Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Andraž Hafner

Razvoj spletne in androidne aplikacije za pomoč gasilcem

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Marko Bajec

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu razvijte mobilno aplikacijo ter spletni portal za evidenco stanja hidrantov in vodnih rezervoarjev, ki se uporabljajo v primerih gašenja požarov. Aplikacijo zasnujte v sodelovanju z gasilci, ki poznajo postopke v zvezi z tem in vam lahko povedo, kakšna informacijska podpora bi jim koristila pri njihovem delu.

Zahvaljujem se mentorju prof. dr. Marku Bajcu za mentorstvo. Zahvaljujem se tudi dekletu in družini, ki so me med pisanjem diplomskega dela podpirali in mi stali ob strani.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
GPS	Global Positioning System	Globalni sistem pozicioniranja
AVD	Android Virtual Machine	Virtualna naprava Android
API	Application Programming Interface	Aplikacijski programski vmesnik
SDK	Software Development Kit	Orodje za razvoj programske opreme
NFC	Near Field Communication	Komunikacija kratkega dosega
SQL	Structured Query Language	Strukturirani povpraševalni jezik za delo s podatkovnimi bazami
REST	Representational State Transfer	Predstavitveni prenos stanja
JSON	JavaScript Object Notation	Objekt JavaScript za izmenjavo podatkov

Povzetek

Naslov: Razvoj spletne in androidne aplikacije za pomoč gasilcem

Avtor: Andraž Hafner

Za izdelavo diplomskega dela sem se odločil, ker sem prostovoljni gasilec v domačem društvu. Trenutno stanje na tem področju je slabo. Vsako leto opravljamo preventivni pregled vseh hidrantov in vodnih rezervoarjev na našem področju in ugotovitve zapisujemo na papir. Ker so ti papirji pospravljeni v arhivu, nam v primeru intervencije ne pomagajo dosti. Po drugi strani pa ima vsak gasilec takrat pri sebi telefon, na katerem bi lahko za katero koli lokacijo v občini pridobil najbližje hidrante in stanje teh. Pri uporabi telefona skupaj z oznako NFC, pa sem se odloči v sklopu aplikacije idelati časovnik, ki ga nastavimo, ko grejo gasilci v notranji napad in nas opozori po določenem času.

Ključne besede: android, GPS, gasilci.

Abstract

Title: Android Application for Intervention Services

I decided to write this diploma thesis because I am a volunteer firefighter in my local fire department. Each year we conduct preventive check-ups of hydrants and water tanks in our region. We write down our observations on a piece of paper. Since these documents are later stored in an archive, they are not very helpful during an intervention. However each firefighter owns a mobile phone that could potentially be used to access the information on the nearest hydrants and their condition, for any location within the municipality. For phones with NFC, I have decided to make a timer as part of the applicationl. The timer can be set when firefighters enter a burning building, and it will give off a warning after a certain period of time has passed.

Keywords: Android, GPS, firefighters.

Poglavje 1

Uvod

V zadnjem času Slovenijo večkrat letno prizadenejo naravne nesreče, požari se dogajajo vsakodnevno, brez informacij o novi prometni nesreči pa ne minejo niti ene dnevne novice. Za pomoč v nesrečah se največkrat najprej obrnemo na gasilce. Pri vseh teh dogodkih ima pomembno vlogo hitrost, saj v prvi minuti požar pogasi že žlička vode, po desetih minutah pa tudi jezero ne bo več dovolj veliko, če nekoliko karikiram. V teh primerih je izjemno pomembno, da imamo gasilci v najkrajšem času omogočen dostop do zadostne količine vode. V zadnjem času se je gasilstvo, dejavnost z več kot 140 let dolgo zgodovino, zelo posodobilo. Poznamo več različnih načinov za obveščanje o intervencijah. Skoraj vsa gasilska vozila imajo vgrajene navigacijske naprave. Še vedno pa veliko stvari zapisujemo na papir. Zapiski so nato skrbno shranjeni v arhivu, saj nam lahko mogoče kdaj pridejo prav.

1.1 Opis problema

Ena takih dejavnosti je pregled hidrantov, ki ga opravimo vsako leto. Pri tem se odpravimo do vsakega hidranta, ga preizkusimo in svoje ugotovitve zapišemo na poseben obrazec, ki ga pospravimo v arhiv. Ko se zgodi požar, nam ti podatki niso na voljo, da bi lahko pogledali, kateri hidrant je najbližji in ali sploh deluje. To je privedlo do ideje za izdelavo aplikacije, v katero

bi zapisovali podatke o hidrantih in bi jih imeli tako vedno pri roki. Druga gasilska dejavnost, ki bi jo lahko optimizirali in posodobili s pomočjo sodobne tehnologije, pa je zapis vstopov notranjih napadalcev v stavbo. Pri vsakem požaru se skupina gasilcev opremi z izolirnimi dihalnimi aparati in vstopi v goreč prostor. Pri tem delu so zelo ogroženi, s seboj pa imajo omejeno količino zraka, ki ponavadi zadošča za 20 do 30 minut dela, odvisno od intenzivnosti. Zato je za vodjo intervencije zelo pomemben podatek, kdo in koliko časa je že v goreči stavbi. Trenutno v ta namen uporabljamo posebno tablo s tabelo, v katero vpišemo podatke o notranjih napadalcih, količini zraka in času vstopa, hkrati pa nastavimo uro, ki nas ob nastavljenem času z zvokom opozori, da preverimo stanje notranjih napadalcev. V ta namen bi rad razvil aplikacijo, ki bi prebrala oznako NFC z zapisom imena in priimka, ki bi jo imel gasilec pritrjeno na obleko. V aplikacijo bi vnesli čas, aplikacija pa bi nas po preteku časa opozorila z zvočnim alarmom.

1.2 Cilji

Zaradi nepraktičnosti in okornosti trenutnega sistema sem se odločil izdelati svojo aplikacijo. Glani cilji aplikacije so:

- iskanje najbližjega hidranta v primeru intervencije,
- vodenje evidence o pregledih hidrantov,
- pregled hidrantov in izvoz seznama v nam berljivi obliki, tako da ga lahko posredujemo naprej, recimo komunalni službi, da odpravi pomanjkljivosti,
- izdelava MySQL podatkovne baze hidrantov ter njihovih pregledov,
- pridobivanje podatkov iz baze s pomočjo storitve REST,
- izdelava spletne aplikacije za pregled podatkov o hidrantih,
- branje oznak NFC,

- prikaz obvestila in sprožitev zvočnega alarma.

Poglavje 2

Uporabljene tehnologije in orodja

V tem poglavju bom opisal programe in tehnologije, ki sem jih uporabil pri izdelavi aplikacije. Razdelim jih lahko na dva dela, saj je aplikacija izdelana v programu Android Studio, v programskem jeziku Java, uporabljena pa sta še GPS in branje oznak NFC. Spletno aplikacijo in storitev, ki vrača podatke o hidrantih, sem razvil v urejevalniku besedila Sublime, v programskem jeziku PHP, s pomočjo poizvedb SQL v podatkovni bazi.

2.1 Android

Mobilni operacijski sistem Android [1] je razvil Google. Temelji na jedru operacijskega sistema Linux in je v prvi vrsti razvit za naprave z zaslonom na dotik, kot so mobilni telefoni in tablice. Vedno pogosteje se pojavlja tudi na pametnih urah, v avtomobilih in televizijah.

Prvič je bil javno predstavljen leta 2007, prva dostopna naprava s sistemom Android 1.0 pa je bila na voljo septembra 2008. Do danes je bil Android deležen številnih večjih ali majših posodobitev, trenutno zadnja različica pa je 9 z imenom Pie, predstavljena avgusta 2018. Ker Android temelji na odprtokodnem jedru, je razvoj aplikacij preprostejši in posledično cenejši, zato

je veliko aplikacij na voljo brezplačno.

Kot zanimivost, Google vse večje izdaje poimenuje z imenom slaščice po abecednem redu. Tako je bila različica 2.3 Gingerbread, 4.0 Ice Cream Sandwich in tako naprej do 7.0 Nougat ter 8.0 Oreo, kar je bila tudi promocijska akcija za enako imenovane piškote.

2.2 Android Studio

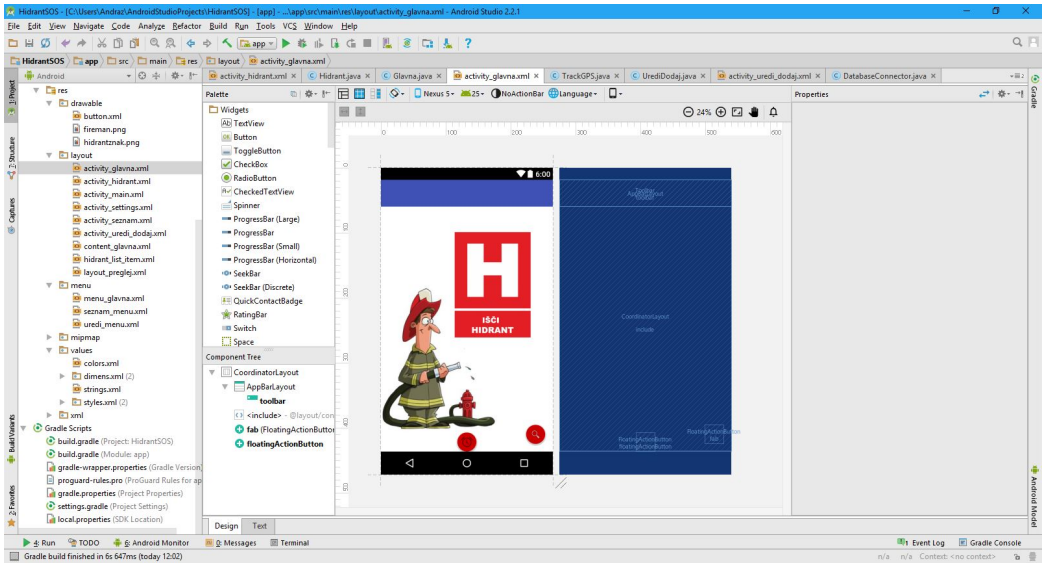
Android Studio [2] je uradno orodje za razvoj aplikacij Android. Predstavljen je bil 16. maja 2013 na Googlovi I/O konferenci. Temelji na programski opremi JetBrains' IntelliJ IDEA in je razvit posebej za razvoj aplikacij Android. Na voljo je za operacijske sisteme Windows, MAc OS X in Linux.

Projekt Android je sestavljen iz več modulov. V AndroidManifest.xml so navedeni podatki o minimalnem SDK, ki ga aplikacija zahteva, o dovoljenjih, ki jih potrebuje za svoje delovanje, in njenih aktivnostih. V modulu Java se nahajajo razredi, znotraj katerih implementiramo delovanje aplikacije. V mapi res pa se nahajajo slike in datoteke XML za oblikovanje, seznam stringov in podobno.

