

[Code](#)[Issues 4](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)

Home

[Edit](#)[New page](#)[Jump to bottom](#)

Elena Neseš edited this page last week · [2 revisions](#)

Welcome to the DisasterMaster wiki!

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Naziv projekta : DisasterMaster

Tim: <TG07.2>

Ime tima : Disaster Master

Nastavnik: Vlado Sruk

+ Add a custom footer

▼ Pages 12

Find a page...

▼ Home

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

- ▶ [1. Opis projektnog zadatka](#)
- ▶ [2. Analiza zahtjeva](#)
- ▶ [3. Specifikacija zahtjeva sustava](#)
- ▶ [4. Arhitektura i dizajn sustava](#)
- ▶ [5. Arhitektura komponenata i razmještaja](#)
- ▶ [6. Ispitivanje programskog rjesenja](#)
- ▶ [7. Tehnologija za implementaciju aplikacije](#)
- ▶ [8. Upute za pustanje u pogon](#)
- ▶ [9. Zaključak i budući rad](#)
- ▶ [Popis literature](#)
- ▶ [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>





1. Opis projektnog zadatka

[Edit](#)[New page](#)[Jump to bottom](#)

en765 edited this page on Nov 15, 2024 · [5 revisions](#)

Cilj projekta je osmisliti i izraditi web aplikaciju koja obavjestava o vremenskim nepogodama. Preciznije, korisnicima omoguciti prijavljivanje nepogoda u njihovoj blizini, pristup korisnim informacijama koje ce im pomoci tijekom tih istih nepogoda. Vlasti kao administratori najvise razine imaju pristup svim prijavama te njihovo odbacivanje ili prihvatanje te generiranje statickih izvjesca. Humanitarne organizacije kao administratori imaju pristup pregledu prijava te mogucnost objavljivanja informacija o resursima, skolinistima i slicno.

Problematika —> Rjesenja

- kreiranje razvijene korisnicke aplikacije po prvi put
- rad u timu i organizacija zajednickog timskog rada
- razumijevanje dosega projektnog zadatka: problem stvaraju detalji kojima se ne zamaras na vrijeme i tek onda shvatis da je bitno donijeti odluku oko svakog dijela implementacije

Korist projekta

- brze dojave o vremenskim nepogodama
- brzi dolazak pomoci na mjesto eventualne nesrece prouzrocene nepogodom
- poboljsanje suradnje izmedu vlasti (civilna zaštita, hitna pomoc) i humanitarnih organizacija
- spriječavanje materijalne stete prouzrocene vremenskom nepogodom
- prevencija prometnih zastoja i drugih oblika problema u prometu

Slicna rjesenja

Disaster Alert

- aplikacija koja omogucuje pregled vremenskih nepogoda, ali ne i njihovo prijavljivanje
- koristi vanjske resurse za informacije o nepogodama
- za odredenu nepogodu sadrzi informacije o vrsti, opis te detaljnu lokaciju (koordinate)
- isto omogucuje stvaranje korisnickog racuna

The screenshot shows a mobile application interface for 'Disaster Alert™'. The top navigation bar includes a blue triangle icon, the app name 'Disaster Alert™', a back arrow, the title 'Volcano - Semeru, Indo...', and a three-dot menu icon. On the left is a vertical sidebar with five icons: a blue exclamation mark, a grey document, a white star, a blue information circle, and a location pin.

VOLCANIC ERUPTION

Volcano - Semeru, Indonesia

Reported: 8 months ago
Updated: 8 minutes ago

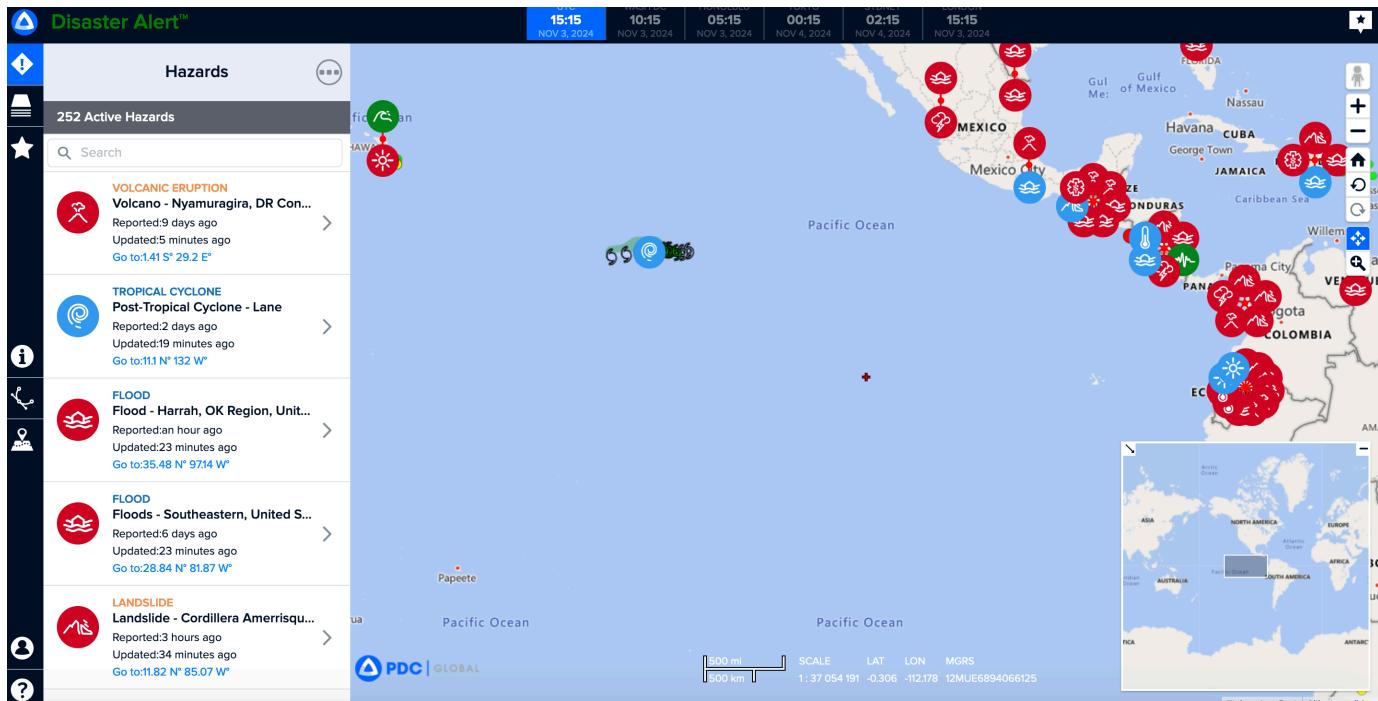
<https://disasteralert.pdc.org/disast...> [SHARE](#)

[MORE INFORMATION](#)

Description

Volcanic activity has been reported for Semeru in Indonesia, by the Volcanic Ash Advisory Centre (VAAC) DARWIN on November 15, 2024, 12:35:00 GMT. This volcanic activity is reported based on aerial, satellite and/or ground observations. Based on the limited data available, it is estimated that 72,200 people, 18,705 households, and \$162 Million (USD) of infrastructure* are within 10 km (6 miles) and more likely to be adversely impacted, and that there is no concentrated population between 10 km (6 miles) and 30 km (18 miles) of the volcano. Additional information will be provided as it becomes available by official sources. Volcano information sources for this report include: HIMAWARI-9, CVGHM *The cost represents the total replacement value of the infrastructure.

Created By: D2P2



Skup korisnika

- korisnici ove aplikacije variraju od gradana do osoba u nadleznim tijelima drzave
- aplikacija ce biti od velike koristi osobama koje vole putovati i zele znati kakvo je stanje na cestama i na samom odredistu putovanja
- takoder ce biti ood velike koristi osobama cija je profesija voznja, bilo kamiona ili buseva, uvijek je korisno biti svjestan vremenskih nepogoda
- zdravstvene ustanove, specificnije vozaci hitne pomoci takoder mogu koristiti aplikaciju kako bi se bolje pripremili u slucaju nesreće (slikama i opisom nepogode mogu bolje ustanoviti drasticnost situacije)
- uz hitnu pomoc i ostale hitne sluzbe mogli bi biti cesti korisnici aplikacije
- korisnicima ce aplikacija biti korisna kako bi znali koje dijelove grada izbjegavati u slucaju nepogoda

Opseg projektnog zadatka

Jednostavne implementacije: ulogiravanje u sustav, pretrazivanje lokacija, slanje prijava, dizajn cijelokupne aplikacije

Zahtjevnije implementacije: kreiranje cijele baze podataka, povezivanje baze sa svim drugim funkcionalnostima, koristenje vanjskih resursa za neke funkcionalnosti kao sto su određivanje lokacije i slanje obavijesti

Moguce nadogradnje

- direktna poveznica sa hitnim sluzbama (shortcut za poziv hitne pomoci ili vatrogasaca unutar mobitlne aplikacije ili klikom misa na gumb koji obavjestava hitne sluzbe)
- dodatne obavijesti o stanju u prometu (zatvorene ceste ili guzve)
- generiranje statistickih izvjesca dostupno gradanima kako bi unaprijed znali u koje doba godine na nekom području je veca koncentracija vremenskih nepogoda (korisno za planiranje putovanja)

+ Add a custom footer

▼ Pages 12

Find a page...

▶ [Home](#)

▼ [1. Opis projektnog zadatka](#)
Problematika —> Rjesenja
Korist projekta
Slicna rjesenja
 Disaster Alert
 Skup korisnika
 Opseg projektnog zadatka
 Moguce nadogradnje

▶ [2. Analiza zahtjeva](#)

▶ [3. Specifikacija zahtjeva sustava](#)

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

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

▶ [6. Ispitivanje programskog rjesenja](#)

▶ [7. Tehnologija za implementaciju aplikacije](#)

▶ [8. Upute za pustanje u pogon](#)

▶ [9. Zaključak i budući rad](#)

▶ [Popis literature](#)

▶ [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>



[Code](#)[Issues 4](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)

2. Analiza zahtjeva

[Edit](#)[New page](#)[Jump to bottom](#)Elena Neseck edited this page last week · [8 revisions](#)

Funkcionalni zahtjevi

ID zahtjeva	Opis	Proriteti	Izvor	Kriterij prihvacanja
F-001	Sustav korisniku omogucava prijavu nepogode	Visok	Zahtjev dionika	Korisnik ima mogucnost ispunjavanja forme za prijavu i uspjesno ju objaviti na sustav
F-001.1	Prijava putem fotografije	Visok	Zahtjev dionika	Korisnik odabire opciju dodavanja slike iz galerije i ona se uspjesno dodaje u fromu
F-001.2	Prijava putem opisa	Visok	Zahtjev dionika	Korisnik dodajte opis svojoj prijavi i uspjesno se dodaje u formu
F-001.3	Prijava lokacije	Visok	Zahtjev dionika	<i>Korisnik odabire lokaciju - korisnik bira koliko detaljno zeli oznacit lokaciju*</i> i uspjesno se azurira unutar sustava
F-002	Sustav omogucuje korisniku prijavu u aplikaciju putem emaila	Visok	Zahtjev dionika	Korisnik se moze registrirati putem e-mail i dobiti povrdu o uspjesnoj registraciji

ID zahtjeva	Opis	Proiriteti	Izvor	Kriterij prihvacanja
F-002.1	Sustav omogucuje opciju prijavu (sign-in) u sustav	Visok	Zahtjev dionika	Registrirani korisnik ima opciju ulogiravanja u postojeci profil putem emaila i lozinke
F-003	Sustav korisniku omogucava primanje obavijesti putem maila	Visok	Povratne infomracije korisnika	Prijavljeni korisnik moze primati obavijesti putem maila
F-003.1	Primanje obavijesti o prijavama povezanim uz korisnikovu	Visok	Povratne informacije korisnika	Prijavljeni korisnik prima obavijesti o prijavama nepogoda na lokacijama u blizini njegove prijave u periodu od 5 minuta nakon sto podnese novu prijavu
F-004	Pregled informacija o resursima i sigurnosnim mjerama	Visok	Zahtjev dionika	Korisnik odabire neku od lokacija ozначенih na interaktivnoj mapi te ako su objavljene informacije o resursima i sigurnosnim mjerama moze ih vidjeti
F-005	Vlasti imaju pristup pregledu svih prijava	Visok	Zahtjev dionika	Korisnik sa ulogom vlasti ima mogucnost pregleda svih prijava unutar njihovog autorativnog područja
F-005.1	Vlasti imaju pregled svih nepregledanih prijava	Visok	Zahtjev dionika	Korisnik s ulogom vlasti ima mogucnost odabira pregleda svih neprovjerenih prijava u obliku liste
F-005.1.1	Vlasti mogu odbiti prijavu	Visok	Zahtjev dionika	Vlasti mogu odbijati prijave koje nisu legitimne i one se brisu iz sustava

ID zahtjeva	Opis	Proiriteti	Izvor	Kriterij prihvacanja
F-005.1.2	Vlasti mogu prihvatiti prijavu	Visok	Zahtjev dionika	Vlasti mogu prihvatiti legitimne objave koje se onda objavljaju na mapi
F-006	Interaktivna mapa omogucuje korisniku pregled prihvacenih prijava u obliku pinga na mapi	Visok	Postojeci sustav	Korisnik moze zoomuranje te odabirom bilo koju od ikonica (pingova) na mapi vidjeti potvrđene prijave na toj lokaciji
F-007	Vlasti mogu zatraziti generiranje statistickog izvjesca	Srednje	Zahtjev dionika	Sustav omogucuje opciju odabira generiranja statistickog izvjesca
F-008	Humanitarne organizacije mogu pregledavati prijave nepogoda	Visoko	Zahtjev dionika	Humanitarne organizacije mogu pregledavati povrdene prijave na temelju kojih ce objavljivati informacije o resursima
F-008.1	Humanitarne organizacije mogu objavljivati informacije o resursima i sigurnosnim mjerama	Zahtjev korisnika	Srednje	Na temelju korisnickih prijava humanitarne organizacije na aplikaciju mogu u obliku opisa postaviti informacije o resursima i sigurnosnim mjerama koje se kasnije prikazuju u korisnickom sucelju

Ostali (nefunkcionalni)zahtjevi

Zahtjevi za održavanje:

ID zahtjeva	Opis	Prioritet
NO-1	Sustav treba biti popraćen "Priručnikom za rad" koji opisuje pravilnu upotrebu sustava	Visok
NO-2	Aplikacije treba biti jednostavna i intuitivna za koristenje	Visok

ID zahtjeva	Opis	Prioritet
NO-3	Sustav treba imati dovoljnu dokumentaciju	Srednje
NO-4	Aplikacija treba biti modularna, kako bi se lakše nadograđivala i prilagođavala novim zahtjevima ili integracijama s vanjskim sustavima	Srednje

Zahtjevi skalabilnosti:

ID zahtjeva	Opis	Prioritet
NS-1	Sustav treba podrzavati veliki broj istovremenih korisnika, posebno tijekom hitnih situacija	Visoki
NS-2	Sustav treba moci u nekoliko minuta obraditi prijavu (odobriti/odbiti), te poslati povratnu informaciju korisniku	Srednje
NS-3	Baza podataka treba podržavati horizontalnu skalabilnost, kao što je sharding, kako bi se smanjilo opterećenje na jednom serveru i omogućio paralelni rad na više instanci baze	Visok
NS-4	Omogućiti asinkronu obradu složenijih operacija u pozadini, čime se smanjuje vrijeme čekanja za korisnike i osigurava da sustav može obraditi veći broj zahtjeva istovremeno	Visok

Zahtjevi pristupacnosti i prilagodljivosti

ID zahtjeva	Opis	Prioritet
NPP-1	Sustav treba biti oblikovan tako da podrzava jednostavno odrzavanje	Visok
NPP-2	Aplikacija mora biti responzivna i usklađena s WCAG standardima pristupačnosti kako bi bila dostupna svim korisnicima [WCAG 2 Overview]	Web Accessibility Initiative (WAI)
NPP-3	Implementacija strategija za visoku dostupnost: Postaviti tehničke mehanizme i strategije za visoku dostupnost, uključujući redundanciju podataka i failover mehanizme koji omogućuju nastavak rada u slučaju kvara	Visok
NPP-4	Sigurnost podataka: Implementirati šifriranje podataka i sigurnosne provjere identiteta	Srednje

ID zahtjeva	Opis	Prioritet
NPP-5	Aplikacija treba imati ugrađeni centar za podršku i mogućnost slanja povratnih informacija putem maila, što će korisnicima omogućiti pomoći unutar aplikacije i povećati zadovoljstvo korisničkim iskustvom	Srednje

Zahtjevi performanse

ID zahtjeva	Opis	Prioritet
NP-1	Aplikacija treba prikazati rezultate korisničkih akcija u roku od 1-2 sekunde za većinu interakcija, čime će se osigurati brza povratna informacija i korisničko iskustvo	Visok
NP-2	Aplikacija treba biti optimizirana kako bi učinkovito koristila resurse poslužitelja i smanjila opterećenje procesora i memorije, osiguravajući stabilan rad čak i uz veliko opterećenje	Srednje

Zahtjevi ogranicenja

ID zahtjeva	Opis	Prioritet
NOG-1	Omogućiti prijavu samo s lokacija unutar regije ili države u kojoj aplikacija operira	Srednje
NOG-2	Postaviti ograničenje broja prijava koje jedan korisnik može podnijeti u kratkom vremenskom razdoblju (npr. jednu prijavu svakih 30 minuta)	Nisko

Dionici

Korisnici

Administrator - vlasti

Administrator - humanitarne organizacije

Narucitelji - Vlado Sruk

Razvojni tim (DisasterMaster grupa)

Aktori

A1 Anonimni korisnik - F-001, F-001.1, F-001.2, F-001.3, F-002, F-004, F-006

A2 Registrirani korisnik - F-001, F-001.1, F-001.2, F-001.3, F-002.1, F-002.2, F-003, F-003.1, F-004, F-006

A3 Administrator vlasti - F-005, F-005.1, F-005.1.1, F-005.1.2, F-007, F-007.1, F-007.2

A4 Administrator humanitarne organizacije - F-004, F-008, F-008.1

A5 Vanjski resursi - F-002.1, F-002.2, F-003, F-003.1, F-006

A6 Baza podataka - Prisutna u gotovo svim zahtjevima

+ Add a custom footer

▼ Pages 12

Find a page...

