SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

ANDROID APLIKACIJA ZA PRONALAŽENJE PUSTOLOVNIH IZLETA

Ante Pupačić

Split, rujan 2020.

SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Preddiplomski sveučilišni studij: **Elektrotehnika i informacijska tehnologija**

Smjer/Usmjerenje: **Komunikacijska i informacijska tehnologija**

Oznaka programa: 114

Akademska godina: 2019./2020.

Ime i prezime: **Ante Pupačić**

Broj indeksa: 4-2017

ZADATAK ZAVRŠNOG RADA

Naslov: **ANDROID APLIKACIJA ZA PRONALAŽENJE PUSTOLOVNIH IZLETA**

Zadatak: Razviti aplikaciju za Android koja će korisnicima omogućiti pronalaženje pustolovnih izleta u srednjodalmatinskoj turističkoj regiji. Predstaviti Android operativni sustav, razvojni paket i Dalvik virtualni stroj. Za razvoj koristiti alat Android Studio te programski jezik Java. Za bazu podataka koristiti Google Firebase platformu.

Rad predan: 02.07.2020.

Datum obrane: 22.07.2020.

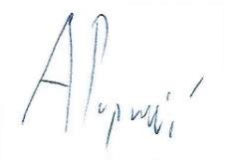
Mentor:

doc. dr. sc. Duje Čoko

IZJAVA

Ovom izjavom potvrđujem da sam završni rad s naslovom "Android aplikacija za pronalaženje pustolovnih izleta" pod mentorstvom doc. dr. sc. Duje Čoke pisao samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući naveo u završnom radu citirao sam i povezao s korištenim bibliografskim jedinicama.

Student



Ante Pupačić

SADRŽAJ

1. UVOD

Android je mobilni operativni sustav, koji je u početku razvio Android lnc, te je prodan Googleu 2005. godine. Temeljen je na modificiranoj Linux 2.6 kernel jezgri. Google, kao i drugi članovi Open Handset Alliance (OHA) surađivali su na dizajnu, razvoju i distribuciji Androida. Trenutno Android projekt otvorenog koda (engl. *Android Open Source Project,* skraćeno AOSP) upravlja ciklusom održavanja i razvoja Androida.

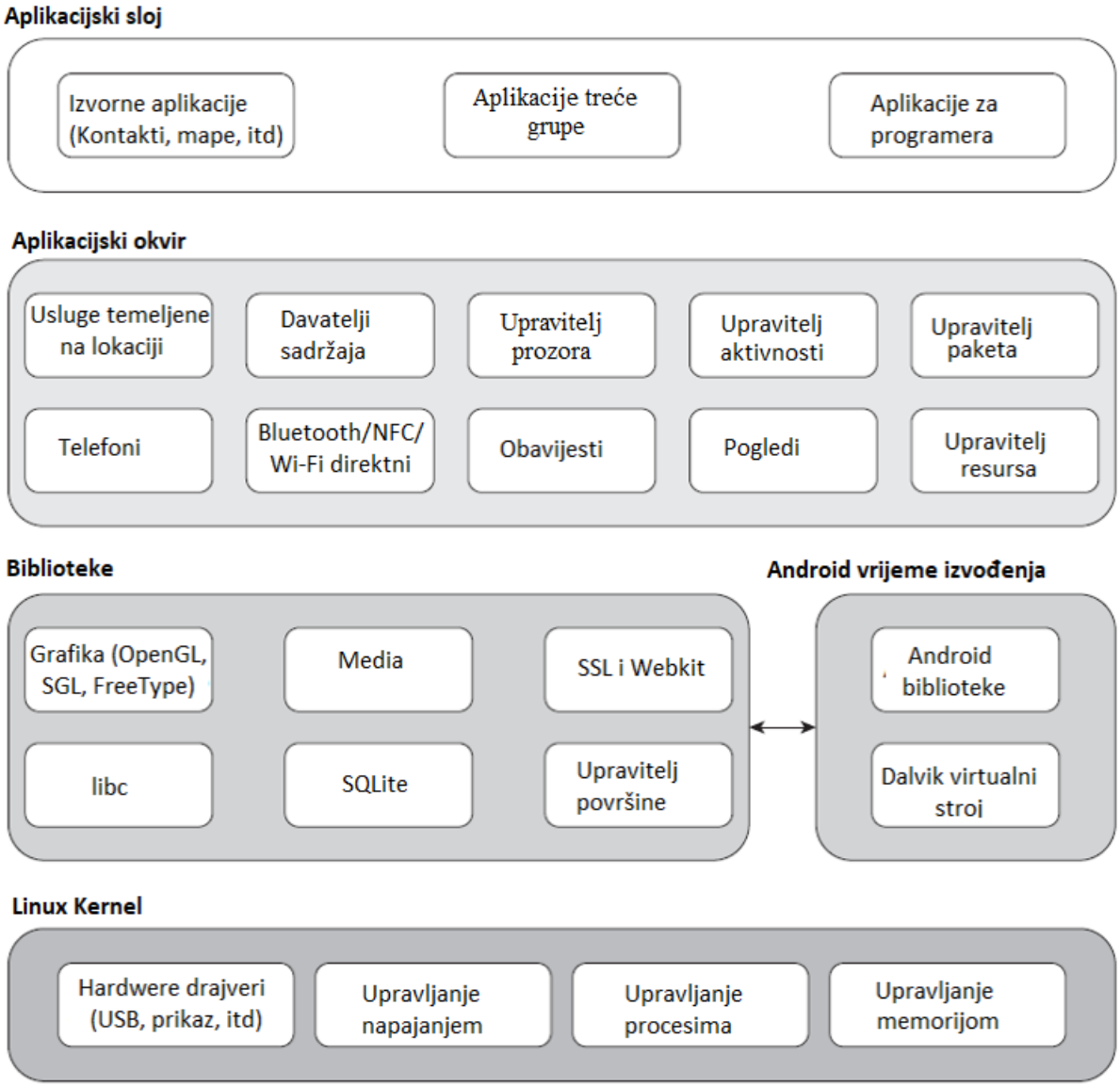
Pametni mobiteli i tableti postaju sve popularniji, operativni sustavi za te uređaje postaju važniji. Android operativni sustav je za uređaje koji se napajaju baterijom i sadrže mnogo hardvera poput prijamnika globalnog sustava pozicionirana (engl. *Global Positioning System,* skraćeno GPS), kamera, svjetlosnih i orijentacijskih senzora i zaslona osjetljivih na dodir. Kao i svi operativni sustavi, Android omogućuje aplikacijama da koriste hardverske značajke putem apstrakcije i pružaju definirano okruženje za aplikacije.

Za razliku od ostalih mobilnih operativnih sustava Appleovog iOS (engl. *iPhone operating system,* skraćeno iOS), Palmovog webOS ili Symbiana, Android aplikacije se pišu u Javi i Kotlinu, te se pokreću na virtualnom stroju. U tu svrhu Android sadrži Dalvik virtualni stroj koji izvršava vlastiti bajt kod. Dalvik je osnovna komponenta, jer izvršava sve Android aplikacije i aplikacijski okvir (engl. *application framework*). Android operativni sustav zasnovan je na modificiranoj Linux 2.6 kernel jezgri. U usporedbi s Linux okruženju, nekoliko je upravljačkih programa i biblioteka modificirano ili novorazvijeno kako bi Android mogao što učinkovitije funkcionirati na mobilnim uređajima i tabletima [1].

U praktičnom djelu je realizirana Android aplikacija za pronalaženje pustolovnih izleta u Srednjoj Dalmaciji, gdje korisnik u nekoliko klikova može pronaći sve potrebne informacije o pustolovnom izletu. Aplikacija je napisana Java programskim jezikom, te razvijena u Android studiju.

2. ANDROID SOFTVERSKI PAKET

Android softverski paket je Linux kernel i kolekcija C / C++ biblioteka izloženo kroz aplikacijski okvir koji pruža usluge, upravljanja vremenom izvršavanja i aplikacijom. Android softverski paket sastoji se od elemenata prikazanih na slici 2.1. [2].



Slika 2.1. Android softverski paket [2]

Linux kernel:

Linux 2.6 kernel nudi osnovne usluge uključujući upravljačke programe hardvera, upravljanje procesima i memorijom, sigurnost, mreža i upravljanje napajanjem. Kernel također pruža sloj apstrakcije između hardvera i ostatka paketa. Obrađuje sve radnje u kojima je Linux dobar, kao što su umrežavanje i niz upravljačkih programa [3].

Android biblioteke:

GNU (engl. *GNU's Not Unix*) biblioteke (engl. *glibs*) su prevelike i komplicirane za mobilne uređaje pa je Android napisao posebnu verziju biblioteka, Bionic biblioteke koje su manjih dimenzija 200K, uklonjene su neke komplicirane C++ značajke. Najznačajnije je to da nema C++ iznimaka (engl. *exceptions*). Implementirana je specijalna procesorska nit (engl. *thread*) koja je temeljena na kernel futexes. Bionic biblioteke ne podržavaju u potpunosti POSIX i nije kompatibilan s glibc-om. Sloj biblioteka uključuje skup C / C++ biblioteka koje koriste različite komponente Android sustava i pruža podršku aplikacijskom okviru. Android uključuje raznovrsne C / C++ biblioteke kao što su libc and SSL, kao i sljedeće [3]:

- Medijska biblioteka za reprodukciju audio i video medija.
- Upravitelj površine (engl. *Surface manager*) koji omogućava upravljanje zaslonom.
- Grafička biblioteka koja uključuje SGL i OpenGL za 2D i 3D grafiku.
- SQLite za podršku izvorne baze podataka.
- SSL i WebKit za integrirani web preglednik i internetsku sigurnost [2].

Android vrijeme izvođenja:

Vrijeme izvođenja je što Android telefon čini Android telefonom, a ne mobilnu implementaciju Linux-a. Uključuje osnovne biblioteke i Dalvik virtualni stroj (engl. *Dalvik Virtual Machine*, skraćeno DVM). Android vrijeme izvođenja (engl. *runtime*) je ″motor″ koji pokreće aplikacije i zajedno s bibliotekama tvori osnovni aplikacijski okvir.

- Osnovne biblioteke – Iako je većina Android aplikacija napisano koristeći Java jezik, Dalvik nije Java virtualni stroj. Osnovne Android biblioteke pružaju većinu

funkcionalnosti koje su dostupne u osnovnim Java bibliotekama, kao i Android specificirana (engl. *Android-specific*) C biblioteka.

- Dalvik VM – Dalvik je virtualni stroj temeljen na registru, optimiziran tako da uređaji mogu učinkovito pokrenuti više instanci. Oslanja se na Linux kernelu za rad s procesorskim nitima (engl. *threading*) i upravljanje memorijom na niskoj razini [2].

Aplikacijski okvir:

Aplikacijski okvir pruža klase koje se koriste za stvaranje Android aplikacije. Također pruža općenitu apstrakciju za pristup hardveru, upravlja korisničkim sučeljem i resursima aplikacije [2]. Aplikacijski okvir sadrži:

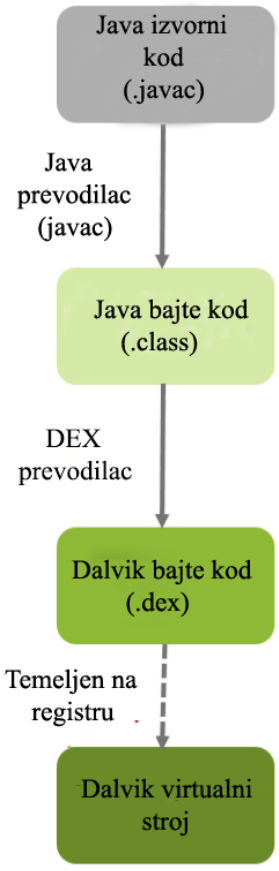
- Bogati i proširivi skup pogleda (engl. *views*) koji se mogu koristiti za izradu aplikacija koje uključuju korisničko sučelje. Pogledi (engl, View) se tvore kao liste, rešetke, tekstni okviri gumbi i web preglednici.
- Skup pružatelja sadržaja (engl. *Set of content providers*) koji imaju zadatak da omoguće aplikacijama pristup podacima druge aplikacije (poput kontakata) ili za dijeljenje vlastitih podataka.
- Upravitelj resursa (engl. *Resource manager*) je odgovoran za pristup ne temeljnim (engl. noncore) izvorima kao što su datoteka s lokaliziranim nizovima slova (engl. string), grafikom i izgledom.
- Upravitelj obavijestima (engl. *Notification manager*) je odgovaran da omogući prikaz svih upozorenja na statusnoj traci (engl. *status bar*).
- Upravitelj aktivnosti (engl. *Activity manager*) odgovoran je za upravljanje životnim ciklusom i pruža zajedničku navigaciju.
- Upravitelj lokacije (engl. *Location manager*) odgovoran je za aktiviranje upozorenja kada korisnik uđe ili napusti određeni geografski položaj.
- Upravitelj paketa (engl. *Package manager*) odgovoran je za dobivanje podataka o instaliranim paketima na uređaju.
- Upravitelj prozora (engl. *Window manager*) odgovoran je za stvaranje pogleda i izgleda.
- Upravitelj telefonije (engl. *Telephony manager*) odgovoran je za podešavanje mrežne veze i za sve informacije o uslugama na uređaju [3].

