UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Erik Pipan

Razvoj mobilne rešitve z uporabo različnih programskih ogrodij

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE

MENTOR: viš. pred. dr. Aljaž Zrnec

Ljubljana, 2014

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00439 / 2013
Datum: 10.4.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ERIK PIPAN**

Naslov: **RAZVOJ MOBILNE REŠITVE Z UPORABO RAZLIČNIH PROGRAMSKIH OGRODIJ**
THE DEVELOPMENT OF MOBILE APPLICATION USING VARIOUS PROGRAMMING FRAMEWORKS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Zaradi vse večjega povpraševanja po mobilnih napravah se povečuje tudi povpraševanje po mobilnih aplikacijah. Tukaj se pojavi vprašanje, kako razviti aplikacijo, ki se bo izvajala na vseh aktualnih platformah. V prvem delu diplomske naloge predstavite ogrodje PhoneGap in nativno ogrodje za razvoj mobilnih aplikacij na platformi Android. PhoneGap omogoča razvoj mobilne aplikacije za več platform. Tukaj predstavite tudi HTML, CSS in JavaScript ter pripadajoče knjiznice JQuery in JQuery Mobile. V okviru drugega dela naloge primerjajte razvoj testne aplikacije z uporabo obeh ogrodij. Pri tem izpostavite poglavitne prednosti in slabosti obeh, ter podajte priporočila glede uporabe omenjenih ogrodij.

Mentor:
viš. pred. dr. Aljaž Zrnec

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:
prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a **Erik Pipan**,

z vpisno številko **63100225**,

sem avtor/-ica diplomskega dela z naslovom:

Razvoj mobilne rešitve z uporabo različnih programskih ogrodij

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  viš. pred. dr. Aljaž Zrnec

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 14.3.2014 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se mentorju viš. pred. dr. Aljažu Zrnecu za nasvete in pomoč pri izdelavi diplomske naloge.

Posebej se zahvaljujem svoji družini, ki so mi v času študija vseskozi stali ob strani ter me spodbujali.

Kazalo

Seznam uporabljenih kratic in simbolov

ADT	angl. *Android Development Tools* – orodja za razvoj Android aplikacij
AJAX	angl. *Asynchronous JavaScript and XML* – asinhroni JavaScript in XML
API	angl. *Application Programming Interface* – vmesnik za programiranje aplikacij
CDT	angl. *Eclipse's C/C++ Development Tooling* – eclipse orodja za razvoj C in C++ aplikacij
CSS	angl. *Cascade Style Sheet* – kaskadne stilske podloge
DOM	angl. *Document Object Model* – model, ki predstavlja interaktivnost z objekti
DP	angl. *Density Pixel* – merska enota za resolucijo naprav v različnih kontekstih
HTML	angl. *Hyper Text Markup Language* – jezik za označevanje nadbesedila
IDE	angl. *Integrated Development Environment* – integrirano razvojno okolje
JDT	angl. *Java Development Tools* – orodja za razvoj javanskih aplikacij
PDT	angl. *Eclipse's PHP Development Tools* – eclipse orodja za razvoj PHP aplikacij
SDK	angl. *Software Development Kit* – paket za razvoj programske opreme
W3C	angl. World Wide Web Consortium – organizacija za spletne standarde
XHTML	angl. *Extensible Hypertext Markup Language* – označevalni jezik, ki ima enak namen kot HTML, vendar je usklajen s sintakso XML
XML	angl. *eXtenisble Markup Language* – razširljiv označevalni jezik

Povzetek

Cilj diplomskega dela je razvoj mobilne rešitve z uporabo različnih programskih ogrodij. Prvi del diplomskega dela je namenjen predstavitvi uporabljenih tehnologij. Drugi in najbolj obsežni del diplomskega dela vsebuje predstavitev razvoja mobilne aplikacije in je razdeljen na tri dele. Prvi del vsebuje opis metodologije in ideje razvoja aplikacije ter analize, kjer določimo zahteve in specifikacije zahtev mobilne aplikacije. Drugi del je namenjen opisu razvoja mobilne aplikacije v avtohtonem ogrodju in tretji del je namenjen opisu razvoja mobilne aplikacije v PhoneGap oziroma hibridnem ogrodju. V tretjem delu diplomskega dela pa predstavimo primerjavo med avtohtonim razvojnim ogrodjem in Phonegap oziroma hibridnim razvojnim ogrodjem.

Ključne besede: mobilne aplikacije, Android, Android aplikacija, hibridno ogrodje, avtohtono ogrodje.

Abstract

The following Bechelor's thesis discusses the development of mobile application using various programming frameworks. The first part of the thesis is devoted to the presentation of the technologies and tools used in mobile application development. The second and most extensive part of the thesis presents the development of mobile applications and is divided into three parts. The first part describes methodology, application development idea and analysis which sets out the requirements and specifications of the requirements for mobile applications. The second part is dedicated to the description of the development of mobile application with native framework and the third part is dedicated to the description of the development of mobile application with hybrid framework. The third part of the thesis discusses a comparison between native and hybrid frameworks.

Keywords: mobile applications, Android, Android application, hybrid framework, native framework.

1. UVOD

V zadnjih letih se je mobilna tehnologija začela izredno hitro razvijati. Mobilno napravo si sedaj lasti skoraj vsak izmed nas in postala je skoraj neizogiben del našega vsakdana. S strani največjih informacijskih koorporacij smo v obdobju zadnjih let bili priča razvoju novih mobilnih operacijskih sistemov. Vzporedno se je povečalo tudi število mobilnih aplikacij za posamezni mobilni operacijski sistem. Vsaka od njih pa za razvoj uporablja svoj programski jezik. Poleg tega se je podoba svetovnega spleta v zadnjem desetletju temeljito spremenila. K temu je pripomogla široka skupnost razvijalcev in z razvojem spletnih tehnologij se je hitrost izmenjevanja informacij bistveno povečala. Tako je vzporedno z razvojem mobilnih platform začelo nastajati veliko ogrodij, ki za razvoj mobilnih aplikacij uporabljajo spletne tehnologije. Zaradi tega smo se odločili, da bomo v okviru diplomske naloge opisali razvoj mobilne aplikacije FileBrowser z dvema razvojnima ogrodjema, avtohtonim (angl. native framework) in hibridnim (angl. hybrid framework). Na podlagi tega smo lahko prišli do prednosti in slabosti obeh pristopov in podali priporočila razvijalcem glede uporabe predstavljenih ogrodij.

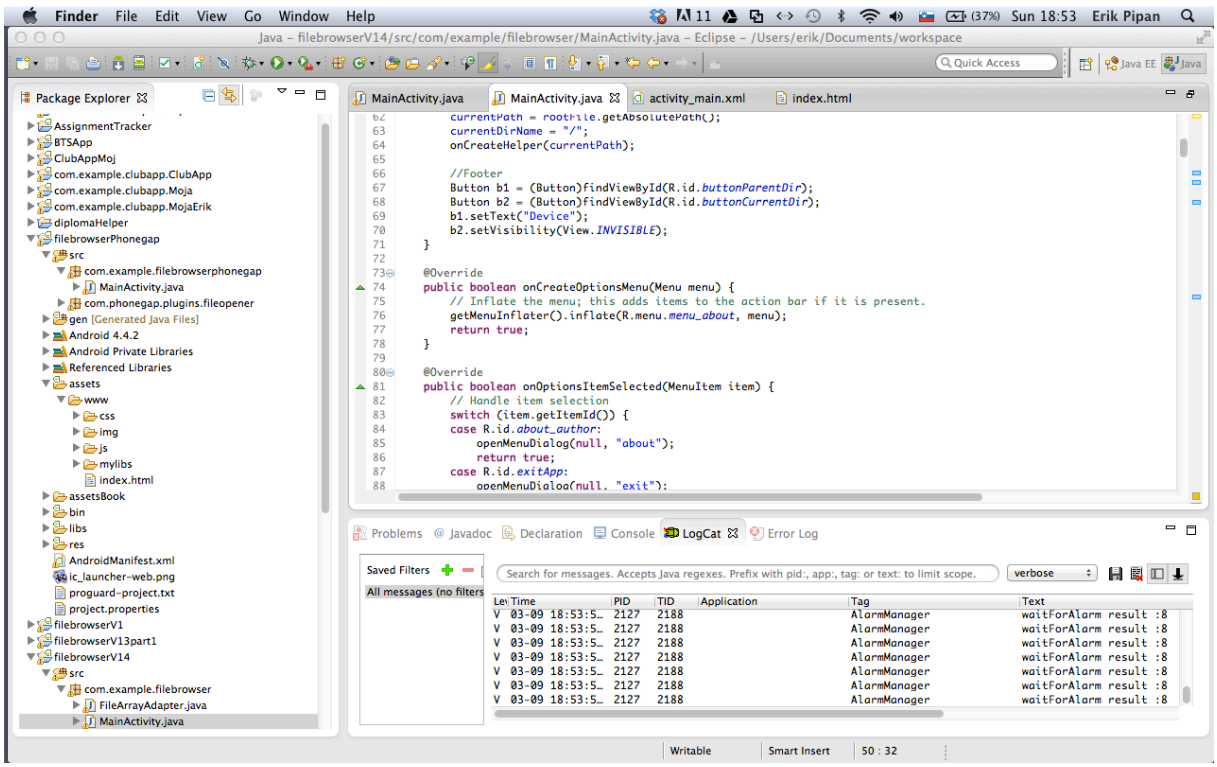
V nadaljevanju diplomskega dela smo najprej predstavili tehnologije, ki se uporabljajo v okviru posameznega programskega ogrodja. Nato smo predstavili postopek razvoja aplikacije z uporabo dveh različnih ogrodij in izvedli primerjavo med njima z uporabo več kriterijev, na podlagi česar smo prišli do priporočil glede uporabe posameznega ogrodja. Na koncu smo v zaključku podali sklepne ugotovitve.

2. UPORABLJENE TEHNOLOGIJE

2.1. Razvojno okolje Eclipse

Eclipse [9] je odprtokodno programsko razvojno okolje. Vsebuje osnovni delovni prostor in razširljiv sistem za vtičnike, ki pomagajo pri oblikovanju delovnega okolja. Eclipse program je večinoma napisan v programskem jeziku Java in se uporablja za razvoj javanskih aplikacij. S pomočjo različnih vtičnikov se lahko Eclipse uporablja tudi za razvoj aplikacij v drugih programskih jezikih, kot so: Ada, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Perl, PHP, Python, R, Ruby(vključno z Ruby on Rails ogrodjem), Scala, Clojure, Groovy, Scheme in Erlang. Okolje lahko uporabimo tudi za razvoj paketov za program Mathematica. Razvojno okolje vsebuje Eclipse JDT, ki predstavljajo orodja za programiranje v programskem jeziku Java in Scala, Eclipse CDT za programiranje v C/C++ in Eclipse PDT za razvijanje v programskem jeziku PHP.

V okviru diplomske naloge smo okolje Eclipse uporabili za razvoj avtohtone mobilne aplikacije kot tudi hibridne mobilne aplikacije. Na sliki 1 spodaj je prikazan izgled razvojnega okolja Eclipse.



Slika 1: Razvojno okolje Eclipse.