► [Home](#)

► [1. Opis projektnog zadatka](#)

▼ [2. Analiza zahtjeva](#)

- Funkcionalni zahtjevi
- Ostali (nefunkcionalni)zahtjevi
- Zahtjevi za održavanje:
- Zahtjevi skalabilnosti:
- Zahtjevi pristupacnosti i prilagodljivosti
- Zahtjevi performanse
- Zahtjevi ogranicenja
- Dionici
- Aktori

► [3. Specifikacija zahtjeva sustava](#)

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

► [5. Arhitektura komponenata i razmještaja](#)

► [6. Ispitivanje programskog rjesenja](#)

▶ [7. Tehnologija za implementaciju aplikacije](#)

▶ [8. Upute za pustanje u pogon](#)

▶ [9. Zaključak i budući rad](#)

▶ [Popis literature](#)

▶ [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>



[Code](#)[Issues 4](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)

3. Specifikacija zahtjeva sustava

[Edit](#)[New page](#)[Jump to bottom](#)

Elena Neseck edited this page last week · [5 revisions](#)

Aktori

1. Korisnik

-slati prijave

-pregledavati stare prijave

-Vidjeti informacije o resursima, sklonistima i slicno

2. Registrirani korisnik

-sve funkcionalnosti obicnog (annimnog) korisnika

-Prijavljivanje u sustav

-Dobivanje obavijesti o statusu prijave

3. Vlasti

-pregled svih prijava

-Prihvatanje ili odbijanje prijava

-Generiranje statistickog izvjesca

4. Humanitarne organizacija

-pregled prijava

-Objavljanje korisnih informacija za pomoc korisnicima(gradanima)

5. Disaster Master web aplikacija

- obrada zahtjeva korisnika (prijava u sustav, slanje prijave nepogode)

6. Vanjski resursi

- omogucuje koristenje interaktivne mape
- omogucuje slanje obavijesti
- omogucuje autentifikaciju i sigurnost pri prijavi u aplikaciju

7. Baza podataka

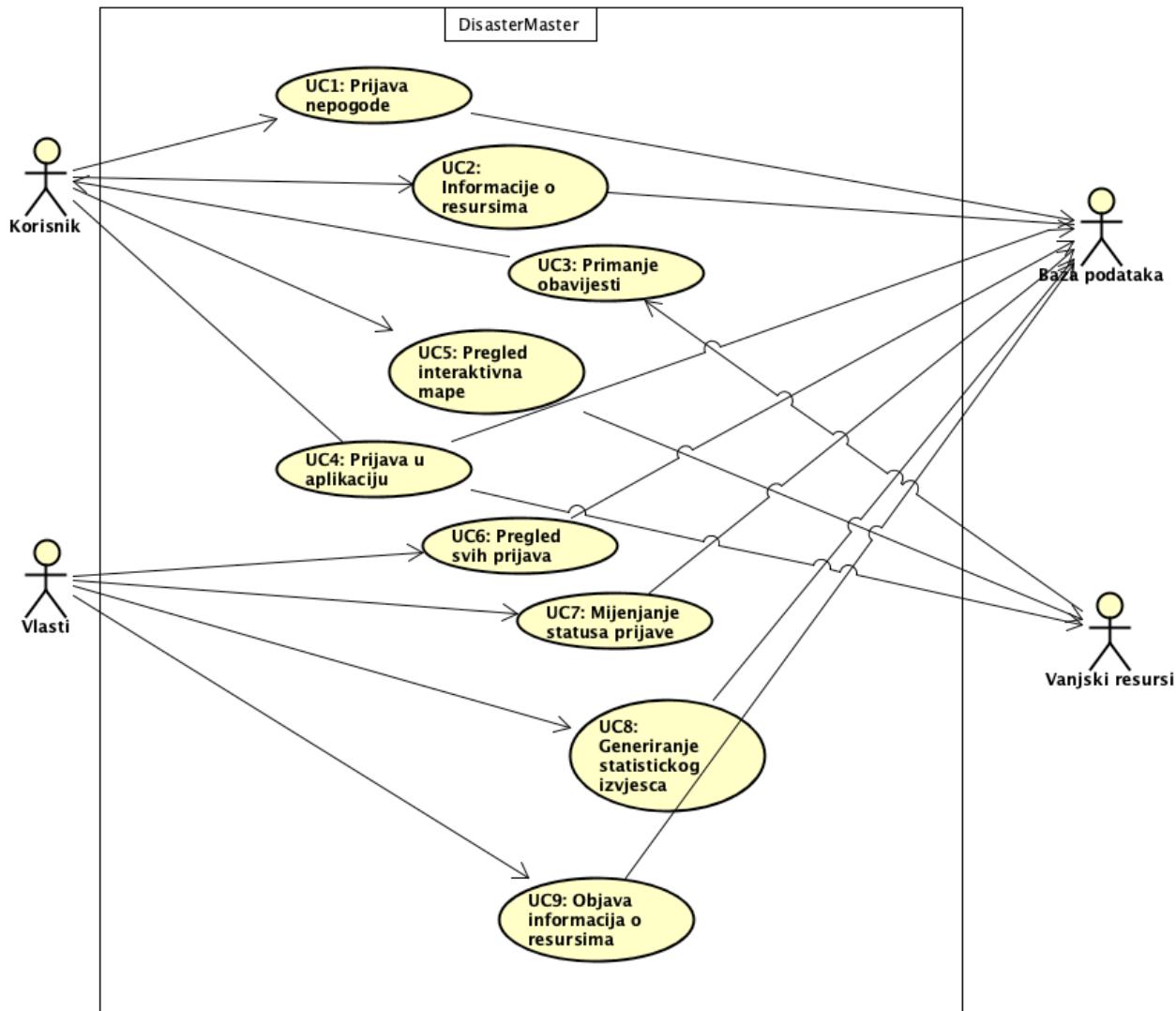
- spremanje podataka o korisnicima
- spremanje podataka o prijavama
- spremanje podataka o resurima, sklonistima...

Obrasci uporabe

Opis obrazaca uporabe

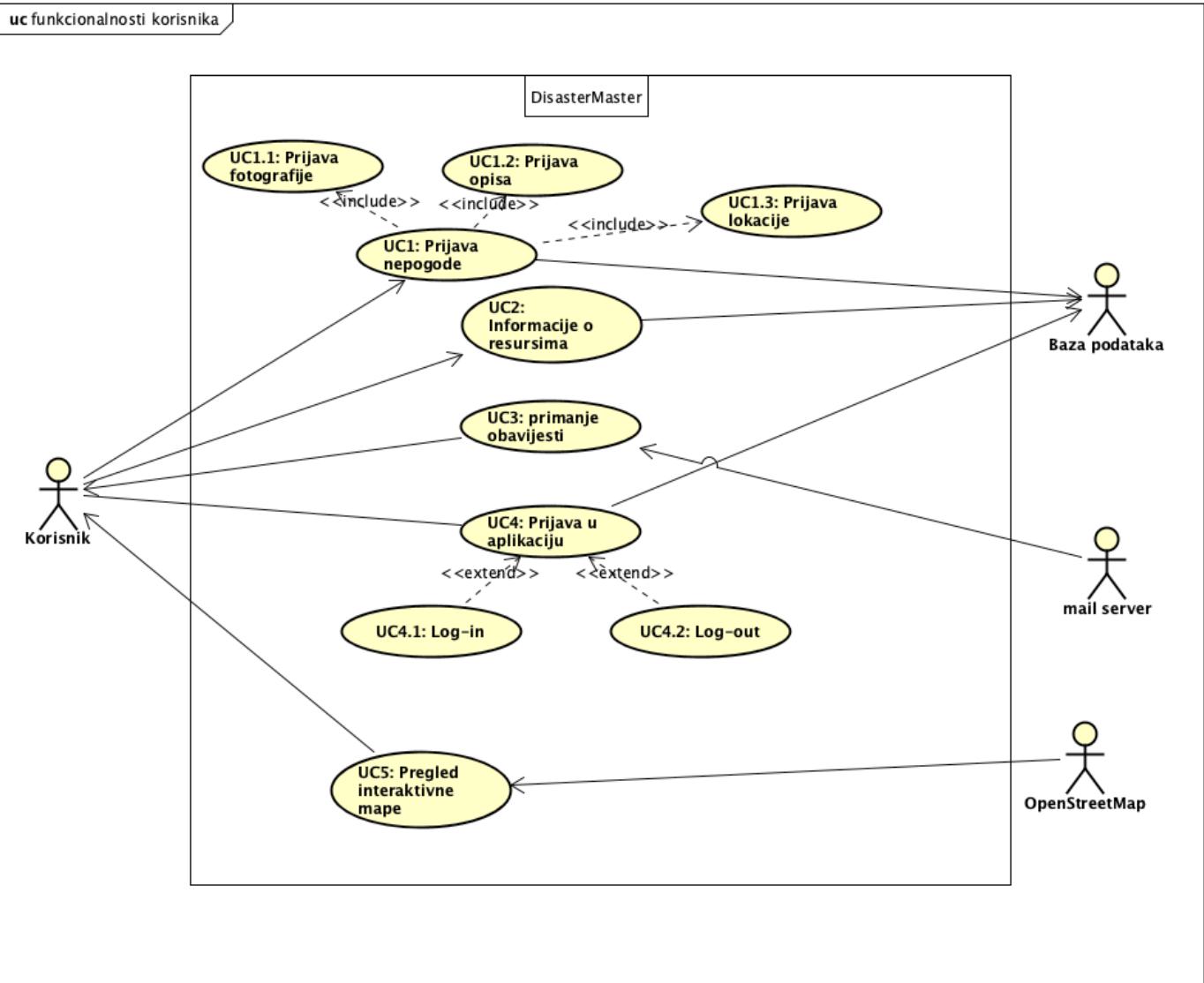
Funkcionalni zahtjevi razradeni u obliku obrazaca uporabe

