Mladi za napredek Maribora 2018

35. srečanje

PREDSTAVITEV MOBILNE APLIKACIJE FireNav

Inovacijski predlog: Računalništvo

Avtor: TOMAS KOVAČIČ, PATRICK KAČIČ, MATIC ŠUŠTERIČ
Mentor: ZDRAVKO KAČIČ
Šola: SREDNJA ELEKTRO-RAČUNALNIŠKA ŠOLA MARIBOR

Maribor, 8.1.2018

Mladi za napredek Maribora 2018

35. srečanje

PREDSTAVITEV MOBILNE APLIKACIJE FireNav

Inovacijski predlog: Računalništvo

Maribor, 8.1.2018

KAZALO

KAZALO SLIK

ZAHVALA

Najprej bi se radi zahvalili našemu mentorju, ki nam je z njegovim znanjem in izkušnjami pomagal pri pisanju naloge, kot tudi pri izvedbi samega sistema oz. aplikacije. Brez njegove pomoči ne bi naredili tako uspešne naloge.

1. Povzetek

Idejo za izdelavo inovacijskega predloga, ki je izveden v obliki mobilne aplikacije, smo dobili od gasilskega društva, kjer so nam razložili, da večina gasilskih društev nima ustrezno poenotenega informacijsko-komunikacijskega sistema za pomoč pri intervenciji. Zato smo analizirali, na katerih področjih dela se še pojavlja nuja po takšnem sistemu. Pri ugotavljanju, kakšen informacijsko-komunikacijski sistem gasilska društva trenutno uporabljajo, smo prišli do zaključka, da bi s predloženim inovacijskim predlogom lahko prispevali predvsem pri izboljšanju sistema avtomatske navigacije, ki ga gasilci uporabljajo kot pomoč pri vožnji na kraj izrednega dogodka. Trenutno večina gasilskih društev uporablja storitev, ki prestreže poziv od centra za obveščanje in ga namesto na pozivnik, pošlje v obliki SMS-sporočila na mobilni telefon. Ob tem uporabljajo še različne druge aplikacije, kot je npr. FireAlert, ki ob sprejemu pravilnega SMS-sporočila sproži alarm. Kaj pa po sproženem alarmu? Trenutno gasilci prispejo do gasilskega doma, kjer se hitro preoblečejo in usedejo v gasilsko vozilo, nato pa morajo ročno vpisati naslov nesreče v eno od svojih navigacijskih naprav (telefon, garmin…), kar je pogosto lahko zamudno opravilo in lahko povzroči nepotrebne zamude pri prispetju na kraj dogodka. Kot rešitev omenjenega problema podajamo inovacijski predlog, s katerim predlagamo avtomatizacijo postopka sprejema sporočila o izrednem dogodku iz centra za obveščanje in vnosa podatkov v izbran navigacijski sistem. Razvita aplikacija po sprejemu SMS-sporočila (v tem primeru SMS od centra za obveščanje), opravi analizo besedila sporočila s postopki avtomatske obdelave besedil in iz prejete vsebine sporočila izloči naslov, ki ga nato pretvori v obliko zapisa URL, kot vhodnega podatka za aplikacijo Google Maps ter sproži avtomatsko navigacijo. Razvita aplikacija je namenjena prvenstveno uporabi na tablici z operacijskim sistemom Android, ki bi bila nameščena v gasilsko vozilo, uporabiti pa jo je mogoče tudi na mobilnih telefonih z operacijskim sistemom Android. Aplikacijo smo razvili v programskem jeziku Java, ki je za android najbolj primeren. Izvedbo aplikacije smo zastavili tako, da smo najprej definirali funkcionalnost sistema, to nato uvedli v diagram poteka, s pomočjo tega pa nato izvedli programsko kodo za ciljno napravo.

Summary

The idea to make the innovation proposal – the mobile application - came from a local firefighter community, where they gave us a hint about having nothing automated to help them with interventions. That is where we kicked in. After a lot of talking with our team about what should be automated in the first place, we came to a conclusion, that the response time to the place of an incident was the thing we needed to improve. Even today every firefighting department uses a service, which intercepts the notification, and sends it to a firefighter in the shape of a text message instead of using a pager. Firefighters are already using an app called “FireAlert”, which makes their job easier, as the text message they receive triggers an alarm, which notifies them about the incident in a case they miss the

text message sound. What comes next? Until today firefighters had to come to the fire station, where they changed clothes, sat in to the firetruck, then they had to manually insert the address of the incident in to one of their navigation devices (telephone, Garmin…). Here we started developing an application, which works as a text message interceptor. Upon receiving a text message in the right format (in this case a standardized text message format from the PSAP), the application goes through the text message and separates the address from it, transforms it in to a Google Maps URL and automatically gives you directions to the place of the incident. The application would be active on a (preferably) tablet with an Android OS, which would be mounted inside an emergency vehicle. We used Java programming language, as it tends to work with the Android better than any other programming languages. The workflow contained a rough idea of what the application was supposed to look like. After a lot of thinking, we transformed that idea into a flowchart, from the flowchart directly into the code and after developing the code we tested it on the Android device.

2. Problem

Poznate občutek, ko je vsaka sekunda pomembna. To je še posebej pomembno v situacijah, ko so ogrožena človeška življenja in je pomembna vsaka sekunda. Gasilci morajo pri reševanju upoštevati veliko ključnih navodil, zaradi katerih je reševanje hitro oz. predvsem uspešno. Ko se mora gasilec na primer ob drugi uri zjutraj zbuditi in pohiteti do zbirnega mesta v gasilskem domu, je zelo težko ostati pozoren in fokusiran na vse te postopke in navodila, ob tem pa ostati še zbran pri čim hitrejšem vpisovanju in iskanju navigacije za naslov? Ob upoštevanju tega smo prepoznali problem, ki ga bomo rešili z ustreznim sistemom.

Današnji svet sloni na tehnologiji in v naših življenjih so vse bolj prisotni mobilni telefoni, zato smo se odločili, da bomo predlagan sistem izvedli v obliki mobilne aplikacije, ki bo nameščena na mobilni napravi v reševalnem vozilu in bo omogočala avtomatsko nastavljanje navigacije do kraja izrednega dogodka ter s tem pridobili dragocen čas, ki je v teh primerih še kako pomemben, predvsem pa bomo gasilcem omogočili boljšo zbranost, saj jim ne bo treba ubadati z iskanjem in vstavljanjem naslova izrednega dogodka.

Predlagan sistem bo preko pozivnika oz. sporočila centra za obveščanje Republike Slovenije dobil potrebne informacije. Bolj specifično gre za to, da bo iz SMS-sporočila razbral lokacijo izrednega dogodka in jo avtomatsko pretvoril v ustrezno obliko zapisa, ki omogoča navigacijo do kraja dogodka.

3 Funkcionalnost sistema

Ko smo načrtovali sistem smo vedeli, da mora biti sistem pri delovanju zelo zanesljiv in ne sme priti do nobenih napak, saj so podatki, ki jih sistem ponuja ključnega pomena za uspešno reševanje v primeru nesreč. Delovanje sistema smo razdelili na 4 dele. Od pridobivanja specifičnega SMS-sporočila, pa vse do končnega zapisa, ki predstavlja vhodni podatek navigacijskega sistema.

Gasilci so še pred kratkim uporabljali pozivnike, na katere so prejemali obvestila o nesrečah, nato pa so namesto pozivnikov pričeli uporabljati svoje pametne telefone s pomočjo storitve Intervencije.net. Ta storitev pošilja SMS-sporočila na mobilne telefone. Sporočila, poslana na mobilne telefone so enaka, kot sporočila, poslana na pozivnike.

