

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

SafeBear

Tim: <TG02.3>

Leteći Medvjedić

Članovi i pozicije:

Elma Vuran - voditelj, full-stack Lana Zekušić - frontend, dizajn Ivan Pastorčić - backend, baza, dokumentacija, DevOps Natali Žegarac - backend, baza Silvija Rade - frontend Krešimir Horvat - frontend Iva Sulić - frontend

Nastavnik: Vlado Sruk

Pristup aplikaciji

Aplikaciju SafeBear se može koristiti putem web preglednika.

Izvorni kod se nalazi na [GitHub-u projekta](#)

Više informacija o projektu pogledajte na našem [Wiki-ju](#)

Trenutna verzija aplikacije: **V2.0.0**

Cilj projekta

Ovaj projekt je rezultat timskog rada u sklopu projektnog zadatka kolegija [Programsko inženjerstvo](#) na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu.

Cilj ovog studentskog timskog projekta je osmisлити и razviti aplikaciju (SafeBear) koja olakšava prijavljivanje i upravljanje prirodnim nepogoda. Aplikacija bi omogućila bržu reakciju i učinkovitiju koordinaciju i komunikaciju između građana, vlasti i humanitarnih organizacija. Također će pružiti odličan uvid u potrebe pogodjenih područja i stanovništva koje su se našli u katastrofi.

Analiza problema

U vrijeme prirodnih nepogoda kao što su požari, potresi, poplave, jakе oluje te odroni ključno je djelovati brzo i efikasno kako bi posljedice katastrofa bile što manje i kako bi

nepogoda utjecala na što manji broj ljudi. Razdoblje tijekom i nakon ovakvih nepogoda izrazito je izazovno i stresno za građane te im je potrebno osigurati svu pomoć koju mogu dobiti.

Trenutni je problem tijekom kriznih situacija loša informiranost što vodi raznim problemima. Korisnici informacije dobijaju putem raznih portala, informacije su raspršene te ponekad neusklađene, s naglaskom na veća središta. Komunikacija između građana je skoro nepostojeća te zapravo u kriznoj situaciji znamo samo što se događa s najbližima. Često ljudi nisu sigurni kome je najbolje obratiti se za pomoć te kako to najjednostavnije napraviti. Sporo prikupljanje informacije predstavlja veliki problem i može znatno utjecati na smanjen postotak spašenih ljudi.

Humanitarne organizacije i spasilačke službe ovise o individualnim prijavama, a vlasti nemaju dobar pregled podataka za krizne situacije što vodi sporoj reakciji što može biti pogubno za život. Humanitarne organizacije i vlast također nemaju dobru međusobnu komunikaciju, što vodi tome da nije ni poznato koliko smo zapravo spremni za krizne situacije, koje količine resursa imamo i slično. Hitne službe imaju ograničene resurse koji se često ne koriste optimalno, upravo zbog nedostatka koordinacije između različitih sudionika.

Građanima je potreban jednostavan način za informiranje vlasti o nepogodi koja ih je zadesila. Prijava specifičnosti situacije olakšava vlastima i humanitarnim organizacijama uvid u trenutni razvoj događaja na terenu.

Cilj naše aplikacije je kvalitetno rješiti navedene probleme integracijom prijava u stvarnom vremenu, praćenjem i distribucijom informacija među građanima, vlastima i organizacijama. Također želimo olakšati reakciju na prirodne nepogode administrativnim tijelima, stvoriti potpunu bazu podataka za postojeće skladištene resurse te omogućiti bolju podršku stradalima tijekom prirodnih nepogoda.

Zahtjevi korisnika

Aplikacija će biti usmjerena na više grupa korisnika, s prikladnim funkcionalnostima.

Za potrebe brze prijave nepogode ili samo olakšanje onima koji ne žele napraviti korisnički račun, prijave će biti moguće napraviti anonimno. U tom slučaju, anonimni korisnik biti će u mogućnosti prijavu pratiti preko jedinstvenog identifikacijskog broja prijave.

Registrirani će korisnici moći pratiti svoje prijašnje prijave, prijavljivati svoje potrebe u kriznim situacijama, kao npr. potrebe za hranom, skloništem i medicinskom intervencijom. Također će imati pristup interaktivnoj mapi i news feedu za pregled aktualnih prijavi. Imati će opciju subskripcije na obavijesti putem push notifikacija ili e-maila za nove prijave u željenim regijama Republike Hrvatske. Aplikacija će im također omogućiti pregled informacija za reagiranje na krizne situacije te pregled akcija pomoći koje organiziraju humanitarne organizacije.

Vlasti, specifično Ministarstvo obrane Republike Hrvatske (MORH) i Ministarstvo Unutarnjih Poslova (MUP), će imati pregled svih prijava nepogoda te aktualnih skladištenih resursa u vlasti humanitarnih organizacija. Podaci će im biti dostupni za stvaranje statističkih izvještaja za prirodne nepogode.

Humanitarne organizacije će moći putem aplikacije obavještavati o radnim akcijama te pregled aktualnih prijavljenih potreba građana. Moći će dodavati informacije o trenutnim resursima koje posjeduju koji su važni u kriznim situacijama, poput zaliha vode, hrane, kapacitetima skloništa, količina kutija za prvu pomoć, vreći pjeska i slično.

Administratori će upravljati prijavama, nadgledati sigurnosti podataka te osigurati nesmetan rad aplikacije.

Potencijalne koristi

Aplikacija će poboljšati komunikaciju između građana, humanitarnih organizacija i vlasti u stvarnom vremenu.

Kroz centraliziranu platformu, alociranje resursa za krize će biti olakšano vlastima i humanitarnim organizacijama.

Građani će biti kontinuirano informirani o prirodnim nepogodama u blizini i moći će doprinositi kroz prijavljivanje nepogoda.

Poboljšat će se sigurnost građana jer će brže dobivati informacije o prirodnih nepogodama u njihovoј blizini. Također će moći prijavljivati opasnosti koje opažaju čime mogu spasiti živote.

Kroz aplikaciju će se skupljati korisni podaci o prirodnim nepogodama i potrebama građana, što će vlastima omogućiti analizu u svrhu poboljšanja pripremljenosti na prirodne nepogode.

Ograničeni resursi države mogu se optimalno rasporediti. Kroz bolju koordinaciju i uvid u potrebe, humanitarne organizacije mogu pomoći onima kojima je najpotrebnije.

Vlasti i organizacije dobivaju relevantne podatke za istraživanje učinaka prirodnih nepogoda i planiranje prevencije.

Postojeća slična rješenja

FEMA App (SAD)

FEMA aplikacija primarno je usmjerena na pripremu te oporavak od prirodnih nepogoda, dok je SafeBear usmjerena na izvještavanje o prirodnim nepogodama u stvarnom vremenu. Izvještavanje o prirodnim nepogodama je obveza Nacionalne meteorološke službe, dok SafeBear ovisi o korisničkim prijavama. SafeBear također omogućava korištenje

interaktivne mape za pregled prijava koja je ažurirana u stvarnom vremenu. FEMA je zbog ovisnosti o Nacionalnoj meteorološkoj službi ograničena na Sjedinjene Američke Države, dok bi se SafeBear mogao implementirati na globalnoj razini. (Prvobitno će biti ograničen na Republiku Hrvatsku). Integracija autoriteta u FEMA aplikaciji uključuje samo državne agencije, dok SafeBear uključuje i humanitarne organizacije kao ključna tijela.[\[1\]](#)

MyShake App (Kalifornija)

MyShakeApp je aplikacija koju je razvilo Sveučilište u Kaliforniji, Berkeley. Osigurava korisnicima upozorenja nekoliko sekundi prije nego što se osjeti podrhtavanje tla. Aplikacija se temelji na ShakeAlert sustavu Američke geološke službe (USGS) za područja s visokom seizmičkom aktivnosti, poput Kalifornije, Oregonia i Washingtona. Nadalje, prikuplja i analizira podatke iz mreže seizmičkih senzora i obrađuje ih koristeći algoritme koji brzo prepoznaju značajne pokrete tla. Ako se otkrije potres, MyShake aplikacija odmah šalje obavijesti korisnicima u blizini pogodjenih područja, što im omogućuje da poduzmu mjere zaštite prije nego što podrhtavanje postane snažnije. Aplikacija je ograničena samo na potrese za razliku od SafeBear aplikacije.[\[2\]](#)

Google Crisis Map

Google Crisis Map alat je koji je prikazivao podatke za prirodne katastrofe u stvarnom vremenu, omogućujući korisnicima brz pristup ključnim sigurnosnim informacijama. Integriranjem podataka iz pouzdanih izvora, mapa nudi detalje o događajima poput požara, uragana, poplava i drugih. Prikazuje zone evakuacije, lokacije skloništa i uvjete u prometu uživo, što pomaže ljudima da donesu informirane odluke u kriznim trenucima. Google Crisis Map primarno je služio za vizualizaciju, dok je SafeBear platforma s fokusom na korisnike. Većina funkcionalnosti je djelomično integrirana u Google Maps te Google Crisis Map nije u funkciji od 31.3.2021.[\[3\]](#)

Skup korisnika koji bi mogao biti zainteresiran za ostvareno rješenje

Gradani, posebno osobe u područjima sklonima prirodnim nepogodama, koji žele što brže prijaviti opasnost i/ili dobiti pouzdane informacije o trenutnim prirodnim nepogodama te sigurnosnim mjerama za iste. Kroz interaktivnu mapu, obavijesti o nepogodama te akcijama pomoći, uvijek će moći biti informirani.

Administrativna tijela, tj. osobe na čelu istih, kojima je neophodno dobivanje pravovremenih informacija kako bi mogli pokrenuti odgovarajuće reakcije u skladu s problemom. Kroz pregled prijava nepogoda i resursa, dobili bi pregled stvarne situacije koji nije omogućen trenutnim alatima u Republici Hrvatskoj.

Humanitarne organizacije koje nude pomoć tijekom kriznih situacija bi mogle koristiti informacije iz aplikacije kako bi mogli učinkovitije planirati akcije pomoći te kvalitetnije pružati i organizirati resurse.

Osim grupa za koje su planirane funkcionalnosti unutar aplikacije, interes za aplikaciju bi moglo imati volonterske skupine koje su namijenjene pomoći stradalima u kriznim situacijama i organizacije javne službe, poput vatrogasaca, policije, bolnica i sličnih koje igraju ključnu ulogu u prvobitnoj pomoći građanima tijekom prirodnih nepogoda.

Mogućnost prilagodbe rješenja

Trenutno aplikacija sve prirodne nepogode tretira jednako, tj. ne bilježi dodatne podatke o specifičnim prirodnim nepogodama. Moguća prilagodba bi uključivala različito tretiranje različitih nepogoda i praćenje različitih podataka, npr. percipiranu jačinu za potrese, prijavljivanje oštećenih nekretnina te razinu štete.

Aplikacija bi mogla biti prilagođena specifičnim regionalnim potrebama. Na primjer, za područja sklona potresima, mogu biti omogućene dodatne funkcionalnosti za upozorenja u stvarnom vremenu, dok će u regijama gdje su češće poplave aplikacija naglasiti značajke kao što su popis skloništa i informacije o poplavnim valovima.

Aplikacija može biti opremljena opcijama za više jezika, što bi omogućilo korištenje u različitim dijelovima svijeta. Uz to, bila bi omogućena slanja regionalno specifičnih upozorenja, kao i personalizacija korisničkog sučelja ovisno o jeziku i lokaciji korisnika, čime bi se poboljšalo korisničko iskustvo.

Opseg projektnog zadatka

Osnovne funkcionalnosti uključuju prijavljivanje prirodnih nepogoda i potreba te praćenje istih, obavještavanje korisnika o novim prijavama, informiranje o akcijama pomoći, sigurnosnim mjerama, skloništima u blizini te pregled prijašnjih prijava.

U skladu s dijagramom obrazaca uporabe, definirane su različite uloge korisnika, uključujući anonimne i registrirane korisnike, administratore, vlasti te humanitarne organizacije. Svaka od ovih uloga ima različite ovlasti i pristup određenim funkcionalnostima u aplikaciji, čime se omogućava specifičan pristup podacima i mogućnostima aplikacije za svaki tip korisnika.

Aplikacija zahtijeva pouzdanu bazu podataka koja će omogućiti sigurno pohranjivanje informacija o prijavljenim nepogodama, korisničkim profilima, resursima i povijesti događaja. Baza podataka će omogućiti brzi dohvati informacija za korisnike i službe koje trebaju ažurirane podatke u stvarnom vremenu.

Kako bi se osigurala sigurnost korisničkih podataka, implementirat će se osnovne sigurnosne mjere, kao što je autentifikacija korisnika i zaštita od neovlaštenog pristupa.

Autentifikacija korisnika će koristiti OAuth 2.0. Ove mjere su ključne kako bi se očuvala privatnost korisnika i spriječile potencijalne zloupotrebe podataka.

Moguće nadogradnje projektnog zadatka

Dodavanje dodatnih tipova korisnika, odnosno prilagodba aplikacije za dodatne uloge, poput volonterskih ili donatorskih organizacija, javnih službi i sličnih.

Povezivanje sa senzorima za monitoring poplava, seizmičkih aktivnosti i kvalitete zraka koji automatski šalju upozorenja.

Dodavanje dodatnih društvenih funkcionalnosti, poput međusobne komunikacije unutar aplikacije, dijeljenje resursa i slično.

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-01	Aplikacija mora omogućiti lociranje prijava na interaktivnoj mapi.	Srednji	Zahtjev dionika	Korisnik može vidjeti lokacije prijašnjih prijava na interaktivnoj mapi
F-02	Sustav omogućuje korisnicima kreiranje računa pomoću e-mail adrese.	Visok	Zahtjev dionika	Korisnik se može registrirati e-mailom, primiti e-mail za potvrdu i uspješno se prijaviti.
F-02.1	Sustav obavještava korisnika o prikupljanju i načinu korištenja osobnih podataka pri registraciji.	Nizak	Postojeći sustav	Korisnik prije stvaranja računa dobiva push obavijest s privolom za prikupljanje podataka.
F-02.1.1	Korisnik može odbiti prikupljanje podataka.	Nizak	Postojeći sustav	Korisnik ima opciju odbiti prikupljanje osobnih podataka na push obavijesti.
F-03	Sustav omogućuje korisnicima prijavu u sustav s pomoću postojećeg korisničkog računa.	Visok	Zahtjev dionika	Korisnik se može prijaviti u sustav e-mailom i lozinkom.
F-04	Aplikacija građanima omogućava prijavu informacija o prirodnim nepogodama.	Visok	Zahtjev dionika	Korisnik može prijaviti vrstu, lokaciju, sliku i koordinate nepogode te dodati kratki opis.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-05	Aplikacija građanima omogućuje praćenje statusa njihovih prijava.	Srednji	Zahtjev dionika	Korisnik ima uvid u stanje svojih prijava.
F-05.1	Aplikacija obavještava korisnika o promjenama statusa njegovih prijava putem e-maila.	Srednji	Zahtjev dionika	Korisnik prima obavijesti o promjenama statusa prijava putem e-maila.
F-05.2	Aplikacija obavještava korisnika o promjenama statusa njegovih prijava putem push notifikacija.	Srednji	Zahtjev dionika	Korisnik prima obavijesti o promjenama statusa prijava putem push notifikacija.
F-05.3	Registrirani korisnik može pratiti status svojih prijava.	Visok	Zahtjev dionika	Korisnik ima pregled prijašnjih prijava nepogoda i njihovog statusa u sklopu profila unutar aplikacije.
F-05.4	Anonimni korisnik ima mogućnost praćenja svojih prijava nepogoda.	Visok	Zahtjev dionika	Anonimni korisnik može putem jedinstvenog identifikacijskog broja prijave u aplikaciji tražiti uvid u status prijave.
F-06	Aplikacija obavještava korisnika o novim prijavljenim nepogodama.	Visok	Zahtjev dionika	Korisnik prima obavijesti o novim prijavljenim nepogodama.
F-06.1	Aplikacija obavještava korisnika o novim prijavljenim nepogodama putem push notifikacija.	Visok	Zahtjev dionika	Korisnik prima obavijesti o novim prijavljenim nepogodama putem push notifikacija.
F-06.2	Aplikacija obavještava korisnika o novim prijavljenim nepogodama putem e-maila.	Visok	Zahtjev dionika	Korisnik prima obavijesti o novim prijavljenim nepogodama putem e-maila.
F-06.3	Aplikacija obavještava korisnika o novim prijavljenim nepogodama ovisno o korisnikovom izboru.	Visok	Zahtjev dionika	Korisnik prima obavijesti o novim prijavljenim nepogodama ovisno o izboru regija.
F-06.3.1	Korisnik može izabrati regije za koje želi primati	Srednji	Zahtjev dionika	Korisnik ima opciju izbora regija za primanje obavijesti

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
	obavijesti.			o novim prirodnim nepogodama.
F-06.4	Aplikacija obavještava korisnika o novim prijavljenim nepogodama ovisno o korisnikovoj lokaciji.	Visok	Zahtjev dionika	Korisnik prima obavijesti o novim prijavljenim nepogodama ovisno o svojoj lokaciji.
F-07	Aplikacija omogućuje korisnicima pristup važnim informacijama za krizne situacije.	Srednji	Zahtjev dionika	Korisnik unutar aplikacije može pristupiti važnim informacijama u slučaju prirodnih nepogoda
F-07.1	Aplikacija omogućuje korisnicima pristup važnim informacijama o sigurnosnim mjerama u kriznim situacijama.	Srednji	Zahtjev dionika	Korisnik unutar aplikacije može pristupiti važnim informacijama o sigurnosnim mjerama u slučaju prirodnih nepogoda.
F-07.2	Aplikacija prikazuje korisnicima informacije o najbližim skloništima u kriznim situacijama.	Srednji	Zahtjev dionika	Korisnik unutar aplikacije može pristupiti informacijama o najbližim skloništima u slučaju prirodnih nepogoda.
F-08	Aplikacija omogućava slanje anonimne prijave.	Visok	Zahtjev dionika	Korisnik može prijaviti nepogodu bez prijave u sustav.
F-08.1	Status anonimne prijave se može pratiti putem jedinstvenog identifikacijskog broja.	Srednji	Zahtjev dionika	Korisnik može pratiti status anonimne prijave putem jedinstvenog identifikacijskog broja.
F-09	Aplikacija građaninu omogućuje prijavu potreba uz prijavu nepogode.	Visok	Zahtjev dionika	Korisnik može prijaviti svoje potrebe neposredno nakon prijave prirodne nepogode.
F-09.1	Aplikacija građaninu omogućuje prijavu potreba za hranom uz prijavu nepogode.	Visok	Zahtjev dionika	Korisnik može prijaviti svoje potrebe za hranom neposredno nakon prijave prirodne nepogode.
F-09.2	Aplikacija građaninu omogućuje prijavu potreba za skloništem uz	Visok	Zahtjev dionika	Korisnik može prijaviti svoje potrebe za skloništem neposredno nakon prijave

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-09.3	prijavu nepogode. Aplikacija građaninu omogućuje prijavu potreba za medicinskom intervencijom uz prijavu nepogode.	Visok	Zahtjev dionika	Korisnik može prijaviti svoje potrebe za medicinskom intervencijom neposredno nakon prijave prirodne nepogode.
F-10	Aplikacija vlastima omogućava pristup informacijama o svim prijavama u sustav.	Nizak	Zahtjev dionika	Vlasti imaju pristup podacima o prijavama u sustav.
F-11	Aplikacija omogućava vlastima pristup svim podacima vezanimi uz prirodne nepogode.	Nizak	Zahtjev dionika	Vlasti imaju pristup podacima o prijavama prirodnih nepogoda.
F-12	Aplikacija humanitarnim organizacijama omogućuje pregled potreba građana.	Visok	Zahtjev dionika	Vlasti imaju pristup podacima o potrebama građana.
F-12.1	Aplikacija humanitarnim organizacijama omogućuje filtriranje potreba građana.	Nizak	Kreativnost	Humanitarne organizacije imaju opciju filtriranja ovisno o vrsti potrebe građana.
F-13	Aplikacija humanitarnim organizacijama omogućuje dodavanje informacija o dostupnim resursima.	Srednji	Zahtjev dionika	Humanitarne organizacije imaju opciju dodati informacije o dostupnim resursima za krizne situacije unutar aplikacije.
F-13.1	Dostupni resursi uključuju: voda, hrana, kapacitet skloništa, kutija za prvu pomoć, vreća pijeska, grijalica.	Srednji	Zahtjev dionika	Humanitarne organizacije imaju unutar aplikacije opciju dodati informacije o dostupnim količinama vode, hrane, kutija za prvu pomoć, vreći pijeska i grijalica te trenutnom kapacitetu skloništa za krizne situacije.
F-13.1.1	Humanitarne organizacije imaju pregled količine resursa.	Nizak	Kreativnost	Humanitarne organizacije mogu vidjeti količinu dostupnim resursa unutar

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-14	Aplikacija humanitarnim organizacijama omogućuje dodavanje informacija o akcijama pomoći.	Srednji	Zahtjev dionika	Humanitarne organizacije imaju opciju dodati informacije o budućim akcijama pomoći unesrećenima.
F-15	Aplikacija administratorima omogućava pregled svih prijava.	Visok	Zahtjev dionika	Administratori mogu u aplikaciji vidjeti sve prijave nepogoda.
F-16	Aplikacija administratorima omogućava promjenu statusa prijave.	Visok	Zahtjev dionika	Administratori mogu promijeniti status prijave u aplikaciji.
F-16.1	Aplikacija administratorima omogućava odobravanje prijave.	Visok	Zahtjev dionika	Administratori mogu odobriti prijave u aplikaciji.
F-16.2	Aplikacija administratorima omogućava odbijanje prijave.	Visok	Zahtjev dionika	Administratori mogu odbiti prijave u aplikaciji.
F-17	Aplikacija mora omogućiti korisniku brisanje spremlijenih osobnih podataka.	Srednji	Postojeći sustav	Korisnik može unutar postavki za svoj profil tražiti brisanje svojih osobnih podataka.
F-17.1	Aplikacija mora korisniku dati obavijest o pokušaju brisanja podataka putem e-maila.	Nizak	Postojeći sustav	Korisnik dobiva e-mail potvrdu za pokušaj brisanja osobnih podataka.
F-18	Aplikacija mora korisnika obavijestiti o načini prikupljanja i korištenja osobnih podataka.	Nizak	Postojeći sustav	Korisnik pri prvom korištenju aplikacije dobiva push obavijest s privolom za prikupljanje i korištenje osobnih podataka.
F-18.1	Korisnik mora moći odbiti prikupljanje i korištenje osobnih podataka.	Nizak	Postojeći sustav	Korisnik može odbiti prikupljanje i korištenje osobnih podataka.
F-19	Aplikacija mora	Visok	Postojeći	Administrator sustava mora

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
	administratoru omogućiti dodavanje novog korisnika		sustav	imati mogućnost dodavanja novog korisnika specifikacijom korisničkog imena, e-mail adrese i odabira ovlasti
F-19.1	Ovlasti koje pojedini korisnik može imati jesu: ROLE_USER, ROLE_AUTHORITY, ROLE_HUMANITARIAN, ROLE_ADMIN	Visok	Postojeći sustav	Administrator prilikom dodavanja novog korisnika u sustav mora moći podesiti ovlast korisnika na jednu od navedenih: ROLE_USER, ROLE_AUTHORITY, ROLE_HUMANITARIAN, ROLE_ADMIN
F-20	Aplikacija mora administratoru omogućiti pretraživanje trenutnih podataka u bazi podataka	Srednji	Postojeći sustav	Administrator mora moći pretraživati podatke iz baze podataka kao što su korisnici i prijave nepogoda
F-20.1	Administrator mora moći pretraživati prijave nepogoda	Srednji	Postojeći sustav	Administrator mora moći pretražiti specifičnu prijavu nepogode prema njenom identifikacijskom broju
F-20.2	Administrator mora moći pretraživati korisnike aplikacije	Srednji	Postojeći sustav	Administrator mora moći pretražiti specifičnog korisnika aplikacije koristeći njegovu e-mail adresu
F - 21	Administrator mora moći obrisati korisnika iz baze podataka	Srednji	Postojeći sustav	Administrator mora moći obrisati korisnika iz baze podataka odabirom njegovog profila i pritiskom na gumb 'Delete user'
F - 22	Korisnik mora moći promijeniti svoje korisničko ime u sustavu	Nizak	Postojeći sustav	Korisnik mora moći promijeniti svoje korisničko ime u sustavu upisom novog imena i odabirom 'Change username'
F - 23	Korisnik se mora moći odjaviti iz sustava	Visok	Postojeći sustav	Korisnik se mora moći odjaviti iz sustava pritiskom na 'Log out' gumb

Ostali zahtjevi

ID zahtjeva	Opis	Prioritet
NF-01	Aplikacija slijedi Opću uredbu o zaštiti osobnih podataka	Visok
NF-02	Aplikacija mora biti responzivna na različitim uređajima	Nizak
NF-03	Aplikacija mora biti usklađena sa standardima pristupačnosti	Nizak
NF-04.1	Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje.	Visok
NF-04.1.1	Sustav treba imati dovoljnu dokumentaciju.	Visok
NF-04.1.1.1	Kôd sustava treba biti dokumentiran prema "Code Conventions for the Java Programming Language" dostupnim na Oracle .	Visok
NF-04.1.2	Sustav treba biti opisan putem dokumenta oblikovanja /SRS/.	Visok
NF-04.1.3	Sustav treba biti popraćen "Priručnikom za rad" koji opisuje pravilnu upotrebu sustava.	Visok
NF-05	Prijave nepogoda su omogućene samo unutar Republike Hrvatske	Visok
NF-05.1	Korisnik može prijaviti nepogodu ne više od 30 km od trenutne lokacije	Visok
NF-06	Korisnik ne može prijaviti prirodnu nepogodu sat vremena nakon prethodne prijave, ako takva postoji	Visok
NF-07	Sustav koristi Ouath2.0 autorizaciju korisnika	Visok

Dionici

Navesti dionike koji imaju interes u ovom sustavu ili su nositelji odgovornosti. To su prije svega korisnici, ali i administratori sustava, naručitelji, razvojni tim. Navesti aktore koji izravno koriste ili komuniciraju sa sustavom. Oni mogu imati inicijatorsku ulogu, tj. započinju određene procese u sustavu ili samo sudioničku ulogu, tj. obavljaju određeni posao. Za svakog aktora navesti funkcionalne zahtjeve koji se na njega odnose.