1. Visokorazinski dijagram svih funkcionalnosti aplikacije

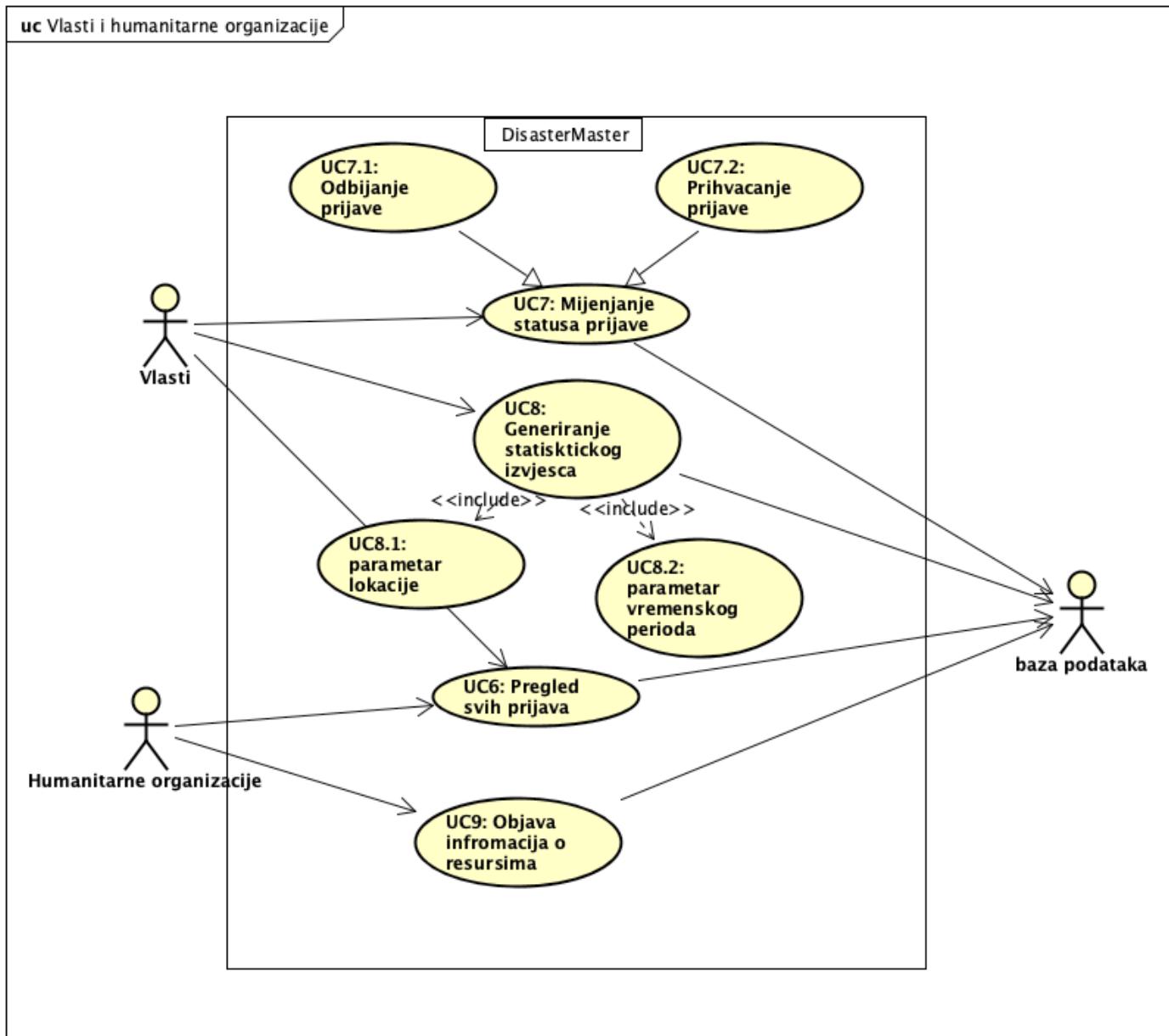
uc general functions

2. Vrste korisnika

2.1 Dijagram funkcionalnosti Korisnika:

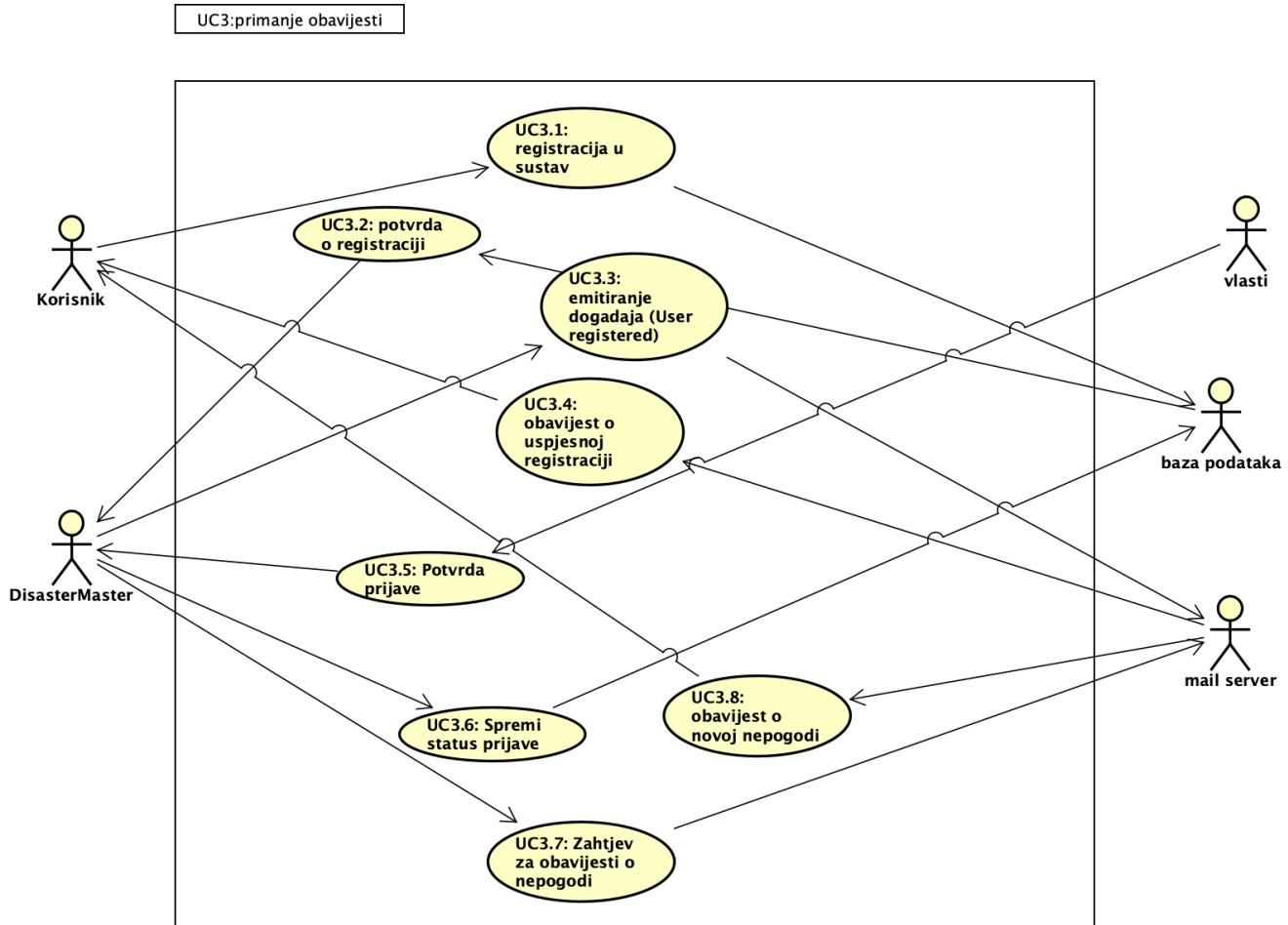


2.2 Dijagram funkcionalnosti Vlasti i Humanitarnih organizacija:

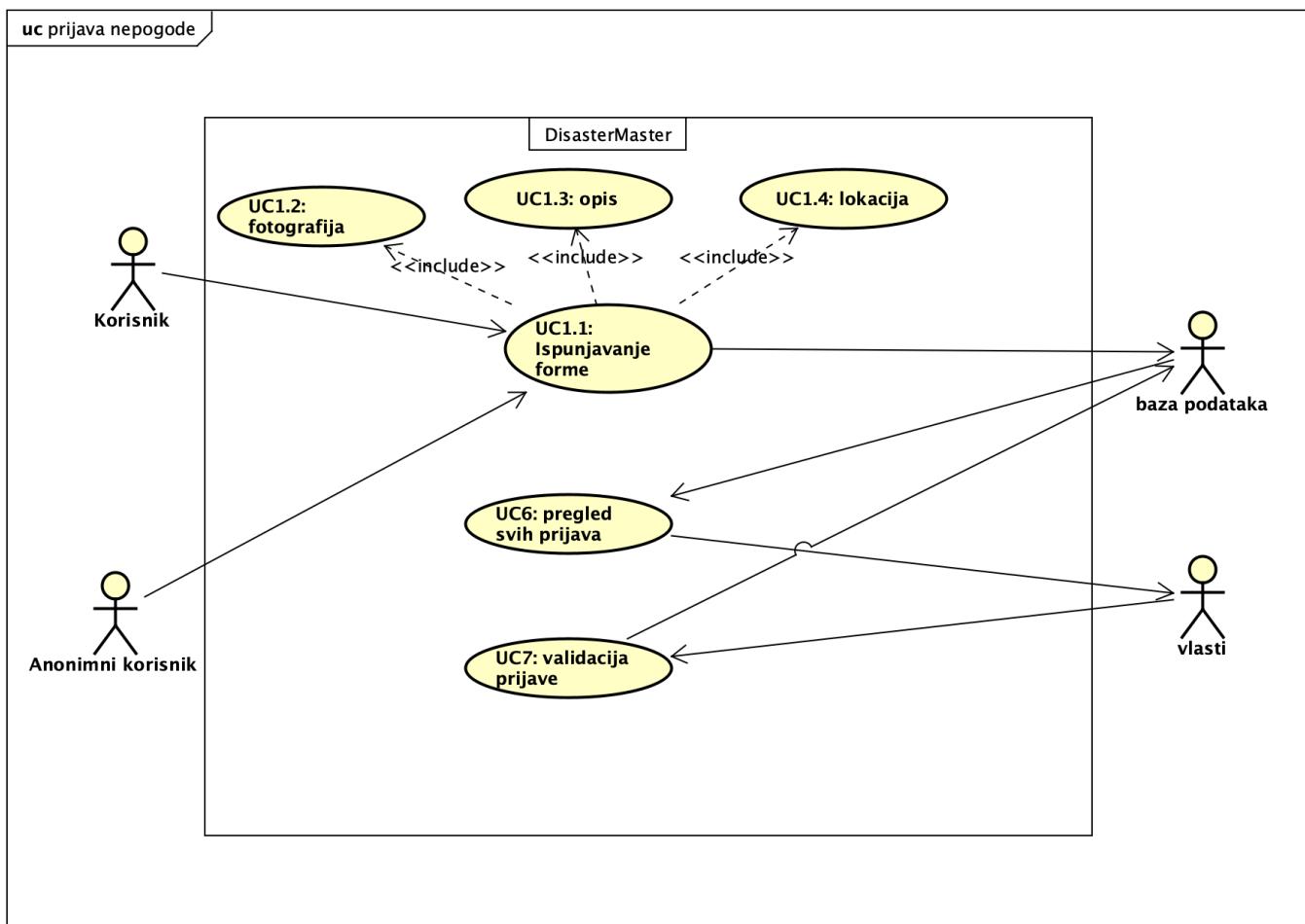


3. Dijagrami glavnih funkcionalnosti

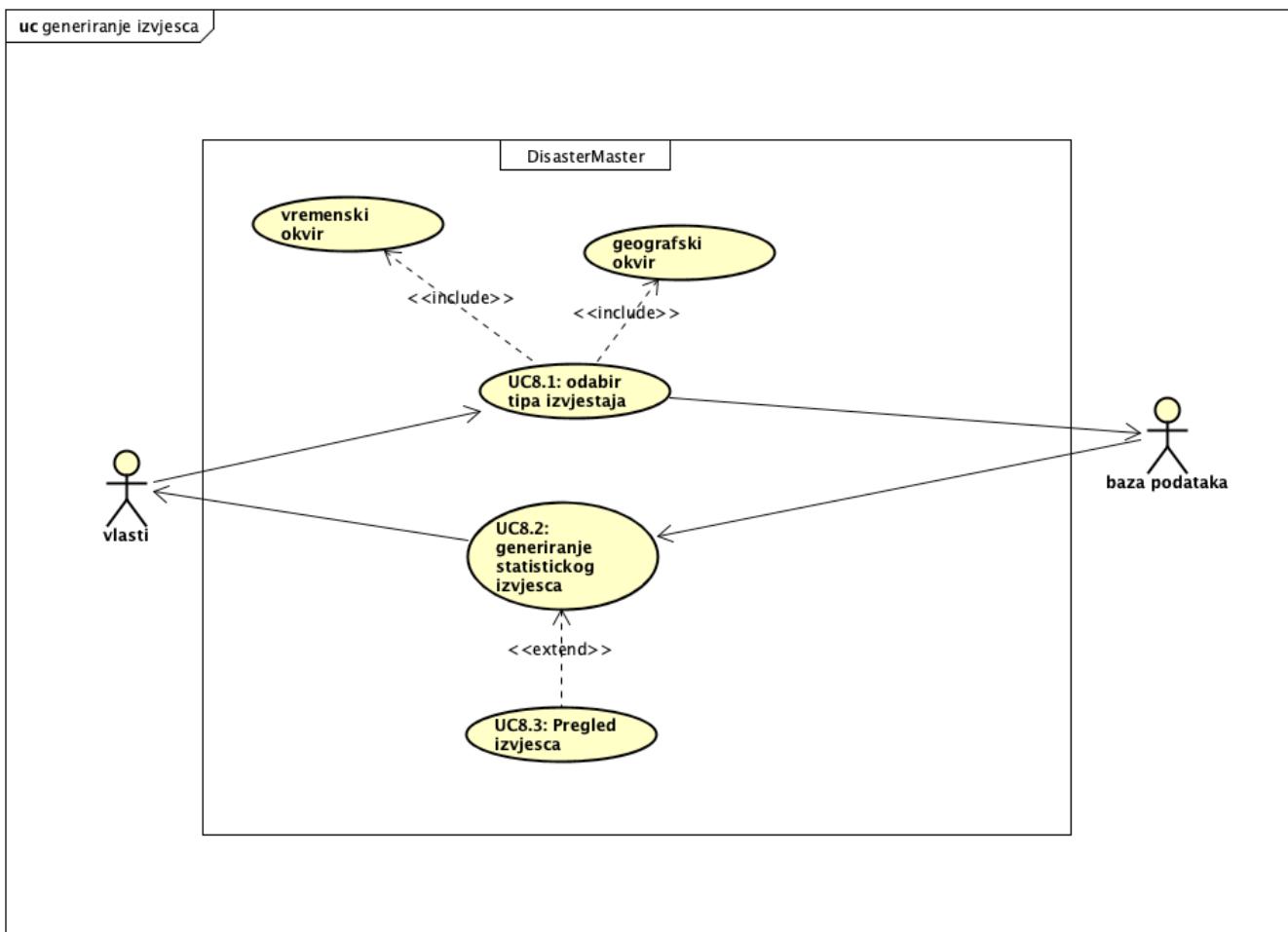
Dijagram procesa primanja obavijesti putem maila:



Dijagram procesa prijave nepogode:



Dijagram generiranja statistickog izvjesca:



Dijagram slanja informacija o resursima i najblizim sklonistima:

Razrada obrazaca

UC1: Prijava nepogode

- glavni sudionik: Korisnik
- cilj: Omoguciti prijavu nepogode u sustav
- sudionici: baza podataka, DisasterMaster web aplikacija
- preduvjet: -
- opis osnovnog tijeka:
 1. Korisnik odabire opciju za prijavom nepogode
 2. Sustav mu prikazuje formular za prijavu
 3. Korisnik ispunjava formular (vrsta nepogode, kratki opis, fotografija, lokacija)
 4. Sustav pohranjuje prijavu u bazu podataka
- Opis mogucih odstupanja:
 3. a) korisnik ne unese podatke potrebne za pohranjivanje prijave

UC2: Pregled informacija o resursima

- glavni sudionik: Korisnik
- cilj: Prikazati korisniku informacije o resursima, sklonistima i mjerama opreza
- sudionici: DisasterMaster web aplikacija, humanitarne organizacije
- preduvjet: humanitarne organizacije su objavile informacije
- opis osnovnog tijeka:
 1. Korisnik odabire opciju za prikazom informacija
 2. Sustav otvara stranicu koja prikazuje informacije
- Opis mogucih odstupanja:
 2. a) Humanitarne organizacije nisu objavile potrebne informacije

UC3: Primanje obavijesti

- glavni sudionik: Korisnik
- cilj: omoguciti korisniku primanje obavijesti putem emaila
- sudionici: mail server, DisasterMaster web aplikacija
- preduvjet: korisnik je registriran u sustav te je odabrao opciju za primanje obavijesti
- opis osnovnog tijeka:
 1. Korisnik se registrirao u sustav
 2. a) Korisniku putem mail dolazi automatska poruka o uspjesnoj registraciji
 3. Korisnik je odabrao opciju subscribe kojom odabire da mu se salju mailovi
 4. a) kada vlasti prihvate (validiraju) prijavu korisniku na mail stize obavijest o novoj nepogodi
- Opis mogucih odstupanja:

greska u sustavu zbog koje mail nebude poslan sustav ne uspije poslati obavijest zbog greske s mail serverom

UC4: Prijava u aplikaciju

- glavni sudionik: Korisnik
- cilj: omoguciti stvaranje korisnickog racuna
- sudionici: baza podataka, vanjski resursi (OAuth)
- preduvjet: -
- opis osnovnog tijeka:
 1. Korisnik odabire opciju prijave u aplikaciju

2. Korisnik odabire log-in(a) ili sign-in(b) 3.a) Ako je odabrao log-in upisuje korisnicko ime i lozinku 4.a) Aplikacija potvrduje log-in i koristenje se nastavlja sa korisnickim racunom
3.b) Ako je odabrao sign-in upisuje korisnicko ime, lozinku i mail 4.b) Dobiva potvrdu o kreiranom racunu i koristenje se nastavlja sa korisnickim racunom

- Opis mogucih odstupanja:

3.a) korisnik je upisao krivo ime i lozinku 3.b) doslo je do greske u bazi podataka te korisnicki racun nije dobro spremlijen 4.b) korisnik nije dobio potvrdu o kreiranom racunu zbog greske pri koristenju vanjskih resursa

UC5: Pregled interaktivne mape

- glavni sudionik: Korisnik
- cilj: koristenje interaktivne mape
- sudionici: vanjski resursi (OpenStreetMap)
- preduvjet: -
- opis osnovnog tijeka:

1. Korisnik ulazi u aplikaciju
2. Zoomiranjem moze vidjeti detaljnije lokacije
3. Koristenjem trazilice moze vidjeti odredenu lokaciju proizvoljne prezicnosti(drzava, grad, kvart, ulica)

- Opis mogucih odstupanja:

greska u radu sa vanjskim resursima (OpenStreetMap)

UC6: Pregled svih prijava

- glavni sudionik: Vlasti
- cilj: mogucnost pregleda svih prijava kako bi se utvrdile one vjerodostojne
- sudionici: Korisnici (kao osobe koje salju te prijave), baza podataka
- preduvjet: postoje prijave za određeno područje
- opis osnovnog tijeka:

1. Vlasti ulaze u aplikaciju kao administratori te zatrazuju pregled prijava
2. Aplikacija prikazuje sve prijave iz baze podataka

- Opis mogucih odstupanja:

zbog velikih broja prijava ili opterecenja sustava ucitavanje prijava moze trajati duze od ocekivanog prilikom pregleda prijava moze doci do prekida veze s bazom podataka i to onemoguci pregled prijava

UC7: Mijenjanje statusa prijave

- glavni sudionik: Vlasti
- cilj: prihvatanje vjerodostojnih prijava, te odbijanje nelegitimnih
- sudionici: baza podataka
- preduvjet: postoje prijave koje bi se mogle odbiti/prihvati
- opis osnovnog tijeka:
 1. Vlasti odabiru odredenu prijavu te joj mijenjaju status
 2. Aplikacija registrira promjenu statusa prijave te se tako sprema u bazu podataka
 3. Prihvacene prijave se prikazuju kao pinovi na interaktivnoj mapi
- Opis mogucih odstupanja:

greska u radu sustava zbog koje se odrene prijave ne registriraju sa dobrim statusom

UC8: Generiranje statistickog izvjesca

- glavni sudionik: Baza podataka
- cilj: generiranje statistickog izvjesca na temelju prijava u odredenu vremenskom periodu ili području
- sudionici: Vlasti
- preduvjet: postoji dovoljno prijava kako bi se kreiralo izvjesce
- opis osnovnog tijeka:
 1. Vlasti zatraze generiranje izvjesca
 2. Aplikacija im nudi opciju filtriranja po određenim parametrima kako bi se bolje specifiralo izvjesce (vremenski period i lokacija)
 3. Vlasti odabiru parametre za generiranje izvjesca
 4. Iz baze podataka se generira izvjesce te prikazuje korisniku(vlastima)
- Opis mogucih odstupanja:

krivo generirano izvjesce

UC9: Objava informacija o resursima

- glavni sudionik: Humanitarne organizacije
- cilj: objavljivanje korisnih informacija u slučaju nepogode
- sudionici: baza podataka
- preduvjet: postoje prijave na temelju kojih su humanitarne organizacije odlucile postaviti informacije
- opis osnovnog tijeka:
 1. Humanitarne organizacije na temelju pregleda prijava odlucuju koje informacije će objaviti

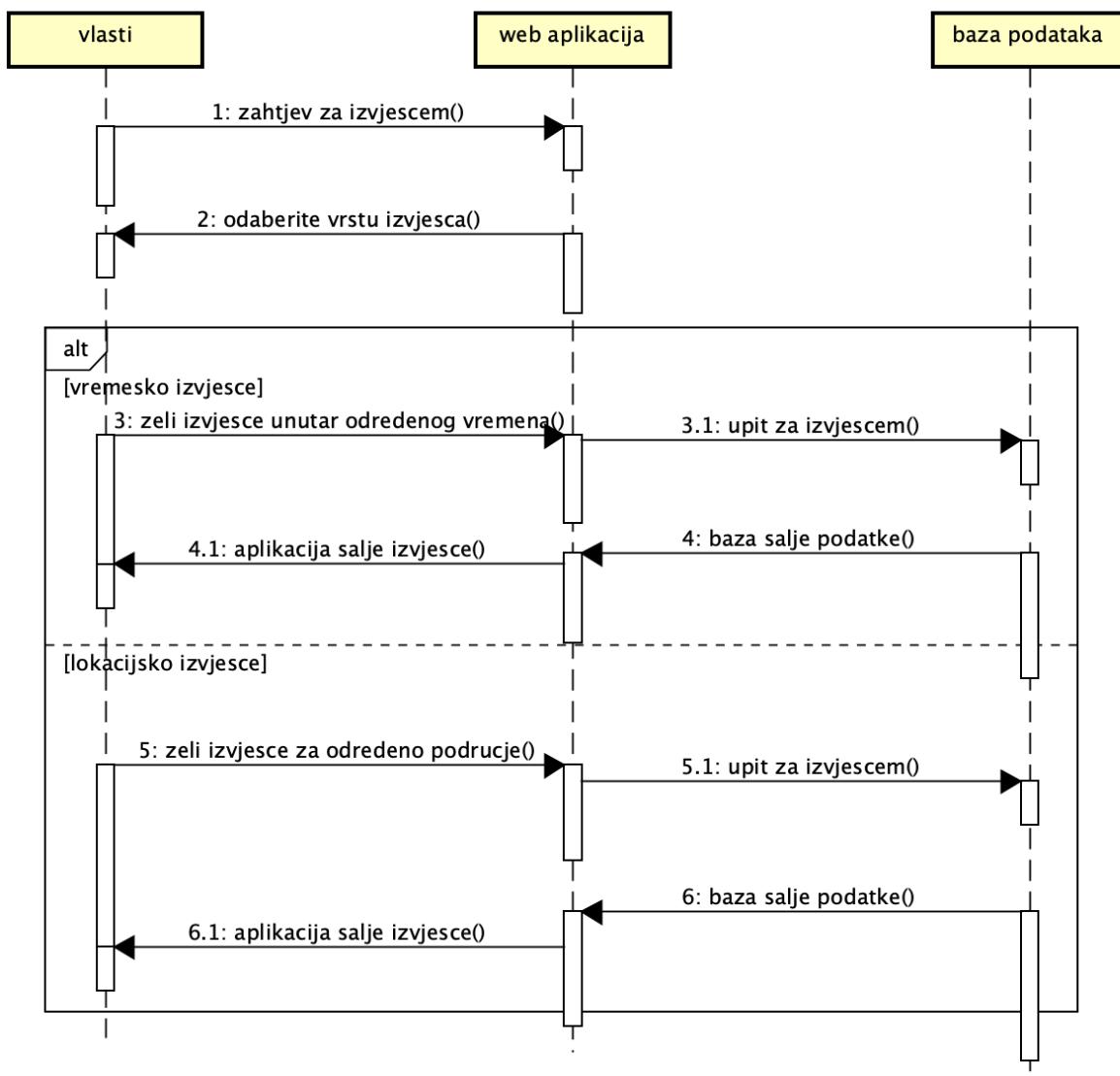
2. Odabiru opciju postavljanja informacija
3. Sustav uspjesno registrira njihove obavijesti
4. Sustav objavljuje informacije na glavnoj stranici aplikacije

- Opis mogucih odstupanja:

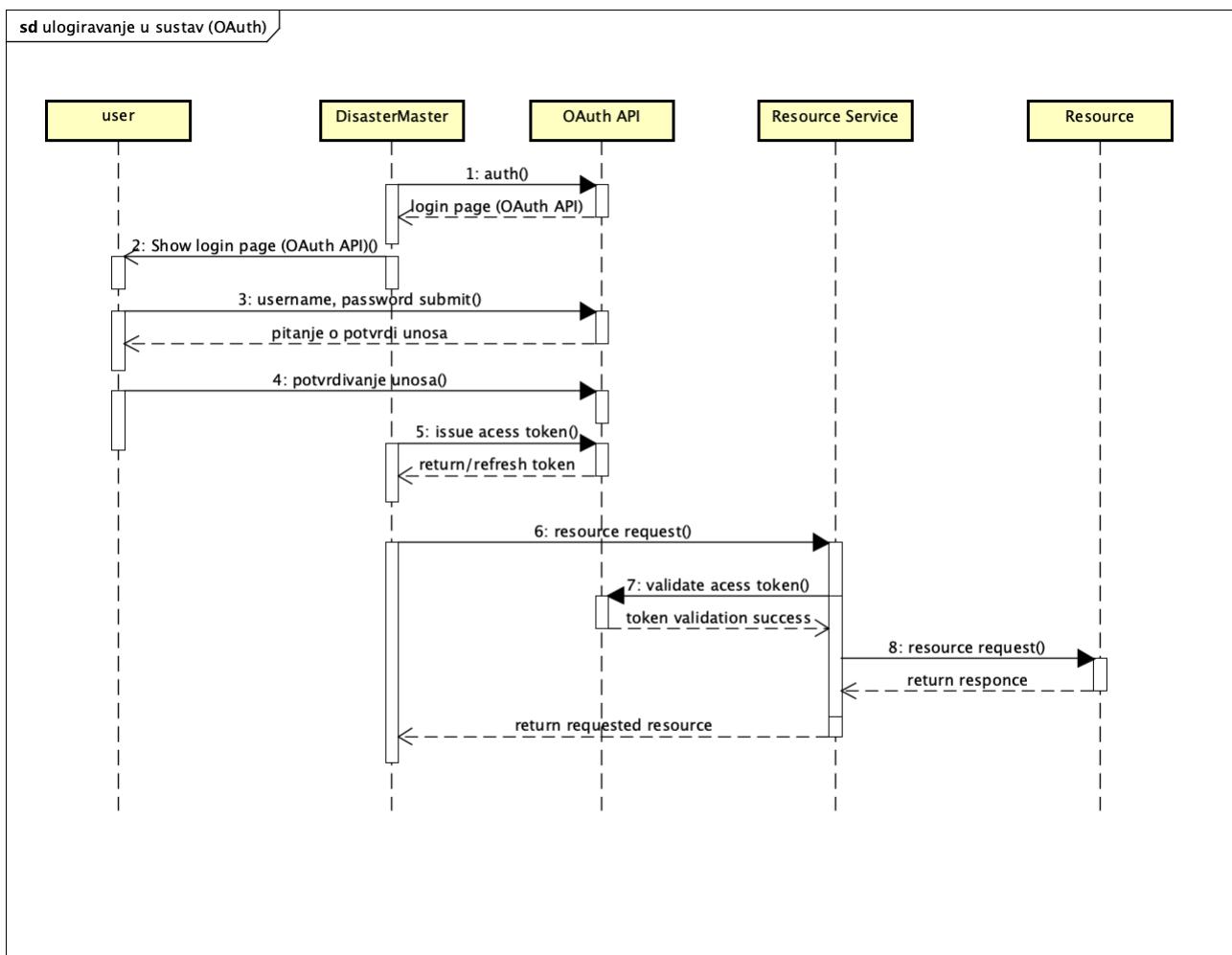
greska u sustavu zbog koje se informacije ne prikazuju