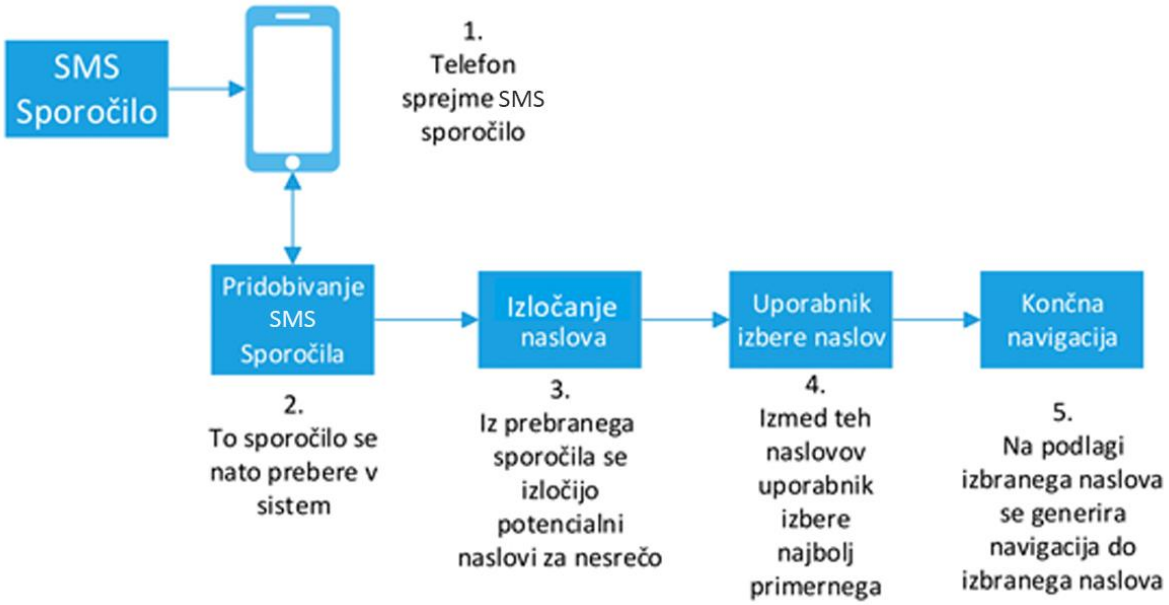
Slika 1: Posredovanje sporočil na pozivnik in mobilni telefon [1]

Kaj se torej zgodi ob delovanju predlaganega sistema. Sistem kot vhodni podatek vzame pridobljeno SMS-sporočilo, ki vsebuje podatke o naslovu nesreče. Izhodni podatek sistema pa je naslov končnega cilja navigacije, ki je bil pridobljen iz SMS-sporočila. Da pridemo do končnega rezultata oz. cilja je potrebno iti skozi vsako fazo obdelave z vhodnim podatkom (SMS-sporočilo). Pri obdelavi SMS-sporočila je pomembna struktura sporočila, ki ga sistem prejme. Sporočilo je sestavljeno iz več ključnih podatkov, ki pripomorejo pri obdelavi npr.:

POZAR – RDECI BREG 23 -- Posilja: ReCo-122, RIC:0890297, IL:059074870

To je primer SMS-sporočila, ki pride v sistem. Struktura tega sporočila je zgrajena iz treh pomembnih delov. Prvi del je na začetku sporočila in sporoča, za kakšno nesrečo gre. Ta del sporočila je omejen s pomišljajem. Za pomišljajem je napisana vsebina sporočila. Ta vsebuje tudi ključni podatek – naslov izrednega dogodka. Vsebina se konča z dvema pomišljajema in besedo »Posilja«. Če SMS- sporočilo nima takšne strukture, sistem javi napako, saj je zgrajen ob domnevi, da ima vhodno sporočilo opisano strukturo. To pa pomeni, da sistem ni namenjen zgolj gasilcem, pač pa ga lahko uporabimo za kakršnekoli namene. Pri tem je pomembno le, da ima sporočilo opisano strukturo.

Ko sistem iz SMS-sporočila izloči pravilen naslov, ga nato pretvori v obliko, ki omogoča zagon navigacijskega sistema.



Slika 2: Diagram poteka načrtovanega sistema

4 Platforma Android

Preden smo začeli z izdelavo aplikacije, smo morali določiti, na kateri platformi bo ta delovala. Odločitev ni bila težka, saj je **Android** po statistiki Statista [2] leta 2015 pokrival 81.2% uporabnikov.

4.1 Kaj je platforma Android

Platforma Android je operacijski sistem za pametne mobilne telefone ter ostale prenosne naprave, ki temelji na jedru Linux. Razvija ga Google v sodelovanju s podjetji združenja Open Handset Alliance (OHA). Android je v prvi vrsti namenjen mobilnim telefonom, začenja pa se tudi njegova uporaba v drugih napravah, kot so: pametne ure, avtomobili, televizije itd. Večina ljudi, ki so že slišali za platformo Android, zmotno misli, da je platforma Android namenjena zgolj zahtevnejšim uporabnikom. Temu ni tako. Platforma Android je namenjena tudi uporabnikom, ki nimajo veliko predznanja in izkušenj z napravami, kot so npr. namizni/prenosni računalniki, mobilni telefoni itd., ki omogočajo uporabo aplikacij preko različnih kompleksnih interaktivnih programskih vmesnikov.[3]

4.2. Zgodovina

Android Inc. so ustanovili Andy Rubin, Rich Miner, Nick Sears in Cris Whiteeta, leta 2003 v Kalifornij, v mestu Palo Alto. Leta 2005 je podjetje prevzelo podjetje Google, ki je želelo

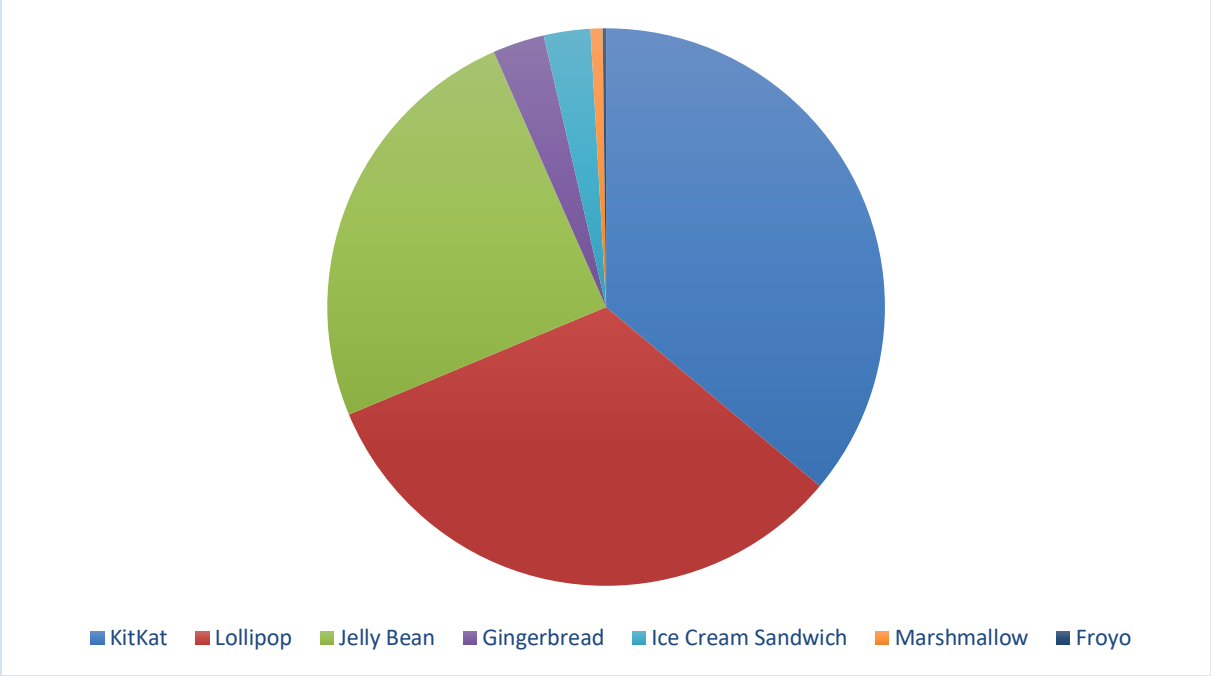
vstopiti na trg mobilnih naprav. Takrat so pričeli z razvojem operacijskega sistema za mobilne naprave, ki bi naj temeljil na jedru Linux.[4]

