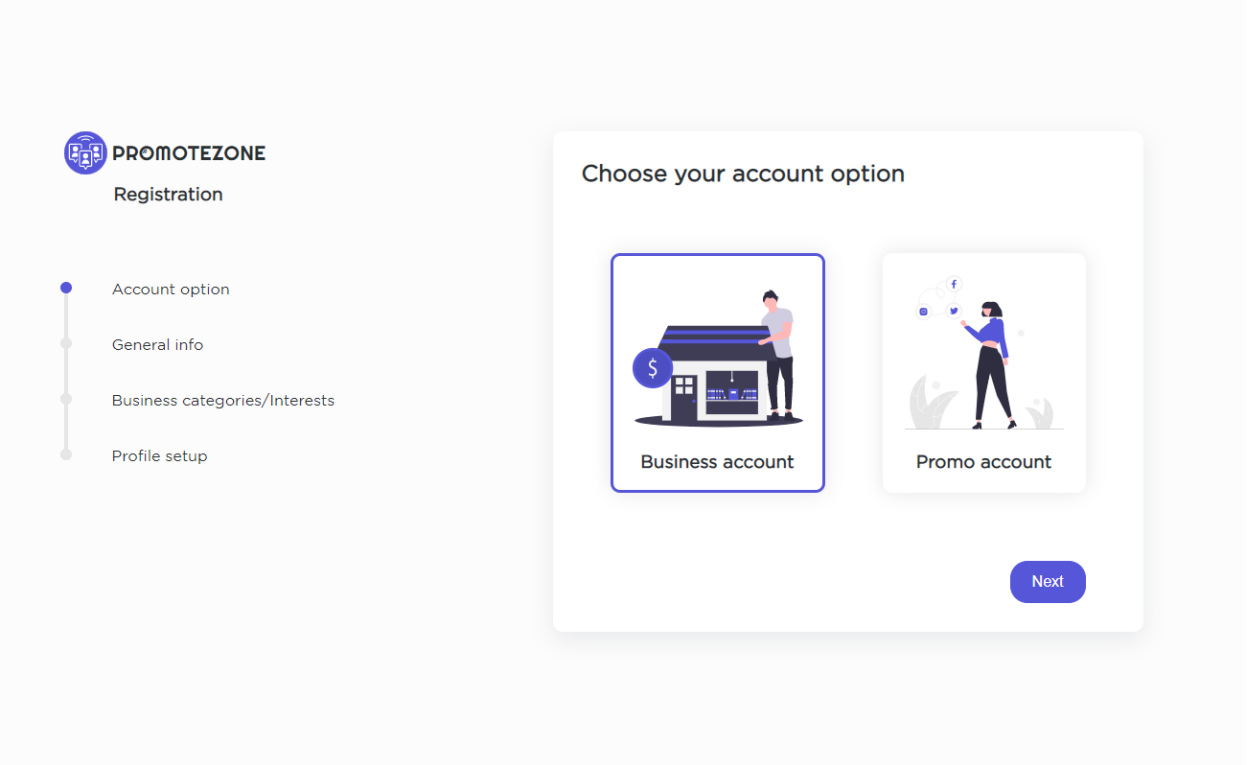
Na aplikaciji Promotezone omogućeno je traženje brendova i promotora po određenim kategorijama te po lokaciji odnosno po državi i gradu. U aplikaciju je moguće registriranje na dva načina: Business account za brendove i Promo account za promotore. Svaka vrsta profila ima mogućnost gledanja i pretraživanja druge vrste.

Slika 7: Naslovna stranica aplikacije Promotezone



Slika 8: Proces registracije u aplikaciju Promotezone.

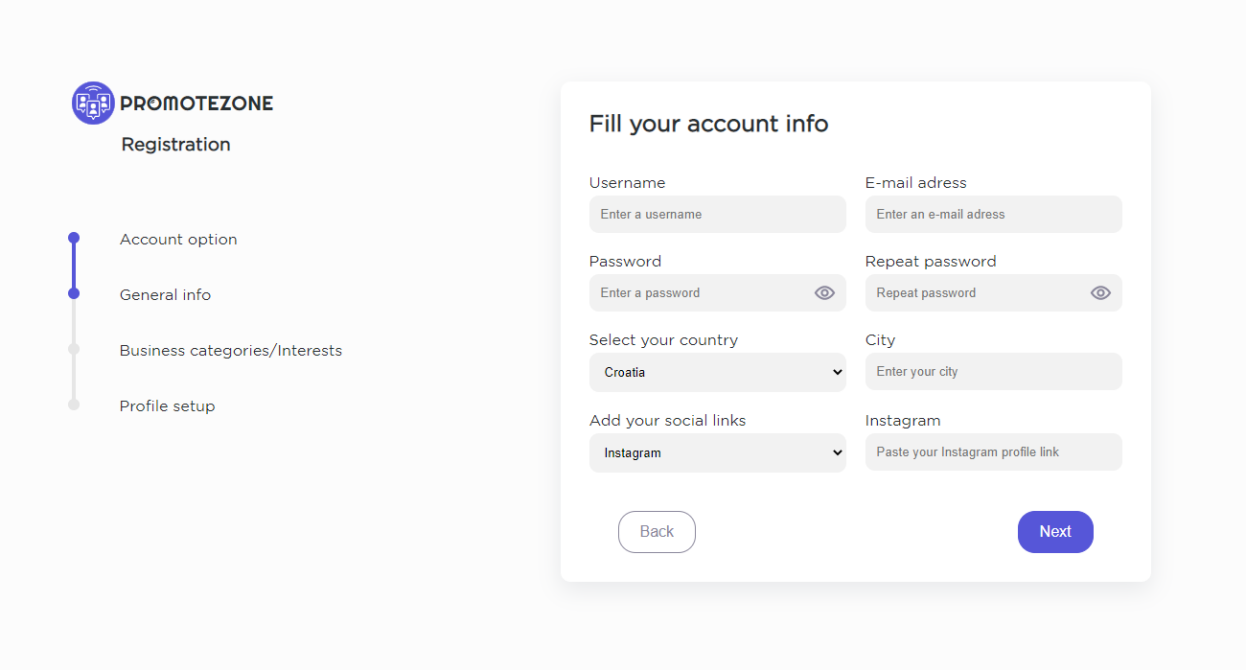
Korak 1: odabir vrste računa



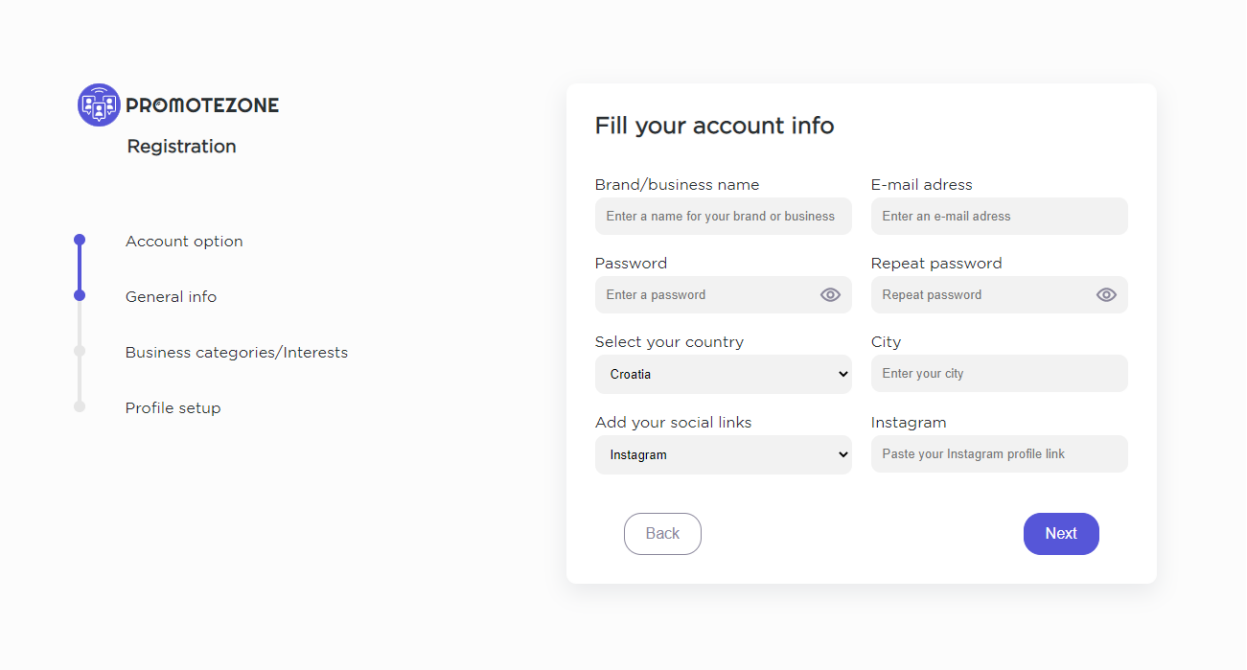
Omogućen je odabir dvije vrste računa:

1. Business account za brendove/tvrtke
2. Promo account za promotore

Slika 9: Proces registracije u aplikaciju Promotezone. Korak 2: Popunjavanje podataka o računu – Business account



Slika 10: Proces registracije u aplikaciju Promotezone. Korak 2: Popunjavanje podataka o računu – Promo account

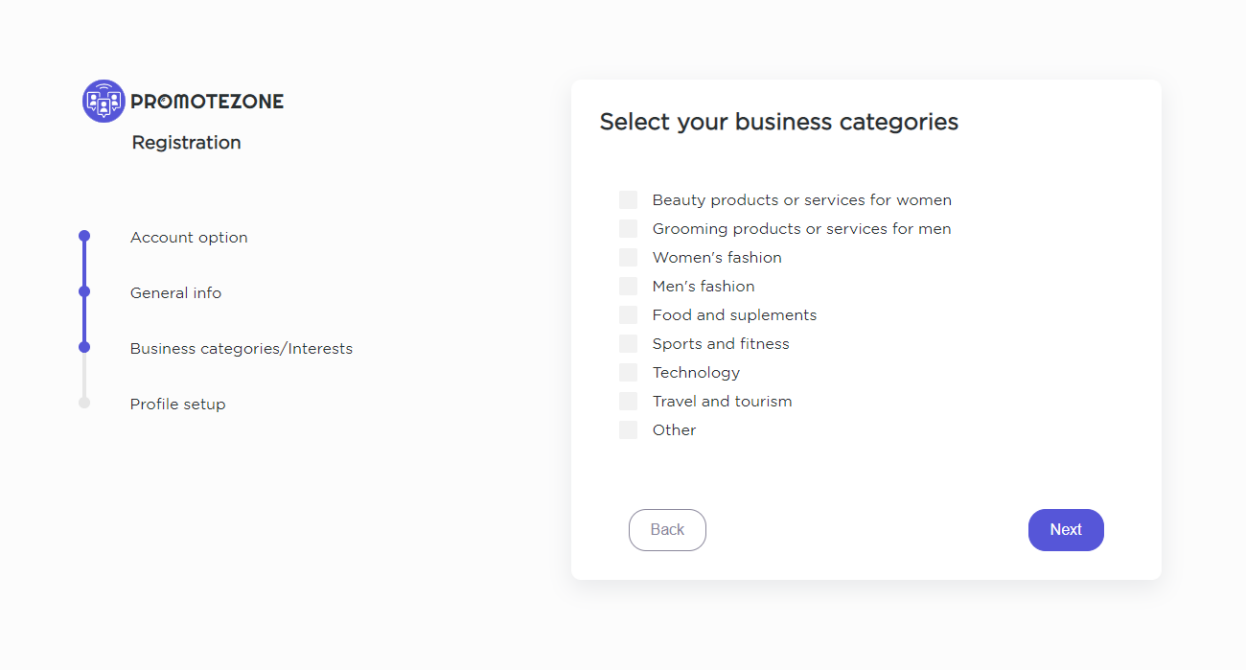


Ovisno o tome koju je vrstu računa korisnik odabrao u prvom koraku, u koraku broj dva popunjava podatke o svojem računu. Podatci su u principu vrlo slični za obje vrste računa međutim razlika je u tome što business account ima Brand/business name, a promo account ima Username.

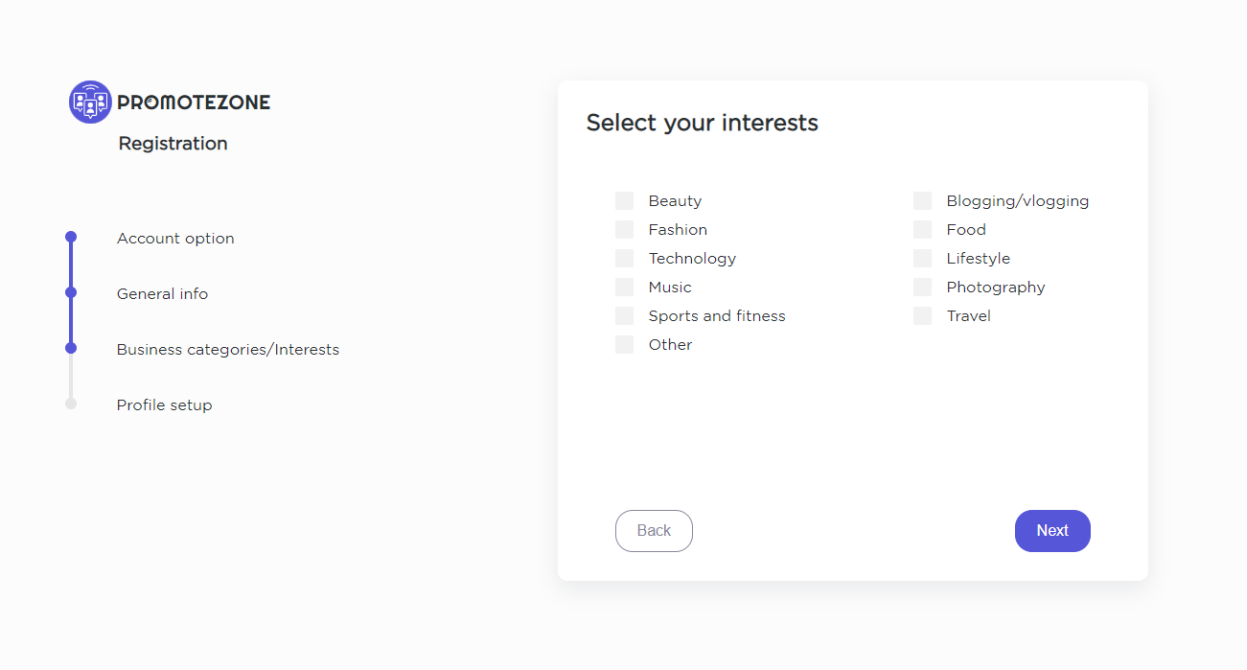
Sljedeći podaci koje korisnik treba popuniti su redom:

1. Email adresa koju će kasnije koristi za logiranje u aplikaciju.
2. Lozinka i ponavljanje lozinke kako bi se osiguralo da korisnik nije pogriješio prilikom unosa. Također prilikom klika na ikonu oka na desnoj strani input polja moguće je prikazati unesenu lozinku i ponovno je sakriti
3. Iz padajućeg izbornika potrebno je odabrati državu brenda/promotora.
4. Grad branda/promotora.
5. Zadnji podaci su opcionalni, a oni uključuju povezivanje društvenih mreža sa profilom na način da korisnik kopira link od svojeg profila na nekoj društvenoj mreži te ga zalijepi u pripadajuće input polje. Iz padajućeg izbornika korisnik odabere koju društvenu mrežu želi povezati sa svojim profilom te se ovisno o tome mijenja input polje desno u koje korisnik može zalijepiti link. Društvene mreže koje je moguće dodati su Instagram, Facebook, TikTok, YouTube i LinkedIn.

Slika 11: Odabiranje poslovnik kategorija brenda



Slika 12: Odabiranje polja interesa promotora



Sljedeći korak je odabiranje kategorija ili interesa pod kojim se račun želi svrstavati. Po tim kategorijama ili interesima će računi druge vrste moći filtrirati taj račun. Ovisno o tome koju vrstu računa korisnik kreira dostupne su kategorije odnosno interesi:

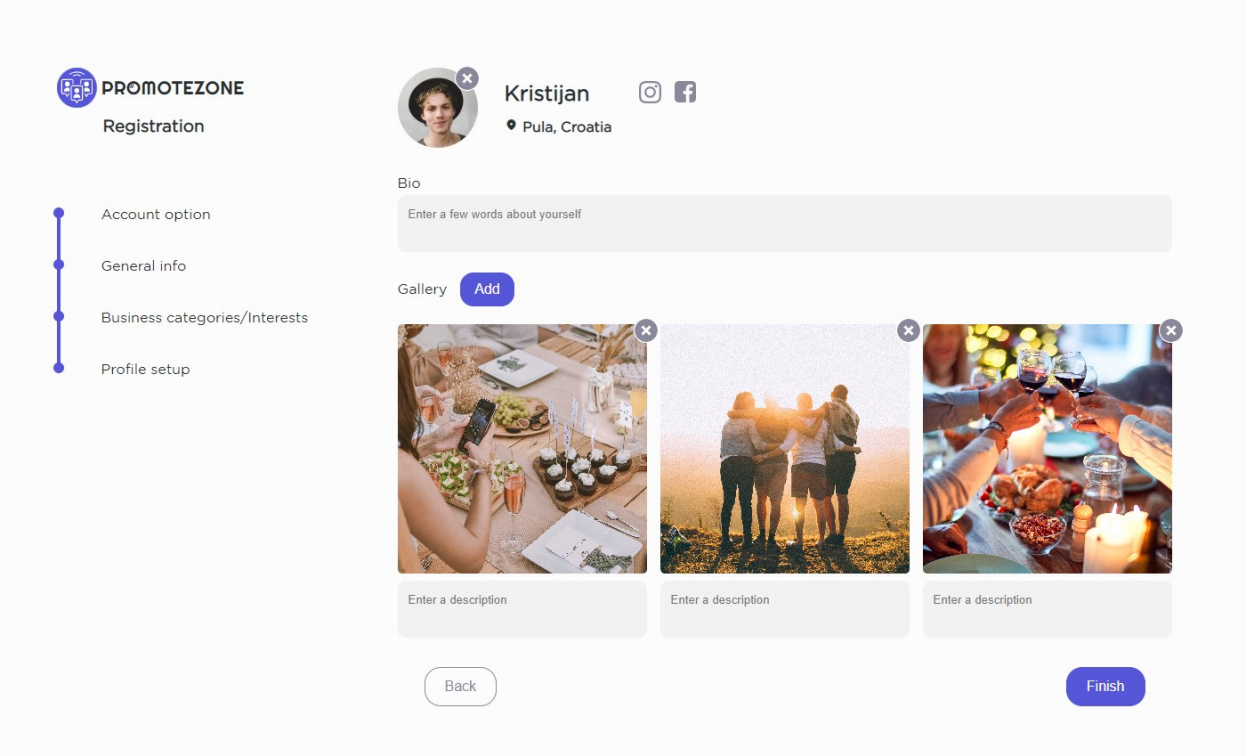
Za Business account:

1. Proizvodi i usluge za uljepšavanje za žene
2. Proizvodi i usluge za uljepšavanje za muškarce
3. Ženska moda
4. Muška moda
5. Hrana i dodaci prehrani
6. Sport i fitnes
7. Tehnologija
8. Putovanje i turizam
9. Ostalo

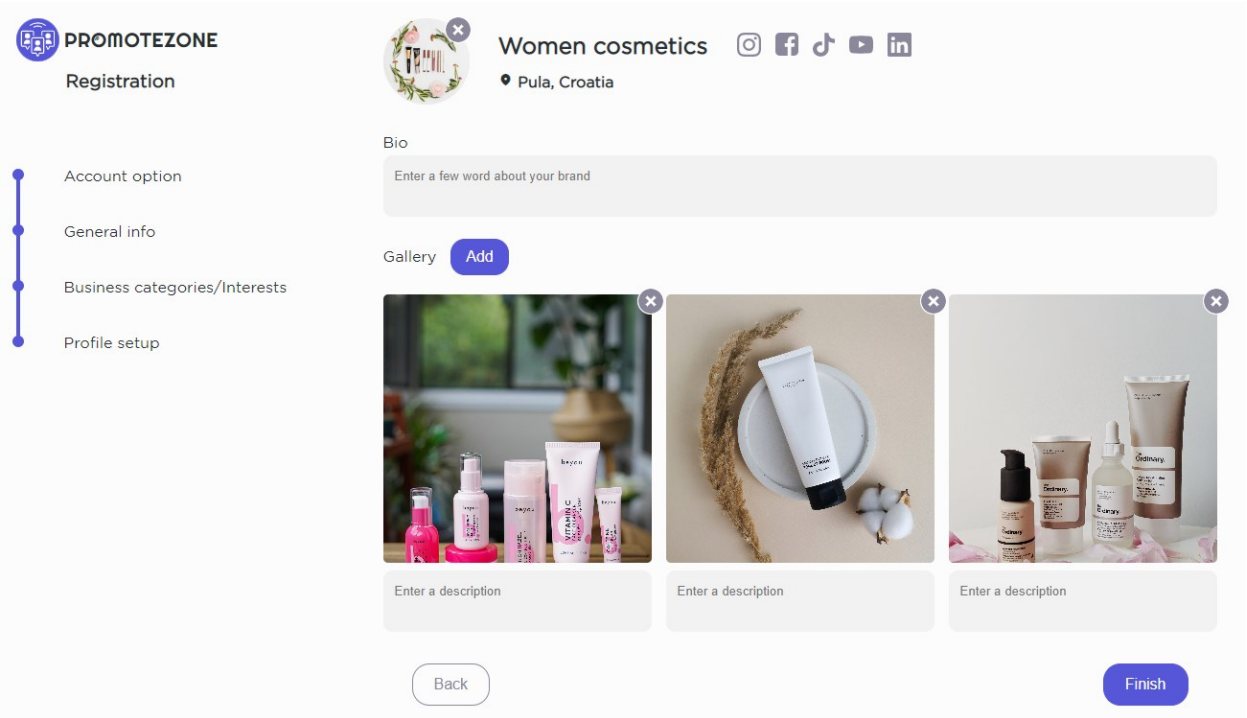
Za Promo account:

1. Ljepota
2. Blogging ili vlogging
3. Moda
4. Hrana
5. Tehnologija
6. Životni stil
7. Glazba
8. Fotografija
9. Sport i fitnes
10. Putovanja
11. Ostalo

Slika 13: Zadnji korak: kreacija profila – Promo account

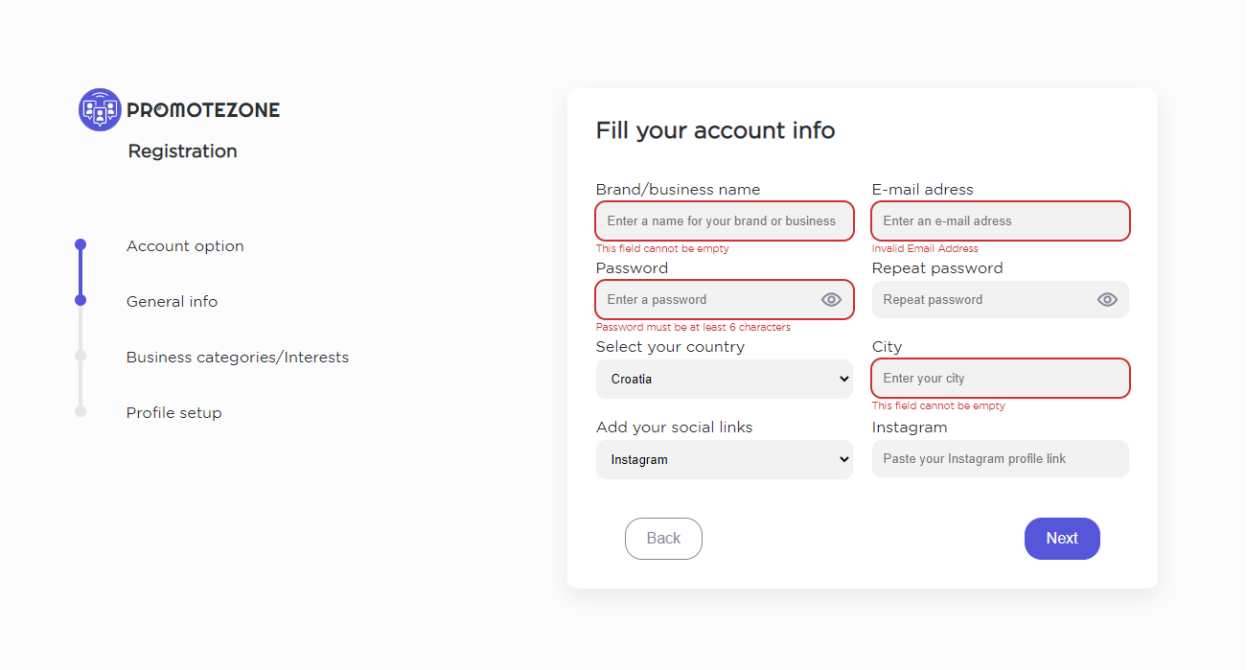


Slika 14: Zadnji korak: kreacija profila – Business account



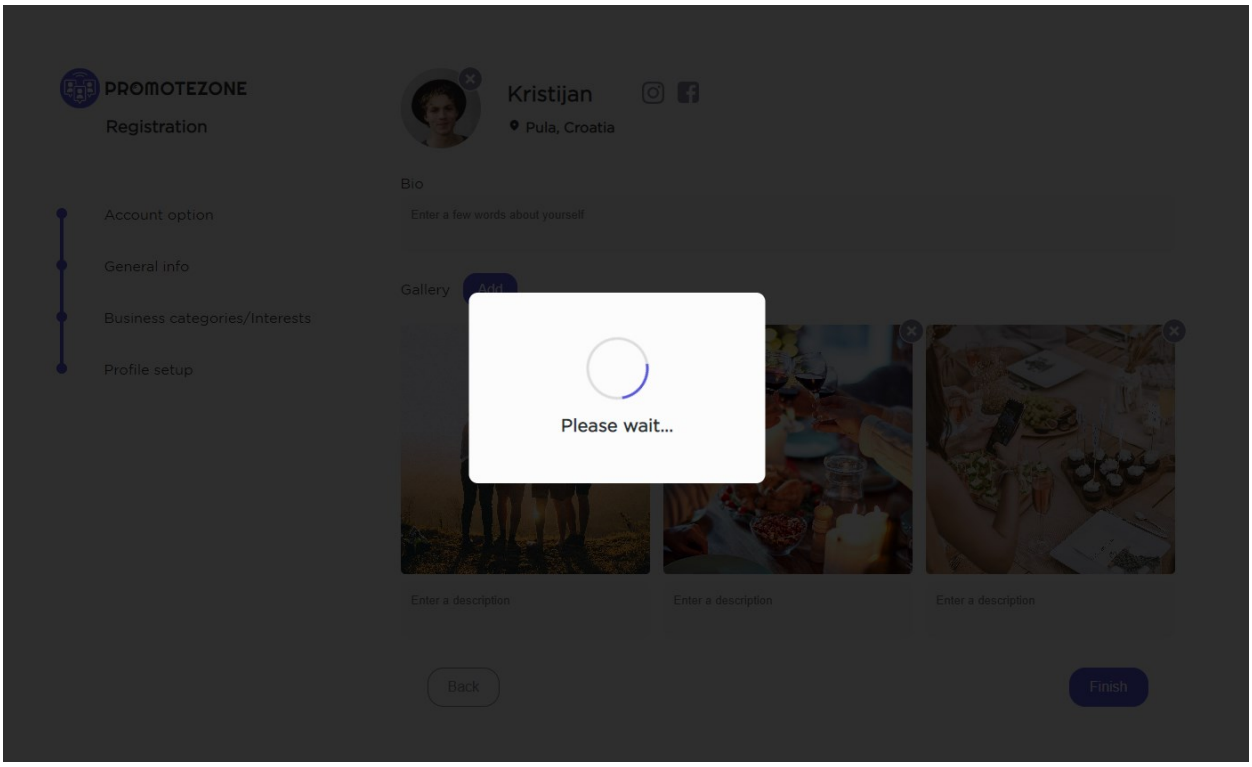
U zadnjem koraku korisnik može urediti svoj profil koji će računi druge vrste moći vidjeti. Može dodati profilnu sliku, opis profila te slike kojima želi dočarati svoj brand ili sebe kao promotora. Za svaku pojedinačnu sliku je omogućeno dodavanje opisa. Također, sliku je moguće obrisati klikom na x ikonu u gornjem desnom kutu svake pojedinačne slike. Uz ime profila (Brand/bussines name za Business account ili Username za Promo account) stoje ikone društvenih mreža koje je korsinik odlučio povezati sa svojim profilom. Klikom na pojedinu ikonu odlazimo na profil korisnika na toj društvenoj mreži. Na profilu će se prikazati ikone za samo one društvene mreže koje je korisnik odlučio povezati sa svojim profilom. To mogu biti sve društvene mreže, ali ne mora biti niti jedna. Također, ispod imena profila nalazi se lokacija koju je korisnik prethodno unio.

Slika 15: Validacija podataka



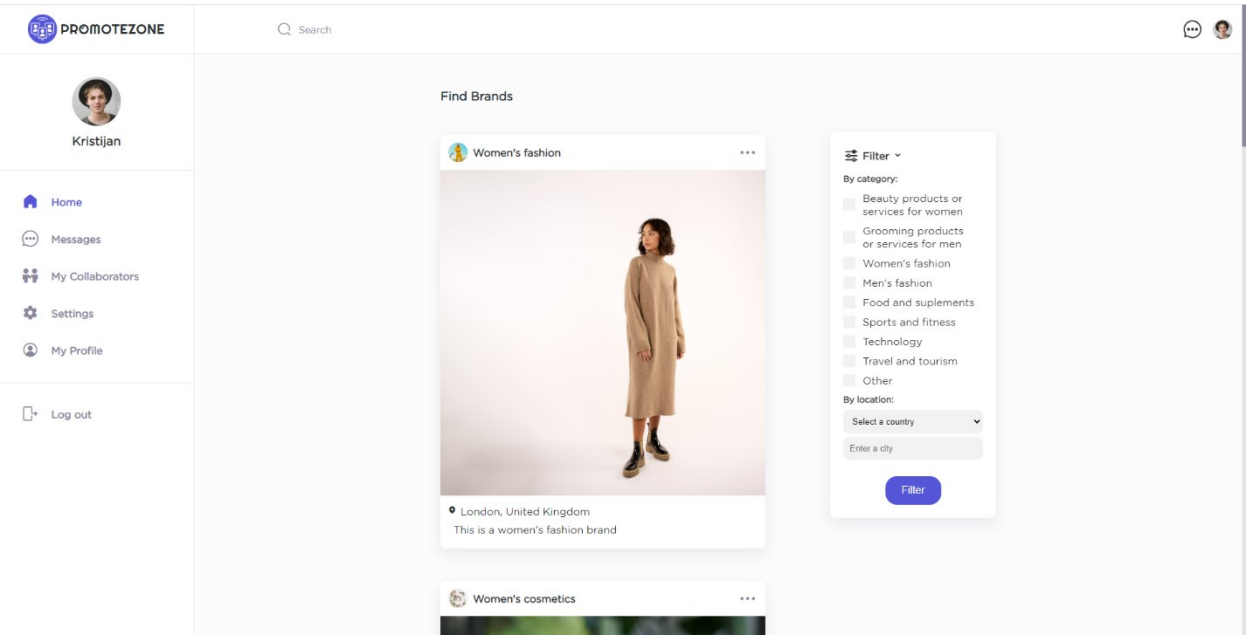
Kada korisnik klikne na gumb Finish u zadnjem koraku provjerava se da li su svi podaci validno uneseni. Ukoliko nisu, aplikacija vraća korisnika na korak na kojemu podaci nisu validno uneseni i upozorava ga koje točno sve podatke treba provjeriti.

Slika 16: Loading modal prilikom kreiranja profila



Kada je korisnik završio sa kreacijom svog profila može kliknuti na gumb Finish. Ako su svi podaci validno uneseni kreće proces kreacije računa. To može potrajati nekoliko trenutaka ovisno o tome koliko je slika korisnik stavio na profil. Dok proces traje, korisnik vidi loading modal. Kada se sve slike uploadaju u bazu, proces kreacije računa je gotov te je korisnik registriran u aplikaciju.

Slika 17: Home stranica aplikacije

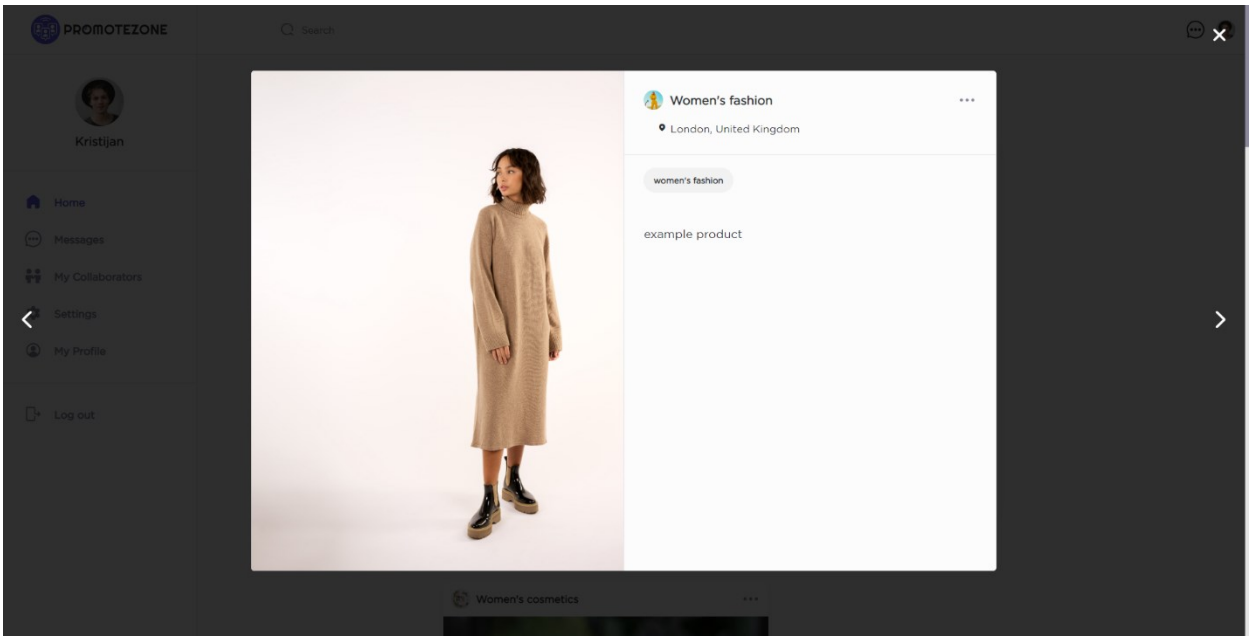


Nakon što se korisnik ulogira, odnosno kreira račun, ako ga prije nije imao, dolazi na home stranicu aplikacije. Na home stranici aplikacije može vidjeti račune druge vrste. Dakle, ukoliko se registrirao kao business račun može vidjeti promo račune, a ukoliko se registrirao kao promo račun može vidjeti business račune.

Na lijevoj strani aplikacije nalazi se na navigacijska traka koja je prisutna kroz cijelu aplikaciju. Ona sadržava linkove na druge stranice kao i link za izlogiravanje iz aplikacije. Također, u njoj se nalazi avatar sa korisničkom slikom profila i imenom te klikom na njega korisnika navigiramo na njegov korisnički profil.

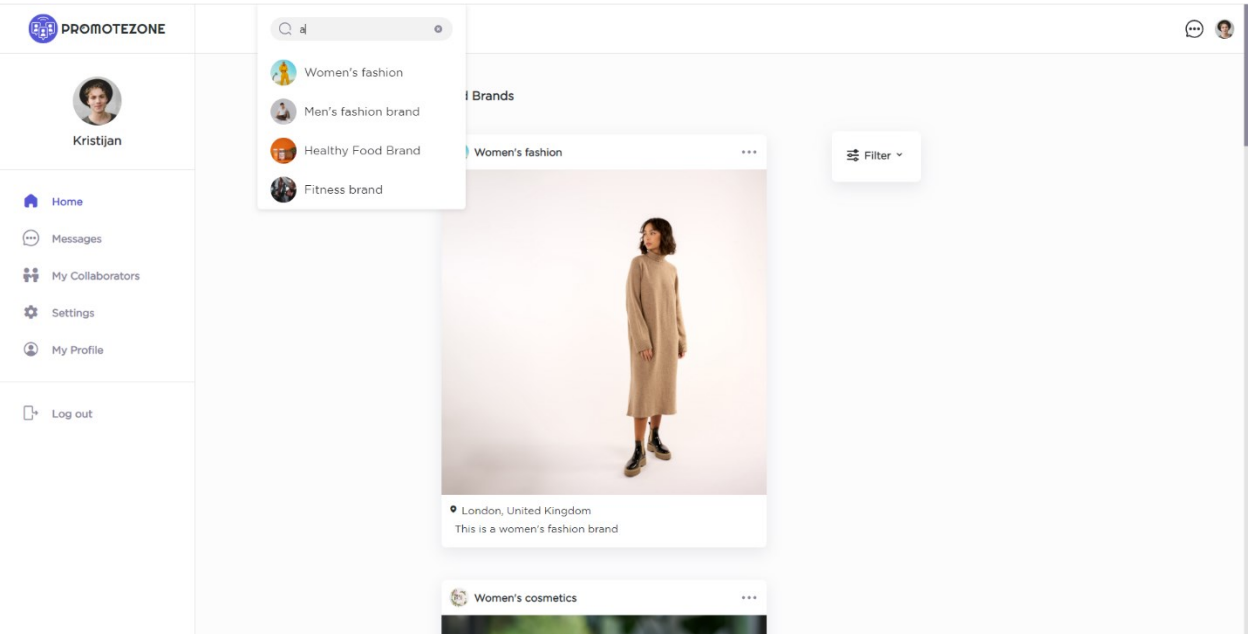
Na desnoj strani home stranice nalazi se padajući izbornik za filtriranje računa. On sadrži razne kategorije za filtriranje koje svaki račun odabire prilikom registracije. Također je omogućeno filtriranje po lokaciji tako da se odabere država iz padajućeg izbornika ili se upiše željeni grad.