2.2. Programski jezik Java

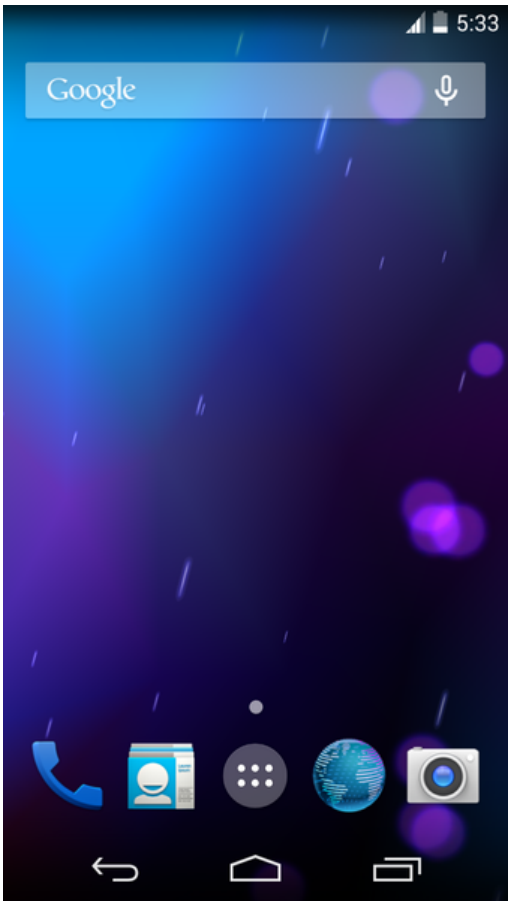
Java [18, 27] je razredno usmerjen in objektno orientiran programski jezik. Zasnovan je z namenom, da ga ni potrebno na različnih sistemih na novo prevajati. Java aplikacije so tipično prevedene v datoteke s končnico `.class`, ki se lahko poganjajo na vseh Java virtualnih strojih neodvisno od arhitekture sistema. Java je eden izmed najbolj popularnih programskih jezikov, predvsem za spletne aplikcije tipa odjemalec-strežnik s približno 9 milijoni razvijalcev.

V okviru diplomskega dela bomo programski jezik Java uporabljali predvsem za razvoj avtohtone mobilne aplikacije.

2.3. Android

Android [7, 10] je operacijski sistem, ki temelji na jedru operacijskega sistema Linux in je primarno zasnovan za mobilne naprave z zaslonom na dotik kot so pametni telefoni in tablični računalniki. Operacijski sistem Android je razvilo podjetje Android, Inc., katerega je Google najprej finančno podprl in kasneje kupil.

Android je odprtokodni operacijski sistem in aplikacije, ki jih operacijski sistem podpira so napisane v programskem jeziku Java ter so lahko nadaljno uporabljene tudi za prodajo na trgovini Google (angl. Google Play Store). Na sliki 2 spodaj je prikazan domači zaslon naprave Android [26].



Slika 2: Domači zaslon na Android napravi.

2.3.1. Android razvoj programske opreme

Android razvoj programske opreme [2] je proces s pomočjo katerega so narejene nove aplikacije za operacijski sistem Android. Aplikacije so običajno razvite na podlagi programskega jezika Java vključno z Android SDK.
Od julija 2013 je bilo razvitih 1 milijon mobilnih aplikacij in prenesenih več kot 25 milijard.

2.3.2. Android SDK

Android SDK [3] je programski razvojni paket, ki vključuje celovit nabor razvojnih orodij. Ta vključujejo razhroščevalnik, knjižnice, emulator, dokumentacijo, vzorce kode ter vodiče.
Uradno podprto integrirano razvojno okolje je Eclipse, ki uporablja Androidova razvojna orodja ADT. Razvijalci lahko razvijajo tudi s pomočjo poljubnega urejevalnika teksta in XML datotek in nato uporabijo orodja ukazne vrstice.
Android aplikacije se zapakirajo v datoteke, za katere je značilna končnica `.apk` in so shranjene v direktoriju `/data/app` Android operacijskega sistema.

2.4. PhoneGap

PhoneGap [3, 4, 21] je mobilno razvojno ogrodje razvito s strani podjetja Nitobi, ki ga je kasneje kupilo podjetje Adobe Systems. Razvijalcem programske opreme omogoča graditi mobilne aplikacije z internetnimi tehnologijami kot so HTML, JavaScript in CSS. Rezultat so hibridne aplikacije, ki niso ne samo spletne aplikacije niti samo avtohtone. Uporabljajo tako spletne tehnologije kot avtohtone programske vmesnike API.
Osnova ogrodja PhoneGap temelji na programski tehnologiji Apache Cordova [8]. Apache Cordova je odprtokodna programska tehnologoja, ki je namenjena razvoju mobilnih aplikacij z uporabo spletnih tehnologij, kot so HTML, JavaScript in CSS. S PhoneGap mobilnim razvojnim ogrodjem lahko razvijemo aplikacije za platforme: Android, iOS, Windows Phone, Symbian, Bada, WebOS in BlackBerry.

2.4.1. HTML

HTML [30, 32] je označevalni jezik za izdelavo spletnih strani in drugih informacij, ki so prikazane v spletnem brskalniku. Jezik HTML je napisan v obliki HTML elementov, ki so sestavljeni iz značk, zapisanih v lomljenih oklepajih (npr. `<h1>`) znotraj vsebine spletne strani. HTML značke so običajno zapisane v parih, kot naprimer `<h1>` in `</h1>`. Prva značka se imenuje začetna značka (ang. *start tag*) in druga značka se imenuje končna značka (ang. *end tag*). Med značke lahko razvijalci dodajajo tekst, več značk, komentarje in druge tipe tekstovno orientirane vsebine.

Namen spletnega brskalnika je branje HTML dokumentov in grajenje vizualnih spletnih strani. Brskalnik ne prikaže HTML značk vendar te značke uporablja za interpretiranje vsebine na strani.

HTML elementi tvorijo gradnike vseh spletnih strani. HTML omogoča uporabo slik in objektov in lahko se uporablja za izdelovanje spletnih obrazcev. Lahko vsebuje skripte napisane v jezikih kot je JavaScript, ki vpliva na interaktivnost spletne strani.

Spletni brskalniki lahko tudi uporabljajo kaskadne stilske podloge CSS, ki definirajo postavitev in pogled tekstovnega in drugega materiala.

2.4.2. JavaScript

JavaScript [28, 20] je dinamični programski jezik. Najbolj je uporabljen kot del spletnih strani, ki omogočajo uporabo skript na strani klienta za interakcijo z uporabnikom, kotrolo nad brskalnikom, asinhronsko komunikacijo in spremembo vsebine, ki je prikazana. Uporabljena je prav tako na strani strežnika in za razvoj video iger ter mobilnih aplikacij.

2.4.2.1. jQuery

jQuery [14] je medplatformna JavaScript knjižnica izdelana za poenostavitev razvoja spletne interakcije strani HTML na strani klienta. jQuery je trenutno najbolj popularna odprtokodna in prostodostopna knjižnica JavaScript programskega jezika. Sintaksa jQuery je oblikovana tako, da omogoča preprostejšo navigacijo dokumenta, selekcijo elementov DOM, izdelovanje animacij, obravnavanje dogodkov (ang. *events*) in razvoj AJAX. Prav tako jQuery omogoča izdelovanje vtičnikov (ang. *plugin*).

2.4.2.1.1. jQuery Mobile

jQuery Mobile [15] je na dotik optimizirano spletno ogrodje (znano tudi kot JavaScript knjižnica ali mobilno ogrodje). Trenutno se razvija s strani projektne skupine jQuery. Razvoj se osredotoča na izdelovanje ogrodja, ki je kompatibilno s širšim naborom pametnih telefonov in tabličnih računalnikov. Izdelano je bilo predvsem zaradi vse večjega trga pametnih telefonov in tabličnih računalnikov. Ogrodje jQuery Mobile je kompatibilno z drugimi mobilmi ogrodji za razvoj mobilnih aplikacij kot so PhoneGap in Worklight.

2.4.3. CSS

Kaskadne stilske podloge [29] pomeni stilovni programski jezik, ki se uporablja za opis izgleda in formata dokumenta, napisanega v označevalnem jeziku. Poleg tega, da je programski jezik najbolj uporabljan za oblikovanje spletnih strani in vmesnikov napisanih v HTML in XHTML je lahko uporabljen tudi za dokumente XML.
CSS je primarno izdelan, da omogoča delitev vsebine dokumenta od predstavitve dokumenta vključujoč postavitev, barve in pisave.

3. MEDPLATFORMSKI RAZVOJ APLIKACIJE

Razvili bomo dve enaki aplikaciji, ki bosta imeli iste funkcionalnosti, vendar bo razvoj potekal z uporabo različnih ogrodij za razvoj aplikacij. Vsa potrebna orodja za razvoj aplikacije smo namestili pred samim razvojem. Ta so opisana v poglavju 2. Ker imamo že nekaj izkušenj s programskim jezikom Java, ki je jedro operacijskega sistema Android, smo se odločili za razvoj Android aplikacije. Ker pa to znanje želimo razširiti, smo se za razvoj druge aplikacije odločili uporabiti dodatno ogrodje PhoneGap, ki pa uporablja spletne programske jezike, kot je Javascript, HTML in CSS.

Idejo za aplikacijo smo dobili, ko smo želeli kar najhitreje dostopati do željenih datotek, kot so fotografije ali določeni dokumenti. Aplikacijo te vrste smo se odločili razviti zato, ker aplikacija te vrste uporablja eno izmed najbolj pomembnih funkcij naprave, delo z datotekami. Na podlagi tega lahko podrobno opišemo primerjavo med razvojnima okoljema, ki smo ju uporabili. Aplikacija pregledovalnik datotek je mobilna aplikacija, ki zagotavlja uporabniški vmesnik za lažji pregled nad datotečnim sistemom. Namen aplikacije je urejen pregled map in datotek. Služi kot pomoč pri pregledovanju datotek, ki se nahajajo v pomnilniku naprave in so predhodno dosegljive le preko integrirane aplikacije Moje datoteke v sistemu Android.

3.3. Aplikacija

3.3.1. Uporabniške zahteve

Razviti želimo mobilno aplikacijo, ki vsebuje podobne funkcionalnosti, kot jih ponuja vsaka aplikacija tipa upravitelj datotek. Avtohtono aplikacijo bo možno poganjati le na mobilnih napravah z operacijskim sistemom Android, hibridno pa na večini ostalih mobilnih napravah. Upoštevane naj bodo vse lastnosti mobilnih naprav in v odvisnosti od tega naj bodo funkcionalnosti temu prilagojene. Aplikacija bo nameščena lokalno. Zaradi samega namena aplikacije, le ta ne bo uporabljala baze podatkov. Podatki, ki so potrebni za pravilno delovanje aplikacije se nahajajo v pomnilniku naprave ali pa na zunanjem pomnilniku naprave. Ker bo aplikacijo uporabljala le ena oseba, ki jo bo naložila na svojo napravo ne bomo potrebovali kakršnekoli registracije. Ker bo aplikacija nameščena lokalno ne potrebuje povezave z internetom. Mobilna aplikacija z imenon brskalnik datotek naj omogoča brskanje po datotečnem sistemu naprave, zagotavlja naj dostop do slik, glasbe, video posnetkov, dokumentov in drugih datotek na Android napravi. Uporabnik lahko dostopa do vseh datotek na napravi. Aplikacija naj omogoča tudi enostavno premikanje po direktorijih v povratni

smeri. Prostor v spodnjem delu aplikacije naj bo namenjen prikazu zgodovine direktorijev, ki jih je uporabnik odprl. Uporabniški vmesnik naj omogoča preprost pregled nad funkcionalnostimi , ki jih aplikacija nudi ter prijaznost do uporabnika in preprostost uporabe. Pri napačni uporabi funkcionalnosti aplikacije naj aplikacija uporabnika primerno obvesti z obvestili oziroma opozorilnimi okni.