Prihod odprtokodne mobilne platforme Android so objavili novembra 2007 z ustanovitvijo združenja Open Handset Alliance (OHA). V to združenje so danes vključena velika podjetja s področja proizvodnje računalniških naprav, programske opreme, mobilni operaterji in ponudniki mobilnih telefonov. Leta 2007 je sledila tudi prva izdaja programskega orodja. Avgusta 2008 je bil napovedan trg aplikacij Android Market, na katerem je danes vse več aplikacij in mobilnih iger [5]

Platforma Android je od nastanka pa do danes doživela veliko nadgradenj. Vsaka novejša različica prinaša večjo podporo širokemu naboru strojne opreme in številne izboljšave, ki so podrobneje prikazane v spodnji tabeli 1, odstotna uporaba pa je prikazana s tortnim grafikonom na sliki 3.[4]

VERZIJA	DODATKI/POPRAVKI
Android 1.6 – Donut	Android Market omogoči lažje iskanje, pretvorba besedila v govor, podporo za zaslon WVGA resolucije.
Android 2.1 - Eclair	E-pošta (več računov hkrati), Bluetooth 2.1, SMS, fotoaparat (digitalni zoom, flash etc.), virtualna tipkovnica, brskalnik (HTML5), Google Maps 3.1.2.
Android 2.2 - Froyo	Hitrost, spomin, integriranje Chrome z JavaScript, izboljšana podpora Microsoft Exchange, USB povezovanje, preklapljanje med jeziki, klicanje ob podpori Bluetootha, Adobe Flash, namestitev aplikacij na pomnilniško kartico, WiFi hotspot, Android Market samostojno posodabljanje aplikacij.
Android 2.3 - Gingerbread	Podpora za veliko ločljivost (WXGA), SIP VoIP, izboljšana virtualna tipkovnica, kopiranje in lepljenje besed, Bass Boost, upravljanje porabe baterije, Download Manager, video format VP8. Dodana nova senzorja giroskop in barometer.
Android 3.0 - Honeycomb	Optimiziran uporabniški vmesnik, hiter dostop do obvestil, večopravilnost, poenostavljeno kopiranje in lepljenje, fotoaparat (fokus, bliskavica, zoom, sprednja kamera), galerija v celozaslonskem načinu, videoklepet Google Talk, izboljšana strojna oprema, podpora za večjedrne procesorje.
Android 4.0 – Ice Cream Sandwich	Obilica popravkov in dodatnih funkcionalnosti sistema; izstopajo strojno pospešen grafični vmesnik, prenova grafičnega vmesnika, odklepanje s prepoznavo obraza, nov spletni brskalnik, izboljšana aplikacija za kamero, nadzor uporabe

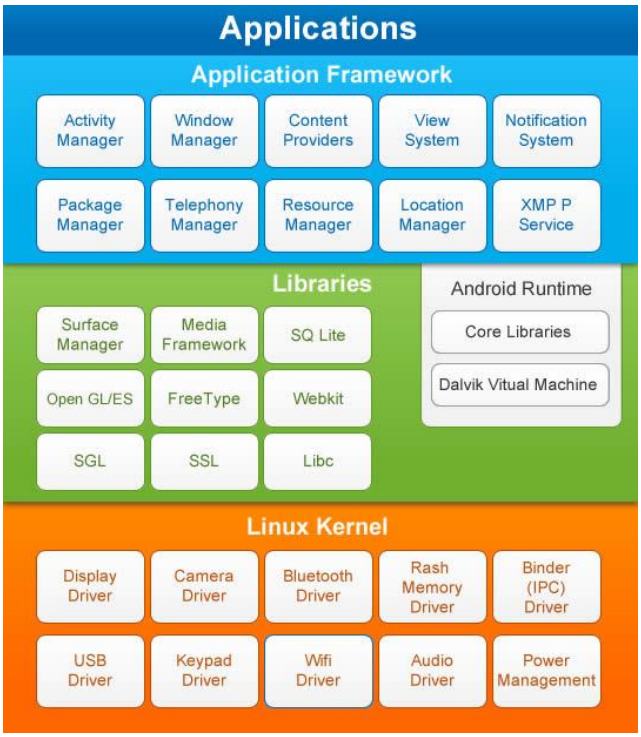
	podatkov itd.
Android 4.1 – Jelly Bean	Omejen dostop do aplikacij – starševski nadzor, Bluetooth smart support, avtomatsko predlaganje pri pisanju, Support za Hebrejski in arabski jezik, izboljšan zvok.
Android 4.4 - KitKat	Glasovno upravljanje »Ok Google«, Celozaslonski način dela, pametni imenik, izboljšana optimizacija podatkov in hitrost sistema, ID klicanje, nov način SMS, Dodani Emoji, povezava med tiskalniki, Quickoffice.
Android 5.0 - Lollipop	Novi Material Design, upravljanje na različnih napravah, obvestila na zaklenjenem zaslonu, izboljšan način uporabe baterije, dodatni jeziki, Tap & play itd.
Android 6.0 - Marshmallow	Kontekstno iskanje iz ključnih besed v aplikacijah, Doze mode, USB-C podpora, dovoljenja se prikažejo ob vsaki aplikaciji posebej in še veliko več.



Slika 3: Najbolj uporabljane različice Android OS (1.4.2015)

4.3 Arhitektura platforme Android

Odprtokodno platformo Android v grobem sestavlja pet slojev (slika 4). Vsak sloj lahko uporablja storitve, ki jih ponuja sloj pod njim [6].



Slika 4: Arhitektura operacijskega sistema Android [7]

Celotni operacijski sistem Android je zgrajen na osnovi jedra Linux, ki predstavlja osnovni sloj z nekaterimi dodatnimi arhitekturnimi spremembami, ki jih je naredil Google. Jedro deluje kot vmesna plast med strojno opremo in programskim delom. Med pomembnejše osnovne funkcije, ki jih Android uporablja z jedrom, spadajo upravljanje s pomnilnikom, upravljanje procesov, mrežna podpora, gonilniki in varnost [6].

Naslednji sloj so knjižnice, ki omogočajo obdelavo različnih vrst podatkov. Knjižnice so napisane v programskem jeziku C/C++ in so specifične za določeno strojno opremo. Pomembnejše knjižnico so: Surface Manager, Media framework, SQLite, SGL, SSL. V sloju knjižnic je tudi zagonsko okolje [6].

Glavni del zagonskega okolja predstavlja ART, ki izboljšuje splošno učinkovitost izvajanja in s tem zmanjšuje porabo energije, prinaša pa hitrejše izvajanje aplikacij ter izboljšavo delovanja pomnilnika [6].

Naslednji sloj je aplikacijsko ogrodje, ki omogoča razvijalcem, da so razvite aplikacije prikazane enako na vseh mobilnih napravah s podporo operacijskega sistema Android. Omogoča tudi dostop do knjižnic iz naslednje ravni. Pomembnejši bloki, ki sestavljajo aplikacijsko ogrodje so:

- **Upravljanje aktivnosti (ang. Activity Manager):** upravlja aktivnost življenjskega cikla posamezne aplikacije in omogoča komunikacijo med njimi.
- **Ponudnik vsebin (ang. Content Provider):** aplikacijam omogoča dostop do deljenih podatkov, kot so npr. stiki, in omogoča deljenje podatkov med aplikacijami.