Razvojni tim Administratori Korisnici Vlasti - MUP i MORH Humanitarne organizacije

Aktori i njihovi funkcionalni zahtjevi:

A-1.1 korisnik (kao inicijator) može: F-01; F-02; F-03; F-04; F-05; F-06.3.1; F-07; F-07.1; F-07.2; F-08; F-08.1; F-09; F-09.1; F-09.2; F-09.3; F-23; F-22

A-1.2 korisnik (kao sudionik) može: F-05.1; F-05.2; F-06; F-06.1; F-06.2; F-06.3; F-06.4

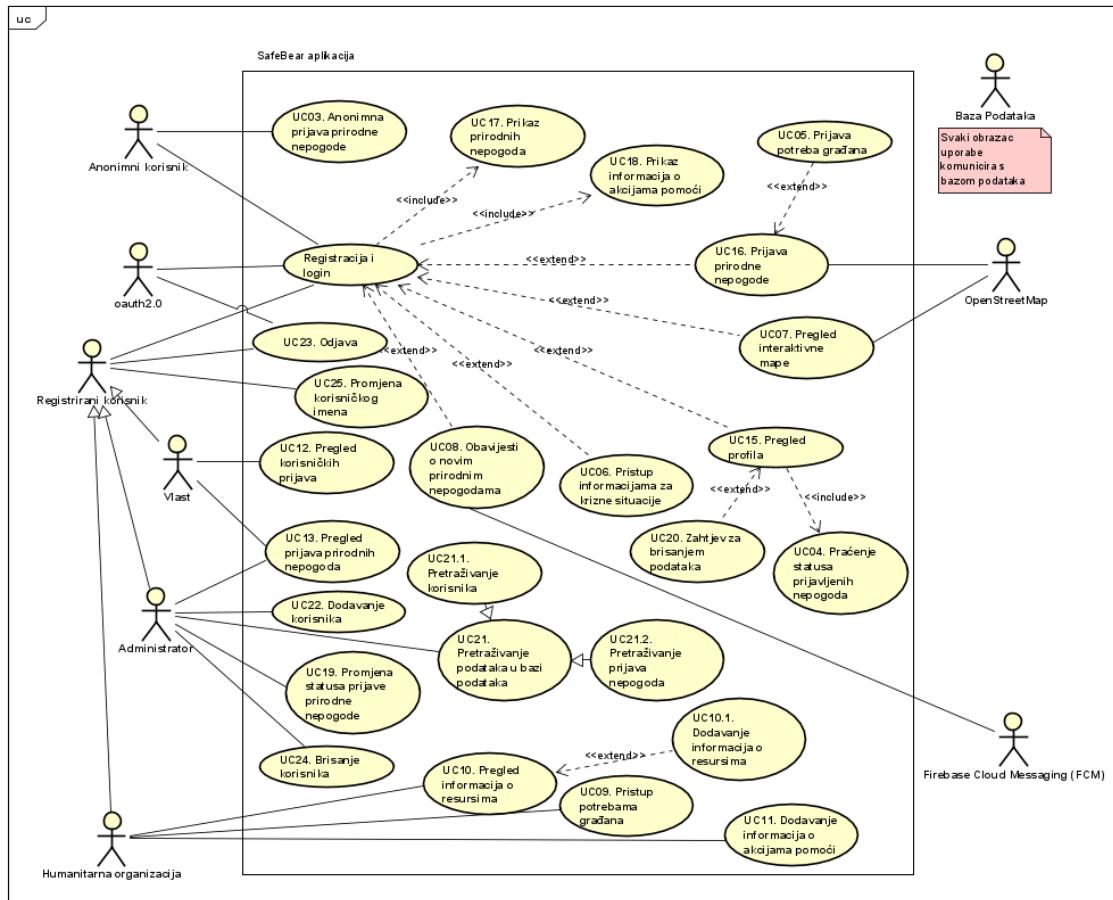
A-2 vlast (inicijator) može: F-10; F-11; F-12

A-3 humanitarna organizacija (inicijator) može: F-13; F-14

A-4 administrator (inicijator) može: F-15; F-16; F-16.1; F-16.2; F-19; F-19.1; F-20; F-20.1; F-20.2; F-21

Obrasci uporabe

Visokorazinski dijagram obrazaca uporabe cijelog sustava



Ključne funkcionalnosti

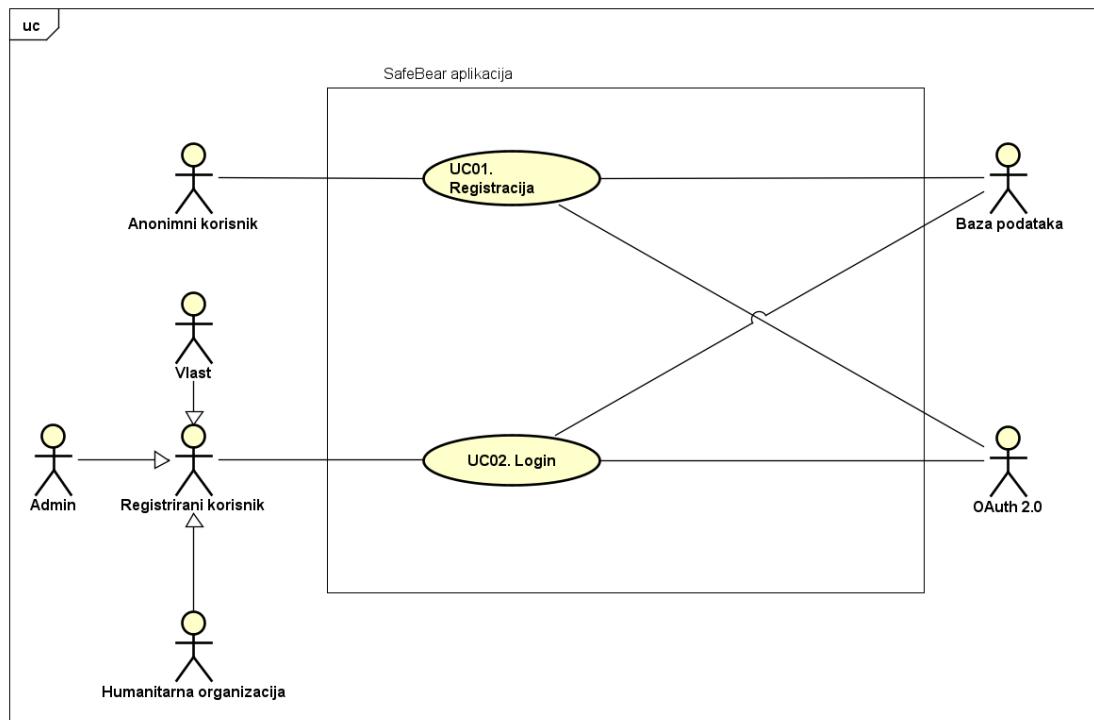
Razlaganje dijagrama na grupe koje se odnose na ključne značajke, poput procesa prijave korisnika, pregleda unutar aplikacije, prijava i praćenja prirodnih nepogoda te upravljanja podacima.

Svaki obrazac koji ne uključuje anonimnog korisnika podrazumijeva da on to može postati kroz ključnu funkcionalnost Registracija i login. Svaki obrazac koji ne uključuje vlast, humanitarne organizacije ili administratora ih ne prikazuje jer nisu nužni za prikazane obrasce, ali se podrazumijeva da se generaliziraju na registriranog korisnika. Ako je registrirani korisnik povezan s bilo kojim obrascem umjesto na UC02 Login ili ključnu funkcionalnost Registracija i login se

podrazumijeva da je povezan na njih i prati ostale obrasce s visokorazinskog dijagrama do dotičnog obrasca.

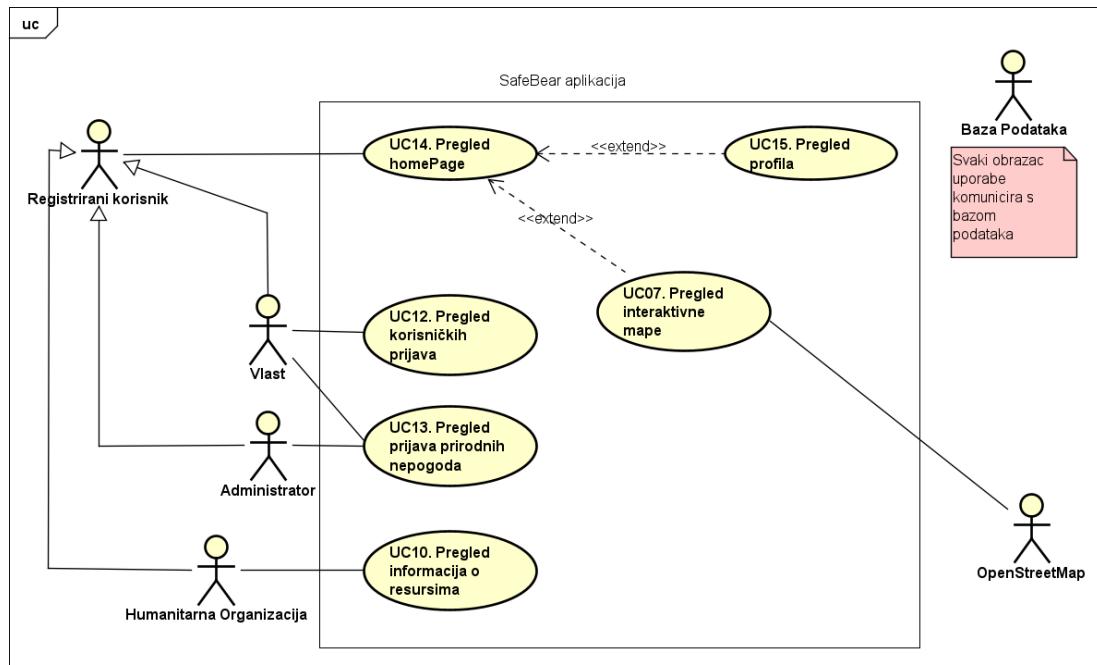
Registracija i login

Uključeni obrasci uporabe: UC01 Registracija, UC02 Login.



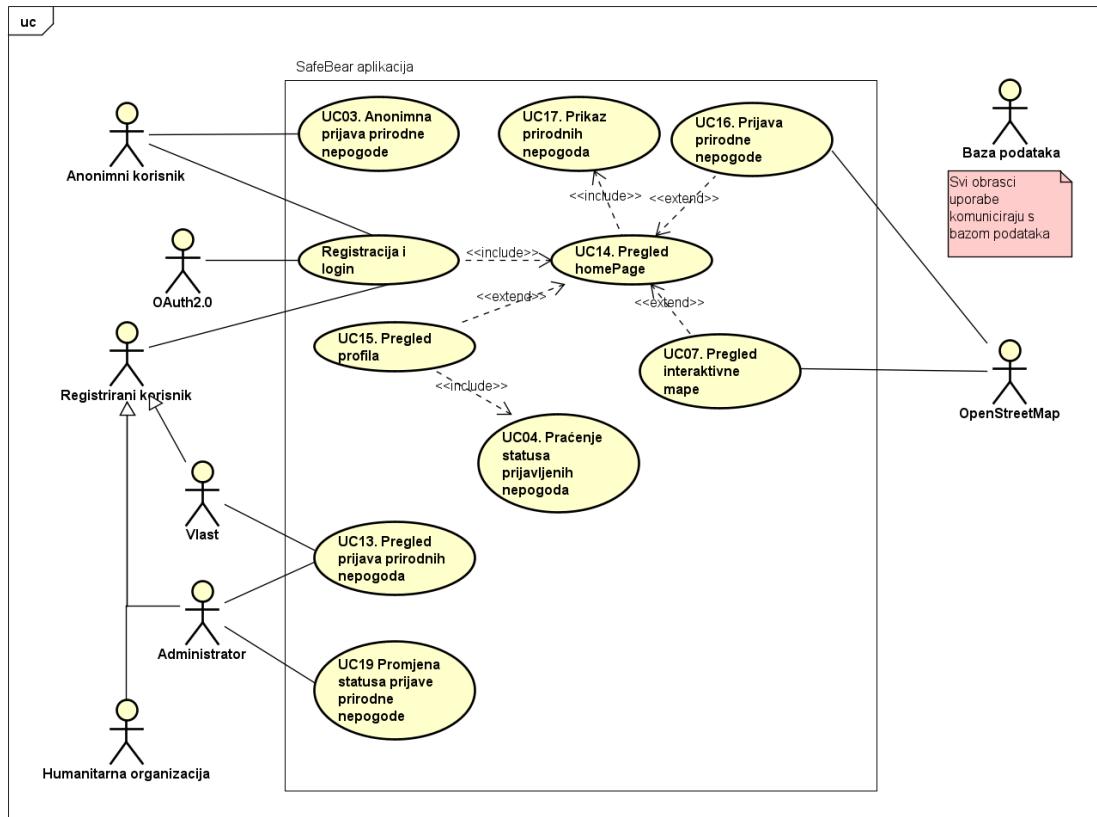
Pregledi unutar aplikacije

Uključeni obrasci uporabe: UC14 Pregled homePage, UC15 Pregled profila, UC07 Pregled interaktivne mape, UC12 Pregled korisničkih prijava, UC10 Pregled informacija o resursima, UC13 Pregled prijava prirodnih nepogoda.



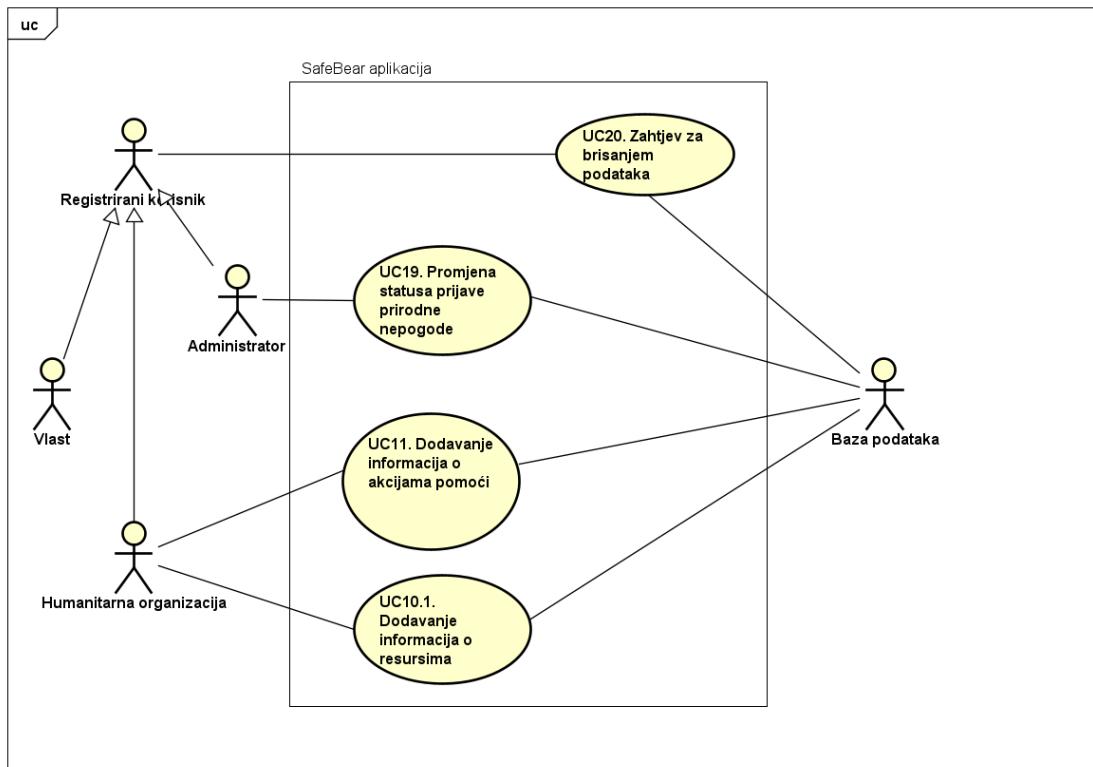
Prijava i praćenje prirodnih nepogoda

Uključeni obrasci uporabe: UC13 Pregled prijava prirodnih nepogoda, UC03 Anonimna prijava prirodne nepogode, UC04 Praćenje statusa prijavljenih nepogoda, UC07 Pregled interaktivne mape, UC16 Prijava prirodne nepogode, UC17 Prikaz prirodnih nepogoda, UC19 Promjena statusa prijave prirodne nepogode. UC14 Pregled homePage i UC15 Pregled profila su korišteni za pojašnjavanje jasnosti implementacije.



Upravljanje podacima

Uključeni obrasci uporabe: UC20 Zahtjev za brisanjem podataka, UC19 Promjena statusa prijave prirodne nepogode, UC10.1 Dodavanje informacija o resursima, UC11 Dodavanje informacija o akcijama pomoći.



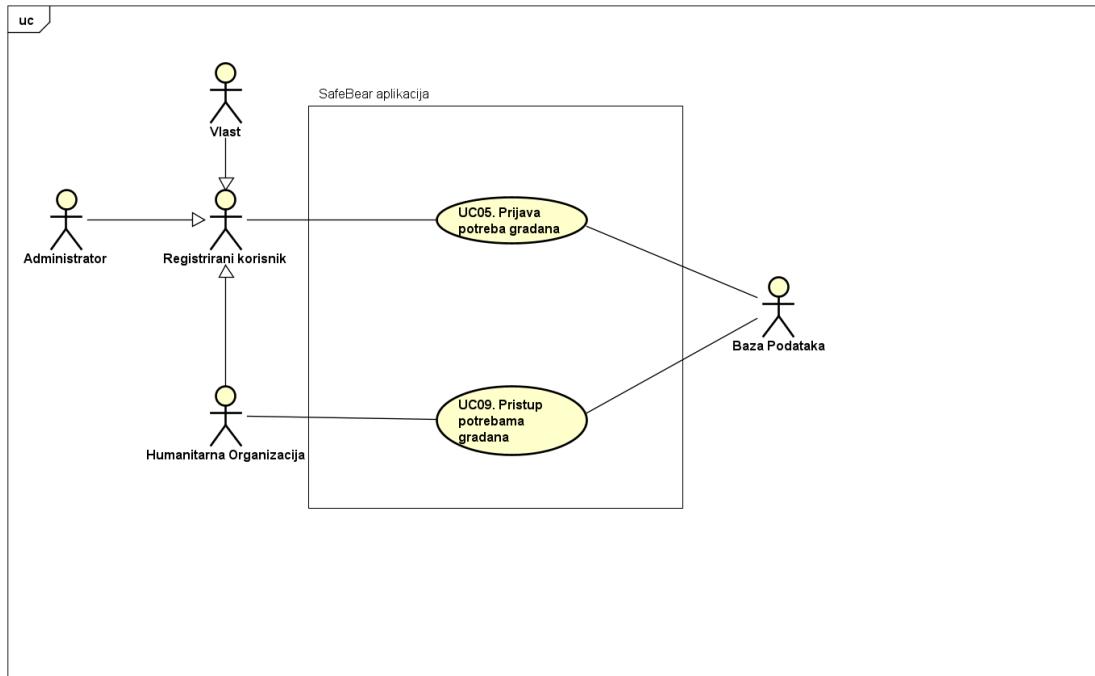
Osnovni poslovni procesi

Ovdje su obrasci uporabe vezani uz praćenje potreba i obavještavanje o promjenama, koji su ključni za rad sustava u hitnim situacijama, kao što su prijave i praćenje potreba građana, informacije i akcije pomoći za krizne situacije te obavještavanje o novim nepogodama.

Svaki obrazac koji ne uključuje anonimnog korisnika podrazumijeva da on to može postati kroz ključnu funkcionalnost Registracija i login. Svaki obrazac koji ne uključuje vlast, humanitarne organizacije ili administratora ih ne prikazuje jer nisu nužni za prikazane obrasce, ali se podrazumijeva da se generaliziraju na registriranog korisnika. Ako je registrirani korisnik povezan s bilo kojim obrascem umjesto na UC02 Login ili ključnu funkcionalnost Registracija i login se podrazumijeva da je povezan na njih i prati ostale obrasce s visokorazinskom dijagrame do dotičnog obrasca.

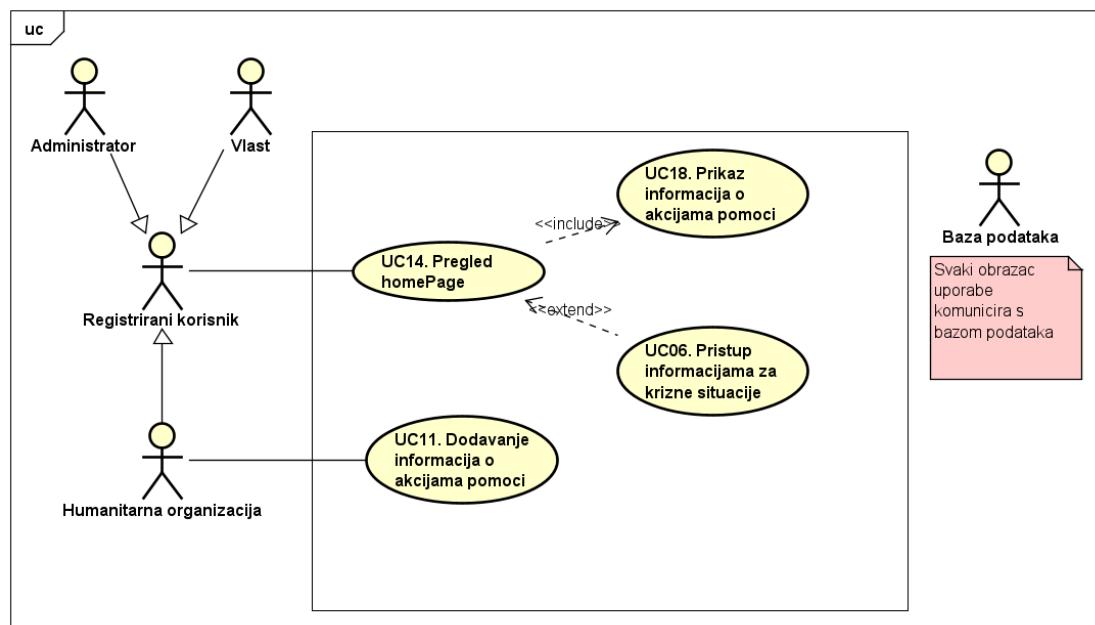
Prijava i praćenje potreba građana

Uključeni obrasci uporabe: UC05 Prijava potreba građana, UC09 Pristup potrebama građana.