Aplikacijski sloj:

Sve aplikacije izvorne (engl. *native*) i treća grupa (engl. *third party*) su izgrađene na aplikacijskom sloju pomoću istih API (engl. *Application Programming Interface*, skraćeno API) biblioteka. Aplikacijski sloj pokreće se u Android vremenu izvođenja (engl, *Android runtime*) koristeći klase i usluge dostupne iz aplikacijskih okvira [2].

2.1. Dalvik virtualni stroj

Android koristi vlastiti virtualni stroj (engl. *Virtual Machine*, skraćeno VM) koji je poznat kao Dalvikov virtualni stroj (engl. *Dalvik Virtual Machine*, skraćeno DVM). DVM koristi posebni bajt kod, zato jer se izvorni Java bajt kod ne može izravno izvršiti na Android sustavima. Android zajednica pruža alat (dx) koji omogućava pretvaranje datoteku Java klase u izvršnu datoteku Dalvik (dex) (*Slika 2.1.1. Dalvik virtualni stroj*).



Slika 2.1.1. Dalvik virtualni stroj [4]

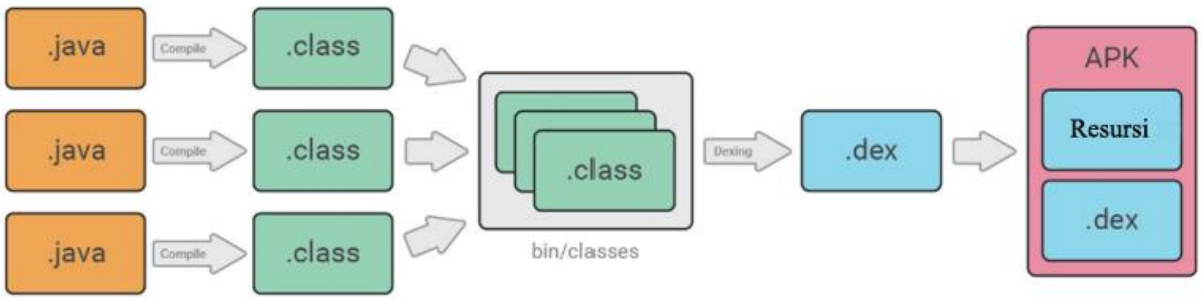
Implementacija DVM-a dobro je optimizirana, kako bi bila što učinkovitija na mobilnim uređajima koji su opremljeni s prilično sporim (jednim) CPU-om, ograničenim kapacitetom baterije. DVM je implementiran tako da omogućuje uređaju da izvršava više virtualnih strojeva na učinkovit način.

DVM se oslanja na modificiranoj Linux jezgri za sve potencijalne funkcionalnosti upravljanja procesnim nitima i memorijom niske razine. Uz Android 2.2, provedene su neke velike promjene u JVM infrastrukturi. Do verzije 2.2, Java virtualni stroj (engl. *Java Virtual Machine*, skraćeno JVM) je bio tumač (engl. *interpreter*), sličan originalnom JVM rješenju razvijenom s Java 1.0. Iako je Android rješenje uvijek imalo učinkoviti tumač, zato što je bio tumač nije moga generirati vlastiti kod. Izdanjem Androida 2.2, u snopu rješenja ugrađen je "na vrijeme" (engl. *Just-in-time*, skraćeno JIT) prevodilac, koji prevodi Dalvik-ov bajt kod u mnogo učinkovitiji strojni kod (slično kao kod C prevoditelja).

Android će implementirati dodatne značajke za JIT i "skupljač smeća" (engl. *Garbage Collection,* skraćeno GC), što će dodatno poboljšati performanse sustava.

2.1.1. DEX datoteka

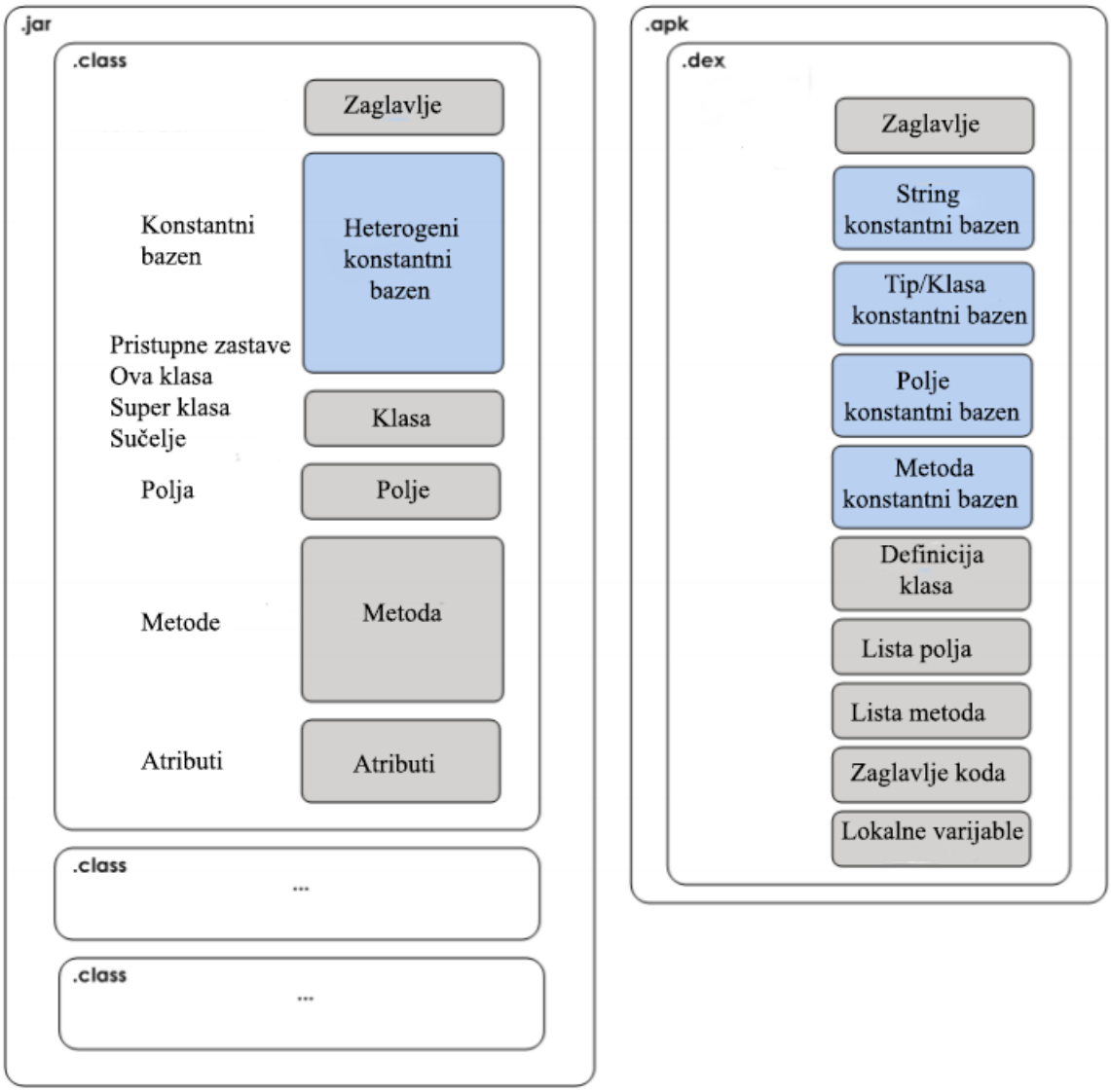
Dalvik izvršni (engl. *Dalvik Executable,* skraćeno .dex) je Dalvik bajt kod koji je preveden s Java bajt koda pomoću dx alata (zamijenjen alatom D8 od API-ja 28). Ovaj format dizajniran je za sustave koji imaju ograničenu brzinu procesora i memorije. Više klasa je uključeno u dex datoteku (*Slika 3.1.1. Postupak generiranja novog APK-a*) [4].



Slika 2.1.1.1. Postupak generiranja novog APK-a [4]

U standardnim Java okruženjima izvorni kod kompajlira se u bajt kod koji se pohranjuje unutar .class datoteka. U toku izvođenja JVM čita .class datoteke. Svaka klasa u Java kodu bit će rezultirana jednom .class datotekom.

Na Android platformi Java izvorni kod kompajliran je unutar .class datoteka. Nakon generiranja .class datoteka, "dx" alat koristi se za pretvorbu .class datoteka u .dex ili Dalvik izvršnu datoteku. .class datoteka sadrži samo jednu klasu, .dex datoteka sadrži više klasa koja se izvršava na Dalvik virtualnom stroju. .dex datoteka je optimizirana za efikasno korištenje memorije i dijeljenje podataka (*Slika 3.1.1 Usporedba .class format datoteke i .dex format datoteke*).



Slika 3.1.1 Usporedba .class format datoteke i .dex format datoteke [5]

.Dex format datoteke sadrži "zajedničke bazene" specifične za tip kao primarni mehanizam za očuvanje memorije. Konstantni bazen (engl. *Constant pool*) pohranjuje vrijednosti konstanti korištene unutar klase, što uključuje vrijednosti nizova slova (engl. *string*) koje se koriste unutar koda, kao i polje, varijable, klase, sučelje i imena metoda (engl. *methods*). Umjesto da se vrijednosti pohranjuju unutar klase, upućuju se na njihov indeks u konstantnom bazenu. U slučaju .class datoteka, svaka klasa ima svoju privatnu, heterogenu konstantu bazena. Sve vrste konstanti (polja, varijable, klase itd.) miješaju se zajedno. Usporedbom s .dex datotekom koja sadrži mnogo klasa, a te klase dijele iste vrste konstantnih bazena. Umnožavanje konstanti u .class datotekama eliminira se u .dex formatu.

Omogućujući klasama dijeljene konstanti bazena, ponavljanje konstantnih vrijednosti zadržava se na minimumu. Osnovni cilj korištenja zajedničkih konstantnih bazena je ušteda memorije.

Kada se analizira koliko svaki odjeljak .class datoteke zauzme: najveći dio zauzme konstantni dio [pool] (61% datoteke), a ne metode koji predstavljaju 33%. Ostali dijelovi .class datoteke dijele 5%. Tako je zaključeno da optimizacijom konstantnog bazena može rezultirati značajnom uštedom memorije (*Tablica 2.1.1.1. Ušteda memorije kod Jar datoteke i dex datoteke*).

Tablica 2.1.1.1. Ušteda memorije kod Jar datoteke i dex datoteke [5]

Kod	Nekomprimirane JAR datoteke (bajt)	Komprimirane JAR datoteke (bajt)	Nekomprimirane DEX datoteke (bajt)
Zajdeničke biblioteke sustava	21.445,320 (100%)	10,662,048 (50%)	10,311,972 (48%)
Aplikacija web preglednika	470,312 (100%)	232,065 (49%)	209,248 (44%)
Aplikacija budilice	119,200 (100%)	61,658 (52%)	53,020 (44%)

Optimizacija dijeljenja memorije ne dolazi besplatno. Strategija skupljanja smeća mora poštovati dijeljenje memorije. Skupljanje smeća je neovisno između aplikacija, iako postoji

dijeljenje neke memorije, jer je svaka aplikacija odvojeni postupak s odvojenim virtualnim strojem i odvojenom hrpom (engl. *heap*). Trenutna strategija u Dalvik skupljaču smeća treba zadržati markirane bitove ili bitove koji upućuju da je određeni objekt "dostupan" i zbog toga se ne smije odvoziti smeće, odvojeno od ostale memorije. Ako su markirani bitovi pohranjeni s objektima na hrpi, svi bi zajednički objekti odmah postali "ne zajednički" tijekom svakog ciklusa odvoza smeća kada sakupljač smeća prođe kroz hrpu i postavi markirane bitove [5].

