

**SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN**

mRanger

PROJEKT U SKLOPU STEM REVOLUCIJA U ZAJEDNICI



Varaždin, 2019.

Sadržaj

1. Uvod	1
2. Instaliranje potrebnih programa.....	2
3. Izrada prvog projekta u Android Studio	3
3.1 Izrada novog projekta.....	3
3.2 Prvi koraci u Android Studio alatu.....	7
3.3 Pokretanje projekta.....	16
4. Dodavanje novog zaslona, slike i gumba	19
4.1 Dodavanje novog zaslona.....	19
4.2 Dodavanje slike	21
4.3 Dodavanje gumba.....	26
4.4 Prijelaz s jednog zaslona na drugi	28
4.5 Kreiranje layout-a.....	31
5. Objekti i klase	34
5.1 Dodavanje nove klase u Android Studio-u	35
6. Izrada mRanger aplikacije	37
6.1 Kreiranje potrebnih datoteka	38
6.2 Dodavanje koda u klase.....	39
6.3 Dodavanje koda u activity datoteke	43
6.3 Dodavanje koda u layout datoteke	45
6.4 Još par sitnih izmjena	48
6.5 Sinkronizacija Gradlea	50
7. Logičko programiranje	52
7.1 Uvod u Prolog	52
7.1.1 Tipovi podataka.....	53
7.1.2 Činjenice i pravila	54
7.1.3 Pravila.....	56
7.2 Pisanje i izvršavanje Prolog programa	56
7.3 Provjera godišnjeg doba	62
7.4 Provjera prosječne temperature	63
7.5 Pozivanje Prolog programa iz PHP-a.....	65
7.6 Usporedba PHP i Prolog rješenja	67
7.6.1 Provjera godišnjeg doba u PHP-u	67
7.6.1 Provjera prosječne temperature u PHP-u	69
7.7 Dodatni Prolog materijali	70

7.7.1	Liste	70
7.7.2	Rekurzije	70
7.7.3	Posebni ugrađeni predikati	74
7.8	Prikaz baze podataka i web stranice.....	75

1. Uvod

Ova dokumentacija je nastala na **Fakultetu organizacije i informatike u Varaždinu**, u sklopu kolegija Analiza i razvoj programa (pod mentorstvom **Doc. dr. sc. Zlatko Stapić** i **Dr. sc. Boris Tomaš**) te kolegija Logičko programiranje (pod mentorstvom **Prof. dr. sc. Sandra Lovrenčić**).

Za izradu ovog projekta i društveno korisne dokumentacije je zaslužan **mRanger tim**:

- **Fabijan Josip Kraljić** (*AIR + LP*)
- **Jakov Kristović** (*AIR + LP*)
- **Karlo Pavleka** (*AIR + LP*)
- **Kristijan Perković** (*AIR*)
- **Kristijan Žebčević** (*AIR*)

Cilj ovog dokumenta je upoznati mladež s osnovama razvoja mobilnih aplikacija i same robotike. Ciljana skupina učenici srednjih škola, kao i svi oni koji žele naučiti nešto više na ovu temu. U ovom dokumentu ćemo se potruditi što jednostavnije proći kroz proces izrade Android aplikacije koja će služiti za daljinsko upravljanje mBot Ranger robotom, na koji ćemo se povezati putem Bluetooth-a.

Kako bi što lakše pratili ovaj dokument, bez obzira na to što ćemo veliki dio gradiva mi sami objasniti, podrazumijevamo da imate minimalna predznanja o tome što su klase, objekti i funkcije ... ili bar da ćete se potruditi to malo proučiti prije izrade ove aplikacije **korak po korak**. 😊

Aplikacija će imati sljedeće funkcionalnosti:

- Povezivanje na mBot Ranger putem Bluetooth-a
- Daljinsko upravljanje robotom
- Mogućnost podešavanja brzine kretanja
- Izbjegavanje sudara robota i prepreke
- Bilježenje temperature zraka u bazu podataka

2. Instaliranje potrebnih programa

Našu aplikaciju ćemo izraditi u Android Studiu, a to je razvojno okruženje koje služi za kreiranje Android mobilnih aplikacija. Detaljni prikaz instalacije ćemo preskočiti, no postoje razni sadržaji na internetu koji prikazuju kako instalirati Android Studio (npr. [Instalacija Android Studia](#)).

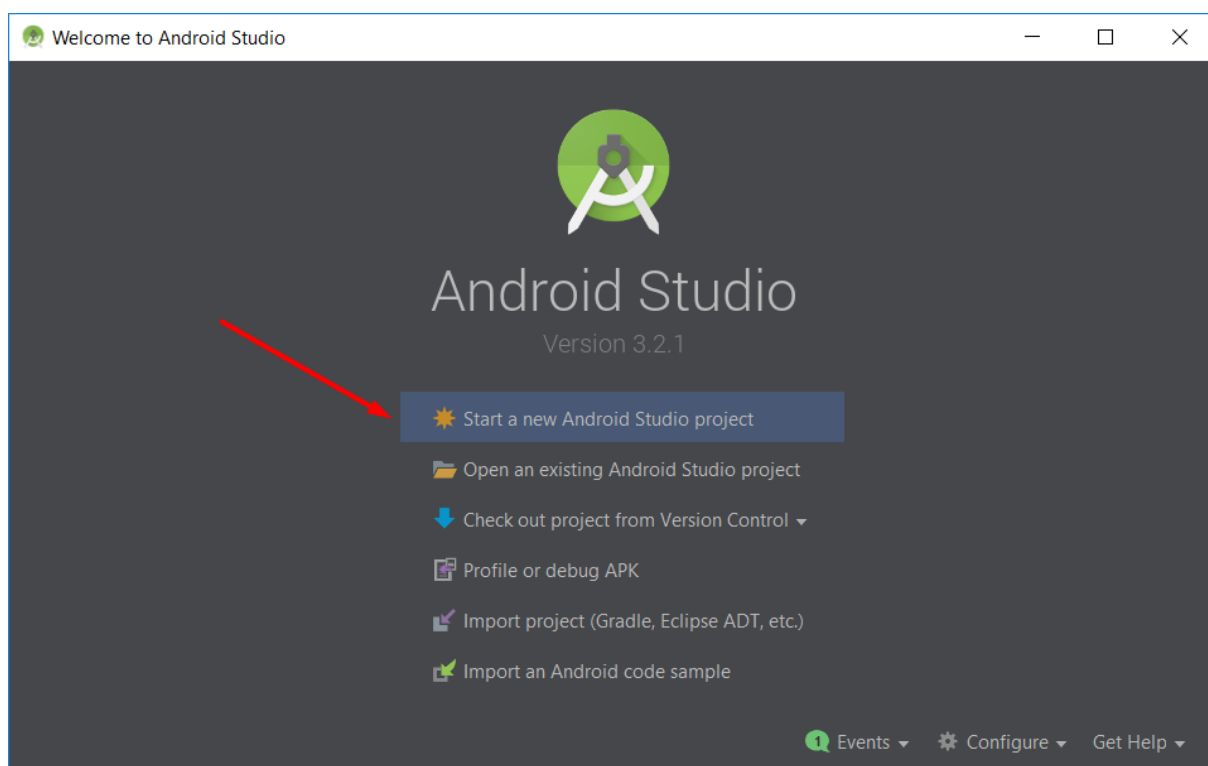
Android Studio se može preuzeti na: [Android Studio - Službena stranica](#)

3. Izrada prvog projekta u Android Studio

U nastavku, sve upute ćemo se potruditi prikazati uz slikovne materijale i komentare kako bi što lakše predočili postupak izrade aplikacije. Na pojedinim slikama se mogu pronaći obojeni okviri i strelice kako bismo usmjerili vašu pozornost na bitne elemente.

Želimo Vam puno sreće prilikom izrade vaše prve aplikacije za kontroliranje metalnog ljubimca. 😊

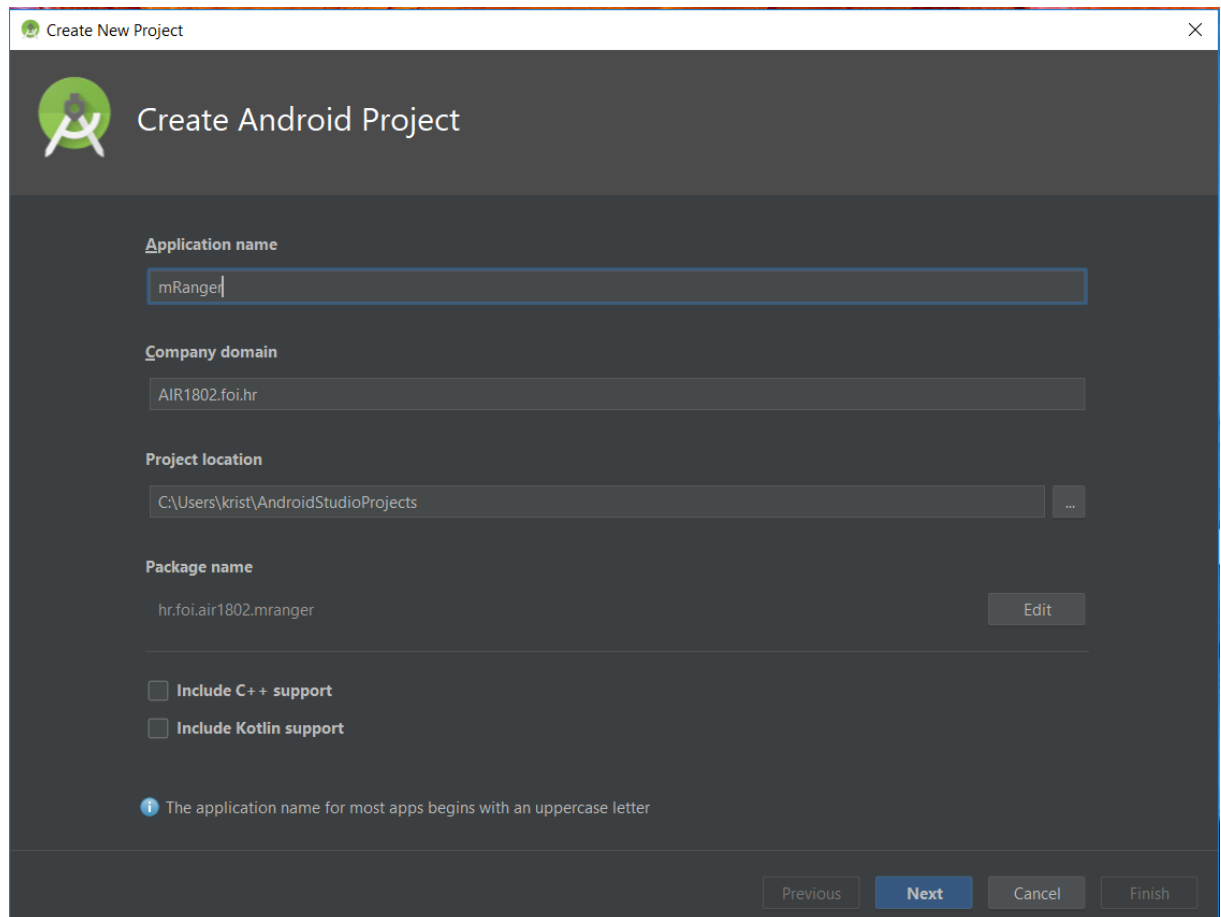
3.1 Izrada novog projekta



Slika 1. Početni prozor Android Studio

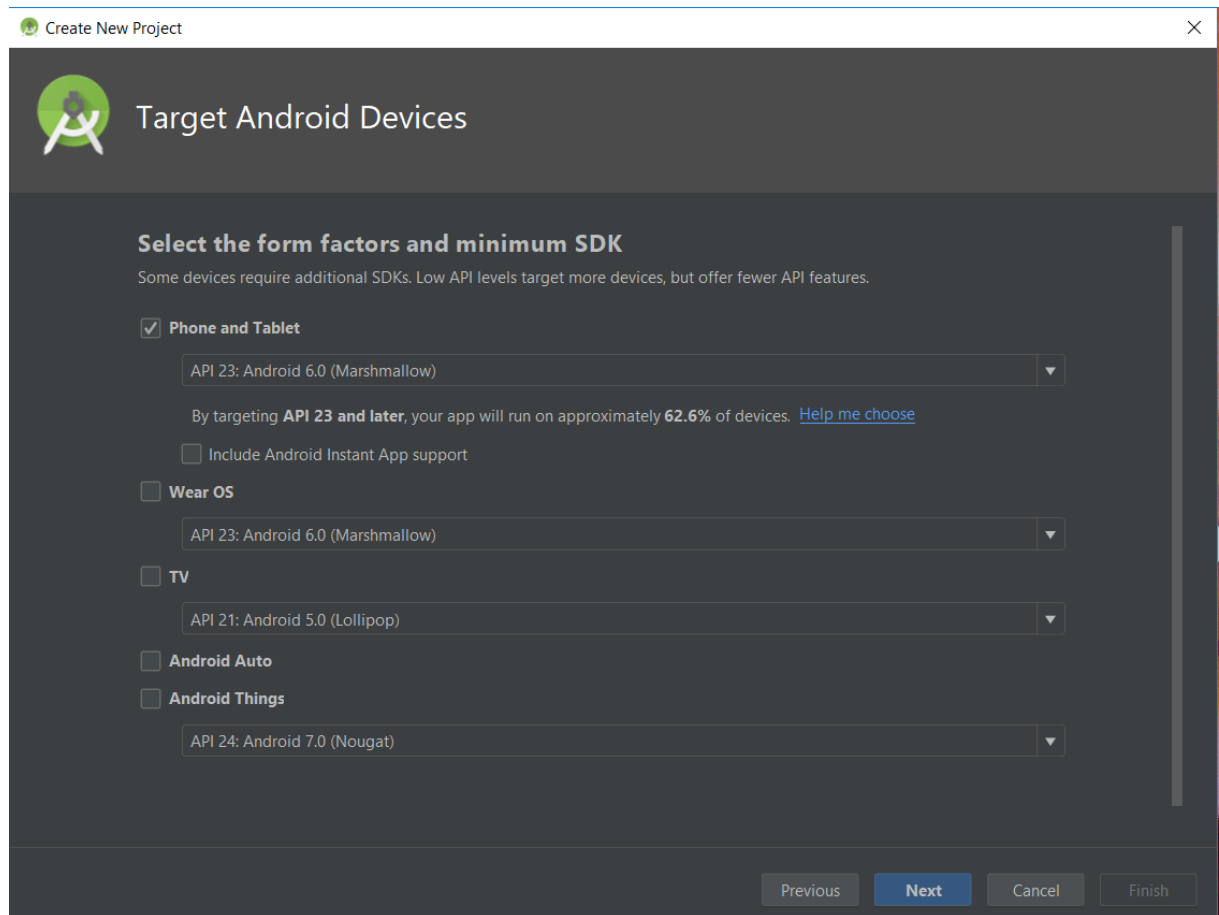
Prilikom pokretanja Android Studia otvara se prozor prikazan na slici iznad. Odaberemo prvu opciju iz prozora, *Start a new Android Studio project*, kojom ćemo započeti kreiranje novog projekta.

Time nam se otvara sljedeći prozor



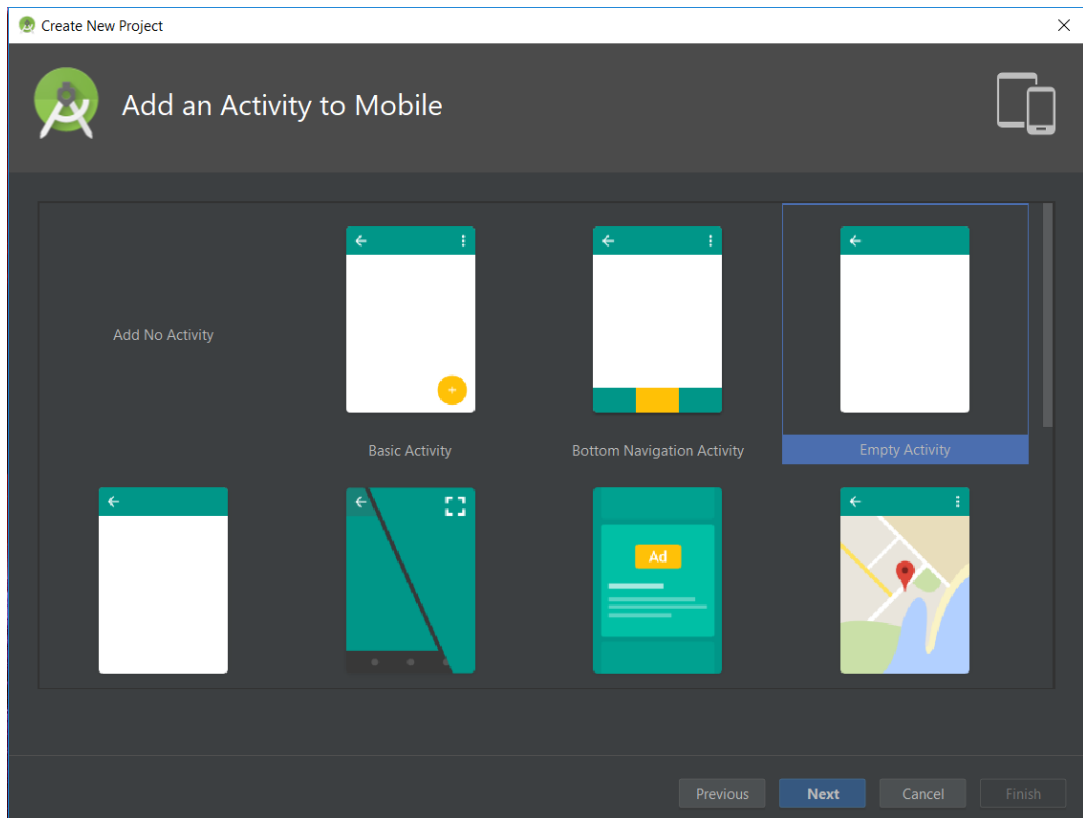
Slika 2. Imenovanje projekta

U ovom prozoru cilj je odrediti naziv projekta, podatke o tvrtki ako ona postoji (projekt će biti smješten u mapu *C:\Users\imeKorisnika\AndroidStudioProjects\tvrka\mRanger*). Ostale opcije ostavljate takve kakve jesu i kliknete na *Next*.



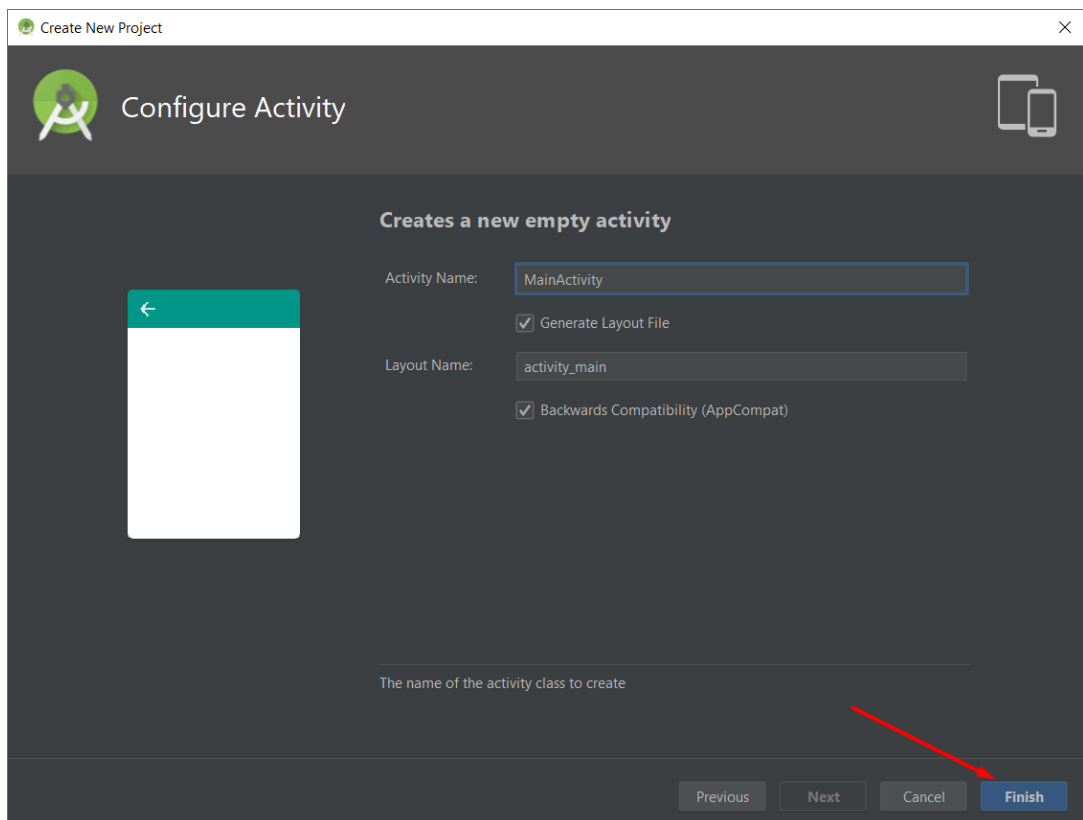
Slika 3. Odabir ciljanih uređaja

Nakon imenovanja projekta potrebno je odabrati vrste uređaja na kojima će se aplikacija izvoditi. U našem slučaju odabiremo *Phone and Tablet* opciju, te nakon toga biramo minimalnu verziju operacijskog sustava za android uređaje (minimalni zahtjevi sustava). Mi odabiremo predloženu opciju *API 23:Android 6.0 (Marshmallow)* prilikom čega dobivamo informaciju o postotku uređaja na kojemu će biti moguće koristiti aplikaciju. Zatim kliknemo *Next*.