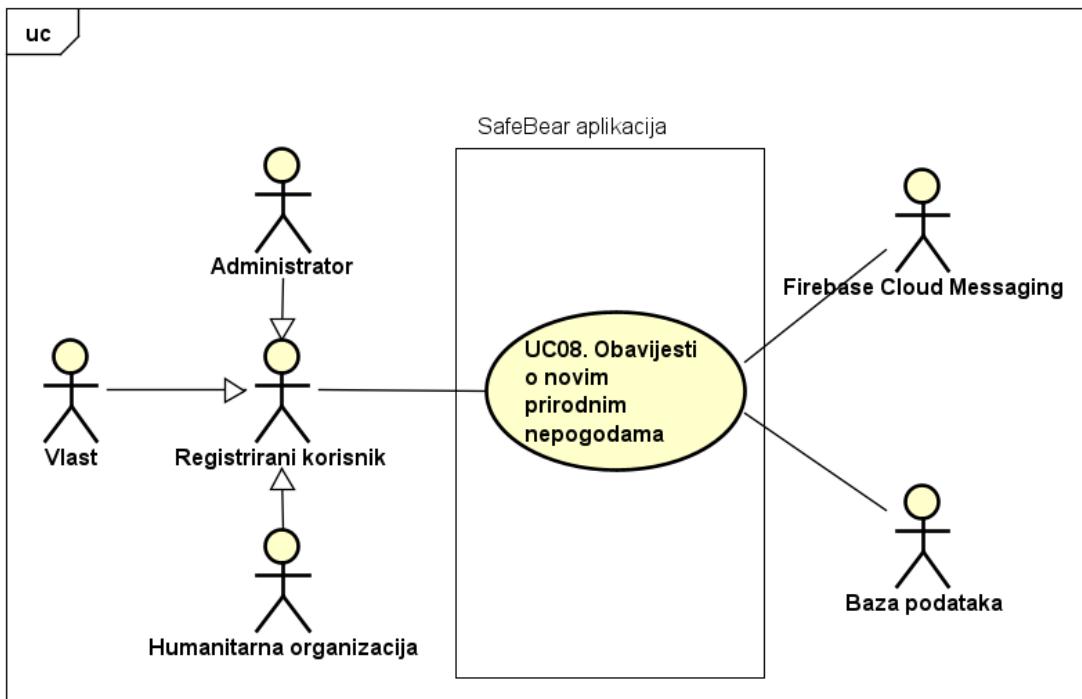
Informacije i akcije pomoći za krizne situacije:

Uključeni obrasci uporabe: UC06 Pristup informacijama za krizne situacije, UC11 Dodavanje informacija o akcijama pomoći, UC18 Prikaz informacija o akcijama pomoći. UC14 Pregled homePage je korišten za pojašnjavanje jasnosti implementacije.



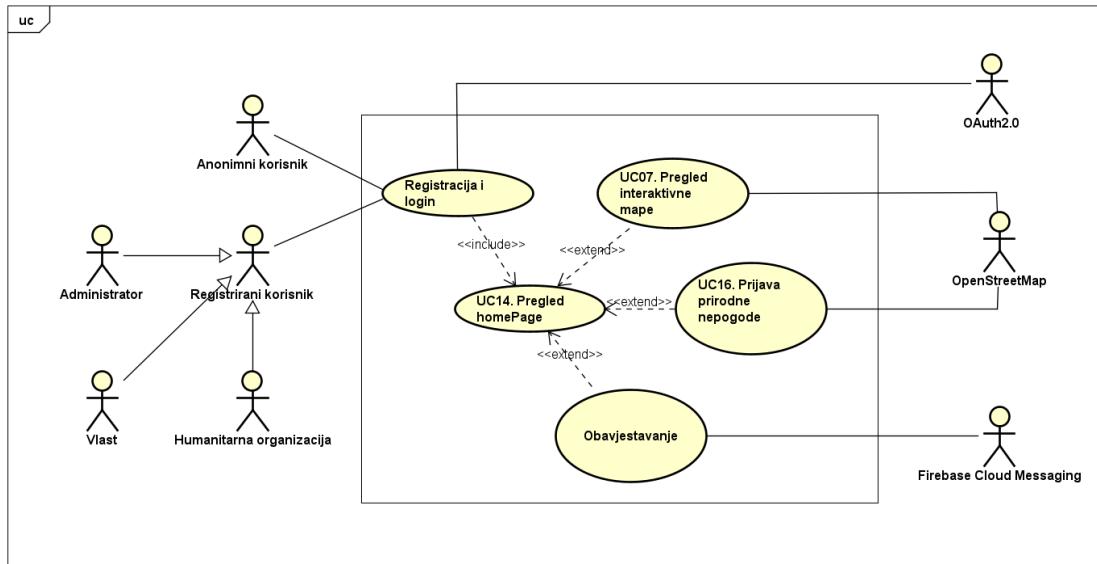
Obavještavanje

Uključeni obrasci uporabe: UC08 Obavijesti o novim prirodnim nepogodama. Trenutni prikaz je vrlo jednostavan, no moduliran je zbog mogućnosti nadogradnje u budućnosti.



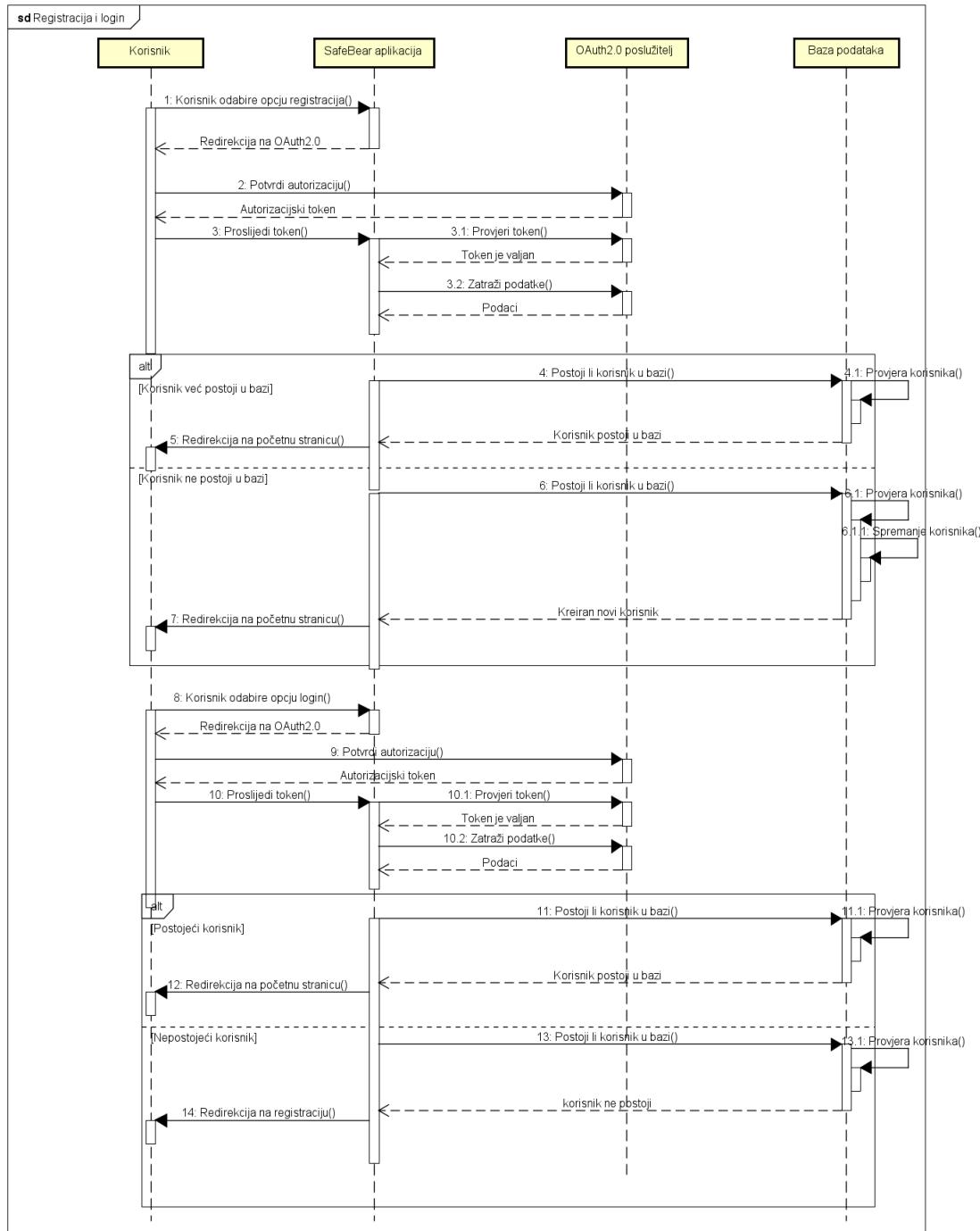
Integracije

Uključeni obrasci uporabe: UC01 Registracija, UC02 Login, UC07 Pregled interaktivne mape, UC08 Obavijesti o novim prirodnim nepogodama, UC16 Prijava prirodne nepogode
Dijagram obrasca uporabe prikazuje integraciju s OAuth 2.0 za autentifikaciju korisnika (UC01 i UC02), s OpenStreetMap API-jem za prikaz interaktivne mape (UC07, UC16) i prijavu nepogoda te Firebase Cloud Messaging (FCM) za slanje push obavijesti korisnicima (UC08). UC01 i UC02 su prikazani kroz ključnu funkcionalnost Registracija i login. UC08 je prikazan kroz osnovni poslovni proces Obavjestavanje. UC14 Pregled homePage je korišten za pojašnjavanje jasnosti implementacije.



Sekvencijski dijagrami

Ovaj sekvencijski dijagram prikazuje proces registracije i logina korisnika. Proces započinje kada korisnik odabire opciju "registracija" prilikom čega ga aplikacija preusmjerava na OAuth2.0 server za autentifikaciju gdje korisnik odobrava autorizaciju. Nakon toga OAuth2.0 korisniku šalje autorizacijski token koji korisnik prosljeđuje aplikaciji. Aplikacija potom na servisu za autentifikaciju provjerava valjanost tokena i prikuplja podatke o korisniku. Potom se u bazi podataka provjerava postoji li već korisnik s tim podacima: ako da, korisnika se preusmjerava na početnu stranicu, ako ne, u bazi podataka se kreira novi korisnik te se potom korisnika preusmjerava na početnu stranicu. U slučaju logina, korisnik odabire opciju "login" prilikom čega ga aplikacija preusmjerava na OAuth2.0 server za autentifikaciju gdje korisnik odobrava autorizaciju. Nakon toga OAuth2.0 korisniku šalje autorizacijski token koji korisnik prosljeđuje aplikaciji. Aplikacija potom na servisu za autentifikaciju provjerava valjanost tokena i prikuplja podatke o korisniku. Potom se u bazi podataka provjerava postoji li već korisnik s tim podacima: ako da, korisnika se preusmjerava na početnu stranicu, ako ne, korisnika se preusmjerava na registraciju.



Ovaj sekvenčni dijagram prikazuje proces pregleda statusa prijavljene nepogode od strane korisnika u aplikaciji. Proces započinje kada korisnik pozove funkciju za pregled statusa prijavljene nepogode (pregled statusa nepogode()). Aplikacija šalje zahtjev bazi podataka da dohvaca informacije o statusu prijavljene nepogode (dohvati podatke o nepogodi). Nakon što baza podataka obradi zahtjev, vraća podatke aplikaciji. Nakon što aplikacija primi podatke o prijavljenoj nepogodi, koristi alt blok kako bi provjerila status

nepogode i prikazala odgovarajuću poruku korisniku. Alternativni slučajevi za prikaz statusa:

1. Podaci su prihvaćeni:

Ako su podaci o nepogodi prihvaćeni, aplikacija prikazuje informacije o prijavljenoj nepogodi uz status "prihvaćeno" (prikaži podatke + status "prihvaćeno").

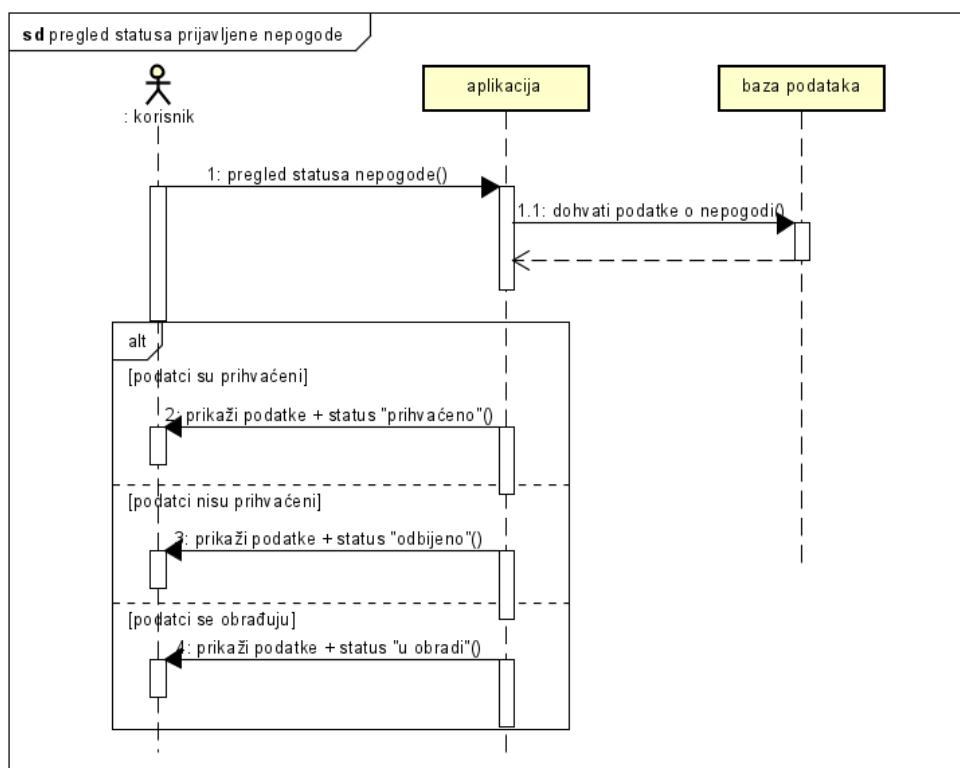
2. Podaci nisu prihvaćeni:

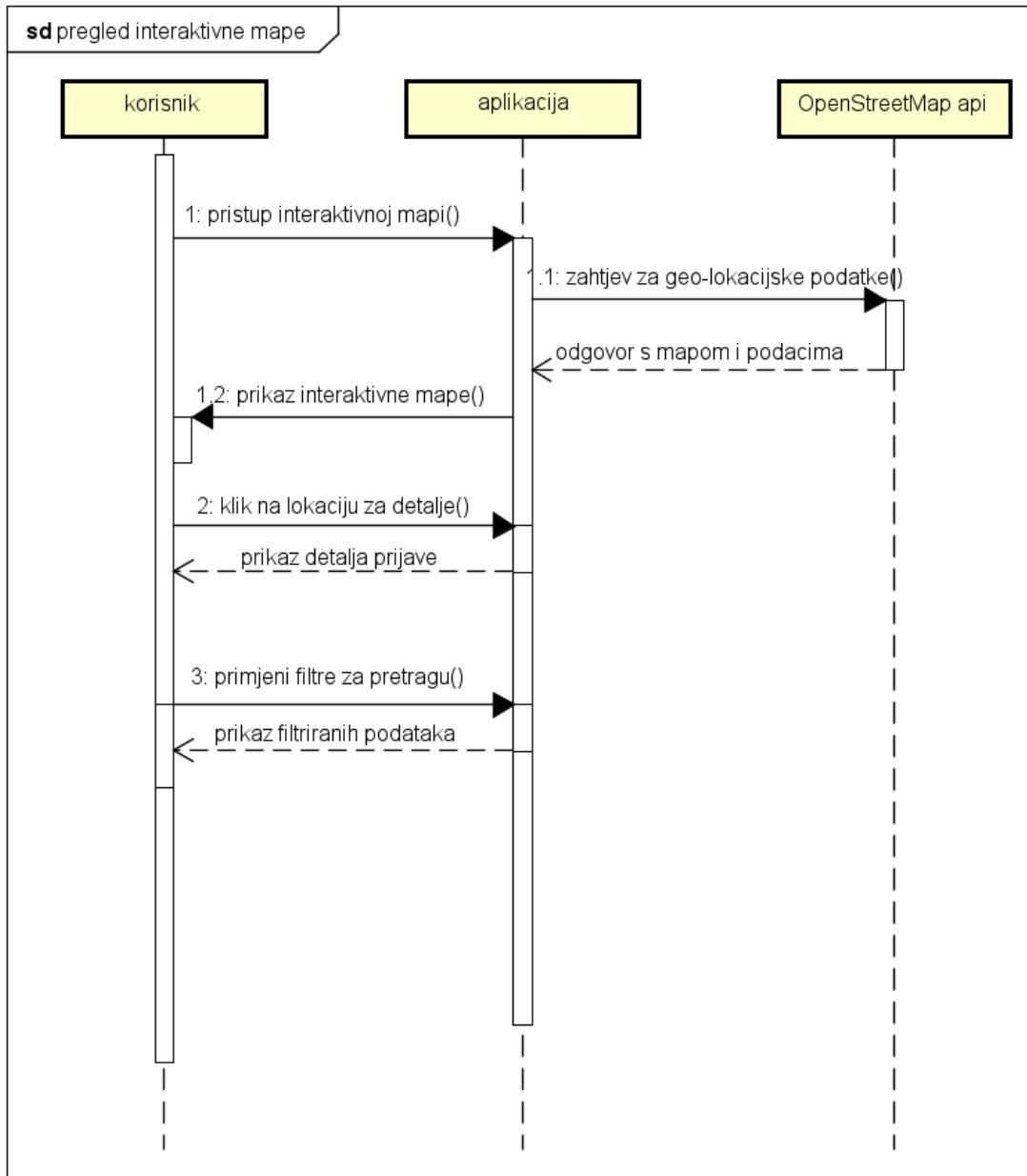
Ako su podaci odbijeni, aplikacija prikazuje informacije uz status "odbijeno" (prikaži podatke + status "odbijeno").

3. Podaci se obrađuju:

Ako su podaci još uvijek u obradi, aplikacija prikazuje informacije uz status "u obradi" (prikaži podatke + status "u obradi").

Nakon što je prikazan odgovarajući status, proces pregleda statusa prijavljene nepogode završava.

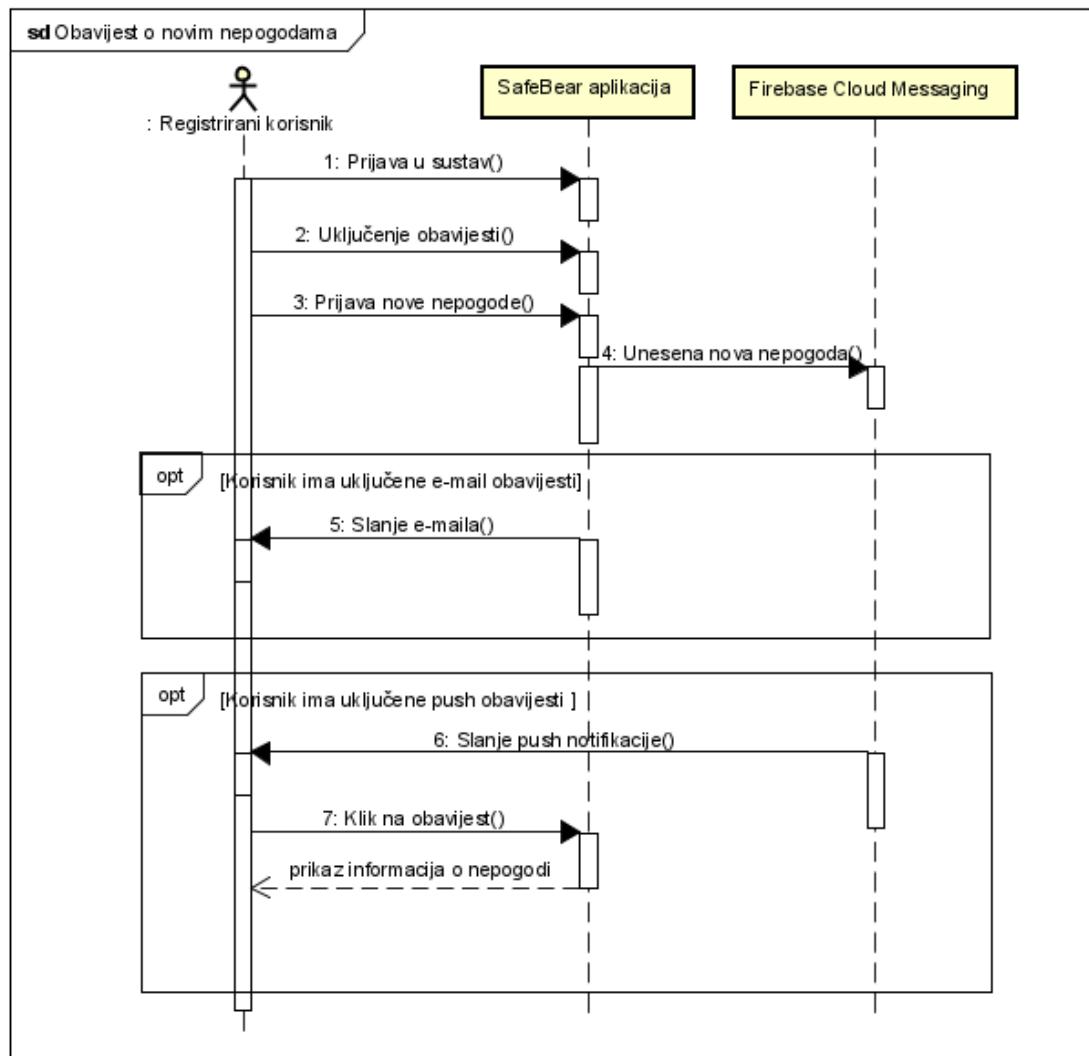




Slanje e-maila - SafeBear aplikacija šalje e-mail korisniku s obavijesti o novoj nepogodi (metoda Slanje e-maila()).

Push obavijest (ako korisnik ima uključene push obavijesti):

Slanje push notifikacije - SafeBear aplikacija, uz pomoć Firebase Cloud Messaging, šalje push notifikaciju korisniku (metoda Slanje push notifikacije()). Korisnik klikne na primljenu obavijest. Nakon klika, korisniku se prikazuje dodatna informacija o novoj nepogodi.



