



HoodClassics je web aplikacija osmišljena za sve one koji žele dublje upoznati mjesto koje posjećuju - kroz lokalne priče, legende i skrivena značenja koja ga čine posebnim. Zamislite da istražujete grad kroz digitalnu kartu s pinovima: svaki pin predstavlja priču, legendu ili zanimljivost koju su podijelili drugi korisnici. Kada kliknete na pin, otvara vam se objava s informacijama i pričama vezanim za tu lokaciju, pružajući vam autentičan pogled na lokalni život i kulturu.

Aplikacija omogućuje registriranim korisnicima, turistima i lokalnim stanovnicima, da doprinesu sadržaju - lokalci dijele priče o poznatim grafitima, povjesnim građevinama, ili omiljenim mjestima koja možda ne biste našli u klasičnim vodičima. Svojim doprinosima obogaćuju iskustvo svih posjetitelja, čineći HoodClassics dinamičnom zajednicom u kojoj je svaki doprinos priča s dušom.

Vrste korisnika: * Turisti i lokalci: Turisti mogu pregledavati i prijavljivati sadržaj, dok lokalci stvaraju objave i dijele priče o mjestima koja poznaju. Lokalci mogu označiti više lokacija kao svoj dom i dijeliti informacije o svom susjedstvu. *

Moderatori: Brinu o kvaliteti sadržaja pregledom prijavljenih objava te odlučuju o njihovom zadržavanju ili uklanjanju. Aplikacija također nudi pretragu po tagovima za lakše istraživanje specifičnih tema, poput sporta ili ulične umjetnosti. Tako korisnici mogu filtrirati sadržaj i pronaći mjesta koja odgovaraju njihovim interesima - recimo, stadion, mural ili legendarno mjesto vezano za nogomet.

Potencijalna korist ovog projekta

Aplikacija HoodClassics ima nekoliko potencijalnih koristi koje mogu unaprijediti turističko iskustvo i jačati lokalne zajednice:

Povećava autentičnost turističkog doživljaja: HoodClassics omogućuje turistima da dožive destinacije na dublji, autentičan način. Umjesto klasičnih informacija, turisti dobivaju pristup lokalnim pričama, legendama i povijesti, što im pomaže da bolje razumiju identitet i kulturu mjesta koje posjećuju.

Podržava lokalne zajednice i kulturu: Kroz objave koje stvaraju lokalci, aplikacija čuva i prenosi specifične priče i lokalne znamenitosti koje nisu uvijek poznate široj javnosti. Ovakav način dijeljenja lokalne povijesti i tradicije može povećati ponos zajednice i privući pažnju na mesta i događaje koji možda nisu dio mainstream turizma.

Pomaže u boljoj organizaciji putovanja: Korištenjem tagova i filtara, korisnici mogu lakše pronaći točno ono što ih zanima - npr., ljubitelji sporta mogu otkriti stadione ili murale povezane s lokalnim klubovima, dok zaljubljenici u umjetnost mogu saznati više o grafitima i uličnim umjetnicima. Pruža sigurno i korisno iskustvo kroz moderiranje: Moderatorima aplikacija omogućava provjeru sadržaja, čime se smanjuje rizik od širenja lažnih informacija i omogućuje korisnicima da uživaju u kvalitetnim i provjerenim pričama.

Potencijal za ekonomski doprinos: Promovirajući lokalne atrakcije i manje poznata mjesta, aplikacija može doprinijeti distribuciji turističkog prometa prema manje posjećenim dijelovima grada ili regije, čime lokalne zajednice mogu imati finansijsku korist kroz povećan interes posjetitelja. Kombinacijom autentičnog sadržaja, angažmana zajednice i sigurnosnih mjera, HoodClassics ima potencijal postati moćan alat za turističke agencije, lokalne vlasti i putnike koji traže dublje povezanosti s destinacijama koje posjećuju.

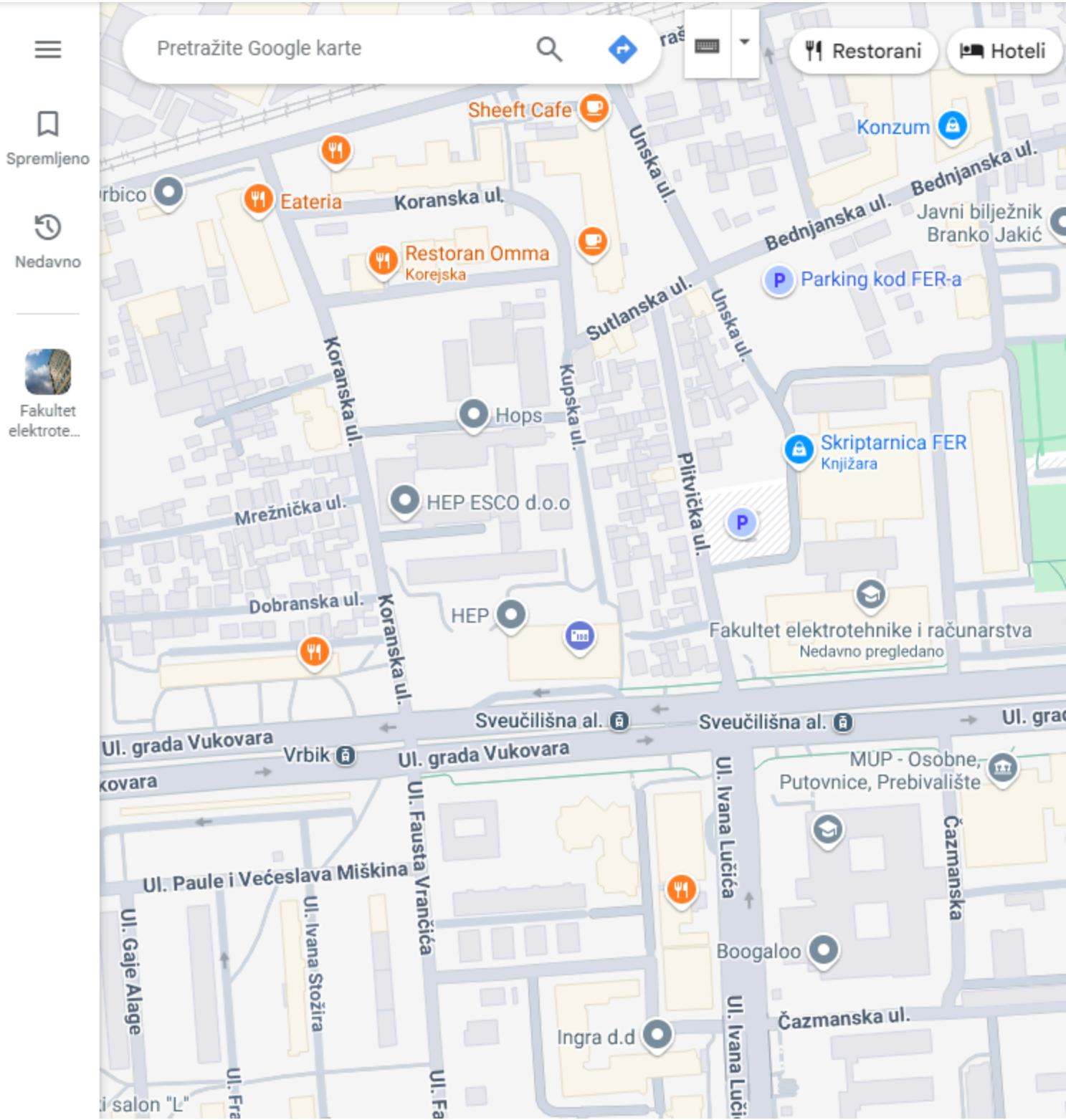
Postojeća slična rješenja

Google Maps

Najsličniji proizvod našoj aplikaciji jest Google Maps. To je popularna web aplikacija koja nudi brojne funkcionalnosti za navigaciju, pretraživanje lokacija i planiranje putovanja. Omogućuje detaljne upute za vožnju, hodanje, bicikliranje i javni prijevoz, uz podatke o prometu i gužvama. Nudi pretraživanje mjesta kao što su restorani, trgovine i bolnice, uz recenzije i radno vrijeme. Također uključuje 360° Street View, prikaz ulica, offline karte, i mjerjenje udaljenosti između točaka.

Google Maps također prikazuje popularne turističke destinacije, kulturne znamenitosti i atrakcije što je funkcionalnost najbliskija našoj aplikaciji. Korisnici mogu pretraživati muzeje, povjesne spomenike, parkove i druge zanimljive lokacije, uz recenzije, fotografije i detalje poput radnog vremena i cijena ulaznica. Aplikacija nudi i preporuke za obližnje atrakcije te vodiče za turističke rute, što pomaže korisnicima u planiranju izleta. Naša se aplikacija razlikuje po tome što ima fokus na lokalne priče i legende. Za razliku od Google Mapsa, koji prvenstveno služi navigaciji i pronalaženju općih informacija o lokacijama (kao što su restorani, hoteli, znamenitosti), HoodClassics omogućava korisnicima da istraže kulturni i povjesni kontekst mjesta kroz priče koje dijele lokalci. Aplikacija se oslanja na lokalne korisnike za stvaranje sadržaja, dok Google Maps objavljuje recenzije i ocjene za lokacije. Također, aplikacija omogućuje korisnicima pretragu kroz tagove, što je korisno za filtriranje sadržaja prema interesima.

Slika 1.1: Google Maps



Kao što je vidljivo na slici 1.1, Google Maps je interaktivna karta sa pinovima kao što će biti i naš proizvod, ali je Google Maps ograničen na pregledavanje područja i pretraživanje ustanova kao što su restorani i hoteli. Cilj naše aplikacije jest pridonijeti pretraživanju karte novu dimenziju predstavivši među-korisničku interakciju, spajajući turiste sa lokalnom zajednicom.

Trip Advisor

Još jedna popularna turistička aplikacija je TripAdvisor. To je web platforma koja korisnicima pruža informacije za planiranje putovanja, uključujući recenzije, ocjene i preporuke o hotelima, restoranima, znamenitostima i turističkim aktivnostima širom svijeta. TripAdvisor je namijenjen korisnicima koji traže detaljne informacije i preporuke od drugih putnika kako bi lakše isplanirali putovanje i otkrili najbolje aktivnosti u destinaciji. HoodClassics i TripAdvisor imaju sličnu osnovnu funkciju – pomažu korisnicima da otkriju lokacije i aktivnosti u destinacijama, ali se razlikuju u nekoliko ključnih aspekata kao što su vrsta i stvaranje sadržaja te funkcija filtriranja po temama sadržaja. Dok TripAdvisor pruža tradicionalne informacije za planiranje putovanja, temeljen na recenzijama i ocjenama, HoodClassics nudi dublje i personalizirane lokalne priče, koje pomažu turistima da se povežu s destinacijom na autentičniji način.

Slika 1.2: Trip Advisor

Essential Zagreb

Pick a category to filter your recs

The screenshot shows the TripAdvisor interface for 'Essential Zagreb'. At the top, there are five filter buttons: 'Essentials' (selected), 'Travelers' Choice', 'Family friendly', 'Hidden gems', and 'Museums'. Below these, a section titled 'Things to do' is displayed. Two items are shown: 'Tkalčićeva Street' and 'Upper Town Gornji Grad'. Each item has a thumbnail image, a title, a rating icon, a numerical rating, and a category tag.

Attraction	Image	Title	Rating	Category
Tkalčićeva Street		Tkalčićeva Street	4.5 stars (1,919 reviews)	Historic Walking Areas, Points of Interest & Landmarks
Upper Town Gornji Grad		Upper Town Gornji Grad	4.5 stars (2,269 reviews)	Historic Walking Areas, Neighborhoods

Tkalčićeva Street

4.5 1,919

Historic Walking Areas, Points of Interest & Landmarks

Upper Town Gornji Grad

4.5 2,269

Historic Walking Areas, Neighborhoods

Slika 1.2 predstavlja sučelje Trip Advisora kada potražimo grad Zagreb na stranici. Naša će aplikacija, baš kao i Trip Advisor, imati tagove po kojima će se moći pretraživati sadržaj, ali ti tagovi će biti specifičniji i više orijentirani na tematsko filtriranje.

Skup korisnika zainteresiran za ostvareno rješenje

Skup korisnika koji bi mogao biti zainteresiran za aplikaciju HoodClassics uključuje raznolike grupe, jer aplikacija nudi jedinstveno iskustvo istraživanja kroz lokalne priče i kulturne informacije. Prva skupina su turisti, koji putuju i žele dublje razumjeti lokalnu kulturu, povijest i tradicije. Ovi korisnici traže autentična iskustva i žele naučiti više o mjestima koja posjećuju iz perspektive lokalaca, a ne samo kroz turističke vodiče.

Druga skupina su lokalni stanovnici, koji žele podijeliti svoje priče, povijest i legende s turistima i drugim korisnicima. Oni mogu koristiti aplikaciju kao način za očuvanje lokalne kulture i tradicije, a istovremeno se angažirati u zajednici i sudjelovati u oblikovanju imidža svog grada ili naselja.

Opseg projektnog zadatka

Opseg projektnog zadatka obuhvaća sve tehničke, dizajnerske, funkcionalne i sigurnosne aspekte, kao i implementaciju sustava za korisničku interakciju, modifikaciju sadržaja i moderaciju.

- *Izrada korisničkog sučelja (UI):* Aplikacija će imati intuitivno korisničko sučelje koje omogućava lako kretanje kroz nju. Sučelje će prikazivati kartu, pinove koji označavaju specifične lokacije, te objave povezane uz te lokacije.
- *Interaktivna karta:* Bit će implementirana interaktivna karta s pinovima koji predstavljaju točno određene lokacije. Svaki pin će biti povezan s objavom koja sadrži specifične informacije, poput lokalnih priča, legendi ili kulturnih detalja.
- *Dizajn i razvoj funkcionalnosti za registraciju korisnika:* Postoje tri tipa korisnika aplikacije - registriani, neregistrirani i moderator. Registrirani korisnik može biti turist ili lokalac. Samo lokalac može raditi objave dok i turist i lokalac mogu prijaviti objave za neprimjereni ili netočan sadržaj. Korisnik koji je neregistriran je automatski turist, ali ne može ustupiti u interakciju s objavama (npr. označiti da mu se objava sviđa ili prijaviti objavu). Korisnik lokalac je lokalac na mjestima koja unaprijed označi kao svoj dom, može ih biti više, ne samo jedno.
- *Kreiranje sustava za objave:* Korisnicima lokalcima omogućit će se stvaranje objava vezanih uz specifične lokacije. Objave uključuju tekstualni sadržaj i tagove koji pomažu u opisivanju kulturnih ili povijesnih značajki

mjesta.

- *Implementacija tagova i pretrage:* Aplikacija će omogućiti korisnicima da pretražuju objave prema tagovima kao što su sport, umjetnost, povijest i druge tematske kategorije. Time će korisnici moći filtrirati sadržaj koji ih zanima i otkriti relevantne informacije vezane uz njihove interese.
- *Moderatorski sustav:* Bit će razvijen sustav za moderatora koji će pregledavati prijavljene objave. Moderatori će imati mogućnost odlučiti treba li objavu ukloniti ili odobriti.
- *Provjera sadržaja:* Osim moderatora, aplikacija će imati i druge sustave za osiguravanje uklanjanja neprimjerenog sadržaja s platforme na vrijeme.
- *Testiranje funkcionalnosti:* Bit će provedeno testiranje aplikacije kako bi se osigurala pouzdanost svih funkcionalnosti, uključujući interaktivnost karte, prijavu korisnika, te kreiranje i prikaz objava. Testiranja će se fokusirati na uočavanje i ispravljanje mogućih bugova i problema.

Moguće nadogradnje projektnog zadatka

U budućnosti, aplikaciju HoodClassics možemo nadograditi dodavanjem novih funkcionalnosti koje će dodatno obogatiti korisničko iskustvo i povećati angažman.

Uvođenjem sustava nagrađivanja, korisnici bi mogli biti motivirani za aktivnije sudjelovanje u aplikaciji. Na primjer, korisnici bi mogli biti nagrađeni za stvaranje kvalitetnih objava, komentiranje ili interakciju s postojećim sadržajem. Sortiranje objava prema popularnosti ili stvaranje lista najaktivnijih korisnika (najboljih objavitelja) također bi doprinijelo većoj dinamici i angažmanu unutar aplikacije.

Aplikacija bi mogla omogućiti korisnicima generiranje prilagođenih turističkih ruta temeljenih na interesima i lokacijama koje su označene s pinovima. Kroz suradnju s turističkim agencijama, moglo bi se integrirati plaćene ili sponzorirane ture, čime bi se turistima olakšalo planiranje posjeta s dodatnim informacijama i podrškom.

Cilj izrade projekta

Cilj ovog projekta nije samo stvaranje funkcionalne i inovativne aplikacije, već i učenje i usavršavanje vještina potrebnih za razvoj softvera. Kroz rad na projektu, upoznajemo se s različitim alatima i tehnologijama koji su nam na raspolaganju, što nam omogućava da steknemo praktično iskustvo u njihovoј primjeni. Ovo uključuje rad s bazama podataka, izradu korisničkih sučelja, implementaciju funkcionalnosti te osiguravanje kvalitete i sigurnosti aplikacije.

Također, projekt nam pruža priliku da dobijemo uvid u to kako izgleda rad u timu. Učimo se učinkovitoj kolaboraciji, pri čemu je ključno razumjeti važnost poštene

podjele zadataka i odgovornosti među članovima tima. Rad u timu zahtijeva međusobnu komunikaciju, povjerenje i spremnost da pomognemo kolegama kada se suočimo s izazovima.

Razvijajući aplikaciju, suočavamo se sa stvarnim problemima i zajedno tražimo rješenja, što jača našu sposobnost rješavanja problema, kao i tehničke i socijalne vještine. Svaki član tima ima priliku doprinijeti projektu na svoj način, a zajednički rad omogućuje nam da učimo jedni od drugih i unaprijedimo se u područjima gdje smo manje vješti.

Osim tehničkog znanja, kroz ovaj proces učimo i o dinamici timskog rada, uključujući važnost dobre organizacije, planiranja i međusobne podrške. Ovaj projekt nije samo korak prema profesionalnom razvoju, već i prilika za osobni rast kroz suradnju i zajednički trud.

- Funkcionalni zahtjevi: Opis funkcionalnosti aplikacije (npr., registracija korisnika, prijava slučajeva, pregled statusa pomoći).
- Nefunkcionalni zahtjevi: npr. Zahtjevi vezani uz performanse, sigurnost, skalabilnost i korisničko sučelje.

Funkcionalni zahtjevi

tablica 2.01: funkcionalni zahtjevi | ID zahtjeva | Opis | Prioritet | Izvor | Kriteriji prihvaćanja | | -----

F-01	Sustav omogućuje korisnicima kreiranje računa pomoću korisničkog imena.	Visok	Zahtjev dionika	Korisnik se može registrirati koristeći korisničko ime i lozinku.	F-02	Sustav omogućuje korisnicima prijavu pomoću postojećeg Google računa.
Srednji	Zahtjev dionika	Korisnik se može prijaviti u sustav s postojećim Google računom i koristiti funkcionalnosti aplikacije.	F-03	Sustav omogućuje oporavak lozinke putem e-maila.	Nizak	Zahtjev dionika
Korisnik može zatražiti resetiranje lozinke, primiti poveznicu za resetiranje i uspješno resetirati lozinku.	F-04	Sustav podržava tri različite korisničke uloge: lokalac, turist, moderator.	Visok	Postojeći sustav	Korisnik može imati jednu od tri moguće uloge, pri registraciji, korisnik aplikacije bira mjesto stanovanja. Dok je unutar granica tog mjesta, registrirani korisnik može poprimiti ulogu lokalca.	F-05
Sustav omogućuje korištenje aplikacije bez prijave.	Visok	Povratne informacije korisnika	Korisnik može koristiti aplikaciju bez prijave, u kojem se slučaju implicitno uzima uloga turista koji može koristiti aplikaciju uz određena ograničenja.	F-06	Sustav prikazuje kartu Visok	Postojeći sustav
Sustav se bazira na interaktivnoj karti	F-07	Lokalac može napraviti objavu.	Visok	Zahtjev dionika	Aplikacija mora podržavati mogućnost lokalca da na kartu doda pin koji sadrži objavu.	Objava može sadržavati tekst, sliku i tagove.
	F-08	Aplikacija mora omogućiti filtriranje pinova po tagovima.	Srednji	Zahtjev dionika	Korisnik može označiti objavu sa tagom I filtrirati postaje objave s tim tagovima.	F-09
		Aplikacija omogućava interakciju s objavama registriranim korisnicima.	Visok	Zahtjev		

dionika | Aplikacija mora omogućiti svim registriranim korisnicima da označe objavu sa "sviđa mi se" ili "ne sviđa mi se" i imati mogućnost da prijave neprimjerjen sadržaj i netočne objave. || F-10 | Aplikacija ima korisnike s ulogom moderatora. | Visok | Zahtjev dionika | Moderator može pregledati prijave i ukloniti objave/pin-ove. || F-11 | Sustav mora omogućiti rangiranje objava. | Nizak | Zahtjev dionika | Aplikacija mora omogućiti rangiranje pinova u nekom području po broju ljudi koji su ih označili sa "sviđa mi se". || F-12 | Moderator može brisati objave | Srednji| Postojeći sustav | Aplikacija mora omogućiti moderatoru da obriše bilo koju objavu za koju procjeni da je neprimjerena ili pogrešna. || F-13 | Prikaz pinova na krati | Visok| Postojeći sustav | Aplikacija mora prikazivati dostupne pinove za odabранo područje na karti. Ti pinovi simboliziraju turističke lokacije sa objavama korisnika. || F-14 | Uz objavu je označen jedan od ponuđenih tagova | Srednji| Zahtjev dionika | Korisniku koji izrađuje objavu su ponuđeni unaprijed određeni tagovi kako bi kasnije drugi korisnici lakše našli njihovu objavu. || F-15 | Prijava neprimjerenoog sadržaja | Visok| Zahtjev dionika | Korisnik, ukoliko vidi neprimjereni sadržaj na nekoj objavi, ima opciju prijaviti tu objavu koja se potom šalje moderatoru na pregled. || F-16 | Prijava neistinitog sadržaja | Srednji| Zahtjev dionika | Korisnik, ukoliko vidi netočan sadržaj na nekoj objavi, ima opciju prijaviti tu objavu koja se potom šalje moderatoru na pregled. || F-17 | Dobivanje uloge moderator kroz sustav aplikacije | Nizak| Zahtjev dionika | Korisnik dobiva ulogu moderatora ukoliko napravi dovoljno uspješnih objava. Uspješne objave se ocijenjuju po kriterijima kao što su njihova točnost, primjerenoost i popularnost ("lajkovi"). || F-18 | Aplikacija nudi fleksibilnost korištenja korisničkog računa| Visok| Povratne informacije korisnika | Korisnik može isti račun koji koristi na računalu koristiti i na mobilnom uređaju || F-19 | Mobilna aplikacija se može koristiti bez korisničkog računa| Nizak| Povratne informacije korisnika | Korisnik može koristiti mobilnu aplikaciju bez prijave u sustav || F-20 | Korisnik ima opciju za odjavu iz svog računa| Srednji| Povratne informacije korisnika | Aplikacija korisniku pruža opciju za "log out" odnosno odjavu iz korisničkog računa |

Ostali zahtjevi

Nefunkcionalni zahtjevi i zahtjevi domene primjene dopunjaju funkcionalne zahtjeve. Oni opisuju kako se sustav treba ponašati i koja ograničenja treba poštivati (performanse, korisničko iskustvo, pouzdanost, standardi kvalitete, sigurnost...).