Sekvencijski dijagrami

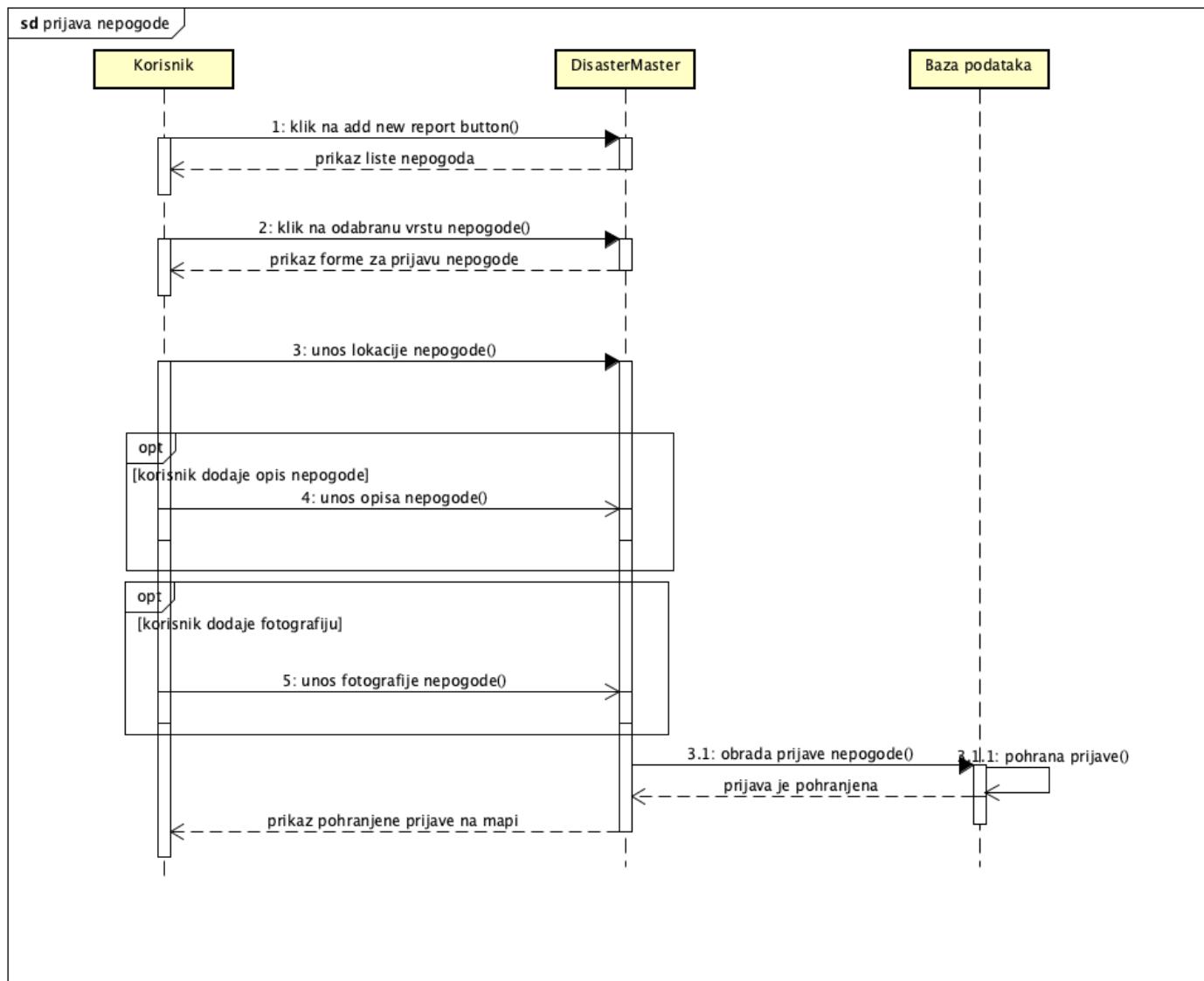
generiranje izvjesca:



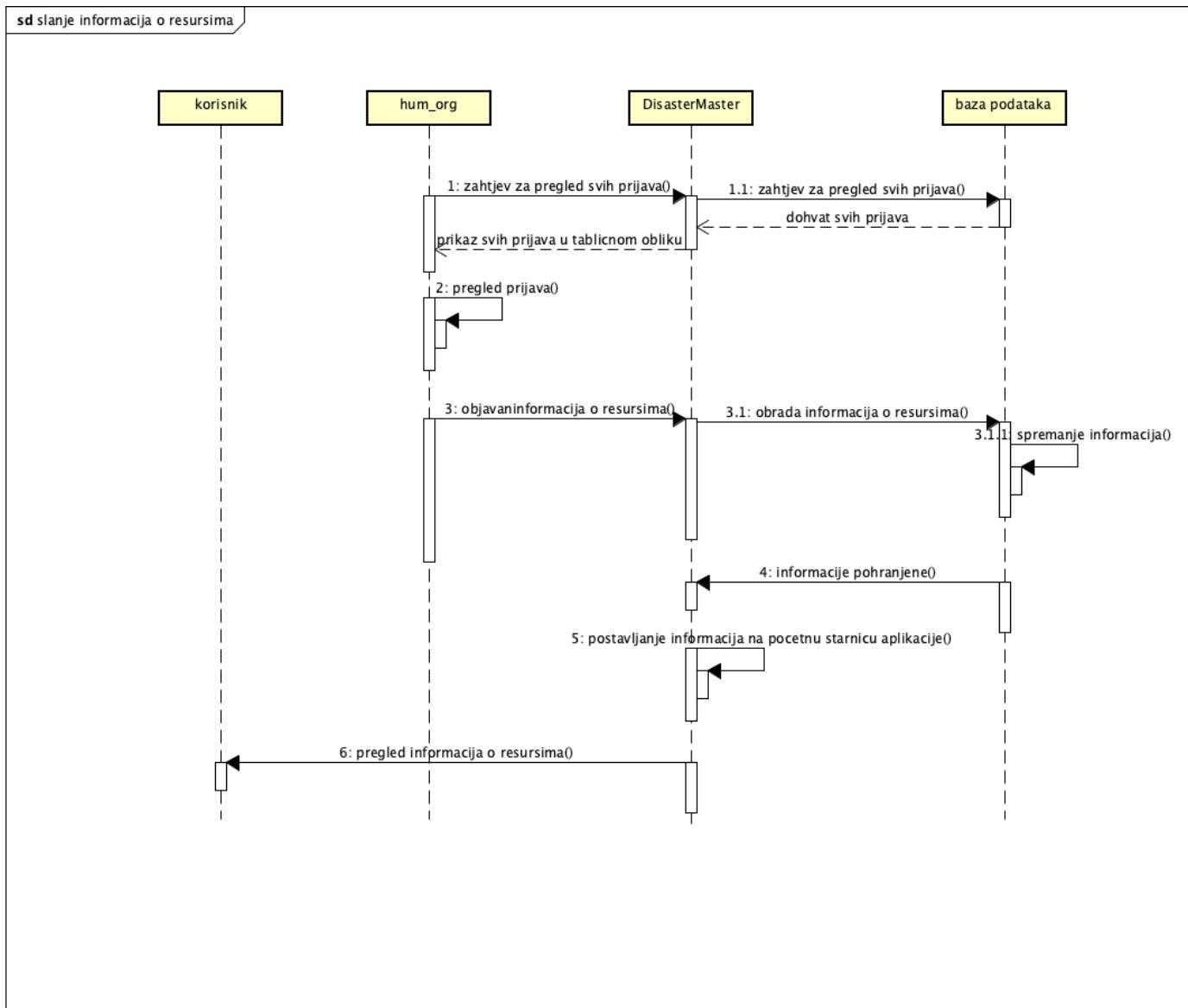
log-in pomocu OAuth:



prijava nepogode:



slanje informacija o resursima od strane humanitarnih organizacija:



+ Add a custom footer

▼ Pages 12

Find a page...

- ▶ [Home](#)
- ▶ [1. Opis projektnog zadatka](#)
- ▶ [2. Analiza zahtjeva](#)
- ▼ [3. Specifikacija zahtjeva sustava](#)
 - Aktori
 - Obrasci uporabe
 - Opis obrazaca uporabe
 - 1. Visokorazinski dijagram svih funkcionalnosti aplikacije

- 2. Vrste korisnika
 - 3. Dijagrami glavnih funkcionalnosti
- Razrada obrazaca
- Sekvencijski dijagrami

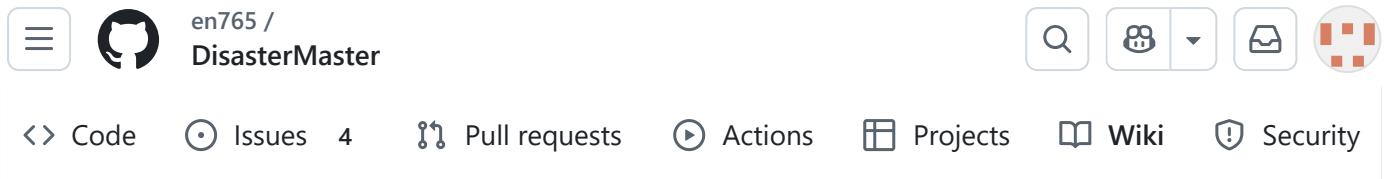
- ▶ [4. Arhitektura i dizajn sustava](#)
- ▶ [5. Arhitektura komponenata i razmještaja](#)
- ▶ [6. Ispitivanje programskog rjesenja](#)
- ▶ [7. Tehnologija za implementaciju aplikacije](#)
- ▶ [8. Upute za pustanje u pogon](#)
- ▶ [9. Zaključak i budući rad](#)
- ▶ [Popis literature](#)
- ▶ [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>





4. Arhitektura i dizajn sustava

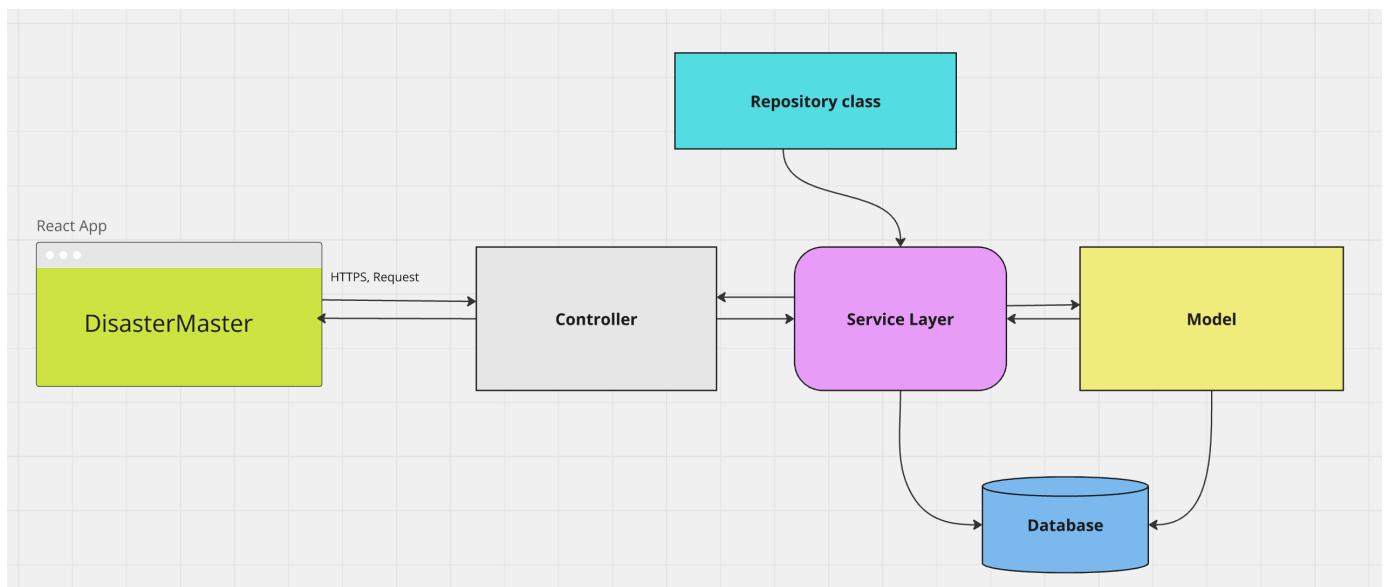
[Edit](#)
[New page](#)
[Jump to bottom](#)

Elena Neseck edited this page 5 days ago · [8 revisions](#)

Arhitektura sustava

Ovim poglavljem opisana je arhitektura aplikacije za prijavu i pracenje vremenskih nepogoda DisasterMaster. Odabrani stil arhitekture je mikroservis zbog potrebe za skalabilnosti i neovisnim razvojem ključnih funkcionalnosti. Na frontendu koristen je Reach.js, dok se za implementaciju backenda koristi Spring Boot. Koristeni su i sigurnosni potokoli koji su navedeni kasnije.

Primjer dijagrama Spring Boot flow arhitekture:



Spring Boot arhitektura prati slojni dizajn s nekoliko glavnih komponenti: Controller layer (upravlja HTTP zahtjevima), Service layer (poslovna logika koja implementira razne funkcionalnosti), Repository Layer (upravlja pristupom podatcima i njihovom pohranom), Model layer (definira podatkovne objekte)

Opis arhitekture

- Stil arhitekture: Arhitektura mikroservisa pogodna je razvoj aplikacije jer omogucava implementiranje svake funkcionalnosti kao zasebnog servisa. Ovo omogućava prilagođeno skaliranje specifičnih dijelova sustava ovisno o opterećenju, kao što je povećanje resursa za prijave nepogoda tijekom krize. Nadalje svaki servis može biti razvijen, implementiran i testiran odvojeno, što omogucava brzu reakciju na korisnicke zahtjeve. Uz model mikroservisa, arhitektura sadrži i neke elemente slojevite arhitekture:
 - Klijentski sloj: prikazuje korisnicko sučelje kroz koje korisnici komuniciraju s aplikacijom (React.js)
 - Sloj servisa: implementira sve funkcionalnosti aplikacije kao što su prijava nepogode, obavijesti i statistika (Spring Boot, REST API)
 - Sloj integracije: olaksava komunikaciju između mikroservisa. Korištenjem API gatewaya osigurava se pouzdanost i otpornost sustava, čak i pri povećanom opterećenju. (Amazon API Gateway, HTTP)
 - Sloj podataka: omogućava pohranu i obradu podataka te brzu obradu informacija (PostgreSQL, AWS S3)
 - Sloj infrastrukture i sigurnosti: upravljanje autentifikacijom, enkripcija i zastita podataka (OAuth 2.0, JWT za provjeru identiteta korisnika)
 - Sloj obavijesti: osigurava slanje obavijesti putem razlicitih kanala (u našem slučaju maila) kako bi se promptno informirali korisnici (Mail Server, SMTP, HTTPS)
- Mrežni protokoli: Za komunikaciju između klijenta i poslužitelja koristiti će se HTTP/HTTPS implementirani u Spring Bootu. Prijenos podataka između front i backenda u JSON formatu odvijet će se putem REST API-ja. OAuth omogućava sigurnu prijavu u sustav putem tokena uz koji će se koristiti i JWT. Sto se tice protokola slanja obavijesti integriran će biti SMTP koristeci Spring Email Starter.
- Globalni upravljački tok:
 1. Početna interakcija: Korisnik (npr. građanin, vlast) pristupa aplikaciji, registrira se ili se prijavljuje.
 2. Unos podataka: Korisnik prijavljuje nepogodu ili traži informacije.
 3. Obrada podataka: Podaci se pohranjuju u bazu podataka i analiziraju.
 4. Obavijesti i akcije: Ako je prijava odobrena, obavijesti se šalju korisnicima putem maila.
 5. Povratna informacija: Korisnik dobiva potvrdu ili povratnu informaciju o statusu prijave ili akciji.

Obrazloženje odabira arhitekture

Odabir mikroservisne arhitekture temelji se na nekoliko ključnih cimbenika: fleksibilnost, skalabilnost i održivost. Mikroservisi omogućuju podjelu aplikacije na manje, samostalne jedinice koje se mogu razvijati, testirati i implementirati neovisno, što olakšava održavanje i unapređivanje aplikacije. Također, mikroservisna arhitektura omogućuje horizontalnu skalabilnost jer je moguće skalirati pojedine servise prema potrebi, čime se povećava efikasnost u upravljanju resursima.

Principi oblikovananja kao sto je povecanje kohezije i smanjenje povezanosti, pomogli su u stvaranju mikroservisa koji komuniciraju putem jasno definiranih API-ja i protokola cime se smanjila meduovisnost i povećala fleksibilnost unutar sustava.

Kao alternativu razmatrali smo klijent-posluzitelj arhitekturu koja je jednostavna za implementaciju, ali postavlja ogranicenja kada je rjec o skalabilnosti i fleksibilnosti. Klijent-posluzitelj arhitektura je prikladna za manje sustave, dok mikroservisna pruza bolje rjesnje za veće, dinamicne aplikacije poput DisasterMastera.

Organizacija sustava na visokoj razini

- Korisnicko sucelje implementirano React.jsom korisniku omogucava komunikaciju s ostatom aplikacije (mikroservisa) putem API gatewaya
- Mikroservisi su manji moduli spicificnih funkcionalnosti:
 - upravljanje korisnicima, prijava nepogoda, primanje/slanje obavijesti, pregled informacija o resursima, generiranje izvjestaja, interaktivna mapa
 - Baza podataka je PostgreSQL ---- Dodajem kasnije kad deployamo aplikaciju
 - Datoteci sustav ---- Dodajem kasnije kad deployamo aplikaciju

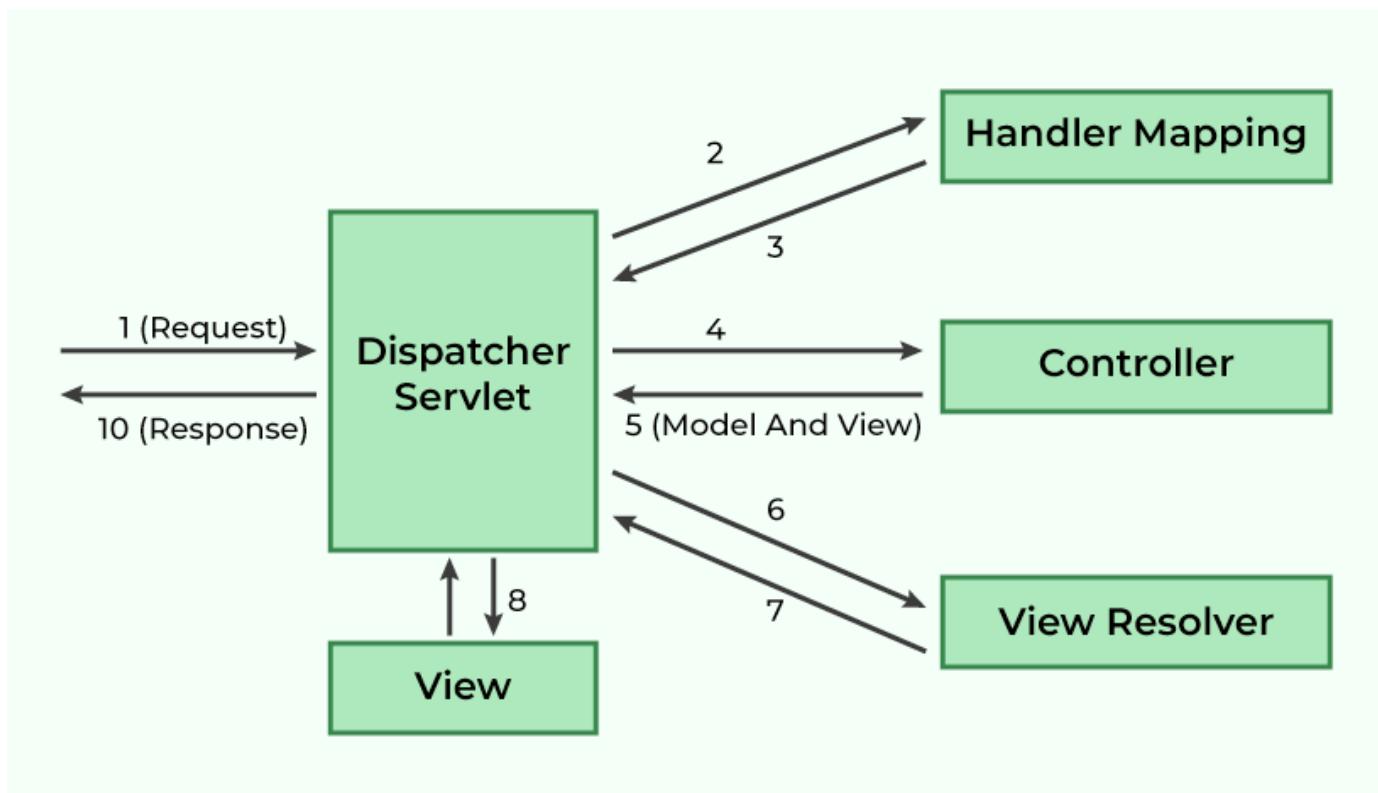
Organizacija aplikacije

U arhitekturi aplikacije koristeci MVC framework u Spring Bootu te React.js za frontend, aplikacija sadrži tri glavna sloja:

1. Frontend (View): Implementiran u React.js, frontend pruža korisničko sučelje koje komunicira s backendom putem HTTP zahtjeva. Frontend koristi REST API kako bi dobio podatke, prikazao ih i omogućio interakcije korisnika, kao što su prijava nepogoda ili ažuriranje profila.
2. Kontroler (Controller): U Spring Boot-u kontroleri obrađuju zahtjeve iz frontenda. Svaki zahtjev se prima putem API endpointa, a kontroler zatim poziva odgovarajuće servise (poslovnu logiku) kako bi obradio podatke. Nakon obrade, kontroler vraća odgovor natrag na frontend.