Slika 4. Odabir početnog zaslona

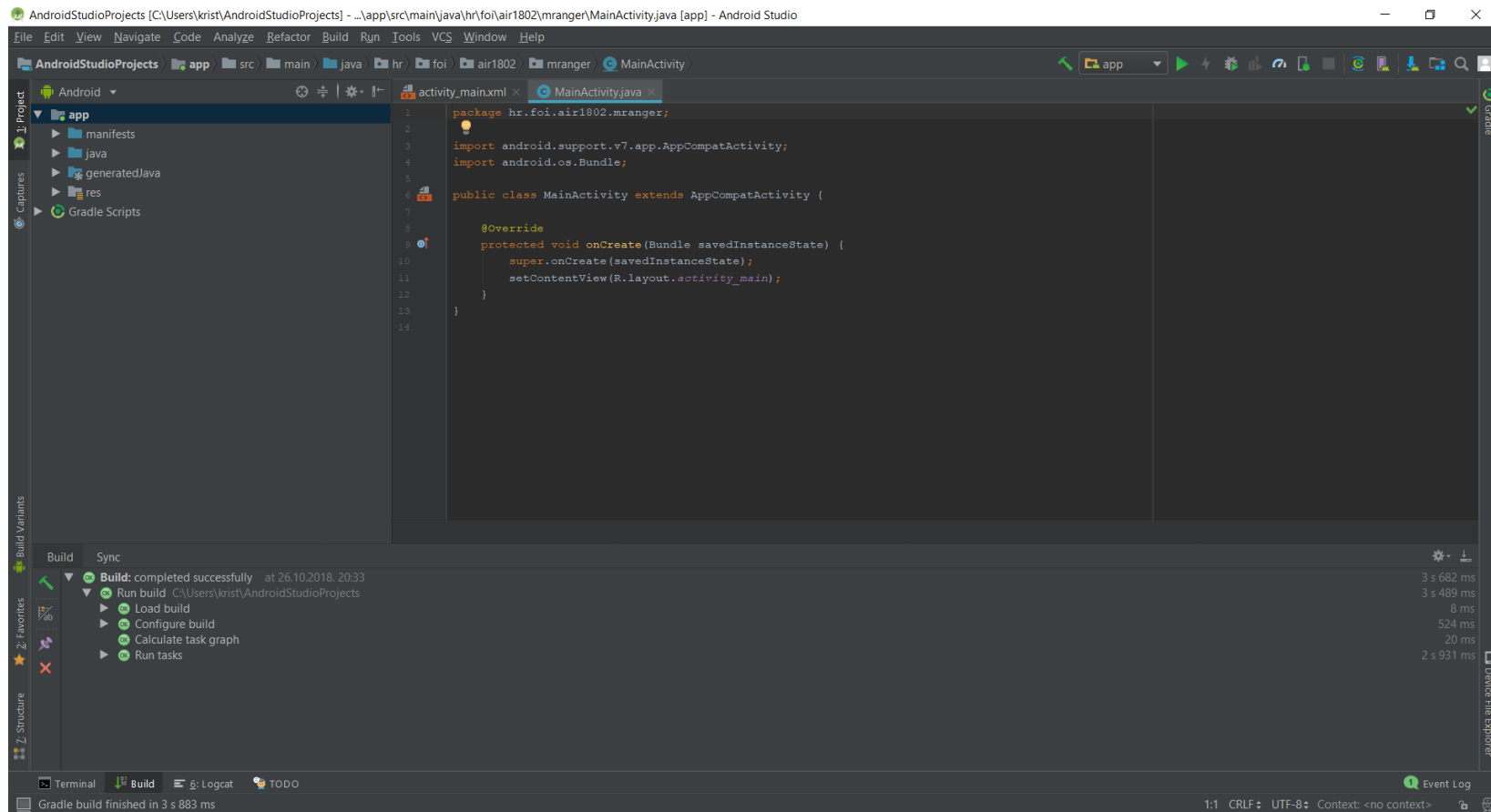
U ovom prozoru imamo mogućnost jednog od predložaka za izgled početnog zaslona aplikacije. Odabiremo predloženi zaslon *Empty Activity* i kliknemo *Next*.



Slika 5. Naziv početnog zaslona

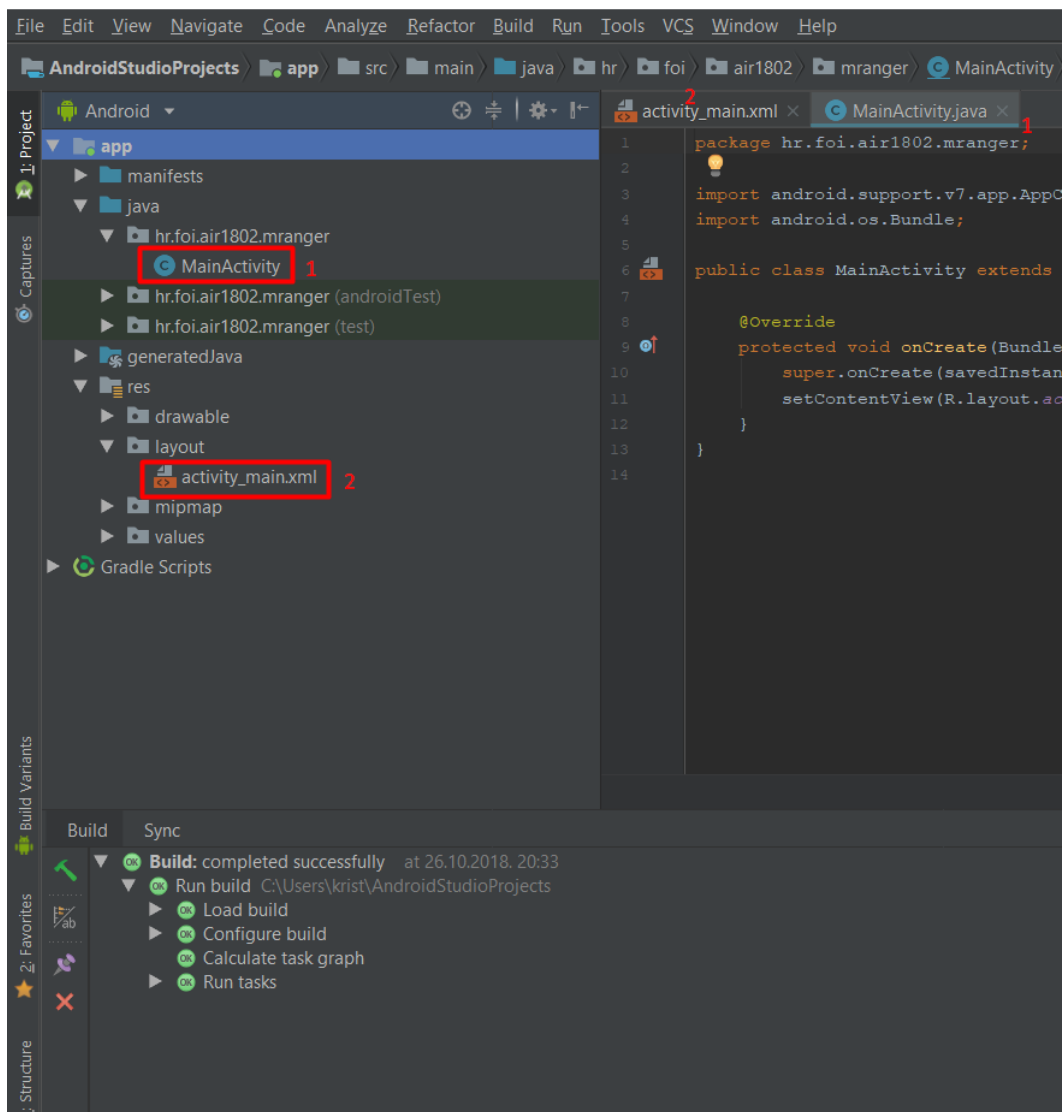
Na slici 5. je moguće odabrati željeno ime za početni zaslon, međutim ostavljamo sve kako je zadano i kliknemo na *Finish*.

3.2 Prvi koraci u Android Studio alatu



Slika 6. Sučelje Android Studio alata nakon otvaranja projekta

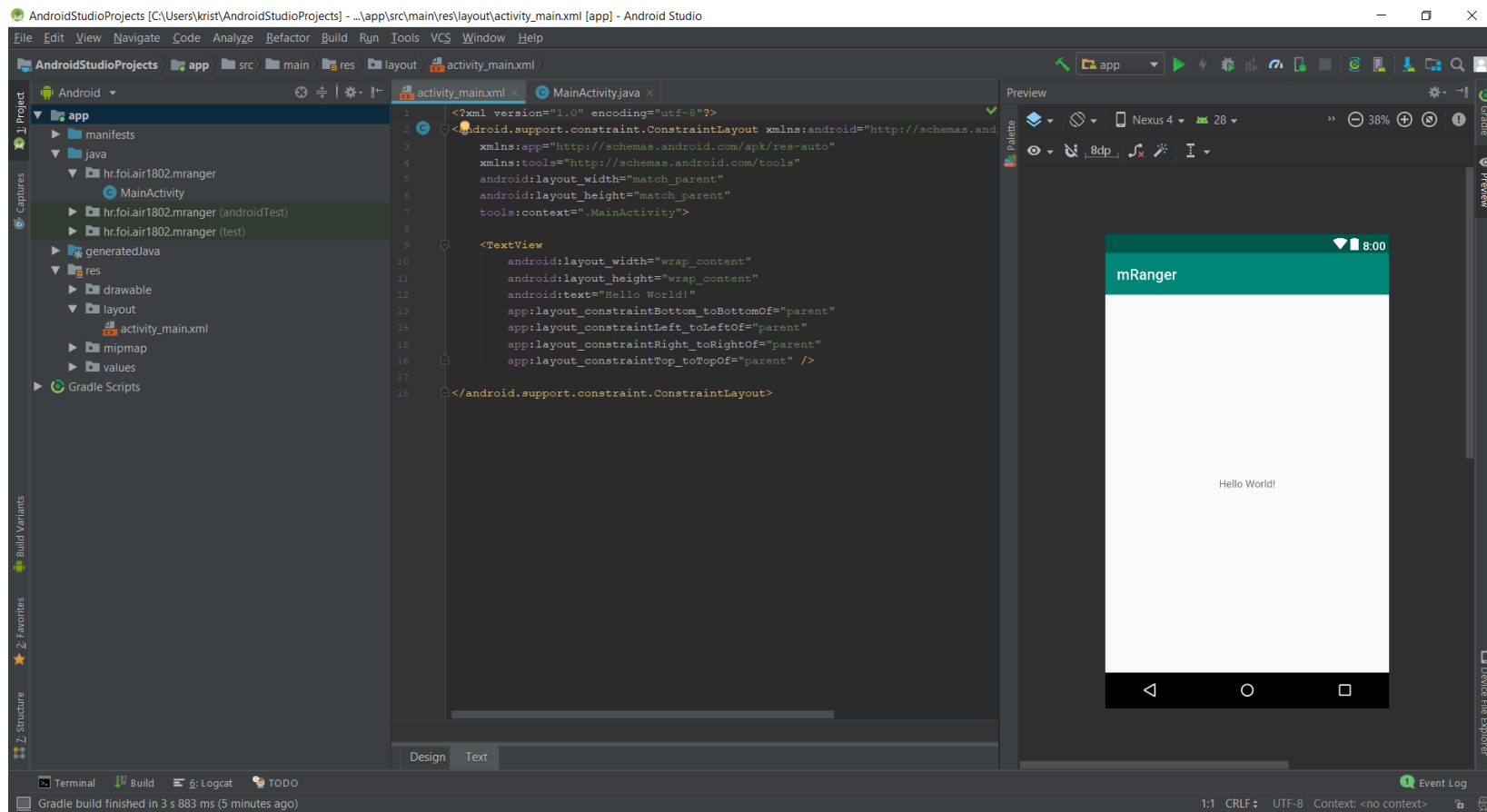
Sljedeća slika prikazuje sučelje Android Studio alata nakon otvaranja našeg novog izrađenog projekta. Na zaslonu nam se po početnim postavkama prikazuju dva taba (*activity_main.xml* i *MainActivity.java*).



Slika 7. java i res mape

Na slici broj 7. vidimo početnu strukturu mape našeg projekta. Bitne mape su *java* i *res*. U *java* mapi, u podmapi sa nazivom naše tvrtke i aplikacije, nalazi se programski kod našeg početnog zaslona kojeg smo prethodno nazvali *MainActivity*. U mapi *res* u podmapi *Layout* nalazi se *activity_main.xml* u kojem se uređuje grafički izgled početnog zaslona naše aplikacije.

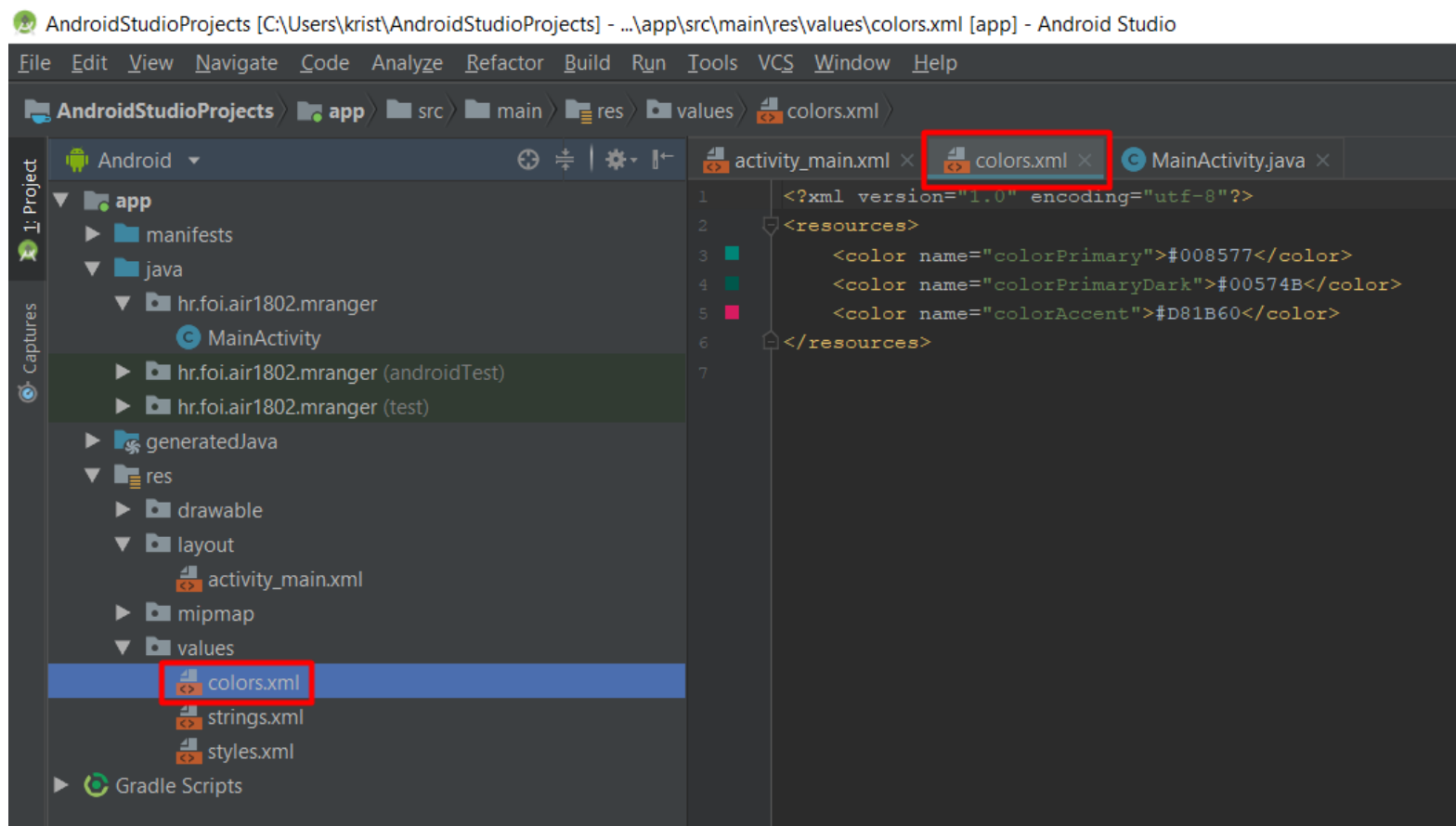
U dokumentu *MainActivity.java* nalazi se programski kod koji se izvršava na početnom zaslonu.



Slika 8. XML datoteka

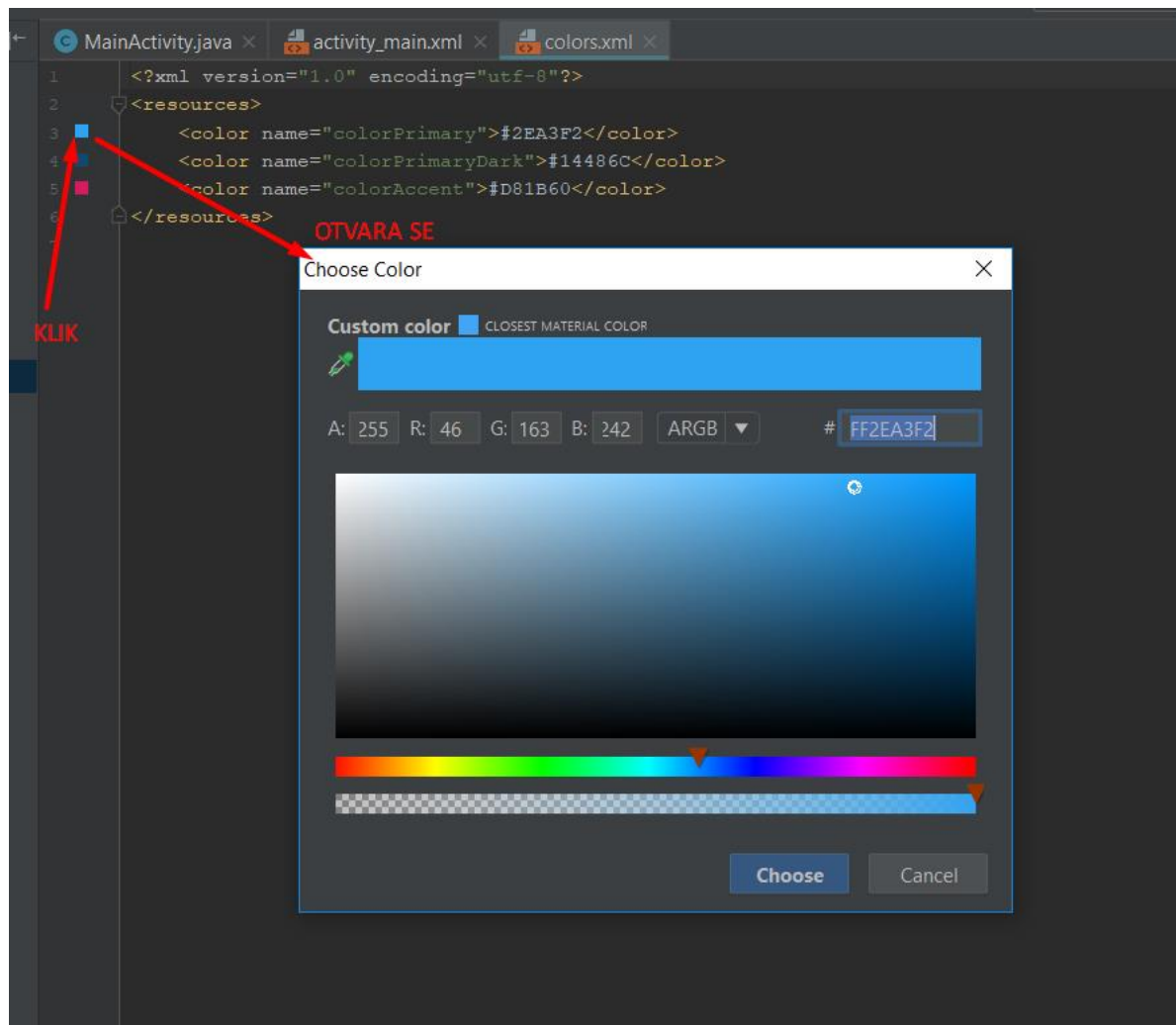
U dokumentu `activity_main.xml` se nalazi programski kod u XML jeziku, koji definira grafički izgled našeg zaslona te podataka koje ćemo prikazivati na njemu. Po početnim postavkama Android Studio alata, na sredini zaslona se prikazuje tekst *Hello World!* Taj tekst možemo također uređivati u ovom XML dokumentu, no o tome ćemo reći nešto više ubrzo.

