ANDROID APLIKACIJA ZA PRIKAZ I ANALIZU PODATAKA MJERENJA

Bajivić, Antonio

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:023552

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-16**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij računarstva

ANDROID APLIKACIJA ZA PRIKAZ I ANALIZU PODATAKA MJERENJA

Završni rad

Antonio Bajivić

Osijek, 2019.

SADRŽAJ

1. UVOD……..1

1.1. Zadatak završnog rada 1

2. OPIS KORIŠTENIH TEHNOLOGIJA 3

2.1. Operacijski sustav Android 3

2.2. Android Studio 4

2.3. Programski jezik Java 5

2.4. XML 6

2.5. Firebase 7

3. RAZVOJ APLIKACIJE 9

3.1. Stvaranje projekta 9

3.2. Funkcionalnosti aplikacije 12

4. ZAKLJUČAK 22

5. LITERATURA 23

6. SAŽETAK 25

7. ABSTRACT 26

8. ŽIVOTOPIS 27

9. PRILOZI……...28

1. UVOD

Razvojem automobilske industrije i tehnologije općenito, ističe se potreba za automatizacijom određenih proračuna kako se na tome ne bi gubilo vrijeme. Pritom se misli na to kako je danas brzina, uz učinkovitost, ključ uspjeha. Svakoj automobilskoj tvrtki cilj je napraviti bolje vozilo od konkurentnog, no prije puštanja proizvoda na tržište moraju se analizirati i testirati određene mogućnosti. Jedno od takvih testiranja koje vozilo mora proći je test na dinamometru [1]. Tim se ispitivanjem saznaje koliko vozilo ima konjskih snaga, koja mu je maksimalna brzina, okretni moment i slično. Dinamometar je mjerni instrument koji služi za mjerenje sila. Problem s kojim se automehaničarske radnje susreću je financijski. Takvi instrumenti koštaju po nekoliko desetaka tisuća kuna, a razlog takvoj cijeni je što se uz fizički dio dobije i elektronički. Ovaj će se problem u radu pokušati riješiti idejom o mogućem umanjivanju cijene dinamometra ako se napravi mobilna aplikacija koja bi bila dostupna svima pod pretpostavkom da svi automehaničari imaju mobilni uređaj koji koristi Android operacijski sustav. Cilj aplikacije je provođenje proračuna koje bi inače provodilo računalo ugrađeno u strukturu dinamometra.

U drugom poglavlju opisane su tehnologije korištene u izradi aplikacije. Razne prednosti pojedinačne tehnologije su iskorištene i zatim su se te prednosti povezale kako bi dobili rezultat koji se tražio. U sljedećem poglavlju opisan je proces izrade aplikacije i najvažnije funkcionalnosti aplikacije. S obzirom da se prednosti svakog alata za izradu ove aplikacije treba spojiti u kompoziciju treće poglavlje je opisalo pojedinosti korištenja tih alata te spajanje istih. Četvrto poglavlje zaključuje ovaj rad te objašnjava zašto je ovo rješenje jednostavnije i smislenije. Nakon što se sazna kako aplikacija funkcionira jednostavnije je shvatit potrebu za ovim načinom rješavanja problema.

1.1. Zadatak završnog rada

Zadatak ovog rada je napraviti Android aplikaciju koja će izvoditi fizičke i matematičke račune [2] potrebne za izračun konjskih snaga čime će se izbaciti potreba korištenja računala na dinamometru te tako napraviti dinamometar pristupačniji svima. Aplikacija će imati mogućnosti poput unošenja teksta, višestrukog odabira (npr. materijal testnog valjka, vrstu pogona i drugo), gumbove za izračune i prebacivanje na sljedeće aktivnosti, a prilikom korištenja baze podataka prikazuje se graf za određeno vozilo na kojemu se nalaze podaci mjerenja, slika vozila. Tekstualni okviri će služiti ili za opis aktivnosti koja se mora napraviti u određenom dijelu tog procesa ili za

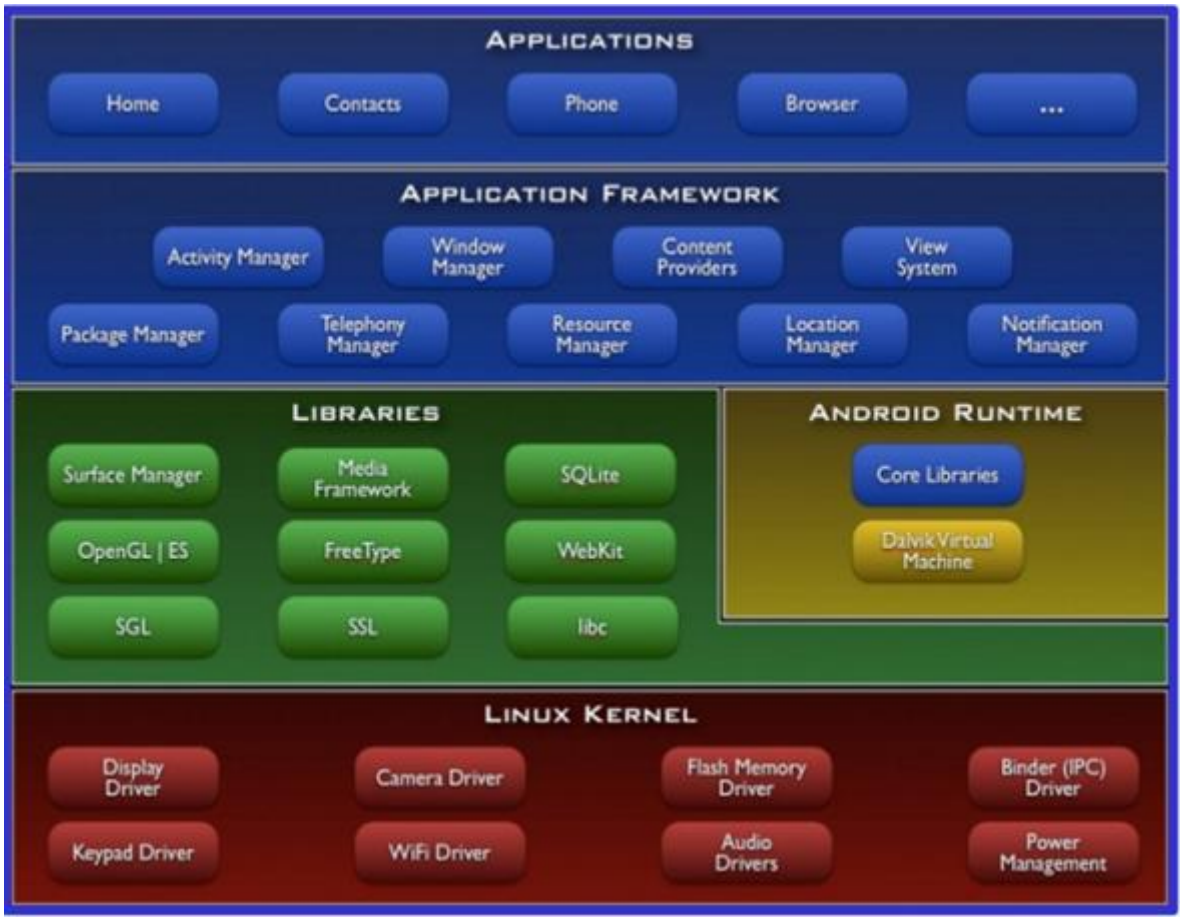
izbacivanje rezultata korak po korak. Radi jednostavnosti aplikacija neće biti u stvarnom vremenu provođenja mjerenja zbog nepristupačnosti dinamometra ili makete.

