2020

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

ZAKLJUČNA NALOGA

SPLETNI INFORMACIJSKI SISTEM ZA NAKUP IN PRODAJO KOLESARSKE OPREME

MARKO RAZPET

RAZPET

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Spletni informacijski sistem za nakup in prodajo kolesarske opreme

(Web-based information system for the purpose of buying and selling cycling equipment)

Ime in priimek: Marko Razpet
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Iztok Savnik

Koper, oktober 2020

Ključna dokumentacijska informacija

Ime in PRIIMEK: Marko RAZPET

Naslov zaključne naloge: Spletni informacijski sistem za nakup in prodajo kolesarske opreme

Kraj: Koper

Leto: 2020
Število listov: 54 Število slik: 20
Število referenc: 45

Mentor: doc. dr. Iztok Savnik

Ključne besede: informacijski sistem, uporabniški vmesnik, kolesarska oprema, spletna aplikacija, spletne storitve, Angular

Izvleček:
Glavni namen zaključne naloge je razvoje enega izmed prvih spletnih informacijskih sistemov za nakup in prodajo kolesarske opreme. V prven delu zaključne naloge je najprej predstavljen postopek razvoja informacijskega sistema, standardi za oblikovanje uporabniških vmesnikov ter ogrodja za implementacijo spletnih informacijskih sistemov. V drugem delu naloge je opisan postopek načrtovanja in izvedbe spletnega informacijskega sistema za nakup in prodajo kolesarske opreme. Razvoj načrtovanega sistema obsega oblikovanje podatkovne baze MySQL, izdelavo spletne aplikacije z ogrodjem Angular, ter uporabo spletnih storitev za komunikacijo med podatkovno bazo in spletno aplikacijo.

Key document information

Name and SURNAME: Marko RAZPET

Title of the final project paper: Web-based information system for purpose of buying and selling cycling equipment

Place: Koper

Year: 2020

Number of pages: 54 Number of figures: 20

Number of references: 45

Mentor: Assist. Prof. Iztok Savnik, PhD

Key document information: information system, user interface, cycling equipment, web application, web service, Angular

Abstract:

The main goal of the final project is to develop a web-based information system for buying and selling cycling equipment. The first part of the final project paper presents the methodologies for developing web-based information systems, the standards for designing user interfaces, and the modern frameworks for the implementation of web-based information systems. The second part of the paper describes the design and the implementation of the web-based information system for the cycling store. The development of the system includes the design of a MySQL database, creation of a web application with the framework Angular, and the use of web services to implement communication between the database and the created web application.

ZAHVALA

Zahvaljujem se mentorju dr. Iztoku Savniku za svetovanju in pomoč pri izdelavi zaključne naloge. Prav tako se zahvaljujem dragi punci Niki Lap za podporo in pomoč med izdelovanjem. Zahvaljujem se tudi kolegu Anžetu Vedetu za gostovanje podatkovne baze, uporabljene za razvoj.

KAZALO VSEBINE

KAZALO SLIK IN GRAFIKONOV

SEZNAM KRATIC

REST	arhitekturni stil spletnih storitev (angl. Representational State Transfer)
SOAP	arhitekturni stil spletnih storitev (angl. Simple Object Access Protocol)
SQL	strukturirani povpraševalni jezik za delo s podatkovnimi bazami (angl. Structured Query Language)
XML	razširljivi označevalni jezik (angl. Extensible Markup Language)
DOM	objekt za definicijo logičnih struktur dokumentov (angl. Document Object Model)
HTML	označevalni jezik za izdelavo spletnih strani (angl. Hyper Text Markup Language)
CSS	kaskadne stilske podloge (angl. Cascading Style Sheets)
SOA	arhitektura, ki temelji na storitvah (angl. Service-oriented architecture)
URL	enolični krajevnik vira (angl. Uniform Resource Locator)

1 UVOD

Informacijski sistemi v današnjem svetu predstavljajo eno od najpomembnejših orodij modernih organizacij. Uporaba takih sistemov omogoča številne prednosti, kot sta na primer znižanje stroškov in povišanje produktivnosti pri izvajanju procesov [5]. V zadnjih nekaj letih so vedno bolj uporabljani računalniško podprti spletni informacijski sistemi. Imenovana različica sistemov, v primerjavi z ostalimi različicami, ponuja lažji dostop do sistema in omogoča veliko število uporabnikov. Zaradi velikega zanimanja za spletne informacijske sisteme, so se razvile tudi nove tehnologije, ki omogočajo lažjo implementacijo takih sistemov [23]. Ker je rezultat izdelave takih sistemov pogosto spletna stran ali spletna aplikacija, je precej poudarka na uporabniškem vmesniku in na lepotnem izgledu aplikacije. Posledica je lahko premajhen poudarek na preostalih pomembnih komponentah, kot je na primer dobro načrtovana podatkovna baza in varen način komunikacije med spletno aplikacijo in podatkovno bazo [43].

V prvem delu naloge so na splošno predstavljeni informacijski sistemi in razvojni modeli, ki se uporabljajo pri razvoju takih sistemov. Predstavljene so tudi posamezne faze razvojnih modelov ter uporabniški vmesniki in njihova vloga v informacijskih sistemih. Sledenje določenim pravilom v razvojnih modelih je lahko ključnega pomena za razvoj kakovostne rešitve. Prvi del zaključne naloge vsebuje tudi opis standardnih pristopov pri oblikovanju uporabniških vmesnikov, ki so pomembni za doseganje pozitivne uporabniške izkušnje in pripomorejo k zadovoljstvu uporabnikov informacijskega sistema.

Za razvoj spletnih informacijskih sistemov se je v zadnjih nekaj letih pojavilo precej novih tehnologij, ki razvijalcem olajšujejo izdelavo komponent spletnih informacijskih sistemov, zato prvi del zaključne naloge vsebuje tudi nekaj informacij o najbolj priljubljenih tehnologijah za razvijanje spletnih aplikacij, ki predstavljajo uporabniški vmesnik informacijskih sistemov. Opisani so tudi standardi pri implementacijah spletnih servisov za komunikacijo med podatkovno bazo in uporabniškim vmesnikom.

Drugi del diplomske naloge predstavlja razvoj specifičnega spletnega informacijskega sistema za nakup in prodajo kolesarske opreme. Zanimanje za kolesarstvo se v Sloveniji zadnjih nekaj let precej povečuje, vendar na spletu ni nobenega dovolj specifičnega orodja, s katerim bi kolesarji lahko podrobno iskali želeno opremo. Z namenom omogočanja take funkcionalnosti, je bil razvit spletni informacijski sistem, katerega razvoj je predstavljen v tretjem in četrtem poglavju.

V tretjem poglavju je predstavljena analiza že obstoječih podobnih platform, ki omogočajo oddajanje oglasov. Nadaljevanje poglavja opisuje načrtovanje novega spletnega informacijskega sistema ter definicije podatkovnih modelov, načrt podatkovne baze in uporabniškega vmesnika ter izbiro orodij za implementacijo. V četrtem poglavju je predstavljena fizična izdelava sistema, ki vključuje urejanje podatkovne baze MySQL, kreiranje spletne aplikacije z ogrodjem Angular ter implementacijo spletnih storitev tipa REST.

2 RAZVOJ SPLETNIH INFORMACIJSKIH SISTEMOV

2.1 Informacijski sistemi in uporabniški vmesniki

V tem poglavju so na splošno predstavljeni informacijski sistemi, njihove komponente ter prednosti, ki jih ponujajo računalniško podprti informacijski sistemi. V drugem delu poglavja so predstavljeni uporabniški vmesniki in pomen uporabniške izkušnje. Predstavljena so tudi načela načrtovanja uporabniških vmesnikov, ki stremijo k čim boljši uporabniški izkušnji.

2.1.1 Informacijski sistemi

Informacijski sistem (v nadaljevanju IS) je množica med seboj odvisnih komponent, ki zbirajo, procesirajo, hranijo in porazdeljujejo podatke ter s tem nudijo uporabnikom sistema oskrbo z informacijami, pomembnimi za temeljne in odločitvene procese [8].
Uporaba IS omogoča številne prednosti, najpomembnejši sta znižanje stroškov in povišanje produktivnosti pri izvajanju procesov. Podjetja z dobro zasnovanimi IS imajo tako lahko prednost pred konkurenco, saj se poveča kakovost storitev, ponudijo se tudi dodatne možnosti ki prej niso bile omogočene. Z znižanjem stroškov izvajanja procesov, se vložene investicije podjetij hitreje povrnejo, kar omogoča uporabo sredstev za nadaljnji razvoj storitev in večjo konkurenčnost na trgu.
Računalniško podprti IS omogočajo hrambo velikih količin podatkov in lahek dostop do le-teh, do njih je mogoč tudi dostop z oddaljenih virov. Obdelava podatkov, predvsem numerična, se izvaja z visoko hitrostjo, prav tako pa procesiranje podatkov predstavlja zajetno manjši strošek v primerjavi s sistemom, kjer obdelava poteka brez uporabe računalniških sistemov [5].
Na IS se zanašajo poslovna podjetja in druge organizacije, saj jim nudijo možnost izvajanja potrebnih opravil in komunikacijo s strankami in dobavitelji. Število velikih podjetij, ki so zgrajena na temeljih računalniških informacijskih sistemov je vedno večje.
Primeri takih podjetij so na primer Google, eBay in Amazon [45].

V sklopu zaključne naloge je pojem informacijski sistem omejen na računalniško podprte informacijske sisteme.

2.1.1.1 Spletni informacijski sistemi

Spletni IS je vrsta informacijskega sistema, ki z uporabo spletnih tehnologij omogoča uporabnikom izmenjavo podatkov in dostop do storitev. Tak sistem je sestavljen iz kombinacije ene ali več komponent s specifično funkcionalnostjo. Tem komponentam

rečemo tudi spletne aplikacije [23]. Vsi spletni IS spadajo v kategorijo računalniško podprtih IS, saj za dostop do spleta nujno potrebujemo računalniški sistem.

Informacijski sistemi take vrste so sestavljeni iz naslednjih osnovnih komponent:

- *strojna oprema*: delovne postaje, računalniki, strežniki, vhodno-izhodne enote (računalniška miška, tipkovnica, ekran, tiskalnik …);
- *programska oprema*: programi, ki omogočajo nadzor in koordinacijo strojne opreme ter obdelavo podatkov;
- *podatkovna baza*: zbirka tabel ali datotek, ki vsebujejo podatke v sistemu.
- *komunikacijska oprema*: povezovalni sistem, preko katerega si lahko računalniške enote med seboj izmenjujejo podatke;
- *postopki*: dokumentacija sistema, navodila za izvajanje postopkov, da lahko zgoraj našteti sistemi delujejo pravilno;
- *ljudje*: uporabniki strojne in programske opreme [32].

V zadnjih nekaj letih so se spletni IS vedno bolj izboljševali in izpopolnjevali, ter tako pridobili številne prednosti pred ostalimi vrstami informacijskih sistemov.
Ena izmed teh prednosti je možnost dostopa do spletnega informacijskega sistema z različnih platform. Minimalna zahteva za uporabo spletnih aplikacij je zmožnost poganjanja spletnih brskalnikov. V današnjih časih imajo to možnost vsi množično uporabljeni operacijski sistemi, kot so Microsoft Windows, različice Linux operacijskih sistemov in sistemi Mac OS. Naslednja velika prednost je poenostavljeno vzdrževanje in upravljanje sistema, saj se take storitve ne izvajajo v okolju uporabnika, temveč na strežnikih, kjer je spletna aplikacija postavljena. V večini primerov uporabniku ni potrebno storiti ničesar, da lahko uporablja novejšo različico spletne aplikacije, saj se posodobitev aplikacije izvede s strani vzdrževalcev sistema na strežnikih [23].
Velika prednost informacijskih sistemov, ki so dostopni prek spleta, je varnost podatkov. Z dovolj dobro urejenimi varnostnimi sistemi na strežnikih, namenjenih za gostovanje spletnih aplikacij, je lahko zagotovljena odlična varnost podatkov, z zaščito pred vdori v osebne podatke. Ker so vsi podatki zbrani na enem mestu, je zaščita podatkov olajšana, vsa logika za zavarovanje podatkov je uporabljena le na gostujočih strežnikih in zanje skrbijo profesionalni informacijski tehniki, ki so za take storitve usposobljeni. Po vzpostavljeni varnosti na strežnikih, je potrebno le še zagotoviti, da uporabniki lahko do teh podatkov dostopajo le prek varne povezave, potrjene z avtorizacijo [37].

2.1.1.2 Potek razvoja metodologij informacijskih sistemov

V zgodovini so se metodologije za razvoj IS s časom spreminjale. Do leta 1970 so se sistemi razvijali brez formalnih metodologij, saj jih do takrat še ni bilo. Rešitve so nastale

po principu »ad hoc«, torej so reševale zgolj specifične tehnične probleme, ki jih je bilo potrebno rešiti [1]. V tem času so imeli ključno vlogo pri razvoju informacijskega sistema programerji, ki so izvajali rešitve.
V obdobju od leta 1970 do 1980, oziroma v času zgodnjega obdobja metodologij, so se definirali življenjski cikli razvoja informacijskih sistemov, nastale so tehnike podatkovnega in procesnega modeliranja. Bolj poudarjeni so postali zajemi zahtev ter analiza in načrtovanje sistema.
V naslednjih desetih letih je sledilo obdobje metodologij, v času katerega se je uveljavilo strateško načrtovanje, v uporabo pa sta prišli dve vrsti razvoja: iterativni razvoj ter razvoj z uporabo prototipov. Metodologije v tem obdobju so bile potratne, saj poudarek ni bil na načrtovanju sistema, vendar na procesu izdelave in postopnemu prilagajanju glede na nove zahteve, v procesu izdelave pa je bilo potrebno ponovno izvajanje določenih faz.
Po letu 2000 so se pojavile agilne metodologije, ki so omogočile hitro in relativno poceni prilagajanje spremembam zahtev v sistemu [35].

2.1.2 Uporabniški vmesniki in uporabniška izkušnja

Uporabniški vmesnik je okolje, v katerem se dogajajo interakcije med napravo, strojem in človekom, uporabnikom. Cilj vsake take interakcije je nadziranje naprave s strani uporabnika, medtem ko naprava daje povratne informacije, ki pomagajo pri odločitvenem procesu uporabnika [7]. Oblika uporabniškega vmesnika mora biti izdelana tako, da je na izgled privlačna ter naredi uporabniško izkušnjo nezahtevno, efektivno in prijetno, prijazno uporabniku. Tako se poveča možnost, da bo uporabnik zadovoljen z uporabo uporabniškega vmesnika in ga v bodoče ponovno uporabil [21].

2.1.2.1 Uporabniška izkušnja

Po mednarodnem standardu ergonomije interakcij človeka ISO 9241-210, se uporabniška izkušnja nanaša na osebna dojemanja in osebne odzive med uporabo določenega izdelka, sistema ali storitve. Uporabniška izkušnja vključuje vse uporabnikove občutke, mnenja, preference, psihične in fizične odzive, obnašanja in dosežke, ki se zgodijo pred, med in po uporabi. Na uporabniško izkušnjo vplivajo posamezna dejanja med uporabo. Vsak pritisk gumba doprinese k izdelavi mnenja uporabnika o sistemu, ki ga uporablja [15].

2.1.2.2 Pomen uporabniškega vmesnika in uporabniške izkušnje v aplikaciji

V današnjem svetu, kjer se informacije širijo z veliko hitrostjo in se tako okoliščine stalno spreminjajo, obstaja precej različnih aplikacij, ki služijo enakemu namenu, torej so med seboj konkurenčne. V največ primerih je aplikacija edini način interakcije med

uporabnikom in lastniškim podjetjem aplikacije. Poslovne prioritete lastnikov aplikacij so pridobivanje zaslužka, povečanje prepoznavnosti znamke in povečanje odzivnosti uporabnikov, zato mora biti aplikacija boljša od drugih konkurenčnih aplikacij na trgu.
Za izdelavo čim boljše aplikacije je pomembno razumevanje oblikovanja aplikacij. Za uporabnika aplikacije mora biti uporaba efektivna in nezahtevna, saj bo tako dosežena pozitivna uporabniška izkušnja.

2.1.2.3 Načela načrtovanja uporabnosti

Po knjigi Donalda Normana, »The Design of Everyday Things« iz leta 1988, je za oblikovanje ključnih šest principov. To so konsistentnost, vidnost, privoščljivost, povezanost, povratne informacije in omejitve. Našteti atributi se osredotočajo na dejavnike, ki naredijo uporabniško izkušnjo čim bolj olajšano in prijetno. Principi, ki jih je navedel avtor Donald Norman, še vedno vplivajo na sodobno oblikovanje spletnih aplikacij.
*Načelo konsistentnosti***:** Odkrivanje vzorcev je eden izmed najboljših načinov za učenje uporabnika. Če podobna dejanja med uporabo aplikacije uporabnika pripeljejo do podobnih rezultatov, bo uporabnik zaznal vzorec dogajanja aplikacije ter se lažje naučil uporabe. Elementi v dizajnu, ki so si med seboj podobni, naj imajo podobne funkcije. Na primer, uporabnik se nauči, da je v aplikaciji izbočena površina z oznako v resnici gumb, ki se lahko pritisne. Ob naslednjem srečanju z izbočena površino, ki nosi oznako, jo bo uporabnik interpretiral kot gumb, ki se ga lahko pritisne.
Nekonsistentnost pripelje do slabše uporabniške izkušnje, saj stvari ne bodo delovale tako, kakor uporabnik pričakuje da bodo. Siljenje uporabnika k učenju izjem med uporabo, bo pripeljalo do kognitivnega bremena ter bo povzročilo odpor do uporabe. Pri oblikovanju aplikacije je potrebno slediti standardnim načinom uporabe že znanih elementov.
Kot v primeru z gumbom, morajo biti procedure in obnašanje aplikacije konsistentne pri podobnih opravilih in dejanjih.