Predložene boje našeg početnog zaslona su nijanse zelene. Idemo to malo izmijeniti, ne želimo zelenu aplikaciju ! 😊



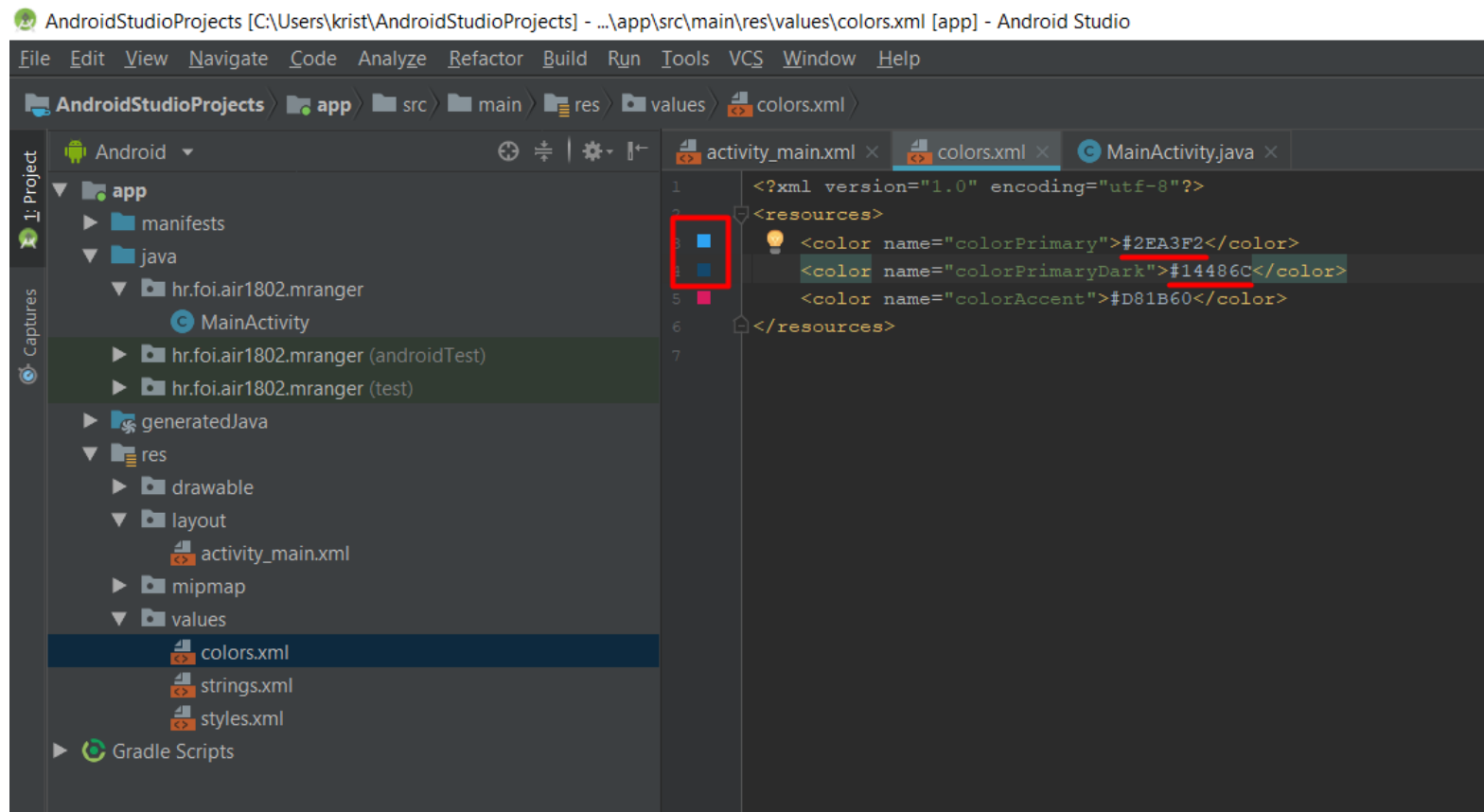
Slika 9. Promjena boja zaslona 1/2

U prethodno spomenutom **res** mapi, imamo mapu **values** koja sadrži datoteku **colors.xml** gdje možemo promijeniti boje zaslona naše aplikacije. Kao što možete vidjeti, boje zaslona su zapisane u heksadecimalnom zapisu.



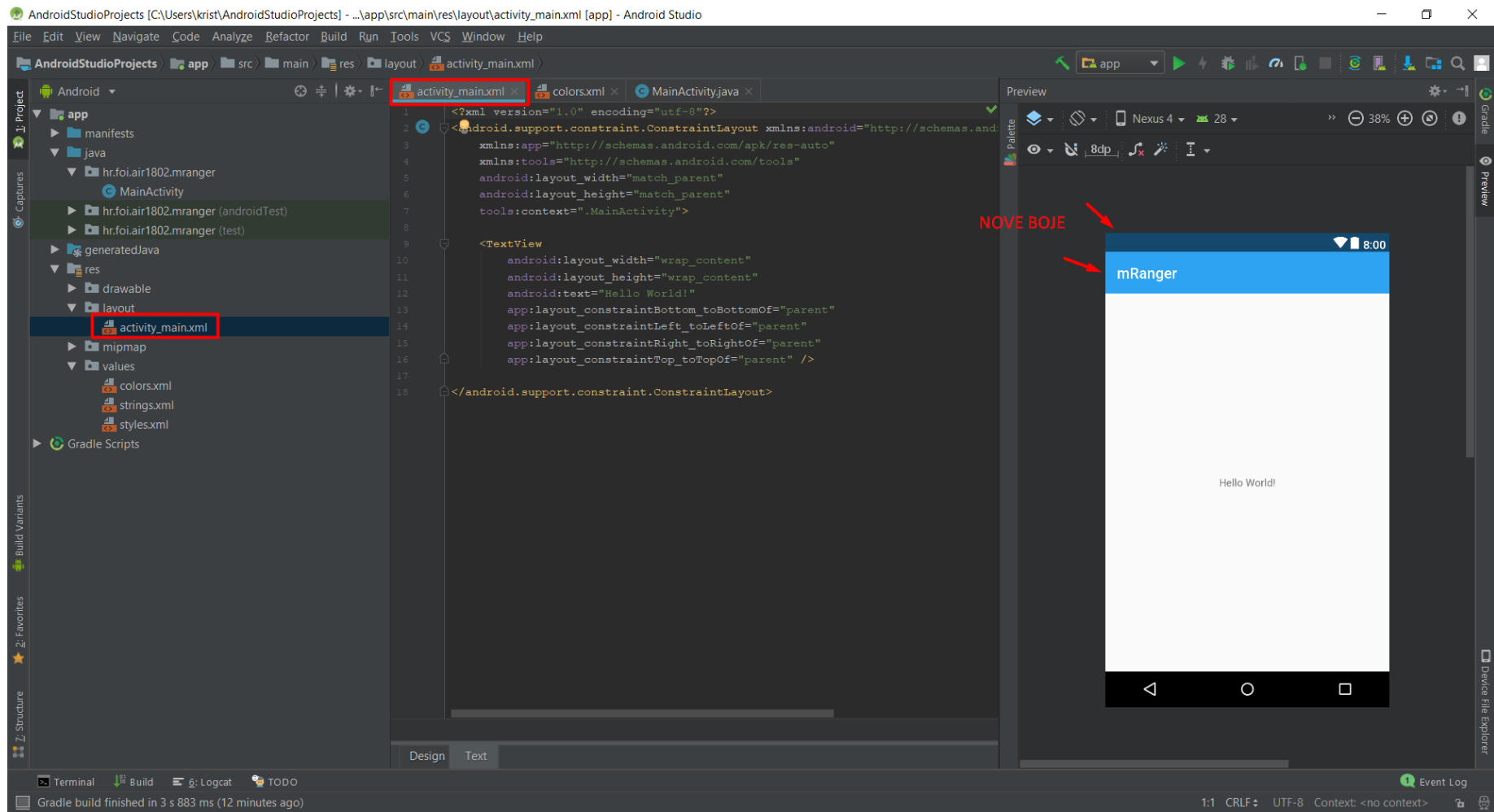
Slika 10. Promjena boja zaslona 2/2

Klikom na kvadratić sa bojom pokraj linije koja sadrži heksadecimalni zapis boje, otvara nam se prozor gdje možemo odabrati željenu boju na vrlo jednostavan način, bez potrebe da znamo stvarni heksadecimalni kod boje.



Slika 11. Promjena heksadecimalnog zapisa boje

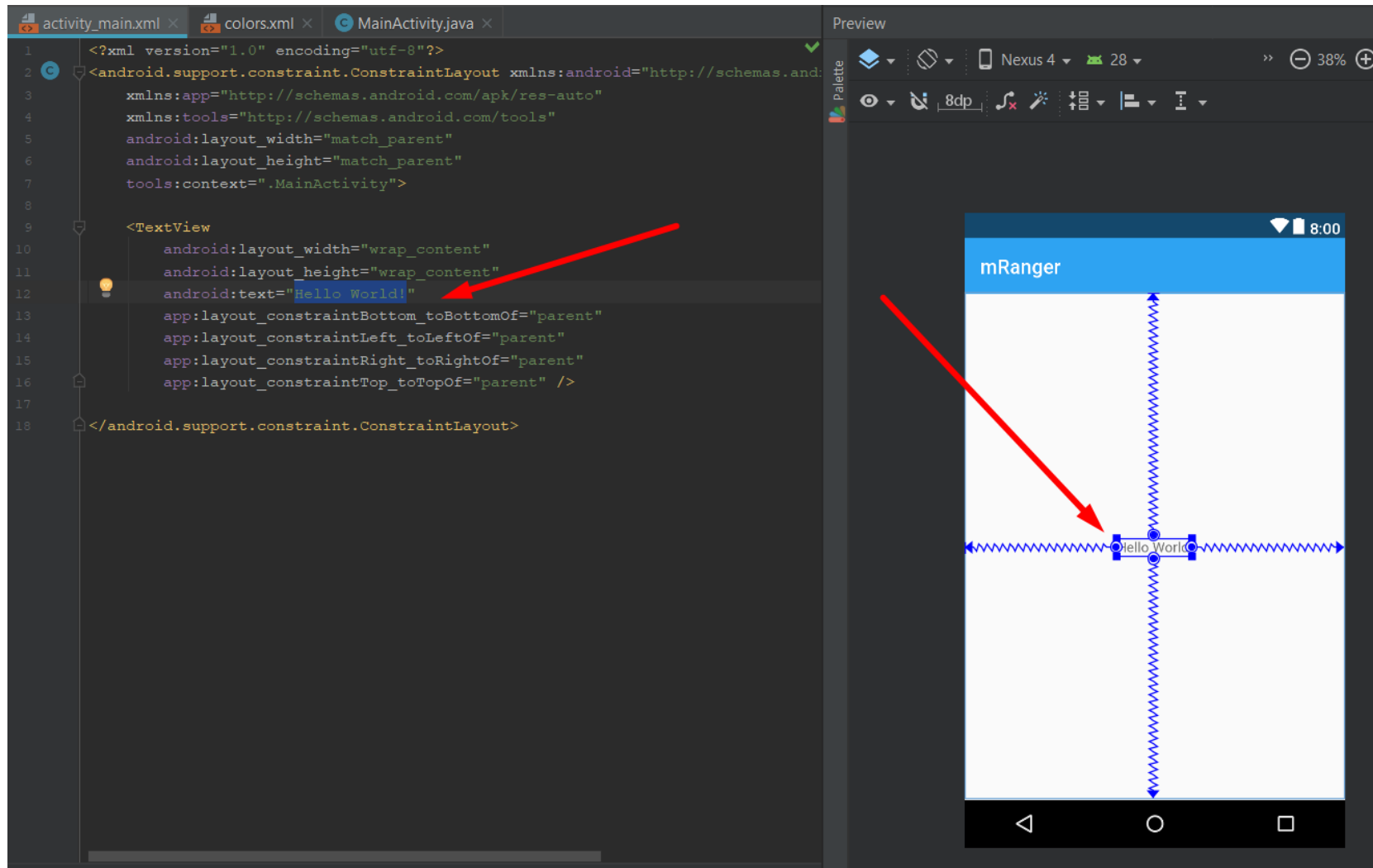
Nakon odabira željenih boja, možemo vidjeti kako se promijenio heksadecimalni kod u **colors.xml** dokumentu.

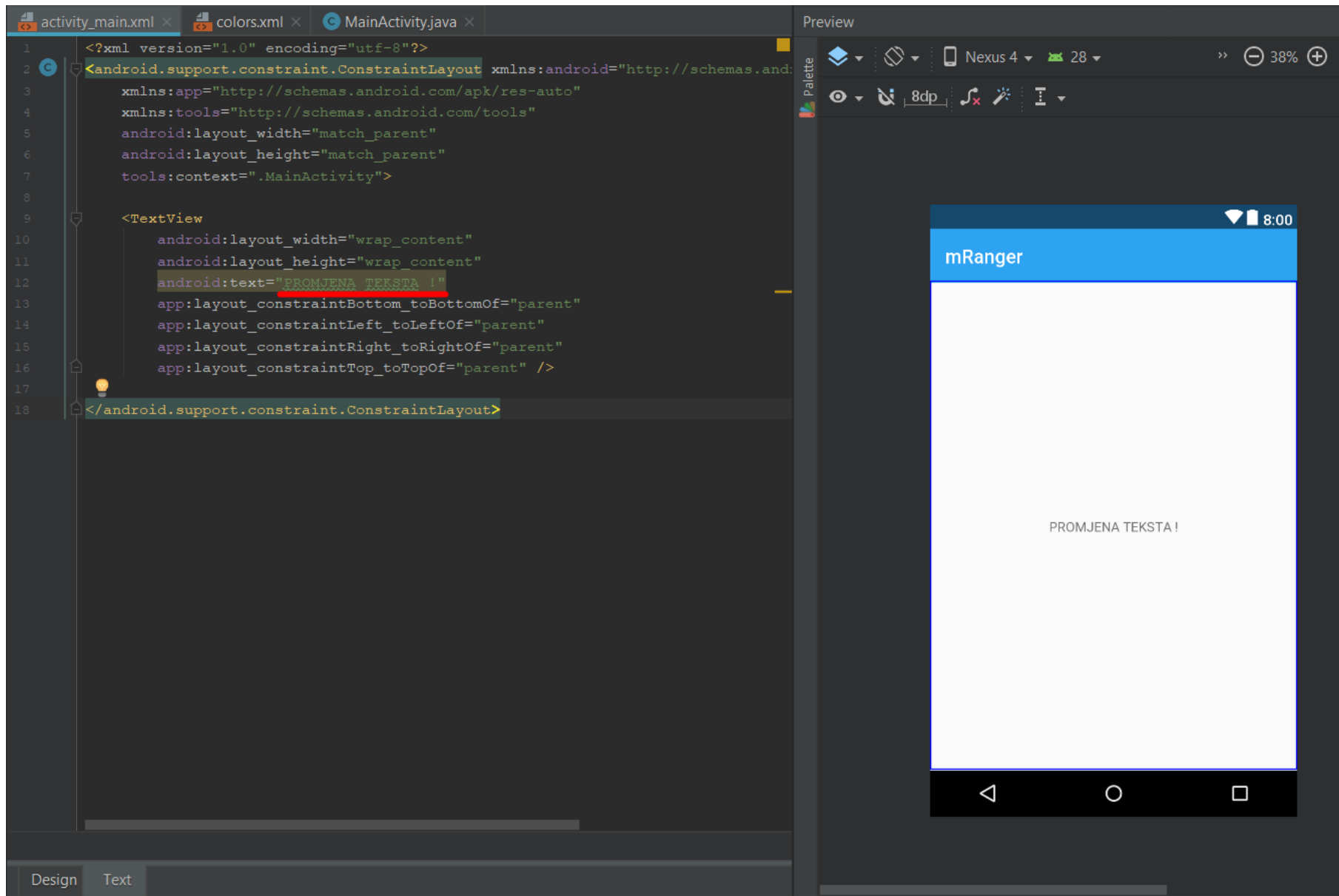


Slika 12. Promjena boja zaslona

Otvorimo li *activity_main.xml*, možemo vidjeti prikaz našeg zaslona te uočiti da su se boje izmijenile.

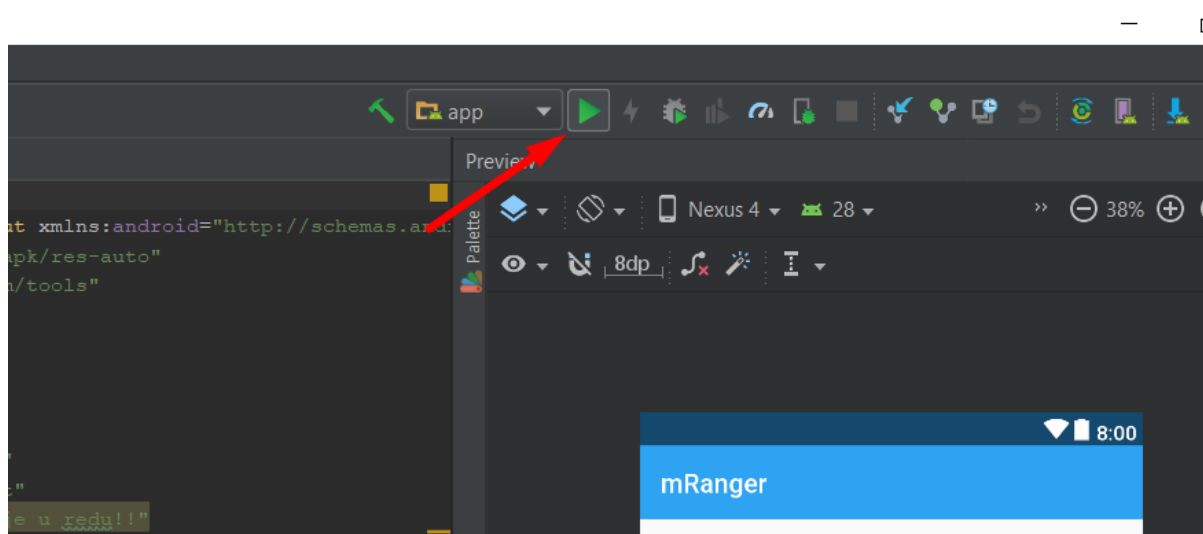
Također, idemo malo izmijeniti i ovaj tekst *Hello World!* Koji se sam od sebe stvorio u našoj aplikaciji prilikom kreiranja. Ako malo bolje promotrimo *activity_main.xml* uočavamo da postoji redak gdje je taj tekst zapisan.





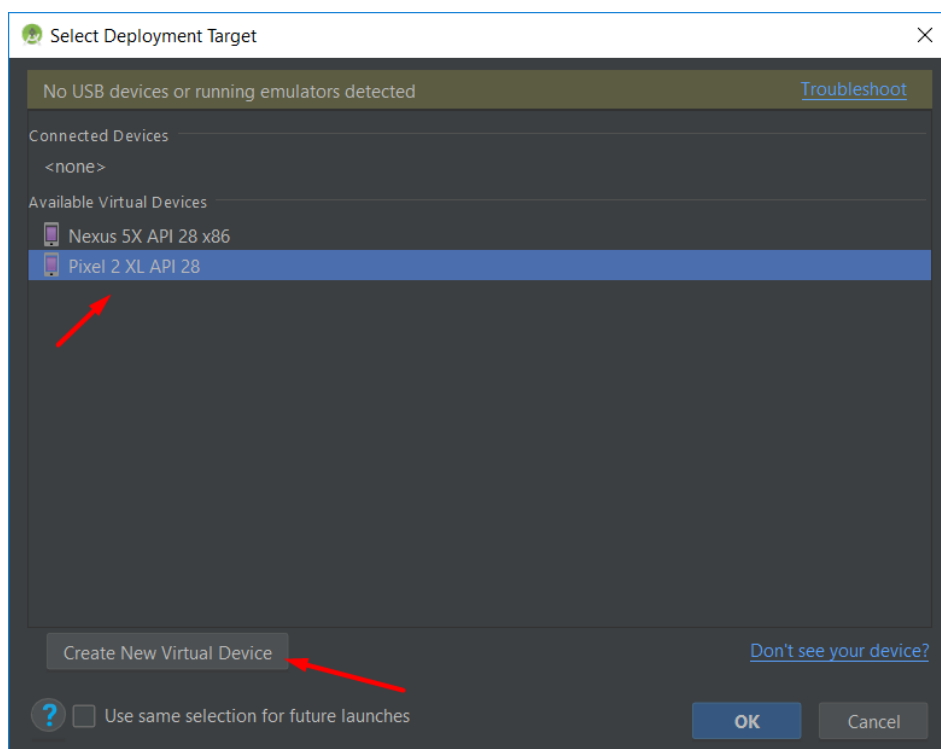
Slika 14. Promjena teksta

3.3 Pokretanje projekta



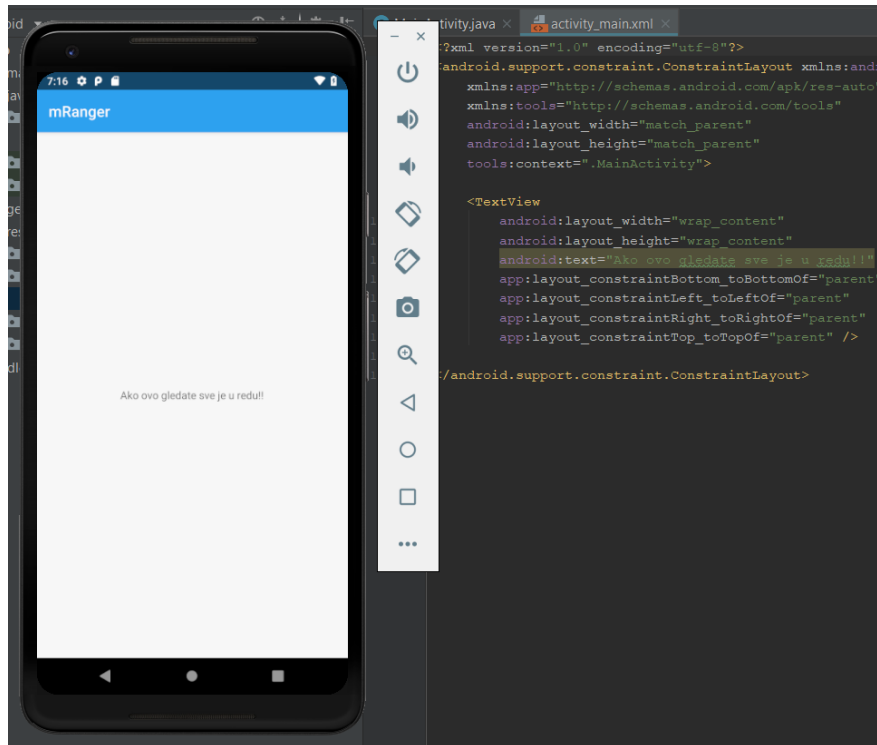
Slika 15. Gumb za pokretanje projekta

Kada poželimo vidjeti našu aplikaciju na djelu, u gornjem desnom dijelu alatne trake se nalazi gumb za pokretanje aplikacije.