3.3.2. Funkcionalne in nefunkcionalne zahteve aplikacije

Na podlagi opisanih želenih funkcionalnosti [1, 19] in zahtev smo naredili seznam specifikacij zahtev mobilne aplikacije:

- avtohtona aplikacija se bo uporabljala na mobilnih napravah z nameščenim operacijskim sistemom Android, hibridna pa lahko tudi na iPhone, Windows Phone, BlackBerry 10 in PlayBook OS, Bada, Symbian, webOS, Tizen, Ubuntu Touch ter Firefox OS. Aplikacija je namenjena uporabi na mobilnih telefonih, prav tako pa bodo lahko aplikacijo uporabljali tudi uporabniki tabličnih računalnikov;
- aplikacija naj omogoča enostavno in preprosto uporabo;
- aplikacija bo nameščena lokalno, kar pomeni, da ne bo uporabljala dostopa do interneta;
- uporabnik bo lahko aplikacijo uporabljal takoj po namestitvi, brez potrebnega kakršnegakoli prijavljanja, registriranja ali podpisovanja;
- podatki, ki jih bo aplikacija potrebovala so shranjeni v pomnilniku naprave ali pa na zunanjem pomnilniku naprave;
- s pritiskom na tipko "`back`" bomo prikazali opozorilno okno, ki uporabnika vpraša, če želi zapreti aplikacijo;
- s pritiskom na tipko "`menu`" bomo lahko izbirali med dvemi možnostimi, prva možnost so informacije o avtorju in druga možnost je izhod iz aplikacije;
- aplikacijo bodo lahko poganjale naprave z nameščenima operacijskim sistemom Android od verzije 2.2 (Froyo) dalje;
- uporabniški vmesnik naj bo enostaven in dobro pregleden;
- aplikacijo sestavljajo tri elementi, glava, telo in noge;
- glava avtohtone aplikacije bo razdeljena na tri dele. Na levi strani imamo gumb "`Domov`", ki bo ob pritisku uporabnika prikazal domači direktorij oziroma root direktorij. Srednji del glave je namenjen prikazu pomnilnika, iz katerega beremo podatke. Tukaj bomo prikazali vse pomnilnike, ki so dosegljivi na napravi. Desni del glave bo namenjen gumbu, ki bo zaprl trenutni direktorij in prikazal starševski direktorij. Glava hibridne aplikacije je nekoliko drugačna, saj imamo na levi strani gumb z imenom "`Nazaj`", ki ob pritisku prikaže starševski direktorij, srednji del

glave predstavlja ime aplikacije in desni del glave predstavlja gumb "Domov", ki ima enako funkcionalnost kot avtohtona aplikacija;

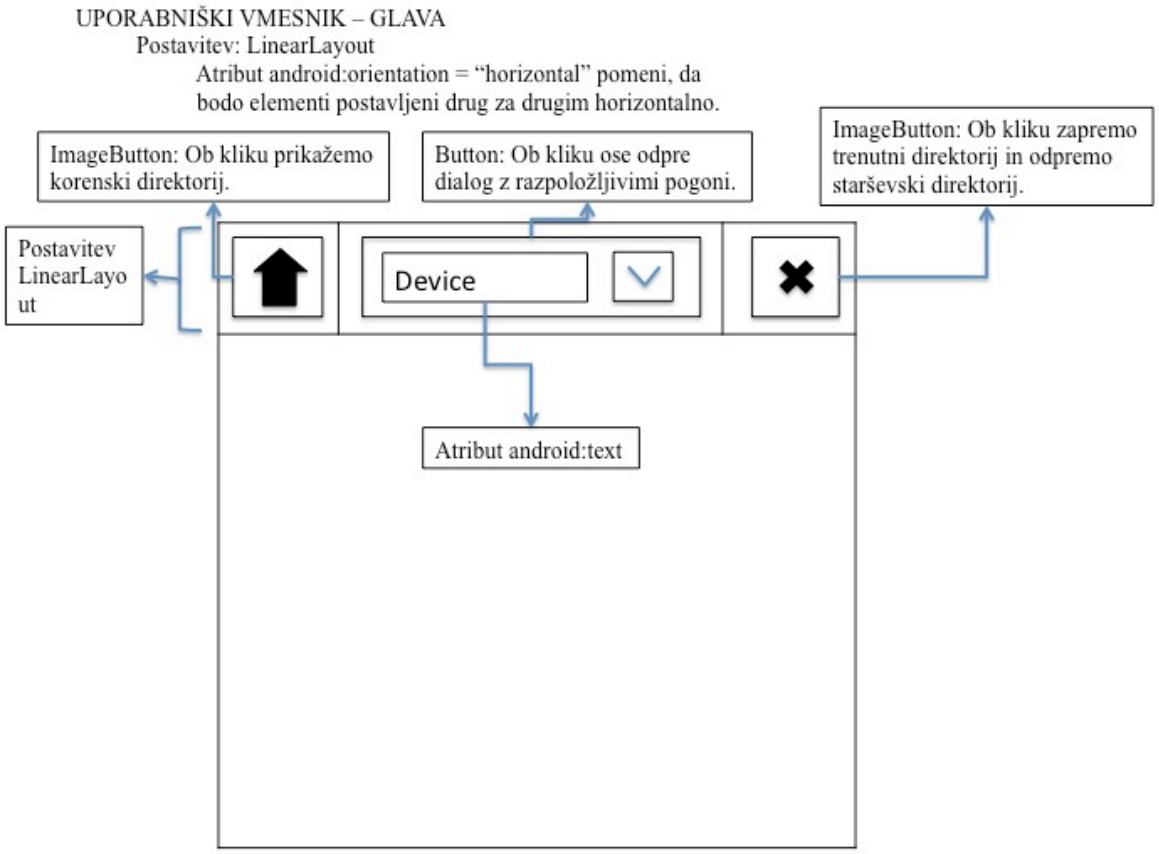
- telo aplikacije bo namenjeno prikazovanju datotek v trenutnem direktoriju. Vsaka datoteka bo v svoji vrstici. Vrstice bomo prikazovali dinamično. Vsaka vrstica bo sestavljena iz ikone, ki predstavlja tip datoteke in bo prikazana levo ter imena datoteke, ki bo zasedalo večji del prostora na desno;
- noga aplikacije je namenjena prikazovanju starševskega in trenutnega direktorija v katerem se nahajamo. Imena direktorijev bodo prikazana na dveh gumbih. Ob pritisku na gumb, ki predstavlja starševski direktorij bomo izbrani direktorij tudi odprli;
- uporabniški vmesnik bomo z obema okoljema (avtohtono in hibridno) poskušali narediti podobnega. To velja za vse aktivnosti (avtohtona aplikacija) in vsak omrežni izgled (hibridna aplikacija);
- v primeru, da pride do kakršnekoli napake, kar pomeni pri uporabi funkcionalnosti aplikacije, bomo uporabnika o tem obvestili;

3.4. Razvoj aplikacije z avtohtonim ogrodjem

Večina Android aplikacij neizogibno potrebuje neko obliko uporabniškega vmesnika. V tem poglavju bomo predstavili vse elemente uporabniškega vmesnika, ki smo jih uporabili za razvoj mobilne aplikacije. Uporabniški vmesnik bomo predstavili v treh delih, glava, telo in noga. Ikone, ki jih aplikacija vsebuje, smo pridobili s spleta [13].

3.4.1. Glava uporabniškega vmesnika

Glava uporabniškega vmesnika je razdeljena na tri dele. Skrajno levo se nahaja gumb, ki predstavlja gumb "Domov", ki uporabnika preusmeri na korenski direktorij. Srednji del glave predstavlja gumb, ki nam privzeto prikazuje ime korenskega direktorija naprave. Desni del glave predstavlja gumb, ki zapre trenutni direktorij. Na sliki 3 lahko vidimo skico, ki predstavlja glavo uporabniškega vmesnika. V datotekah XML [1], ki so namenjene gradnji uporabniškega vmesnika uporabimo `dp` [22] mersko enoto, ki naredi našo aplikacijo kompatibilno z različnimi gostotami in resolucijami zaslonov.



Slika 3: Skica glave uporabniškega vmesnika aplikacije.

Za gradnjo glave uporabniškega vmesnika smo uporabili postavitev `LinearLayout` [1]. Znotraj te postavitve smo uporabili dodatno postavitev `LinearLayout`, ki predstavlja glavo uporabniškega vmesnika in znotraj te postavitve smo uporabili naslednje gradnike: `ImageButton` in `Button`. Uporabili smo dva elementa `ImageButton`, en predstavlja gumb na levi strani uporabniškega vmesnika, drugi na desni strani uporabniškega vmesnika. Gradnik `Button` pa uporabimo na sredini glave uporabniškega vmesnika.

```
<ImageButton
 android:id="@+id/button_home"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_weight="0.18"
 android:src="@drawable/home_icon"
 android:background="@android:drawable/btn_default"
 android:onClick="openRootDirectory" />
```

Zgornji izsek vsebuje gradnik `ImageButton`. Uporabili smo ga za preusmeritev uporabnika na korenski direktorij. Ta podatek je statičen in se ne spreminja. Akcija oziroma dogodek, ki uporabnika preusmeri bo potekalo v programski kodi (kasneje bomo opisali, kako poteka proces preusmeritve) in zato potrebujemo atribut `android:id`. S pomočjo tega atrubuta

lahko v programski kodi enostavno ta element tudi poiščemo in uporabimo. V nadaljevanju opazimo atribut `android:layout_width`, ki predstavlja širino elementa. V tem primeru mu nastavimo vrednost `0dp`, to pa zato, da lahko uporabimo procente. Ko nastavimo atribut `android:layout_weight` na določen procent to pomeni, da se bo sedaj širina prilagajala vsem napravam na katerih bomo poganjali aplikacijo. Atribut `android:layout_height` predstavlja višino elementa, ki mu v tem primeru nastavimo vrednost "`wrap_content`", kar pomeni, da bo višina elementa enaka vsebini elementa. Tukaj je potrebno omeniti še atribut `android:src`, ki elementu doda sliko, v našem primeru ikono. To tudi pojasni samo ime elementa. Predstavimo naj še atribut `android:background`, ki elementu doda privzeto ozadje. Ta atribut uporabimo zaradi tega, ker ko starševskemu elementu `LinearLayout` dodamo ozadje, to vpliva na vse otroške elemente in posledično vsi otroški elementi prevzamejo (ne v celoti) ozadje starševskega elementa. Nazadnje uporabimo še atribut `android:onClick`, ki predstavlja referenco s programsko kodo. To pomeni, da tekst, ki se nahaja med dvema navednicama lahko uporabimo kot metodo v javanski kodi. V našem primeru imamo tekst "`openRootDirectory`" in v javanski kodi nato definiramo metodo `openRootDirectory()`.

```
public void openRootDirectory(View view){

        //kreiramo novo datoteko, ki predstavlja koreninski direktorij
        File rootFileHomeButton = rootFile;

        //posodobimo ime sredinskega gumba
        final Button drivesButton = (Button) findViewById(R.id.buttonDrive);
        drivesButton.setText("Device");

        //spremenljivka values predstavlja vsebino oziroma
        //seznam, ki jo prikazujemo v telesu aplikacije
        values = rootFileHomeButton.list();

        //posodobimo trenutno pot in trenutno ime direktorija
        currentPath = rootFileHomeButton.getAbsolutePath();
        currentDirName = "/";

        //prikažemo obvestilo, da se uporabnik nahaja na koreninskem direktoriju
        Toast.makeText(getBaseContext(), "Root directory!",
        Toast.LENGTH_SHORT).show();

        //pokličemo metodo, ki nam prikaže vsebino želenega direktorija
        onCreateHelper(rootFileHomeButton.getAbsolutePath());

}
```

Ob kliku na levi gumb v glavi uporabniškega vmesnika uporabnika preusmerimo na korenski direktorij.