*Načelo vidnosti***:** Uporabnik lahko s pogledom na uporabniški vmesnik odkrije omogočene funkcije na podlagi vidnih uporabnih elementov. Uporabnost in učljivost aplikacije se izboljšata, če uporabnik zlahka opazi možnosti in nadzorne elemente.
Elementi, ki so namenjeni uporabi, naj bodo jasno vidni in naj bodo na mestih, kjer jih uporabnik pričakuje. Postavitev elementov za upravljanje na nepričakovana mesta ima skoraj podoben učinek, kot če bi bili ti elementi skriti. Funkcijo aplikacije, ki nima vidnega elementa za sprožitev je težko najti za nove uporabnike. Primer je uporaba aplikacij, ki ima implementirane funkcije s podrsavanjem. Za vešče uporabnike je lahko taka funkcija samoumevna, vendar je za nove uporabnike nedosegljiva, če ne poznajo dostopa do nje.

Tak dostop do funkcij ni prepovedan, vendar je pametno obdržati dodatno vidno komponento, ki bo imela dostop do funkcije.
Princip vidnosti ne pomeni, da morajo vse funkcije imeti pripadajoče prožilne elemente na ekranu. V kompleksnejših aplikacijah bi bil posledično ekran, ki ga vidi uporabnik, preveč nasičen z gumbi in podobnimi elementi, zato bi bilo težko najti pravega.
V kompleksnih aplikacijah z velikim številom omogočenih funkcij je zato smiselno združiti več komponent v menije, ki ob interakciji uporabnika prikažejo združene komponente. Še ena rešitev za aplikacije, ki imajo veliko funkcij, je skrivanje upravljalnih elementov, ki jih uporabnik ne bo potreboval v določenih kontekstih.
Načelo vidnosti torej predlaga postavitev, kjer so vse potrebne komponente za nadzor aplikacije na vidnih mestih, kjer jih uporabnik pričakuje.

*Načelo privoščljivosti***:** Oblikovalci uporabniških vmesnikov si lahko privoščijo, da z vizualno prestavitvijo definirajo uporabe in način delovanja določene kontrolne komponente. Standardni primer za razlago načela privoščljivosti je vratna kljuka. Okrogla kljuka uporabniku namiguje, da jo je potrebno zavrteti, če hoče uporabnik doseči, da se vrata odprejo. Ravna ploskev na mestu kljuke uporabniku namiguje da mora na ploskev pritisniti, pritrjen ročaj pa namiguje, da ga je treba potegniti.
V grafičnih uporabniških vmesnikih se princip privoščljivosti uporablja na kontrolnih komponentah. Komponente so vizualno predstavljene na način, ki uporabniku namiguje, ali je komponenta namenjena pritiskanju, podrsavanju ali preklapljanju. Ena izmed najbolj pogostih tehnik je vizualna predstavitev gumbov. Gumbi so največkrat predstavljeni kot izbočena površina, z uporabo senc in osvetlitve površine se doseže tri-dimenzionalni učinek. Težave pri vizualnih predstavitvah se pojavijo pri komponentah s kompleksnimi funkcijami. Primer take komponente je kontekstni menu, ki se pojavi ob dolgem pritisku na napravah z dotikom, ali pa z desnim klikom računalniške miške na namiznih napravah. Vizualna predstavitev take komponente je zahtevna, zato si oblikovalci uporabniških vmesnikov lahko pomagajo z opisi orodja. Opis orodja je največkrat prikazan kot ikona z oznako informacije, na katero lahko uporabnike klikne, da tekst z opisom postane viden.
Oblikovalci uporabniških vmesnikov torej lahko za definicijo funkcionalnosti upravljalnih elementov uporabijo vizualne predstavitve, ki uporabniku namigujejo pravilen način uporabe, pri oblikovanju vmesnikov pa se morajo oblikovalci držati že določenih standardov za vizualni prikaz, s katerimi so uporabniki v večini primerov že seznanjeni.

*Povratne informacije***:** Uporabnik ob pritisku na gumb pričakuje povratno informacijo. Če povratne informacije ni, uporabnik ne more vedeti ali je bil pritisk na gumb v sistemu zaznan. Uporabnik mora zato ugibati, ali mora na gumb pritisniti ponovno, ali pa počakati na odziv sistema. Princip povratnih informacij priporoča implementacijo odziva na interakcije uporabnika z uporabniškim vmesnikom. Uporabnik naj prejme povratne

informacije ne glede na uspešnost uporabljene funkcionalnosti, torej prejeti mora tudi informacijo o neuspešnosti obdelave vnesene zahteve. Obstajata dva tipa povratnih informacij: aktivacijske in vedenjske povratne informacije. V skupino aktivacijskih povratnih informacij spadajo vizualni, čutni in zvočni odzivi na uporabnikove interakcije z uporabniškim vmesnikom. Ko uporabnik pritisne na gumb, je lahko odziv na klik podan z animacijo gumba, ki simulira fizični premik gumba. Na napravah z dotikom se lahko ob pritisku na gumb sproži vibracija, ki uporabnika obvesti o zaznani uporabi gumba. Pritisk na gumb je lahko potrjen tudi z zvočnim efektom. V skupino vedenjskih povratnih informacij spadajo odzivi, ki nakazujejo uspešno ali neuspešno obdelavo vnosa v sistemu. Primer take povratne informacije je potrdilno sporočilo, ki se pojavi ob pošiljanju elektronske pošte. Po pritisku na gumb za pošiljanje, se lahko prikaže pojavno okno s potrdilnim sporočilom. Če je od določene zahteve v sistemu pričakovano, da se izvaja dlje časa, je smiselno prikazati vizualno predstavitev napredka. V primeru da stanje napredka ni znano, je smiselno prikazati »spinner«, ki predstavlja povratno informacijo o obdelavi zahtevane funkcionalnosti. V uporabniškem vmesniku je torej smiselna uporaba povratnih informacij na vsako uporabo funkcionalnosti, saj je tako uporabnik obveščen o uspešnosti vnesene zahteve.

*Omejitve***:** Uporabniški vmesniki morajo biti oblikovani tako, da uporabnik ne more doseči neveljavnega stanja. V neveljavnem stanju uporabniški vmesnik ne deluje več pravilno, to pa slabo vpliva na uporabniško izkušnjo. Dostop do neveljavnega stanja se onemogoči z omejitvami v uporabniškem vmesniku. Do neveljavnega stanja lahko pride z vnosom nepravilnih podatkov, na primer vnos števila ki ga polje v tabeli podatkovne baze ne podpira. Vsi kontrolni elementi uporabniškega vmesnika morajo biti torej implementirani tako, da uporabniku ne omogočajo vnosa nesprejemljivih podatkov. Uporabnik mora s poljubnim zaporedjem ukazov vedno priti v dovoljeno stanje [27].

2.2 Razvoj informacijskih sistemov

V tem poglavju so predstavljene standardne faze v življenjskem ciklu razvoja IS. Za vsako fazo so predstavljene naloge, ki naj bi jih posamezna faza vsebovala. Pri načrtovanju spletnega IS za nakup in prodajo kolesarske opreme je bil uporabljen razvojni model z uporabo prototipa. Tak razvojni model je način razvoja IS, pri katerem je prototip na voljo zgodaj v razvojnem procesu končnega produkta, in služi kot pomoč pri razumevanju sistema. Stranke lahko hitro preverijo, ali so bile njihove zahteve v razvoju uresničene, ter o odkritih napakah ali pomanjkljivostih obvestijo razvijalce [34].

2.2.1 Življenjski cikel razvoja informacijskega sistema

Razvoj IS sledi določenemu življenjskemu ciklu, to je razvojni model, ki določa zaporedje faz razvoja. Tipične faze v razvoju informacijskih sistemov so: začetek razvoja, analiza IS, načrtovanje IS, tehnična izvedba ter vzdrževanje. Delitev na naštete faze razvoja ni striktna. Ljudje, ki razvijajo IS, lahko izberejo drugačen nabor razvojnih faz, ali pa posamezne razvojne faze tudi drugače definirajo. V primeru, da se med določeno fazo v razvoju ugotovi pomanjkljivosti ene izmed predhodnih faz, je dovoljeno vračanje na predhodno fazo v kateri je bila pomanjkljivost ugotovljena. Razvoj IS se lahko deli tudi v več ciklov, kjer vsak razvojni cikel vsebuje vse razvojne faze. Pri izvedbi vsakega cikla se lahko pridobi določeno znanje, ki se uporabi pri izvedbi naslednjega cikla. Z vsako razvojno fazo se pridobijo nove informacije, iz katerih se naredijo sklepi in rezultati izvedbe razvojne faze. Rezultati so lahko pogoj za kvalitetno izvedbo naslednje faze [10].

2.2.2 Začetek razvoja informacijskega sistema

V življenjskem ciklu razvoja IS je na prvem mestu razvojna faza, ki se imenuje začetek razvoja. Cilji v začetku razvoja so definicija problema ter izvedba študije izvedljivosti projekta, ki rešuje problem. Pri definiciji problema izvajalci informacijskega sistema ugotavljajo kaj je razlog za nastanek problema, kje se problem pojavi ter kdo so ljudje, ki ta problem občutijo. Definicija problema lahko razkrije priložnosti, zato izvajalci IS ugotavljajo možnosti reševanja problema, ki jim bodo koristile. Pri izvedbi študije izvedljivosti je potrebno ugotoviti, ali je reševanje zadanega problema mogoče izvesti. Izvajalci IS morajo v sklopu študije definirati tehnično izvedljivost projekta in ugotoviti, ali je z znanimi tehnologijami mogoče priti do rešitve. Vse razpoložljive rešitve niso vedno v skladu z zakoni in kulturnimi omejitvami, zato je potrebno ugotoviti, ali je projekt operativno izvedljiv ter ne krši zakonov in podobnih omejitev. Opredeljena mora biti ekonomska zahtevnost projekta in ali je projekt ekonomsko upravičen. Zahtevni projekti so

lahko za izvajalce veliko finančno breme, zato je treba ugotoviti ali je investicija v projekt smiselna. Izvajalci projekta morajo ugotovljene rezultate v fazi začetka razvoja informacijskega sistema predati svojemu vodstvu. Ko upravljalno vodstvo odobri projekt, se lahko prične naslednja faza v razvojnem ciklu [14].

2.2.3 Sistemska analiza

Druga faza v razvojnem ciklu je sistemska analiza. Namen te faze je, da se izvajalci projekta bolje seznanijo s problemom in potrebami, ki so sprožile proces razvoja določenega IS. Podrobna analiza lahko zahteva tudi natančno preučevanje področja, iz katerega izvira problem. Z natančnim preučevanjem določenega področja izvajalci bolje razumejo kako potekajo procesi v okolju. Z boljšim razumevanjem procesov v okolju, izvajalci opazijo potrebe po spremembah, ki so potrebne za reševanje problemov in izpolnjevanje zahtev. Pomembna je komunikacija med izvajalci IS in strankami, ki po zaključku razvoja dobijo končni rezultat v uporabo. Stranke, oziroma uporabniki izdelanega IS morajo natančno definirati svoje zahteve in pričakovanja, ki naj bi jih končni izdelek vseboval. Vseh zahtev in pričakovanj s strani stranke je lahko tudi preveč za razpoložljiva sredstva in čas, kar lahko pomeni nezmožnost izpolnitve vseh zahtev. V primeru preobsežnih zahtev za implementacijo, je potreben dogovor za definicijo prioritet funkcionalnosti med izvajalci IS in strankami. Rezultat v zaključnih delih sistemske analize pogosto zahteva posodobitev končnih ciljev, ki so bili definirani v fazi začetka razvoja. Po končani analizi se lahko odkrije, da je načrtovan projekt preveč ali pa premalo obsežen, zato lahko izvajalci projekt prilagodijo na sprejemno zahtevnost. Če se projekt spremeni, je potrebna ponovna izvedba študije izvedljivosti iz faze začetka razvoja IS. Na podlagi vseh rezultatov sistemske analize, se izvajalci odločijo, ali se bo projekt razvoja IS nadaljeval, ali pa bo preklican [44].

2.2.3.1 Zbiranje informacij

Prvi del sistemske analize se imenuje razumevanje obstoječega sistema. V tem delu izvajalci IS zbirajo informacije, ki jih pridobijo od ključnih deležnikov, ki lahko vplivajo na sistem, ali pa lahko sistem vpliva na njih. Za zbiranje informacij obstajajo številne klasične in moderne tehnike. V skupino klasičnih tehnik spada proučevanje obstoječe dokumentacije sistema in proučevanje drugih rešitev, ki so že izdelane. Informacije se zajamejo tudi s pomočjo vprašalnikov, individualnih ali skupinskih razgovorov z udeleženci v sistemu. Izmed modernih tehnik zbiranja informacij, se pogosto uporablja skupinsko načrtovanje. To je skupna delavnica, na katero so povabljeni menedžerji in uporabniki sistema, ter projektni tim. Skupinsko načrtovanje omogoča hitro reševanje

konfliktov, zmanjša možnost nastanka napak v kasnejših fazah razvoja in pripomore k hitrejšemu razvoju končnega izdelka [37].

2.2.3.2 Identifikacija izboljšav

Z zbranimi informacijami iz prvega dela sistemske analize, izvajalci lažje razumejo sistem, v katerem se pojavlja problem. V zbranih informacijah se identificirajo pomanjkljivosti, ki jih lahko izrazijo že uporabniki obstoječega sistema, ali pa jih izvajalci z razumevanjem sistema odkrijejo sami. Na podlagi identificiranih pomanjkljivosti, se raziščejo možnosti za izboljšave [37].

2.2.3.3 Izdelava koncepta novega sistema

Ob zaključku sistemske analize se sestavi pisno poročilo, ki vključuje koncept novega sistema. Predlog novega sistema predstavlja izboljšave in rešitve za definirane pomanjkljivosti starega sistema. Ideja novega sistema se predstavi strankam, ki lahko med in po predstavitvi komunicirajo in delijo povratne informacije z mnenji o predlaganih rešitvah [44].

2.2.4 Načrtovanje

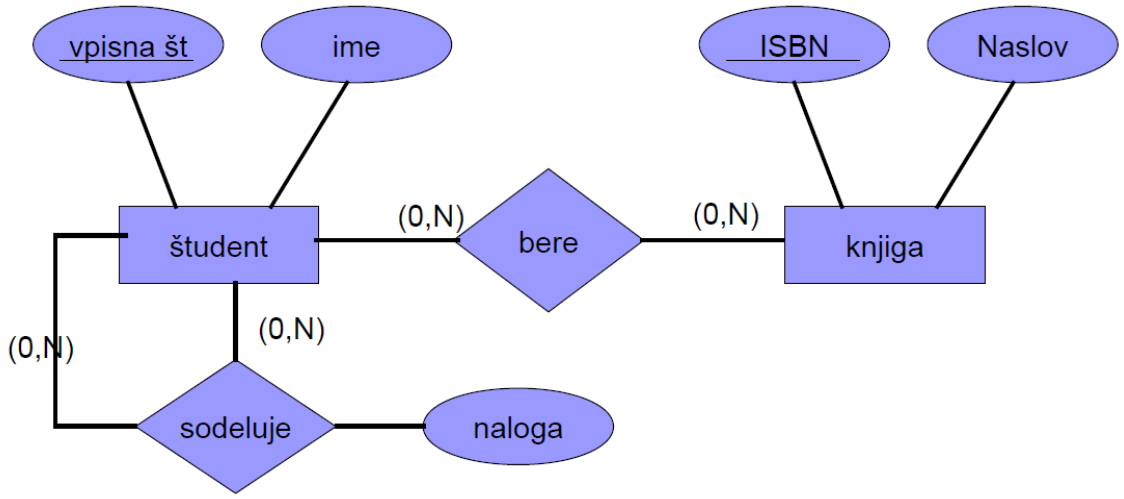
Načrtovanje informacijskega sistema je ključni del razvojnega cikla. Dobro načrtovanje je v večini primerov izdelave IS pogoj za dobro kvaliteto končnega izdelka.

2.2.4.1 Načrtovanje arhitekture IS

Arhitektura informacijskega sistema definira tehnologije, ki se uporabljajo v kasnejših delih razvojnega cikla. Za definicijo tehnologij je potrebno najprej izdelati podatkovne modele, ki se največkrat predstavljajo z uporabo entitentno-relacijskega diagrama. Potrebno je tudi izdelati načrte podatkovnega toka v sistemu z diagrami toka podatkov in definirati procese v informacijskemu sistemu z diagrami stanj. Procesi in tok podatkov v bodočem IS se definirajo na podlagi informacij, ki so bile pridobljene s sistemsko analizo [44].