Slika 16. Odabir virtualnog uređaja

Nakon toga prikazuje se zaslon za odabir virtualnog uređaja na kojem želimo pokrenuti aplikaciju. Možemo odabrati jedan od već ponuđenih ili pak dodati novi uređaj. Klik na *OK*.



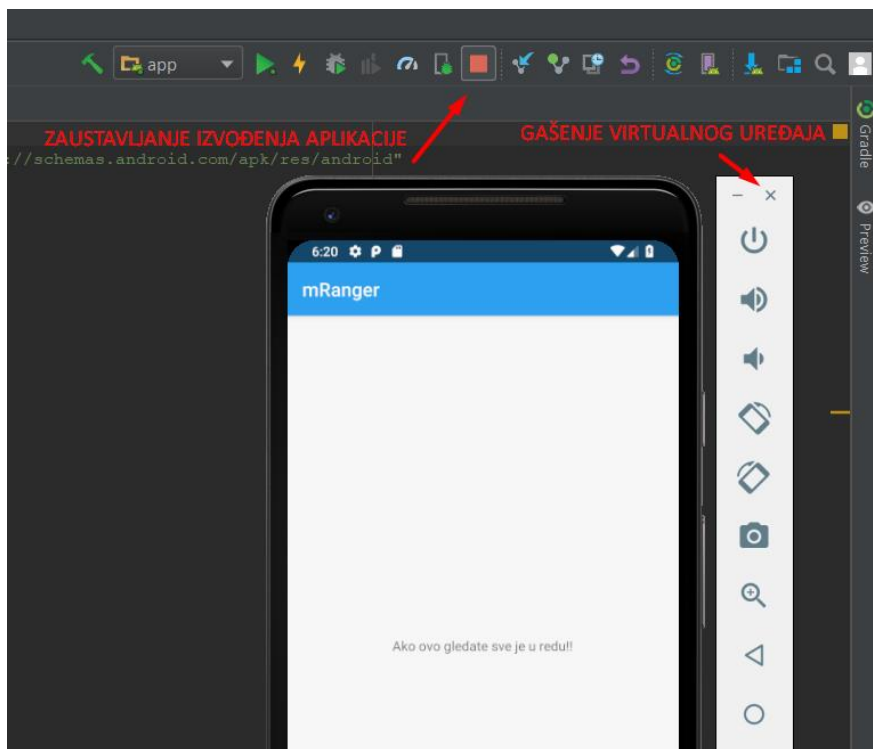
Slika 17. Prikaz aplikacije na virtualnog uređaju uslijed pokretanja

Prilikom pokretanja virtualnog uređaja automatski se pokrene i prikaže na zaslonu naša aplikacija.

Važno je napomenuti:

- Prilikom programiranja u Android Studio alatu, jednom kada pokrenemo virtualni uređaj nije potrebno gasiti, već samo minimiziramo dok radimo promjene u kodu.
- Ukoliko vaše računalo nema minimalno 8 GB RAM memorije, vrlo vjerojatno nećete moći koristiti značajku virtualnog uređaja, no tada imate opciju priključiti svoj pravi uređaj putem USB, te odabrati ga prilikom pokretanja kao na slici 16.
- Emulator nema savršeno ugrađen rad s Bluetooth-om te se preporučuje za rad s našom aplikacijom spajanje vlastitog Android uređaja.

Prilikom zaustavljanja izvođenja aplikacije (pogledati sliku 18. ispod) ne gasimo virtualni uređaj, već zaustavljamo samo izvođenje aplikacije pritiskom na *Stop* znak koji se pojavi nakon pokretanja u gornjem desnom dijelu alatne trake.

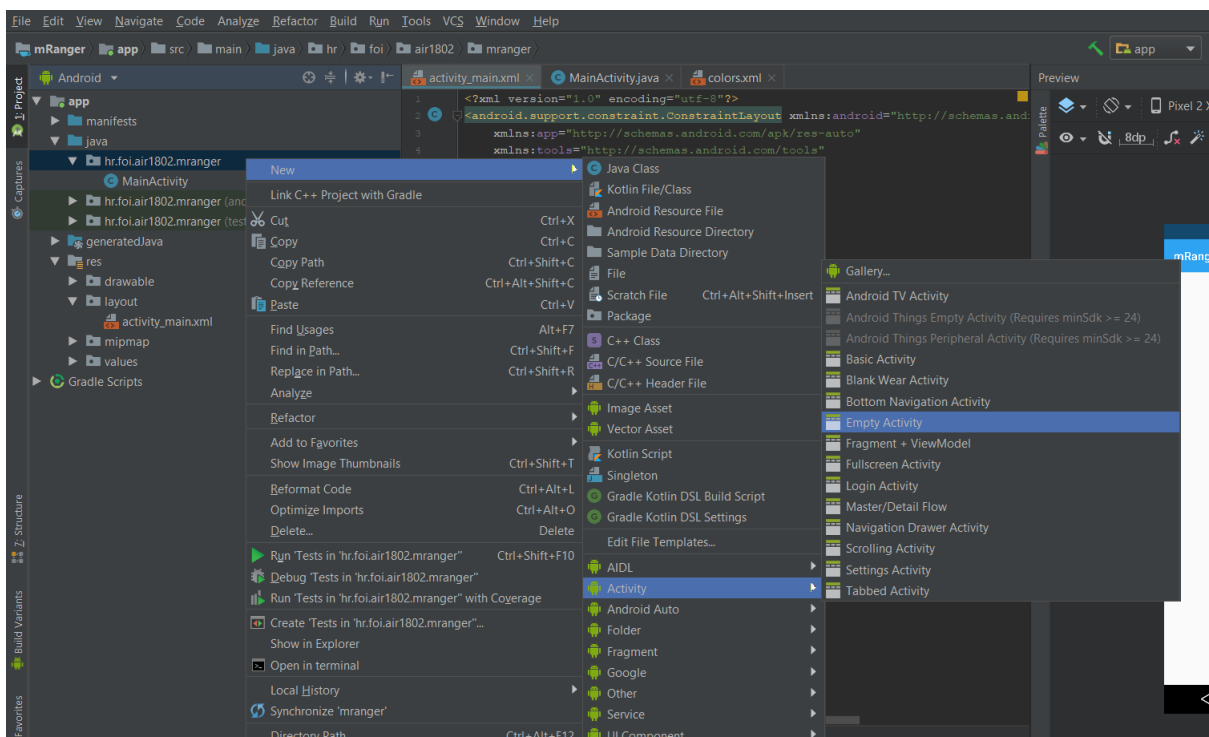


Slika 18. Prekid izvođenja aplikacije

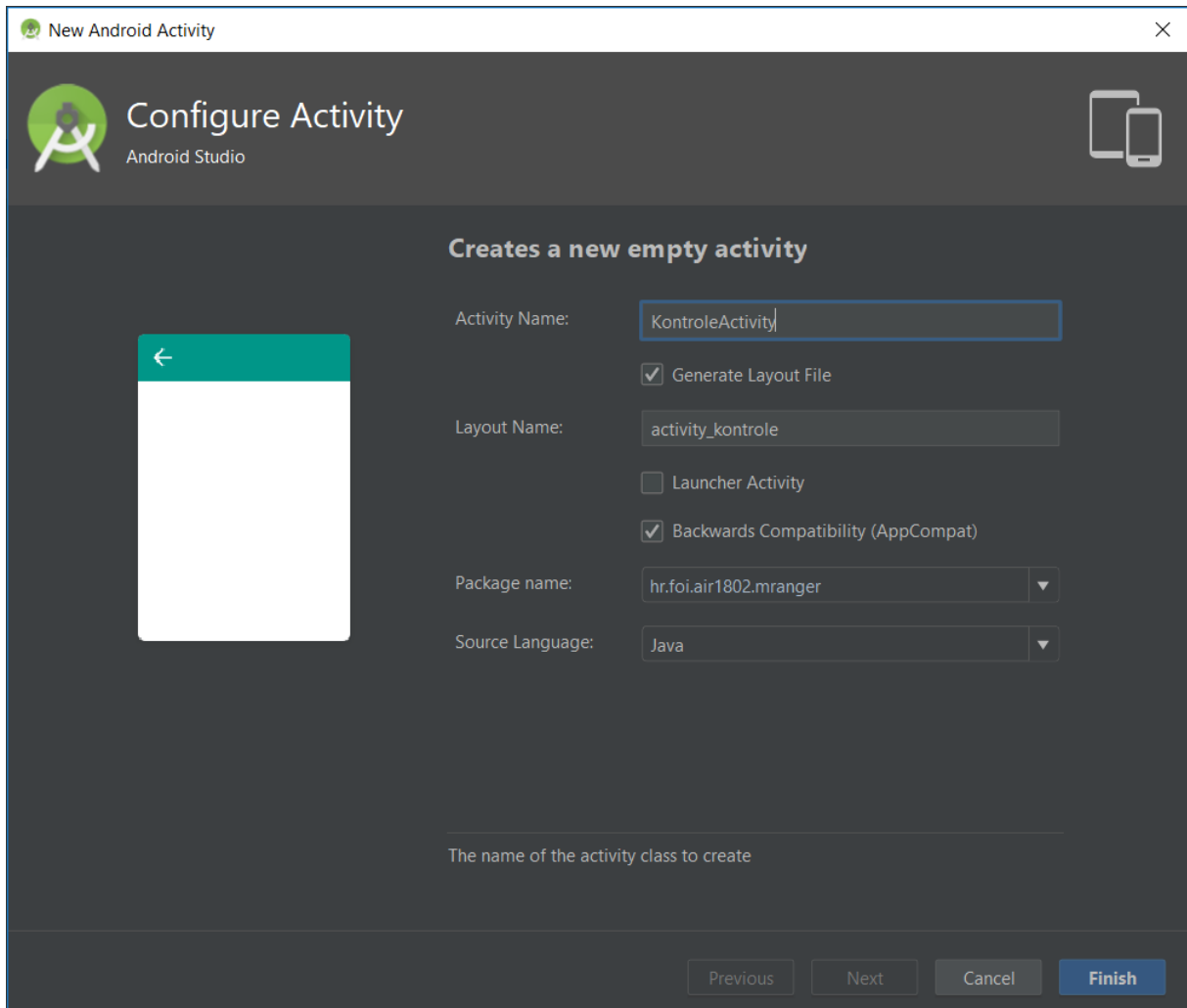
4. Dodavanje novog zaslona, slike i gumba

U prethodnom poglavlju stvorili smo i uredili početni zaslon sa pripadajućim datotekama, *MainActivity.java* i *activity_main.xml*. Pošto se najprije moramo spojiti Bluetooth-om na robota, taj početni zaslon će nam služiti za spajanje mobilnog uređaja na robot putem Bluetooth-a. Isto tako nakon spajanja, želimo imati zaslon putem kojeg ćemo upravljati robotom, tako da ćemo sada dodati novi zaslon u naš projekt.

4.1 Dodavanje novog zaslona

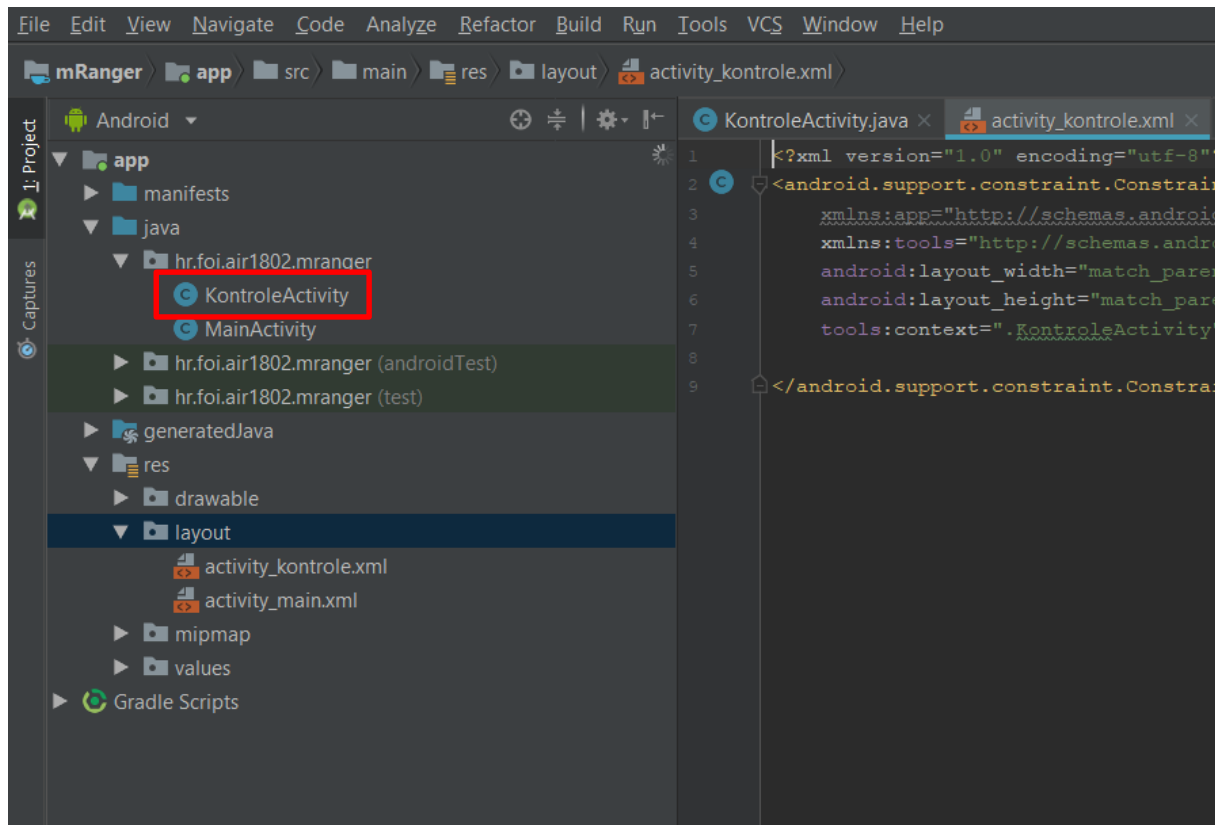


Slika 19. Kreiranje novog zaslona (nove aktivnosti)



Slika 20. Imenovanje novog zaslona (nove aktivnosti)

Novokreirani zaslona će sadržavati kontrole (joystick) za upravljanje mBot uređajem te ga stoga nazivamo *KontrolaActivity*. Razlog takvome nazivu je nekakva praksa da programeri u androidu različite zaslone smatraju i nazivaju aktivnostima.

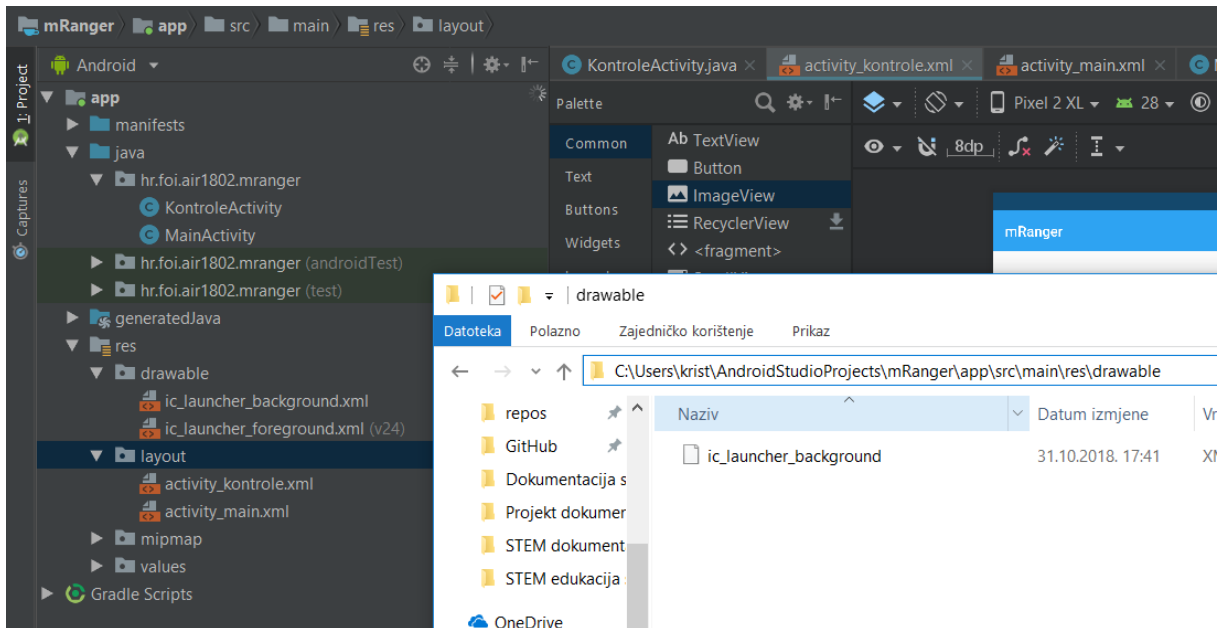


Slika 21. Stvoren novi zaslon

Uz svaki zaslon dolaze dvoje datoteke, jedna za *layout* u kojoj se uređuje izgled. Druga je *Java klasa* u kojoj se nalazi pripadajući kod koji će se izvršavati za taj zaslon.

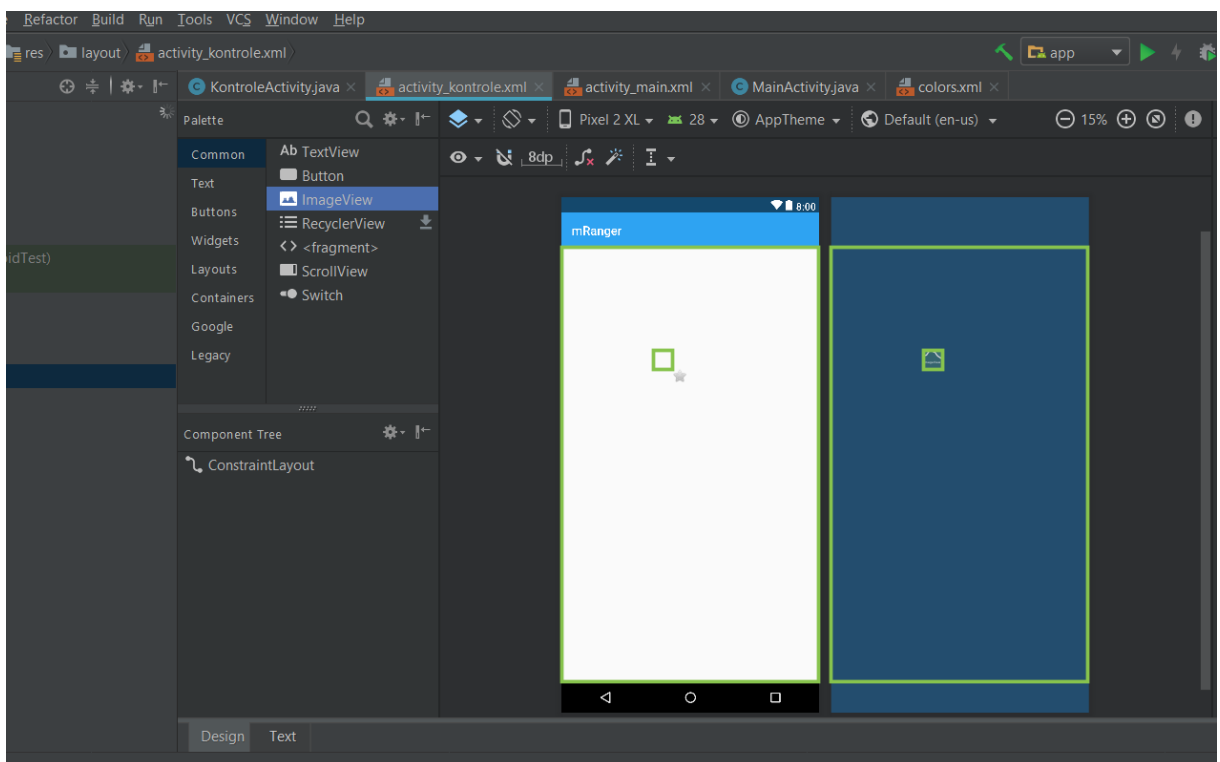
4.2 Dodavanje slike

U sjedećem poglavlju pokazati ćemo kako se slika dodaje na zaslon, te koji su sve koraci potrebni za to. Kako bi razlikovali ovaj novi zaslon, na njega ćemo staviti sliku koju ćemo poslije iskoristiti. Kako bi slike dodavali na zaslon u Android Studio-u, te slike moraju biti u posebnoj projektnoj mapi.