Zahtjevi za performanse

tablica 2.02: nefunkcionalni zahtjevi za performanse ID zahtjeva Opis Prioritet			

-----	NF-1.1 Sustav treba omogućiti brzo učitavanje karte. Visok NF-1.2 Aplikacija mora učinkovito upravlјati velikim brojem korisničkih objava i pinova bez usporavanja performansi. Srednji NF-1.3 Sustav treba omogućiti brzo		

pretraživanje objava i tagova. | Srednji |

Zahtjevi za iskustvo korisnika

tablica 2.03: nefunkcionalni zahtjevi za iskustvo korisnika | ID zahtjeva | Opis | Prioritet |

|-----

|-----

|-----| | NF-2.1 | Korisničko sučelje treba biti intuitivno i jednostavno za korištenje za sve tipove korisnika. | Visok | | NF-2.2 | Korisnici trebaju moći navigirati kroz aplikaciju s minimalnim brojem klikova. | Nizak |

Zahtjevi za održavanje

tablica 2.04: nefunkcionalni zahtjevi za održavanje | ID zahtjeva | Opis | Prioritet |

|-----

|-----

|-----

|-----| | NF-3.1 | Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje. | Visok | | NF-3.2 | Sustav treba imati dovoljnu dokumentaciju. | Visok | | NF-3.3 | Kod sustava treba biti dokumentiran prema "Code Conventions for the Java Programming Language" dostupnim na [Oracle](#). | Srednji | | NF-3.4 | Sustav treba biti opisan putem dokumenta oblikovanja /SRS/. | Visok | | NF-3.5 | Sustav treba biti popraćen "Priručnikom za rad" koji opisuje pravilnu upotrebu sustava. | Nizak | | NF-3.6 | Sustav treba imati "Plan implementacije" za pravilno postavljanje sustava. | Visok |

Zahtjevi za sigurnost i skalabilnost

tablica 2.05: nefunkcionalni zahtjevi za sigurnost i skalabilnost | ID zahtjeva | Opis | Prioritet |

|-----

|-----

|-----

|-----| | NF-4.1 | Aplikacija treba osigurati sigurnost privatnih podataka korisnika. | Visok | | NF-4.2 | Aplikacija treba biti skalabilna i podržavati proširenje za nove lokacije, korisničke grupe i objave. | Visok | | NF-4.3 | Sustav treba omogućiti korištenje na različitim uređajima. | Srednji | | NF-4.4 | Sustav treba omogućiti jednostavno dodavanje novih funkcionalnosti i integraciju s vanjskim sustavima. | Srednji |

Dionici

1. Vlasnik
2. Registrirani korisnici:
 - lokalci
 - turisti
 - moderatori
3. Neregistrirani korisnici (automatski postavljeni kao turisti)

4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

Neregistrirani korisnik (sudionik) može: * poslati zahtjev za registraciju * koristiti aplikaciju bez registracije uz ograničenja prilikom čega automatski zaprima ulogu turista

Turist (inicijator) može: * pregledavati kartu i objave * filtrirati kroz objave pomoću tagova * označiti objave s oznakom svida mi se ili ih prijaviti za neprimjeren sadržaj

Lokalac (inicijator) može: * isti zahtjevi kao turist * dodatno, može objavljivati sadržaj * prijaviti postojeću objavu za netočan sadržaj

Moderator (inicijator) može: * isti zahtjevi kao turist * pregledavati i po potrebi uklanjati prijavljeni sadržaj * dati ulogu moderatora drugom korisniku (uz ograničenja)

Baza podataka (sudionik): * pohranjuje sve podatke o registriranim korisnicima * pohranjuje sve objave * pohranjuje sve interakcije sa postojećim objavama

Google Oauth (sudionik): * autentificira registraciju i prijavu korisnika putem njihovog Google računa

Open Street Map (sudionik): * služi za prikaz karte na kojoj su "pinovi" * služi za određivanje grada iz danih koordinata

Obrasci uporabe

U nastavku se nalazi tablica sa obrascima uporabe. Uz svaki su priloženi funkcionalni zahtjevi koji su povezani uz njih

tablica 3.01: obrasci uporabe

Obrazac uporabe	Naziv	Funkcionalni zahtjevi
UC-01	Registracija	F-01, F-04
UC-02	Prijava u aplikaciju	F-02, F-04
UC-03	Pristup kao gost	F-05
UC-04	Google prijava	F-02, F-03
UC-05	Pogrešna prijava u sustav	F-02
UC-06	Izrada objave	F-07, F-09, F-14
UC-07	Pregled karte	F-06, F-13
UC-08	Pregled objave	F-09, F-13
UC-09	Prijava neprimjerenog	F-09, F-10, F-15, F-16

Obrazac uporabe	Naziv	Funkcionalni zahtjevi
	sadržaja	
UC-10	“Lajkanje” objave	F-09, F-11
UC-11	Brisanje objave	F-10, F-12
UC-12	Pretraživanje i filtriranje objava	F-08, F-14
UC-13	Responzivnost na mobilnom uređaju	F-18, F-19
UC-14	Dobivanje uloge moderatora	F-10, F-17
UC-15	odjava iz korisničkog računa	F-20

Opis obrazaca uporabe

UC-01: Registracija

Korisnik stvara novi račun u aplikaciji koristeći korisničko ime i lozinku. Tijekom registracije, korisnik unosi osnovne podatke i odabire mjesto stanovanja. U odabranom mjestu stanovanja korisnik dobiva ulogu “lokalca”, što mu omogućava stvaranje objava povezivanih s lokacijom.

Povezani funkcionalni zahtjevi: F-01, F-04

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvaranje novog korisničkog računa
- **Sudionici:** Korisnik
- **Preduvjet:** Korisnik nema postojeći račun

Opis osnovnog tijeka:

1. Korisnik otvara aplikaciju i odabire opciju “Registracija” (F-01).
2. Unosi osnovne podatke: korisničko ime i lozinku (F-01).
3. Korisnik odabire mjesto stanovanja (F-01).
4. Sustav provjerava unesene podatke na valjanost (F-01).
5. Korisnički račun se uspješno kreira, korisnik postaje “lokalac” za odabранo mjesto (F-04).

Opis mogućih odstupanja:

1. **Nevaljani podaci:** Sustav prikazuje obavijest s uputama za ispravak (F-01).
2. **Korisnik već postoji:** Sustav obavještava da korisničko ime već postoji (F-01).

UC-02: Prijava u aplikaciju

Korisnik unosi svoje vjerodajnice (korisničko ime i lozinku) kako bi pristupio

aplikaciji. Nakon uspješne prijave, korisnik može koristiti funkcionalnosti aplikacije dostupne njegovoj ulozi. Registrirani korisnici mogu pregledavati objave drugih korisnika i interaktivnu kartu. Korisnik s ulogom "lokalca" može stvarati objave za svoje mjesto stanovanja.

Povezani funkcionalni zahtjevi: F-02, F-04

- **Glavni sudionik:** Korisnik
- **Cilj:** Pristup korisničkom računu
- **Sudionici:** Korisnik
- **Preduvjet:** Postojeći račun korisnika

Opis osnovnog tijeka:

1. Korisnik unosi vjerodajnice (korisničko ime i lozinku) (F-02).
2. Sustav provjerava valjanost vjerodajnica (F-02).
3. Uspješna prijava omogućuje korisniku pristup funkcionalnostima aplikacije (F-04).
4. Korisnik može pregledavati sadržaj prema svojoj ulozi (F-02).

Opis mogućih odstupanja:

1. **Pogrešne vjerodajnice:** Sustav prikazuje obavijest o grešci i daje mogućnost ponovnog unosa (F-02).

UC-03: Pristup kao gost

Gostujući korisnici mogu koristiti aplikaciju bez potrebe za prijavom. Imaju ograničene mogućnosti, poput pregledavanja interaktivne karte i čitanja objava. Ne mogu aktivno sudjelovati u aplikaciji (npr. "lajkanje" ili stvaranje objava).

Povezani funkcionalni zahtjevi: F-05

- **Glavni sudionik:** Gost
- **Cilj:** Korištenje aplikacije bez prijave
- **Sudionici:** Gost
- **Preduvjet:** Nije potrebna prijava

Opis osnovnog tijeka:

1. Gost otvara aplikaciju (F-05).
2. Pristupa karti i pregledava dostupne objave (F-05).

Opis mogućih odstupanja:

1. **Ograničenje funkcionalnosti:** Gost ne može "lajkati" ili dodavati objave (F-05).

UC-04: Google prijava

Korisnici mogu koristiti Google račun za prijavu u aplikaciju. Ovo omogućava brzo i jednostavno pristupanje aplikaciji bez potrebe za dodatnim vjerodajnicama. Nakon prijave, funkcionalnosti aplikacije dostupne su prema korisničkoj ulozi.

Povezani funkcionalni zahtjevi: F-02, F-03

- **Glavni sudionik:** Korisnik
- **Cilj:** Brza prijava putem Google računa
- **Sudionici:** Korisnik, Google sustav autentifikacije
- **Preduvjet:** Korisnik ima Google račun

Opis osnovnog tijeka:

1. Korisnik odabire opciju "Google prijava" (F-02).
2. Sustav preusmjerava korisnika na Google autentifikaciju (F-03).
3. Korisnik unosi vjerodajnice za Google račun (F-03).
4. Uspješnom prijavom, korisnik pristupa aplikaciji (F-04).

Opis mogućih odstupanja:

1. **Google prijava neuspješna:** Sustav prikazuje obavijest s uputama za ponovno pokušavanje (F-03).

UC-05: Pogrešna prijava u sustav

Ako korisnik unese netočne vjerodajnice (korisničko ime ili lozinku), sustav obavještava korisnika o grešci i pruža mogućnost ponovnog unosa podataka. Ovo sprječava neovlašteni pristup aplikaciji.

Povezani funkcionalni zahtjevi: F-02

- **Glavni sudionik:** Korisnik
- **Cilj:** Obavijestiti korisnika o pogrešnim vjerodajnicama i omogućiti ponovni unos
- **Sudionici:** Korisnik
- **Preduvjet:** Postojeći račun korisnika

Opis osnovnog tijeka:

1. Korisnik unosi vjerodajnice (korisničko ime i lozinku) (F-02).
2. Sustav provjerava valjanost vjerodajnica (F-02).
3. Ako vjerodajnice nisu valjane, sustav obavještava korisnika o grešci (F-02).

Opis mogućih odstupanja:

1. **Ponovni pokušaj:** Korisnik ima mogućnost ponovnog unosa vjerodajnica (F-02).

UC-06: Izrada objave

Korisnici s ulogom "lokalca" mogu dodati objave koje se povezuju s lokacijom na karti. Objava može sadržavati tekst, sliku i unaprijed definirane oznake (tagove) koje olakšavaju filtriranje i pretraživanje.

Povezani funkcionalni zahtjevi: F-07, F-09, F-14

- **Glavni sudionik:** Lokalac
- **Cilj:** Dodavanje nove objave povezane s lokacijom
- **Sudionici:** Lokalac
- **Preduvjet:** Korisnik ima ulogu "lokalca"

Opis osnovnog tijeka:

1. Lokalac otvara opciju za dodavanje objave (F-07).
2. Unosi tekstualni opis, dodaje sliku i odabire relevantne oznake (F-14).
3. Sustav povezuje objavu s određenom lokacijom na karti (F-07).
4. Objavu je moguće pregledavati i interagirati s njom (F-09).

Opis mogućih odstupanja:

1. **Nevaljani unos:** Sustav upozorava na pogreške poput nedostatka obaveznih podataka (F-07).

UC-07: Pregled karte

Korisnici mogu pregledavati interaktivnu kartu s prikazanim objavama. Moguće je pregledati detalje svake objave klikom na njen pin.

Povezani funkcionalni zahtjevi: F-06, F-13

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregled interaktivne karte i objava
- **Sudionici:** Korisnik
- **Preduvjet:** Aplikacija je otvorena

Opis osnovnog tijeka:

1. Korisnik otvara interaktivnu kartu (F-06).
2. Na karti se prikazuju pinovi koji predstavljaju objave (F-13).

3. Korisnik može kliknuti na pin i pregledati detalje objave (F-13).

Opis mogućih odstupanja:

1. **Nema objava:** Korisnik dobiva obavijest da za odabranu područje nema dostupnih objava (F-13).

UC-08: Pregled objave

Korisnik može otvoriti i pregledati detalje pojedine objave, uključujući njen tekstualni sadržaj, slike i oznake.

Povezani funkcionalni zahtjevi: F-09, F-13

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati detalje određene objave
- **Sudionici:** Korisnik
- **Preduvjet:** Postoji objava na karti

Opis osnovnog tijeka:

1. Korisnik odabire pin na karti (F-13).
2. Sustav prikazuje detalje objave (F-09).

Opis mogućih odstupanja:

1. **Neispravan pin:** Korisnik dobiva obavijest da objava nije dostupna (F-09).

UC-09: Prijava neprimjerenog sadržaja

Korisnik može prijaviti objavu koja sadrži neprimjeren sadržaj. Prijave se šalju moderatoru na pregled i daljnju akciju.

Povezani funkcionalni zahtjevi: F-09, F-10, F-15, F-16

- **Glavni sudionik:** Korisnik
- **Cilj:** Obavijestiti moderatora o neprimjerenom sadržaju
- **Sudionici:** Korisnik, Moderator
- **Preduvjet:** Postoji neprimjerena objava

Opis osnovnog tijeka:

1. Korisnik odabire opciju za prijavu objave (F-15).
2. Unosi razlog prijave (neprimjeren sadržaj, neistinit sadržaj itd.) (F-16).
3. Sustav šalje prijavu moderatoru na pregled (F-10).

Opis mogućih odstupanja:

1. **Nedostatak razloga:** Sustav upozorava korisnika da unese razlog prijave (F-15).

UC-10: "Lajkanje" objave

Korisnici mogu ocijeniti objavu tako da je označe sa "sviđa mi se" ili "ne sviđa mi se". Ove ocjene koriste se za rangiranje objava.

Povezani funkcionalni zahtjevi: F-09, F-11

- **Glavni sudionik:** Korisnik
- **Cilj:** Ocijeniti objavu
- **Sudionici:** Korisnik
- **Preduvjet:** Postoji objava na karti

Opis osnovnog tijeka:

1. Korisnik otvara detalje objave (F-09).
2. Klikne na opciju "sviđa mi se" ili "ne sviđa mi se" (F-11).
3. Sustav bilježi ocjenu i ažurira rangiranje objave (F-11).

Opis mogućih odstupanja:

1. **Već ocijenjena objava:** Sustav onemogućuje korisniku ponovno ocjenjivanje iste objave (F-11).

UC-11: Brisanje objave

Moderator može obrisati objavu koju smatra neprimjerenom ili netočnom. Ovo osigurava da sadržaj na platformi ostane relevantan i kvalitetan.

Povezani funkcionalni zahtjevi: F-10, F-12

- **Glavni sudionik:** Moderator
- **Cilj:** Uklanjanje neprimjerene ili netočne objave
- **Sudionici:** Moderator
- **Preduvjet:** Moderator je prijavljen u aplikaciju

Opis osnovnog tijeka:

1. Moderator pregledava prijave za neprimjereni ili netočan sadržaj (F-10).
2. Odabire objavu za brisanje (F-12).
3. Sustav traži potvrdu akcije (F-12).
4. Moderator potvrđuje brisanje, a sustav uklanja objavu iz prikaza

(F-12).

Opis mogućih odstupanja:

1. **Objava nije dostupna:** Sustav prikazuje obavijest da je objava već uklonjena (F-12).

UC-12: Pretraživanje i filtriranje objava

Korisnik može pretraživati i filtrirati objave koristeći unaprijed određene tagove kako bi brzo pronašao relevantan sadržaj.

Povezani funkcionalni zahtjevi: F-08, F-14

- **Glavni sudionik:** Korisnik
- **Cilj:** Pronalaženje specifičnih objava na temelju ključnih riječi ili tagova
- **Sudionici:** Korisnik
- **Preduvjet:** Postoje objave s tagovima

Opis osnovnog tijeka:

1. Korisnik unosi ključne riječi ili odabire tagove za filtriranje (F-08, F-14).
2. Sustav pretražuje objave i prikazuje rezultate (F-08).
3. Korisnik pregledava rezultate filtriranja na karti ili u listi (F-14).

Opis mogućih odstupanja:

1. **Nema rezultata:** Sustav obavještava korisnika da nema objava koje zadovoljavaju kriterije (F-08).

UC-13: Responzivnost na mobilnom uređaju

Aplikacija se prilagođava različitim veličinama ekrana, omogućavajući jednostavno korištenje na mobilnim uređajima.

Povezani funkcionalni zahtjevi: F-18, F-19

- **Glavni sudionik:** Korisnik
- **Cilj:** Omogućiti jednostavno korištenje aplikacije na mobilnom uređaju
- **Sudionici:** Korisnik
- **Preduvjet:** Korisnik koristi mobilni uređaj

Opis osnovnog tijeka:

1. Korisnik otvara aplikaciju na mobilnom uređaju (F-18).
2. Aplikacija se automatski prilagođava veličini ekrana (F-19).

3. Korisnik koristi funkcionalnosti aplikacije na intuitivan način (F-19).

Opis mogućih odstupanja:

1. **Problemi s prikazom:** Korisnik prijavljuje poteškoće s prikazom funkcionalnosti na mobilnom uređaju (F-18).

UC-14: Dobivanje uloge moderatora

Korisnik može dobiti ulogu moderatora nakon ispunjenja određenih kriterija, poput stvaranja uspješnih objava.

Povezani funkcionalni zahtjevi: F-10, F-17

- **Glavni sudionik:** Korisnik
- **Cilj:** Postati moderator aplikacije
- **Sudionici:** Korisnik
- **Preduvjet:** Korisnik ispunjava kriterije za dobivanje uloge

Opis osnovnog tijeka:

1. Sustav prati korisničke aktivnosti, uključujući uspješnost objava (F-17).
2. Korisnik dostiže određeni broj uspješnih objava (F-17).
3. Sustav automatski dodjeljuje korisniku ulogu moderatora (F-10).
4. Korisnik dobiva obavijest o promjeni uloge (F-17).

Opis mogućih odstupanja:

1. **Nepotpun ispunjenje kriterija:** Sustav ne dodjeljuje ulogu moderatora dok se kriteriji ne ispune (F-17).

UC-15: Odjava iz korisničkog računa

Korisnik može odjaviti svoj račun iz aplikacije i prekinuti trenutnu sesiju.

Povezani funkcionalni zahtjevi: F-20

- **Glavni sudionik:** Korisnik
- **Cilj:** Prekinuti aktivnu sesiju korisničkog računa
- **Sudionici:** Korisnik
- **Preduvjet:** Korisnik je prijavljen

Opis osnovnog tijeka:

1. Korisnik odabire opciju "Odjava" (F-20).
2. Sustav traži potvrdu za odjavu (F-20).

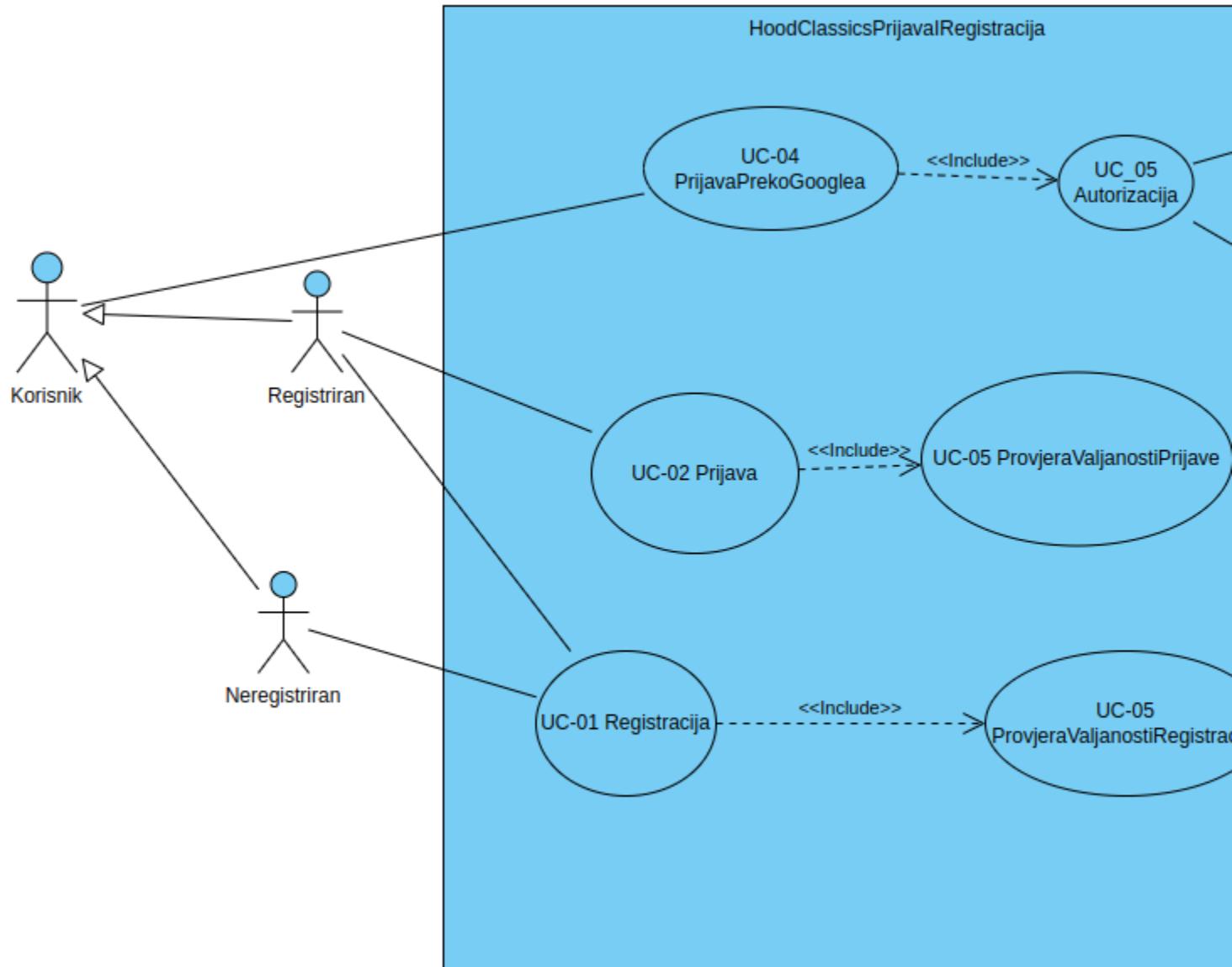
3. Korisnički račun se odjavljuje, a korisnik se preusmjerava na početni ekran (F-20).

Opis mogućih odstupanja:

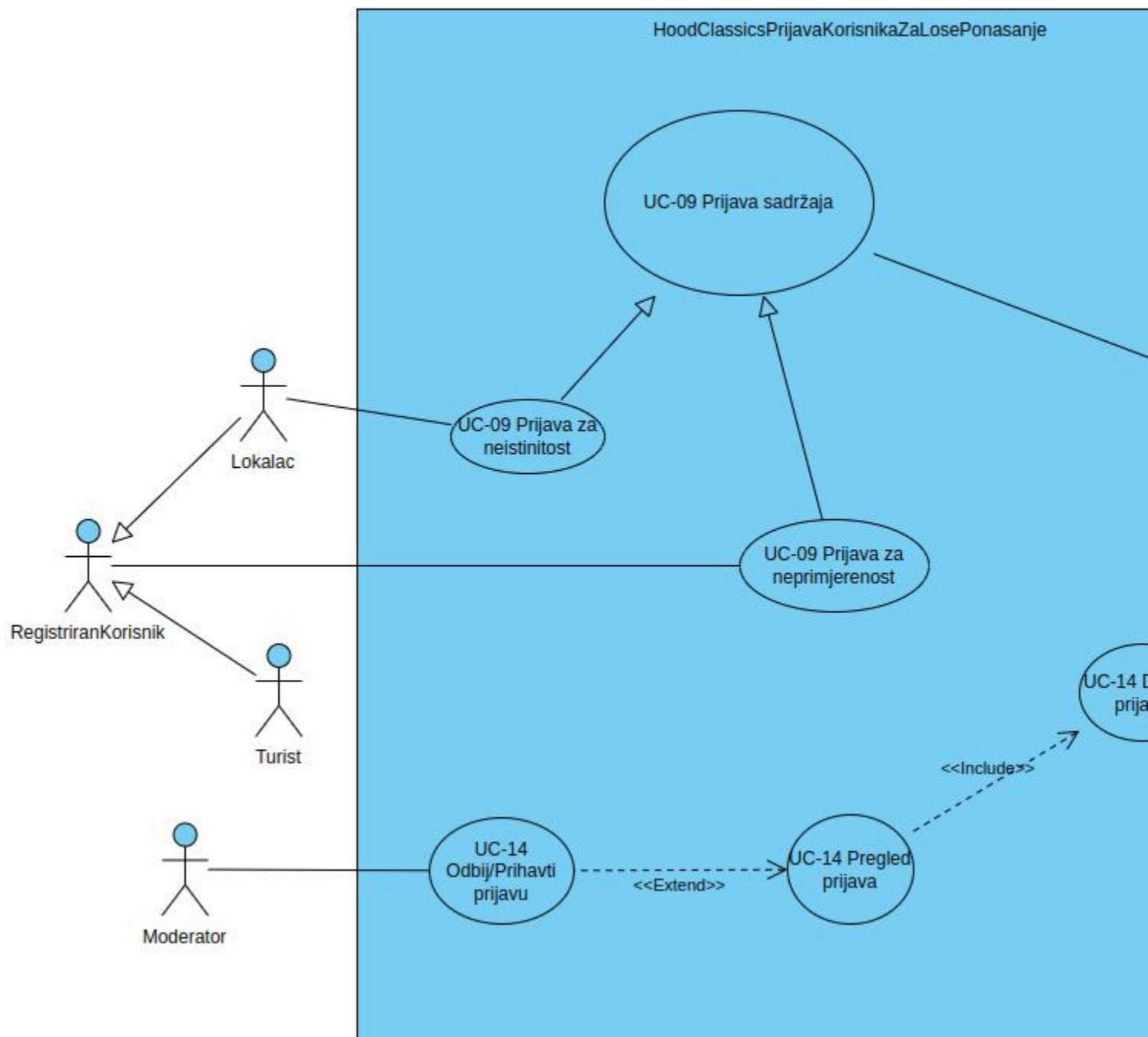
1. **Problemi s odjavom:** Sustav prikazuje obavijest o pogrešci i omogućava ponovno pokušavanje (F-20).