Vgrajen je tudi emulator naprave Android, za katerega lahko določimo, na kateri napravi in kateri različici Androida bomo preizkusili aplikacijo. Seveda pa lahko povežemo tudi svoj telefon in na njem preizkusimo, ali vse deluje pravilno. Slika 2.1 prikazuje vmesnik programa Android Studio.

2.3 Sublime

Sublime [10] je napredni urejevalnik besedila. V njem lahko pišemo v večini programskih jezikov, ki jih tudi primerno obarva. V primeru, da katerega jezika ne pozna, skoraj zagotovo obstaja dodatek zanj, ki ga lahko namestimo znotraj urejevalnika. Pozna zelo veliko bližnjic, ki jih lahko tudi uredimo po svoje, predlaga pa tudi funkcije ki jih lahko uporabimo. Program je plačljiv. Licenca se nanaša na uporabnika, zato lahko en uporabnik z eno licenco



Slika 2.1: Grafični vmesnik programa Android Studio.

program namesti na več računalnikov, obstaja pa tudi možnost brezplačnega preizkusa.

2.4 Java

Java [4] je objektno usmerjen, prenosljiv programski jezik, ki ga je razvil James Gosling s sodelavci v podjetju Sun Microsystems. Projekt, ki se je na začetku (leta 1991) imenoval Oak (hrast), je bil razvit kot zamenjava za C++. Jave ne smemo zamenjevati z jezikom JavaScript, ki ima podobno ime ter podobno, C-jevsko, skladnjo. Različica Java 1.0 je bila objavljena leta 1996, zadnja različica pa je 8.0 (marec 2014). Javo vzdržuje in posodablja Oracle - Sun Microsystems.

2.5 PHP

PHP [6] je razširjen odprtokodni programski jezik, ki se izvaja na strežniku in omogoča razvoj dinamičnih spletnih vsebin. PHP deluje tako, da primerno

teče na strežniku, kjer iz izvorne kode PHP generira spletno stran HTML. Prva različica PHP, kot ga poznamo danes, je bila PHP 3, javnosti dostopna leta 1998. Trenutno na največ strežnikih tečeta različici 5 ali 7, najnovejša različica, izdana decembra 2018, pa je PHP 7.3.

2.6 GPS

Globalni sistem pozicioniranja [3] je satelitski navigacijski sistem, ki se uporablja za določanje točne lege in časa kjer koli na Zemlji ali v zemeljski tirnici. Njegovi sateliti na potovanju okrog Zemlje uporabljajo srednjo krožno tirnico.

Sistem GPS je zasnovalo obrambno ministrstvo ZDA, ki ga tudi upravlja. Prosto ga lahko uporablja vsakdo, ki ima ustrezen sprejemnik. Razdeljen je na tri odseke: vesoljskega, nadzornega in uporabniškega. Vesoljski odsek vključuje satelite GPS, nadzorni zemeljske postaje, ki skrbijo za nadzorovanje poti satelitov, usklajevanje njihovih atomskih ur in nalaganje podatkov, ki jih oddajajo sateliti. Uporabniški odsek sestavljajo civilni in vojaški sprejemniki GPS, ki razberejo časovne podatke iz večjega števila satelitov in nato izračunajo lego sprejemnikov s postopkom triangulacije.

2.7 NFC

Near Field Communication [5] je visokofrekvenčna komunikacijska tehnologija kratkega dosega, ki deluje na razdalji do 10 cm. Trenutno se je tehnologija NFC najbolj prijela ravno pri mobilnih telefonih, ker telefon lahko nastopa v treh različnih vlogah. Prva vloga je kot kartica, kjer telefon nadomesti obstoječe kartice NFC (npr.: kontrola pristopa), druga vloga je čitalec, kjer s telefonom preberemo obstoječo kartico, tretja pa način P2P, kjer si dve enakovredni napravi izmenjujeta podatke.

2.8 REST

REST [7] predstavlja arhitekturo, kjer je vsak vir predstavljen kot spletna storitev z enoznačnim naslovom URL. REST sam po sebi ni standardiziran protokol, opredeljuje pa, kako uporabljati obstoječe standarde. Načelo REST je uporaba protokola HTTP, tako kot je modeliran. Za dostopanje in spreminjanje virov se tako uporabljajo standardne metode HTTP: GET, PUT, POST in DELETE.

2.9 SQL

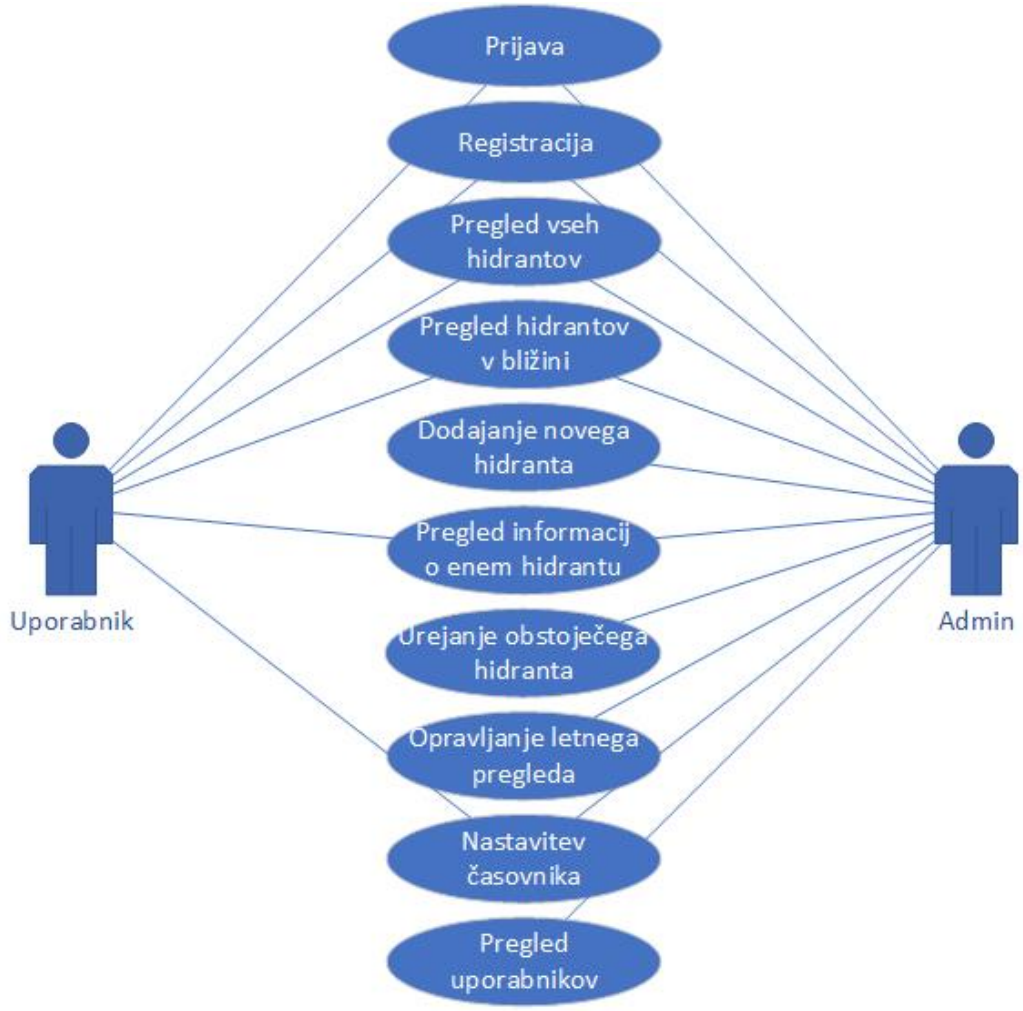
SQL [9] ali strukturirani povpraševalni jezik za delo s podatkovnimi bazami (angl. Structured Query Language) je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkovnimi zbirkami, s programskimi stavki, ki posnemajo ukaze v naravnem jeziku.

Poglavje 3

Načrt aplikacije

Pri načrtovanju aplikacije sem se najprej posvetil aktivnostim za katere želim, da jih aplikacija omogoča. Za pomoč pri tem je bil izdelan model primerov uporabe, ki ga prikazuje slika 3.1. Aplikacija mora omogočati pregled seznama vseh hidrantov ter samo hidrantov, ki so v bližini. Za nastavljanje oddaljenosti potrebujemo polje, kamor to vpišemo, ter možnost pridobitve trenutne lokacije s pomočjo senzorja GPS. Ko uporabnik vpiše razdaljo in začne pregledovati hidrante, mora aplikacija omogočati pregled podrobnosti o posameznem hidrantu. Da lahko pregleduje hidrante, jih mora imeti možnost dodati in jih kasneje izbrisati ali urediti. Ker želimo z aplikacijo poenostaviti tudi letni pregled hidrantov, mora aplikacija smiselno ponuditi možnost vnosa pregleda. Ker ne želimo, da imajo vsi dostop do teh podatkov in možnosti urejanja, mora alikacija vsebovati tudi možnost prijave in registracije. Vse te podatke je treba tudi shraniti, za kar potrebujemo podatkovno bazo, v katero se bodo podatki zapisovali, ter storitev s pomočjo katere bo potekala komunikacija med aplikacijo in podatkovno bazo. S tem sem nekako zaključil celoto v povezavi s hidranti in aplikacijo. Ker sem želel, da aplikacija nadomesti tudi časovnik notranjih napadalcev, sem moral omogočiti branje z oznake NFC in nastavljanje odštevalnika časa. Želeli smo imeti tudi pregled nad preventvnimi pregledi hidrantov na spletni strani, za kar sem moral razviti tudi spletno stran s podstranjo za pregled posameznega hidranta ter

povezavo s podatkovno bazo za pridobivanje podatkov. Na spletni strani smo potrebovali tudi nadzor nad uporabniki, kamor se lahko prijavijo samo administratorji ter administratorske pravice dodajajo tudi drugim uporabnikom ali jih izbrišejo.

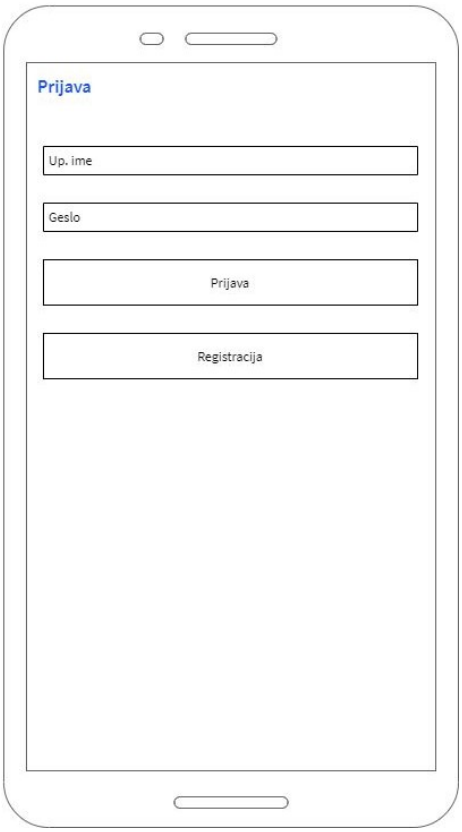


Slika 3.1: Model primerov uporabe.