3. Poslovna logika i model (Service i Model): Sloj poslovne logike obuhvaća servise koji obrađuju zahtjeve iz kontrolera, izvršavajući poslovne operacije kao što su validacija podataka, autentifikacija ili manipulacija podacima. Model predstavlja strukture podataka koje se koriste unutar aplikacije i pohranjuju u bazi podataka. Svaki model preslikava strukture baze podataka pomoću JPA ili drugih ORM alata.
4. Baza podataka (Repository Layer): Pomoću JPA ili drugih ORM tehnologija, Spring Boot omogućava komunikaciju s bazom podataka. Repozitoriji komuniciraju s poslovnom logikom, a svaki model podataka može biti povezan s jedinstvenom bazom podataka ako koristimo arhitekturu mikrousluga.

Dijagram mvc arhitekture u spring bootu:



Baza podataka

Opis tablica

Location

Atribut	Tip podatka	Opis varijable
locationId	DOUBLE	identifikator lokacije za prijave, PK

Atribut	Tip podatka	Opis varijable
name	STRING	ime grada, ulice, mjesta prijave
latitude	DOUBLE	Jedinstveni identifikator lokacije
longitude	DOUBLE	Jedinstveni identifikator lokacije
address	STRING	vlastiti atribut lokacije
city	STRING	vlastiti atribut lokacije
zipCode	INT	vlastiti atribut lokacije

Photo

Atribut	Tip podatka	Opis varijable
photoid	DOUBLE	identifikator fotografije priložene uz prijavu, PK
reportId	DOUBLE	identifikator prijave uz koju je vezana fotografija, FK
photoURL	STRING	identifikator fotografije
uploadedAt	DATETIME	vlastiti atribut, vrijeme prijave fotografije
userId	DOUBLE	identifikator korisnika koji prijavljuje nepogodu, FK

Report

Atribut	Tip podatka	Opis varijable
reportId	DOUBLE	Jedinstveni identifikator prijave nepogode, PK
userId	DOUBLE	identifikator korisnika koji prijavljuje nepogodu , FK, PK
disasterType	STRING	vlastiti atribut, vrsta tipa nepogode
status	BOOLEAN	vlastiti atribut, status prijave (odbijena ili prihvaćena)
locationId	DOUBLE	identifikator lokacije vezane uz prijavu, FK
descriptionOfDisaster	STRING	vlastiti atribut, tekstualni opis prijave
createdAt	DATETIME	vlastiti atribut, vrijeme prijave nepogode
photoid	DOUBLE	identifikator fotografije priložene uz prijavu, FK

Information

Atribut	Tip podatka	Opis varijable
informationId	INT	Jedinstveni identifikator objave o informacijama, PK
userId	DOUBLE	identifikator korisnika koji objavljuje informacije, FK
informationType	STRING	vlastiti atribut vrste tipa informacije (sklonište, resurs)
createdAt	DATETIME	vlastiti atribut vremena objave informacije
locationId	DOUBLE	identifikator lokacije vezane uz objavu informacije
descriptionOfInformation	STRING	vlastiti atribut, tekstualni opis informacije

Hum_org

Atribut	Tip podatka	Opis varijable
userId	DOUBLE	Jedinstveni identifikator korisnika, FK, PK
orgName	STRING	vlastiti atribut imena humanitarne organizacije

Citizen

Atribut	Tip podatka	Opis varijable
userId	DOUBLE	Jedinstveni identifikator korisnika, FK, PK
firstName	STRING	vlastiti atribut, ime korisnika
lastName	STRING	vlastiti atribut, prezime korisnika
isAnonymous	BOOLEAN	vlastiti atribut, status anonimnosti korisnika (je li korisnik ulogiran u aplikaciju ili ne)

User

Atribut	Tip podatka	Opis varijable
userId	DOUBLE	Jedinstveni identifikator korisnika, PK

Atribut	Tip podatka	Opis varijable
roleId	INT	identifikator tipa uloge korisnika
password	STRING	vlastiti atribut šifra korisničkog računa
e-mail	STRING	vlastiti atribut, email korisnika (koji koristi za prijavu u aplikaciju)
roleId	INT	FK, PK

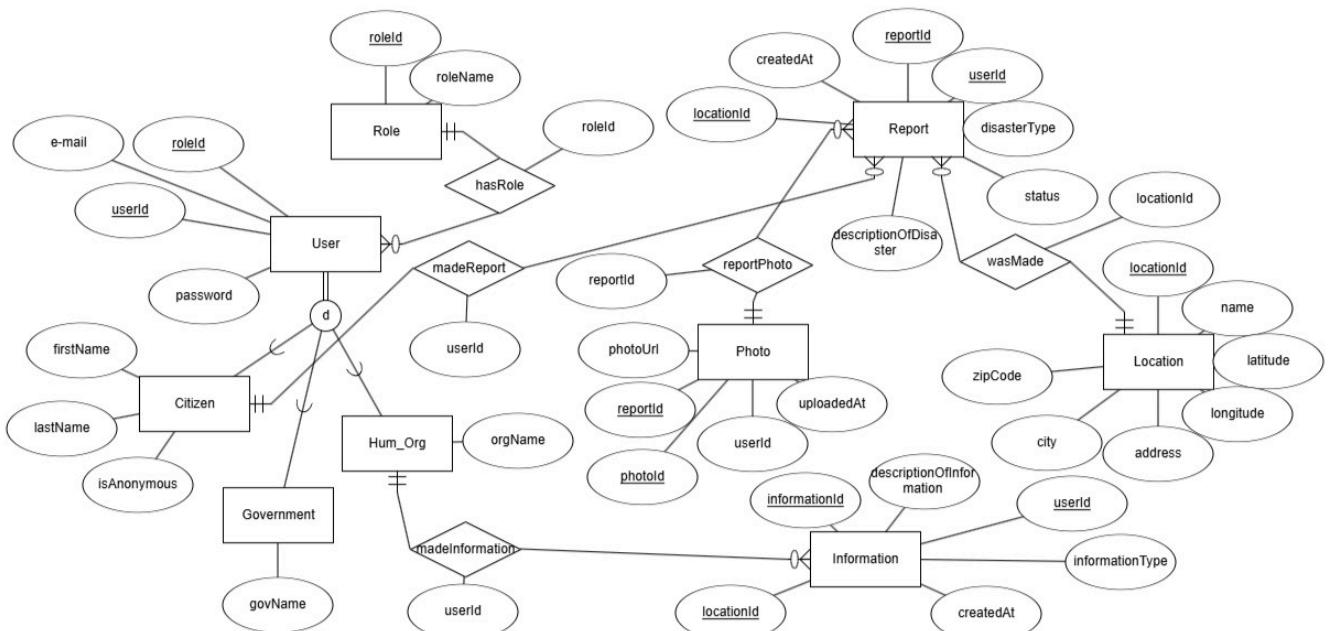
Government

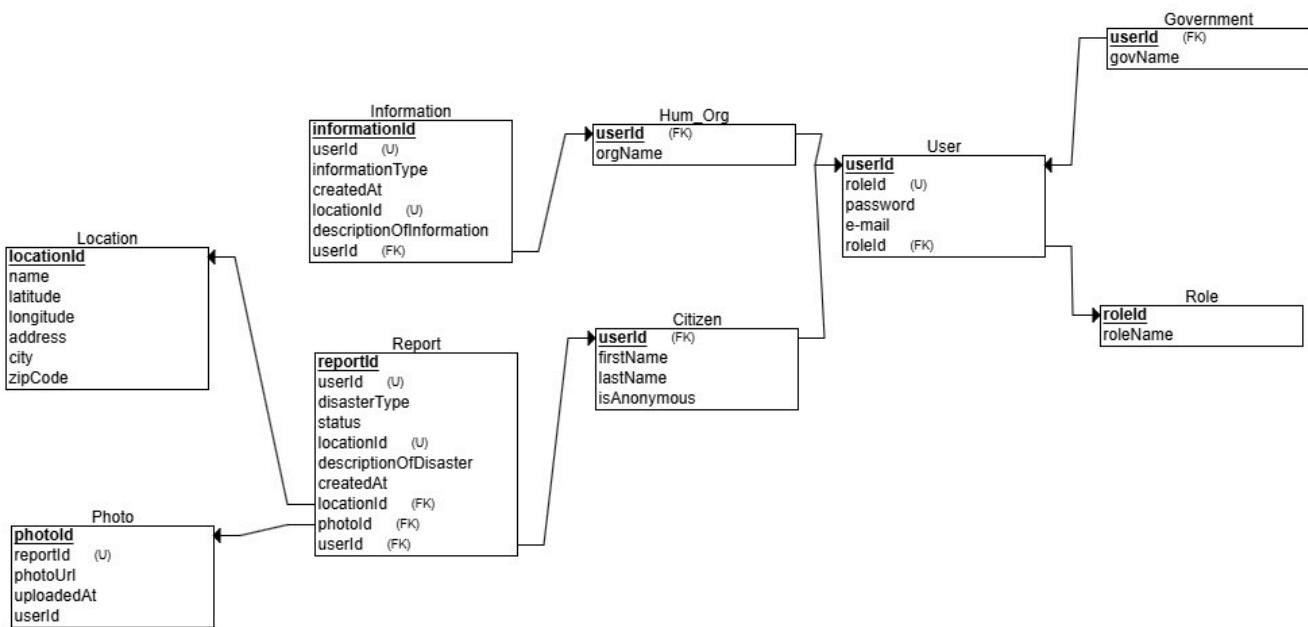
Atribut	Tip podatka	Opis varijable
userId	DOUBLE	Jedinstveni identifikator korisnika, PK, FK
govName	STRING	vlastiti atribut imena nadležnog tijela

Role

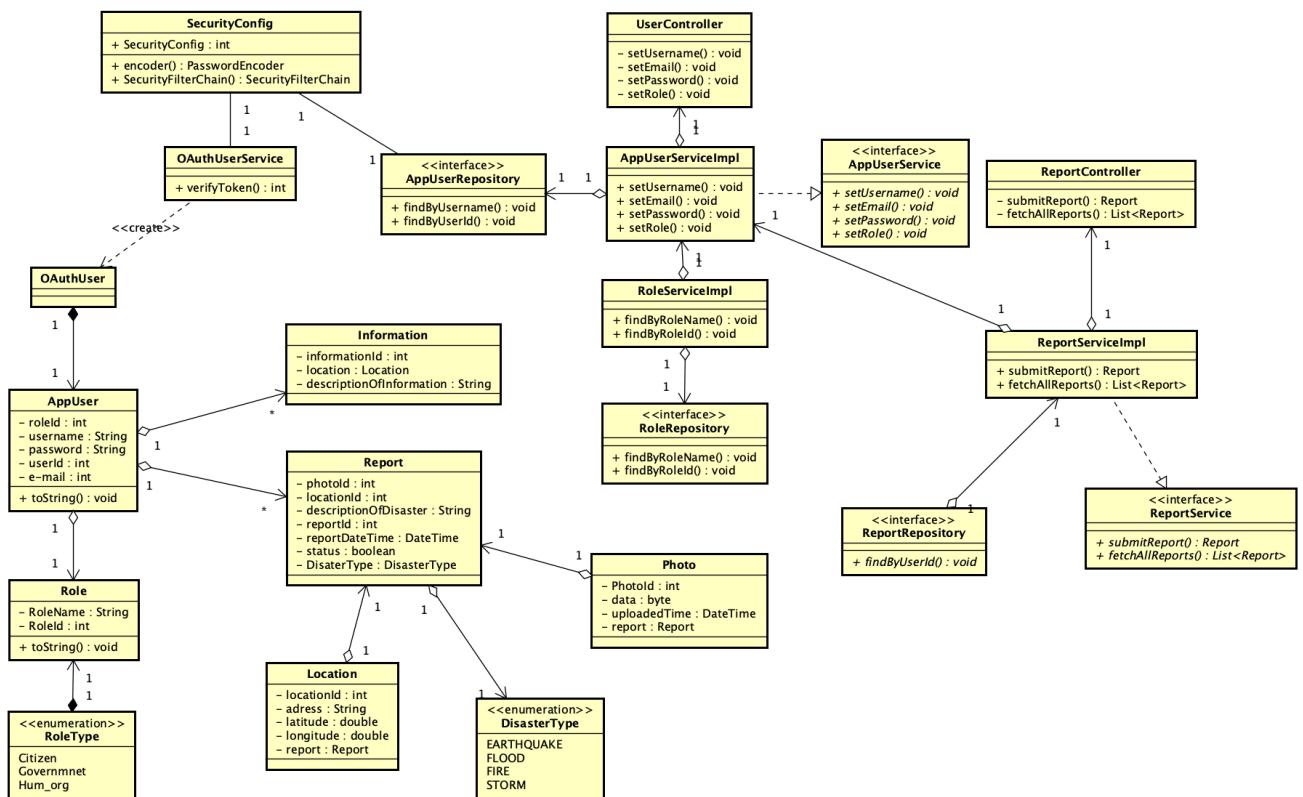
Atribut	Tip podatka	Opis varijable
roleId	INT	jedinstvani identifikator uloge korisnika, PK
rolename	STRING	vlastiti atribut imena uloge korisnika

Dijagram baze podataka



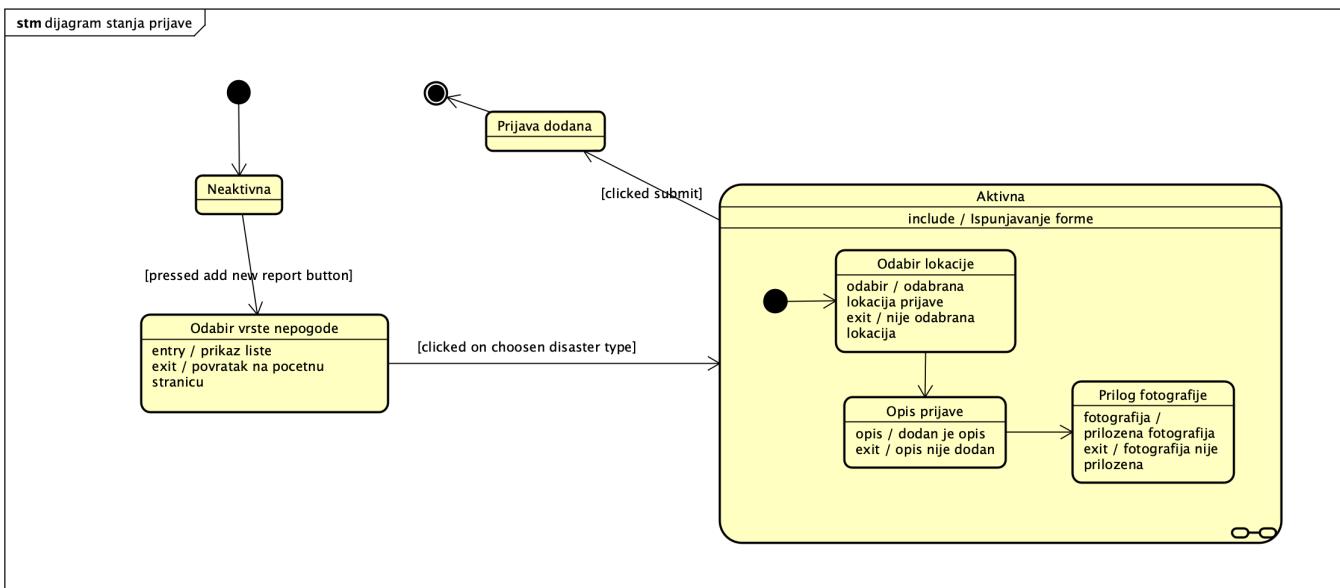


Dijagram razreda



Dijagram stanja

dijagram stanja prijave nepogode



Stanja dijagrama su: Neaktivna, Odabir vrste napogode, Aktivna/ispunjavanje forme (Odabir lokacije, Opis prijave, Prilog fotografije), Prijava dodana, Zavrsno stanje (prijava dodana je zavrsno stanje prijave nepogode)

Korisniku se klikom na gumb 'Add new weather reports' prikazuje lista nepogodi

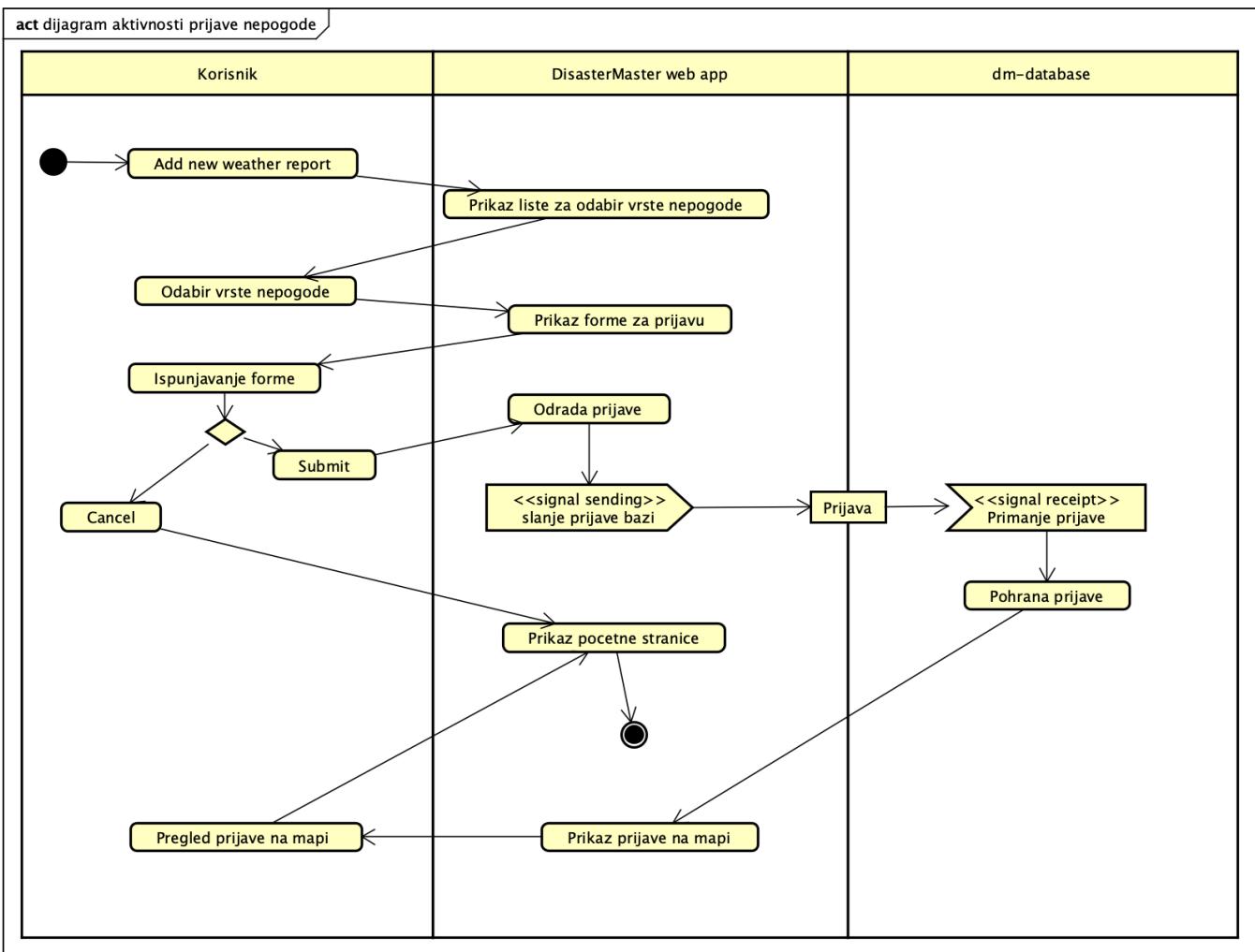
Klikom na odredenu vrstu nepogode prikazuje se forma za prijavu