Dijagrami obrazaca uporabe

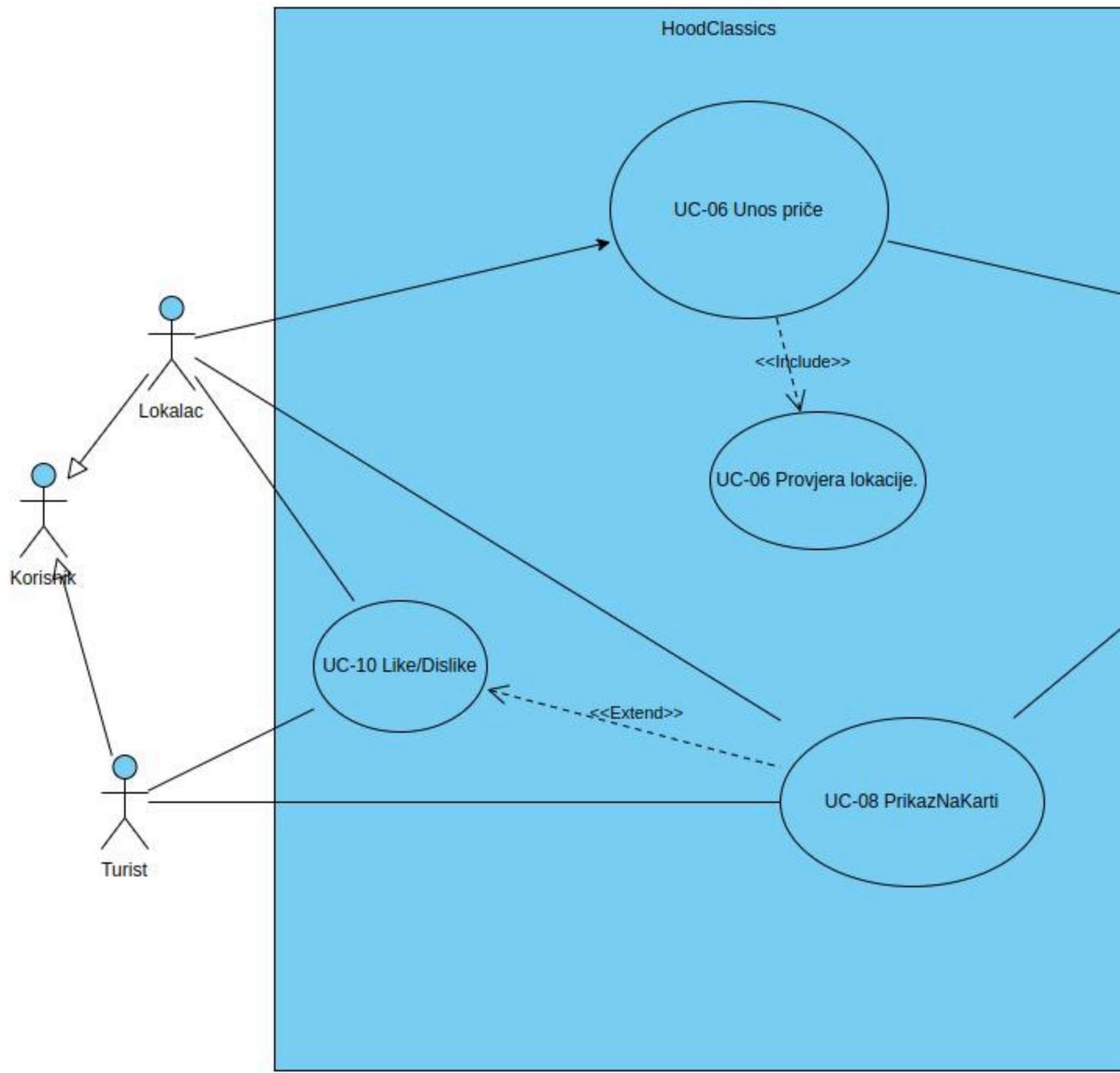
dijagram 3.03: UML prijava i registracija



dijagram 3.05: UML sustava prijave korisnika za loše ponašanje

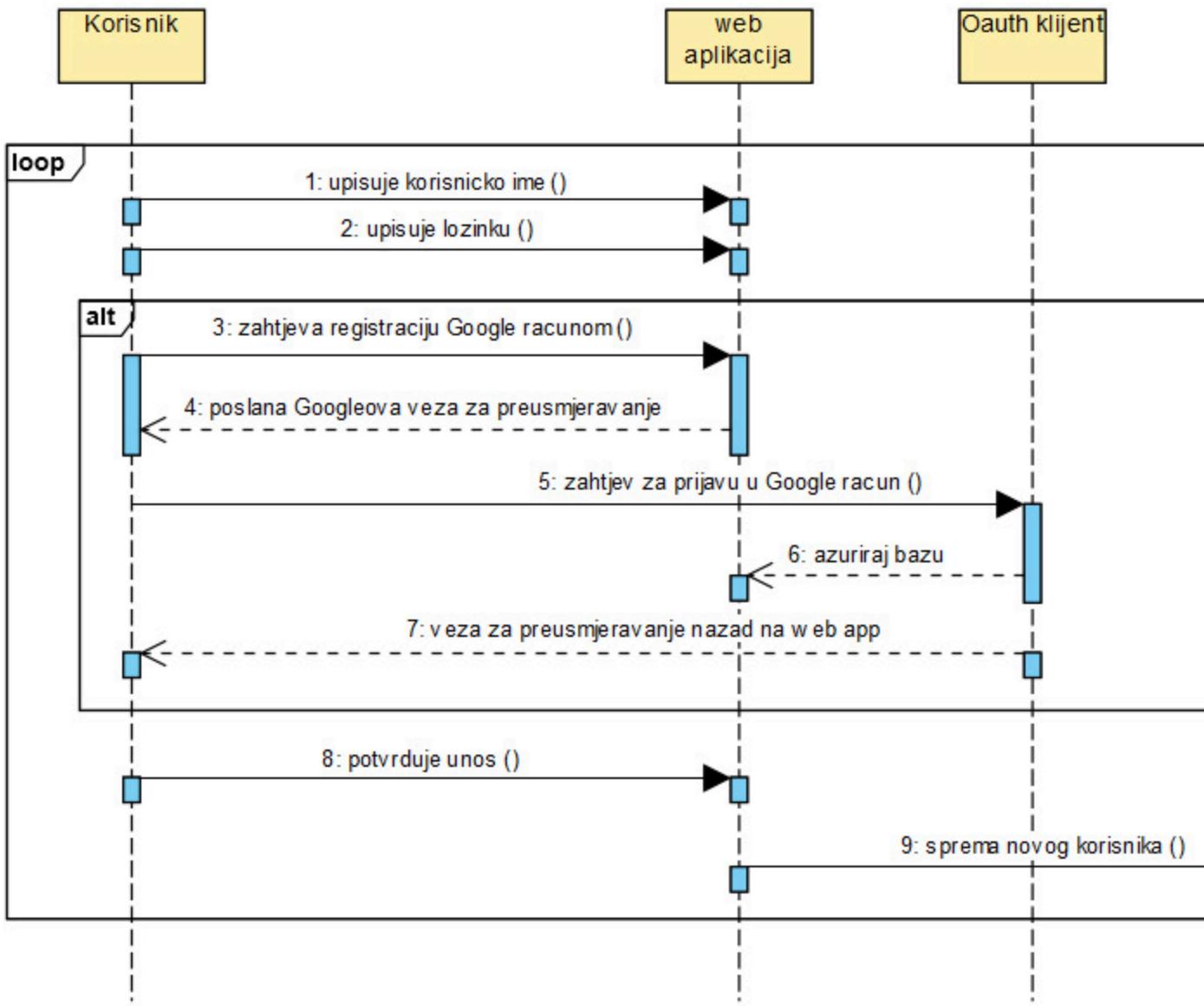


dijagram 3.06: UML Tok Sadržaja



Sekvencijski dijagrami

dijagram 3.07: sekvencijski dijagram registracije korisnika



> Ovaj dijagram je povezan sa obrascem uporabe UC-01.

Novi korisnik želi se registrirati na aplikaciju, pri čemu sustav prikuplja njegove podatke, verificira ih i kreira račun koji se spremi u bazu podataka:

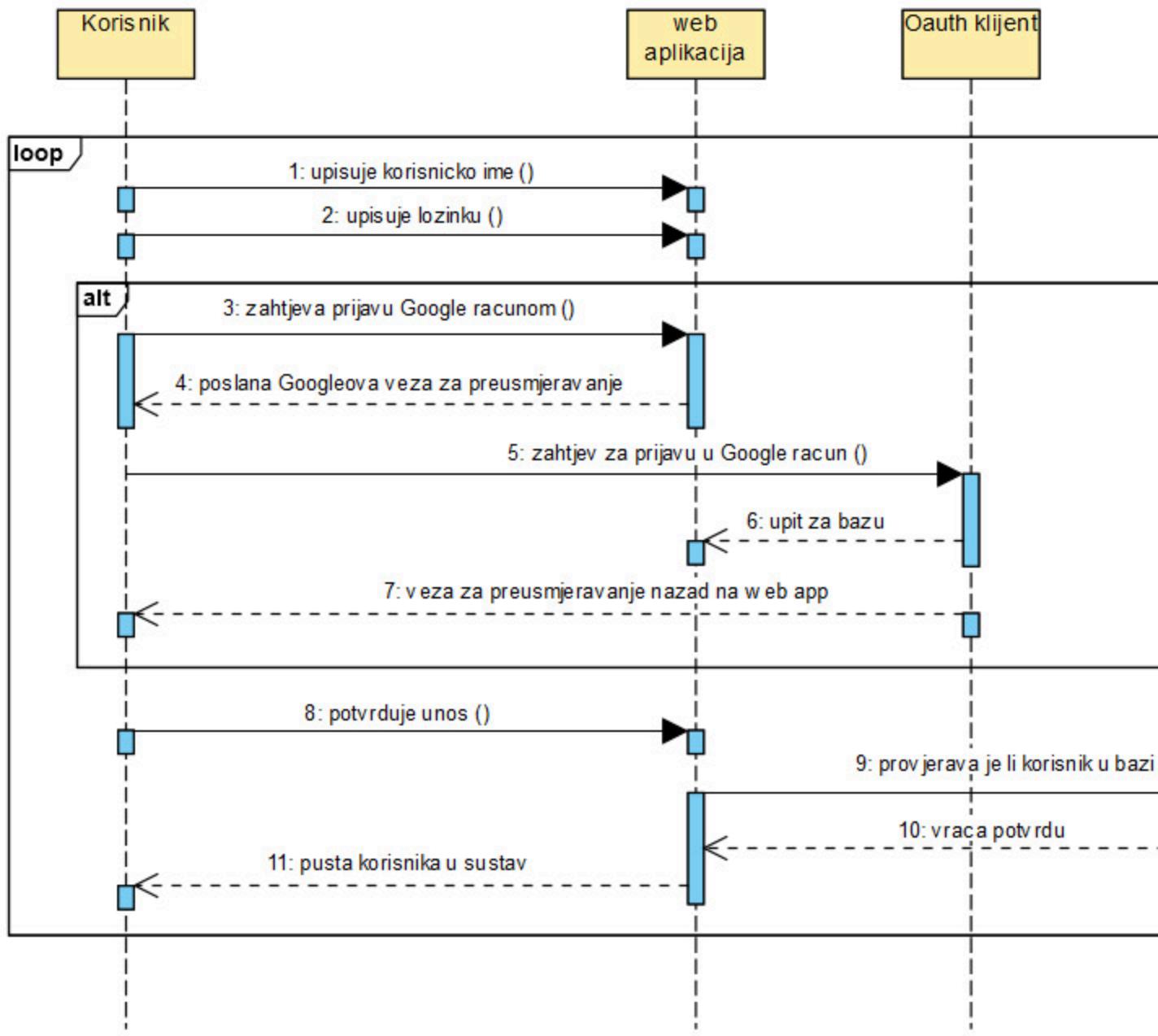
Registracija izradom korisničkog računa * Korisnik: 1. upisuje korisničko ime 2. upisuje lozinku za svoj račun 3. potvrđuje unos pritiskom na gumb

- web aplikacija:
 1. šalje bazi zahtjev za spremanje novog korisnika
- baza podataka
 1. spremi novog korisnika

Registracija koristeći postojeći Google račun * Korisnik: 1. odabire opciju registracije putem Google računa

- web aplikacija:
 1. šalje korisniku Googleovu poveznicu za preusmjeravanje
- Korisnik:
 2. Oauth klijentu šalje zahtjev za prijavu u Google račun
- Oauth klijent
 1. šalje web aplikaciji da treba ažurirati bazu podataka
 2. šalje korisniku poveznicu za preusmjeravanje nazad na webb aplikaciju
- web aplikacija:
 2. šalje bazi zahtjev za spremanje novog korisnika
- baza podataka
 1. sprema novog korisnika

dijagram 3.08: sekvensijski dijagram uspješne prijave korisnika



> Ovaj dijagram je povezan sa obrascem uporabe UC-02.

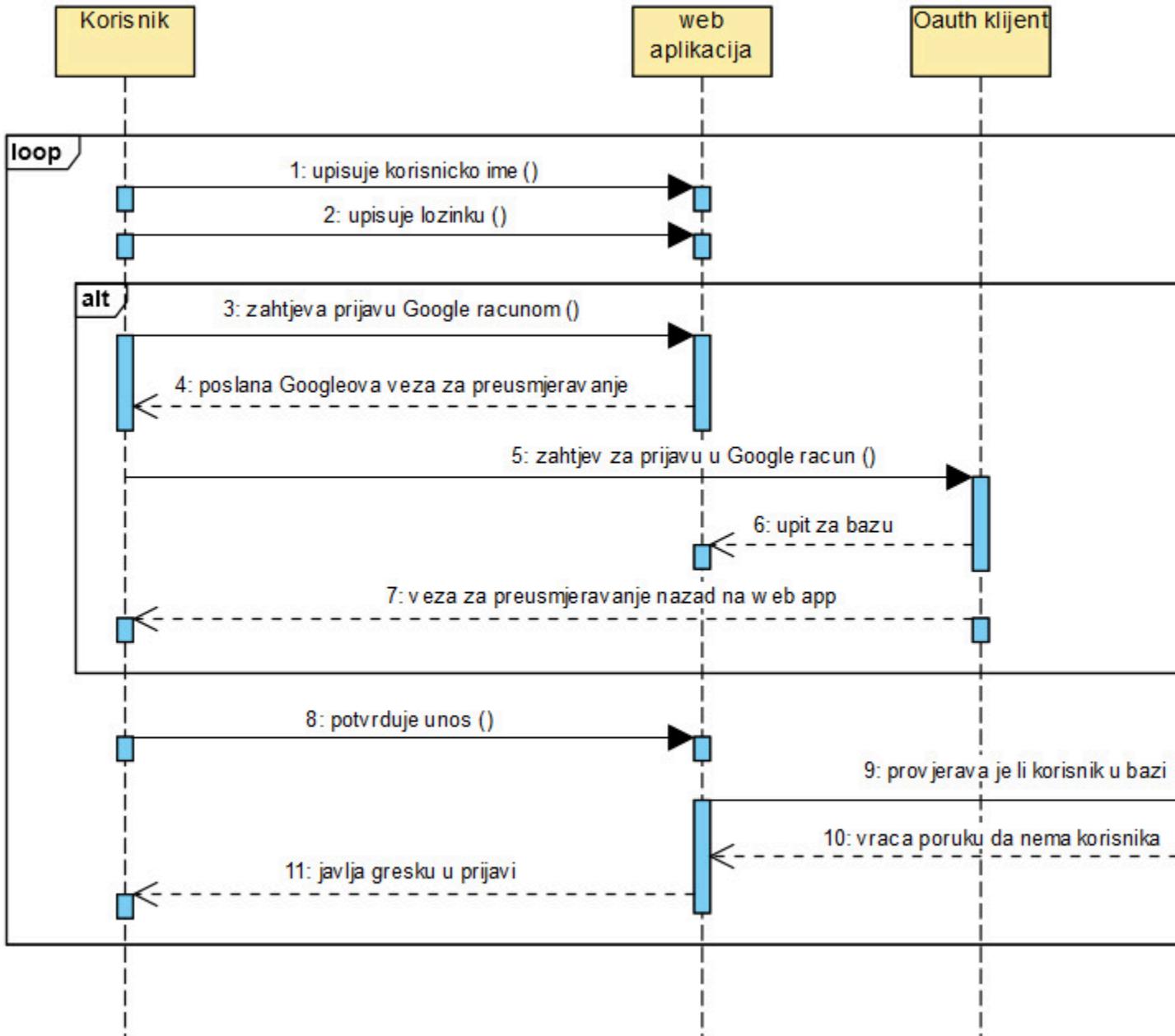
Registrirani korisnik želi se prijaviti u aplikaciju. Sustav provjerava vjerodajnica korisnika i odobrava pristup: *Prijava izradom korisničkog računa* * Korisnik: 1. upisuje korisničko ime 2. upisuje lozinku za svoj račun 3. potvrđuje unos pritiskom na gumb

- web aplikacija:
 1. dopušta korisniku korištenje funkcionalnosti aplikacije

Prijava koristeći postojeći Google račun * Korisnik: 1. odabire opciju prijave putem Google računa

- web aplikacija:
 1. šalje korisniku Googleovu poveznicu za preusmjeravanje
- Korisnik:
 2. Oauth klijentu šalje zahtjev za prijavu u Google račun
- Oauth klijent
 1. šalje korisniku poveznicu za preusmjeravanje nazad na webb aplikaciju
- web aplikacija:
 1. dopušta korisniku korištenje funkcionalnosti aplikacije

dijagram 3.09: sekvencijski dijagram pogrešne prijave korisnika



> Ovaj dijagram je povezan sa obrascem uporabe UC-05.

Korisnik pokušava prijavu, ali unosi netočne vjerodajnice, zbog čega sustav odbija pristup: *Prijava izradom korisničkog računa* * Korisnik: 1. upisuje korisničko ime 2. upisuje lozinku za svoj račun 3. potvrđuje unos pritiskom na gumb

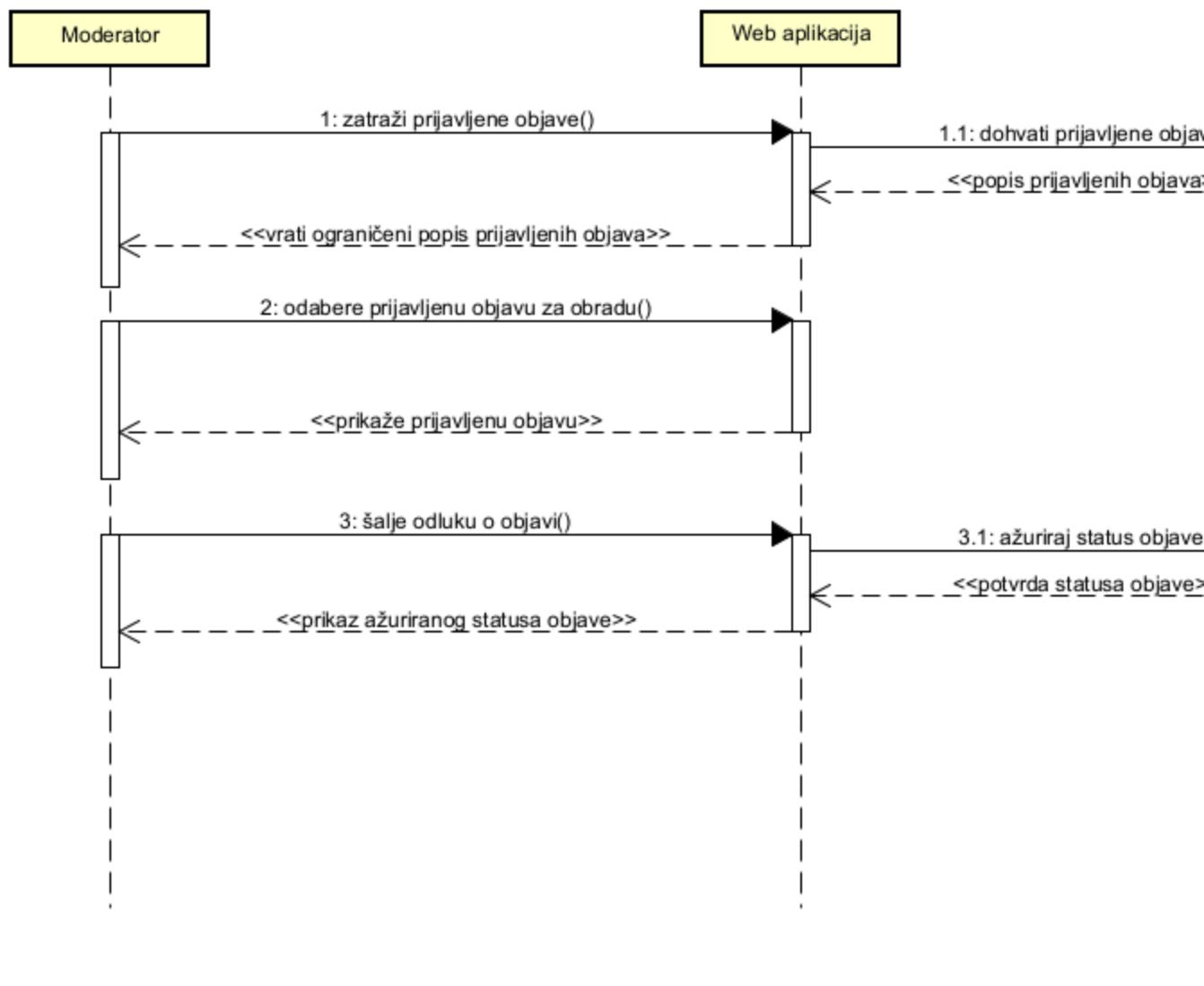
- web aplikacija:
 1. javlja korisniku poruku da su podaci krivo uneseni

Prijava koristeći postojeći Google račun * Korisnik: 1. odabire opciju prijave putem Google računa

- web aplikacija:
 1. šalje korisniku Googleovu poveznicu za preusmjeravanje
- Korisnik:
 2. Oauth klijentu šalje zahtjev za prijavu u Google račun
- Oauth klijent:
 1. šalje korisniku poveznicu za preusmjeravanje nazad na webb aplikaciju
- web aplikacija:
 1. javlja korisniku poruku da su podaci krivo uneseni

dijagram 3.10: sekvenički dijagram moderatora i baze

sd d2: sekvenčni dijagram- moderator i baza

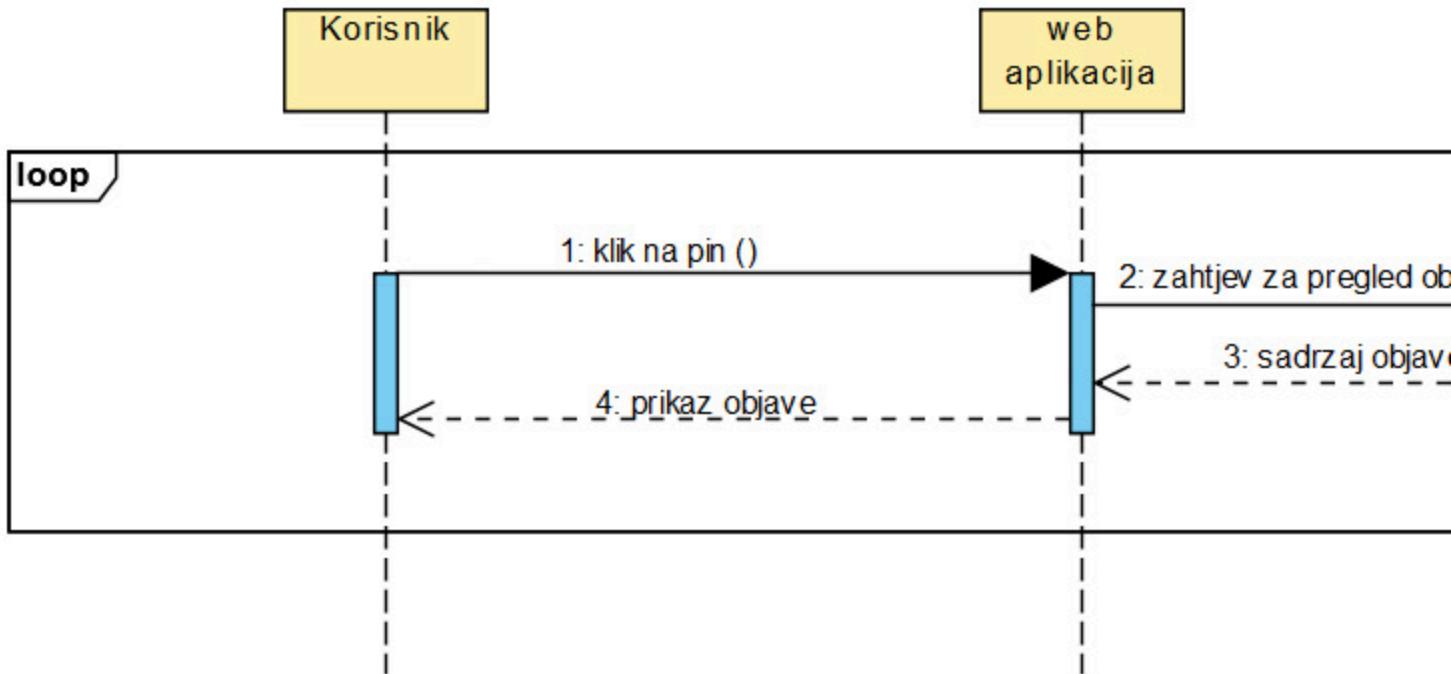


> Ovaj dijagram je povezan sa obrascem uporabe UC-11.