Po tem je sledilo sestavljanje zaslonov in zaslonskih slik, da bi aktivnosti čim bolje razvrstili in omogočili čim boljšo in preprosto uporabo. Pri tem smo morali razmisliti, da aktivnosti, ki jih potrebujemo v času intervencije, bolj izpostavimo, druge aktivnosti pa razvrstimo tako, da nas takrat ne ovirajo.

V ta namen sem začel z razvojem grafičnega vmesnika na papir, kasneje pa sem za to uporabil programsko opremo. Pri tem sem izdelal žični diagram vseh aktivnosti.

Ob zagonu aplikacije se prikaže aktivnost za prijavo v aplikacijo (slika 3.2), na kateri se s prijavo premaknemo na glavno aktivnost. V primeru, da uporabniškega imena še nimamo, lahko z gumbom "Registracija"odpremo aktivnost za registracijo (slika 3.3). Po uspešni registraciji nas aktivnost usmeri nazaj na prijavo.

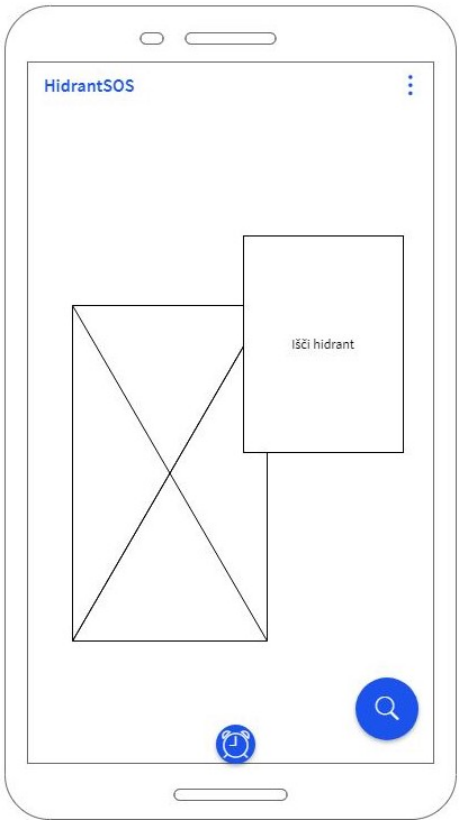


Slika 3.2: Žični diagram - Prijava

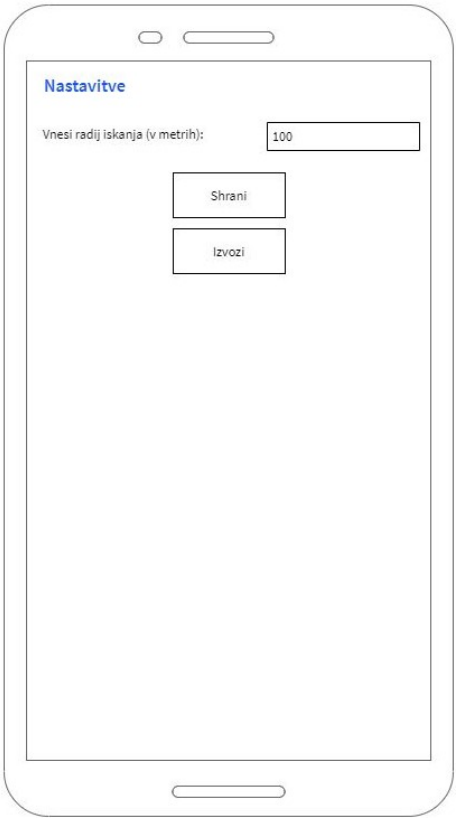
Slika 3.3: Žični diagram - Registracija

Na glavni aktivnosti (slika 3.4) so postavljeni trije gumbi: glavni velik gumb za prikaz bližnjih hidrantov, gumb v spodnjem desnem kotu za prikaz celotnega seznama ter gumb spodaj na sredini za prikaz časovnika notranjih napadalcev. V zgornjem desnem kotu je meni, od koder lahko dostopamo do

nastavitev ali se odjavimo iz aplikacije. V nastavitvah (slika 3.5) so polja za vnos razdalje, gumb shrani in gumb za izvoz podatkov.



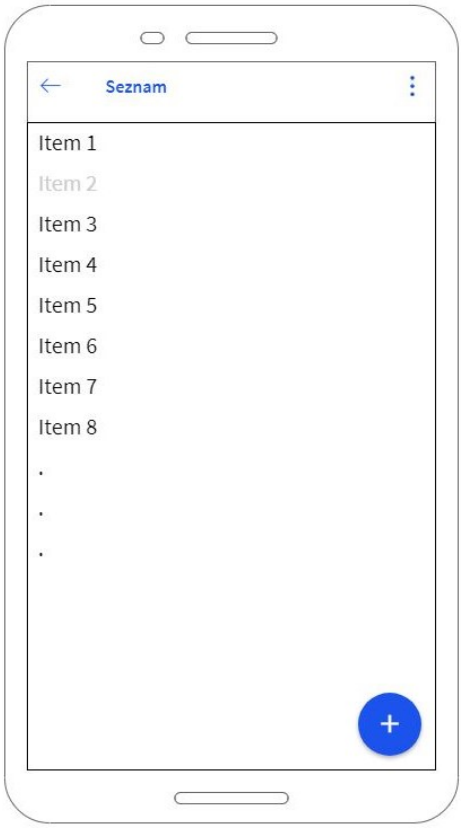
Slika 3.4: Žični diagram - Glavna aktivnost



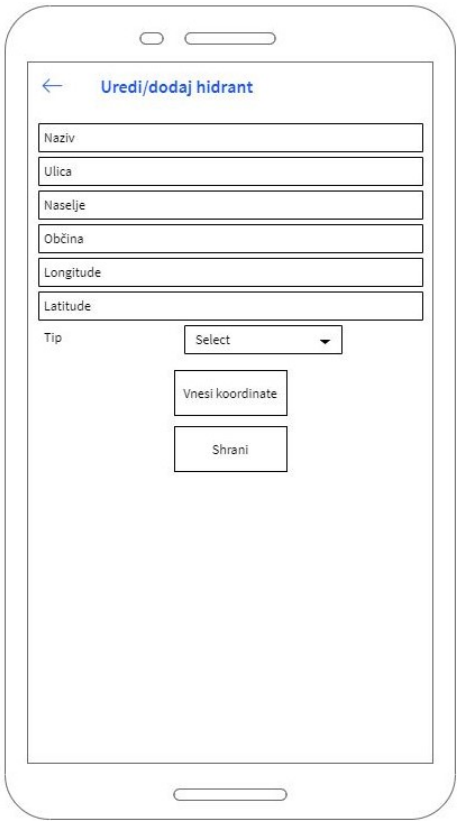
Slika 3.5: Žični diagram - Nastavitve

Pri prikazu seznama (slika 3.6) sem se odločil samo za eno aktivnost, podatki pa se prikažejo različno, odvisno od tega, na kateri gumb na prvi strani pritisnemo. Aktivnost je sestavljena iz seznama hidrantov in gumba za dodajanje novega hidranta, enako dejanje pa lahko izvedemo tudi iz zgornjega menija. V primeru, da uporabnik ni administrator, se zgornji meni in gumb za dodajanje hidranta ne prikažeta. Dve aktivnosti sem združil tudi v primeru dodajanja in urejanja hidrantai (slika 3.7). V primeru dodajanje hidranta do aktivnosti pridemo, če pri seznamu kliknemo na gumb dodaj hidrant. Če pa ga želimo urediti, moramo najprej priti na aktivnost

s podrobnostmi o hidrantu, kjer lahko v zgornjem meniju izberemo urejanje hidranta. V tem premeru se tudi polja že izpolnejo z znanimi podatki, ki jih lahko uredimo. Te aktivnosti navadni uporabnik ne vidi.



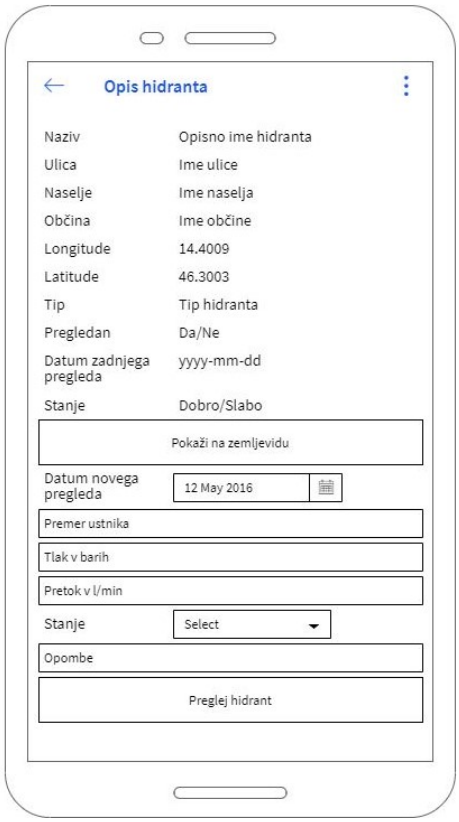
Slika 3.6: Žični diagram - Seznam



Slika 3.7: Žični diagram - Uredi/dodaj

S pritiskom na posamezni hidrant iz seznama se odpre naslednja aktivnost, na kateri so prikazani podrobni podatki (slika 3.8). Ta aktivnost je sestavljena iz seznama lastnosti hidranta, gumba, ki nas preusmeri v aplikacijo z zemljevidi in tam prikaže točko, kjer je hidrant, ter vnosnih polj za vnos vrednosti pregleda in gumba za potrditev. V zgornjem desnem kotu sta v meniju možnosti urejanja in izbrisa hidranta. V primeru, da uporabnik ni administrator, menija in polj za vnos pregleda ne vidi (slika 3.9).