Entitetno-relacijski diagram: ER diagram je vrsta strukturnega diagrama, ki se izdela za pomoč pri načrtovanju podatkovne baze. Uporablja se za opis podatkovne strukture v IS in odvisnosti med podatki. Diagram je sestavljen iz entitet, entitetnih atributov in povezav med entitetami. Entiteta je opredeljiv predmet v sistemu, ki je lahko oseba, vloga, objekt,

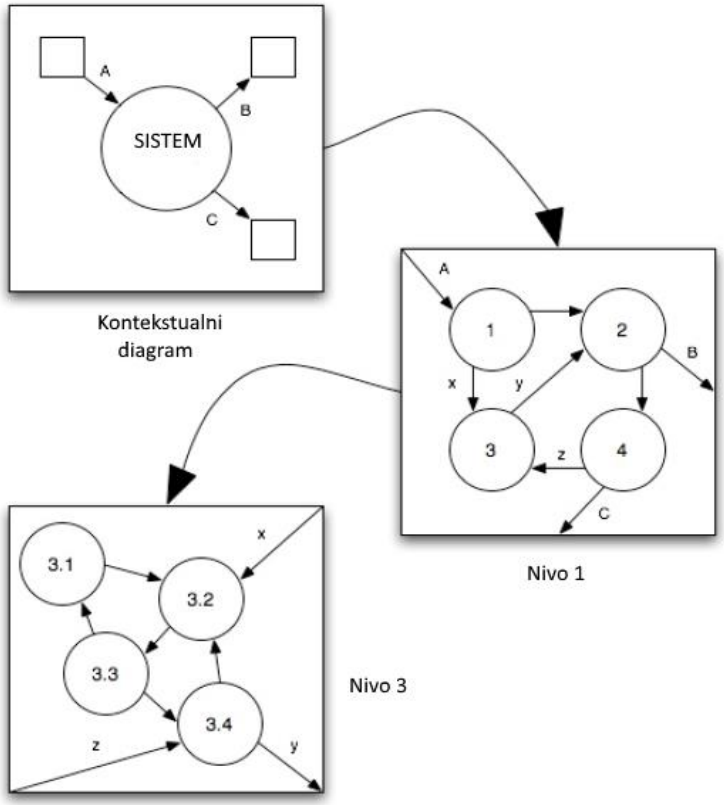
koncept ali dogodek, ki se pojavlja kot dejavnik v IS. Vsaka entiteta ima atribute, ki so lahko različnih podatkovnih tipov, in opisujejo lastnosti entitete. Definiran mora biti tudi primarni ključ entitete, ki kasneje unikatno opredeljuje zapis podatka v podatkovni bazi. Če se entiteta nanaša na drugo entiteto v diagramu, se za identificiranje povezave med njima uporablja tuji ključ, ki pa ni nujno unikaten. Zapisi podatkov iste entitete si delijo enake vrednosti tujega ključa, če se nanašajo na isti zapis podatka druge entitete.
Povezave v ER diagramu pomenijo odvisnost med posameznimi entitetami. V primeru diagrama, predstavljenega na sliki 1, je definirana povezava med študentom in knjigo. Ta povezava pomeni, da študent lahko bere knjigo in da je knjiga lahko brana s strani študenta. Opredeljene so tudi številnosti povezave, primer (0, N) na levi strani povezave pomeni, da študent lahko bere od 0 do poljubnega števila N knjig. Številnost (0, N) na desni strani povezave pa pomeni, da posamezno knjigo lahko bere od 0 do poljubnega števila N študentov. V ER diagramu so entitete običajno predstavljene s pravokotnimi oblikami in napisom imena entitete, kar je vidno na sliki 1. Atributi entitete so lahko napisani v pravokotniku, ki vsebuje ime entitete, ali pa so posamezno predstavljeni v elipsah, z imeni atributov. Atribut, ki je primarni ključ entitete, mora biti posebno označen, v primeru na sliki 1, je ime primarnega ključa podčrtano. Povezave med identitetami so lahko definirane z romboidno obliko in imenom povezave [25].



Slika 1: Primer ER diagrama. (vir: [33])

Diagram toka podatkov (DFD): DFD opisuje tokove podatkov od uporabnikov in zunanjih objektov v sistem in obratno, ter tokove podatkov znotraj sistema. Diagram opisuje tudi, kako se podatki znotraj sistema obdelujejo in hranijo. Za modeliranje diagrama je nujno poznavanje procesov in funkcionalnosti znotraj sistema. DFD je sestavljen iz več diagramov, ki opisujejo sistemske procese na različnih nivojih podrobnosti. Diagram na najvišjem nivoju se imenuje kontekstualni diagram – nivo 0, kjer je sistem predstavljen kot ena sama komponenta in umeščen v okolje zunanjih dejavnikov, ki vnašajo in prejemajo

podatke IS. Z vsakim nižjim nivojem diagrama je sistem opisan bolj podrobno. V diagramu nivoja 1 je predstavljena dekompozicija sistema na osnovne sistemske procese, diagram nivoja 2 pa predstavlja dekompozicijo posameznega sistemskega procesa. Končni diagram ima nivo N, kjer so dekomponirani vsi procesi v sistemu in diagrama ni mogoče več razčleniti na več delov. Pomembno je, da so v vsakem diagramu vsebovani vsi podatki višjega nivoja, hkrati pa se ne smejo dodajati novi podatki.
Primer diagrama podatkovnega toka je opisan na sliki 2, kjer del slike z oznako »kontekstualni diagram« predstavlja diagram nivoja 0 oziroma kontekstualni diagram. Del slike z oznako »Nivo 1« predstavlja diagram nivoja 1, kjer so procesi označeni s številkami, tok podatkov pa je označen s puščicami, ki nosijo črkovne oznake. »Nivo 3« na sliki označuje diagram nivoja 3, kjer je natančneje predstavljen proces 3 iz diagrama nivoja 1 [31].



Slika 2: Nivoji diagrama podatkovnega toka. (vir: [31])

2.2.4.2 Načrtovanje podatkovne baze

Naloga načrtovanja podatkovne baze (v nadaljevanju PB) je pogosto zahtevna naloga, saj ne gre zgolj za preprosto razporeditev zapisov podatkov. Lahko se zgodi, da v prihodnosti dostop do PB potrebuje več različnih programov in aplikacij z različnimi storitvami in funkcionalnostmi, ki niso vnaprej predvidene. Načrtovalec PB mora posledično paziti, da oblika baze dovoljuje nadaljnje prilagoditve in nadgradnje [2].

Predstavitev načrtovanja PB je v zaključni nalogi omejena na *relacijske podatkovne baze*
Proces načrtovanja PB vključuje več korakov:
Analiza zahtev: V kolikor razvojni cikel ni vseboval sistemske analize, ali pa so informacije iz faze sistemske analize nepopolne, je potreben ponovni zajem informacij. Za načrtovanje PB je ključnega pomena, da načrtovalec pozna naravna razmerja med podatki, ter kakšne so funkcionalne zahteve, ki so odvisne od procesov, ki se vršijo v sistemu. Izbere se programska platforma za izdelavo in upravljanje PB. Rezultat analize zahtev sta specifikacija zahtev, ki jim mora PB ustrezati, ter podatkovni slovar, ki se nanaša na opis podatkov v informacijskem sistemu in opisuje vse predvidene entitetne tipe. V splošnem je podatkovni slovar že definiran z ER diagramom, izdelanim v predhodnem delu načrtovanja informacijskega sistema [42].
Konceptualno načrtovanje podatkovne baze: V tem koraku načrtovanja PB, se klasificirajo entitete in atributi, identificirajo se generalizacijske hierarhije in definirajo razmerja med entitetami. Pri načrtovanju PB se lahko pojavi problem pri odločitvi, ali je določen podatkovni element entiteta, atribut ali razmerje. Klasifikacija entitet in atributov narekuje, naj entitete vsebujejo opisne podatke in naj so večvrednostni atributi definirani kot entitete. Atributi, ki najbolj direktno opisujejo določeno entiteto, naj bodo nanjo pripeti. Identifikacija generalizacijske hierarhije pomeni, da se v sistemu pojavlja več različnih entitet, ki imajo skupne generične lastnosti. V takih primerih je smiselno ustvariti splošno entiteto z generičnimi lastnostmi, kateri se ostale entitete podredijo. Podrejene entitete tako vsebujejo le specifične lastnosti, saj so generične že vsebovane v splošni entiteti. Definicija razmerij med entitetami pomeni identificiranje, ali je določeno razmerje rekurzivno, binarno ali ternarno. Smiselno je tudi identificirati redundantna, odvečna razmerja. Redundantna razmerja pogosto predstavljajo redundanco pri prevajanju konceptualnih shem v relacijske sheme, kar v splošnem pomeni dodatno delo pri normalizaciji relacijskih tabel [38].

Integracija shem in pogledov: V primerih, kjer je podatkovna baza velika in je pričakovano veliko uporabnikov in različnih aplikacij, ki bodo do PB dostopale, je smiselna integracija shem, s čimer omejimo možnost funkcionalnosti določenim uporabnikom, kjer je to potrebno [38].

Transformacija konceptualne sheme v SQL tabele: V tem koraku se entitete, definirane v konceptualnem načrtovanju PB, prevedejo v SQL tabele. Pridobljena SQL tabela ima isto informacijsko vsebino kot originalna entiteta, iz katere je bila razvita. Entitete, ki so odvisne od drugih entitet, se pretvorijo v SQL tabele z dodanimi tujimi ključi entitet od katerih so odvisne. Naredijo se tudi tabele, izpeljane iz binarnih razmerij tipa N-N, razmerij, ki so rekurzivna in tipa N-N, ter razmerij, ki so ternarna ali višjega reda. Tako narejene tabele vsebujejo tuje ključe vseh entitet v razmerju [41].

Normalizacija SQL tabel: Podatkovne baze, ki jih definira ena sama velika tabela, lahko povzročijo veliko redundantnih zapisov podatkov in daljše čase v izvajanju poizvedb v tabeli. Poveča se tudi zahtevnost popravljanja in brisanja zapisov podatkov. Naštetim težavam se lahko načrtovalci PB izognejo z normalizacijo SQL tabel. Obstajajo tri splošne oblike normalizacije, to so prva, druga in tretja normalna oblika (v nadaljevanju NO). Oblike se stopnjujejo ena za drugo, kar pomeni, da mora PB zadostovati kriterijem prve NO, da se lahko normalizira na drugo NO. Podobno velja tudi za prehod iz druge na tretjo obliko. Z normalizacijo do tretje normalne oblike, se načrtovalci izognejo anomalijam v zapisih podatkov, obenem pa preprečijo podvajanje in se zaščitijo pred neželeno izgubo podatkov [39].

Fizično načrtovanje: V tem koraku se načrtuje postavljanje indeksov na tabelah v PB. Namen indeksov je optimizacija časa poizvedb v tabelah PB. Načrtovalci postavijo indekse na najbolj primerne stolpce v tabelah SQL. Primerni stolpci so tisti, na katere se bodo poizvedbe sklicevale v največ primerih. Načrtovalci se morajo tudi odločiti, ali je za določeno tabelo smiselno uporabiti indeks. V nekaterih primerih je bolj smiselno opustiti uporabo indeksa, saj bi tako po nepotrebnem koristili prostor v PB. Odločitev, na katerem stolpcu naj se postavi indeks, zahteva vnaprejšnjo poznavanje poizvedb, ki se bodo izvajale v PB [40].

2.2.4.3 Načrtovanje uporabniškega vmesnika

Uporabniški vmesnik (v nadaljevanju UV) v informacijskem sistemu predstavlja vlogo modula, ki omogoča izmenjavo podatkov med podatkovno bazo in uporabniki. Oblikovalci UV se osredotočajo na potrebe uporabnika, ki mora imeti možnost uporabe vseh funkcionalnosti, ki so bile definirane v konceptu novega IS. Na podlagi definiranih funkcionalnosti, se oblikovalci odločijo za primerne elemente, ki jih lahko uporabniki koristijo za izvajanje želenih opravil. Kot je definirano že v poglavju 2.1.2, morajo biti oblikovalci pozorni na postavitve elementov, s katerimi upravljajo uporabniki. Pomembno je tudi, da se oblikovalci držijo standardnih elementov, ki so jih uporabniki že naučeni. Primer takih elementov so gumbi, tekstovna polja, spustni seznami in tako dalje [22].

2.2.4.4 Načrtovanje komunikacije med podatkovno bazo in uporabniškim vmesnikom

V tem koraku se načrtuje povezava med že načrtovano podatkovno bazo in uporabniškim vmesnikom. Izvajalci se odločajo, s katerimi tehnologijami se bo izvajal prenos podatkov med PB in UV. Pri izbiri morajo upoštevati tudi varnost podatkov. Odločiti se morajo za tehnologije, ki podpirajo način varnega dostopa do podatkov, v primeru da je bila varnost podatkov zahteva v konceptu novega sistema.

2.2.5 Izvedba

V fazi izvedbe se izvršijo načrti, narejeni v predhodni fazi razvojnega cikla.

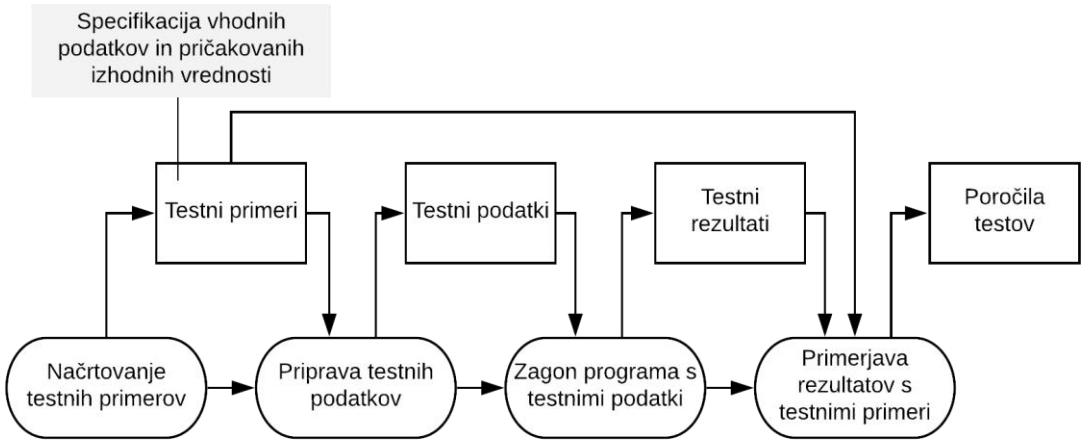
2.2.5.1 Gradnja sistema

Posamezniki v projektnem timu izvajalcev pričnejo z implementacijo načrtovanih delov IS. V kolikor za izvedbo posameznih delov IS tehnologije še niso bile izbrane, izvajalci tehnologije izberejo ob pričetku izvedbe. Komponente sistema lahko razvijajo sami, ali pa se, kjer bi bil razvoj od čistega začetka nesmiseln, odločijo za nakup posameznih komponent. Sistemski administratorji poskrbijo za pravilno delovanje strežnikov, kjer se bo nahajal glavni del, vključno s podatkovno bazo. Administratorji PB načrtovano strukturo implementirajo s pomočjo sistemov za uporabo podatkovnih baz. Razvijalci se običajno razdelijo na dve skupini. Prva skupina se ukvarja z implementacijo »front end« izdelave IS, kar pomeni da izdelujejo uporabniške vmesnike, ki bodo vidni uporabnikom. Druga skupina razvijalcev izdeluje »back end« sistema, ki poskrbi za pravilno komunikacijo med podatkovno bazo in uporabniškim vmesnikom, ki ga izdela prva skupina razvijalcev. Po končani izdelavi komponent informacijskega sistema, se izvrši namestitev komponent v okolje. Smiselno je, da se komponente najprej namestijo v testno okolje, ki je po sistemskih lastnostih identično okolju, namenjenemu za končno uporabo IS.

2.2.5.2 Testiranje

Namen testiranja IS je iskanje napak, preden se končni izdelek preda končnim uporabnikom. Po standardu IEEE 829, se za testiranje ustvari 8 različnih dokumentov, ki pokrivajo vse faze testiranja. *Prvi dokument* opisuje načrtovanje testiranja, ki okvirno določa potek testiranja, kdo bo testiral, ter kako in kdaj bo testiranje potekalo. V *drugem dokumentu* se opisane generalne določitve testov in pričakovani rezultati. *Tretji dokument* predstavlja specifikacije testa, v katerem so natančno določeni testni primeri in pričakovani rezultati. Preizkuševalci, ki so po navadi del projektnega tima, izvedejo serijo testnih primerov. Najprej oblikujejo testne primere za posamezne komponente IS. Testni primeri vsebujejo testne vhodne podatke in pričakovane rezultate, ki ustrezajo testnim vhodnim podatkom. S *četrtim dokumentom* se določijo praktični postopki izvedbe testiranja primerov, ki so specifično opisani v tretjem dokumentu. *Peti dokument* je poročilo predaje v testiranje, v katerem je navedena vsebina, predana v postopek testiranja. Po predaji se testni vhodni podatki vstavijo v posamezne komponente IS. Dejanska izvedba testiranja je zabeležena v *šestem dokumentu*. V ta dokument se zapiše dnevnik s seznamom vseh izvedenih testov z detajli. Incidenti, ki se zgodijo med testom, se zapišejo v poročilo

oziroma *sedmi dokument* s seznami dogodkov, ki jih je potrebno dodatno raziskati. Rezultate, ki jih vrne IS, preizkuševalci primerjajo s pričakovanimi rezultati. Primerjava rezultatov se zapiše v poročilo testiranja, ki predstavlja *osmi dokument*.
Primer testiranja posamezne komponente IS je grafično prikazan v sliki 3. Vsi testni primeri morajo biti povezani za zahtevami, saj so najhujše napake tiste, ki povzročijo nepravilno delovanje zahtevanih funkcionalnosti IS. Preizkuševalci poizkušajo porušiti pravilno delovanje sistema s testi, za katere obstaja verjetnost, da razkrijejo še ne odkrito napako. Testiranje je najbolj učinkovito, ko ga opravljajo neodvisni preizkuševalci. Razvijalec, ki je naredil posamezno komponento, najverjetneje sistema ne bo testiral na destruktiven način, zato ni najbolj primeren za izvajanje testov. Izčrpno testiranje ni mogoče, saj je število možnih poti izvajanja programa že pri manjših IS lahko prevelika, da bi lahko preizkusili vse možnosti izvajanja [29].