Ispunjavanjem svih elemenata forme i klikom na gumb 'submit' prijava se sprema te prikazuje na mapi pocetne stranice

Klikom na gumb 'cancel' u bilo kojem trenutku korisnika se vracanje na pocetnu stranicu, te njegova prijava nije spremljena

Dijagram aktivnosti

dijagram aktivnosti prijave nepogode



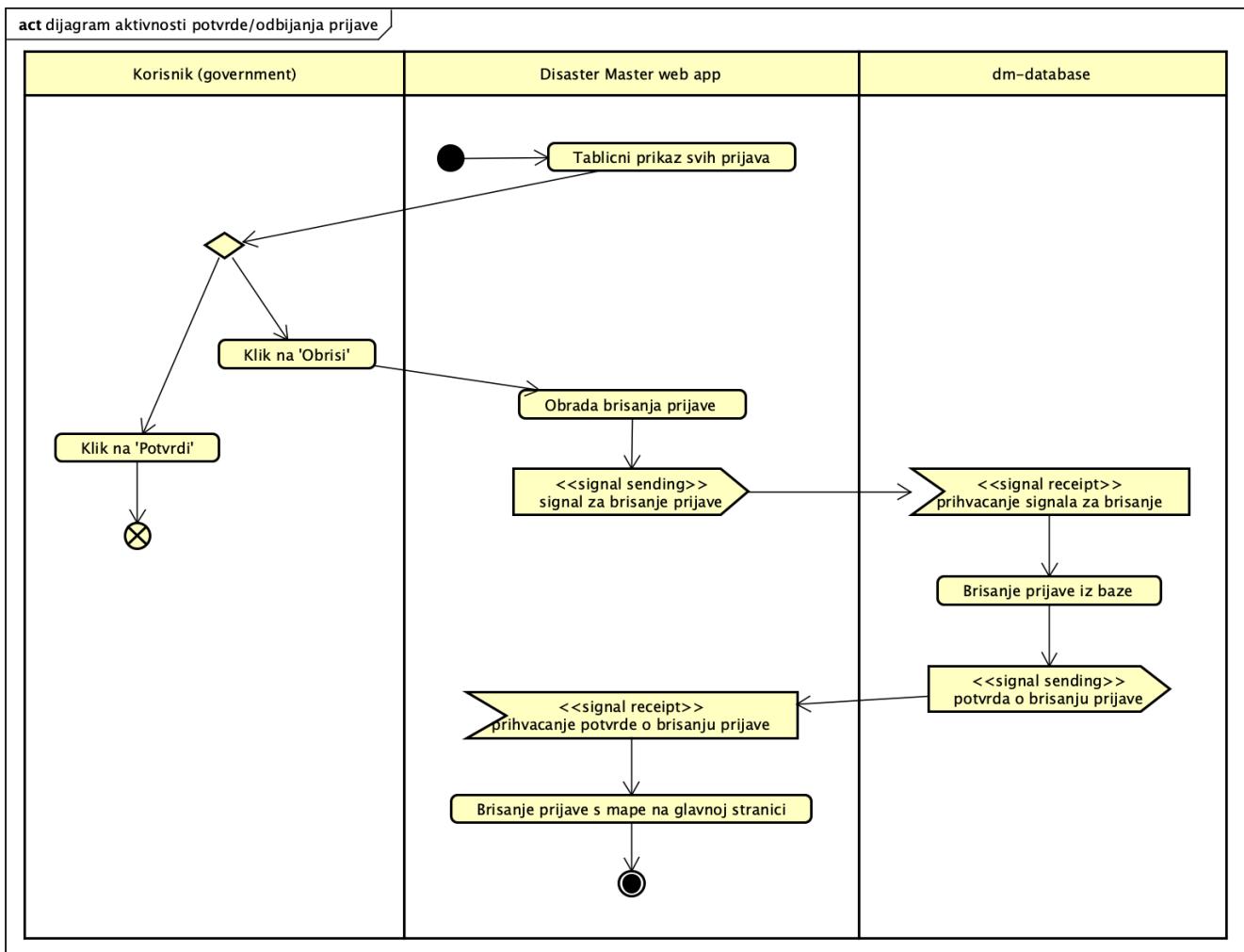
Dijagram aktivnosti prikazuje tok akcija tijekom procesa prijave nove nepogode

Akori su: Korisnik, Disaster Master web aplikacija i baza podataka

Na pocetni cvor se nadovezuje na klik gumba za prijavu nove nepogode cime zapocinje proces prijave nepogode

Završni cvor se nadovezuje na prikaz pocetne stranice cime zavrsava proces prijave nepogode (i u slučaju submita i u slučaju canclea)

dijagram aktivnosti prihvatanja/odbijanja prijave



Dijagram aktivnosti prikazuje proces prihvacanja/odbijanja prijave korisnika od strane vlasti

Aktori su: Korisnik (governmnet), Disaster Master web aplikacija, baza podataka

Proces zapicnje tablicnim prikazom svih prijava korisniku s ulogom government

Proces završava ili zavrsetkom procesa toka kod prihvacanja prijave (prijava samo ostaje u bazi i prikazana na mapi) ili završnim cvorom nakon brisanja prijave

+ Add a custom footer

▼ Pages 12

Find a page...

► [Home](#)

► [1. Opis projektnog zadatka](#)

► [2. Analiza zahtjeva](#)

▶ [3. Specifikacija zahtjeva sustava](#)

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

Arhitektura sustava

Primjer dijagrama Spring Boot flow arhitekture:

[Opis arhitekture](#)

Obrazlozenje odabira arhitekture

Organizacija sustava na visokoj razini

Organizacija aplikacije

Baza podataka

Opis tablica

Location

Photo

Report

Information

Hum_org

Citizen

User

Government

Role

Dijagram baze podataka

Dijagram razreda

Dijagram stanja

dijagram stanja prijave nepogode

Dijagram aktivnosti

dijagram aktivnosti prijave nepogode

dijagram aktivnosti prihvacanja/odbijanja prijave

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

▶ [6. Ispitivanje programskog rjesenja](#)

▶ [7. Tehnologija za implementaciju aplikacije](#)

▶ [8. Upute za pustanje u pogon](#)

▶ [9. Zaključak i budući rad](#)

▶ [Popis literature](#)

▶ [Prikaz aktivnosti grupe](#)

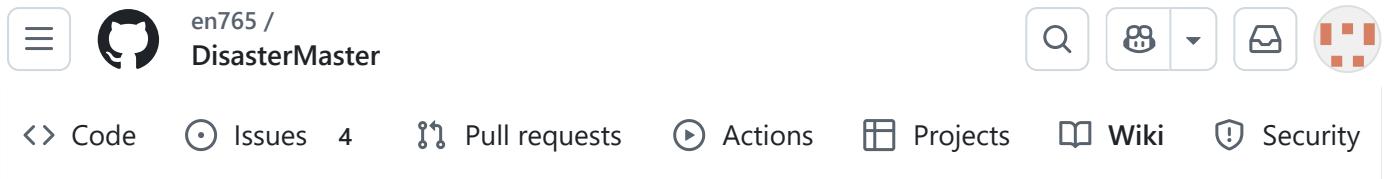


+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>



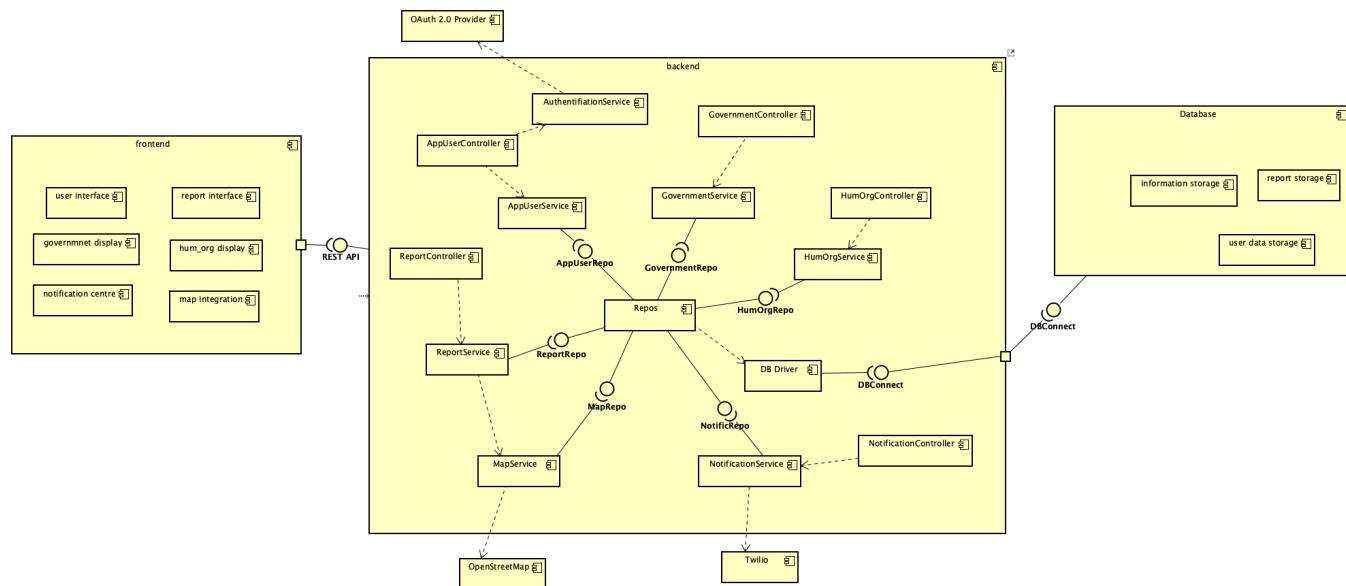


5. Arhitektura komponenata i razmještaja

[Edit](#)
[New page](#)
[Jump to bottom](#)

Elena Neseck edited this page 5 days ago · [6 revisions](#)

Dijagram komponenti



Komponente:

- frontend: predstavlja sva korisnicka sucelja pristuna pri koristenju aplikacije (user, report, government, hum_org, notification, map)
- backend: komponenta koja prestavlja okolinu za izvodenje svih logickih opracija, povezivanje s bazom podataka, obraduje korisnicke zahtjeve
- Database: predstavlja bazu podataka aplikacije, sadrzi spremnike za sve skupine podataka (user data, reports, hum_org information)
- Vanjski servisi: Twilio, OpenStreetMap i OAuth koji su integrirani u sustav
- Podkomponente backenda:

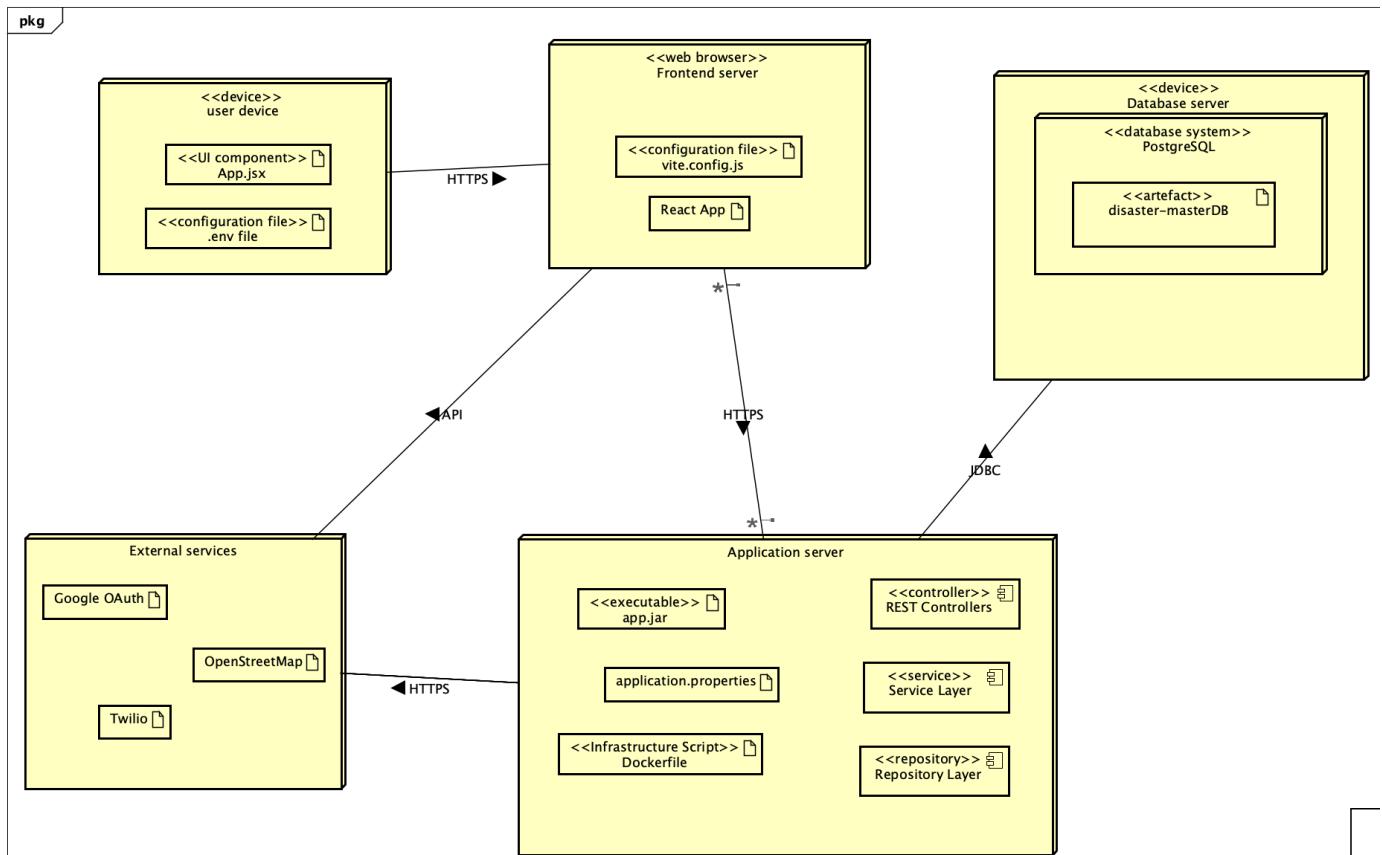
Controlleri: ReportController (obraduje poslane prijave nepogoda) i NotificationController (salje obavijesti korisnicima), AppUserController (obraduje prijave i ostale funkcionalnosti korisnika), GovernmentController (obraduje funkcije administratora/vlasti), HumOrgController (obraduje funkcije humanitarnih organizacija kao korisnika)

Services: Opcenito provjeravaju ispravnost podataka poslanih na controllere, povezuju backend s vanjskim servisima te pozivaju repository layer kako bi spremili provjerene podatke u bazu (ReportService, MapService, NotificationService, AppUserService, GovernmentService, HumOrgService, AuthenticationService)

Repos: predstavlja repository layer koji provodi CRUD operacije nad podatcima te sluzi kao poveznica s bazom

DB Driver: sluzi kao pomoc pri uspostavljanju veze s bazom podataka

Dijagram razmjestaja



- User device node:

App.jsx - predstavlja frontend komponentu korisnickog sucelja

.env file - varijable koje koristi korisnicko racunalo pri izvodenju operacija (API keys, URLs, ...)

- Frontend server node: prijelaz izmedu korisnika i backenda sustava

vite.config.js - konfiguracija za runnanje i buildanje React aplikacije

React App - kompajljana verzija frontend dijela aplikacije

- Application server node: backend jedinica (API zahjtevi, logicke operacije, operacije nad bazom)

app.jar - kompajljana backend aplikacija (Spring Boot)

application.properties - konfiguracijski file za backend operacije (API zahtjevi, povezivanje s bazom)

Dockerfile - skripta za deployment backend aplikacije

Controller Layer - API zahtjevi iz frontenda

Service Layer - logicke operacije

Repository Layer - komunikacija s bazom podataka

- Database server node:

disaster-masterDB - predstavlja PostgreSQL schemu koja sadrzi podatke za aplikaciju

- External service node:

Google OAuth, OpenStreetMap, Twilio

+ Add a custom footer

Pages 12

Find a page...