2. OPIS KORIŠTENIH TEHNOLOGIJA

Korištene tehnologije za izradu ove aplikacije bit će objašnjene i opisane u ovom poglavlju. Najprije je opisan rad operacijskog sustava Android, zatim korištenje Android Studio razvojnog alata i programski jezik Java te uporaba XML-a.

2.1. Operacijski sustav Android

Razvojem mobilnih uređaja nastajala je sve veća potreba za stvaranje sustava koji bi dodatno olakšao korištenje takvih uređaja uz neke nove mogućnosti. Android Inc. je osnovana 2003. godine s namjerom razvijanja programa za pametne mobilne uređaje. Njihovi programi su uzimali u obzir korisnička preferiranja i lokaciju. Tvrtka se 2005. godine približila propasti, ali ju preuzima Google. Koristeći tu priliku Google je otvorio vrata prema tehnologiji za mobilne uređaje. Kako bi prednjačili pred svojim konkurentnima Google tvrtka se dosjetila kako bi se trebao razviti javni standard za mobitele, tablete i slično, te su tako inicirali osnivanje konzorcija *Open Handset Alliance (OHA).* Prilikom osnivanja udruzi su pristupile brojne utjecajne firme iz tog područja tehnologija, a Android je u konačnici predstavljen javnosti. Android je tada predstavljen kao operacijski sustav, da bi godinu dana kasnije bio ugrađen na prve uređaje. Predstavljen je kao mobilna platforma otvorenoga koda koji je zasnovan na Linux 2.6 jezgri. Prvi takav uređaj bio je T-Mobile G1 izrađen iz tajvanske firme proizvođača pametnih telefona HTC. Iako je bio otvorenog koda, Google proizvođačima nije dopuštao korištenje Android zaštićenog imena dok se uređaj ne ispostavi kompatibilnim prema *Compatibility Definition Document (CDD)* normi. Ova platforma je pisana u C/C++ jeziku, ali je većina aplikacija napisana programskim jezikom Java. Kako bi mobilne aplikacije pisane Javom bile ostvarive, morale su se koristiti Android razvojne programske komponente (engl. *Android Software Development Kit,* SDK). Kako bi se pokretale Java aplikacije bez prethodnog postojanja *Java Virtual Machine,* Android je koristio virtualni stroj Dalvik, a od inačice 5.0 koristi se *Android Runtime* (ART) [5]. Isticanje Google-ovih proizvoda ističe se i u samoj mrežnoj trgovini Androida nazvana Google Play. Mrežna trgovina služi za pronalazak aplikacija koje korisnik preuzima, a ovisno o izdavatelju one su besplatne ili se prethodno plaćaju. Arhitektura Android sustava [3] razlaže se na više razina (***Sl.2.1***), a prva je Linux razina s potrebnim driverima. Slijedi druga razina gdje su knjižnice pisane u C/C++ programskom jeziku.



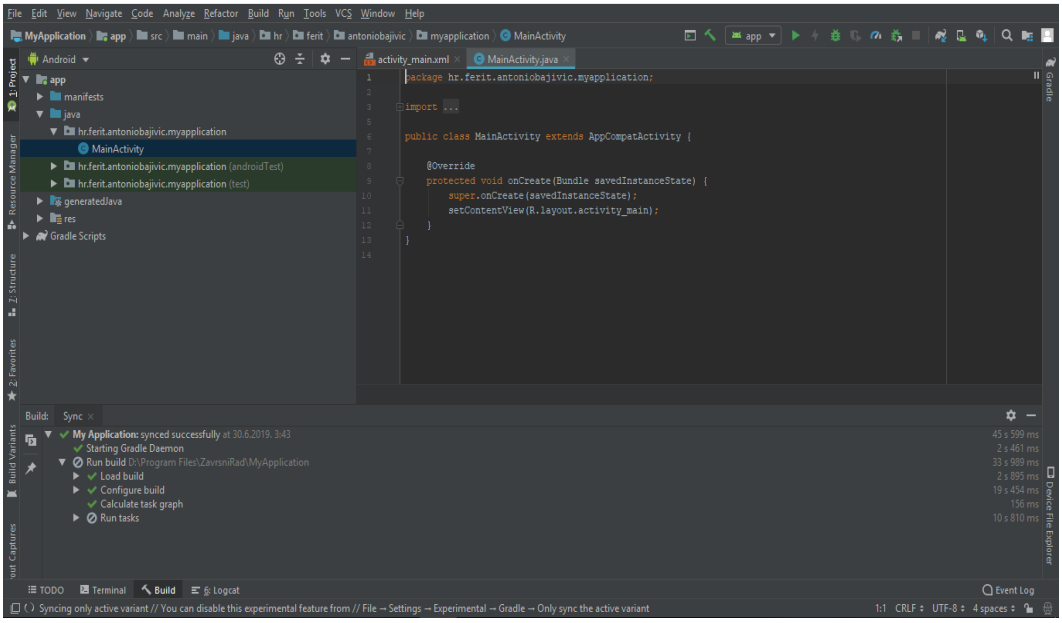
Sl.2.1. *Arhitektura Android sustava*[1]

Sljedeća razina je sloj koji pokreće aplikacije, točnije Android Runtime. Iznad toga dolazi aplikacijski okvir s mehanizmima koji ubrzavaju i olakšavaju pisanje aplikacija.