Ker sem želel, da je dostop do časovnika avtomatski od dotiku oznake



Slika 3.8: Žični diagram - Opis (administratorski pogled)

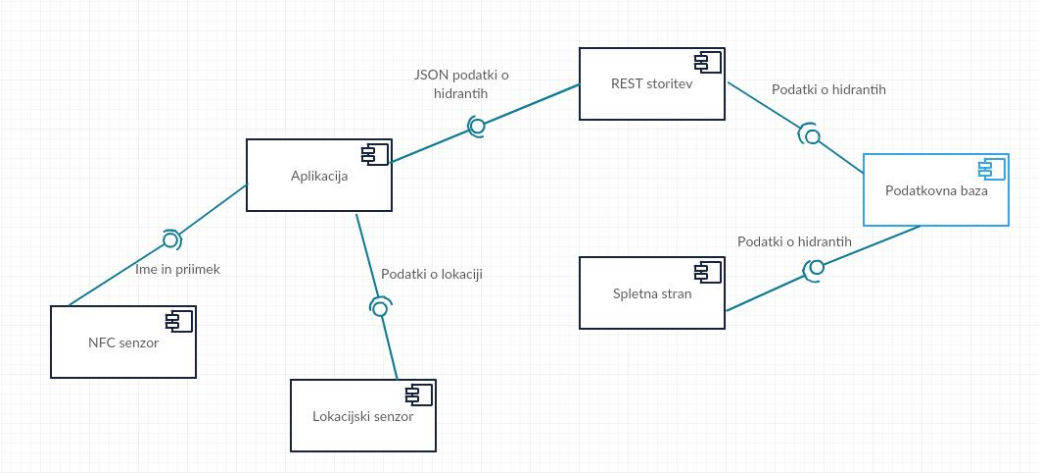
Slika 3.9: Žični diagram - Opis

NFC, skoraj ni bilo potrebe po gumbu za dostop do te aktivnosti. Ker včasih vseeno želimo preveriti preostali čas, je bilo smiselno dodati gumb na prvo aktivnost za dostop do časovnika. Aktivnost s časovnikom vsebuje polje s prebranim imenom in priimkom, možnost vnosa časa ter gumbe za kontrolo časovnika (slika 3.10).

Izdelan je bil tudi komponentni diagram, ki prikazuje posamezne komponente aplikacije in povezavo med njimi (slika 3.11). Pri tem smo se osredotočili na to, katere aktivnosti oziroma komponente aplikacije za svoje delovanje potrebujejo katero od drugih komponent. V tem primeru aplikacija za svoje delovanje potrebuje trenutno lokacijo, ki jo pridobimo s pomočjo

Slika 3.10: Žični diagram - Časovnik

senzorja GPS, za delovanje časovnika pa podatke, ki jih z oznake NFC preberemo z zato namenjenim senzorjem. Vendar nam lokacija ne koristi veliko, če nimamo podatkov o hidrantih. Ti so shranjeni v podatkovni bazi, do katere ne dostopamo neposredno. Zato z aplikacijo dostopamo do storitve, ki je na strežniku. Ta nato naredi povpraševanje v podatkovni bazi in nam vrne ustrezne podatke v aplikacijo. Ločeno od androidne aplikacije se na strežniku nahaja spletna stran, ki neposredno dostopa do podatkovne baze in prikazuje podatke.



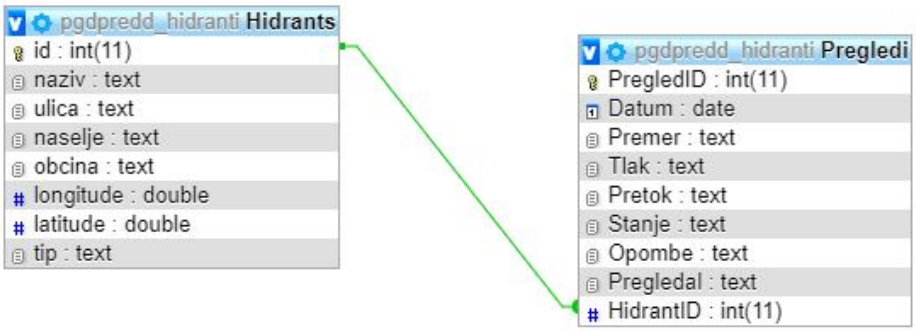
Slika 3.11: Komponentni diagram.

Po pregledu, katere podatke o hidrantu želimo hraniti, podatkovna baza vsebuje dve tabeli, ki sta med seboj povezani z id-jem hidranta, kakor prikazuje slika 3.12. V podatkovni bazi je še tabela z uporabniki, za potrebe registracije in prijave v aplikacijo.
Prva tabela se imenuje Hidrants in vsebuje podatke o posameznem hidrantu.

- id: zaporedna številka hidranta, doda se avtomatsko,
- naziv: ime hidranta, tako da ga čim lažje najdemo (hidrant pri trgovini ...),
- ulica: na kateri ulici stoji hidrant,
- naselje: v katerem naselju se nahaja hidrant,
- obcina: v kateri občini se nahaja hidrant,
- longitude: zemljepisna dolžina,
- latitude: zemljepisna širina,
- tip: nadtalni, podtalni ...

Druga tabela z imenom Pregledi pa vsebuje podatke o posameznem pregledu:

- PregledID: zaporedna številka pregleda, doda se avtomatsko,
- Datum: datum pregleda,
- Premer: premer ročnika, s katerim se je opravil pregled,
- Tlak: tlak, ki ga je dosegel hidrant pri pregledu,
- Pretok: izmerjen pretok vode,
- Stanje: podatek o delovanju hidranta,
- Opombe: opažanja,
- HidrantID: tuji ključ id iz tabele Hidrants.



Slika 3.12: Model podatkovne baze

Tabela z uporabniškimi podatki se imenuje Gasilci:

- ID: zaporedna številka uporabnika, doda se avtomatsko,
- UpIme: uporabniško ime uporabnika,
- Ime: ime uporabnika,
- Priimek: priimek uporabnika,

- Geslo: geslo, zaščiteno z MD5,
- Admin: podatek o tem, ali je uporabnik tudi administrator.

Poglavje 4

Pregled in analiza sorodnih aplikacij

Našel sem malo podobnih aplikacij. Najbolj podobni sta dve mobilni aplikaciji, eno od njiju sem tudi preizkusil, druga pa je namenjena zgolj uporabi znotraj podjetja. Vsi podatki in aplikacije na to temo so prvenstveno namenjeni komunalnim podjetjem in v tem pogledu ne implementirajo pomembnih funkcionalnosti za gasilce, kot je na primer iskanje najbližjega hidranta. Našel sem tudi spletno stran, ki prikazuje javno dostopne podatke občin, med katerimi so tudi lokacije hidrantov.

4.1 Mariborski vodovod d. d.

Apliakcija je prosto dostopna v trgovini Google Play. Za uporabo se ni treba registrirati. Znotraj aplikacije so kontakt podjetja Mariborski vodovod, ter povezavi do Facebooka in spletne strani. Ena izmed funkcionalnosti je tudi Karta hidrantov s klikom nanjo nas aplikacija preusmeri na spletno stran `https://fusiontables.google.com/data?docid=1ye1M5cgWBBfd5o93JhASnaDTOjFlWWhEu` . Tam so trije zavihki in vsak prikazuje svojo obliko prikaza. V prvem je seznam vseh hidrantov, v drugem zavihku pa so kartice hidrantov. Še najbolj zanimiv je tretji zavihek, kjer so na zemljevidu prikazani vsi hidranti, s

klikom nanje pa se odprejo podrobni podatki. Skozi aplikacijo ne moremo spreminjati nobenih podatkov o hidrantu, prav tako ne moremo vnesti podatkov o pregledu. Tudi možnosti prikaza glede na oddaljenost ne omogoča.

4.2 JKP Žalec - hidranti

Aplikacija je le ena izmed številnih podobnih aplikacij istega razvijalca. Aplikacije so ločene glede na občino in tip komunalnih storitev ki jih ponujajo. Aplikacije nisem mogel preizkusiti, saj je za uporabo od administratorja treba pridobiti geslo, sem pa nekaj podatkov razbral iz zaslonskih slik. Aplikacija omogoče iskanje objekta, dodajanje dogodka (verjetno okvara, pregled ...) ter pregledovanje seznamov. Možna sta tudi pregled posameznega hidranta in urejanje nekaterih podatkov o njem.

4.3 iObčina

IObčina je spletni informacijski sistem istega podjetja, kot je razvilo aplikacijo JKP Žalec - hidranti. Znotraj informacijskega sistema imamo možnost iskati, pregledovati in analizirati elemente na zemljevidu. Imamo možnost več različnih pogledov, z različno količino podatkov. Eden izmed nivojev podatkov je tudi nivo s prikazom hidrantov. Ta infomacijski sistem ima zelo veliko pokritost in lahko vidimo lokacije hidrantov po skoraj celotni Sloveniji, vendar sem po pregledu lokacij v naši občini odkril, da je seznam zelo pomanjkljiv, saj manjka več kot pol hidrantov. Tudi kakšnih dodatnih podatkov o hidrantu ni na voljo.

Poglavje 5

Opis rešitve

5.1 Pridobivanje podatkov iz podatkovne baze

Pridobivanje podatkov lahko razdelimo na dva dela. Prva je storitev REST, ki teče na stražniku in glede na zahtevo vrne zbirko podatkov v formatu JSON. Drugi pa je razred DatabaseConnector.java, ki znotraj aplikacije najprej skrbi za pravilni zahtevek na strežnik, potem pa odgovor tudi pravilno oblikuje za nadaljnjo uporabo.

5.1.1 Storitev REST

Na strežniku sem ustvaril storitev REST [8], ki glede na metodo in podane parametre vrača pravi odgovor JSON. V storitvi najprej preberem vse parametre in jih razčlenim ter podam dostopne podatke do podatkovne baze. Potem glede na metodo sestavim stavek SQL, ki ga kasneje izvedem. V primeru metode GET, izvedem stavek SELECT, ter pridobim podatke o hidrantih ali pregledih. Pri metodi PUT posodobim zapis v podatkovni bazi, pri metodi POST pa dodam nov zapis. Dodana je tudi metoda DELETE ki nam izbriše določen hidrant ali pregled. Podatek, nad katero tabelo izvajamo poizvedbo, prejmem prek parametra.

```
// create SQL based on HTTP method
```

```
switch ($method) {
  case 'GET':
    if ($keyPregled) {
      $sql = "select * from `$table` WHERE
         $keyPregled=$valPregled ORDER BY Datum DESC
         ".($key?" LIMIT $key":''); break;
    }
    else {
      $sql = "select * from `$table`".($key?" WHERE
         id=$key":''); break;
    }
  case 'PUT':
    $sql = "update `$table` set $set where id=$key";
        break;
  case 'POST':
    $sql = "insert into `$table` set $set"; break;
  case 'DELETE':
    $sql = "delete from `$table` where id=$key";
       break;
}
```

Tudi odgovor strežnika je odvisen od metode in pridobljenih podatkov. V primeru, da iz podatkovne baze ne prejmemo nobenih podatkov, strežnik vrne kodo odgovora 404, kar pomeni, da podatki niso bili najdeni. Ko je poizvedba uspešna, strežnik vrne kodo 200, kar pomeni, da je zahtevano uspešno izvedel. Če pošljemo zahtevek z metodo GET, vrne seznam JSON, v ostalih primerih pa vrstico v tabeli, ki je bila na novo dodana, spremenjena ali izbrisana.