Ovaj sekvencijski dijagram prikazuje proces obrade statusa prijavljene nepogode od strane administratora u aplikaciji. Administrator šalje upit aplikaciji za dohvaćanje podataka o prijavljenoj nepogodi (metoda upit za prijavljenu nepogodu()). Aplikacija šalje zahtjev bazi podataka za potrebne informacije (metoda dohvati podatke()) i prima podatke natrag. Za svaku prijavljenu nepogodu, administrator pregledava valjanost prijave unutar petlje označene kao "loop [za svaku prijavu]". Alternativni tokovi:

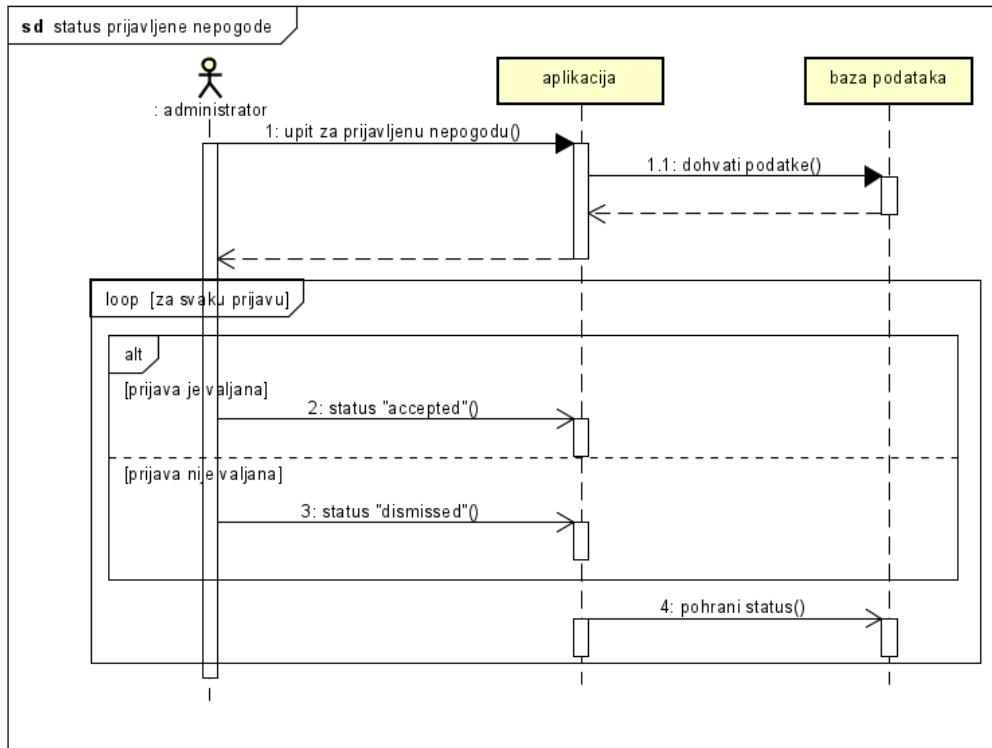
1. Ako je prijava valjana:

Status "accepted" - Administrator postavlja status prijave na "accepted" (metoda status "accepted"()).

2. Ako prijava nije valjana:

Status "dismissed" - Administrator postavlja status prijave na "dismissed" (metoda status "dismissed"()).

Nakon što je prijava obrađena i status određen, aplikacija pohranjuje novi status u bazu podataka (metoda pohrani status()).



Opis obrazaca uporabe

UC01 Registracija

- Glavni sudionik: Anonimni korisnik
- Cilj: Omogućiti korisniku registraciju u sustav putem e-maila.
- Sudionici: Korisnik, Google OAuth 2.0 autentifikacijski poslužitelj
- Preduvjet: Korisnik nije već registriran u sustav.
- Opis osnovnog tijeka:
 1. Korisnik odabire opciju za registraciju putem Google računa.
 2. Sustav preusmjerava korisnika na Googleov OAuth 2.0 autentifikacijski poslužitelj.(F-02)
 3. Korisnik se prijavljuje na Google (ako već nije priavljen) i daje dopuštenje za dijeljenje svojih podataka s aplikacijom.
 4. Googleov autentifikacijski poslužitelj šalje autentifikacijski token aplikaciji.
 5. Aplikacija prima i dekodira token te iz njega dohvaća korisnikove podatke (e-mail, ime i prezime).
 6. Aplikacija provjerava postoji li korisnik već u sustavu s tim e-mailom: Ako postoji, sustav prikazuje poruku da je korisnik već registriran i nudi opciju prijave. Ako ne

postoji, aplikacija sprema korisnikove podatke (e-mail, ime i prezime) u bazu podataka kao novog korisnika.

7. Nakon uspješne registracije, korisnik je automatski prijavljen i preusmjerena na početnu stranicu (homePage).

- Opis mogućih odstupanja:

1. Ako korisnik odbije dati dozvolu za dijeljenje podataka, sustav prekida registraciju i vraća korisnika na početnu stranicu s obavijesti o neuspjeloj registraciji.(F-02.1)
2. Ako dođe do greške pri komunikaciji s Googleovim autentifikacijskim poslužiteljem, sustav obavještava korisnika o tehničkom problemu i nudi ponovno pokušavanje kasnije.

UC02 Login

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima prijavu u sustav pomoću Google OAuth 2.0 autentifikacije.
- Sudionici: Registrirani korisnik, Google OAuth 2.0 autentifikacijski poslužitelj
- Preduvjet: Korisnik je prethodno registriran.
- Opis osnovnog tijeka:
 1. Sustav preusmjerava korisnika na Googleov OAuth 2.0 autentifikacijski poslužitelj.
 2. Korisnik se prijavljuje na Google (ako već nije prijavljen) i daje dopuštenje za dijeljenje svojih podataka s aplikacijom.
 3. Google autentifikacijski poslužitelj šalje autentifikacijski token aplikaciji.
 4. Aplikacija prima token, dekodira ga i dohvaća korisnikove podatke (e-mail, ime).
 5. Ako korisnik postoji u sustavu, sustav prihvata prijavu i preusmjerava korisnika na početnu stranicu.
 6. Ako korisnik ne postoji, sustav prikazuje obavijest o neuspjeloj prijavi i nudi opciju registracije.
 7. Sustav prikazuje korisniku početnu stranicu.
 8. Korisnik ima pristup svom profilu i drugim funkcionalnostima.
- Opis mogućih odstupanja:
 1. Ako korisnik odbije dati dozvolu za dijeljenje podataka pri Google autentifikaciji, sustav prekida prijavu i vraća korisnika na početnu stranicu.
 2. Ako korisnik ne može pristupiti svom računu zbog tehničkih problema, sustav nudi opciju za podršku.

UC03 Anonimna prijava prirodne nepogode

- Glavni sudionik: Anonimni korisnik
- Cilj: Omogućiti anonimnu prijavu prirodne nepogode bez potrebe za registracijom.
- Sudionici: Anonimni korisnik
- Preduvjet: Korisnik je na početnoj stranici aplikacije.

- Opis osnovnog tijeka:
 1. Korisnik unosi podatke o prirodnoj nepogodi (vrsta nepogode, lokacija, opis).
 2. Sustav generira jedinstveni ID za prijavu.
 3. Korisnik može dodati dodatne informacije, poput slika.
 4. Sustav spremi prijavu i šalje korisniku jedinstveni identifikacijski broj.
 5. Korisnik može kasnije pratiti status prijave putem jedinstvenog ID-a (F-08.1).
- Opis mogućih odstupanja:
 1. Ako korisnik ne unese dovoljno informacija, sustav traži dodatne podatke.
 2. Ako sustav ne može obraditi prijavu, korisnik dobiva obavijest o grešci.
 3. Ako se prijava podudara s prethodnom, sustav obavještava korisnika o duplikatu.

UC04 Praćenje statusa prijavljenih nepogoda

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima praćenje statusa prijavljenih nepogoda.
- Sudionici: Registrirani korisnik
- Preduvjet: Korisnik je napravio najmanje jednu prijavu nepogode.
- Opis osnovnog tijeka:
 1. Korisnik ulazi u pregled statusa prijava unutar profila.
 2. Sustav prikazuje status prijava (u obradi, odobreno, odbijeno) ako postoje.
 3. Ako je status promijenjen, korisnik prima obavijest putem e-maila i push notifikacija (F-05.1, F-05.2).
- Opis mogućih odstupanja:
 1. Ako se status prijave ne ažurira pravovremeno, administrator šalje obavijest korisnicima.
 2. Ako prijava nije u sustavu, korisnik dobiva obavijest o nemogućnosti pronađaska statusa.

UC05 Prijava potreba građana

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti građanima prijavu svojih potreba (hrana, voda, medicinska pomoć) nakon prijave nepogode.
- Sudionici: Registrirani korisnik
- Preduvjet: Korisnik je prijavio prirodnu nepogodu.
- Opis osnovnog tijeka:
 1. Građanin unosi svoje potrebe u aplikaciji (hrana, voda, medicinska pomoć).
 2. Sustav povezuje ove potrebe s korisnikom te ih pohranjuje.
 3. Građanin prima obavijest o eventualnim promjenama statusa njegove prijave potreba.
- Opis mogućih odstupanja:
 1. Ako korisnik ne unese sve potrebne informacije, sustav traži dodatne podatke.

UC06 Pristup informacijama za krizne situacije

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima pristup ključnim informacijama tijekom kriznih situacija.
- Sudionici: Registrirani korisnik, Administrator
- Preduvjet: Korisnik je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Korisnik ulazi u pregled homePage.
 2. Sustav generira listu ključnih informacija relevantnih za trenutnu kriznu situaciju.
 3. Korisnik pregledava informacije, uključujući resurse, dostupna skloništa, i upute za postupanje.
 4. Administrator redovito ažurira informacije o situaciji i resursima.
- Opis mogućih odstupanja:
 1. Ako su informacije nepotpune ili netočne, korisnik ima mogućnost prijave administratoru za ispravak.

UC07 Pregled interaktivne mape

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućavanje registriranim korisnicima pristup interaktivnoj mapi koja prikazuje lokacije prijavljenih nepogoda.
- Sudionici: Registrirani korisnik, OpenStreetMap API
- Preduvjet: Korisnik je uspješno prijavljen u aplikaciju i pristupa stranici s interaktivnom mapom.
- Opis osnovnog tijeka:
 1. Korisnik pristupa interaktivnoj mapi.
 2. Mapa prikazuje sve prijave prirodnih nepogoda na temelju geo-lokacije.
 3. Aplikacija šalje zahtjev za prikazivanje mape s podacima o prijavama prema OpenStreetMap API-ju
 4. OpenStreetMap API vraća mapu s označenim lokacijama prijava nepogoda
 5. Korisnik može kliknuti na označene lokacije da bi dobio detalje o vrsti nepogode
 6. Korisnik može filtrirati podatke prema vrsti nepogode, raspoloživim resursima ili statusu prijave.
- Opis mogućih odstupanja:
 1. API ne može dohvatiti podatke zbog mrežnih problema (na primjer, ako OpenStreetMap API nije dostupan ili se događa prekid u mrežnoj vezi)
 2. Korisnik ne može vidjeti mapu ako njegov uređaj nije povezan s internetom ili ako postoje problemi s prikazom mape u aplikaciji

UC08 Obavijesti o novim prirodnim nepogodama

- Glavni sudionik: Registrirani korisnik

- Cilj: Omogućavanje korisnicima primanje obavijesti o novim prirodnim nepogodama u stvarnom vremenu.
- Sudionici: Registrirani korisnik, Firebase Cloud Messaging (FCM)
- Preduvjet: Korisnik je prijavljen u sustav i pretplaćen na obavijesti.
- Opis osnovnog tijeka:
 1. Kada nastane nova prijava prirodne nepogode, aplikacija SafeBear validira i šalje podatke o događaju FCM servisu. 1.1. FCM šalje push obavijest na uređaje svih korisnika koji su omogućili primanje obavijesti putem push notifikacija. 1.2. Aplikacija šalje e-mail obavijest korisnicima koji su omogućili primanje obavijesti putem push notifikacija.
 2. Korisnik prima obavijest na svom uređaju.
 3. Korisnik klikne na obavijest za više informacija o prijavljenoj nepogodi.
 4. Sustav pruža informacije o trenutnom statusu nepogode i sigurnosnim uputama.
- Opis mogućih odstupanja:
 1. Korisnikov uređaj nije povezan s internetom i ne može primiti obavijest; obavijest se šalje kada se uređaj ponovno poveže
 2. Korisnik nije omogućio primanje obavijesti; korisnik može omogućiti obavijesti u postavkama aplikacije

UC09 Pristup potrebama građana

- Glavni sudionik: Humanitarna organizacija
- Cilj: Humanitarne organizacije mogu pregledavati prijavljene potrebe građana (hrana, voda, medicinska pomoć).
- Sudionici: Humanitarna organizacija
- Preduvjet: Humanitarna organizacija je registrirana u sustavu.
- Opis osnovnog tijeka:
 1. Humanitarna organizacija pristupa sustavu.
 2. Organizacija pregledava prijavljene potrebe građana, uključujući zahtjeve za hranom, vodom i skloništima.
 3. Sustav omogućava filtriranje prijava prema prioritetu, geografskoj lokaciji i vrsti pomoći.
 4. Sustav omogućava praćenje napretka dostave pomoći.
- Opis mogućih odstupanja:
 1. Ako organizacija ne može ispuniti zahtjev zbog nedostatka resursa, sustav automatski obavještava korisnike o odgodi.

UC10 Pregled informacija o resursima

- Glavni sudionik: Humanitarna organizacija
- Cilj: Organizacije pregledavaju raspoložive resurse i njihovo stanje.
- Sudionici: Humanitarna organizacija, administrator

- Preduvjet: Organizacija je registrirana i ima pristup resursima.
- Opis osnovnog tijeka:
 1. Organizacija pristupa sustavu i pregledava dostupne resurse (hrana, voda, medicinska oprema).
 2. Sustav prikazuje informacije o trenutnom stanju resursa, uključujući količinu i lokaciju.
 3. Organizacija odlučuje koji resursi će biti distribuirani i u kojem prioritetu.
 4. Sustav ažurira status resursa u realnom vremenu.
 5. Organizacija može dodati nove resurse u sustav ili ažurirati postojeće.
- Opis mogućih odstupanja:
 1. Ako resursi nisu dostupni ili nisu ažurirani, sustav šalje upozorenje organizaciji.
 2. Ako se resursi nalaze na neispravnoj lokaciji, sustav nudi mogućnost ispravka ili preusmjeravanja.
 3. Ako su resursi nepravilno označeni, administrator mora potvrditi ili ispraviti informacije.

UC10.1 Dodavanje informacija o resursima

- Glavni sudionik: Humanitarna organizacija
- Cilj: Dodavanje novih resursa i njihovih količina od strane humanitarnih organizacija.
- Sudionici: Humanitarna organizacija
- Preduvjet: Organizacija je registrirana i ima pristup sustavu.
- Opis osnovnog tijeka:
 1. Organizacija unosi podatke o novim resursima u sustav (vrsta resursa, količina, lokacija).
 2. Sustav automatski provodi validaciju unesenih podataka (ispravnost podataka, dostupnost resursa).
 3. Organizacija potvrđuje unos podataka, a sustav ažurira status resursa.
 4. Resursi su dostupni drugim organizacijama putem sustava.
 5. Sustav omogućuje organizaciji da kasnije ažurira ili ukloni resurse.
- Opis mogućih odstupanja:
 1. Ako uneseni podaci nisu valjani, sustav obavještava organizaciju o grešci i traži ispravke.
 2. Ako resursi nisu dodani zbog tehničkog problema, sustav pokušava ponovno unositi podatke.
 3. Ako resursi nisu dostupni na prijavljenoj lokaciji, organizacija mora obavijestiti sustav za izmjenu podataka.

UC11 Dodavanje informacija o akcijama pomoći

- Glavni sudionik: Humanitarna organizacija

- Cilj: Omogućiti organizacijama unos informacija o svojim akcijama pomoći.
- Sudionici: Humanitarna organizacija
- Preduvjet: Organizacija je registrirana u sustavu i ima pristup funkcionalnostima unosa akcija pomoći.
- Opis osnovnog tijeka:
 1. Organizacija pristupa sučelju za unos akcija pomoći.
 2. Organizacija unosi podatke o akciji (opis, naziv).
 3. Sustav provodi validaciju podataka (provjera unesenih informacija i mogućih sukoba s postojećim akcijama).
 4. Organizacija potvrđuje unos podataka.
 5. Informacije o akciji pomoći postaju vidljive svim korisnicima i organizacijama u sustavu.
- Opis mogućih odstupanja:
 1. Ako organizacija unese netočne podatke, sustav prikazuje upozorenje i zahtjeva korekciju.
 2. Ako akcija nije potvrđena zbog nesuglasica s postojećim podacima, sustav automatski upozorava organizaciju za pojašnjenje.
 3. Ako uneseni podaci ne odgovaraju standardima sustava, administrator može tražiti od organizacije da ih ažurira.

UC12 Pregled korisničkih prijava

- Glavni sudionik: Vlast
- Cilj: Pregledati prijave korisnika
- Sudionici: Vlast
- Preduvjet: Vlasti ima pristup sustavu.
- Opis osnovnog tijeka:
 1. Vlasti pristupaju sučelju za pregled prijava.
 2. Sustav prikazuje sve prijave korisnika.
 3. Vlasti filtriraju prijave prema prioritetima ili potrebama.
 4. Vlasti donose odluku o dalnjoj akciji ili obavještavaju korisnike.
- Opis mogućih odstupanja:
 1. Ako su podaci iz prijave netočni, vlasti mogu označiti prijavu kao nevažeću.

UC13 Pregled prijava prirodnih nepogoda

- Glavni sudionik: Administrator, vlast
- Cilj: Pregledati sve prijave prirodnih nepogoda u sustavu.
- Sudionici: Administrator, vlast
- Preduvjet: Administrator ili vlast ima pristup sustavu.
- Opis osnovnog tijeka:
 1. Administrator ili vlast pristupa sučelju za pregled prijava nepogoda.

2. Sustav prikazuje sve prijave nepogoda.
 3. Administrator ili vlast mogu prema različitim filterima (vrsta nepogode, status, regija) sortirati sadržaj.
 - Opis mogućih odstupanja:
1. Ako prijavu nije moguće obraditi, sustav obaveštava administratore o grešci.
 2. Ako je prijava nepravilno označena, administrator mora izvršiti korekciju.

UC14 Pregled homePage

- Glavni sudionik: Registrirani korisnik
- Cilj: Prikaz početne stranice s glavnim informacijama.
- Sudionici: Registrirani korisnik
- Preduvjet: Korisnik je ulogiran u sustav.
- Opis osnovnog tijeka:
 1. Korisnik pristupa početnoj stranici sustava.
 2. Sustav prikazuje najnovije informacije o prijavama nepogoda, potrebama građana, resursima i akcijama pomoći.
 3. Na početnoj stranici su vidljive i najvažnije obavijesti (novosti, promjene u statusu nepogoda).
 4. Korisnik može pristupiti dubljim informacijama ili se prebaciti na druge sekcije sustava.
 5. Početna stranica je optimizirana za prikaz na različitim uređajima (mobilni, desktop).
- Opis mogućih odstupanja:
 1. Ako stranica nije ispravno učitana, sustav automatski pokušava ponovno učitati sadržaj.
 2. Ako korisnik ne vidi ažurirane obavijesti, sustav šalje obavijest za ručno osvježavanje.

UC15 Pregled profila

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima pregled i ažuriranje svog profila.
- Sudionici: Registrirani korisnik
- Preduvjet: Korisnik je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Korisnik pristupa svom profilu u sustavu.
 2. Sustav prikazuje informacije o korisniku (korisničko ime, povijest prijava).
 3. Korisnik može ažurirati svoje osobne podatke, postavke obavijesti ili lozinku.
 4. Korisnik potvrđuje promjene koje sustav automatski spremi.
 5. Ažurirani podaci postaju vidljivi svim relevantnim organizacijama ili vlastima.
- Opis mogućih odstupanja:

1. Ako korisnik unese neispravan format za lozinku ili osobne podatke, sustav prikazuje upozorenje o pogrešnim podacima.

UC16 Prijava prirodne nepogode

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti registriranim korisnicima prijavu prirodnih nepogoda uz precizno označavanje lokacije.
- Sudionici: Registrirani korisnik, OpenStreetMap API
- Preduvjet: Korisnik je registriran i prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Korisnik pristupa sučelju za prijavu nepogode.
 2. Sustav koristi OpenStreetMap API za učitavanje interaktivne mape i omogućava korisniku unos lokacije nepogode putem karte.
 3. Korisnik označava točnu lokaciju na karti uz pomoć interaktivnog sučelja koje pruža OpenStreetMap API.
 4. Korisnik dodaje prateće informacije (vrstu nepogode, kratak opis) te opcionalno fotografiju.
 5. Sustav provodi validaciju unesenih podataka.
 6. Prijava se šalje sustavu i postaje vidljiva relevantnim vlastima ili organizacijama.
- Opis mogućih odstupanja:
 1. Ako korisnik unese netočne podatke, sustav automatski traži korekciju.
 2. Ako API za prikaz mape nije dostupan, korisnik može ručno unijeti adresu ili koordinate.
 3. Ako prijava nije moguća zbog tehničkog problema, korisniku se nudi alternativa za prijavu.
 4. Ako je korisnik napravio prijavu prije manje od sat vremena, sustav šalje obavijest da pokuša ponovno kasnije.
 5. Ako je prijava van Republike Hrvatske, sustav obaviještava korisnika da aplikacija SafeBear trenutno nije implementirana van Republike Hrvatske.
 6. Ako korisnik pokuša napraviti prijavu više od 30 km od korisnikove trenutne lokacije, sustav obaviještava korisnika da ne može raditi prijave za udaljene lokacije.

UC17 Prikaz prirodnih nepogoda

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima prikaz relevantnih prijava prirodnih nepogoda koje su evidentirane u sustavu.
- Sudionici: Registrirani korisnik
- Preduvjet: Korisnik je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Korisnik ulazi u pregled homePage i automatski dobija prikaz prijava nepogoda.

2. Korisnik može filtrirati ili sortirati nepogode prema kriterijima kao što su datum, tip nepogode.
 3. Klikom na određenu nepogodu, korisnik dobiva detaljan pregled, uključujući informacije o ozbiljnosti, nadležnoj organizaciji, poduzetim akcijama i eventualnim potrebama za pomoć.
 4. Korisnik može odabrati da prati određenu prijavu kako bi dobivao obavijesti o promjenama statusa ili novim ažuriranjima.
- Opis mogućih odstupanja:
 1. Ako sustav ne može učitati listu prijava zbog tehničkog problema, korisniku se prikazuje poruka o grešci i savjetuje ponovno pokušavanje kasnije.

UC18 Prikaz informacija o akcijama pomoći

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima prikaz informacija o akcijama pomoći.
- Sudionici: Registrirani korisnik
- Preduvjet: Korisnik je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Korisnik pristupa pregledu homePage.
 2. Sustav prikazuje listu aktivnih i prošlih akcija pomoći s osnovnim podacima (naziv).
 3. Klikom na određenu akciju, korisnik dobiva detaljan opis akcije pomoći.
 4. Korisnik može pregledati i ažuriranja ili komentare vezane uz akciju.
- Opis mogućih odstupanja:
 1. Ako sustav ne može učitati listu akcija, prikazuje se poruka o grešci i korisnik se upućuje na tehničku podršku.
 2. Ako filtriranje nije moguće zbog neispravnih unosa, sustav prikazuje poruku za prilagodbu kriterija pretrage.

UC19 Promjena statusa prijave prirodne nepogode

- Glavni sudionik: Administrator
- Cilj: Omogućiti administratorima ažuriranje statusa prijava prirodnih nepogoda kako bi informirali korisnike o napretku rješavanja prijava.
- Sudionici: Administrator
- Preduvjet: Administrator ima pristup sustavu.
- Opis osnovnog tijeka:
 1. Administrator pristupa sučelju za upravljanje prijavama prirodnih nepogoda.
 2. Administrator odabire prijavu kojoj želi promijeniti status.
 3. Sustav prikazuje trenutne informacije o prijavi, uključujući trenutni status i detalje.
 4. Administrator ažurira status prijave (u obradi, riješeno, na čekanju).
 5. Sustav pohranjuje promjenu i šalje obavijest korisnicima vezanim uz prijavu.
- Opis mogućih odstupanja:

1. Ako se promjena statusa ne može pohraniti zbog tehničkog problema, administrator dobiva poruku o grešci i savjet za ponovno pokušavanje kasnije.

UC20 Zahtjev za brisanjem podataka

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima podnošenje zahtjeva za brisanje svojih podataka iz sustava.
- Sudionici: Registrirani korisnik, administrator
- Preduvjet: Korisnik je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Korisnik pristupa svom profilu i odabire opciju za podnošenje zahtjeva za brisanje podataka.
 2. Sustav traži potvrdu zahtjeva za brisanje kako bi se spriječila slučajna brisanja.
 3. Korisnik potvrđuje zahtjev za brisanje podataka.
 4. Sustav automatski obavještava administratora o zahtjevu za brisanje.
 5. Administrator pregleda zahtjev, odobrava ga i potvrđuje korisniku da su podaci trajno obrisani.
- Opis mogućih odstupanja:
 1. Ako korisnik pokuša podnijeti zahtjev za brisanje bez valjane autorizacije, sustav ga obavještava o potrebi za prijavom.
 2. Ako zahtjev za brisanje ne može biti obrađen zbog tehničkog problema, korisnik dobiva poruku o grešci i upute za daljnje korake.