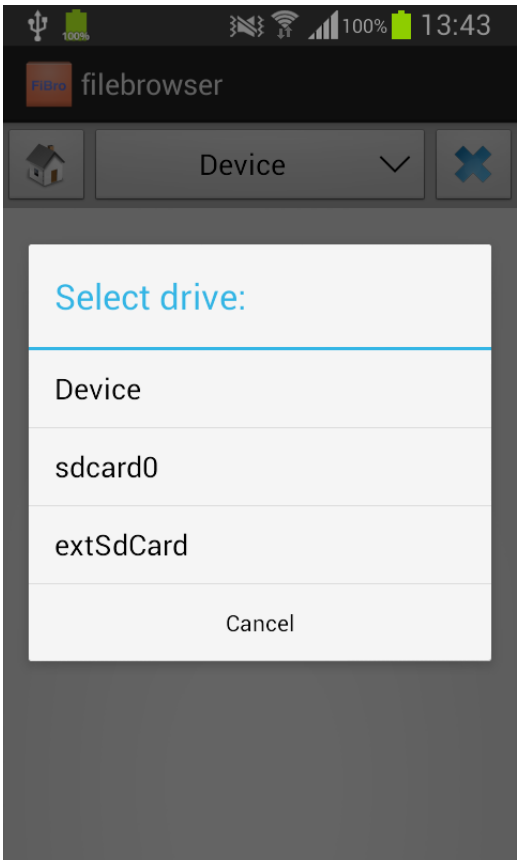
```
<Button
 android:id="@+id/buttonDrive"
 android:layout_width="0dp"
 android:layout_height="50dp"
 android:layout_weight="0.64"
 android:text="@string/button_header_middle"
 android:drawableRight="@drawable/dialog_down_arrow_icon_32"
 android:onClick="openDriveDialog"
 android:background="@android:drawable/btn_default" />
```

Zgornji izsek vsebuje gradnik `Button`. Uporabili smo ga za prikaz rezpoložljivih direktorijev v napravi. Atributi `android:layout_width`, `android:layout_height` in `android:layout_weight` predstavljajo dimenzije elementa, ki smo jih predhodno že omenili. Atribut `android:text` predstavlja privzeti tekst, ki bo prikazan na gumbu. Dejanski tekst preberemo iz datoteke XML. Imenuje se `Strings.xml` in tukaj so definirani vsi elementi tipa `String`. Sledi atribut `android:drawableRight`, ki določeno ikono izriše na desno stran gumba. V našem primeru ikono, ki predstavlja puščico navzdol.

```
public void openDriveDialog(View view){

	. . .

	final Button drivesButton = (Button) findViewById(R.id.buttonDrive);

	//kreiramo nov dialog
	AlertDialog.Builder alertDialog = new
		AlertDialog.Builder(MainActivity.this);

	//dialogu določimo naslov
	alertDialog.setTitle("Select drive:");

	//metoda sprejme tabelo elementov, ki jih nato prikažemo v dialogu
	alertDialog.setItems(drivesArray, new DialogInterface.OnClickListener()
	{
		//ob kliku na element dialoga, metoda pridobi pozicijo dialoga
		@Override
		public void onClick(DialogInterface dialog, int which) {

			//dialogu nastavimo tekst
			drivesButton.setText(drivesArrayFinal[which]);

		}
	});

	//za razrešitev dialoga potrebujemo negativni gumb, ki mu določimo ime
	alertDialog.setNegativeButton("Cancel", new
		DialogInterface.OnClickListener() {

		@Override
		public void onClick(DialogInterface dialog, int which) {

		}
	});

	//dialog prikažemo
	alertDialog.show();

}
```

Ob kliku na sredinski gumb se odpre okno, ki je objekt razreda `AlertDialog`. Dialog se pojavi ob kliku na sredinski gumb glave uporabniškega vmesnika, zato moramo dialog implementirati znotraj poslušalcev dogodkov. Tukaj predstavimo dialog, ki uporabniku omogoča lažji pregled in premikanje po direktorijski strukturi. Ob kliku na katerikoli pogon naprave, nam vsebino pogona aplikacija prikaže v telesu uporabniškega vmesnika.

Na desni strani glave uporabniškega vmesnika se nahaja gumb, ki pa zapre trenutni direktorij v katerem se uporabnik nahaja in prikaže starševski direktorij. Ima podobno strukturo kot gumb na levi strani glave uporabniškega vmesnika.



Slika 4: Razpoložljivi pogoni v napravi.

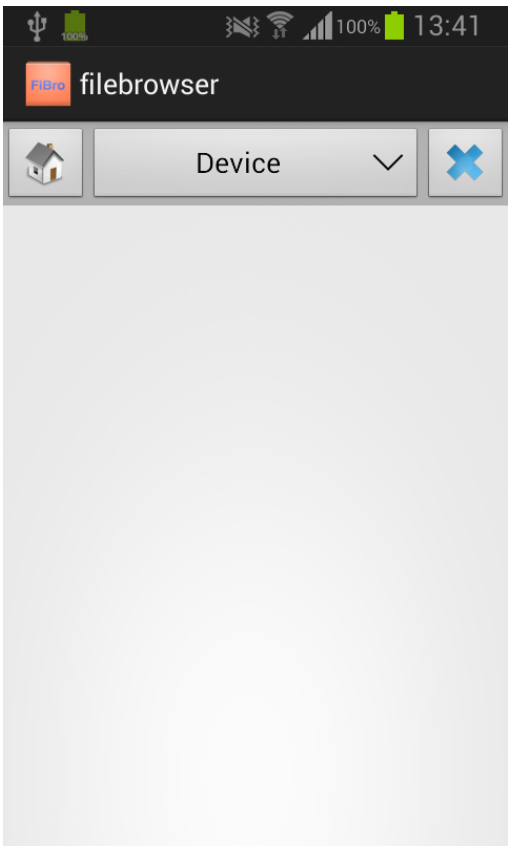
Zgornja slika 4 prikazuje okno, ki predstavlja element `dialog` z razpoložljivimi pogoni v napravi.

```
<ImageButton
 android:id="@+id/button_close"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_weight="0.18"
 android:src="@drawable/close_icon48"
 android:background="@android:drawable/btn_default"
 android:onClick="closeCurrentDirectory"/>
```

Zgornji izsek predstavlja gradnik `ImageButton`. Uporabili smo ga za zapiranje trenutnega direktorija oziroma odpiranje starševskega direktorija. Vse atribute, ki jih gradnik implementira smo predhodno že omenili.

```
public void closeCurrentDirectory(View view){

	//currPath predstavlja trenutno pot
	String currPath = currentPath;

	//currDirName predstavlja ime trenutnega direktorija
	String currDirName = currentDirName;

	//trenutno pot razcepimo na različna imena direktorijev
	String[] splitPath = currPath.split("/");

	//preveri, ce smo na nivoju root direktorija
	if(splitPath.length != 0){
		//preveri, ce smo na nivoju sdCard0
		if(splitPath[splitPath.length - 1].equals("sdcard0")){

			. . .

		//preveri, ce je na nivoju extSdCard
		}else if(splitPath[splitPath.length - 1].equals("extSdCard") ||
		splitPath[splitPath.length - 1].equals("external_sd")){

			. . .

		//prikaz obvestila, da se že nahajamo na koreninskem direktoriju
		}else if(currDirName.equals("/")){

			. . .

		}else{

			//zgradimo pot, ki predstavlja starša
			String parentPath = buildPath(splitPath);

			//kreiramo starševsko datoteko
			File parentFile = new File(parentPath);

			//pridobimo ime starševskega direktorija
			String parentDirName = splitPath[splitPath.length - 2];

			//posodobimo trenutno pot in trenutno ime
			currentPath = parentPath;
			currentDirName = parentDirName;

			//posodobimo seznam z vsebino, ki jo nato prikažemo
			values = parentFile.list();

			//prikažemo želeni direktorij
			onCreateHelper(parentPath);
		}
	}else{

		Toast.makeText(getBaseContext(), "Root directory!",
		Toast.LENGTH_SHORT).show();

	}
}
```

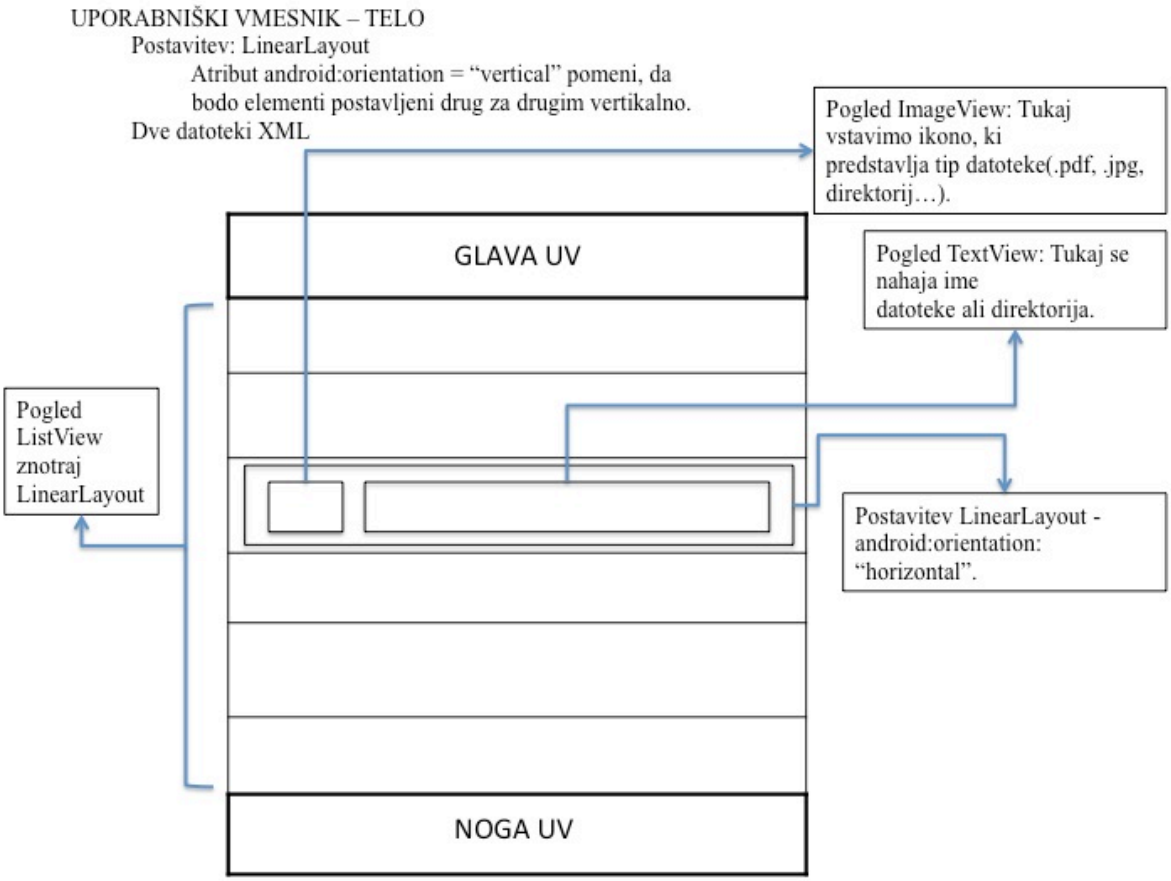
Zgornji izsek programske kode se izvede ob kliku na desni gumb v glavi uporabniškega vmesnika aplikacije. Na sliki 5 spodaj je prikazan dokončan uporabniški vmesnik, ki predstavlja glavo mobilne aplikacije.



