



☰

 m / S...

🔍

Type / to search



⌵

+

⌵

🕒

🔗

📁

👤

<>

Code

🕒

Issues

🔗

Pull requests

🎬

Actions

📁

Projects

📖

Wiki

🛡️

Security

11

Home

Edit

New page

Mona Mihoković edited this page 1 hour ago · [6 revisions](#)

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Razmijeni Ulaznice

Grupa: TG07.3

Tim: Swappet

- Ivan Vjekoslav Rođak - voditelj
- Goran Torbica
- Mona Mihoković
- Patrick Mraz
- Maja Blažok
- Dominik Mandić
- Paško Zekić

Nastavnik: Vlado Sruk

Pristup aplikaciji

Našoj aplikaciji možete pristupiti [ovdje](#).

▼ Pages 13

Find a page...

▼ Home

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Razmijeni Ulaznice

Grupa: TG07.3

Tim: Swappet

Nastavnik: Vlado Sruk

Pristup aplikaciji

▶ 1. Opis projektnog zadatka

▶ 2. Analiza zahtjeva

▶ 3. Specifikacija zahtjeva sustava

▶ 4. Arhitektura i dizajn sustava

▶ 5. Arhitektura komponenata i razmj...

▶ 6. Ispitivanje programskog rješenja

▶ 7. Tehnologije za implementaciju a...

▶ 8. Upute za puštanje u pogon

▶ 9. Zaključak i budući rad

Više informacija o samom projektu možete naći u našem [Wiki-ju](#), te izvorni kod na [glavnoj grani projekta](#).

Swappet



- [A. Popis literature](#)
- [B. Dnevnik promjena dokumentacije](#)
- [C. Prikaz aktivnosti grupe](#)

[+ Add a custom sidebar](#)

Clone this wiki locally

<https://github.com/monamihokovic>

m / S...

Q

Type / to search

+

<>

CodeIssuesPull requestsActionsProjectsWikiSecurity

11

1. Opis projektnog zadatka

Edit

New page

pz55296 edited this page on Nov 10, 2024 · [9 revisions](#)

Web aplikacija za kupnju i zamjenu ulaznica

1. Cilj projektnog zadatka

Cilj projekta *Swappet* je razviti inovativnu platformu za kupnju, prodaju i zamjenu ulaznica za događaje, kao što su koncerti, sportska natjecanja, predstave i slična okupljanja. Aplikacija je osmišljena kao rješenje problema koji se javlja kad korisnici unaprijed kupe ulaznice, ali zbog nepredviđenih okolnosti ne mogu prisustvovati događaju. Swappet korisnicima omogućava legalnu, brzu i sigurnu razmjenu ili prodaju ulaznica, smanjujući potrebu za korištenjem nesigurnih kanala prodaje i izloženost prevarama.

2. Problematika zadatka

U današnje vrijeme sve je lakše kupiti ulaznice online, ali korisnicima je često problem preprodati ili zamijeniti ulaznice ako dođe do promjena planova. Postoje brojni rizici prilikom prodaje putem društvenih mreža ili nereguliranih platformi, gdje se korisnici suočavaju s prevarama, lažnim profilima, manipulacijom cijenama te visokim naknadama za prodaju. Nadalje, aktualne platforme rijetko omogućuju izravnu zamjenu ulaznica između korisnika, zbog čega korisnici gube mogućnost za jednostavan pronalazak odgovarajuće zamjene.

▼ Pages13

Find a page...

▶ Home

▼ 1. Opis projektnog zadatka

Web aplikacija za kupnju i zamjenu ulaznica

1. Cilj projektnog zadatka

2. Problematika zadatka

3. Potencijalna korist projekta

4. Postojeća slična rješenja i usporedba

5. Skup korisnika zainteresiranih za rješenje

6. Mogućnost prilagodbe rješenja

7. Opseg projektnog zadatka

▶ 2. Analiza zahtjeva

▶ 3. Specifikacija zahtjeva sustava

▶ 4. Arhitektura i dizajn sustava

▶ 5. Arhitektura komponenata i razmj...

▶ 6. Ispitivanje programskog rješenja

▶ 7. Tehnologije za implementaciju a...

▶ 8. Upute za puštanje u pogon

Swappet nastoji riješiti ove izazove putem kontrolirane, sigurne i interaktivne platforme koja nudi:

- **sigurno okruženje za razmjenu ulaznica**
- **funkciju jednokratne zamjene** u kojoj obje strane odmah prihvaćaju transakciju
- **automatsko uklanjanje nevažećih oglasa** po isteku datuma događaja
- **povezivanje sa sustavima prognoze vremena i katalogima izvođača** kako bi korisnici imali dodatne informacije prilikom kupovine ili zamjene.

3. Potencijalna korist projekta

Swappet nudi značajnu korist za korisnike i širu zajednicu:

- **Sigurna kupnja i zamjena:** Uklanjanjem posrednika i nesigurnih oglašivačkih kanala, korisnici izravno komuniciraju i pregovaraju sa zainteresiranim stranama, što smanjuje mogućnost prevare.
- **Prilagodljivost zahtjevima korisnika:** Korisnici mogu prilagoditi svoje oglase prema vrsti ulaznice ili željenom događaju u slučaju zamjene. Uz dodatne informacije o događaju i vremenskim uvjetima, korisnici donose informiranije odluke.
- **Edukacija i podrška lokalnoj zajednici:** Platforma promovira sigurno i odgovorno ponašanje na tržištu ulaznica, educirajući korisnike o prednostima legalnog poslovanja.

4. Postojeća slična rješenja i usporedba

Na tržištu trenutno postoje razne aplikacije i web stranice za kupnju i preprodaju ulaznica, među kojima su popularne:

- **StubHub:** Globalna platforma za preprodaju

- [9. Zaključak i budući rad](#)
- [A. Popis literature](#)
- [B. Dnevnik promjena dokumentacije](#)
- [C. Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

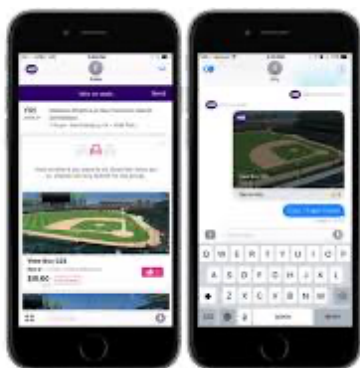
<https://github.com/monamihokovic>



ulaznica. Njihov sustav omogućava kupovinu, ali ne uključuje mogućnost zamjene ulaznica jedan-na-jedan. Također, StubHub ima visoke naknade.

- **TicketSwap:** Sigurna platforma za prodaju i kupovinu ulaznica koja uključuje autentifikaciju ulaznica kako bi se smanjio rizik prevare. No, TicketSwap se fokusira na preprodaju ulaznica, a ne na zamjenu.
- **Viagogo:** Popularna platforma za preprodaju ulaznica, ali ima mnogo pritužbi na visoke naknade i nedovoljnu sigurnost korisnika.

Glavna prednost *Swappeta* nad postojećim rješenjima je funkcionalnost zamjene ulaznica, čime se korisnicima nudi alternativa klasičnoj kupovini i prodaji. Pored toga, Swappet integrira obavijesti i informacije o vremenskim uvjetima, dok mnoge konkurentske platforme nemaju takve značajke.



Prikaz aplikacije StubHub

5. Skup korisnika zainteresiranih za rješenje

Korisnici koji bi bili zainteresirani za ovu aplikaciju uključuju:

- **Ljubitelji koncerata i sportskih događanja:** Osobe koje redovno kupuju ulaznice za događanja i povremeno ih trebaju zamijeniti ili prodati.
- **Putnike:** Osobe koje su kupile ulaznice, ali su

ih prisiljene prodati zbog promjena u rasporedu putovanja.

- **Zajednice i grupe:** Grupe prijatelja ili članovi obitelji koji žele osigurati zamjenu ulaznica za događaje u slučaju iznenadnih promjena.
- **Mladi i studenti:** Korisnici koji se često susreću s promjenama u planovima zbog financijskih i drugih razloga te traže sigurnu platformu za zamjenu.

6. Mogućnost prilagodbe rješenja

Swappet je razvijen s fleksibilnošću na umu, što znači da je prilagodljiv različitim potrebama i zahtjevima korisnika i tržišta:

- **Dodavanje podrške za različite vrste događanja:** U budućnosti se može proširiti na ulaznice za festivale, kazališne predstave, i konferencije.
- **Integracija s dodatnim vanjskim servisima:** Na primjer, integracija s alatima za digitalne karte ili sustavima provjere autentičnosti ulaznica radi veće sigurnosti.
- **Skalabilnost na međunarodnoj razini:** Aplikacija može biti prilagođena različitim tržištima uz lokalizaciju jezika i opcija plaćanja.
- **Mobilna aplikacija:** Iako je aplikacija responzivna, kreiranje posebne mobilne aplikacije dodatno bi olakšalo upotrebu platforme i povećalo zadovoljstvo korisnika.

7. Opseg projektnog zadatka

Opseg ovog projekta uključuje:

- **Osnovnu funkcionalnost za kupnju, prodaju i zamjenu ulaznica:** Osiguranje glavnih funkcija koje omogućavaju jednostavno postavljanje oglasa, pregled i filtriranje dostupnih ulaznica

te upravljanje interakcijama između korisnika.


- **Korisničke profile s mogućnošću prilagodbe:** Razvoj korisničkog sučelja koje omogućava kreiranje i prilagodbu profila, uključujući opcije za postavke privatnosti i praćenje preferencija korisnika.
- **Sigurnosne značajke:** Implementacija osnovnih mehanizama provjere identiteta kako bi se smanjio rizik od zlouporabe.
- **Modul za administraciju i izvještavanje:** Modul koji omogućava administratorima pregled aktivnosti na platformi, rješavanje potencijalnih sporova, generiranje izvještaja i praćenje učestalosti korištenja platforme.
- **Sustav notifikacija:** Korisnici primaju notifikacije putem aplikacije i emaila o važnim promjenama, kao što su odgovori na oglase i prilike za zamjenu.

Swappet projekt ima cilj omogućiti korisnicima sigurno iskustvo u kupnji, prodaji i zamjeni ulaznica uz minimalne rizike te usmjeriti tržište ulaznica prema transparentnijem modelu poslovanja. Ova platforma predstavlja rješenje za rastuće potrebe potrošača, dok istovremeno promovira sigurnu razmjenu ulaznica među korisnicima.

Swappet




☰

 m / S...


🔍


Type / to search




+

🕒










<> Code


🔗 Issues

 Pull requests

🎬 Actions

 Projects

 Wiki

 Security 11

2. Analiza zahtjeva

Edit

New page

Mona Mihoković edited this page yesterday · [9 revisions](#)

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	I
F-001	Sustav omogućuje korisnicima prijavu putem Google računa.	Visok	Zah dior
F-002	Sustav omogućuje pregled liste dostupnih oglasa za različite događaje.	Visok	Zah dior
F-003	Sustav omogućuje korisnicima podnošenje oglasa za prodaju ili zamjenu	Visok	Zah dior

▼ Pages 13

Find a page...

▶ Home

▶ 1. Opis projektnog zadatka

▼ 2. Analiza zahtjeva

Funkcionalni zahtjevi

Nefunkcionalni zahtjevi

Zahtjevi za performanse

Zahtjevi za sigurnost

Zahtjevi za dostupnost

Zahtjevi za održavanje

Zahtjevi za korisničko iskustvo (UX)

Dionici

Dionik 1: Korisnici

Dionik 2: Administrator

Dionik 3: Razvojni tim

Dionik 4: Naručitelj

Aktori i njihovi funkcionalni zahtjevi

A-1: Kupac (inicijator)

A-2: Prodavatelj (inicijator i sudionik)

A-3: Administrator (inicijator i sudionik)

A-4: Razvojni tim (sudionik)

A-5: Naručitelj (inicijator)

	ulaznica.		
F-004	Sustav omogućuje korisnicima kupovanje ulaznica.	Visok	Zah dior
F-005	Sustav omogućuje korisnicima zamjenu ulaznica.	Visok	Zah dior
F-006	Sustav omogućuje registriranim korisnicima pregled prošlih transakcija.	Srednji	Zah dior
F-007	Sustav šalje e-mail obavijest korisniku o kupljenoj ulaznici.	Visok	Pov info kori
	Sustav šalje e-mail obavijest		Pov

▸ 3. Specifikacija zahtjeva sustava
▸ 4. Arhitektura i dizajn sustava
▸ 5. Arhitektura komponenata i razmj...
▸ 6. Ispitivanje programskog rješenja
▸ 7. Tehnologije za implementaciju a...
▸ 8. Upute za puštanje u pogon
▸ 9. Zaključak i budući rad
▸ A. Popis literature
▸ B. Dnevnik promjena dokumentacije
▸ C. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/monamihokovic>



F-008	korisniku o ponuđenoj zamjeni.	Visok	info kori
F-009	Sustav omogućuje automatsko uklanjanje oglasa nakon isteka datuma događaja.	Visok	Zah dior
F-010	Sustav omogućuje korisnicima filtere za pretraživanje oglasa prema različitim kriterijima.	Srednji	Zah dior
F-011	Sustav omogućuje korisnicima „lajkanje“ ili „dislajkanje“ oglasa.	Nizak	Zah dior
F-012	Sustav omogućuje administratorima pregled prijava o zlouporabi korisničkih računa	Visok	Zah dior
	Sustav omogućuje		

F-013	administatorima deaktivaciju korisničkih računa	Visok	Zaht dior
F-014	Sustav omogućuje administratorima generiranje izvještaja na temelju povijesti transakcija i prosječne cijene prodane ulaznice	Srednji	Zaht dior
F-015	Sustav omogućava korisniku pregled informacija o vremenskoj prognozi za vrijeme održavanja događaja iz oglasa	Srednji	Pos sus

Nefunkcionalni zahtjevi

Zahtjevi za performanse

ID zahtjeva	Opis	Prioritet
NF-1.1	Sustav treba omogućiti brz odziv na akcije korisnika, s vremenom odziva manjim od 3 sekunde.	Srednji
	Sustav treba	

NF-1.2	omogućiti istovremeno korištenje od strane najmanje 500 korisnika bez smanjenja performansi.	Srednji
NF-1.3	Sustav mora biti optimiziran za rad na mobilnim uređajima, tabletima i računalima.	Visok

Zahtjevi za sigurnost

ID zahtjeva	Opis	Prioritet
NF-2.1	Sustav mora omogućiti autentifikaciju korisnika putem Google računa.	Srednji

Zahtjevi za dostupnost

ID zahtjeva	Opis	Prioritet
NF-3.1	Sustav treba biti dostupan 98% vremena, s maksimalnim vremenom prekida od 10 sati mjesečno.	Srednji
NF-3.2	Web stranica mora omogućiti korisnicima pristup s bilo kojeg uređaja uz kompatibilnost s	Visok

	najnovijim preglednicima.	
--	---------------------------	--

Zahtjevi za održavanje

ID zahtjeva	Opis	Prioritet
NF-4.1	Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje i proširivanje novih funkcionalnosti.	Srednji
NF-4.2	Sustav treba imati dostatnu dokumentaciju za korisnike i administratore.	Srednji
NF-4.3	Kôd sustava treba biti dokumentiran prema industrijskim standardima, uključujući komentare u kodu i uputama.	Niski

Zahtjevi za korisničko iskustvo (UX)

ID zahtjeva	Opis	Prioritet
NF-5.1	Sustav treba imati intuitivno sučelje koje omogućava jednostavno kreiranje i upravljanje korisničkim računima putem Google prijave.	Visok
	Sučelje treba biti	

NF-5.2	responzivno i optimizirano za mobilne uređaje, omogućujući korisnicima jednako kvalitetno iskustvo na svim platformama.	Visok
NF-5.3	Navigacija kroz web stranicu mora biti jednostavna i jasna, s minimalnim brojem koraka za obavljanje osnovnih funkcija.	Srednji

Dionici

Dionik 1: Korisnici

Korisnici su ključni dionici koji aktivno koriste sustav, bilo da su registrirani ili neregistrirani, i obavljaju razne aktivnosti poput prijave, pregleda oglasa, objave oglasa, komunikacije s drugim korisnicima, te obavljaju transakcije. Korisnici mogu biti podijeljeni u nekoliko kategorija prema njihovoj aktivnosti:

- **Kupci** koji kupuju ulaznice za događanja.
- **Prodavači** koji prodaju ili razmjenjuju svoje ulaznice.
- **Korisnici zainteresirani za zamjenu** ulaznica.

Dionik 2: Administrator

Administrator ima ključnu ulogu u održavanju sustava, praćenju aktivnosti korisnika, upravljanju korisničkim računima i rješavanju nesuglasica. Administratori mogu imati pristup svim funkcijama sustava i mogu intervenirati u slučaju nepravilnosti, te generirati izvještaje o aktivnostima sustava.

Dionik 3: Razvojni tim

Razvojni tim je odgovoran za tehničku izradu sustava. Oni implementiraju funkcionalnosti, održavaju infrastrukturu, odgovorni su za sigurnost sustava te za implementaciju novih značajki. Ovaj tim osigurava da sustav radi kako je predviđeno te da je usklađen s tehničkim zahtjevima.