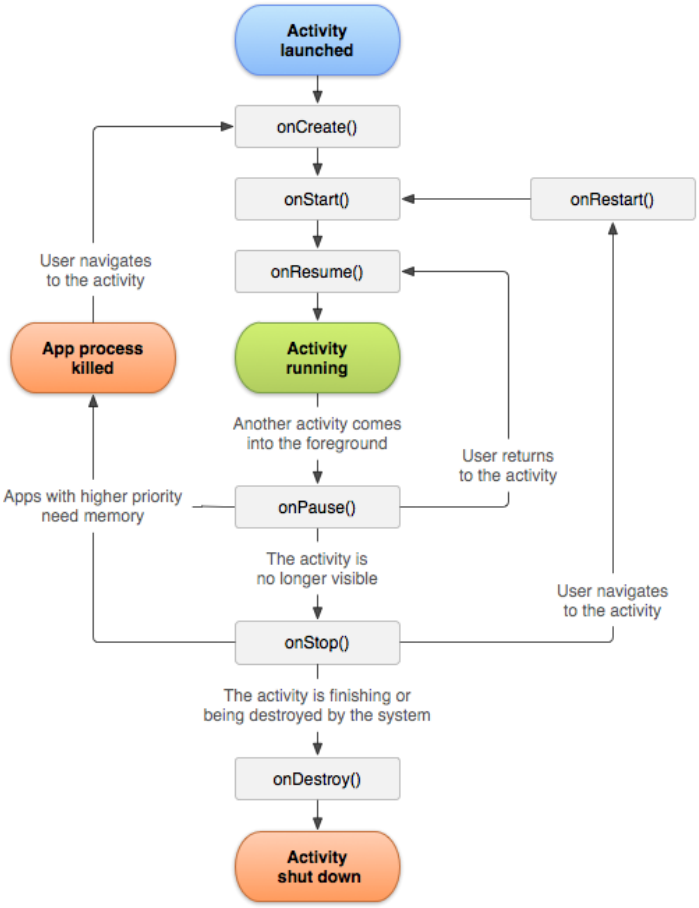
- **Upravitelj telefonije (ang. Telephony Manager):** vsebuje programske vmesnike, namenjene telefoniji.
- **Upravitelj virov (ang. Resource Manager):** zagotavlja dostop do virov, ki niso del aplikacije, kot so: lokalni nizi, grafični elementi, razporejevalci vsebin.
- **Upravitelj virov (ang. Notification Manager):** skrbi za prikazovanje obvestil v vrstici stanja sistema.
- **Pogledi (ang. Views):** so osnovni gradniki uporabniškega vmesnika. To so razni gumbi, tekstovna polja, mreže in navsezadnje tudi brskalniki.

Aplikacije so najvišji sloj arhitekture operacijskega sistema Android. Vključene so že v osnovni različici Androida ob nakupu pametnega mobilnega telefona. To so npr. sporočanje, e-pošta, budilka, koledar, stiki, zemljevid in podobno. Vse aplikacije so napisane v programskem jeziku Java [6].

4.4 Življenjski cikel aktivnosti Android aplikacije

Koncept aktivnost predstavlja posamezni zaslon aplikacije z uporabniškim vmesnikom, na primer seznam prejetih sporočil. Druga aktivnost lahko vsebuje vsebino sporočil, tretja pa na primer izbiro prejemnika posameznega sporočila. Aktivnost ima štiri pomemba stanja življenjske dobe. Med preklopom na drugo aktivnost, se aktivnost, ki je trenutno delovala, neha izvajati in jo sistem potisne v zgodovinski sklad. S tem ohrani podatke za kasnejše nadaljnjo izvajanja. Aktivnost lahko iz sklada tudi odstranimo, če je več ne potrebujemo ali pa nočemo po nepotrebnem polniti pomnilnika. Življenjski cikel aktivnosti je prikazan na sliki 5. Štiri pomembnejša stanja življenjske dobe aktivnosti so:

- **Aktivna aktivnost (ang. Active activity):** aktivnost je prikazana na zaslonu v ospredju in se odziva na uporabnikove akcije (dotik zaslona, pritiske tipk …).
- **Prekinjena aktivnost (ang. Paused activity):** aktivnost se ne odziva na uporabnikove akcije, vendar je še vedno vsaj delno vidna na zaslonu (izbira opcij, potrjevanje dialogov …).
- **Ustavljena aktivnost (ang. Stopped activity):** aktivnost na zaslonu ni več vidna, a je še vedno na zgodovinskem skladu.
- **Uničena aktivnost (ang. Destroyed activity):** aktivnost ni na zaslonu in je tudi ni več na zgodovinskem skladu.



Slika 5: Življenjski cikel aktivnosti aplikacije Android [8].

4.5 Prednosti platforme Android

Največja prednost platforme Android v primerjavi z ostalimi je v tem, da je odprtokodna in brezplačna. To je dobro tudi za uporabnike, saj imajo na razpolago mobilne naprave s še več funkcionalnostmi in večje število aplikacij, od tega večina brezplačnih. Domači proizvajalci lahko ceneje in lažje realizirajo svoje ideje in jih ponudijo na trg. Razvijalci lahko koristijo vse možnosti strojnih platform, kot so npr. zaslon na dotik, GPS, kamera, mikrofon, kompas ipd. Aplikacije lahko vključujejo module izvedene v jeziku Java, JavaScript, HTML in CSS. Za to je na voljo tudi brezplačno razvojno okolje SDK (Software development kit). Vse naprave se samodejno sinhronizirajo z Googlovimi storitvami in omogočajo enostavno posodobitev, ko je na voljo novejša različica [9].

5 Zagonsko in razvojno okolje

Za razvoj Android aplikacij potrebujemo zagonsko in razvojno okolje. Med razvojna okolja sodita tudi Eclipse, z vtičnikom ADT in Android Studio. Za testiranje aplikacij lahko uporabljamo navidezno napravo, ki je vključena v programskem paketu Android SDK, ali pa uporabljamo kar dejanske naprave z operacijskim sistemom Android.

5.1 Razvojno okolje Android Studio

Razvojno okolje Android studio (slika 6) je odprtokodni in brezplačni program, ki ga je razvilo podjetje Google. Namenjen je razvoju aplikacij na platformi Android. Prvič je bil predstavljen 13.5.2013 na konferenci Google I/O. Temelji na programski opremi JetBrains IntelliJ IDEA in je namenjen za razvoj Android aplikacij. Prva stabilna različica je bila »Android Studio v1.0«, ki je bila izdana v decembru 2014, trenutna različica pa je »Android studio v2.0 Preview 5« in je bila izdana 14 januarja 2016. Razvojno okolje Android Studio je na voljo za operacijske sisteme Windows, Linux in Mac OS X. [10]

Slika 6: Logotip razvojnega okolja Android Studio[11]

5.2 Android SDK

Android SDK je programski paket, ki vsebuje celovit paket orodji za razvijalce. Vsebuje izvorno kodo, razvojno orodje, navidezno napravo (Emulator), dokumentacijo, razhroščevalnik, knjižnice, ki so ključnega pomena pri izgradnji Android aplikacij. Knjižnice so napisane v programskem jeziku Java, poganja pa jih navidezni stroj ART. Navidezna naprava (Emulator) omogoča izvajanje aplikacije, ki jo razvijamo na navidezni ciljni napravi in je v veliko pomoč pri testiranju aplikacij. V programskem paketu so na začetku samo osnovna orodja, v nadaljevanju pa moramo dodajati komponente, ki jih potrebujemo pri razvoju

kompleksnejših aplikacij. Orodja SDK dodajamo preko managerja SDK, ki ga vključuje program Android Studio. Android SDK vsebuje tudi starejše verzije platforme Android, tako da lahko razvijalci razvijajo aplikacije na starejših platformah.[12]

5.3 Android API

API je ogrodje, ki ima unikatno oznako za določeno verzijo platforme Android. Operacijski sistem razvijalci ves čas razvijajo in posodabljajo, prav tako pa tudi ogrodje API. Vsak novejši API je združljiv s predhodnim, vendar le do določene verzije. Razvijalci morajo pri razvoju določiti najnižjo in ciljno številko (slika 7) različice API.[12]