▶ [Home](#)

▶ [1. Opis projektnog zadatka](#)

▶ [2. Analiza zahtjeva](#)

▶ [3. Specifikacija zahtjeva sustava](#)

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

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

Dijagram komponenti

Komponente:

Dijagram razmjestaja

- ▶ [6. Ispitivanje programskog rjesenja](#)
- ▶ [7. Tehnologija za implementaciju aplikacije](#)
- ▶ [8. Upute za pustanje u pogon](#)
- ▶ [9. Zaključak i budući rad](#)
- ▶ [Popis literature](#)
- ▶ [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>





Code

Issues 4

Pull requests

Actions

Projects

Wiki

Security

6. Ispitivanje programskog rjesenja

[Edit](#)[New page](#)[Jump to bottom](#)Elena Neseck edited this page 19 hours ago · [7 revisions](#)

Ispitivanje sustava i komponenti

Ispitivanje funkcionalnosti aplikacije odradeno je koristenjem Selenium WebDrivera te jUnit testova. Ispitivanje komponenti odradeno je skriptama napisanim za Selenium WebDriver kao i rucno testiranje za odredene slucejeve

ID	Opis	Koraci	Podatci	Ocekivani izlaz	Stvarni izlaz	Status
1	Provjera unosa postojeće lokacije (Redovan slučaj)	1. U search tab mape upisati postojecu lokaciju 2. Kliknuti 'search' button	Lokacija: 'London'	Prikaz Londona na mapi	Prikaz Londona na mapi	OK
2	Provjera unosa nepostojeće lokacije (Izazivanje greske)	1. U search tab mape upisati postojecu lokaciju 2. Kliknuti 'search' button	Lokacija: 'InvalidLocation'	Alert poruka 'Location not found'	Alert poruka 'Location not found'	OK

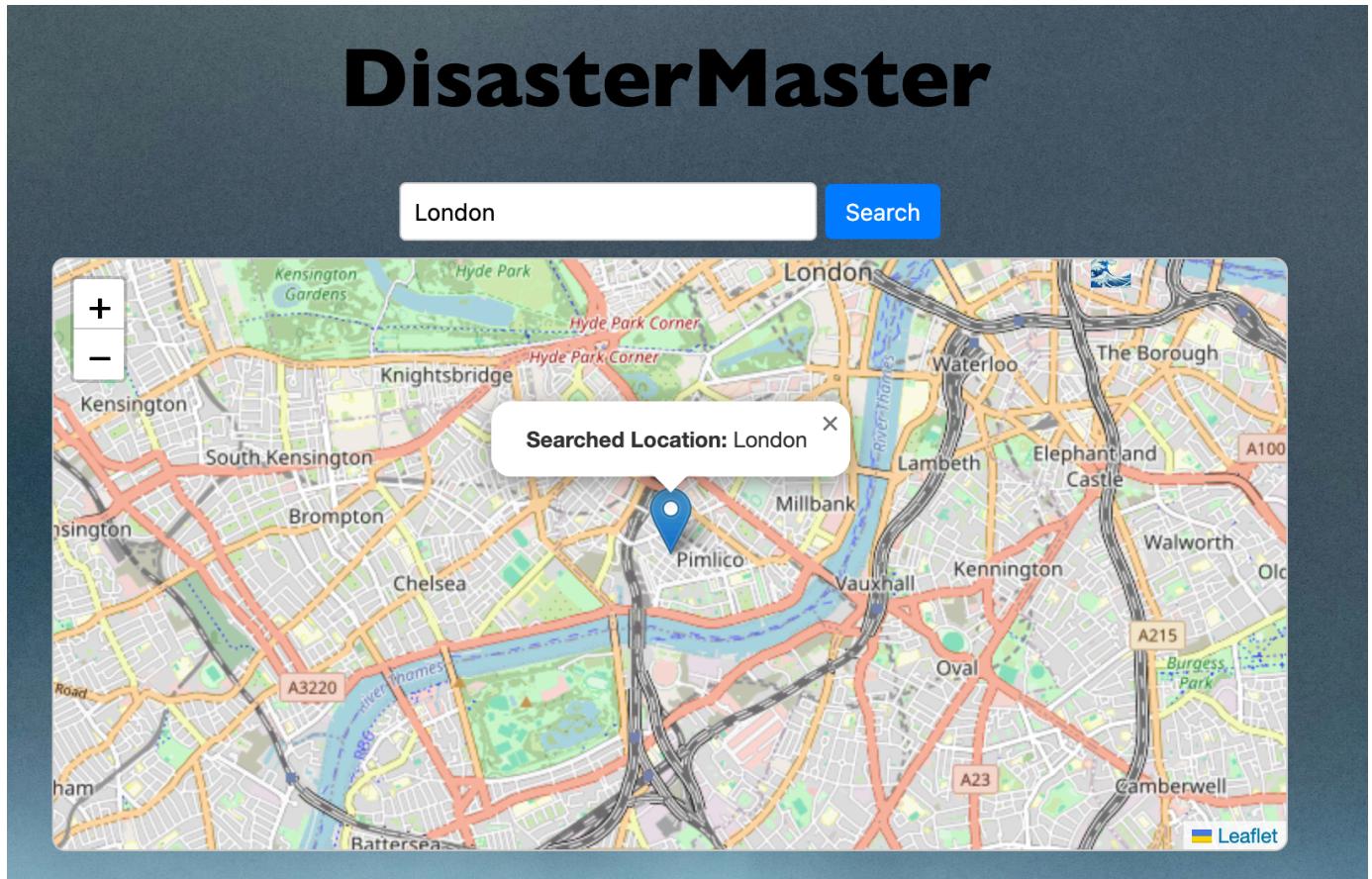
ID	Opis	Koraci	Podatci	Ocekivani izlaz	Stvarni izlaz	Status
3	Provjera praznog unosa (Rubni uvjet)	1. Search tab ostaviti prazan 2. Kliknuti 'search' button	prazan	nema dogadaja	nema dogadaja	OK
4	Provjera dodavanja prijave na mapu (Redovan slučaj)	1. Kliknuti na 'add new weather report' button 2. Odabratи vrstu nepogode 3. Unijeti zeljenu lokaciju 4. Opcionalno dodati opis nepogode i fotografiju 5. Kliknuti na 'submit' button	Vrsta nepogode: 'Fire' Lokacija: 'London'	prikaz novo unesene nepogode na mapi pocetne stranice	prikaz novo unesene nepogode na mapi pocetne stranice	OK
5	Provjera dodavanja prijave na mapu bez unesene lokacije (Rubni uvjet)	1. Kliknuti na 'add new weather report' button 2. Odabratи vrstu nepogode 3. Lokaciju ne ispuniti 4. Kliknuti	Vrsta nepogode: 'Fire'	nema promjene na karti nakon klikanja 'submit' buttona	nema promjene na karti nakon klikanja 'submit' buttona	OK

ID	Opis	Koraci	Podatci	Ocekivani izlaz	Stvarni izlaz	Status
		na 'submit' button				
6	Login putem google accounta (Redovan slučaj)	1. Kliknuti na login gumb 2. Odabratи login with google 3. Nakon odabira korisničkog računa vratit se na početnu stranicu 4. Refreshati stranicu	korisnički račun	prikaz imena na početnoj stranici	prikaz imena na početnoj stranici	OK
7	Pretraga skloništa (Redovan slučaj)	1. Preko izbornika na početnoj stranici otići na 'Nearest shelves' prikaz 2. U search tab upisati željenu lokaciju 3. Kliknuti search button	'London'	prikaz skloništa u Londonu	prikaz skloništa u Londonu	OK

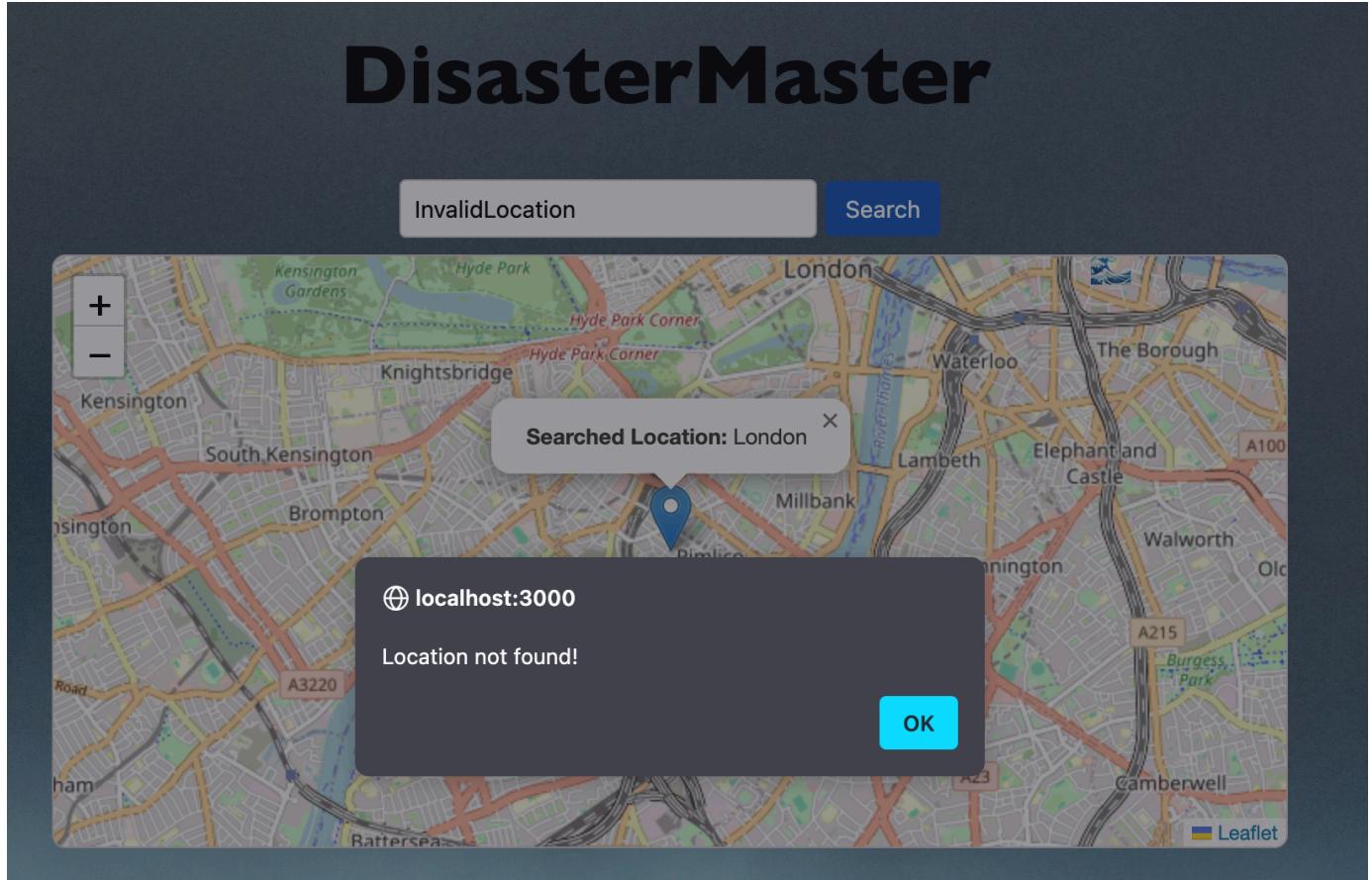
Izlaz skripte za testiranje:

```
Map is still active after empty search.  
enesek@beatsworking test % node Map.test.js  
Browser window maximized to full screen.  
Map is displayed.  
Search button clicked with a valid location.  
Map should now be centered on London.  
Search button clicked with an invalid location.  
Alert for invalid location shown correctly.  
Alert dismissed successfully.  
Search button clicked with an empty search bar.  
Map is still active after empty search.
```

Prikaz unosa lokacije 'London'



Prikaz unosa nepostojeće lokacije



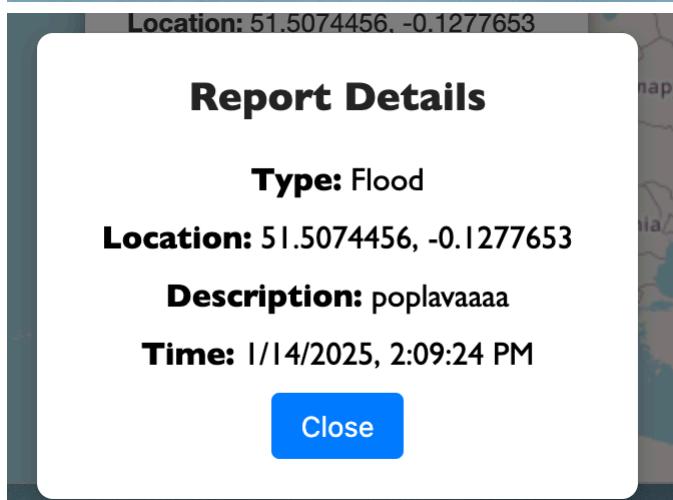
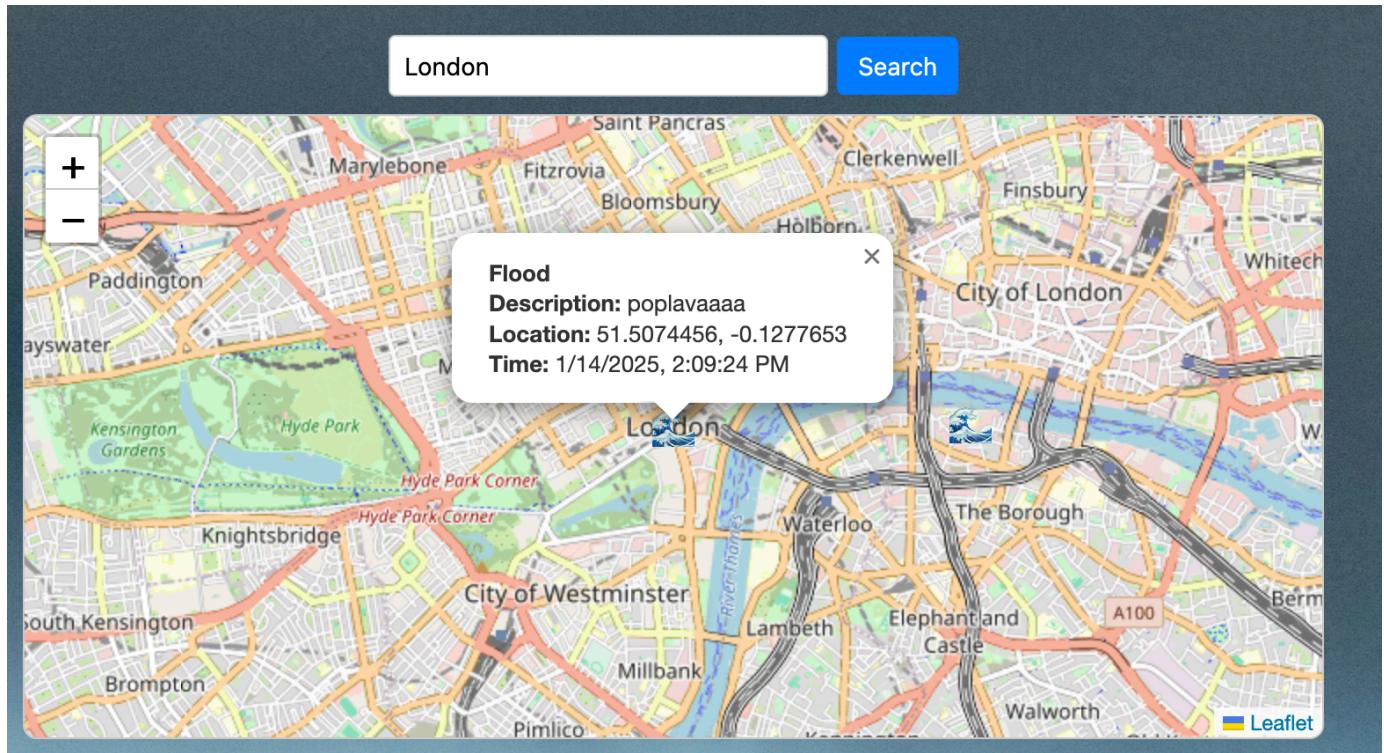
Izlaz skripte za testiranje

```
AddWeatherReports button clicked, list of reports shown for 🌏 Earthquake at Lisbon.  
🌐 Earthquake icon selected.  
Weather report form loaded for 🌏 Earthquake  
Form element located inside form-content.  
Location entered in search bar: Lisbon  
Search button clicked, location set to Lisbon.  
Description added to the form for 🌏 Earthquake in Lisbon.  
Report form submitted for 🌏 Earthquake in Lisbon.  
AddWeatherReports button clicked, list of reports shown for 🌏 Earthquake at Rome.  
🌐 Earthquake icon selected.  
Weather report form loaded for 🌏 Earthquake  
Form element located inside form-content.  
Location entered in search bar: Rome  
Search button clicked, location set to Rome.  
Description added to the form for 🌏 Earthquake in Rome.  
Report form submitted for 🌏 Earthquake in Rome.  
AddWeatherReports button clicked, list of reports shown for 🌏 Earthquake at Athens.  
🌐 Earthquake icon selected.  
Weather report form loaded for 🌏 Earthquake  
Form element located inside form-content.  
Location entered in search bar: Athens  
Search button clicked, location set to Athens.  
Description added to the form for 🌏 Earthquake in Athens.  
Report form submitted for 🌏 Earthquake in Athens.  
AddWeatherReports button clicked, list of reports shown for 🔥 Fire at Barcelona.  
🔥 Fire icon selected.  
Weather report form loaded for 🔥 Fire  
Form element located inside form-content.  
Location entered in search bar: Barcelona  
Search button clicked, location set to Barcelona.  
Description added to the form for 🔥 Fire in Barcelona.  
Report form submitted for 🔥 Fire in Barcelona.  
AddWeatherReports button clicked, list of reports shown for 🔥 Fire at Naples.  
🔥 Fire icon selected.  
Weather report form loaded for 🔥 Fire  
Form element located inside form-content.  
Location entered in search bar: Naples  
Search button clicked, location set to Naples.  
Description added to the form for 🔥 Fire in Naples.  
Report form submitted for 🔥 Fire in Naples.  
AddWeatherReports button clicked, list of reports shown for 🔥 Fire at Paris.  
🔥 Fire icon selected.  
Weather report form loaded for 🔥 Fire  
Form element located inside form-content.  
Location entered in search bar: Paris  
Search button clicked, location set to Paris.  
Description added to the form for 🔥 Fire in Paris.  
Report form submitted for 🔥 Fire in Paris.  
AddWeatherReports button clicked, list of reports shown for 💦 Flood at Amsterdam.  
💦 Flood icon selected.  
Weather report form loaded for 💦 Flood  
Form element located inside form-content.  
Location entered in search bar: Amsterdam  
Search button clicked, location set to Amsterdam.  
Description added to the form for 💦 Flood in Amsterdam.  
Report form submitted for 💦 Flood in Amsterdam.  
AddWeatherReports button clicked, list of reports shown for 💦 Flood at Berlin.  
💦 Flood icon selected.
```

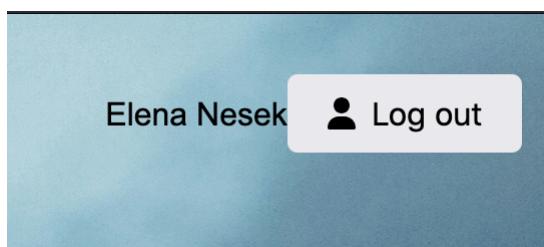
Prikaz novih prijava na mapi glavne stranice



Prikaz detalja prijave sa mape glavne stranice



Izlaz nakon logina google korisničkim računom



Izlaz nakon pretrage skloništa

Nearest Shelters

Enter your location to find nearby shelters:

Park Lane
Coordinates: 51.5110157, -0.1580033

Kennington Park History Hut
Coordinates: 51.4824903, -0.1100478

Search

Leaflet | © OpenStreetMap contributors

+ Add a custom footer

▼ Pages 12

Find a page...

- ▶ [Home](#)
- ▶ [1. Opis projektnog zadatka](#)
- ▶ [2. Analiza zahtjeva](#)
- ▶ [3. Specifikacija zahtjeva sustava](#)
- ▶ [4. Arhitektura i dizajn sustava](#)
- ▶ [5. Arhitektura komponenata i razmještaja](#)
- ▼ [6. Ispitivanje programskog rjesenja](#)
 - Ispitivanje sustava i komponenti
 - Izlaz skripte za testiranje:
 - [Prikaz unosa lokacije 'London'](#)

- Prikaz unosa nepostojeće lokacije
- Izlaz skripte za testiranje
- Prikaz novih prijava na mapi glavne stranice
- Prikaz detalja prijave sa mape glavne stranice
- Izlaz nakon logina google korisničkim računom
- Izlaz nakon pretrage skloništa

▶ [7. Tehnologija za implementaciju aplikacije](#)

▶ [8. Upute za pustanje u pogon](#)

▶ [9. Zaključak i budući rad](#)

▶ [Popis literature](#)

▶ [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>



en765 / DisasterMaster

Code Issues 4 Pull requests Actions Projects Wiki Security

7. Tehnologija za implementaciju aplikacije

[Edit](#)[New page](#)[Jump to bottom](#)

Elena Neseck edited this page last week · [2 revisions](#)

1. Programski jezici

- Projekt je razvijen koristeći JavaScript (ECMAScript 2021) za frontend i Java (verzija 17) za backend. JavaScript je odabran zbog svoje svestranosti i široke podrške za moderne preglednike, dok Java nudi stabilnost i pouzdanost za poslovnu logiku aplikacije.

