SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2062

Numerička virtualna tipkovnica pod Android operacijskim sustavom

Gordan Markuš

Zagreb, lipanj 2011.

IZVORNIK

Zahvala

Zahvaljujem svom mentoru prof. dr. sc. Mariu Cifreku na pomoći i iskazanom strpljenju pri izradi ovog završnog rada. Srdačno zahvaljujem dipl. ing. Saši Mrvošu, vanjskom suradniku na Zavodu za elektroničke sustave i obradu informacija, na vodstvu, izrazitoj pomoći i ustupljenim materijalima.

.

Sadržaj

1. Uvod 1

2. Opis razvojnog sustava za mobilne aplikacije 2

2.1. Eclipse 2

2.2. Android SDK i ADT 2

2.3. Stvaranje projekta 5

2.4. Struktura aplikacije 6

2.5. Struktura XML datoteke 7

3. Teorija izvedbe 11

3.1. Pokazna aplikacija 12

3.1.1. Definicija korisničkog sučelja pomoću XML datoteke 14

3.1.2. Dodjeljivanje funkcije aplikaciji 16

4. Programska izvedba virtualne tipkovnice 19

4.1. Definicija korisničkog sučelja 19

4.2. Implementacija klasa 22

4.2.1. Klasa LatinKeyboard 22

4.2.2. Klasa LatinKeyboardView 23

4.2.3. Klasa VirtualKeyboard 24

4.3. Izlazni proizvod 28

5. Diskusija 30

Zaključak 31

Literatura 32

Skraćenice 33

Sažetak 34

Summary 35

Popis slika

Slika 2.1 Android AVD and SDK Manager 3
Slika 2.2 Android emulator 4
Slika 2.3 Izbornik novog Android projekta 5
Slika 3.1 Standardna Android tzv. "qwerty" tipkovnica 11
Slika 3.2 Primjer starog mobilnog uređaja s tipkovnicom 12
Slika 3.3 Dijagram toka demo aplikacije 13
Slika 3.4 Korisničko sučelje demo aplikacije 16
Slika 4.1 Hijerarhijska struktura tipkovnice 19
Slika 4.2 Izbor tipkovnice 28
Slika 4.3 Numerička virtualna tipkovnica 29
Slika 4.4 Simbolička tipkovnica 29

Popis tabela

Tabela 2.1 Vrste prikaza 8
Tabela 2.2 Elementi grafičkog korisničkog sučelja 9
Tabela 3.1 Korišteni elemetni GUI-a 14
Tabela 3.2 Metode za dodavanje funkcionalnosti 17
Tabela 4.1 Funkcije i opis tipki 21
Tabela 4.2 Klasa LatinKeyboard 22
Tabela 4.3 Klasa Key 23
Tabela 4.4 Klasa LatinKeyboardView 23
Tabela 4.5 Deklaracija klase 25
Tabela 4.6 Metode klase VirtualKeyboard - incijalizacija 25
Tabela 4.7 Metode klase VirtualKeyboard - pomoćne funkcije i ispis 26
Tabela 4.8 Metode klase VirtualKeyboard - specifične ugrađene metode 27

1. Uvod

Zadatak ovog završnog rada bio je programski ostvariti virtualnu tipkovnicu pod Android operacijskim sustavom koja će emulirati klasične numeričke tipkovnice na starijim mobilnim aparatima. Android operacijski sustav danas koriste mnogi pametni telefoni te razvoj aplikacija za navedeni operacijski sustav je jednostavan i besplatan. Za izradu aplikacije, odnosno virtualne tipkovnice, koristi se Eclipse IDE te programski jezik Java. Dodak koji se koristi za razvoj aplikacija za Android operacijski sustav je Android SDK te ADT za Eclipse. U određenom poglavlju objasnit ću način instalacije i upotrebu navedenog razvojnog sustava sa svim potrebnim dodacima.

Motivacija ovog rada leži u tome da je osnovna virtualna "qwerty" tpikovnica koju koriste mobilni uređaji s Android operacijskim sustavom nepriklada i neprecizna zbog nedovoljne rezolucije detekcije dodira ekrana osjetljivog na dodir. Uz funkcionalnost numeričke tipkovnice, navedena virtualna tipkovnica zadržat će pojedine funkcionalnosti standardne Android tipkovnice.

Krajnji cilj je razviti tipkovnicu koja najviše odgovara korisniku koristeći se spojem modernih i starijih tipkovnica.

2. Opis razvojnog sustava za mobilne aplikacije

2.1. Eclipse

Eclipse je besplatno višejezično okruženje za razvoj programa (engl.*software development environment*). U sebi sadrži IDE te niz raznih dodataka koji služe za razvoj aplikacija u različitim programskim jezicima. Najčešće korišten programski jezik u Eclipse-u je Java, ali je moguće i programiranje u ostalim programskim jezicima:

- C/C++
- Ruby
- Phyton
- Perl
- PHP
- COBOL

Za svaki programski jezik koristi se različiti IDE, tako se za razvoj Java aplikacija koristi Eclipse JDT, a za C/C++ programe koristi Eclipse CDT.

2.2. Android SDK i ADT

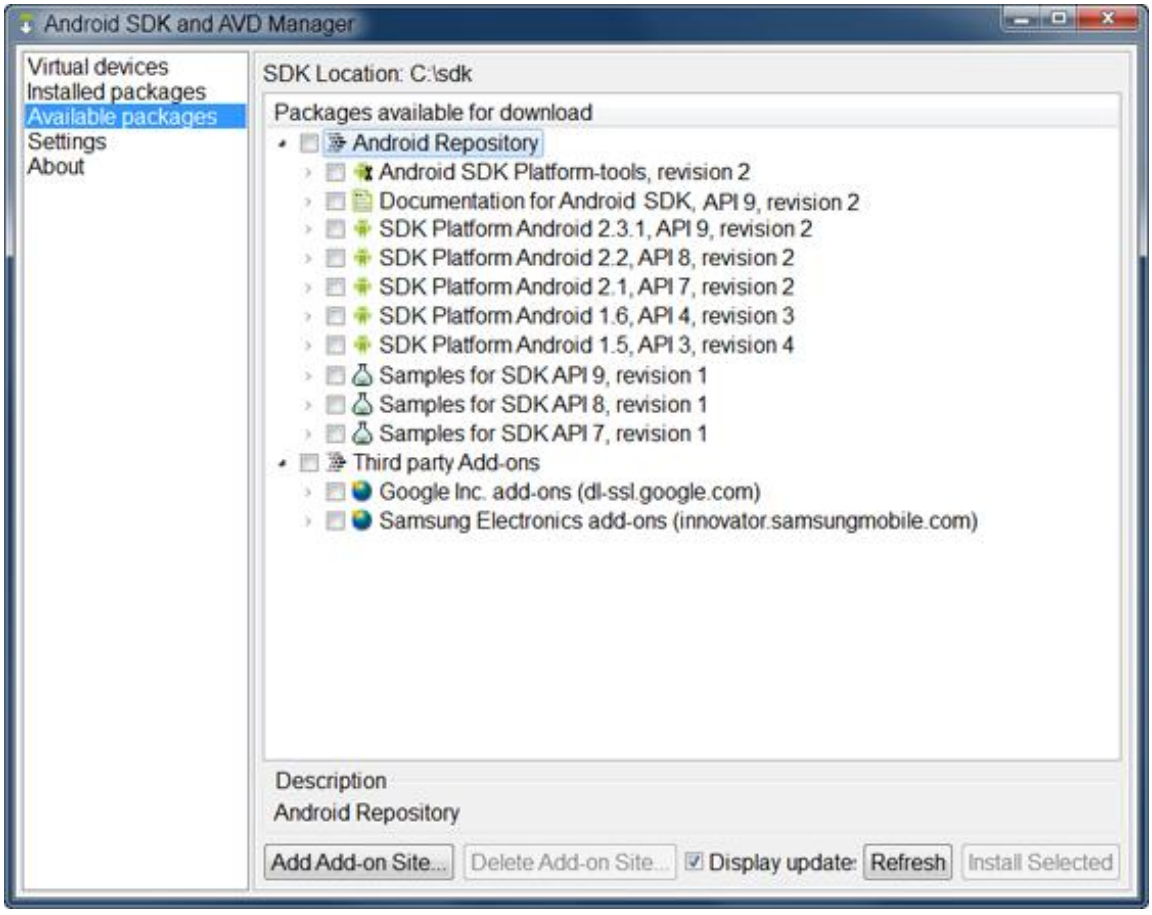
Prije razvoja Android aplikacija u potrebno je postaviti dodatak Android SDK u Eclipse. Android SDK nije potpuno razvojno okruženje - sadrži samo bitne SDK alate, koji nam služe za preuzimanje ostalih SDK komponenata. Android SDK je skupina alata koji omogućuju stvaranje aplikacija za Android operacijski sustav.