Provjera uključenosti funkcionalnih zahtjeva u obrasce uporabe

Redni broj	Naziv obrasca uporabe	Funkcionalni zahtjevi
UC01	Registracija	F-02, F-02.1, F-02.1.1, F-18, F-18.1
UC02	Login	F-03
UC03	Anonimna prijava prirodne nepogode	F-08, F-08.1
UC04	Praćenje statusa prijavljenih nepogoda	F-05, F-05.3
UC05	Prijava potreba građana	F-09, F-09.1, F-09.2, F-09.3
UC06	Pristup informacijama za krizne situacije	F-07
UC07	Pregled interaktivne mape	F-01
UC08	Obavijesti o novim prirodnim nepogodama	F-06, F-06.1, F-06.2, F-06.3, F-06.3.1
UC09	Pristup potrebama građana	F-12, F-12.1

Redni broj	Naziv obrasca uporabe	Funkcionalni zahtjevi
UC10	Pregled informacija o resursima	F-13.1.1
UC10.1	Dodavanje informacija o resursima	F-13
UC11	Dodavanje informacija o akcijama pomoći	F-14
UC12	Pregled korisničkih prijava	F-10
UC13	Pregled prijava prirodnih nepogoda	F-11, F-15
UC14	Pregled homePage	/
UC15	Pregled profila	/
UC16	Prijava prirodne nepogode	F-04
UC17	Prikaz prirodnih nepogoda	F-01
UC18	Prikaz informacija o akcijama pomoći	/
UC19	Promjena statusa prijave prirodne nepogode	F-16, F-16.1, F-16.2
UC20	Zahtjev za brisanjem podataka	F-17, F-17.1
UC21	Pretraživanje podataka u bazi podataka	F-20
UC21.1	Pretraživanje korisnika	F-20.2
UC21.2	Pretraživanje prijava nepogoda	F-20.1
UC22	Dodavanje korisnika	F-19
UC23	Odjava	F-23
UC 24	Brisanje korisnika	F-21
UC 25	Promjena korisničkog imena	F-22

Arhitektura

Izbor arhitekture

Odabrana arhitektura za SafeBear sustav je višeslojna arhitektura s Model-View-Controller (MVC) obrascem u aplikacijskom sloju.

Ova je izabrana zbog jasne podjele odgovornosti, skalabilnosti, sigurnosti te fleksibilnosti. Detaljnije, višeslojna arhitektura omogućava jasnu podjelu među različitim funkcionalnostima sustava, što olakšava razvoj i održavanje. Razdvajanjem klijentskog, aplikacijskog te sloja baze podataka omogućećemo skaliranje svakog sloja neovisno, ovisno o potrebama u razvoju. Višeslojna arhitektura također omogućuje dodatne sigurnosne mјere, kao što su zaštita aplikacijskog sloja kroz poslužitelje za autorizaciju i autentifikaciju. MVC (Model View Controller) arhitektura u aplikacijskom sloju omogućava timovima paralelan rad na sučelju, logici te upravljuju podacima.

Organizacija sustava

U sustavu je primijenjen tip arhitekture klijent-poslužitelj s centraliziranim bazom podataka. Na klijentskoj sloju korisnici komuniciraju s aplikacijom putem web preglednika. Ovaj sloj upravlja prikazom i interakcijom korisničkog sučelja. Aplikacijski sloj je poslužiteljski sloj koji sadrži logiku aplikacije. Unutar ovog sloja obavlja se upravljanje prijavama, obavještavanje o nepogodama, dodavanje informacija o resursima i ostalo u skladu s funkcijskim zahtjevima sustava. Baza podataka je centralizirana, te pohranjuje sve korisničke podatke, podatke o prijavama prirodnih nepogoda, resursima, akcijama pomoći, humanitarnim organizacijama te lokacijama. Klijentski sloj komunicira s aplikacijskim slojem, koji komunicira dalje s bazom podataka.

Organizacija aplikacije (frontend, backend, MVC)

Frontend (Klijentski sloj) implementiran je koristež tehnologiju React. Komunikacija s backendom se odvija putem REST API-ja. On sadrži sve što je potrebno za interakciju s korisnikom, primarno podijeljen na različite preglede, npr. homePage, profil, interaktivna mapa. Unutar različitih pregleda, korisniku se prikazuju informacije vezane uz nepogode, kao što su prijašnje prijave prikazane u formatu objave i oznake na mapi, sigurnosne mjere, akcije pomoći i slično.

Backend (Aplikacijski sloj) organizira se prema MVC arhitekturi. Modeli obuhvaćaju podatkovne strukture i logiku koja povezuje aplikaciju s bazom podatka (Hibernate). Uloga view komponente je u ovoj arhitekturi ograničena na API odgovore, frontend sloj preuzima podatke u JSON formatu te ih prikazuje korisnicima. Controlleri uključuju logiku koja obrađuje zahtjeve korisnika, validira podatke i koristi Model komponentu. Također se brine za autentifikaciju i autorizaciju korisnika.### ## #

Frontend sloj komunicira s Controllerima na backendu putem HTTP protokola (REST API).

Identifikacija podsustava

U sustavu SafeBear mogu se identificirati sljedeći podsustavi:

Podsustav za autentifikaciju i autorizaciju koji omogućava prijavu i registraciju korisnika, upravljanje sesijama i sigurnosnim postavkama. Koristi OAuth 2.0 za prijavu putem vanjskih identiteteskih davatelja.

Podsustav za upravljanje prijavama prirodnih nepogoda koji omogućava prijavu prirodnih nepogoda, praćenje prijava te pregled statusa. Podržava komunikaciju između korisniku i nadležnih tijela.

Podsustav za obavještavanje i informiranje koji prikuplja podatke o nepogodama i šalje obavijesti korisnicima putem push notifikacija ili e-maila. Sadrži prikaz interaktivne mape i prikaz informacija o akcijama pomoći.

Podsustav za upravljanje resursima kroz koji humanitarne organizacije mogu dodavati resurse i informacije o akcijama pomoći. Informacije o akcijama pomoći dostupne su drugim korisnicima za pregled.

Preslikavanje na radnu platformu

Frontend je implementiran pomoću tehnologije React.

Backend server je postavljen na Java Spring Boot.

Za bazu podataka se koristi relacijska baza podataka implementirana pomoću H2 tehnologije.

API poslužitelji koji se koriste su REST za komunikaciju između frontend i backend sloja.

Spremišta podataka

Centralizirana relacijska baza podataka čuva sve podatke o korisnicima, prijavama nepogoda, resursima i akcijama pomoći.

Mrežni protokoli

HTTP/HTTPS: Za sigurnu komunikaciju između klijentskog i aplikacijskog sloja. OAuth 2.0: Koristi se za autentifikaciju i autorizaciju putem vanjskih identitetskih davatelja.

Globalni upravljački tok

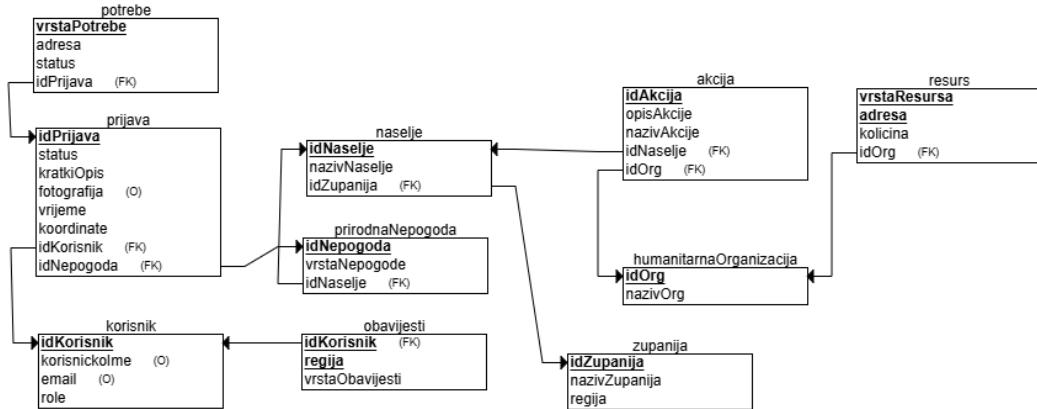
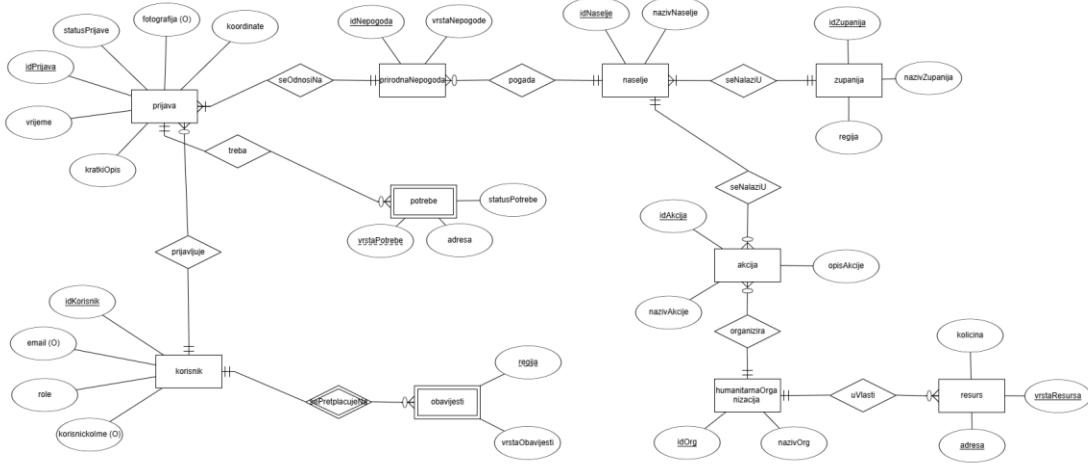
Korisnik se prijava ili registrira (autentifikacija). Aplikacijski sloj obrađuje korisničke zahtjeve za prijavu nepogoda, pregled informacija o resursima, i obavijesti o akcijama pomoći. Backend kontrolira pristup podacima u bazi podataka kroz autorizaciju i validaciju podataka. Korisnici primaju ažurirane podatke putem klijentskog sloja, a obavijesti u stvarnom vremenu koristeći push notifikacije.

Sklopoškoprogramski zahtjevi

1. Serverska strana 1.1. Poslužitelj mora imati dovoljno resursa za podršku aplikacijskog sloja i baze podataka.
2. Klijentska strana 2.1. Klijenti trebaju moderan web preglednik ili mobilnu aplikaciju koja komunikaciju putem REST API-ja.

Baza podataka

Dijagram baze podataka



Opis tablica

humanitarnaOrganizacija

Atribut	Tip podatka	Opis varijable
idOrg	INT	Jedinstveni identifikator organizacije
nazivOrg	VARCHAR(100)	Naziv organizacije

naselje

Atribut	Tip podatka	Opis varijable
idNaselje	INT	Jedinstveni identifikator naselja
nazivNaselje	VARCHAR(50)	Naziv naselja
idZupanija (FK)	INT	Jedinstveni identifikator županije

resurs

Atribut	Tip podatka	Opis varijable
adresa	VARCHAR(100)	Adresa resursa
vrstaResursa	VARCHAR(100)	Sklonište ili vrsta resursa
idOrg (FK)	INT	Jedinstveni identifikator humanitarne organizacije
kolicina	INT	Količina resursa ili kapacitet skloništa

akcija

Atribut	Tip podatka	Opis varijable
idAkcija	INT	Jedinstveni identifikator akcije
opisAkcije	VARCHAR(1000)	Opis akcije
nazivAkcije	VARCHAR(100)	Naziv akcije
idNaselje (FK)	INT	Jedinstveni identifikator za naselje
idOrg (FK)	INT	Jedinstveni identifikator humanitarne organizacije

prirodnaNepogoda

Atribut	Tip podatka	Opis varijable
idNepogoda	INT	Jedinstveni identifikator nepogode
vrstaNepogode	VARCHAR(50)	Vrsta prirodne nepogode
idNaselje (FK)	INT	Jedinstveni identifikator za naselje

zupanija

Atribut	Tip podatka	Opis varijable
idZupanija	INT	Jedinstveni identifikator županije
nazivZupanija	VARCHAR(50)	Naziv županije
regija	VARCHAR(20)	Jedinstveni identifikator za regiju

korisnik

Atribut	Tip podatka	Opis varijable
idKorisnik	INT	Jedinstveni identifikator korisnika
korisnickoIme	VARCHAR(20)	Korisničko ime
role	VARCHAR(20)	Uloga
email	VARCHAR(50)	Email adresa

prijava

Atribut	Tip podatka	Opis varijable
idPrijava	INT	Jedinstveni identifikator prijave

Atribut	Tip podatka	Opis varijable
status	VARCHAR(20)	Status prijave
kratkiOpis	VARCHAR(1000)	Kratki opis prijave
fotografija	VARCHAR(50)	Putanja do fotografije
koordinate	VARCHAR(50)	Geografske koordinate lokacije nepogode
vrijeme	TIMESTAMP	Vrijeme prijave
idKorisnik (FK)	INT	Jedinstveni identifikator korisnika
idNepogoda (FK)	INT	Jedinstveni identifikator nepogode

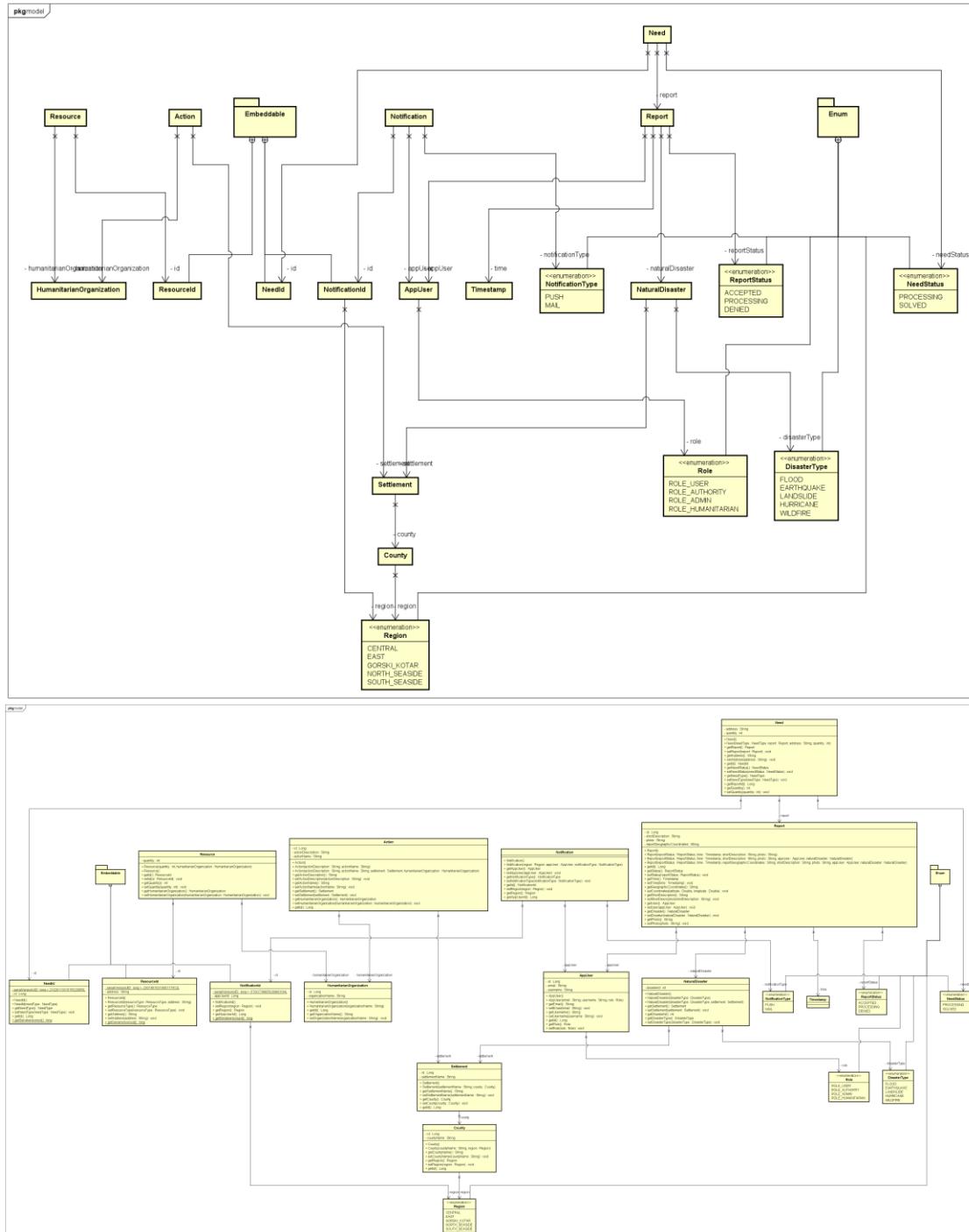
obavijesti

Atribut	Tip podatka	Opis varijable
vrstaObavijesti	VARCHAR(50)	Vrsta obavijesti
idKorisnik (FK)	INT	Jedinstveni identifikator korisnika
regija	VARCHAR(20)	Jedinstveni identifikator za regiju

potrebe

Atribut	Tip podatka	Opis varijable
vrstaPotrebe	VARCHAR(50)	Vrsta potrebe
adresa	VARCHAR(100)	Adresa gdje je potrebna pomoć
status	VARCHAR(20)	Status potrebe
idPrijava (FK)	INT	Jedinstveni identifikator prijave

Dijagram razreda



Prva slika prikazuje jednostavan prikaz razreda s fokusom na veze.

Opis Dijagrama Razreda

Ovaj dijagram razreda prikazuje strukturu sustava za praćenje i reagiranje na prirodne katastrofe. U sustavu su definirane glavne klase koje omogućuju prijavu i obradu izvještaja o katastrofama, alokaciju resursa, obavijesti korisnicima i zadovoljavanje potreba pogodjenih područja.

Opći opis strukture

Sustav se sastoji od nekoliko glavnih komponenti:

Akcije pomoći (Action) – Održava informacije o aktivnostima koje humanitarne organizacije poduzimaju u reagiranju na prirodne nepogode.

Atribut	Tip podatka	Opis varijable
id	Long	Jedinstveni identifikator akcije
actionDescription	String	Kratki opis akcije
actionName	String	Naziv akcije
settlement	Settlement	Referenca na naselje
humanitarianOrganization	HumanitarianOrganization	Referenca na humanitarnu organizaciju

Resursi (Resource) – Prikazuje dostupne resurse (npr. hranu, vodu, medicinske potrepštine) koji se koriste za pružanje pomoći.

Atribut	Tip podatka	Opis varijable
id	ResourceId	Jedinstveni kompozitni ključ resursa
quantity	int	Dostupna količina resursa
humanitarianOrganization	HumanitarianOrganization	Referenca na humanitarnu organizaciju

Opis ključa

Atribut	Tip podatka	Opis varijable
resourceType	ResourceType (Enum)	Vrsta resursa
address	String	Adresa na kojoj se resurs nalazi

Atribut	Tip podatka	Opis varijable
id	Long	Jedinstveni identifikator reporta
reportStatus	ReportStatus (Enum)	Status prijave nepogode
time	Timestamp	Vrijeme prijave nepogode

Atribut	Tip podatka	Opis varijable
shortDescription	String	Kratki opis nepogode
photo	String	Putanja do slike
reportGeographicCoordinates	String	Koordinate lokacije nepogode

Atribut	Tip podatka	Opis varijable
id	NotificationId	Jedinstveni kompozitni identifikator reporta
notificationType	NotificationType (Enum)	Vrsta notifikacije
appUser	AppUser	Referenca na korisnika

Opis ključa

Atribut	Tip podatka	Opis varijable
region	Region (Enum)	Regija uz koju se veže obavijest
appId	Long	ID korisnika

Atribut	Tip podatka	Opis varijable
id	Long	Jedinstveni identifikator korisnika
email	String	E-mail korisnika
username	String	Korisničko ime korisnika
role	Role (Enum)	Uloga korisnika

Atribut	Tip podatka	Opis varijable
id	needId	Jedinstveni kompozitni identifikator potrebe
address	String	Adresa vezana uz potrebu
needStatus	NeedStatus (Enum)	Status potrebe

Opis ključa

Atribut	Tip podatka	Opis varijable
needType	NeedType (Enum)	Vrsta potrebe
id	Long	ID reporta

Atribut	Tip podatka	Opis varijable
id	Long	Jedinstveni identifikator korisnika
organizationName	String	Naziv humanitarne organizacije

Atribut	Tip podatka	Opis varijable
disasterId	int	Jedinstveni identifikator nepogode
disasterType	DisasterType (Enum)	Vrsta prirodne nepogode
settlement	Settlement	Referenca na naselje

Atribut	Tip podatka	Opis varijable
id	Long	Jedinstveni identifikator naselja
settlementName	String	Naziv naselja
county	County	Referenca na županiju

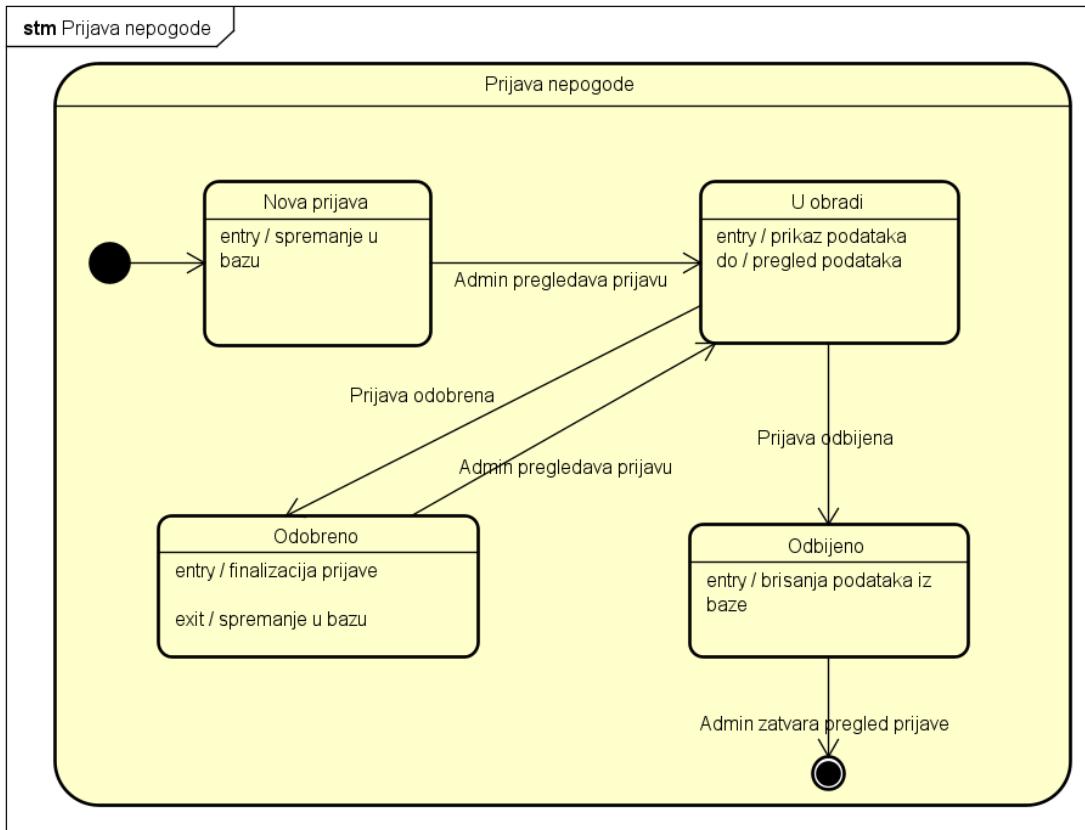
Atribut	Tip podatka	Opis varijable
id	Long	Jedinstveni identifikator županije
countyName	String	Naziv županije
region	Region (Enum)	Regija u kojoj se županija nalazi

Opis Enum klasa

Region – Prikazuje različite regije: CENTRAL, EAST, GORSKI_KOTAR, NORTH_SEASIDE, SOUTH_SEASIDE
 DisasterType – Vrste katastrofa: FLOOD, EARTHQUAKE, LANDSLIDE, HURRICANE, WILDFIRE
 NeedType – Tipovi potreba: WATER, FOOD, CAPACITY, FIRST_AID_KIT, SANDBAG, HEATER
 NotificationType – Tipovi obavijesti: PUSH, MAIL
 ReportStatus – Status izvještaja: ACCEPTED, PROCESSING, DENIED
 NeedStatus - Status potrebe: PROCESSING, SOLVED
 ResourceType - Vrste resursa: WATER, FOOD, CAPACITY, FIRST_AID_KIT, SANDBAG, HEATER
 Role - Vrste uloga: ROLE_USER, ROLE_AUTHORITY, ROLE_ADMIN, ROLE_HUMANITARIAN

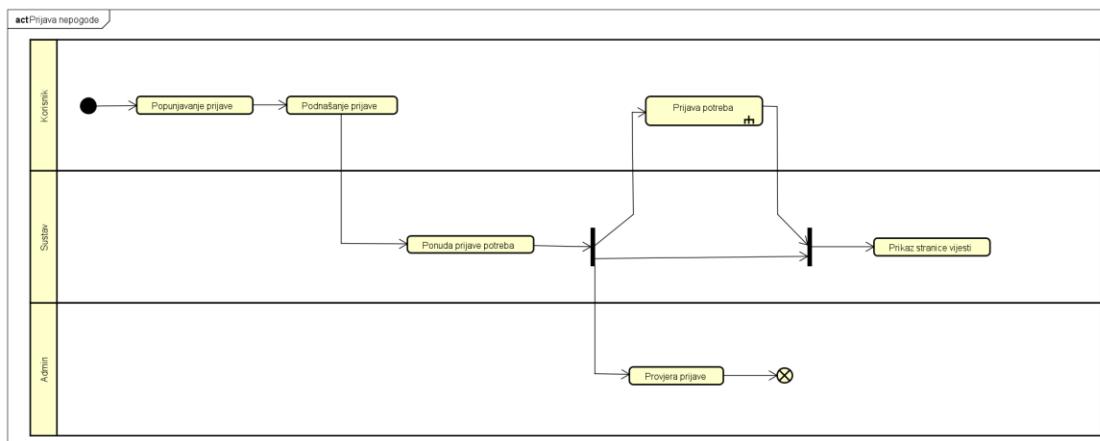
Dijagram stanja

Dijagramom je prikazano prijavljivanje nepogoda. Prijava nepogode ima 4 stanja ovisno o fazi obrade: Nova prijava, U obradi, Odobreno i Odbijeno. Prijava se odbija u slučaju neprimjerenog sadržaja ili duplicitne prijave.