2.1.2. Zygote

Svaka aplikacija pokreće se u vlastitoj instanci VM-a, instance VM-a moraju se pokrenuti brzo kada se pokrene nova aplikacija te otisak memorije VM mora biti minimalan. Android koristi koncept nazvan Zygote kako bi omogućio dijeljenje koda u VM instancama i brzo pokrenuo nove VM instance. Zygote dizajn pretpostavlja da postoji značajan broj osnovnih biblioteka klasa (engl. *core library classes*) i odgovarajućih struktura hrpa koje se koriste kod velikog broja aplikacija. Sve strukture hrpa su uglavnom za čitanje. To su podaci i klase koje aplikacija koristi, ali ih nikad ne mijenjaju. Te su karakteristike iskorištene za optimizaciju dijeljenja memorije u različitim procesima.

Zygote je VM proces koji počinje u vremenu pokretanja sustava. Kada započne proces Zygote inicijalizira Dalvik VM koji učitava i incijalizira temeljne biblioteke klase. Osnovne biblioteke klasa su samo za čitanje i stoga su dobar kandidat za učitavanje i dijeljenje preko procesa. Jednom kada se Zygote inicijalizira, sjedit će i čekati "socket" zahtjev (engl, *socket requests*) koji dolazi iz vremena izvođenja koje upućuje da treba razdvojiti nove VM instance temeljene na instanci Zygote VM. Virtualni strojevi s "hladnim" pokretanjem traju dugo i mogu biti smetnja izolaciji svake aplikacije u svom VM-u. Spajanjem novih VM procesa iz Zygote, vrijeme pokretanja se minimizira.

Osnovne biblioteke klasa (engl. *The core library classes*) koje se dijele na VM instance uglavnom se samo čitaju, ali napisane su putem aplikacija. Kad se klase upisuju u memoriju iz zajedničkog Zygote procesa koji je kopiran u dječji proces (engl. *forked child process*) aplikacije VM-a. Ponašanje "kopiraj-na-napiši (engl. *copy-on-write*)" omogućava maksimalno dijeljenje memorije dok zabranjuje aplikacijama međusobno "uplitanje" (engl. *interfering*) i pruža sigurnost kroz aplikaciju i proces granica (engl. *boundary*). U tradicionalnom Java VM dizajnu sve instance

VM-a imat će cijelu kopiju osnovne biblioteke klase datoteka i sve povezane hrpe objekata. Memorija se ne dijeli na više instance [5].

3. ANDROID SDK

Razvoj Androida započinje s Android SDK-om (engl. *Software Development Kit*, skraćeno SDK). Iako postoji puno različitih programskih jezika i mnoštvo IDE-ova (engl. Integrated Development Environments, skraćeno IDE) koje možete koristiti za izradu aplikacije, prednost SDK-a je što se ne mijenja.

SDK nudi izbor alata potrebnih za izgradnju Android aplikacija ili kako bi se osiguralo da se proces izvrši što je najlakše moguće. Bez obzira završi li se stvaranje aplikacije s Java, Kotlinom ili C #, potreban je SDK za pokretanje na Android uređaju i pristup jedinstvenim značajkama OS-a. SDK nudi mogućnost upotrebe emulatora za testiranje gotovih aplikacija, nadziranje uređaja i obavljanje niza drugih stvari. Android SDK također dolazi u kompletu s Android integriranim razvojnim okruženjem u kojem se izvode programi, te s mnogim alatima koji služe za pristup i upravlje aplikacijom.

Android SDK se sastoji od nekoliko komponenata:

- Platformni alati (engl. *Platform - tools*).
- Ugradbeni alati (engl. *Build – tools*).
- SDK – alati (engl. *SDK -tools*).
- Android program za uklanjanje pogrešaka (engl. *The Android Debug Bridge*, skraćeno ADB).
- Android emulator.

Alati pružaju svojevrsni most između Android Studio-a i emulatora kako bi aplikacija bila prikladno pakirana i testirana tijekom razvoja.

Najvažniji dijelovi SDK paketa su SDKtools. Ovi alati su potrebni bez obzira na verziju Androida. Svi ovi alati će stvoriti APK – koji prevara Java kod u Android-ovu aplikaciju koja se može pokrenuti na mobilnom uređaju. SDK uključuju alate za izradu, alate za uklanjanje pogrešaka i alate za slike (engl. *image tools*). Primjer jednog takvog alata je DDMS koji omogućava korištenje Android monitor uređaja (engl. *Android device monitor*) za provjeru statusa Android uređaja.

Ugradbeni i Platformni alati su razdvojeni tako da se mogu zasebno ažurirati. Ti alati su potrebni za izgradnju Androidovih aplikacija. Zipalign alat (engl. *Zipalign tool*) optimizira aplikaciju da koristi minimalnu memoriju prilikom pokretanja prije generiranja konačnog APK-a, i apksigner koji potpisuje APK za naknadnu provjeru [6].

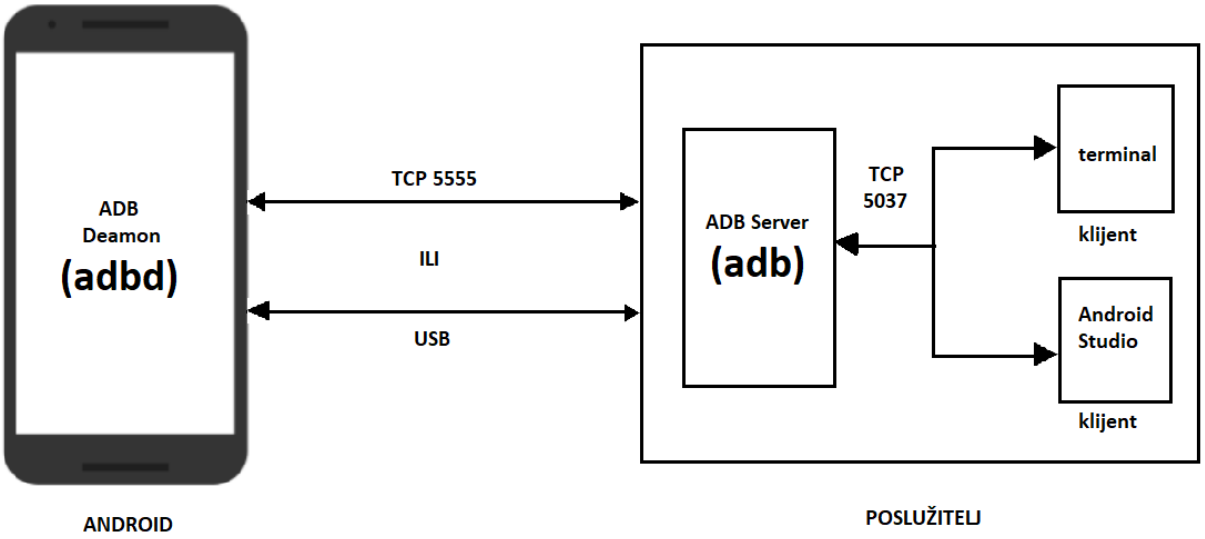
3.1. Android program za uklanjanje pogrešaka

Android program za uklanjanje pogrešaka je program koji omogućuje komunikaciju s bilo kojim Android uređajem. Oslanja se na Platform-alatu kako bi razumio verziju Androida koja se koristi na navedenom uređaju i stoga je uključen u paket Platform-alati.

ADB je prilagodljivi alat naredbenog retka (engl. *command-line*) koji omogućuje komunikaciju s uređajem. Naredba adb olakšava razne akcije uređaja, poput instaliranja i uklanjanje pogrešaka aplikacija, a omogućuje pristup Unix školjci (engl. *Unix shell*) koja se koristiti za pokretanje naredbi na uređaju. To je klijent-poslužiteljski program koji sadrži tri komponente:

- Klijent, koji šalje naredbe. Radi na razvojnom stroju. Klijent se poziva s terminala naredbenog retka izdavanjem adb naredbe.
- Daemon (adbd), koji izvršava naredbe na uređaju, pokreće se kao pozadinski proces na svakom uređaju.
- Poslužitelj, koji upravlja komunikacijom između klijenta i daemona, pokreće se kao pozadinski proces na razvojnom stroju.

Kada se pokrene ADB klijent, klijent prvo provjerava postoji li postupak ADB poslužitelja, ako nema pokreće se proces na poslužitelju. Kad se poslužitelj pokrene, on se veže za lokalni TCP (engl. *Transmission Control Protocol*, skraćeno TCP) port 5037 i sluša naredbe poslane od adb klijenata – svi adb klijenti koriste port 5037 za komunikaciju s adb poslužiteljem (*slika 3.1.1 ADB*).



Slika 3.1.1 ADB [7]

Poslužitelj zatim postavlja veze na sve pokrenute uređaje. Locira emulatore skeniranjem portova neparnih brojeva u rasponu 5555 do 5585, to je raspon koji koristi prvih 16 emulatora. Tamo gdje poslužitelj pronađe adb daemon (adbd), uspostavlja vezu s tim priključkom. Svaki emulator koristi par uzastopnih ulaza (numerirani priključak za veze na konzoli i neparni priključak za adb veze). Na primjer:

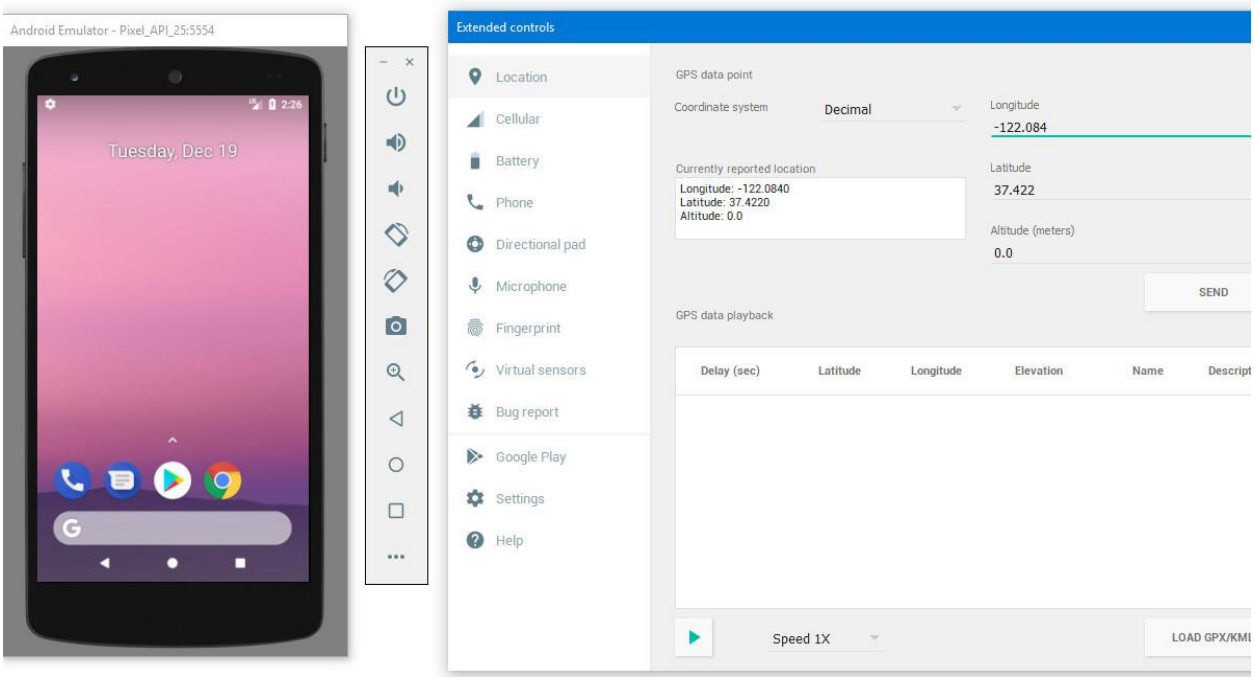
- Emulator 1, konzola: 5554
- Emulator 1, adb: 5555
- Emulator 2, konzola: 5556
- Emulator 2, adb: 5557
- Itd.

Emulator spojen na adb priključak 5555 jednak je emulatoru čija konzola sluša na priključku 5554.

Nakon što poslužitelj uspostavi veze na sve uređaje, koristi se adb naredba za pristup tim uređajima. Budući da poslužitelj upravlja vezama s uređajima i rukuje naredbama s više adb klijenata, može se upravljati s bilo kojeg uređaja s bilo kojeg klijenta (ili sa skripte) [8].