Moderator prima prijavu od korisnika (turista ili lokalca), pregledava je, provjerava njen sadržaj, i zatim odlučuje hoće li je ukloniti ili odobriti: *

Moderator: 1. zatraži prijavljene objave * web aplikacija: 1. dohvati prijavljene objave
 2. vrati ograničeni popis prijavljenih objava * Moderator: 2. odabire prijavljenu objavu za pregled i obradu
 3. šalje odluku o objavi sustavu * web aplikacija: 3. ažurira status objave u bazi 4. ažurira prikaz objave u aplikaciji

dijagram 3.11: sekvenčni dijagram pregleda objave



> Ovaj dijagram je povezan sa obrascem uporabe UC-08.

Korisnik pokušava pregledati pojedinačnu objavu za željenu lokaciju * Korisnik: 1. nakon pronađaska zanimljivog pina, pritisne ga za detalje objave * Aplikacija: 1. dohvati tekst objave 2. vrati čitavu objavu korisniku

Arhitektura sustava

1. Stil arhitekture

Za ovaj sustav odabrana je **MVC (Model-View-Controller)** arhitektura, s jasno definiranim ulogama između backend-a i frontend-a, temeljenom na **Spring Boot** za backend i **React** za frontend. MVC arhitektura omogućava:

- **Odvajanje odgovornosti:** MVC obrazac omogućuje jasnou podjelu između modela (podaci), prikaza (korisničko sučelje) i kontrolera (poslovna logika), što olakšava razvoj i održavanje aplikacije.
- **Fleksibilnost:** Razdvajanje slojeva omogućava lakšu modifikaciju bilo kojeg dijela sustava bez utjecaja na ostale dijelove.
- **Testabilnost:** S obzirom na jasnou podjelu, testiranje svakog sloja (modela, kontrolera, prikaza) može se obaviti neovisno.

Arhitektura sustava je podijeljena u sljedeće slojeve:

- **Controller sloj:** Upravljanje HTTP zahtjevima, komunikacija s poslovnom logikom, te vraćanje odgovora prema korisniku.
- **Model sloj:** Predstavlja podatke, tj. objekte koji se koriste u aplikaciji. U Spring Boot aplikaciji, ovo je obično entitet povezani s bazom podataka.

- **View sloj:** U React-u, odgovoran je za prikazivanje podataka korisniku putem dinamičkih sučelja.

2. Podsustavi

Sustav se sastoji od tri glavne komponente:

- **Frontend podsustav (React):**
 - **View** u MVC-u implementiran je pomoću **React-a**. React se koristi za razvoj dinamičkog korisničkog sučelja.
 - Komunicira s backendom putem **REST API-a**, šaljući HTTP zahtjeve i obrađujući odgovore u JSON formatu.
 - Korisničko sučelje omogućava korisnicima da interagiraju s aplikacijom, dok React upravlja dinamičkim prikazom podataka i odgovara na korisničke akcije.
- **Backend podsustav (Spring Boot):**
 - **Controller sloj:** U Spring Boot-u, kontroleri obrađuju HTTP zahtjeve i odgovaraju korisnicima s podacima u JSON formatu. Kontroler također prosljeđuje podatke između modela (poslovne logike) i prikaza (frontend-a).
 - **Service sloj:** Poslovna logika sustava, koja se implementira kao zaseban sloj, odgovorna je za obradbu podataka i donošenje poslovnih odluka.
 - **Repository sloj:** Ovaj sloj upravlja podacima, komunicira s bazom podataka pomoću ORM alata poput **Spring Data JPA** i pruža podatke kontroleru.
 - **Model sloj:** Predstavlja entitete koji su povezani s bazom podataka. Ti entiteti obično uključuju korisničke podatke, proizvode, narudžbe itd.
- **Baza podataka:**
 - **PostgreSQL** se koristi za pohranu podataka. Relacijska baza podataka omogućava pohranu strukturiranih podataka s jasno definiranim relacijama između entiteta.
 - **Spring Data JPA** (u kombinaciji s Hibernate-om) koristi se za interakciju s bazom podataka, omogućujući jednostavno upravljanje podacima.

3. Preslikavanje na radnu platformu

Sustav je trenutačno implementiran lokalno, a bit će postavljen na online platformu pomoću **Docker-a**, što omogućuje lakšu distribuciju i skalabilnost aplikacije.

4. Spremišta podataka

- **PostgreSQL** je odabran za pohranu podataka zbog njegove robusnosti i podrške za relacije među podacima.
- Podaci koji se pohranjuju uključuju:

- Korisničke podatke (autentifikacija i autorizacija).
- Podaci o objavljenim pričama

5. Mrežni protokoli

- **HTTP/HTTPS** protokoli koriste se za komunikaciju između **React** frontend-a i **Spring Boot** backend-a.
- **HTTPS** osigurava šifriranje podataka tijekom prijenosa, čime se povećava sigurnost aplikacije, osobito kod osjetljivih podataka poput prijava i autentifikacije.

6. Globalni upravljački tok

- **Početna interakcija korisnika:**
 - Korisnik putem **React** sučelja šalje zahtjev prema backendu (Spring Boot) (npr. prijava, dohvaćanje podataka).
 - Zahtjev se šalje putem **HTTP/HTTPS** protokola prema **Controller** sloju u backendu.
- **Obrada zahtjeva u backendu:**
 - **Controller sloj** u Spring Boot-u prima zahtjev, proslijeđuje ga **Service sloju**, koji obrađuje poslovnu logiku.
 - Ako je potrebno, **Service sloj** komunicira s **Repository slojem** za dohvati ili pohranu podataka u bazu podataka.
- **Povratna informacija korisniku:**
 - Nakon obrade zahtjeva, backend vraća odgovor u JSON formatu.
 - **React** frontend obrađuje ovaj odgovor i dinamički ažurira sučelje, prikazujući podatke korisnicima.

7. Obrazloženje odabira arhitekture

Izbor MVC arhitekture temelji se na sljedećim principima:

- **Visoka kohezija:**
 - Svaki sloj (Model, View, Controller) ima jasno definiranu odgovornost. To omogućava jednostavno testiranje, razvoj i održavanje.
- **Niska povezanost:**
 - Slojevi međusobno komuniciraju preko jasno definiranih sučelja (npr. između Controller sloja i Service sloja), čime se smanjuje međusobna povezanost i olakšava održavanje sustava.
- **Jednostavnost implementacije i održavanja:**
 - MVC pristup omogućava lakše upravljanje sustavom jer se poslovna logika, korisničko sučelje i podaci drže odvojeni. Također omogućava lakšu integraciju novih funkcionalnosti.
- **Performanse:**
 - Kako je sustav temeljen na monolitnoj MVC arhitekturi, nema potrebe za složenim mrežnim pozivima između različitih servisa, što smanjuje latenciju.

- **Sigurnost:**
 - MVC pristup omogućuje centraliziranu implementaciju sigurnosnih mjera, poput autentifikacije i autorizacije, na razini kontrolera.

8. Organizacija sustava na najvišoj razini apstrakcije

1. Klijent-poslužitelj

- **Frontend (React):**
 - **View** u MVC arhitekturi implementiran je pomoću React-a, koji prikazuje podatke korisnicima i omogućuje interakciju s aplikacijom.
 - React šalje HTTP zahtjeve backendu putem **REST API-ja**.
- **Backend (Spring Boot):**
 - **Controller sloj** obrađuje zahtjeve, proslijeđuje ih **Service sloju** i vraća odgovore klijentima u JSON formatu.
 - **Model sloj** predstavlja entitete povezane s bazom podataka, dok **Repository sloj** upravlja podacima.

2. Baza podataka

- **PostgreSQL** se koristi kao centralizirana relacijska baza podataka koja pohranjuje sve relevantne podatke aplikacije.
- **Spring Data JPA** omogućuje laganu interakciju s bazom podataka putem objektnog modela.

3. Grafičko sučelje

- **React** omogućava responzivno korisničko sučelje.
- Sučelje šalje zahtjeve backendu putem **REST API-ja**, a backend vraća odgovore koje React koristi za ažuriranje prikaza korisnicima.

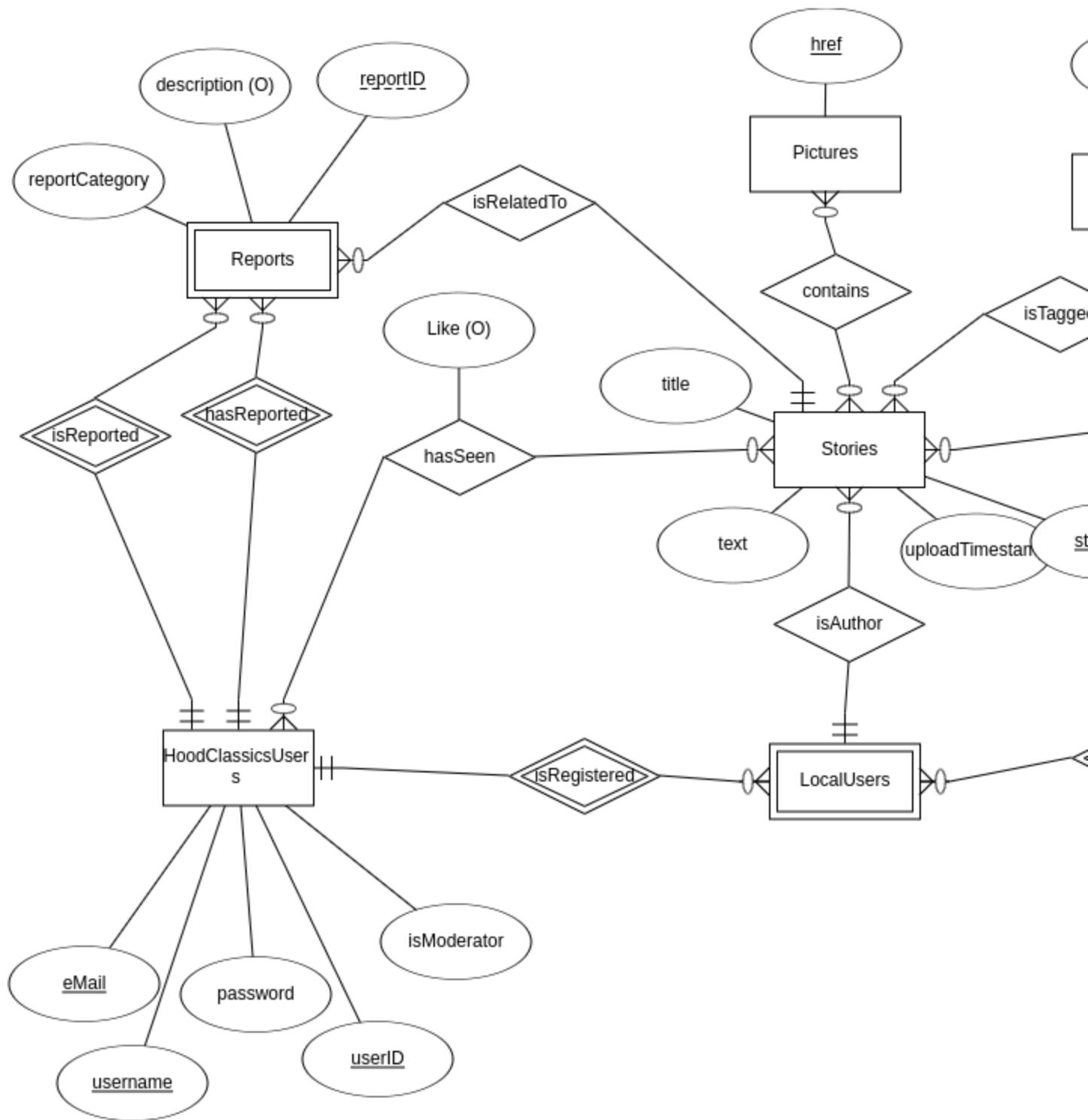
Ova arhitektura temelji se na **MVC obrascu**, osiguravajući jasan razdjeljak između modela, prikaza i poslovne logike, dok korištenje **Spring Boot-a** i **React-a** omogućava jednostavno, skalabilno i učinkovito upravljanje aplikacijom.

Baza podataka

Odabrali smo PostgreSQL bazu podataka implementirali ju kroz java spring framework i deployali ju preko besplatne verzije Rendera.

Dijagram baze podataka

dijagram 4.01: ER dijagram web aplikacije HoodClassics



Dijagram 4.01 prikazuje bazu podataka aplikacije **HoodClassics** i opisuje strukturu podataka, odnose između entiteta i njihova svojstva. On prikazuje kako se podaci u aplikaciji **HoodClassics** organiziraju i povezuju. Struktura omogućuje detaljno praćenje priča, korisničkih interakcija i geografskih podataka. Ključni odnosi između korisnika, priča, prijava i geografskih entiteta omogućavaju

skalabilnost i proširivost sustava, što je važno za buduće nadogradnje aplikacije.

Entiteti i njihova svojstva 1. HoodClassicsUsers

- Svojstva: - userID (jedinstveni identifikator korisnika)
- username (korisničko ime)
- eMail (e-mail adresa korisnika)
- password (lozinka)
- isModerator (flag koji označava je li korisnik moderator)
- Odnosi: - Može prijaviti priče (veza hasReported prema entitetu Reports). - Može pregledavati priče (veza hasSeen prema entitetu Stories). - Može biti lokalni korisnik (veza isRegistered prema entitetu LocalUsers).

2. Reports

- Svojstva:
 - reportID (jedinstveni identifikator prijave)
 - description (opis prijave)
 - reportCategory (kategorija prijave)
- Odnosi:
 - Povezan s korisnikom koji je podnio prijavu (IsReported prema HoodClassicsUsers).
 - Odnosi se na određenu priču (isRelatedTo prema Stories).

3. Stories

- Svojstva:
 - storyID (jedinstveni identifikator priče)
 - title (naslov priče)
 - text (tekst priče)
 - uploadTimestamp (vrijeme učitavanja priče)
- Odnosi:
 - Ima slike (contains prema Pictures).
 - Oznake (tagove) kroz vezu isTagged prema Tag.
 - Povezana s lokacijama (isLocatedAt prema Coordinates).
 - Kreira ih lokalni korisnik (isAuthor prema LocalUsers).
 - Može biti prijavljena kroz vezu isRelatedTo prema Reports.

4. Pictures

- Svojstva:
 - href (link na sliku).
- Odnosi:
 - Povezane su s pričama (contains prema Stories).

5. Tag

- Svojstva:
 - tagID (jedinstveni identifikator taga).
 - tagName (naziv taga).
- Odnosi:
 - Koristi se za označavanje priča (isTagged prema Stories).

6. Coordinates

- Svojstva:
 - latitude (geografska širina).
 - longitude (geografska dužina).

- Odnosi:
 - Povezane s pričama kroz vezu `isLocatedAt`.
 - Pripadaju određenom gradu (`belongsTo` prema `Town`).

7. **Town**

- Svojstva:
 - `townID` (jedinstveni identifikator grada).
 - `townName` (naziv grada).
 - `state` (država kojoj grad pripada).
 - `county` (županija).
- Odnosi:
 - Ima geografske koordinate (`belongsTo` prema `Coordinates`).
 - Dio je određene države (`isPartOf` prema `Country`).

8. **Country**

- Svojstva:
 - `countryID` (jedinstveni identifikator države).
 - `countryName` (naziv države).

9. **LocalUsers**

- Nasljeđuje svojstva iz entiteta `HoodClassicsUsers` kroz vezu `isRegistered`.
- Odnosi:
 - Mogu biti autori priča (`isAuthor` prema `Stories`).
 - Pripadaju određenim gradovima (`ISALocalOf` prema `Town`).

Ključni odnosi - Prijave (Reports): Korisnici (`HoodClassicsUsers`) mogu prijaviti priče kroz vezu `hasReported`. Prijava je povezana s određenom pričom putem `isRelatedTo`.

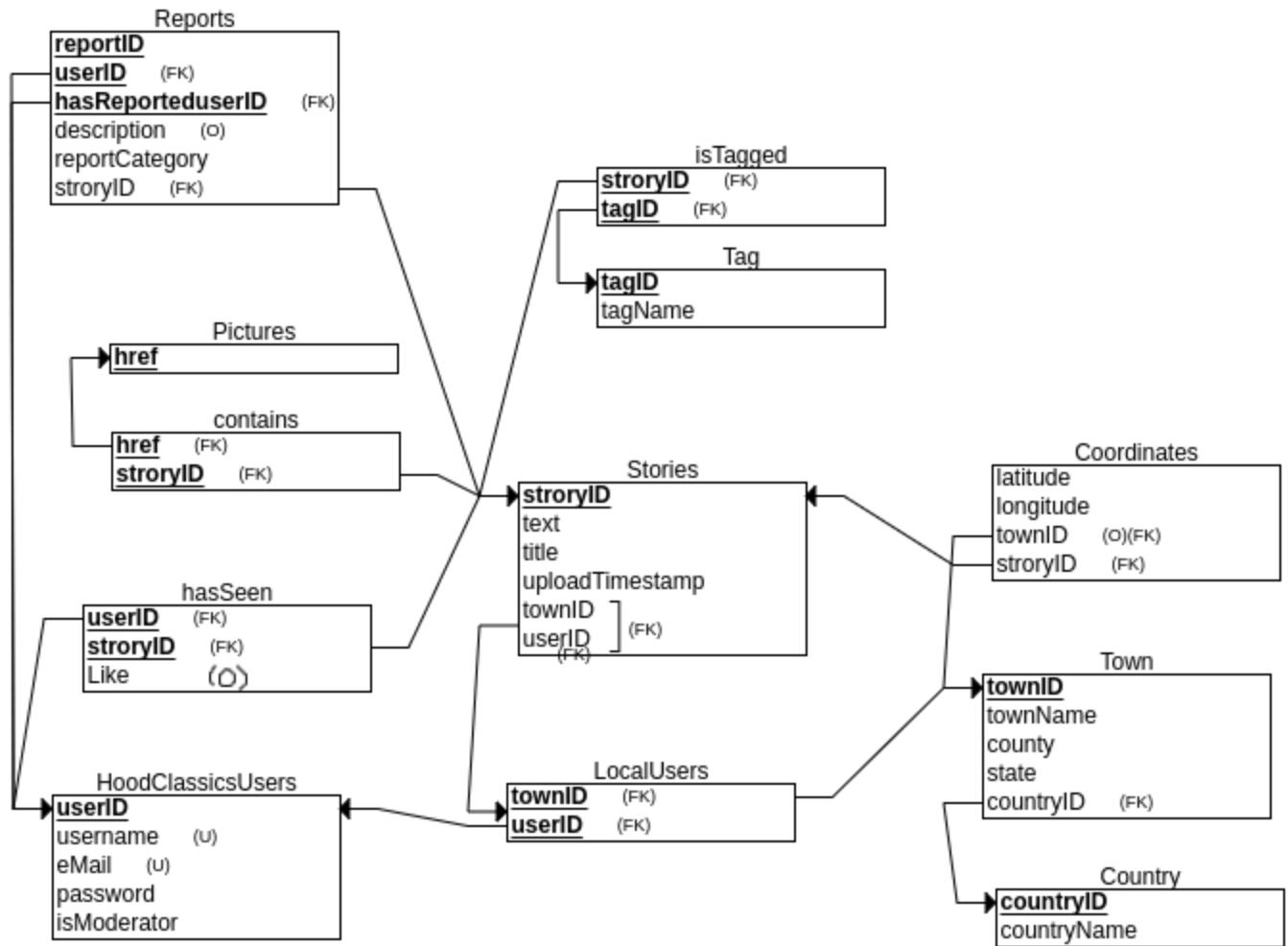
- Priče (Stories):

- Sadrže slike kroz vezu `contains`.
- Označene su tagovima kroz vezu `isTagged`.
- Lokacija priče povezana je putem `isLocatedAt` s entitetom `Coordinates`.
- Lokalni korisnici su autori priča (`isAuthor`).

- Geografija:

- Koordinate (`Coordinates`) pripadaju određenim gradovima (`belongsTo`).
- Gradovi (`Town`) dio su država (`isPartOf` prema `Country`).

dijagram 4.02: relacijska shema web aplikacije HoodClassics



Dijagram 4.02 prikazuje relacijsku shemu koja predstavlja dizajn baze podataka za aplikaciju **HoodClassics**. Opisuje tablice (entitete), njihova svojstva (attribute) i veze između njih kroz strane ključeve (FK) i primarne ključeve (PK). Ona definira funkcionalnosti aplikacije **HoodClassics** kroz međusobno povezane tablice. Osigurava organizaciju podataka o korisnicima, pričama, prijavama, oznakama, slikama i geografskim lokacijama. Struktura je fleksibilna i omogućuje skalabilnost za buduće proširenje funkcionalnosti, poput naprednjeg sustava moderiranja, mobilne aplikacije ili integracije novih značajki.