Slika 18: Karusel slika profila



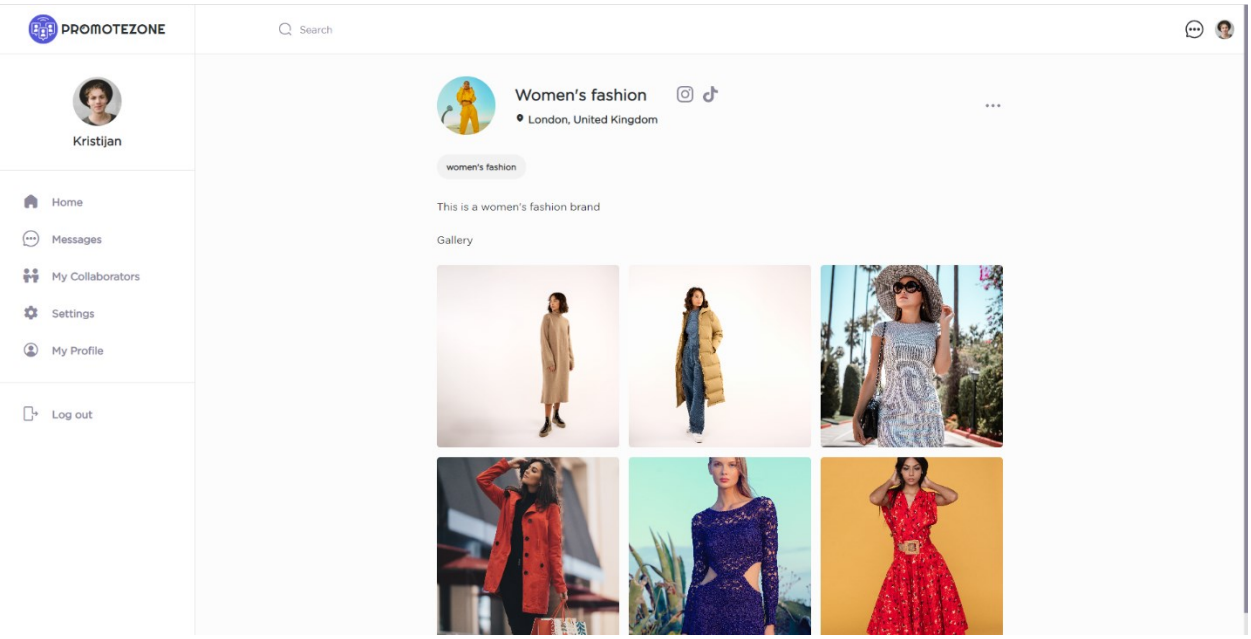
Prilikom klika na pojedinu karticu profila otvara se karusel sa svim slikama koje se nalaze na tom profilu i sa njihovim pojedinačnim opisima. Slike se uvijek vrte u krug i može se listati lijevo ili desno klikom na strelice koje se nalaze na obje strane ekrana. Također iznad opisa slike navedene su kategorije pod koje račun pripada kao i ime računa i lokacija.

Slika 19: Polje za pretraživanje



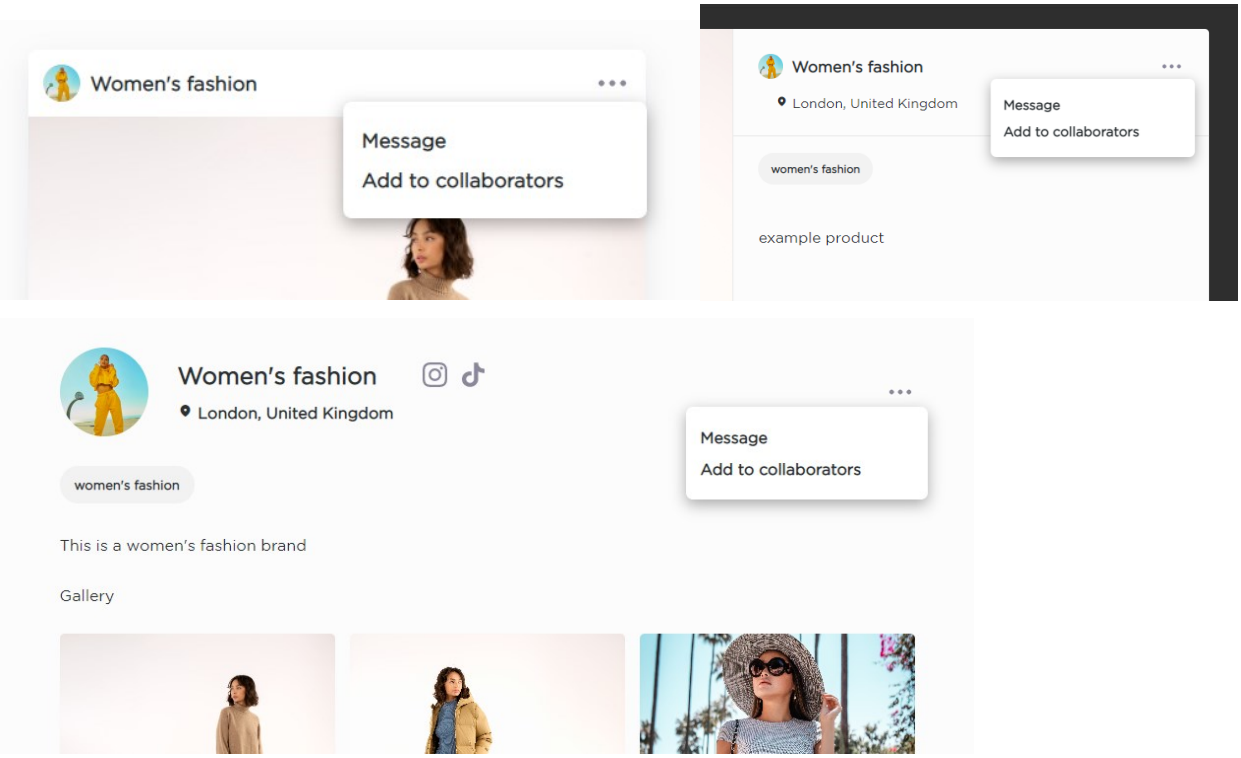
U gornjoj navigacijskoj traci nalazi se polje za pretraživanje kojim možemo pretraživati račune po imenu. Klikom na neki od ponuđenih računa otvara se pripadajući korisnički profil.

Slika 20: Korisnički profil



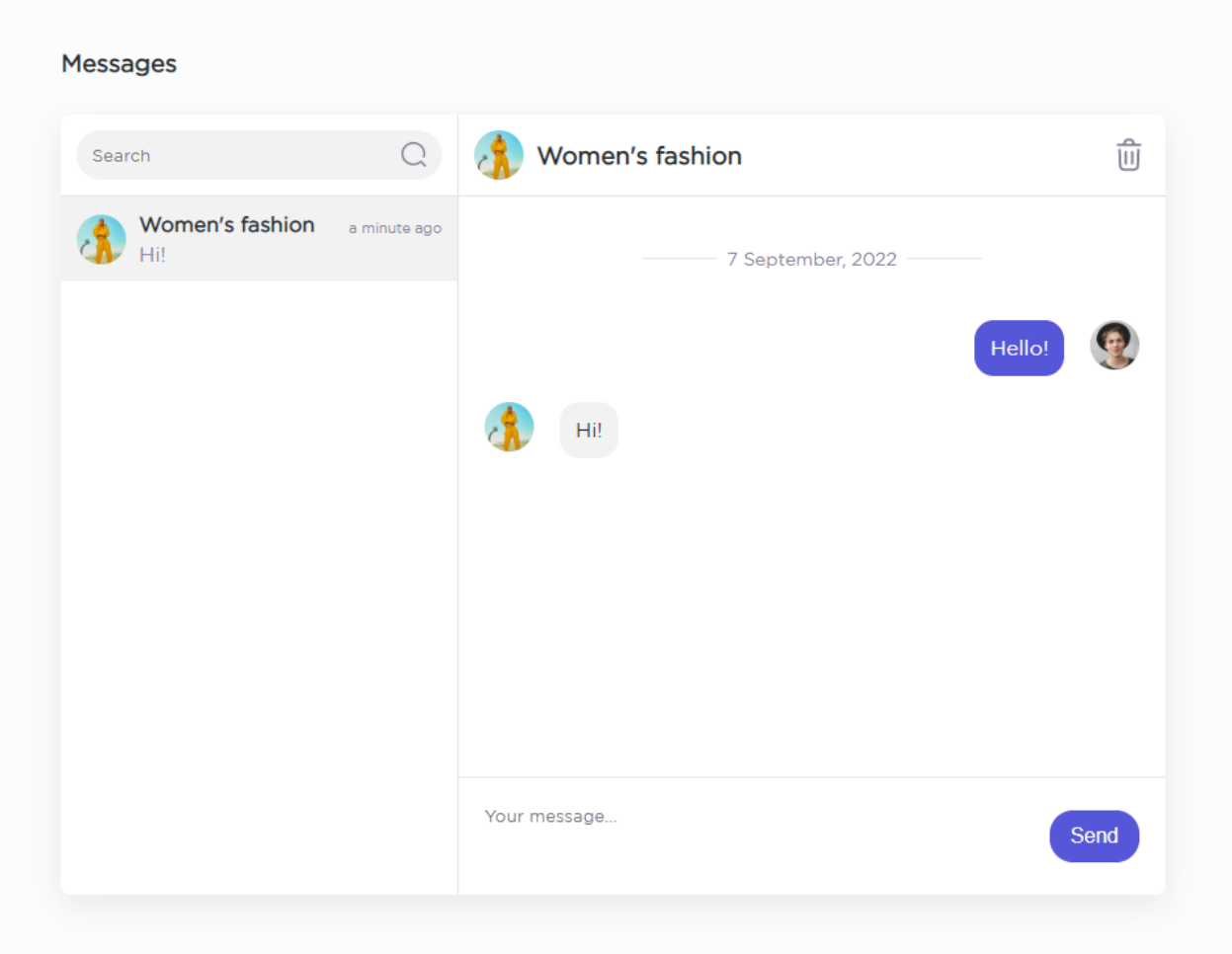
Klikom na ime profila na kartici profila, u karuselu profila ili u padajućem izborniku pretraživanja otvara se korisnički profil koji sadrži sve informacije o profilu: sliku profila, ime, povezane društvene mreže, lokaciju, kategorije profila, opis i sve slike. Klikom na pojedinu sliku također se otvara karusel slika na već prikazani način.

Slika 21: Padajući izbornik za više mogućnosti



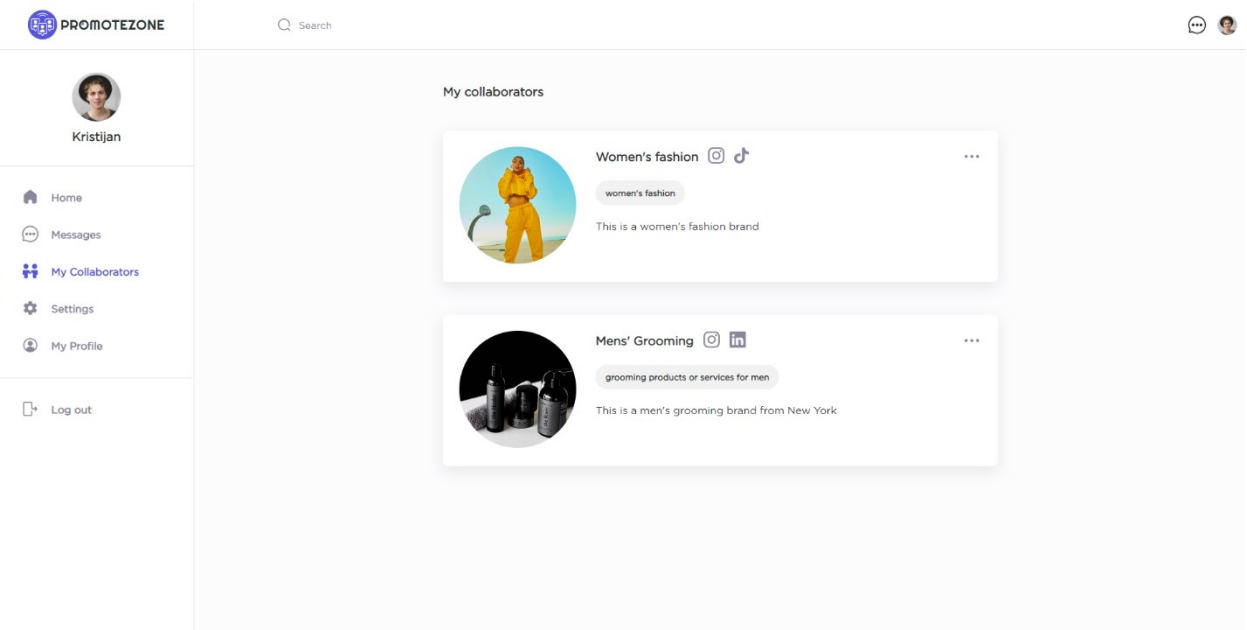
Ako korisnik prijeđe mišem preko tri točkice koje se nalaze u desnom kutu kartice profila, karusela profila i stranice profila otvara se padajući izbornik sa više mogućnosti. Klikom na „Message“ aplikacija korisnika navigira na „Messages“ stranicu i otvara razgovor sa tim profilom. Klikom na „Add to collaborators“ taj se profil dodaje na korisnikovu listu suradnika koju je moguće vidjeti na stranici „My Collaborators“. Ako se profil nalazi na listi suradnika umjesto „Add to collaborators“ pisat će „Remove from collaborators“ te klikom na to taj se profil briše iz liste suradnika.

Slika 22: Poruke



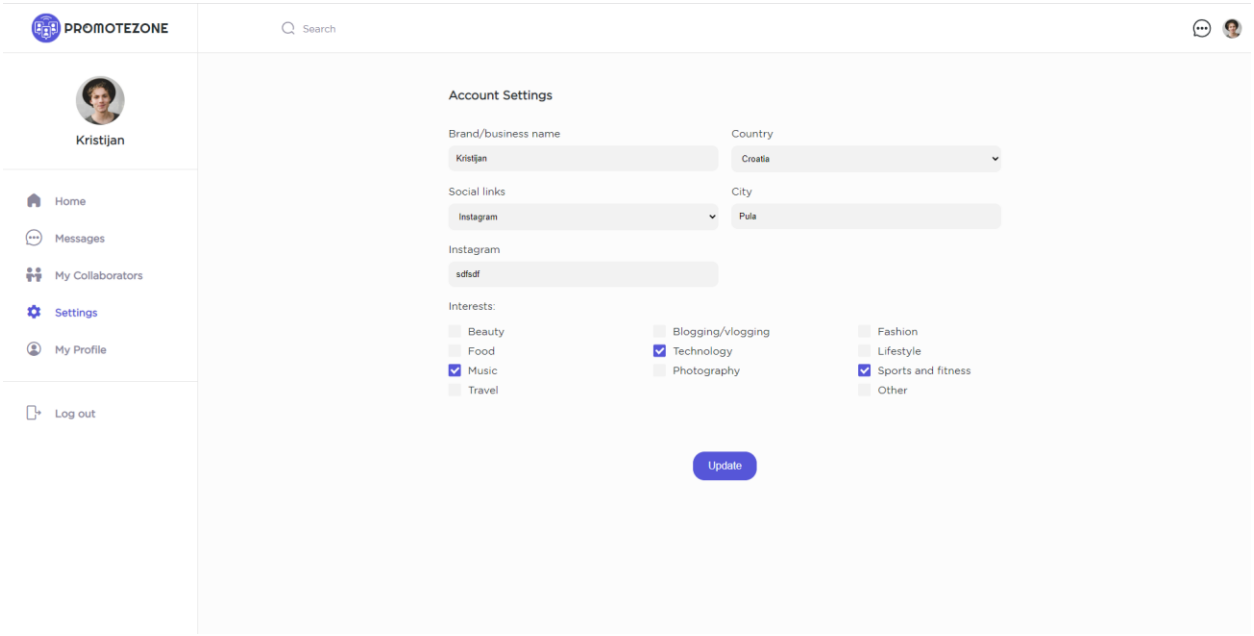
U aplikaciji je također omogućeno razmjenjivanje poruka korisnika u realnom vremenu. Na slici vidimo jedan takav razgovor. Na lijevoj strani nalazi se popis svih razgovora. Razgovor je moguće pretraživati unosom imena u gornjem lijevom kutu. Također svaki razgovor je moguće izbrisati klikom na ikonu kante za smeće u gornjem desnom kutu pojedinog razgovora.

Slika 23: Lista suradnika



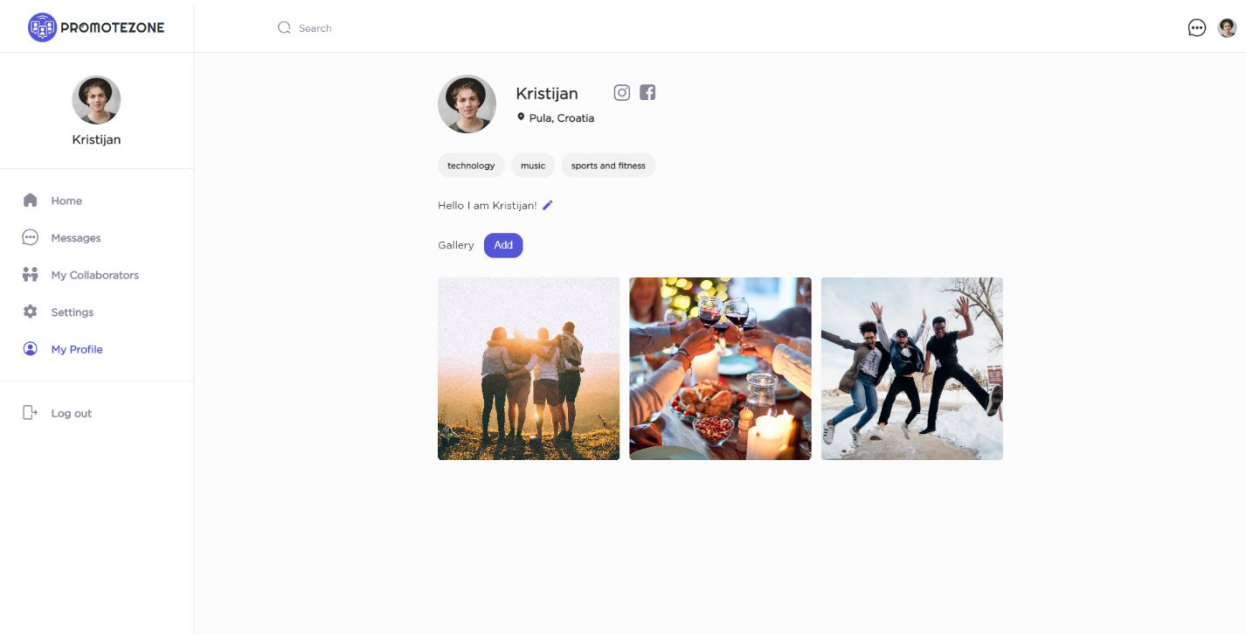
U aplikaciji je omogućeno dodavanje profila na svoju listu suradnika kako bi svi bili na jednom mjestu. Klikom na pojedinu karticu suradnika otvara se njegov korisnički profil. Također, na već opisani način se prelaženjem miša preko tri točkice u kutu otvara padajući izbornik sa više mogućnosti gdje se taj profil može izbrisati iz liste suradnika ili se može otvoriti razgovor s njime.