Slika 3: Potek testiranja komponente informacijskega sistema. (vir: [29])

2.2.5.3 Migracija

Po uspešno zaključenem testiranju, se komponente IS migrirajo v okolje, ki je namenjeno za končno uporabo IS. Če se izvajalci in stranke dogovorijo drugače, se lahko za končno okolje uporabi tudi testno okolje. Tedaj migracija komponent ni več potrebna. V kolikor je končni IS izdelan kot nadgradnja starejše verzije IS, je potrebna migracija vseh podatkov in datotek, prisotnih v starem sistemu [10].

2.2.5.4 Dokumentiranje

Izdelava dokumentacija je pomembna za razvijalce in uporabnike IS, zato obstajata dve verziji dokumentacije. Prva je *sistemska dokumentacija,* ki pomaga razvijalcem pri

razumevanju delovanja informacijskega sistema. Pomen sistemske dokumentacije se izkaže pri vzdrževanju sistema in pri odkritju napak s strani uporabnikov po končni predaji. Tedaj je sistemska dokumentacija ključnega pomena za hitro reševanje napak, saj razvijalci hitreje najdejo dele sistema, pri katerih se pojavljajo napake. Praviloma se izdelava dokumentacije izvede med gradnjo sistema, pri izdelavi pa izdelujejo vsi razvijalci IS. Izdelava med gradnjo sistema je najbolj smiselna, saj razvijalci takrat o delovanju sistema vedo največ. Druga verzija dokumentacije je *uporabniška dokumentacija*, ki je namenjena končnim uporabnikom IS. Uporabniška dokumentacija vsebuje napotke za pravilno uporabo IS iz uporabniškega vidika, kar lahko pomaga uporabnikom k lažji uporabi sistema. Izdelava uporabniške dokumentacije je lahko izvedena med gradnjo sistema in ni nujno, da jo izdelajo razvijalci. Izdelavo lahko izvedejo tudi drugi člani projektnega tima, ki poznajo zahteve in procese v IS [28].

2.2.6 Vzdrževanje

V fazi vzdrževanja se izvajajo popravila v IS in vse druge aktivnosti, ki ohranjajo končni produkt v stanju obratovanja. Vzdrževanje IS pomeni spreminjanje programskega produkta po izdaji z namenom odpravljanja napak, izboljšanja zmogljivosti sistema, prilagajanja novim zahtevam ali prilagajanja produkta v primerih sprememb okolja. Običajno je cena vzdrževanja precej višja od cene razvoja, saj faza vzdrževanja praviloma traja precej dlje kot faza razvoja [30].

2.3 Tehnični razvoj spletnih informacijskih sistemov

Pri izvajanju spletnega informativnega sistema v sklopu zaključne naloge je bila uporabljena programska oprema, izbrana izmed možnosti, predstavljenih v tem poglavju. Za pridobitev podatkov v spletni aplikaciji so bili uporabljeni spletni servisi, zato je v tem poglavju omenjena tudi uporaba arhitekture, ki temelji na storitvah.

Za izvedbo spletnega informacijskega sistema je smiselno izbrati programsko opremo, ki bo najbolj ustrezna za naravo dela, ki ga narekuje načrt IS. Izvajalci se pri izbiri programske opreme odločajo med odprtokodno in plačljivo programsko opremo. Delo z odprtokodno programsko opremo pomeni, da med uporabo določenega programa s pomanjkljivo dokumentacijo, razvijalec nima dostopa do tehnične podpore avtorjev programske opreme. Plačljiva programska oprema je pogosto uporabniku bolj prijazna od odprtokodne, največkrat iz vidika uporabniških vmesnikov, kar lahko pohitri proces razvoja [17].

2.3.1 Programska oprema za razvijanje podatkovnih baz

Pri razvoju podatkovne baze je izbor programske opreme odvisen predvsem od tipa podatkovne baze, ki je cilj te naloge. Podatkovna baza je lahko tipa SQL ali pa NoSQL, zato mora razvijalec pri izbiri programske opreme biti pozoren, da program podpira zastavljeno obliko PB. Med razvijem je precej pomembna možnost testiranja PB z izvajanjem poizvedb. Večina programov dandanes to lastnost ponuja in obenem vrača statistike poizvedb s časi izvajanj, ter tako vrača povratne informacije o efektivnosti dizajna PB. V nadaljevanju so našteta najbolj predlagana orodja za razvoj PB.

dbForgeStudio: Namizni program za MySQL in MariaDB podatkovne baze, ki z brezplačnim planom ponuja najbolj osnovne funkcije za upravljanje PB, kot je dizajniranje podatkovne baze in podpora izvajanja poizvedb. S plačljivim planom uporabnik pridobi možnost kreiranja rutin in izvajanja drugih operacij v podatkovnih bazi, kot je vstavljanje, posodabljanje in brisanje podatkov. Program ponuja vizualno urejanje podatkovne baze z uporabo diagramov. Vsebovana so tudi orodja za olajšano izdelavo varnostnih kopij podatkovne baze. Program je narejen zgolj v različici za operacijske sisteme Microsoft Windows.

Vertabelo: Plačljiva spletna aplikacija za načrtovanje PB tipa SQL, ki omogoča sodelovanje med več uporabniki na istem projektu. Podprte so podatkovne baze PostgreSQL, MySQL, Microsoft SQL Server, SQLite, Oracle, IBM DB2 in HSQLDB. Člani uporabniške ekipe imajo lahko različne pravice, na primer lastnik, urejevalec ali pregledovalec. Aplikacija ne more izvajati poizvedb v bazi, saj se ne more povezati z PB. Aplikacija ima uporabniku prijazen pristop za vizualno urejanje podatkovne baze z diagrami. V aplikacijo je mogoče uvoziti obstoječo PB in urejati njen dizajn. Med oblikovanjem se lahko izdela SQL skripta, ki se jo lahko uporabi za udejanjenje načrtovane baze. Ker je aplikacija dostopna prek spleta, lahko uporabnik do nje dostopa z vseh platform, ki podpirajo uporabo internetnega brskalnika.

Lucidchart: Spletna aplikacija, namenjena izdelovanju različnih diagramov, s katerimi lahko uporabniki načrtujejo PB. Aplikacija ponuja brezplačno ali plačljivo uporabo. Uporabniki brezplačnega plana imajo dostop do osnovnih funkcionalnosti, plačniki pa imajo na voljo tudi možnost sodelovanja z drugimi uporabniki na istem projektu in dostop do drugih funkcionalnosti aplikacije, na voljo pa imajo tudi več podatkovnega prostora v aplikaciji, kar omogoča izvedbo večjih projektov.

SqlDBM: Podobno kot Vertabelo, je SqlDBM spletna aplikacija za načrtovanje PB tipa SQL, ki omogoča sodelovanje med več uporabniki. Podprti sta dve podatkovni bazi MySql

in Microsoft SQL Server. Aplikacija omogoča uvažanje obstoječih PB in nadaljnje urejanje le-teh. Na voljo je tudi brezplačna uporaba aplikacije, vendar so funkcionalnosti v tej verziji precej omejene, uporabnik lahko na primer naenkrat dela le na enem projektu.

Aqua Data Studio: Plačljiv namizni program, ki podpira operacijske sisteme Microsoft Windows, Linux in Apple macOS. Program omogoča razvoj in urejanje več kot 28 podatkovnih baz tipa SQL in NoSQL, torej podpira upravljanje večine uporabljanih PB. Podobno kot program dbForgeStudio, je na voljo povezava s PB, kar dovoljuje izvajanja poizvedb in drugih operacij za spreminjanje podatkov v PB. Slabost orodja Aqua Data Studio je, da je izmed ponujenih orodij najdražje.

MySQL Workbench: Brezplačen namizni program, ki je podprt na operacijskem sistemu Microsoft Windows in na nekaterih različicah sistema Linux. Program ponuja precejšnjo izbiro orodij za kreiranje, urejanje in testiranje PB, vendar je uporaba omejena zgolj na podatkovne baze MySQL [2].

2.3.2 Ogrodja za implementacijo spletnih aplikacij

V današnjem času je na voljo veliko različnih ogrodij za razvijanje spletnih aplikacij. Vsaka spletna aplikacija je sestavljena iz uporabniškega vmesnika ter API-ja, ki skrbi za komunikacijo med uporabniškim vmesnikom in PB. V nadaljevanju so našteta med razvijalci priljubljena ogrodja za implementacijo spletnih aplikacij. Veliko ogrodij, namenjenih izdelavi uporabniškega vmesnika v spletnih aplikacijah, je zgrajenih na osnovi programskega jezika JavaScript in si delijo precej podobnosti [19]. Vsa našteta ogrodja ponujajo načine implementacije, s katerimi se lahko dosežejo isti cilji. Razlikujejo se predvsem v zahtevnosti učenja, kjer je Angular ogrodje najtežje, sledi mu React in nato Vue.js z najlažjo stopnjo zahtevnosti [39].

2.3.2.1 Angular

Angular je ogrodje, ki se uporablja za razvoj spletnih, namiznih in mobilnih aplikacij. Razvijanje aplikacij v Angular ogrodju vključuje uporabo programskih in označevalnih jezikov TypeScript, HTML in CSS. Ogrodje Angular podpira »Single Page« aplikacije. To so aplikacije, pri katerih se ob zagonu naloži ena sama HTML stran, ki vsebuje dinamične komponente. Taka aplikacija lahko v ozadju komunicira s storitvami na strežnikih brez osveževanja celotne HTML strani, kar običajno pomeni hitrejšo odzivnost aplikacije in boljšo uporabniško izkušnjo. Velika prednost ogrodja Angular je organizacija programske kode z moduli. Vsak modul, imenovan tudi komponenta, predstavlja del uporabniškega vmesnika, in je lahko neodvisen od drugih modulov. Z modularnostjo je precej lažja

delitev dela, v kolikor aplikacijo razvija več ljudi hkrati [26]. Angular uporabljajo številne velike organizacije, kot so Google, Microsoft in Paypal [19].

2.3.2.2 React

React je knjižnica programskega jezika JavaScript, vendar ga mnogi razvijalci kategorizirajo kot ogrodje. React knjižnica je bila prva, ki je uporabila arhitekturo na osnovi komponent, pozneje se je po njej zgledoval tudi Angular in ogrodje Vue.js. Prednost te knjižnice je neodvisnost od drugih tehnologij, ki se uporabljajo za razvoj določene spletne aplikacije. Uporabljajo se virtualni DOM elementi, ki se vstavljajo v HTML stran. V primerjavi z ogrodjem Angular, React ponuja več fleksibilnosti, saj mora uporabnik v primeru uporabe ogrodja Angular uporabljati zgolj tehnologije, ki jih le-ta ponuja. React uporabljata organizaciji Facebook in Instagram [9].

2.3.2.3 Vue.js

Vue.js je ogrodje, ki v zadnjem času najbolj pridobiva na priljubljenosti. Ogrodje je precej podobno knjižnici React, saj se pri obeh uporabljajo virtualni DOM elementi. Prednost ogrodja Vue.js je dokumentacija, v kateri so opisane rešitve za veliko večino težav, s katerimi se razvijalci pri razvoju lahko srečajo. Ogrodje je zelo dobro optimizirano, kar pomeni, da aplikacija uporablja manj sredstev za delovanje, kot pri drugih ogrodjih [4]. Kljub svoji enostavnosti, pa je lahko to ogrodje dvorezen meč, saj dovoljuje »slabo kodo«, kar otežuje reševanje težav, ki se lahko pojavijo [10].

2.3.3 Uporaba SOA (arhitektura, ki temelji na storitvah)

SOA je način dizajniranja programov, kjer se za komunikacijo med komponentami sistema uporabljajo omrežni komunikacijski protokoli. V taki arhitekturi sta prisotni dve vlogi, to sta ponudnik storitve in potrošnik storitve. Ponudnik storitve mora poskrbeti, da je storitev vzdrževana in vedno na voljo za uporabo s strani potrošnika. Naloga ponudnika je tudi, da potrošnika informira o pravilni uporabi storitve in definira zahteve in stroške za uporabo.
Potrošnik storitve mora za uporabo poravnati morebitne stroške, ki jih določi ponudnik, ter tako pridobi pravico do uporabe storitve. Vzdrževanje storitev je načeloma nezahtevno opravilo, saj so storitve medsebojno neodvisne, kar pomeni da spreminjanje določene storitve ne vpliva na drugo. Prednost arhitekture SOA je enostavna dostopnost, saj so po navadi dostopne z uporabo zahtev preko spleta [12].

2.3.3.1 SOAP spletni servisi

SOAP (Simple Object Access Protocol) je specifikacija protokola sporočil za izmenjavo informacij in implementacijo spletnih storitev. Sporočila so sestavljena z uporabo formata XML, kar omogoča enostavno razhroščevanje v primeru napak na nivoju uporabe spletnih storitev. Slabost SOAP protokola je količina odvečnih podatkov pri pošiljanju sporočil s kratko vsebino, saj je količina podatkov za specifikacijo izbrane storitve lahko večja od same vsebine sporočila [36].

2.3.3.2 REST spletni servisi

REST (Representational State Transfer) je stil programske arhitekture, ki definira nabor omejitev, ki se jih je treba držati pri izdelavi spletnih storitev. V zadnjih nekaj letih je postal pristop izdelave spletnih storitev s stilom REST precej popularen. Razlog za naraščanje popularnosti je preprostost stila ter manjša velikost sporočil pri zahtevah storitev, v primerjavi s protokolom SOAP. Zahteve s pristopom REST so definirane z naslednjimi kriteriji:
Osnovni URL: Naslov URL kaže na osnovno lokacijo na strežniku, ki je namenjen obdelavi zahtev s strani potrošnika (primer: http://api.example.com/collection/).
Tip medija: Definicija tipa podatka, ki ga potrošnik pošlje skupaj z zahtevo.
HTTP metoda: Način zahteve, ki jo potrošnik hoče izvesti. Seznam HTTP metod:

- GET – prejem podatkov;
- POST – vnos določenega podatka v določeno zbirko podatkov;
- PUT – zamenjava določenega podatka v določeni zbirki, ali pa vnos podatka, če le-ta v zbirki še ne obstaja;
- PATCH – posodobitev določenih lastnosti podatka v določeni zbirki;
- DELETE – izbris določenega podatka iz določene zbirke.

Dodatne možnosti pri zahtevi so določitev v glavi klica, primer dodatne možnosti so poverilnice za avtorizacijo klica zahteve [13].

Na strani ponudnika storitve je priporočena uporaba HTTP kode statusa, ki jih vsebuje odgovor na zahtevo. V kodi statusa je povzeto, kaj se zgodi z zahtevo po obdelavi na strežniku, ki ponuja storitev. Primer je koda »200 – OK«, ki jo lahko ponudnik vrne v odgovoru po uspešni obdelavi. Še en primer je koda »401 – Unauthorized«, ki pomeni da je potrošnik v glavi zahteve podal napačne podatke za avtorizacijo zahteve, ali pa podatkov sploh ni vstavil [3].

3 NAČRTOVANJE SPLETNEGA INFORMACIJSKEGA SISTEMA ZA NAKUP IN PRODAJO KOLESARSKE OPREME

Cilj zaključne naloge je bil postaviti spletni informacijski sistem, ki bi uporabnikom omogočal prodajo in nakup koles in kolesarske opreme prek oglaševanja v spletni aplikaciji. Načrtovanje in implementacijo spletnega IS smo izvedli s pristopom razvoja IS s prototipom.

3.1 Predstavitev problema

V Sloveniji se število kolesarjev povečuje, leta 2015 je raziskava Evropske kolesarske federacije pokazala, da Slovenija na lestvici kolesarskega barometra zaseda sedmo mesto izmed 27 držav Evropske unije. Na lestvici prodanih koles na število prebivalcev, pa Slovenija zaseda prvo mesto. Raziskava je pokazala, da novo kolo vsako leto kupi 12% populacije [6]. Težave lahko nastopijo pri nakupu novih ali rabljenih koles s pomočjo spletnega iskanja. Oglasi, ki so namenjeni prodaji kolesa ali kolesarske opreme, se trenutno nahajajo na spletnih straneh, kot so bolha, Salomonov spletni oglasnik in Facebook Marketplace. Za prodajo koles v Sloveniji ni določene spletne strani, ki bi med uporabniki poznana kot glavna za ta namen, kakor je na primer za motorna vozila stran avto.net. Prednost spletne strani, ki je določena namenu prodaje specifičnih izdelkov, je boljši način iskanja želenega izdelka. Običajno je na taki spletni strani ponujen širok spekter iskalnih filtrov, ki uporabniku omogočajo specifično iskanje. Spletne strani, namenjene splošnim oglasom, vsebujejo zgolj splošne filtre iskanja, kar pogosto za iskalce oglasov ni dovolj. Ker v Sloveniji nimamo določene strani za objavljanje oglasov za kolesa, morajo uporabniki preverjati več spletnih strani, da lahko najdejo ustrezen izdelek.