Zaradi dodatnih zahtev pri preverjanju se v primeru, da poizvedbo izvajamo nad tabelo"Gasilci", izvedejo drugačni stavki SQL. Uporabljata se samo metodi GET in POST. Pri metodi GET preverimo ali v tabeli obstaja

par uporabniškega imena in gesla. V primeru ujemanja vrnemo vse podatke o uporabniku. Pri metodi POST prav tako najprej preverimo, ali uporabnik obstaja. V primeru, da ne obstaja, ga dodamo v tabelo in vrnemo številko vrstice, ki je bila dodana.

```
// create SQL based on HTTP method
switch ($method) {
    case 'GET':
      $password = array_shift($request);
      $sql = "select * from `$table` WHERE UpIme='".
         $key."' AND Geslo='".$password."'"; break;
    case 'POST':
      $sql = "select * from `$table` WHERE UpIme='".
         $key."'";
      $exist = mysqli_query($link,$sql);
      if ($exist->num_rows) {
        echo -1;
        mysqli_close($link);
        die();
      }
      else {
        $sql = "insert into `$table` set $set";
      }
      break;
}
```

5.1.2 DatabaseConnector.java

Razred DatabaseConnector.java je v aplikaciji zadolžen za pošiljanje zahtevkov na strežnik in urejanje prejetih podatkov. Sestavljen je iz metod convertInputStreamToString, insertHidrant, preglejHidrant, getAllHidrantsInRadious, exportHidrants, exportPregledi, getOneHIdrant, deleteHidrant,

login in register.

Metodo convertInputStreamToString uporabimo v vseh ostalih metodah in zgolj pretvori InputStream, ki ga sprejmemo s strežnika v niz, ki ga uporabimo v metodi.

Metodo insertHidrant uporabljam, tako za ustvarjanje novega hidranta v bazi kot za posodabljanje obstoječega. Kot parameter metodi podamo podatke o hidrantu, ki jih združi v objekt JSON, od prvega parametra id pa je odvisno, ali posodobimo hidrant ali dodamo novega. V primeru, da je id enak 0, se ustvari nov zahtevek POST, v telesu katerega pošljemo podatke o hidrantu in ustvarimo nov hidrant. Če želimo posodobiti določene hidrant, kot prvi parameter podamo id tega hidranta in tako se ustvari zahtevek PUT s podatki o hidrantu ter posodobi hidrant.

```
public void insertHidrant(long id, String naziv,
   String ulica, String naselje, String obcina,
   double longitude, double latitude, String tip) {

        InputStream inputStream = null;
        String result = "";
        try {
            JSONObject newHidrant = new JSONObject()
               ;
            newHidrant.put("naziv", naziv);
            newHidrant.put("ulica", ulica);
            newHidrant.put("naselje", naselje);
            newHidrant.put("obcina", obcina);
            newHidrant.put("longitude", longitude);
            newHidrant.put("latitude", latitude);
            newHidrant.put("tip", tip);
            String message = newHidrant.toString();

            HttpClient httpclient = new
```

```
    DefaultHttpClient(); // Naredimo HTTP
     cliend
HttpResponse httpResponse;
if (id == 0){
    HttpPost httpPost = new HttpPost("
       http://hidranti.pgd-preddvor.si/
       api.php/Hidrants");
    httpPost.setEntity(new StringEntity(
       message, "UTF8"));
    httpPost.setHeader("Content-type", "
       application/json");
    httpResponse = httpclient.execute(
       httpPost); // Naredimo POST
       request
}
else {
    HttpPut httpPut = new HttpPut("http
       ://hidranti.pgd-preddvor.si/api.
       php/Hidrants/"+id);
    httpPut.setEntity(new StringEntity(
       message, "UTF8"));
    httpPut.setHeader("Content-type", "
       application/json");
    httpResponse = httpclient.execute(
       httpPut); // Naredimo PUT request
}
inputStream = httpResponse.getEntity().
   getContent(); // Response sprejmemo
   kot inputStream
if(inputStream != null){
    result = convertInputStreamToString(
```

```
                inputStream); // Pretvorimo
                inputStream v String
            }

            else
                result = "Not working!";

        } catch (Exception e) {
            Log.d("InputStream", e.
               getLocalizedMessage());
        }
}
```

Metoda preglejHidrant deluje podobno. Tako kot parametre prejme vse podatke o pregledu in id hidranta ki ga pregledujemo. Tako se ustvari zahtevek POST, ki doda nov vpis v tabelo Pregledi.

Tudi v metodo getAllHidrantsInRadious sta združeni dve funkciji. Če metodi kot parameter pošljemo zemljepisno dolžino in širino, po prejetju vseh hidrantov vrne samo tiste, ki so oddaljeni manj, kot znaša tretji parameter, ki sporoča oddaljenost. V primeru, da kot dolžino in širino pošljemo vrednost nič, metoda vrne vse hidrante, ne glede na oddaljenost. S tem sem tudi poskrbel, da v primeru, da nam ne uspe pridobiti lokacije GPS, še vedno dobimo seznam hidrantov.

Metodi exportHidrants in exportPregledi pridobita vse podatke o vseh hidrantih in vseh pregledih in jih vrneta kot niz vrednosti, ločenih z vejico.

Metoda getOneHIdrant kot parameter sprejme id hidranta in pridobi vse podatke o hidrantu in podatke o zadnjem pregledu. Te podatke vrne v obliki Cursorja, ki je kasneje uporabljen za prikaz podatkov o hidrantu.

Metoda deleteHidrant poskrbi, da iz podatkovne baze Hidrants izbriše hidrant z id-jem, ki ga podamo kot parameter.

Zadnji dve metodi sta nemenjeni upravljanju uporabnikov.

Prva metoda login kot parameter sprejme uporabniško ime in kodirano geslo ter na strežniku preveri, ali uporabnik obstaja. V primeru, da uporabnik obstaja, vrne objekt JSON z vsemi podatki o uporabniku.

Metoda register kot parameter prejme uporabniško ime, ime, priimek in kodirano geslo ter vrne številčno vrednost ki jo prejme s strežnika. Vrednost je v primeru uspešne registracije pozitivna, v nasprotnem primeru pa vrne -1.

5.2 Lokacija

Ker nas v primeru intervencije zanimajo samo hidranti, ki so v naši bližini, je treba razdaljo od nas do vsakega hidranta izračunati. Za to potrebujemo našo trenutno lokacijo, nato pa uporabimo prilagojen Pitagorov izrek, saj moramo upoštevati zaobljenost zemlje.

Če želimo pridobiti trenutno lokacijo, moramo najprej v AndroidManifest.xml zapisati pravice za uporabo lokacijskega senzorja, saj ga drugače aplikacija ne more uporabljati.

```
<uses-permission android:name="android.permission.
   ACCESS_FINE_LOCATION" />
```

Za pridobivanje lokacije uporabim razred TrackGPS.java, v katerem najprej preverim, ali ima aplikacija dostop aziroma ali lahko pridobi lokacijo, kar najprej preizkusim z manj natančnim pridobivanjem lokacije s pomočjo mobilnega omrežja, potem pa še s pomočjo senzorja GPS. V razredu TrackGPS.java sta še metodi getLongitude() in getLatitude(), ki vračata trenutno zemljepisno dolžino in širino in ju kličemo v aplikaciji.
Izračun oddaljenosti hidranta od trenutne lokacije:

```
double lonH = c.getDouble(2);  //get longitude from
   database
double latH = c.getDouble(3);  //get latitude from
   database
```

```
final int Re = 6371; // Radius of the earth

Double latDistance = Math.toRadians(latH - lati);
   //lati = my latitude
Double lonDistance = Math.toRadians(lonH - longi);
   //longi = my longitude
Double a = Math.sin(latDistance / 2) * Math.sin(
   latDistance / 2)
           + Math.cos(Math.toRadians(lati))* Math.cos
              (Math.toRadians(latH))
           * Math.sin(lonDistance / 2) * Math.sin(
              lonDistance / 2);
Double ce = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1
    - a));
Double distance = Re * ce * 1000; // convert to
   meters
```

5.3 Aplikacija - iskanje in pregled hidrantov

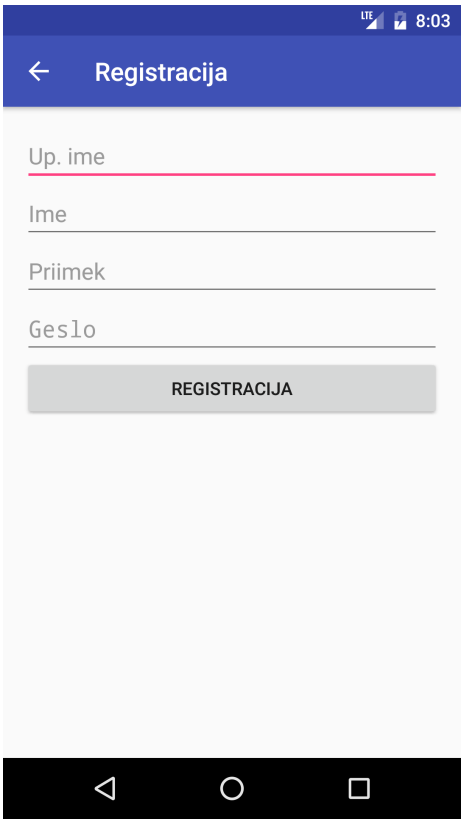
Aplikacija je sestavljena iz več zaslonov oziroma aktivnosti. Te so prijava, registracija, začetni zaslon, seznam hidrantov, prikaz enega hidranta, urejanje in dodajanje hidranta in nastavitve, ločeno pa še aktivnost za odštevanje časa notranjim napadalcem, ki jo predstavim posebej.

5.3.1 Prijava

Prijavna aktivnost se prikaže kot prva v primeru, da v aplikacijo še nismo prijavljeni. V vnosna polja moramo vnesti uporabniško ime in geslo ter pritisniti Prijava. V primeru napačnih podatkov se prikaže obvestilo, v primeru, da je prijava uspešna, pa nas preusmeri na glavno aktivnost. Ob tem se

tudi v SharedPreferences zapišejo vsi podatki o uporabniku, da so na voljo znotraj aplikacije. Ti podatki ostanjo shranjeni tudi, če aplikacijo zapremo in se ne odjavimo. Takrat se ob zagonu takoj odpre glavna aktivnost, ni se treba ponovno prijaviti.

Slika 5.1: Zaslonska slika prijavnega zaslona.



Slika 5.2: Zaslonska slika registracijskega zaslona.