Slika 24: Postavke računa



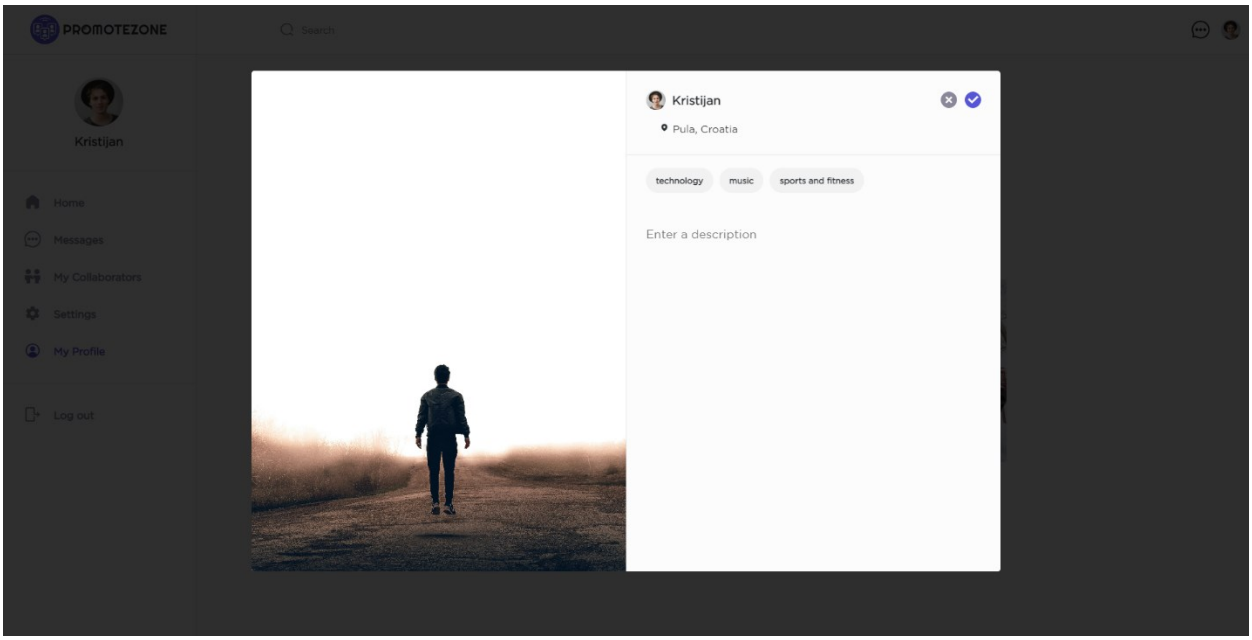
U postavkama je moguće promijeniti unesene podatke računa kao što su: ime računa, lokacija, povezane društvene mreže te kategorije u koje profil pripada.

Slika 25: Profil trenutnog korisnika

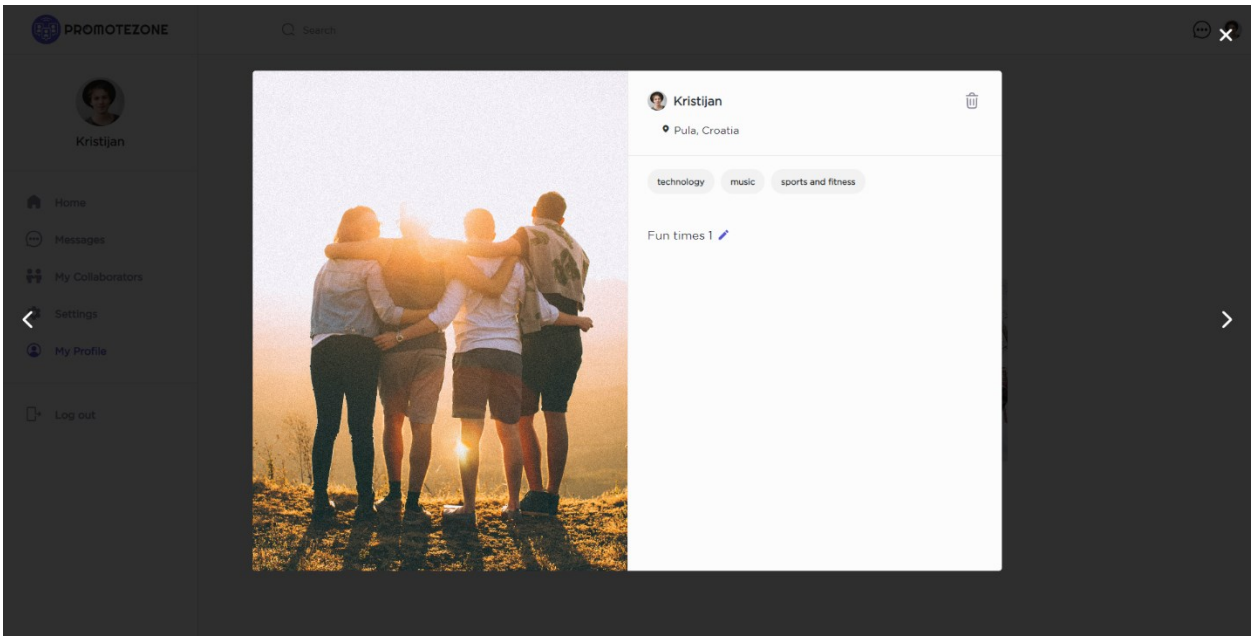


Korisnik može vidjeti svoj profil i mijenjati podatke na njemu. Klikom na sliku profila može dodati drugu sliku profila. Klikom na ikonu olovke pored opisa može promijeniti opis svog profila. Također može dodati novu sliku na svoj profil klikom na gumb „Add“. Korisnik također može uređivati opise slika koje se već nalaze na njegovom profilu klikom na ikonu olovke koja se nalazi pored opisa pojedine slike te ih također može izbrisati klikom na ikonu kante za smeće koja se nalazi u gornjem desnom kutu karusela pojedine slike.

Slika 26: Dodavanje nove slike na profil



Slika 27: Uređivanje ili brisanje postojećih slika



7.2. KORIŠTENE TEHNOLOGIJE

7.2.1. VUE JS

Vue JS je, uz React i Angular jedan od tri najpopularnija frontend razvojna okvira za Javascript programski jezik. Njegov tvorac je Evan You i prvo izdanje je nastalo 2014. godine. Vue nudi brojne mogućnosti za developere i time im olakšava posao razvoja aplikacija jer bi korištenjem vanilla JavaScripta proces razvoja bio mnogo duži i teži. Prema anketi provedenoj od strane Stack Overflowa, stranice za pitanja i odgovore namijenjene za profesionalne programere i programere entuzijaste, provedene 2022. godine Vue JS je prema popularnosti treći od spomenutih tehnologija, odmah ispod Angulara sa 18.82% odgovora.

Vue JS služi za izradu dinamičkih single page web aplikacija. Single page aplikacija unutar Vue JS – a funkcionira na način da se sastoji se od jedne index.html datoteke koja služi kao korijen u koji se dinamično učitava sadržaj. Korisnik ima dojam da prelazi s jedne stranice u drugu međutim to zapravo nije slučaj već se stranice prikazuju dinamično. To je puno bolji, brži i efikasniji način konstruiranja aplikacija od klasičnog načina u kojem je svaka stranica zasebna html datoteka. Na taj je način prelazak između stranica momentalan jer aplikacija ne mora učitavati svaku zasebnu stranicu.

Vue JS dolazi sa određenim alatima i konceptima koji olakšavaju proces izrade aplikacija i štedi vrijeme jer bi korištenjem običnog JavaScripta proces bio puno teži i zahtjevniji. Srž Vue JS – a su komponente. Svaka aplikacija ima jednu komponentu roditelja koja se obično zove App.vue. Ta se komponenta umeće u korijen index.html datoteku i to je ulazna točka u Vue aplikaciju. Sve ostale komponente su komponente djeca komponente App.vue. Komponente djeca će biti zasebne stranice koje želimo prikazati koje također mogu imati neke svoje komponente djecu. To mogu biti komponente koje se koriste na više mjesta te ih želimo dijeliti sa ostalim komponentama te ih ima smisla odvojiti u zasebnu komponentu. To može biti primjerice neki padajući izbornik koji koristimo na više mjesta u aplikaciji ili kartica profila koja se ponavlja za svaki profil i sl. U Vue JS – u naglasak se stavlja na ponovnom iskorištavanju koda.

Svaka komponenta se sastoji od template dijela, script dijela i style dijela. U template dijelu piše se html koji može biti obogaćen direktivama koje dolaze iz Vue – a. U script dijelu piše se JavaScript, dakle logika koja je potrebna za izvršavanje funkcionalnosti pojedine komponente. U style dijelu piše se CSS za svaku komponentu.

7.2.2. FIREBASE

Firebase je Google-ova platforma za kreiranje web i mobilnih aplikacija. Nastao je 2011. godine kao neovisna kompanija dok ju 2014. godine nije preuzeo Google. Firebase u svojoj suštini nudi uslugu backenda za aplikacije. Nudi raznovrsne pakete koji su potrebni u razvoju većine aplikacija poput Cloud Firestore baze podataka, autentifikacije korisnika, storage-a za datoteke, hostinga i sl. S tim paketima aplikacija komunicira putem funkcija definiranih iz Firebase-a. Pakete koje se žele koristiti potrebno je instalirati u aplikaciju. Nakon toga je moguće zvati definirane funkcije koje dolaze iz tih paketa koje će komunicirati sa Firebase serverima te raditi određene akcije u bazi koja se nalazi u oblaku.

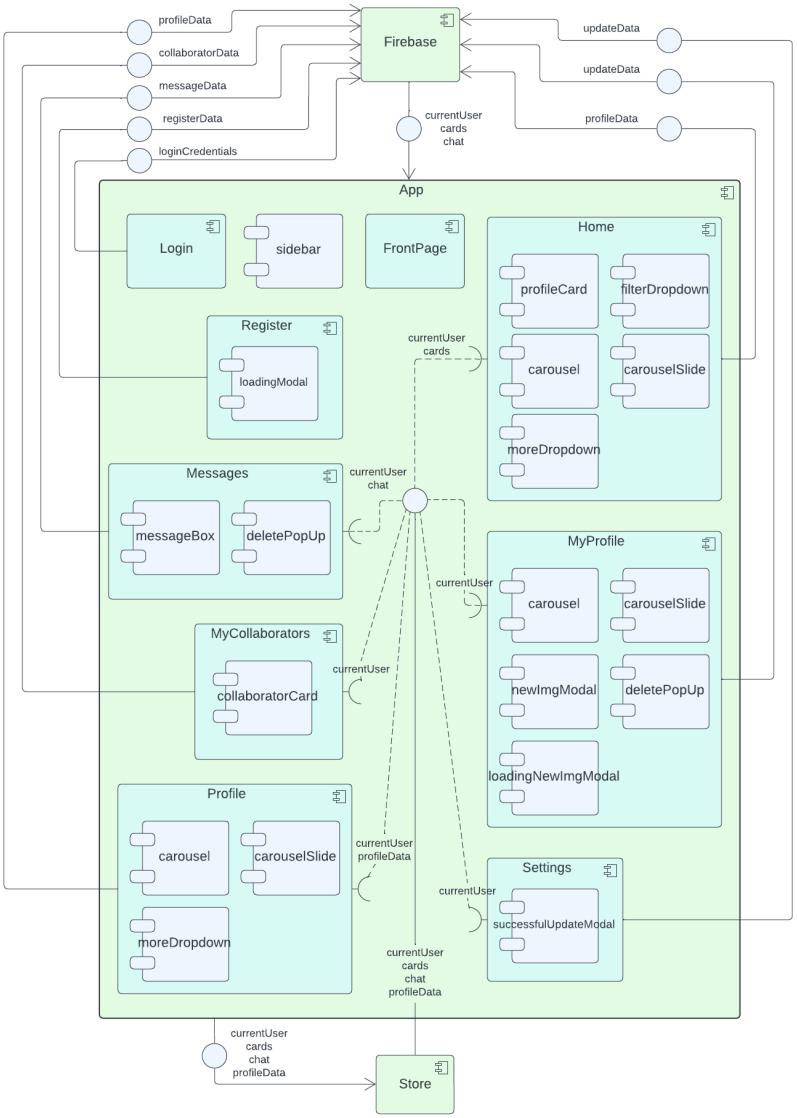
7.2.3. SASS

SASS je CSS predprocesor. Sastoji se od dvije sintakse, originalna SASS sintaksa i SCSS (Sassy CSS) sintaksa koja je vrlo slična klasičnom CSS - u. SASS je u suštini proširena verzija CSS - a odnosno kako se to popularno voli reći CSS na steroidima. Nudi puno više mogućnosti od CSS - a poput definiranja varijabli, objekata, funkcija, mixina i sl. On je u svojoj suštini jezik za skriptiranje koji se prije izvršavanja aplikacije prevede u klasični CSS koji preglednik može razumjeti. Zbog toga je SASS CSS predprocesor.

7.3. ARHITEKTURA APLIKACIJE

U ovome će dijelu biti objašnjena arhitektura aplikacije Promotezone, kako je ona strukturirana i bit će prikazani i objašnjeni neki najvažniji dijelovi koda po komponentama. Za vizualizaciju komponenti aplikacije te kako su one povezane između sebe i sa Firebase – om možemo se poslužiti UML komponentnim dijagramom.

Dijagram 3: UML komponentni dijagram aplikacije Promotezone



UML komponentni dijagram služi za prikazivanje komponenti aplikacije i vizualizaciju toka podataka između komponenti.

Aplikacija Promotezone sastoji se od jedne glavne App komponente u kojoj se nalaze sve ostale komponente. To su komponente koje će prikazivati zasebne stranice poput FrontPage komponente, Login komponente, Register komponente koja sadrži loadingModal komponentu, Home komponente koja sadrži profileCard, filterDropdown, carousel, carouselSlide i moreDropdown komponente, Messages komponente koja sadrži messageBox i deletePopUp komponente, MyCollaborators komponente koja sadrži collaboratorCard komponentu, Profile komponente koja sadrži carousel, carouselSlide i moreDropdown komponente, MyProfile komponente koja sadrži carousel, carouselSlide, newImgModal, deletePopUp i loadingNewImgModal komponente te Settings komponente koja sadrži successfulUpdateModal komponentu. Također unutar App komponente nalazi se komponenta sidebar koja će se prikazivati na svakoj stranici koju korisnik može vidjeti ako je ulogiran.

Također na dijagramu je korištenjem tzv. interface – a (kružići) moguće prikazati koje to podatke komponente pružaju, primaju te zahtijevaju. Kada se korisnik ulogira ili registrira, u Firebase se iz Login ili Register komponente šalju podaci za logiranje odnosno registriranje korisnika. Firebase zatim obrađuje te podatke te u App komponentu šalje podatke o trenutnom korisniku, njegovim razgovorima i dohvaća profile koji će se prikazivati na Home komponenti. App komponenta zatim u globalnu store komponentu prosljeđuje te podatke kako bi im sve druge komponente mogle pristupiti. Korištenjem tzv. dependency veze koja se označava iscrtanom linijom i polukruga označavamo da određena komponenta zahtijeva neke podatke. Svaka komponenta za koju korisnik mora biti ulogiran zahtijevat će njegove podatke. Home komponenta također će zahtijevati podatke o profilima koje mora prikazati. Messages komponenta zahtijevat će podatke o porukama trenutnog korisnika. Profile komponenta koja predstavlja neki korisnički profil zahtijevat će podatke o tom profilu. Ti podaci bit će spremljeni u store i proslijeđeni u Profile komponentu kada korisnik klikne na ime ili sliku profila koji se prikazuju u određenim komponentama.

Kada korisnik doda nekog drugog korisnika u listu suradnika, obriše ga ili otvori razgovor sa njime, komponenta koja sadrži tu funkcionalnost (ona koja sadrži moreDropdown komponentu) šalje podatke u Firebase koji zatim ažurira potrebne dokumente. Ona također zatim ažurira lokalni store kako ne bismo morali ponovno zvati dohvaćanje iz Firebase – a. Kada korisnik ažurira neke svoje podatke bilo na MyProfile komponenti ili na Settings komponenti, u Firebase se šalju podaci za ažuriranje te se također ažurira i lokalni store. Kada korisnik pošalje poruku, Messages komponenta će proslijediti podatke poruke u Firebase koji će zatim ažurirati potrebne dokumente. Budući da koristimo „onSnapshot“ funkciju iz Firebase – a koja okida dohvaćanje za svaki novi upis u tu kolekciju, promjene će odmah biti vidljive i u aplikaciji.

7.3.1. App komponenta

Po objašnjenom principu rada SPA (Single Page Aplikacija) „PromoteZone“ aplikacija se sastoji od jedne App.vue komponente. Vue radi tako da sve ostale komponente odnosno stranice budu komponente djeca te komponente. Sav kod koji se napiše u App.vue komponenti odrazit će se na cijelu aplikaciju.

Zbog načina na koji se ponaša App komponenta, ona može biti dobro mjesto za staviti elemente koji će se ponavljati kroz cijelu aplikaciju. Poput primjerice navbara odnosno navigacijske trake koja će biti prisutna kroz cijelu aplikaciju. Na slici 23 prikazan je dio koda navbara kojeg želimo prikazivati kroz cijelu aplikaciju pod uvjetom da imamo korisnika. Za to koristimo v-if direktivu iz Vue – a koja će prikazati određeni sadržaj ako je neki uvjet istinit. U ovom slučaju ako postoji korisnik tj. ako je on ulogiran. To će spriječiti pojavljivanje navbara na mjestima na kojima to ne želimo poput na naslovnoj stranici te stranici za logiranje i registraciju. Slična je stvar sa sidebarom aplikacije kojeg želimo prikazati kroz cijelu aplikaciju ako je korisnik ulogiran.