Dionik 4: Naručitelj

Naručitelj je osoba ili organizacija koja inicira razvoj sustava i ima interes u njegovom uspješnom implementiranju. Naručitelj definira osnovne zahtjeve, očekivanja i funkcionalnosti koje sustav mora podržavati, te ima nadzor nad projektom.

Aktori i njihovi funkcionalni zahtjevi

A-1: Kupac (inicijator)

Kupac je korisnik koji pretražuje i kupuje ulaznice za događanja. Kupac može biti inicijator jer započinje proces kupnje ulaznica ili zamjene putem sustava.

Funkcionalnosti koje Kupac može obaviti:

- **Pregledavanje dostupnih oglasa za ulaznice prema različitim filterima.**
 - Funkcionalni zahtjev: **F-002** - Sustav omogućuje pregled liste dostupnih ulaznica za različite događaje.
 - Funkcionalni zahtjev: **F-010** - Sustav omogućuje korisnicima filtere za pretraživanje oglasa prema različitim kriterijima.
- **Pokretanje transakcija za kupnju ulaznica.**
 - Funkcionalni zahtjev: **F-004** - Sustav omogućuje korisnicima kupovanje

ulaznica.

- **Slanje zahtjeva za zamjenu ulaznica.**
 - Funkcionalni zahtjev: **F-005** - Sustav omogućuje korisnicima podnošenje zahtjeva za zamjenu ulaznica, s mogućnošću prihvatanja ili odbijanja od strane druge strane.
- **Primanje e-mail obavijesti o kupljenim ulaznicama ili ponuđenim zamjenama.**
 - Funkcionalni zahtjevi: **F-007** i **F-008** - Sustav šalje e-mail obavijesti korisnicima o kupljenim ulaznicama i ponuđenim zamjenama.

A-2: Prodavatelj (inicijator i sudionik)

Prodavatelj je korisnik koji objavljuje oglase za prodaju ili zamjenu ulaznica. Prodavatelj može inicirati transakcije ili odgovarati na zahtjeve drugih korisnika.

Funkcionalnosti koje Prodavatelj može obaviti:

- **Objavljivanje oglasa za prodaju ili zamjenu ulaznica.**
 - Funkcionalni zahtjev: **F-003** - Sustav omogućuje korisnicima podnošenje oglasa za prodaju ili zamjenu ulaznica, uz detalje o događaju i ulaznicama.
- **Pregled statusa svojih oglasa i povijesti transakcija.**
 - Funkcionalni zahtjev: **F-006** - Sustav omogućuje registriranim korisnicima pregled prošlih transakcija.
- **Primanje obavijesti o interesu za njihove ulaznice.**
 - Funkcionalni zahtjev: **F-007** - Sustav šalje e-mail obavijesti korisnicima o kupljenim ulaznicama.
- **Odgovaranje na zahtjeve za zamjenu**

ulaznica.

- Funkcionalni zahtjev: **F-005** - Sustav omogućuje korisnicima podnošenje zahtjeva za zamjenu ulaznica.

A-3: Administrator (inicijator i sudionik)

Administrator upravlja sustavom, prati aktivnosti korisnika i intervenira u slučaju nepravilnosti.

Administrator ima ovlasti za upravljanje korisničkim računima i generiranje izvještaja.

Funkcionalnosti koje Administrator može obaviti:

- **Upravljanje korisničkim računima.**
 - Funkcionalni zahtjevi: **F-012** i **F-013** - Sustav omogućuje administratorima pregled prijava o zlouporabi i deaktivaciju korisničkih računa.
- **Generiranje izvještaja o aktivnostima korisnika.**
 - Funkcionalni zahtjev: **F-014** - Sustav omogućuje administratorima generiranje izvještaja na temelju povijesti transakcija i prosječne cijene ulaznica.
- **Praćenje prijava i nepravilnosti.**
 - Funkcionalni zahtjev: **F-012** - Sustav omogućuje pregled prijava o zlouporabi korisničkih računa.
- **Upravljanje oglasima.**
 - Funkcionalni zahtjev: **F-009** - Sustav omogućuje automatsko uklanjanje oglasa nakon isteka datuma događaja.

A-4: Razvojni tim (sudionik)

Razvojni tim odgovoran je za tehničku izvedbu, implementaciju i održavanje sustava. Tim kontinuirano radi na unaprjeđenju funkcionalnosti i osiguravanju performansi.

Funkcionalnosti koje Razvojni tim može obaviti:

- **Implementacija funkcionalnosti za prijavu korisnika putem Google računa.**
 - Funkcionalni zahtjev: **F-001** - Sustav omogućuje prijavu putem Google računa.
- **Održavanje performansi sustava i osiguravanje tehničke podrške.**
 - Nefunkcionalni zahtjevi: **NF-1.1** i **NF-1.2** - Sustav mora omogućiti brz odziv i podržavati istovremeno korištenje velikog broja korisnika.
- **Omogućavanje responzivnosti sustava na svim platformama.**
 - Nefunkcionalni zahtjevi: **NF-1.3** i **NF-5.2** - Sustav mora biti optimiziran za mobilne uređaje i imati intuitivno sučelje.

A-5: Naručitelj (inicijator)

Naručitelj postavlja osnovne zahtjeve za sustav i nadgleda razvoj kako bi sustav bio u skladu s poslovnim ciljevima.

Funkcionalnosti koje Naručitelj može obaviti:

- **Definiranje osnovnih zahtjeva za sustav.**
 - Funkcionalni zahtjevi: **F-002, F-003, F-004, F-005** - Naručitelj definira funkcionalnosti koje omogućuju korisnicima pregled, objavu, kupnju i zamjenu ulaznica.
- **Praćenje napretka razvoja sustava.**
 - Nefunkcionalni zahtjev: **NF-4.1** - Sustav treba biti jednostavan za održavanje i proširivanje novih funkcionalnosti.

- **Provjera usklađenosti sustava s funkcionalnim i tehničkim zahtjevima.**
 - Nefunkcionalni zahtjevi: **NF-1.1** do **NF-5.3**
 - Sustav mora zadovoljavati definirane zahtjeve za performanse, sigurnost, dostupnost i korisničko iskustvo.

Swappet



m / S...

Q

Type / to search

+

<>

CodeIssuesPull requestsActionsProjectsWikiSecurity

11

3. Specifikacija zahtjeva sustava

Edit

New page

Mona Mihoković edited this page 4 hours ago · [60 revisions](#)

UC1 - Registriraj i prijavi korisnika

Element	Opis
Namjena	Omogućavanje korisnicima pristup sustavu putem prijave ili registracije
Opis	Korisnici se mogu registrirati i prijaviti u aplikaciju koristeći svoj Google račun, čime se automatizira unos podataka.
Glavni aktor	Korisnik
Preduvjeti	Korisnik ima aktivni Google račun.
Opis osnovnog tijeka	<div>1. Korisnik na početnoj stranici odabire opciju "Registracija ili prijava s Google" u aplikaciji.</div> <div>2. Aplikacija preusmjerava korisnika na Google stranicu za autentifikaciju.</div> <div>3. Korisnik unosi potrebne podatke te odabire opciju za podnošenje zahtjeva.</div> <div>4. Korisnik odobrava aplikaciji pristup podacima s Google računa.</div> <div>5. Aplikacija provjerava postoji</div>

▼ Pages13

Find a page...

▶ Home

▶ 1. Opis projektnog zadatka

▶ 2. Analiza zahtjeva

▼ 3. Specifikacija zahtjeva sustava

UC1 - Registriraj i prijavi korisnika

UC2 - Pregledaj oglase

UC3 - Objavi oglase

UC4 - Zamijeni ulaznice

UC5 - Prilagodi prikaza oglasa temeljem korisničkih preferencija

UC6 - Pregledaj i upravljaj transakcijama

UC7 - Automatski ukloni i ažuriraj oglase

UC8 - Pregledaj korisničke račune

UC9 - Pregledaj prijavu o korisničkoj zlouporabi

UC10 - Deaktiviraj korisnički račun

UC11 - Generiraj izvještaj o aktivnostima i obavijesti korisnika

UC12 - Poveži se s vanjskim servisima za prikaz informacija o

https://github.com/monamihokovic/Swappet/wiki/3.-Specifikacija-zahtjeva-sustava

Page 1 of 20

	<p>li korisnik u bazi. Ako ne postoji, vrši se registracija, ako postoji, vrši se prijava</p> <p>6. Ako je registracija, aplikacija pohranjuje korisničke podatke i završava registraciju.</p> <p>7. Ako je prijava, aplikacija autentificira korisnika i omogućuje pristup aplikaciji.</p>
Opis mogućih odstupanja	<p>5.a)Korisnik odbija pristup</p> <p>1. Google obrađuje pogrešku i ispisuje poruku da je korisnik odbio pristup</p> <p>2.Proces se nastavlja u koraku 1 osnovnog tijeka</p> <p>5.b)Tehnička greška tijekom registracije/prijave</p> <p>1. Google obrađuje pogrešku i ispisuje poruku da je došlo do tehničke greške</p> <p>2.Proces se nastavlja u koraku 2 osnovnog tijeka</p>

vremenskoj prognozi

UC13 - Kupi ulaznice

1. Dijagram obrazaca uporabe za ključne funkcionalnosti
2. Dijagram obrazaca uporabe za korisničke uloge
3. Dijagram obrazaca uporabe za osnovne poslovne procese
4. Dijagram obrazaca uporabe za kritične sustave i integracije

Dijagrami obrazaca uporabe

Sekvencijski dijagrami

Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

- [4. Arhitektura i dizajn sustava](#)
- [5. Arhitektura komponenata i razmj...](#)
- [6. Ispitivanje programskog rješenja](#)
- [7. Tehnologije za implementaciju a...](#)
- [8. Upute za puštanje u pogon](#)
- [9. Zaključak i budući rad](#)
- [A. Popis literature](#)
- [B. Dnevnik promjena dokumentacije](#)
- [C. Prikaz aktivnosti grupe](#)

 Add a custom sidebar

Clone this wiki locally

<https://github.com/monamihokovic/Swappet/wiki/3.-Specifikacija-zahtjeva-sustava>


UC2 - Pregledaj oglase

Element	Opis
Namjena	Omogućavanje korisnicima pregled dostupnih oglasa na sustavu
Opis	Korisnik pregledava dostupne oglase na platformi
Glavni aktor	Korisnik
Preduvjeti	-
Opis	1. Korisnik na početnoj stranici odabire opciju "Pregledaj

osnovnog tijeka	oglas". 2. Aplikacija prikazuje popis dostupnih oglasa.
Opis mogućih odstupanja	-

UC3 - Objavi oglas

Element	Opis
Namjena	Sustav omogućuje korisnicima da objave vlastite oglase
Opis	Registrirani korisnici mogu objavljivati vlastite oglase za ulaznice.
Glavni aktor	Korisnik
Preduvjeti	Korisnik mora biti registriran.
Opis osnovnog tijeka	1. Korisnik na početnoj stranici odabire opciju "Objavi oglas". 2. Korisnik unosi podatke o ulaznicama (naziv, datum, mjesto). 3. Sustav stvara oglas te ga čini vidljivim i drugim korisnicima.
Opis mogućih odstupanja	-

UC4 - Zamijeni ulaznice

Element	Opis
	Omogućavanje korisnicima

Namjena	zamjenu ulaznica
Opis	Korisnik pronalazi oglas s ponuđenom razmjenom, te daje ponudu, vlastitu kartu koju nudi kao zamjenu
Glavni aktor	Korisnik
Preduvjeti	Korisnik mora biti registriran te oglas s kojim se želi zamijeniti mora biti oglas otvoren za zamjene
Opis osnovnog tijeka	<ol style="list-style-type: none">1. Korisnik na početnoj stranici odabire opciju "Pregledaj oglase".2. Aplikacija prikazuje popis dostupnih oglasa.3. U slučaju pronalaska odgovarajućeg, korisnik šalje zahtjev za razmjenu4. Sustav obavještava drugog korisnika o razmjeni te čeka njegov odgovor5. Sustav odgovor drugog korisnika sprema u bazu podataka6. Sustav obavještava prvog korisnika o prihvatanju/odbijanju razmjene ulaznica
Opis mogućih odstupanja	<p>3.a) Nemogućnost pronalaska odgovarajućeg zahtjeva za razmjenu ulaznica:</p> <ol style="list-style-type: none">1. Sustav ispisuje poruku da nema odgovarajućih ponuda2. Proces se nastavlja u koraku 1 osnovnog tijeka.

UC5 - Prilagodi prikaza oglasa temeljem korisničkih preferencija

Element	Opis
Namjena	Praćenje preferencija korisnika
Opis	Aplikacija bilježi korisničke "lajkove" i "dislajkove" na oglase te prilagođava prikaz temeljen na tim preferencijama.
Glavni aktor	Korisnik
Preduvjeti	Korisnik je prijavljen u aplikaciju.
Opis osnovnog tijeka	<ol style="list-style-type: none"> 1. Korisnik označava oglas s "lajkom" ili "dislajkom". 2. Aplikacija pohranjuje ovu informaciju za buduću prilagodbu prikaza. 3. Aplikacija prikazuje oglase koji odgovaraju korisničkim preferencijama.
Opis mogućih odstupanja	-

UC6 - Pregledaj i upravljaj transakcijama

Element	Opis
Namjena	Sustav omogućuje korisniku pregled i upravljanje transakcijama
Opis	Korisnici mogu pregledavati povijest svojih transakcija u aplikaciji te upravljati istima.
Glavni	

aktor	Korisnik
Preduvjeti	Korisnik je prijavljen u aplikaciju.
Opis osnovnog tijeka	<ol style="list-style-type: none"> 1. Korisnik na početnoj stranici odabire opciju "Povijest transakcija". 2. Aplikacija prikazuje popis svih prethodnih transakcija (kupovina, prodaja, zamjena). 3. Korisnik može filtrirati popis prema datumu, vrsti ili statusu transakcije.
Opis mogućih odstupanja	<ol style="list-style-type: none"> 2.a) Sustav se ne uspijeva povezati s bazom podataka <ol style="list-style-type: none"> 1. Sustav ispisuje poruku o pogrešci s povezivanjem na bazu podataka. 2. Proces se nastavlja u koraku 1 osnovnog tijeka

UC7 - Automatski ukloni i ažuriraj oglase

Element	Opis
Namjena	Sustav automatski uklanjanja i ažuririra oglase
Opis	Oglasi se automatski uklanjaju kada istekne datum događaja.
Glavni aktor	Sustav
Preduvjeti	Oglas ima datum događaja.
Opis osnovnog	<ol style="list-style-type: none"> 1. Sustav provjerava datum događaja oglasa. 2. Ako je datum prošao, oglas se automatski uklanja iz baze

tijeka	podataka. 3. Oglas više nije vidljiv korisnicima.
Opis mogućih odstupanja	-

UC8 - Pregledaj korisničke račune

Element	Opis
Namjena	Sustav omogućava Administratoru uvid u popis korisnika i njihove aktivnosti radi lakše kontrole i kvalitetnijeg rada aplikacije
Opis	Administrator pregledava korisničke račune
Glavni aktor	Administrator
Preduvjeti	Administrator ima pristup administratorskom sučelju.
Opis osnovnog tijeka	1. Administrator na početnoj stranici odabire opciju "Upravljanje korisnicima". 2. Administrator pregledava popis korisnika i njihove aktivnosti.
Opis mogućih odstupanja	-

UC9 - Pregledaj prijavu o korisničkoj zlouporabi

Element	Opis

Namjena	Uloga administratora
Opis	Administrator pregledava prijave zlouporaba korisničkih računa
Glavni aktor	Administrator
Preduvjeti	Administrator ima pristup administratorskom sučelju.
Opis osnovnog tijeka	<ol style="list-style-type: none"> Administrator na početnoj stranici odabire opciju "Upravljanje prijavama zlouporabe". Administrator na popisu prijave zlouporabe korisničkih račune može vidjeti detaljan opis prijave korisnika Administrator razmotri prijavu i unese odluku
Opis mogućih odstupanja	-

UC10 - Deaktiviraj korisnički račun

Element	Opis
Namjena	Uloga administratora
Opis	Administrator deaktivira korisničke račune
Glavni aktor	Administrator
Preduvjeti	Administrator ima pristup administratorskom sučelju.
	<ol style="list-style-type: none"> Administrator na početnoj stranici odabire opciju "Upravljanje korisnicima".