3.2. Android emulator

Android emulator simulira Android uređaje na računalu tako da se može testirat aplikacija na raznim uređajima i Android API razinama, a da nije potreban fizički uređaj (*Slika 3.2.1 Android Emulator – Pixel_API_25:5554*). Da bi to mogli koristiti potreban nam je android sustav slike (engl. *Android system image*) koji je dizajniran da se može izvoditi na hardveru računala.



Slika 3.2.1 Android Emulator – Pixel_API_25:5554 [9]

Emulator pruža gotovo sve mogućnosti pravog Android uređaja. Može simulirati dolazne telefonske pozive i tekstualne poruke, odrediti lokaciju uređaja, simulirati različite brzine mreže, simulirati rotaciju i druge hardverske senzore, pristupiti Google Play Store-u i još mnogo toga.

, Testiranje aplikacija na emulatoru je brže i jednostavnije nego na fizičkom uređaju. Podaci se brže prenose na emulator nego na uređaj spojen preko USB-a. Emulator dolazi s unaprijed definiranom konfiguracijom za razne Android telefone, tablete, Wear OS i Android TV uređaje.

Upravitelj Android virtualnog uređaja (engl. *Android Virtual Device Manager*) služi za odabir verzije Androida koja se želi oponašat, zajedno sa specifikacijama uređaja (veličina zaslona, performanse, …). Svaki Android virtualni uređaj (engl. *Android Virtual Device*, skraćeno AVD) funkcionira kao neovisni uređaj s vlastitom privatnom pohranom za korisničke podatke, SD karticom, itd. Emulator prema zadanim postavkama pohranjuje korisničke podatke, podatke SD

(engl. *Secure Digital*) i brza memorija (engl. *cache*) u direktorij specifičan za taj AVD. Kada se pokrene emulator, on učitava podatke o korisničkim podacima i SD kartici iz AVD direktorija.

Kad se koristi AVD s razinom API-ja 25 ili višim, emulator pruža simuliranu Wi-Fi pristupnu točku ("AndroidWifi"), te se automatski povezuje na nju.

Android emulator ne uključuje virtualni hardver za: Bluetooth, NFC, umetanje / izbacivanje SD kartice, slušalice priključene na uređaj, USB [9].

3.3. Životni ciklus aplikacije

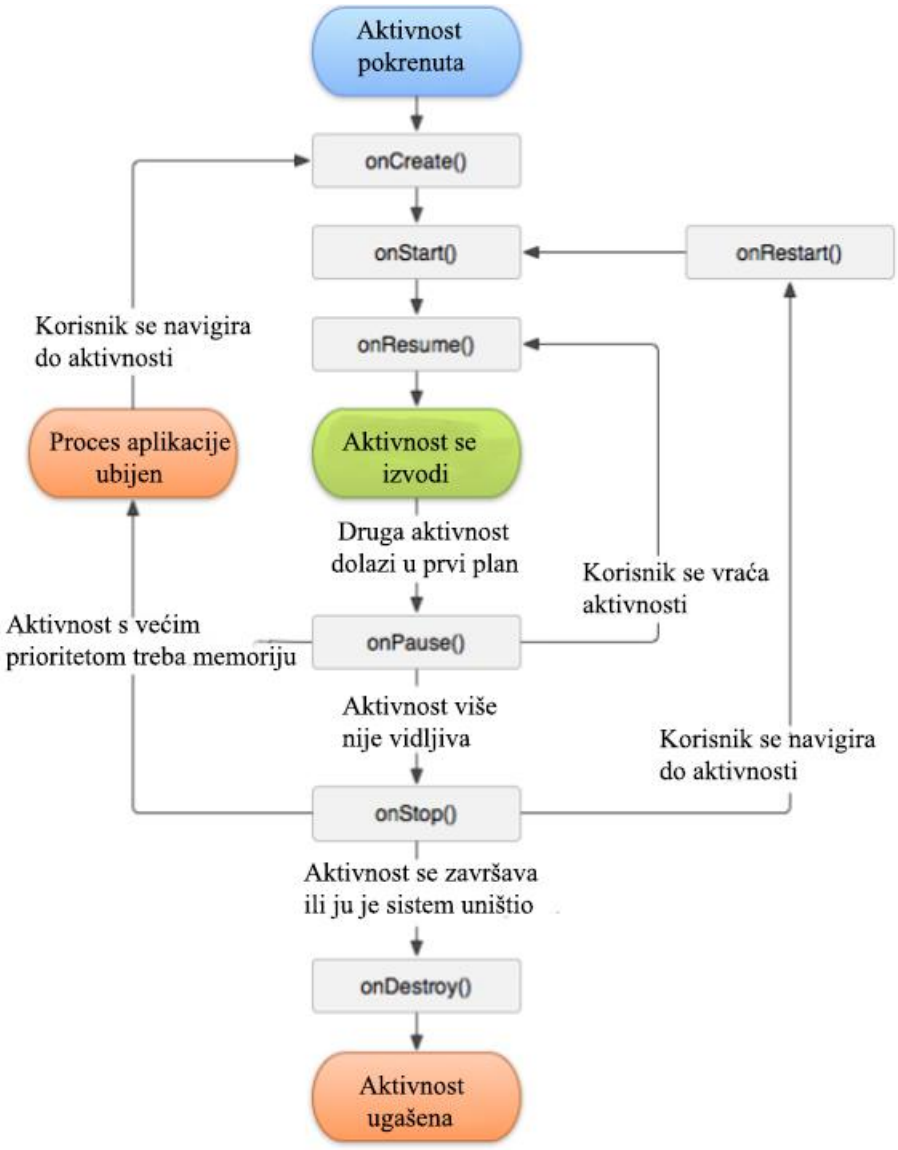
Android aktivnosti prolaze kroz nekoliko faza, od kada se prvo učitavaju do zatvaranja. Obrađuje se kao "događaj" (engl. *event*) unutar koda kada korisnik na neki način promijeni stanje aplikacije: pokretanjem, pauziranjem, zatvaranjem itd. Događaji se odvijaju unutar metoda (nizovi uputa unutar vitičastih zagrada), možemo koristiti metode i odlučiti što želimo da se dogodi unutar svake faze. Odatle potječe linija @Override: što znači da koristimo metodu koja nije potpuno nova, već ona koja se uvijek izvodi (obično naslijeđena od superklasa) čime developer preskače linije koda i dodaje promjene [10].

Za navigaciju prijelaza između faza životnog ciklusa aktivnosti, klasa aktivnosti (engl. activity class) pruža temeljni skup od 7 povratnih poziva:

- onCreate() - Poziva se kada je aktivnost prvi put kreirana. Ovdje se rade sve uobičajene statičke postavke: stvoriti prikaze, povezati podatke na liste itd. Ova metoda pruža skup koji sadrži prethodno zamrznuto stanje aktivnosti (engl. *activity's previously frozen state*), ako ga je bilo. Uvijek slijedi onStart().
- onRestart() – Poziva se nakon zaustavljanja aktivnosti prije nego što se ona ponovno pokrene. Uvijek slijedi onStart().
- onStart() – Poziva se kada aktivnost započne interakciju s korisnikom. U ovom trenutku aktivnost se nalazi na vrhu stoga s korisničkim unosom koje ide na njega. Uvijek slijedi onPause().

- onPause() – Poziva se kada aktivnost nije u prvom planu, više se ne može fokusirati procesne niti pri prijelazu u zaustavljeno / skriveno ili uništeno stanje. Aktivnost je i dalje vidljiva korisniku, pa se preporučuje da bude vizualno aktivna i nastavi ažuriranje korisničkog sučelja. Provedba ove metode mora biti vrlo brza jer sljedeća aktivnost neće biti nastavljena dok se ova metoda ne vrati. Nakon toga slijedi ili onResume() ako se aktivnost vrati u prvi plan, ili onStop() ako postane nevidljiva za korisnika.
- onStop() – Poziva se kada aktivnost više nije vidljiva korisniku. To se može dogoditi zato što se nova aktivnost pokreće na vrhu, postojeća se pokreće ispred ove ili se uništava. Obično se koristi za zaustavljanje animacija i osvježavanje korisničkog sučelja itd. Nakon toga slijedi ili onRestart() ako se aktivnost vraća u interakciju s korisnikom ili onDestroy() ako se aktivnost zaustavi.
- onDestroy() – Završni poziv koji primate prije nego što aktivnost bude uništena. To se može dogoditi ili zato što aktivnost završava (*Activity#finish*), ili zato što sustav privremeno uništava ovu instancu aktivnosti radi uštede prostora. Ova dva scenarija mogu se razlikovati metodom Activity#isFinishing [11].

Sustav poziva svaki od tih povratnih poziva dok aktivnost ulazi u novo stanje (*Slika 4.1.1. Pojednostavljena ilustracija životnog ciklusa aktivnosti*).



Slika 3.3.1. Pojednostavljena ilustracija životnog ciklusa aktivnosti [12]

Kad korisnik napušta aktivnost, sustav poziva metode da rastavi (engl. *dismantle*) aktivnosti. U nekim slučajevima rastavljanje je samo djelomična: aktivnost i dalje ostaje u memoriji (primjerice kada korisnik prebaci na drugu aplikaciju) i još se vijek može vratiti na početnu aktivnost. Ako je korisnik vrati na tu aktivnost, aktivnost će se nastaviti ondje gdje je korisnik stao. Uz nekoliko iznimaka, aplikacije su ograničene od započinjanja aktivnosti kada se pokreću u pozadini.

Vjerojatnost sustava da ubije (engl. *killing*) određeni proces, zajedno sa aktivnostima u njemu ovisi o stanju aktivnosti u tom trenutku. Stanje aktivnosti i izbacivanje iz memorije pružaju više informacija o odnosu između stanja i ranjivosti na izbacivanju.

Aktivnosti u sustavu upravljaju se kao stog aktivnosti. Kada se pokrene nova aktivnost, ona se obično postavlja na vrh trenutnog stoga i postaje aktivna. Prethodna aktivnost je ispod nje u stogu i neće više doći na početak stoga (u prvi plan) sve dok nova aktivnost ne prestane. Na zaslonu mogu biti vidljive jedna ili više aktivnosti.

Akitivnost ima četiri stanja:

- Ako je neka aktivnost u prvome planu zaslona (na najvišoj poziciji stoga), ona je aktivna ili pokrenuta. To je obična aktivnost s kojom korisnik trenutno komunicira.
- Ako je neka aktivnost izgubila fokus, ali je i dalje predstavljena korisniku, ona je vidljiva. Moguće je ako se nova aktivnost koja nije maksimalne veličine ili transparentno fokusira na vrh aktivnosti, druga aktivnost ima viši položaj u načinu rada s više prozora ili nije fokusirana u trenutnom načinu prikaza. Takva aktivnost je potpuno aktivna (održava sve podatke o stanjima i ostaje vezana za upravitelja prozora).
- Ako je neka aktivnost u potpunosti zamračena drugom aktivnošću, on se zaustavlja ili skriva. Još uvijek zadržava sve podatke o stanju i članovima, međutim više nije vidljiva korisniku i sustav će je ubiti kada je memorija potrebna negdje drugdje.
- Sustav može ispustiti aktivnost iz memorije ako zatraži da završi ili je jednostavno ubije, čime ju uništava. Kad se aktivnost prikaže korisniku, ponovno se pokreće i vraća u prethodno stanje [11].

Postoje tri ključne petlje koje su zanimljive u okviru aktivnosti:

- Cijeli životni vijek aktivnosti događa se između prvog poziva onCreate() do jednog konačnog poziva onDestroy(). Aktivnost će obaviti sva globalna stanja

u onCreate() i osloboditi sve preostale resurse u onDestroy(). Procesne nit koja se pokreće u pozadini za preuzimanje podataka s mreže. Procesna nit se stvara u onCreate(), a zatim se zaustavlja u onDestroy().

- Vidljivi vijek trajanja aktivnosti događa se između poziva onStart() do odgovarajućeg poziva onStop(). Za to vrijeme korisnik može vidjeti aktivnost na zaslonu, iako nije u prvom planu i interaktira s korisnikom. Između ove dvije metode zadržavaju se resursi koji su potrebni da bi se korisniku prikazale aktivnosti. Metode onStart() i onStop() mogu se pozvati više puta jer aktivnost postaje vidljiva i skrivena od korisnika.
- Vijek trajanja glavne aktivnosti događa se između poziva onResume() do odgovarajućeg poziva onPause(). Za to vrijeme aktivnost je vidljiva, aktivna i u interakciji s korisnikom. Aktivnost često može prolaziti između nastavljenih i zaustavljenih stanja (npr. kada uređaj spava, kada se isporučuje rezultat aktivnosti, kada se isporučuje nova namjera (kod ove metode treba biti jednostavan)).