2.2. Android Studio

Android Studio (***Sl.2.2***) je službeno integrirano razvojno okruženje (engl. *Integrated development environment*, IDE) namijenjeno za izradu aplikacije koje koriste Android operacijski sustav.

[1] Izvor: <https://informatika.buzdo.com/pojmovi/mobile-3.htm> (svibanj, 2019.)



Sl.2.2. *Započinjanje projekta u Android Studio*

Izgrađen je na temelju *JetBrains' IntelliJ IDEA* programskoj podršci (softveru) specifično dizajniranoj za razvijanje *Android* aplikacija. Objavljen je na *Google*-ovoj konferenciji razvojnih inženjera. Prva stabilna inačica objavljena je 2014. godine[8]. Dostupan je i na drugim operacijskima sustavima. Neke od boljih značajki su specifično refaktoriranje i brzi popravci, čarobnjaci na temelju predloška za stvaranje uobičajenih Android-ovih dizajna i komponenti, poseban emulator (*Android Virtual Device*) za pokretanje i uklanjanje pogrešaka u aplikacijama unutar *Android Studija*. Kako bi se lakše refaktorirao ili popravio kôd nude se informacije za ispravak unutar linije kôda koja se trenutno piše te izbacuje određene probleme. Neki od problema ili potencijalnih opasnosti može biti upućivanje objekata na koje se odnosi odabrani objekt kako programer ne bi pogriješio u korištenju koncepta nasljeđivanja. Takav način upozorenja može izbaciti informaciju o povratnoj vrijednosti metode, lambdi i izrazu operatera, opisnih vrijednosti. Također se unutar *Android Studija* nudi profiliranje performansi kako bi se moglo pratiti kako računalo reagira na izradu aplikacije [8].

2.3. Programski jezik Java

Inženjeri iz tvrtke *Sun Microsystems* su 1991. godine započeli s razvojem ovog programskog jezika. James Gosling i Patrick Naughton u prvom redu su zaslužni za stvaranje *Jave*. Počeo je kao dio projekta Green i umalo je ostao imenovan kao *Oak* zbog hrasta kojeg je James gledao kroz prozor svoga ureda. Objavljen je pod konačnim imenom *Java* u studenom 1995. kao objektno

orijentirani programski jezik[9]. Kod *Jave* se kôd nalazi u klasama. Prednost *Jave* u ono vrijeme bilo je izvođenje programa bez ikakve preinake, ako korisnikov operacijski sustav sadrži *Java Virtual Machine* (*JVM*). Međutim, to je predstavljalo problem drugim jezicima, npr. *C*. Taj programski jezik ubraja se u skupinu viših programskih jezika što znači da su jednostavniji za čitanje, pisanje i programiranje u višim programskim jezicima. Baš zato što je *Java* programski jezik jednostavan za naučiti i jednostavniji za uporabu *Java* je danas jedan od najzastupljenijih jezika za pisanje *Android* aplikacija. Ovaj je jezik objektno orijentiran i ne ovisi o platformi što znači lakše kretanje među računalnim sustavima. Ukoliko *Java* nije instalirana na suvremenim računalima postoji mogućnost da se ne učitaju animacije, aplikacije ili čak internetske stranice. Školski primjer za ispis teksta se sastoji od šest linija koda (***Sl.2.3***).

Sl.2.3. *Primjer kôda za funkciju ispisa*[2]

2.4. XML

XML je kratica za *EXtensible Markup Language* što znači da nije klasični programski jezik nego jezik za označavanje, a u ovom slučaju proširivi jezik za označavanje. Nastao je kao potreba za stvaranjem jezika koji će biti čitljiv i razumljiv ljudima, ali i računalnim programima. Zbog želje za jednostavnošću kod je pisan unutar oznaka koje označavaju početak elemenata <> i završetak elementa </>[11]. Upravo zbog tog formata nalik je *HTML*-u. *XML* dokument se sastoji od zaglavlja i sadržaja dokumenta koji je omeđen *XML* oznakama [3]. Sličnost s *HTML*-om je velika, ali kod proširivog jezika za označavanje nudi mogućnosti stvaranja oznaka koje ga i čine „proširivim“ (***Sl.2.4***) .

[2] Izvor: <https://examples.javacodegeeks.com/desktop-java/ide/eclipse/eclipse-tutorial-beginners/> (svibanj, 2019.)

```xml
<?xml version="1.0"?>
<CAT>
  <NAME>Izzy</NAME>
  <BREED>Siamese</BREED>
  <AGE>6</AGE>
  <ALTERED>yes</ALTERED>
  <DECLAWED>no</DECLAWED>
  <LICENSE>Izz138bod</LICENSE>
  <OWNER>Colin Wilcox</OWNER>
</CAT>
```

Sl.2.4. *Primjer XML kôda*[3]

Neke od mogućih upotreba *XML*-a su: razmjena, pohrana i povećanje dostupnost podataka. *XML* se često primjenjuje u definiranju matematičkih formula, lakši je za opis tvrdnji i poveznica s tvrdnjama, kao jezik za integraciju električkih sustava (*CIM*) [11].

2.5. Firebase

Firebase je platforma koju je razvila tvrtka *Firebase Inc.*, no *Google* ju preuzima 2014. godine. Ova platforma nudi 18 usluga, a neke od njih su analitičke usluge, baza podataka u stvarnom vremenu, pohrana podataka (npr. slike, videozapis), usluge autentifikacije i mnoge druge[13]. Preko 1.5 milijun aplikacija ju koriste jer se mogu koristiti i za *Android* i *iOS* mobilne aplikacije i internetske stranice. Jednostavno korištenje *Firebase* usluge zahtijeva pristup Internetu jer se svi podaci nalaze na platformi. Korištenjem *Firebase* baze podataka omogućen je i unos i mijenjanje podataka u stvarnom vremenu. Baza podataka strukturirana je hijerarhijski poput stabla s određenim čvorovima koji služe kao razine (***Sl.2.5***).

[3] Izvor: <https://hr.wikipedia.org/wiki/XML> (svibanj, 2019.)