Tablice i njihovi atributi 1. HoodClassicsUsers - Atributi: - userID (PK) - jedinstveni identifikator korisnika. - username (U) - korisničko ime (unikatno). - eMail - e-mail adresa korisnika. - password - lozinka. - isModerator - indikator je li korisnik moderator. - Odnosi: - Može pregledavati priče (hasSeen veza s tablicom Stories). - Može prijaviti priče (Reports). - Lokalni korisnici su podskup ove tablice (LocalUsers).

2. Reports

◦ Atributi:

- reportID (PK) - jedinstveni identifikator prijave.

- userID (FK) – korisnik koji je prijavio priču.
- hasReporteduserID (FK) – korisnik na kog se odnosi prijava.
- description – opis prijave.
- reportCategory – kategorija prijave.
- storyID (FK) – priča koja je prijavljena.
- Odnosi:
 - Povezuje korisnike i prijave kroz userID i storyID.

3. Stories

- Atributi:
 - storyID (PK) – jedinstveni identifikator priče.
 - text – sadržaj priče.
 - title – naslov priče.
 - uploadTimestamp – vrijeme učitavanja priče.
 - userID (FK) – autor priče.
 - townID (FK) – grad povezan s pričom.
- Odnosi:
 - Može imati slike (Pictures).
 - Oznake kroz vezu (isTagged s tablicom Tag).
 - Lokacija priče (Coordinates).

4. Pictures

- Atributi:
 - href – URL slike.
 - storyID (FK) – priča kojoj slika pripada.
- Odnosi:
 - Povezuje slike i priče (contains veza s Stories).

5. Tag

- Atributi:
 - tagID (PK) – jedinstveni identifikator taga.
 - tagName – naziv taga.
- Odnosi:
 - Povezani s pričama (isTagged veza s Stories).

6. Coordinates

- Atributi:
 - latitude – geografska širina.
 - longitude – geografska dužina.
 - townID (FK) – grad povezan s koordinatama.
 - storyID (FK) – priča povezana s lokacijom.
- Odnosi:
 - Lokacije su povezane s pričama i gradovima.

7. Town

- Atributi:
 - townID (PK) – jedinstveni identifikator grada.
 - townName – naziv grada.
 - county – županija.
 - state – država kojoj grad pripada.
 - countryID (FK) – država povezana s gradom.
- Odnosi:
 - Gradovi su povezani s državama i pričama.

8. Country

- Atributi:
 - countryID (PK) - jedinstveni identifikator države.
 - countryName - naziv države.
- Odnosi:
 - Povezana s gradovima (Town).

9. LocalUsers

- Atributi:
 - userID (FK) - korisnik iz tablice HoodClassicsUsers.
 - townID (FK) - grad povezan s lokalnim korisnikom.
- Odnosi:
 - Lokalni korisnici su autori priča (Stories).

Ključni odnosi između tablica 1. **Priče i korisnici:** Priče (Stories) su povezane s korisnicima (HoodClassicsUsers) putem userID (autor priče) i mogu se prijavljivati kroz Reports. 2. **Lokacije i priče:** Priče imaju lokacije definirane putem Coordinates koje su povezane s određenim gradovima (Town). 3. **Gradovi i države:** Gradovi pripadaju određenim državama (Country) kroz strani ključ countryID. 4. **Slike i priče:** Svaka slika (Pictures) povezana je s jednom pričom putem storyID. 5. **Oznake i priče:** Priče mogu imati više oznaka kroz vezu isTagged s tablicom Tag.

NAPOMENA vezana uz implementaciju: korisnik može biti lokalac na više mesta jer bi htjeli da ne bude ograničen objavljivati na jednom mjestu

Opis tablica

U nastavku su navedene tablice koje opisuju bazu podataka aplikacije HoodClassics. Uz svaku tablicu je priložen kratki opis o svrsi tablice i atributima koje koristi.

HoodClassicsUsers

*tablica 4.01: HoodClassicsUsers | Atribut | Tip podatka | Opis varijable |
|-----|-----|-----| | userID | INT | Jedinstveni
identifikator | | username | VARCHAR(n) | Jedinstveno korisničko ime | | eMail |
VARCHAR(n) | Jedinstvena e-mail adresa | | password | VARCHAR(n) | Korisnička
lozinka | | isModerator | BOOL | Označava je li korisnik moderator |*

Tablica 4.01 pohranjuje podatke o korisnicima aplikacije, uključujući osnovne informacije poput korisničkog imena, e-mail adrese i lozinke. Također bilježi je li korisnik moderator. Tablica koristi tip podatka **INT** za userID jer je potrebno imati jedinstveni identifikator koji se lako može inkrementirati i koristi kao primarni ključ.

VARCHAR(n) se koristi za username, eMail i password jer ove vrijednosti variraju u duljini, a potrebno je osigurati pohranu alfanumeričkih podataka. Duljina n može se odabratи prema maksimalnim očekivanim vrijednostima (npr., 50 za korisnička imena).

BOOL za `isModerator` omogućava jednostavno praćenje statusa korisnika (je li moderator ili nije), uz minimalnu memorijsku potrošnju.

Reports

tablica 4.02: Reports | Atribut | Tip podatka | Opis varijable |
|-----|-----|-----| | reportID | INT | Jedinstveni identifikator | | userID | INT | ID korisnika koji je prijavljen | | hasReportedUserID | INT | ID korisnika koji je podnio prijavu | | description | VARCHAR(n) | Sadržaj prijave | | reportCategory | VARCHAR(n) | Kategorija prijave | | storyID | INT | ID povezane priče |

U tablici 4.02 se pohranjuju prijave vezane za objave ili korisnike. Sadrži informacije o tome tko je prijavio, tko je prijavljen, sadržaj prijave, kategoriju prijave i povezanost s objavom. **INT** za `reportID`, `userID`, `hasReportedUserID` i `storyID` omogućava jednostavno referenciranje i povezivanje između tablica. **VARCHAR(n)** se koristi za `description` i `reportCategory` kako bi se omogućila fleksibilnost unosa tekstualnih podataka, poput detalja prijave i njene kategorije. Duljina n može se prilagoditi očekivanom sadržaju.

Like

tablica 4.03: Like | Atribut | Tip podatka | Opis varijable |
|-----|-----|-----| | userID | INT | ID korisnika koji je video priču | | storyID | INT | ID priče koju je korisnik video | | hasSeen | BOOL | Označava je li priča bila pregledana |

Tablica 4.03 prati interakcije korisnika s objavama, uključujući informaciju o tome je li korisnik video određenu priču. **INT** za `userID` i `storyID` služi za povezivanje korisnika i priča s njihovim jedinstvenim identifikatorima.

BOOL za `hasSeen` osigurava pohranu binarne informacije o tome je li korisnik pregledao priču.

Stories

tablica 4.04: Stories | Atribut | Tip podatka | Opis varijable |
|-----|-----|-----| | storyID | INT | Jedinstveni identifikator | | text | VARCHAR(n) | Sadržaj priče | | title | VARCHAR(n) | Naslov priče | | uploadTimestamp | TIMESTAMP | Vrijeme objave priče | | townID | INT | ID mjesta povezano s pričom | | userID | INT | ID korisnika koji je objavio priču |

Tablica 4.04 je tablica u kojoj se pohranjuju sve objave korisnika, uključujući tekst, naslov, vrijeme objave, te povezanost s korisnikom i mjestom. **INT** za `storyID`, `townID` i `userID` omogućava referenciranje objava, mjesta i korisnika, olakšavajući njihovo povezivanje.

VARCHAR(n) za `text` i `title` omogućava pohranu tekstualnog sadržaja i naslova priča različite duljine, a n se prilagođava očekivanim ograničenjima (npr., naslov do 100 znakova, tekst do nekoliko tisuća).

TIMESTAMP za uploadTimestamp koristi se za precizno bilježenje vremena objave, što je ključno za kronološko praćenje.

Coordinates

tablica 4.05: Coordinates | Atribut | Tip podatka | Opis varijable |
|-----|-----|-----| | latitude | FLOAT | Geografska širina lokacije priče | | longitude | FLOAT | Geografska dužina lokacije priče | | townID | INT | ID povezanog mjesta | | storyID | INT | ID povezane priče |

Tablica 4.05 bilježi geografske koordinate povezane s objavama. Omogućava prikaz lokacije priče na interaktivnoj karti. **FLOAT** za latitude i longitude omogućava pohranu geografskih koordinata s velikom preciznošću, što je neophodno za prikaz na interaktivnoj karti.

INT za townID i storyID povezuje lokaciju s mjestom i pričom.

LocalUsers

tablica 4.06: LocalUsers | Atribut | Tip podatka | Opis varijable |
|-----|-----|-----| | townID | INT | ID mjesta | | userID | INT | ID lokalnog korisnika |

Tablica 4.06 povezuje lokalne korisnike s mjestima koja su označili kao svoj dom, omogućujući im stvaranje sadržaja specifičnog za ta mjesta. **INT** za townID i userID omogućava povezivanje korisnika s mjestima koja smatraju svojim domom, koristeći njihove jedinstvene identifikatore.

Town

tablica 4.07: Town | Atribut | Tip podatka | Opis varijable |
|-----|-----|-----| | townID | INT | Jedinstveni identifikator mjesta | | townName | VARCHAR(n) | Ime mjesta | | county | VARCHAR(n) | Naziv županije | | state | VARCHAR(n) | Naziv države | | countryID | INT | ID države |

Tablica 4.07 pohranjuje podatke o mjestima poput imena mjesta, županije, države i povezanog identifikatora države. **INT** za townID i countryID omogućava jedinstvenu identifikaciju mjesta i država, dok **VARCHAR(n)** za townName, county, i state omogućava fleksibilnu pohranu tekstualnih naziva.

Country

tablica 4.08: Country | Atribut | Tip podatka | Opis varijable |
|-----|-----|-----| | countryID | INT | Jedinstveni identifikator države | | countryName | VARCHAR(n) | Naziv države |

Tablica 4.08 sadrži informacije o državama, uključujući naziv države i jedinstveni identifikator. **INT** za countryID služi kao jedinstveni identifikator država, dok **VARCHAR(n)** za countryName omogućava pohranu naziva država različite

duljine. ### Pictures *tablica 4.09: Pictures | Atribut | Tip podatka | Opis varijable*
| |-----|-----|-----| | href | VARCHAR(n) | URL slike | |
| storyID | INT | ID povezane priče |

U tablicu 4.09 se pohranjuju URL-ovi slika povezanih s objavama, čime se obogaćuje vizualni sadržaj priča. **VARCHAR(n)** za href koristi se za pohranu URL-ova slika, s fleksibilnošću da primi putanju različite duljine. INT za storyID povezuje sliku s odgovarajućom pričom.

Tag

tablica 4.10: Tag | Atribut | Tip podatka | Opis varijable
|-----|-----|-----| | tagID | INT | Jedinstveni identifikator
oznake | | tagName | VARCHAR(n) | Naziv oznake |

Tablica 4.10 pohranjuje informacije o oznakama koje korisnici mogu koristiti za kategorizaciju priča (npr. sport, umjetnost). INT za tagID koristi se za jedinstvenu identifikaciju oznaka, dok **VARCHAR(n)** za tagName omogućava pohranu naziva oznaka, fleksibilno podržavajući različite duljine naziva.

isTagged

tablica 4.11: isTagged | Atribut | Tip podatka | Opis varijable
|-----|-----|-----| | storyID | INT | ID priče koja je
označena | | tagID | INT | ID povezane oznake |

Tablica 4.11 povezuje priče s oznakama, omogućujući pretraživanje i filtriranje priča prema tematskim kategorijama. INT za storyID i tagID omogućava povezivanje priča s odgovarajućim oznakama, koristeći njihove jedinstvene identifikatore.

Dijagram razreda

dijagram 4.03: dijagram razreda web aplikacije HoodClassics

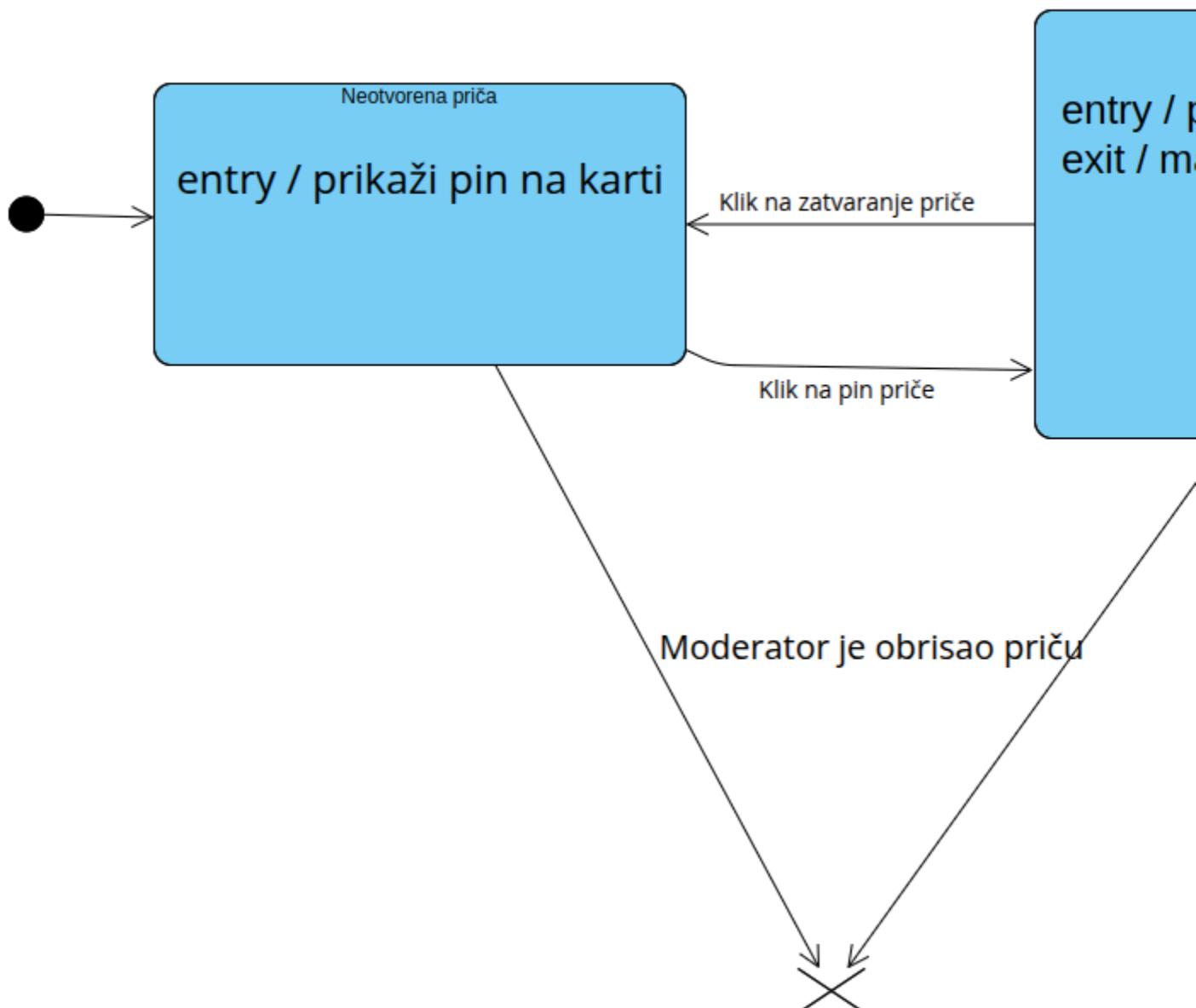
© ⌂ Hood

- (f) ↗ isModerator
- (f) ↗ password
- (f) ↗ userId
- (f) ↗ username
- (f) ↗ email
- (m) ⌂ setEmail
- (m) ⌂ setUsername
- (m) ⌂ setIsModerator
- (m) ⌂ getUsername
- (m) ⌂ getEmail
- (m) ⌂ setPassword
- (m) ⌂ toString()
- (m) ⌂ getUserId
- (m) ⌂ getPassword

Dinamičko ponašanje aplikacije

UML dijagrami stanja

dijagram 4.04: UML dijagram stanja za pregled priče u objavi



Dijagram prikazuje kako stanje priče (objekta) mijenja svoje stanje tijekom vremena, ovisno o interakcijama korisnika i uvjetima unutar sustava. Fokus je na omogućavanju pregleda priče u objavi, uključujući događaje koji utječu na pristupačnost priče.

Objekt sustava: Priča unutar objave.

Stanja objekta:

1. **Neotvorena priča**
 - Početno stanje u kojem priča nije pregledana od strane korisnika.
 - Korisnik nije izvršio nikakvu radnju na priči.
2. **Otvorena priča**
 - Priča je pregledana nakon što korisnik klikne na objavu.
 - Korisnik ima mogućnost čitanja sadržaja priče.
3. **Nedostupna priča** (implicitno stanje)
 - Stanje koje nastaje kada moderator obriše priču. U ovom slučaju, priča više nije dostupna za pregled bez obzira na korisničku interakciju.

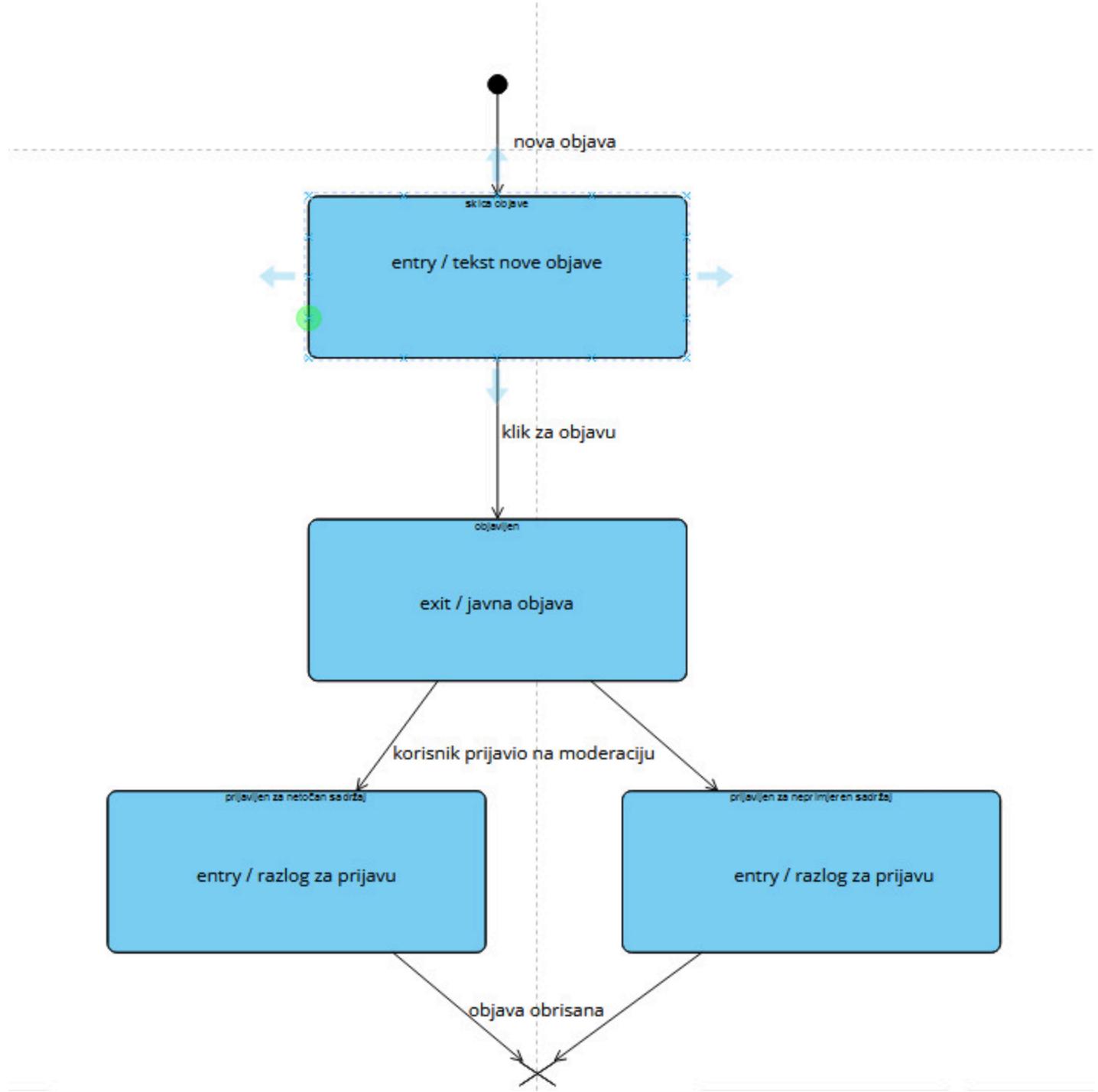
Dogadaji i uvjeti prijelaza:

1. **Klik na objavu**
 - Dogadaj koji uzrokuje prijelaz iz stanja *Neotvorena priča* u stanje *Otvorena priča*.
 - Preduvjet: Priča mora postojati i ne smije biti obrisana.
2. **Moderatorska akcija: Brisanje priče**
 - Uzrokuje prijelaz u implicitno stanje *Nedostupna priča*.
 - Priča postaje trajno nedostupna korisnicima, bez mogućnosti povratka u prethodna stanja.

Hijerarhija stanja:

Za ovaj dijagram nije potrebna složena hijerarhija stanja, jer je dinamika sustava jednostavna. Sva stanja su ravnopravna i povezana prijelazima.

dijagram 4.05: UML dijagram stanja za izradu objave



Dijagram prikazuje kako stanje objave (objekta) mijenja svoje stanje tijekom vremena, ovisno o interakcijama korisnika i uvjetima unutar sustava. Fokus je na omogućavanju stvaranja i upravljanja objavama, uključujući prijave zbog različitih razloga.

Objekt sustava:

Objava u sustavu.

Stanja objekta:

1. Skica

- Početno stanje u kojem objava nije dovršena ni objavljena.
- Korisnik unosi ili uređuje sadržaj, no objava još nije vidljiva drugim korisnicima.

2. Objavljena objava

- Stanje u kojem je objava završena i postala javno dostupna.
- Korisnici mogu pregledavati sadržaj objave.

3. Prijavljena za neispravan sadržaj

- Stanje u kojem je objava prijavljena zbog tehničkih ili formalnih problema (npr. neispravan format ili netočan sadržaj).
- Objavu može prijaviti bilo koji korisnik. Moderator zatim pregledava prijavu i poduzima odgovarajuće radnje.

4. Prijavljena za neprimjeren sadržaj

- Stanje u kojem je objava prijavljena zbog kršenja pravila zajednice (npr. uvredljiv sadržaj, govor mržnje).
- Nakon prijave, moderator može odlučiti obrisati ili urediti objavu.

Dogadaji i uvjeti prijelaza:

1. Spremanje skice

- Dogadaj koji uzrokuje prelazak u stanje *Skica*.
- Preduvjet: Korisnik započinje unos sadržaja, ali ne izvršava objavu.

2. Objava sadržaja

- Uzrokuje prijelaz iz stanja *Skica* u stanje *Objavljena objava*.
- Preduvjet: Korisnik dovršava unos sadržaja i potvrđuje objavu.

3. Prijava za neispravan sadržaj

- Uzrokuje prijelaz iz stanja *Objavljena objava* u stanje *Prijavljena za neispravan sadržaj*.
- Preduvjet: Korisnik prijavljuje objavu zbog tehničkih ili formalnih razloga.

4. Prijava za neprimjeren sadržaj

- Uzrokuje prijelaz iz stanja *Objavljena objava* u stanje *Prijavljena za neprimjeren sadržaj*.
- Preduvjet: Korisnik prijavljuje objavu zbog kršenja pravila zajednice.

5. Moderatorska akcija: Brisanje ili izmjena

- Moderator odlučuje o dalnjem postupanju s prijavljenim objavama, što može uključivati:
 - Brisanje objave (implicitno stanje: *Nedostupna objava*).
 - Povratak objave u stanje *Objavljena objava* nakon uređivanja.