Eclipse ne sadrži dovoljno alata za razvoj aplikacija za Android operacijski sustav. Android nudi posebni dodatak za Eclipse IDE, Android ADT, koji omogućuje i također zantno olakšava razvoj Android aplikacija. Android ADT proširuje mogućnosti Eclipse-a, omogućuje jednostavno postavljanje Android projekata i dizajniranje korisničkog sučelja. Po završetku projekta Android ADT omogućuje stvaranje **.apk** datoteka koje služe za distribuciju razvijene aplikacije.

Android ADT se preuzima na sljedeći način: **Help > Install New Software > Add > ...** Navedene alate je potrebno nazvati te preuzeti sa sljedeće stranice (navedeni link se navodi unutar URL polja): https://dl-ssl.google.com/android/eclipse

Koristeći *Android SDK and AVD Manager* (Slika 2.1) preuzimamo potrebne komponente i alate za naše razvojno okruženje. Pristupamo mu na sljedeći način:

- u Eclipse-u izaberemo padajući izbornik **Window > Android SDK and AVD Manager**
- iz Windows-a pokrenemo **SDK Manager.exe** datoteku
- u Linux-u preko terminala otvorimo **tools/** direktorij unutar Android SDK te zatim izvršimo naredbu **android**



Slika 2.1 Android AVD and SDK Manager

Preuzimamo potrebne komponente za razvoj Android aplikacija pomoću korisničkog sučelja (Slika 2.1). Komponente preuzimamo iz Android repozitorija koji nudi sljedeće skupine komponenata:

- SDK alati - služe za uklanjanje pogrešaka i testiranje aplikacije

- SDK platformski alati - sadrže platformski ovisne alate za razvijanje aplikacije
- Android platforme
- USB drivere za Windows-e
- Primjeri - primjeri aplikacija za svaku Android razvojnu platformu
- Dokumenatcija - lokalna kopija posljednje dokumentacija za Android API (engl. *Application Programming Interface*)

Uz navedene komponente i alate moguće je preuzeti *Third party Add-ons* koji omogućuju razvijanje okruženja za specifičnu Android vanjsku biblioteku. Primjer takvih dodataka je Google Maps biblioteka.

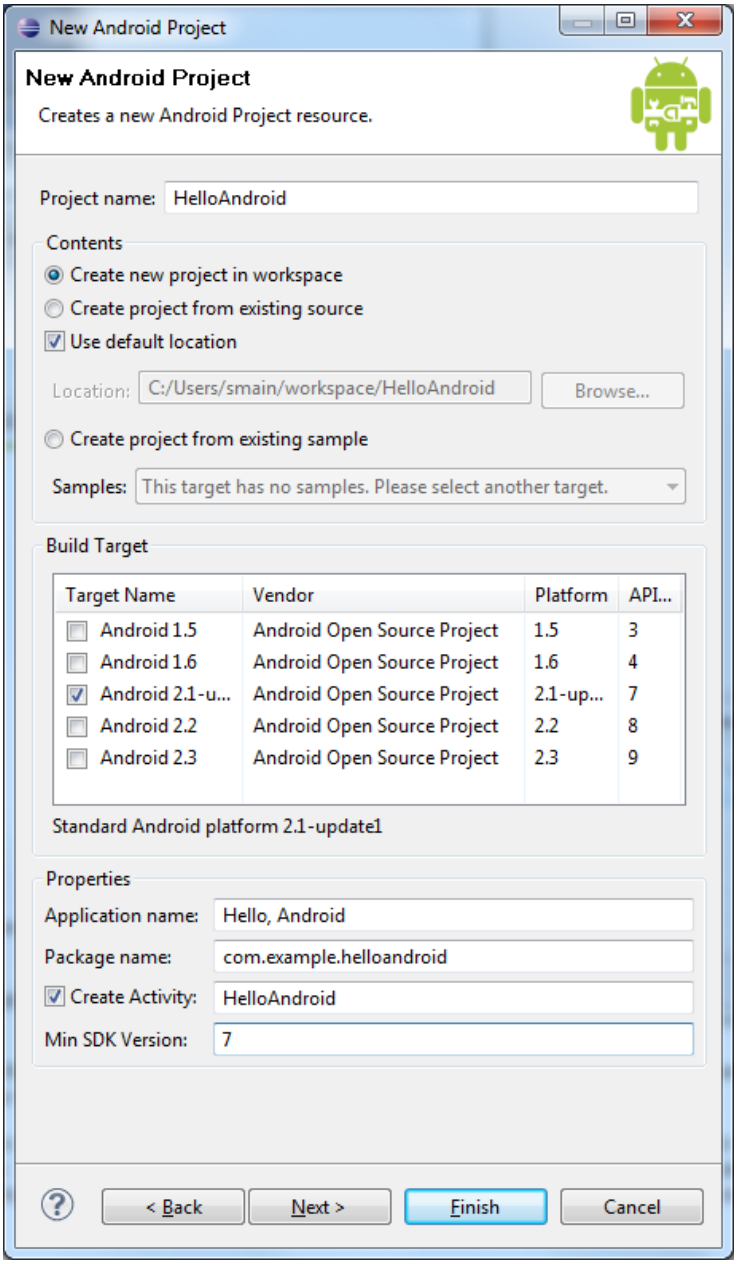
Android SDK and AVD Manager nudi opciju stvaranja emulatora za testiranje aplikacija na virtualnom uređaju - AVD (Slika 2.2). Izbornik nudi mogućnost izbora i proizvoljnog odabira različitih *hardware* i *software* performansi koje će biti emulirane.

Slika 2.2 Android emulator

2.3. Stvaranje projekta

U ovom potpoglavlju objasniti će se stvaranje temeljnog Android projekta i njegovu strukturu. Podrazumijeva se instalacija Eclipse-a i preuzimanje potrebnih paketa za ravoj Android aplikacija (ADT i SDK) i stvaranje vlastitog AVD-a.

Sljedeći korak stvaranje novog Android projekta na sljedeći način: **File > New > Project > Android Project**. Pojavljuje se izbornik u kojem postavljamo osnovne postavke projekta (Slika 2.3).



Slika 2.3 Izbornik novog Android projekta

Potrebno je ispuniti sljedeća polja:

- *Project Name* - ime direktorija u kojem se nalaze projektne datoteke
- *Application Name* - ime aplikacije koje će se prikazati na Android uređaju
- *Package Name* - ime paketa pod kojim će se nalaziti sav potreban kôd. Ime paketa mora biti jedinstveno, također preporučljiva je dosljednost pri nazivanju paketa
- *Create Activity* - ime klase koja će se automatski stvoriti. Ona će biti podklasa klase `Activity`. Klasa `Activity` omogućuje rad aplikacije, može stvarati korisničko sučelje ili obavljati određen zadatak. Aplikacija može sadržati više takvih klasa, ali se njima pristupa odvojeno
- *Min SDK Version* - vrijednost koja određuje minimalni nivo API-a potreban za korištenje aplikacije
- *Build Target* - izbor platforme kompajliranja. Potrebno je imati uređaj jednake ili naprednije platforme; u protivnom, aplikaciju neće biti moguće pokrenuti

2.4. Struktura aplikacije

Nakon popunjavanja polja prema navedenim uputama (Slika 2.3), automatski se generiran kostur Android aplikacije. Android aplikacija je strukturirana u stablo direktorija. Struktura Android aplikacije je specifična i potrebno ju je dobro proučiti. Unutar samog projekta nalazi se više datoteka i direktorija:

- *AndroidManifest.xml* - xml datoteka koja opisuje aplikaciju koja se pokreće i koje komponente koristi aplikacija
- *build.xml* - Ant skripta za kompajliranje i instaliarnje aplikacije
- *assets/*: - direktorij koji sadrži statičke datoteke koje želimo uz prenjeti uz aplikaciju
- *gen/:* - sadrži automatski generirane datoteke
- *src/:* - sadrži Java *source* kôdove
- *res/:* - sadrži ikone, definicije grafičkih korisničkih sučelja i XML datoteke

U zadanom primjeru unutar src/: direktorija nalazi se datoteka HelloAndroid.java koja sadrži aktivnost `HelloAndroid` te metodu `onCreate()` koja se poziv pokretanjem

aktivnosti. Aktivnosti su javne klase koje nasljeđuju osobine od **android.app.Activity** klase. Metoda `onCreate()` u sebi sadrži inicijalizaciju i definiranje korisničkog sučelja, dok `import` linije sadrže sve klase koje se namjeravaju referencirati.

```
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroid extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Kôd 2.1 Automatski generirana aktivnost

Komunikaciju korisnika s mobilnom aplikacijom omogućuje GUI (engl. *Graphic User Interface*). Korisničko sučelje Android aplikacija sastoji se u većini slučajeva od različitih elemenata korisničkog sučelja (*widget*). *Widget* je objekt preko kojeg korisnik komunicira s aplikacijom; to su na primjer tipke, polja, labele ili padajući izbornici. Specifičnost programiranja Android aplikacija je u tome, što je moguće uz stvaranja grafičkog korisničkog sučelja putem Java kôda, stvaranje korisničkog sučelja pomoću XML datoteka.

Dinamičko instanciranje *widget*-a je rezervirano za kompliciranije scenarije kada nije unaprijed poznat izgled sučelja, već je izgled uvjetovan ulaznim parametrima.

2.5. Struktura XML datoteke

Definiranje korisničkog sučelja pomoću XML datoteka izrazito je korisno iz više razloga: jednostavnost, hijerarhijska struktura, promjenjivost, povezanost podelemenata sučelja i nepodložnost promjenama izvornog kôda. Referenciranje elemenata korisničkog sučelja (*widget*) u Java kôdu preko jedinstvenog `android:id` atributa omogućava jednostavno programiranje funkcija sučelja. Zbog navedenih razloga XML postaje sve korišteniji u definiciji grafičkog korisničkog sučelja.

Na vrhu XML datoteke određuje se tip prikaza. Tip prikaza naziva se *layout*, on name služi grupiranju više podelemenata korisničkog sučelja u cijelinu. Tip prikaza sugerira u kakvom su odnosu objekti djeca (Tabela 2.1). Klase tipa `Layout` su podklase klase `ViewGroup`.

Klasa	**Opis**
`FrameLayout`	Okvir koji se ponaša kao jedinstven objekt
`GridView`	Prikazuje mrežu s M redova i N stupaca
`Gallery`	Horizontalni pomičini prikaz slika
`LinearLayout`	Organizira djecu u jedan red ili stupac
`ListView`	Prikazuje pomičnu listu s jednim redom
`RelativeLayout`	Omogućava postavljanje relativnog odnosa među objektima djecom i roditeljima
`ScrollView`	Vertikalni pomični niz elemenata
`Spinner`	Prikazuje jedan izabrani element liste u predefiniranom prostoru
`SurfaceView`	Omogućuje direktan pristup prostoru za crtanje
`TabHost`	Tablični izbor koji omogućuje promjenu prozora pri pritisku
`TableLayout`	Izgrađuje tablicu s proizvoljnim brojem redova i stupaca; untar svake ćelije se nalazi zaseban *widget*
`ViewFlipper`	Lista koja prikazuje jedan po jedan element u određenim intervalima

Tabela 2.1 Vrste prikaza

Unutar *layouta* se nalaze elementi korisničkog sučelja (*widget*) koji su podklase `View`. Takvnih elemenata ima mnogo te služe za ispunjavanje različitih zadataka ovisno o zahtjevima. Sve vrste elemenata korisničkog sučelja sadrže temeljne atribute kao što su:

- `android:id`
- `android:layout_width / android:layout_height`
- `android:text`

Osim navedenih temeljnih atributa svaka klasa elemenata ima zasebne specifične atribute ili funkcionalnosti te se iz tog razloga različiti elementi korisničkog sučelja koriste za izvršenje različitih zadaća. Specifične funkcionalnosti se definiraju preko XML datoteke, a njihova funkcija im se pridodaje unutar glavnog programa (`Activity`), ako postoji, ili klase koja generira korisničko sučelje.

U nastavku su opisani od autora izdvojeni elementi korisničkog sučelja s opisom i specifičnim atributima i metodama (Tabela 2.2).

Element korisničkog sučelja	Opis, specifične metode ili atributi

`EditText`	Nasljeđuje `TextView`; promjenjivo polje
	`string getText()`
	`void setText()`
	`void selectAll()`

`Button`	Tipka `android:onClick`

`CheckBox`	Okvir za izbor
	`void setChecked()`
	`boolean isChecked()`

`Scroller`	Omogućuje klizeći prijelaz
`RadioButton`	Gumb s dva stanja; sklopka `void toggle()`
`PopupMenu`	Implementira iskačući izbornik

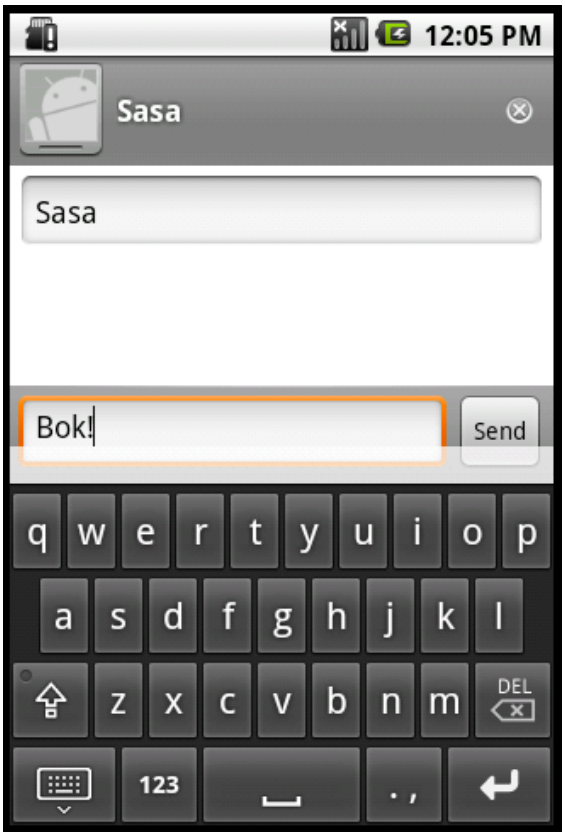
`NumberPicker`	Omogućuje korisniku izbor brojeva unutar predefiniranog raspona
	`void setMaxValue(int )`
	`void setMinValue(int )`
	`void setValue(int )`

Tabela 2.2 Elementi grafičkog korisničkog sučelja

Specifiničnost izrade android aplikacije leži u automatskom generiranju **R.java** datoteke. Unutar navedene datoteke se nalaze statičke vrijednosti koje jednoznačno referenciraju objekte koje smo stvorili. Time se znatno olakšava pristupanje pojedinom objektu. Uređivanje **R.java** datoteke od strane korisnika je moguće, ali nije preporučljivo. Statičke vrijednosti su tipa `final int` te su zapisane u heksadekatskom formatu.

3. Teorija izvedbe

Kao što je navedeno u uvodu temeljna zamisao ovog rada je razviti novu napredniju virtualnu tipkovnicu osjetljivu na dodir za mobilne uređaje s Android operacijskim sustavom. To će se ostvariti oponašanjem već postojeće tipkovnice za mobilne uređaje s Android operacijskim sustavom (Slika 3.1) uz izgled i funkcionalnost tipkovnica kakve su se koristile na starim mobilnim uređajima (Slika 3.2).



Slika 3.1 Standardna Android tzv. "qwerty" tipkovnica

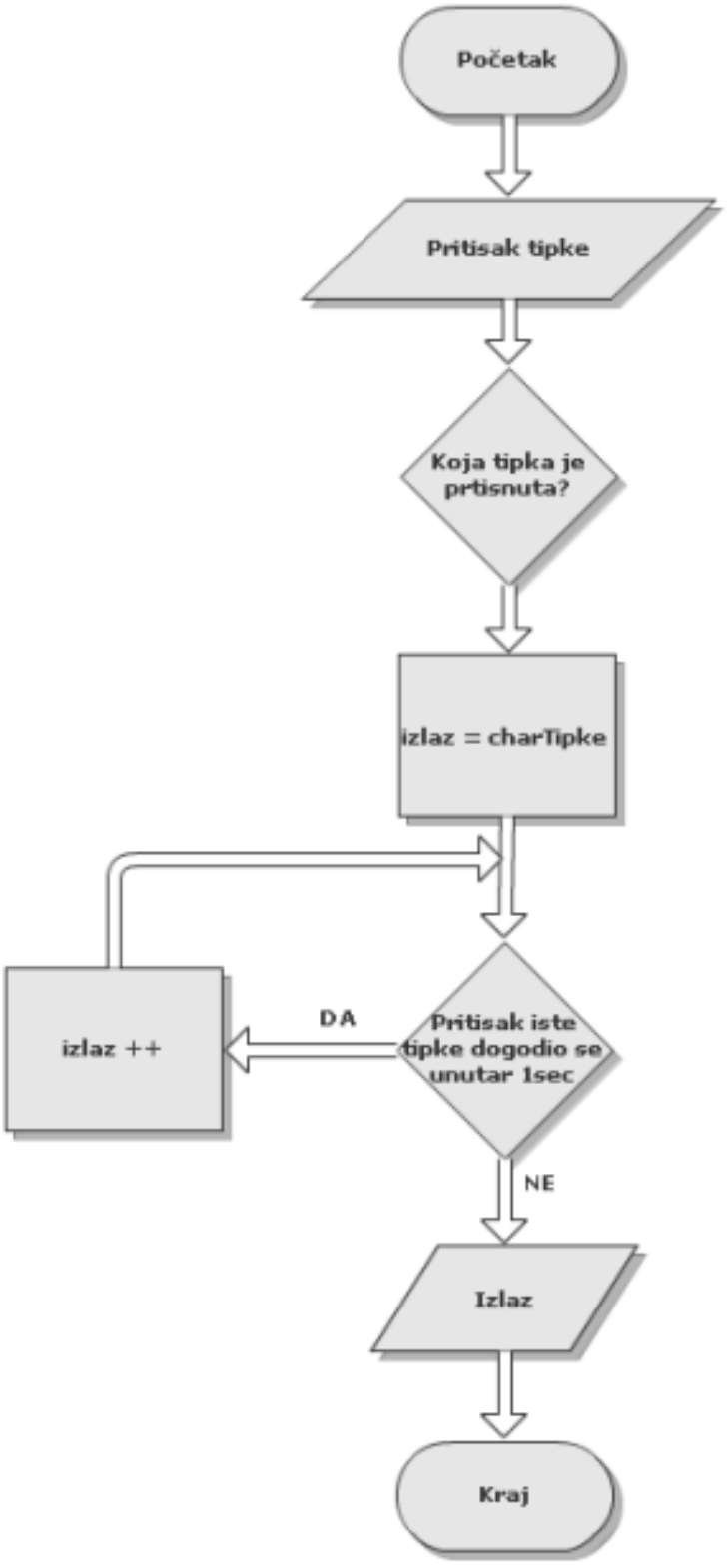