Sl.2.5. *Prikaz podataka u Firebase bazi podataka*[4]

Povezana je s uslugom autentifikacije pomoću *WebSocket*-a. Još jedna zanimljiva usluga ove platforme je *Firebase Storage* koja omogućava spremanje generiranih sadržaja poput slika ili videozapisa. Korištenjem *Google Cloud Storage*-a na kojem se spremaju podaci, omogućeno je korištenje *Firebase* i *Google Cloud*-a. Kao i prethodna usluga, povezana je s uslugom autentifikacije. Ako se netko odluči koristit provjeru autentičnosti *Firebase ML-Kit* pojednostavljuje i omogućuje postupak prijave. Istim *ML-Kit*-om je ponuđeno nešto što je karakteristično za aplikacije utemeljene na strojnom učenju (npr. prepoznavanje lica, teksta, skeniranje kodova) [15].

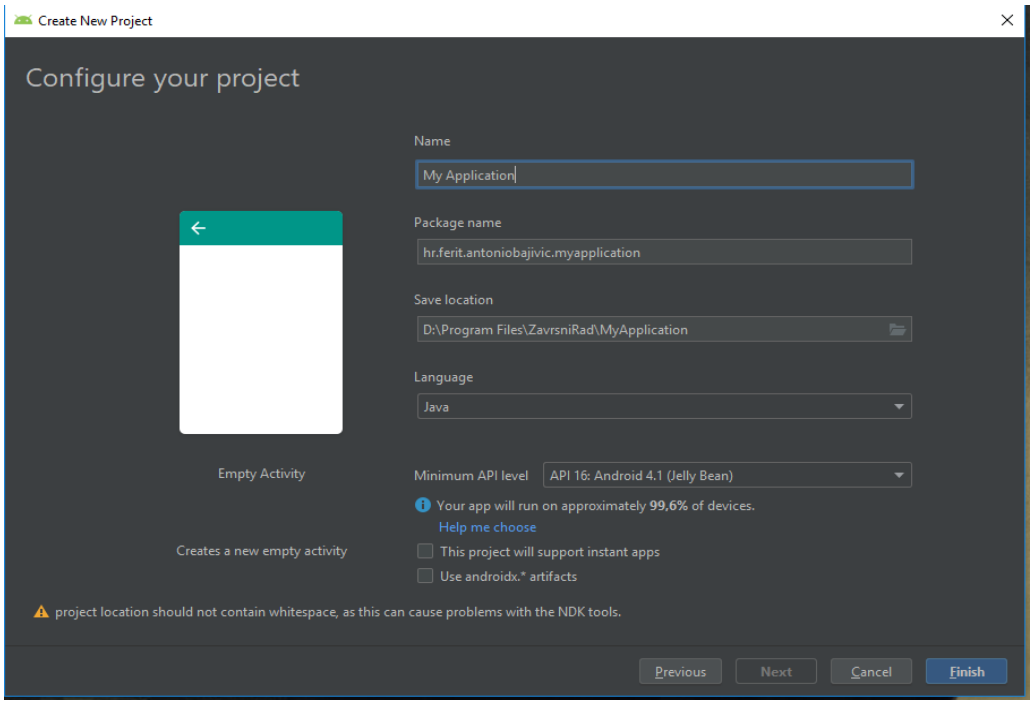
[4] Izvor: <https://firebase.google.com/docs/database/> (svibanj, 2019.)

3. RAZVOJ APLIKACIJE

Ovo poglavlje služi kako bi se opisao razvoj aplikacije za mjerenje snage izražene u ekvivalentu konjskih snaga. Ideja za ovo proizlazi iz videa u kojima se izvode testiranja na dinamometru i daljnjim informiranjem kako dinamometar funkcionira, koliko košta i zašto to toliko košta. Dolazi se do ideje kako je moguće smanjiti cijenu i pojednostaviti ovakav proces izradom aplikacije i drugačijeg dizajna dinamometra. Ovom aplikacijom pokušat će se dočarati kako se dolazi do konjskih snaga uz znanje određenih podataka o dinamometru. Ova će aplikacija zato biti nalik prototipu jer nije napravljena za obrađivanje podataka (u ovom slučaju mjerenja) u stvarnom vremenu. Kroz ovo poglavlje bit će objašnjeno programsko rješenje te koraci u izradi i implementaciji aplikacije. Aplikacija je nazvana *DynoApp*.

3.1. Stvaranje projekta

Ulaskom u *Android Studio* nude se mogućnosti stvaranja novog projekta, učitavanje projekta iz drugih izvora i druge. Kreiranjem novog projekta najčešće se odabire prazna aktivnost (engl. *Activity)* te se odabire ime prve, a ujedno i glavne aktivnosti. *Android Studio* nudi mogućnost za odabiranje razine *API*-ja te pomoć ukoliko se želi vidjeti koliko uređaja koristi koju razinu. Time se završava proces stvaranja novog projekta (***Sl.3.1***).



Sl.3.1. *Konfiguracija novog projekta*

Projekt se sastoji od tri glavna dijela: *manifests*, *java* i *res*. Struktura je zamišljena tako da se u *manifest* dijelu nalaze važne informacije ili određene potvrde koje su potrebne za izradu aplikacije. Ako aplikacija zahtijeva korištenje interneta mora se upisati određena linija kôda (***Sl.3.2***) unutar *AndroidManifest.xml* (***Sl.3.3***) datoteke kako bi se zatražila dozvola za pristup.

```
1
2  <uses-permission android:name="android.permission.INTERNET" />
3
```

Sl.3.2. *Dozvola za pristup Internetu*[5]

[5] Izvor: <https://java2blog.com/add-internet-permission-in-androidmanifest-android-studio/> (svibanj, 2019.)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.java2blog.helloworldapp">
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".HelloWorldActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