Opis osnovnog tijeka	2. Administrator na popisu korisnika s njihovim aktivnostima odabire korisnike koje je potrebno deaktivirati 3. Administrator uklanja račune koje je potrebno ukloniti
Opis mogućih odstupanja	-

UC11 - Generiraj izvještaj o aktivnostima i obavijesti korisnika

Element	Opis
Namjena	Sustav omogućava korisniku da generira izvještaj u svrhu uvida u podatke korištenja sustava u nekom periodu
Opis	Administrator generira izvještaj na temelju povijesti transakcija i prosječnog vremena prodaje/zamijene oglasa
Glavni aktor	Administrator
Preduvjeti	Administrator ima pristup administratorskom sučelju.
Opis osnovnog tijeka	1. Administrator na početnoj stranici odabire opciju "Upravljanje korisnicima". 2. Administrator generira izvještaj na temelju informacija uzetih iz baze podataka 3. Administrator generirani izvještaj šalje korisniku
Opis	3.a) Zakašnjelo izvještavanje korisnika: Sustav automatski

mogućih odstupanja	šalje podsjetnik administratoru ili ga obavještava o kašnjenju izvještaja.
-------------------------------	--

UC12 - Poveži se s vanjskim servisima za prikaz informacija o vremenskoj prognozi

Element	Opis
Namjena	Sustav omogućava korisniku pregled informacija o vremenskoj prognozi za vrijeme održavanja događaja iz oglasa
Opis	Aplikacija se povezuje s vanjskim servisima za dobivanje informacija o vremenskoj prognozi
Glavni aktor	Korisnik
Preduvjeti	Korisnik je prijavljen u aplikaciju.
Opis osnovnog tijeka	<ol style="list-style-type: none"> 1. Korisnik pregledava dostupne oglase 2. Aplikacija šalje upit vanjskom servisu za dobivanje podataka. 3. Aplikacija prikazuje informacije o vremenskoj prognozi
Opis mogućih odstupanja	<p>2.a) Sustav se ne uspijeva povezati s vanjskim servisom za dobivanje informacija o vremenskoj prognozi:</p> <ol style="list-style-type: none"> 1. Sustav ispisuje poruku o pogrešci s povezivanjem na vanjskim servisom za

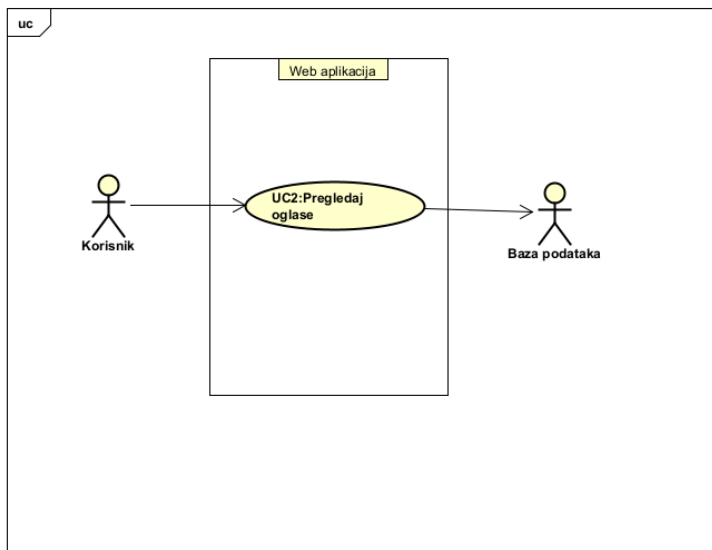
	<p>informacije o vremenskoj prognozi</p> <p>2. Proces se nastavlja u koraku 1 osnovnog tijeka</p>
--	---

UC13 - Kupi ulaznice

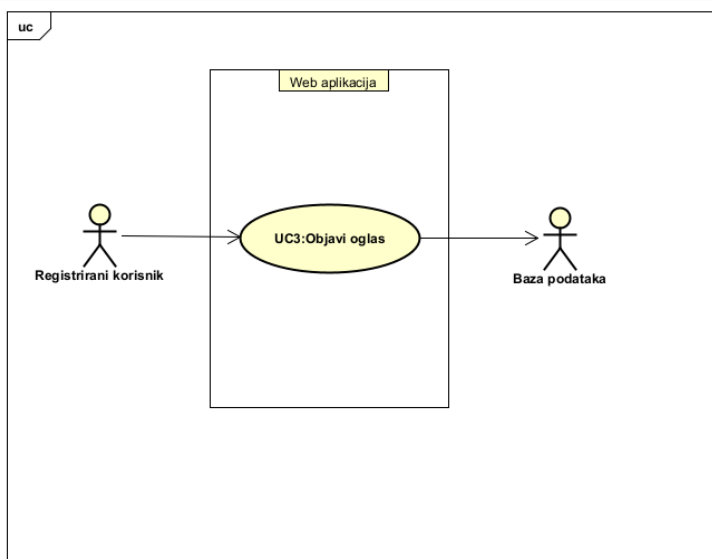
Element	Opis
Namjena	Omogućavanje korisnicima kupnju ulaznica
Opis	Korisnik pronalazi oglas koji želi kupiti, te se klikom na gumb izvršava kupnja
Glavni aktor	Korisnik
Preduvjeti	Korisnik mora biti registriran te oglas koji želi kupiti mora biti aktivan
Opis osnovnog tijeka	<ol style="list-style-type: none"> 1. Korisnik na početnoj stranici odabire opciju "Pregledaj oglase". 2. Aplikacija prikazuje popis dostupnih oglasa. 3. U slučaju pronalaska odgovarajućeg, korisnik kupuje oglas. 4. Sustav obavještava korisnika čiji je oglas o prodanoj ulaznici.
Opis mogućih odstupanja	<ol style="list-style-type: none"> 1. Oglas više nije dostupan. 2. Sustav ne uspijeva obavijestiti prodavatelja o kupnji zbog tehničkog problema.

1. Dijagram obrazaca uporabe za ključne funkcionalnosti

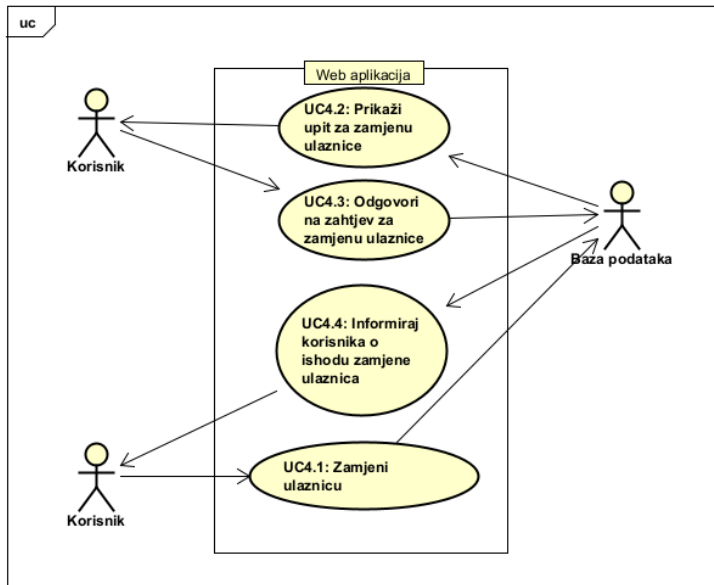
▼ Pregledavanje oglasa



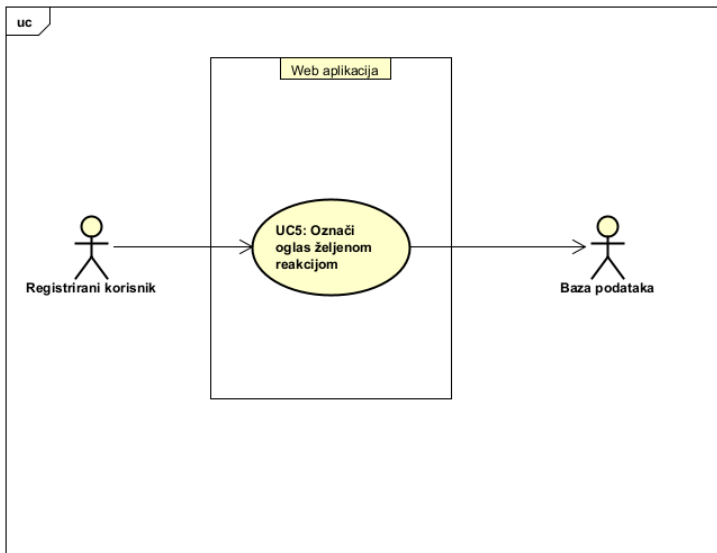
▼ Objavljivanje oglasa



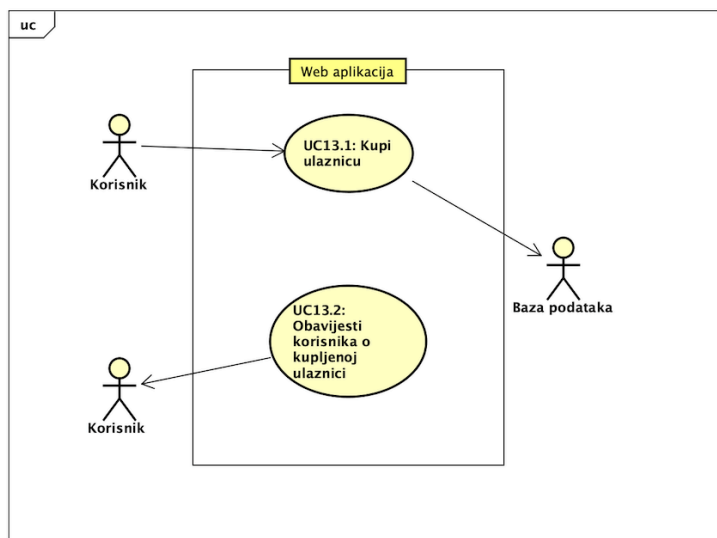
▼ Zamjena ulaznica



▼ Prilagodba prikaza oglasa temeljem korisničkih preferencija

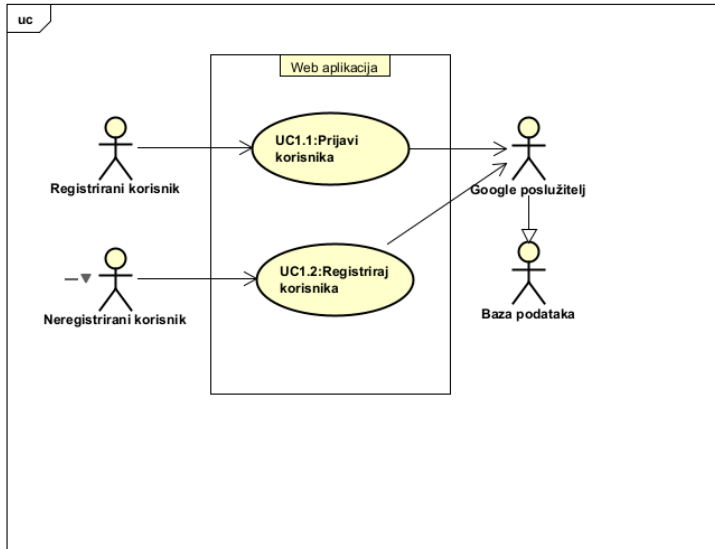


▼ Kupnja ulaznica

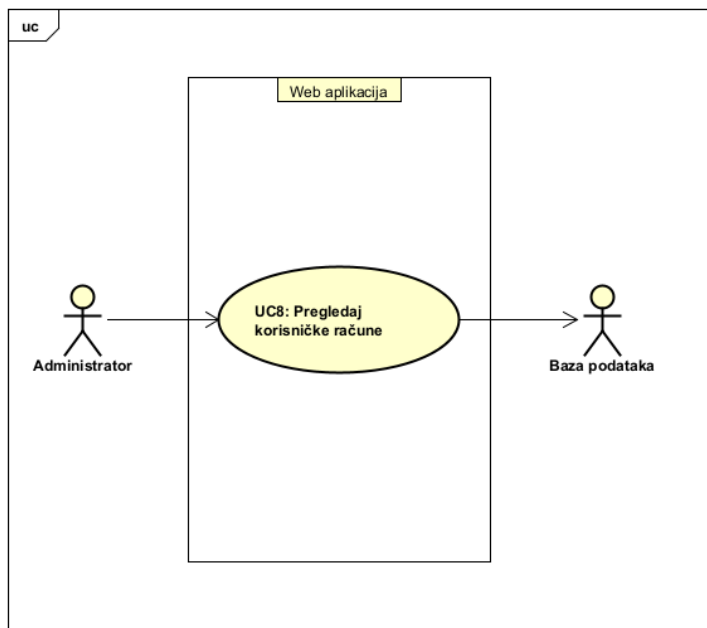


2. Dijagram obrazaca uporabe za korisničke uloge

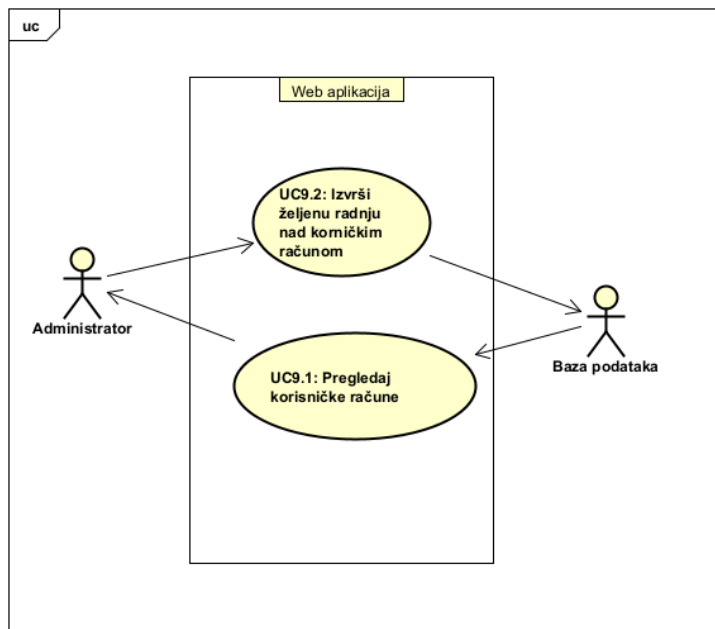
▼ Registracija i prijava korisnika



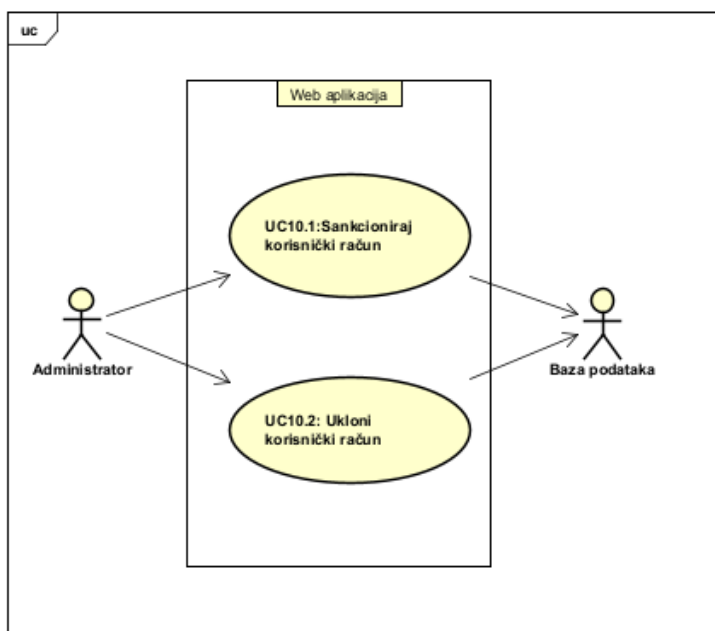
▼ Pregled korisničkih računa



▼ Pregled prijavljenih korisničkih računa

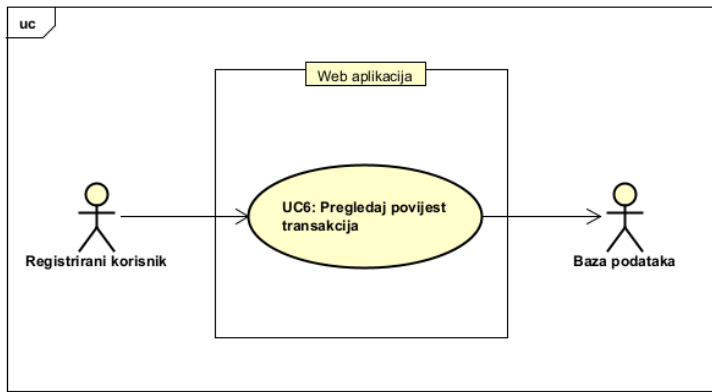


▼ Sankcioniranje i uklanjanje korisničkih računa

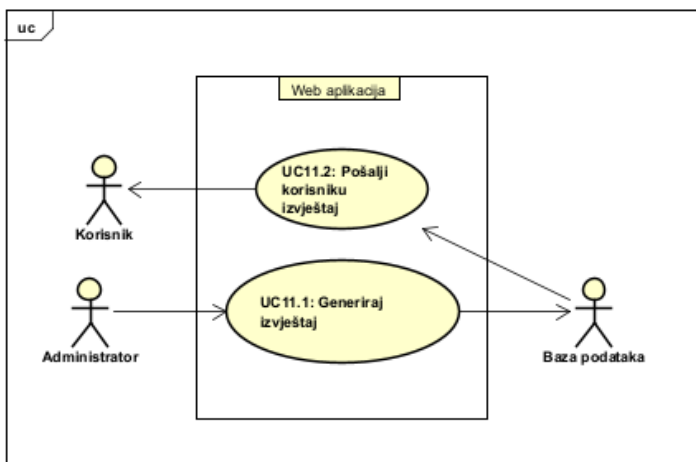


3. Dijagram obrazaca uporabe za osnovne poslovne procese

▼ Pregled i upravljanje transakcijama

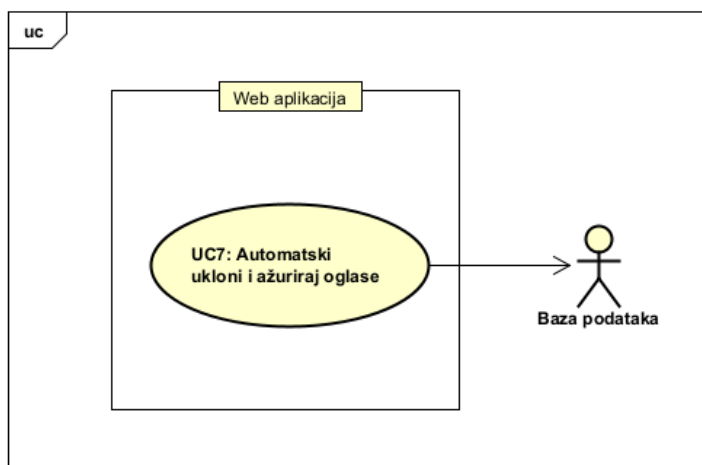


▼ Generiranje izvještaja o aktivnostima i obavještavanje korisnika



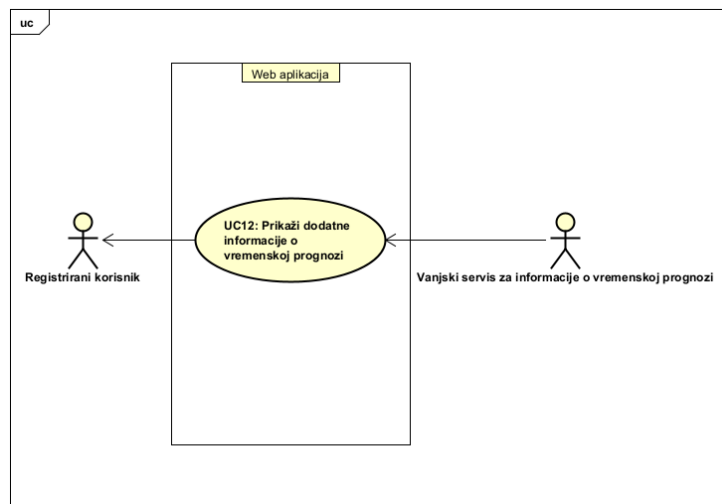
4. Dijagram obrazaca uporabe za kritične sustave i integracije