Slika 5: Dokončana glava uporabniškega vmesnika.

3.4.2. Telo uporabniškega vmesnika

V telesu uporabniškega vmesnika aplikacije prikazujemo vsebino želenega direktorija. Pri gradnji telesa aplikacije smo uporabili postavitev `LinearLayout` in znotraj te pogled `ListView` [1, 2, 25, 11, 5]. Na sliki 6 je predstavljena skica telesa uporabniškega vmesnika.



Slika 6: Skica telesa uporabniškega vmesnika aplikacije.

```
<LinearLayout
 android:layout_width="match_parent"
 android:layout_height="0dp"
 android:layout_weight="0.50"
 android:orientation="vertical">

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

</LinearLayout>
```

Zgornji odsek programske kode predstavlja postavitev `LinearLayout`, ki vsebuje pogled `ListView`. `ListView` prikazuje seznam vertikalno premikajočih elementov. Elementi seznama so samodejno vstavljeni v seznam z uporabo adapterja, ki navadno pridobi podatke iz tabele ali podatkovne baze in pretvori vsak pridobljen podatek v pogled, ki je nato vstavljen v seznam (angl. *list*).

```
<ImageView
 android:id="@+id/imgIcon"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"/>

<TextView
 android:id="@+id/txtTitle"
 android:layout_width="300dp"
 android:layout_height="fill_parent"
 android:gravity="center_vertical"
 android:paddingLeft="10dp"
 android:textSize="18sp"
 android:textStyle="bold"/>
```

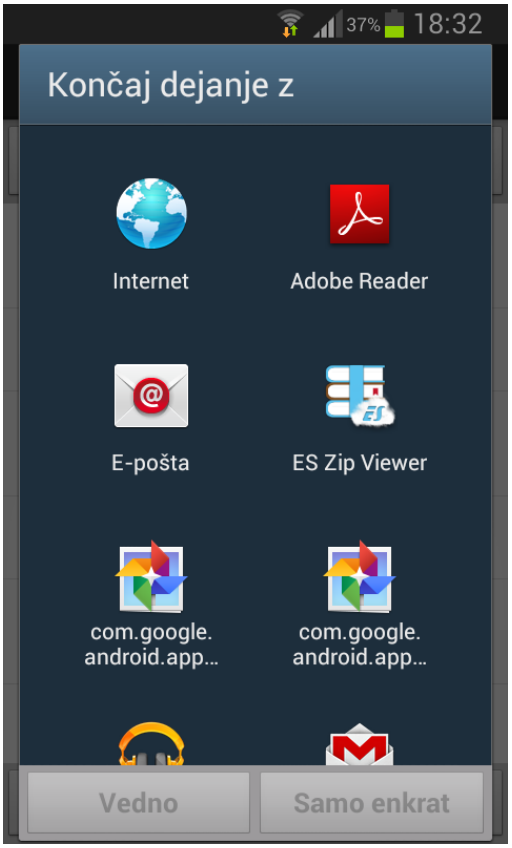
Zgornji odsek predstavlja dva pogleda uporabniškega vmesnika. Skupaj predstavljata eno vrstico v telesu aplikacije. Prvi pogled `ImageView` predstavlja prostor, kjer bo postavljena ikona, ki predstavlja tip datoteke, drugi pogled `TextView` pa predstavlja prostor, kjer bo postavljen tekst, ki predstavlja ime posameznega direktorija oziroma datoteke.

```
public class FileArrayAdapter extends ArrayAdapter<String> {

	Context context;
	int resource;
	String[] data;

	/*Konstruktor, ki sprejme 3 parametre:
	1. parameter: Context - Aktivnost, v kateri bomo uporabili listview
	2. parameter: id vira
	3. parameter: vsebina direktorija, ki ga želimo prikazati
	*/
	public FileArrayAdapter(Context context, int resource, String[] data) {
		super(context, resource, data);
		this.context = context;
		this.resource = resource;
		this.data = data;
	}

	//metoda getView() poveže XML postavitev z JAVA objekti
	@Override
	public View getView(int position, View convertView, ViewGroup parent) {

		//layout naredi povezavo med XML elementi in Java View objektom
		LayoutInflater inflater =
		(LayoutInflater)context.getSystemService(
			Context.LAYOUT_INFLATER_SERVICE);

		//dejanska povezava med xml datoteko in java objektom
		View rowView = inflater.inflate(R.layout.listview_item_row,
			parent, false);

		//kreiranje javanskih objektov
		ImageView imageView =
			(ImageView)rowView.findViewById(R.id.imgIcon);
		TextView textView =
			(TextView)rowView.findViewById(R.id.txtTitle);

		//posameznemu elementu določimo tekst
		textView.setText(data[position]);

		//glede na končnico imena datoteke spremenimo sliko
		String text = data[position];
		if(text.contains(".pdf")){
			imageView.setImageResource(R.drawable.pdf_icon_48);
		}else if(!text.contains(".")){
			imageView.setImageResource(R.drawable.folder_icon_48);

		. . .

		}else{
			imageView.setImageResource(R.drawable.ini_file_icon48);
		}

		return rowView;
	}

}
```

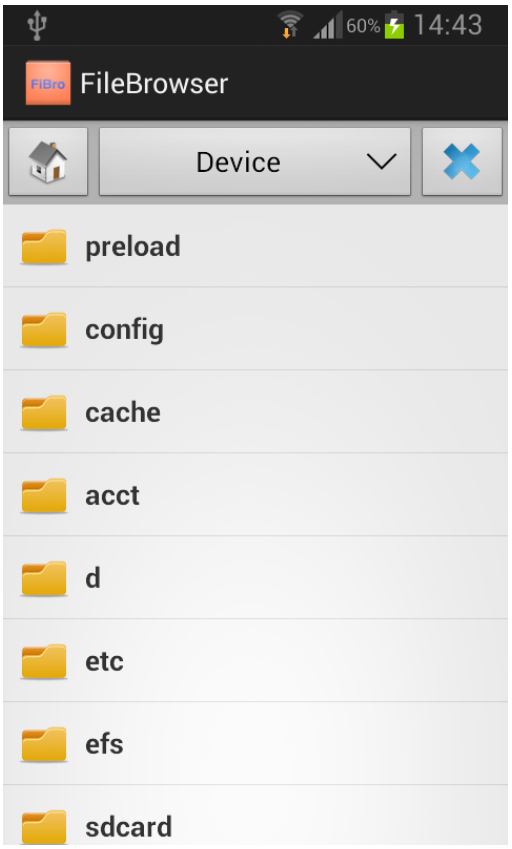
Zgornji izsek predstavlja javanski razred `FileArrayAdapter`. `FileArrayAdapter` razširja razred `ArrayAdapter`. `ArrayAdapter` je razred, ki je podprt s tabelo poljubnih elementov.

```
public void onCreateHelper(String path){

. . .

//Kreiramo instanco razreda FileArrayAdapter
FileArrayAdapter adapter = new FileArrayAdapter(this,
       R.layout.listview_item_row, values);

//pogledu ListView nastavimo adapter
listview.setAdapter(adapter);

//pogledu nastavimo poslušalec klika
listview.setOnItemClickListener(new AdapterView.OnItemClickListener() {

       //za vsak klik preverimo, če je željena datoteka direktorij ali ne
       @Override
       public void onItemClick(AdapterView<?> parent, final View view,
              int position, long id) {

              . . .

              File clickedFile = new File(newPath);
              if(!item.contains(".")){

                     //preveri, če je datoteka prazna
                     if(clickedFile.list() != null ){

                            //preveri, če je datoteka direktorij
                            if(clickedFile.isDirectory()){

                                   //posodobimo vrednosti
                                   values = clickedFile.list();
                                   //rekurzivni klic
                                   onCreateHelper(newPath);
                            }
                     }else{

                               . . .

                     }
              }else if(item.contains(".pdf")){
                     //s pomočjo elementa intent odpremo določen tip datotek
                     Intent intent = new Intent(Intent.ACTION_VIEW);
                     intent.setData(Uri.fromFile(clickedFile));
                     startActivity(intent);
              }else{
                     //drugih primerih uporabnik izbere program za odpiranje
                     Intent intent = new Intent(Intent.ACTION_VIEW);
                     intent.setDataAndType(Uri.fromFile(clickedFile),
                            "application/*");
                     startActivity(intent);
              }
       }
  });
}
```

V zgornjem odseku predstavimo metodo v kateri kreiramo instanco razreda `FileArrayAdapter`, v katerem povežemo podatke, ki jih želimo predstaviti v jedru aplikacije s pripadajočimi pogledi. Vsak element tabele povežemo s pripadajočim pogledom `ListView` in nato to povežemo še s pogledom `ListView`, ki nam dejansko prikaže vse elemente seznama. Ob kliku na datoteko preverimo, če je le-ta direktorij. Če je direktorij, ta direktorij odpremo, če ni pa imamo dve možnosti za odpiranje datotek. Prvo preverimo, če se ime datoteke konča s `.pdf`. Če je pogoj pravilen uporabimo objekt `Intent` [23, 4] in odpremo datoteko PDF s pripadajočim programom Adobe Reader. Če je pogoj napačen pomeni, da datoteka ni tipa PDF in ob kliku na to vrsto datoteke se odpre dialog, kjer izberemo aplikacijo za odpiranje te vrste datoteke (slika 7) [6]. Na sliki 8 prikažemo dokončano telo uporabniškega vmesnika aplikacije.



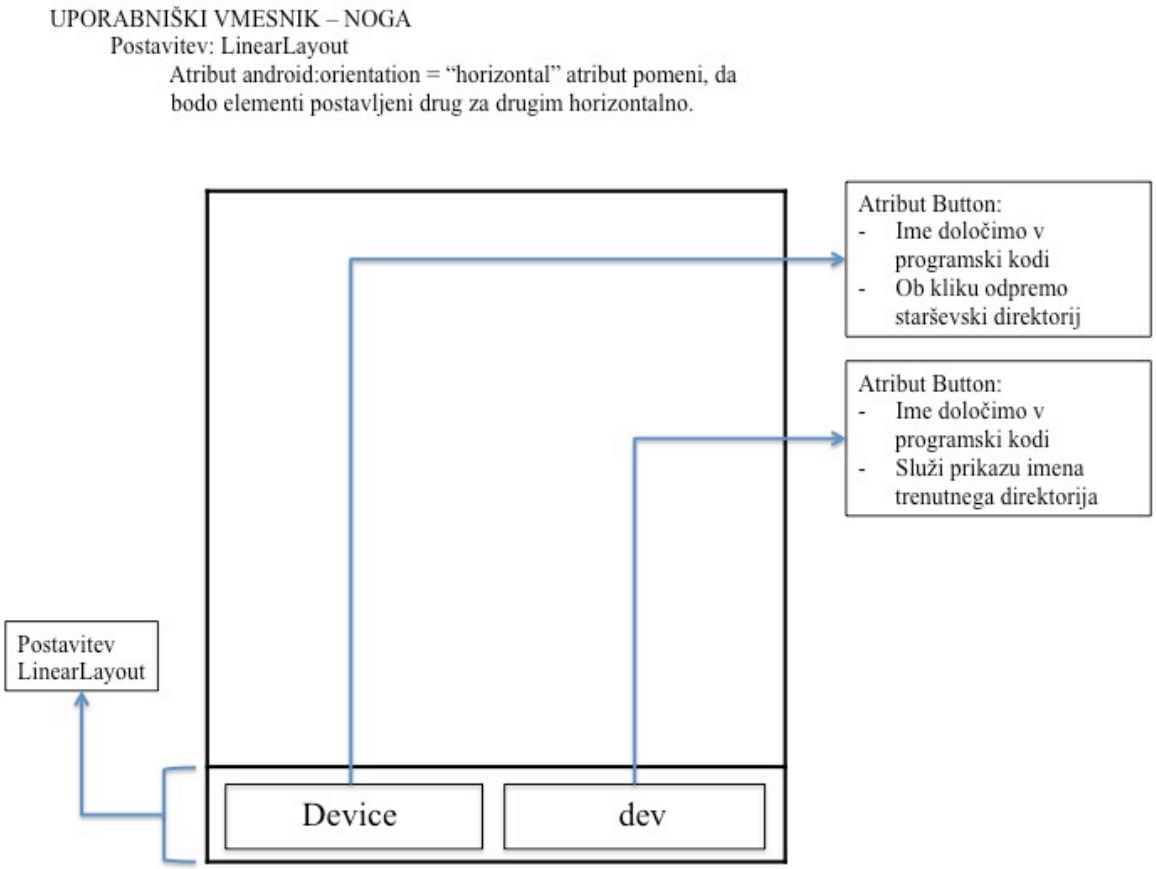
Slika 7: Možnosti nativne aplikacije za odpiranje željene datoteke.