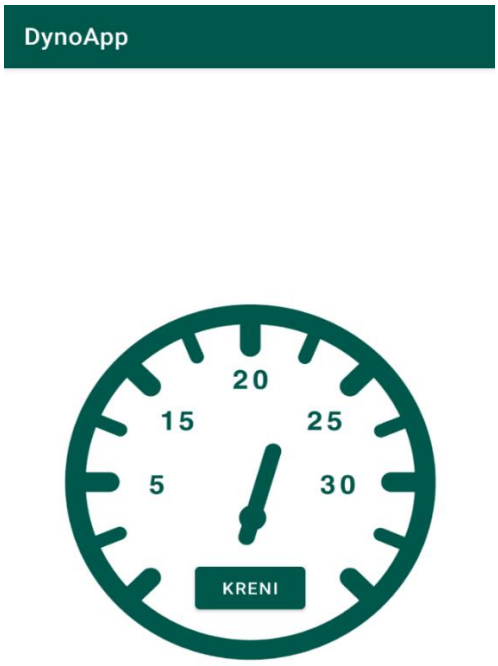
Sl.3.3. *AndroidManifest.xml sa zahtjevom za dozvolu pristupa internetu*[6]

Prateći strukturu projekta slijedi *java* dio u kojem se nalazi kôd pisan u Java jeziku. U ovom dijelu mogu se vidjeti klase koje su potrebne za određenu aktivnost i metode (funkcije) koje obavljaju određene zadatke potrebne za izračune, a uz to prebacivanje na sljedeći ekran te prosljeđivanje podataka. Kako sada slijedi *res* dio, što je zapravo skraćeno od *resources* (hrv. resursi), uočava se kako se u tom dijelu zapravo spremaju podaci poput nekih rečenica (programski: tip podataka string), slika i slično. Koristeći resurse lakše je nešto mijenjati ako se ode pod tu kategoriju projekta bez da se mora puno pretraživati po određenim *XML* datotekama. *XML* datoteke služe kao datoteke rasporeda gdje se dodaju funkcionalni dijelovi koji će biti prikazani. Nastavi li se pratiti struktura, uočava se značajka nazvana *gradle*. Ukoliko se želi dodati određena „biblioteka“ u aplikaciju to se može napraviti ovdje. Za rad ove aplikacije trebat će implementirati dodatne biblioteke uz standardno dobivene od *Android Studija*. Biblioteke u programiranju služe kako bi se po potrebi dodale određene nadogradnje ili usluge, a ovom radu će poslužiti biblioteka za bazu podataka i sve funkcije koje idu s njom. U ovom slučaju to su Java klase, iako postoje biblioteke i u drugim programskim jezicima.

[6] Izvor: <https://java2blog.com/add-internet-permission-in-androidmanifest-android-studio/> (svibanj, 2019.)

3.2. Funkcionalnosti aplikacije

Prva aktivnost s kojom se korisnik susreće (***Sl.3.4***) je *activity_main* (***Sl.3.5***) u XML datoteci ili *MainActivity* (***Sl.3.6***).



Sl.3.4. *Prikaz sučelja - prva aktivnost*

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView android:id="@+id/image_view_speedometer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/speedometer"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="@id/fab_next"
        app:layout_constraintEnd_toEndOf="@id/fab_next"
        android:layout_marginBottom="25dp"
        tools:ignore="ContentDescription" />

    <com.google.android.material.button.MaterialButton android:id="@+id/fab_next"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/kreni"
        app:layout_constraintTop_toTopOf="@id/image_view_speedometer"
        app:layout_constraintBottom_toBottomOf="@id/image_view_speedometer"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="180dp"/>
```

Sl.3.5. *XML kôd za prvu aktivnost*

```java
public class MainActivity extends BaseActivity {

    @Override
    protected void setupUi() {
    }

    @Override
    protected int getLayout() { return R.layout.activity_main; }

    @Override
    protected void onNextClick() { goTo(CylinderActivity.class); }

}
```

Sl.3.6. *Java kôd za prvu aktivnost*

Dohvaćanje elementa gumba omogućeno je pomoću određene identifikacijske oznake i uočljiva je metoda koja se koristi za prebacivanje na sljedeću aktivnost; *CylinderActivity()*. Prikazano je kako *MainActivity* (***Sl.3.6***) koristi *BaseActivity*. *BaseActivity* (***Sl.3.7***) je temeljna klasa koja će predstavljati prototip svake aktivnosti te metode koje se ne moraju svaki puta iznova ponavljati unutar svake aktivnosti.

```
public abstract class BaseActivity extends AppCompatActivity {

    //region BIND
    @Optional
    @OnClick(R.id.fab_next)
    protected void onNextClick() {}
    //endregion

    @Override
    public final void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(getLayout());
        ButterKnife.bind(this);
        setupUi();
    }

    protected abstract void setupUi();

    @LayoutRes
    protected abstract int getLayout();

    protected final void goTo(Class<? extends BaseActivity> activityClass, Parcelable ... parcelables) {
        Intent intent = new Intent(this, activityClass);
        for(Parcelable parcelable : parcelables) {
            intent.putExtra(parcelable.getClass().getSimpleName(), parcelable);
        }
        startActivity(intent);
    }
}
```

Sl.3.7. *BaseActivity*

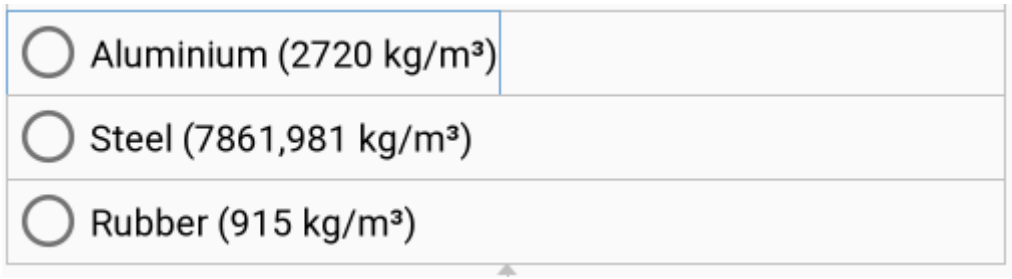
Unutar klase *BaseActivity* klase inicijaliziraju se određene metode kao što su *setupUI*, *getLayout* koja je zadužena za dohvaćanje rasporeda. Metoda koja je potrebna svakoj aktivnosti za prelazak u novu aktivnost je *goTo* metoda. Zadnja metoda, koja se nalazi u *ButterKnife* biblioteci, dodatno sprema vezano označene (engl. *bind*) metode, koristeći znak „@“. *ButterKnife* biblioteka olakšava raspoređivanje, u ovom slučaju *OnClick* metode, kako bi se smanjilo dupliciranje koda na svakoj aktivnosti. Metoda *goTo* od svake aktivnosti prima naziv klase te koristeći *Parcelable* sučelje prebacuje *Datu*, a određene podatke koje spremimo kao važne i prenosive, na sljedeću aktivnost. Za provjeru je li korisnik nešto unio na svakoj aktivnosti napravljeno je sučelje *IRule* u kojem se nalazi metoda *isValid* pomoću koje se provjerava može li se nastaviti dalje s izračunima i prebacivanjem na drugu aktivnost.

Na sljedećoj aktivnosti korisnika se traži unos teksta koji se ostvaruje s *EditText* elementom. Takav element, uz standardne opcije za dimenziju, može birati tip podatka za unošenje (broj ili tekst) te se uz pomoć nagovještaja korisnika upućuje što treba unijeti. Koristeći *TextView* priložen je kratki opis što se na aktivnosti odvija i što je korisnikov zadatak. Od korisnika se u ovom slučaju traži unos radijusa i visine valjka koji se nalazi na korisničkome dinamometru u metrima. Pritiskom na gumb *Izračunaj* povlače se unosi iz *EditTexta* te se koristi formula 3.1.

(3.1.) $$V = r^2 * h * \pi \, [m^3]$$

Nakon što se korisniku u tekstualnom okviru prikaže rezultat ponuđen je gumb koji je smješten pod ID *fab_next* pomoću kojeg se prelazi na novu aktivnost. FAB je skraćenica od *Floating Action Button* što u prijevodu znači plutajući akcijski gumb.

Sljedeća aktivnost zadužena je za izračun mase (3.2,) valjka. Na ovoj aktivnosti pojavljuje se novi element koji se zove *RadioGroup* (***Sl.3.8***).



Sl.3.8. *Prikaz RadioGroup elementa*

RadioGroup je element koji se sastoji od više *RadioButton*-a (***Sl.3.9***), odnosno korisniku se nudi višestruki odabir uz kratki opis zašto je uopće ponuđen višestruki odabir.