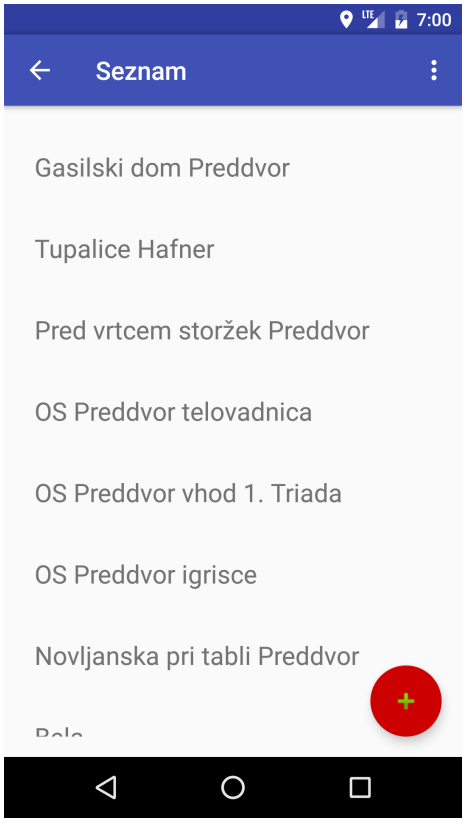
5.3.2 Registracija

V primeru, da uporabniškega imena in gesla še nimamo, se lahko registriramo. Oddati moramo svoje uporabniško ime, pravo ime in priimek ter geslo. V primeru napake ali če uporabnik že obstaja, se to izpiše na zaslonu. V primeru, da je registracija uspešna, prav tako dobimo obvestilo, hkrati pa nas preusmeri nazaj na prijavo, kjer se z novimi podatki prijavimo v aplikacijo.

5.3.3 Glavni zaslon

Začetni zaslon (slika 5.3) sem želel ohraniti čim bolj preprost, ker v času intervencije ni časa za iskanje pravega gumba. Glavni gumb je namenjen iskanju bližnjih hidrantov in je čim večji. Oblikovan je kot znak za hidrant z napisom "IŠČI HIDRANT". Ob kliku nanj se odpre seznam vseh hidrantov, ki so od nas oddaljeni manj, kakor smo nastavili v nastavitvah. Spodaj desno je plavajoči gumb za pregled celotnega seznama hidrantov, spodaj na sredini pa gumb za dostop do časovnika, če do njega ne dostopamo s pomočjo oznake NFC. V zgornji vrstici najdemo še povezavo do nastavitev.

Slika 5.3: Zaslonska slika glavnega zaslona.



Slika 5.4: Zaslonska slika seznama hidrantov.

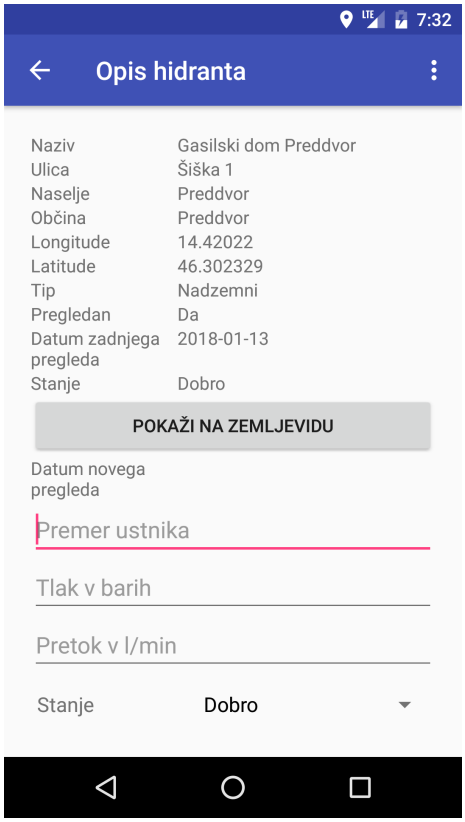
5.3.4 Seznam hidrantov

Na tem zaslonu (slika 5.4) se prikažejo vsi hidranti oziroma tisti, ki so znotraj določenega območja. Za prikaz tega seznama sem uporabil gradnik ListViewCompact, znotraj katerega dodajam vrstice, ki imajo svoje oblikovanje določeno v datoteki hidrant_list_item.xml. V razredu Seznam.java pokličem metodo databaseConnector.getAllHidrantsInRadious, ki vrne Cursor z vsemi hidranti, nato pa za vsakega dodam vrstico v seznam. Ob kliku na posamezni hidrant, se odpre nova aktivnost za pregled podatkov o posameznem hidrantu. Kot dodatek se pošlje še id tega hidranta, da vemo, katere podatke moramo prikazati. Spodaj desno se nahaja plavajoči gumb za dodajanje novega hidranta, enaka povezava pa se nahaja tudi zgoraj v meniju.

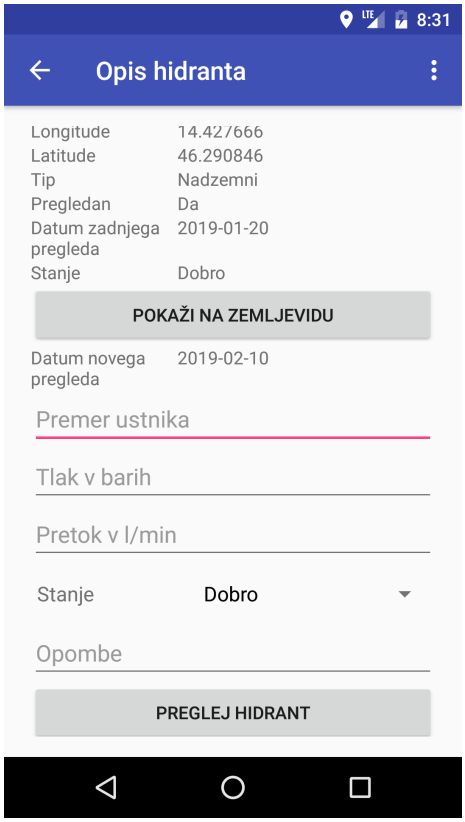
5.3.5 Pregled podatkov o hidrantu

Ob kliku na posamezni hidrant na seznamu se odpre podroben opis hidranta (slika 5.5). Vidimo lahko naziv hidranta, ulico, naselje ter občino, v kateri se nahaja. Prikazani sta tudi zemljepisna višina in širina ter tip hidranta. Podatke za izpis hidranta priskrbi metoda databaseConnector.getOneHidrant, ki vrne Cursor s podatki, med katerimi so tudi podatki o datumu zadnjega pregleda in stanju takrat. Pod vsemi temi podatki se nahaja gumb "Prikaži na zemljevidu". Ob kliku nanj se odpre aplikacija Google zemljevidi, znotraj katere se prikaže oznaka, kjer stoji hidrant.

Na zaslonu te aktivnosti se nahajajo tudi polja za vnos podatkov o pregledu hidranta (slika 5.6). To je polje za izbiro datuma, v katerem je že izpisan današnji datum, s klikom nanj pa izberemo nov datum. V naslednja tri polja vnesemo premer ročnika, tlak in pretok vode, ki smo izmerili pri testu hidranta. V zbirnem seznamu izberemo, ali je stanje hidranta dobro ali slabo, dodamo lahko opombe, ki so pomembne, kadar je stanje slabo, da pristojne službe vedo, kaj je treba popraviti, ter pritisnemo na gumb "Preglej hidrant". V tem trenutku se doda nov vpis v tabelo Pregledi, posodobi pa se tudi zgornji prikaz. Pri tem se dodata tudi ime in priimek osebe, ki je



Slika 5.5: Prikaz podatkov o hidrantu.



Slika 5.6: Polja za vnos podatkov o pregledu hidranta.

v aplikacijo prijavljena in je pregledala hidrant. Polj, namenjenih pregledu hidranta, ne vidimo v primeru, da nismo administratorji. To storim tako, da pri nalaganju pogleda preverim tip uporabnika in nepotrebna polja skrijem.

```
if (getSharedPreferences("User", MODE_PRIVATE).
   getBoolean("isAdmin",false) == false){
      premerEditText.setVisibility(View.GONE);
      tlakEditText.setVisibility(View.GONE);
      pretokEditText.setVisibility(View.GONE);
      opombeEditText.setVisibility(View.GONE);
      preglejTableRow.setVisibility(View.GONE);
      datumTableRow.setVisibility(View.GONE);
```

```
        stanjeLinearLayout.setVisibility(View.GONE);
}
```

5.3.6 Dodajanje in urejanje podatkov o hidrantu

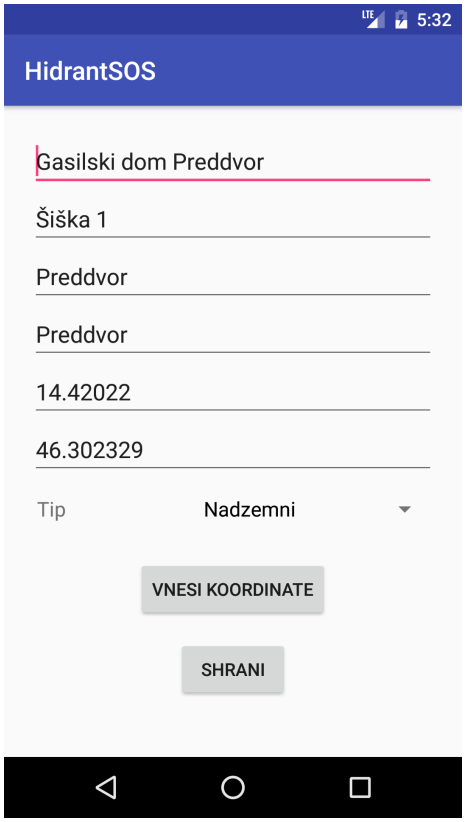
Ta aktivnost je enaka tako za dodajanje novega hidranta kot za urejanje obstoječega (slika 5.7). V aktivnosti na začetku preverim, ali so bili ob prejšnji aktivnosti, ob ukazu za zagon podani dodatni podatki o hidrantu. V primeru dodajanja novega hidranta teh podatkov nimamo, zato se naložijo prazna polja, v katera vnesemo podatke o hidrantu. V primeru urejanja obstoječega hidranta se znani podatki izpišejo v ustreznih poljih. Za pomoč pri dodajanju hidranta je tipka "Vnesi koordinate", ob kliku na katero pridobimo trenutno zemljepisno dolžino in širino in jo zapišemo v ustrezna polje. To nam zelo olajša vnos, če stojimo poleg hidranta.

Poleg urejanja lahko na prikazu posameznega hidranta izberemo tudi brisanje. Pri tem se pokaže opozorilno okno, ki nas opozori, da dejanja ne bo mogoče razveljaviti in bodo vsi podatki o hidrantu izbrisani, kot prikazuje slika 5.8.