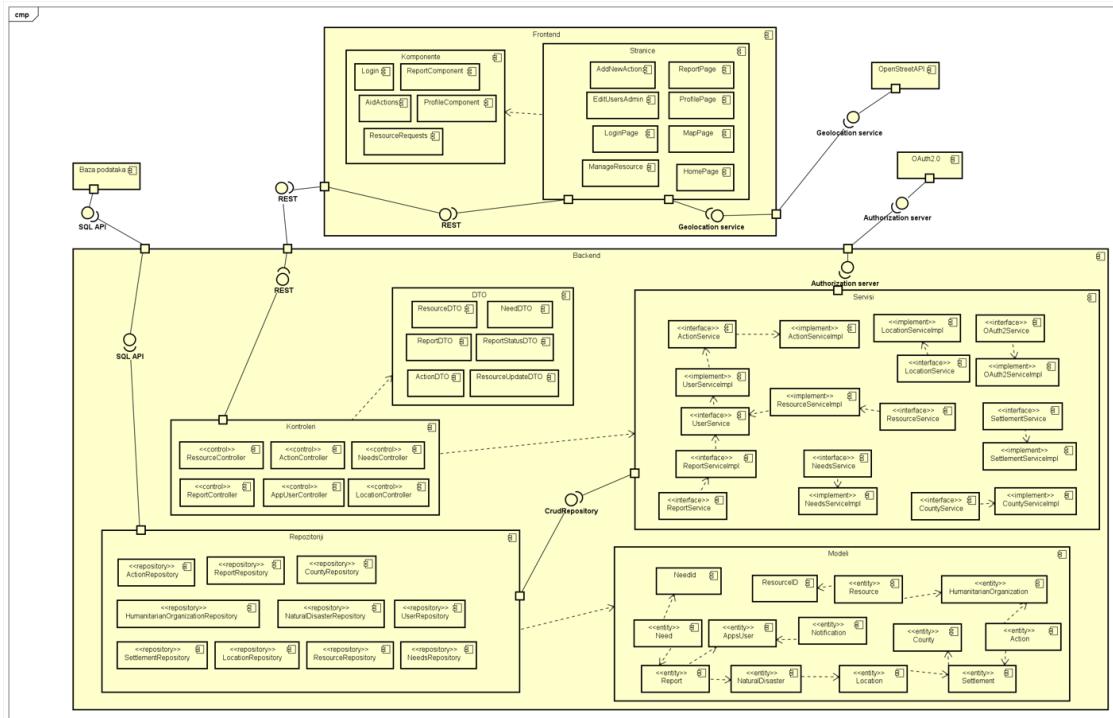
Dijagram aktivnosti

Dijagram aktivnosti prikazuje aktivnost prijave nepogode te popratnu prijavu potreba. Administrator može pregledati te po potrebi odbaciti prijavu nakon što je podnesena u sustav.



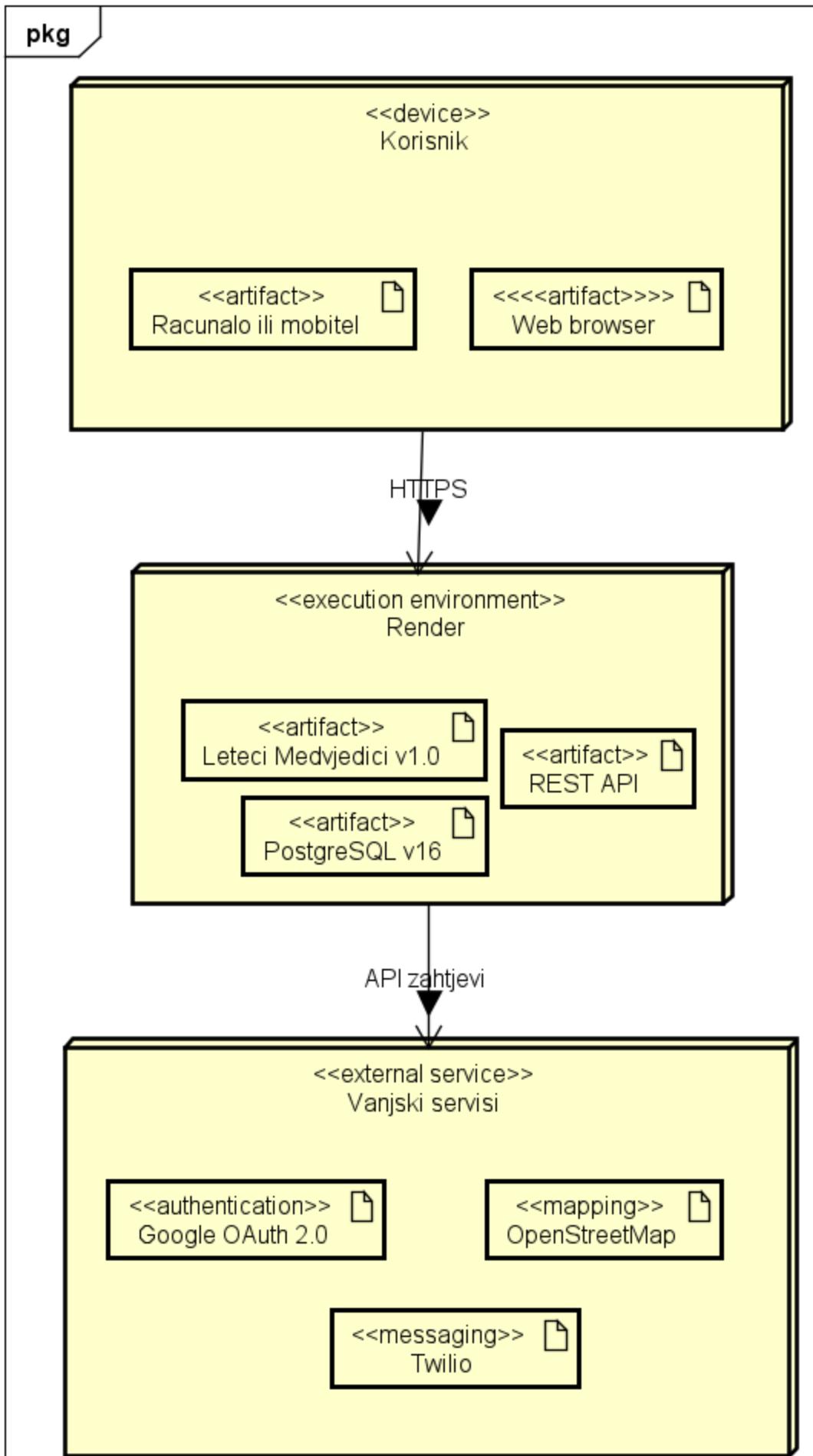
Dijagram komponenata

U nastavku je dijagram komponenata za aplikaciju SafeBear.



Dijagram razmještaja

Ovaj dijagram prikazuje razmještaj komponenata softverskog sustava u kojem korisnik putem uređaja (računala ili mobitela) ostvaruje interakciju s aplikacijom. U nastavku je specifikacijski dijagram razmještaja.



Uvod u testiranje komponente

Za ispitivanje implementiranih funkcionalnosti korištene su **JUnit 5** i **Mockito** biblioteke, koje omogućuju izolaciju i simulaciju komponenti. Testiranje je provedeno s ciljem provjere ponašanja pojedinih komponenti u različitim scenarijima, uključujući:

- Redovne slučajeve (ispravni ulazi i očekivani ishodi),
- Rubne uvjete (maksimalne i minimalne vrijednosti ulaza),
- Izazivanje pogrešaka (neispravni podaci i neočekivani ulazi),
- Poziv nepostojećih ili nedefiniranih funkcionalnosti.

Uz JUnit 5 i Mockito, korišten je **Maven** kao alat za upravljanje projektom i izvršavanje testova. Cilj testiranja bio je osigurati ispravnu funkcionalnost implementiranih metoda, uz detaljno praćenje reakcija sustava i izolaciju zavisnih komponenti kako bi se postigli precizni rezultati.

Postupak testiranja

1. Priprema testnog okruženja:

- Koristeći anotacije **@Mock** i **@InjectMocks**, simulirane su zavisnosti testirane komponente, dok su stvarni razredi testirani u izolaciji.
- Za postavljanje zajedničkih podataka i inicijalizaciju testnog okruženja korištena je metoda **@BeforeEach**, koja osigurava konzistentnost svih testova.

2. Izrada i izvršavanje testova:

- Kreirani su testni slučajevi koji pokrivaju:
 - Validaciju ulaznih podataka u redovnim slučajevima,
 - Obradu rubnih uvjeta i graničnih vrijednosti,
 - Rukovanje neispravnim unosima podataka,
 - Ponašanje sustava prilikom neočekivanih ili nedefiniranih scenarija.
- Testovi su pokrenuti pomoću **Maven test** naredbe, a rezultati su automatski zabilježeni u izvještajima, uključujući podatke o broju pokrenutih testova, vremenu izvršavanja te detalje o neuspjelim slučajevima.

3. Metodologija testiranja:

- Za svaki testni slučaj korišteni su konkretni ulazni podaci te očekivani rezultati definirani pomoću metoda **assert** iz JUnit-a.
- Ponašanje simuliranih objekata provjereno je pomoću funkcije **verify** iz Mockito biblioteke, koja osigurava da su zavisnosti testirane komponente pozvane na ispravan način.

4. Pokretanje testova u izolaciji:

- Simulacijom zavisnih komponenti osigurano je da se testovi izvršavaju bez oslanjanja na vanjske resurse, poput baze podataka ili vanjskih API-ja.
- Rezultati su prikazali uspješnost pojedinih testova, trajanje svakog testa te identificirali eventualne pogreške.

Zaključak

Provedeno testiranje omogućilo je uvid u ponašanje implementiranih funkcionalnosti u različitim scenarijima, uključujući redovne slučajeve, rubne uvjete i rukovanje greškama. Testiranje je osiguralo pouzdanost sustava i omogućilo identificiranje potencijalnih problema, čime je omogućena daljnja optimizacija implementiranih metoda.

Rezultati su zabilježeni u automatski generiranim izvještajima te su pokazali visoku pokrivenost i pouzdanost implementiranih funkcionalnosti u različitim uvjetima.

Test Cases for Validation of User Account (Registration)

Test Case ID	Test Description	Input Data	Test Steps	Expected Results	Actual Results	Status
TC_1_0_1	Validacija email-a s valjanim podacima	email: "email@example.com", imen: "John Doe"	Pozvati metodu processOAuthPostLogin sa gore navedenim podacima.	Kreira korisnika s valjanim podacima.	Korisnik uspješno kreiran.	PAS
TC_1_0_2	Validacija email-a s nevaljanim e-mailom	email: "invalid-email", imen: "John Doe"	Pozvati metodu processOAuthPostLogin sa nevaljanim e-mailom.	Metoda baca illegalArgumentException zbog nevaljanog e-mail formata.	Korisnik kreiran s nevaljanim e-mailom.	FAIL
TC_1_0_3	Validacija imena s maksimalnom dužinom znakova	email: "email@example.com", imen: 50 znakova	Pozvati metodu processOAuthPostLogin sa imenom ("A".repeat(50))	Korisnik kreiran jer je ime unutar dopuštene dužine znakova.	Korisnik uspješno kreiran.	PAS
TC_1_0_4	Validacija email-a s nevaljanim e-mailom	email: "invalid_email@"	Pozvati metodu processOAuthPostLogin	Metoda baca illegalArgumentException.	Korisnik kreiran s nevaljanim e-mailom.	FAIL

Test Case ID	Test Case Description	Input Data	Test Steps	Expected Results	Actual Results	Status
TC_1_0	Validacija imena korisnika koji je dulje od dozvoljene duljine.	"email@example.com", ime: "John Doe"	metodu processOAuthPostLogin s imenom duljim od 50 znakova.	IllegalArgumentException jer je ime dulje od dopuštene dužine.	imenom duljim od 50 znakova.	L
TC_1_0	Validacija imena korisnika koji je nepostojeci.	"unknown_user", email: "email@example.com", ime: "John Doe"	Pozvati metodu koja pokušava obraditi nepostojeće polje poput username.	Metoda baca UnsupportedOperationException za neprepoznato polje (username).	Nije bačena iznimka UnsupportedOperationException za neprepoznato polje (username).	FAI L

Provedeno je **jedinično testiranje komponente OAuth2ServiceImpl** s ciljem provjere ispravnosti funkcionalnosti obrade korisničkih podataka nakon OAuth2 autentifikacije. Testiranje je provedeno koristeći alate kao što su **JUnit 5** za pisanje i izvršavanje testova te **Mockito** za simulaciju ovisnosti, dok je upravljanje projektom i testovima omogućeno putem **Mavena**.

Cilj testiranja bio je osigurati:

- Ispravnu obradu korisničkih podataka nakon prijave putem OAuth2,
- Održavanje sustava u rubnim slučajevima, poput maksimalne duljine korisničkog imena,
- Rukovanje pogrešnim ili neispravnim podacima, poput neispravnih email adresa,
- Provjeru ponašanja sustava prilikom unosa neočekivanih podataka ili nepostojećih polja.

Postupak testiranja

Postavljanje testnog okruženja:

- Koristeći anotacije **@Mock** i **@InjectMocks**, simuliran je repozitorij UserRepository kako bi se izbjeglo oslanjanje na stvarnu bazu podataka.
- Metoda **@BeforeEach** osigurala je inicijalizaciju mock objekata i pripremu zajedničkog okruženja za sve testove.

Izrada testova:

Testovi su napisani kako bi pokrili ključne funkcionalnosti metode `processOAuthPostLogin`, uključujući:

- **Redovne slučajeve:** Obrada valjanog korisničkog imena i email adrese.
- **Rubne uvjete:** Maksimalna dopuštena duljina korisničkog imena (50 znakova).
- **Neispravne podatke:** Email adrese s pogrešnim formatom ili korisnička imena dulja od dopuštene granice.
- **Nepostojeća polja:** Simulacija obrade podataka s nepoznatim ili nepodržanim poljima (npr. `username`).

Izvršavanje testova:

- Testovi su pokrenuti pomoću **Mavena** (`mvn test`) unutar lokalnog razvojnog okruženja.
- Rezultati testiranja zabilježeni su u automatski generiranim izvještajima unutar direktorija `target/surefire-reports/`, uključujući broj uspješnih i neuspješnih testova, trajanje izvršavanja i detalje o svakom neuspjelom testu.

Verifikacija rezultata:

- Rezultati svakog testa uspoređeni su s očekivanim ishodima pomoću metoda `assert` iz JUnit-a, uključujući `assertThrows` za provjeru iznimki u rubnim i pogrešnim slučajevima.
- Simulirani repozitorij (`UserRepository`) provjeren je pomoću metode `verify` iz Mockito biblioteke, kako bi se osiguralo da su pozivi na spremanje ili dohvatanje korisnika izvršeni na očekivani način.

Zaključak

Testiranje je omogućilo detaljan uvid u ponašanje metode `processOAuthPostLogin` unutar komponente `OAuth2ServiceImpl`. Identificirani su ključni problemi, poput neispravne obrade nepostojećih polja ili neočekivanih unosa, što ukazuje na potrebu dodatnih provjera kako bi se poboljšala otpornost sustava.

Rezultati testiranja zabilježeni su u skladu s najboljim praksama jediničnog testiranja, pokrivajući redovne slučajeve, rubne uvjete i pogrešne ulaze, čime je osigurana pouzdanost implementiranih funkcionalnosti.

Ovo je JUnit testni izvještaj za testove unutar klase `OAuth2ServiceImplTest`:

```
Test set: hr.fer.progi.backend.service.impl OAuth2ServiceImplTest Tests run: 5, Failures: 3, Errors: 0, Skipped: 0, Time elapsed: 5.552 s <<< FAILURE! -- in
hr.fer.progi.backend.service.impl OAuth2ServiceImplTest
hr.fer.progi.backend.service.impl OAuth2ServiceImplTest.processOAuthPostLogin_UnknownField -- Time elapsed: 0.038 s <<< FAILURE! org.opentest4j.AssertionFailedError:
```

Expected java.lang.UnsupportedOperationException to be thrown, but nothing was thrown.

Test Cases for Validation of User Account (Log In)

Test Case	Test	Input Data	Test Steps	Expected Results	Actual Results	STATUS
CASE ID	Description					
TC_2_0_1	Redovni slučaj	email: email@example.com	1. Simulirati autentifikaciju pomoću OAuth2Authentication Token. 2. Pozvati metodu loadCurrentUser().	Korisnik uspješno prijavljen s pripadajućim podacima.	Korisnik uspješno prijavljen.	PASSED
TC_2_0_2	Nepostojeca funkcionalnost	email: "Anonimni korisnik"	1. Očistiti kontekst sigurnosti pomoću SecurityContextHolder.clearContext(). 2. Pozvati metodu loadCurrentUser().	Metoda baca WrongInputException zbog pokušaja prijave s nepostojećim korisničkim računom.	Metoda bacila WrongInputException.	PASSED
TC_2_0_3	Izazivanje pogreške	email: unknown@example.com	1. OAuth2Authentication Token s nepostojećim e-mailom. 2. Pozvati metodu loadCurrentUser().	Metoda baca WrongInputException zbog nepostojećeg korisnika.	Metoda bacila WrongInputException.	PASSED
TC_2_0_4	Rubni uvjet	email: u@x.com	1. Simulirati autentifikaciju pomoću OAuth2Authentication Token s e-mailom na granici valjanosti. 2. Pozvati metodu loadCurrentUser().	Korisnik uspješno prijavljen s pripadajućim podacima.	Korisnik uspješno prijavljen.	PASSED

Provedeno je **jedinično testiranje komponente UserServiceImpl** s ciljem provjere ispravnosti funkcionalnosti učitavanja trenutnog korisnika iz OAuth2 autentifikacije i baze podataka. Testiranje je provedeno koristeći **JUnit 5** za pisanje i izvršavanje testova te **Mockito** za simulaciju ovisnosti, dok je upravljanje projektom i testovima omogućeno putem **Mavena**.

Cilj testiranja bio je osigurati:

- Ispravnu obradu korisničkih podataka dobivenih putem OAuth2 autentifikacije,
- Održavanje sustava u rubnim slučajevima, poput krajnjih granica ulaznih podataka (npr. kratke email adrese),
- Rukovanje pogrešnim ili neispravnim podacima, poput anonimnog korisnika ili korisnika koji nije pronađen u bazi.

Postupak testiranja

Postavljanje testnog okruženja:

- Koristeći anotacije **@Mock** i **@InjectMocks**, simuliran je repozitorij UserRepository kako bi se izbjeglo oslanjanje na stvarnu bazu podataka.
- Metoda **@BeforeEach** osigurala je inicijalizaciju mock objekata i pripremu zajedničkog okruženja za sve testove.

Izrada testova:

Testovi su napisani kako bi pokrili ključne funkcionalnosti metode `loadCurrentUser`, uključujući:

- **Redovne slučajeve:** Obrada valjanih korisničkih podataka dobivenih putem OAuth2 autentifikacije.
- **Rubne uvjete:** Učitavanje korisnika s rubnim vrijednostima podataka, poput kratkog formata email adrese.
- **Neispravne podatke:** Simulacija situacija poput anonimnog korisnika ili email adrese koja ne postoji u bazi.
- **Korisnici koji nisu pronađeni:** Testiranje ponašanja sustava kada autentifikacija postoji, ali korisnik ne postoji u bazi podataka.

Izvršavanje testova:

- Testovi su pokrenuti pomoću **Mavena** (`mvn test`) unutar lokalnog razvojnog okruženja.
- Svi testovi su uspješno prošli, što je prikazano u izvještaju
- Rezultati testiranja zabilježeni su u automatski generiranim izvještajima unutar direktorija `target/surefire-reports/`.

Verifikacija rezultata:

- Rezultati svakog testa uspoređeni su s očekivanim ishodima pomoću metoda assert iz JUnit-a, uključujući assertThrows za provjeru iznimki u slučaju anonimnih korisnika i nepostojećih zapisa u bazi.
- Simulirani repozitorij (UserRepository) provjeren je pomoću metode verify iz Mockito biblioteke, kako bi se osiguralo da su pozivi na dohvata podataka iz baze izvršeni na očekivani način.

Zaključak

Testiranje je omogućilo detaljan uvid u ponašanje metode loadCurrentUser unutar komponente UserServiceImpl. Svi testovi su uspješno prošli, što pokazuje da implementacija ispravno obrađuje valjane podatke, rukuje rubnim uvjetima i ispravno detektira neispravne unose ili anonimne korisnike.

Rezultati testiranja zabilježeni su u skladu s najboljim praksama jediničnog testiranja, osiguravajući pouzdanost implementiranih funkcionalnosti u različitim scenarijima.