```
<RadioGroup
    android:id="@+id/radio_group_materials"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_marginTop="16dp"
    app:layout_constraintTop_toBottomOf="@id/text_view_material_title"
    app:layout_constraintStart_toStartOf="@id/text_view_material_title"
    app:layout_constraintEnd_toEndOf="parent">

</RadioGroup>
```

Sl.3.9. *XML kôd za RadioGroup*

Korisnikov je zadatak na ovoj aktivnosti odabrati jedan od ponuđena tri materijala na koji se prislanjaju gume vozila zbog formule za izračun mase (3.2.) Korisnik odabire između aluminija, čelika i gume s njihovim gustoćama u opisu odabira. Za izračun mase se također pritišće gumb *Izračunaj* (***Sl.3.10***) te se obavlja više funkcionalnosti.

```
private void calculateMass() {
    this.mass = MaterialType.values()[materialIndex].density * cylinderData.getVolume();
}
```

Sl.3.10. *Java kôd za funkciju Izračunaj*

Koristeći *enum*[7] tip klase (***Sl.3.11***.), konstante gustoća ovih materijala spremljene su u istu.

```
public enum MaterialType {

    ALUMINIUM( density: 2720,  name: "Aluminij"),
    STEEL( density: 7861.093,  name: "Čelik"),
    RUBBER( density: 915,  name: "Guma");

    public final double density;
    public final String name;

    MaterialType(double density, String name) {
        this.density = density;
        this.name = name;
    }

}
```

Sl.3.11. *Enum tip klase za vrstu materijala valjka*

[7] Enum tip klase predstavlja posebnu klasu koja sadrži skup konstanti

Unutar metode za izračun mase (***Sl.3.10***.) pristupa se toj klasi i dohvaćaju se vrijednosti kako bi se u konačnici izračunala masa.

Prije nego se korisnik prebaci na ovu aktivnost aplikacija je za zadatak imala prenijeti rezultat volumena koji se prethodno izračunao. Aktivnost kojoj je poslana neka vrijednost također ga na poseban način mora deklarirati u svojoj Java datoteci. Nakon klika na *Izračunaj,* aplikacija za zadatak ima otkriti koji je *RadioButton* korisnik odabrao unutar *RadioGroup* kako bi se znalo koju gustoću će koristiti u formuli.

(3.2.) $$m = \rho * V \text{ [kg]}$$

Nakon što se korisniku prikaže masa valjka u mjernoj jedinici kilogrami, pritiskom na gumb *fab_next* odlazi se na sljedeću aktivnost gdje se predaje masa jednog valjka. Ovdje se iz starih aktivnosti pod *parcelable*[8] predaje masa i radijus valjka. Korisnik ponovno ima višestruki odabir gdje bira pogon vozila koji je ključan za izračun momenta inercije valjka (3.2-3). Ponovno, koristeći *enum* tip klase, postoje vrijednosti svakog pogona spremljene u određene konstantne varijable. Formula kaže kako će se izračunati moment za jedan valjak, što bi značilo da aplikacija funkcionira samo za motocikle, a to bi uvelike ograničilo mogućnosti automehaničarskim radionicama. Dakle, nakon odabira pogona za testno vozilo i pritiskom na gumb *Izračunaj* aplikacija je ponovno treba provjeriti koji je *RadioButton* odabran te će s odabranim brojem (oznaka n; 3.3.) množiti moment inercije jednog valjka. Ponovno postoji gumb za napuštanje trenutne aktivnosti za daljnji postupak u mjerenju.

(3.3.) $$I = (\frac{m}{2} * r^2) * n \, [kg/m2]$$

U svrhu „prototipne“ realizacije ove ideje korisniku će se ponuditi odabir tri marke vozila, ovisno o tome što je odabrao na prethodnoj aktivnosti za pogon vozila. Svi podaci o vozilima, pogonu nekog vozila i drugo, bit će spremljeno u *Firebase* bazu podataka. S obzirom na to da se moraju uvesti pristupne metode za *Firebase,* koriste se određene linije koda (***Sl.3.12***).

[8] Parcel predstavlja kontejner podataka koji mogu biti proslijeđeni kroz klase, a Parcelable sučelje sadrži metode za rukovanje odabranim podacima

```
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
```

Sl.3.12. *Linije za uvoz Firebase podataka*

Uvođenjem reference na kolekciju podataka iz baze, postavljaju se podaci tako da se dohvaća sve potrebno za pristupanje slici i provjerava se jesu li jednaki s odabranim (***Sl.3.13***).