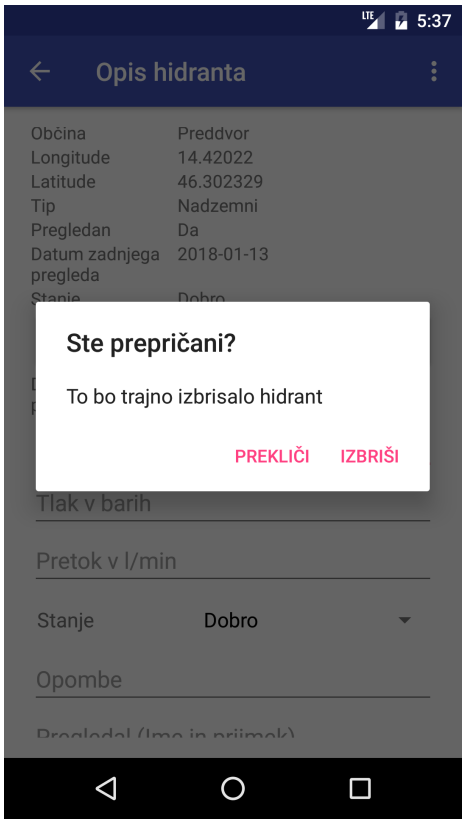
5.3.7 Nastavitve

V nastavitvah nastavljamo, v kakšnem območju želimo iskati hidrant. To je še posebej pomembno takrat, kadar posredujemo na različno gosto naseljenih predelih, zato se tudi gostota hidrantov razlikuje. V strnjenem naselju se je kot optimalna pokazala nastavitev od 100 do 200 metrov, na redkeje poseljenem območju pa je treba nastaviti tudi več kot 500 metrov, če želimo pridobiti podatke za vsaj en hidrant.

Poleg teh nastavitev je v nastavitvah tudi gumb za izvoz vseh podatkov o hidrantih in pregledih. Podatke aplikacija shrani na notranji pomnilnik telefona, v za to ustvarjeno mapo, ki jo ustvari ob prvi izvedbi izvoza. Podatki se zabeležijo v dve ločeni datoteki, poimenovani "HidrantIzvoz"ali "PregledIzovz"ter trenutni datum in ura, da se podatki med seboj ne prepisujejo,



Slika 5.7: Polja za urejanje in dodajanje hidranta.



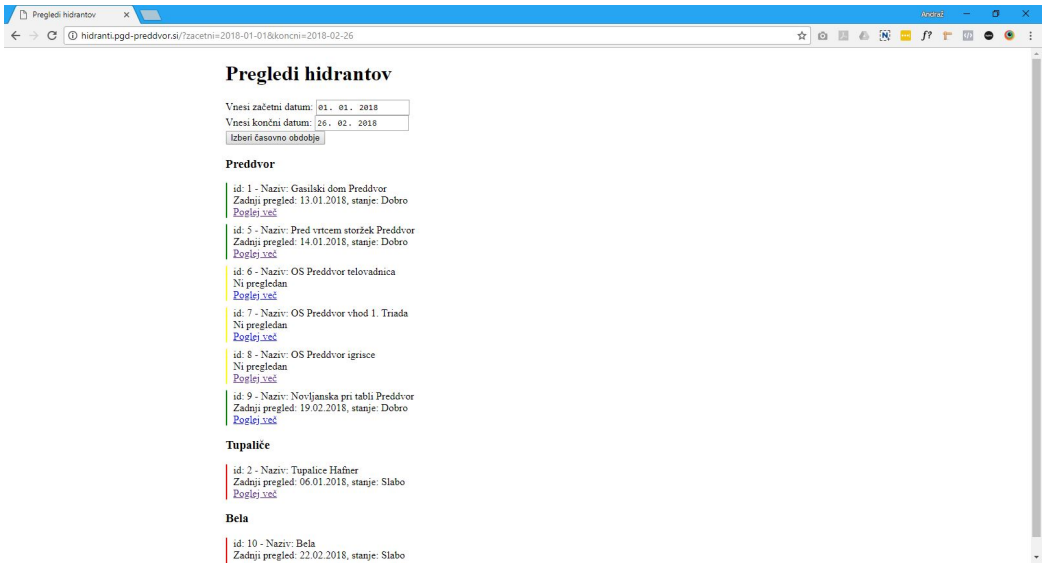
Slika 5.8: Opozorilo pred izbrisom hidranta.

vedno pa tudi lahko vidimo, kdaj smo naredili izvoz. Podatki so v formatu .csv, pri katerem so posamezni podatki ločeni z vejico.

5.4 Spletna stran za pregled hidrantov

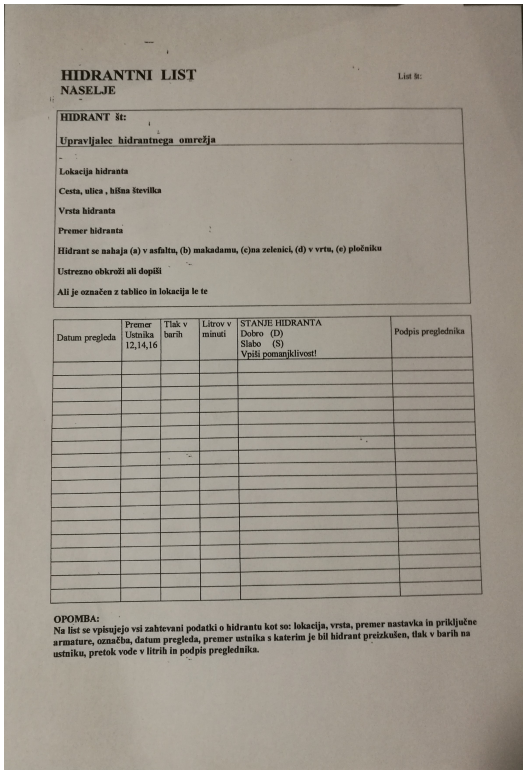
Poleg pregleda seznama hidrantov na mobilnem telefonu sem podobno aplikacijo razvil tudi za ogled na spletni strani (slika 5.9). V korenskem imeniku poddomene, kjer se izvaja tudi storitev REST, sem ustvaril dve spletni aplikaciji PHP. Prva stran index.php služi kot seznam vseh hidrantov. Razvrščeni so po naseljih, za lažje pregledovanje, poleg imena pa je prikazan tudi naslov. Pri vsakem hidrantu sta prikazana tudi datum zadnjega pregleda in stanje.

Stanje je označeno tudi z barvami na levi strani vrstice. Zelena pomeni, da je bil hidrant pregledan in je v dobrem stanju, rdeča pomeni, da je stanje slabo, če je črta rumena, hidrant še ni bil pregledan. Pogosto nas zanimajo pregledi znotraj določenega časovnega območja, npr. v zadnjem letu. Zato lahko nad seznamom hidrantov v dveh poljih izberemo začetni in končni datum ter s tipko "Izberi časovno obdobje"potrdimo izbiro. Datumi se zapišejo kot parametri naslova URL, seznam se posodobi, pod imeni hidrantov pa se prikažejo zadnji pregledi znotraj časovnega obdobja. Stanja so enako poudarjena z barvami.



Slika 5.9: Prikaz seznama hidrantov in oznak z barvami.

Ker sem želel popolnoma odpraviti arhiv v papirnati obliki, sem na drugi strani razvil podrobni pregled posameznega hidranta s podatki vseh pregledov. Pri tem sem se vizualno želel čim polj približati obliki (slika 5.10), ki smo jo uporabljali do zdaj, da tudi drugi čim lažje pregledujejo podatke (slika 5.11). Do seznama lahko pridemo tako, da kliknemo na ime hidranta ali napis "Poglej več". Pri tem se odpre nova stran, na kateri so vsi podatki o hidrantu ter seznam vseh pregledov. V primeru, da hidrant pregleda še nima opravljenega, se to izpiše.



HIDRANTNI LIST
NASELJE

List št:

HIDRANT št:

Upravljalec hidrantnega omrežja

Lokacija hidranta

Cesta, ulica , hišna številka

Vrsta hidranta

Premer hidranta

Hidrant se nahaja (a) v asfaltu, (b) makadamu, (c)na zelenici, (d) v vrtu, (e) pločniku

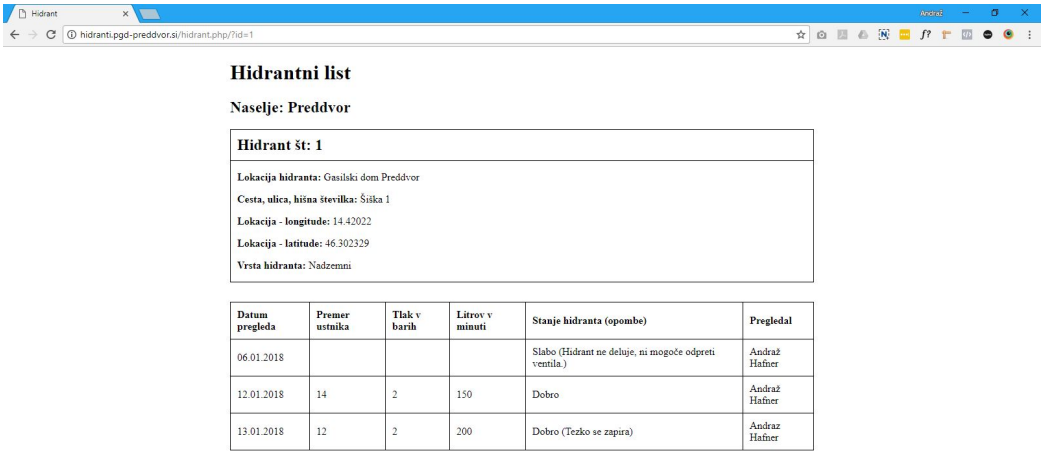
Ustrezno obkroži ali dopiši

Ali je označen z tablico in lokacija le te

Datum pregleda	Premer Ustnika 12,14,16	Tlak v barih	Litrov v minuti	STANJE HIDRANTA Dobro (D) Slabo (S) Vpiši pomanjkljivost!	Podpis preglednika

OPOMBA:
Na list se vpisujejo vsi zahtevani podatki o hidrantu kot so: lokacija, vrsta, premer nastavka in priključne armature, označba, datum pregleda, premer ustnika s katerim je bil hidrant preizkušen, tlak v barih na ustniku, pretok vode v litrih in podpis preglednika.

Slika 5.10: Hidrantni list, ki smo ga uporabljali do zdaj.



Hidrantni list

Naselje: Preddvor

Hidrant št: 1

Lokacija hidranta: Gasilski dom Preddvor
Cesta, ulica, hišna številka: Šiška 1
Lokacija - longitude: 14.42022
Lokacija - latitude: 46.302329
Vrsta hidranta: Nadzemni

Datum pregleda	Premer ustnika	Tlak v barih	Litrov v minuti	Stanje hidranta (opombe)	Pregledal
06.01.2018				Slabo (Hidrant ne deluje, ni mogoče odpreti ventila.)	Andraž Hafner
12.01.2018	14	2	150	Dobro	Andraž Hafner
13.01.2018	12	2	200	Dobro (Tezko se zapira)	Andraz Hafner

Slika 5.11: Prikaz podatkov in pregledov posameznega hidranta.