Ovo je JUnit testni izvještaj za testove unutar klase UserServiceImplTest:

```
Test set: hr.fer.progi.backend.service.impl.UserServiceImplTest Tests run: 4, Failures: 0,
Errors: 0, Skipped: 0, Time elapsed: 6.001 s -- in
hr.fer.progi.backend.service.impl.UserServiceImplTest
```

Test Cases - input short description

Test Case ID	Test Case Description	Input Data	Steps to Execute	Expected Output	Actual Output	Status
TC_3_01	Redovni slučaj: Valjan kratak opis	settlementName: "Zagreb", disasterType: "FLOOD", shortDescription: "Poplava u Zagrebu", photo: "photo.jpg"	Pozvati metodu newRepo s ispravnim disasterType: "FLOOD", shortDescription: "Poplava u Zagrebu", photo: "photo.jpg"	Kreiran Report objekt s ispravnim podacima.	Report uspješno kreiran s odgovarajućim podacima.	PAS
TC_3_02	Rubni uvjet: Maksimalna duljina kratkog opisa	settlementName: "Split", disasterType: "HURRICANE", shortDescription: "duljine"	Pozvati metodu newRepo s rubnim uvjetom duljine opisa	Kreiran Report objekt jer je duljina unosa valjana.	Report uspješno kreiran s maksimalnom duljinom	PAS

Test						
Case ID	Test Case Description	Input Data	Steps to Execute	Expected Output	Actual Output	Status
TC_3_03	Izazivanje pogreške: Nedostaje kratak opis	"A".repeat(255), photo: "photo.jpg"	Pozvati metodu newRepo srt bez kratkog opisa.	Kreiran Report objekt s null kratkim opisom (trenutna implementacija ne baca iznimku).	opisa. Report kreiran s null vrijednoć u za opis.	PAS
TC_3_04	Nepostojeća funkcionalnost: Neobičan unos	settlementName: "Rijeka", disasterType: EARTHQUAKE, shortDescription: "A", photo: "photo.jpg"	Pozvati metodu newRepo srt s neobičnim unosom.	Kreiran Report objekt s neobičnim im unosom u kratkom opisu.	Report uspješno kreiran s posebnim znakovima .	PAS
TC_3_05	Rubni uvjet: Minimalna duljina kratkog opisa	settlementName: "Zadar", disasterType: FLOOD, shortDescription: "A", photo: "photo.jpg"	Pozvati metodu newRepo srt s opisom duljine 1 znak.	Kreiran Report objekt jer je duljina unosa valjana.	Report uspješno kreiran s minimalnom duljinom opisa.	PAS
TC_3_06	Rubni uvijet: Duljina kratkog opisa veća od maksimalne(za 1)	settlementName: "Šibenik", disasterType: WILDFIRE, shortDescription: "A".repeat(256), photo: "	Pozvati metodu newRepo srt s opisom duljine 256 znakova .	Bacanje iznimke IllegalArgumentException jer duljina prelazi maksimalnu dopuštenu granicu.	Nije bačena nikakva iznimka, Report kreiran bez greške.	FAIL

Test Case ID	Test Case Description	Input Data	Steps to Execute	Expected Output	Actual Output	Status
		"photo.jpg"				

Provedeno je **jedinično testiranje komponente ReportServiceImpl** s ciljem provjere ispravnosti funkcionalnosti prijavljivanja prirodnih nepogoda i povezivanja s relevantnim entitetima (korisnicima, naseljima i vrstama nepogoda). Testiranje je provedeno koristeći **JUnit 5** za pisanje i izvršavanje testova te **Mockito** za simulaciju ovisnosti, dok je upravljanje projektom i testovima omogućeno putem **Mavena**.

Cilj testiranja bio je osigurati:

- Ispravno kreiranje novih izvještaja o prirodnim nepogodama s valjanim podacima,
- Održavanje sustava u rubnim slučajevima, poput maksimalne duljine opisa nepogode,
- Rukovanje nedostajućim podacima, poput izostavljenog kratkog opisa,
- Provjeru ponašanja sustava prilikom unosa neobičnih ili neočekivanih vrijednosti.

Postupak testiranja

Postavljanje testnog okruženja:

- Koristeći anotacije **@Mock** i **@InjectMocks**, simulirani su repozitoriji **ReportRepository**, **SettlementRepository** i **NaturalDisasterRepository**, kao i ovisnosti o servisima **UserServiceImpl** i **OAuth2Service**, kako bi se izbjeglo oslanjanje na stvarne resurse.
- Dodatno je, pomoću refleksije, ručno postavljena inicijalizacija ovisnosti unutar **ReportServiceImpl**, kako bi se omogućila precizna kontrola nad ponašanjem simuliranih komponenti.
- Metoda **@BeforeEach** osigurala je inicijalizaciju mock objekata i pripremu zajedničkog okruženja za sve testove.

Izrada testova:

Testovi su napisani kako bi pokrili ključne funkcionalnosti metode **newReport**, uključujući:

- **Redovne slučajeve:** Kreiranje izvještaja s valjanim opisima, tipovima nepogoda i naseljima.
- **Rubne uvjete:** Provjera ponašanja sustava pri unosu maksimalne dopuštene duljine opisa (255 znakova).
- **Nedostajući podaci:** Rukovanje situacijama gdje je kratak opis nepogode izostavljen.
- **Neobični unosi:** Obrada neuobičajenih znakova u opisu nepogode, poput emojija i specijalnih znakova.

Izvršavanje testova:

- Testovi su pokrenuti pomoću **Mavena** (`mvn test`) unutar lokalnog razvojnog okruženja.
- 5/6 testova je prošlo, što je prikazano u izvještaju
- Rezultati testiranja zabilježeni su u automatski generiranim izvještajima unutar direktorija `target/surefire-reports/`.

Verifikacija rezultata:

- Rezultati svakog testa uspoređeni su s očekivanim ishodima pomoću metoda `assert` iz JUnit-a.
- Simulirani repozitoriji i servisi provjereni su pomoću metode `verify` iz Mockito biblioteke kako bi se osiguralo da su svi pozivi na spremanje i dohvata podataka izvršeni ispravno.

Zaključak

Testiranje je omogućilo detaljan uvid u ponašanje metode `newReport` unutar komponente `ReportServiceImpl`. Skoro svi testovi su uspješno prošli, što pokazuje da implementacija ispravno obrađuje valjane podatke, rukuje rubnim uvjetima i ispravno detektira nedostajuće ili neobične podatke.

Rezultati testiranja zabilježeni su u skladu s najboljim praksama jediničnog testiranja, pokrivajući redovne slučajeve, rubne uvjete i neispravne ulaze, čime je osigurana pouzdanost implementiranih funkcionalnosti.

Ovo je JUnit testni izvještaj za testove unutar klase `ReportServiceImplTest`:

```
Test set: hr.fer.progi.backend.service.impl.ReportServiceImplTest Tests run: 6, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 6.442 s <<< FAILURE! -- in
hr.fer.progi.backend.service.impl.ReportServiceImplTest
hr.fer.progi.backend.service.impl.ReportServiceImplTest.newReport_ExceedsMaxShortDescriptionLength -- Time elapsed: 0.059 s <<< FAILURE!
org.opentest4j.AssertionFailedError: Expected java.lang.IllegalArgumentException to be thrown, but nothing was thrown.
```

Test Cases - Location

Test Case ID	Test Case Description	Input Data	Steps to Execute	Expected Output	Actual Output	Status
TC_4_01	Redovni slučaj	settlementName: "Zagreb", disasterType: POPLAVA	Pozvati metodu newReport s valjanom	Report uspješno kreiran.	Report kreiran sa lokacijom: Zagreb	PASSED

Test Case ID	Test Case Description	Input Data	Steps to Execute	Expected Output	Actual Output	Status
TC_4_02	Izazivanje pogreške	settlementName: "Nepoznat", disasterType: POPLAVA	Pozvati metodu newReport s nevaljonom lokacijom.	IllegalArgumentException je bačen.	Nije bačena iznimka, lokacija nije pronađena.	FAIL
TC_4_03	Izazivanje pogreške	settlementName: "", disasterType: POPLAVA	Pozvati metodu newReport s praznom lokacijom.	IllegalArgumentException je bačen.	Nije bačena iznimka, naziv lokacije je prazan.	FAIL
TC_4_04	Nepostojeća funkcionalnost	settlementName: "    ", disasterType: POPLAVA	Pozvati metodu newReport s neobičnom lokacijom.	Report uspješno kreiran.	Report kreiran sa lokacijom:   	PAS

Provedeno je **jedinično testiranje komponente ReportLocationServiceImpl** s ciljem provjere ispravnosti funkcionalnosti prijavljivanja lokacija povezanih s izveštajima o prirodnim nepogodama. Testiranje je provedeno koristeći **JUnit 5** za pisanje i izvršavanje testova te **Mockito** za simulaciju ovisnosti, dok je upravljanje projektom i testovima omogućeno putem **Mavena**.

Cilj testiranja bio je osigurati:

- Ispravno povezivanje izvještaja o nepogodama s valjanim lokacijama,
- Održavanje sustava u rubnim slučajevima, poput neuobičajenih naziva lokacija,
- Rukovanje pogrešnim ili neispravnim podacima, poput nepostojećih lokacija,
- Provjeru ponašanja sustava prilikom unosa praznih naziva lokacija.

Postupak testiranja

Postavljanje testnog okruženja:

- Koristeći anotacije **@Mock** i **@InjectMocks**, simulirani su repozitoriji ReportRepository, SettlementRepository i NaturalDisasterRepository, kao i ovisnosti o servisima UserService i OAuth2Service, kako bi se izbjeglo oslanjanje na stvarne resurse.
- Ručno su inicializirane poljske ovisnosti unutar ReportServiceImpl koristeći refleksiju kako bi se osigurala potpuna kontrola nad mock objektima.
- Metoda **@BeforeEach** osigurala je pripremu zajedničkog okruženja za sve testove.

Izrada testova:

Testovi su napisani kako bi pokrili ključne funkcionalnosti metode newReport, uključujući:

- **Redovne slučajeve:** Dodavanje izvještaja s valjanim nazivima lokacija.
- **Rubne uvjete:** Provjera ponašanja sustava prilikom unosa neuobičajenih naziva lokacija (npr. emoji).
- **Neispravne podatke:** Rukovanje unosima nepostojećih lokacija, koje sustav ne može pronaći u bazi.
- **Prazne vrijednosti:** Validacija da se naziv lokacije ne može ostaviti prazan.

Izvršavanje testova:

- Testovi su pokrenuti pomoću **Mavena** (`mvn test`) unutar lokalnog razvojnog okruženja.
- Dva testa nisu prošla, zbog nepravilne validacije podataka, što je prikazano u izvještaju
- Rezultati testiranja zabilježeni su u automatski generiranim izvještajima unutar direktorija `target/surefire-reports/`.

Verifikacija rezultata:

- Rezultati svakog testa uspoređeni su s očekivanim ishodima pomoću metoda `assert` iz JUnit-a, uključujući `assertThrows` za provjeru iznimki u slučaju neispravnih ili praznih lokacija.
- Simulirani repozitoriji i servisi provjereni su pomoću metode `verify` iz Mockito biblioteke kako bi se osiguralo da su pozivi na spremanje i dohvata podataka izvršeni prema očekivanjima.

Zaključak

Testiranje je omogućilo detaljan uvid u ponašanje metode newReport unutar komponente ReportLocationServiceImpl. Dva testa nisu prošla, što ukazuje na problem s validacijom

unosa neispravnih lokacija (poput nepostojećih ili praznih naziva). Identificirani problemi sugeriraju potrebu za dodatnim provjerama kako bi se poboljšala otpornost sustava.

Rezultati testiranja zabilježeni su u skladu s najboljim praksama jediničnog testiranja, pokrivajući redovne slučajeve, rubne uvjete i neispravne ulaze, čime je osigurana pouzdanost implementiranih funkcionalnosti u većini scenarija.

Ovo je JUnit testni izvještaj za testove unutar klase ReportLocationServiceImplTest:

```
Test set: hr.fer.progi.backend.service.impl.ReportLocationServiceImplTest Tests run: 4,
Failures: 2, Errors: 0, Skipped: 0, Time elapsed: 5.188 s <<< FAILURE! -- in
hr.fer.progi.backend.service.impl.ReportLocationServiceImplTest
hr.fer.progi.backend.service.impl.ReportLocationServiceImplTest.newReport_InvalidLoca
tion -- Time elapsed: 0.047 s <<< FAILURE! org.opentest4j.AssertionFailedError: Expected
java.lang.IllegalArgumentException to be thrown, but nothing was thrown.
```

Test cases - coordinates

Test Case ID	Test Case Description	Input Data	Test Steps	Expected Results	Actual Results	STAT US
TC_5_01	Redovni slučaj: Valjani unos koordinata	45.8150,15.9 819	Pozvati metodu addLocations s koordinatama '45.8150,15.98 19'.	Kreirana lokacija s odgovarajućim koordinatama.	Lokacija uspješno kreirana.	PASS
TC_5_02	Rubni uvjet: Maksimalne i minimalne koordinate	180.0,90.0; -180.0,-90.0	Pozvati metodu addLocations s rubnim koordinatama.	Lokacija uspješno kreirana za rubne vrijednosti.	Lokacija uspješno kreirana za rubne vrijednosti.	PASS
TC_5_03	Izazivanje pogreške: Negativne vrijednosti izvan granica	-200.0,91.0	Pozvati metodu addLocations s koordinatama '-200.0,91.0'.	Lokacija spremljena sa i vraćene nepravne koordinatama.	Lokacija spremljena sa nepravnim koordinatama.	FAIL
TC_5_04	Nepostojeća funkcionalnost: String	invalid_coordinates	Pozvati metodu addLocations s vrijednošću 'invalid_coordi	Lokacija spremljena sa tekstualni	Lokacija spremljena sa vrijednošću 'invalid_coordi	FAIL

Test Case ID	Test Case Description	Input Data	Test Steps	Expected Results	Actual Results	STAT US
	umjesto koordinata		nates':.	unos.	nates'.	
TC_5_05	Rubni uvijet(max + 1): Vrijednosti iznad granica	181.0,91.0	Pozvati metodu addLocations koordinatama '181.0,91.0'.	Lokacija nije spremlijen a; bacanje greške zbog neispravnih vrijednosti.	Lokacija uspješno spremljena s neispravnim vrijednostima.	FAIL
TC_5_06	Rubni uvijet(min - 1): Vrijednosti ispod granica	-181.0,-91.0	Pozvati metodu addLocations koordinatama '-181.0,-91.0'.	Lokacija nije spremlijen a; bacanje greške zbog neispravnih vrijednosti.	Lokacija uspješno spremljena s neispravnim vrijednostima.	FAIL

Provedeno je **jedinično testiranje komponente** LocationServiceImpl s ciljem provjere ispravnosti funkcionalnosti dodavanja lokacija u sustav. Testiranje je provedeno koristeći alate kao što su **JUnit 5** za pisanje i izvršavanje testova te **Mockito** za simulaciju ovisnosti, dok je upravljanje projektom i testovima omogućeno putem **Mavena**.

Cilj testiranja bio je osigurati:

- Ispravnost spremanja valjanih geografskih koordinata.
- Održavanje sustava u rubnim slučajevima, poput granica dozvoljenih vrijednosti koordinata.
- Provjeru ponašanja sustava prilikom unosa neispravnih podataka.

Postupak testiranja

1. Postavljanje testnog okruženja:

- Koristeći **Mockito**, simuliran je repozitorij LocationRepository kako bi se ovisnost o bazi podataka zamijenila mock objektima.

- Za inicijalizaciju testnih podataka korišten je @BeforeEach, kojim su definirani zajednički objekti za sve testove.

2. Izrada testova:

- Testovi su napisani za ključne funkcionalnosti metode addLocation, uključujući:
 - Validaciju ispravnih koordinata.
 - Obradu rubnih uvjeta (maksimalne i minimalne vrijednosti koordinata).
 - Rukovanje neispravnim ili nelogičnim unosima podataka.

3. Izvršavanje testova:

- Testovi su pokrenuti pomoću **Mavena** (mvn test) unutar lokalnog razvojnog okruženja.
- Rezultati testiranja zabilježeni su u automatski generiranim izvještajima koji prikazuju broj pokrenutih testova, vrijeme izvođenja, te uspješne i neuspješne testove.

4. Verifikacija rezultata:

- Rezultati svakog testa uspoređeni su s očekivanim ishodima pomoću assert metoda iz JUnit-a.
- Simulacije repozitorija su provjerene pomoću verify funkcija iz Mockito-a kako bi se osiguralo ispravno ponašanje mock objekata.

Zaključak

Testiranje je osiguralo uvid u ispravnost implementacije metode addLocation unutar komponente LocationServiceImpl. Identificirani su slučajevi koji zahtijevaju dodatne provjere za neispravne unoše (npr. negativne koordinate) kako bi se poboljšala otpornost sustava. Rezultati testiranja dokumentirani su u skladu s najboljim praksama jediničnog testiranja.

Ovo je JUnit testni izvještaj za testove unutar klase LocationServiceImplTest:

```
Test set: hr.fer.progi.backend.service.impl.LocationServiceImplTest Tests run: 6, Failures: 4, Errors: 0, Skipped: 0, Time elapsed: 5.953 s <<< FAILURE! -- in
hr.fer.progi.backend.service.impl.LocationServiceImplTest
hr.fer.progi.backend.service.impl.LocationServiceImplTest.addLocation_InvalidCoordinat
es -- Time elapsed: 0.043 s <<< FAILURE! org.opentest4j.AssertionFailedError: expected:
not
```

Test cases for quantity of resources

Test Case ID	Test Case Description	Input Data	Test Steps	Expected Results	Actual Results	STATUS
--------------	-----------------------	------------	------------	------------------	----------------	--------

Test Case ID	Test Case Description	Input Data	Test Steps	Expected Results	Actual Results	STATUS
TC_6_01	Redovni slučaj	Količina: 20	Klik na unos, upiši količinu 20, potvrди.	Sustav sprema unos količine 20.	Sustav je spremio količinu 20.	PASS
TC_6_02	Nepostojeća funkcionalnost	Ne postoji resurs	Odabir resursa koji ne postoji, potvrди.	Sustav vraća grešku 'Resource not found'.	Prikazana greška: 'Resource not found'.	PASS
TC_6_03	Izazivanje pogreške	Količina: -10	Klik na unos, upiši količinu -10, potvrди.	Sustav vraća grešku 'Quantity cannot be negative'.	Prikazana greška: 'Quantity cannot be negative'.	PASS
TC_6_04	Rubni uvjet	Količina: 0	Klik na unos, upiši količinu 0, potvrdi.	Sustav sprema unos količine 0.	Sustav je spremio količinu 0.	PASS

Provedeno je **jedinično testiranje komponente ResourceServiceImpl** s ciljem provjere ispravnosti funkcionalnosti ažuriranja resursa u humanitarnim organizacijama. Testiranje je provedeno koristeći **JUnit 5** za pisanje i izvršavanje testova te **Mockito** za simulaciju ovisnosti, dok je upravljanje projektom i testovima omogućeno putem **Mavena**.

Cilj testiranja bio je osigurati:

- Ispravno ažuriranje količina resursa s valjanim unosima,
- Održavanje sustava u rubnim slučajevima, poput nulte količine resursa,
- Rukovanje pogrešnim podacima, poput negativne količine,
- Provjeru ponašanja sustava prilikom unosa nepostojećih resursa.

Postupak testiranja

Postavljanje testnog okruženja:

- Koristeći anotacije **@Mock** i **@InjectMocks**, simulirani su repozitoriji **ResourceRepository** i **HumanitarianOrganizationRepository**, kao i ovisnost o servisu **UserService**, kako bi se izbjeglo oslanjanje na stvarne resurse.

- Metoda **@BeforeEach** osigurala je pripremu zajedničkih objekata i mock podataka, uključujući mock objekt Resource s pridruženim ID-em i pripadajućom humanitarnom organizacijom.

Izrada testova:

Testovi su napisani kako bi pokrili ključne funkcionalnosti metode updateResource, uključujući:

- **Redovne slučajeve:** Ažuriranje resursa s valjanom količinom.
- **Rubne uvjete:** Provjera ponašanja sustava kada je količina resursa jednaka nuli.
- **Neispravne podatke:** Validacija da negativne količine nisu dozvoljene.
- **Nepostojeće resurse:** Simulacija situacija u kojima resurs nije pronađen u bazi podataka.

Izvršavanje testova:

- Testovi su pokrenuti pomoću **Mavena** (`mvn test`) unutar lokalnog razvojnog okruženja.
- Svi testovi su uspješno prošli, što je prikazano u izvještaju
- Rezultati testiranja zabilježeni su u automatski generiranim izvještajima unutar direktorija `target/surefire-reports/`.

Verifikacija rezultata:

- Rezultati svakog testa uspoređeni su s očekivanim ishodima pomoću metoda `assert` iz JUnit-a, uključujući `assertThrows` za provjeru iznimki u slučaju neispravnih unosa ili nepostojećih resursa.
- Simulirani repozitoriji provjereni su pomoću metode `verify` iz Mockito biblioteke kako bi se osiguralo da su pozivi na spremanje i dohvati podataka izvršeni prema očekivanjima.

Zaključak

Testiranje je omogućilo detaljan uvid u ponašanje metode updateResource unutar komponente `ResourceServiceImpl`. Svi testovi su uspješno prošli, što potvrđuje da implementacija ispravno rukuje redovnim unosima, rubnim uvjetima i neispravnim podacima. Sustav se ispravno ponaša u slučajevima kada resurs ne postoji ili kada se unese negativna količina, pružajući odgovarajuće iznimke.

Rezultati testiranja zabilježeni su u skladu s najboljim praksama jediničnog testiranja, pokrivajući redovne slučajeve, rubne uvjete i pogrešne ulaze, čime je osigurana pouzdanost implementiranih funkcionalnosti.

Ovo je JUnit testni izvještaj za testove unutar klase `ResourceServiceImplTest`:

Test set: hr.fer.progi.backend.service.impl.ResourceServiceImplTest Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 7.138 s -- in
hr.fer.progi.backend.service.impl.ResourceServiceImplTest

Ukupno testova : 29 , Ukupno otkrivenih grešaka: 10

Ispitivanje sustava

Za ispitivanje sustava korišten je alat Selenium IDE, koji omogućuje snimanje i automatizaciju korisničkih akcija unutar preglednika kako bi se testiralo ponašanje cijelog sustava u stvarnim uvjetima korištenja. Cilj testiranja bio je provjeriti interakciju svih komponenti sustava te identificirati potencijalne probleme, posebice u redovnim slučajevima, rubnim uvjetima i situacijama koje uključuju pogreške ili poziv nepostojećih funkcionalnosti.

Testni slučajevi uključivali su unos valjanih i nevaljanih podataka, simulaciju korisničkih akcija poput prijave, podnošenja izvještaja i prijave potreba te validaciju očekivanih rezultata. Tijekom testiranja sustav je odgovarao kako je predviđeno, uz evidentirane iznimke za nevalidne unose. Rezultati su dokumentirani s pripadajućim ulaznim podacima, očekivanim i stvarnim ishodima te statusom prolaska svakog slučaja.

Test Cases - Ispitivanje sustava

Test				
Case ID	Test Case Description	Test Steps	Expected Outcome	Actual Outcome
TC_01	Valid Emergency Report Submission	1. Pokrenuti aplikaciju. 2. Kliknuti na "Log in" – unijeti korisničko ime i lozinku. 3. Kliknuti na "Report" i ispuniti formu sa valjanim podatcima (vrsta nepogode, mjesto, kratak opis). 4. Kliknuti Submit.	Submit report uspješno pokrenut. Na novom ekranu poruka: "Report Submitted Successfully!"	Submit report uspješno pokrenut. Na novom ekranu poruka: "Report Submitted Successfully!"
TC_02	Invalid Emergency Report Submission (Missing Location)	1. Pokrenuti aplikaciju. 2. Kliknuti na "Log in" – unijeti korisničko ime i lozinku. 3. Kliknuti na "Report" i ispuniti formu sa nepotpunim	Submit report nije uspio. Na ekranu poruka: "Failed to submit report!".	Submit report nije uspio. Na ekranu poruka: "Failed to submit report!".

Test Case ID	Test Case Description	Test Steps	Expected Outcome	Actual Outcome
TC_03	Invalid Emergency Report Submission (Invalid Location)	<p>podatcima (vrsta nepogode, kratak opis).</p> <p>4. Kliknuti Submit.</p> <p>1. Pokrenuti aplikaciju.</p> <p>2. Kliknuti na “Log in” – unijeti korisničko ime i lozinku.</p> <p>3. Kliknuti na “Report” i ispuniti formu sa krivim podatcima (vrsta nepogode, krivo mjesto, kratak opis).</p> <p>4. Kliknuti Submit.</p>	<p>Submit report nije uspio. Na ekranu imamo poruku: “Please enter a valid location from the list”.</p>	Submit report nije uspio. Na ekranu imamo poruku: “Please enter a valid location from the list”.
TC_04	Edge Case for Reporting a Need (1)	<p>1. Pokrenuti aplikaciju.</p> <p>2. Kliknuti na “Log in” – unijeti korisničko ime i lozinku.</p> <p>3. Kliknuti na “Report” i ispuniti formu sa krivim podatcima (vrsta nepogode, krivo mjesto, kratak opis).</p> <p>4. Kliknuti Submit.</p> <p>5. Kliknuti na “Report a need”.</p> <p>6. Odabratи hranu i kliknuti na količinu 1.</p>	<p>Uspješan submit report, uspješno prijavljivanje potrebe. Na ekranu poruka: Needs submitted successfully!</p>	Uspješan submit report, uspješno prijavljivanje potrebe. Na ekranu poruka: Needs submitted successfully!
TC_05	Invalid Input for Reporting a Need (-1)	<p>1. Pokrenuti aplikaciju.</p> <p>2. Kliknuti na “Log in” – unijeti korisničko ime i lozinku.</p> <p>3. Kliknuti na “Report” i ispuniti formu sa krivim podatcima (vrsta nepogode, krivo mjesto, kratak opis).</p> <p>4. Kliknuti Submit.</p> <p>5. Kliknuti na “Report a need”.</p> <p>6. Odabratи hranu i kliknuti na količinu -1.</p>	<p>Uspješan submit report, ali neuspješno prijavljivanje potrebe. Na ekranu poruka: Please enter valid number!</p>	Uspješan submit report, ali neuspješno prijavljivanje potrebe. Na ekranu poruka: Needs submitted successfully!