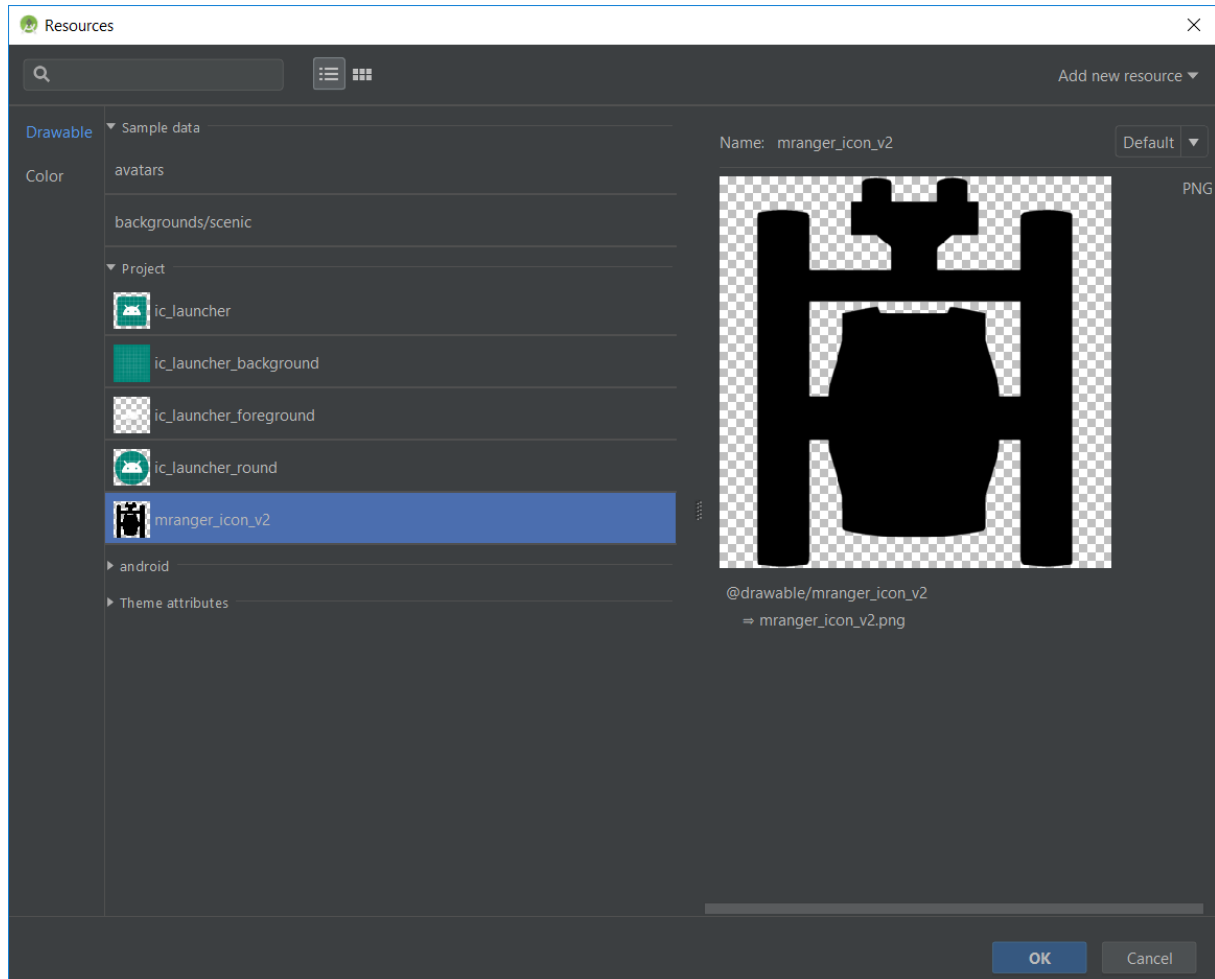
Slika 22. Dodavanje slike u mapu projekta

Na slici se također nalazi putanja do spomenute mape (mapa *drawable*) u koju prije dodavanja slike je potrebno postaviti željenu sliku. Ime slike mora obavezno biti napisano malim slovima, te ako se radi o više riječi, one moraju biti odvojene donjom crtom („_“).



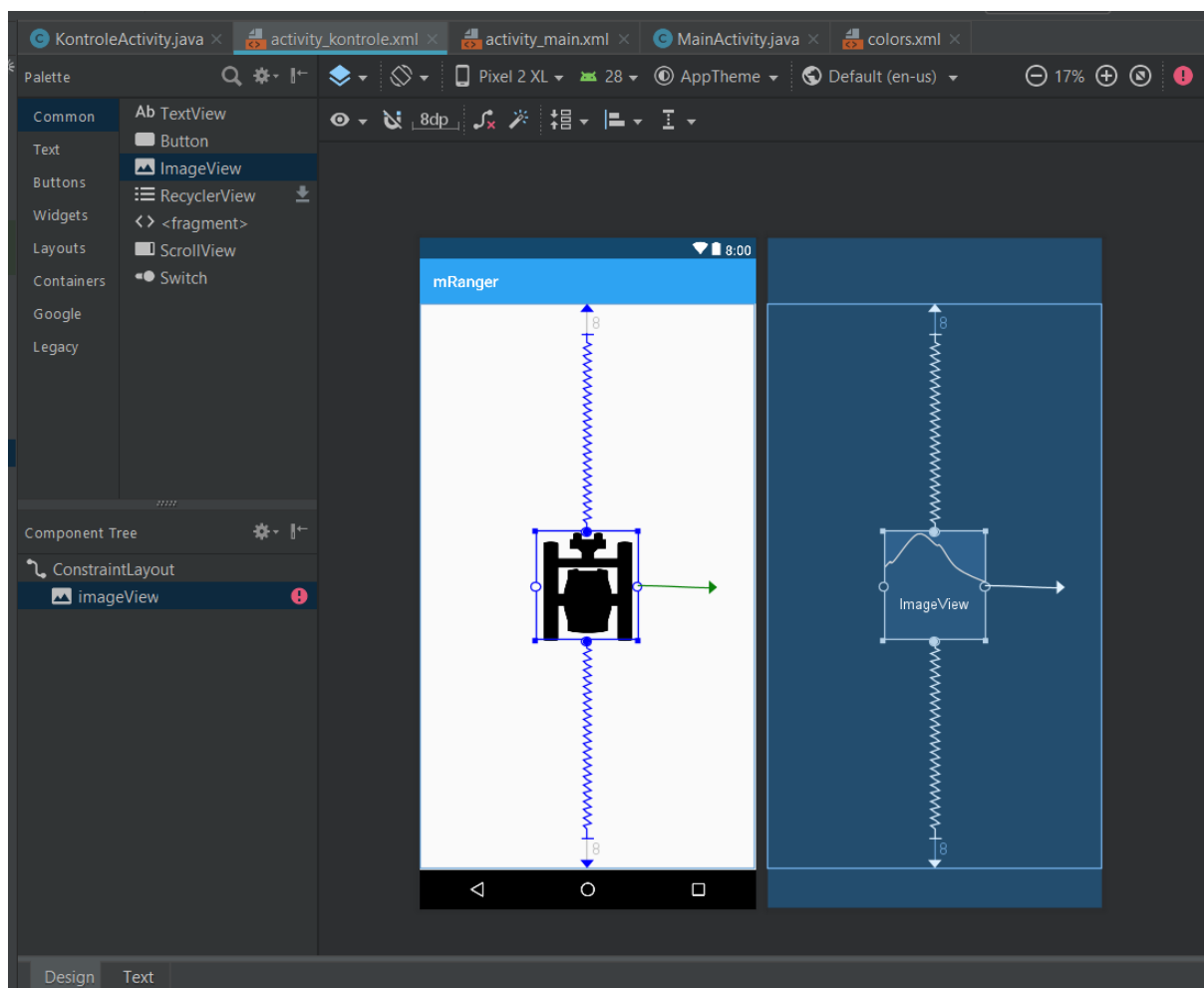
Slika 23. Dodavanje slike na zaslon

Kako bi dodali sliku na zaslou, unutar *Design* taba dodajemo kontrolu *ImageView* koja predstavlja okvir unutar koje će se slika prikazivati. Kontrola se na zaslou dodaje po principu povuci i spusti (*eng. drag and drop*).



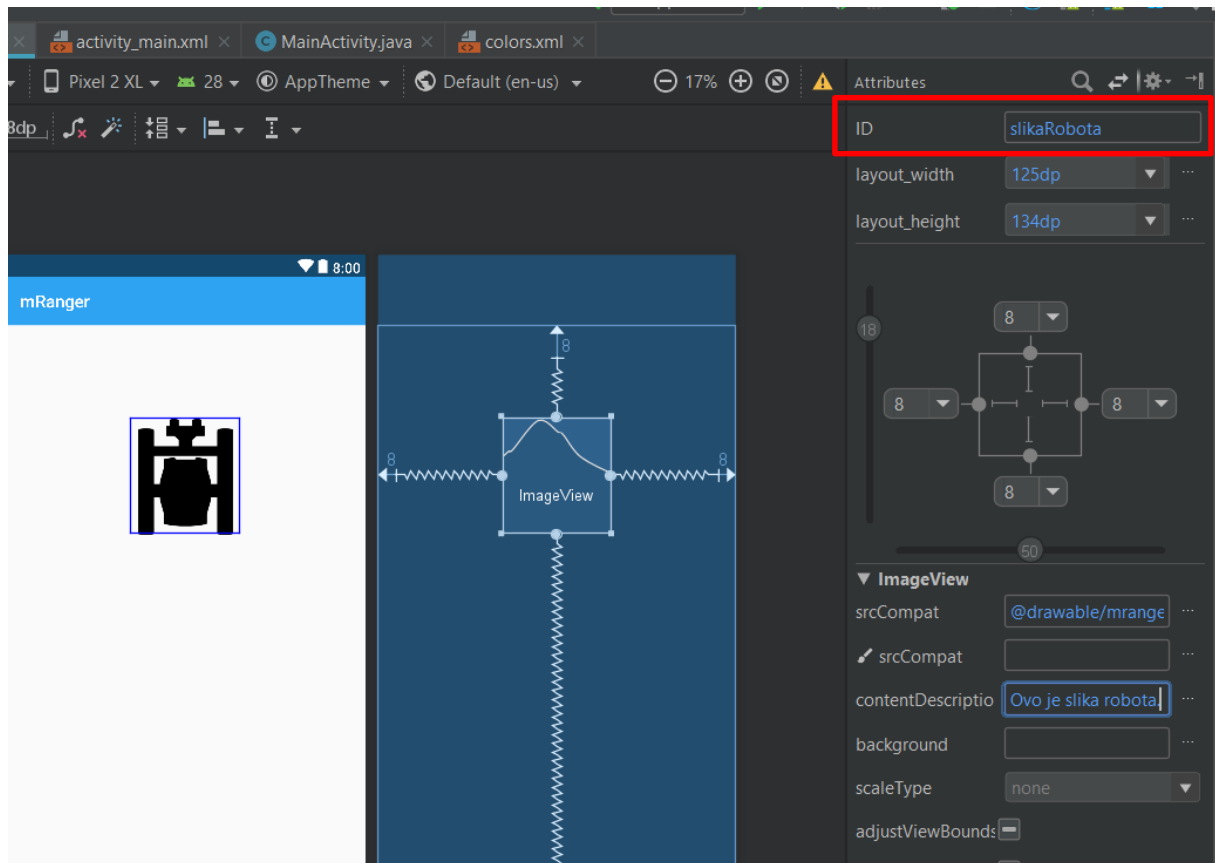
Slika 24. Odabir slike koja će se prikazati

Nakon ispuštanja kontrole, otvara nam se sljedeći prozor gdje biramo sliku koja će se prikazati u okviru. Tu ćemo na odabir imati neke predložene slike od Android Studio-a, kao i one koje smo dodali u *drawable* mapu.



Slika 25. Smještaj slike na zaslonu

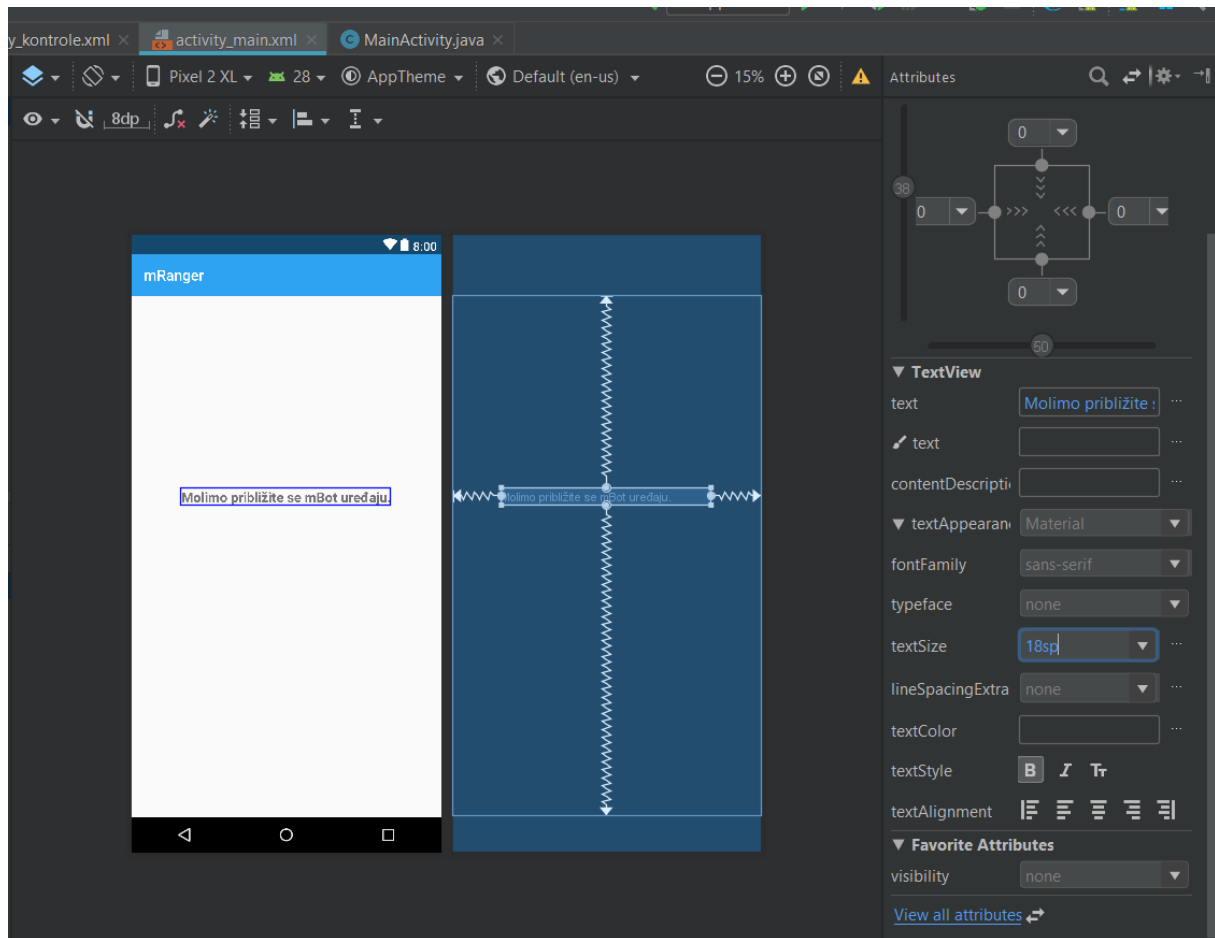
Kako bi se naša slika nalazila na željenom mjestu, a moramo uzeti u obzir da postoje zasloni različitih veličina, potrebno je sliku „usidriti“ sa svih njezinih strana okvira. Sliku usidravamo povlačenjem crte od točke sa sredine njezinog okvira pa sve do krajnjih rubova zaslona i/ili drugih kontrola.



Slika 26. Dodavanje id-a i opisa slike

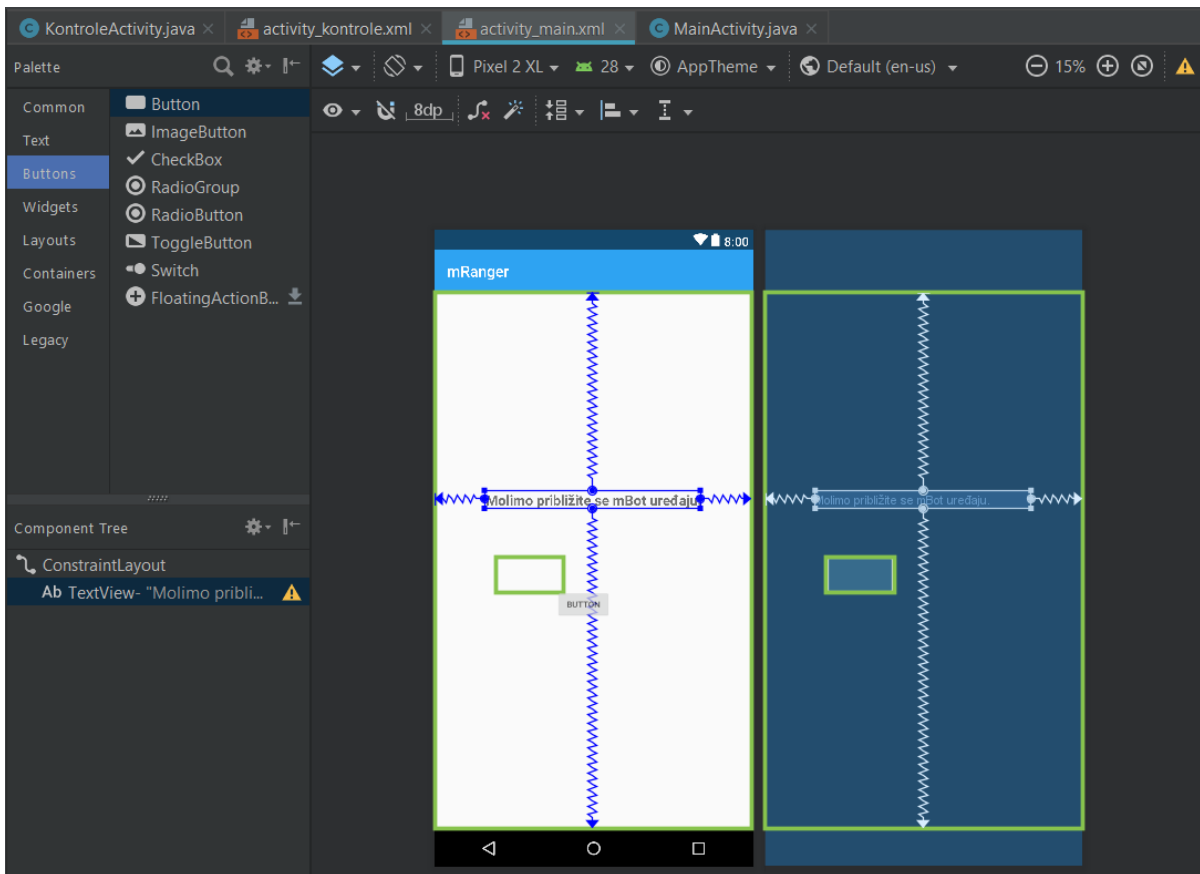
Novo dodanoj kontroli koja sadrži sliku dodajemo *id* kako bi je mogli prepoznati, te dajemo opis i ostale atribute po želji (proučite).

4.3 Dodavanje gumba



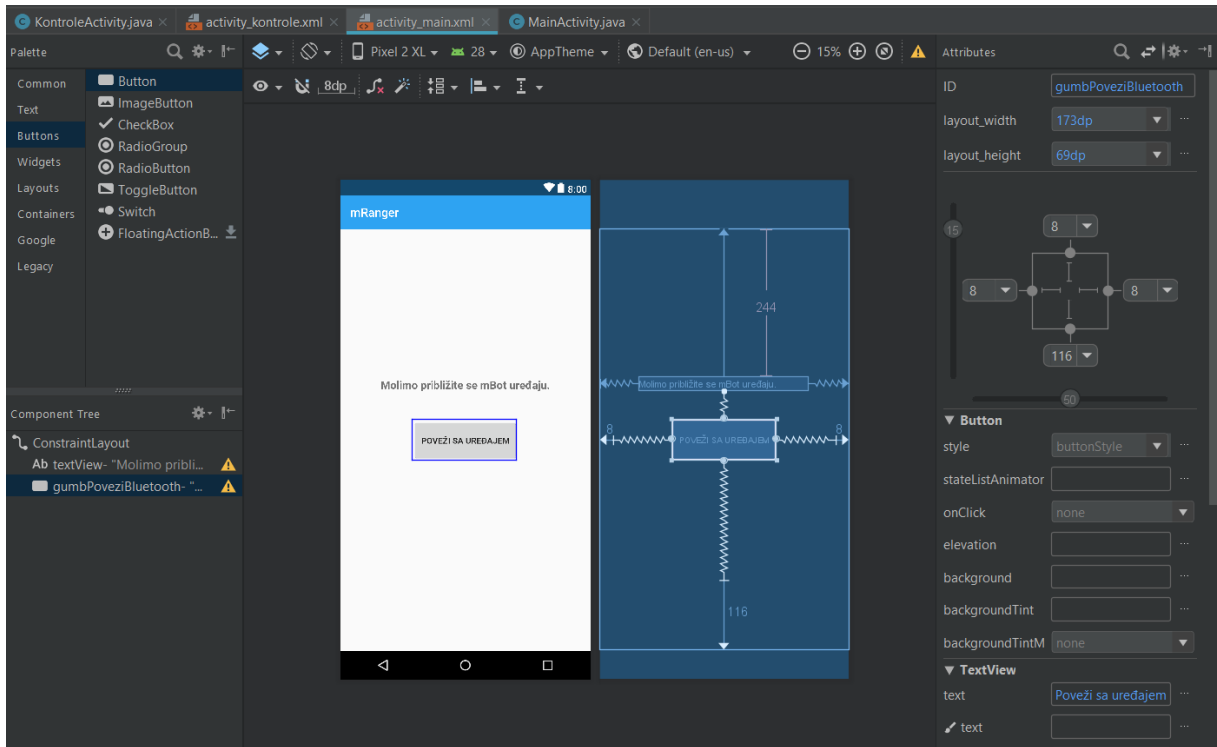
Slika 27. Sidrenje tekstualnog okvira

Iz istog razloga kao i kod slike sidrimo kontrolu tekstualnog okvira na željenu poziciju na zaslonu. Pošto je ovo naš početni zaslon, on se prvi otvara i na njemu se nalazi aktivnost povezivanja robota putem Bluetooth-a. Kako bi započeli proces povezivanja, dodajemo gumb pomoću kojeg ćemo to izvršiti.