Slika 8: Dokončano telo uporabniškega vmesnika.

3.4.3. Noga uporabniškega vmesnika

V nogi uporabniškega vmesnika aplikacije se znotraj postavitve `LinearLayout` nahajata dva gumba, ki sta postavljena horizontalno. Na sliki 9 lahko opazimo skico noge uporabniškega vmesnika aplikacije.



Slika 9: Skica noge uporabniškega vmesnika aplikacije.

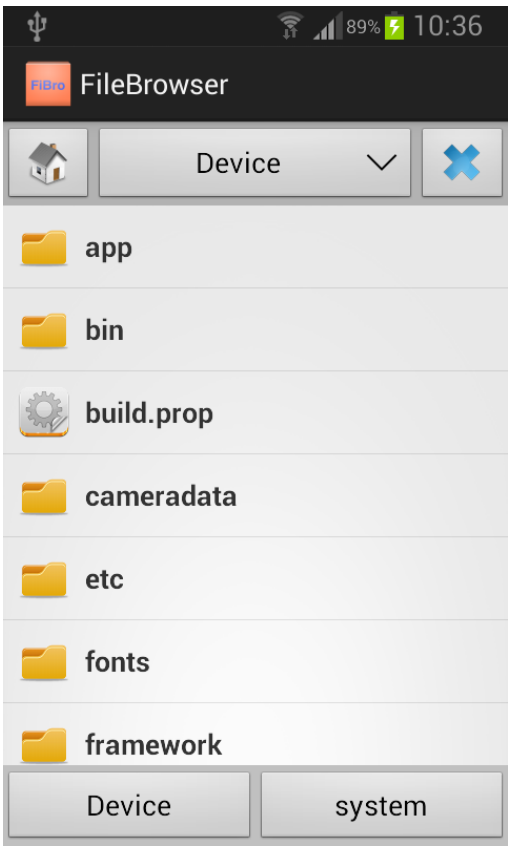
```
<LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal"
 android:background="#C0C0C0"
 android:paddingTop="3dp" >

 <Button
  android:id="@+id/buttonParentDir"
  android:layout_width="0dp"
  android:layout_height="wrap_content"
  android:layout_weight="0.50"
  android:background="@android:drawable/btn_default"
  android:onClick="closeCurrentDirectory" />

 <Button
  android:id="@+id/buttonCurrentDir"
  android:layout_width="0dp"
  android:layout_height="wrap_content"
  android:layout_weight="0.50"
  android:background="@android:drawable/btn_default"/>
</LinearLayout>
```

Zgornji odsek kode predstavlja dva gumba, ki se nahajata znotraj postavitve `LinearLayout`. Levi gumb prikazuje ime starševskega direktorija, desni gumb pa ime trenutnega direktorija v

katerem se nahajamo. Ob kliku na gumb, ki predstavlja starševski direktorij odpremo ta direktorij. Če se nahajamo na korenskem direktoriju naprave desnega gumba ne prikažemo. V tem primeru ime trenutnega direktorija predstavlja levi gumb.

```
Button b1 = (Button)findViewById(R.id.buttonParentDir);
Button b2 = (Button)findViewById(R.id.buttonCurrentDir);
b1.setText("Device");
b2.setVisibility(View.INVISIBLE);
```

V zgornjem odseku programske kode s pomočjo metode `findViewById()` poiščemo željeni gumb, ki mu nato nastavimo tekst, ki predstavlja ime gumba. Gumbu lahko nastavimo tudi vidnost in sicer s pomočjo metode `setVisibility()`. Končen izgled aplikacije je prikazan na sliki 10.



Slika 10: Končni izgled nativne aplikacije.

3.5. Razvoj aplikacije s hibridnim ogrodjem Phonegap

V tem delu bomo predstavili vse dele, ki sestavljajo aplikacijo. To so datoteke HTML, JavaScript in CSS. Implementacijo razvoja aplikacije z ogrodjem PhoneGap bomo opisali

tako, da bomo sprva predstavili uporabniški vmesnik aplikacije in nato za vsak posamezni del uporabniškega vmesnika predstavili še pripadajočo programsko kodo.

3.5.1. Glava uporabniškega vmesnika

Glava uporabniškega vmesnika aplikacije je sestavljena iz dveh povezav in imena aplikacije. Na levi strani se nahaja povezava na starševski direktorij, vendar zaradi pripadajočih programskih knjižnic uporabnika preusmeri na korenski direktorij. Na sredini glave uporabniškega vmesnika se nahaja ime aplikacije, ki se ne spreminja in na desni strani se nahaja povezava, ki nas preusmeri na domačo stran aplikacije oziroma na vrhnji nivo aplikacije.

```
<div data-role="header" data-position="fixed">
      <a id="backBtn" href="#" data-icon="arrow-l" data-
            theme="d">Nazaj</a>
      <h1>Raziskovalec</h1>
      <a id="homeBtn" href="index.html" data-icon="home" data-
            theme="d">Domov</a>
</div>
```

Zgornji odsek programske kode, ki je napisana v programskem jeziku HTML predstavlja glavo uporabniškega vmesnika aplikacije. Element HTML `div` sestavljajo različni atributi. Tukaj je potrebno omeniti atribute `data-*` [31], ki predstavljajo povezavo z oblikovno datoteko in omogočajo atraktivni pogled za mobilne naprave. Atribute `data-*` uporabljajo oblikovne knjižnice, katere nam poenostavijo izdelovanje uporabniškega vmesnika. V tem delu naj omenimo atribut `data-role='header'`, s pomočjo katerega bo glava aplikacije primerno oblikovana in postavljena na vrh zaslona naprave. Sledi atribut `data-position='fixed'`, ki omogoča, da se glava uporabniškega vmesnika ne spreminja z vsebino. Povezava HTML, ki je postavljena na levi strani glave uporabniškega vmesnika vsebuje atribut `data-icon='arrow-l'`, ki omogoča preoblikovnje klasične povezave HTML v gumb. Pomembno je opisati še atribut `data-theme='d'` [16], ki omogoča celovito preoblikovanje elementa. Za ta atribut lahko nastavimo različne vrednosti, "`a`", "`b`", "`c`", "`d`" in "`e`". Vsaka vrednost predstavlja različno oblikovanje. Sledi element HTML `h1`, ki predstavlja naslov aplikacije, ki se ne spreminja. Element, ki je postavljen na desni strani uporabniškega vmesnika pa predstavlja povezavo.

```
//inicializacija spremenljivk
var backBtn = $('#backBtn');
var homeBtn = $('#homeBtn');

//odziv na klik gumba nazaj
backBtn.click(function(){

	//preverimo, če starševski direktorij obstaja
	if( parentDir != null ){

		//prikažemo vsebino direktorija
		listDir(parentDir);

		//rekurzivni klic trenutne funkcije
		clickItemAction();
	}
});

//odziv na klik gumba domov
homeBtn.click(function(){

	//preverimo, če koreninski direktorij obstaja
	if( root != null ){
		listDir(root);
		clickItemAction();
	}
});
```

V zgornjem delu programske kode predstavimo odzive na klik posameznega gumba glave uporabniškega vmesnika. Na začetku inicializiramo spremenljivki, ki predstavljata gumba. Funkcija `listDir()` prikaže vsebino direktorija, ki ga kot paramater pošljemo v metodo. To metodo bomo podrobneje opisali v nadaljevanju. Potrebno je omeniti še funkcijo `clickItemAction()`. To je funkcija, ki vsebuje zgornji odsek kode in ob klicu funkcije izvedemo rekurzivni klic, ki nam omogoča, da lahko tudi po prikazu vsebine novega direktorija dostopamo do funkcionalnosti gumbov.

3.5.2. Telo uporabniškega vmesnika

Telo uporabniškega vmesnika sestavlja seznam elementov, ki predstavljajo vsebino direktorija. Ob kliku na vsak element seznama preverimo ali je direktorij ali datoteka. Na podlagi tega tudi ustrezno ukrepamo z odpiranjem poddirektorija ali z odpiranjem datoteke.

```
<div data-role="content">
	<ul id="iconlistview" data-role="listview"></ul>
</div>
```

Zgornji odsek programske kode napisane v programskem jeziku HTML predstavlja telo uporabniškega vmesnika aplikacije. Tukaj naj omenimo element `ul`, ki predstavlja seznam, v

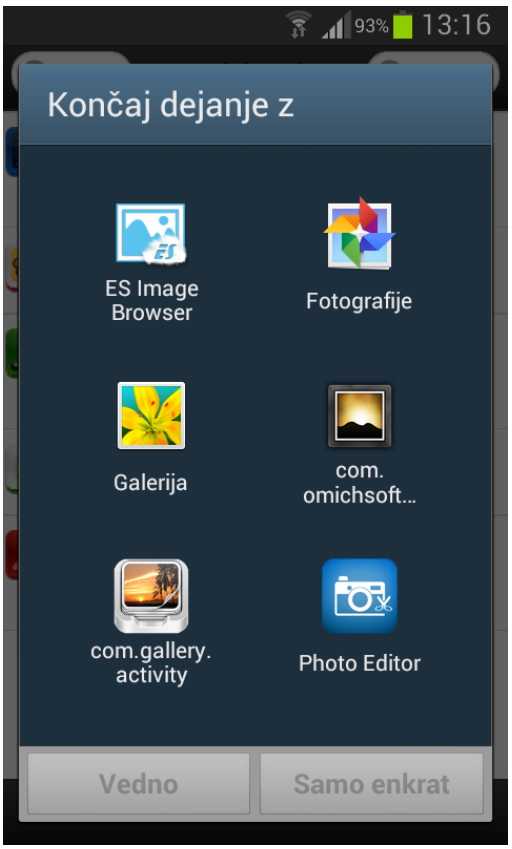
katerega dinamično vstavljamo vsebino. Ta element vsebuje atribut `data-role='listview'` [17], ki omogoča primerno oblikovanje elementov seznama in prilagajanje zaslonu aplikacije.

```
function getFileSystem(){

	//pridobimo datotečni sistem
	window.requestFileSystem(LocalFileSystem.PERSISTENT, 0,
		function(fileSystem){

			//po uspešnem klicu datotečnega sistema nastavimo
			//koreninski direktorij
			root = fileSystem.root;

			//prikažemo vsebino direktorija
			listDir(root);
		}, function(evt){

			//ob neuspešnem klicu napako zapišemo v dnevnik
			console.log("File System Error: "+evt.target.error.code);
		}
	);
}
```