▼ Automatsko uklanjanje i ažuriranje oglasa



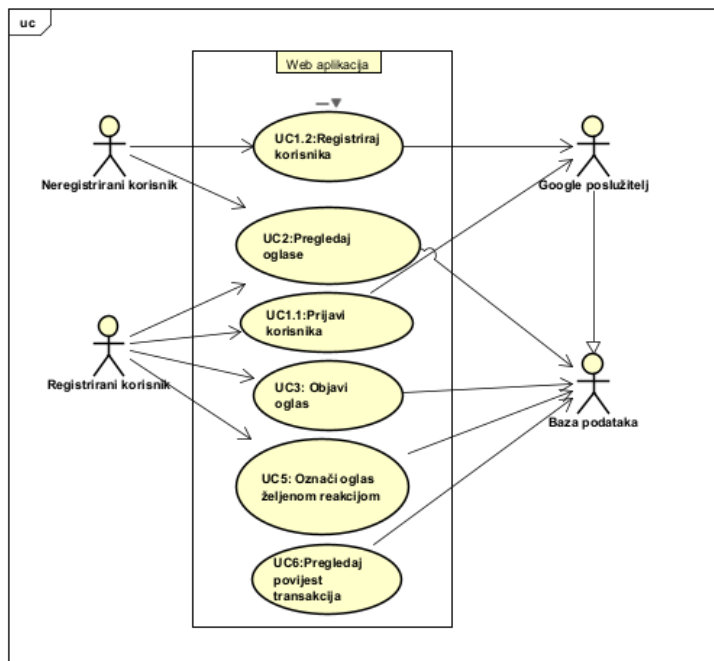
▼ Povezivanje s vanjskim servisima za prikaz

informacija o vremenskoj prognozi

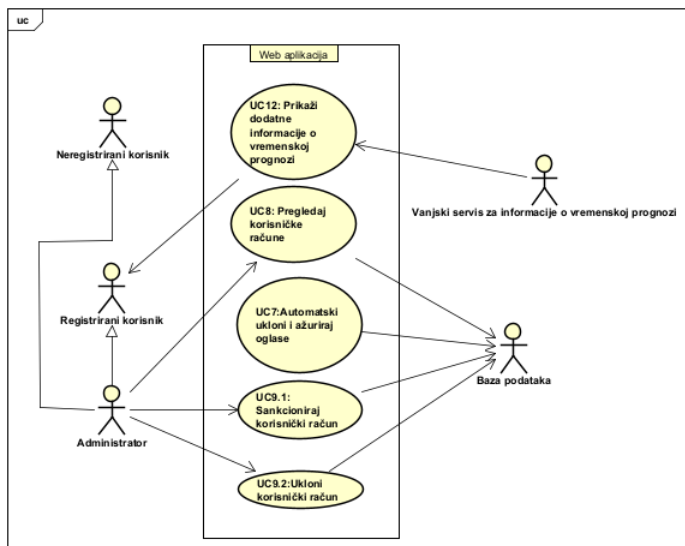


Dijagrami obrazaca uporabe

▼ Dijagram obrasca uporabe 1

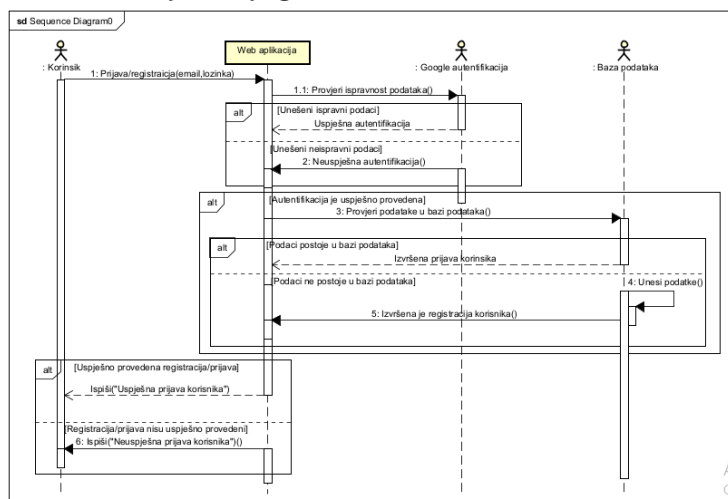


▼ Dijagram obrasca uporabe 2

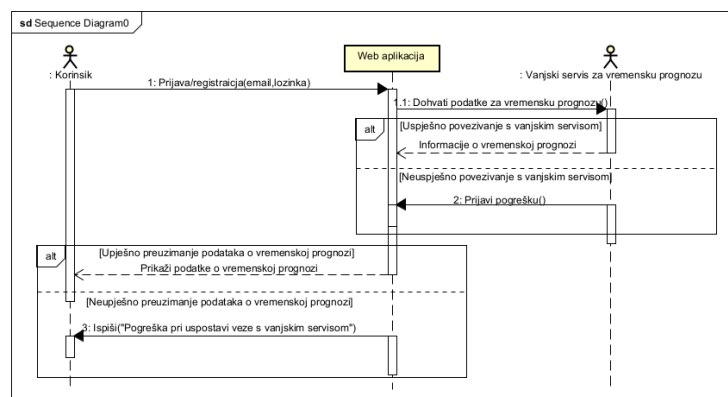


Sekvencijski dijagrami

▼ Sekvencijski dijagram 1



▼ Sekvencijski dijagram 2



Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

Redni broj	Naziv obrasca uporabe	Funkcionalni zahtjevi
UC1	Registriraj i prijavi korisnika	F-001
UC2	Pregledaj oglase	F-002, F-010
UC3	Objavi oglase	F-003
UC4	Zamijeni ulaznice	F-005, F-008
UC5	Prilagodi prikaz oglasa temeljem korisničkih preferencija	F-011
UC6	Pregledaj i upravljaj transakcijama	F-006, F-010
UC7	Automatski ukloni i ažuriraj oglase	F-009
UC8	Pregledaj korisničke račune	F-012
UC9	Pregledaj prijavu o korisničkoj zlouporabi	F-012
UC10	Deaktiviraj korisnički račun	F-013
UC11	Generiraj izvještaj o aktivnostima i obavijesti korisnika	F-014
	Poveži se s	

UC12	vanjskim servisima za prikaz informacija o vremenskoj prognozi	F-015
UC13	Kupi ulaznice	F-004, F-007

Swappet



m / S...

Q Type / to search

<> Code Issues Pull requests Actions Projects Wiki Security 11

4. Arhitektura i dizajn sustava

Edit

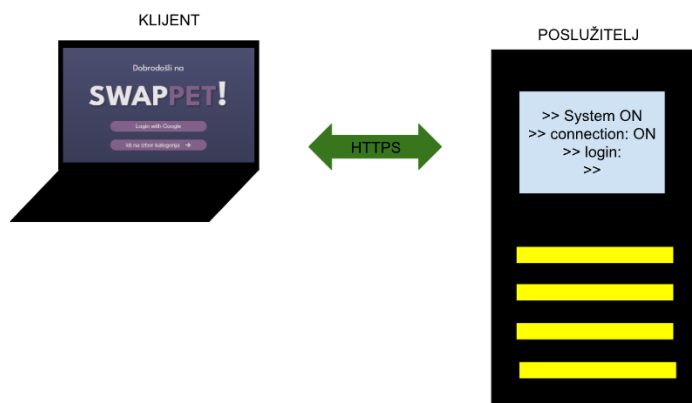
New page

Dominik Mandić edited this page 2 hours ago · [30 revisions](#)

Arhitektura sustava

Opis arhitekture

• **Stil arhitekture:** Na najvišoj razini, arhitektura aplikacije prati model **klijent-poslužitelj**. Za stil klijent-poslužitelj odlučili smo se jer omogućuje jednostavno upravljanje podacima i resursima na poslužitelju, čime je olakšano održavanje, sigurnost i ažuriranje sustava. Klijentima smanjuje potrebe za korištenje resursa njihovih računala. Omogućuje skalabilnost za veći broj korisnika. Također ovaj stil je relativno jednostavan za izvođenje, a zadovoljavao je naše potrebe i zahtjeve stoga nije bilo potrebe za odabirom nekog stila koji je zahtjevniji za implementaciju.



Slika 4.1: vizualizacija klijent-poslužitelj

• **Podsustavi:**

Pages 13

▶ Home

▶ 1. Opis projektnog zadatka

▶ 2. Analiza zahtjeva

▶ 3. Specifikacija zahtjeva sustava

▼ 4. Arhitektura i dizajn sustava

Arhitektura sustava

Opis arhitekture

Obrazloženje odabira
arhitektureOrganizacija sustava na
visokoj razini

Organizacija aplikacije

Baza podataka

Opis tablica

tipDog

korisnik

oglas

nadtransakcija

transakcija

ulaznica

jeTip

voliOglas

jeUkljucen

seMijenja

1. Klijent, razvijen u Reactu, predstavlja korisničko sučelje i osigurava interakciju s korisnicima.
2. Poslužitelj, implementiran u Spring Bootu, ima zadatak obraditi sve zahtjeve korisnika, rukovati poslovnom logikom i komunikacijom s bazom podataka.
3. Baza podataka, PostgreSQL, služi kao centralizirani sustav pohrane svih podataka potrebnih za rad aplikacije, uključujući korisničke informacije, oglase, događaje i druge relevantne entitete.

• **Preslikavanje na radnu platformu:** Za deployment aplikacije odabran je Render, cloud platforma koja pruža jednostavno postavljanje i upravljanje aplikacijama. Docker se koristi za standardizaciju okruženja, a GitHub Actions za automatizaciju build i deploy procesa. Render je odabran zbog fleksibilnosti, skalabilnosti, podrške za moderne alate te besplatnih opcija.

• **Spremišta podataka:** Podatci su pohranjeni pomoću relacijska baza podataka, zbog svoje brzine dohvata i upisa za manje do srednje velike sustave. Organizacija podatak je strukturirano i organizirana pomoću tablica, koje su međusobno povezane jasnim odnosima.

• **Mrežni protokoli:** Klijent komunicira s poslužiteljem putem HTTP protokola, šaljući zahtjeve za potrebne podatke ili akcije. Baza i poslužitelj komuniciraju putem PostgreSQL protokola kojim, JPA sa servera "prevodi" upite za podatke u SQL upite i na taj način dobiva i daje podatke bazi.

• **Globalni upravljački tok:**

1. Početna interakcija korisnika sa sustavom.
2. Oblikovanje HTTP zahtjeva na klijentu koji se upućuje poslužitelju.

spor

deaktiviranOglas

Dijagram baze podataka

Dijagram razreda

Dinamičko ponašanje aplikacije

UML dijagrami stanja

UML dijagrami aktivnosti

▸ [5. Arhitektura komponenata i razmj...](#)

▸ [6. Ispitivanje programskog rješenja](#)

▸ [7. Tehnologije za implementaciju a...](#)

▸ [8. Upute za puštanje u pogon](#)

▸ [9. Zaključak i budući rad](#)

▸ [A. Popis literature](#)

▸ [B. Dnevnik promjena dokumentacije](#)

▸ [C. Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/monamihokovic/Swappet/wiki/4.-Arhitektura-i-dizajn-sustava>



3. Obrada HTTP zahtjeva na poslužitelju, aktivacija potrebnih slojeva (kontroler, poslovna logika) te slanje/zahtjev za podacima bazi podatka)
4. Dohvaćanje/Pohrana podataka u bazi podataka i slanje podataka poslužitelju.
5. Odgovor na HTTP zahtjev korisniku, vrćanjem odgovarajuće poruke i resursa.
6. Prikaz podataka/poruke korisniku

• **Sklopovskoprogramski zahtjevi:** Na poslužitelju je potreban moderan operacijski sustav, koji je sigurnosno održavan, kako bi mogao podržati rad modernog Java JDK (17 ili noviji) te baze podatak u PostgreSQL-u kroz PgAdmin4. Također poslužitelj mora posjedovati internetsku vezu kako bi mogao komunicirati s klijentima. Klijentu je potreban pristup modernom web pregledniku kako bi mogao pristupiti uslugama naše web aplikacije.

Obrazloženje odabira arhitekture

• **Izbor arhitekture temeljen na principima oblikovanja:** Arhitektura klijent-poslužitelj je odabrana zbog:

1. Skalabilnost = arhitektura omogućava relativno jednostavnu prilagodbu u slučaju povećanja broja korisnika koji koriste web stranicu. To je ponajviše omogućeno kroz dodavanje novih računalnih resursa na postojeći poslužitelj čime se omogućuje povećanje propusnosti web usluge.
2. Sigurnost = sigurnost je olakšana zbog centralizirane baze podataka i obrade zahtjeva, jer nema potrebe za dodatnim slanjem podatka između komponenti koje bi trebalo štititi u komunikacijskim kanalima.
3. Fleksibilnost = moguće je da odvojeni timovi rade na razvoju backend-a, frontend-a i baze

jer sustavi nisu međusobno povezani način implantacije već samo izmjenjuju potrebne informacije.

4. Niska povezanost = ponajprije vidljiva u komunikaciji baze podataka i poslužitelja, gdje su pomoću JPA u potpunosti odvojeni podatci od izvora u bazi. Stoga je moguće zamijeniti bazu nekom drugom i ne bi se javili veći problemi u radu cijele aplikacije.

• Razmatrane alternative:

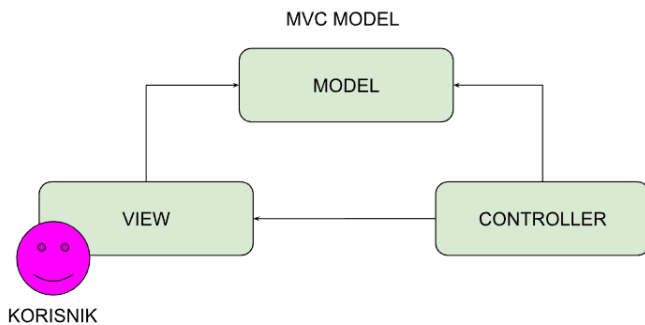
1. Monolitna arhitektura: Sustav bi bio implementiran kao jedna velika aplikacija, u kojoj su svi slojevi (UI, poslovna logika i baza podataka) povezani. Od monolita smo odustali zbog težeg skaliranja i održavanja, jer promjena jednog dijela može utjecati na cijeli sustav. Također takva arhitektura nudi nisku fleksibilnost za buduće proširenje.
2. Mikroservisna arhitektura: Svaka funkcionalnost bila bi implementirana kao neovisna usluga, npr. zasebni servisi za korisničke račune, administratorske račune, kupnju, razmjenu, prijavu, ... Od mikroservisa smo odustali zbog značajno veće složenosti izrade takvog sustava, uz naglasak na potrebe sigurnosti i sinkronizacije. Kako potrebe naše stranice nisu zahtijevale veću skalabilnost ili modularnost nije bilo potrebe za otežavanjem izrade uz marginalno bolje rezultate rada aplikacije.