Ulazni zahtjevi ovog projekta su da korisnik može u bilo kojem trenutku birati između standardne tipkovnice i tipkovnice koja će biti razvijena. Također korisniku je potrebno omogućiti određene funkcionalnosti koje postoje na Android tipkovnici, a stariji mobilni uređaji ih nemaju. Na primjer: mogućnost sakrivanja tipkovnice, prelazak iz pisanja teksta u pisanje simbola i brojeva.

Slika 3.2 Primjer starog mobilnog uređaja s tipkovnicom

Pogledom na stari mobilni uređaj (Slika 3.2), vidimo da je potrebno sačuvati temeljnih 12 tipki kao što su postojale na starim mobilnim uređajima s očuvanim rasporedom. Implementacija početne aplikacije koja će jednako oponašati staru tipkovnicu je početni zadatak koji će se izvršiti u svrhu upoznavanja i razumijevanja programiranja Android aplikacija. Nakon toga će se krenuti u stvaranje upotrebljive numeričke virtualne tipkovnice pod Android operacijskim sustavom.

3.1. Pokazna aplikacija

Dijagram toka navedene aplikacije dan je na sljedećoj stranici dokumenta (Slika 3.3). Ovisno o pritisku određene tipke potrebno je ispisati traženo slovo, znak ili broj. Također potrebno je nadzirat frekvenciju uzastopnog pritiska iste tipke, kao što je to na starijim mobilnim uređajima. Navedena aplikacija neće biti u mogućnosti koristiti se kao stvarna tipkovnica na mobilnim uređajim s Android operacijskim sustavnom, niti neće biti u stanju napisani tekst i simbole prosljeđivati dalje.



Slika 3.3 Dijagram toka demo aplikacije

3.1.1. Definicija korisničkog sučelja pomoću XML datoteke

Za početak potrebno je implementirati GUI koji oponaša tipkovnicu na starijm mobilnim aparatima. Definiciju GUI-a izvršiti ćemo uređivanjem XML datoteke. Vrsta odnosa između elemenata GUI-a demo aplikacije biti će postavljen kao `RelativeLayout`. Definiranjem ovog odnosa olakšana je manipulacija položaja elemenata GUI-a u međusobnom odnosu. Zatim je potrebno odrediti koliko elemenata grafičkog korisničkog sučelja aplikacija zahtijeva. U svrhu ove aplikacije koristit ćemo 13 *widgeta*; dvanaest tipki i jedno promjenjivo polje koje sadrži tekst. Definicija navedenih elemenata GUI-a i njihovih specifičnih atributa navedeni su u nastavku (Tabela 3.1).

Element GUI-a	Atributi

`TextView`	`android:id="@+id/tekstbox"`
	`android:layout_width="fill_parent"`
	`android:layout_height="75px"`
	`android:text=""`
	`android:textSize="18sp"`
	`android:layout_below="@+id/lejaut"`
	`android:layout_alignParentLeft="true"`
	`android:singleLine="false"`
	`android:autoText="true"`

`Button (1..12)`	`android:id`
	`android:layout_width="110px"`
	`android:layout_height="85px"`
	`android:padding="10px"`
	`android:text`
	`android:layout_below`
	`android:layout_alignParentLeft`
	`android:layout_toRightOf`
	`android:onClick`

Tabela 3.1 Korišteni elemetni GUI-a

Promjenjivo polje za prikaz teksta je samo jedno te su navedeni svi njezini potrebni atributi. Atribut `android:text` definira inicijalno stanje teksta koji se nalazi u polju. Definirano je da je polje pri pokretanju aplikacije prazno kako bi se moglo omogućiti pisanje iz početka pri svakom pokretanju aplikacije. Atribut `android:singleLine` određuje mogućnost pisanja unutar više redova. Poželjno je da se isključi ova opcija. Ostali atributi nam služe za pozicioniranje polja unutar prikaza, njegove dimenzije te font ispisa.

Sve tipke kao elementi sučelja sadrže tri indentična atributa, a to su:

- `android:padding`
- `android:layout_height`
- `android:layout_width`

Oni su jedaki za sve tipke jer su očito svi istih dimenzija te je razmak među njima jednolik. Vidljivo je da se nalaze mnogi atributi za pozicioniranje tipki u relativnom odnosu, što nam omogućuje `RelativeLayout` te `android:id` koji nam jednoznačno određuje svaku tipku. Atribut `android:text` određuje s kojim će tekstom biti ispunjen tipka, što nam služi korisniku za raspoznavanju pojedinih tipki. Atribut `android:onClick` je specifičan za interaktivne elemente grafičkog sučelja koju su klase `Button`. Njime se definira ime metode koja će biti pozvana pritiskom na tipku. Navedeni atribut je izuzetno koristan jer nam omogućuje direktno definiranje metode unutar izvornog kôda bez da smo prije referencirali tipku unutar kôda, već isključivo unutar XML datoteke. Istu metodu mogu pozivati više tipki. Implementacija metode biti će objašnjena u sljedećem potpoglavlju. Važno je napomenuti da se metoda `onClick` pojedine tipke može također definirati isključivo iz izvornog kôda, gdje se svakoj tipki pristupa pomoću jedinstvenog `android:id`. Time bi se izgubilo na preglednosti kôda te bi zanemarili jednu važnu blagodat programiraju Android aplikacija u ovom razvojnom okruženju.

Nakon postavljanja XML datoteke na način koji je opisan; izgrađeno je sljedeće korisničko grafičko sučelje (Slika 3.4).

Slika 3.4 Korisničko sučelje demo aplikacije

3.1.2. Dodjeljivanje funkcije aplikaciji

Nakon definiranja korisničkog sučelja aplikacije potrebno je uvesti funkcionalost elemetima sučelja. Dodavanje funkcionalnosti aplikacije vršimo kroz uređivanje izvornog kôda. Izvorni kôd se nalazi unutar *src/* direktorija. U početku se automatski generira kôd od strane Eclipse-a kao što je prije navedeno. Uređivanjem navedenog generiranog kôda pridodaje se funkcionalnost aplikaciji.

Prvo što je potrebno napraviti je uvesti `import` linije jer u slučaju da one ne postoje Eclipse IDE ne može pronaći ugrađene metode specifične za pojedine elemente korisničkog sučelja. Za ovu aplikaciju potrebno je uvesti sljedeće datoteke:

- `android.view.View`
- `android.widget.Button`
- `android.widget.TextView`

Dodavanje biblioteke za tipke (`android.widget.Button`) nije potrebno jer smo metodu `onClick` za svaku pojedinu tipku definirali unutar XML datoteke, ali preporučeno je da se doda. Biblioteka za polje s promjenjivim tekstom se dodaje (`android.widget.TextView`) jer će se koristiti njezine ugrađene metode te će se ponašati kao varijabla. Deklaracija varijable tipa `TextView` izvršava se na sljedeći način:

```
TextView polje;
...
@Override
public void onCreate(Bundle savedInstanceState)
...
polje=(TextView)findViewById(R.id.tekstbox);
```

Kôd 3.1 Deklaracija varijable

Pomoću jedinstvene adrese zapisane unutar **R.java** datoteke pristupamo pojedinom elementu kojeg smo definirali unutar XML datoteke. Važno je primjetiti da je potrebno izvršiti *cast* varijable prije referenciranja elementa sučelja.

Unutar XML datoteke definirane su `onClick` metode, a sad ih je potrebno referencirati i dodati im funkcionalnost. Za navedenu svrhu izgrađene su sljedeće funkcije (Tabela 3.2):

Prototip funkcije	Opis
`public void klik(View)`	Funkcija kao ulazni parametar dobiva korisničko sučelje; ovakvih funkcija ima koliko i tipki (12); jedina razlika između njih je `char` koji stvaraju i prosljeđuju funkciji `applyFormat`
`private void applyFormat(char)`	Funkcija kao ulazni parametar dobiva varijablu tipa `char` te ju spaja s već postojećim tekstom, a zatim ga ispisuje na prikaz pomoću ugrađene metode `setText`
`public final void polje.setText(CharSequence)`	Ugrađena metoda specifična za prikaz tipa `TextView;` kao ulazni parametar dobiva varijablu tipa `string` ili `char` te ju ispisuje unutar prikaza

Tabela 3.2 Metode za dodavanje funkcionalnosti

Nakon definicije funkcije pojedinih tipki javlja se problem detekcije višestrukog pritiska iste tipke unutar zadanog vremenskog intervala. Unutar aplikacije potrebno je izvršiti

dohvat vremena i brojanje frekvencije uzastopnog pritiska iste tipke. To se izvodi dohvatom sistemskog vremena nakon pojedinog pritiska tipke te ga nakon sljedećeg uzastopnog pritiska iste tipke umanjuje za novo sistemsko vrijeme, ako se vremenski interval dva uzastopna pritiska nalazi unutar 1 sekunde potrebno je ispisati sljedeće slovo u nizu. Dohvat programskog vremena vrši se pomoću ugrađene metode `currentTimeMillis`. Navedena metoda nalazi se unutar biblioteke `java.lang.System`.

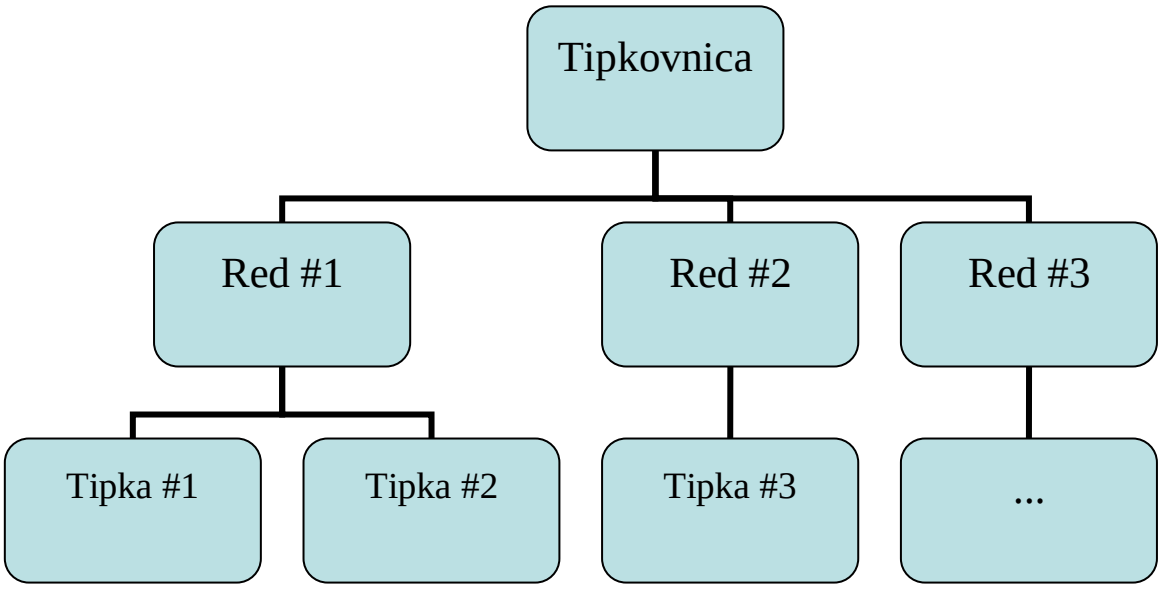
Navedena funkcionalnost se neće implementirati unutar aplikacije zbog nedostatka vremena. Uz to ovo je samo demo aplikacija koja može pomoći čitatelju daljni razvoj i nadogradnju aplikacije. Implementacija stvarne funkcionalne virtualne tipkovnice biti će obrađena u sljedećem poglavlju.

4. Programska izvedba virtualne tipkovnice

Korištenjem ugrađene klase `KeyboardView` znatno je olakšana implementacija tipkovnice za Android operacijski sustav. Metode koje koristi navedena klasa su specifične za nju te će njihova upotreba i funkcija biti objašnjena u daljnjim poglavljima.

4.1. Definicija korisničkog sučelja

Definicija grafičkog korisničkog sučelja biti će izvršena pomoću XML datoteka. Klasa `KeyboardView` ima specifičan način oblikovanja grafičkog korisničkog sučelja. Sučelje se dijeli hijearhijski u stablo (Slika 4.1).



Slika 4.1 Hijerarhijska struktura tipkovnice

Također na takav način se izgrađuje XML datoteka. Tipkovnica sadrži atribute koji definiraju veličinu svih tipki, položaj tipkovnice i razmak među tipkama. Tipke sadrže atribute koji kasnije služe za dodavanje funkcije pojedine tipke, dimenzije tipke te njihov oblik prikaza (slika ili znakovi). Primjer deklaracije tipkovnice prikazan je kôdom 4.1.