```
private CollectionReference collectionReference = FirebaseFirestore.getInstance().collection("Vehicles");

@Override
protected void setupUi() {
    brandData = new BrandData(
            (DriveData) getIntent().getParcelableExtra(DriveData.class.getSimpleName())
    );
    collectionReference.whereEqualTo(
            "driveType",
            DriveType.values()[
                    brandData.getDriveData().getDriveTypeIndex()
                    ].multiplier
    )
            .get()
            .addOnSuccessListener(queryDocumentSnapshots -> {
                if (queryDocumentSnapshots != null) {
                    List<BrandType> brands = new ArrayList<>();
                    Map<String, Object> data;
                    for (QueryDocumentSnapshot document : queryDocumentSnapshots) {
                        data = document.getData();
                        brands.add(
                                new BrandType(
                                        (String) data.get("name"),
                                        Integer.valueOf(data.get("driveType") + ""),
                                        Double.valueOf(data.get("rpm") + ""),
                                        (String) data.get("imageUrl"),
                                        (String) data.get("graphUrl")
                                )
                        );
                    }
                    setRadioGroup(brands);
                }
            });
```

Sl.3.13. *Uvođenje reference na kolekciju i postavljanje tipa vozila*

Za ovu aktivnost potreban je samo radijus. S obzirom na to da program, koji se nalazi u sklopu realnog dinamometra, istovremeno računa broj okretaja valjka po minuti, brzinu vozila (3.4.), kutnu brzinu (3.5.) i kutno ubrzanje (3.6.) te prikazuje graf iz baze podataka, za svako će vozilo biti spremljeni ti podaci. Cjelobrojni broj će predstavljati broj okretaja valjka, a ti će čvorovi sadržavati ime vozila i grafove s mjerenjima koja su već obavljena. RPM varijabla će predstavljati cjelobrojni broj okretaja u minuti. Radi lakše pretvorbe brzine u formulu se dodaje 60/1000.

Varijabla „v“ predstavlja brzinu, ω će predstavljati kutnu brzinu, slovo „a“ kutno ubrzanje, a „t1“ će predstavljati vrijeme potrebno valjku kako bi napravio puni krug.

(3.4.) $$v = \ 2 * r * \pi * RPM * \frac{60}{1000} \ [km/hr]$$

(3.5.) $$\omega = \frac{\pi * RPM}{60} \ [rad/s]$$

(3.6.) $$a \ = \frac{\omega}{t1} \ [rad/s2]$$

Korisniku se odabirom marke ispišu ovi podaci, a ukoliko želi vidjeti graf svog mjerenja i znati moment svog ili testnog motora, stisnut će gumb za prijelaz na novu aktivnost.

Na toj aktivnosti će biti vidljiv naziv vozila koje je odabrano i graf za koje su mjerenja rađena. Ime i graf će morati povući iz *Firebase* baze podataka (***Sl.3.14***).

Sl.3.14. *Prikaz sučelja GraphActivity*

Tekstualni dijelovi će prikazati vozilo za koje je graf napravljen (***Sl.3.14***) i jednu od najvažnijih stvari za izračun konjske snage, a to je okretni moment motora (3.7.), oznaka τ.

(3.7.) $$\tau = a * I \ [kgm]$$

Kao što je vidljivo, moment motora je umnožak kutnog ubrzanja i momenta inercije koji se izračunao na četvrtoj aktivnosti nakon ponuđenog višestrukog odabira. On je bio zadnja ključna varijabla za izračun konjskih snaga motora. U konačnici, kada sakupi sve ove podatke, korisnik pritiskom na gumb, koji predstavlja plutajući akcijski gumb, ponovno otvara aktivnost (***Sl.3.15***) na kojoj će biti prikaza na slika odabranog testnog vozila iz baze podataka, odnosno tekst koji prikazuje rezultat.

Sl.3.15. *Konačni rezultat*

Pritiskom na *Izračunaj* aplikacija konačno provodi posljednji izračun putem formule (3.8.) i tako se dolazi do traženog rješenja. U funkciji programa za izračun konjskih snaga mora se uzeti u obzir činjenica da su sve varijable dovele do rezultata u mjernoj jedinici Watt, a aplikacija će napraviti i pretvorbu (3.9.) u konjske snage koristeći omjer 1HP = 745.7W

(3.8.) $$PW = \tau * \omega\ [W]$$

(3.9.) $$P = \frac{PW}{745.7}\ [HP]$$

S konačnim rezultatom se dokazalo kako ovakav način može ponuditi jednostavnije rješenje te je aplikacijski test uspješno proveden. Unošenjem i biranjem podataka u svakom koraku aplikacije, točnije u svakoj aktivnosti, provodili su se određeni izračuni i na kraju su doveli do željenog i točnog rezultata.

4. ZAKLJUČAK

Kako je korištenje mobilnih uređaja neizbježno, u ovome se radu htjelo približiti rješenje problema koji nosi dinamometar, a najveći su cijena i dimenzije koje zauzima u radionici. Kroz ovaj završni rad korištena je Android aplikacija kao rješenje. Za izradu aplikacije trebalo je povezati znanje fizike i programiranja kako bi se došlo do rješenja koje je praktično i lako upotrebljivo. Za izradu se koristio Android Studio razvojno okruženje koje, osim što je popularno i raširenо, pruža mnoge usluge kao npr. povezivanje s online bazom podataka koja je besplatna i može biti korištena u stvarnom vremenu. Pokušalo se napraviti aplikaciju koja ima dobar aplikacijski protok tj. jednostavno kretanje kroz aplikaciju s jasnim uputama i napomenama. Kada bi korisnik napravio svoj model dinamometra i koristio jednostavnije senzore za mjerenje okretaja bez stalnog praćenja u ekran, lako može izračunati sve podatke koji su potrebni za vozilo koje je postavljeno na mjerni instrument. Zbog ovakvog rješenja dinamometar (skraćeno dyno) bio bi pristupačniji te bi se mjerenja mogla obavljati u više gradova, a ne samo na nekoliko mjesta, kao što je slučaj u Republici Hrvatskoj. Budući da je aplikacija prototipske ili čak poticajne namjere (poticaj za izradom jednostavnijeg dinamometra), višestruki odabir je sveden na par