Organizacija sustava na visokoj razini

- **Klijent-poslužitelj:** Klijent, razvijen u Reactu, predstavlja korisničko sučelje i osigurava interakciju s korisnicima. Klijent komunicira s poslužiteljem putem HTTP protokola, šaljući zahtjeve za potrebne podatke ili akcije. Poslužitelj, implementiran u Spring Bootu, ima zadatak obraditi sve zahtjeve korisnika, rukovati poslovnom logikom i komunikacijom s bazom podataka
- **Baza podataka:** Baza podataka, PostgreSQL, služi kao centralizirani sustav pohrane svih podataka potrebnih za rad aplikacije, uključujući korisničke informacije, oglase, događaje i druge relevantne entitete. Također baza pomoću triggera upravlja razmjenama i prodajama, jer oblikuje odgovor sustava na odluke korisnika vezane uz njihove oglase. (primjerice ako korisnik odbije zamjenu, baza "reaktivira" karte i obilježava transakciju kao nevažeću)
- **Grafičko sučelje:** Sustav koristi web aplikaciju kao korisničko sučelje. Korisnici pristupaju aplikaciji putem web preglednika, što omogućava jednostavan pristup s bilo kojeg uređaja. Tehnologija oblikovanja je React koji se naslanja na HTML, CSS, JavaScript za dinamički prikaz podatka s obzirom na podražaje koje korisnik daje. Povezivanje s backend-om je provedeno putem RESTful API-ja koristeći HTTP zahtjev.

Organizacija aplikacije

Organizacija aplikacije prati MVC strukturu, gdje se frontend razvijen u Reactu smatra kao "View" sloj. Ovaj sloj prikazuje podatke korisnicima i omogućuje im interakciju s aplikacijom, šaljući zahtjeve prema backendu. Backend, razvijen u Spring Bootu, ima ulogu "Controllera", gdje se zahtjevi s frontenda obrađuju, pokreću se potrebni servisi te se korisniku kao odgovor šalju obrađeni podatci. Servisi primjenjuju poslovnu logiku aplikacije; omogućuju obradu i manipulaciju podataka te njihovo prosljeđivanje prema bazi podataka. Konačno, "Model" predstavlja sloj baze podataka, gdje se podaci trajno pohranjuju i čuvaju na PostgreSQL poslužitelju.



Slika 4.2: MVC struktura

Baza podataka

Baza podataka temelji se na relacijskom modelu izvedenom u PostgreSQL-u. Relacijski model je odabran zbog učinkovitog i jednostavnog upravljanje srednje velikom količinom podataka, koju bi sustav namijenjen velikom broju korisnika trebao generirati. Zadaća tablice je pohrana, umetanje, izmjena te dohvat podataka. Također je idejno zamišljeno da baza obavlja veći dio „sustavskih“ promjena podataka kroz okidače. Glavna namjena sustava je omogućavanje prodaje i zamjene ulaznica za razne događaje te je stoga naglasak sustava na upravljanju oglasima sa ulaznicama. Baza podataka uključuje sljedeće tablice:

- tipDog
- jeTip (*nastao zbog veze N na N tablica oglas i tipDog*)
- oglas
- voliOglas (*nastao zbog veze, označavanja oglasa koji se korisniku ne sviđaju, N na N tablica oglas i korisnik*)
- korisnik
- jeUključen (*nastao zbog veze, jer više korisnika treba odobriti jednu transakciju, N na N tablica korisnik i transakcija*)
- nadtransakcija
- transakcija
- seMijenja (*nastao zbog veze, jer moguće je zamijeniti jednu "vrjedniju" kartu za više "manje vrijednih", N na N tablica ulaznica i transakcija*)
- ulaznica
- spor
- deaktiviranOglas

Opis tablica

tipDog

tipDog: tablica koja sadrži podatke o tipu događaja (koncert, rock, izložba, ...) te odgovarajuću šifru za tip događaja.

Atribut	Tip podatka	Opis varijable
idDog (PK)	INT	Jedinstveni identifikator tipa događaja
nazivtipa	VARCHAR(100)	Naziv tipa događaja

korisnik

korisnik: tablica koja sadrži podatke o svim korisnicima, koji su potrebi za prijavu (za administratore i kupce), postavljanje oglasa te provođenje transakcije.

Atribut	Tip podatka	Opis varijable
email (PK)	VARCHAR(50)	Jedinstveni identifikator korisnika
username	VARCHAR(50)	Jedinstveno ime korisnika u sustavu
idKorisnik	INT	Jedinstveni identifikator korisnika iz vanjskog servisa za autentifikaciju i prijavu
uloga	INT	Uloga unutar stranice koju korisnik ima

		(korisnik ili administrator)
koristi	INT	Je li korisnik deaktiviran ili ima pravo punog pristupa

oglas

oglas: tablica koja sadrži sve potrebne podatke o oglasu (osim karata koje su predmet oglasa).
Povezana je N na 1 s tablicom korisnik preko email-a.

Atribut	Tip podatka	Opis varijable
idOglas (PK)	INT	Jedinstveni identifikator oglasa
datum	TIMESTAMP	Datum i vrijeme održavanja događaja iz oglasa
ulica	VARCHAR(50)	Ulica održavanja događaja iz oglasa
grad	VARCHAR(50)	Grad održavanja događaja iz oglasa
kucnibr	VARCHAR(50)	Kućni broj održavanja događaja iz oglasa

opis	VARCHAR(1000)	Opis događaja u oglasu
aktivan	INT	Indikator aktivnosti oglasa
tipOglas	INT	Indikator radi li se o zamjeni ili prodaji
opisZamjene	VARCHAR(1000)	Opis prihvatljivih oglasa za zamjenu, ako je odabrana zamjena
email (FK)	VARCHAR(50)	Jedinstveni identifikator korisnika

nadtransakcija

nadtransakcija: tablica koja sadrži podatak o nadtransakciji izvedenoj nad nekim lancem razmjene.

Atribut	Tip podatka	Opis varijab
idNadtransakcije (PK)	INT	Jedinstveni identifikator nadtransakc
ukupno	INT	Indikator izvršenja događaja iz transakcije

dvNadPocetak	TIMESTAMP	Datum i vrijeme pokretanja nadtransakc

transakcija

transakcija: tablica koja sadrži podatak o transakciji izvedenoj nad nekom ulaznicom koja je dio oglasa. Povezana je N na 1 s tablicom ulaznica preko idUlaznice. Povezana je N na 1 s tablicom nadtransakcije preko idNadtransakcije.

Atribut	Tip podatka	Opis varijab
idTransakcije (PK)	INT	Jedinstveni identifikator transakcije
uspjesna	INT	Indikator izvršenja događaja iz transakcije
dvPocetka	TIMESTAMP	Datum i vrijeme pokretanja transakcije
idUlaznice (FK)	INT	Jedinstveni identifikator ulaznice
idNadtransakcije (FK)	INT	Jedinstveni identifikator nadtransakc

ulaznica

ulaznica: tablica sadrži podatke o ulaznici, (bez same adrese i vremena održavanja koncerta koji su navedeni u oglasu). Omogućuje da više korisnika može kupiti karte iz jednog oglasa (primjerice osoba A prvu kartu, osoba B 2 i 3 kartu, ako se u jednom oglasu prodaju 3 karte). Povezana je s tablicom oglas N na 1 preko idOglas.

Atribut	Tip podatka	Opis varijable
idUlaznice (PK)	INT	Jedinstveni identifikator ulaznice
red	INT	Red mjesta u prostoru održavanja događaja
broj	INT	Broj mjesta u prostoru održavanja događaja
vrstaUlaznice	INT	Šifra vrste ulaznice (VIP, obična, ...)
cijena	FLOAT	Cijena događaja u €
idOglas (FK)	INT	Jedinstveni identifikator oglasa kojem ulaznica pripada

jeTip

jeTip: tablica koja omogućuje da jedan oglas može biti više različitih tipova (npr. rock i koncert, ili rock i opera,...). Povezana je N na 1 s tablicom događaj preko idOglas , te N na 1 s tablicom tipDog preko idDog.

Atribut	Tip podatka	Opis varijable
idDog (PK)(FK)	INT	Jedinstveni identifikator tipa događaja
idOglas (PK)(FK)	INT	Jedinstveni identifikator oglasa

voliOglas

voliOglas: tablica koja omogućuje da jedan korisnik može (ne)voljeti i (ne)željeti da mu se prikazuju više specifičnih oglasa. Povezana je N na 1 s tablicom događaj preko idOglas , te N na 1 s tablicom korisnik preko email.

Atribut	Tip podatka	Opis varijable
email (PK) (FK)	VARCHAR(50)	Jedinstveni identifikator korisnika
idOglas (PK) (FK)	INT	Jedinstveni identifikator oglasa
voli	INT	Atribut za iskazivanje (ne)naklonosti prema oglasu

jeUkljucen

jeUkljucen: tablica koja omogućuje postojanje lanca razmijene, kada više od 2 korisnika treba potvrditi zamjenu kako bi se transakcija obavila. Povezana je N na 1 s tablicom transakcija preko idTransakcije, te N na 1 s tablicom korisnik preko email.

Atribut	Tip podatka	Opis varijable
email (PK) (FK)	VARCHAR(50)	Jedinstveni identifikator korisnika
idTransakcije (PK)(FK)	INT	Jedinstveni identifikator transakcije
odluka	INT	Indikator prihvatanja transakcije

seMijenja

seMijenja: tablica koja omogućuje razmjenu ulaznice u transakciji razmijene. Povezana je N na 1 s tablicom transakcija preko idTransakcije, te N na 1 s tablicom ulaznica preko idUlaznice.

Atribut	Tip podatka	Opis varijable
idUlaznice (PK)(FK)	INT	Jedinstveni identifikator ulaznice
idTransakcije (PK)(FK)	INT	Jedinstveni identifikator transakcije

spor

spor: tablica koja omogućuje podnošenje spora protiv korisnika, koji krši neku odrednicu korištenja aplikacije. Povezana je 1 na 1 s tablicom korisnik preko tuzioEmail, te 1 na 1 s tablicom korisnik preko tuzenEmail.

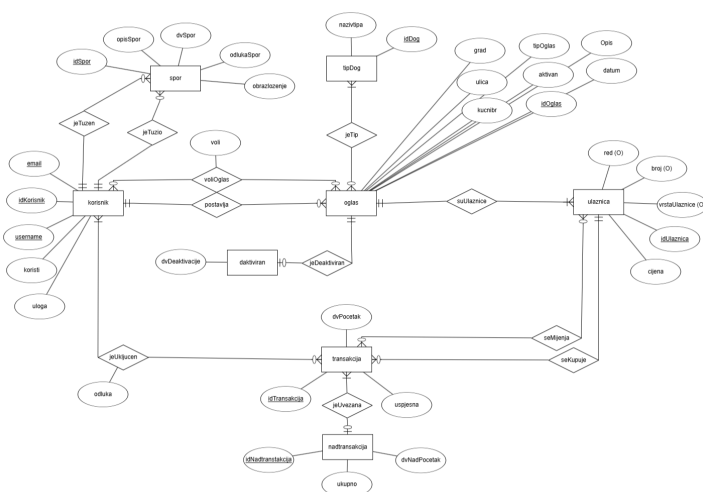
Atribut	Tip podatka	Opis varijal
idSpor (PK)	INT	Jedinstveni identifikator spora
opisSpor	VARCHAR(1000)	Detaljan opis spora koji popunjava korisnik koji tuži
dvSpor	TIMESTAMP	Datum i vrijeme kreiranja spoja
odlukaSpor	INT	Odluka administrat o sporu
obrazlozenje	VARCHAR(1000)	Obrazložen administrat o odluci spo
tuzioEmail (FK)	VARCHAR(50)	Email korisnika ko je podnio sp
tuzenEmail (FK)	VARCHAR(50)	Email korisnika protiv kojeg podnesen spor

deaktiviranOglas

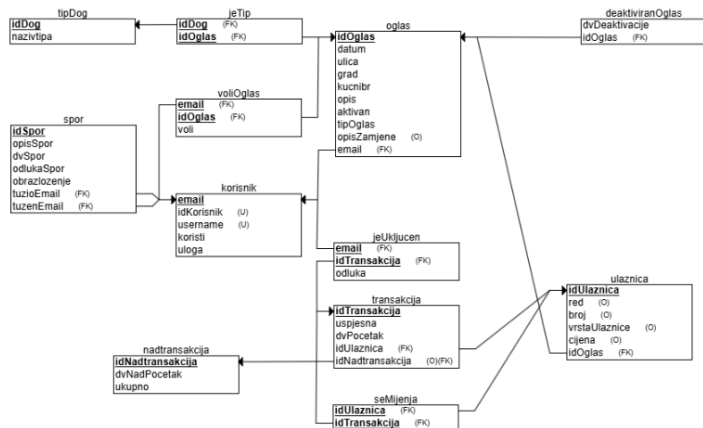
deaktiviranOglas: tablica koja omogućuje privremenu deaktivaciju oglasa (do n dana), prije čijeg je isteka moguće reaktivirati oglas i ponovno ga ponuditi na tržištu. Povezana je 1 na 1 s tablicom oglas preko idOglas.

Atribut	Tip podatka	Opis varijable
idOglas (PK, FK)	INT	Jedinstveni identifikator oglasa
dvDeaktivacije	TIMESTAMP	Datum i vrijeme deaktivacije oglasa

Dijagram baze podataka



Slika 4.3: Relacijski dijagram baze podataka



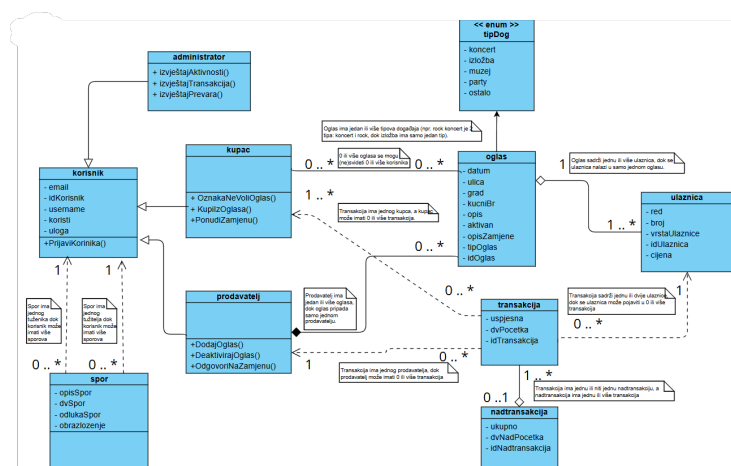
Dijagram razreda

*Razred **korisnik** je u bazi podataka spremljen kao jedan entitet, jer korisnik može u jednoj transakciji biti kupac, a u drugoj korisnik te stoga nije potrebno jasno razdvajati te funkcije. Administrator sustava je planiran u sljedećoj iteraciji sustava te će se za njega stvoriti novi entitet u tablici, koji će omogućiti da se administratorima dopuste sve tražene vrijednosti.

Razred **nadtransakcija** opisuje vezanost više transakcija koje sudjeluju u lancima razmjena.

Page 17 of 20

Enum **tipDog** sadrži kategorie događaja kojima oglas može pripadati.

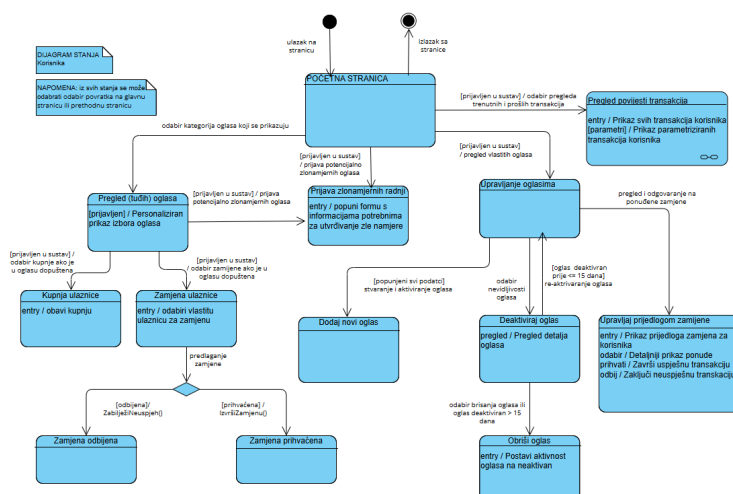


Slika 4.5: Dijagram razreda

Dinamičko ponašanje aplikacije

UML dijagrami stanja

Korisnik ulaskom na početnu stranicu može odabrati pregledavanje oglasa, upravljanje oglasima, povijest svojih transakcija te prijavu zlonamjernih oglasa. Prijava je uvjet za sva stanja osim za pregledavanje oglasa koje može obaviti i neregistrirani korisnik. Unutar pregleda oglasa može svoje rezultate prilagoditi tako da izabere želi li vidjeti oglase zamjene i/ili kupnje. Ako se odluči za zamjenu daje prijedlog zamjene, te ovisno o odgovoru osobe kojoj je zamjenu predložio, nastavlja sa postupkom ili u slučaju odbijanja transakcija se zapisuje a korisnik nastavlja svoje daljnje pregledavanje stranice. Ako korisnik tijekom pregledavanja vidi sumnjivi oglas može ga prijaviti kao potencijalno zlonamjerman (ili neistinit oglas). Ako se korisnik odluči za pregled povijesti transakcija, prikazati će mu se sve prošle i trenutne transakcije neovisno o uspješnosti istih. Odabirom na upravljanje oglasima, korisnik može odabrati stvaranje novog oglasa, deaktiviranje postojećeg, prijavom za sumnjive oglase ili pregled ponuda za oglase u kojima je označio da je zainteresiran za zamijene. Kada korisnik deaktivira oglas nudi mu se dodatna mogućnost brisanja oglasa ukoliko isto smatra potrebnim.