3.2 Analiza spletnega informacijskega sistema za nakup in prodajo

Pred začetkom načrtovanja novega spletnega informacijskega sistema so bile zbrane informacije med nekaterimi uporabniki spletnih IS, ki ponujajo nalaganje in ogled oglasov za različne izdelke.

3.2.1 Zbiranje informacij

Uporabnikom spletnih IS, ki omogočajo ogled in nalaganje oglasov za različne izdelke, smo postavili naslednja vprašanja.

- »Kateri spletni informacijski sistem bi izbrali za oddajo ali iskanja oglasa za kolo ali kolesarske izdelke?«;
- »Ali v izbranem spletnem IS opazite pomanjkljivosti ali nevšečnosti?«;

- »Ali v izbranem spletnem IS vidite možnosti za izboljšave?«.

Uporabniki so kot odgovor na prvo vprašanje večinoma navajali slovensko spletno stran *Bolha*, spletno in mobilno aplikacijo *Letgo*, ter funkcionalnost socialnega omrežja Facebook, imenovano *Facebook Marketplace*. V nadaljevanju so naštete pomanjkljivosti posamezne aplikacije oziroma spletne strani.

Bolha: Spletna stran bolha.com je med slovenskimi uporabniki zelo priljubljena za iskanje in nalaganje spletnih oglasov. Uporabniki so za izbrano spletno stran navedli, da mobilna verzija spletne strani ni zasnovana najbolje, gumb za oddajanje oglasa je skrit v meniju. Spletna stran omogoča tudi nalaganje oglasov brez fotografij izdelka, kar pa uporabnikom ni všeč. Kot možne izboljšave spletne strani so uporabniki navedli bolj uporabniku prijazen mobilni pogled, ter uvedbo obveznega dodajanja fotografij med nalaganjem oglasa. Pomanjkljivi so tudi filtri pri iskanju, zaradi majhnega števila možnosti pri označevanju filtrov je izbor oglasov manj podroben.

Letgo: Mobilna in spletna aplikacija Letgo je bila med uporabniki precej priljubljena, vendar je julija 2019 prenehala z delovanjem v Sloveniji. Vprašani uporabniki so med uporabo našli pomanjkljivost nepravilnega delovanja iskanja.

Facebook Marketplace: Funkcionalnost socialnega omrežja Facebook, ki jo zaradi prenehanja delovanja priljubljene aplikacije Letgo, uporablja vedno več uporabnikov. Prednost storitve Marketplace je obvezno dodajanje fotografije izdelka ob oddaji oglasa, kar je dobrodošlo pri iskalcih oglasov. Pomanjkljivost storitve je odsotnost filtriranja oglasov, pri iskanju pa je omogočena le izbira kategorije izdelka.

3.2.2 Identifikacija izboljšav

Iz zbranih podatkov lahko sklepamo, da uporabnikom pri uporabi spletnih IS za namen prodaje in kupovanja izdelkov, precej pomeni dobro zasnovan uporabniški vmesnik. Spletna ali mobilna aplikacija takega IS mora imeti torej uporabniku prijazen uporabniški vmesnik. Možna izboljšava pri spletni strani Bolha, bi bila boljša postavitev elementov uporabniškega vmesnika mobilne verzije, ki so namenjeni pomembnim funkcionalnostim aplikacije. Zagotoviti je treba tudi, da funkcionalnosti določenega sistema delujejo pravilno. Pri aplikaciji Letgo bi bila smiselna izboljšava popravilo iskalnega modula.

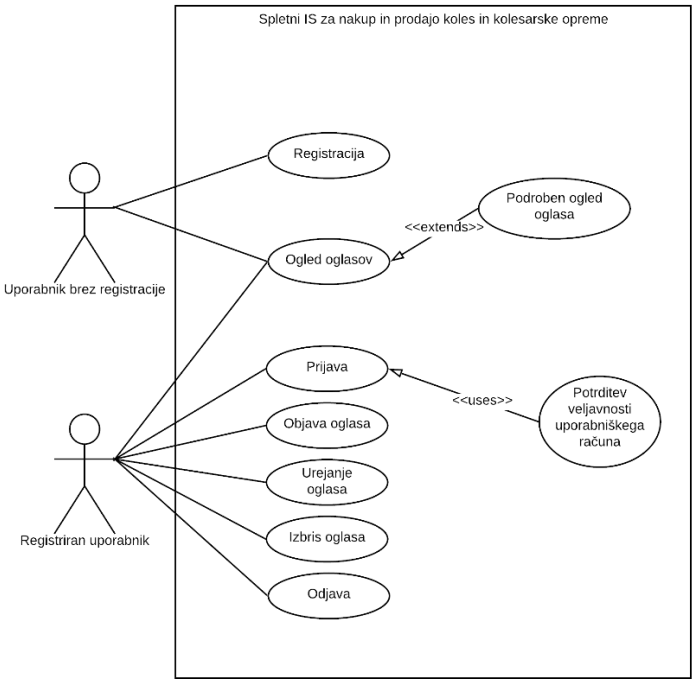
3.2.3 Izdelava koncepta novega sistema

Nov sistem, ki bo omogočal objavljanje oglasov za nakup ali prodajo koles in kolesarske opreme bo uporabnikom dostopen prek uporabe spletne aplikacije. Ker je vsebina spletnega IS omejena zgolj na kolesa in kolesarsko opremo, je smiselno omogočiti funkcionalnost podrobnega iskanja izdelkov. Uporabniški vmesnik spletne aplikacije bo

preprost, elementi vmesnika bodo postavljeni na mestih, ki bodo dobro vidna oziroma kjer bi jih uporabnik pričakoval.

3.2.3.1 Diagram primerov uporabe

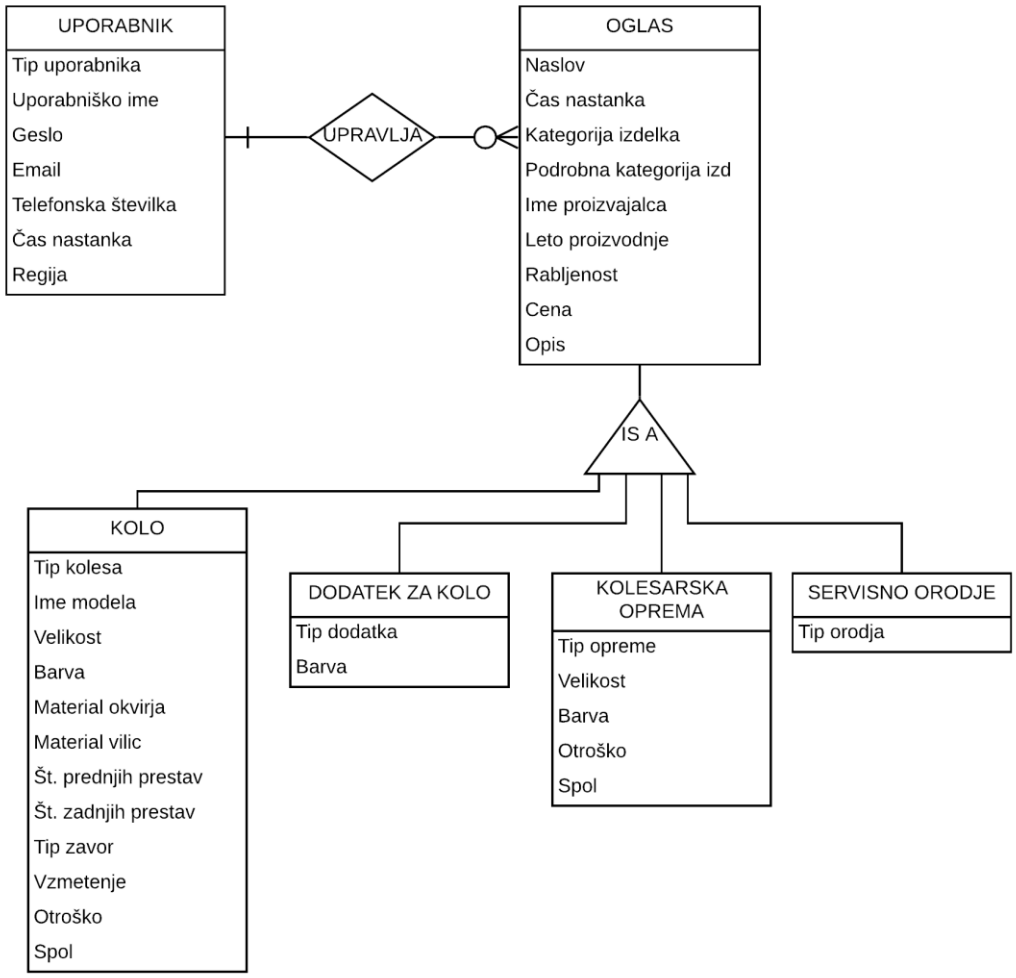
Uporabnik bo brez prijave lahko dostopal do vseh oglasov, ki bodo naloženi v spletni IS. S funkcionalnostjo filtriranja oglasov glede na uporabnikov vnos, bo uporabnik naredil selekcijo med oglasi. Na prvotnem prikazu spletne aplikacije bo na voljo osnovna izbira oglasov, na izbiro bodo kolesa, dodatki za kolo, kolesarska oprema ter servisna orodja. Po izboru splošne kategorije oglasov, se bo prikazal drugi nivo izbire, s katerim bo uporabnik še bolj podrobno izbral kategorijo izdelka. Po izbiri kategorije na drugem nivoju, se bodo prikazali vsi oglasi pripadajoče kategorije. Med prikazanimi oglasi bo lahko uporabnik z uporabo dodatnih filtrov še natančneje omejil izbor oglasov. Z izbiro določenega oglasa bo uporabnik lahko videl podrobnosti izbranega oglasa. Omogočena bo registracija in prijava v sistem, kar bo pogoj za dostop do možnosti kreiranja oglasov. S pogojem prijave v sistem zmanjšamo možnost za morebitne zlorabe funkcionalnosti dodajanja oglasov. S prijavo v sistem bo uporabnik lahko svoje objavljene oglase tudi spreminjal in brisal.



Slika 4: Diagram primerov uporabe.

3.3 Definicija podatkov

V tem poglavju so definirane entitete in relacije, prisotne v načrtovanem spletnem IS. Vizualno so entitete in relacije predstavljene v ER diagramu, predstavljenem na sliki 5.



Slika 5: ER diagram spletnega informacijskega sistema.

3.3.1 Entitete

Uporabnik je oseba, ki bo uporabljala novi spletni IS. Uporabnik bo lahko ustvarjal, posodabljal in brisal svoje oglase z izdelki v sistemu. Za potrebe novega sistema nas zanimajo naslednji podatki uporabnika: tip uporabnika (zasebni uporabnik ali podjetje), uporabniško ime, geslo, e-mail naslov, telefonska številka, čas nastanka uporabnika v sistemu in regija v kateri se uporabnik nahaja.

Oglas je nabor podatkov ki opisuje splošne lastnosti izdelka, ki ga uporabnik prodaja oziroma oddaja. Splošne lastnosti, ki jih entiteta oglas vsebuje, so: naslov/ime oglasa, čas nastanka oglasa, kategorija izdelka, podrobna kategorija izdelka, ime proizvajalca, leto proizvodnje/nakupa, stanje izdelka, cena in opis.

Kolo je tip izdelka, ki zavzema vse kategorije koles. Atributi kolesa, ki jih vsebuje entiteta, so: tip kolesa, ime modela, velikost, barva, material okvirja, material vilic, število prednjih in zadnjih prestav, tip zavor, način vzmetenja, ali je kolo otroško ter za kateri spol je kolo namenjeno.
Dodatek za kolo je entiteta, ki zajema vse dodatke kolesa. Primeri dodatkov za kolo so števci, kolesarske torbe, luči in tako naprej. Za potrebe spletnega IS entiteto opisujeta tip dodatka, in barva dodatka.
Kolesarska oprema je entiteta, s katero je zajeta oprema, ki jo kolesar lahko nosi na kolesu. Atributi kolesarske opreme so: tip opreme, velikost, barva, za kateri spol je oprema namenjena ter ali je oprema namenjena za otroke.
Servisno orodje so vsi pripomočki, namenjeni vzdrževanju ali popravilu kolesa. Entiteta vsebuje zgolj en atribut, to je tip orodja.

3.3.2 Relacije

V entitetno-relacijskem diagramu, prikazanem na sliki 5, sta prisotni le dve relaciji. Prva relacija obstaja med uporabnikom in oglasom, druga pa je generalizacija med entiteto oglas in kategorijami izdelkov.
Relacija uporabnik – oglas je opcijsko binarno razmerje. Uporabnik lahko upravlja z nič ali več (N) oglasi, zato je številnost razmerja ena proti mnogo.
Generalizacija je narejena z namenom posplošitve podobnosti entitet, ki predstavljajo izdelke različnih kategorij. Super-razred v tem razmerju je entiteta »oglas«, podrazredi razmerja pa so entitete »kolo«, »dodatek za kolo«, »kolesa« in »servisno orodje«.

3.4 Načrtovanje podatkovne baze

Zaradi manjšega obsega entitet in relacij, je korak integracije shem in pogledov izpuščen.

3.4.1 Konceptualno načrtovanje podatkovne baze

V tem koraku se klasificirajo entitete, atributi, relacije in hierarhična razmerja v informacijskem sistemu. V definiranih entitetah je smiselna ločitev atributov »tip uporabnika« in »regija« entitete »uporabnik«. Našteta atributa postaneta novi entiteti sistema, saj obstaja možnost, da bodo v prihodnosti spletnega IS zahtevani dodatni atributi teh entitet. Primer take zahteve bi bilo spreminjanje določenega tipa uporabnika, saj obstaja možnost spreminjanja pravic določenega tipa. Novi entiteti se imenujeta imenovani »uporabniška pravica« in »regija«. Z nastankom dveh novih entitet se definirata tudi dve novi razmerji. Prvo razmerje je med entitetama »uporabnik« in »uporabniška pravica«, ki je številnosti 1:1, vsak uporabnik ima lahko natanko eno uporabniško pravico. Drugo

razmerje obstaja med entitetama »uporabnik« in »regija«. Tudi to razmerje je številnosti 1:1, vsak uporabnik prebiva natanko v eni regiji. Ostale relacije med entitetami so že definirane v poglavju 3.3.2, prav tako je definirano hierarhično razmerje med super-razredom »oglas« in različnimi tipi izdelkov, ki v razmerju predstavljajo podrazrede. Vse entitete in razmerja so predstavljena na sliki 6.