Slika 28: Dio template dijela App komponente

```
<template>
  <div id="app">
    <div class="nav" v-if="store.currentUser">
      <div class="nav__logo">
        <router-link to="/" href="#">
          <img
            src="@/assets/Logo-purple.svg"
            alt="PromoteZone-logo"
            height="45"
          />
        </router-link>
      </div>

      <div class="nav__search">
        <input
          placeholder="Search"
          v-model="searchTerm"
          :style="[searchTerm ? { background: '#f2f2f2' } : {}]"
        />
        <icon-library name="searchInput" class="nav__search__icon" />
        <img
          src="@/assets/x.svg"
          class="x-icon"
          v-if="searchTerm"
          @click="searchTerm = ''"
        />
        <div class="nav__search__box" v-if="searchTerm">
          <div class="nav__search__box__results">
            <div
              v-for="card in searchedCards"
              :key="card.id"
              class="nav__search__box__results__result"
            >
              <div
                @click="
                  store.profileData = card;
                  searchTerm = '';
                "
              >
                <router-link
                  :to="{
                    name: 'Profile',
                    params: { profileName: card.name },
                  }"
                >
                  <img :src="card.profileImg" />
                  <p>
                    {{ card.name }}
                  </p>
```

Unutar App komponente nalazi se Firebase funkcija koja se pokreće kada se promijeni stanje autentifikacije korisnika, u prijevodu kada se korisnik registrira, ulogira ili izlogira. To je važno zato što kada imamo ulogiranog korisnika želimo ga automatski preusmjeriti na Home stranicu te želimo u store spremiti njegove podatke kako bi ih ostale komponente mogle koristiti, te također želimo pozvati funkcije za dohvaćanje profila koji će se prikazati na Home stranici te korisnikovih poruka. Ukoliko se stanje autentifikacije korisnika promijenilo na način da nemamo korisnika tj. korisnik se izlogirao, tada ga preusmjeravamo na naslovnu stranicu.

Slika 29: On auth state changed funkcija u App komponenti

```
onAuthStateChanged(auth, async (user) => {
  const currentRoute = router.currentRoute;

  if (currentRoute.name == "Register") {
    setTimeout(async () => {
      if (user) {
        const docRef = doc(db, "accounts", user.uid);
        const docSnap = await getDoc(docRef);
        if (docSnap.exists()) {
          store.currentUser = docSnap.data();
          await profileService.getProfiles();
          await messagesService.getMessages();
        } else {
          // doc.data() will be undefined in this case
          console.log("No such document!");
        }
        if (!currentRoute.meta.needsUser) {
          router.push({ name: "Home" });
        }
      } else {
        store.currentUser = null;
        console.log("No user");
      }
    }, store.loadingTime);
  } else {
    if (user) {
      const docRef = doc(db, "accounts", user.uid);
      const docSnap = await getDoc(docRef);
      if (docSnap.exists()) {
        store.currentUser = docSnap.data();
        await profileService.getProfiles();
        await messagesService.getMessages();
      } else {
        // doc.data() will be undefined in this case
        console.log("No such document!");
      }
      if (!currentRoute.meta.needsUser) {
        router.push({ name: "Home" });
      }
    } else {
      store.currentUser = null;
      console.log("No user");
      if (currentRoute.meta.needsUser) {
        router.push({ name: "FrontPage" });
      }
    }
  }
});
```

Slika 30: Funkcija za dohvaćanje profila

```
import store from "@/store";
import {collection, getDocs, orderBy, query, where, db} from "@/firebase";

export default {

async getProfiles() {
    if (store.currentUser.accType === "business") {
      const q = query(
        collection(db, "accounts"),
        where("accType", "==", "promo"),
        orderBy("name", "asc")
      );
      const querySnapshot = await getDocs(q);
      store.cards = [];
      querySnapshot.forEach((doc) => {
        const data = doc.data();
        store.cards.push({
          id: doc.id,
          accType: data.accType,
          bio: data.bio,
          categories: data.categories,
          city: data.city,
          country: data.country,
          facebook: data.facebook,
          images: data.images,
          instagram: data.instagram,
          linkedIn: data.linkedIn,
          name: data.name,
          profileImg: data.profileImg,
          tiktok: data.tiktok,
          youtube: data.youtube,
          collaborators: data.collaborators,
          chatedWith: data.chatedWith
        });
      });
    } else {
```

Kada imamo ulogiranog korisnika i spremili smo njegove podatke u store, možemo provjeriti tip njegovog računa. Ukoliko je tip njegovog računa jednak „bussiness“ onda dohvaćamo iz Firebase kolekcije račune čiji je tip jednak „promo“. Dohvaćene račune s njihovim podacima spremamo u store kako bi ih mogli ispisati u Home komponenti.

Slika 31: Funkcija za dohvaćanje profila 2. dio

```
    } else {
      const q = query(
        collection(db, "accounts"),
        where("accType", "==", "business"),
        orderBy("name", "asc")
      );
      const querySnapshot = await getDocs(q);
      store.cards = [];
      querySnapshot.forEach((doc) => {
        const data = doc.data();
        store.cards.push({
          id: doc.id,
          accType: data.accType,
          bio: data.bio,
          categories: data.categories,
          city: data.city,
          country: data.country,
          facebook: data.facebook,
          images: data.images,
          instagram: data.instagram,
          linkedIn: data.linkedIn,
          name: data.name,
          profileImg: data.profileImg,
          tiktok: data.tiktok,
          youtube: data.youtube,
        });
      });
    }
  },
}
```

Ukoliko tip računa korisnika nije „business“, dakle „promo“ je, onda dohvaćamo iz Firebase kolekcije račune čiji je tip „business“ i spremamo njihove podatke u store kako bi ih kasnije mogli ispisati u Home komponenti.

Slika 32: Funkcija za dohvaćanje korisnikovih poruka

```
import store from "@/store";
import { collection, onSnapshot, orderBy, query, where, db } from "@/firebase";

export default {
  async getMessages() {
    const q = query(
      collection(db, "chat"),
      where("users", "array-contains", store.currentUser.uid),
      orderBy("lastUpdated", "desc")
    );
    onSnapshot(q, (querySnapshot) => {
      store.chat = [];
      querySnapshot.forEach((doc) => {
        const data = doc.data();
        const user = data.users.filter((user) => user != store.currentUser.uid);
        const chatedWith = store.cards.filter((card) => card.id == user);
          store.chat.push({
            id: doc.id,
            messages: data.messages,
            chatedWith: chatedWith[0],
            lastUpdated: data.lastUpdated
          })
      });
    });
  },
};
```

Kada imamo ulogiranog korisnika također želimo dohvatiti njegove razgovore. Iz Firebase kolekcije „chat“ dohvaćamo sve razgovore koji uključuju trenutnog korisnika. Korištenjem „onSnapshot “ funkcije iz Firebase – a pretplaćujemo se na svaku promjenu u tom upitu iz kolekcije što će osigurati da se dohvaćanje poruka odvija svaki put kada se dogodi promjena, odnosno svaki puta kada korisnik primi ili pošalje novu poruku. Dohvaćeni podaci sastoje se od arraya objekata svih poruka, gdje svaka poruka ima vrijednost, ID korisnika koji ju je poslao i timestamp te od arraya koji sadrži ID – e svih korisnika koji sudjeluju u razgovoru. Prvi korak je utvrditi ID korisnika s kojim trenutni, ulogirani korisnik vodi razgovor. Iz arraya ID – eva korisnika koji sudjeluju u razgovoru filtriramo, odnosno mičemo ID trenutnog korisnika. Sljedeće, iz prethodno dohvaćenih profila korisnika filtriramo sve profile koji ne sadrže prethodno utvrđeni ID korisnika s kojim se vodi razgovor te tako dobivamo sve podatke korisnika s kojim se vodi razgovor. Na kraju, sve podatke spremamo u store kako bi ih mogli koristiti tamo gdje je potrebno.

7.3.2. Store

Slika 33: Globalna store datoteka

```
export default {
  currentUser: null,
  chat: [],
  visibleChat: null,
  specificChatOpened: false,
  width: null,
  messageBoxVisible: true,
  messageListVisible: true,
  loadingTime: 0,
  cards: [],
  filterOptions: {
    categories: [],
    country: "",
    city: "",
  },
  profileData: null,
};
```

Vrlo važan dio aplikacije je globalna store datoteka koja nam služi za dijeljenje podataka između komponenata. Tako se u store datoteci nalazi jedan zadani objekt koji će sadržavati podatke o trenutnom korisniku, podatke o razgovorima trenutnog korisnika, postavljat će se vrijednost vidljivog razgovora kada korisnik klikne na message u moreDropdown komponenti (kako bi se otvorio razgovor baš s tim korisnikom), postavljat će se vrijednost varijable „specificChatOpened“ koja će biti istina ili laž (služi Messages komponenti da ukoliko nije kliknut niti jedan razgovor pokaže onaj prvi), također nalazi se varijabla u kojoj je spremljena širina ekrana koja će diktirati ponašale sidebara, „messageBoxVisible“ i „messageListVisible“ varijabla koje će diktirati ponašanje Message komponente ovisno o širini ekrana, „loadingTime“ varijabla koju prosljeđujemo u App komponentu iz Registera kako bi pričekali sa preusmjeravanjem korisnika dok se ne učitaju sve slike u bazu, cards varijabla koja će sadržavati profile koje prikazujemo na Home komponenti, „filterOptions“ objekt sa vrijednostima za filtriranje koje prosljeđujemo iz filterDropdown komponente u Home komponentu i „profileData“ varijabla u koju će se spremiti podaci o profilu na koji korisnik klikne koji će se koristiti u Profile komponenti.

7.3.3. Register komponenta

Svaka Vue komponenta u svom script dijelu ima jedan zadani objekt. Unutar tog objekta nalazi se data funkcija u kojoj su sadržani svi podaci s kojima ta komponenta barata. To mogu biti podaci koje korisnik popunjava i koji se kasnije spremaju u bazu ili mogu biti podaci koji se mijenjaju ovisno o nekom uvjetu, primjerice da li je neki loading modal otvoren ili nije. U primjeru iz slike 24. vidimo data funkciju komponente Register koja sadrži sve podatke koje korisnik mora popuniti u procesu registracije. Za povezivanje onoga što korisnik upiše u polje za unos sa određenim podatkom u data funkciji koristi se direktiva iz Vue – a koja se zove v-model.

Slika 34: Data dio komponente Register

```
923     getDownloadURL,
924     arrayUnion,
925   } from "@/firebase";
926
927   export default {
928     name: "Register",
929     data: function () {
930       return {
931         store,
932         openLoading: false,
933         loading: true,
934         step: 1,
935         msg: [],
936         visible: "password",
937         visibleRepeat: "password",
938         linkChosen: "Instagram",
939         fileUrl: [],
940         auth: {
941           email: null,
942           password: null,
943           repeatPassword: null,
944         },
945         registration: {
946           accType: "business",
947           name: null,
948           country: "Croatia",
949           city: null,
950           socialLinks: {
951             instagram: null,
952             facebook: null,
953             tiktok: null,
954             youtube: null,
955             linkedIn: null,
956           },
957           categories: [],
958           profileImgReference: null,
959           bio: null,
960           files: [],
961         },
962       };
963     },
964
965     components: {
966       IconLibrary,
967       loadingModal,
968     },
969     methods: {
970       next() {
```

U script dijelu također se može nalaziti objekt svih komponenata koje ta komponenta koristi. Dakle komponente djeca te komponente poput modala, padajućih izbornika i sl.

Nadalje, u script dijelu nalazit će se bilo koje metode odnosno funkcije koje komponenta izvršava.

Slika 35: Metode u Register komponenti

```
methods: {
  next() {
    this.step++;
  },
  back() {
    this.step--;
  },
  showPassword() {
    this.visible = "text";
  },
  hidePassword() {
    this.visible = "password";
  },
  showPasswordRepeat() {
    this.visibleRepeat = "text";
  },
  hidePasswordRepeat() {
    this.visibleRepeat = "password";
  },
  openInput() {
    this.$refs.fileInput.click();
  },
  uploadFile(event) {
    const selectedFiles = event.target.files;
    const self = this;
    for (var i = 0; i < selectedFiles.length; i++) {
      const file = selectedFiles[i];
      if (!file.type.match("image.*")) {
        continue;
      }
      setTimeout(() => {
        this.registration.files.push({
          img: file,
          desc: "",
        });
        const reader = new FileReader();
        reader.onload = function (event) {
          self.fileUrl.push(event.target.result);
        };
        reader.readAsDataURL(file);
      }, i * 1000);
    }
  },
  removeFile(index) {
    this.registration.files.splice(index, 1);
    this.fileUrl.splice(index, 1);
    this.$refs.fileInput.value = null;
```

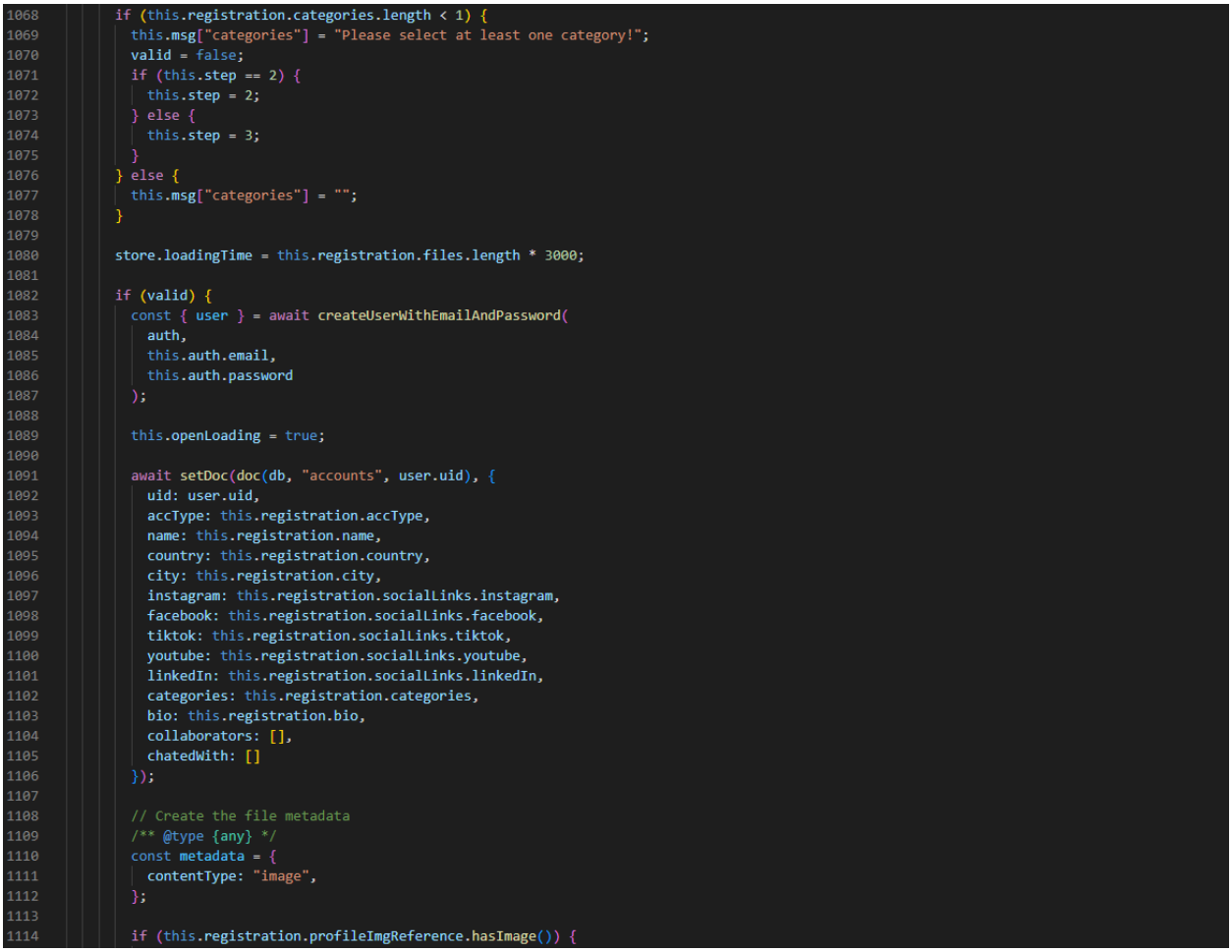
Slika 36: Finish metoda u Register komponenti, 1. dio

```
async finish() {
  let valid = true;

  if (
    /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(this.auth.email)
  ) {
    this.msg["email"] = "";
    valid = true;
  } else {
    this.msg["email"] = "Invalid Email Address";
    this.step = 2;
    valid = false;
  }

  if (!this.auth.password || this.auth.password.length < 6) {
    this.msg["password"] = "Password must be at least 6 characters";
    this.step = 2;
    valid = false;
  } else {
    this.msg["password"] = "";
    valid = true;
  }

  if (this.auth.password !== this.auth.repeatPassword) {
    this.msg["repeatPassword"] = "Password doesn't match";
    this.step = 2;
    valid = false;
  } else {
    this.msg["repeatPassword"] = "";
  }

  if (!this.registration.name) {
    this.msg["name"] = "This field cannot be empty";
    this.step = 2;
    valid = false;
  } else {
    this.msg["name"] = "";
  }

  if (!this.registration.city) {
    this.msg["city"] = "This field cannot be empty";
    this.step = 2;
    valid = false;
  } else {
    this.msg["city"] = "";
  }

  if (this.registration.categories.length < 1) {
```

Najznačajnija metoda u Register komponenti je finish metoda koja se pokreće klikom na gumb „Finish“ u zadnjem koraku registracijskog procesa.

Prvi korak u metodi je provjeriti da li su sva polja ispravno popunjena. Korisnik mora unijeti validnu email adresu, lozinka mora imati barem 6 znamenki, ponovljena lozinka mora biti ista te polja za ime, grad i izabrane kategorije ne smiju biti prazna. Ako bilo koji od tih uvjeta nije zadovoljen postavljaju se relevantne error poruke, korisnika se vraća na korak gdje je napravio grešku u unosu te varijabla valid postaje neistinita i funkcija prestaje sa izvršavanjem.