```
<Keyboard xmlns:android
    android:keyWidth
    android:horizontalGap
    android:verticalGap
    android:keyHeight
    >

    <Row>
        <Key android:codes
            android:keyLabel
            android:keyWidth
            android:keyEdgeFlags
            android:keyIcon
            android:isRepeatable/>
        <Key
            ...
            />
     </Row>
```

Kôd 4.1 Primjer definicije jednog reda tipkovnice s priloženim atributima

Atributi koji opisuju pojedinu tipku i njihov opis su sljedeći:

- `android:codes` - navodimo koji će se znak prema ASCII tablici generirati pritiskom; navedena funkcija će biti programirana u izvornom kôdu; odvajanjem više znakova dodjeljuje se funkcionalnost uzastopnog pritiska tipke (*multitap*)
- `android:keyLabel` - ime prikaza
- `android:keyWidth; android:keyHeight` - dimenzije tipke; standardne vrijednosti deklarirane su prilikom definiranja cijele tipkovnice, a unutar same tipke se dodaju iznimke
- `android:keyIcon` - slika prikaza, ako postoji
- `android:isRepeatable` - mogućnost ponavljanja

Vođen ulaznim zahtjevima izrađena je tipkovnica sa sljedećim tipkama (Tabela 4.1).

Naziv tipke	Funkcija i opis
1 .,	Služi za unos interpunkcijskih znakova i brojke jedan **Redoslijed znakova: . , ? ! : ; 1**
2 abc	**Redoslijed znakova: a b c 2**
3 def	**Redoslijed znakova: d e f 3**
4 ghi	**Redoslijed znakova: g h i 4**
5 jkl	**Redoslijed znakova: j k l 5**
6 mno	**Redoslijed znakova: m n o 6**
7 pqrs	**Redoslijed znakova: p q r s 7**
8 tuv	**Redoslijed znakova: t u v 8**
9 wxyz	**Redoslijed znakova: w x y z 9**
*	**Redoslijed znakova: * + - _ ( ) = "**
0	**Redoslijed znakova: razmak 0**
#	**Redoslijed znakova: # & / % $**
DEL	Briše
SHIFT	Omogućuje ispis velikih slova ili alternativnu funkciju tipke
SYM	Tipka koja otvara simboličku tipkovnicu standardnog Android uređaja
CLOSE	Tipka koja služi za sakrivanje tipkovnice nakon prestanka upotrebe

Tabela 4.1 Funkcije i opis tipki

Emulirana je virtualna numerička tipkovnica koja sadrži pojedine funkcionalnosti standardne Android tipkovnice. Definicijom XML datoteke nije definirana funkcija pojedinih tipki već njezin fizički izgled i temelj za pridodavanje stvarne funkcionalnosti.

4.2. Implementacija klasa

4.2.1. Klasa LatinKeyboard

Navedena klasa služi za stvaranje tipkovnice. Klasa je deklarirana na sljedeći način (Tabela 4.2):

`public class LatinKeyboard extends Keyboard`	

konstruktor	`public LatinKeyboard(Context context, int layoutTemplateResId, CharSequence characters, int columns, int horizontalPadding)`
	`{`
	`super(context, layoutTemplateResId, characters, columns, horizontalPadding);`
	`}`

varijable	`private Key mEnterKey`

prototip funkcije	`void setImeOptions(Resources res, int options)`
	
	`protected Key createKeyFromXml(Resources res, Row parent, int x, int y, XmlResourceParser parser)`

Tabela 4.2 Klasa LatinKeyboard

Funkcija `setImeOptions` nam služi da postavi odgovarajuću vidljivu labelu pri pritisku tipke ako postoji.

Funkcija `createKeyFromXml` stvara tipku na temelju podataka dohvaćenih iz odgovarajuće XML datoteke.

Unutar klase nalazi se također deklaracija klase `Key` (Tabela 4.3).

`static class LatinKey extends Keyboard.Key`	

konstruktor	`public LatinKey(Resources res, Keyboard.Row parent, int x, int y, XmlResourceParser parser)`
	`{`
	`super(res, parent, x, y, parser);`
	`}`

prototip funkcije	`public boolean isInside(int x, int y)`

Tabela 4.3 Klasa Key

Implementacija ove klase omogućuje deklariraciju različitih tipkovnica, ovisno o XML datoteci, koje će biti dostupne korisniku.

4.2.2. Klasa LatinKeyboardView

Klasa je preuzeta iz primjera. Nasljeđuje klasu `KeyboardView`. Klasa služi za dodavanje dodatnih opcija korisničkom sučelju. Procesira se kao posebna tipka (delete, shift ili enter). Deklaracija klase i njezina funkcija nalazi se u nastavku (Tabela 4.4).

`public class LatinKeyboardView extends KeyboardView`	

konstruktor	`public LatinKeyboardView(Context context, AttributeSet attrs, int defStyle)`
	`{`
	`super(context, attrs, defStyle);`
	`}`

prototip funkcije	`protected boolean onLongPress(Key key)`

Tabela 4.4 Klasa LatinKeyboardView

4.2.3. Klasa VirtualKeyboard

Klasa `VirtualKeyboard` dodaje funkciju tipkovnici te se služi prethodno navedenim klasama kao varijablama. Navedena klasa služi se specifičnim funkcijama koje se nalaze u biblioteci klase od koje nasljeđuje svojstva. Također implementirane su pomoćne funkcije koje nam služe za jednostavnije upravljanje zadaće tipkovnice.

4.2.3.1 Varijable

Koriste se sljedeće varijable:

- `private long mLastShiftTime`
- `private KeyboardView mInputView`
- `private StringBuilder mComposing`
- `private int mLastDisplayWidth`
- `private boolean mCapsLock`
- `private LatinKeyboard mSymbolsKeyboard`
- `private LatinKeyboard mSymbolsShiftedKeyboard`
- `private LatinKeyboard mQwertyKeyboard`
- `private LatinKeyboard mQwertyOldKeyboard`
- `private LatinKeyboard mCurKeyboard`
- `private long mLastShiftTime`
- `private long mMetaState`
- `private long mMetaState`
- `private String mWordSeparators`

Varijable tipa `LatinKeyboard` služe da u njima spremamo podatke o stvorenim tipkovnicama unutar XML datoteka ili referenciramo tip postojeće tipkovnice. Varijabla `mInputView` služi za stvaranje izabrane tipkovnice pomoću funkcije `mInputView.setKeyboard()`. Varijabla `mMetaState` pokazuje koje su funkcijske tipke pritisnute, ako postoje. Ostale varijable nalaze se kao pomoćne varijable unutar metoda.

4.2.3.2 Metode i funkcije

Klasa je deklarariana na sljedeći način (Tabela 4.5).