2. Radni okviri i biblioteke

- React (verzija 18): Za razvoj klijentskog dijela aplikacije korišten je React, popularna JavaScript biblioteka za izgradnju interaktivnih korisničkih sučelja. React omogućuje modularan razvoj putem samostalnih komponenti koje su lako ponovno iskoristive i jednostavne za ažuriranje.
- Node.js (verzija 16): Node.js je korišten za razvoj serverske strane u kombinaciji s paketnim menadžerom npm za upravljanje ovisnostima. Njegova asinkrona priroda omogućava obradu više zahtjeva istovremeno, čime se povećava učinkovitost aplikacije.
- Spring Boot (verzija 3.0): Spring Boot je korišten za backend aplikaciju zbog svoje sposobnosti brzog razvoja složenih aplikacija i integracije s različitim bazama podataka. Njegov sustav konfiguracije pruža fleksibilnost i jednostavno povezivanje s alatima poput OAuth.
- OAuth 2.0: Za autentifikaciju korisnika korišten je OAuth 2.0 sustav, koji omogućuje sigurno prijavljivanje putem Google računa. Ovaj pristup eliminira potrebu za lokalnim pohranama korisničkih lozinki, povećavajući sigurnost aplikacije.
- Leaflet API: Korišten za implementaciju interaktivne mape unutar aplikacije. Leaflet API omogućuje prikaz i manipulaciju geografskih podataka u stvarnom vremenu, čineći ga idealnim za funkcionalnosti poput prikaza lokacija ili rute.

3. Baza podataka

- Projekt koristi PostgreSQL (verzija 13) kao relacijsku bazu podataka. PostgreSQL je odabran zbog svoje robusnosti i mogućnosti rada s velikim količinama podataka, kao i zbog podrške za složene upite i relacijske odnose.

4. Razvojni alati

- Visual Studio Code (VS Code) bio je glavni IDE korišten tijekom razvoja zbog svoje fleksibilnosti i podrške za različite jezike i dodatke.
- Git (verzija 2.34) korišten je za verzioniranje koda, omogućavajući timski rad, povijest promjena i rješavanje sukoba u kodu.

5. Alati za ispitivanje

- Selenium (verzija 4.0) korišten je za testiranje korisničkog sučelja i funkcionalnosti aplikacije. Omogućuje automatizirano ispitivanje preglednika kako bi se osigurala ispravnost i funkcionalnost interaktivnih elemenata aplikacije.
- Jest (verzija 27) korišten za jedinično testiranje komponenti razvijenih u Reactu. Njegova jednostavnost i mogućnost simulacije različitih stanja aplikacije čine ga idealnim alatom za frontend testove.

6. Alati za razmjestaj

- Docker (verzija 20.10) korišten je za razmještaj aplikacije. Docker omogućuje kreiranje izoliranih okruženja u obliku kontejnera, čime se osigurava konzistentnost aplikacije bez obzira na okruženje u kojem se pokreće.

7. Cloud oprema

- Aplikacija je hostana na platformi Render, koja omogućuje jednostavnu implementaciju aplikacije s podrškom za automatsko skaliranje i sigurno upravljanje infrastrukturom. Render nudi integriranu podršku za baze podataka, statičke datoteke i API-je, što pojednostavljuje cijeli proces razmještaja.

8. Alati za modeliranje i dijagrame

- Za izradu dijagrama aplikacije korišten je Astah UML. Alat omogućuje vizualizaciju arhitekture aplikacije, dijagrama klase, sekvenci i tijeka rada, čime se olakšava komunikacija unutar tima i bolje razumijevanje sustava.

+ Add a custom footer

▼ Pages 12

Find a page...

▶ [Home](#)

▶ [1. Opis projektnog zadatka](#)

▶ [2. Analiza zahtjeva](#)

- ▶ [3. Specifikacija zahtjeva sustava](#)
- ▶ [4. Arhitektura i dizajn sustava](#)
- ▶ [5. Arhitektura komponenata i razmještaja](#)
- ▶ [6. Ispitivanje programskog rjesenja](#)
- 7. Tehnologija za implementaciju aplikacije**
- ▶ [8. Upute za pustanje u pogon](#)
- ▶ [9. Zaključak i budući rad](#)
- ▶ [Popis literature](#)
- ▶ [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>



en765 / DisasterMaster

Code Issues 4 Pull requests Actions Projects Wiki Security

8. Upute za pustanje u pogon

[Edit](#)[New page](#)[Jump to bottom](#)

Elena Neseck edited this page 19 hours ago · [9 revisions](#)

1. Instalacija lokalno

- za frontend aplikacije potrebna najnovija verzija Node.js

```
nvm install --lts  
# Potvrди instalaciju  
node -v  
npm -v  
#instalacija ovisnosti  
npm install  
#pokretanje  
npm run dev
```



- za backend potrebna najnovija verzija Dockera

```
npm run build-docker  
npm run run-docker
```



- za preuzimanje izvornog koda se klonira Git repozitorij pomoću poveznice <https://github.com/en765/DisasterMaster>. Git repozitorij najlakse je klonirati korištenjem GitHub Desktopa.

2. Postavke

- ukoliko se želi koristiti vlastiti Google Client Secret za autentifikaciju, u .env datoteci se on može promijeniti:

```
GOOGLE_CLIENT_SECRET="zamijeniti s vlastitim"
```

- isto vrijedi i za bazu, ako se želi koristiti vlastita baza podataka na nekom drugom serveru, trebaju se napraviti promjene u datoteci application.properties u mapi IzvorniKod\dm-

be\src\main\resources :

```
spring.application.name=dm-be
spring.datasource.url=jdbc:postgresql://ep-polished-mud-a2kpdk82.eu-central-1.aws.neon.tech/disaster-master-db
spring.datasource.username=disaster-master-db_owner
spring.datasource.password=hvltsu5cRT6N
```

3. Pokretanje aplikacije

- frontend: postaviti se u direktorij IzvorniKod\dm-fe (pomoću naredbe cd IzvorniKod/dm-fe) u terminalu uređivača, zatim pokrenuti aplikaciju pomoću npm run dev nakon čega se na web pregledniku, na adresi <http://localhost:3000/>, može koristiti aplikacija.
- backend: u mapi IzvorniKod\dm-be\src\main\java\dm_be je potrebno pokrenuti program DmBeApplication.java kojim se pokreće server lokalno kod korisnika.
- za pravilan lokalni rad aplikacije potrebno je obaviti oba postupka

4. Primjer za Render platformu (Cloud Deploy)

Frontend i backend je potrebno deployati na zasebnim servisima.

- Frontend:
 - Prijaviti se na [Render](#)
 - Kreirati novi **Web Service** i povezati ga s GitHub Repozitorijem
<https://github.com/en765/DisasterMaster>
 - Odabratи **Language** Node
 - Konfiguriraju se postavke:
 - **Branch:** main
 - **Root Directory:** ./IzvorniKod/dm-fe/src
 - **Build Command:** npm install
 - **Start Command:** npm run dev
- Backend:
 - Prijaviti se na [Render](#)
 - Kreirati novi **Web Service** i povezati ga s GitHub Repozitorijem
<https://github.com/en765/DisasterMaster>
 - Odabratи **Language** Docker
 - Konfiguriraju se postavke:
 - **Root directory:** ./IzvorniKod/dm-be (s obzirom da se Dockerfile nalazi u tom direktoriju neće biti potrebno ručno unijeti gdje se on nalazi u repozitoriju)
 - Pod **Advanced -> Secret Files -> Add Secret File** je potrebno dodati secret **GOOGLE_CLIENT_SECRET** te vaš google secret kako bi se mogla provoditi autentifikacija
- Pokretanje aplikacije:

- o Render će automatski generirati URL za pristup aplikaciji te će to biti npr.
disastermaster.onrender.com

+ Add a custom footer

▼ Pages 12

Find a page...

► [Home](#)

► [1. Opis projektnog zadatka](#)

► [2. Analiza zahtjeva](#)

► [3. Specifikacija zahtjeva sustava](#)

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

► [5. Arhitektura komponenata i razmještaja](#)

► [6. Ispitivanje programskog rjesenja](#)

► [7. Tehnologija za implementaciju aplikacije](#)

[8. Upute za pustanje u pogon](#)

► [9. Zaključak i budući rad](#)

► [Popis literature](#)

► [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>



[Code](#)[Issues 4](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)

9. Zaključak i budući rad

[Edit](#)[New page](#)[Jump to bottom](#)

Elena Neseck edited this page 19 hours ago · [2 revisions](#)

Naš tim je prvenstveno naučio koliko je teško raditi u grupi. Organizacija i komunikacija su stvorile jedan od najvećih problema pri izradi ovog projekta, naravno uz nedostatak znanja o alatima koji su se koristili. Kada neki zadatak nije išao samostalno, uvijek smo ga pokušali rješiti zajedničkim snagama, bilo to uspješno ili neuspješno. Smatramo da smo projekt mogli odraditi puno uspješnije, ali ne zato jer se nismo dovoljno trudili, nego nam je nedostajalo iskustvo u timskom radu i kvalitetnoj raspodjeli zadataka te pravovremenoj komunikaciji o problemima na koje smo naišli. U funkcionalnosti koje nismo uspjeli implementirati ulaze:

- Prijava nepogode: radi na frontendu, ne radi spremanje u bazu
- Primanje obavijesti putem maila
- Korisničko sučelje za vlasti: uključuje prihvaćanje/odbijanje prijava te pregled svih prijava - napravljen je frontend simbolički prikaz
- Generiranje statističkog izvješća
- Korisničko sučelje za humanitarne organizacije: nema korisničkog sučelja za humanitarne organizacije kako bi mogle pregledavati prikaze, ali je uz pomoć javno dostupnog API-ja napravljena stranica koje prikazuje lokacije skloništa

[+ Add a custom footer](#)[Pages 12](#)[Home](#)[1. Opis projektnog zadatka](#)[2. Analiza zahtjeva](#)

- ▶ [3. Specifikacija zahtjeva sustava](#)
- ▶ [4. Arhitektura i dizajn sustava](#)
- ▶ [5. Arhitektura komponenata i razmještaja](#)
- ▶ [6. Ispitivanje programskog rjesenja](#)
- ▶ [7. Tehnologija za implementaciju aplikacije](#)
- ▶ [8. Upute za pustanje u pogon](#)
- [9. Zaključak i budući rad](#)
- ▶ [Popis literature](#)
- ▶ [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>



[Code](#)[Issues 4](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)

Popis literature

[Edit](#)[New page](#)[Jump to bottom](#)

Elena Neseck edited this page last week · [1 revision](#)

Reference i literatura koja nam je pomogla u ostvarivanju projekta:

1. Programsko inženjerstvo, FER ZEMRIS, <https://www.fer.unizg.hr/predmet/proinz>
2. The Unified Modeling Language, <https://www.uml-diagrams.org/>
3. Astah Community, <http://astah.net/editions/uml-new>
4. Spring Boot, <https://spring.io/projects/spring-boot>
5. React, <https://react.dev/>
6. Leaflet, <https://leafletjs.com/reference.html>
7. Leaflet API reference versions, <https://leafletjs.com/reference-versions.html>
8. PostgreSQL, <https://www.postgresql.org/>
9. Disaster Alert fact sheet, <https://www.pdc.org/wp-content/uploads/2018/11/Disaster-Alert-Fact-Sheet-Web.pdf>
10. Disaster Alert App, <https://disasteralert.pdc.org/disasteralert/>
11. Spring, Web MVC Framework, <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html>
12. OAuth API, <https://oauth.net/2/>
13. AstahUML, <https://astah.net/products/astah-uml/>

[+ Add a custom footer](#)[Pages 12](#)[Home](#)[1. Opis projektnog zadatka](#)[2. Analiza zahtjeva](#)

- ▶ [3. Specifikacija zahtjeva sustava](#)
- ▶ [4. Arhitektura i dizajn sustava](#)
- ▶ [5. Arhitektura komponenata i razmještaja](#)
- ▶ [6. Ispitivanje programskog rjesenja](#)
- ▶ [7. Tehnologija za implementaciju aplikacije](#)
- ▶ [8. Upute za pustanje u pogon](#)
- ▶ [9. Zaključak i budući rad](#)
- ▼ [Popis literature](#)
 - Reference i literatura koja nam je pomogla u ostvarivanju projekta:
 - ▶ [Prikaz aktivnosti grupe](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>



[Code](#)[Issues 4](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)

Prikaz aktivnosti grupe

[Edit](#)[New page](#)[Jump to bottom](#)

Elena Neseck edited this page 19 hours ago · [6 revisions](#)

Dnevnik sastanaka

1. Sastanak

- Datum: 14.10.2024.
- Prisustvovali: Elena Neseck, Leonarda Hunski, Petra Boras, Arjana Ivkovic, Nikola Marinovic, Zeljko Capan, Luka Bobic
- Teme sastanka:
 - medusobno upoznavanje ekipe
 - za nacin komunikacije odabrano je prvenstveno Whatsapp, a u slucaju online sastanka Teams
 - odredeno je koji clanovi ekipe ce raditi na kojim dijelovima projekta (samo okvirno backend, frontend, dizajn i slicno)

2. Sastanak

- Datum: 21.10.2024.
- Prisustvovali: Elena Neseck, Arjana Ivkovic, Nikola Marinovic, Zeljko Capan, Luka Bobic
- Teme sastanka:
 - detaljnije razradeno koji clanovi ce sto odradivati
 - zapoceta izrada Github repozitorija, uredena pocetna strana, dodani README i license fajlovi

3. Sastanak - teams

- Datum: 23.10.2024.
- Prisustvovali: Elena Neseck, Leonarda Hunski, Petra Boras, Arjana Ivkovic, Nikola Marinovic, Zeljko Capan, Luka Bobic
- Teme sastanka:

- razredni UseCase scenariji te zapoceta izrada dijagrama
- dogovorene osnovne implementacije aplikacije
- dogovoren dizajn stranice (izgled pocetne stranice vidljive obicnom korisniku)

4. Sastanak

- Datum: 28.10.2024.
- Prisustvovali: Elena Neseck, Leonarda Hunski, Arjana Ivkovic, Nikola Marinovic, Zeljko Capan, Luka Bobic
- Teme sastanka:
 - zajednicko proucavanje Git-a
 - detaljno razradeni i ispravljeni funckjski zahtjevi te cijela specifikacija zahtjeva

5. Sastanak

- Datum: 4.11.2024.
- Prisustvovali: Leonarda Hunski, Petra Boras, Nikola Marinovic
- Teme satanka:
 - pregled trenutnog napredka u razvoju

6. Sastanak

- Datum: 11.11.2024.
- Prisustvovali: Elena Neseck, Leonarda Hunski, Arjana Ivkovic, Nikola Marinovic, Zeljko Capan, Luka Bobic
- Teme satanka:
 - dogovor oko deploymenta
 - dogovor za implementaciju logina uz autentifikaciju
 - razradivanje arhitekture sustava

7. Sastanak

- Datum: 09.12.2024.
- Prisustvovali: Elena Neseck, Leonarda Hunski, Arjana Ivkovic, Nikola Marinovic, Zeljko Capan, Luka Bobic
- Teme satanka:
 - pregled i analiza prve prijave projektnog zadatka
 - dogovor oko zadataka za finalnu prijavu

8. Sastanak

- Datum: 16.12.2024.

- Prisustvovali: Elena Nesešek, Leonarda Hunski, Arjana Ivković, Nikola Marinović, Zeljko Capan, Luka Bobić
- Teme satanka:
 - razrada o funkcionalnosti logina
 - detaljan dogovor oko izgleda korisnickog sučelja

9. Sastanak

- Datum: 07.01.2025.
- Prisustvovali: Arjana Ivković, Nikola Marinović, Zeljko Capan, Luka Bobić
- Teme satanka:
 - podjela zadataka

10. Sastanak

- Datum: 13.01.2025.
- Prisustvovali: Elena Nesešek, Leonarda Hunski, Arjana Ivković, Nikola Marinović, Zeljko Capan, Luka Bobić
- Teme satanka:
 - pregled alfa verzije aplikacije
 - dogovor oko testiranja sustava

Plan rada

Tablica aktivnosti

Aktivnost	Clanovi - Sati
Upravljanje projektom	Elena Nesešek-2
Opis projektnog zadatka	Elena Nesešek-1
Funkcionalni zahtjevi	Elena Nesešek-1
Opis pojedinih obrazaca	Elena Nesešek-2
Dijagram obrazaca	Elena Nesešek-4
Sekvencijski dijagrami	Elena Nesešek-2
Opis ostalih zahtjeva	Elena Nesešek-1

Aktivnost	Clanovi - Sati
Arhitektura i dizajn sustava	Elena Neseck-2
Baza podataka u dokumentaciji	Elena Neseck-1
Dijagram razreda	Elena Neseck-3
Dijagram stanja	Elena Neseck-3
Dijagram aktivnosti	Elena Neseck-3
Dijagram komponenti	Elena Neseck-4
Korištene tehnologije i alati	Elena Neseck-0.5
Ispitivanje programskog rješenja	Elena Neseck-7
Dijagram razmještaja	Elena Neseck-2
Upute za puštanje u pogon	Željko Capan-0.5
Dnevnik sastajanja	Elena Neseck-1
Zaključak i budući rad	Elena Neseck-0.25
Popis literature	Elena Neseck-0.5
Osnovna struktura frontend projekta	Leonarda Hunski-1, Petra Boras-1
Početni izgled korisničkog sučelja uz Canvu	Leonarda Hunski-2, Petra Boras-2
Raspored elemenata na stranici	Leonarda Hunski-5, Petra Boras-3
Navigacijski izbornik s tri opcije, s funkcionalnošću otvaranja i zatvaranja	Leonarda Hunski-2
Svaka opcija vodi na zasebnu stranicu s korisnim savjetima, obrađene sve opcije iz izbornika	Petra Boras-5
Responzivnost više stranica	Petra Boras-3
Login forma-overlay	Petra Boras-1
Footer s gumbovima i linkovima	Petra Boras-2
Gumbi za pretplatu i hitan poziv	Leonarda Hunski-1
Gumb za dodavanje vremenskih nepogoda, koji prikazom otkriva opcije različitih vrsta nepogoda	Leonarda Hunski-2
Funkcionalnost klika na svaku opciju što otvara form	Leonarda Hunski-1
Dizajn form-a osim interaktivne karte	Leonarda Hunski-3

Aktivnost	Clanovi - Sati
Dizajn dodavanja reporta(kako izgleda na mapi te popis u prikazu)	Leonarda Hunski-3
Responzivnost	Leonarda Hunski-5
Interaktivna karta na početnoj stranici	Petra Boras-3
izrada baze podataka	Nikola Marinovic-2, Luka Bobic-2, Arjana Ivkovic-2, Željko Capan - 2
login putem OAuth 2.0	Nikola Marinovic-12, Luka Bobic-10, Arjana Ivkovic-5
implementacija interaktivne mape	Luka Bobic-7
spajanje s bazom podataka	Željko Capan - 6
rad na stavkama za vladu i hum_org	Arjana Ivkovic-5
rad na funkcionalnosti za citizen	Željko Capan - 10
struktura backenda za projekt	Željko Capan - 2
deployment	Željko Capan - 8
upravljanje projektom na Githubu, kontroliranje branchova i merge branchova	Željko Capan - 4

+ Add a custom footer

▼ Pages 12

Find a page...

- ▶ [Home](#)
- ▶ [1. Opis projektnog zadatka](#)
- ▶ [2. Analiza zahtjeva](#)
- ▶ [3. Specifikacija zahtjeva sustava](#)
- ▶ [4. Arhitektura i dizajn sustava](#)
- ▶ [5. Arhitektura komponenata i razmještaja](#)

- ▶ [6. Ispitivanje programskog rjesenja](#)
- ▶ [7. Tehnologija za implementaciju aplikacije](#)
- ▶ [8. Upute za pustanje u pogon](#)
- ▶ [9. Zaključak i budući rad](#)
- ▶ [Popis literature](#)
- ▼ [Prikaz aktivnosti grupe](#)
 - Dnevnik sastanaka
 - Plan rada
 - Tablica aktivnosti

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/en765/DisasterMaster.wiki.git>