Omeniti je potrebno, da je potrebno prvo pridobiti datotečni sistem, šele nato lahko začnemo po njem raziskovati. To predstavlja zgornji odsek programske kode.

```
function listDir(directoryEntry){

	. . .

	//začnemo z branjem direktorija
	directoryReader.readEntries(function(entries){

		//iteriramo skozi vsebino direktorija
		for(var i=0; i<entries.length; ++i){
			var entry = entries[i];

			//v tabelo direktorijev dodajamo direktorij
			//in v tabelo datotek dodamo datoteke
			if( entry.isDirectory && entry.name[0] != '.'
			) dirArr.push(entry);
			else if( entry.isFile && entry.name[0] != '.'
			) fileArr.push(entry);
		}

		//nova sortirana tabela
		var sortedArr = dirArr.concat(fileArr);

		//iteriramo skozi nov seznam vsebine
		for(var i=0; i<sortedArr.length; ++i){
			var entry = sortedArr[i];

			//preverimo ali je direktorij ali datoteka
			if( entry.isDirectory ){

				stringBuilder += "<li
				class='folder'><div><span><img class='ui-li-
				thumb' src='img/folder_icon_32.png'>
				</img></span><span  class='li-span-namedir'>" +
				entry.name + "</span></div></li>";

			}else if( entry.isFile ){
				var nameOfFile = entry.name;

				//preverimo končnico imena
				if(nameOfFile.indexOf(".pdf") >= 0){

					stringBuilder += "<li
					class='file'><div><span><img class='ui-li-
					thumb' src='img/pdf_icon_32.png'>
					</img></span><span class='li-span-
					namedir'>" + entry.name +
					"</span></div></li>";

				}

		//novo vsebino dodamo seznamu
		dirContent.append(stringBuilder);
		//vsebino posodobimo
		dirContent.listview().listview("refresh");

	. . .
}
```

Zgornji odsek programske kode predstavlja funkcijo `listDir()` s pomočjo katere prikažemo vsebino željenega direktorija. Ob kliku na datoteko direktorija se ob uporabi vtičnika [12] odpre dialog s seznamom aplikacij s katerimi želimo odpreti željeno datoteko (slika 11).



Slika 11: Možnosti hibridne aplikacije za odpiranje željene datoteke.

3.5.3. Noga uporabniškega vmesnika

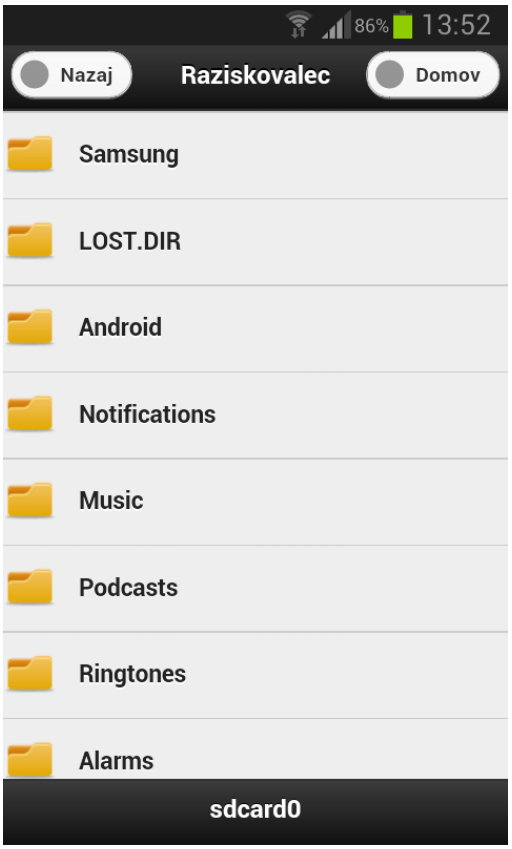
```
<div data-role="footer" data-position="fixed">
	<h1>
		<span id="parDirName"></span>
		<span id="currDirName"></span>
	</h1>
</div>
```

V zgornjem izseku programske kode predstavimo nogo uporabniškega vmesnika aplikacije. V tem delu je potrebno poudariti, da znotraj elementa, ki predstavlja prostor za naslov ustvarimo dva dodatna elementa. Element z atributom `id='parDirName'` predstavlja prostor, ker se bo nahajalo ime starševskega direktorija. Element z atributom `id='currDirName'` pa predstavlja prostor, kjer se bo nahajalo ime trenutnega direktorija. Ker se vsakič, ko aplikacijo zaženemo nahajamo na korenskem direktoriju bomo prikazali le ime tega direktorija. Ob vsakem vstopu v poddirektorij pa na levi strani noge uporabniškega

vmesnika prikažemo starševski direktorij, na desni strani pa prikažemo ime trenutnega direktorija.

```
//trenutno ime direktorija prikažemo v nogi uporabniškega vmesnika desno
$("#currDirName").html(currentDir.name);

//ime starševskega direktorija prikažemo v nogi uporabniškega vmesnika levo
$("#parDirName").html(parentDir.name).show();
```

V zgornjem odseku programske kode predstavimo dva dela, ki sta postavljena na različnih delih v programski kodi. Njuno funkcionalnost pa smo predhodno že opisali. Končni izgled hibridne aplikacije je prikazan na sliki 12.



Slika 12: Končni izgled hibridne aplikacije.

4. PRIMERJAVA RAZVOJNIH OGRODIJ

V okviru tega poglavja bomo opisali primerjavo med avtohtonim ogrodjem Android in hibridnim ogrodjem PhoneGap [24]. Primerjavo bomo opisali na podlagi razvoja mobilne aplikacije z obema ogrodjema. V nadaljevanju so opisane primerjave, ki smo jih srečali ob uporabi posameznega ogrodja.

4.1. Primerjava avtohtonega ogrodja Android in hibridnega ogrodja PhoneGap na podlagi razvoja mobilne aplikacije

Ob razvoju mobilnih aplikacij smo uporabili 6 kriterijev za primerjavo med avtohtonim in hibridnim ogrodjem. To so programski jezik, uporabniški vmesnik s katerim predstavimo prednosti in slabosti implementacije in oblikovanja uporabniškega vmesnika aplikacije, orodja posameznega ogrodja s katerimi predstavimo v kakšni meri ogrodje poenostavi razvoj aplikacije, dokumentacija in viri, ki nam olajšajo razumevanje posameznih metod in funkcij posameznega ogrodja, programske knjižnice s pomočjo katerih pohitrimo razvoj aplikacije in namestitev aplikacije na fizično napravo, ki pokaže dejanske razlike med ogrodjema glede delovanja aplikacije na napravi.

4.1.1. Programski jezik

V nadaljevanju opisa tega kriterija so podani programski jeziki, ki jih posamezno ogrodje podpira.

4.1.1.1. Avtohtona aplikacija

- Java
- XML

Programski jezik Java je prosto dostopen in neodvisen od operacijskega sistema.

4.1.1.2. Hibridna aplikacija

- JavaScript
- HTML
- CSS

Nekatere implementacije programskega jezika JavaScript so prosto dostopne. HTML je del W3C standarda, kateri je prav tako prosto dostopen.

4.1.2. Uporabniški vmesnik

Ker je uporabniški vmesnik eden izmed glavnih delov vsake mobilne aplikacije lahko na ta način primerjamo kakšne pristope in možnosti nudi posamezno ogrodje za razvoj uporabniškega vmesnika mobilne aplikacije.

4.1.2.1 Avtohtona aplikacija

Prednosti:

- Razvojna orodja nam nudijo velik nabor možnosti in nastavitev pri oblikovanju uporabniškega vmesnika
- Veliko različnih možnosti oblikovanja uporabniškega vmesnika posameznega elementa v seznamu (ang. *ListView*)
- Na voljo imamo veliko možnosti za implementacijo posameznih elementov uporabniškega vmesnika

Slabosti:

- Pri oblikovanju uporabniškega vmesnika je veliko časa potrebno nameniti razumevanju delovanja
- Velik nabor možnosti predstavlja večji obseg programske kode in težje razumevanje

4.1.2.2. Hibridna aplikacija

Prednosti:

- Sklicevanje na kaskadne stilske podloge je enostavno in pregledno
- Enostavna in logična postavitev vsebine znotraj elementa seznama

Slabosti:

- Ker je veliko elementov že predhodno oblikovanih z različnimi programskimi knjižnicami se pojavi problem preoblikovanja, kjer moramo eksplicitno poudariti pomembnost oblikovne podlage
- Težave pri stilskem preoblikovanju zaradi programskih knjižnic, ki elemente označevalnega jezika predhodno oblikujejo.

4.1.3. Orodja razvojnih ogrodij

Orodja posameznega ogrodja razvijalcu mobilne aplikacije poenostavijo delo s programsko kodo z različnimi pristopi kot je naprimer samoizpolnjevanje programske kode.

4.1.3.1. Avtohtona aplikacija

Prednosti:

- Razvojna orodja nam olajšajo razvoj s samoizpolnjevanjem programske kode in s tem imamo večjo možnost izbire različnih razvojnih funkcij

Slabosti:

- Za pravilno delovanje moramo veliko časa nameniti analizi problema
- Zapletenost programske kode pri prikazovanju imen direktorijev ko se vračamo nazaj po direktorijski strukturi

4.1.3.2. Hibridna aplikacija

Prednosti:

- Enostavna in preprosta implementacija programske kode v označevalnem jeziku
- Sama struktura dokumenta HTML prihrani veliko dela, saj so deli kot so glava, telo in noge predhodno že definirani
- Manjši obseg programske kode nam omogoča boljši pregled in lažje urejanje
- Enostavno dodajanje elementov v seznam

Slabosti:

- Bolj zapletena uporaba posameznih funkcij

4.1.4. Dokumentacija in viri

S pomočjo dokumentacije in virov glede posameznega ogrodja razvijalcu omogočimo razumevanje posameznih metod in funkcij z uporabo katerih ogrodja poenostavijo razvoj mobilne aplikacije.

4.1.4.1. Avtohtona aplikacija

Prednosti:

- Veliko dobro pregledne in uporabne dokumentacije

Slabosti:

- V dokumentaciji je implementacija in delovanje posameznih elementov pomankljivo razložena

4.1.4.2. Hibridna aplikacija

Prednosti:

- Literatura, ki jo najdemo v knjigah na temo razvoja hibridnih aplikacij je podrobno opisana

Slabosti:

- Malo dokumentacije na temo združevanja jQuery Mobile s PhoneGap ogrodjem za poenostavitev razvoja

4.1.5. Programske knjižnice

Programske knjižnice posameznega ogrodja pripomorejo k hitrejšemu razvoju aplikacije, saj tako razvijalcu mobilne aplikacije ni potrebno implementirati delov, ki se za razvoj mobilne aplikacije pogosto uporabljajo.