5.5 Aplikacija - časovnik notranjih napadalcev

Ob razvoju aplikacije za iskanje in pregledovanje hidrantov sem želel pohitriti in olajšati še eno opravilo ob intervenciji. To je odštevalnik časa, po katerem moramo preveriti stanje z notranjih napadalcev. Ob vsakem požari ali drugačni nesreči, pri kateri obstaja možnost vdihavanja nevarnih snovi, se gasilci opremijo z izolirnimi dihalnimi aparati, poleg katerih imajo jeklenko z zrakom, v kateri je omejena količina tega. Količina zraka zadošča za od 20 do 30 minut dela, odvisno od intenzivnosti. Pri tem je pomembno, da je vodja seznanjen s tem, kaj se dogaja v notranjosti objekta. Ker ima vodja veliko drugega dela, potrebuje opomnik, da po določenem času preveri stanje. Zdaj se uporablja posebna tabela na papirju, kamor se vpišejo imena, neto pa se nastavi ura, podobna tisti, ki jo imamo doma za kuhanje jajc. Ponavadi se nastavi na 10 minut, po preteku časa pa nas opomni z glasnim zvonjenjem. Kot največjo pomanjkljivost dosedanjega sistema vidim velikost table, ki se ponavadi po nastavitvi časa odloži nekje pri gasilskem tovornjaku, kjer se lahko presliši.

Zato sem razvil aktivnost znotraj aplikacije, pri kateri s pomočjo oznake NFC, ki je pritrjena na obleko gasilca, preberemo njegovo ime ter nastavimo časovnik. Ker imamo telefon vedno pri sebi, alarm težje preslišimo.

Za delovanje aplikacije sem najprej v datoteki AndroidManifest.xml nastavil, da se v primeru branja oznake NFC z vpisanim imenom samodejno zažene aplikacija in prava aktivnost ter dovoljenje za uporabo senzorja NFC v napravi.

```
<uses-permission android:name="android.permission
   .NFC" />

 <uses-feature
     android:name="android.hardware.nfc"
     android:required="true" />
```

```
<activity
    android:name=".TimerNapadalcev"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.nfc.action
            .NDEF_DISCOVERED" />

        <category android:name="android.intent.
            category.DEFAULT" />

        <data android:mimeType="text/plain" />
    </intent-filter>
</activity>
```

Po zagonu prave aktivnosti sta na vrhu strani izpisana ime in priimek, ki jih preberemo iz oznake NFC (slika 5.12). V polje vnesemo čas v minutah, po katerem želimo, da nas aplikacija opozori. Po začetku odštevanja imamo možnost to začasno ustaviti ter kasneje nadaljevati ali popolnoma ustaviti. Po preteku časa se to izpiše pod vnosnim poljem, sproži se zvočni alarm ter prikaže obvestilo v obvestilni vrstici (slika 5.13). Zvočni alarm se na izklopi ob ogledu obvestila, ob kliku nanj pa se zopet odpre aktivnost z odštevalnikom časa.

Za prikaz obvestila sem uporabil vgrajen razred Notification.Builder. Ustvaril sem ga v metodi, v kateri sem kot parameter dodal ime gasilca ter napis, koliko časa je poteklo. Nastavil sem mu še barvo obvestilne lučke ter vzorec vibriranja ob obvestilu. Za zvok obvestila sem uporabil privzeti zvok za alarm. Privzeta aktivnost, ki se odpre ob kliku, je TimerNapadalcev, obvestilo pa se izbriše ob kliku.

Slika 5.12: Prikaz imena, vpis minut in odštevanje časa.

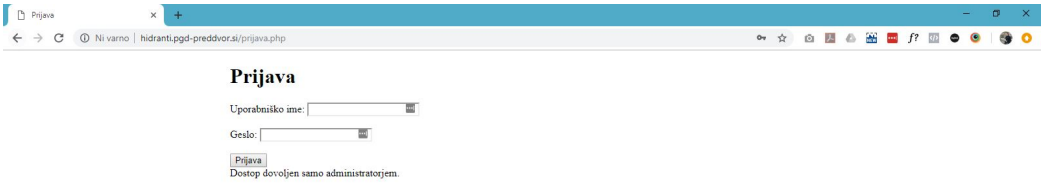
Slika 5.13: Obvestilo o preteku časa.

```
private Notification getNotification(String title,
   String content) {
        Notification.Builder builder = new
           Notification.Builder(this);
        builder.setContentTitle(title);
        builder.setContentText(content);
        builder.setLights(Color.BLUE, 500, 500);
        long[] pattern =
           {500,500,500,500,500,500,500,500,500};
        builder.setVibrate(pattern);
        Uri uri = RingtoneManager.getDefaultUri(
```

```
            RingtoneManager.TYPE_ALARM);
        builder.setSound(uri);
        Intent myintent = new Intent(this,
            TimerNapadalcev.class);
        PendingIntent contentIntent = PendingIntent.
            getActivity(this, 0,
                  myintent, PendingIntent.
                     FLAG_UPDATE_CURRENT);
        builder.setContentIntent(contentIntent);
        builder.setAutoCancel(true);
        return builder.build();
}
```

5.6 Spletna stran - upravljanje uporabnikov

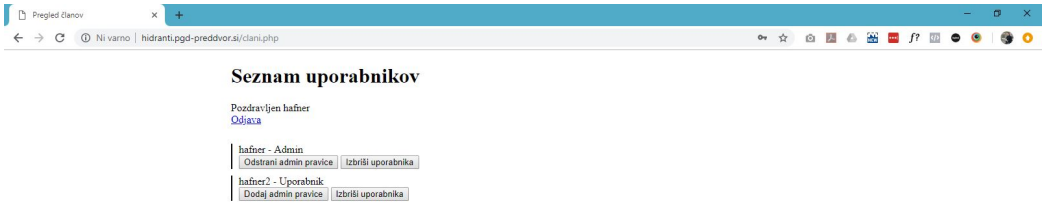
Za namene upravljanja z uporabniki je bil razvit namenski uporabniški vmesnik, v katerega se lahko prijavijo administratorji. Sestavljen je iz strani prijava.php, clani.php in logout.php. Na strani prijava (slika 5.14) so polja za vnos uporabniškega imena in gesla ter gumb za prijavo. V primeru napake (nepravilno uporabniško ime ali geslo, uporabnik ni administrator) se ta izpiše pod gumbom. Ko vnesemo pravo uporabniško ime in geslo, se nastavi sejna spremenljivka uporabnik, odpre pa se stran s seznamom uporabnikov.



Slika 5.14: Prijavna stran za upravljanje uporabnikov.

Na strani člani je tabela s seznamom uporabnikov (slika 5.15). Pri vsakem je možnost spreminjanje njegove vloge (administrator/uporabnik) ter

brisanja. Stran je zaščitena pred nepooblaščenim dostopom, tako da pred odprtjem preveri sejno spremenljivko in nas v primeru, da ni nastavljena, preusmeri na prijavno stran. Gumbi za spreminjanje pravic in brisanje so narejeni kot FORM z dvema skritima poljema (ID uporabnika in trenutna vrednost pravic) ter dvema gumboma SUBMIT. Nato se na podlagi gumba SUBMIT ivede pravi stavek SQL.



Slika 5.15: Prikaz uporabnikov in gumbi za spremembo in izbris.

Odjava je rešena povsem preprosto. Z obiskom spletne strani na naslovu logout.php se izbriše sejna spremenljivka in nas preusmeri na prijavno stran.

Poglavje 6

Sklepne ugotovitve

V sklopu diplomskega dela sta bili razviti androidna in spletna aplikacija za pomoč gasilcem pri njihovem delu. Razvit je bil sistem za vnos hidrantov in njihovo pregledovanje in nadzor. Sistem je bil delno preizkušen v praksi in se je izkazal za učinkovitega, saj zdaj pregled in izpolnjevanje hidrantnega lista vzameta manj časa kot prej. Tudi pregledovanje in pregled, ali smo pregledali vse hidrante, sta hitrejša saj se na spletni strani nepregledani hidranti obarvajo rumeno.

Poleg tega sistema je bil razvit tudi sistem za iskanje najbližjega hidranta. Tudi tega smo preizkusili na gasilski vaji v gosto naseljenem območju. Ker se pri iskanju hidranta vedno osredotočamo na ulico, v kateri se nahajamo, je koristno da aplikacija meri oddaljenost glede na zračno razdaljo. Tako se je izkazalo, da je hidrant v sosednji ulici bližje od tega v ulici, v kateri se nahajamo.

Kot zadnjo aktivnost sem razvil aplikacijo za odštevanje časa notranjim napadalcem. Ta nekako združi v celoto najnujnejši stvari na intervenciji, preskrbo z gasilnim sredstvom in preskrbo gasilca. Aplikacijo smo preizkusili tudi v praksi, kjer se je pokazala kakšna možna izboljšava, potrebujemo pa tudi nekaj časa, da se navadimo na njeno uporabo in nam postane rutina.

6.1 Nadaljni razvoj

Kot sem že omenil, se je z uporabo pokazalo še nekaj dodatnih možnosti za izboljšave in nadaljni razvoj. To bi bile:

- dodati možnost odštevanja časa večim napadalnim skupinam naenkrat;
- shranjevati čase vstopov notranjih napadalcev za kasnejšo analizo;
- izboljšati grafični vmesnik;
- predpomniti podatke o hidrantih, za primer izgube mobilnega signala.

Literatura

[1] Android. Dosegljivo: https://en.wikipedia.org/wiki/Android_Oreo. [Dostopano: 4. 1. 2017].

[2] Android studio. Dosegljivo: https://developer.android.com/studio/intro/index.html. [Dostopano: 4. 1. 2017].

[3] Gps. Dosegljivo: https://sl.wikipedia.org/wiki/Globalni_sistem_pozicioniranja. [Dostopano: 4. 1. 2017].

[4] Java. Dosegljivo: https://sl.wikipedia.org/wiki/Programski_jezik_java. [Dostopano: 4. 1. 2017].

[5] Nfc. Dosegljivo: https://sl.wikipedia.org/wiki/Near_Field_Communication. [Dostopano: 24. 2. 2018].

[6] Php. Dosegljivo: https://sl.wikipedia.org/wiki/PHP. [Dostopano: 24. 2. 2018].

[7] Rest. Dosegljivo: http://eprints.fri.uni-lj.si/2561/1/63080164-MATJA\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{Z\global\mathchardef\accent@spacefactor\spacefactor}\accent20Z\egroup\spacefactor\accent@spacefactor_RAJNAR-Primerjava_uporabe_SOAP_in_REST_za_potrebe_povezave_mobilnih_naprav_s_spletnimi_storitvami.pdf. [Dostopano: 29. 7. 2017].

[8] Rest api. Dosegljivo: https://github.com/mevdschee/php-crud-api. [Dostopano: 24. 2. 2018].

[9] Sql. Dosegljivo: https://sl.wikipedia.org/wiki/SQL. [Dostopano: 29. 7. 2017].

[10] Sublime. Dosegljivo: https://www.sublimetext.com/. [Dostopano: 24. 2. 2018].