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.0%
4.2.x		17	3.0%
4.3		18	0.9%
4.4	KitKat	19	13.4%
5.0	Lollipop	21	6.1%
5.1		22	20.2%
6.0	Marshmallow	23	29.7%
7.0	Nougat	24	19.3%
7.1		25	4.0%
8.0	Oreo	26	0.5%

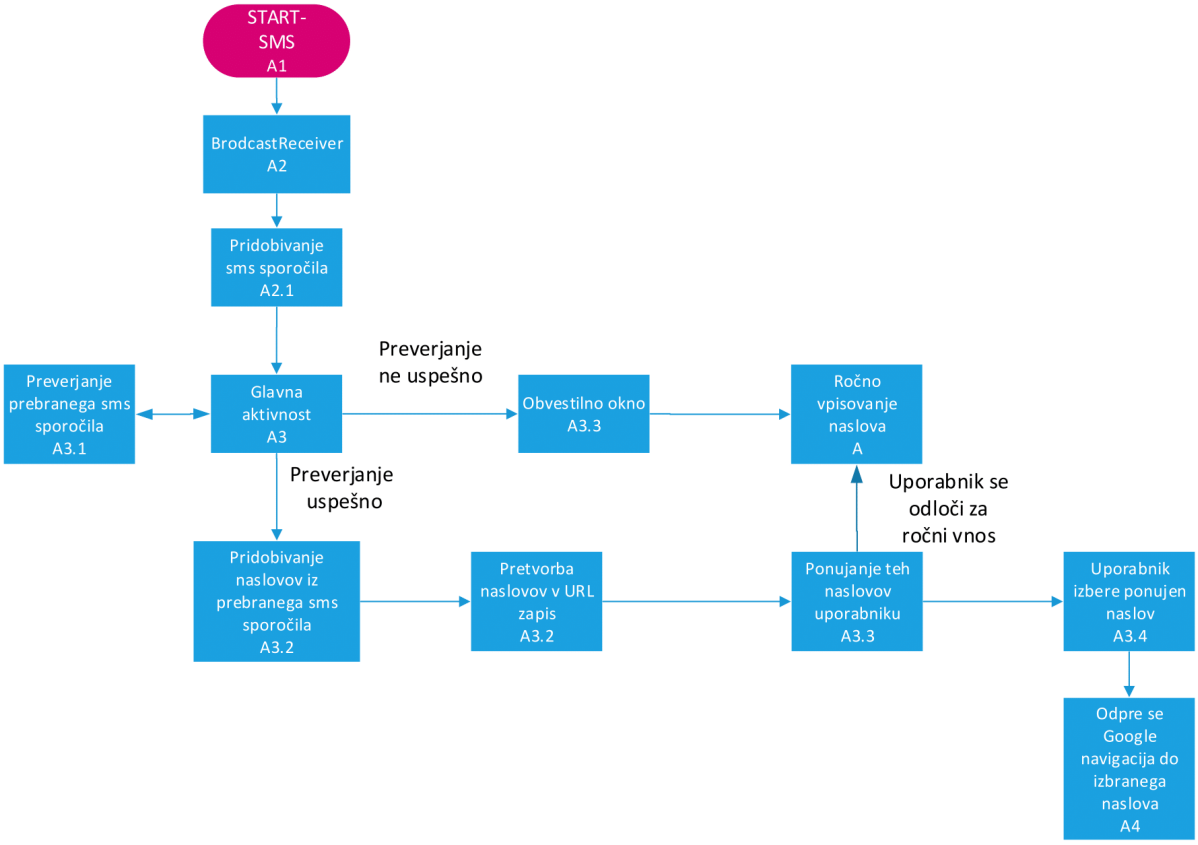
Slika 7: Tabela uporabljenih API različic med uporabniki [13].

5.4 Navidezna mobilna naprava (Emulator)

V paket Android SDK je vključena navidezna mobilna naprava, ki omogoča sprotno testiranje aplikacije in iskanje napak z razhroščevalnikom. Priporočljivo je, da razvijano aplikacijo testiramo na čim več različnih mobilnih napravah, saj je na ta način mogoče, da najdemo napako, ki se morda pojavi smo na nekaterih napravah. Navidezna naprava ima možnost predstavljanja različnih naprav, saj ji lahko določimo velikost in ločljivost zaslona, različico ogrodja API, procesorjev in nastavljanje velikosti delovnega in notranjega pomnilnika. Lahko pa uporabimo tudi dejanske naprave, ki morajo imeti operacijski sistem Android in ki jih priklopimo na računalnik. Izkazalo se je, da je testiranje na dejanskih napravah boljše ter hitrejše, saj navidezne naprave potrebujejo več časa za zagon [12].

6 Razvoj aplikacije

Aplikacijo smo razvijali v razvojnem okolju Android Studio, ki je opisan v poglavju 5. Aplikacija nima še nobenih nadgradenj in je trenutno še prototip. Pri razvoju aplikacije smo uporabili dva programska jezika, to sta Java, za funkcionalnost aplikacije, in XML, ki zajema grafični vmesnik aplikacije. Pri grafičnem vmesniku smo se odločili za preprosto izvedbo, da bo aplikacija omogočala pregled nad dogajanjem in uporabnikom ne bo povzročala težav med uporabo.



Slika 8: Diagram poteka aplikacije FireNav

6.1 Zagon aplikacije

Zagon aplikacije se izvede, ko naprava prejme SMS-sporočilo od specifične številke. Naprava je v vozilu vedno priklopljena na napajanje. S pomočjo možnosti operacijskega sistema Android, ki ima v nastavitvah pod kategorijo »za razvijalce« možnost, da se aplikacija nikoli ne ugasne oz. zaklene, ko je priklopljena na napajanje, omogočimo 24 urno pripravljenost naprave in je tako izključena potreba, da bi morali napravo pred uporabo odkleniti.

Zagon preko SMS-sporočila smo izvedli s pomočjo Java objekta **Brodcast receiverja**, ki dostopa do shranjenjih SMS-sporočil in vedno »posluša«, kaj se dogaja s SMS-sporočili v sistemu. Brodcast receiver deluje tako, da ob prejemu SMS-sporočila preveri, ali je številka pravilna. Če le-ta ustreza, zažene aplikacijo, v kateri je implementiran predlagan sistem.

```java
public class SmsBrodcastReciver extends BroadcastReceiver {

    public static final String SMS_BUNDLE = "pdus";
    public static final String Spec
    public String address;

    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle intentExtras = intent.getExtras();

        if (intentExtras != null) {
            Object[] sms = (Object[]) intentExtras.get(SMS_BUNDLE);
            String smsMessageStr = "";
            for (int i = 0; i < sms.length; ++i) {
                String format = intentExtras.getString( key: "format");
                SmsMessage smsMessage = SmsMessage.createFromPdu((byte[]) sms[i], format);

                address = smsMessage.getOriginatingAddress();
                String smsBody = smsMessage.getMessageBody().toString();
                smsMessageStr += "SMS From: " + address + "\n";
                smsMessageStr += smsBody + "\n";
            }
            Toast.makeText(context, text: "Message Received!", Toast.LENGTH_SHORT).show();
            if (MainActivity.active) {
                if(address.equals(SpecificNum)){
                    MainActivity inst = MainActivity.instance();
                    inst.refreshSmsInbox();
                }
            } else if(address.equals(SpecificNum)) {
                Intent i = new Intent(context, MainActivity.class);
                i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(i);
            }
        }
    }
}
```

Typo: In word 'pdus' more... (Ctrl+F1)

Slika 9: Funkcija onCreate, ki pridobi podatke.