Ukupno testova: 5, Otkriveno pogrešaka: 1

Selenium test case - izvještaji(log Selenium testa)

TC_01

Running 'TC_01' 22:21:43 1.open on / OK 22:21:45 2.setWindowSize on 1550x830 OK
22:21:46 3.click on css=.google-login OK 22:21:46 4.click on css=.report-button OK
22:21:49 5.click on css=.iconContainerStyle:nth-child(3) path OK 22:21:57 6.click on
css=.address-input OK 22:22:06 7.type on css=.address-input with value Split OK 22:22:06
8.sendKeys on css=.address-input with value \${KEY_ENTER} OK 22:22:06 9.click on
css=.address-input OK 22:22:07 10.type on css=.address-input with value Sisak OK
22:22:07 11.click on css=.inputField OK 22:22:07 12.type on css=.inputField with value u
sisku pada kisaaaaaa OK 22:22:08 13.click on css=.submit-button OK 22:22:08 14.click on
css=.go-home-button OK

TC_02

Running 'TC_02' 22:32:16 1.open on / OK 22:32:17 2.setWindowSize on 896x692 OK
22:32:18 3.click on css=.google-login OK 22:32:18 4.click on css=.report-button OK
22:32:20 5.click on css=.iconContainerStyle:nth-child(5) > .paragraphStyle OK 22:32:23
6.click on css=.inputField OK 22:32:34 7.click on css=.inputField OK 22:32:35 8.type on
css=.inputField with value UNOSIMO BEZ LOKACIJE HA HA HA HA(im crashout) OK
22:32:35 9.click on css=.submit-button OK

TC_03

Running 'TC_03' 22:39:27 1.open on / OK 22:39:28 2.setWindowSize on 896x692 OK
22:39:28 3.click on css=.google-login OK 22:39:30 4.click on css=.report-button OK
22:39:32 5.click on css=.iconContainerStyle:nth-child(2) > svg OK 22:39:37 6.click on
css=.inputField OK 22:39:51 7.type on css=.inputField with value POTRESSSČINAAA
AAAAAAAAAAAA OK 22:39:52 8.click on css=.address-input OK 22:39:53 9.type on
css=.address-input with value NE ZNAM DI SAMMMOK 22:39:53 10.click on
css=.submit-button OK 22:39:53 11.assertAlert on Please enter a valid location from the
list. OK 22:39:53 12.close OK

TC_04

Running 'TC_04' 22:47:27 1.open on / OK 22:47:28 2.setWindowSize on 1550x830 OK
22:47:29 3.click on css=.google-login OK 22:47:30 4.click on css=.report-button OK
22:47:32 5.click on css=.iconContainerStyle:nth-child(1) > svg OK 22:47:37 6.click on
css=.address-input OK 22:47:45 7.type on css=.address-input with value Sisak OK
22:47:45 8.click on css=.inputField OK 22:47:46 9.type on css=.inputField with value bla
bla bla gladan sam OK 22:47:46 10.click on css=.submit-button OK 22:47:46 11.click on
css=.report-need-button OK

TC_05

Running 'TC_05' 22:53:40 1.open on / OK 22:53:41 2.setSize on 1550x830 OK
22:53:42 3.click on css=.google-login OK 22:53:42 4.click on css=.report-button OK
22:53:47 5.click on css=.address-input OK 22:54:07 6.click on
css=.iconContainerStyle:nth-child(3) OK 22:54:19 7.click on css=.address-input OK
22:54:19 8.type on css=.address-input with value Split OK 22:54:20 9.click on
css=.inputField OK 22:54:21 10.click on css=.inputField OK 22:54:22 11.type on
css=.inputField with value u sumarku skrivenom medvjeda je mali dommm, oni cak i letee
pogledajeeee, neki ljudi okrutni , opasni ko zmajjj; slistili bi taaaajjj: praviii maliii raaajjj:
NOOO NEĆEEEE , NEĆEEE BOME MOĆ, JER MALII MEDVJEDI ĆE DOOOĆĆ OK 22:54:22
12.click on css=.submit-button OK 22:54:29 13.click on css=.report-need-button OK
22:54:29 14.click on css=.need-item:nth-child(2) > .need-button OK 22:54:31 15.type on
css=.quantity-input with value -1 OK 22:54:31 16.click on css=.quantity-input OK 22:54:32
17.click on css=.submit-button OK

Korištene tehnologije i alati

1. Programski jezici
 - Java 21
 - JavaScript ES6
2. Radni okviri i biblioteke
 - React 18.3.1
 - Spring Boot 3.3.5
3. Baza podataka
 - PostgreSQL 16
4. Razvojni alati
 - IntelliJ IDEA
 - Eclipse
 - VS Code
 - Astah UML
 - Figma
 - Postman
5. Alati za ispitivanje
 - Selenium
6. Cloud platforma
 - Google Cloud (autentifikacija)
7. Alati za razmještaj

- Docker
- Nginx
- Render

React je open-source biblioteka JavaScripta koja se koristi za dizajniranje korisničkog sučelja. Njegove prednosti uključuju:

- virtualni DOM koji omogućuje da se stranica samo jednom prikaže bez obzira na to što korisnik mijenja na stranici. U web razvoju koji koristi klasični DOM nakon svake korisnikove promjene čitava se stranica mora ponovno prikazati kako bi odgovarala učinjenoj promjeni što može usporavati aplikaciju i troškove učiniti skupljima. Reactov virtualni DOM rješava te te probleme tako da detaljno prati gdje se u DOM-u dogodila promjena i ponovno prikaže samo taj specifični dio.
- Komponente. Reactove komponente su izdvojeni dijelovi programskog koda koje se mogu ponovno iskorištavati na različitim mjestima u aplikaciji bez dupliciranja koda. Osim toga, jedna od prednosti komponenata jest mogućnost kombiniranja više malih kako bi se dobila jedna kompleksnija komponenta. Ovakav pristup omogućava razdvojeno razvijanje dijelova programskog koda i lakšu podjelu posla.

Razvojni okvir Java Spring jest open-source razvojni okvir koji se koristi za razvijanje aplikacija koje rade na Javinom virtualnom stroju. Spring Boot je specijalizacija radnog okvira Spring. Glavne prednosti Spring Boota uključuju:

- Automatska konfiguracija projekta. Spring Boot ima mogućnost da automatski odluči koje su ovisnosti potrebne da bi neka aplikacija radila te da ih ažurira kako se aplikacija razvija kako razvojni tim ne bi morao ručno odabirati i unositi te podatke.
- Razvoj samostalnih aplikacija. Spring Boot omogućuje razvoj aplikacija koje se samostalno pokreće bez oslanjanja na vanjske web poslužitelje.

Naš je tim odabrao ovaj radni okvir i biblioteku za razvoj aplikacije poglavito zbog prethodnog iskustva s njima.

1. Instalacija

- Preduvjeti: Java 21, Node.js 16
- Preuzimanje:

```
git clone https://github.com/IvanPastorcic/LeteciMedvjedici.git
```

```
cd LeteciMedvjedici
```

Instalacija ovisnosti:

```
npm install
```

2. Postavke

Baza se može inicijalizirati preko data-dev.sql datoteke.

3. Pokretanje aplikacije

- Razvojno okruženje:

npm start (pokretanje lokalnog frontend servera) java -jar (pokretanje lokalnog backend servera) Frontend server pokrene se na portu 3000, a backend server na portu 8081.

- Render platforma:

Za pokretanje aplikacije na Renderu dovoljno je pokrenuti workflows u akcijama, na main branchu nalaze sve potrebne datoteke za pokretanje na Render platformi.

- Pokretanje aplikacije: Render će automatski preuzeti repozitorij, instalirati ovisnosti i pokrenuti aplikaciju. Nakon deploja, aplikaciji možete pristupiti putem generiranog URL-a (<https://safebear.onrender.com/>).

4. Upute za administratore

Administrator može pristupiti aplikaciji prijavom preko maila l.controller.m@gmail.com (lozinka: prOgi.Projekt1)

Opis pristupa aplikaciji na javnom poslužitelju

Pristup aplikaciji: Aplikacija je dostupna na linku: <https://safebear.onrender.com/>

Običan korisnik može aplikaciji pristupiti tako da se registrira i prijavi preko sučelja. Anoniman korisnik može automatski podnijeti prijavu nepogode odabirom odgovarajuće opcije na početnoj stranici. Humanitarne organizacije i vlasti koje bi željele imati račun u našoj aplikaciji moraju se javiti administratoru sustava na email (l.controller.m@gmail.com) sa zamolbom da im se kreira račun sa odgovarajućim ovlastima. Naš administrator će im se povratno javiti kada račun bude kreiran u sustavu. Ograničenja: server dopušta do 2500 prijava nepogoda dnevno zbog OpenCage Geocoding API-ja

Za testiranje funkcionalnosti u sustav su uneseni inicijalni korisnici sa posebnim ovlastima. Njihovim sučeljima može se pristupiti preko slijedećih emailova:

HUMANITARNA ORGANIZACIJA l.humanitarna.m@gmail.com lozinka: Progi.Projekt3

VLAST predstavnikvlasti@gmail.com lozinka: pRogi.pRojekt2

ADMIN l.controller.m@gmail.com lozinka: prOgi.Projekt1

Izrada ovog projektnog zadatka zahtjevala je značajan vremenski angažman, posebno u fazama istraživanja, planiranja i implementacije.

Glavni tehnički izazovi su bili integracija i upoznavanje s različitim tehnologijama. Izazov je riješen timskim radom i istraživanjem dodatnih izvora.

Tijekom izrade projekta naučili smo primjenjivati nove tehnologije te smo razvili timske vještine

Kako bismo projekt realizirali brže i kvalitetnije, posebno bi bilo korisno dodatno znanje iz područja upravljanja projektima, posebno u smislu bolje procjene potrebnog vremena i resursa.

Perspektive za nastavak rada u projektnoj grupi su obećavajuće. Tim je pokazao visok stupanj motivacije i fleksibilnosti.

- F-02.1
- F-05.1
- F-05.2
- F-06
- F-07.2
- F-10
- F-17.1
- F-18.1
- NF-02
- NF-03
- NF-05.1

Literatura

Reference

1. FEMA, "FEMA App," FEMA.gov. [Online]. Dostupno: <https://www.fema.gov/> [Datum pristupa: 10. studenoga 2024.]
2. MyShake, "MyShake App," MyShake.org. [Online]. Dostupno: <https://myshake.berkeley.edu/> [Datum pristupa: 10. studenoga 2024.]
3. A. Li, "Google winding down standalone Crisis Map website given Search, Maps integration," 9to5Google, 2021. [Online]. Dostupno: <https://9to5google.com/2021/02/17/google-crisis-map/> [Datum pristupa: 10. studenoga 2024.]

Alati

1. React, "React," [Online] Dostupno: <https://react.dev/> [Datum pristupa: 10. studenoga 2024.]
2. Spring Boot, "Spring Boot," [Online] Dostupno: <https://spring.io/> [Datum pristupa: 10. studenoga 2024.]
3. Apache Maven, "Maven," [Online] Dostupno: <https://maven.apache.org/> [Datum pristupa: 10. studenoga 2024.]

4. H2 Database, "H2 Database," [Online] Dostupno: <https://www.h2database.com/html/main.html> [Datum pristupa: 10. studenoga 2024.]
5. Figma, "Figma," [Online] Dostupno: <https://www.figma.com/> [Datum pristupa: 10. studenoga 2024.]
6. Astah, "Astah UML," [Online] Dostupno: <https://astah.net/> [Datum pristupa: 10. studenoga 2024.]
7. Postman, "Postman," [Online] Dostupno: <https://www.postman.com/> [Datum pristupa: 10. studenoga 2024.]
8. Render, "Render," [Online] Dostupno: <https://render.com/> [Datum pristupa: 10. studenoga 2024.]

Korišteni izvori

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak	I.P.	29.10.2024
0.1.1	Dodan dijagram baze podataka u dokumentaciju	N.Ž.	31.10.2024
0.1.2	Dodana relacijska shema baze podataka u dokumentaciju	N.Ž.	01.11.2024
0.1.3	Dodani funkcionalni zahtjevi	I.P., E.V.	02.11.2024
0.1.4	Popravak dijagrama i sheme baze	I.P.	05.11.2024
0.2	Dopunjavanje opisa projektnog zadatka	I.P.	07.11.2024
0.2.1	Dopunjavanje opisa projektnog zadatka	I.S.	07.11.2024
0.3	Dovršen opis projektnog zadatka	I.P.	08.11.2024
0.3.1	Dodan finalni dizajn baze podataka	I.P.	08.11.2024
0.3.2	Dodan visokorazinski dijagram obrazaca uporabe	I.P.	09.11.2024
0.3.3	Dodan NFZ za ograničenja na prijavu nepogoda	I.P.	09.11.2024
0.3.4	Dodavanje literature za opis projekta	I.P.	10.11.2024
0.3.5	Dodavanje alata u literaturu	I.P.	10.11.2024
0.3.6	Dodavanje pristupa aplikaciji na početnu stranicu	I.P.	10.11.2024
0.3.7	Dodavanje FZ za praćenje prijava	I.P.	10.11.2024
0.3.8	Dodavanje scenarija za registraciju korisnika	I.P.	11.11.2024
0.3.9	Dovršena arhitektura sustava	I.P.	14.11.2024
0.3.10	Dorađeni scenariji za zahtjeve	I.P.	15.11.2024
0.4	Dorađena specifikacija sustava	I.P., E.V., I.S., L.Z.	15.11.2024
1.0	Verzija prve predaje	I.P., E.V.	15.11.2024
1.0.1	Ažurirana dokumentacija u skladu s primjerkom	I.P.	16.1.2025.

Rev.	Opis promjene/dodataka	Autori	Datum
1.0.2	Dorađen dijagram i relacijska shema baze podataka	N.Z	19.1.2025.
1.0.3	Dorađen dijagram klasa	N.Z	19.1.2025.
1.0.4	Dodani FZ i dijagram komponenti	E.V.	19.1.2025.
1.0.5	Dodani ostali dijagrami te alati	I.P.	19.1.2025.
1.0.6	Ažuriran homePage	I.P.	19.1.2025.
1.0.7	Ispravljen visokorazinski dijagram i dijagram komponenti	I.P., E.V.	19.1.2025.
1.0.8	Dorađen dijagram i relacijska shema baze podataka	N.Z	24.1.2025.
1.0.9	Dorađen dijagram klasa	N.Z	24.1.2025.
1.1.1	Neke finalne promjene	I.P.	24.1.2025.
1.1.2	Dodane upute za puštanje u pogon	E.V	24.1.2025.
2.0.0	Verzija finalne predaje	I.P., E.V.	24.1.2025.

Dnevnik sastajanja

1. sastanak

- Datum: 14. listopada 2024.
- Prisustvovali: E. Vuran, L. Zekušić, I. Pastorčić, N. Žegarac, S. Rade, K. Horvat, I. Sulić
- Teme sastanka:
- Uspostavljena Discord grupa za komunikaciju
- Podjela članova tima na backend i frontend
- Rasprava o ideji projekta

2. sastanak

- Datum: 20. listopada 2024.
- Prisustvovali: E. Vuran, L. Zekušić, I. Pastorčić, N. Žegarac, S. Rade, K. Horvat, I. Sulić
- Teme sastanka:
- Uspostava GitHub repozitorija
- Definiranje funkcijskih i nefunkcijskih zahtjeva aplikacije

3. sastanak

- Datum: 24. listopada 2024.
- Prisustvovali: E. Vuran, I. Pastorčić, K. Horvat, I. Sulić
- Teme sastanka:
- Definiranje i razrada UML use case i sekvencijskih dijagrama

4. sastanak

- Datum: 28. listopada 2024.
- Prisustvovali: E. Vuran, L. Zekušić, I. Pastorčić, N. Žegarac, S. Rade
- Teme sastanka:
- Definiranje funkcionalnosti pojedinih stranica aplikacije
- Dogovor oko rada na dizajnu UI
- Diskusija o arhitekturi baze podataka

5. sastanak

- Datum: 30. listopada 2024.
- Prisustvovali: E. Vuran, I. Pastorčić, N. Žegarac
- Teme sastanka:
- Predstavljanje i daljna diskusija vezana uz bazu podataka
- Dodatno informiranje oko tehnologije SpringBoot i podjela poslova na backendu

6. sastanak

- Datum: 3. studenoga 2024.
- Prisustvovali: E. Vuran, I. Pastorčić, N. Žegarac, L. Zekušić, S. Rade, I. Sulić, K. Horvat
- Teme sastanka:
- Prezentacija dosad napravljenog posla na frontendu (login, register, report) te daljnja diskusija oko dizajna stranica
- Prezentacija dosad napravljenog posla na backendu (Spring projekt, granulacija po paketima)
- Razrada plana rada za idući tjedan

7. sastanak

- Datum: 8. studenoga 2024.
- Prisustvovali: E. Vuran, L. Zekušić, I. Sulić
- Teme sastanka:
- Rasprava o zaostacima
- Prijedlog bolje razdjеле posla i bolje komunikacije u timu

8. sastanak

- Datum: 11. studenoga 2024.
- Prisustvovali: E. Vuran, L. Zekušić, I. Pastorčić, N. Žegarac, S. Rade, K. Horvat, I. Sulić
- Teme sastanka:
- Završna podjela posla prije međuispita
- Definiranje internih rokova za predaju

9. sastanak

- Datum: 9. siječnja 2025.

- Prisustvovali: E. Vuran, L. Zekušić, I. Pastorčić, N. Žegarac, S. Rade, I. Sulić
- Teme sastanka:
- Rasprava o funkcionalnostima aplikacije vezanim uz alfa verziju
- Podjela poslova među članovima tima i dogovaranje rokova
- Rasprava i dogovor oko testiranja aplikacije

10. sastanak

- Datum: 16. siječnja 2025.
- Prisustvovali: E. Vuran, I. Pastorčić, N. Žegarac, S. Rade
- Teme sastanka:
- Prikaz rada članova
- Planovi za backend i dokumentaciju
- Daljnja raspodjela poslova za buduće tjedne

Plan rada

Tjedan	Aktivnost na projektu	Angažirani članovi
3.	> uspostava komunikacije među članovima tima > uspostava GitHub repozitorija > definiranje funkcijskih i nefunkcijskih zahtjeva	> E.V., I.P., L.Z., S.R., N.Ž., I.S., K.H. > I.P. > E.V., I.P., L.Z., S.R., N.Ž., I.S., K.H.
4.	> razrada UML dijagrama	> E.V., I.P., I.S.
5.	> dizajn aplikacije > razrada dijagrama baze podataka > dokumentacija > stvaranje SpringBoot projekta > Login i Register stranice > Report page	> L.Z. > I.P., N.Ž. > I.P., E.V. > E.V. > I.S. > K.H.
6.	> backend modeli > OAuth2.0 protokol i Google cloud > dokumentacija > rad na bazi podataka > backend za Report > AppUser backend	> I.P. > E.V. > I.P., I.S., S.R. > I.P. > N.Ž. > E.V.
7.	> dokumentacija > spajanje frontenda i backenda > Homepage > Report page	> I.P., I.S., L.Z., N.Ž., E.V. > E.V. > S.R. > K.H., E.V.
10.	> deployment > ažuriranje baze podataka > dodavanje potrebnih endpointa > dodavanje OpenStreetMap API	> N.Ž. > N.Ž. > E.V. > I.S.
11.	> HomePage (vlasti, admin, humanitarne), dodavanje potreba > Deployment, baza podataka, rad sa geografskim koordinatama > Dodavanje rola, spajanje frontenda i backenda > Profile page, pojedinačni report i need > Dodavanje humanitarne akcije i baratanje resursima	> L.Z. > N.Ž. > E.V. > S.R. > I.S.
12.	> završni radovi na backendu > dokumentacija > završni	> N.Ž., E.V. > I.P. >

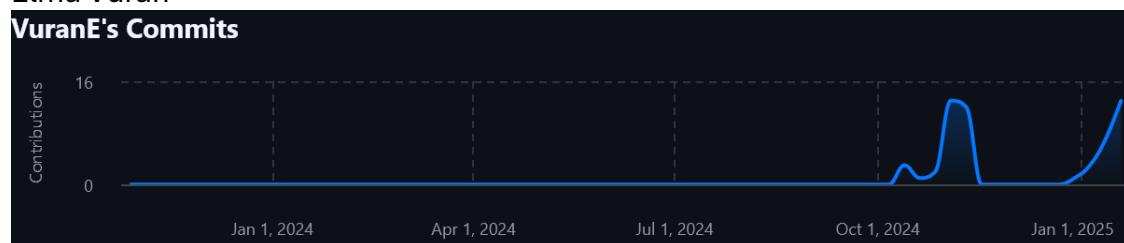
Tjedan	Aktivnost na projektu	Angažirani članovi
	radovi na frontendu	L.Z., I.S., S.R., K.H.

Tablica aktivnosti

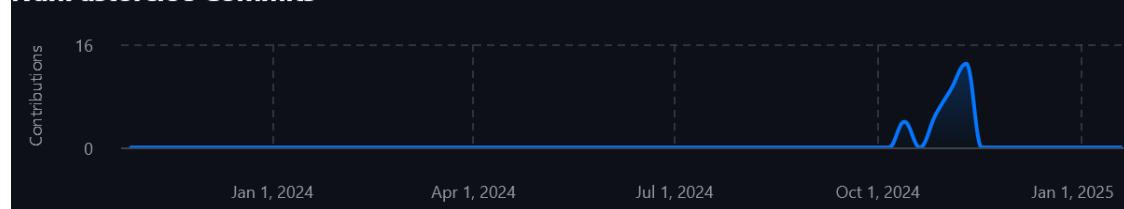
Dijagram pregleda promjena

Elma Vuran

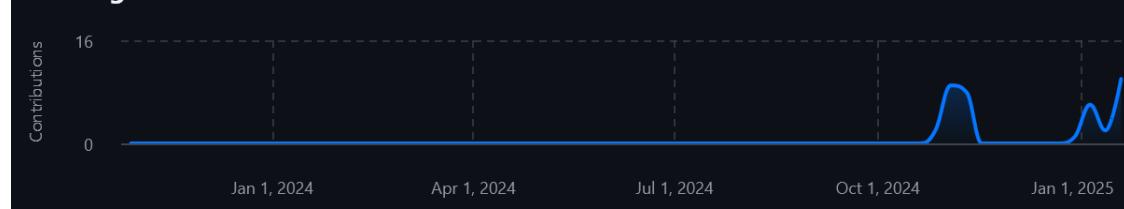
VuranE's Commits



IvanPastorcic's Commits



natalizegarac's Commits



ivasulic's Commits



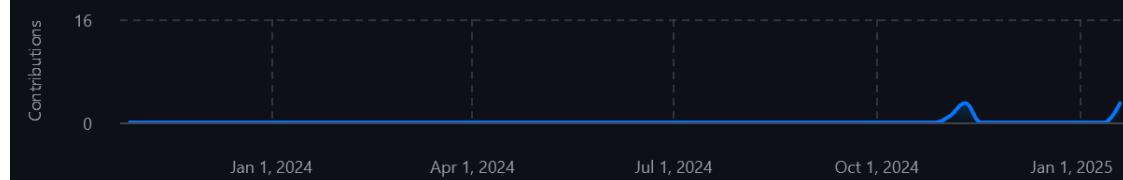
Izekusic's Commits



SilvijaRade's Commits



kresohorvat's Commits



Kjučni izazovi i rješenja

- Zaključno
- Opis izazova: kašnjenje u razvoju, pogrešna tehnička implementacija, nepoštivanje zadanih rokova
- Rješenja: dodatni rad na projektu, proučavanje tehničke dokumentacije