Slika 28. Dodavanje gumba

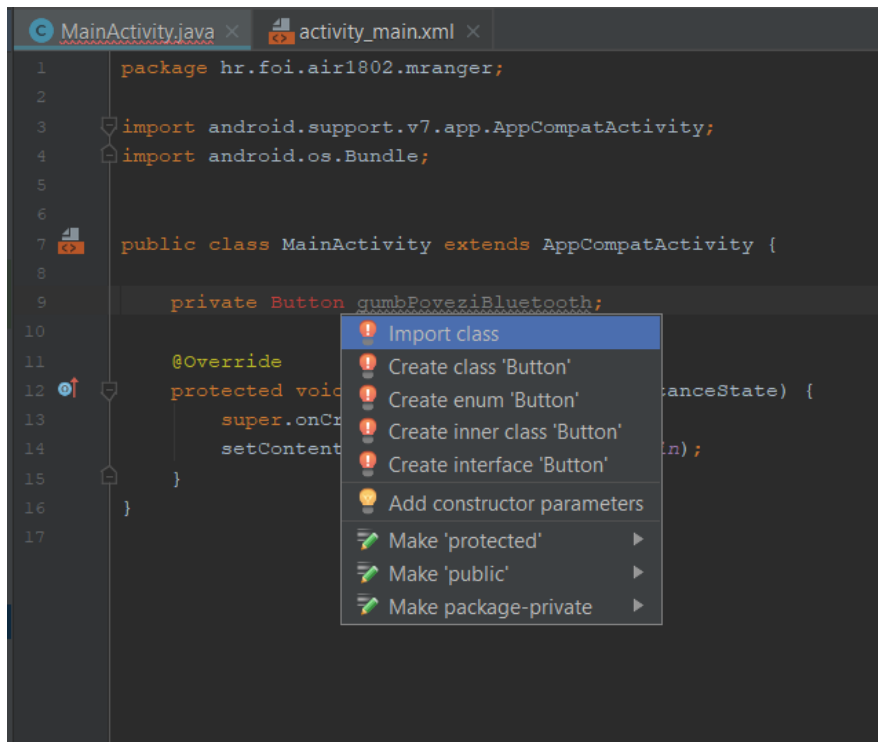
Istim principu kako smo dodali kontrolu za slikovni okvir, dodajemo kontrolu gumba na zaslon. Sidrimo je isto kao i kod kontrole slikovnog okvira. Također dodajemo joj *id* (taj naziv *id*-a će poslije biti jako važan u kodu), kao i tekst koji će se prikazivati na njemu, te ostale attribute po želji.



Slika 29. Kontrola za gumb

4.4 Prijelaz s jednog zaslona na drugi

Polozicioniramo se unutar *MainActivity.java* datoteke, te u njoj pišemo pripadajući kod za prijelaz na zaslom *KontrolaActivity.java*.

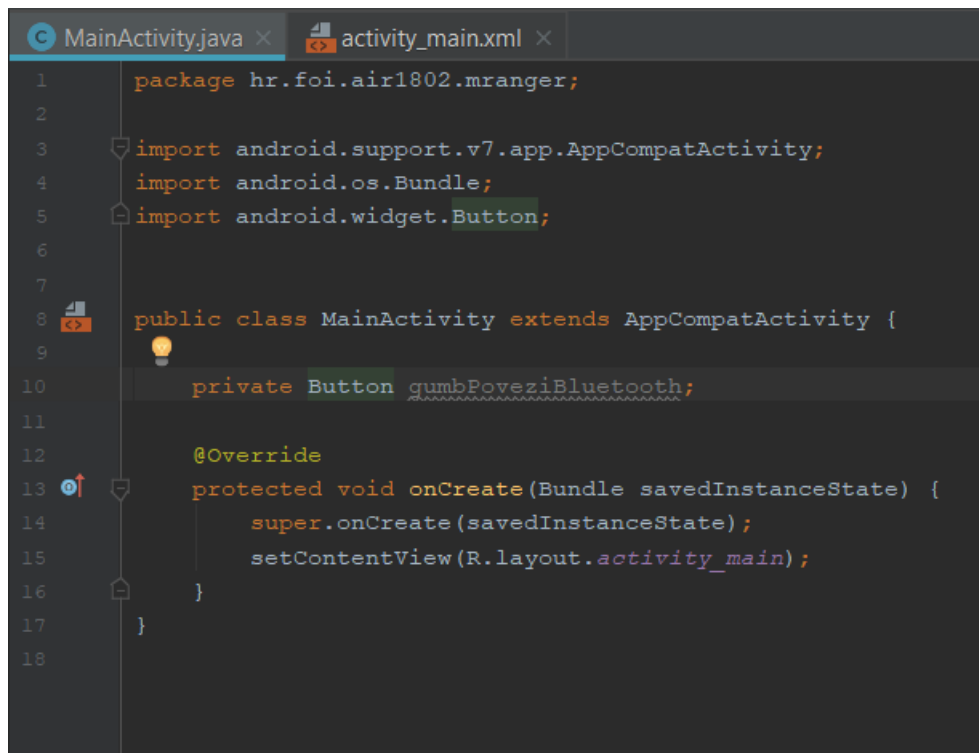


Slika 30. ALT + Enter kombinacija tipaka

Napomena: ova kombinacija tipki je vrlo važna, ona omogućuje brzo izvršavanje unaprijed predviđenih opcija, te služi kao prva opcija koju ćemo poduzeti ukoliko dođe do nekih greški.

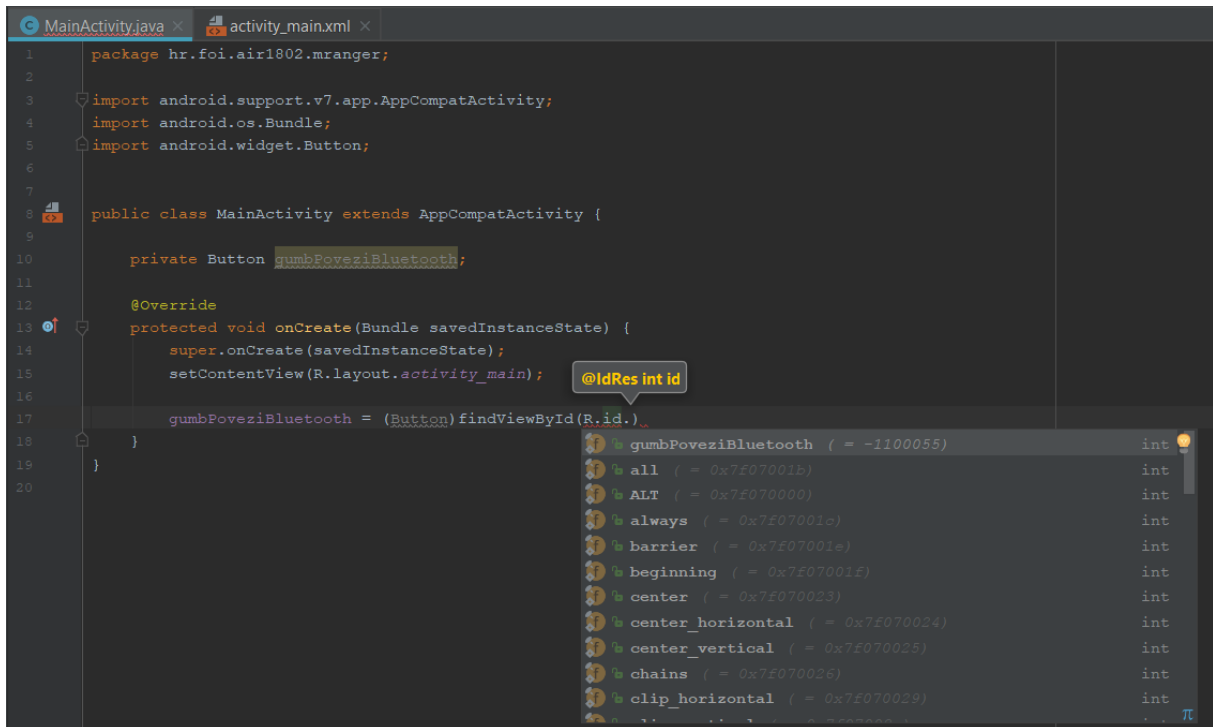
Na primjer, na slici poviše je napisana sljedeća linija koda gdje stvaramo novu privatnu varijablu tipa *Button* koju nazivamo po želji, ali zbog jednostavnosti isto onako kao što je i *id* našeg gumba.

Vidimo kako se pojavila greška kod tipa varijable jer nismo posebno uključili biblioteku koja se koristi za rad s tim tipom kontrole, te jednostavnim *Alt + Enter* dobivamo nekoliko mogućih opcija za rješavanje tog problema. U ovom slučaju je potrebno odabrati *Import Class* kako bi dodali biblioteku i otklonili pogrešku.



```
1 package hr.foi.air1802.mranger;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.widget.Button;
6
7
8 public class MainActivity extends AppCompatActivity {
9     private Button gumbPoveziBluetooth;
10
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16     }
17 }
18
```

Slika 31. Dodana biblioteka



```

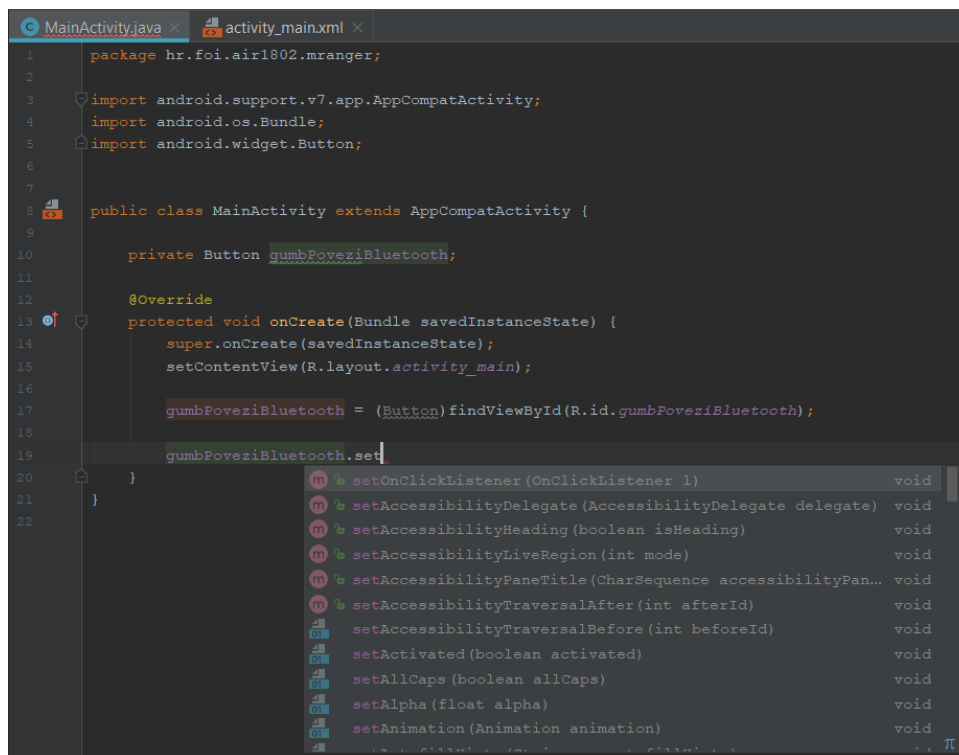
1 package hr.foi.air1802.mranger;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.widget.Button;
6
7
8 public class MainActivity extends AppCompatActivity {
9
10     private Button gumbPoveziBluetooth;
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16
17         gumbPoveziBluetooth = (Button) findViewById(R.id.);
18     }
19 }

```

The screenshot shows an IDE with MainActivity.java open. The code defines a MainActivity class that extends AppCompatActivity. It has a private Button variable named gumbPoveziBluetooth. In the onCreate method, it calls findViewById(R.id.) to find the button. An autocomplete popup is visible over the R.id. parameter, listing various resource IDs such as gumbPoveziBluetooth, all, ALT, always, barrier, beginning, center, center_horizontal, center_vertical, chains, and clip_horizontal.

Slika 32. Dohvaćanje gumba preko id-a

Varijabli koju smo prethodno stvorili u kodu dajemo vrijednost, odnosno povezujem je sa kontrolom pripadajućeg gumba koji se nalazi na našem zaslonu. Primjetimo kako korištenje točke kod određenih elemenata nam također daje više mogućnosti što olakšava tipkanje koda kao i pronalaženje one prave.



```

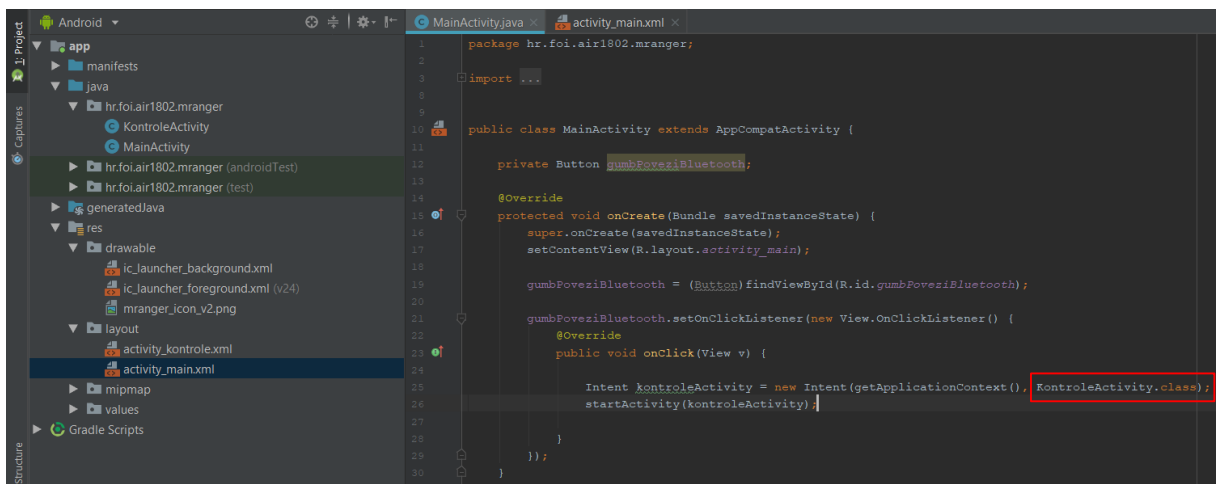
1 package hr.foi.air1802.mranger;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.widget.Button;
6
7
8 public class MainActivity extends AppCompatActivity {
9
10     private Button gumbPoveziBluetooth;
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16
17         gumbPoveziBluetooth = (Button) findViewById(R.id.gumbPoveziBluetooth);
18
19         gumbPoveziBluetooth.set
20     }
21 }

```

The screenshot shows the same MainActivity.java code as in Slika 32. The code is now more complete, with the findViewById call using the full resource ID R.id.gumbPoveziBluetooth. In the onCreate method, the line gumbPoveziBluetooth.set is being typed, and an autocomplete popup is showing a list of methods available for the Button object, including setOnClickListener, setAccessibilityDelegate, setAccessibilityHeading, setAccessibilityLiveRegion, setAccessibilityPansTitle, setAccessibilityTraversalAfter, setAccessibilityTraversalBefore, setActivated, setAllCaps, setAlpha, and setAnimation.

Slika 33. Dodavanje setOnClickListener metode

Dodavanje *setOnClickListener* metode na gumb omogućujemo izvršavanje aktivnosti koje ćemo mi odrediti prilikom klika na određeni gumb na zaslonu.



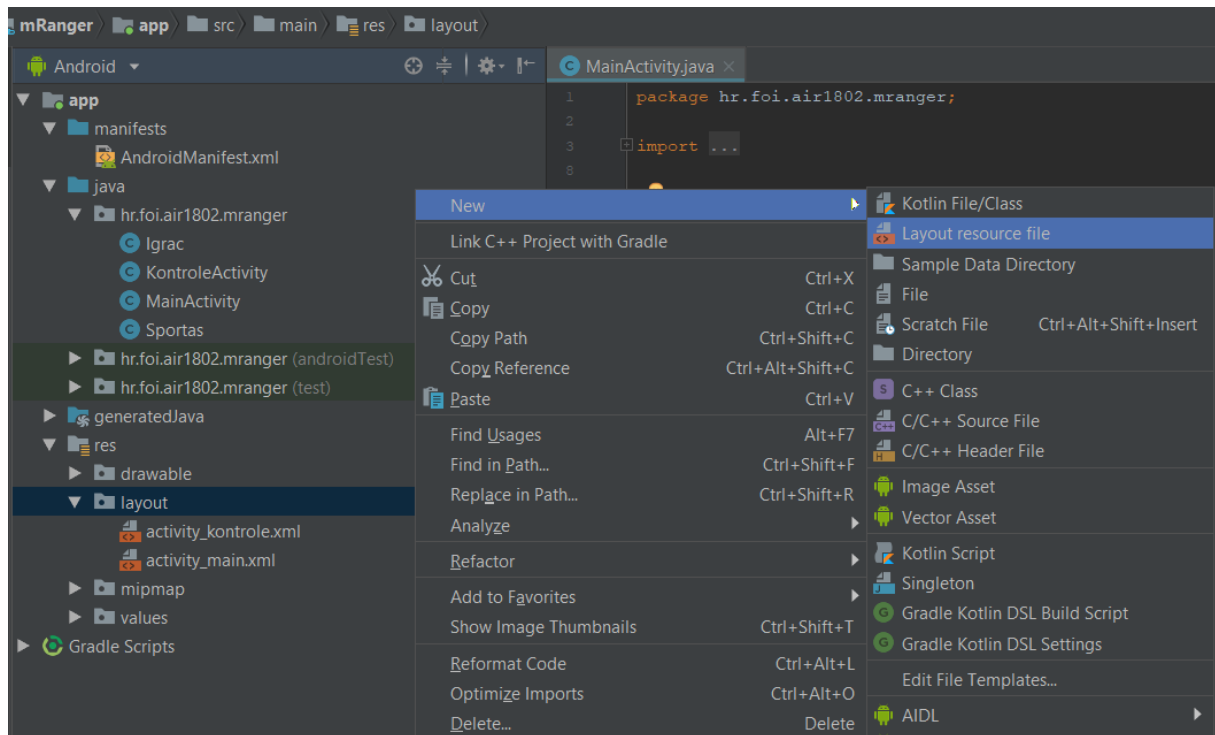
Slika 34. Dodavanje koda koji se izvršava na klik gumba

Unutar *onClick* metode nalazi kod pomoću kojeg se prilikom pritiska na gumb prelazi sa početnog zaslona na željeni, što je u ovom slučaju *KontroleActivity.java*.

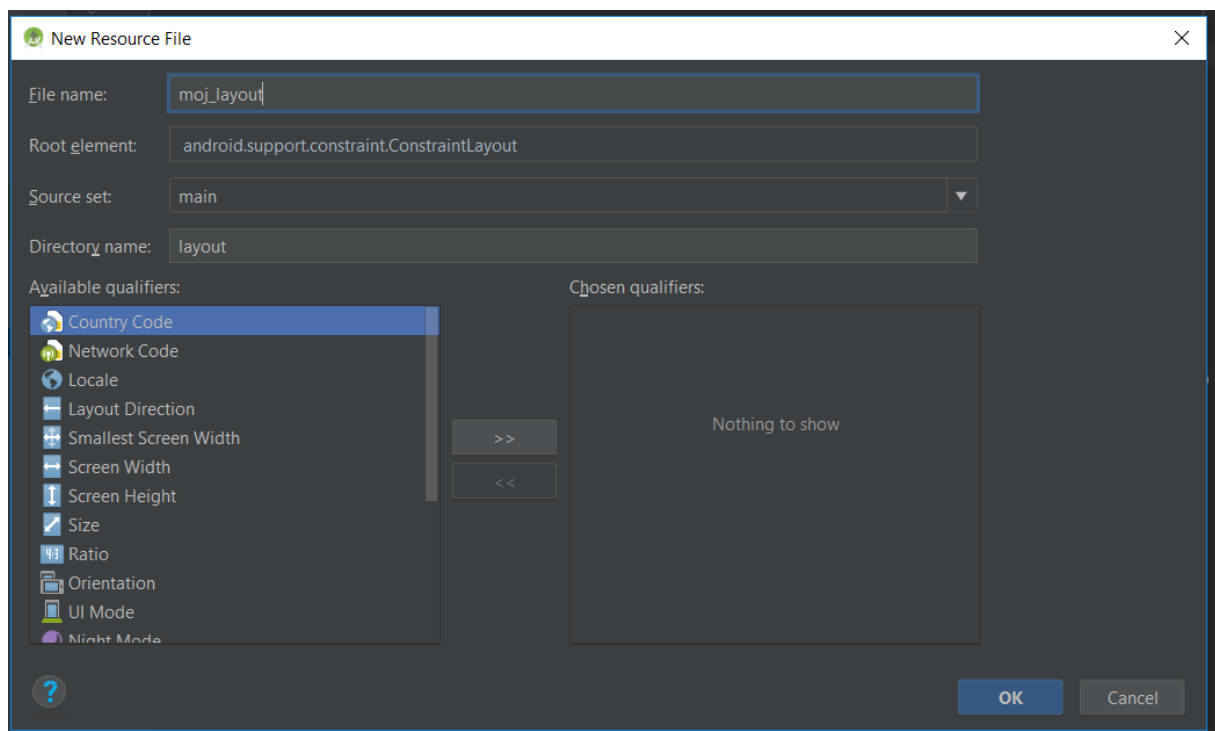
4.5 Kreiranje layout-a

Još jedna vještina koja će Vam biti potrebna za izradu ovog projekta je dodavanje nove *layout* datoteke. Kao što ste i dosad već mogli vidjeti, kreiranjem nekog novog *activity-a*, kreira se i njegova datoteka za izgled tipa *.xml*, koja se stavlja u **layout** mapu. Ta *layout* datoteka služi kako bi svojim kodom vizualno definirala izgled tog zaslona.