6.2 Pridobivanje SMS-sporočila

Gasilci so še pred kratkim uporabljali pozivnike, na katere so prejemali obvestila o nesrečah, nato pa so namesto pozivnikov pričeli uporabljati svoje pametne telefone s pomočjo storitve Intervencije.net. Ta storitev pošilja SMS-sporočila na mobilne telefone. Sporočila, poslana na mobilne telefone so enaka, kot sporočila, poslana na pozivnike.

Pri izvedbi modula pridobivanja SMS-sporočila, uporabimo Java objekt **BrodcastReceiver**, ki sprejema vsa možna sistemska sporočila. V tem primeru uporabljen objekt **BrodcastReceiver** posluša, kdaj sistem pridobi SMS-sporočilo. Za določanje, kateri **BrodcastReceiver** naj uporabimo, moramo nastaviti Filter, ki sistemu pove, na kaj naj bo pozoren.

```xml
<receiver android:name=".SmsReceiver"  android:exported="true">
    <intent-filter android:priority="999">
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
```

Slika 10: Intent Filter, ki bo poslušal prejeta SMS-sporočila.

Nato moramo izvesti za ta sprejemnik Java razred (**class)**, ki ga poimenujemo SmsReceiver. Vsak BrodReceiver namreč potrebuje svoj **razred**, v katerem določimo, kaj se naj zgodi s pridobljenimi podatki.

SmsReceiver najprej preveri, ali je sistem že aktiven ali ne, nato pa na podlagi tega izvede pridobivanje SMS-sporočila. Če sistem še ni bil aktiven, bo **SmsReceiver** pričel s postopkom izločitve teksta iz SMS-sporočila, nato pa z uporabo objekta **Intent** (Java objekt, ki omogoča zagon drugih dejavnosti) zagnal glavno dejavnost, zraven pa prenesel še podatek, ki vsebuje tekst SMS-sporočila.

Če pa ugotovi, da je sistem že aktiven in deluje, pa bo samo poklical funkcijo **GetSms**, ki se nahaja v glavnem razredu oz. aktivnosti. Zakaj je potreben dvojen način delovanja? Prednost je v hitrosti delovanja, kajti če sistem že deluje, se samo pokliče funkcija GetSma, ki že sama prebere in osveži pridobljene podatke, tako odpade en postopek in tako pridobimo na hitrosti delovanja.

```
public void getSms(){
    String num = "070365764";
    Uri uri = Uri.parse("content://sms/inbox");
    Cursor c = getContentResolver().query(uri, projection: null, selection: null, selectionArgs: null, sortOrder: null);

    String body = null;
    String address = null;

    if(c.moveToFirst()){
        address = c.getString(c.getColumnIndexOrThrow( s: "address")).toString();
        if(address.equals(num)){
            body = c.getString(c.getColumnIndexOrThrow( s: "body")).toString();
        }
    }
    c.close();
    onUpdate(body);
}
```

Slika 11: Funkcija getSms

6.3 Pridobivanje naslova iz SMS-sporočila

Ko preberemo SMS-sporočilo v tekstovni obliki, je potrebno iz njega izločiti naslov izrednega dogodka, pri čemer pa se stvari zapletejo, saj SMS-sporočilo, ki ga sistem sprejme ni zmeraj enako zapisano – natančen enotni format zapisa sporočila namreč ni definiran. Tako ni mogoče preprosto določiti, od kod do kod je potrebno prebrati sporočilo, saj je naslov lahko zapisan na začetku ali pa na koncu sporočila. Vendar pa je SMS-sporočilo zapisano tako, da po podanih podatkih (kot so kraj, vzrok, osebe itd.) vključuje zmeraj tudi ključno besedo: **Posilja:ReCo.** Pri tem je zelo pomemben tudi znak »:«**,** saj algoritem, ki izloča naslov, SMS-sporočilo zmeraj najprej razcepi pri znaku »:«**,** nato pa uporabi prvo polovico SMS-sporočila, ki vsebuje podatke o nesreči.

Algoritem, ki iz SMS-sporočila vrne naslov, uporablja dva indeksa: **X** in **Y**. Najprej pregleda celotno SMS-sporočilo, vsak znak posebej, dokler ne najde zadnjega mesta, kjer se nahaja številka (hišna). Indeks te pozicije se dodeli spremenljivki **X**, nato pa funkcija še enkrat pregleda sporočilo, vendar ne od začetka, temveč od pozicije vrednosti **X**, dokler ne pride do presledka. Takšen postopek je potreben zato, ker imajo nekateri naslovi za številko še črko,

npr. 23a, 12b, … Ko definiramo vrednost spremenljivke **Y**, uporabimo metoda **Substring (**začetek, konec**)**, ki sprejme dva parametra. Prvi parameter določa, od kod se bo določen niz začel, drugi parameter pa določa, kje se bo končal. Za vrednost prvega parametra uporabimo vrednost **0**, drugi parameter pa je spremenljivka **Y.** Tako smo s tem postopkom pridobili niz, ki vsebuje morebiten naslov.

```
for(int i=0; i<sporocilo.length(); i++){
    if(Character.isDigit(sporocilo.charAt(i))) {
        x = i;
    }
}
if(x != 0){
    for(int j=x; j<sporocilo.length(); j++){
        if(sporocilo.charAt(j) == ' '){
            y=j;
            break;
        }
    }
```

Slika 12: Algoritem za obdelavo sporočila

Če upoštevamo, da mora biti aplikacija kar se da zanesljiva, je najbolj pomemben del aplikacije izbira pravilnega naslova kraja dogodka. Če bi naredili sistem, ki popolnoma avtomatsko izbere naslov kraja dogodka, bi se lahko zgodilo, da ta iz različnih razlogov zmeraj ne bi bil pravilno določen in bi bile lahko posledice tega katastrofalne. Zato smo se pri izbiri naslova odločili, da bo končno odločitev izbral uporabnik. Zato z izvedenim algoritmom izberemo več možnosti, ki jih sistem ponudi uporabniku.

6.4 Ponujanje naslova uporabniku

V prejšnjem poglavju smo opisali, kako smo izločili niz znakov, ki predstavljajo morebitni naslov. V naslednjem koraku moramo izločene podatke v ustrezni obliki ponuditi uporabniku. To smo izvedli na naslednji način.

Najprej s pomočjo metode **split(znak)** (ki kot parameter sprejme izraz). Metoda split vrne tabelo nizov, izračunanih z razdelitvijo niza okoli zadetkov danega izraza. V našem primeru hočemo iz enega niza pridobiti tabelo vseh besed niza, torej vsako posamezno besedo. To storimo tako, da v metodo split vstavimo znak presledek. Tako napolnimo tabelo, ki vsebuje posamezne besede iz niza z morebitnim naslovom.

V naslednjem koraku moramo iz posameznih besed sestaviti ustrezne morebitne naslove. To izvedemo na naslednji način.