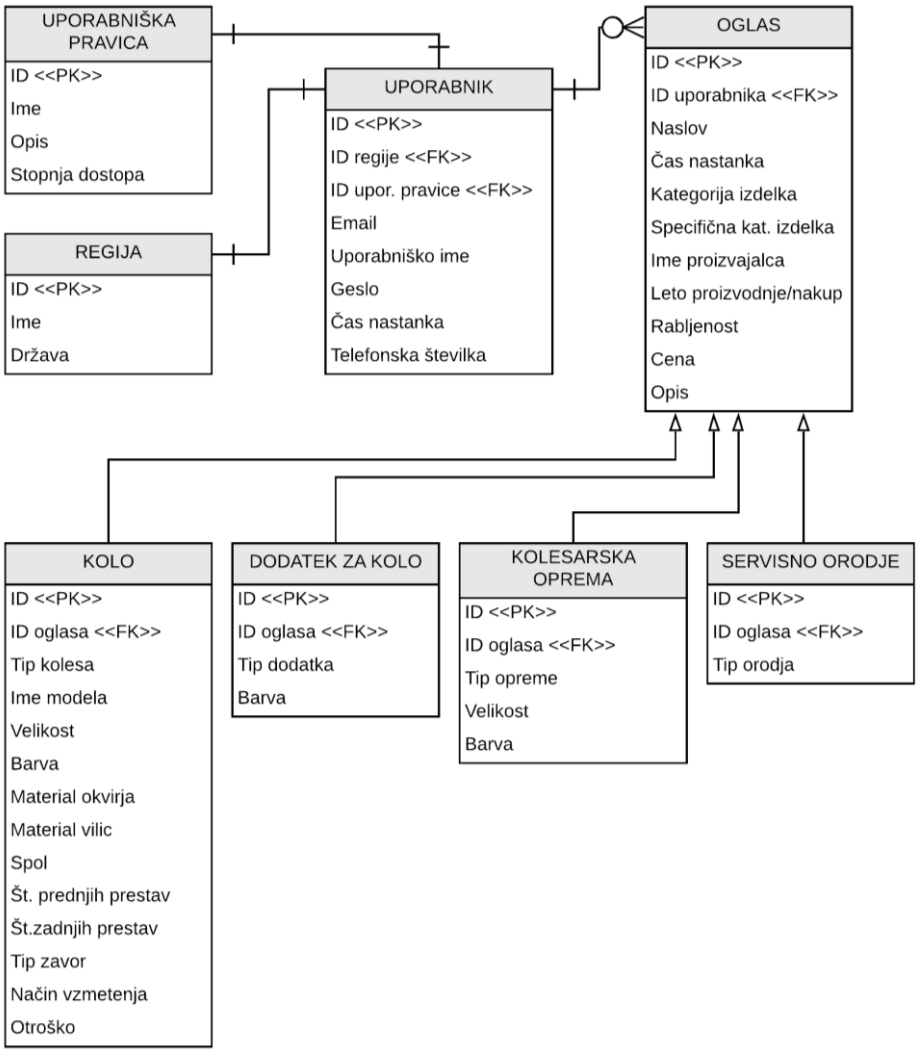
3.4.2 Pretvorba konceptualnega modela v SQL tabele

Konceptualni model načrtovane podatkovne baze ne vsebuje nobenega rekurzivnega razmerja, razmerja tipa N-N. Prav tako nobeno od razmerij ni ternarnega ali višjega reda, zato v pretvorbi konceptualnega modela ni bilo potrebe po pretvorbi razmerij.
Entitete, ki so od drugih entitet neodvisne, se prevedejo v SQL tabele, ki imajo isto informacijsko vsebino kakor entitete, iz katerih so bile tabele razvite. V načrtovani podatkovni bazi se tako entiteti »uporabniška pravica« in »regija« prevedeta v istoimenski SQL tabeli. Pri prevajanju ostalih entitet načrtovane PB, ki so medsebojno odvisne, pa je bilo potrebno prevajanje v SQL tabele z dodanimi tujimi ključi drugih entitet. V nadaljevanju so opisane pretvorbe odvisnih entitet.
Uporabnik: Entiteta je odvisna od entitet »uporabniška pravica« in »regija«, zato prevedena SQL tabela vsebuje dva tuja ključa. Prvi tuji ključ se imenuje »ID regije« in se navezuje na primarni ključ tabele »regija«. Drugi tuji ključ se imenuje »ID uporabniške pravice« in se navezuje na primarni ključ tabele »uporabniška pravica«.
Oglas: Entiteta je odvisna od entitete »uporabnik«, zato se pri prevajanju v SQL tabelo doda tuji ključ »ID uporabnika«, ki se navezuje na primarni ključ tabele »uporabnik«.
Kolo, Dodatek za kolo, Kolesarska oprema, Servisno orodje: Vse štiri entitete so podrazredi entitete »Oglas«, zato se pri prevajanju vsake izmed naštetih entitet, doda tuji ključ. Ime tujega ključa je »ID oglasa« in se navezuje na primarni ključ tabele »oglas«.

3.4.3 Normalizacija SQL tabel

Cilj pri normalizaciji podatkovne baze v sklopu načrtovanega spletnega IS je tretja normalna oblika. To pomeni, da mora podatkovna baza najprej ustrezati prvi in drugi normalni obliki. Pogoj prve normalne oblike je, da vsak stolpec vsebuje le podatke, ki so posameznih vrednosti, množice vrednosti niso dovoljene. Temu pogoju ustrezajo vse SQL tabele načrtovane podatkovne baze. Načrtovana podatkovna baza izpolnjuje tudi pogoje druge normalne oblike. Pogoj za drugo normalno obliko je, da baza zadostuje pogojem prve NO, ter da je vsak atribut SQL tabele, ki ni primarni ključ, odvisen od celotnega primarnega ključa. V načrtovani PB so vsi atributi, odvisni zgolj od celotnega primarnega ključa. Podatkovna baza zadostuje tudi pogojem tretje NO, saj so vsi atributi posamezne

SQL tabele neodvisni drug od drugega. Tako se onemogoči možnost pojavljanja anomalij pri vnašanju, posodabljanju ali brisanju zapisov.



Slika 6: SQL tabele podatkovne baze.

3.4.3 Fizično načrtovanje

V tem delu je opisana postavitev indeksov na posameznih SQL tabelah, kjer je to smiselno. Tabeli »uporabniška pravica« in »regija« sta namenjeni manjši zalogi podatkov, zato postavljanje indeksov na teh dveh tabelah ni smiselno. V nadaljevanju so opisane postavitve indeksov za posamezne SQL tabele.

Uporabnik: Večina poizvedb, ki se bodo izvajale v tabeli »uporabnik«, bo kot ključ iskanja uporabljala primarni ključ tabele, ali stolpec »Email«. Prvi indeks je torej postavljen na stolpcu »ID«, drugi pa na stolpcu »Email«.

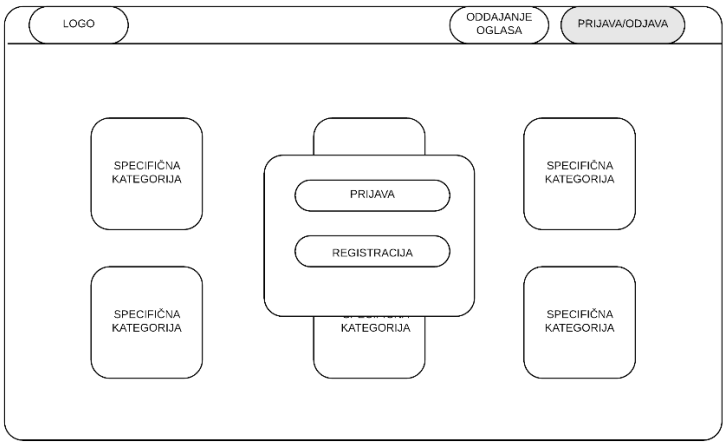
Oglas: Tabela »Oglas« bo tabela z največ vpisi, zato je smiselna postavitev indeksov na več stolpcih tabele. Poizvedbe, izvedene v tej tabeli, se bodo največkrat sklicevale na

stolpec »Specifična kategorija izdelka«, zato se prvi indeks postavi na ta stolpec. Naslednje pogostejše poizvedbe se bodo sklicevale na stolpca »ID uporabnika« in »Ime proizvajalca« katerima pripadata drugi in tretji indeks.
Kolo, Dodatek za kolo, Kolesarska oprema, Servisno orodje: V vseh štirih tabelah se bodo klicale podobne poizvedbe, ki se bodo sklicevale na stolpec, prisoten v naštetih tabelah. To je »ID oglasa«, zato se na vsaki izmed naštetih tabel postavi indeks na imenovanem stolpcu.

3.5 Načrtovanje uporabniškega vmesnika

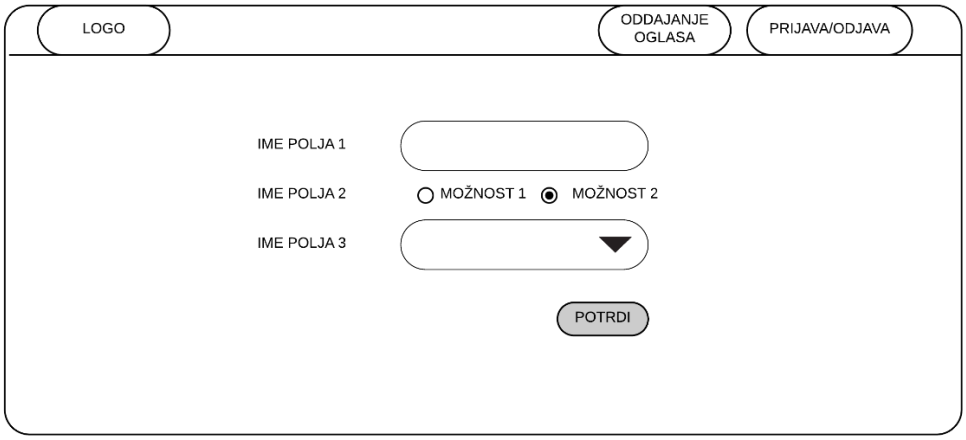
Dizajn spletne aplikacije, ki bo uporabniški vmesnik načrtovanega IS, mora biti preprost, uporabniku uporaba aplikacije ne sme biti zahtevna. V dizajnu bodo uporabljeni zgolj elementi, ki so deli standardnih podobnih aplikacij in bodo uporabniku znani. Uporabljeni kontrolni elementi bodo gumbi, »radio button« gumbi, spustni seznami in polja z vnosom teksta. Pri oblikovanju je potrebno paziti, da so gumbi za pomembne funkcionalnosti na mestih, kjer jih uporabnik pričakuje. V nadaljevanju je za vsak korak v uporabi aplikacije opisan princip oblikovanja.

Pogled z izbiro kategorij izdelkov: V prvotnem pogledu se uporabniku omogoči izbiro izdelkov. V prvem koraku lahko uporabnik izbere kategorijo izdelka, v drugem pa specifično kategorijo izdelka, glede na izbrano kategorijo iz prvega koraka. Obenem mora imeti tudi možnost prijave v sistem, kar uporabniku dovoljuje kreiranje oglasov. Pogled je predstavljen na sliki 7. Na sliki so kategorije predstavljene kot veliki gumbi, ki vizualno predstavljajo posamezno kategorijo. Možnost prijave se nahaja na desni strani menija na vrhu strani. Desna stran menija je namenjena logotipu oziroma naslovu spletne aplikacije, ki uporabnika pripelje na uvodno stran aplikacije. Gumb za oddajanje oglasa se prikaže le ob prijavi uporabnika. Klik na gumb za prijavo sproži prikaz pojavnega okna z gumboma za prijavo in registracijo. Meni se nahaja v zgornjem delu vsakega pogleda v aplikaciji.



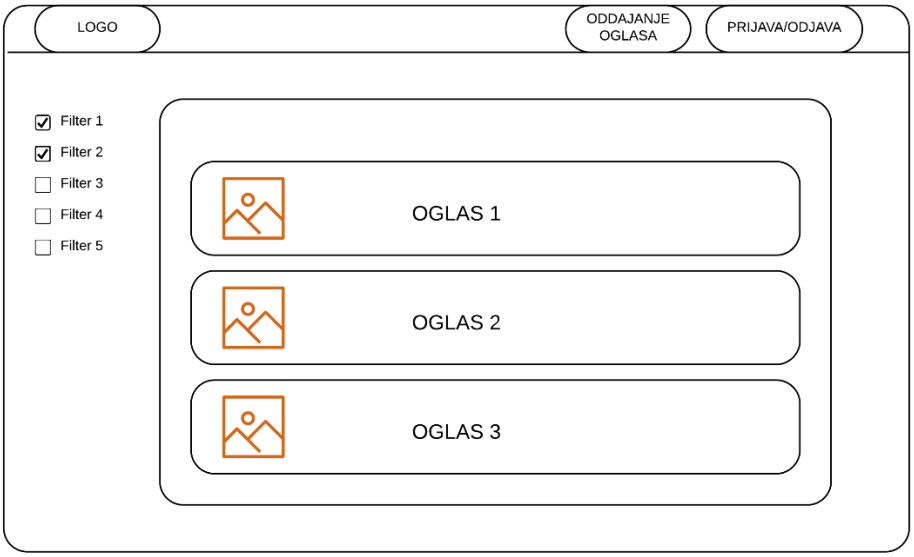
Slika 7: Pogled z izbiro kategorije in prijavnim oknom.

Pogled z vnašanjem podatkov: Podatki se vnašajo v koraku registracije uporabnika in pri vnašanju oglasa. Uporabljene so štiri različice kontrolnih elementov, ki so predstavljene na sliki 8. Kontrolni element, ki na sliki pripada »polju 1«, je tekstovno polje in je namenjeno vnašanju teksta. »Polje 2« predstavlja izbiro možnosti z uporabo »radio button« gumbov, »polje 3« pa predstavlja možnost izbire iz spustnega menija, ki je primeren za večje število možnosti. Potrditev vnosa podatkov se izvede z gumbom, ki se nahaja v spodnjem delu pogleda za vnašanje podatkov.



Slika 8: Pogled z vnašanjem podatkov.

Pogled z izbiro oglasov: Pogled, ki je predstavljen na sliki 9, predstavlja korak, kjer uporabnik izbira oglase določene kategorije. Na levi strani pogleda se nahajajo filtri za podrobnejšo opredelitev oglasov, za prikaz je uporabljen kontrolni element »checkbox«, ki uporabniku omogoča izbiro več možnosti. Oglasi so prikazani kot veliki gumbi s slikami in napisi, ki predstavljajo splošne podatke oglasa.



Slika 9: Pogled z izbiro oglasov.

3.6 Izbira ogrodij za implementacijo spletnega informacijskega sistema

Za implementacijo spletnega IS sta bili izbrani ogrodji *Angular* in *Apache CXF*.
Ogrodje *Angular* je bilo izbrano z namenom implementacije spletne aplikacije. Razlog za izbiro so izkušnje z uporabo tega ogrodja, ter knjižnice z implementiranimi uporabniškimi elementi, ki ustrezajo zahtevam načrtovanega uporabniškega vmesnika.
Ogrodje *Apache CXF* je bilo izbrano z namenom implementacije komunikacije med podatkovno bazo in spletno aplikacijo. Prav tako so razlog za izbiro tega ogrodja izkušnje z uporabo. Ogrodje omogoča implementacijo spletnih storitev tipa REST in SOAP.

3.7 Izbira programske opreme za implementacijo

Za urejanje podatkovne baze je bil izbran brezplačen program *MySQL Workbench.* Podatkovna baza, ki je bila pred začetkom projekta že postavljena na strežniku, uporablja tip MySQL, kar je bil razlog za izbiro programa.
Za implementacijo spletne aplikacije je bil izbran program *Visual Studio Code*. Program podpira večino programskih jezikov, med drugimi tudi jezik »TypeScript«, ki ga uporablja ogrodje »Angular«. Razširitve, ki jih ponuja program, omogočajo poenostavljeno izdelavo komponent in storitev ter nameščanje dodatnih knjižnic.
Za uporabo ogrodja »Apache CXF« je bil izbran program *Eclipse IDE*. Program omogoča dostop do repozitorija »Maven« z uporabo istoimenske razširitve. Iz repozitorija se lahko uvažajo knjižnice ogrodja »Apache CXF«.

4 IMPLEMENTACIJA

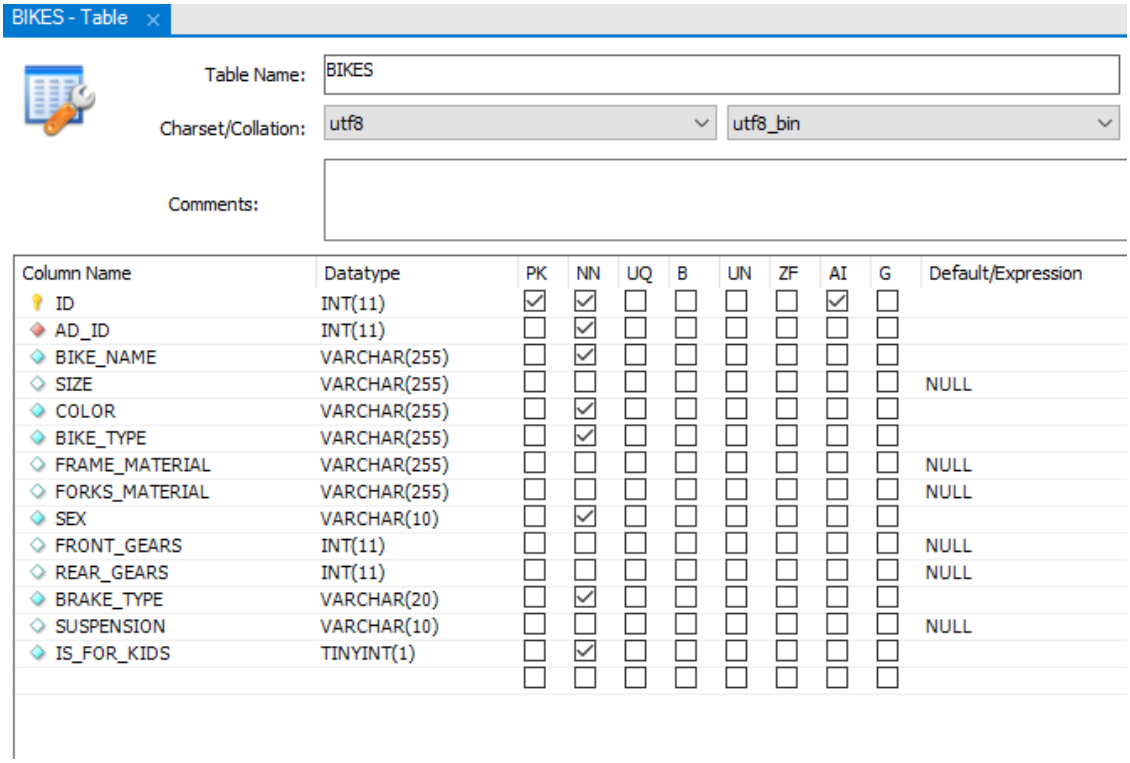
V tem poglavju je opisana implementacija spletnega IS za nakup in prodajo kolesarske opreme. Pred začetkom implementacije je bila podatkovna baza tipa MySQL že postavljena na določenem strežniku.

4.1 Implementacija podatkovne baze

Z uporabo programa »MySQL Workbench« je bila vzpostavljena povezava s podatkovno bazo. S pomočjo grafičnega vmesnika v programu so bile izdelane SQL tabele brez pisanja SQL stavkov.