Međutim, ta *layout* datoteka se može i samostalno napraviti kako bi definirali izgled nekih drugih elemenata. Mi ćemo dalje u kodu definirati izgled *ListView* elementa (liste koja će prikazivati detektirane Bluetooth uređaje) te ćemo za njega morati izraditi posebnu *layout* datoteku.

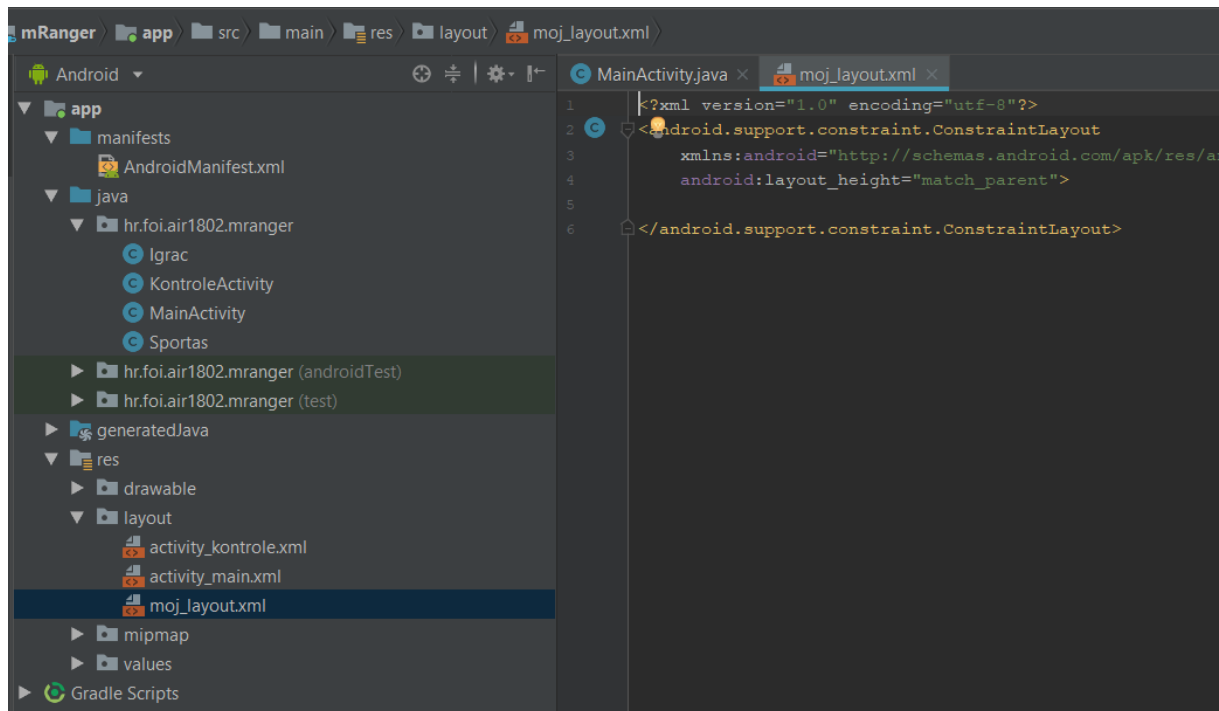


Slika 35. Dodavanje nove layout resource datoteke



Slika 36. Imenovanje layout resource datoteke

Dajemo datoteci željeno ime i dalje stisnemo OK. (Ne diramo ništa više 😊)



Slika 37. Dodana nova layout resource datoteka

Ovako bi trebao izgledati rezultat kreiranja nove *layout resource* datoteke, a kod koji će potrebno biti dodati, moći će te kopirati također iz ovog dokumenta kasnije kada dodemo do tog dijela.

5. Objekti i klase

Sljedeća stvar koju Vam moramo objasniti prije nego li možemo nastaviti sa izradom aplikacije, su pojmovi **klasa**, **objekt** i **metoda**.

Objekt u realnom svijetu, može biti recimo: automobil ili bicikl, a u programskoj igri lopta ili neki lik.

Svaki objekt je definiran stanjem i ponašanjem; na primjer, stanje lika u igri mogu biti **atributi** poput: broj godina, ime, prezime, visina, težina itd., a ponašanje lika možemo definirati **metodama** poput : Trči, Skači, Spavaj. Ispod slijedi dio koda kojim ćemo to bolje objasniti.

```
public class Sportas {  
    public Integer BrojGodina;  
    public String Ime;  
    public String Prezime;  
  
    public Float Visina;  
    public Float Tezina;  
  
    public void JediRucak(Sportas sportas)  
    {  
        // Pretpostavili smo da nakon ručka uvijek dobije 1 kg dodatne težine  
        sportas.Tezina = sportas.Tezina + 1;  
    }  
    public void Narasti(Sportas sportas, float centimetri)  
    {  
        //Možemo prosljediti željenog sportaša i željeni iznos  
        //za koji će narasti.  
        sportas.Visina = sportas.Visina + centimetri;  
    }  
    public void Trci()  
    {  
        // Ovdje napišemo što se točno događa prilikom trčanja sportaša  
        // Naravno, ovo je samo u cilju boljeg objašnjenja pojma jer ne možemo  
        // baš natjerati varijablu neku da trči. :)  
    }  
}
```

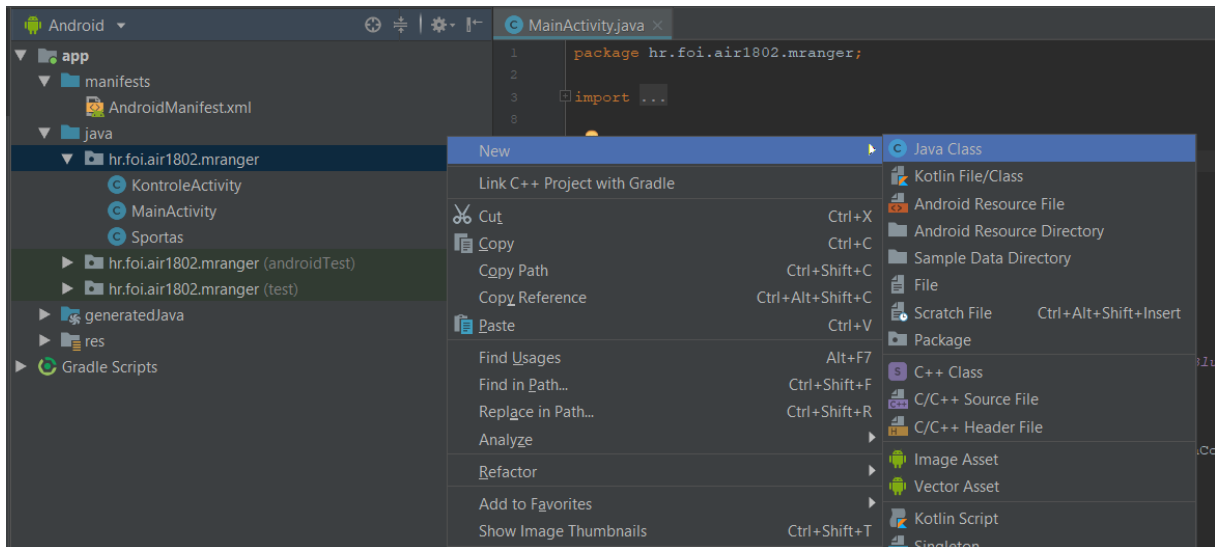
(Dvostrukim klikom na kod možete ga kopirati)

U programu je objekt opisan varijablama koje mu određuju stanje i metodama koje mu određuju ponašanje.

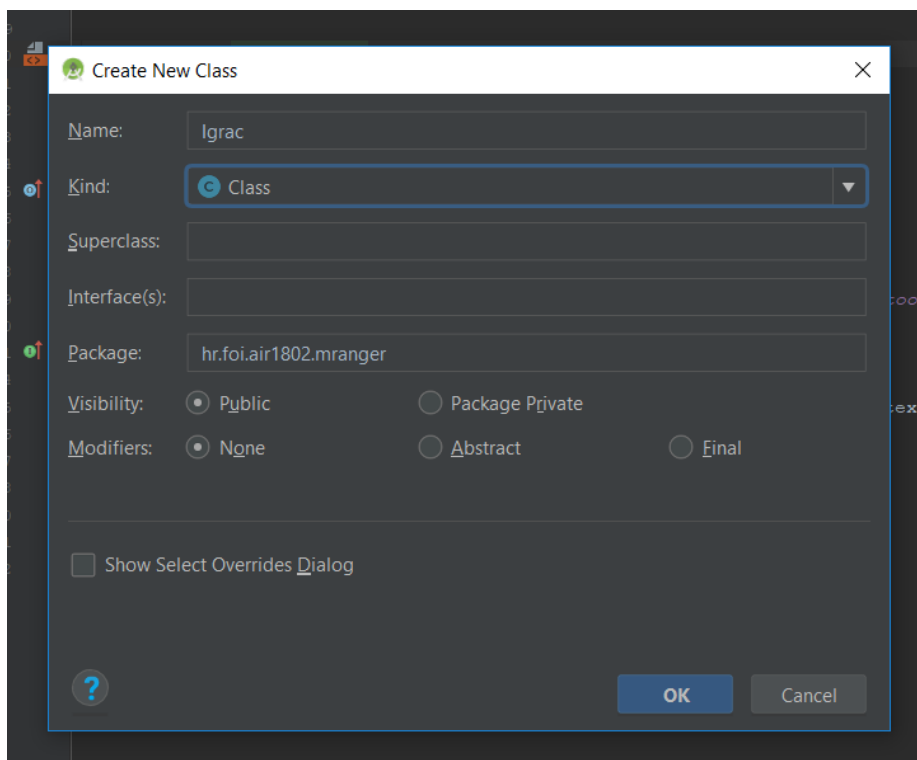
Klasa predstavlja nacrt (predložak) objekta (eng. class). Stoga se uvijek prvo kreiraju klase na temelju kojih se proizvode objekti. Na primjer, kada kreiramo klasu za sportaša, u igri možemo kreirati više likova koji mogu imati drugačije attribute (različite godine, imena, visine ...) i ponašanja (drugačije metode).

5.1 Dodavanje nove klase u Android Studio-u

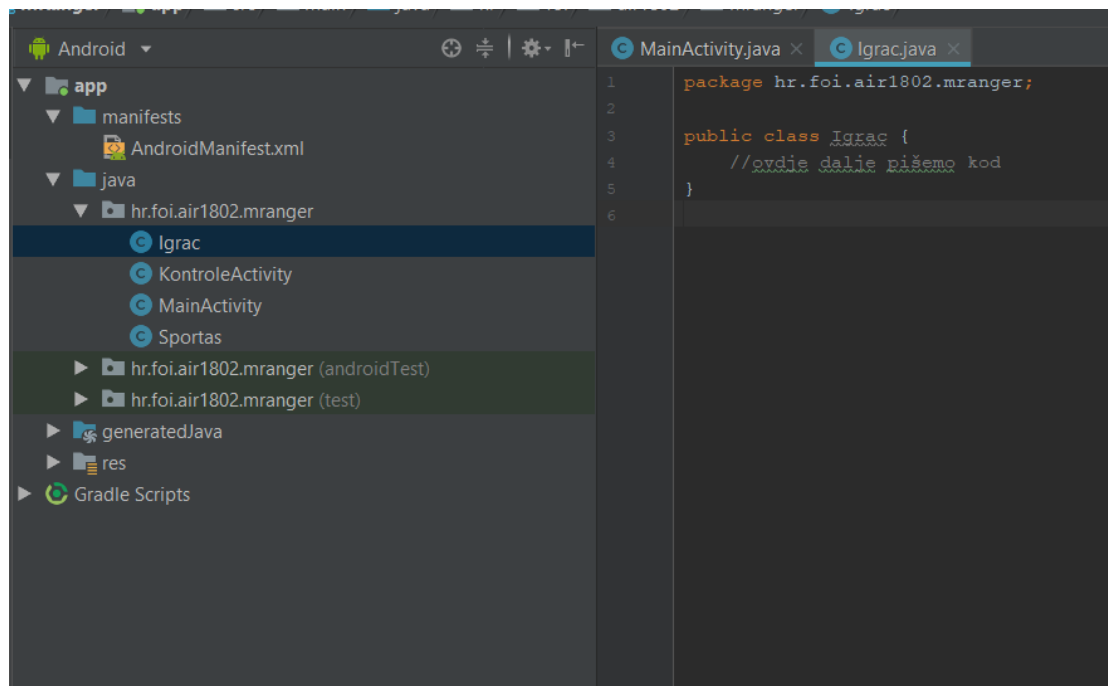
U ovom odlomku ćemo Vam prikazati kako se kreiraju i dodaju nove klase u projekt. Na slikama možete popratiti pojedine korake.



Slika 38. Dodavanje nove klase u projekt



Slika 39. Unosimo željeno ime i tip: Class



Slika 40. Dodana je nova klasa u projekt

Od ovog trenutka, upoznati ste sa svim potrebnim predznanjima kako bi mogli kreirati ovu aplikaciju prateći naše daljnje upute, te svi koraci od sada pa na dalje se baziraju na ovim prethodnim objašnjenim.

Naravno, sav kod ćemo Vam mi objasniti i priložiti kako bi ga Vi mogli zalijepiti na predviđena mjesta.

6. Izrada mRanger aplikacije

Sada slijedi najbitniji dio ovog edukacijskog dokumenta, a to je zapravo izrada aplikacije mRanger. Kako smo svjesni da ste vi još uvijek novi u radu sa Android Studiom, te možda izradom aplikacije bilo kakve vrste uopće, ovo ćemo vam nastojati objasniti na najjednostavniji mogući način.

Prvo ćemo od Vas zatražiti da izradite potrebne *Activity-e* odnosno zaslone, potrebne **klase** te potrebne **resource layout** datoteke, a zatim ćemo Vam priložiti kod koji ćete moći kopirati u te datoteke kako bi aplikacija radila.

Sav kod, sve klase i metode koje smo izradili smo Vam objasnili te ćete moći shvatiti i zašto su oni tu gdje jesu, odnosno koja je njihova zadaća u ovoj cijeloj priči.

Kad smo sve to rekli, krenimo na posao. 😊

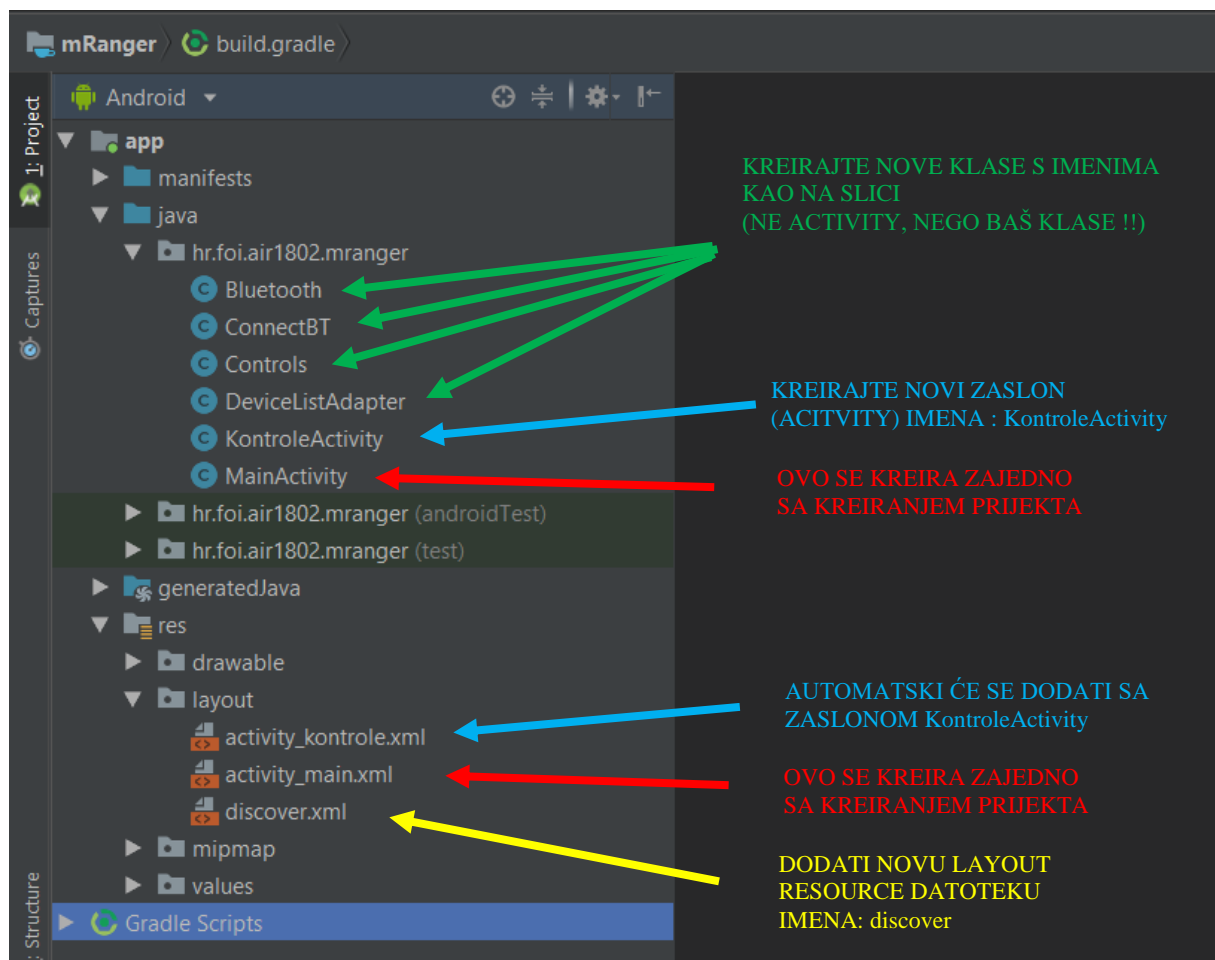
POTREBNO JE PAŽJIVO PRATITI UPUTE !

SRETNO ♥

6.1 Kreiranje potrebnih datoteka

Prije nego što počnete sa izradom ovog projekta, zatvorite trenutni projekt na kojem smo dosad radili i pokazivali osnove rada u Android Studio-u, te kreirajte novi projekt gdje ćete pod *Application name* staviti **mranger**, te pod *Company domain* staviti **air1802.foi.hr**, kako ne bih morali mijenjati niti jednu liniju prilikom kopiranja koda, jer smo ga i mi također nazvali na taj način.

(*Project location* možete odabrati po vašoj volji)



Slika 41. Popis datoteka projekta

Pobrinite se da izradite sve poviše navedene datoteke sa pravim imenom. Nakon što ih kreirate, ne morate se brinuti za njihov sadržaj, odnosno kod jer će te unutra sve izbrisati prije nego što kopirate naš.

6.2 Dodavanje koda u klase

Prethodno smo kreirali 4 klase : **Bluetooth**, **ConnectBT**, **Controls** i **DeviceListAdapter**.

Sada ćemo u jednu po jednu od njih zalijepiti pripadajući kod.

- 1) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **Bluetooth** klasu, iz koje se prethodno tome sve obrisali.

```
package hr.foi.air1802.mranger;

import android.Manifest;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.support.v4.app.ActivityCompat;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;

/**
 * Klasa koja služi za uključivanje/isključivanje Bluetootha, otkirvanje uređaja i
 * stvaranje konekcije
 */
public class Bluetooth {

    public static BluetoothAdapter myBluetoothAdapter;
    public static ArrayList<BluetoothDevice> myBluetoothDevices;
    public static ListView listaDiscoveredDevices;
    public static DeviceListAdapter myDeviceListAdapter;
    private static String deviceAddress;
    public static String EXTRA_ADDRESS = "device_address";

    /**
     * Metoda koji služi za stvaranje novog Bluetooth adaptera
     */
    public static void createBluetoothAdapter() {
        myBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    }

    /**
     * Metoda koja služi za uključivanje/isključivanje Bluetootha
     * @param context - parametar u kojem prosljeđujemo trenutni kontekst
     */
    public static void enableDisableBluetooth(Context context) {

        if (myBluetoothAdapter == null) { //ne podržava
            hr.foi.air1802.mranger.Bluetooth
            Toast.makeText(context, "Ovaj uređaj ne podržava Bluetooth
konekciju.", Toast.LENGTH_LONG).show();
        }
        if (!myBluetoothAdapter.isEnabled()) { //hr.foi.air1802.mranger.Bluetooth nije
uključen
            Intent enableBTIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE); //uključiti
            context.startActivity(enableBTIntent);
        }
        if (myBluetoothAdapter.isEnabled()) { //hr.foi.air1802.mranger.Bluetooth
uključen
            myBluetoothAdapter.disable(); //isključiti

            myBluetoothDevices.clear(); //ukoliko smo već otkrili neke, čistimo
            listaDiscoveredDevices.setAdapter(null);

            Toast.makeText(context, "Bluetooth isključen.", Toast.LENGTH_LONG).show();
        }
    }
}
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

- 2) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **ConnectBT** klasu, iz koje se prethodno tome sve obrisali.