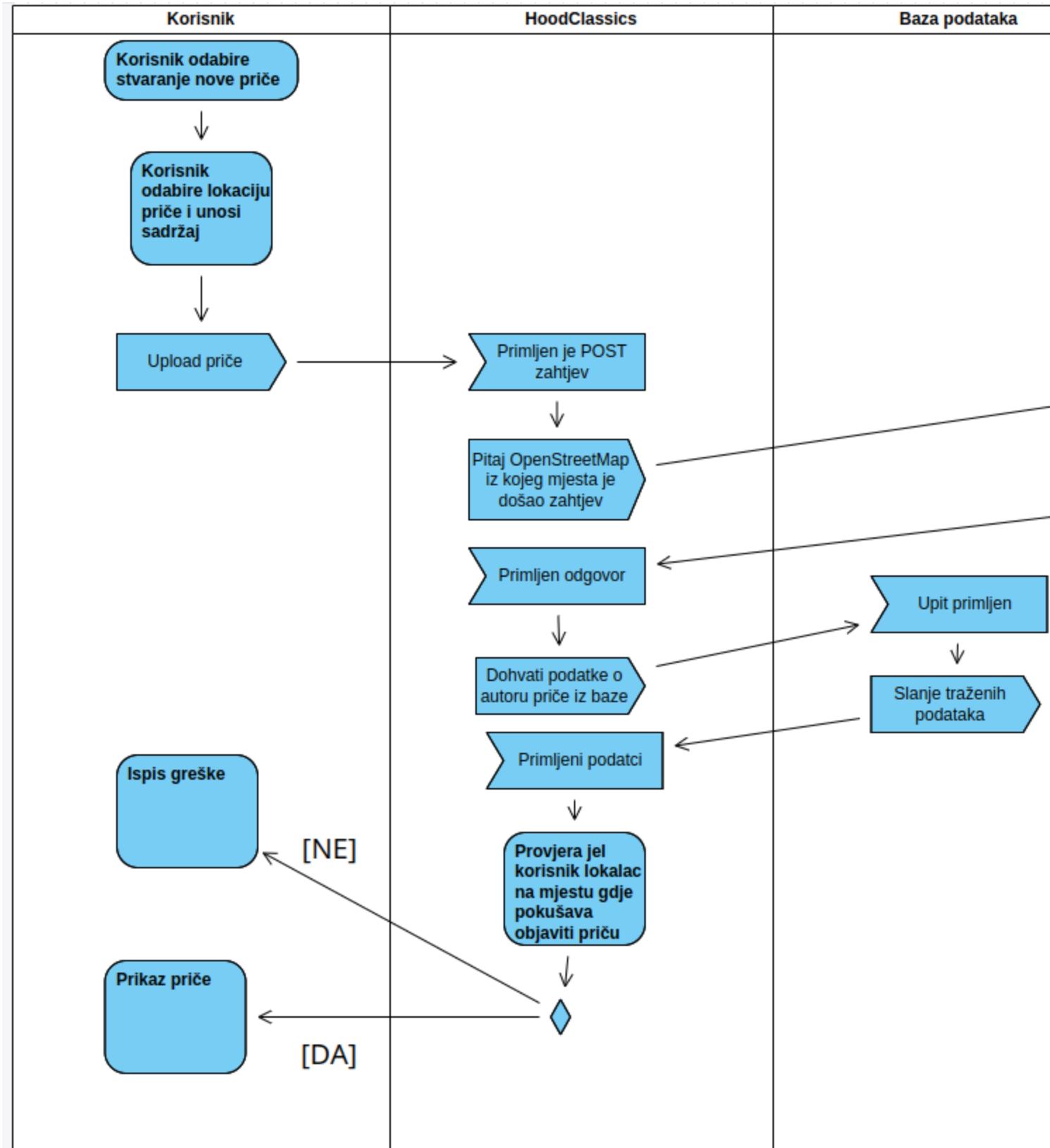
Hijerarhija stanja:

- *Skica* i *Objavljena objava* su osnovna stanja.
- *Prijavljena za neispravan sadržaj* i *Prijavljena za neprimjeren sadržaj* predstavljaju nadređene podkategorije za prijavljene objave.
- U slučaju brisanja, sustav prelazi u implicitno stanje *Nedostupna objava*

koje je trajno i nepovratno.

UML dijagrami aktivnosti

dijagram 4.06: UML dijagram aktivnosti za izradu objave



UML dijagram aktivnosti za proces izrade nove objave prikazuje tijek izvršavanja koraka u sustavu prilikom kreiranja priče i njene objave u aplikaciji. Dijagram jasno razdvaja radnje korisnika, web aplikacije, baze podataka i vanjskog servisa.

OpenStreetMap te prikazuje međusobnu interakciju ovih elemenata.

Identifikacija procesa:

Proces koji se modelira je **Izrada nove objave**. Glavni cilj procesa je omogućiti korisniku da kreira i objavi novu priču na lokaciji kojoj pripada. Ako korisnik nije lokalac na odabranoj lokaciji, sustav prikazuje grešku.

Aktivnosti unutar procesa:

1. Korisničke radnje:

- **Odabir stvaranja nove priče:** Korisnik inicira proces odabirom opcije za stvaranje nove priče.
- **Odabir lokacije i unos sadržaja:** Korisnik specificira lokaciju priče i unosi tekstualni sadržaj.
- **“Upload” priče:** Korisnik potvrđuje unos i šalje zahtjev za objavu.

2. Radnje web aplikacije:

- **Primanje POST zahtjeva:** Sustav prima zahtjev za objavu priče.
- **Komunikacija s OpenStreetMap:** Sustav šalje koordinate odabrane lokacije OpenStreetMap servisu za dobivanje naziva mjesta.
- **Dohvaćanje podataka o autoru priče:** Sustav preuzima podatke o korisniku iz baze kako bi provjerio njegov status lokalca.
- **Provjera korisnikova statusa:** Sustav provjerava je li korisnik lokalac na odabranoj lokaciji.

3. Interakcije s vanjskim sustavima:

- **OpenStreetMap:** Prima upit s koordinatama i vraća adresu vezanu uz njih.
- **Baza podataka:** Prima zahtjev za dohvaćanje korisničkih podataka i vraća potrebne informacije.

4. Ishodi:

- **Objava priče:** Ako je korisnik lokalac, priča se uspješno objavljuje.
- **Prikaz greške:** Ako korisnik nije lokalac, sustav prikazuje poruku o pogrešci.

Tok procesa:

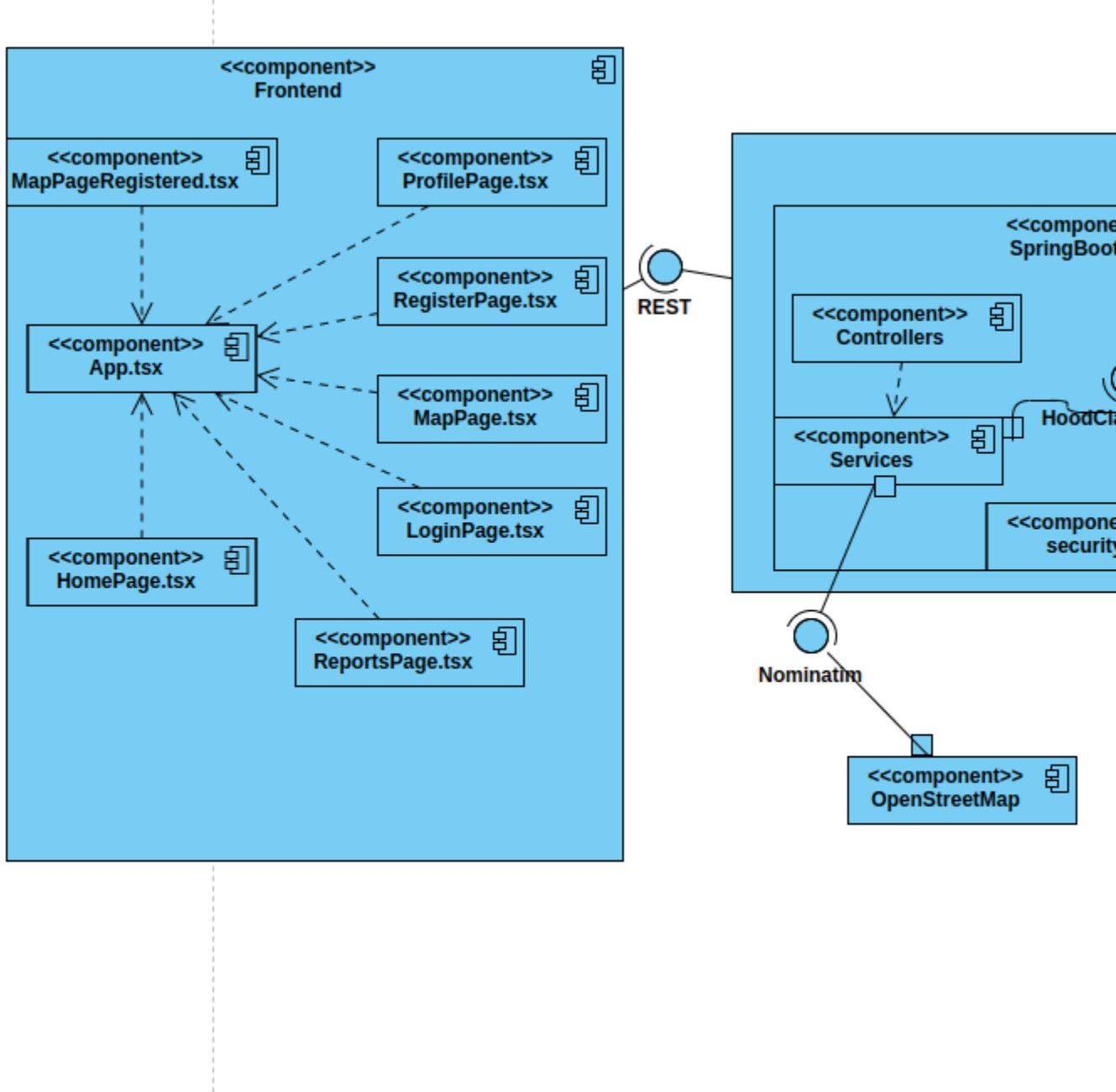
1. Početni pseudo-čvor označava početak procesa.
2. Korisnik inicira proces biranjem opcije za stvaranje priče.
3. Aplikacija prima zahtjev i šalje upit OpenStreetMap servisu za dobivanje naziva mjesta na temelju koordinata.
4. Nakon što se vrati odgovor od OpenStreetMap, sustav preuzima podatke o korisniku iz baze.
5. Provodi se provjera statusa korisnika:
 - Ako je korisnik lokalac na odabranoj lokaciji, sustav objavljuje priču.
 - Ako nije, prikazuje poruku o pogrešci.
6. Proces završava objavom priče ili prikazom greške.

Pseudo-čvorovi:

- Početni čvor:** Označava početak procesa.
- Odluka:** Provjera je li korisnik lokalac vodi prema dvije grane aktivnosti (objava ili greška).
- Završni čvor:** Označava kraj procesa nakon objave priče ili prikaza greške.

Dijagram komponenata

dijagram 5.01: dijagram komponenata sustava



Dijagram 5.02 je dijagram komponenata aplikacije *HoodClassics* koji pruža pregled ključnih funkcionalnih komponenti aplikacije, njihovih odgovornosti i međusobne povezanosti. Dijagram se temelji na dvoslojnoj arhitekturi, gdje su jasno odvojeni *frontend* i *backend*, čime se osigurava modularnost i lakše

održavanje sustava. Frontend se fokusira na korisničko sučelje, dok backend obrađuje zahtjeve, povezuje se s bazom podataka i integrira vanjske API-je. Ovakva arhitektura omogućuje lako održavanje, skalabilnost i sigurnost aplikacije.

1. Frontend komponenta Frontend predstavlja korisnički sloj aplikacije i implementiran je pomoću React.js tehnologije. Sadrži glavnu komponentu App.tsx, koja povezuje sve ostale funkcionalne stranice aplikacije:

- **Podkomponente frontenda:**
 - HomePage.tsx - Početna stranica aplikacije, gdje korisnici mogu započeti navigaciju kroz aplikaciju.
 - LoginPage.tsx - Stranica za prijavu korisnika.
 - RegisterPage.tsx - Stranica za registraciju novih korisnika.
 - ProfilePage.tsx - Stranica za prikaz i uređivanje korisničkog profila.
 - MapPage.tsx i MapPageRegistered.tsx - Stranice koje prikazuju kartu s integracijom OpenStreetMap, prilagođene za goste i registrirane korisnike.
 - ReportsPage.tsx - Stranica za pregled i prijavu problema unutar zajednice.
- **Poveznice:**

Frontend komunicira s backendom putem **HTTPS REST API-ja** koji omogućuje dohvaćanje i slanje podataka (npr. za korisničku prijavu, registraciju i rad s kartama). Statički resursi (index.html) učitavaju se iz backend poslužitelja.

2. Backend komponenta Backend aplikacije implementiran je pomoću **Spring Boot frameworka** i sadrži dvije glavne komponente: - **SpringBootApp** - **SpringDBHandler**

a) **SpringBootApp**

Ova komponenta predstavlja srce backend logike, koja obrađuje korisničke zahtjeve i pruža funkcionalnost aplikacije. Sastoji se od pet ključnih podkomponenti:

1. **Controllers:**

- Ovi moduli obrađuju ulazne zahtjeve (HTTP GET i POST) od frontenda. Smješteni su u direktoriju rest. Svaki kontroler ima specifičnu ulogu:
 - HomeController - Obrada početnih zahtjeva; vraća statičke resurse (index.html).
 - StoryController - Upravljanje pričama (dohvat, kreiranje, brisanje).
 - UserController - Upravljanje korisnicima, uključujući identifikaciju njihovog grada.
- Kontroleri stvaraju sesiju i pozivaju odgovarajuće servise kako bi izvršili tražene operacije.

HomeController

```

package hoodclassics opp rest;

import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;

@RestController
@RequestMapping("/")
public class HomeController {

    @GetMapping({"/", "/map", "/mapRegistered", "/profilePage"})
    public ModelAndView welcome() {
        return new ModelAndView("forward:index.html");
    }

    // For testing purposes
    /*
    @GetMapping("/")
    public String welcome() {
        return "Welcome!";
    }
    */
}

}

```

Ova klasa HomeController je REST kontroler koji pripada Spring Boot aplikaciji. Klasa upravlja HTTP zahtjevima za određene rute:

1. Rute koje obrađuje:

- GET zahtjevi za /, /map, /mapRegistered i /profilePage preusmjeravaju korisnika na datoteku index.html koristeći metodu welcome. Ovo omogućava posluživanje klijentske aplikacije iz jednog središnjeg ulaza.

2. Preusmjeravanje:

- ModelAndView("forward:index.html") govori aplikaciji da se zahtjev preusmjeri na statičku datoteku index.html, čime se omogućuje prikaz glavne stranice aplikacije.

3. Zakomentirana metoda za testiranje:

- Zakomentirana metoda welcome vraća tekstualni odgovor "Welcome!" kao jednostavan primjer ili za testiranje funkcionalnosti bez posluživanja klijentskih datoteka.

Ključne napomene: - @RestController označava ovu klasu kao komponentu za obradu REST zahtjeva. - @RequestMapping("/") definira osnovni URL za sve zahtjeve koje kontroler obrađuje.

StoryController

REST kontroler StoryController za upravljanje pričama u aplikaciji. Ključne značajke su:

1. **Osnovne rute** (@RequestMapping("/api/story")):
 - @GetMapping("/{story_id}"): Dohvaća detalje pojedine priče prema ID-u.
 - @GetMapping("/stories"): Dohvaća priče unutar danog radijusa na temelju koordinata.
 - @PostMapping: Omogućuje dodavanje novih priča, tagova i prijava.
 - @PostMapping("/delete/{story_id}"): Briše priču ako korisnik ima prava (moderator ili autor).

2. **Sigurnost i provjere:**

- Provjera prava za moderatora (userService.isModerator()) kod zahtjeva za prijave i brisanje priča.

3. **Usluge i ovisnosti:**

- Koristi StoryService za manipulaciju podacima o pričama i UserService za provjeru korisničkih prava.

4. **Upravljanje prijavama:**

- Rute poput submitReport i getReports omogućuju korisnicima prijavljivanje neispravnih sadržaja te moderatorima dohvaćanje prijava.

5. **Dodatne klase:**

- CreateStoryBody, AddTagToStoryBody, i TaggedStoriesBody služe za mapiranje podataka zahtjeva na objekte.

UserController

REST kontroler UserController za upravljanje korisnicima i njihovim interakcijama s aplikacijom. Ključni dijelovi su:

1. **Rute za upravljanje korisnicima:**

- @GetMapping("/list"): Vraća popis svih korisnika.
- @GetMapping("/moderator"): Provjerava je li korisnik moderator, uz odgovarajuće HTTP statuse.

2. **Dodavanje korisnika u gradove:**

- @PostMapping("/town"): Prima podatke o gradovima (koordinate) i poziva userService.addToTown za dodavanje korisnika u navedene gradove.

3. **Pomoćna klasa TownRequest:**

- Služi za mapiranje JSON zahtjeva s popisom gradova, uključujući njihove geografske koordinate (lat i lng).

4. **Sigurnost (komentirano):**

- Kod za testiranje sigurnosnih konteksta (getAuthorities, getName) prikazan je, ali nije aktivan.

5. **Services:**

- Posrednik između kontrolera i ostalih komponenti sustava. Ovdje se odvijaju poslovne operacije i komunikacija s vanjskim servisima.
- Komponenta Service koristi **OpenStreetMap API** putem **Nominatim geocoding servisa** za dohvaćanje koordinata i identifikaciju gradova.

- Servisi također komuniciraju s bazom podataka (putem Repos sučelja) za obradu i upravljanje podacima.

6. **Repos:**

- Sučelja smještena u direktoriju dao, omogućuju manipulaciju podacima u bazi. Svaki repo sučelje odnosi se na određenu tablicu u bazi podataka (npr. korisnici, priče, gradovi).
- Povezani su sa Servisima, omogućujući jednostavan i strukturiran rad s bazom.

7. **Domain:**

- Klase koje predstavljaju modele podataka u skladu s ER dijagramom. Svaka klasa mapira tablicu u bazi podataka i sadrži atribute poput korisničkih podataka, priča, lokacija itd.

8. **Security:**

- Ova komponenta implementira autentifikaciju i autorizaciju korisnika putem **Google OAuth 2.0** protokola.
- Funkcionalnosti uključuju prijavu korisnika, upravljanje sesijama i zaštitu API poziva.
- Komponenta je povezana s vanjskim servisom GoogleOAuth pomoću "ručice" koja šalje zahtjeve, dok "kružić" predstavlja povratne odgovore.

b) SpringDBHandler

Ova komponenta obuhvaća funkcionalnost interakcije s bazom podataka. Kroz "Spring Data" i **Repos** sučelja omogućuje kreiranje, dohvaćanje, ažuriranje i brisanje podataka unutar baze. Baza podataka generira se iz koda, a struktura podataka temelji se na klasama iz **Domain** komponente.

Vanjske komponente: 1. **OpenStreetMap API:**

- Komponenta zadužena za dohvaćanje geografskih informacija (karte, koordinati gradova). Backend komunicira s njom koristeći **Nominatim API**.

2. **GoogleOAuth:**

- Autentifikacijska komponenta koja omogućuje korisnicima prijavu putem Google računa. Implementirana je pomoću **OAuth 2.0 protokola**.

Poveznice između elemenata - Frontend i Backend:

Frontend šalje zahtjeve na backend putem REST API-ja. Backend obrađuje zahtjeve, poziva servise i vraća odgovore frontendu.

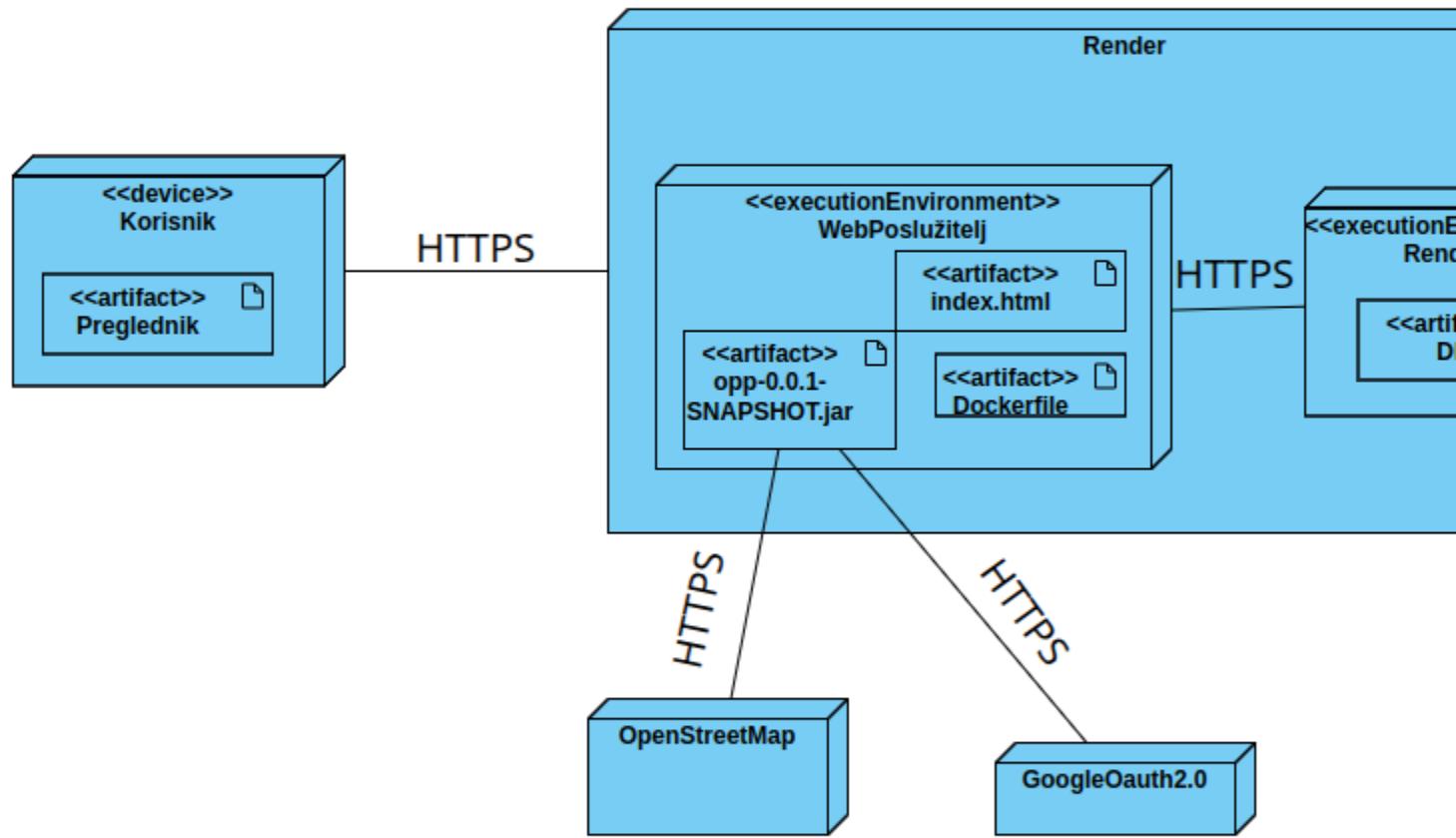
- Backend i Vanjski servisi:

- Servisi unutar backenda šalju zahtjeve vanjskim API-jevima (OpenStreetMap, GoogleOAuth) kako bi dobili dodatne podatke potrebne za obradu korisničkih zahtjeva. - **Backend i Baza podataka:**

SpringDBHandler osigurava pristup bazi podataka koristeći sučelja iz **Repos** komponente.

Dijagram razmještaja

dijagram 5.02: dijagram razmještaja sustava



Dijagram 5.02 je UML dijagram razmještaja prikazuje fizički i virtualni raspored komponenata aplikacije, njihovu međusobnu povezanost i način komunikacije s vanjskim servisima. Ovaj dijagram obuhvaća dva glavna elementa: **Korisnik** i **Render**. Cilj dijagrama je ilustrirati kako aplikacija funkcioniira na visokoj razini, uključujući interakcije korisnika s aplikacijom, unutarnju obradu zahtjeva te integraciju s vanjskim servisima.