Slika 37: Finish metoda u Register komponenti, 2. dio



```
1068        if (this.registration.categories.length < 1) {
1069          this.msg["categories"] = "Please select at least one category!";
1070          valid = false;
1071          if (this.step == 2) {
1072            this.step = 2;
1073          } else {
1074            this.step = 3;
1075          }
1076        } else {
1077          this.msg["categories"] = "";
1078        }
1079
1080        store.loadingTime = this.registration.files.length * 3000;
1081
1082        if (valid) {
1083          const { user } = await createUserWithEmailAndPassword(
1084            auth,
1085            this.auth.email,
1086            this.auth.password
1087          );
1088
1089          this.openLoading = true;
1090
1091          await setDoc(doc(db, "accounts", user.uid), {
1092            uid: user.uid,
1093            accType: this.registration.accType,
1094            name: this.registration.name,
1095            country: this.registration.country,
1096            city: this.registration.city,
1097            instagram: this.registration.socialLinks.instagram,
1098            facebook: this.registration.socialLinks.facebook,
1099            tiktok: this.registration.socialLinks.tiktok,
1100            youtube: this.registration.socialLinks.youtube,
1101            linkedIn: this.registration.socialLinks.linkedIn,
1102            categories: this.registration.categories,
1103            bio: this.registration.bio,
1104            collaborators: [],
1105            chatedWith: []
1106          });
1107
1108          // Create the file metadata
1109          /** @type {any} */
1110          const metadata = {
1111            contentType: "image",
1112          };
1113
1114          if (this.registration.profileImgReference.hasImage()) {
```

Ako je varijabla valid istinita tj. svi podaci su točno ispunjeni funkcija kreće sa daljnjim izvršavanjem. Poziva se asinkrona funkcija iz Firebase – a za stvaranje korisnika u koju prosljeđujemo podatke koje je korisnik unio, njegov i email i lozinku. Asinkrona funkcija, što i sama riječ govori, znači da se funkcija ne izvršava sinkrono tj. ne izvršava se u točnom momentu pozivanja. To je tako zato što ona mora pozvati neki zahtjev na neki udaljeni server što se ne može izvršiti momentalno. Zato asinkrone funkcije vraćaju tzv. promise odnosno obećanje da će se to nešto izvršiti i funkcija ide dalje. Jedno od rješenja za asinkrone funkcije je korištenje ključnih riječi async i await. Prije imena funkcije u ovom slučaju finish funkcije stavimo ključnu riječ async kako bi naznačili da funkcija sadrži asinkroni kod. U pozivanju neke asinkrone funkcije, u ovom slučaju funkcije iz Firebase – a za kreiranje korisnika stavimo ključnu riječ await i time naznačimo kako želimo pričekati rezultat koji će nam ta funkcija vratiti. U ovom slučaju kada se funkcija za

kreiranje korisnika izvrši ona nam vraća tog kreiranog korisnika kojeg potom spremamo u varijablu user. Nakon što imamo kreiranog korisnika tada možemo pristupiti kreiranju dokumenta u Firestore bazi podataka za tog korisnika i spremiti njegove podatke.

Slika 38: Finish metoda u Register komponenti, 3. dio

```
if (this.registration.profileImgReference.hasImage()) {
  this.registration.profileImgReference.generateBlob((blobData) => {
    let profileImgName = this.registration.name + "-profile-image.png";

    const storageRef = ref(
      storage,
      this.registration.name + `(${user.uid})/` + profileImgName
    );
    const uploadProfileImg = uploadBytesResumable(
      storageRef,
      blobData,
      metadata
    );
    uploadProfileImg.on(
      "state_changed",
      (snapshot) => {
        const progress =
          (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
        console.log("Upload is " + progress + "% done");
        switch (snapshot.state) {
          case "paused":
            console.log("Upload is paused");
            break;
          case "running":
            console.log("Upload is running");
            break;
        }
      },
      (error) => {
        switch (error.code) {
          case "storage/unauthorized":
            break;
          case "storage/canceled":
            break;
          case "storage/unknown":
            break;
        }
      },
      async () => {
        const profileImgUrl = await getDownloadURL(
          uploadProfileImg.snapshot.ref
        );
        await updateDoc(doc(db, "accounts", user.uid), {
          profileImg: profileImgUrl,
        });
      }
    );
```

Sljedeći korak je učitavanje profilne slike u Firebase Storage. Generiramo binarne podatke za sliku te pristupamo storage – u i kreiramo u njemu mapu za novog korisnika i krećemo u učitavanje slike. U konzoli nam se ispisuje postotak izvršenja učitavanja. Kada je učitavanje gotovo pozivanjem funkcije iz Firebase – a dobivamo adresu odnosno URL te slike u Firebase Storage – u. Kada imamo adresu profilne slike možemo ažurirati dokument koji je prethodno stvoren u bazi podataka za novokreiranog korisnika i u njega dodati URL do profilne slike.

Slika 39: Finish metoda u Register komponenti, 4. dio

```
for (let i = 0; i < this.registration.files.length; i++) {
  setTimeout(async () => {
    let imagesName =
      this.registration.name + "-image-" + Date.now() + ".png";
    const storageRef = ref(
      storage,
      this.registration.name + `(${user.uid})/` + imagesName
    );
    const uploadImges = uploadBytesResumable(
      storageRef,
      this.registration.files[i].img,
      metadata
    );

    uploadImges.on(
      "state_changed",
      (snapshot) => {
        const progress =
          (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
        console.log("Upload is " + progress + "% done");
        switch (snapshot.state) {
          case "paused":
            console.log("Upload is paused");
            break;
          case "running":
            console.log("Upload is running");
            break;
        }
      },
      (error) => {
        switch (error.code) {
          case "storage/unauthorized":
            break;
          case "storage/canceled":
            break;
          case "storage/unknown":
            break;
        }
      },

      async () => {
        const imagesUrl = await getDownloadURL(
          uploadImges.snapshot.ref
        );
        let imagesDesc = this.registration.files[i].desc;
        const image = { url: imagesUrl, desc: imagesDesc };
        await updateDoc(doc(db, "accounts", user.uid), {
          images: arrayUnion(image),
```

Zadnji korak koji je preostao je učitvanje slika koje je korisnik stavio na profil zajedno sa njihovim opisima. Korištenjem for petlje za svaku sliku izvršavamo funkciju učitavanja na isti način kao što je već prikazano za sliku profila. Zbog toga što sada učitavamo više slika odjednom, a učitavanje u bazu je asinkrono stavljen je timeout između svake iteracije for petlje kako bi se sačuvao redoslijed slika u bazi na način na koji ih je korisnik posložio na svom profilu. Nakon što učitavanje pojedine slike završi i dobijemo URL, možemo ažurirati dokument u bazi za novokreiranog korisnika i dodati mu objekt pojedine slike sa njenim URL – om i opisom.

7.3.4. Router

Router datoteka govori Vue – u koje različite rute tj. koje stranice želimo imati u aplikaciji. Svaku instanciranu rutu moramo povezati sa njenom komponentom. U rutama je definiran njihov put i URL. Svaku rutu možemo zaštiti od ne ulogiranih korisnika ako je za nju korisnik potreban.

Slika 40: Router

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../views/Home.vue'
import store from '@/store'

Vue.use(VueRouter)

const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home,
    meta: {
      needsUser: true,
    }
  },
  {
    path: '/FrontPage',
    name: 'FrontPage',
    component: () => import('../views/FrontPage.vue')
  },
  {
    path: '/Register',
    name: 'Register',
    component: () => import('../views/Register.vue')
  },
  {
    path: '/Login',
    name: 'Login',
    component: () => import('../views/Login.vue')
  },
  {
    path: '/MyCollaborators',
    name: 'MyCollaborators',
    component: () => import('../views/MyCollaborators.vue'),
    meta: {
      needsUser: true,
    }
  },
  {
    path: '/Messages',
    name: 'Messages',
    component: () => import('../views/Messages.vue'),
    meta: {
      needsUser: true,
    }
  },
  {
    path: '/Settings',
```

Slika 41: Router 2. dio

```
const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL,
  routes
})

router.beforeEach((to, from, next) => {
  console.log('Old route', from.name, '->', to.name, 'user', store.currentUser);

  const noUser = store.currentUser === null;

  if (noUser && to.meta.needsUser) {
      next('FrontPage');
  } else {


      next();
  }

});

export default router
```

Korištenjem „beforeEach“ funkcije koju možemo izvršavati nad Vue router objektom možemo provjeriti da li je za rutu na koju se korisnik zaputio potrebno biti ulogiran. Ukoliko je to slučaj, korisnik je automatski preusmjeren na naslovnu strancu aplikacije. To je važno kako bi zaštitili rute od toga da im ne ulogirani korisnik pristupi ručno, ukucavanjem njihovog URL – a u browseru.

7.3.5. Login komponenta

Kada korisnik klikne na gumb „Login“ u Login komponenti pokreće se funkcija „login“ koja će ukoliko su svi podaci popunjeni zvati funkciju iz Firebase – a zaduženu za logiranje korisnika koja kao parametre prima uneseni email i lozinku. Ukoliko je korisnik unio krivi mail ili lozinku o tome ga se obavještava shodnom porukom.

Slika 42: Script dio Login komponente

```
<script>
import { auth, signInWithEmailAndPassword } from "@/firebase";

export default {
  name: "Login",
  data() {
    return {
      email: null,
      password: null,
      visiblePassword: "password",
    };
  },

  methods: {
    login() {
      if (this.email == null || this.password == null) {
        alert("Please fill out all fields");
      } else {
        signInWithEmailAndPassword(auth, this.email, this.password)
          .then((userCredential) => {
            const user = userCredential.user;
            console.log(user);
          })
          .catch((error) => {
            if (error.code === "auth/wrong-password") {
              alert("Wrong password");
            } else {
              if (error.code === "auth/user-not-found") {
                alert("User with given email doesn't exist");
              } else {
                alert(error.message);
              }
            }
            console.error(error);
          });
      }
    },
    showPassword() {
      this.visiblePassword = "text";
    },
    hidePassword() {
      this.visiblePassword = "password";
    },
  },
};
</script>
```

7.3.6. Home komponenta

Kada imamo ulogiranog korisnika i obavili smo dohvaćanje svih potrebnih podataka u Home komponenti možemo ispisati sve kartice profila dohvaćenih korisnika. To radimo tako da komponentu profile card korištenjem v-for direktive ponavljamo za svaku karticu odnosno profil iz, u ovom slučaju filtriranih profila, zbog toga što je omogućeno njihovo filtriranje. U profile card komponentu prosljeđujemo tzv. prop koji smo nazvali card koji nam predstavlja jednu karticu profila.

Slika 43: Dio template – a Home komponente

```
<template>
  <div class="home">
    <div class="home__content">
      <h3 class="home__title">
        {{
          store.currentUser.accType === "business"
            ? "Find Promoters"
            : "Find Brands"
        }}
      </h3>
      <profile-card
        v-for="card in filteredProfiles"
        :key="card.id"
        :card="card"
        @open="
          carouselVisible = true;
          visibleImage = 0;
          cardData = card;
        "
      />
      <div class="home__message" v-if="filteredProfiles.length < 1">
        There are no
        {{ store.currentUser.accType === "business" ? "promoters" : "brands" }}
        that meet the filter requirements.
      </div>
      <transition name="carousel">
        <carousel
          @next="next"
          @prev="prev"
          @close="carouselVisible = false"
          v-if="carouselVisible"
        >
          <carousel-slide
            class="carouselImg"
            v-for="(image, index) in cardData.images"
            :key="image.url"
            :index="index"
            :visibleImage="visibleImage"
          >
            <div class="img-slide">
              <div class="img-slide__heading-mobile">
                <div @click="store.profileData = cardData">
                  <router-link
                    :to="{
                      name: 'Profile',
                      params: { profileName: cardData.name },
                    }"
                  >
```

Slika 44: Profile card komponenta

```
<template>
  <div class="card">
    <div class="card__header">
      <router-link
        :to="{ name: 'Profile', params: { profileName: card.name } }"
      >
        <img
          :src="card.profileImg"
          @click="store.profileData = card"
          class="profileImg"
        />
        <span @click="store.profileData = card">{{ card.name }}</span>
      </router-link>
      <more-dropdown :profile="card" />
    </div>
    <div class="card__img">
      <img :src="card.images[0].url" @click="openCarousel()" />
    </div>
    <div class="card__location">
      <span>{{ card.city }}, {{ card.country }}</span>
    </div>
    <div class="card__description">
      {{ card.bio }}
    </div>
  </div>
</template>

<script>
import moreDropdown from "@/components/moreDropdown.vue";
import store from "@/store";

export default {
  name: "profileCard",
  props: ["card"],
  data() {
    return {
      store,
    };
  },
```

Profile card komponenta sadrži kod koji želimo ponavljati za svaki profil. Dakle, želimo da svaki profil ima svoju karticu koja će sadržavati ime profila sa profilnom slikom, glavnu sliku unutar kartice, lokaciju te opis profila. Podatke ispisujemo korištenjem proslijeđenog propa card koji predstavlja jednu karticu profila.

Unutar computed dijela Vue komponente stavljaju se bilo kakva izračunata svojstva koja želimo koristiti, primjerice filtrirani profili. Radimo funkciju koja će nam vratiti profile prema određenim parametrima za filtriranje koje je korisnik unio. Ukoliko korisnik nije izabrao niti jedan parametar, što je početno odnosno zadano stanje, vratit će se svi profili spremljeni u store. Ukoliko su izabrani neki parametri, funkcija vraća profile iz store - a filtrirane prema tim parametrima. Parametri za filtriranje su kategorije u koje spada taj profil i lokacija koja se sastoji od države i grada. Na isti se način unutar App komponente vrši pretraga profila po imenu koje korisnik ukuca u polje za unos u navbaru.

Slika 45: Computed dio Home komponente

```
computed: {
  imagesLength() {
    return this.cardData.images.length;
  },

  filteredProfiles() {
    if (
      this.filterValues.categories.length < 1 &&
      !this.filterValues.country &&
      !this.filterValues.city
    ) {
      return store.cards;
    }

    return store.cards.filter((card) =>
      this.filterValues.categories.length > 0
        ? this.filterValues.categories.some((value) =>
            card.categories.includes(value)
          ) &&
          card.country.includes(this.filterValues.country) &&
          card.city
            .toLowerCase()
            .includes(this.filterValues.city.toLowerCase())
        : card.country.includes(this.filterValues.country) &&
          card.city
            .toLowerCase()
            .includes(this.filterValues.city.toLowerCase())
    );
  },
},
```

7.3.7. Carousel komponenta

Carousel je komponenta koja u sebi sadrži posebnu komponentu carousel slide koja predstavlja jednu sliku. Carousel slide korištenjem v-for direktive ponavljamo za svaku sliku koja se nalazi na profilu te postavljamo tzv. propove index koji će biti jednak indeksu slike unutar arraya i visible image koji će biti jednak visibleImage varijabli. Carousel komponenta prima događaje next koji zove next funkciju koja prikazuje sljedeću sliku, prev događaj koji poziva funkciju koja prikazuje prethodnu sliku i close događaj koji zatvara karusel. Carousel komponenta će biti vidljiva ako je varijabla „carouselVisible“ istinita. Ona postaje istinita kada korisnik klikne na sliku unutar profile card komponente ili na određenu sliku koja se nalazi na profilu (ovisno o tome u kojoj komponenti koristimo carousel komponentu).

Slika 46: Karusel slika na profilu

```
<transition name="carousel">
  <carousel
    @next="next"
    @prev="prev"
    @close="carouselVisible = false"
    v-if="carouselVisible"
  >
    <carousel-slide
      class="carouselImg"
      v-for="(image, index) in cardData.images"
      :key="image.url"
      :index="index"
      :visibleImage="visibleImage"
    >
      <div class="img-slide">
        <div class="img-slide__heading-mobile">
          <div @click="store.profileData = cardData">
            <router-link
              :to="{
                name: 'Profile',
                params: { profileName: cardData.name },
              }"
            >
              <img class="profileImg" :src="cardData.profileImg" />
              <h4>
                {{ cardData.name }}
              </h4>
            </router-link>
          </div>
          <more-dropdown :profile="cardData" />
        </div>
        <img class="image" :src="image.url" />
        <div class="img-slide__content">
          <div class="img-slide__content__heading">
            <div @click="store.profileData = cardData">
              <router-link
                :to="{
                  name: 'Profile',
                  params: { profileName: cardData.name },
                }"
              >
                <img class="profileImg" :src="cardData.profileImg" />
                <h3>
                  {{ cardData.name }}
                </h3>
              </router-link>
            </div>
            <more-dropdown :profile="cardData" />
          </div>
```

Slika 47: Next i prev funkcije

```
next() {
  if (this.visibleImage >= this.imagesLength - 1) {
    this.visibleImage = 0;
  } else {
    this.visibleImage++;
  }
},
prev() {
  if (this.visibleImage <= 0) {
    this.visibleImage = this.imagesLength - 1;
  } else {
    this.visibleImage--;
  }
},
```

Unutar next i prev funkcija osigurano je da slike idu u krug. Ako je korisnik na zadnjoj slici unutar liste a klikne na strelicu za sljedeću sliku prikazuje mu se prva slika u listi. Isto tako ako se korisnik nalazi na prvoj slici unutar liste, a klikne na strelicu za prethodnu sliku prikazuje mu se zadnja slika u listi.

Slika 48: Carousel komponenta

```
<template>
  <transition name="carousel">
    <div class="carousel">
      <button @click="close()" class="close">
        <font-awesome-icon icon="fa-solid fa-xmark" />
      </button>
      <button @click="next()" class="next">
        <font-awesome-icon icon="fa-solid fa-chevron-right" />
      </button>
      <button @click="prev()" class="prev">
        <font-awesome-icon icon="fa-solid fa-chevron-left" />
      </button>

      <slot></slot>
    </div>
  </transition>
</template>

<script>
export default {
  name: "Carousel",
  methods: {
    close() {
      this.$emit("close");
    },
    next() {
      this.$emit("next");
    },
    prev() {
      this.$emit("prev");
    },
  },
};
</script>

<style lang="scss">
</style>
```

Unutar carousel komponente nalaze se gumbovi za mijenjanje slika (strelice) i gumb za zatvaranje karusela (x). U ovisnosti o tome koji je gumb korisnik kliknuo šalje se događaj u komponentu roditelja koji zatim okida određenu funkciju. Korištenjem slot taga komponenta carousel prikazuje ono što je navedeno u komponenti roditelja između carousel tagova.

Slika 49: Carousel Slide komponenta

```
<template>
    <div class="carouselSlide" v-show="visibleImage === index">
      <slot></slot>
    </div>
</template>

<script>
export default {
  name: "CarouselSlide",
  props: ["visibleImage", "index"],
};
</script>

<style>
.carouselSlide {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
}
</style>
```

Kako smo korištenjem v-for direktive stvorili carousel slide komponentu za svaku sliku unutar liste, moramo odrediti za koju ga točno sliku želimo prikazati (kako bi izbjegli da se pojavljuje za svaku sliku). Carousel slide prima propove „visibleImage“ i index“. Korištenjem tih propova možemo odrediti koji će se točno carousel slide prikazati. Bit će prikazan onaj carosel slide gdje su propovi „visibleImage“ i „index“ jednaki. Kada korisnik klikne na karticu profila „visibleImage“ se postavlja na 0 što predstavlja prvu sliku u listi jer indexi počinju od nule. Također kako svaka instanca carousel slide komponente prima svoj index kao prop, možemo usporediti te dvije vrijednosti i prikazati željenu sliku.

7.3.8. More Dropdown komponenta

Kako je prikazano u opisu funkcionalsnoti, kada korisnik želi započeti razgovor s nekim drugim korisnikom, u padajućem izborniku za više mogućnosti mora kliknuti na „Message“. To će pokrenuti funkciju „messageUser“ koja će provjeriti da li je trenutni korisnik već vodio razgovor s tim korisnikom. Ukoliko nije, u Firebase – u kreirat će se novi dokument za taj razgovor sa potrebnim poljima „messages“ koje će za sada biti prazno, „users“ koje će sadržavati ID – e korisnika koji vode razgovor i „lastUpdated“ koje će sadržavati sadašnji timestamp. Također će se ažurirati dokument trenutnog korisnika te dokument korisnika s kojim je razgovor započet tako da će se u polje „chatedWith“ dodati ID od onog drugog korisnika. Također ćemo shodno tome ažurirati podatke trenutnog korisnika u store – u tako da ćemo u polje „chatedWith“ dodati ID korisnika s kojim je započet razgovor. Prije nego korisnika odvedemo na rutu „Messages“ potrebno je odrediti razgovor koji želimo prikazati. U store – u postavljamo varijablu „visibleChat“ koja će biti jednaka ID – u korisnika s kojim je započet razgovor.

Slika 50: Funkcija za kreiranje razgovora s određenim korisnikom