Slika 4.6: Dijagram stanja

UML dijagrami aktivnosti

[illegible]

Swappet 

m / S...

Q Type / to search

<> Code Issues Pull requests Actions Projects Wiki Security 11

5. Arhitektura komponenata i razmještaja

Edit

New page

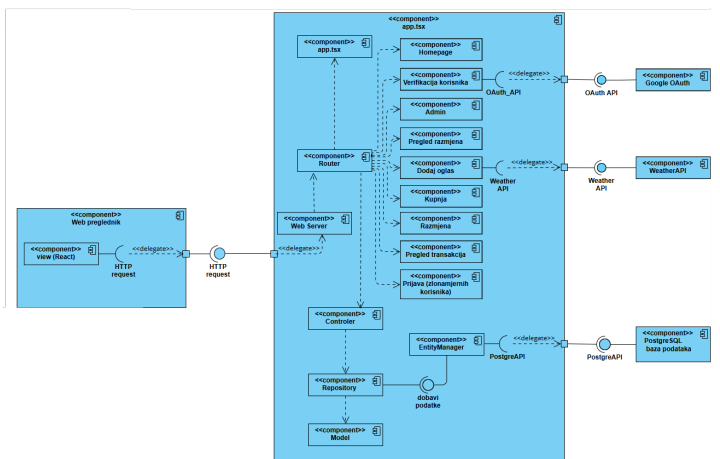
Dominik Mandić edited this page yesterday · [6 revisions](#)

Dijagram komponenata

Komponente smo najopćenitije podijelili na:

- komponente korisnika (Web preglednik)
- komponente poslužitelja (Swappet)
- komponente baze (PostgreSQL baza)
- komponente vanjskih servisa (GoogleOAuth, WeatherAPI)

Komponente korisnika su izvedene u Reactu, kako bi korisnici na vizualno privlačan način pristupali našem sadržaju. Komponente poslužitelja su izvedene u Spring boot-u programskim jezikom Java, kako bi bile u skladu sa zahtjevom o objektno orijentiranom backend-u. Baza je izvedena kroz PostgreSQL bazu podataka, a poslužitelj implementira i koristi JPA za pristup objektima koji su mu potrebni.



Slika 5.1 Dijagram komponenata

Pages 13

Home

1. Opis projektnog zadatka

2. Analiza zahtjeva

3. Specifikacija zahtjeva sustava

4. Arhitektura i dizajn sustava

5. Arhitektura komponenata i razmj...

Dijagram komponenata

Dijagram razmještaja

6. Ispitivanje programskog rješenja

7. Tehnologije za implementaciju a...

8. Upute za puštanje u pogon

9. Zaključak i budući rad

A. Popis literature

B. Dnevnik promjena dokumentacije

C. Prikaz aktivnosti grupe

[+ Add a custom sidebar](#)

Dijagram razmještaja

Clone this wiki locally

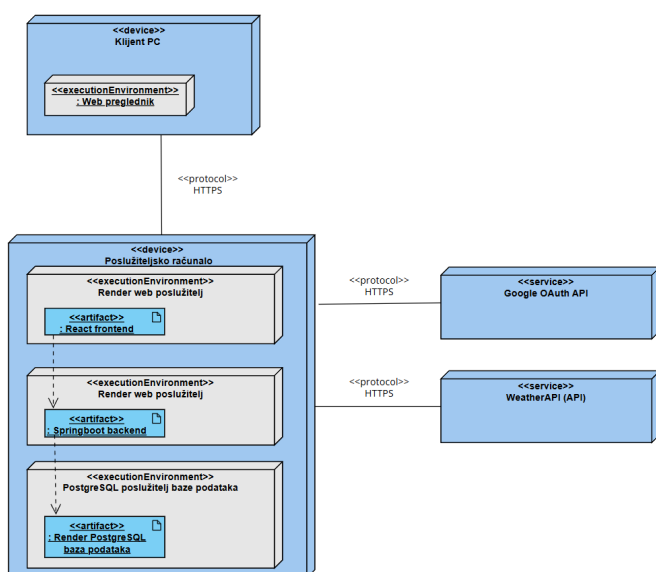
<https://github.com/monamihokovic>



Za rad aplikacije su nam potrebna:

- 2 uređaja (1. za serversku stranu, 2. za korisničku stranu)
- 2 vanjska servisa

Poslužiteljsko računalo je smješteno na Render online usluzi na kojoj je izvedena i PostgreSQL baza. Korisnik usluzi pristupa sa svog računala i HTTPS zahtjevima dohvaća tražene materijale i resurse.




Slika 5.2 Dijagram razmještaja

Swappet




☰

 m / S...

🔍

Type / to search



▼


+

▼

🕒

🔗

📁



<> Code

🕒 Issues

🔗 Pull requests

🎬 Actions

📁 Projects

📖 Wiki

🛡️ Security 11

6. Ispitivanje programskog rješenja

Edit

New page

pz55296 edited this page 3 minutes ago · [19 revisions](#)

1. TEST: Prijava korisnika s validnim podacima

1. Funkcionalnost koju testiramo:

Testira se postupak prijave korisnika na sustav koristeći validne korisničke podatke.

2. Ispitni slučaj:

Ulazni podaci:

- Email: rodjowara@gmail.com
- Lozinka: (unesena, ali nepravilna)

Očekivani rezultat:

- Sustav treba pokušati prijaviti korisnika. Budući da je lozinka nepravilna, prijava bi trebala završiti greškom, i sustav treba ispravno prikazati poruku o grešci (npr. "Neispravna lozinka").

Dobiveni rezultat:

- Prijava korisnika bila je neuspješna zbog nepravilne lozinke. Sustav je ispravno prikazao poruku o grešci.

3. Postupak provođenja ispitivanja:

Ispitni slučaj testira funkcionalnost prijave korisnika na sustav.

▼ Pages 13

Find a page...

▶ Home

▶ 1. Opis projektnog zadatka

▶ 2. Analiza zahtjeva

▶ 3. Specifikacija zahtjeva sustava

▶ 4. Arhitektura i dizajn sustava

▶ 5. Arhitektura komponenata i razmj...

▼ 6. Ispitivanje programskog rješenja

1. TEST: Prijava korisnika s validnim podacima

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

2. TEST: TestKomponenti2 - Kreiranje događaja s nepotpunim podacima

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

3. TEST: Kupovina karata s

https://github.com/monamihokovic/Swappet/wiki/6.-Ispitivanje-programskog-rješenja

Page 1 of 13

- Prvo se otvara stranica na URL-u
`https://swappet-app-iod2.onrender.com/` .
- Nakon toga, klikne se na gumb za prijavu.
- Na sljedećem koraku korisnik unosi email
(`rodjowara@gmail.com`), a zatim klikne na gumb za potvrdu.
- U sljedećem koraku, unosi se nepravilna lozinka i pokušava se prijaviti.
- Prilikom prijave, sustav je trebao prikazati odgovarajuću poruku o grešci, što je i bio očekivani rezultat.
- Na kraju, test završava, a pregled pogreške pokazuje da je sustav ispravno detektirao problem.

4. Kôd za testiranje:

Kôd za ovaj test nalazi se u

<https://github.com/monamihokovic/Swappet/blob/main/Dokumentacija/src/SeleniumCode/TestKomponenti1.java>.

2. TEST: TestKomponenti2 - Kreiranje događaja s nepotpunim podacima

1. Funkcionalnost koju testiramo:

Testira se postupak kreiranja događaja u sustavu s nepotpunim podacima, gdje korisnik ne unosi sve potrebne informacije.

2. Ispitni slučaj:

Ulazni podaci:

- Kategorija: "Koncerti"
- Datum: 2025-02-15
- Vrijeme: 18:30
- Vrsta transakcije: "Kupnja"
- Cijena ulaznice: 100
- Količina ulaznica: 1

maksimalnim brojem karata

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

4. TEST: Slanje zahtjeva za razmjenu karata

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

5. TEST: Potvrda oglasa koji nije označen za razmjenu

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

6. TEST: Uspješna deaktivacija oglasa od strane korisnika

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

1. TEST: Otvaranje stranice za prijavu

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

2. TEST: Odabir kategorija

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

- Red i sjedalo: 5 i 12
- Tip ulaznice: "Obična"
- Opis događaja: "Ovo je koncert poznatog benda."
- Ulica: "Glavna ulica"
- Grad: "Zagreb"
- **Nepotpuni podaci:** Kućni broj nije unesen.

Očekivani rezultat:

- Sustav bi trebao baciti grešku jer su podaci nepotpuni (nedostaje kućni broj), što znači da predaja nepotpunih podataka izaziva grešku u sustavu.

Dobiveni rezultat:

- Sustav prepoznaje da su podaci nepotpuni, pokazuje poruku o grešci i ne dopušta kreiranje događaja.

3. Postupak provođenja ispitivanja:

Ispitni slučaj testira funkcionalnost kreiranja događaja u sustavu s nepotpunim podacima.

- Prvo se otvara stranica za kreiranje događaja na URL-u `https://swappet-app-iod2.onrender.com/createEvent`.
- Na stranici, odabire se kategorija "Koncerti" putem radio gumba.
- Unosi se datum (2025-02-15) i vrijeme (18:30), a zatim se odabire vrsta transakcije "Kupnja".
- Unosi se cijena (100) i količina ulaznica (1), te se unose red (5) i sjedalo (12).
- Odabire se tip ulaznice kao "Obična".
- Unosi se opis događaja ("Ovo je koncert poznatog benda.") i ulica ("Glavna ulica"), te grad ("Zagreb").
- Međutim, kućni broj nije unesen, što čini

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

3. TEST: Predaja oglasa

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

4. TEST: Kupnja oglasa

1. Funkcionalnost koju testiramo:

2. Ispitni slučaj:

3. Postupak provođenja ispitivanja:

4. Kôd za testiranje:

▶ [7. Tehnologije za implementaciju a...](#)

▶ [8. Upute za puštanje u pogon](#)

▶ [9. Zaključak i budući rad](#)

▶ [A. Popis literature](#)

▶ [B. Dnevnik promjena dokumentacije](#)

▶ [C. Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/monamihokovic>



podatke nepotpunima.

- Kada korisnik pokuša predati obrazac klikom na gumb "Kreiraj događaj!", sustav prepoznaje grešku zbog nedostatka kućnog broja i odbija kreiranje događaja.
- Očekivana greška je detektirana i sustav ispravno reagira s odgovarajućom porukom o grešci.

4. Kôd za testiranje:

Kôd za ovaj test nalazi se u

<https://github.com/monamihokovic/Swappet/blob/main/Dokumentacija/src/SeleniumCode/TestKomponenti2.java>.

3. TEST: Kupovina karata s maksimalnim brojem karata

1. Funkcionalnost koju testiramo:

Testira se postupak kupovine karata za događaj s maksimalnim brojem karata.

2. Ispitni slučaj:

Ulazni podaci:

- Korisnik odabire događaj.
- Povećava cijenu putem klizača.
- Kupuje maksimalan broj karata (broj karata prikazan u aplikaciji).

Očekivani rezultat:

- Sustav treba omogućiti odabir i kupovinu maksimalnog broja karata, a nakon što se klikne na gumb za kupovinu, korisnik će završiti postupak s kupovinom željenog broja karata.

Dobiveni rezultat:

- Kupovina karata uspješno je završena. Sustav je ispravno dozvolio kupovinu maksimalnog broja karata, a cijena je prilagođena na temelju postavki u aplikaciji.

3. Postupak provođenja ispitivanja:

Ispitni slučaj testira funkcionalnost kupovine karata za događaj s maksimalnim brojem karata.

- Prvo se otvara stranica na URL-u
`https://swappet-app-iod2.onrender.com/advertisements`.
- Nakon toga, korisnik odabire kategoriju i odabrani događaj.
- Povećava se broj karata pomoću klizača.
- Zatim, klikne se na gumb za povećanje broja karata dok se ne dostigne maksimalni broj.
- Na kraju, korisnik klikne na gumb za kupovinu i test završava uspješno.

4. Kôd za testiranje:

Kôd za ovaj test nalazi se u

<https://github.com/monamihokovic/Swappet/blob/main/Dokumentacija/src/SeleniumCode/TestKomponenti3.java>.

4. TEST: Slanje zahtjeva za razmjenu karata

1. Funkcionalnost koju testiramo:

Testira se funkcionalnost slanja zahtjeva za razmjenu karata

2. Ispitni slučaj:

Ulazni podaci:

- Odabir ponude s padajuće liste: Prvi ponuđeni element u listi.

Očekivani rezultat:

- Sustav treba omogućiti selektiranje ponude putem padajuće liste i pravilno inicirati zahtjev za razmjenu putem gumba.
- Nakon selektiranja ponude, gumb za slanje zahtjeva treba biti omogućeno i treba biti moguće izvršiti klik na gumb kako bi se zahtjev poslao.

Dobiveni rezultat:

- Zahtjev za razmjenu karata uspješno je poslan.
- Sustav je omogućio selektiranje prvog elementa iz padajuće liste i iniciranje zahtjeva za razmjenu na temelju odabranog.

3. Postupak provođenja ispitivanja:

Ispitni slučaj testira funkcionalnost slanja zahtjeva za razmjenu karata:

- Prvo se otvara stranica na URL-u
`https://swappet-app-iod2.onrender.com/advertisements`.
- Zatim se pričekaju potrebni elementi za učitavanje, uključujući padajuću listu (`<select>` element).
- U sljedećem koraku, odabire se prvi element u toj listi putem `Select` klase.
- Nakon što je element odabran, klikne se na gumb koji inicira zahtjev za razmjenu karata.
- Na kraju, test završava i pokazuje da je zahtjev za razmjenu uspješno poslan.

4. Kôd za testiranje:

Kôd za ovaj test nalazi se u

<https://github.com/monamihokovic/Swappet/blob/main/Dokumentacija/src/SeleniumCode/TestKomponenti4.java>.

5. TEST: Potvrda oglasa koji nije označen za razmjenu

1. Funkcionalnost koju testiramo:

Testira se scenarij u kojem korisnik pokušava potvrditi razmjenu oglasa koji nije prethodno označen za razmjenu, što izaziva grešku.

2. Ispitni slučaj:

Očekivani rezultat:

- Sustav treba spriječiti potvrdu oglasa jer nije označen za razmjenu.
- Očekuje se da korisnik vidi odgovarajuću poruku o grešci (npr. "Oglas nije označen za razmjenu.").

Dobiveni rezultat:

- Klikom na gumb dolazi do greške jer sustav sprječava potvrdu oglasa koji nije označen za razmjenu.

3. Postupak provođenja ispitivanja:

Ovaj test simulira pokušaj potvrde razmjene bez odabira valjanih opcija:

1. Otvara se stranica na URL-u
`https://swappet-app-iod2.onrender.com/user/trades`.
2. Čeka se da gumb za potvrdu postane klikabilan.
3. Pokušava se kliknuti na gumb za potvrdu razmjene oglasa.
4. Budući da nisu odabrane valjane opcije, dolazi do greške.
5. Sustav treba prikazati odgovarajuću poruku o grešci, što ukazuje da je test uspješno izazvao grešku.

4. Kôd za testiranje:

Kôd za ovaj test nalazi se u

<https://github.com/monamihokovic/Swappet/blob/main/Dokumentacija/src/SeleniumCode/TestKomponenti5.java>.

6. TEST: Uspješna deaktivacija oglasa od strane korisnika

1. Funkcionalnost koju testiramo:

Testira se scenarij u kojem korisnik uspješno deaktivira oglas putem korisničkog sučelja.

2. Ispitni slučaj:

Očekivani rezultat:

- Oglas je uspješno deaktiviran, a sustav potvrđuje deaktivaciju putem poruke ili promjene statusa oglasa.

Dobiveni rezultat:

- Klikom na gumb za deaktivaciju sustav uspješno deaktivira oglas i prikazuje odgovarajuću potvrdu.

3. Postupak provođenja ispitivanja:

Test simulira deaktivaciju oglasa od strane korisnika:

1. Otvara se stranica na URL-u
`https://swappet-app-iod2.onrender.com/user/trades`.
2. Čeka se da gumb za deaktivaciju oglasa postane klikabilan.
3. Klikne se na gumb za deaktivaciju oglasa.
4. Sustav potvrđuje da je oglas uspješno deaktiviran, čime se završava test.

4. Kôd za testiranje:

Kôd za ovaj test nalazi se u

<https://github.com/monamihokovic/Swappet/blob/main/Dokumentacija/src/SeleniumCode/TestKomponenti6.java>.

1. TEST: Otvaranje stranice za prijavu

1. Funkcionalnost koju testiramo:

Testira se uspješno otvaranje stranice za prijavu na sustav.

2. Ispitni slučaj:

Scenarij:

- Korisnik pristupa aplikaciji i otvara stranicu za prijavu koristeći odgovarajući URL.

Očekivani rezultat:

- Sustav uspješno otvara stranicu za prijavu, prikazujući potrebne elemente za unos korisničkih podataka.

3. Postupak provođenja ispitivanja:

1. Otvara se početna stranica aplikacije na URL-u `https://swappet-app-iod2.onrender.com/`.
2. Korisnik klikne na link za prijavu s oznakom `Prijava` ili na odgovarajući gumb na početnoj stranici.
3. Sustav preusmjerava korisnika na stranicu za prijavu (URL: `/login`).
4. Stranica za prijavu se učitava, prikazujući elemente za unos emaila i lozinke.