Elementi dijagrama razmještaja 1. Korisnik - Artifact: Preglednik (browser). Korisnik komunicira s aplikacijom putem web-preglednika, šaljući *HTTPS zahtjeve* za funkcionalnosti kao što su prijava, registracija, dohvaćanje priča, te rad s

kartama. - **Komunikacija:** Korisnik šalje zahtjeve direktno na *WebPoslužitelj* unutar komponente *Render*.

2. Render

- **Execution Environment:** WebPoslužitelj.
 - **Artifacti:**
 - index.html – Statički resurs za frontend aplikacije.
 - opp.0.0.1-SNAPSHOT.jar – Radna verzija aplikacije (backend). Upravo ovaj artifact obrađuje sve zahtjeve korisnika.
 - Dockerfile – Konfiguracijska datoteka koja definira varijable okruženja, postavlja radnu okolinu i pokreće aplikaciju unutar Docker kontejnera.
 - **Komunikacija:**
 - Backend (opp.0.0.1-SNAPSHOT.jar) obrađuje zahtjeve i komunicira s različitim servisima:
 1. **OpenStreetMap** – Slanje **HTTPS zahtjeva** za prikaz karte i dohvaćanje koordinata kako bi se pronašao grad.
 2. **Google OAuth 2.0** – Slanje **HTTPS zahtjeva** za autentifikaciju korisnika putem Google računa.
 3. **RenderDB** – Slanje **HTTPS zahtjeva** za pristup i upravljanje podacima u bazi podataka.
 - **Execution Environment:** RenderDB.
 - **Artifact:** DB (baza podataka) – Baza u kojoj se pohranjuju podaci aplikacije, poput korisničkih računa, priča, oznaka, i drugih informacija potrebnih za rad aplikacije.

Funkcionalnosti aplikacije na široj razini 1. Korisnički zahtjevi:

Korisnik putem preglednika šalje **HTTPS zahtjeve** za različite funkcionalnosti, uključujući: - **Prijavu i registraciju** – Autentifikacija putem Google OAuth 2.0 servisa. - **Dohvat priča** – Dobivanje podataka iz baze (RenderDB). - **Rad s kartama** – Prikaz i navigacija na karti pomoću OpenStreetMap servisa.

2. Obrada na WebPoslužitelju:

- **Dockerfile:** Postavlja okruženje za aplikaciju i pokreće backend komponentu opp.0.0.1-SNAPSHOT.jar. Ova komponenta:
 - Obradjuje korisničke zahtjeve.
 - Servira statičke datoteke (index.html) za frontend aplikaciju.
 - Komunicira s:
 - **OpenStreetMap** za prikaz i navigaciju na karti.
 - **Google OAuth 2.0** za autentifikaciju korisnika.
 - **RenderDB** za rad s podacima (korištenje baze podataka).

3. Komunikacija s bazom podataka:

- WebPoslužitelj šalje zahtjeve prema **RenderDB**, koja upravlja svim podacima aplikacije. Ovo uključuje korisničke račune, priče, geografske oznake i druge potrebne informacije.

Ovo poglavlje treba opisati provedena ispitivanja implementiranih funkcionalnosti na razini komponenti i sustava. Fokus je na odabiru i izvedbi ispitnih slučajeva koji obuhvaćaju redovne, rubne uvjete i testiranje grešaka, kao i upotrebu odgovarajućih alata za provedbu testiranja.

Ispitivanje komponenti

Pronalaženje mjesta iz koordinata i obrnuto | Opis | Ulagni podatci | Očekivani podatci | Dobiveni rezultati |

|:-----:|:-----:|:-----:|:-----:|
Uspješno nalaženje lokacije iz adrese | "New York, USA" | "New York" | "New York" | | **Neuspješno nalaženje lokacije iz adrese** (lokacija ne postoji) | "" | empty | empty | | **Uspješno nalaženje države iz adrese** | "New York, USA" | "USA" | "USA" | | **Nesuspješno nalaženje lokacije iz adrese** | "" | empty | empty | | **Uspješno pronalaženje adrese iz koordinata** | 40.7128, -74.0060 | "City of New York, United States" | "City of New York, United States" | | **Neuspješno pronalaženje adrese iz koordinata** | 92, 182 | empty | empty |

Postupak tesiranja smo proveli uz pomoć JUnit-a i Mockito-a. Testove smo evaluirali tako da smo pokrenuli klasu za testiranje.

link: ## funkcionalnost koju testiramo (ime) | Opis | Ulagni podatci | Očekivani podatci | Dobiveni rezultati |

|:-----:|:-----:|:-----:|:-----:|
Uspješno nalaženje lokacije iz adrese | "New York, USA" | "New York" | "New York" | | **Neuspješno nalaženje lokacije iz adrese** (lokacija ne postoji) | "" | empty | empty | | **Uspješno nalaženje države iz adrese** | "New York, USA" | "USA" | "USA" |

Postupak tesiranja smo proveli uz pomoć JUnit-a i Mockito-a. Testove smo evaluirali tako da smo pokrenuli klasu za testiranje.

link: # Ispitivanje sustava

Sljedeći testovi su napravljeni pomoću alata Selenium

1. RegisterTest

Koraci: 1. Otvoriti stranicu. 2. Pritisnuti gumb **Sign Up** —> Korisnik je preusmjeren na stranicu za registraciju. 3. Unijeti **username**: "testUser". 4. Unijeti **password**: "123". 5. Pritisnuti gumb **Register** —> Korisnik je preusmjeren na stranicu **Profile**. 6. Na stranici **Profile**, odabrati lokaciju na karti odakle je korisnik. 7. Pritisnuti gumb **Save Places** —> Korisnik je preusmjeren na kartu.

Očekivani izlaz: Korisnik uspješno registriran i preusmjeren na kartu s odabranim lokacijama.

Dobiveni izlaz: Test uspješno završen; korisnik je registriran i preusmjeren prema očekivanjima.

Rubni slučaj:

- Pokušaj registracije korisnika s postojećim **usernameom** prikazuje poruku: "Korisničko ime već postoji."

slika 6.05: RegisterTest

Running 'registerUser'

1. open on / **OK**
2. setWindowSize on 1800x1040 **OK**
3. click on linkText=Sign Up **OK**
4. click on id=:r0: **OK**
5. type on id=:r0: with value testUser **OK**
6. click on id=:r1: **OK**
7. mouseOver on css=.MuiButton-contained **OK**
8. type on id=:r1: with value password **OK**
9. click on css=.MuiButton-contained **OK**
10. Trying to find css=.leaflet-container... **OK**
11. click on css=.MuiButton-root **OK**

'registerUser' completed successfully

2. LoginTest

Koraci: 1. Otvoriti stranicu. 2. Pritisnuti gumb **Sign In** —> Korisnik je preusmjeren na stranicu za prijavu. 3. Unijeti **username**: "testUser". 4. Unijeti **password**: "123". 5. Pritisnuti gumb **Login** —> Korisnik je preusmjeren na kartu.

Očekivani izlaz: Korisnik uspješno prijavljen i preusmjeren na kartu.

Dobiveni izlaz: Test uspješno završen; korisnik je prijavljen prema očekivanjima.

Rubni slučajevi:

- Pogrešno **username** ili **password** prikazuje poruku: "Pogrešno korisničko ime ili lozinka."

slika 6.05: LoginTest

Running 'loginTest'

1. open on / **OK**
2. setWindowSize on 1800x1040 **OK**
3. mouseOver on linkText=Sign Up **OK**
4. click on linkText=Sign In **OK**
5. click on id=:r0: **OK**
6. type on id=:r0: with value test **OK**
7. click on id=:r1: **OK**
8. mouseOver on css=.MuiButton-contained **OK**
9. type on id=:r1: with value 123 **OK**
10. click on css=.MuiButton-contained **OK**

'loginTest' completed successfully

3. AddPinTest

Koraci: 1. Otvoriti stranicu. 2. Pritisnuti gumb **Sign In** —> Korisnik je preusmjeren na stranicu za prijavu. 3. Unijeti **username**: "testUser". 4. Unijeti **password**: "123". 5. Pritisnuti gumb **Login** —> Korisnik je preusmjeren na kartu. 6. Pritisnuti gumb **+** za dodavanje priče. 7. Kliknuti na kartu na željenu lokaciju za

dodavanje priče —> Otvara se ladica s desne strane. 8. Unijeti **title**: "Test". 9. Unijeti **text**: "Test adding story". 10. Označiti tag priče. 11. Pritisnuti gumb **Add Pin**.

Očekivani izlaz: Priča je uspješno dodana na odabranu lokaciju na karti.

Dobiveni izlaz: Test uspješno završen; priča je dodana prema očekivanjima.

Rubni slučaj:

- Korisnik nije lokalac na odabranoj lokaciji — prikazuje poruku o grešci: "Nije moguće dodati priču jer niste lokalac."

4. ReportTest

Koraci: 1. Otvoriti stranicu. 2. Pritisnuti gumb **Sign In** —> Korisnik je preusmjeren na stranicu za prijavu. 3. Unijeti **username**: "testUser". 4. Unijeti **password**: "123". 5. Pritisnuti gumb **Login** —> Korisnik je preusmjeren na kartu. 6. Kliknuti na **pin** postojeće priče. 7. U ladici na desnoj strani odabratи razlog za prijavu i unijeti opis prijave. 8. Pritisnuti gumb **Report**.

Očekivani izlaz: Priča je uspješno prijavljena, a sustav bilježi prijavu.

Dobiveni izlaz: Test uspješno završen; prijava priče evidentirana prema očekivanjima.

slika 6.06: ReportTest

Running 'reportTest'

1. open on / **OK**
2. setWindowSize on 1800x1040 **OK**
3. click on linkText=Sign In **OK**
4. click on id=:r0: **OK**
5. type on id=:r0: with value test **OK**
6. click on id=:r1: **OK**
7. doubleClick on id=:r1: **OK**
8. type on id=:r1: with value 123 **OK**
9. click on css=.MuiButton-contained **OK**
10. Trying to find css=.leaflet-marker-icon:nth-child(3)... **OK**
11. runScript on window.scrollTo(0,0) **OK**
12. mouseDown on css=.MuiSelect-select **OK**
13. mouseUp on css=.MuiBackdrop-invisible **OK**
14. click on css=body **OK**
15. click on css=.MuiButtonBase-root:nth-child(1) **OK**
16. click on id=:r3: **OK**
17. type on id=:r3: with value Test reporta **OK**
18. click on css=.css-1gm71ie **OK**

'reportTest' completed successfully

API testovi

U svrhu ispitivanja funkcionalnosti poslužitelja, napisano je nekoliko Postman kolekcija čijim se pokretanjem na poslužitelj šalje niz zahtjeva te uspoređuju stvarni i očekivani odgovori. Ovaj se pristup pokazao izrazito korisnim u otkrivanju pogrešaka i služi kao jamac stabilnosti sustava pri integraciji novih promjena.

Testovi su građeni u skladu s bazom podataka popunjrenom unaprijed definiranim testnim vrijednostima. Za izgradnju se baze koristi [SQL skripta](#) koju je potrebno izvršiti prije svakog pokretanja poslužitelja. Prije pokretanja svake kolekcije potrebno je izvršiti skriptu, pa ponovo pokrenuti poslužitelj.

U nastavku su navedene kolekcije te neki ilustrativni primjeri iz svake.

Autentifikacija

U ovoj su kolekciji ispitani sljedeći slučajevi: Registracija novog korisnika, registracija postojećeg korisnika, prijava postojećeg korisnika, odjava, odjava dok korisnik nije prijavljen, prijava nakon registracije, pokušaj prijave neispravnom zaporkom, pokušaj prijave nepostojećim korisničkim imenom.

Registracija novog korisnika

- **Što se ispituje:** POST /register koji prima korisničko ime i zaporku te u BP dodaje novog korisnika.
- **Očekivani odgovor:** 200 OK
- **Stvarni odgovor:** 200 OK

slika 6.01: Postman sučelje koje prikazuje sadržaj zahtjeva i da je ispit uspješan

POST

localhost:8080/register

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value
<input checked="" type="checkbox"/> username	new_test
<input checked="" type="checkbox"/> password	123
Key	Value

Body Cookies Headers (11) Test Results (1/1)

Filter Results ▾

PASSED Register new user

Registracija postojećeg korisnika (rubni slučaj)

- **Što se ispituje:** POST /register koji prima korisničko ime i zaporku, ali ne bi trebao dodati novog korisnika u BP, već javiti da korisnik već postoji.
- **Očekivani odgovor:** 403 FORBIDDEN “This username is already taken”
- **Stvarni odgovor:** 403 FORBIDDEN “This username is already taken”

slika 6.02:Postman sučelje koje prikazuje sadržaj zahtjeva i da je ispit uspješan

The screenshot shows the Postman interface with the following details:

- HTTP Method:** POST
- URL:** localhost:8080/register
- Headers:** (9) - x-www-form-urlencoded
- Body:** (9 items)

Key	Value
username	test
password	123
Key	Value
- Test Results:** (1/1)
 - 1 That username is already taken

Stvaranje objava i dohvaćanje objava sa poslužitelja

U ovoj su kolekciji ispitani sljedeći slučajevi: Dohvaćanje objave kao turist, tj. neautentificirani korisnik, dohvaćanje popisa objava na karti kao turist, dohvaćanje objave kao prijavljeni korisnik, pokušaj dohvaćanja nepostojeće objave, dohvaćanje popisa objava na karti kao prijavljeni korisnik, pokušaj dohvaćanja objava koje nisu u definiranom radiusu oko korisnika, stvaranje nove objave na mjestu gdje je korisnik lokalac, dohvaćanje popisa objava nakon stvaranja nove objave, pokušaj stvaranja objave na mjestu gdje korisnik nije lokalac, dohvaćanje objave s mjesta gdje korisnik nije lokalac.

Pokušaj dohvaćanja nepostojeće objave

- **Što se ispituje:** GET /api/story/100 koji pokušava dohvatiti objavu sa nepostojećim ID-om (100).
- **Očekivani odgovor:** 404 NOT FOUND Body: {}
- **Stvarni odgovor:** 404 NOT FOUND Body: {}

Ovaj je ispitni zahtjev bio posebno važan jer se pomoću njega otkrilo da poslužitelj nije obrađivao pogrešan ID kako je namijenjeno, već se događala greška (500

INTERNAL SERVER ERROR). Nakon otkrića, funkcionalnost je proširena tako da podržava i taj slučaj.

slika 6.03: Postman sučelje koje prikazuje sadržaj zahtjeva i da je ispit uspješan

The screenshot shows the Postman interface for a GET request to `localhost:8080/api/story/100`. The 'Scripts' tab is selected, containing the following Mocha-style test code:

```
1 pm.test("Failed to fetch non-existing story", function () {
2     pm.response.to.have.status(404);
3
4     const expectedBody = {};
5
6     const responseBody = pm.response.json();
7     pm.expect(responseBody).to.deep.equal(expectedBody);
8 });
9
```

The 'Test Results' tab shows one test result: **PASSED** Failed to fetch non-existing story.

Tagovi i pretraživanje po tagovima

U ovoj su kolekciji ispitani sljedeći slučajevi: Dodavanje taga objavi u ulozi turista, tj. neprijavljenog korisnika, dohvaćanje popisa objava filtriranih po 2 taga u ulozi turista, pokušaj dodavanja taga objavi koja je njime već označena, pokušaj dodavanja nepostojećeg taga objavi, pokušaj dodavanja taga nepostojećoj objavi, dodavanje novog taga objavi, dohvaćanje popisa objava filtriranih po novom tagu.

Dohvaćanje popisa objava filtriranih po 2 taga u ulozi turista

- **Što se ispituje:** GET /api/story/taggedstories sa tijelom zahtjeva { "latitude" : "45.8150", "longitude" : "15.9819", "radius" : "10", "tags" : ["1", "2"] }
- **Očekivani odgovor:** 200 OK Body: [{ "longitude": 15.979814251930389, "latitude": 45.81466552408981, "storyId": 1, "likes": 1 }, { "longitude": 15.971270141762062, "latitude": 45.800873703358214, "storyId": 2, "likes": 0 }]

- **Stvarni odgovor:** 200 OK Body: [{ "longitude": 15.979814251930389, "latitude": 45.81466552408981, "storyId": 1, "likes": 1 }, { "longitude": 15.971270141762062, "latitude": 45.800873703358214, "storyId": 2, "likes": 0 }]

slika 6.04: Postman sučelje koje prikazuje sadržaj zahtjeva i da je ispit uspješan

HTTP Tags / Fetch pins with 2 tags (tourist)

POST localhost:8080/api/story/taggedstories

Params Authorization Headers (10) Body Scripts Settings

Body (raw JSON)

```

1  {
2    "latitude" : "45.8150",
3    "longitude" : "15.9819",
4    "radius" : "10",
5    "tags" : [
6      "1",
7      "2"
8    ]
9  }

```

Body Cookies (1) Headers (11) Test Results (1/1) | ⏪

Filter Results ▾

PASSED Fetched correct story pins

Korištene tehnologije i alati

U ovom poglavlju su nabrojani alati i tehnologije korišteni prilikom razvoja naše web aplikacije. Uz svaki su navedeni njihova funkcija i zašto su odabrani.

tablica 7.01: korištene tehnologije i alati

redni broj	ime tehnologije/ alata	verzija	kategorija
01	Java	21.0.5	programske jezike
02	TypeScript	5.6.3	programske jezike
03	Spring Boot	3.3.5	radni okvir
04	React	18	radni okvir

redni broj	ime tehnologije/ alata	verzija	kategorija
05	Vite	5.4.10	radni okvir
06	Node	22.10.5	radni okvir
07	Open Street map	-	radni okvir
08	Nominatim	-	radni okvir
09	PostgreSQL	16	baza podataka
10	Eclipse IDE	4.34.0	razvojni alat
11	Visual Studio Code	1.96.3	razvojni alat
12	IntelliJ IDea Ultimate	2024.3.1.1	razvojni alat
13	IntelliJ Community		razvojni alat
14	Git	2.43	razvojni alat
15	Postman	11.23.3	alat za ispitivanje
16	DevTools	-	alat za ispitivanje
17	Selenium	-	alat za ispitivanje
18	Docker	20.10.	alat za razmještaj

Programski jezici

Java (verzija 21.0.5)

Java je univerzalni, objektno orijentirani programski jezik poznat po stabilnosti i dugotrajnoj podršci. U našem projektu Java je odabrana za razvoj poslužiteljskog dijela (backenda) aplikacije jer je industrijski standard i pruža bogatu ekosustav alata i biblioteka. Verzija 21.0.5 je dugoročno podržavana (LTS), što osigurava pouzdanost i sigurnosna ažuriranja.

TypeScript (verzija 5.6.3)

TypeScript je nadogradnja JavaScripta koja uvodi statičko tipiziranje, što doprinosi većoj sigurnosti i održivosti koda. U našem projektu koristi se za razvoj klijentskog dijela (frontenda) aplikacije jer je industrijski standard i omogućava kvalitetniji razvoj složenih aplikacija. Verzija 5.6.3 osigurava kompatibilnost s modernim bibliotekama i alatima.

Radni okviri

Spring Boot (verzija 3.3.5)

Spring Boot je popularan radni okvir za razvoj Java aplikacija koji olakšava postavljanje poslužitelja i upravljanje ovisnostima. Odabran je za backend zbog svoje stabilnosti, fleksibilnosti i mogućnosti integracije s različitim alatima i bazama podataka. Verzija 3.3.5 pruža napredne funkcionalnosti koje pojednostavljaju razvoj web aplikacija.

React (verzija 11.13.3)

React je biblioteka za razvoj korisničkih sučelja koja omogućava izradu interaktivnih i responzivnih web aplikacija pomoću komponentnog pristupa. U našem projektu koristi se za frontend, omogućujući brzo i učinkovito razvijanje

sučelja. Verzija 11.13.3 je industrijski standard s podrškom za modernu ekosustav.

Vite (verzija 5.4.10)

Vite je alat za brzi razvoj frontend aplikacija koji koristi naprednu tehniku "hot module replacement". Koristi se za razvoj i testiranje aplikacije u Reactu, omogućujući brzo učitavanje i responzivnost tijekom razvoja. Verzija 5.4.10 osigurava pouzdanu podršku za moderne projekte.

Node.js (verzija 22.10.5)

Node.js je runtime okruženje koje omogućava izvođenje JavaScript koda izvan preglednika. Koristi se za podršku razvojnog procesu frontenda, kao i za upravljanje ovisnostima pomoću npm-a. Verzija 22.10.5 osigurava stabilnost i kompatibilnost s modernim alatima.

OpenStreetMap (OSM)

OpenStreetMap je platforma za izradu besplatnih i otvorenih kartografskih podataka. U našem projektu koristi se za prikaz interaktivnih karata koje omogućuju korisnicima navigaciju i pregled turističkih lokacija. Njegova besplatna priroda čini ga idealnim za našu aplikaciju.

Nominatim

Nominatim je alat za geokodiranje koji omogućava pretvaranje adresa u geografske koordinate. Koristi se zajedno s OpenStreetMap-om za pretragu lokacija u aplikaciji. Njegova integracija osigurava preciznost i besplatno rješenje za geolokaciju.

Baza podataka

PostgreSQL (verzija 16)

PostgreSQL je moćan sustav za upravljanje relacijskim bazama podataka. U našem projektu koristi se kao glavna baza podataka zbog svoje pouzdanosti, skalabilnosti i podrške za složene upite. Verzija 16 osigurava najnovije značajke i optimizacije.

Razvojni alati

Eclipse IDE (verzija 4.34.0)

Eclipse je integrirano razvojno okruženje koje se koristi za razvoj Java aplikacija. Odabранo je za backend razvoj jer smo već upoznati s alatom, čime se smanjuje vrijeme učenja. Verzija 4.34.0 nudi poboljšanu podršku za moderne Java projekte.

Visual Studio Code (verzija 1.96.3)

Visual Studio Code je popularno integrirano razvojno okruženje koje podržava širok spektar jezika i alata. U našem projektu koristi se za razvoj i testiranje frontenda zbog jednostavnosti, fleksibilnosti i podrške za proširenja.

IntelliJ IDEA Ultimate (verzija 2024.3.1.1)

IntelliJ IDEA Ultimate je napredno razvojno okruženje koje pruža bogat skup alata za razvoj aplikacija u Javi i TypeScriptu. Koristi se za frontend i backend razvoj zbog svoje duboke integracije s alatima poput Spring Boota i Reacta.

IntelliJ Community

IntelliJ Community je besplatna verzija IntelliJ IDEA koja pruža osnovne značajke za razvoj aplikacija. Koristi se za backend razvoj kao pristupačna alternativa Ultimate verziji.

Git (verzija 2.43)

Git je sustav za verzioniranje koji omogućava praćenje promjena u kodu i suradnju u timu. Koristi se za upravljanje kodom na svim razinama projekta, osiguravajući povijest i sigurnost podataka.

Postman (verzija 11.23.3)

Postman je alat za testiranje REST API-ja koji omogućava slanje zahtjeva i analizu odgovora. Koristi se za ispitivanje backend funkcionalnosti i provjeru ispravnosti API poziva.

Alati za ispitivanje

DevTools

DevTools su alati integrirani u preglednike za ispitivanje i otklanjanje grešaka u frontend aplikacijama. Koriste se za analizu performansi i stilova korisničkog sučelja.

Selenium

Selenium je alat za automatizirano testiranje web aplikacija. U našem projektu koristi se za provjeru ispravnosti korisničkog sučelja kroz simulaciju interakcije s aplikacijom.

Alati za razmještaj

Docker (verzija 20.10)

Docker je platforma za kontejnerizaciju koja omogućava jednostavno pakiranje i razmještaj aplikacija. Koristi se za osiguravanje dosljednog okruženja između razvoja i produkcije.

Dodatni alati

Za komunikaciju i planiranje koristili su se: - WhatsApp - Discord - Trello

Ovaj odjeljak dokumentacije treba dati detaljne smjernice za instalaciju, konfiguraciju, pokretanje i administraciju aplikacije. Cilj je olakšati postavljanje aplikacije na razvojnem, ispitnom i produkcijskom okruženju.