Ustvarimo novo tabelo **ArrayNaslovi** tipa niz in spremenljivko **dela** tipa niz. Najprej tabelo pregledamo od zadaj naprej, besedo po besedo (zadnja beseda je številka hiše in pravilen naslov ima vedno številko hiše na zadnjem mestu) in vsako posamezno prebrano besedo shranimo v spremenljivko **dela**, vendar se pri dodajanju posamezne besede nova prebrana beseda doda vedno pred prejšnjo. Istočasno pa vsebino spremenljivke **dela** dodamo v

končno tabelo **ArrayNaslovi**. Na koncu tabela **ArrayNaslovi** vsebuje vse izločene podatke iz SMS-sporočila. Tako imamo npr. v tabeli podatke: "požar", "rdeči", "breg", "23". Ti podatki so shranjeni v naslednjem vrstnem redu:

- 23
- breg 23
- rdeči breg 23
- požar rdeči breg 23

Iz zgornjega zapisa je razvidno, da je Rdeči breg 23 pravilen naslov.

```
final String arrNas[] = sporocilo.split( regex: " ");
String dela = "";

for(int i=arrNas.length-1; i>0; i--) {
    dela = " " + arrNas[i] + dela;
    arrayNaslovi.add(dela);
}
```

Slika 13: Dodeljevanje besed v spremenljivko dela in arrayNaslovi

6.5 Pretvorba niza v obliko URL, primerno za navigacijo

Za pretvarjanje izločenega znakovnega niza znakov naslova v ustrezno obliko URL uporabimo funkcijo OpenMaps(niz). V tej funkciji ustvarimo spremenljivko gmmIntentUri tipa URI. V tej spremenljivki s pomočjo funkcije createUrl(niz) ustvarimo niz oz. URL naslov, ki je primeren kot vhodni podatek navigacijskega sistema. Funkcija createUrl v nizu, ki ga vstavimo kot parameter funkcije, zamenja presledke z znakom **+** in pred tem doda niz, ki je potreben pri ustvarjanju navigacije in sicer v obliki:
"https://www.google.com/maps/dir/?api=1&destination=" + izločen znakovni niz + "&travelmode=driving".

```
public String createUrl(String naslov) {
    String UrlNaslov = naslov.replace( oldChar: ' ', newChar: '+');
    System.out.println(UrlNaslov);
    String delaj = "https://www.google.com/maps/dir/?api=1&destination=Lovrenc+na+Pohorju+," + UrlNaslov + "&travelmode=driving";
    return delaj;
}
```

Slika 14: Funkcija createUrl

Tako definiran znakovni niz predstavlja vhodni podatek navigacijskega sistema, ki na osnovi tega določi navigacijsko pot do želenega naslova.

```
public void openMaps(String naslov){
    Uri gmmIntentUri = Uri.parse(createUrl(naslov));
    System.out.println(gmmIntentUri);
    Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
    mapIntent.setPackage("com.google.android.apps.maps");
    startActivity(mapIntent);
}
```

Slika 15: Funkcija openMaps

Definiran sistem bo izveden za uporabo na mobilnih napravah v obliki mobilne aplikacije na sistemu Android. Podrobnosti razvoja aplikacije so podane v poglavju 6.

6.6 Uporabniški vmesnik

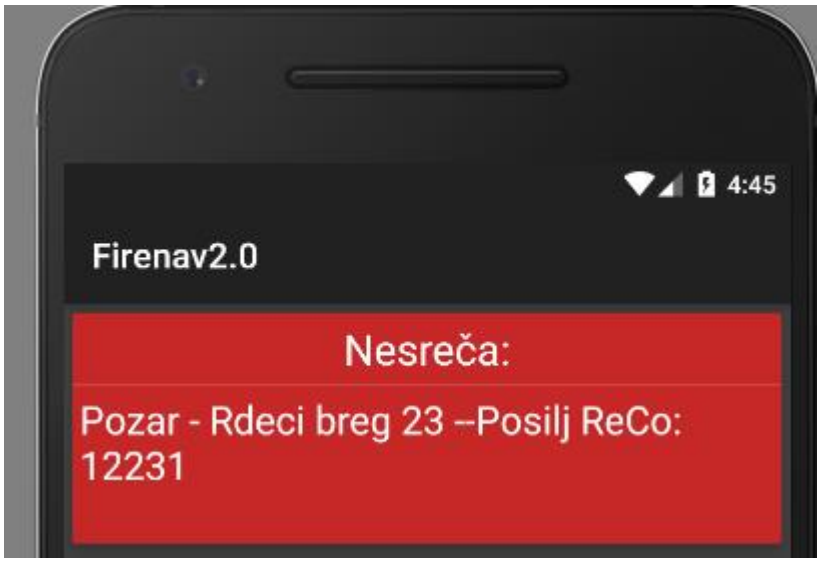
V tem poglavju bomo predstavili izgled in funkcionalnosti uporabniškega vmesnika razvite aplikacije.

Ko se aplikacija odpre, se pokaže slika aplikacije. Uporabniški vmesnik je sestavljen iz ene slike in nima nobenih prehodov. Sestavljen je iz treh elementov UI uporabniškega vmesnika (»CardView«). CardView je element UI, ki predstavlja oz. ima izgled kartice, ki odstopa od ozadja.

Prva kartica je sestavljena iz dveh besedil. Prvo besedilo oz. beseda nakazuje, za kakšen izredni dogodek gre, pod njo pa je vsebina celotnega SMS. Tako lahko uporabniki vidijo, za kakšne vrste izrednega dogodka gre.

```
<android.support.v7.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/cardViewSms"
    android:layout_gravity="center"
    app:cardCornerRadius="2dp"
    app:cardElevation="8dp"
    android:layout_margin="5dp"
    app:cardBackgroundColor="@color/cardBG">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="24sp"
            android:layout_gravity="center"
            android:text="Nesreča:"
            android:layout_margin="4dp"
            android:textColor="@color/whiteText"/>
        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="@color/divider"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="22sp"
            android:layout_margin="4dp"
            android:id="@+id/smsSporocilo"
            android:layout_gravity="center"
            android:textColor="@color/whiteText"/>
    </LinearLayout>
</android.support.v7.widget.CardView>
```

Slika 16: XML datoteka, ki vsebuje prvi CradView

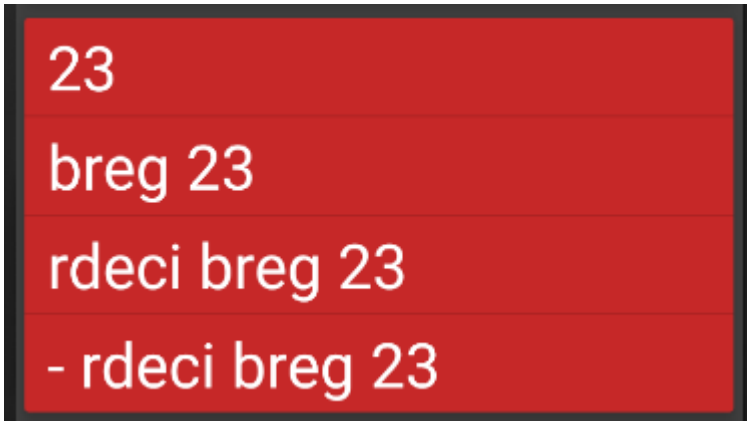


Slika 17: Izgled prvega CardViewa

Druga kartica »ListView« prikazuje vsebino seznama, ki vsebuje vse možne naslove, ki so bili izbrani iz SMS-sporočila. Ko uporabnik izbere na želen naslov, se zažene in odpre aplikacija Google Maps in ponudi navigacijo do izbranega naslova. Dizajn posamezne vrstice v seznamu določimo preko druge datoteke xml.

```
<android.support.v7.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/cardViewList"
    android:layout_gravity="center"
    app:cardCornerRadius="2dp"
    app:cardElevation="8dp"
    android:layout_margin="5dp"
    app:cardBackgroundColor="@color/cardBG">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <ListView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/listView"/>

    </LinearLayout>
</android.support.v7.widget.CardView>
```