4. Kôd za testiranje:

Kôd za ovaj test nalazi se u

<https://github.com/monamihokovic/Swappet/blob/main/Dokumentacija/src/SeleniumCode/TestLogin.png>.

2. TEST: Odabir kategorija

1. Funkcionalnost koju testiramo:

Testira se odabir kategorija i preusmjeravanje korisnika na prikaz oglasa nakon klika na gumb "Nastavi".

2. Ispitni slučaj:

Ulazni podaci:

- Lista kategorija: [1, 2, 3, 8] (primjer odabira kategorija).
- Kategorije se odabiru klikom na njih.

Očekivani rezultat:

- Korisnik odabire željene kategorije.
- Nakon klika na gumb „Nastavi“, korisnik se preusmjerava na stranicu koja prikazuje oglase prema odabranim kategorijama.

3. Postupak provođenja ispitivanja:

1. Otvara se stranica koja prikazuje dostupne kategorije oglasa.
2. Korisnik odabire kategorije klikom na željene kategorije s popisa [1] .
3. Nakon odabira kategorija, korisnik klikne na gumb „Nastavi“.
4. Sustav preusmjerava korisnika na stranicu koja prikazuje oglase prema odabranim kategorijama.

4. Kôd za testiranje:

Kôd za ovaj test nalazi se u

<https://github.com/monamihokovic/Swappet/blob/main/Dokumentacija/src/SeleniumCode/Test2.png>.

3. TEST: Predaja oglasa

1. Funkcionalnost koju testiramo:

Testira se proces predaje oglasa, uključujući popunjavanje podataka o događaju, te preusmjeravanje na stranicu za odabir kategorija nakon uspješne predaje.

2. Ispitni slučaj:

Ulazni podaci:

- Kategorija: npr. „Koncerti“.
- Vrijeme događaja: „20:00“.
- Datum događaja: „2025-02-15“.
- Cijena (ako je oglas za prodaju): „50“.
- Opis razmjene (ako je oglas za razmjenu): „Zamjena za druge karte“.
- Vrsta transakcije: „Prodaja“ ili „Razmjena“.
- Količina ulaznica: „2“.
- Tip ulaznice: „VIP“.
- Opis događaja: „Glazbeni spektakl“.
- Ulica događaja: „Ilica“.
- Kućni broj: „15“.
- Grad događaja: „Zagreb“.

Očekivani rezultat:

- Klikom na gumb „Kreiraj događaj!“ sustav upisuje podatke u bazu podataka.
- Ako je korisnik prethodno ulogiran, preusmjerava ga na stranicu za odabir kategorija.

3. Postupak provođenja ispitivanja:

1. Otvori se stranica za predaju oglasa.
2. Korisnik unosi potrebne podatke:
 - Kategorija događaja.
 - Vrijeme i datum događaja.
 - Cijenu (ako je prodaja) ili opis razmjene (ako je razmjena).
 - Vrstu transakcije.
 - Količinu i tip ulaznica.
 - Opis događaja.
 - Lokaciju događaja (ulica, kućni broj, grad).
3. Klikne na gumb „Kreiraj događaj!“.
4. Ako su svi podaci ispravno uneseni, oglas se pohranjuje u bazu podataka.
5. Korisnik se preusmjerava na stranicu za odabir kategorija.

4. Kôd za testiranje:

Kôd za ovaj test nalazi se u <src/SeleniumCode/Test3.jpg>.

4. TEST: Kupnja oglasa

1. Funkcionalnost koju testiramo:

Testira se proces kupnje ulaznica za određeni oglas, uključujući odabir broja ulaznica, slanje podataka o kupnji na backend te ispis poruke o uspješnoj kupnji.

2. Ispitni slučaj:

Ulazni podaci:

- Nema ulaznih podataka izravno od korisnika osim broja ulaznica koje odabire klikom na gumb „plus“ ili „minus“.

Očekivani rezultat:

- Klikom na gumb „Kupi“ šalju se podaci na backend:
 - ID oglasa.
 - Email korisnika (ako je ulogiran).
 - Broj ulaznica koje je korisnik odabrao.
- Ako je kupnja uspješna, na korisničkom sučelju prikazuje se poruka „Ulaznice kupljene“.

3. Postupak provođenja ispitivanja:



1. Otvori se stranica oglasa.
2. Korisnik koristi gumb „plus“ ili „minus“ kako bi odabrao broj ulaznica za određeni oglas.
3. Ako je korisnik ulogiran, gumb „Kupi“ postaje dostupan.
4. Klikom na gumb „Kupi“ šalju se podaci na backend:
 - ID oglasa.
 - Email korisnika.
 - Broj ulaznica.
5. Backend obrađuje zahtjev za kupnju.
6. Ako je kupnja uspješna, na korisničkom sučelju prikazuje se poruka „Ulaznice kupljene“.









4. Kôd za testiranje:

Kôd za ovaj test nalazi se u <src/seleniumCode/Test4.jpg>.

Swappet



 m / S...



<> Code Issues Pull requests Actions Projects Wiki Security 11

7. Tehnologije za implementaciju aplikacije

Edit

New page

Ivan Vjekoslav Rođak edited this page 46 minutes ago · [9 revisions](#)

Korištene tehnologije i alati

1. Programski jezici:

- Java 21
- JavaScript 16.13
- HTML5

Java je robusni, objektno orijentirani jezik idealan za poslovne aplikacije, izradu serverske logike te je također pogodan za skalabilne sustave. JavaScript omogućuje dinamičnu, interaktivnu logiku na klijentskoj strani, pružajući korisnicima dobro iskustvo korištenja web aplikacije. HTML je temelj za strukturiranje web-stranica, koji je gotovo pa norma u prikazu i organizaciji sadržaja u preglednicima.

2. Radni okviri i biblioteke:

- React 18.3
- Node.js 20.1
- Spring Boot 3.4.2

▼ Pages 13

- ▶ [Home](#)
- ▶ [1. Opis projektnog zadatka](#)
- ▶ [2. Analiza zahtjeva](#)
- ▶ [3. Specifikacija zahtjeva sustava](#)
- ▶ [4. Arhitektura i dizajn sustava](#)
- ▶ [5. Arhitektura komponenata i razmj...](#)
- ▶ [6. Ispitivanje programskog rješenja](#)
- ▼ [7. Tehnologije za implementaciju a...](#)
 - Korištene tehnologije i alati
- ▶ [8. Upute za puštanje u pogon](#)
- ▶ [9. Zaključak i budući rad](#)
- ▶ [A. Popis literature](#)
- ▶ [B. Dnevnik promjena dokumentacije](#)
- ▶ [C. Prikaz aktivnosti grupe](#)

[+ Add a custom sidebar](#)

React je popularan frontend okvir koji omogućuje stvaranje dinamičnih korisničkih sučelja. Njegova virtual DOM tehnologija osigurava brže performanse i lakšu manipulaciju prikaza korisničkog sučelja. Node.js omogućuje pokretanje JavaScript-a na serverskoj strani, pružajući brz i skalabilan način za rukovanje zahtjevima. Verzija 20.1 donosi sigurnosna i performansna poboljšanja, koja osiguravaju dulju upotrebljivost sustava. Spring Boot je radni okvir za programski jezik Java koji omogućuje programiranje servera („backend“).

3. Baza podataka:

- PostgreSQL 16
- korištena relacijska baza
- način pristupa: USER "pone" WITH PASSWORD '*****' sa svim privilegijama

PostgreSQL je besplatna baza podataka, otvorenog koda koja nudi provjeren i optimiziran alat za skladištenje podataka. Verzija 16 poboljšava performanse i sigurnost, osiguravajući pouzdano čuvanje podataka.

4. Razvojni alati:

- IntelliJ IDEA 2024.2.3
- VS Code
- Eclipse
- pgAdmin 4
- Git 2.39

Clone this wiki locally

<https://github.com/monamihokovic>



IntelliJ je napredni IDE s puno značajki poput automatskog dovršavanja koda i integriranih alata za ispravljanje grešaka, primarno u našem projektu korišten za backend tj. Spring Boot. Visual Studio Code je jednostavan, ali moćan editor s podrškom za brojne jezike, stoga smo ga koristili pri izmjeni dijelova koda koji nisu pisani u Javi. Eclipse je IDE, poznat po prilagodljivosti i podršci za velike projekte, stoga smo ga primarno koristili za razvoj frontenda, tj. React „radnog okvira“. PgAdmin 4 je besplatan alat za administraciju PostgreSQL baza podataka koji pruža napredne alate za vizualizaciju i upravljanje podacima. Git je standardni sustav za verzioniranje koji omogućuje timovima učinkovitu suradnju i praćenje promjena u kodu, što je nama kao timu također olakšalo dijeljenje koda.

5. **Alati za ispitivanje:** Jedinični, integracijski ili UI ispitni slučajevi

- Selenium 4.0, Playwright

Selenium je moderan i besplatan alat za automatizirano testiranje web-aplikacija koji olakšava testiranje, te čak nudi opcije testiranje u kojima korisnik ne mora napisati niti liniju koda već samo klikati na sučelju.

6. **Alati za razmještaj:**

- Docker 20.10

Docker omogućuje stvaranje, distribuciju i pokretanje aplikacija unutar kontejnera, osiguravajući dosljedno okruženje.

7. **Cloud platforma:**



- Render









Render je jednostavna, besplatna, ali moćna platforma za hosting aplikacija, omogućujući brzo postavljanje i skaliranje uz minimalnu konfiguraciju.

Korišteni alati i tehnologije odabrani su zbog preporuka nositelja predmeta ili prethodnog iskustva u njima.

Swappet



 m / S...



<> Code Issues Pull requests Actions Projects Wiki Security 11

8. Upute za puštanje u pogon

Edit

New page

Ivan Vjekoslav Rođak edited this page 50 minutes ago · [17 revisions](#)

1. Instalacija

• Preuvjeti:

- Java 17 ili novije
- Node.js 20.14.0 ili novije
- PostgreSQL 16 ili novije
- Git 2.30 ili novije
- Docker 27.3.1 ili novije

- **Preuzimanje:** Klonirati Git repozitorij korištenjem sljedeće naredbe

```
git clone
https://github.com/monamihokovic/Swappet.git
```

```
cd Swappet
```

Instalacija ovisnosti:

```
npm install
```

2. Postavke

- **Konfiguracijske datoteke:** Treba namjestiti .env datoteku (IzvorniKod/Swappet/frontend/.env) ovisno pokreće li se aplikacija u lokalnom ili deployanom okruženju. Za backend ne treba ništa ručno postavljati već se automatski prebacuje kad Render overridea vrijednost spremljenih varijabli.

▼ Pages 13

▶ [Home](#)

▶ [1. Opis projektnog zadatka](#)

▶ [2. Analiza zahtjeva](#)

▶ [3. Specifikacija zahtjeva sustava](#)

▶ [4. Arhitektura i dizajn sustava](#)

▶ [5. Arhitektura komponenata i razmj...](#)

▶ [6. Ispitivanje programskog rješenja](#)

▶ [7. Tehnologije za implementaciju a...](#)

▼ [8. Upute za puštanje u pogon](#)

Opis pristupa aplikaciji na javnom poslužitelju

▶ [9. Zaključak i budući rad](#)

▶ [A. Popis literature](#)

▶ [B. Dnevnik promjena dokumentacije](#)

▶ [C. Prikaz aktivnosti grupe](#)

[+ Add a custom sidebar](#)

- Lokalno:

```
REACT_APP_BACKEND_URL=http://localhost:8081
```

- Deployano:

```
REACT_APP_BACKEND_URL=https://swappet-backend.onrender.com
```

- **Postavke baze podataka:**

- Inicijalizirati bazu podataka pokretanjem SQL skripte
(Dokumentacija/baza/strater_packovi/potpuna_baza_sve_unutra)

```
npm run db:migrate
```

```
npm run db:seed
```

3. Pokretanje aplikacije

- **Razvojno okruženje:**

- Frontend:

```
cd frontend npm run start
```

- Backend:

```
cd backend  
./mvnw spring-boot:run
```

- **Produksijsko okruženje:**

- Prevođenje aplikacije:

```
npm run build
```

- **Pokretanje poslužitelja:**

```
npm start
```

- **Provjera rada:** <https://swappet-app-iod2.onrender.com>

Clone this wiki locally

```
https://github.com/monamihokovic
```



4. Upute za administratore

Smjernice za administratore aplikacije nakon puštanja u pogon:

- **Pristup administratorskom sučelju:**
 - administrator se treba ulogirati pomoću svojeg računa kao svaki drugi korisnik
 - na vrhu stranice nalazi se gumb „Admin usluge“ koji se koristi za pristup administratorskom sučelju
 - gumb „Admin usluge“ će se pojaviti samo korisnicima koji su u bazi podataka označeni kao administratori
 - samo administrator može učiniti drugog korisnika administratorom (na početku je zadan inicijalni administrator)
- **Redovito održavanje:**
 - Arhiviranje baze podataka.
 - Pregled logova.
 - Ažuriranje aplikacije:
 - povući novu verziju iz Git repozitorija
 - pokrenuti workflowove iz Github actionsa za rebuildanje frontend i backend image-a
 - nakon uspješnog builda Render će automatski redeployati ažuriranu aplikaciju
- **Rješavanje problema:** Kako pristupiti logovima i dijagnosticirati greške (npr. logs/error.log ili docker logs).

5. Primjer za Render platformu (Cloud Deploy)

Render je popularna cloud platforma za jednostavno smještanje aplikacija.

- **Priprema repozitorija:**

- Osigurajte da vaš projekt ima datoteku `render.yaml` ili `Dockerfile` za konfiguraciju deploja.
- Primjer `render.yaml`:

```
services:  
  
- type: web  
  
name: my-web-app  
  
env: node  
  
buildCommand: npm install && npm run build  
  
startCommand: npm start  
  
plan: free
```

- **Postavljanje na Render:**

- Prijavite se na [Render](#).
- Kreirajte novi **Web Service** i povežite ga s vašim GitHub repozitorijem (i za frontend i za backend).
- Kreirajte novu PostgreSQL bazu podataka
- Konfigurirajte postavke (npr. build i start komande).
- Dodajte environment varijable:
 - Za frontend:
 - `REACT_APP_BACKEND_URL`
 - Za backend:
 - `DB_DRIVER`

- DB_PASS
 - DB_URL
 - DB_USERNAME
 - FRONTEND_URL
 - MAIL_PASSWORD
 - MAIL_USERNAME
 - OAUTH_REDIRECT_URI
 - SPRING_JPA_DATABASE_PLATFORM
 - SPRING_PROFILES_ACTIVE
- **Pokretanje aplikacije:**

Render će automatski preuzeti repozitorij, instalirati ovisnosti i pokrenuti aplikaciju. Nakon deploja, aplikaciji možete pristupiti putem generiranog URL-a (u ovom slučaju <https://swappet-app-iod2.onrender.com>).

Opis pristupa aplikaciji na javnom poslužitelju

Pristup aplikaciji

- Otvorite URL aplikacije u pregledniku (<https://swappet-app-iod2.onrender.com>)
- Ulogirajte se pomoću Google računa ili nastavite kao gost
- Pristup administratorskom sučelju opisan u 4. odjeljku


Ograničenja:

- Besplatan plan na Renderu može uzrokovati privremenu nedostupnost aplikacije zbog hibernacije
- Vrijeme podizanja aplikacije nakon hibernacije traje do 2 minute

Swappet




☰

 m / S...

🔍

Type / to search




+

🕒

🔗

📁



<> Code

🔖 Issues

🔗 Pull requests

🔄 Actions

📁 Projects

📖 Wiki

🛡 Security 11

9. Zaključak i budući rad

Edit

New page

Dominik Mandić edited this page 40 minutes ago · [5 revisions](#)

Projektni zadatak grupe Swappet, u sklopu predmeta [Programsko inženjerstvo](#) je bila izrada stranice za razmjenu ili kupnju karata za događanja. Kroz projekt nas je vodila potreba korisnika za platformom na kojoj bi mogli drugim korisnicima ponudi krivo kupljene karte, ili karte događaja kojima iz raznih razloga ne mogu sudjelovati. Projekt je trajao 14 tjedana, uz dva ocjenjivanja (u 7 tjednu i 14 tjednu).

Tijekom provedbe susreli smo se s nizom problema koje smo podijeli na 3 glavne kategorije:

1) **Timski rad** (držanje rokova, vremenska zavisnost, ...)

Timski rad je tijekom projekta bio jedan od najizazovnijih dijelova. Posebno su se isticali problemi u izvedbi dijelova programa koji su međusobno visoko povezani, a rade ih dvije ili više osoba. Takvi dijelovi projekta su iziskivali iznimno jasnu i preciznu komunikaciju te usklađenost vremenskih rokova. U početku projekta nismo imali dovoljno preciznu komunikaciji koja je također lako vidljiva. No, kako je projekt napredovao počeli smo koristiti precizniju komunikaciju ali i dokumentaciju u kojoj smo jasno i precizno zapisivali informacije potrebne drugima za njihov dio posla.

▼ Pages 13

Find a page...

▶ Home

▶ 1. Opis projektnog zadatka

▶ 2. Analiza zahtjeva

▶ 3. Specifikacija zahtjeva sustava

▶ 4. Arhitektura i dizajn sustava

▶ 5. Arhitektura komponenata i razmj...