Upute za lokalno testiranje

1. Instalacija

- **Preduvjeti:** Node.js 16, Java 17, Maven 3.9.9, PostgreSQL 14
- **Preuzimanje:** Preuzmite kod s main brancha (instalirajte zip ili ga klonirajte)

Instalacija ovisnosti: Odite u frontend folder i pokrenite redom

```
npm install
```

```
npm run build
```

Prvom naredbom smo instalirali potrebne ovisnosti, a drugom smo stvorili statičke fileove koji se nalaze u “dist” folderu. Kopirajte sadržaj dist foldera i premjestite ga u backend folder u resources/static (napravite static folder) **2. Postavke** Prije pokretanja backenda potrebno je namjestiti varijable okruženja;

```
dbURL=jdbc:postgresql://localhost:5432/dbname
```

```
dbUser=username
```

```
dbPassword=password
```

```
GOOGLE_ID=oauthGoogleID
```

```
GOOGLE_SECRET=oauthGoogleSecret
```

```
VITE_BASE=https://hoodclassics.onrender.com
```

VITE_BASE može ostati i prazan ako se aplikacija pokreće lokalno

3. Pokretanje aplikacije pokrenite java kod “OppApplication” i upišite “localhost:8080” u Vaš preglednik.

5. Primjer za Render platformu (Cloud Deploy)

Render je popularna cloud platforma za jednostavno smještanje aplikacija.

- **Priprema repozitorija:**

- Osigurajte da vaš projekt sadrži Dockerfile za konfiguraciju deploya.
- Primjer Dockerfile-a za ovu aplikaciju:

```
FROM node:16 AS frontend-builder
```

```

WORKDIR /app/frontend

ARG VITE_BASE
ENV VITE_BASE=${VITE_BASE}

COPY frontend/ .

RUN echo "VITE_BASE=${VITE_BASE}" && npm install && npm run build

FROM openjdk:17-jdk-slim AS backend-builder

WORKDIR /app/backend

COPY backend/ .

COPY --from=frontend-builder /app/frontend/dist /app/backend/src/main/resources

RUN chmod +x mvnw

RUN ./mvnw clean package -DskipTests

FROM openjdk:17-jdk-slim

WORKDIR /app

COPY --from=backend-builder /app/backend/target/opp-0.0.1-SNAPSHOT.jar /app/

EXPOSE 8080

CMD ["java", "-jar", "opp-0.0.1-SNAPSHOT.jar"]

```

- **Postavljanje na [Render](#):**

- Prijavite se na [Render](#).
- Kreirajte novi **Web Service** i odaberite opciju za Docker deployment.
- Povežite se s vašim GitHub repozitorijem koji sadrži navedeni Dockerfile.
- Dodajte potrebne environment varijable:
 - `VITE_BASE=https://hoodclassics.onrender.com`
 - DATABASE_URL=jdbc:postgresql://db-host:5432/dbname
 - DB_USER=username
 - DB_PASSWORD=password
 - GOOGLE_ID=oauthGoogleID
 - GOOGLE_SECRET=oauthGoogleSecret

- **Pokretanje aplikacije:**

Render će automatski prepoznati Dockerfile, izgraditi sliku i pokrenuti aplikaciju prema zadanim postavkama.

Nakon deploja, aplikaciji možete pristupiti putem generiranog URL-a (npr. <https://hoodclassics.onrender.com>).

- **Održavanje:**

- Render podržava automatsko ponovno deployanje kad se naprave promjene u repozitoriju (CI/CD).
- Pogledajte dashboard za logove, greške i performanse aplikacije.

- **Napomena:**

- U besplatnom planu, aplikacija može imati ograničen broj aktivnih sati dnevno i može spavati nakon nekog vremena neaktivnosti.
- Pobrinite se da environment varijable budu pravilno postavljene jer će nedostatak bilo koje ključne varijable sprječiti ispravno pokretanje aplikacije.

Pristup aplikaciji

- Nakon deploja, korisnici mogu pristupiti aplikaciji putem URL-a generiranog od strane Render platforme.

- **Ograničenja:**

- Pristup aplikaciji može biti sporiji prilikom prvog otvaranja zbog "spavanja" aplikacije u besplatnom planu.
- Pobrinite se da aplikacija radi na HTTPS protokolu radi sigurnosti.
- Za produkcijsku upotrebu preporučuje se korištenje plaćenih planova zbog stabilnijeg rada i dodatnih resursa. # Opis prisutpa aplikaciji na javnom poslužitelju **Pristup aplikaciji** Aplikaciji HoodClassics trenutno se može pristupiti preko web preglednika na linku: <https://hoodclassics.onrender.com> Ako želite pristupiti svim mogućnostima aplikacije potrebno se je prijaviti i nakon prijave odabrati mjesto na kojem želite objavljivati priče. Naravno možete i bez prijave samo gledati priče tako da kliknete na gumb "Try as a tourist". Nakon prijave, kako biste objavili priču, kliknite na gumb "**add**" i odaberite mjesto na karti gdje biste htjeli objaviti priču. Klikom na bilo koji pin na karti imate mogućnost like i dislikea i pristup detalnjem sadržaju priče.

Ciljevi projekta

Cilj ovog projekta bio je razviti web aplikaciju HoodClassics koja omogućava dublje i autentičnije istraživanje turističkih destinacija. Aplikacija je zamišljena kao most između turista i lokalnih zajednica, pružajući platformu za dijeljenje priča, legendi i povijesnih zanimljivosti. Kroz te sadržaje korisnici mogu dobiti bogat kulturni i emocionalni kontekst mjesta koja posjećuju, što klasični turistički vodiči često ne nude. Osnovne funkcionalnosti aplikacije uključuju interaktivnu kartu za pregledavanje sadržaja, sustav tagova za tematsko filtriranje objava te opciju moderiranja koja osigurava da je sav sadržaj prikidan, točan i pouzdan. Lokalni korisnici imaju mogućnost dodavanja objava vezanih uz specifične lokacije, dok turisti te objave mogu pregledavati, označavati kao zanimljive ("lajkati") i pretraživati prema vlastitim interesima.

Izazovi tijekom razvoja

Razvoj aplikacije bio je izazovan proces tijekom kojeg smo se suočili s brojnim tehničkim i organizacijskim problemima. Jedan od značajnih izazova bio je proces implementacije i deploy-a aplikacije na produkcijsku platformu, no taj problem smo uspješno riješili. Također, sigurnosni aspekt aplikacije postavljen je u skladu s trenutnim znanjem i raspoloživim vremenom, dok je funkcionalnost geocodinga, koja omogućava pretvaranje koordinata u adrese, implementirana i potpuno funkcionalna. Ovi izazovi potaknuli su nas na dublje istraživanje i razvoj rješenja, čime smo značajno proširili svoje tehničke kompetencije.

Tijekom rada na projektu stekli smo brojna nova znanja, uključujući implementaciju prijave putem Google računa, integraciju interaktivne karte, deployment web aplikacije te korištenje alata za testiranje aplikacija. Osim tehničkih vještina, razvili smo i praktične sposobnosti poput rada u timu, praćenja rokova i osiguravanja dosljednog napretka projekta. Kroz ovaj proces jasno smo identificirali područja u kojima bismo mogli unaprijediti svoju učinkovitost, kao što su znanja o testiranju, deploymentu i sigurnosti. Sada, s ovim iskustvom, vjerujemo da bismo sličan projekt mogli realizirati brže i kvalitetnije.

Neki od predviđenih funkcionalnih zahtjeva nisu implementirani, funkcionalni zahtjevi F-03, koji uključuje oporavak lozinke putem e-maila, te funkcionalni zahtjevi F-11, koji omogućava rangiranje objava. Oni nisu realizirani zbog nedostatka vremena i tehničkog znanja. Ti nedostatci predstavljaju izazov, ali i priliku za buduće iteracije aplikacije.

Ograničenja i budući razvoj

Ograničenja s kojima smo se suočili tijekom projekta uključivala su vremenske restrikcije i manjak financijskih sredstava. Mnogi alati i servisi koji bi ubrzali razvoj zahtjevali su pretplate ili jednokratna plaćanja, što je nadilazilo naš budžet. Unatoč tome, uspjeli smo razviti funkcionalan proizvod koji može služiti kao temelj za daljnji razvoj.

Perspektive za budući rad na ovom projektu uključuju nadogradnju aplikacije s neispunjениm funkcionalnim zahtjevima (F-03 i F-11), razvoj mobilne verzije aplikacije za različite operativne sustave te dodavanje dodatnih značajki koje bi obogatile korisničko iskustvo. Također, skaliranje aplikacije za podršku većem broju korisnika te unaprjeđenje sigurnosnih značajki predstavljaju važan korak u njenom razvoju. Projekt HoodClassics ostvario je svoj primarni cilj i pružio platformu za unapređenje turističkog iskustva, istovremeno promovirajući lokalnu kulturu i baštinu. Proces rada bio je vrijedan izvor znanja i iskustva za sve članove tima, a rezultat projekta je temelj za daljnji razvoj i potencijalno širenje aplikacije.

tablica A.01: Dnevnik promjena dokumentacije Rev. Opis promjene/dodatka Autori Datum --- :----- :----- :----- 0.1 inicijalni home page M. Piskač 28.10.2024. 0.2 napravljen predložak D. Strmečki 4.11.2024 0.3 nova verzija dokumentacije za projekt D. Strmečki 11.11.2024. 2.1 dodatne revizije prve verzije dokumentacije za projekt D. Strmečki 11.11.2024. 0.4 preseljen wiki sa starog gita D. Strmečki 15.11.2024. 1.1 opis projektnog zadatka D. Strmečki 15.11.2024. 2.2 funkcionalni i nefunkcionalni zahtjevi D. Strmečki 15.11.2024. 4.1 dodane slike za opis projekta D. Strmečki 15.11.2024. 2.3 ažurirani prioriteti funkcionalnih zahtjeva D. Strmečki 10.1.2025.					
B.1 dodani održani sastanci D. Strmečki 10.1.2025. B.2 ažuriran plan rada D. Strmečki 10.1.2025. 0.5 dodani članovi tima na Home D. Strmečki 14.1.2025. 0.6 dodane poveznice za kod aplikacije na home D. Strmečki 14.1.2025. 1.2 dodan cilj projekta u opis D. Strmečki 14.1.2025. 4.2 dodavanje novih dijelova predloška D. Strmečki 14.1.2025. 2.4 specificiranje funkcionalnih zahtjeva D. Strmečki 15.1.2025. 3.1 Obrasci uporabe D. Strmečki 15.1.2025. 3.2 opis obrazaca uporabe D. Strmečki 15.1.2025. 7.1 nabranjanje i opis tehnologija D. Strmečki 15.1.2025. A.1 dodani izvori u popis literature D. Strmečki 15.1.2025. 0.7 popravljene poveznice za kod aplikacije na home H. Radoš 16.1.2025. 2.5 dodavanje funkcionalnih zahtjeva koji nedostaju D. Strmečki 16.1.2025. 2.6 popravljanje funkcionalnih zahtjeva H. Radoš 16.1.2025. 3.3 opis obrazaca uporabe H. Radoš 16.1.2025. B.3 dodani održani sastanci D. Strmečki 16.1.2025. 4.3 ažuriranje arhitekture i dizajna sustava D. Strmečki 16.1.2025. 4.4 ažuriranje arhitekture i dizajna sustava H. Radoš 16.1.2025. 2.7 dodavanje aktora koji nedostaju D. Strmečki 17.1.2025. 3.4 popravljanje opisa obrazaca uporabe D. Strmečki 17.1.2025. 0.8 popravljeni sekvencijski dijagrami D. Strmečki 18.1.2025. 4.5 dijagrami stanja i aktivnosti H. Radoš 18.1.2025. 5.1 dijagrami komponenti i razmještaja H. Radoš 18.1.2025. 3.5 popravljen opis sekvencijskih dijagrama D. Strmečki 19.1.2025. 4.6 popravak dijagrama stanja i aktivnosti H. Radoš 19.1.2025. 7.2 dodavanje korištenih tehnologija D. Strmečki 19.1.2025. 8.1 dodane upute za puštanje u pogon H. Radoš 19.1.2025. 1.3 popravci D. Strmečki 24.1.2025. 4.7 dodavanje opisa er dijagrama i relacijske sheme D. Strmečki 24.1.2025. 4.8 dodavanje dijagrama stanja D. Strmečki 24.1.2025. 6.1 dokumentiranje ispitivanja u Postman-u I. Bevanda 24.1.2025. 6.2 dokumentiranje ispitivanja u Selenium-u D. Strmečki 24.1.2025. 6.3 dodavanje alata za komunikaciju u dokumentaciju D. Strmečki 24.1.2025. 9.1 dodavnaje zaključka D. Strmečki 24.1.2025.					

Napomena: Naš je tim mijenjao git prilikom izrade projekta, te je Dnevnik promjena dokumentacije trenutno sastavljen na način da uključuje revizije sa oba. Također, promjene u Dnevniku promjena dokumentacije nisu reflektirane u tablici.

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>
7. Material Design components, <https://mui.com/material-ui/>
8. React documentation, <https://react.dev/>
9. Spring Security documentation, <https://docs.spring.io/spring-security/site/docs/current/api/index.html>,
10. Spring documentation, <https://docs.spring.io/spring-framework/reference/index.html>
11. Tripadvisor, <https://www.tripadvisor.com/>
12. Google maps, <https://www.google.com/maps>

Dnevnik sastajanja

1. sastanak * *Datum:* 14. listopada 2024. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, D. Strmečki * *Teme sastanka:* * uspostavljena Discord I Whatsapp grupa svih članova tima * inicijalne ideje za temu našeg projekta

2. sastanak * *Datum:* 15. listopada 2024. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki * *Teme sastanka:* * dogovaranje teme za projekt * sastavljanje kratkog plana projekta za predlaganje teme profesoru * inicijalna raspodjela uloga unutar tima

3. sastanak * *Datum:* 21. listopada 2024. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki * *Teme sastanka:* * raspravljanje o detaljima projekta * predstavljanje dijagrama profesoru i demonstratoru * konzultacije sa profesorom i demonstratorom

4. sastanak * *Datum:* 25. listopada 2024. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki * *Teme sastanka:* * osnovni funkcionalni zahtjevi * ideje za nefunkcionalne zahtjeve

5. sastanak * *Datum:* 28. listopada 2024. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki * *Teme sastanka:* * prezentiranje funkcionalnih zahtjeva * konzultacije sa profesorom i demonstratorom

6. sastanak * *Datum:* 4. studenog 2024. * *Prisustvovali:* H. Radoš, M. Piskač, D. Strmečki * *Teme sastanka:* * prezentiranje napretka sa projektom * konzultacije sa profesorom i demonstratorom

7. sastanak * *Datum:* 8. studenog 2024. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki * *Teme sastanka:* * daljnja raspodjela poslova koji su ostali * dogovor za dijelove projekta koje planiramo završiti u prvom ciklusu

8. sastanak * *Datum:* 11. studenog 2024. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki * *Teme sastanka:* * prezentiranje dosadašnjeg napretka projekta

9. sastanak

* *Datum:* 15. studenog 2024.
* *Prisustvovali:* H. Radoš, I. Bevanda
* *Teme sastanka:*
* Deploy aplikacije
* dogovor vezan uz Google Secrets
* Spajanje backend i frontend dijelova aplikacije

10. sastanak * *Datum:* 5. prosinca 2024.

* *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
* *Teme sastanka:*
* Dovršavanje radova za prvi ciklus nastave
* Provjera deploymenta aplikacije

11. sastanak

* *Datum:* 9. prosinca 2024.
* *Prisustvovali:* H. Radoš
* *Teme sastanka:*
* konzultacije s profesorom vezane uz bodove za tim

12. sastanak

* *Datum:* 9. prosinca 2024.
* *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
* *Teme sastanka:*
* Raspodjela bodova po članovima tima

13. sastanak

* *Datum:* 16. prosinca 2024.
* *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
* *Teme sastanka:*
* Online konzultacije s profesorom

14. sastanak

* *Datum:* 21. prosinca 2024.
* *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
* *Teme sastanka:*

- * Revizija podjele poslova projekta
- * Priprema za demonstraciju alfa verzije
- * Analiza napretka u prvom ciklusu nastave
- * Dinamičko ponašanje aplikacije, razmještaj komponenti, usklađivanje dokumentacije i ispitivanje aplikacije

15. sastanak

- * *Datum:* 6. siječnja 2025.
- * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
- * *Teme sastanka:*
 - * Implementacija praćenja lokacije korisnika
 - * Poboljšanje autorizacije
 - * Dogovor o funkcionalnostima alfa verzije aplikacije:
 - Registracija, prikaz karte, objave na karti
 - Određivanje statusa korisnika (turist/lokalac)
 - * Planiranje za preostale funkcionalnosti (sustav tagova, prijave korisnika, potencijalna nova funkcionalnost „upit“)

16. sastanak

- * *Datum:* 7. siječnja 2025.
- * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
- * *Teme sastanka:*
 - * Konzultacije s profesorom o demonstraciji alfa verzije aplikacije

17. sastanak

- * *Datum:* 7. siječnja 2025. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
- * *Teme sastanka:*
 - * Komentari na sigurnosne aspekte:
 - Uvođenje dodatnih sigurnosnih mjera u autentifikaciju korisnika
 - * Ubrzavanje razvoja backend dijela
 - * Razgovor o dodatnim funkcionalnostima aplikacije:
 - Zoom mape
 - Dodavanje lajkova i potencijalno rangiranje objava

18. sastanak

- * *Datum:* 12. siječnja 2025. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
- * *Teme sastanka:*
 - * popravljanje funkcionalnosti aplikacije (rađenje objava od strane korisnika)
 - * frontendov dio aplikacije vezan uz izradu objava

19. sastanak

- * *Datum:* 13. siječnja 2025. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
- * *Teme sastanka:*
 - * Konzultacije s profesorom o demonstraciji alfa verzije aplikacije
 - * Komentari na: - limit za tekst u objavi - log out - pin kojim se korisnik kreće po mapi potencijalno staviti da je prikayan druge boje - korištenje Seleniuma za

testiranje aplikacije

20. sastanak

* *Datum:* 15. siječnja 2025. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki

* *Teme sastanka:*

* podjela zadataka za ostatak obaveza koje se trebaju dovršiti do kraja tjedna

* komentiranje korištenih tehnologija za dokumentiranje na wikiju

21. sastanak

* *Datum:* 19. siječnja 2025. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki

* *Teme sastanka:*

* uljepšavanje koda

* točnost novih dijagrama

22. sastanak

* *Datum:* 24. siječnja 2025. * *Prisustvovali:* H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki

* *Teme sastanka:*

* popravak deploymenta * dogovori oko testiranja * dogovori oko dokumentacije

Plan rada

Zadatak	Rok	Zaduženi	Status
Tema projekta	15.10.2024.	Cijeli tim	Odrađeno
Dijagrami obrazaca uporabe	21.10.2024.	H. Radoš	Odrađeno
Nabranje funkcijskih zahtjeva	28.10.2024.	Cijeli tim	Odrađeno
Planiranje dizajna Sekvencijski dijagrami	27.10.2024.	Frontend podtim	Odrađeno
Priprema za prezentiranje napretka	4.11.2024.	D. Strmečki	Odrađeno
Izrada početne stranice	11.11.2024.	Cijeli tim	Odrađeno
Spajanje s bazom podataka	1.11.2024.	Frontend podtim	Odrađeno
Implementacija interaktivne karte s pinovima	20.11.2024.	H. Radoš	Odrađeno
Dizajn i razvoj registracije korisnika	12.1.2025.	Backend podtim	Odrađeno
Kreiranje sustava za	12.1.2025.	Backend podtim	Odrađeno

Zadatak	Rok	Zaduženi	Status
objave i tagove			
Moderatorski sustav	13.1.2025.	Backend podtim	Odrađeno
Testiranje alfa verzije aplikacije	12.1.2025.	Cijeli tim	Odrađeno
Implementacija osnovnih funkcionalnosti alfa verzije (registracija, prikaz karte, objave)	13.1.2025.	Cijeli tim	Odrađeno
Implementacija sustava za pretragu objava po tagovima	16.1.2025.	Backend podtim	Odrađeno
Implementacija dodatnih funkcionalnosti (zoom, lajkanje, prijave)	18.1.2025.	Backend i frontend podtimovi	Odrađeno
Testiranje finalne verzije aplikacije	13.1.2025.	Cijeli tim	Odrađeno
Provjera finalne verzije projekta	18.1.2025.	Cijeli tim	Odrađeno

Napomena:

- Cijeli tim:** H. Radoš, I. Bevanda, D. Ćubelić, J. Borić, M. Piskač, D. Strmečki
- Frontend podtim:** J. Borić, M. Piskač
- Backend podtim:** I. Bevanda, D. Ćubelić

Tablica aktivnosti

Tablica aktivnosti prikazuje koliko je svaki član tima pridonio kojoj aktivnosti. Vrijednosti su izražene u u satima i svatko ih zapisuje za sebe.

aktivnosti	I. Bevanda	J. Borić	D. Ćubelić	M. Piskač	H. Radoš	D. Strmečki
Upravljanje projektom	0	0	0	0	20	0
Opis projektnog zadatka	2	2	2	2	2	2
Funkcionalni zahtjevi	1	1	0	1	1	3
Dijagram	0	0	0	0	4	4

aktivnosti	I. Bevanda	J. Borić	D. Ćubelić	M. Piskač	H. Radoš	D. Strmečki
obrazaca						
Sekvencijski dijagrami	0	0	0	0	0	4
Opis ostalih zahtjeva	0	0	0	0	0	3
Opis projekta	0	0	0	0	0	4
Arhitektura i dizajn sustava	11	0	0	0	0	0
Baza podataka	4	0	0	0	3	0
opis baze podataka	0	0	0	0	0	2
Dijagram razreda	0	0	1	0	0	0
Dijagram stanja	0	0	0	0	1	1
Dijagram aktivnosti	0	0	0	0	1	1
Dijagram komponenti	0	0	0	0	1	1
Korištene tehnologije i alati	10	0	7	0	0	2
Ispitivanje programskog rješenja	8	2	0	2	15	0
Dijagram razmještaja	0	0	0	0	1	1
Upute za puštanje u pogon	1	0	0	0	2	0
Dnevnik sastajanja	0	0	0	0	0	4
Zaključak i budući rad	0	0	0	0	0	2
Popis literature	1	0	0	0	0	1
izrada aplikacije	10	60	0	25	10	0
izrada baze podataka	8	0	0	0	0	0
spajanje s bazom podataka	2	0	2	0	3	0
dokumentiranje plana projekta i	0	0	0	0	0	3

aktivnosti	I. Bevanda	J. Borić	D. Ćubelić	M. Piskač	H. Radoš	D. Strmečki
uključenosti						
dokumentacija						
specifikacije						
obrazaca	0	0	0	0	0	4
uporabe						

HoodClassics

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Projekt - HoodClassics

Tim: <TG02.4>

HoodClassics

Nastavnik: Vlado Sruk

Članovi Tima:

- Hrvoje Radoš - *voditelj projekta*
- Ivan Bevanda - *razvoj backenda aplikacije*
- Duje Ćubelić - *razvoj backenda aplikacije*
- Jakov Borić - *razvoj frontenda aplikacije*
- Mateo Piskač - *razvoj frontenda aplikacije*
- Dora Strmečki - *dokumentacija*

Poveznica za GitHub projekta:

<https://github.com/PROGI-HoodClassics/HoodClassics>

Upute za korištenje

Aplikaciji HoodClassics trenutno se može pristupiti preko web preglednika na linku: <https://hoodclassics.onrender.com> Ako želite pristupiti svim mogućnostima aplikacije potrebno se je prijaviti i nakon prijave odabrati mjesto na kojem želite

objavljivati priče. Naravno možete i bez prijave samo gledati priče tako da kliknete na gumb "Try as a tourist". Nakon prijave, kako biste objavili priču, kliknite na gumb "+" i odaberite mjesto na karti gdje biste htjeli objaviti priču. Klikom na bilo koji pin na karti imate mogućnost like i dislikea i pristup detaljnog sadržaju priče.

Izvorni kod aplikacije

Kod aplikacije nalazi se na GitHubu projekta i podijeljen je na backend i frontend dijelove aplikacije. Kod backend razvoja nalazi se u folderu "backend" na poveznici <https://github.com/PROGI-HoodClassics/HoodClassics/tree/dev/backend>, a kod za frontend dio aplikacije u folderu "frontend" na poveznici <https://github.com/PROGI-HoodClassics/HoodClassics/tree/dev/frontend>.