```
async messageUser() {
  if (!store.currentUser.chatedWith.includes(this.profile.id)) {
    await addDoc(collection(db, "chat"), {
      messages: [],
      users: [store.currentUser.uid, this.profile.id],
      lastUpdated: Date.now()
    });
    await updateDoc(doc(db, "accounts", store.currentUser.uid), {
      chatedWith: arrayUnion(this.profile.id),
    });
    await updateDoc(doc(db, "accounts", this.profile.id), {
      chatedWith: arrayUnion(store.currentUser.uid),
    });
    store.currentUser.chatedWith.push(this.profile.id);
  }
  let visibleChat;
  store.chat.forEach((element) => {
    if (element.chatedWith.id === this.profile.id) {
      visibleChat = element.id
    }
  });
  store.visibleChat = visibleChat;
  store.specificChatOpened = true;
  router.push({ name: "Messages" });
},
```

Kada korisnik klikne gumb „Add to collaborators“ pokreće se funkcija „addToMyCollaborators“ koja korištenjem Firebase funkcije „updateDoc“ ažurira račun trenutnog korisnika i dodaje ID tog profila u polje „collaborators“. Također na isti način ažurira i lokalne podatke u store – u. Kada korisnik klikne gumb „Remove from collaborators“ pokreće se funkcija za brisanje tog računa iz korisnikove liste suradnika. Prvo je potrebno stvoriti novu listu suradnika iz koje ćemo izfiltrirati ID profila koji želimo obrisati te ćemo korištenjem „updateDoc“ funkcije iz Firebase – a ažurirati račun trenutnog korisnika i vrijednost polja „collaborators“ postaviti na novokreiranu listu. Također ćemo istu stvar napraviti i sa lokalnim podacima u store – u. U „computed“ dijelu nalazi se funkcija koja vraća da li se trenutni profil nalazi na korisnikovoj listi suradnika (dakle istina ili laž). U ovisnosti o tome da li se taj profil nalazi na korisnikovoj listi suradnika ili ne, prikazat će se potrebni gumb.

Slika 51: Funkcije vezane uz listu suradnika

```
  async addToMyCollaborators() {
    await updateDoc(doc(db, "accounts", store.currentUser.uid), {
      collaborators: arrayUnion(this.profile.id),
    });
    store.currentUser.collaborators.push(this.profile.id);
  },
  async removeFromMyCollaborators() {
    let newCollaborators = store.currentUser.collaborators.filter(
      (id) => id !== this.profile.id
    );
    await updateDoc(doc(db, "accounts", store.currentUser.uid), {
      collaborators: newCollaborators,
    });
    store.currentUser.collaborators = newCollaborators;
  },

},

computed: {
  collabBtn() {
    return store.currentUser.collaborators.includes(this.profile.id);
  },
},
```

7.3.9. Messages komponenta

Kada korisnik napiše poruku i klikne na gumb „send“ pokreće se „sendMessage“ funkcija. Prvo se provjerava da li poruka uopće postoji, tj. da li je korisnik popunio polje za unos poruke prije nego što je kliknuo „send“. Ako korisnik nije napisao poruku, a kliknuo je „send“ funkcija staje sa izvršavanjem. Ako poruka postoji, stvaramo objekt poruke sa njenom vrijednosti (onime što je korisnik napisao), ID – em korisnika i timestampom kad je ona poslana. Sa stvorenim objektom ažuriramo relevantni razgovor u Firebase – u. Na kraju ispražnjavamo polje za unos poruke i scrollamo razgovor na dno kako bi se uvijek vidjela zadnja poruka. Za svaku poruku zatim želimo izračunati kada je ona poslana, jer u bazu spremamo samo timestamp koji moramo preračunati u točno vrijeme. Ne želimo prikazati vrijeme slanja za svaku poruku, već samo onda kada se promijeni datum.

Slika 52: Metode vezane uz slanje poruke

```
scrollChatToBottom() {
  let messageArea = this.$refs.msgBox;
  messageArea.scrollTop = messageArea.scrollHeight;
},

async sendMessage(convo) {
  if (this.message) {
    let message = {
      value: this.message,
      id: store.currentUser.uid,
      time: Date.now(),
    };
    await updateDoc(doc(db, "chat", convo.id), {
      messages: arrayUnion(message),
      lastUpdated: Date.now(),
    });
    this.message = null;
    this.scrollChatToBottom();
  }
},

sentAt(message, previousMessage) {
  if (
    !previousMessage ||
    moment(message.time).format("D MMMM, YYYY") !==
      moment(previousMessage.time).format("D MMMM, YYYY")
  ) {
    return moment(message.time).format("D MMMM, YYYY");
  }
},
```

Slika 53: Brisanje razgovora

```
openDeletePopUp(deleteChatData, deleteChatDataIndex) {
  this.deleteChatData = deleteChatData;
  this.deleteChatDataIndex = deleteChatDataIndex;
  this.deletePopUpVisible = true;
},

async deleteChat() {
  await deleteDoc(doc(db, "chat", this.deleteChatData.id));
  await updateDoc(doc(db, "accounts", store.currentUser.uid), {
    chatedWith: arrayRemove(this.deleteChatData.chatedWith.id),
  });
  await updateDoc(doc(db, "accounts", this.deleteChatData.chatedWith.id), {
    chatedWith: arrayRemove(store.currentUser.uid),
  });
  store.currentUser.chatedWith = store.currentUser.chatedWith.filter(
    (id) => id !== this.deleteChatData.chatedWith.id
  );
  this.deletePopUpVisible = false;
  if (this.deleteChatDataIndex === 0) {
    store.visibleChat = this.searchedChats[this.deleteChatDataIndex].id;
  } else {
    store.visibleChat = this.searchedChats[this.deleteChatDataIndex - 1].id;
  }
},
```

Kada korisnik klikne ikonicu kante za smeća koja se nalazi u gornjem desnom kutu pojedinog razgovora, iz „messageBox“ komponente, komponente koja sadrži pojedini razgovor, šalje se događaj „delete“ u parent komponentu „Messages“ sa podacima potrebnima za brisanje („deleteChat“ funkcija nalazi se u komponenti „Messages“). Događaj „delete“ okida „openDeletePopUp“ funkciju u kojoj prosljeđujemo dobivene podatke u zasebne varijable kako bi ih kasnije mogli koristiti za brisanje te prikazujemo „deletePopUp“ komponentu u kojoj se korisniku postavlja pitanje da li je siguran da želi izbrisati određeni razgovor. Ukoliko klikne „Yes“, razgovor se briše tako što se zove „deleteDoc“ funkcija iz Firebase – a u kojoj brišemo dokument sa prethodno dobivenim ID – em. Također je potrebo ažurirati račune korisnika koji su vodili razgovor te iz polja „chatedWith“ maknuti ID od drugog korisnika. Također je na isti način potrebno ažurirati lokalne podatke u store – u. Kada brisanje završi zatvaramo „deletePopUp“ i postavljamo sljedeći razgovor koji će biti vidljiv.

7.3.10. My Collaborators komponenta

U „computed“ dijelu „MyCollaborators“ komponente vraćaju se prethodno spremljeni profili iz store – a koji se nalaze na korisnikovoj listi suradnika. Korištenjem v-for direktive iz Vue – a, za svaki profil koji se nalazi na korisnikovoj listi suradnika ponavljamo collaborator card komponentu koja će biti zadužena za prikazivanje podataka jednog profila. Ako je lista suradnika prazna prikazat će se poruka „You don't have any collaborators yet!“.

Slika 54: Ispisivanje liste suradnika

```
<template>
  <div class="page">
    <div class="myCollaborators">
      <h3 class="myCollaborators__title">My collaborators</h3>
      <p class="myCollaborators__msg" v-if="collaborators.length < 1">You don't have any collaborators yet!</p>
      <collaborator-card
        v-for="card in collaborators"
        :key="card.id"
        :card="card"
      />
    </div>
  </div>
</template>

<script>
import store from "@/store";
import collaboratorCard from "@/components/collaboratorCard.vue";

export default {
  name: "MyCollaborators",
  data() {
    return {
      store,
    };
  },
  components: {
    collaboratorCard,
  },

  computed: {
    collaborators() {
      return store.cards.filter((card) => store.currentUser.collaborators.includes(card.id));
    }
  }
};
</script>
```

7.3.11. Settings i My Profile komponente

Kada korisnik klikne na gumb „Update“ u „Settings“ komponenti pokreće se funkcija koja prvo provjerava da li su podaci validno uneseni (polja za ime, grad i kategorije ne mogu bit prazna) te ukoliko je to istina ažurira podatke trenutnog korisnika korištenjem „updateDoc“ Firebase funkcije. Također promjene spremamo i u lokalni store.

Slika 55: Funkcija za ažuriranje podataka računa u Settings komponenti

```
methods: {
  async updateAccountData() {
    let valid = true;
    if (!this.updateValues.name) {
      this.msg.name = "This field cannot be empty";
      valid = false;
    } else {
      this.msg.name = null;
    }
    if (!this.updateValues.city) {
      this.msg.city = "This field cannot be empty";
      valid = false;
    } else {
      this.msg.city = null;
    }
    if (this.updateValues.categories.length < 1) {
      this.msg.categories = "You need to select at least one category!";
      valid = false;
    } else {
      this.msg.categories = null;
    }

    if (valid) {
      await updateDoc(doc(db, "accounts", store.currentUser.uid), {
        name: this.updateValues.name,
        city: this.updateValues.city,
        country: this.updateValues.country,
        categories: this.updateValues.categories,
        instagram: this.updateValues.instagram,
        facebook: this.updateValues.facebook,
        tiktok: this.updateValues.tiktok,
        youtube: this.updateValues.youtube,
        linkedIn: this.updateValues.linkedIn,
      });
      store.currentUser.name = this.updateValues.name;
      store.currentUser.city = this.updateValues.city;
      store.currentUser.country = this.updateValues.country;
      store.currentUser.categories = this.updateValues.categories;
      store.currentUser.instagram = this.updateValues.instagram;
      store.currentUser.facebook = this.updateValues.facebook;
      store.currentUser.tiktok = this.updateValues.tiktok;
      store.currentUser.youtube = this.updateValues.youtube;
      store.currentUser.linkedIn = this.updateValues.linkedIn;
      this.successModalVisible = true;
    }
  },
},
```

Slika 56: Neke od funkcija za ažuriranje podataka na profilu u My profile komponenti

```
async updateBio() {
  await updateDoc(doc(db, "accounts", store.currentUser.uid), {
    bio: this.updateData.bio,
  });
  store.currentUser.bio = this.updateData.bio;
  this.bioUpdate = false;
},
openInput() {
  this.$refs.fileInput.click();
},
uploadFile(event) {
  const selectedFiles = event.target.files;
  const file = selectedFiles[0];
  if (!file.type.match("image.*")) {
    return;
  }
  this.updateData.file = { img: file, desc: "" };
  const reader = new FileReader();
  reader.onload = (event) => (this.fileUrl = event.target.result);
  reader.readAsDataURL(file);
},
discardAddImg() {
  this.updateData.file = null;
  this.$refs.fileInput.value = null;
},
async addNewImg(desc) {
  // Create the file metadata
  /** @type {any} */
  const metadata = {
    contentType: "image",
  };
  let imageName = store.currentUser.name + "-image-" + Date.now() + ".png";
  const storageRef = ref(
    storage,
    store.currentUser.name + `(${store.currentUser.uid})/` + imageName
  );
  const uploadImg = uploadBytesResumable(
    storageRef,
    this.updateData.file.img,
    metadata
  );

  this.loading = true;

  // Listen for state changes, errors, and completion of the upload.