Sve aktivnosti će implementirati onCreate (paket) kako bi napravili početnu postavku, može se također implementirati onPause() da izvrši promjene na podacima te pripremi se za pauziranje u interakciji s korisnikom, onStop() za rukovanje kada više nije vidljivo na zaslonu. Potrebno je pozvati super klasu prilikom primjene ovih metoda [11].

4. VERZIJE ANDROIDA

Svake godine Google objavi neku novu verziju Androida, svaka verzija je dobila ime po nekoj slastici. Trenutno ima 17 verzija (*slika 5.1 Lista android verzija*), najnovija je Android 10 koja je izašla 2019 godine.

Slika 5.1. Lista Android verzija [13]

Android 1.0 i 1.1 (bez imena):

Obje su verzije prve komercijalne verzije. Službeno su objavljene u 2008. i 2009. godine. Prva Android komercijalna verzija stavljena je na HTC uređaju. Ove verzije objavljene su bez imena. Značajke Androida 1.0: Google mape, kamera, Gmail, kontakti, Google sinkronizacija

Web-preglednika, bežična podrška WiFi i Bluetooth. Android 1.1: poruke dodaje i spremi privitak (engl. *save attachment*), pruža recenzije i pojedinosti o pretraživanju korisnika na kartama.

Android 1.5 (CupCake):

Android 1.5 izdan je u travnju 2009. Prva verzija sa službenim kodnim imenom CupCake. Donio je značajke u dizajnu korisničkog sučelja i ažurirao nekoliko novih značajki. Nova usluga prijenosa na YouTube i Picasa poput prijenosa videozapisa i fotografija. Podrška u MPEG-4, snimanje videozapisa, poboljšavanje mogućnosti kopiranja i lijepljenja web pretraživača.

Android 1.6 (Donut):

Android inačica 1.6, kodnim imenom Donut. Objavljena je u rujnu 2009. Sadrži različite značajke: podržava veliku veličinu zaslona, pružanje značajki galerije i fotoaparata, poboljšanje brzine i sistemske aplikacije.

Android 2.0 i 2.1 (Eclair):

Kodno ime za Android 2.0 je Eclair. Objavljen je u listopadu 2009., a verzija 2.1 objavljena je u prosincu 2009. Značajke su: ažuriranje korisničkog sučelja, podrška uživo "Live Wallpaper", podrška Bluetooth 2.1, poboljšanje Google karte, male promjene API –ja.

Android 2.2 (Froyo):

Objavljen je u svibnju 2010. Godine s kodnim imenom Froyo. Značajke su: podrška animiranom GIF, WiFi podrška Hotspot-u funkcionalnosti, poboljšana brzina, prijenos podataka podržan u pregledniku, podrška brojčanoj i alfanumeričkoj lozinki.

Android 2.3 i 2.4 (Gingerbread):

Gingerbread je na tržište izašao u prosincu 2010. Godine. Službeno je najavljen za Nexus S android telefon gdje je Google surađivao sa Samsungom. Poboljšano je kopiranje / lijepljenje, ažuriran je dizajn sučelja, podrška za društvene mreže, jednostavna upotreba tipkovnice.

Android 3.0, 3.1 i 3.2 (Honeycomb):

Android 3.0 objavljen je u veljači 2011, nakon čega su brzo slijedili 3.1 i 3.2 u srpnju i kolovozu 2011. Poboljšana aplikacija Gmail, ažurirano 3D korisničko sučelje, sinkronizacija medija sa SD karticom, Google e-knjige, video chat Google talk, podrška Adobe Flash u pregledniku, WiFi veze i zaključavanje visokih performansi, kineski rukopis.

Android 4.0 (Ice-Cream Sandwich):

Ice-Cream Sandwich objavljen je u listopadu 2011. Googleov pokušaj sintetiziranja Honeycomb. Neke značajke: poboljšan unos teksta i provjera pravopisa, izravni WiFi, objekt za uređenje fotografija, poboljšanje u ispravljanju tipkovnice, zaključavanje lica, poboljšanje razlučivosti video zapisa, poboljšane performanse kamere, do 16 kartica u web pregledniku.

Android 4.1, 4.2 i 4.3 (Jelly Bean):

Android 4.1 izašao je u srpnju 2012 s kodnim imenom Jelly Bean. Google sada (engl. *Google now*) je glavna značajka Jelly Bean-a. Koristi se kad god se želi pretraživati podatke s Google računa i podatke o lokaciji sa Android uređaja radi prikupljanja potrebnih podataka. Mnoge druge značajke su: glasovno pretraživanje, glatko korisničko sučelje, poboljšana aplikacija za kameru, poboljšana sigurnost, upisivanje glasa, više korisničkih računa samo na tabletu, podrška za 4k rezoluciju, podržava Bluetooth niske energije, dvosmjerna podrška za tekst i drugi jezik, podržava USB audio, poboljšano zaključavanje zaslona, mogućnost postavljanja glasnoće dolaznih poziva i prikaz upozorenja o poruci, podrška za osnovne emojije.

Android 4.4 (KitKat):

Android 4.4 s kodnim imenom KitKat, koji je Google objavio u rujnu 2013. Značajke su: snimanje na zaslonu, KitKat dodaje značajku u Google Now koja se naziva "OK Google" te omogućava korisniku pristup Googleu bez dodirivanja mobilnog telefona, GPS podrška, izvanmrežna glazbena podrška, ažuriranje korisničkog sučelja za navigaciju i alarm na Googleovoj karti, predstavljeni emoji na Google tipkovnici.

Android 5.0 i 5.1 (Lollipop):

Android 5.0 pod kodnim imenom Lollipop. Objavljen je u studenom 2014. Podrška vremenu izvođenja. Značajke su: ušteda baterije ne nekim uređajima, poboljšano korisničko sučelje, novi materijalni dizajn, ispravak pogrešaka, višestruka podrška za SIM kartice, glasovni poziv visoke kvalitete.

Android 6.0 (Marshmallow):

Marshmallow je izašao u svibnju 2015. Nove značajke su: ovjera za otiske prstiju, USB podrška za C tip, ušteda baterije 'Sleep Mode', model dozvole aplikacije-OPT (slanje zahtjeva za dozvolu), novi emojiji.

Android 7.0 (Nougat):

Android Nougat objavljen je u kolovozu 2016. Najavljen je s novi načinom dijeljena ekrana i značajkom čuvanja podataka. Ostale značajke su: omogućuje više zadataka, omogućuje način s više prozora, poboljšava upravitelja memorije (engl. *storage manager*), poboljšanje prikaza dodira.

Android 8.0 (Oreo):

Android 8.0 je izašao pod nazivom Oreo u kolovozu 2017. Ažurirane su neke nove značajke: podrška slike u slici (engl. *Picture in Picture,* skraćeno PIP), podrška za više zaslona, podrška za Google Play, prilagodljive ikone, poboljšan sustav obavijesti.

Android 9.0 (Pie):

Android Pie 9.0 je 16. verzija Android mobilnog operativnog sustava. Javno je objavljena u kolovozu 2018. godine. Značajke su: nova navigacija gestama, prilagodljiva baterija i svjetlina, poboljšane sigurnosne značajke, digitalni prosperitet, novi izbornik za bolju pristupačnost, nove prečice (engl. *screenshot*) za snimke zaslona, lakša rotacija zaslona, poboljšanje glasnoće i zvuka, mračni način odabira, lakši odabir teksta, više informacija o obavijestima [14].

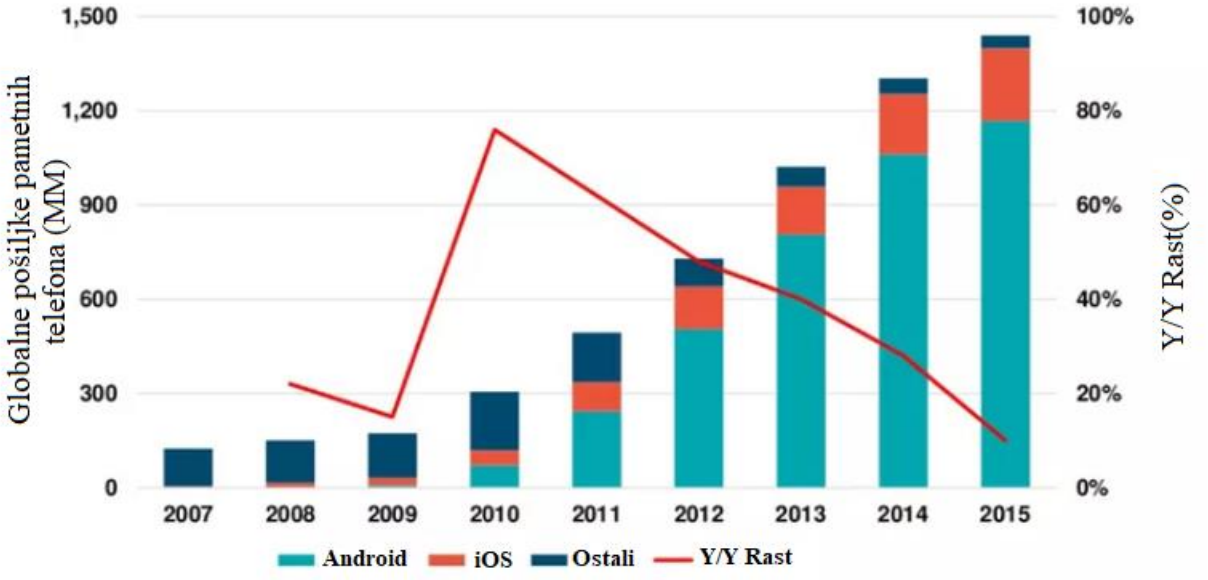
Android 10 (Q):

Android 10 izašao je u rujnu 2019. godine. Donosi jedan od najnovijih načina za kontrolu privatnosti, prilagođavanja telefona i izvršavanja radnji. Ostale značajke su: natpis uživo (engl. *Live Caption*) – na telefonu se automatski reproduciraju mediji s titlovima, pametan odgovor (engl. *Smart Reply*) – poduzima akciju dok odgovarate, pojačavač zvuka (engl. *Sound Amplifier*)-telefon pojačava zvuk, filtrira pozadinski šum i podešava kako se najbolje čuje, bez obzira na to što se radi (npr. razgovor s prijateljem, gledanje televizije, itd). Navigacija gestama (engl. *Gesture Navigation*) – pokreti su sada brži i intuitivniji nego ikad prije. Tamna tema (engl. *Dark theme*) – koristi crnu boju kako bi duže održala bateriju na životu, mijenja izgled aplikacija. Najnoviji uređaji koji mijenja igre – 5G uređaji dostupni su samo na Androidu, tako se može iskusiti najnovije i najbolje značajke koje se savijaju i kreću brže nego ikad prije. Drži podatke privatnim s više kontrola, brže dobivanje sigurnosnog ažuriranja, digitalno blagostanje (engl. *Digital Wellbeing*), način fokusiranja (engl. *Focus mode*) – za isključivanje ometajućih aplikacija.

Obiteljska veza (engl. *Family Link*) – mogućnost postavljanja vremenskog ograničenja zaslona, pregled aktivnosti aplikacija, upravlja aplikacijama i ograničava sadržaj [15].

5. USPOREDBA ANDROIDA I IOSa

Na tržištu operativnih sustava za pametne telefone i tableta postoji nekoliko različitih operativnih sustava, tržištem dominiraju samo dva Android i iOS. Najpopularniji na tržištu je Android koji se aktivno koristi na više od 2 milijarde uređaja, dok iOS koristi više od 1.3 milijarde uređaja (*Graf 6.1 Zastupljenost operativnih sustava kod pametnih telefona*).



Graf 6.1 Zastupljenost operativnih sustava kod pametnih telefona [16]

Performanse:

iOS su slabijih karakteristika nego što su to pametni uređaji koji rade na Android operativnom sustavu. Apple proizvodi dobro optimizirane uređaje, gdje se iz svakog komada hardvera izvlači maksimum, što rezultira vrhunskim radom na pametnom telefonu. Neki Android uređaji imaju više RAM-a i bolju rezoluciju zaslona od Appleovih, što ne znači da su bolji u tom segmentu. Starije verzije Apple uređaja pokazuju se bolje od nekih novih Android uređaja, koji bi prema postavkama na papirima trebali raditi brže. iOS uređaji zbog sjajne optimizacije hardvera i softvera imaju bolje performanse od Android uređaja.