▶ 6. Ispitivanje programskog rješenja

▶ 7. Tehnologije za implementaciju a...

▶ 8. Upute za puštanje u pogon

9. Zaključak i budući rad

▶ A. Popis literature

▶ B. Dnevnik promjena dokumentacije

▶ C. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

2) Tehnički (implementacijski) (funkcionalnosti komponenti, povezivanje komponenti, testiranje, ...)

<https://github.com/monamihokovic>

Implementacija je predstavljala problem zbog slabe upoznatosti članova tima s radnim okvirom Spring boot, koji je naveden kao nužan za izvođenje projekta. Većinu vremena prvih 4-5 tjedana projekta potrošeno na upoznavanje i namještanje radnog okvira Spring boot na backend-u i radnog okvira React na frontend-u. Sazrijevanjem projekta, naglasak problema se pomaknuo na interakciju i suradnju raznih dijelova programske podrške. Kako bi tome doskočili organizirali smo zajedničke sastanke u kojima bi promotrili problem koji se javio, iz njegova izvorišta (primjerice baze podataka) i odredišta problema u kojem se on jasno vidio (primjerice ne postojanje neke vrijednosti u krajnjem prikazu na frontendu).

3) Vanjske kompatibilnosti (deploy, GoogleOauth, ...)

Problem vanjskih kompatibilnosti je pokazao izražen, ali općenit problem, ne pretjerano jasnih uputa i dokumentacije vanjskih alata. Unutra projektnog zadatka smo trebali koristiti više vanjskih alata, od kojih je većina imala više verzija, od kojih je ona besplatna bila najstriktnija za implementaciju i trebali smo prilagođavati rad naše aplikacije za rad iste. No, kada smo ih implementirali i naučili sve posebnosti tog servisa njegova kasnija primjena je bila uspješna.

Projekt je bio podijeljen na dva dijela pisanje dokumentacije i programskog koda. Prvih 7 tjedana projekta je bilo više posvećeno pisanju dokumentacije te osmišljavanju arhitekturnog rješenja programske potpore. Ostatak vremena je bio ponajviše utrošen na implementaciju i testiranje programskog koda. Tijekom projekta smo izbrusili svoja znanja korištenja CSS-a, Jave, SQL-a te oblikovanja baze podataka. Nova znanja smo dobili iz područja dokumentiranja programske potpore, vođenja i rada u grupama te rada s radnim okvirima React i Spring Boot. Također kako je ovaj projekt bio prvi veći projekt većini članova, dobili smo značajno iskustvo u povezivanju dijelova aplikacije. Projekt je mogao biti brže i kvalitetnije uz veća znanja korištenja radnih okvira i realnije i preciznije postavljanje rokova za izvršavanje zadataka.

Daljnji rad na projektu bi se mogao primarno realizirati kroz dovršavanje funkcionalnosti koje nisu izvedene:

- lanaca razmjene (izvedena baza i logika za cijelu obradu vezanih odluka)
- vanjskog servisa za prikaz informacija o izvođačima

Aplikacija bi se mogla još više prilagoditi korisnicima te im pružiti još bolje iskustvo uvođenjem nekih inovativnih rješenja kroz:

- chatboot za informacije o trenutnim oglasima
- sustav autentifikacije pomnuđenih ulaznica sa organizatorom događaja
- sustavom za procjenu cijene karata pri prodaji
- implementiranjem sustava plaćanja unutar web-stranice
- specijaliziranom aplikacijom za mobilne uređaje (pametne telefone)

Swappet



m / S...

Q

Type / to search

+

<>

Code

⦿

Issues

🔗

Pull requests

▶

Actions

📁

Projects

📖

Wiki

🛡️

Security

11

A. Popis literature

Edit

New page

Patrick Mraz edited this page 8 hours ago · [4 revisions](#)

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. The Unified Modeling Language, <https://www.uml-diagrams.org/>
3. Astah Community, <http://astah.net/editions/uml-new>
4. ERDPlus, <https://erdplus.com>
5. Genuine Coder, <https://youtu.be/tlaWE9WthSQ?si=HKAdRnXU3eAkHEzF>
6. Code with Arjun, https://youtu.be/O_XL9oQ1_To?si=5BhAqaRLGeGF8Vo5
7. Amigoscode, <https://www.youtube.com/watch?v=9SGDpanrc8U&t=2717s>
8. Github, <https://docs.github.com/en/get-started/start-your-journey/git-and-github-learning-resources>
9. Postman, <https://www.postman.com>
10. freeCodeCamp.org, <https://www.youtube.com/watch?v=31KTdfRH6nY>
11. Stack Overflow, <https://stackoverflow.com>
12. Online Visual Paradigm <https://online.visual-paradigm.com>
13. <https://www.britannica.com/technology/client-server-architecture> (pristup 15.1.2025)
14. <https://cs.uwaterloo.ca/~m2nagapp/courses/C>

▼ Pages 13

Find a page...

▶ Home

▶ 1. Opis projektnog zadatka

▶ 2. Analiza zahtjeva

▶ 3. Specifikacija zahtjeva sustava

▶ 4. Arhitektura i dizajn sustava

▶ 5. Arhitektura komponenata i razmj...

▶ 6. Ispitivanje programskog rješenja

▶ 7. Tehnologije za implementaciju a...

▶ 8. Upute za puštanje u pogon

▶ 9. Zaključak i budući rad

A. Popis literature

▶ B. Dnevnik promjena dokumentacije

▶ C. Prikaz aktivnosti grupe

+

 Add a custom sidebar

Clone this wiki locally

[S446/1195/Arch_Design_Activity/ClientServer.pdf](#) (pristup 15.1.2025)

<https://github.com/monamihokovic>



15. <https://medium.com/nerd-for-tech/client-server-architecture-explained-with-examples-diagrams-and-real-world-applications-407e9e04e2d1> (pristup 12.11.2024)
16. [https://www.fer.unizg.hr/_download/repository/UML_zadaci_za_vjezbu_-_nadopuna_sveucilisnog_prirucnika\[1\].pdf](https://www.fer.unizg.hr/_download/repository/UML_zadaci_za_vjezbu_-_nadopuna_sveucilisnog_prirucnika[1].pdf) (pristup 23.1.2025)
17. Weather Api, <https://www.weatherapi.com/>

Swappet



m / S...

Q

Type / to search

+

<>

CodeIssuesPull requestsActionsProjectsWikiSecurity

11

B. Dnevnik promjena dokumentacije

Edit

New page

Dominik Mandić edited this page 37 minutes ago · [14 revisions](#)

Rev.	Opis promjene/dodatka	Autori	Datumi	<div>▼ Pages 13</div> <div><div>Find a page...</div><div><div>▶ Home</div><div>▶ 1. Opis projektnog zadatka</div><div>▶ 2. Analiza zahtjeva</div><div>▶ 3. Specifikacija zahtjeva sustava</div><div>▶ 4. Arhitektura i dizajn sustava</div><div>▶ 5. Arhitektura komponenata i razmj...</div><div>▶ 6. Ispitivanje programskog rješenja</div><div>▶ 7. Tehnologije za implementaciju a...</div><div>▶ 8. Upute za puštanje u pogon</div><div>▶ 9. Zaključak i budući rad</div><div>▶ A. Popis literature</div><div>B. Dnevnik promjena dokumentacije</div><div>▶ C. Prikaz aktivnosti grupe</div></div><div><div>+ Add a custom sidebar</div></div><div>Clone this wiki locally</div></div>
0.1	Stvorena dokumentacija i unesene početne informacije	Mona	30.10.	
0.2	Minimalna promjena strukture	Ivan	2.11.	
0.3	Unesen predložak	Paško	5.11.	
0.4	Analiza zahtjeva = preliminarne verzija, Opis projektnog zadatka	Paško, Dominik	10.11.	
0.5	Specifikacija zahtjeva sustava = preliminarne verzija	Paško	11.12.	
0.6	Arhitektura i dizajn sustava = početno unošenje	Dominik, Mona, Goran	11.12.	
0.7	Implementacija i korisničko sučelje = početno unošenje	Ivan, Maja, Patrick	12.12.	
	Specifikacija			

0.8	zahtjeva sustava = početno unošenje	Paško	13.12
0.9	Završno unošenje promjena i priprema za 1 predaju	Dominik, Paško, Ivan, Mona	15.12
1.0	Prva predaja = sve pripremljeno	svi	16.12
1.1	Prepravak baze podataka, zbog novih tablica	Dominik	26.12
1.2	Dodana prva verzija dijagrama stanja i aktivnosti	Dominik	26.12
1.3	Dodana druga verzija dijagrama stanja i aktivnosti	Dominik	10.1.
1.4	Specifikacija zahtjeva sustava popravljivanje postojećih i dodavanje novih UC dijagrama	Paško	12.1.
1.5	Prepravak baze podataka, zbog novih tablica	Dominik	15.1.
1.6	Prepravak Analize zahtjeva te dodavanje sekvencijskih dijagrama	Paško	17.1.
2.0	Preuzet novi predložak	Dominik	17.1.
2.1	Završena reorganizacija u	Dominik,	21.1.

<https://github.com/monamihokovic>


	skladu s novim predloškom	Mona	
2.2	Popunjena nova Arhitektura i dizajn sustava	Dominik, Goran	21.1.
2.3	Popunjena Tehnologije za implementaciju aplikacije	Mona	21.1.
2.4	Popunjene Upute za puštanje u pogon	Goran	21.1.
2.5	Popravljanje funkcionalnih zahtjeva i aktera	Mona	23.1.
2.6	Popunjena Arhitektura komponenata i razmještaja	Dominik	23.1.
2.7	Popravljanje uključenosti ključnih funkcionalnosti u obrascima uporabe zbog promjene funkcionalnih zahtjeva	Mona	23.1.
2.8	Popunjen Zaključak i budući rad	Dominik	24.1.
2.9	Popunjeno Ispitivanje programskog rješenja	Paško	24.1.

Swappet



m / S...

Q

Type / to search

+

<>

CodeIssuesPull requestsActionsProjectsWikiSecurity

11

C. Prikaz aktivnosti grupe

Edit

New page

Dominik Mandić edited this page 30 minutes ago · [15 revisions](#)

Dnevnik sastajanja

1. sastanak

- Datum: 19. listopada 2024.
- Prisustvovali: Mona Mihoković, Patrick Mraz, Maja Blažok, Goran Torbica, Ivan Vjekoslav Rođak
- Teme sastanka:
 - Pregled funkcionalnih i nefunkcionalnih zahtjeva
 - TODO: urediti GitHub, napraviti Discord server

2. sastanak

- Datum: 31. listopada 2024.
- Prisustvovali: Mona Mihoković, Patrick Mraz, Maja Blažok, Goran Torbica, Dominik Mandić, Ivan Vjekoslav Rođak
- Teme sastanka:
 - rute
 - revizija baze
 - planovi za daljnji rad i rokovi

3. sastanak

- Datum: 2. prosinca 2024.
- Prisustvovali: Mona Mihoković, Patrick Mraz,

▼ Pages13

Find a page...

▶ Home

▶ 1. Opis projektnog zadatka

▶ 2. Analiza zahtjeva

▶ 3. Specifikacija zahtjeva sustava

▶ 4. Arhitektura i dizajn sustava

▶ 5. Arhitektura komponenata i razmj...

▶ 6. Ispitivanje programskog rješenja

▶ 7. Tehnologije za implementaciju a...

▶ 8. Upute za puštanje u pogon

▶ 9. Zaključak i budući rad

▶ A. Popis literature

▶ B. Dnevnik promjena dokumentacije

▼ C. Prikaz aktivnosti grupe

Dnevnik sastajanja

Plan rada

Tablica aktivnosti

Dijagram pregleda promjena

Maja Blažok, Goran Torbica, Dominik Mandić,
Ivan Vjekoslav Rođak

- Teme sastanka:
 - roadmap za daljnji rad na projektu
 - podjela poslova unutar grupe

4. sastanak

- Datum: 3. siječnja 2025.
- Prisustvovali: Mona Mihoković, Dominik Mandić, Ivan Vjekoslav Rođak
- Teme sastanka:
 - razmjena ulaznica
 - lanci razmjene

5. sastanak

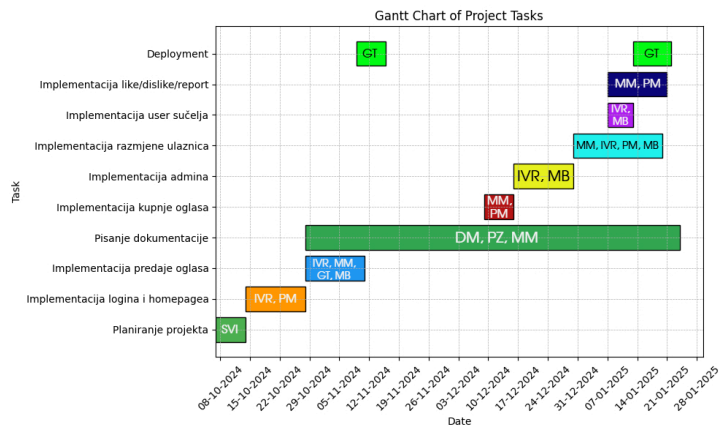
- Datum: 17. siječnja 2025.
- Prisustvovali: Mona Mihoković, Patrick Mraz, Maja Blažok, Dominik Mandić, Ivan Vjekoslav Rođak
- Teme sastanka:
 - zajedničko debuggiranje aplikacije
 - dodavanje još nekih funkcionalnosti
 - planovi za daljnji rad i rokovi

+ Add a custom sidebar

Clone this wiki locally

https://github.com/monamihokovic

Plan rada



Tablica aktivnosti

Kontinuirano osvježavanje

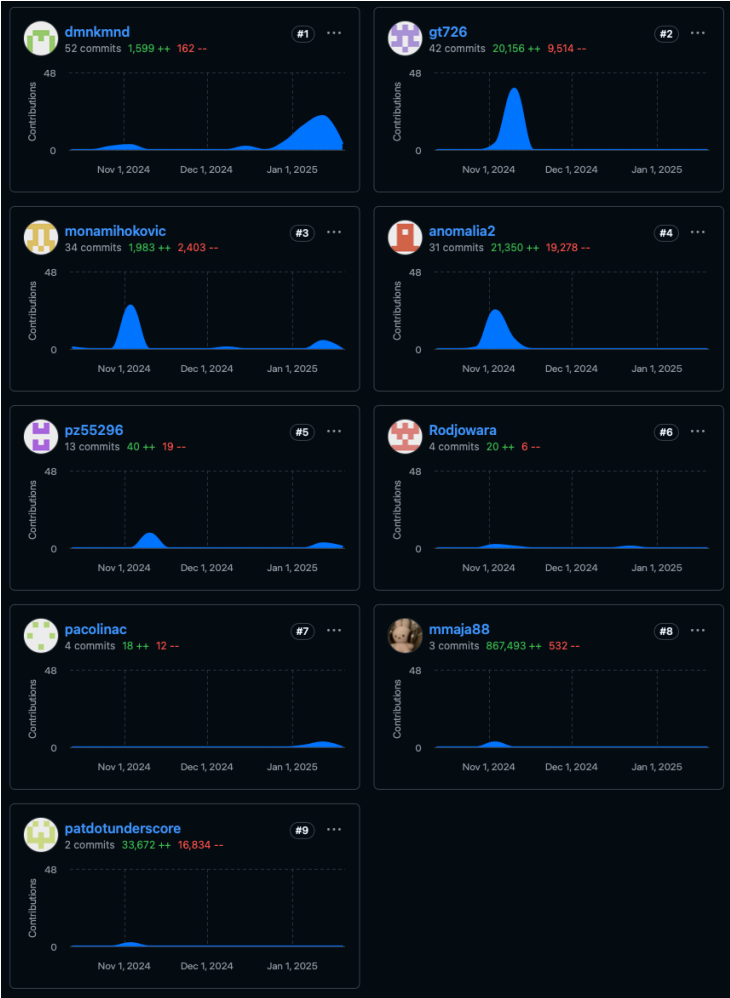
Napomena: Doprinosi u aktivnostima treba navesti u satima po članovima grupe po aktivnosti. Potrebno je navesti koliko je sati koja osoba uložila u pojedinu komponentu, možete oblikovati tablicu ili ispisati za svaku osobu.

Komponenta	Ivan Vjekoslav Rođak	Goran Torbica	Mc Miho
Upravljanje projektom	20h	5h	5h
Opis projektnog zadatka			
Funkcionalni zahtjevi			2h
Opis pojedinih obrazaca			
Dijagram obrazaca			1h
Sekvencijski dijagrami			
Opis ostalih zahtjeva			
Arhitektura i dizajn sustava	8h	2h	2h
Baza podataka			
Dijagram razreda			

Dijagram stanja			
Dijagram aktivnosti			
Dijagram komponenti			
Rad na backendu	50h	13h	20h
Ogledni dizajn aplikacije			
Rad na frontendu			
Izrada baze podataka			
Izrada triggera u bazi podataka			
Ispitivanje programskog rješenja	35h	10h	6h
Dijagram razmještaja			
Deployment		21h	
Upute za puštanje u pogon		1h	
Dnevnik sastajanja	1h		
Zaključak i budući rad	3h		
GitHub i dokumentacija			12h
Izrada			

prezentacije projekta			3h
--------------------------	--	--	----

Dijagram pregleda promjena



Swappet