4.1.5.1. Avtohtona aplikacija:

Prednosti:

- Enostavna uporaba metod ki nam pohitrijo delo z datotečnim sistemom naprave
- Za implementacijo funkcionalnosti (npr. seznam) lahko uporabimo veliko različnih pristopov in metod
- Boljše razumevanje programske kode zaradi uporabe več javanskih razredov in razločevanja funkcijskega in oblikovnega dela

Slabosti:

- Razumevanje delovanja relacij med javanskimi razredi in razredi uporabniškega vmesnika ter njihovo povezovanje zahteva veliko časa

4.1.5.2. Hibridna aplikacija:

Prednosti:

- Programske knjižnice, ki vsebujejo kaskadne stilske podloge nam prihranijo veliko dela, saj so glavni deli privzeto že oblikovani

- Enostavna implementacija programske kode s pomočjo programske knjižnice jQuery Mobile

Slabosti:

- Nekatere programske knjižnice preveč preoblikujejo posamezne dele HTML dokumenta, kar nam kasneje predstavlja težavo pri lastnem obikovanju

4.1.6. Namestitev aplikacije na fizično napravo

Namestitev aplikacije na fizično napravo pokaže dejanske pomanjkljivosti določenega programskega ogrodja in s tem tudi samo kakovost ogrodja.

Ob namestitvi avtohtone aplikacije:

- Hiter zagon
- Gladki prehodi med prikazi direktorijev
- Ob rotiranju zaslona je prehod delujoč in gladek
- Neprekinjeno delovanje
- Hitro delovanje

Ob namestitvi hibridne aplikacije in v primerjavi z avtohtono aplikacijo:

- Počasnejši zagon
- Počasnejši prehodi med prikazi direktorijev
- Pri rotiranju zaslona se aplikacija zapre in preide v stanje čakanja
- Več napak in občasno prenehanje delovanja

4.2. Priporočila glede uporabljenih ogrodij

Katero ogrodje bo uporabljeno je predvsem odvisno od posameznega razvijalca. Tako bo nekdo, ki ima veliko izkušenj s programskim jezikom Java izbral razvoj z avtohtonim ogrodjem, nekdo drugi, ki ima več izkušenj z spletnimi tehnologijami pa bo raje izbral razvoj s hibridnim ogrodjem. Omenimo naj, da se veliko časa nameni razvoju z avtohtonim ogrodjem. Ta čas, ki ga vložimo v razvoj mobilne aplikacije se nam kasneje povrne ko opazimo, da mobilna aplikacija deluje brezhibno in po pravilih, ki smo jih zasnovali pred samim začetkom razvoja. V primerjavi s hibridno aplikacijo, kjer je razvoj mobilne aplikacije razmeroma hiter pa se ob koncu razvoja lahko pojavijo neželene težave, ki jih je potrebno rešiti in katerim lahko namenimo tudi polovico ali več razvojnega časa. Če se odločimo za razvoj mobilne aplikacije z avtohtonim ogrodjem obenem vlagamo trud v prihodnost, saj se

bomo seznanili in naučili več programskih jezikov, kar bomo kasneje lahko s pridom uporabljali. Če pa razvijamo mobilno aplikacijo samo s hibridnim ogrodjem, je velika verjetnost, da se avtohtonih programskih jezikov za posamezno platformo ne bomo naučili in v primeru, da se razvoj hibridnega ogrodja prekine s strani proizvajalcev bomo morali poseči po avtohtonih ogrodjih za razvoj mobilnih aplikacij.

5. SKLEPNE UGOTOVITVE

V teoretičnem delu diplomskega dela smo opisali tehnologije in razvojna orodja, ki smo jih kasneje uporabili za razvoj mobilne aplikacije. V praktičnem delu diplomskega dela smo nato opisali metodologijo in idejo za razvoj aplikacije ter naredili analizo zahtev aplikacije. Sledil je podroben opis razvoja mobilne aplikacije za avtohtono in hibridno oziroma PhoneGap ogrodje. Končni rezultat so dve delujoči mobilni aplikaciji. Pri razvoju aplikacije z avtohtonim ogrodjem Android zaradi obsežne dokumentacije in literature ter spletnih forumov nismo naleteli na večje težave. V primerjavi s hibridnim ogrodjem PhoneGap lahko opazimo manj literature, to pa predvsem zaradi začetka distribucije ogrodja, ki je sledilo nekaj let za avtohtonim Android ogrodjem. Aplikaciji, katerih primarna naloga je raziskovanje in brskanje po direktorijski strukturi lahko uporabljamo na katerikoli napravi, ki ima nameščen operacijski sistem Android. Omenimo naj tudi, da aplikacijo, ki je razvita s hibridnim ogrodjem lahko uporabimo na kateremkoli operacijskem sistemu, ki to ogrodje podpira. Vse funkcionalnosti in lastnosti, ki so bile zahtevane smo tudi implementirali. Dosegli smo dva pomembna cilja, prvi je enostavna uporaba aplikacije, čemur je pripomogel tudi dobro izdelan uporabniški vmesnik, ki nudi dober pregled nad vsemi funkcionalnostimi mobilne aplikacije in drugi je optimizacija aplikacije glede hitrosti in tekočega delovanja. Prvi cilj velja za razvoj aplikacije z obema ogrodjema, avtohtonim in hibridnim, drugi cilj pa velja predvsem za avtohtono ogrodje.

Skozi podroben opis razvoja mobilne aplikacije v praktičnem delu diplomskega dela smo se dobro seznanili z novimi ogrodji in orodji ter tehnikami razvoja mobilnih aplikacij, sprva za specifično platformo Android in nato še za druge mobilne platforme. Svoje znanje programiranja smo nadgradili na področju programskega jezika Java, ki smo ga predhodno sicer že poznali ter spletnih programskih jezikov kot je JavaScript.

Cilj diplomske naloge, razvoj mobilne rešitve z uporabo različnih programskih ogrodij je bil torej dosežen. S tem smo razširili svoje tehnično znanje na področju programiranja in to predstavlja veliko prednost pri razvoju novih mobilnih aplikacij po katerih je trenutno povpraševanje zelo veliko.

LITERATURA IN VIRI

[1] Anuzzi J.Jr., Darcey L., Conder S. (2013), Introduction to Android Application Development, Michigan, Shepherd Inc., poglavja 1, 2 in 3.

[2] Lee W. (2013), Android Application Development Cookbook, Indiana, John Wiley & Sons Inc., poglavja 1, 2 in 3.

[3] Myer T. (2012), Beginning PhoneGap, Indiana, John Wiley & Sons Inc., poglavja 1, 2, 3, 11, 12.

[4] Natili G. (2013), PhoneGap 3, Birmingham, Packt Publishing Ltd., poglavja 3, 6, 9, 11.

Spletni viri

[1] Academia (2014), Functional requirements. [Online] Dostopno na: <https://www.academia.edu/4028933/Functional_Requirements_of_mobile_application_for_fishermen> [Dostopno 10.03.2014]

[2] Android (2014), Developer tools. [Online] Dostopno na: <https://cordova.apache.org/#about> [Dostopno 15.02.2014]

[3] Android, (2014), Exploring the SDK. [Online] Dostopno na: <https://developer.android.com/sdk/exploring.html> [Dostopno 18.02.2014]

[4] Android (2014), Intent. [Online] Dostopno na: <http://developer.android.com/reference/android/content/Intent.html> [Dostopno 11.02.2014]

[5] Android (2014), List View. [Online] Dostopno na: <http://developer.android.com/guide/topics/ui/layout/listview.html> [Dostopno 22.02.2014]

[6] Android (2014), Supported Media Formats. [Online] Dostopno na: <http://developer.android.com/guide/appendix/media-formats.html> [Dostopno 22.02.2014]

[7] Android (2014), Welcome. [Online] Dostopno na: <http://developer.android.com/about/index.html> [Dostopno 12.02.2014]

[8] Apache Cordova (2013), About Apache Cordova. [Online] Dostopno na: <https://cordova.apache.org/#about> [Dostopno 28.02.2014]

[9] Eclipse (2014), About Eclipse. [Online] Dostopno na: <https://www.eclipse.org/org/> [Dostopno 01.03.2014]

[10] EngineersGarage (2012), Android. [Online] Dostopno na: <http://www.engineersgarage.com/articles/what-is-android-introduction> [Dostopno 01.03.2014]

[11] EzzyLearning (2012), Customizing Android ListView Items. [Online] Dostopno na: <http://www.ezzylearning.com/tutorial.aspx?tid=1763429> [Dostopno 17.02.2014]

[12] GitHub (2014), FileOpener. [Online] Dostopno na: <https://github.com/markeeftb/FileOpener> [Dostopno 02.02.2014]

[13] IconArchive (2014), Icon archive. [Online] Dostopno na: <http://www.iconarchive.com/> [Dostopno 06.02.2014]

[14] jQuery (2014), jQuery. [Online] Dostopno na: <http://jquery.com/> [Dostopno 12.03.2014]

[15] jQuery Mobile (2014), About. [Online] Dostopno na: <http://jquerymobile.com/about/> [Dostopno 02.03.2014]

[16] JQuery Mobile (2012), Themes. [Online] Dostopno na: <http://demos.jquerymobile.com/1.1.1/docs/api/themes.html> [Dostopno 03.03.2014]

[17] JQuery Mobile (2014), Listviews. [Online] Dostopno na: <http://demos.jquerymobile.com/1.3.0-rc.1/docs/demos/widgets/listviews/> [Dostopno 14.02.2014]

[18] Java (2014), About Java. [Online] Dostopno na: <http://www.java.com/en/about/> [Dostopno 04.03.2014]

[19] Mobileapp-development (2013), Writing Specifications for Mobile App. [Online] Dostopno na: <http://mobileapp-development.com/blog/writing-specifications-for-a-mobile-app-development-project.aspx> [Dostopno 06.01.2014]

[20] Mozilla (2014), JavaScript. [Online] Dostopno na: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> [Dostopno 07.03.2014]

[21] PhoneGap (2014), PhoneGap. [Online] Dostopno na: <http://phonegap.com/> [Dostopno 08.03.2014]

[22] StackOverflow (2014), Difference between px, dp, dip and sp. [Online] Dostopno na: <http://stackoverflow.com/questions/2025282/difference-between-px-dp-dip-and-sp-in-android> [Dostopno 15.02.2014]

[23] StackOverflow (2014), How to open pdf via intent. [Online] Dostopno na: <http://stackoverflow.com/questions/10530416/how-to-open-a-pdf-via-intent-from-sd-card> [Dostopno 16.02.2014]

[24] Uva (2007), Framework comparison method. [Online] Dostopno na: <http://dare.uva.nl/document/356530> [Dostopno 09.03.2014]

[25] Vogella (2014), Using lists in Android (ListView). [Online] Dostopno na: <http://www.vogella.com/tutorials/AndroidListView/article.html> [Dostopno 17.02.2014]

[26] Wikipedia (2014), Android operating system. [Online] Dostopno na: <http://en.wikipedia.org/wiki/Android_(operating_system)> [Dostopno 10.01.2014]

[27] Wikipedia (2014), Java programming language, [Online] Dostopno na: <http://en.wikipedia.org/wiki/Java_(programming_language)> [Dostopno 11.03.2014]

[28] Wikipedia (2014), JavaScript. [Online] Dostopno na: <http://en.wikipedia.org/wiki/JavaScript> [Dostopno 10.03.2014]

[29] W3 (2014), Cascading Style Sheets. [Online] Dostopno na: <http://www.w3.org/Style/CSS/> [Dostopno 10.03.2014]

[30] W3 (2014), HTML. [Online] Dostopno na: <http://www.w3.org/community/webed/wiki/HTML> [Dostopno 11.03.2014]

[31] W3 (2014), jQuery Mobile Data Attributes. [Online] Dostopno na: <http://www.w3schools.com/jquerymobile/jquerymobile_ref_data.asp> [Dostopno 02.03.2014]

[32] W3 (2013), What is HTML. [Online] Dostopno na:
<http://www.w3.org/community/webed/wiki/HTML/Training/What_is_HTML%3F>
[Dostopno 08.03.2014]