Zaslon:

Pametni uređaji imaju zaslon baziran na IPS, LCD ili OLED tehnologiji. OLED tehnologija ima nekoliko prednosti pred IPS tehnologijom. OLED tehnologija značajno štedi bateriju jer svaki piksel emitira svoje svjetlo pa nema potrebe za pozadinskim osvjetljenjem. Vidljivost je dobra iz različitih kuteva, ali emitira i manje svjetla. Prednost IPS zaslona je ta što je jeftiniji od OLED zaslona. Android i iOS imaju podjednake karakteristike i tehnologije zaslona.

Kamera:

Kamera je jedna od važnijih elemenata pametnog telefona, koja može utjecati na odluku kupca koji pametni uređaj da kupi. Apple je godinama proizvodio kvalitetne kamere visokih performansi s kojim je dominirao tržištem. Danas se dio pametnih telefona baziranih na Android operativnom sustavu približio standardu Apple, a neki su ga i prestigli. Huawei koji koristi Android operativni sustav ima ugrađene kamere koje imaju performanse kvalitetnih fotoaparata. Android i iOS imaju podjednake karakteristike i tehnologije kamera.

Aplikacije:

Osnovna razlika između iOS i Android sustava je u zatvorenosti, odnosno otvorenosti sustava. Android je sustav otvorenog koda koji je okrenut idejama programera i promjenama, dok je struktura iOS-a zatvorena. Korisnici Android uređaja unutar Play Store-a imaju više od 2.6 milijuna različitih aplikacija koje mogu preuzeti, dok je na App Store-u za iOS korisnike dostupno više od 2.2 milijuna aplikacija. Android sustav ima veću ponudu, dok korisnici iOS sustava uglavnom prvi dobivaju nove nadogradnje i funkcije aplikacija. Android omogućava da korisnici mogu nabavljati aplikacije od trećih strana (npr. Amazon AppStore), te omogućava instaliranje testnih aplikacija putem .apk datoteke.

Mogućnost odabira:

Apple posljednjih godina gubi na svojoj inovativnosti, pa tako novi modeli ne nude ništa novo i spektakularno. Apple uređaji su uglavnom sličnog dizajna i karakteristika. Android pametni uređaji imaju znatno bogatiju i šareniju ponudu. Pametni uređaji s Android operativnim sustavom svake godine predstavi mnoštvo kompanija (Samsung, Huawei, Xiaomi, …), što daje veliku širinu na tržištu. Uređaje kineskih proizvođača bazirani na Android operativnom sustavu su se probili na tržištu zbog svoje niske cijene, a karakteristikama konkuriraju Apple i svim ostalim uređajima. Android uređaji imaju veću mogućnost odabira u odnosu na iOS.

Prilagodba potrebama korisnika:

IOS korisnicima donosi dizajn koji nije u potpunosti prilagodljiv i zbog toga se neće moći podesiti prema svim željama korisnika. Android je puno prilagodljiviji i dopušta korisnicima da sustav prilagođavaju svojim željama i potrebama i to u različitim segmentima. Android operativni sustav svojom otvorenošću osigurava veliku razinu prilagodljivosti korisnicima, ali i proizvođačima. IOS osigurava sigurnost sustava, ali sprječava korisnike u potpunoj prilagodbi uređaja.

Nadogradnje:

Zbog visokih cijena pametnih telefona većina korisnika kupuje svoje pametne telefone svake dvije godine. Kako bi uređaj kvalitetno funkcionirao i pratio trendove i nove aplikacije, potrebno je da se stalno nadograđuje. Zbog malog broja Apple uređaja na tržištu, nadogradnje se mogu puno brže razvijati i plasirati na uređaje korisnika. Android svake godine ima stotine različitih novih modela uređaja, što znatno usporava pripremu nadogradnju. Ponekad prođe nekoliko mjeseci kako bi se napravila kvalitetna nadogradnja Android sustava, što može utjecati na nesmetan rad korisnika. iOS uređaji svoje nadogradnje dobivaju redovito i s novim nadogradnja uglavnom dođu i nove super značajke. Android nadogradnja ovisi o tome koliko je kvalitetan korisnikov uređaj. Top modeli dobivaju najviše nadogradnji, dok će oni slabiji i zastarjeli modeli ponekad ostati i bez novih nadogradnji.

Umjetna inteligencija:

Umjetna inteligencija upravlja određenim dijelom zadataka na pametnom uređaju. Proizvođači tu karakteristiku sve više naglašavaju, tako da će i korisnici ubrzo shvatiti kako je umjetna inteligencija bitan saveznik za kvalitetniji i efikasniji rad na pametnom telefonu. Google je napravio veliki pomak na području umjetne inteligencije razvojem Google Assistant koji je pametni virtualni asistent. Google Assistant nauči navike korisnika i zatim pomaže u obavljanju svakodnevnih zadataka. Asistenti će u budućnosti igra veliku ulogu kod odabira pametnih telefona.

Cijena:

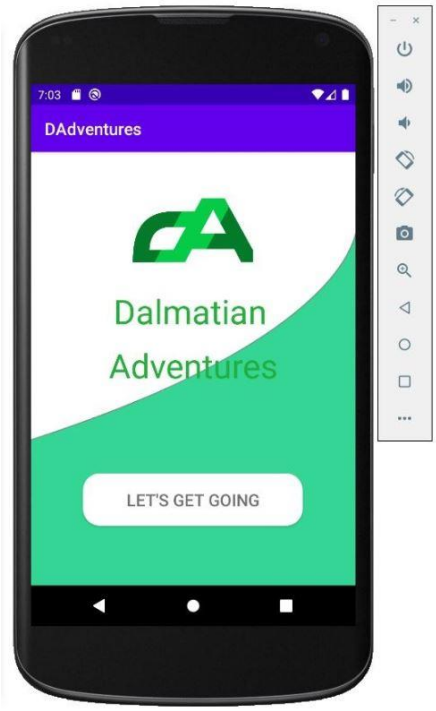
Apple uređaji su na tržištu pozicionirani kao top brend i zbog toga imaju iznimno visoke cijene. Posljednji modele korisnik će platiti i više od 1.000 dolara, što je cijena koju si mnogi korisnici ne mogu priuštiti. Cijena Android uređaja ovisi o karakteristikama samog uređaja.

Android uređaji kineskih proizvođača postali su popularni zbog svoje male cijene izrade dostupni su po vrlo niskim cijenama krajnjim korisnicima. Takvi uređaji po kvaliteti su uvijek bili znatno ispod top brendova, posljednjih godina porasla je kvaliteta izrade takvih uređaja, koji sada dolaze s vrhunskim karakteristikama i usklađenim hardverom [17].

6. PRAKTIČNI RAD: APLIKACIJA ZA PRONALAŽENJE PUSTOLOVNIH IZLETA

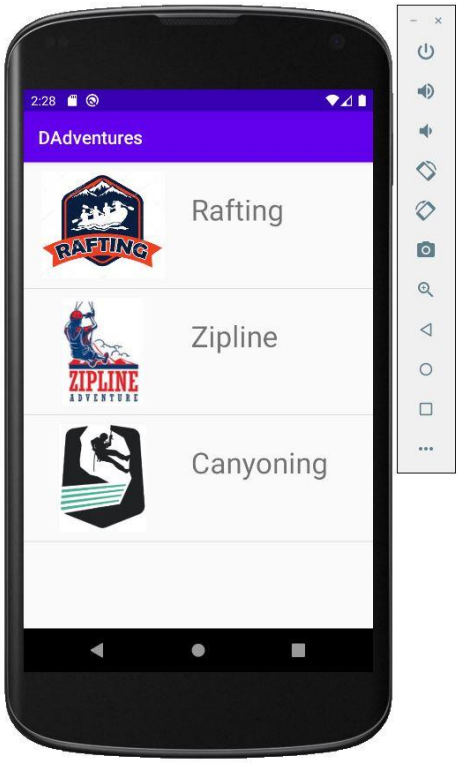
U praktičnom radu je realizirana Android aplikacija koja pomaže pronaći turistima i svim ostalim gostima pustolovne aktivnosti (rafting, zipline, kanjoning) unutar Srednje Dalmacije. Gdje korisnik aplikacije može u nekoliko klikova pronaći željeni pustolovni izlet. Projekt je realiziran tako da korisnik aplikacije može naći sve informacije o željenoj aktivnosti te kontaktirati agenciju za eventualnu narudžbu. Narudžbe se izvršavaju nekoliko dana unaprijed, da se može organizirati potrebna oprema, vodiči te prijevoz. Pustolovni izleti koji se nalaze unutar aplikacije su rafting, zipline i kanjoning. Rafting na rijeci Cetini te kanjoning koji su veoma popularni brojnim gostima zbog prekrasne prirode i osvježavajuće rijeke na visokim temperaturama tijekom ljeta. Zipline na planini iznad grada Omiša i na planini Kozjak, koji uključuje planinarenje te spuštanje niz planinu. Cjelokupni kod projekta se nalazi na GitHubu: https://github.com/apupac00/ZavrsniRad.

Aplikacija je napisana programskom jeziku Java, te razvijena u Android studiu. Ime aplikacije je Dalmatinske Avanture (engl. *Dalmatian Adventures*) (*Slika 6.1. Početni aktivnost aplikacije*).



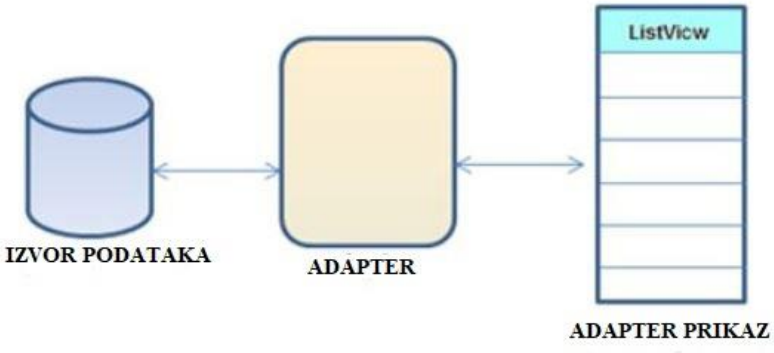
Slika 6.1. Početna aktivnost aplikacije

Klikom na dugme (engl. *button*) “LET’S GET GOING” otvara se aktivnost koja sadrži tri moguća odabira Rafting, Zipline i Canyoning, realizirano pomoću ListView (*Slika 6.2. Aktivnost s ponudama pustolovnih izleta*).



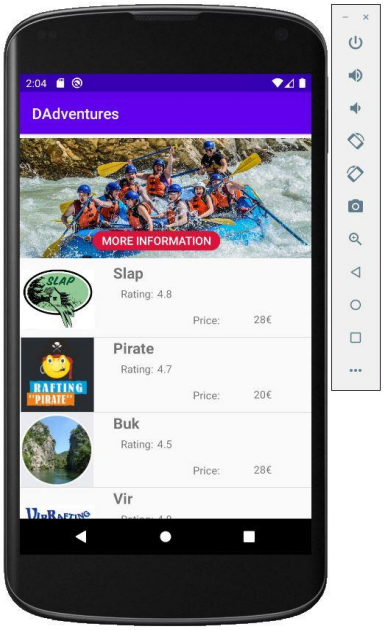
Slika 6.2. Aktivnost s ponudama pustolovnih izleta

Android ListView je prikaz koji grupira stavke i prikazuje ih na okomitom popisu koji se može pomicati. Stavke popisa automatski se ubacuje na popis pomoću adaptera koji povlači sadržaj iz izvora, poput polja ili baze podataka (*Slika 6.1.2. ListView*) [18].