4.1.1 Izdelava tabel

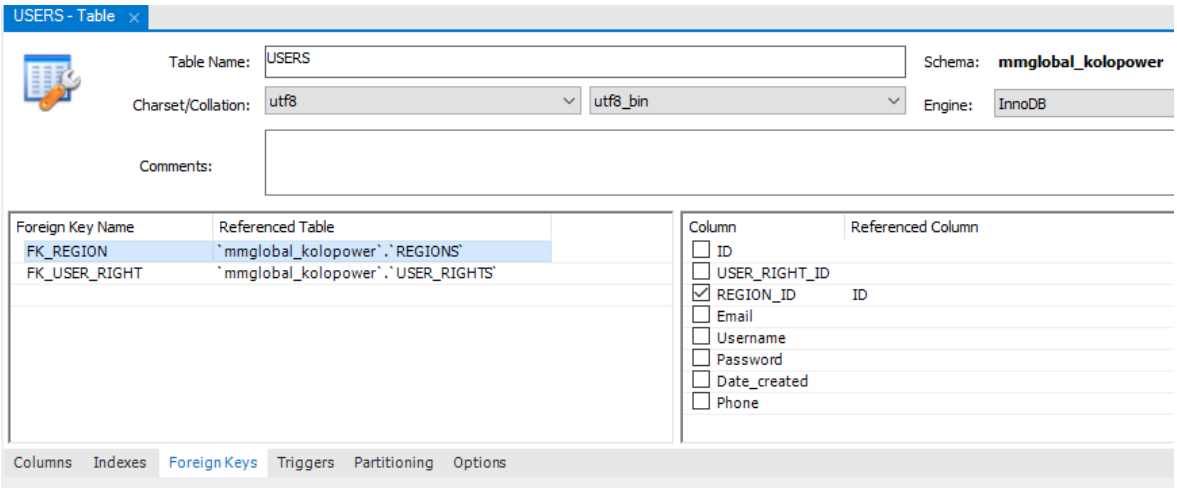
Na sliki 10 je predstavljena izdelava SQL tabele v programu za primer tabele »Kolo«. V prvem stolpcu grafičnega vmesnika je vstavljeno ime stolpca kreirane SQL tabele, v drugem stolpcu pa je določen tip podatka. Naslednji stolpci določajo ali je stolpec primarni ključ, ali je dovoljen vnos NULL in tako naprej. Na prikazani sliki vrstica z imenom stolpca »ID« predstavlja primarni ključ tabele, vrstica z imenom stolpca »AD_ID« pa predstavlja tuji ključ v tabeli. Zaradi izogibanja težavam s šumniki v imenih stolpcev in tabel, so bili uporabljeni prevodi v angleškem jeziku.



Slika 10: Izdelava SQL tabele v programu MySQL Workbench.

4.1.2 Izdelava razmerij med tabelami

Po končani izdelavi SQL tabel, je potrebno za definicijo razmerij povezati tuje ključe z ustreznimi primarnimi ključi drugih tabel. Na sliki 11 je predstavljena implementacija razmerij s tabelo »Uporabnik«. Prvo razmerje, ki obstaja med tabelo »Uporabnik« in tabelo »Regija«, je definirano s tujim ključem »REGION_ID«, ki se nanaša na primarni ključ tabele »Regija«. Podobno je storjeno tudi za drugo razmerje, ki obstaja med tabelo »Uporabnik« in »Uporabniška pravica«. Enaka operacija se izvede še v drugih tabelah, ki vsebujejo tuji ključ oziroma so v razmerju z drugimi tabelami.



Slika 11: Izdelava razmerij med SQL tabelami v programu MySQL Workbench.

4.2 Izdelava uporabniškega vmesnika z ogrodjem Angular

Z uporabo ogrodja Angular v programu Visual Studio Code, je bil narejen uporabniški vmesnik. Izbrani deli uporabniškega vmesnika so bili smiselno razdeljeni na komponente, ki jih omogoča ogrodje.

Prva izdelana komponenta uporabniškega vmesnika je bil meni, ki je v vsakem koraku aplikacije prikazan na zgornji strani ekrana. Gumbi v meniju se prilagajajo glede na stanje prijave uporabnika, primer je gumb za oddajanje oglasa, ki se omogoči šele, ko se uporabnik prijavi.

Komponente, izdelane v ogrodju Angular, med seboj komunicirajo z uporabo »service« komponent, v katerih lahko poslušajo določen objekt ali metodo, ter ob spremembi objekta ali sprožitvi metode prejmejo nove podatke, ki se lahko prikažejo v aplikaciji. Primer je objekt z imenom »loggedIn«, prikazan na sliki 12, vsebuje ga komponenta z imenom »LoginService«. Na sliki 13 je prikazana obdelava podatkov, ki se zgodi ob spremembi objekta z imenom »loggedIn«, v komponenti, ki ta objekt posluša.

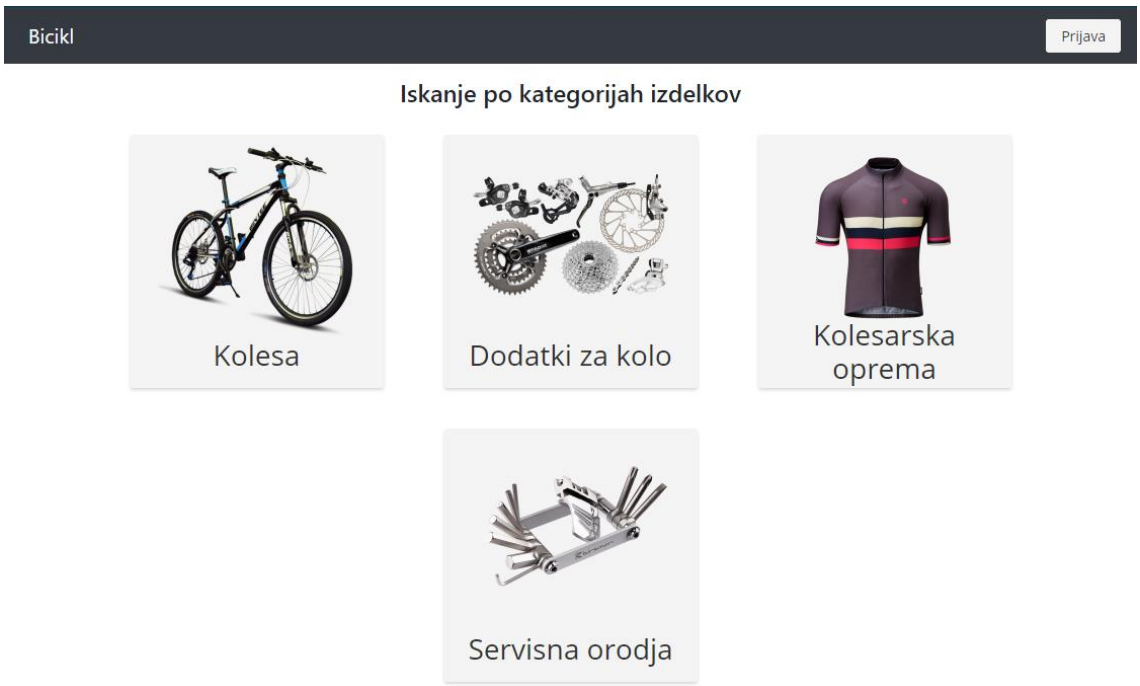
```
export class LoginService {

  public loggedIn = new Subject<boolean>();
```

Slika 12: Primer objekta, ki ga komponente lahko poslušajo.

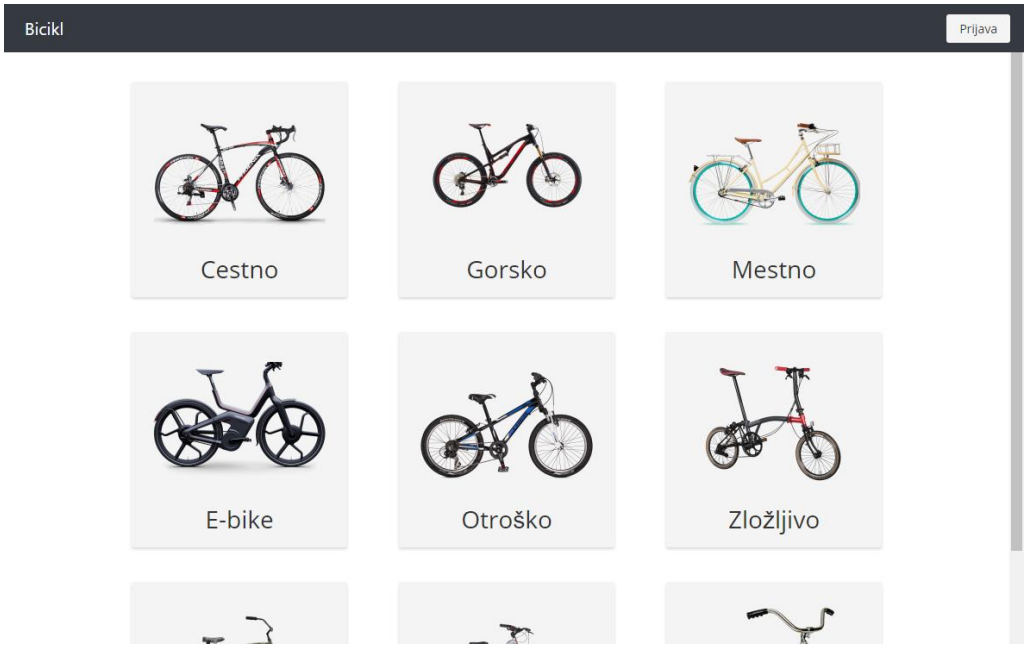
```
this.loginService.loggedIn.subscribe(
  value => {
    this.isLoggedIn = value;
    if (value === true && this.loginService.getGoogleUser() !== null) {
      if (this.loginService.getLoggedUser() !== undefined) {
        this.user = this.loginService.getLoggedUser()[0];
        this.loginService.setLoginDialog(false);
      }
    } else if (value === true) {
      this.user = this.loginService.getLoggedUser();
    }
    console.log('Is logged in: ' + this.isLoggedIn);

  }
);
```

Slika 13: Primer obdelave podatkov ob spremembi poslušanega objekta.

Naslednji izdelani komponenti sta namenjeni izbiri splošne in specifične kategorije izdelkov. Izdelani so bili veliki gumbi s slikami in napisi, ki uporabniku olajšajo izbiro kategorije. Komponenta za izbiranje splošne kategorije je predstavljena v sliki 14, komponenta za izbiro specifične kategorije pa v sliki 15, kjer je vsebina gumbov odvisna od izbrane splošne kategorije.

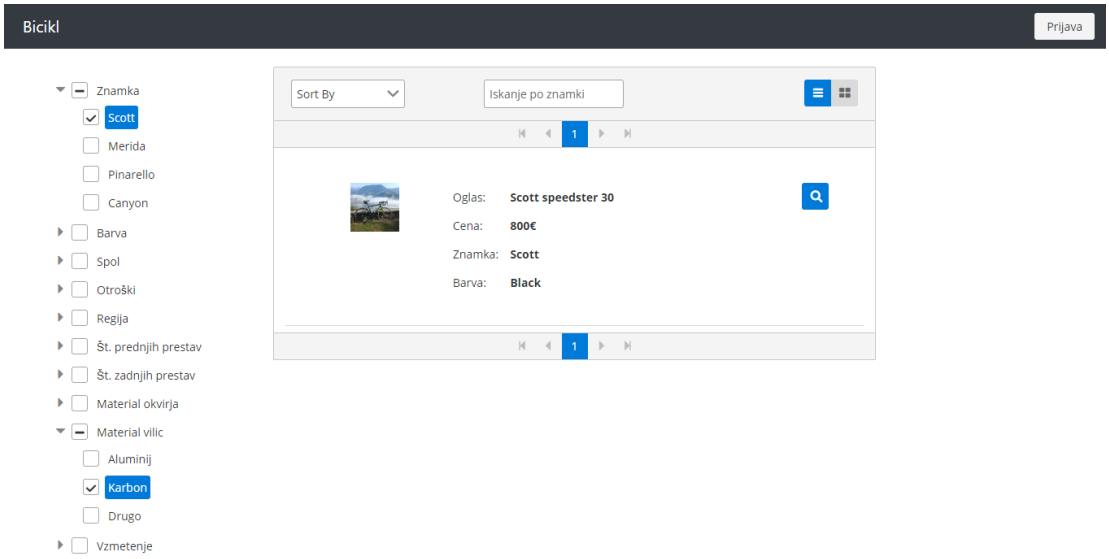


Slika 14: Prvi pogled pri uporabi aplikacije, izbira kategorije izdelka.



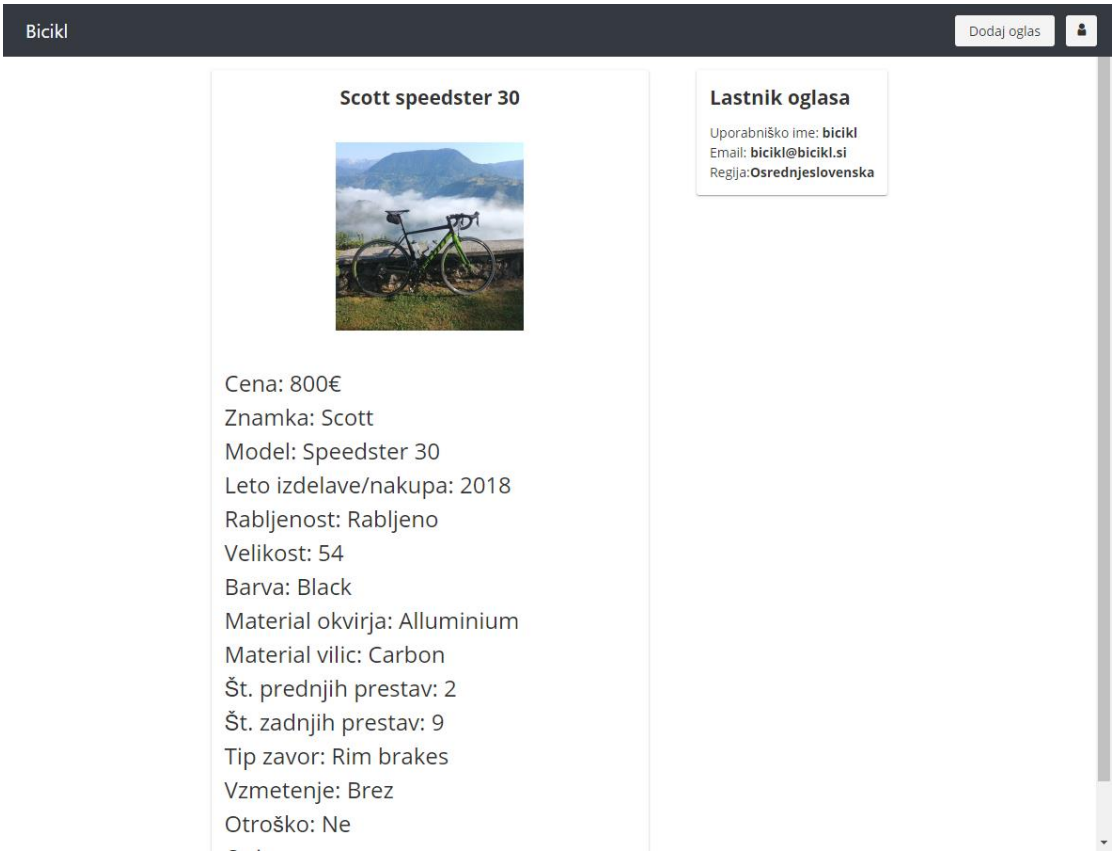
Slika 15: Drugi pogled pri uporabi aplikacije, izbira specifične kategorije izdelka.

Naslednja komponenta je namenjena najbolj podrobnemu iskanju izdelkov in je predstavljena na sliki 16. Na levi strani komponente so prikazani filtri, s katerimi uporabnik določi željene parametre iskanja izdelka. Na desni strani komponente pa se dinamično prikazujejo izdelki, glede na uporabnikovo izbiro parametrov.



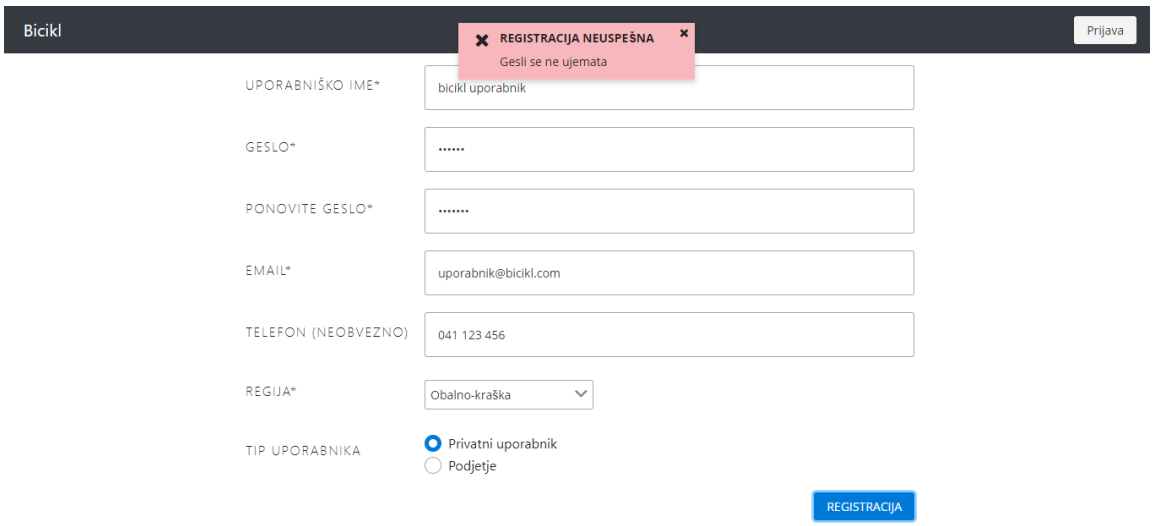
Slika 16: Komponenta za podrobno izbiranje oglasov.