```
public class VirtualKeyboard extends InputMethodService implements KeyboardView.OnKeyboardActionListener
```

Tabela 4.5 Deklaracija klase

Tabela 4.6 prikazuje popis metoda i funkcija za inicijalizaciju klase `VirtualKeyboard`.

Prototip funkcije	Opis
`public void onCreate()`	Prilikom stvaranja dohvaća podatke o separatorima (interpunkcijski znakovi, razmak itd.)
`public void onInitializeInterface()`	Incijalizacija korisničkog sučelja; dohvaća podatke o definiranim tipkovnicama unutar XML datoteke i pridjeljuje ih određenim varijablama
`public View onCreateInputView()`	Funkcija koja emulira tipkovnicu iz podataka definiranih unutar funkcije
`public void onStartInput(EditorInfo attribute, boolean restarting)`	Incijalizacija načina unosa u ovisnosti o tipu teksta koji želimo unjeti
`public void onFinishInput()`	Nakon završenog unosa tipkovnica se sakriva i postavlja u incijalno stanje
`public void onStartInputView(EditorInfo attribute, boolean restarting)`	Omogućuje čuvanje stanja tipkovnice nakon zatvaranja
`private boolean translateKeyDown(int keyCode, KeyEvent event)`	Omogućuje procesiranje posebnih funkcija i tumači pritisak tipke
`public boolean onKeyDown(int keyCode, KeyEvent event)`	Nakon pritiska funkcija odabire akciju koju će napraviti ovisno o pritisnutoj tipci

Tabela 4.6 Metode klase VirtualKeyboard - incijalizacija

Također implementirano je mnogo pomoćnih funkcija koje nam pomažu da procesiramo sve zadaće tipkovnice (Tabela 4.7).

Prototip funkcije	Opis
`private void commitTyped(InputConnection inputConnection)`	Pomoćna funkcija koja služi za pisanje znakova
`private void updateShiftKeyState(EditorInfo attr)`	Osvježava stanje SHIFT tipke ovisno o tipkovnici
`private boolean isAlphabet(int code)`	Provjera znaka da li je uneseno slovo
`private void keyDownUp(int keyEventCode)`	Funkcija uzrokuje prikaz pritisnute tipke
`private void sendKey(int keyCode)`	Procesiranje znaka; pomoćna funkcija
`public void onKey(int primaryCode, int[] keyCodes)`	Implementacija `KeyboardViewListener` unutar sebe poziva razne podfunkcije
`public void onText(CharSequence text)`	Procesiranje pritiska ako je unešen tekst
`private void handleBackspace()`	Pomoćna funkcija implementirana da emulira tipku DEL
`private void handleShift()`	Pomoćna funkcija postavlja stanje tipke SHIFT na pritisak
`private void handleCharacter(int primaryCode, int[] keyCodes)`	Funkcija koja obrađuje pritisnut znak te ga priprema za ispis
`private void handleClose()`	Pomoćna funkcija procesira posebnu tipku za sakrivanje tipkovnice
`private void checkToggleCapsLock()`	Funkcija osvježava stanje tipke SHIFT
`private String getWordSeparators()`	Dohvaća separatore iz *values/string*
`public boolean isWordSeparator(int code)`	Provjera da li je pritisnut znak separator

Tabela 4.7 Metode klase VirtualKeyboard - pomoćne funkcije i ispis

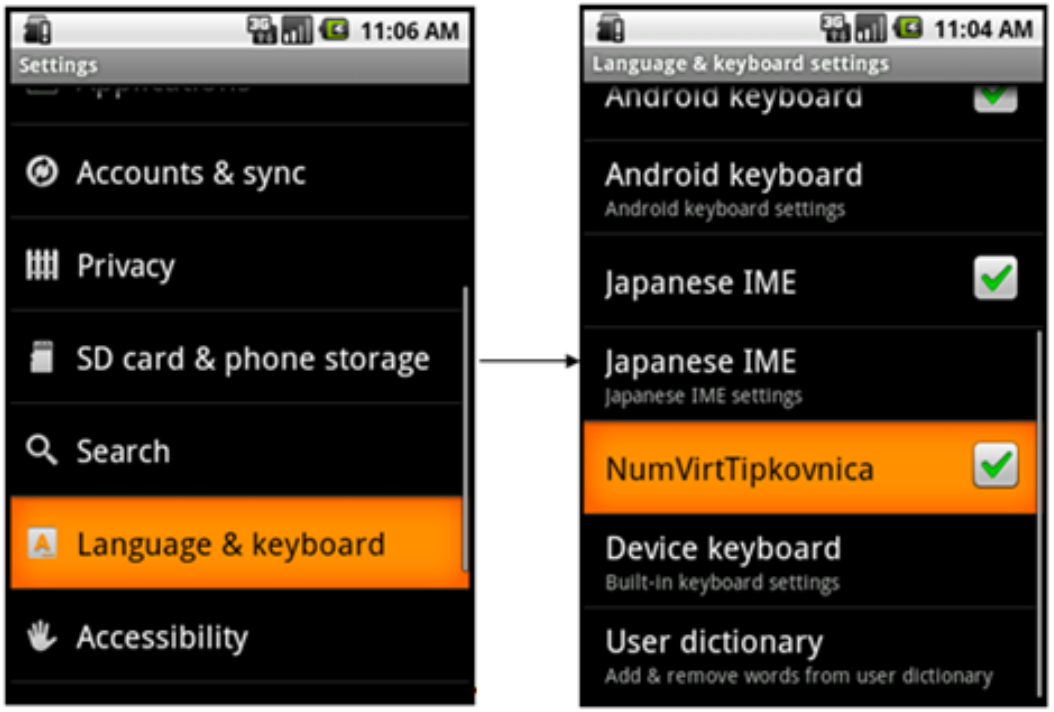
Posebnu pažnju pridaje se metodama koje se nalaze u biblioteci tipkovnice i specifične su za navedenu klasu (Tabela 4.8). Neke metode nisu implementirane zbog mogućnosti zbunjivanja korisnika, što bi uvelike umanjilo funkcionalnost tipkovnice.

Prototip ugrađene metode	Opis zadaće
`public void swipeLeft()`	Preusmjerava se na tipku DEL
`public void swipeRight()`	Omogućuje brzu izmjenu između razvijene tipkovnice i standardne "qwerty" tipkovnice
`public void swipeUp()`	Zadaće ove funkcije nije implementirana zbog praktičnosti
`public void swipeDown()`	Zatvara tipkovnicu
`public void onRelease(int primaryCode)`	Zadaće ove funkcije nije implementirana zbog praktičnosti
`public void onPress(int primaryCode)`	Zadaće ove funkcije nije implementirana zbog praktičnosti

Tabela 4.8 Metode klase VirtualKeyboard - specifične ugrađene metode

4.3. Izlazni proizvod

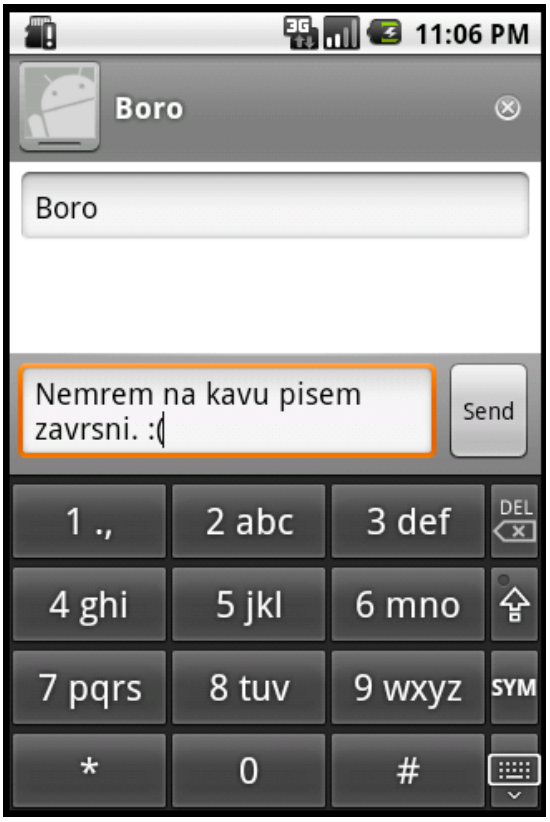
Implementacijom XML datoteke, prikaza, metoda i funkcija iz prethodnih poglavlja te definiranjem njihovih zadaća stvoren je konačni proizvod. Tipkovnica se uključuje za korištenje izbornikom *Settings* (Slika 4.2).



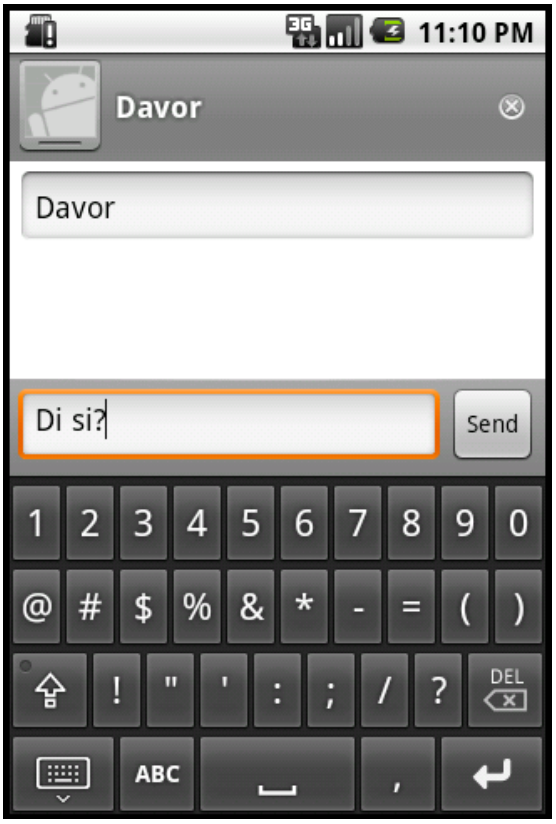
Slika 4.2 Izbor tipkovnice

Nakon uključivanje tipkovnice ona je spremna za korištenje. Prilikom korištenja virtualna numerička tipkovnica uspješno emulira stare tipkovnice uz dodatke modernih tipkovnica (Slika 4.3). Uz navedenu tipkovnicu moguće je pristupiti i standardnoj "qwerty" tipkovnici potezom prsta u desno po tipkovnici, a povratak na emuliranu tipkovnicu se odvija na jednak način. Tipka SYM omogućuje prijelaz u simboličku tipkovnicu kakva postoji na standardnim mobilnim uređajima s Android operacijskim sustavom (Slika 4.4).

Uspješno je izvršen zadatak zadan ulaznim zahtjevima. Krajnji proizvod nalazi se u privitku ovog rada na CD-u, zajedno s razvojnim okruženjem, potrebnim dodacima i literaturom.



Slika 4.3 Numerička virtualna tipkovnica



Slika 4.4 Simbolička tipkovnica

5. Diskusija

Razvijen proizvod uspio je ispuniti ulazne zahtjeve ovog rada, ali uvijek postoji prostor za poboljšanje. Pravu ocjenu aplikacije može dati jedino grupa korisnika nakon testiranja te detaljna analiza njihovih zahtjeva i zamjerki.

Mogućnost prijelaza iz numeričke virtualne tipkovnice u standardnu "qwerty" tipkovnicu u bilo kojem trenutku jednostavnim pokretom po ekranu osjetljivom na dodir značajno poboljšava funkcionalnost tipkovnice. Korisnik može izabrati način korištenja tipkovnice ovisno da li se nalazi u pokretu, kada je preporučljivo koristiti numeričku tipkovnicu zbog preciznosti, ili kada se ne kreće, kada može koristiti standardnu tipkovnicu.