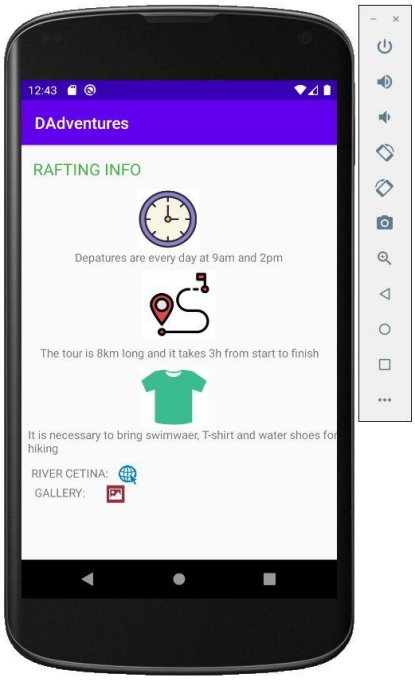
Slika 6.3. ListView [19]

Dalje klikom na jednu od aktivnosti (Rafting, Zipline i Canyoning) otvara se aktivnost s imenima pojedinih firmi i agencija koje daju usluge pojednih pustolovnih izleta, te za one koje žele saznati nešto više o pojedinoj aktivnosti (*Slika 6.3. Aktivnost s ponudama i cijenama pustolovnih izleta*).



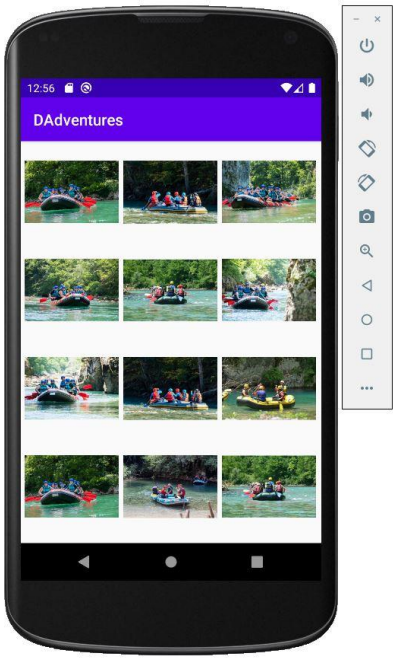
Slika 6.4. Aktivnost s ponudama i cijenama pustolovnih izleta

Klikom na dugme "MORE INFORMATION" otvara se aktivnost s dodatnim informacijama o raftingu (*Slika 6.5. Rafting informacije*).



Slika 6.5. Rafting informacije

Klikom na ikonu galerija otvara se galerija sa slikama rafting realizirana pomoću GridView koji radi na sličnom principu kao i ListView samo što su slike organizirane u stupcima i redcima. Klikom na pojedinu sliku, slika se uveća na cijeli ekran (*Slika 6.6. Galerija*).



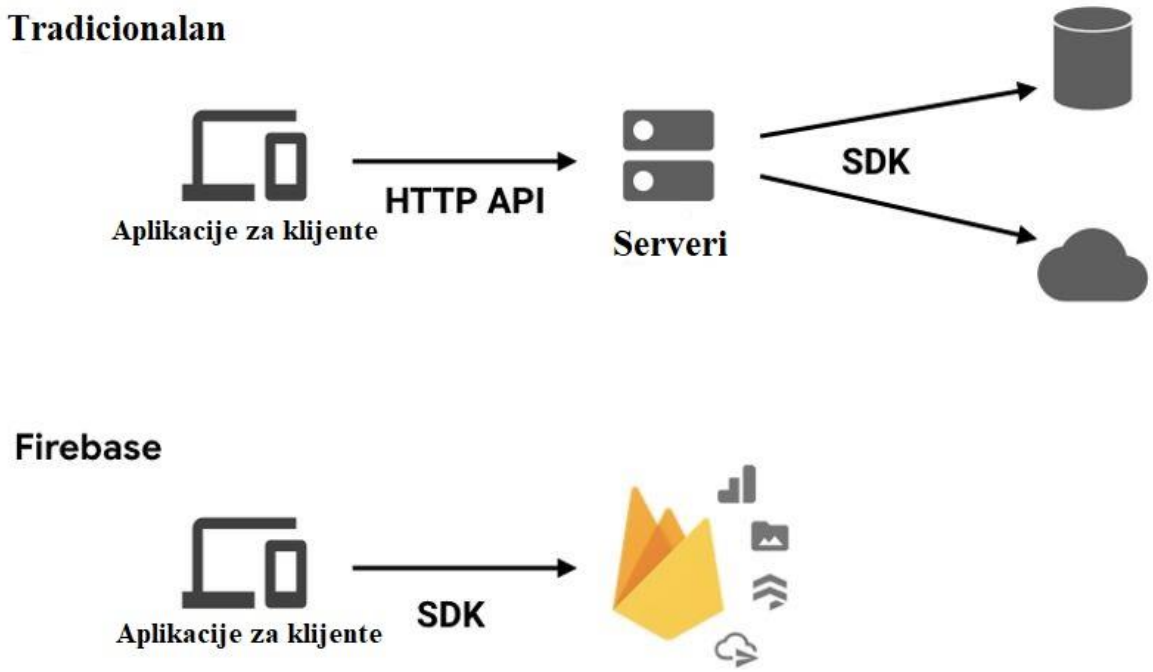
Slika 6.6. Galerija

Klikom na ikonu svjetska mreža (engl. *World Wide Web,* skraćeno WWW) otvara se web stranica s dodatni informacijama o rijeci.

6.1. Firebase

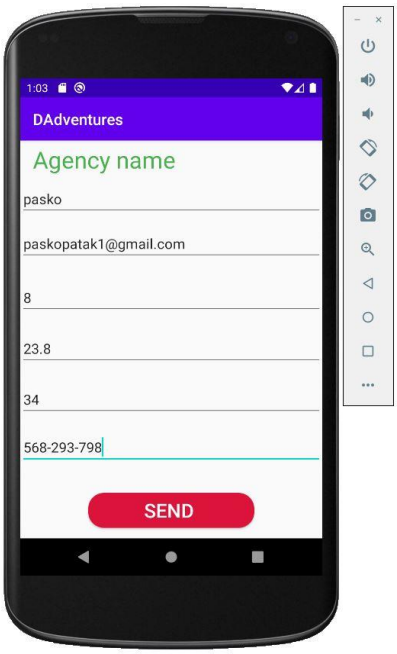
Firebase je Googleova platforma za razvoj mobilnih aplikacija, skup je alata za izgradnju, poboljšanje i razvoj aplikacije. Alati pokrivaju velik dio usluge koje bi programeri obično sami trebali izgraditi, ali ne žele izgraditi jer se trebaju usredotočiti na samu aplikaciju. Firebase uključuje stvari poput analitike, provjere autentičnosti, baze podataka, konfiguracije, pohrane datoteka, push poruka (engl. *push messaging*), itd. Usluge su smještene u oblak (engl. *cloud*) [16].

Paket SDK-a klijenta koje pruža Firebase izravno komunicira s pozadinskim uslugama (engl. *backend services*), bez potrebe za uspostavljanjem bilo kakvog srednjeg softvera između aplikacije i usluge. Korištenjem jedne od usluga, obično se piše kod za pretraživanje baze podataka u aplikaciji klijenta. To je različito od tradicionalnog razvoja aplikacija, što obično uključuje pisanje i softvera za sučelje i pozadinski sustav (engl. *frontend and backend*). Kod sučelja samo poziva krajnje točke API-ja izložene pozadinskom sustavu, a pozadinski kod sam odrađuje posao. Firebase proizvodima zaobilazi se tradicionalni pozadinski sustav, stavljajući posao na klijenta. Administrativni pristupa svakom od ovih proizvoda pruža Firebase konzola (*Slika 6.2.1. Usporedba tradicionalnog pristupa u odnosu na Firebase*) [20].



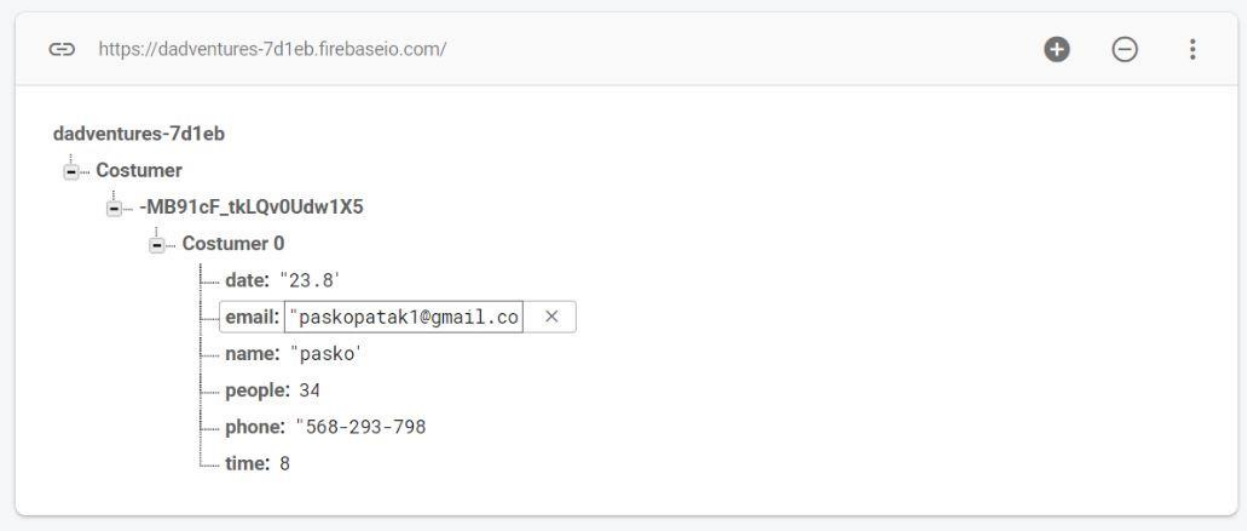
Slika 6.1.1. Usporedba tradicionalnog pristupa u odnosu na Firebase [20]

Podatke koje korisnik unese unutar aplikacije šalju se na Firebase. Podaci koji se šalju su ime, email, broj mobitela, broj osoba, vrijeme i datum kada se žele rezervirati, tako da se može organizirati sve potrebno da se ostvari pustolovni izlet (*Slika 6.2.2. Rezervacija pustolovnog izleta*).



Slika 6.1.2. Rezervacija putolovnog izleta

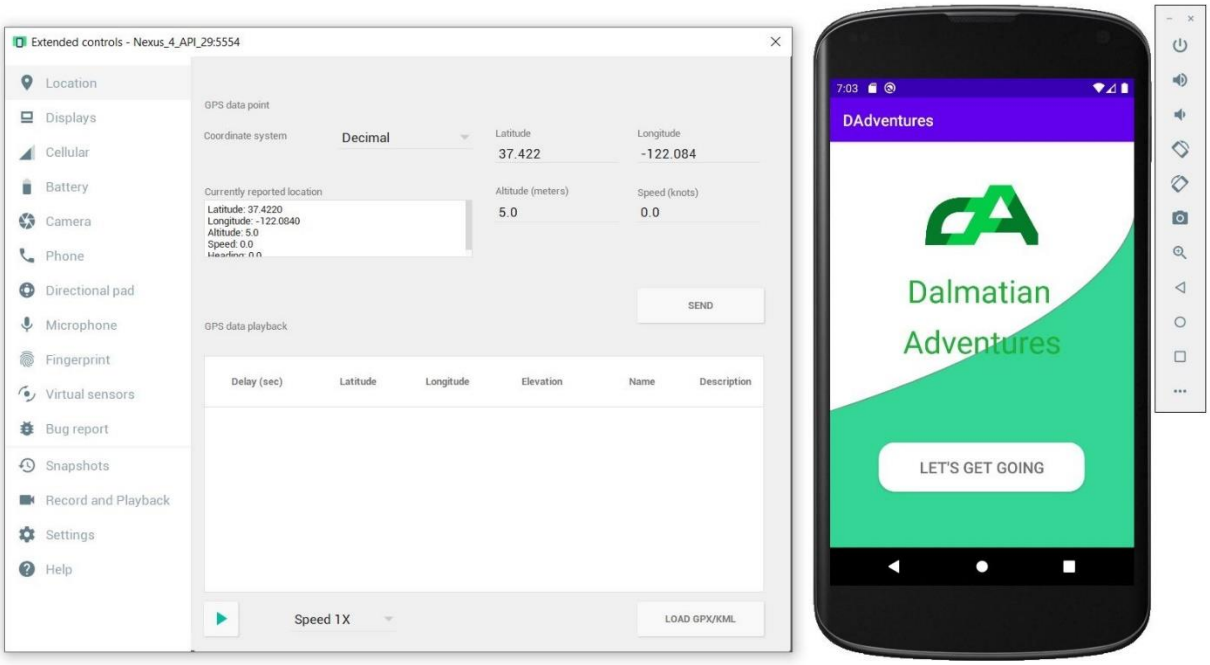
Kada korisnik popuni sva obvezna polja, te klikne na dugme “Send” podaci se šalju na Firebase. Podaci se spremaju na Firebase unutar baze podataka u stvarnom vremenu (engl. *Realtime Database*) u JSON (engl. *JavaScript Object Notation, skraćeno JSON*) formatu (*Slika 6.2.3. Baza podataka u stvarnom vremenu*). Poslani podaci su trajno pohranjeni unutar baze podataka.