Zadnja komponenta z namenom prikazovanja oglasov je prikazana na sliki 17. Na levi strani so predstavljeni vsi podatki izdelka, vsebovanega v oglasu, na desni strani pa so predstavljeni podatki uporabnika, ki je oglas naložil. Komponenta se prikaže, ko uporabnik izbere enega izmed oglasov, prikazanem v zgoraj opisani komponenti.



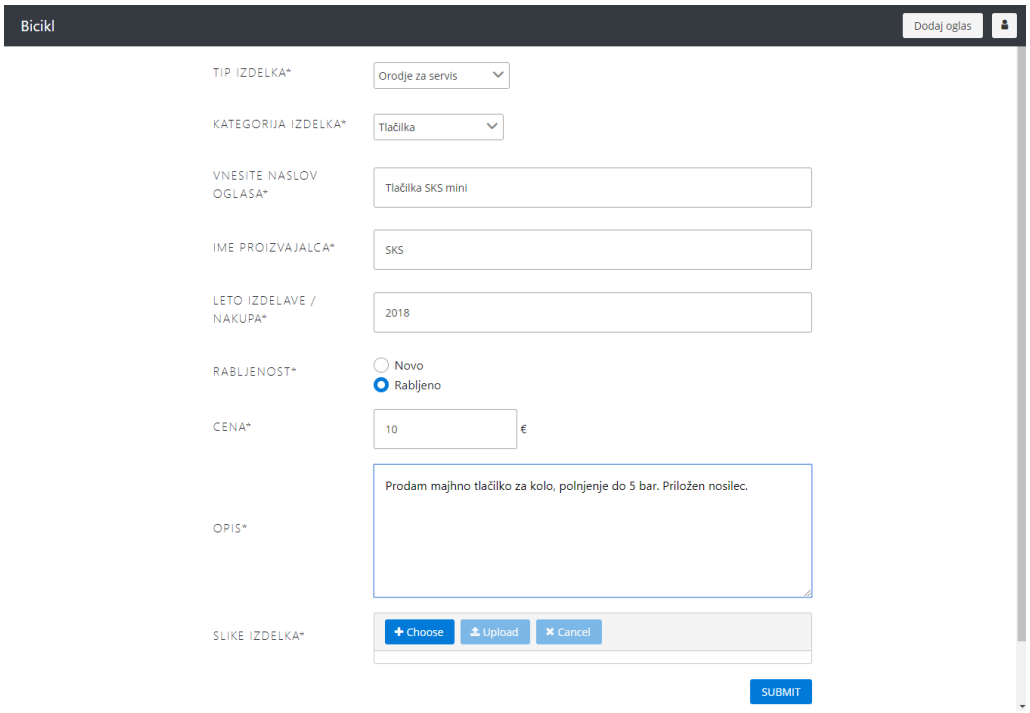
Slika 17: Komponenta s podrobnimi podatki izbranega oglasa.

Komponenta, namenjena registraciji uporabnika, je prikazana na spodnji sliki. Za uspešno registracijo mora uporabnik vnesti vsa polja, ki so označena z zvezdico. Ob pritisku na gumb »registracija«, se preveri vsebina vnosov uporabnika. V primeru, da uporabnik vnese neustrezne podatke, se pojavi obvestilo, ki uporabnika opozori. Na spodnji sliki je prikazano obvestilo, ki se pojavi ob neuspešni ponovitvi gesla.



Slika 18: Komponenta, namenjena registraciji uporabnika.

Naslednja izdelana komponenta je namenjena oddajanju oglasa. Pogoj za prikaz komponente je prijava uporabnika. Uporabnik mora vnesti vse podatke o izdelku, ki jih aplikacija zahteva in so označeni z zvezdico.



Slika 19: Komponenta za nalaganje oglasov.

4.3 Implementacija komunikacije spletne aplikacije s podatkovno bazo

Komunikacija spletne aplikacije s podatkovno bazo je bila narejena s pomočjo ogrodja CXF in programskega vmesnika JPA. Programski vmesnik JPA je bil uporabljen za lažjo pretvorbo podatkov v obliko, ki je sprejemljiva za podatkovno bazo.

4.3.1 Pretvorba podatkov

Spletna aplikacija je narejena tako, da vsa komunikacija poteka z uporabo JSON objektov, ki vsebujejo vse potrebne podatke za pravilno delovanje. Programski vmesnik JPA je namenjen transformaciji tabel iz SQL podatkovne baze v razrede v programskem jeziku Java. Z uporabo knjižnice Jackson, se lahko objekti JSON, ki so poslani prek spletne aplikacije, pretvorijo v objekte razredov, kreiranih z vmesnikom JPA. Tako so vsi podatki, ki jih pošlje spletna aplikacija, pretvorjeni v obliko, ki jo podatkovna baza lahko sprejme.

4.3.2 Izdelava spletnih storitev

Za vsako izmed entitet v spletnem IS, je bila narejena zbirka spletnih storitev, ki omogočajo dodajanje, spreminjanje, prejemanje in brisanje zapisov entitete v podatkovni bazi. V nadaljevanju je opisana spletna storitev za pridobitev podatkov. Storitev je opisana zgolj za entiteto »Uporabnik«, saj so vse ostale spletne storitve narejene po enakem principu.

Ob zagonu aplikacije, ki nudi spletne storitve na strežniku, se vzpostavijo vse končne poti, ki jih spletna aplikacija lahko uporablja. Za primer spletnih storitev, povezanih z entiteto »Uporabnik«, se vzpostavi pot »http://localhost:8080/BiciklService/services/bicikl/users/«. Spletna aplikacija lahko pošlje zahtevo GET, PATCH, POST ali DELETE. Za vsako od teh zahtev je na vzpostavljeni končni poti storitve definirana metoda, ki te zahteve obdela.

Na spodnji sliki je prikazana metoda, ki obdela zahtevo GET. Metoda preveri, ali v podatkovni bazi obstaja uporabnik z določenim e-mail naslovom ali uporabniškim imenom. V klicu REST spletne storitve sta dodana parametra »username« in »email«, ki ju metoda uporabi za poizvedbo v podatkovni bazi.

```
@GET
@Path("/{username}/{email}")
public Response getUserByEmail(
        @Context MessageContext mc,
        @PathParam("username") String username,
        @PathParam("email") String email) {
    EntityManager em = JpaEm.getInstance().createEntityManager();
    try {
        List<User> result = em.createNamedQuery("User.findByUsernameEmail", User.class)
                              .setParameter("username", username)
                              .setParameter("email", email).getResultList();
        if(result.size() != 0) {
            return Response.status(Status.OK).entity(result).build();
        }
        return Response.status(Status.NO_CONTENT).entity(result).build();
    } catch(Exception e) {
        e.printStackTrace();
        return Response.status(Status.INTERNAL_SERVER_ERROR).entity(e.getMessage()).build();
    }
}
```

Slika 20: Obdelava spletne storitve GET.

5 ZAKLJUČEK

Pri razvijanju spletnih informacijskih sistemov poznamo več različnih načinov razvoja, za katere se izvajalci sistemov odločijo na podlagi obsežnosti določenega projekta. V diplomski nalogi so bili predstavljeni klasični modeli razvijanja, ki so namenjeni projektom večjih in manjših obsežnosti. Pri razvoju spletnega IS za nakup in prodajo kolesarske opreme, je bil uporabljen razvoj z uporabo prototipa, saj je obsežnost projekta majhna, tehnologije, ki so bile uporabljene v razvoju, pa dovoljujejo enostavno dodajanje sprememb, v kolikor se zahteve sistema v prihodnosti spremenijo.

Predstavljeni so bili tudi standardi pri oblikovanju uporabniških vmesnikov. Pri načrtovanju uporabniškega vmesnika izdelane spletne aplikacije smo te standarde uporabili, aplikacija pa ima posledično enostaven in hitro odziven uporabniški vmesnik.

Razviti spletni informacijski sistem za nakup in prodajo kolesarske opreme ima v Sloveniji potencial za množično uporabo, saj takšen sistem še ni na voljo. V prihodnje bi bilo potrebno še nekoliko dodelati spletno aplikacijo, da bi bila le-ta uporabnikom še privlačnejša.

6 LITERATURA IN VIRI

[1] Ad hoc, Wikipedia [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Ad_hoc [Dostopano 23.7.2019].

[2] Best database design software of 2019 [Online]. Dosegljivo: https://www.techradar.com/best/best-database-design-software [Dostopano 31.7.2019]

[3] Best Practices for Designing a Pragmatic RESTful API [Online]. Dosegljivo: https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api#restful [Dostopano 1.8.2019]

[4] Comparison with Other Frameworks, Vue.js [Online]. Dosegljivo: https://vuejs.org/v2/guide/comparison.html [Dostopano 1.8.2019]

[5] Diagram stanj Bankomat [Online]. Dosegljivo: http://sasa.musiclab.si/eri1/RACUNALNISTVO/uml/slike/bankomatStanja.gif [Dostopano 31.7.2019]

[6] ECF Cycling barometer [Online]. Dosegljivo: https://ecf.com/resources/cycling-facts-and-figures/ecf-cycling-barometer [Dostopano 1.8.2019]

[7] How important is UI/UX in the modern digital world? [Online]. Dosegljivo: https://cabforward.com/how-important-is-uiux-in-the-modern-digital-world/ [Dostopano 23.7.2019]

[8] Information systems, Wikipedia [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Information_system [Dostopano 17.7.2019].

[9] React vs. Angular: The Complete Comparison [Online]. Dosegljivo: https://programmingwithmosh.com/react/react-vs-angular/ [Dostopano 1.8.2019]

[10] React vs. Vue (vs. Angular) [Online]. Dosegljivo: https://medium.com/fundbox-engineering/react-vs-vue-vs-angular-163f1ae7be56 [Dostopano 1.8.2019]

[11] Representational state transfer, Wikipedia [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Representational_state_transfer [Dostopano 1.8.2019]

[12] Service-Oriented Architecture [Online]. Dosegljivo: https://www.geeksforgeeks.org/service-oriented-architecture/ [Dostopano 1.8.2019]

[13] Software development process, Wikipedia [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Software_development_process [Dostopano: 23.7.2019]

[14] Systems development life cycle, Wikipedia [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Systems_development_life_cycle#System_investigation [Dostopano 28.7.2019]

[15] The Benefits of Web-based Applications (2015) [Online]. Dosegljivo: https://coresolutions.ca/blogs/core-web/the-benefits-of-web-based-applications [Dostopano 22.7.2019].

[16] The Encyclopedia of Human-Computer Interaction, 2nd Ed. [Online]. Dosegljivo: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro [Dostopano: 23.7.2019]

[17] The Pros and Cons of Open Source Software [Online]. Dosegljivo: https://medium.com/4thought-studios/the-pros-and-cons-of-open-source-software-d498304f2a95 [Dostopano 31.7.2019]

[18] The Seven Phases of the System-Development Life Cycle [Online]. Dosegljivo: https://www.innovativearchitects.com/KnowledgeCenter/basic-IT-systems/system-development-life-cycle.aspx [Dostopano 23.7.2019].

[19] Top 10 Web Development Frameworks in 2019 [Online]. Dosegljivo: https://hackr.io/blog/top-10-web-development-frameworks-in-2019 [Dostopano 1.8.2019]

[20] User experience, Wikipedia [Online]. Dosegljivo: https://en.wikipedia.org/wiki/User_experience [Dostopano 24.7.2019]

[21] User Interface, Wikipedia [Online]. Dosegljivo: https://en.wikipedia.org/wiki/User_interface [Dostopano: 23.7.2019]

[22] User Interface Design Basics [Online]. Dosegljivo: http://www.usability.gov/what-and-why/user-interface-design.html [Dostopano 30.7.2019]

[23] Web Based Information Systems (2017) [Online]. Dosegljivo: https://www.ukessays.com/essays/information-systems/web-based-information-systems.php [Dostopano 21.7.2019].

[24] What is build and fix model or ad hoc model? [Online]. Dosegljivo: http://ecomputernotes.com/software-engineering/build-and-fix-model [Dostopano 28.7.2019]

[25] What is Entity Relationship Diagram (ERD)? [Online]. Dosegljivo: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/ [Dostopano 30.7.2019]

[26] Why should you learn Angular in 2019? [Online]. Dosegljivo: https://hackr.io/blog/why-should-you-learn-angular-in-2019 [Dostopano 1.8.2019]

[27] D. Norman (2013), The Design of Everyday Things, New York, Basic Books

[28] R. K. Rainer, B. Prince, Introduction to Information Systems – Sixth edition, John Wiley & Sons Inc., New Jersey, 2015

[29] P. Rogelj (2016) Programsko Inženirstvo: Testiranje [PowerPoint]. Pridobljeno s https://e.famnit.upr.si/pluginfile.php/53194/mod_resource/content/0/PI%20-%2007%20-%20Testiranje.pdf

[30] P. Rogelj (2016) Programsko Inženirstvo: Vzdrževanje [PowerPoint]. Pridobljeno s https://e.famnit.upr.si/pluginfile.php/53195/mod_resource/content/0/PI%20-%2008%20-%20Vzdr%C5%BEevanje.pdf

[31] P. Rogelj (2017) Sistemi 3: Diagram podatkovnega toka (DFD) [PowerPoint]. Pridobljeno s https://e.famnit.upr.si/pluginfile.php/56652/mod_resource/content/0/Sis3%20-%2008%20-%20Diagram%20podatkovnega%20toka_s.pdf

[32] P. Rogelj (2017), Sistemi 3: Informacijski sistemi v poslovnem procesu. [PowerPoint]. Pridobljeno s https://e.famnit.upr.si/pluginfile.php/55415/mod_resource/content/0/Sis3%20-%2002%20-%20Osnove%20IS%20v%20poslovnem%20procesu.pdf .

[33] P. Rogelj (2017) Sistemi 3: Podatkovni modeli [PowerPoint]. Pridobljeno s https://e.famnit.upr.si/pluginfile.php/56296/mod_resource/content/0/Sis3%20-%2007%20-%20Podatkovni%20modeli_short.pdf

[34] P. Rogelj (2017) Sistemi 3: Razvojni modeli IS. [PowerPoint]. Pridobljeno s https://e.famnit.upr.si/pluginfile.php/55811/mod_resource/content/0/Sis3%20-%2004%20-%20Razvojni%20modeli%20IS.pdf

[35] P. Rogelj (2017) Sistemi 3: Strateško načrtovanje in razvoj. [PowerPoint]. Pridobljeno s https://e.famnit.upr.si/pluginfile.php/55810/mod_resource/content/0/Sis3%20-%2003%20-%20Na%C4%8Drtovanje%20in%20razvoj%20IS.pdf

[36] P. Rogelj (2017) Sistemi 3: XML tehnologije in SOA [PowerPoint]. Pridobljeno s https://e.famnit.upr.si/pluginfile.php/56651/mod_resource/content/0/Sis3%20-%2009%20-%20XML%20in%20SOA_short.pdf

[37] P. Rogelj (2017) Sistemi 3: Informacijski sistemi – Zbiranje informacij. [PowerPoint]. Pridobljeno s https://e.famnit.upr.si/pluginfile.php/56288/mod_resource/content/0/Sis3%20-%2006%20-%20Zbiranje%20informacij_short.pdf

[38] I. Savnik (2016) Načrtovanje podatkovnih baz. [PowerPoint]. Pridobljeno s http://osebje.famnit.upr.si/~savnik/predmeti/NPB/npb4-nacrt-pb.pdf

[39] I. Savnik (2016) Normalizacija. [PowerPoint]. Pridobljeno s http://osebje.famnit.upr.si/~savnik/predmeti/NPB/npb7-normalizacija.pdf

[40] I. Savnik (2016) Physical Optimization. [PowerPoint]. Pridobljeno s http://osebje.famnit.upr.si/~savnik/predmeti/NPB/npb8-physical-opt.pdf

[41] I. Savnik (2016) Prevod konceptualnega modela v SQL. [PowerPoint]. Pridobljeno s http://osebje.famnit.upr.si/~savnik/predmeti/NPB/npb6-prevod-sql.pdf

[42] I. Savnik (2016) Uvod v načrtovanje podatkovnih baz. [PowerPoint]. Pridobljeno s http://osebje.famnit.upr.si/~savnik/predmeti/NPB/npb1-uvod.pdf

[43] R. Vidgen, Constructing a web information system development methodology. Information Systems Journal (2002), 12: 247-261.

[44] J. L. Whitten, L. D. Bentley, The context of systems analysis and design methods, v: Paul Ducham (ur.) System analysis and design methods - 7th edition, The McGraw-Hill Companies, New York, 2007.

[45] V. Zwass, Information system (2017) [Online]. Dosegljivo: https://www.britannica.com/topic/information-system [Dostopano 17.7.2019].