Slika 18: Datoteka XML, ki vsebuje drugi CardView



Slika 19: Izgled drugega CardViewa

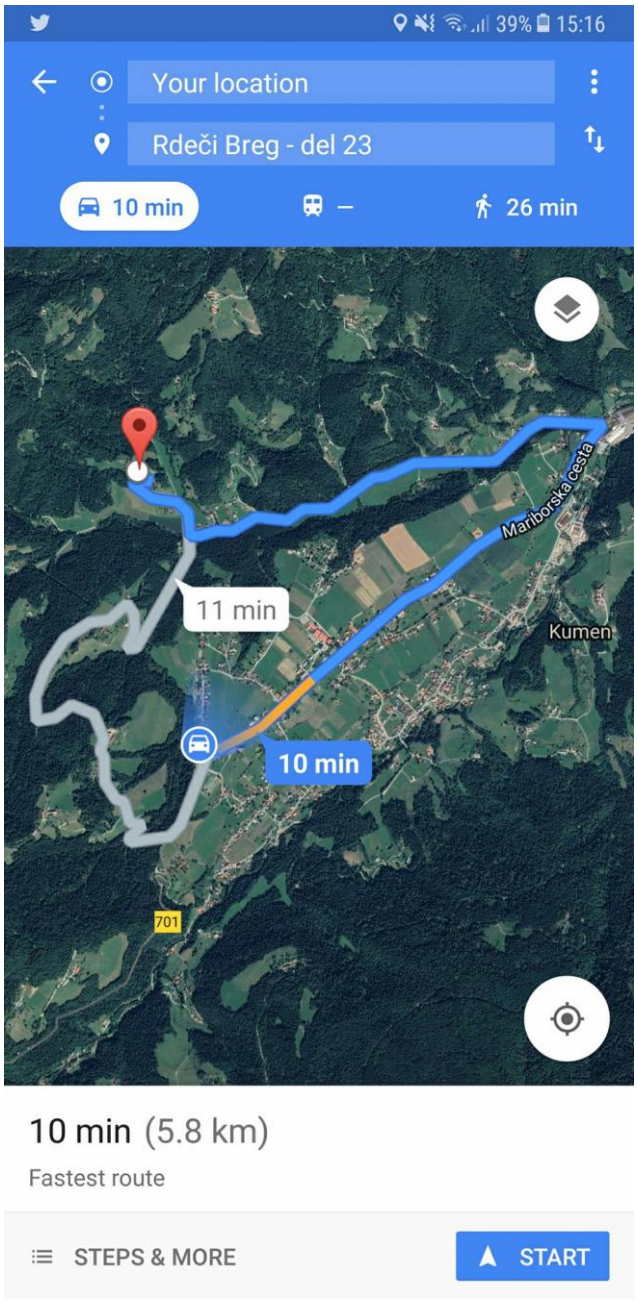
Tretja kartica pa omogoča fizični vnos naslova, če iz seznama ni bilo mogoče izbrati pravega. Pod vnosnim poljem se nahaja gumb, ki ima enako funkcionalnost, kot je izvedena za seznam, ko pritisnemo na izbran vnos v seznamu.

```
<android.support.v7.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/cardViewIsci"
    android:layout_gravity="center"
    app:cardCornerRadius="2dp"
    app:cardElevation="8dp"
    android:layout_margin="5dp"
    app:cardBackgroundColor="@color/cardBG">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Išči naslov"
            android:id="@+id/editText"
            android:textColorHint="@color/whiteText"
            android:textColor="@color/whiteText"
            android:layout_gravity="center"/>

        <Button
            android:id="@+id/btnIsci"
            android:layout_width="wrap_content"
            android:layout_height="35dp"
            android:layout_margin="4dp"
            android:layout_gravity="center"
            android:background="@color/appBG"
            android:textColor="@color/whiteText"
            android:text="IŠČI"/>

    </LinearLayout>
</android.support.v7.widget.CardView>
```

Slika 20: Datoteka XML, ki vsebuje tretji CardView

Slika 21: Izgled tretjega CardViewa

Ko uporabnik iz seznama ponujenih naslovov izbere pravilen naslov, ali pa ročno vnese naslov, se odpre aplikacija Google Maps, ki že vsebuje navigacijo do izbranega oz. vnesenega naslova.



Slika 22: Prikaz navigacije do željenega naslova

Uporabnik mora nato samo še preveriti ali sta naslov in izbrana pot pravilna in pritisniti na START.

7 Sklep

V inovacijskem predlogu smo najprej predstavili funkcionalnost sistema in njegovo delovanje od prejetega SMS-sporočila, do končne navigacije. Predstavili smo tudi tehnologije, potrebne za razvoj zastavljene aplikacije. V nadaljevanju smo podrobno predstavili mobilno platformo Android ter prikazali trenutno najbolj uporabljene različice operacijskega sistema Android.

Ugotovili smo, da je najbolj uporabljana različica 4.4 KitKat. Nato smo razčlenili sloje arhitekture operacijskega sistema in prikazali stanja življenjskega cikla aktivnosti aplikacije Android.

V nadaljevanju smo predstavili uporabljeno prosto dostopno razvojno okolje za razvijanje aplikacij na mobilni platformi Android. To okolje je Android Studio. Poleg tega smo predstavili še dodatke, brez katerih v programskem jeziku Java ne moremo programirati Android aplikacij.

V praktičnemu delu Inovacijskega predloga naloge smo se osredotočil na razvoj mobilne aplikacije FireNav, ki smo jo razvili za mobilne naprave z nameščenim operacijskim sistemom Android. S to aplikacijo želimo izboljšati učinkovitost reševanja in olajšati delo gasilcem. Na koncu smo izvedli preizkušanje aplikacije na fizičnih napravah, kar je pripomoglo k uspešnejšemu ugotavljanju napak v sistemu. Skozi celoten proces razvoja so bili v pomoč prosto dostopni primeri na spletu.

LITERATURA

[1] Slika. Posredovanje sporočil na pozivnik in mobilni telefon[online]. Dostopno na: http://intervencije.net/Images/News/Posredovanje_400_300px.jpg

[2] The Platform War Is Over and Android Won[online]. STATISTA. 2016. Dostopno na: https://www.statista.com/chart/4112/smartphone-platform-market-share/

[3] Platforma [online]. FRI - Android Wiki. 2012. [11.1.2016]. Dostopno na: http://android.fri.uni-lj.si/index.php/Platforma

[4] Android (operating system) [online]. Wikipedija. 2016. [11.1.2016]. Dostopno na: https://en.wikipedia.org/wiki/Android_(operating_system)

[5] Open Handset Alliance [online]. Wikipedija. 2016. [11.1.2016]. Dostopno na: https://en.wikipedia.org/wiki/Open_Handset_Alliance

[6] Android architecture [online]. Tuturials point. 2016. [14.1.2016]. Dostopno na: http://www.tutorialspoint.com/android/android_architecture.htm

[7] Slika. arhitektura operacijskega sistema Android[online]. Dostopna na: https://www.google.si/search?q=Android+-+Architecture&espv=2&biw=1745&bih=859&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiY7761k-7KAhVrIpoKHcYkB3MQ_AUIBigB#imgrc=mCDn3xy3vm3cJM%3A

[8] Slika. življenjski cikel aktivnosti Android aplikacije[online]. Dostopno na: http://developer.android.com/guide/components/activities.html

[9] Kaj je Android? [online]. Slo – Android.si. 2011. [15.1.2016]. Dostopno na: http://slo-android.si/prispevki/kaj-je-android.html

[10] Android Studio [online]. Wikipedija. 2016. [15.1.2016]. Dostopno na: https://en.wikipedia.org/wiki/Android_Studio

[11] Slika. logotip Android Studio[online]. Dostopno na: https://www.google.si/search?q=android+studio+logo&espv=2&biw=1745&bih=859&tbm=isch&imgil=XziqvrMF6h-tnM%253A%253B0TA6ebUmqLf32M%253Bhttp%25253A%25252F%25252Fcodecanyon.net%25252Fitem%25252Fblappy-bird-source-code-android-studio-project%25252F11230906&source=iu&pf=m&fir=XziqvrMF6h-tnM%253A%252C0TA6ebUmqLf32M%252C_&usg=__JNl8DAP6HJa3VDGUL5UGBPNQr0Y%3D&ved=0ahUKEwiNmOv6lO7KAhXBApoKHfS5Co4QyjcIMA&ei=wq67Vo2vDMGF6AT086rwCA#imgrc=XziqvrMF6h-tnM%3A

[12] Android Software development [online]. Wikipedija. 2016. [15.1.2016]. Dostopno na: https://en.wikipedia.org/wiki/Android_software_development

[13] Slika. API za različne verzije Androida[online]. Dostopno na: http://developer.android.com/about/dashboards/index.html