Najveći nedostatak virtualne numeričke tipkovnice su funkcionalnosti tipki "*****" i "**#**". Kod starih mobilnih uređaja navedene tipke su stvarale posebne znakove ili su bile u funkcije tipke SHIFT. Na numeričkoj virtualnoj tipkovnici one služe za unos posebnih znakova, ali je problem što korisnik ne može unaprijed znati koji znak upisuje te koji slijedi. Ovaj problem bi se mogao riješiti implementacijom navedene funkcije unutar `keyDownUp` metode, ali se tada javlja problem preglednosti. Moguće rješenje ovog problema je uklanjanje navedenih tipki i postavljanje dodatnih tipki za prijelaze u različite simboličke tipkovnice.

Položaj tipke SPACE koje se provlačilo godinama kod proizvođača mobilnih uređaja također je postavljao dilemu. Odlučeno je da se ona nalazi unutar tipke "0" kao što je to bilo na starijim Nokia mobilnim uređajima, točnije Nokia 3310 modelima.

Funkcionalnosti modernih tipkovnica numeričke virtualne tipkovnice sadržani su u četvrtom stupcu tipkovnice (Slika 4.3). Ovakvom implementacijom korisničkog sučelja korisnik neće ostati zakinut za sve blagodati koje nude moderne tipkovnice uređaja s Android operacijskim sustavom.

Uz pomoć implementacije virtualne tipkovnice dane ovim radom moguće su jednostavne preinake i nadogradnje prema što funkcionalnijoj i korisniku ugodnijoj tipkovnici.

Zaključak

U okviru završnog rada ostvarena je virtualna numerička tipkovnica za uređaje s Android operacijskim sustavom prilagođena ulaznim zahtjevima. Aplikacije je stovrena pomoću Eclipse razvojnog okruženja uz pomoć posebnih alata za razvoj Android aplikacija. Konačni izlazni proizvod rada je kriptirana datoteka koji služi za pokretanje navedene aplikacije na uređaju s Android operacijskim sustavom.

Tijekom razvoja aplikacije usvojena su vrijedna znanja objektno orijentiranog programiranja, razvoja Android aplikacija i korištenja različitih programskih alata. Zadatak je bio intrigantan, ali zahtijevan. Korištenje primjera i razne literature uvelike su pomogli u razumijevanju danog zadatka i njegovom izvršenju.

Čitatelju ovog rada dan je uvod u programiranje Android aplikacije i navedeni su svi potrebni alati za razvoj navedenih aplikacija. Na temelju postojeće aplikacije, numeričke virtualne tipkovnice, čitatelj ovog rada može lako izgraditi svoju vlastitu virtualnu tipkovnicu za uređaje s Android operacijskim sustavom. Moguće su preinake na postojećoj tipkovnici, ovisno o zahtjevima te dodavanje novih funkcionalnosti i nadogradnje na postojećem kosturu.

Na CD-u koji prilažem kao privitka uz ovaj završni rad nalaze se sljedeće datoteke i direktoriji: direktorij s konačnom aplikacijom, Eclipse razvojni sustav verzija 3.6 (Helios), Android SDK alati, literatura, prezenatacija i dokument završnog rada.

Literatura

[1] APRESS. *Beginning Android 2*. New York. Mark L. Murphy, 2010.

[2] THE PRAGMATIC PROGRAMERS. *Hello, Android 3rd edition*. Dallas. Ed Burnette, 2010.

[3] WILEY PUBLISHING, INC. *Proffesional Android 2 Application Development*. Indianapolis, Reto Meier

[4] WIKIPEDIA, http://en.wikipedia.org/wiki/Android_(operating_system)

[5] ANDROID DEVELOPERS, http://developer.android.com/index.html

[6] ECLIPSE, http://www.eclipse.org/

[7] ANDROID CENTRAL, http://www.androidcentral.com/

Skraćenice

IDE	engl. *Integrated Development Environment*	integrirano razvojno okruženje
ADT	engl. *Android Development Tools*	Android alati za razvoj
SDK	engl. *Software Development Kit*	alati za razvoj programske podrške
AVD	engl. *Android Virtual Device*	virtualni emulator Android uređaja
GUI	engl. *Graphic User Interface*	korisničko grafičko sučelje
LCD	engl. *Liquid Crystal Display*	prikaz tekućim kristalima
USB	engl. *Universal Serial Bus*	univerzalna serijska sabirnica
XML	engl. *Extensible Markup Language*	jezik s dodatnim označavanjem
IME	engl. *Input Method Editor*	stvaratelj načina unosa

Sažetak

Numerička virtualna tipkovnica pod Android operacijskim sustavom

U ovom radu opisan je postupak instalacije i postavljanja svih potrebnih alata za razvoj mobilnih aplikacija za mobilne uređaje koji koriste Android operacijski sustav. Objašnjeni su temeljni dijelovi Android aplikacija te njihova funkcija. Korisnik pomoću ovog rada i predložene literature može lagano započeti s razvojem vlastitih Android aplikacija.

Glavna tema rada je razvoj numeričke virtualne tipkovnice pod Android operacijskim sustavom. Temeljni ulazni zahtjevi su funkcionalnost i jednostavnost. Razvijena tipkovnica omogućuje korisniku jednostavniju upotrebu tipkovnice na mobilnom uređaju s Android operacijskim sustavom. Tipkovnica udružuje funkcionalnost tipkovnica starijih mobilnih uređaja i novih "qwerty" tipkovnica

Unutar ovog dokumenta opisan je proces dizajniranja virtualne tipkovnice prema zadanim zahtjevima.

Ključne riječi: Virtualna tipkovnica, Android operacijski sustav, virtualni emulator, razvojno sučelje, grafičko korisničko sučelje, mobilna aplikacija, mobilni uređaj

Summary

Virtual numeric keyboard developed for Android operating system

In this project we describe the process of installing and setting up all the necessary tools to develop mobile applications for mobile devices using the Android operating system. It explains the fundamental parts of Android applications and their functions. The reader of this document can easily start to develop their own Android applications, with the help of the suggested literature.

The main theme of document is the development of a virtual numeric keyboard developed under the Android operating system. The basic entry requirements are functionality and simplicity. This developed keyboard allows the user to easily use the keyboard on their mobile device which uses an Android operating system. The keyboard combines the functionality of the old mobile devices keyboards and new "qwerty" keyboards.

This document describes the process of virtual keyboard development suited to match specific user demands.

Key words: Virtual keyboard, Android operating system, virtual emulator, development interface, graphical user interface, mobile application, mobile device