vozila jer su bitniji izračuni koji se provode unutar same aplikacije. Prebacivanjem vrijednosti između aktivnosti, neke podatke, kao npr. radijus, nije bilo potrebno ponovno unositi jer aplikacija uz određene funkcije to učini sama. Budući da je *Firebase* baza podataka, baza koja omogućava unos i prijenos podataka u stvarnom vremenu, manji bi problem bio spojiti senzor za okretaje na mobitel koji bi sve to spremao u bazu i poslije koristio za daljnje računanje. Korištenjem operacijskog sustava *Android* i *Android Studio* platforme je postajala mogućnost napraviti aplikaciju. *XML* programski jezik nam je omogućio oblikovanje rasporeda određenih elemenata dok je *Java* programski jezik dao određene odgovornosti i funkcionalnosti svakom od tih elemenata aplikacije. Kako bi mogli spremati određene proizvode, tipove podataka ili u ovom slučaju vozila iskorištena je *Firebase* baza podataka te se spremanjem vrijednosti za svako vozilo moglo doći do konačnog rješenja za koji je namijenjena. Zaključuje se kako bi mjerenje konjskih snaga bi bilo mnogo jednostavnije, „na dlanu“ i svatko bi mogao koristiti aplikaciju jer bi bila široko dostupna.

LITERATURA

[1] Wikipedija, Dynometer, dostupno na: https://en.wikipedia.org/wiki/Dynamometer, (travanj, 2019.)

[2] DTEC, Inertia Dyno Design Guide, dostupno na: http://dtec.net.au/Inertia%20Dyno%20Design%20Guide.htm, (travanj, 2019.)

[3] Wikipedija, XML, dostupno na: https://hr.wikipedia.org/wiki/XML, (svibanj, 2019.)

[4] Wikipedija, Android, dostupno na: https://hr.wikipedia.org/wiki/Android_(operacijski_sustav), (svibanj, 2019.)

[5] D. Radić, Mobilni uređaji – Android operativni sustav, dostupno na: https://informatika.buzdo.com/pojmovi/mobile-3.htm, (svibanj, 2019.)

[6] R. Bandakkanavar, Google Android Operating System, 2015., dostupno na: https://krazytech.com/technical-papers/android-a-smart-phone-operating-system-by-google, (svibanj, 2019.)

[7] Wikipedija, Android Studio dostupno na: https://en.wikipedia.org/wiki/Android_Studio,

(svibanj, 2019.)

[8] Developers, Meet Android Studio, dostupno na: https://developer.android.com/studio/intro, (svibanj, 2019.)

[9] Wikipedija, Java (programski jezik), dostupno na: https://hr.wikipedia.org/wiki/Java_(programski_jezik), (svibanj, 2019.)

[10] Java, dostupno na: https://www.java.com/en/, (svibanj, 2019.)

[11] D. Kirasić, XML tehnologija i primjena u sustavima procesne informatike, Sveučilište u Zagrebu, Fakultet elektrotehničke i računarstva, Zagreb, dostupno na: https://www.ieee.hr/_download/repository/mipro_xml_tekst.pdf, (svibanj, 2019.)

[12] Microsoft, XML za početnike, dostupno na:

https://support.office.com/hr-hr/article/xml-za-po%C4%8Detnike-a87d234d-4c2e-4409-9cbc-45e4eb857d44, (svibanj, 2019.)

[13] Wikipedija, Firebase, dostupno na: https://en.wikipedia.org/wiki/Firebase, (svibanj, 2019.)

[14] Firebase, Firebase Realtime Database, dostupno na: https://firebase.google.com/docs/database/, (svibanj, 2019.)

[15] Amar InfoTech, Firebase, dostupno na: https://www.amarinfotech.com/top-10-benefits-of-having-firebase-for-mobile-app-development.html, (svibanj, 2019.)

SAŽETAK

U završnom radu razvijena je Android mobilna aplikacija za računanje konjskih snaga preko podataka valjka s dinamometra. Korisnik najprije unosi dimenzije valjka na mjernom instrumentu te gumbom *Izračunaj* računa obujam valjka. Korisnik također ima višestruki odabir u slučaju da ne koristi klasični materijal (gumu) za navlake preko valjaka. Korisnik također odabire vrstu pogona što igra ulogu na moment inercije i okretni moment te ih računa ovisno o odabiru. Sve što se do ovog trenutka računalo se računalo se za samo jedan valjak. Nakon odabira vozila dohvaćaju se određeni podaci iz *Firebase* baze podataka poput okretaja u minuti, slika vozila i graf mjerenja tog vozila. U konačnici se računa snaga u Watt-ima i pretvara u konjske snage.

Ključne riječi: analiza, Android, baza podataka, Dyno, Firebase, konjske snage

ABSTRACT

Android Application for Representation and Analisys Measurement Data

In this bachelor's thesis, an Android mobile application for calculating horse power using cylinder's data from dyno is developed. By pressing the button *Izračunaj* the application calculates cylinder volume. User can choose between multiple choices, in case the user is not using classic material (rubber) for cylinder covers. User can also choose the type of drive which is crucial for moment, inertia and torque and the application calculates them, depending on user's choice. All calculations thus far were for only one cylinder. After choosing the vehicle, the DynoApp fetches specific data from Firebase database: like revolutions per minute, a picture of vehicle and measurement chart for that vehicle. Finally, it calculates power in Watts and then the power is being converted into horse power.

Key words: analysis, Android, database, Dyno, Firebase, horse power

ŽIVOTOPIS

Antonio Bajivić je rođen 11. srpnja 1997. godine u Virovitici. Od 2004. do 2012. godine pohađa Osnovnu školu Vladimira Nazora u Virovitici. Za vrijeme osnovnoškolskog obrazovanja bio je sudionik natjecanja iz geografije i matematike. Godine 2012. upisuje Opću gimnaziju Petra Preradovića u Virovitici koju završava u 2016. godine polaganjem ispita državne mature. Tijekom svojeg srednjoškolskog obrazovanja sudjeluje na natjecanjima iz geografije, matematike, latinskog jezika. Godine 2016. upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija na Sveučilištu Josipa Jurja Strossmayera u Osijeku, preddiplomski studij Računarstvo.

PRILOZI (NA DVD-u)

Prilog 1: Završni rad u docx i pdf formatu

Prilog 2: Projekt mobilne aplikacije u *Android Studiju*