Slika 6.1.3. Baza podataka u stvarnom vremenu

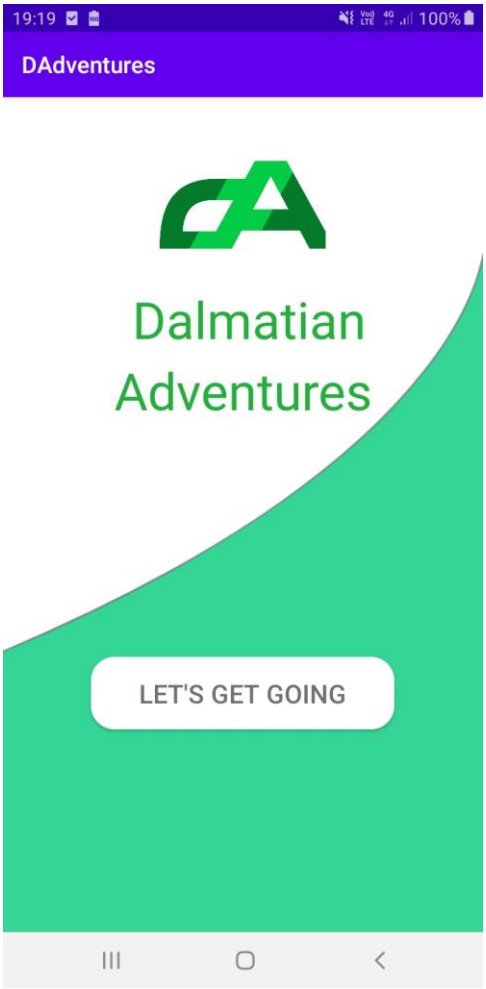
6.2. Testiranje aplikacije

Prvi način testiranja aplikacije se može izvršiti na računalu pomoću Android studio aplikacije koja sadrži virtualni stroj koji emulira stvarne performanse različitih mobilnih uređaja, koje je potrebno instalirati za svaki pojedini uređaj (*Slika 6.3.1. Prikaz aplikacije na Nexus_4 mobilnom uređaju*).



Slika 6.2.1. Prikaz aplikacije na Nexus_4 mobilnom uređaju

Drugi način testiranja aplikacije je da putem USB (engl. *Universal Serial Bus, skraćeno USB*) kabela povežemo stvarni uređaj s računalom na kojem se nalazi kod. Pritom ne treba uzimati jedan dio procesora računala da kreira virtualni stroj što usporava računalo. Nedostatak testiranja aplikacije preko USB-a je taj da možemo prikazati aplikaciju samo na mobilnim uređajima koje posjedujemo (*Slika 6.3.2. Prikaz aplikacije na Samsung SM-J610FN mobilnom uređaju*).



Slika 6.2.2. Prikaz aplikacije na Samsung SM-J610FN mobilnom uređaju

7. ZAKLJUČAK

Android je najzastupljeniji operativni sustav kod mobilnih uređaja. Mobilni uređaji sve više zamjenjuju računala, zato što mogu raditi većinu funkcionalnosti kao i računala. Postoji velik broj Android aplikacija koje se koriste da olakšaju svakodnevicu. Aplikacije se mogu koristiti za kupovinu, zabavu te edukaciju. Android je besplatna razvojna platforma koja se temelji na Linuxu. Otvorenog je koda tako da svatko može testirati vlastite aplikacije bilo to preko virtualnog ili stvarnog uređaja. Virtualni uređaji se sve više svojim performansama približavaju stvarnome uređaju.

Android aplikacija koja je realizirana unutar praktičnog dijela namijenjena je da pomogne pronaći neki od pustolovnih izleta u Srednjoj Dalmaciji, daje osnovne informacije o cijenama i ponudama. Tako korisnik aplikacije može saznati sve potrebne informacije o određenom pustolovnom izletu te naručiti se u nekoliko klikova. Podaci koje korisnik upiše šalju se na Firebase u JSON formatu koji je razvio Google da olakša developerima razvoj aplikacija da ne troše vrijeme na serveru već da se posvete samoj aplikaciji.

LITERATURA

[1] Ahamed Shibly: "Android operativni sustav: Arhitektura, sigurnost, izazovi i rješenja", s Interneta, https://www.researchgate.net/publication/299394606_Android_Operating_System_Architecture_Security_Challenges_and_Solutions, 6. veljače 2020.

[2] Reto Meier: "PROFESSIONAL Android™ 4 Aplication Development", John Wiley and Sons, Indiana, Indianapolis, 2012.

[3] Javad Ahmed Shaheen, Mian Ali Asghar, Abid Hussain: "Android operativni sustav sa svojom arhitekturom i Androidovom aplikacijom s pregledom Dalvik virtualnog stroja ", s Interneta, http://gvpress.com/journals/IJMUE/vol12_no7/3.pdf, 6. veljača 2020.

[4] Tam H. Doan: "Virtualni uređaj u Androidu: sve što trebate znati", s Interneta, https://android.jlelse.eu/virtual-machine-in-android-everything-you-need-to-know-9ec695f7313b, 8. veljače 2020.

[5] David Ehringer: "Arhitektura Dalvik virtualnog stroja", s Interneta, http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf, 18. veljača 2020.

[6] Adam Sinicki: "Android SDK tutorial za početnike", s Interneta, https://www.androidauthority.com/android-sdk-tutorial-beginners-634376/, 10. veljače 2020

[7] Zesoi – FER-a:"Računalna forenzika", s Interneta, http://comfor.zesoi.fer.hr/doku.php?id=racfor_wiki:android:tehnike_otkljucavanja_zakljucanih_android_uredaja, 17. veljače 2020.

[8] Android developers: "Android program za uklanjanje pogrešaka", s Interneta, https://developer.android.com/studio/command-line/adb.html, 17. veljače 2020.

[9] Android developers: "Pokretanje aplikacije na Android emulatoru", s Interneta, https://developer.android.com/studio/run/emulator, 17. veljače 2020.

[10] Adam Sinicki: "Anatomija aplikacije: uvod u životne cikluse aktivnosti", s Interneta, https://www.androidauthority.com/android-app-lifecycle-879968/, 10. veljače 2020.

[11] Android developers: "Aktivnost", s Interneta, https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle), 11. veljače 2020.

[12] Android developers: “Razumjevanje životnog ciklusa aktivnosti”, s Interneta, https://developer.android.com/guide/components/activities/activity-lifecycle.html, 10. Veljače 2020.

[13] Quora: “Zašto je Android Q postao Android 10”, s Interneta, https://www.quora.com/Why-did-Android-Q-become-Android-10, 11. veljače 2020.

[14] JR Raphael: “Verzije Androida: povijest od 1.0 do 11”, s Interneta, https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html, 11. veljače 2020.

[15] Android developers: “Android 10”, s internet, https://www.android.com/android-10/, 17. veljače 2020.

[16] THE VERGE: “Čitava povijest iPhonea i Androida sažeta na dva grafa”, s Interneta, https://www.theverge.com/2016/6/1/11836816/iphone-vs-android-history-charts, 13. veljače 2020.

[17] PCCHIP: “Android vs Ios: Koji je operativni sustav i uređaj idealan za vas?”, s Interneta, https://pcchip.hr/softver/must-have/android-vs-ios-operativni-sustav/, 13. veljače 2020.

[18] Tutorialspoint: “Android ListView”, s Interneta, https://www.tutorialspoint.com/android/android_list_view.htm, 29. lipnja 2020.

[19] CareerRide.com: “Android adapter s kontrolom spinera”, s Interneta, https://www.careerride.com/page/android-adapter-with-spinner-control-595.aspx, 29. lipanj 2020.

[20] Doug Stevenson: “Što je Firebase? Kompletna priča”, s Interneta, https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0, 29. lipanj 2020.

POPIS OZNAKA I KRATICA

ADB (engl, *Android Debug Bridge*)	Android program za uklanjanje pogrešaka
AVD (engl, *Android Virtual Device*)	Android virtualni uređaj
AOSP (engl, *Android Open Source Project*)	Android projekt otvorenog koda
API (engl, *Application Programming Interface*)	Sučelje za programiranje aplikacija
APK (engl, *Android Package Kit*)	Android paket
CPU (engl, *Central processing unit*)	Središnja procesorska jedinica
DEX (engl, *Dalvik Executable format*)	Dalvik izvršni
DVM (engl, *Dalvik Virtual Machine*)	Dalvik virtualni stroj
engl	engleski
GC (engl, *Garbage Collection*)	Skupljač smeća
GNU (engl, *GNU's Not Unix*)	GNU nije Unix
GPS (engl, *Global Positioning System*)	Globalni sustav pozicioniranja
IDE (engl, *Integrated development environment*)	Integrirano razvojno okruženje
iOS (engl, *iPhone Operating System*)	iPhone operativni sustav
IPS (engl, *in-plane switching*)	Unutarnje prebacivanje
JAR (engl, *Java ARchive*)	Java ARhive
JIT (engl, *Just-In-Time*)	Na vrijeme
JSON (engl, *JavaScript Object Notation*)	JavaScript objekt notacija
JVM (engl, *Java Virtual Machine*)	Java virtualni stroj
LCD (engl, *Liquid Crystal Display*)	Zaslon s tekućim kristalima
NFC (engl, *Near Field Communication*)	Komunikacija bliskog polja
OHA (engl, *Open Handset Alliance*)	Open Handset Alliance
OLED (engl, *Organic Light Emitting Device*)	Uređaj koji emitira organsku svjetlost

PIP (engl, *Picture in picture*)	Slika u slici
RAM (engl, *Random Access Memories*)	Nasumičan pristup memoriji
SDK (engl, *Software development kit*)	Oprema za razvoj softvera
SD Card (engl, *Secure Digital Card*)	Sigurna digitalna kartica
SGL (engl, *Scalable Graphics Library*)	Skalabilna grafička biblioteka
SSL (engl, *Secure Socket Layer*)	Sigurni sockets sloj
TCP (engl, *Transmission Control Protocol*)	Protokol kontrole prijenosa
USB (engl, *Universal Serial Bus*)	Univerzalna serijska sabirnica
VM (engl, *Virtual Machine*)	Virtualni stroj
WWW (engl, *World Wide Web*)	Svjetska mreža

SAŽETAK

Završni rad je organiziran tako da se sastoji od dva dijela. U prvom dijelu detaljno je objašnjen Android operativni sustav sa svim komponentama. Dalvik virtualni uređaj se odnosi na testiranje Android aplikacije, gdje virtualni uređaj svojim performansama pokušava što vjernije emulirati stvarni uređaj. Životni ciklus aplikacije od prvog pokretanje aktivnosti pa do njenog konačnog uništenja. Detaljna usporedba Androida i iOS-a operativnog sustava, gdje im je najveća razlika u otvorenosti i zatvorenosti sustava. Verzije Androida od prve verzije 1.0 koja je imala samo nekoliko značajki pa sve do posljednje verzije 10.

U drugom djelu je realiziran praktični dio koji je Android aplikacija za pronalaženje pustolovnih izleta u Srednjoj Dalmaciji napisan u Java programskom jeziku, te razvijen u Android studiju. Korisnik aplikacije može u nekoliko klikova saznati sve potrebne informacije o nekom pustolovnom izletu.

KLJUČNE RIJEČI: Android operativni sustav, Dalvik virtualni stroj, životni ciklus aplikacije, Android studio i Firebase.

SUMMARY

ANDROID APPLICATION FOR FINDING ADVENTURE TRIPS

The final work is organized so that is consists of two parts. The first part explains in detail the Android operating system with all components. Dalvik virtual device refers to testing an Android application, where the virtual device tries to emulate the real device as faithfully as possible with its performance. The application lifecycle from the first start of the activity to its final destruction. A detailed comparison of Android and iOS operating system, where their biggest difference is in the openness and closedness of the system. Android versions from the first version 1.0 which had only a few features all the way to last version 10.

In the second part, a practical part of an Android application for finding adventure trips in Central Dalmatia was written in the Java programming language, and developed in the Android studio. The user of the application can find all the necessary information about an adventure trip in a few clicks.

KEY WORDS: Android operating system, Dalvik virtual machine, application life cycle, Android studio and Firebase.