  uploadImg.on(
```

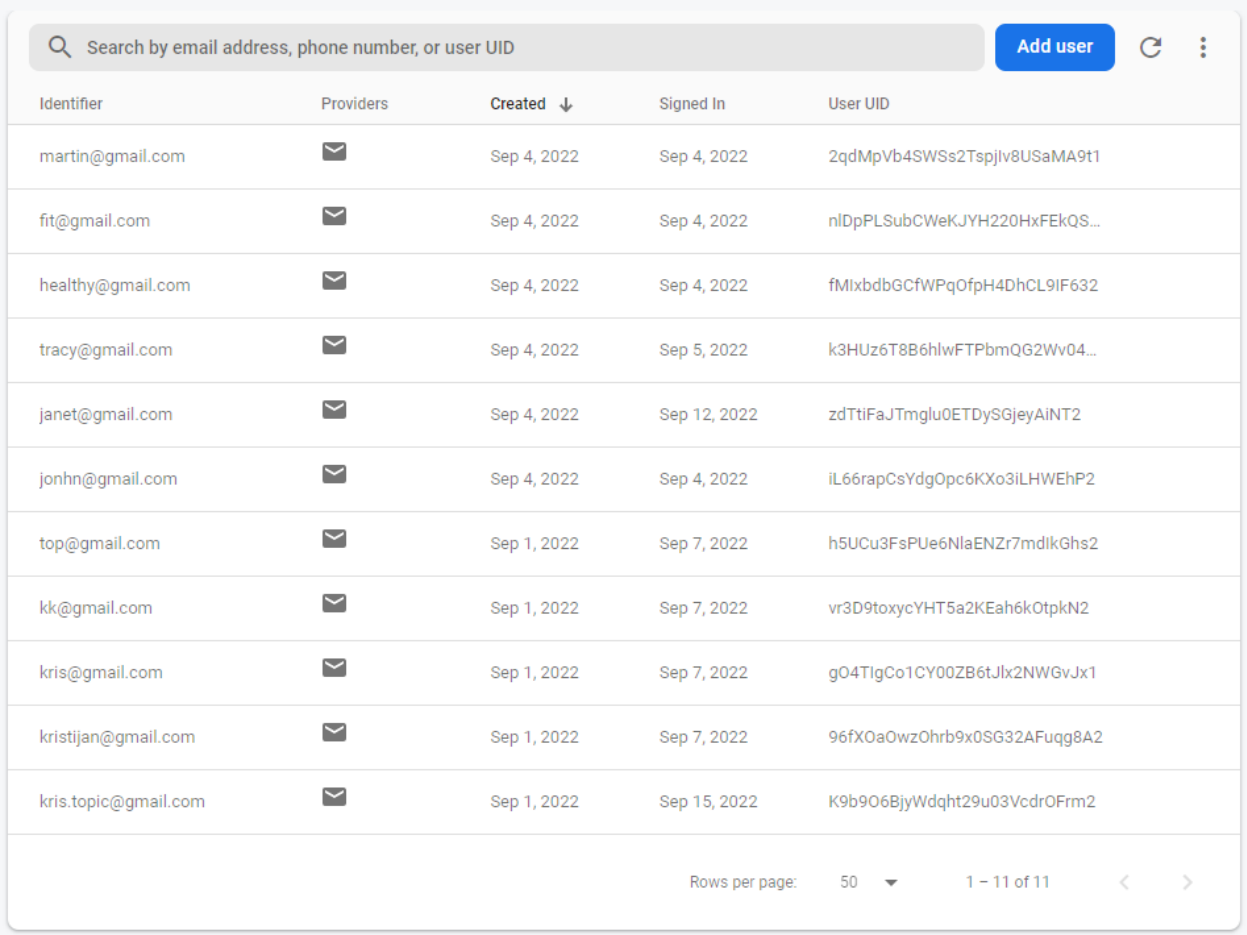
Na sličan način možemo ažurirati i podatke profila u komponenti My profile. Klikom na ikonu olovke otvara se mogućnosti editiranja određenog podatka, te ako korisnik želi spremiti editirano stanje, treba kliknuti na ikonu kvačice što će pokrenuti povezanu funkciju i ažurirati podatke u bazi. Ažuriranje podataka i učitavanje slika u bazu vrši se na sličan način kako je već objašnjeno u prethodnim primjerima.

7.4. KORIŠTENJE FIREBASE - A

7.4.1. Authentication

Firebase sadrži nekoiko paketa funkcija. Prvi korišteni paket u aplikaciji je auth paket koji sadrži funkcionalnosti za autentifikaciju korisnika. Korištenjem tog paketa autentificiramo korisnike korištenjem e-maila i lozinke. E-mail mora bit jedinstven, dakle dva korisnika ne mogu imati isti e-mail.

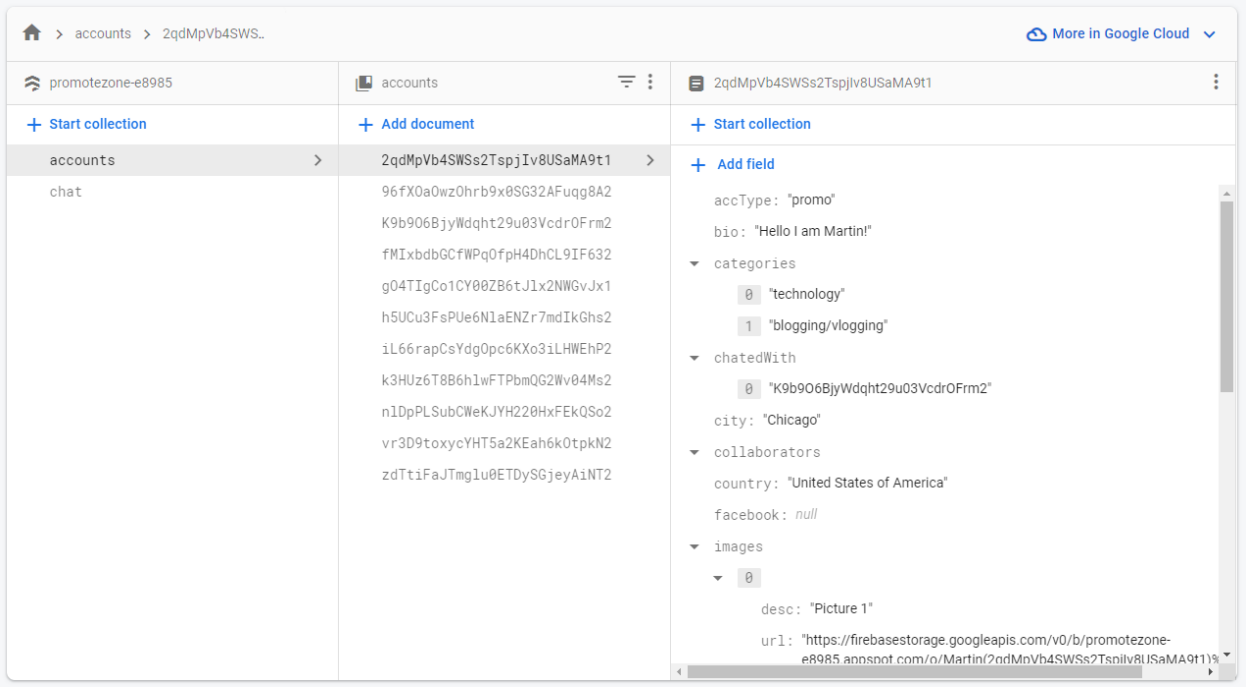
Slika 57: Tablica korisnika u Firebase – u



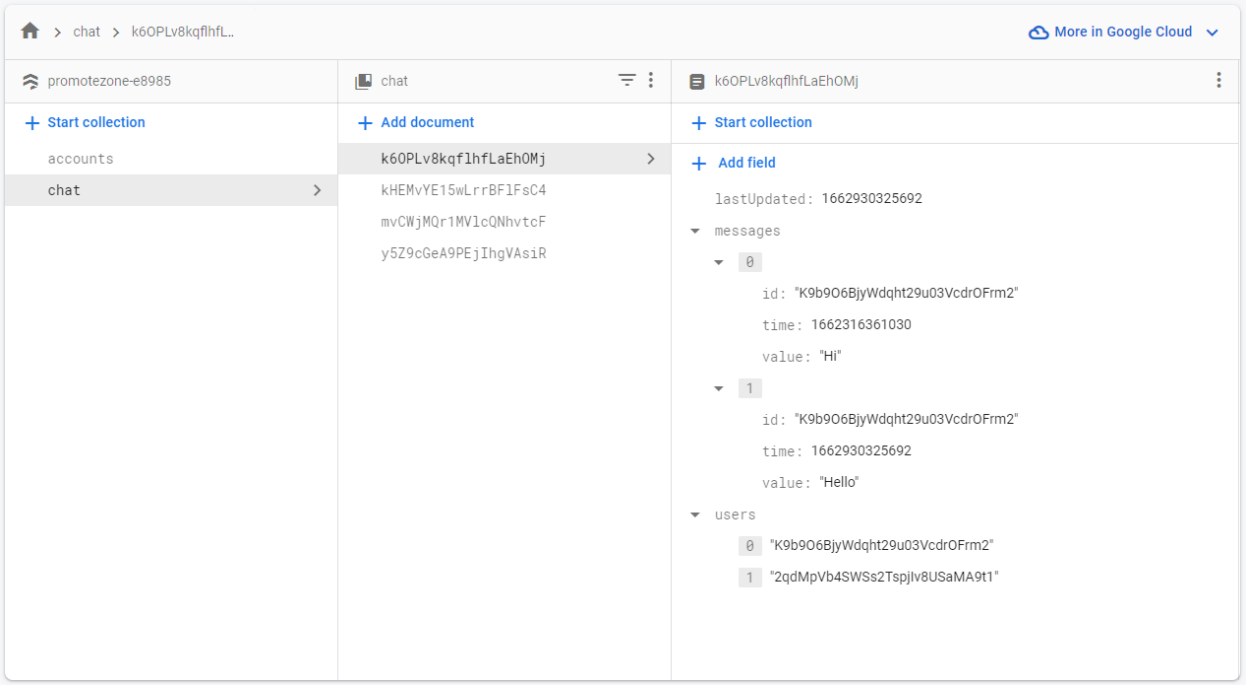
7.4.2. Firestore Database

Također u aplikaciji je korištena Firestore baza podataka koja sadrži dvije kolekcije u koje se spremaju podaci. Kolekcija accounts i kolekcija chat. U kolekciji accounti nalazi se po jedan dokument sa podacima za svakog korisnika, dakle svim onim podacima koje je on popunio u registracijskom procesu i koji se tiču njegovom profila. U kolekciji chat spremaju se svi razgovori. U njoj se nalazi po jedan dokument za svaki razgovor koji sadrži podatke o tom razgovoru.

Slika 58: Kolekcija accounts u Firestore bazi podataka



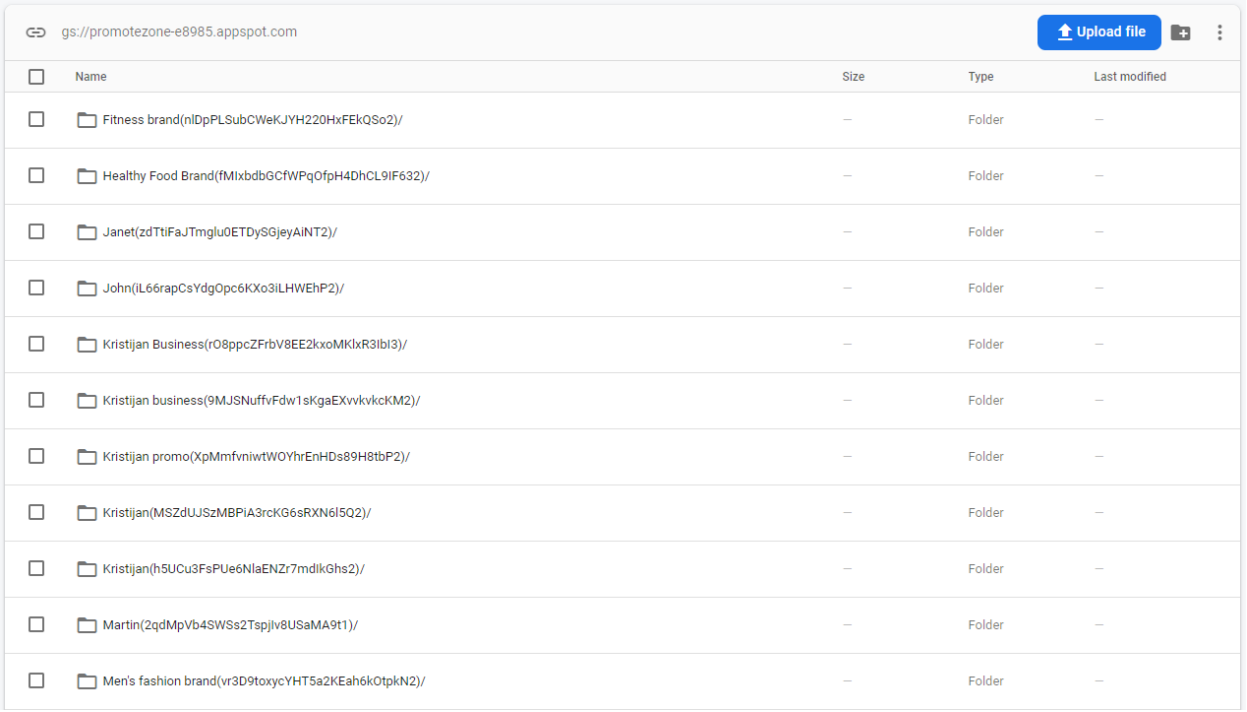
Slika 59: Kolekcija chat u Firestore bazi podataka



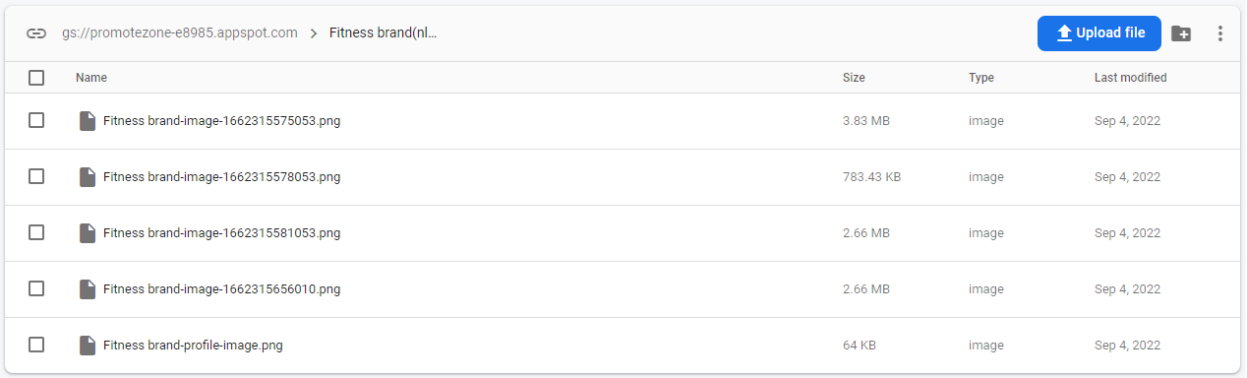
7.4.3. Firebase Storage

Za spremanje datoteka odnosno slika korištena je posebna objektna baza podataka Firebase Storage. U njoj se nalazi folder za svaki kreirani račun u koji se spremaju sve njegove slike.

Slika 60: Folderi u Firebase Storage – u



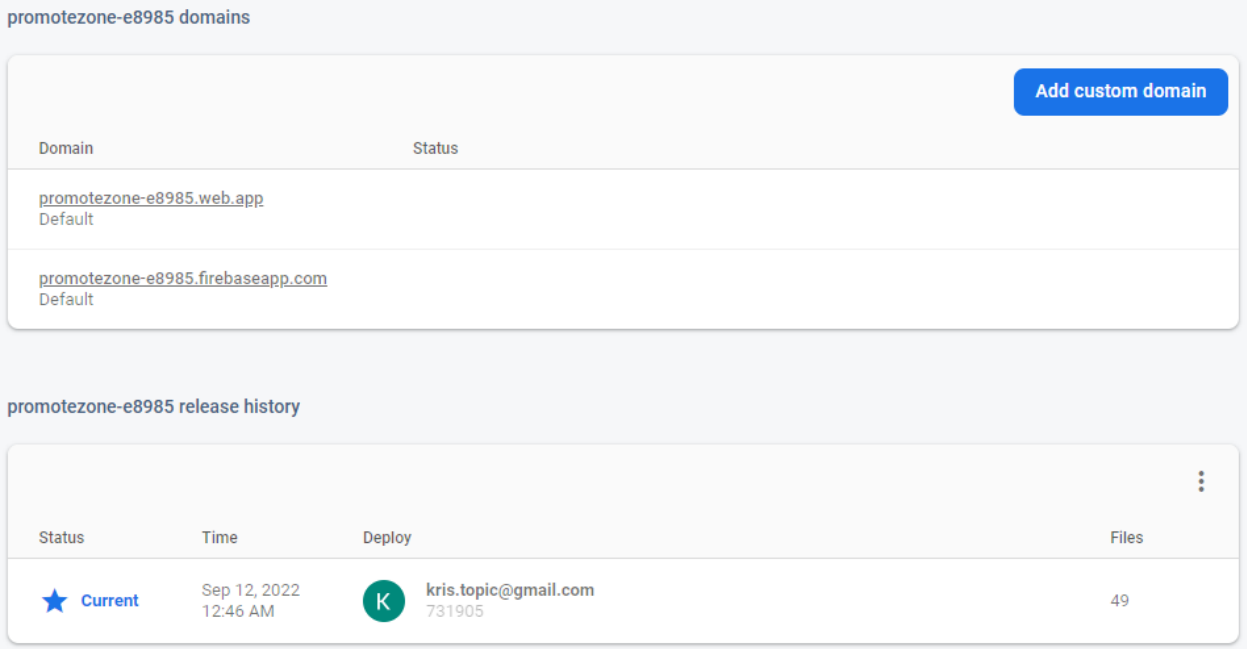
Slika 61: Datoteke u folderu računa



7.4.4. Firebase Hosting

Firebase također nudi i mogućnost hostinga za aplikacije te je i aplikacija Promotezone hostana upravno na Firebase serveru.

Slika 62: Hosting aplikacije Promotezone



8. ZAKLJUČAK

U današnjem vremenu tvrtka, ukoliko želi uspješno poslovati, mora uložiti određenu dozu napora u promociju i marketing kako bi se istaknula od konkurencije. Tvrtke nastoje izgraditi svoje brendove kojima će stvoriti određenu sliku o sebi i svojim proizvodima i uslugama te slati određenu poruku i vrijednosti koje zastupaju. U tim naumima može im pomoći marketing na društvenim mrežama, točnije influencer marketing, koji je danas postao vrlo popularan tip marketinga. Riječ je o marketingu gdje brendovi pronalaze ljude sa određenim utjecajem, obično ljudi koji imaju određen broj pratitelja na društvenim mrežama, koji će promovirati njihove proizvode za određenu naknadu. Takva vrsta marketinga je uspješna zato što ljudi imaju veliku dozu povjerenja u „influencere“ koje prate i veća je vjerojatnost da će proizvod i kupiti.

Aplikacija „Promotezone“ koja je razvijena u ovom diplomskom radu nastoji spojiti brendove sa njihovim promotorima. To čini tako što je omogućena registracija sa dvije vrste računa, jedan za brendove, drugi za promotore. U aplikaciji se oni mogu pronalaziti po određenim kategorijama i lokacijama te međusobno komunicirati i spremati se u svoje liste suradnika. Aplikacija je razvijena korištenjem tehnologija Vue JS, Firebase i SASS.

POPIS LITERATURE

ZNANSTVENI ČLANCI:

1. Gregory T. Gundlach and William L. Wilkie, „The American Marketing Association's New Definition of Marketing: Perspective and Commentary on the 2007 Revision", Journal of Public Policy & Marketing Vol. 28 (2) Fall 2009, 259–264, https://www.unf.edu/~ggundlac/pdfs/pub_07.pdf, (pristupljeno 13. kolovoz 2022.)
2. Sajid SI, „Social Media and Its Role in Marketing", Sajid, Bus Eco J 2016, 7:1, http://repository.embuni.ac.ke/bitstream/handle/123456789/810/social-media-and-its-role-in-marketing.pdf?sequence=1&isAllowed=y, (pristupljeno 28. kolovoz 2022.)

INTERNETSKI IZVORI:

1. American Marketing Association, „What is Marketing? - Definition of Marketing" 2022, https://www.ama.org/the-definition-of-marketing-what-is-marketing/#:~:text=Marketing%20is%20the%20activity%2C%20set,Approved%202017, (pristupljeno 13. kolovoz 2022.)
2. Investopedia, „Marketing in Business: Strategies and Types Explained" 2022, https://www.investopedia.com/terms/m/marketing.asp, (pristupljeno 13. kolovoz 2022.)
3. ISO, „What's in a Brand? Quite a bit, actually" 2020, https://www.iso.org/news/ref2486.html, (pristupljeno 13. kolovoz 2022.)
4. Stack Overflow, „2022 Developer Survey" 2022, https://survey.stackoverflow.co/2022/, (pristupljeno 13. kolovoz 2022.)

5. Investopedia, „Marketing Mix: The 4 Ps of Marketing and How to Use Them“ 2020, https://www.investopedia.com/terms/m/marketing-mix.asp, (pristupljeno 13. kolovoz 2022.)
6. Investopedia, „Trademark“ 2022, https://www.investopedia.com/terms/t/trademark.asp, (pristupljeno 15. kolovoz 2022.)
7. Investopedia, „Brand Awareness“ 2022, https://www.investopedia.com/terms/b/brandawareness.asp, (pristupljeno 18. kolovoz 2022.)
8. Investopedia, „Brand Equity: Definition, Meaning, Effect on Profit Margin, and Examples“ 2021, https://www.investopedia.com/terms/b/brandequity.asp, (pristupljeno 19. kolovoz 2022.)
9. Investopedia, „Social Media Marketing (SMS): What It Is, How It Works, Pros and Cons“ 2022, https://www.investopedia.com/terms/s/social-media-marketing-smm.asp, (pristupljeno 27. kolovoz 2022.)
10. Smart Insights, „Global social media statistics research summary 2022“ 2022, https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/#:~:text=More%20than%20half%20of%20the,social%20media%20is%202h%2029m., (pristupljeno 28. kolovoz 2022.)
11. Datareportal, „Global Social Media Statistics“ 2022, https://datareportal.com/social-media-users, (pristupljeno 28. kolovoz 2022.)
12. Forbes, „What Is Influencer Marketing And How Can Marketers Use It Effectively?“ 2017, https://www.forbes.com/sites/forbescommunicationscouncil/2017/11/14/what-is-influencer-marketing-and-how-can-marketers-use-it-effectively/?sh=75c751ff23d1, (pristupljeno 28. kolovoz 2022.)

JAVNI LINK APLIKACIJE

- https://promotezone-e8985.web.app/FrontPage

JAVNI REPOZITORIJ APLIKACIJE

- https://github.com/KristijanTop/promotezone

POPIS DIJAGRAMA

POPIS SLIKA

SAŽETAK

U ovom radu izložena je funkcionalnost aplikacije „Promotezone“ i njena arhitektura i korištene tehnologije. Kako je danas marketing postao nužan za opstojanje bilo kojeg poslovnog poduhvata, tvrtke su prisiljene tražiti inovativne i kreativne načine za promociju svojih proizvoda kako bi se istaknule od konkurencije i povećale prodaju. Jedan od sve popularnijih načina na koji tvrtke šire svjesnost o svom brendu je promocija preko društvenih mreža korištenjem tzv. influencer marketinga u kojem tvrtke pronalaze utjecajne ljude koji će promovirati njihove proizvode. Aplikacija „Promotezone“ služi za spajanje brendova sa njegovim možebitnim promotorima. Brendovima (tvrtkama) nije uvijek lako naći relevantne promotore u njihovom području i taj problem nastoji riješiti aplikacija „Promotezone“. Također, iz drugog aspekta, može ponuditi nove prilike za promotore. Brendovi i promotori se mogu međusobno tražiti po raznim kategorijama i lokaciji kao i međusobno komunicirati preko live chata te spremati u svoje liste suradnika. Aplikacija je razvijena korištenjem tehnologija Vue JS što je frontend razvojni okvir za JavaScript, Firebase (Google – ova platforma za razvoj aplikacija koja rješava cjelokupni backend dio aplikacije) i SASS (CSS predprocesor) za uređivanje izgleda.

SUMMARY

This paper presents the functionality of the "Promotezone" application, its architecture and the technologies used. As marketing has become necessary for the survival of any business venture today, companies are forced to look for innovative and creative ways to promote their products in order to stand out from the competition and increase sales. One of the increasingly popular ways in which companies spread awareness of their brand is promotion through social media using the so-called influencer marketing in which companies find influential people to promote their products. The "Promotezone" application serves to connect brands with their potential promoters. It is not always easy for brands (companies) to find relevant promoters in their area, and the "Promotezone" application tries to solve this problem. Also, from another aspect, it can offer new opportunities for promoters. Brands and promoters can search for each other by various categories and locations, as well as communicate with each other via live chat and save other users in their lists of collaborators. The application was developed using technologies Vue JS, which is a frontend development framework for JavaScript, Firebase (Google's application development platform that handles the entire backend part of the application) and SASS (CSS preprocessor) for layout editing.