```
package hr.foi.air1802.mranger;

import android.Manifest;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.os.AsyncTask;
import android.support.v4.app.ActivityCompat;
import android.widget.Toast;

import java.io.IOException;

/**
 * Klasa koja služi za ostvarivanje i upravljanje Bluetooth konekcijom nad uređajem
 */
public class ConnectBT extends AsyncTask<Void, Void, Void> // UI thread
{
    Context context;
    Activity activity;

    /**
     * Konsturktor metode
     * @param context - kontekst klase KontrolaActivity.java
     * @param activity - aktivnost klase KontrolaActivity.java
     */
    public ConnectBT(Context context, Activity activity){
        this.context=context;
        this.activity=activity;
    }

    private boolean ConnectSuccess = true; //ako smo došli do ovdje, skoro smo se spojili

    /**
     * Stvaranje bluetooth veze između uređaja.
     * @param devices - parametar kojim se prosljeđuje uređaj
     * @return - metoda uvijek vraća null
     */
    @Override
    protected Void doInBackground(Void... devices) //varijabilan broj parametara, dobro je to tako
    {
        try {
            if (Controls.bluetoothSocket == null || !Controls.isBluetoothConnected)
                //ako nismo spojeni
                {
                    ActivityCompat.requestPermissions(activity, new
                    String[]{Manifest.permission.BLUETOOTH}, 1);
                    ActivityCompat.requestPermissions(activity, new
                    String[]{Manifest.permission.BLUETOOTH_ADMIN}, 1);
                    ActivityCompat.requestPermissions(activity, new
                    String[]{Manifest.permission.BLUETOOTH_PRIVILEGED}, 1);

                    Controls.myBluetoothAdapter =
                    BluetoothAdapter.getDefaultAdapter();//dohvati naš bluetooth uređaj

                    BluetoothDevice bluetoothRobot =
                    Controls.myBluetoothAdapter.getRemoteDevice(Controls.address);

                    //spaja se na adresu uređaja i provjerava da li je slobodna
                    Controls.bluetoothSocket =
                    bluetoothRobot.createInsecureRfcommSocketToServiceRecord(Controls.myUUID);//stvara
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

- 3) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **Controls** klasu, iz koje se prethodno tome sve obrisali.

```
package hr.foi.air1802.mranger;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.view.MotionEvent;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import java.io.IOException;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

/**
 * Klasa koja služi za upravljanje kontrolama na robotu
 */
public class Controls {

    public static String address = null;
    public static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    public static BluetoothAdapter myBluetoothAdapter = null;
    public static BluetoothSocket bluetoothSocket = null;
    public static boolean isBluetoothConnected = false;
    private static float globTemp;
    private static boolean isTemperatureRead=false;
    private static float proslaTemp;

    //podaci za kretanje
    private static byte[] cmd = new byte[13];
    public static final int WRITEMODULE = 2;
    public static final int type = 5;
    private static int DesniMotor = 180;
    private static int LijeviMotor = 180;

    /**
     * Metoda u kojoj se šalju podaci za kretanje robota
     * @param lijeviMotor brzina lijevog motora
     * @param desniMotor brzina desnog motora
     */
    private static void move(int lijeviMotor, int desniMotor) {
        cmd[0] = (byte) 0xff;
        cmd[1] = (byte) 0x55;
        cmd[2] = (byte) 8;
        cmd[3] = (byte) 0;
        cmd[4] = (byte) WRITEMODULE;
        cmd[5] = (byte) type;
        final ByteBuffer buf = ByteBuffer.allocate(4);
        buf.putShort((short) lijeviMotor);
        buf.putShort((short) desniMotor);
        buf.position(0);
        // Read back bytes
        final byte b1 = buf.get();
        final byte b2 = buf.get();
        final byte b3 = buf.get();
    }
}
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

- 4) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **DeviceListAdapter** klasu, iz koje se prethodno tome sve obrisali.

```
package hr.foi.air1802.mranger;

import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.TextView;

import java.util.ArrayList;

/**
 * Klasa DeviceListAdapter služi za prikupljanje svih pronađenih bluetooth uređaja te
 * prikaz istih u Listi.
 * Lista se prikazuje na početnom zaslonu, a puni se na pritisak tipke OTKRIJ UREĐAJE.
 */
public class DeviceListAdapter extends ArrayAdapter<BluetoothDevice> {

    private ArrayList<BluetoothDevice> myDevices;
    private LayoutInflater myLayoutInflater;
    private int myResourceId;

    /**
     * Konstruktor za klasu DeviceListAdapter
     * @param context - parametar u koji prosljeđujemo trenutni kontekst
     * @param resource - parametar u koji prosljeđujemo listu u kojoj se prikazuju
     * uređaji
     * @param devices - parametar u koji se prosljeđuje lista pronađenih uređaja. Oni
     * koji se prikazuju za odabir na zaslonu
     */
    public DeviceListAdapter(Context context, int resource, ArrayList<BluetoothDevice>
    devices) {
        super(context, resource, devices);

        this.myDevices = devices;
        myLayoutInflater = (LayoutInflater)
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        myResourceId = resource;
    }

    /**
     * Metoda koja prikazuje pronađene Bluetooth uređaje u listi, odnosno postavlja ih
     * u listu.
     * @param position - parametar koji označava poziciju uređaja u listi, tj. njegov
     * index.
     * @param convertView - View u kojem se prikazuju pronađeni uređaji.
     * @param parent - grupa u kojoj se nalazi naš View.
     * @return
     */
    public View getView(int position, View convertView, ViewGroup parent) {

        convertView = myLayoutInflater.inflate(myResourceId, null);

        BluetoothDevice device = myDevices.get(position);

        if (device != null) {
            TextView deviceName = convertView.findViewById(R.id.labelDeviceName);
            TextView deviceAdress = convertView.findViewById(R.id.labelDeviceAddress);

            if (deviceName != null) {
                deviceName.setText(device.getName());
            }
        }
    }
}
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

6.3 Dodavanje koda u activity datoteke

Prethodno smo kreirali 2 activity-a : **MainActivity** i **KontrolaActivity**.

Sada ćemo u jednu po jednu od njih zalijepiti pripadajući kod.

- 1) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **MainActivity** datoteku, iz koje se prethodno tome sve obrisali.

```
package hr.foi.air1802.mranger;

import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.Button;
import java.util.ArrayList;

/**
 * Početni zaslon koji se otvara prilikom pokretanja aplikacije.
 * MainActivity nasljeđuje klasu AppCompatActivity te njegove metode i klase.
 * Na zaslonu je omogućeno paljenje i gašenje Bluetooth-a te pronalazak uređaja na
 * kojih se može povezati.
 */
public class MainActivity extends AppCompatActivity implements
    AdapterView.OnItemClickListener {

    //definiranje varijabli
    private Button gumbBTOnOff;
    private Button gumbDiscover;

    /**
     * Metoda koja čeka pritisak na gumb za povratak na prethodni zaslon.
     */
    @Override
    public void onBackPressed() {
        super.onBackPressed();
        finish();
        System.exit(0);
    }

    /**
     * Početna metoda koja se izvrši prilikom pokretanja zaslona.
     * Pronalazak i inicijalizacija elemenata sa zaslona.
     * Prilikom pritiska na gumbBTOnOff se uključuje ili isključuje Bluetooth
     * Prilikom pritiska na gumbDiscover se pronalaze i izlistaju svi mogući uređaji
     * na koji se aplikacija može povezati
     * @param savedInstanceState - parametar koji sadrži prošlo stanje zaslona kako se
     * nebi izgubili podaci prilikom povratka na isti
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        gumbBTOnOff = findViewById(R.id.gumbBTOnOff);
        gumbDiscover = findViewById(R.id.gumbDiscover);
        Bluetooth.listaDiscoveredDevices = findViewById(R.id.listaDiscDevices);
        Bluetooth.listaDiscoveredDevices.setOnItemClickListener(MainActivity.this);
        Bluetooth.myBluetoothDevices = new ArrayList<>();
        Bluetooth.createBluetoothAdapter();

        //Paljenje gašenje Bluetooth-a
        gumbBTOnOff.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

- 2) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **KontrolaActivity** datoteku, iz koje se prethodno tome sve obrisali.

```
package hr.foi.air1802.mranger;

import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;

public class KontrolaActivity extends AppCompatActivity {

    Button gumbDisconnect;
    Button gumbForward;
    Button gumbLeft;
    Button gumbRight;
    Button gumbBackwards;
    Button gumbSporo;
    Button gumbNormalno;
    Button gumbBrzo;

    //temperatura
    Button gumbTemperatura;
    Button gumbPohraniTemperaturu;

    StringBuilder messages;

    /**
     * Metoda koja čeka pritisak na gumb za povratak na prethodni zaslon.
     */
    @Override
    public void onBackPressed() {
        super.onBackPressed();
        Controls.Disconnect(KontrolaActivity.this);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_kontrola);
        Intent newint = getIntent();
        messages = new StringBuilder();
        Controls.address = newint.getStringExtra(Bluetooth.EXTRA_ADDRESS);
        new ConnectBT(getApplicationContext(), KontrolaActivity.this).execute();
    }

    //konekcija
    gumbDisconnect = findViewById(R.id.gumbDisconnect);

    //kretanje
    gumbForward = findViewById(R.id.gumbForward);
    gumbLeft = findViewById(R.id.gumbLeft);
    gumbRight = findViewById(R.id.gumbRight);
    gumbBackwards = findViewById(R.id.gumbBackwards);

    //brzine
    gumbSporo = findViewById(R.id.gumbSporo);
    gumbNormalno = findViewById(R.id.gumbNormalno);
    gumbBrzo = findViewById(R.id.gumbBrzo);

    //boje gumbova brzina
    gumbSporo.setBackgroundResource(android.R.drawable.btn_default);
    gumbNormalno.setBackgroundColor(Color.parseColor("#fed63c"));
    gumbBrzo.setBackgroundResource(android.R.drawable.btn_default);
}
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

6.3 Dodavanje koda u layout datoteke

Prethodno smo kreirali 3 layout datoteke : **activity_kontrola.xml**, **activity_main.xml** i **discover.xml**.

Sada ćemo u jednu po jednu od njih zalijepiti pripadajući kod.

- 1) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **activity_main.xml** datoteku, iz koje se prethodno tome sve obrisali.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Molimo približite se mBot uređaju !"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintHorizontal_bias="0.504"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/gumbBTONOff"
        android:layout_width="200dp"
        android:layout_height="63dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="8dp"
        android:text="Bluetooth: ON/OFF"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <Button
        android:id="@+id/gumbDiscover"
        android:layout_width="200dp"
        android:layout_height="63dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="8dp"
        android:text="Otkrij uređaje"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/gumbBTONOff" />

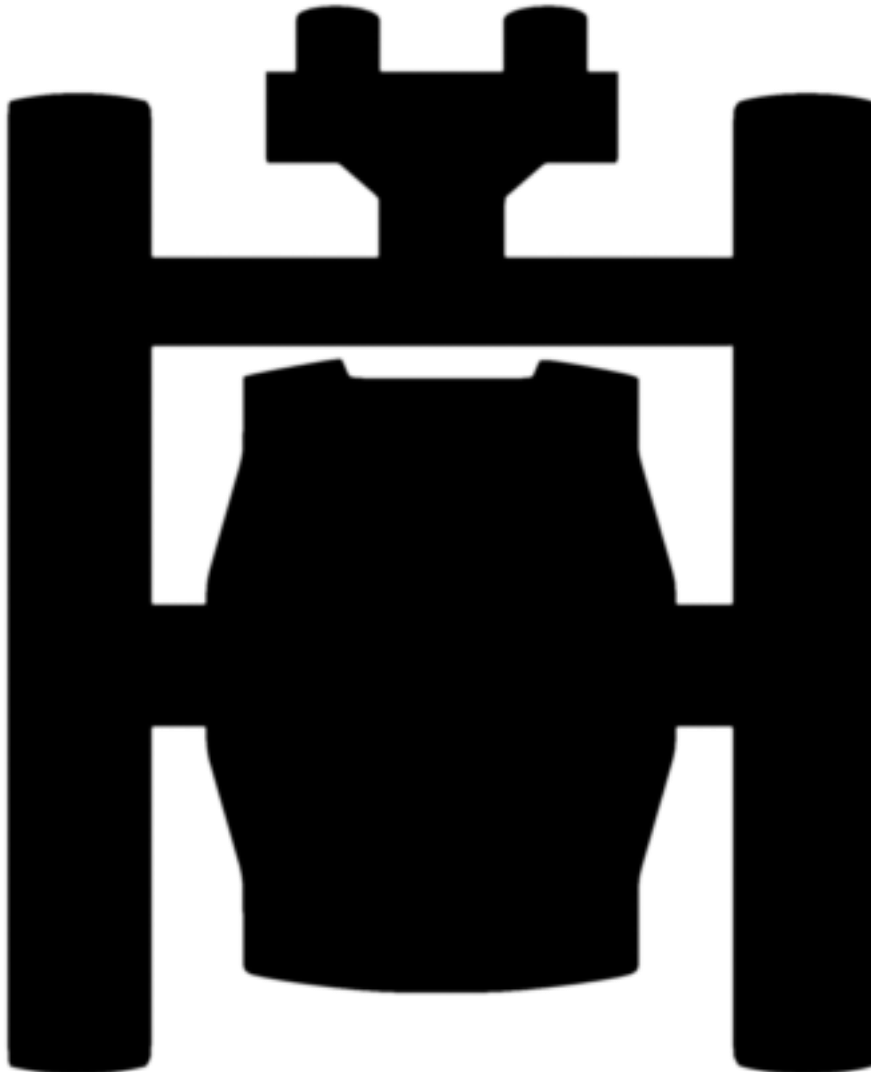
    <ListView
        android:id="@+id/listaDiscDevices"
        android:layout_width="325dp"
        android:layout_height="263dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.511"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/labelUredaji" />

    <TextView
        android:id="@+id/labelUredaji"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

Prije nego što zalijepimo kod za **activity_kontrole.xml**, na njoj se nalazi slika, te ćemo vam ispod priložiti sliku koju će te spremiti na vaše računalo, te zatim dodati u projekt kako je to opisano u poglavlju **4.2 Dodavanje slike**.

Desni klik na sliku te zatim **Spremi kao sliku...** te je nazovite **mranger_icon_v2** u već ponuđenom **PNG** formatu.



Slika 42. Slika za dodati

- 2) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **activity_kontorle.xml** datoteku, iz koje se prethodno tome sve obrisali.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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:id="@+id/textTemperatura"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".KontrolaActivity">

<ImageView
    android:id="@+id/slikaRobota"
    android:layout_width="103dp"
    android:layout_height="130dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:contentDescription="Ovo je slika našeg robota."
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.501"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/gumbDisconnect"
    app:srcCompat="@drawable/mranger_icon_v2" />

<Button
    android:id="@+id/gumbDisconnect"
    android:layout_width="89dp"
    android:layout_height="55dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="Odspoji se"
    app:layout_constraintEnd_toStartOf="@+id/gumbTemperatura"
    app:layout_constraintHorizontal_bias="0.056"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/gumbForward"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="8dp"
    android:text="NAPRIJED"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/slikaRobota" />

<Button
    android:id="@+id/gumbLeft"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="56dp"
    android:layout_marginTop="20dp"
    android:text="Lijevo"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/gumbForward" />

<Button
    android:id="@+id/gumbRight"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

- 3) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **discover.xml** datoteku, iz koje se prethodno tome sve obrisali.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/labelDeviceName"
        android:textSize="15sp"/>

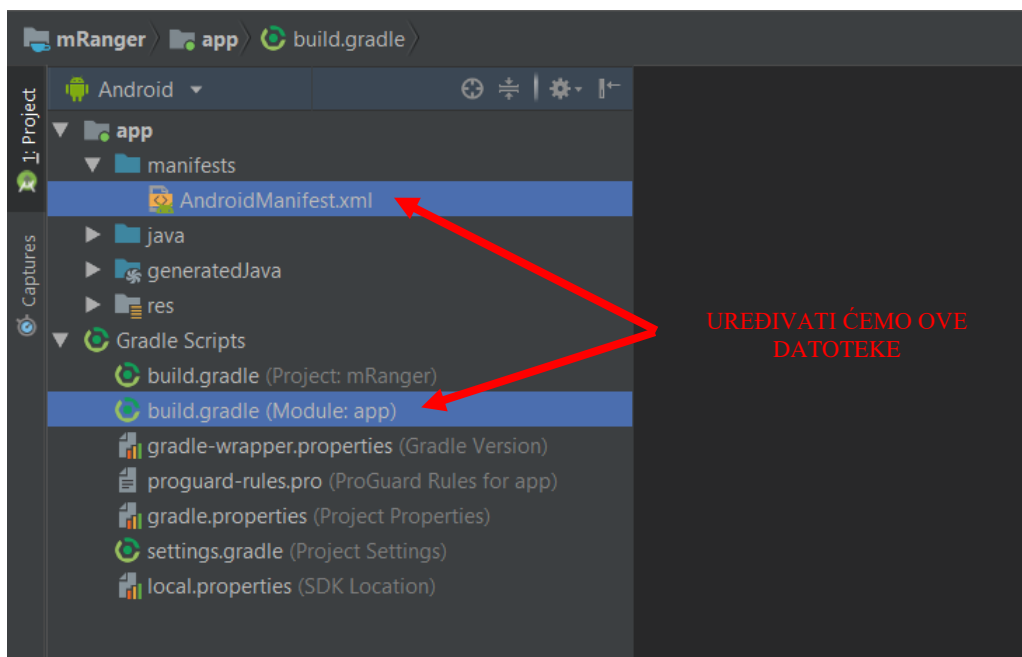
    <TextView
        android:id="@+id/labelDeviceAddress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="15sp"
        app:layout_constraintTop_toBottomOf="@+id/labelDeviceName"
        tools:layout_editor_absoluteX="0dp" />

</android.support.constraint.ConstraintLayout>
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

6.4 Još par sitnih izmjena

Dodali smo kod u skoro sve potrebne datoteke. Naime još samo moramo izmijeniti kod na dva mjesta.



Slika 43. Dodatne datoteke za uredit

- 1) Sljedeći kod treba u cijelosti kopirati i zalijepiti u **AndroidManifest.xml** datoteku, iz koje se prethodno tome sve obrisali.

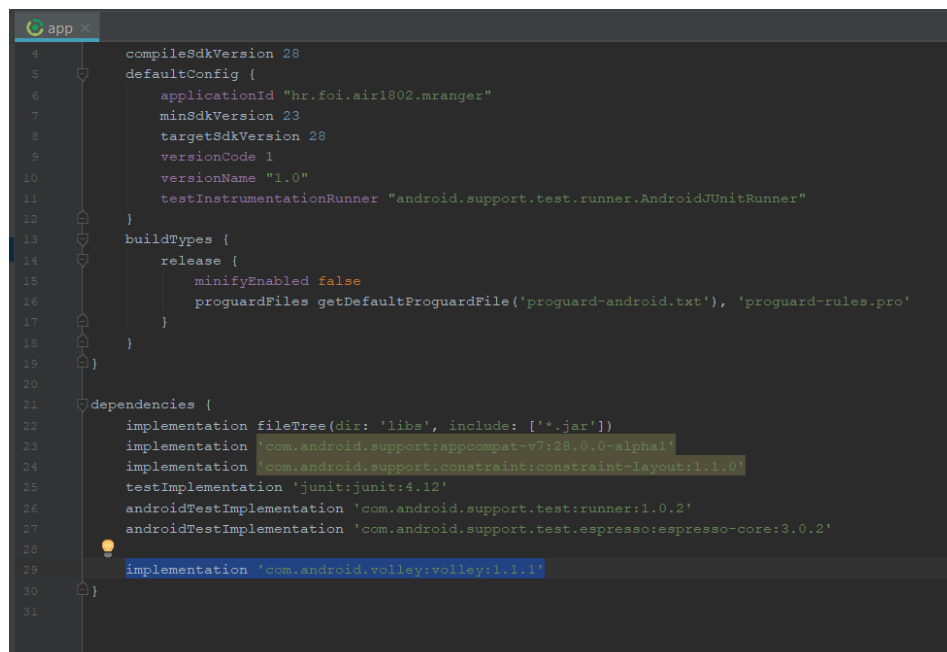
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="hr.foi.air1802.mranger">

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity" android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".KontrolaActivity"
            android:screenOrientation="portrait"></activity>
    </application>
</manifest>
```

- 2) Sljedeći kod treba **samo jednu liniju** kopirati i zalijepiti u **build.gradle (Module: app)** datoteku, iz koje se prethodno tome niste ništa obrisali !!!



```
4   compileSdkVersion 28
5   defaultConfig {
6       applicationId "hr.foi.air1802.mranger"
7       minSdkVersion 23
8       targetSdkVersion 28
9       versionCode 1
10      versionName "1.0"
11      testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
12  }
13  buildTypes {
14      release {
15          minifyEnabled false
16          proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
17      }
18  }
19  }
20  }
21  dependencies {
22      implementation fileTree(dir: 'libs', include: ['*.jar'])
23      implementation 'com.android.support:appcompat-v7:28.0.0-alpha1'
24      implementation 'com.android.support.constraint:constraint-layout:1.1.0'
25      testImplementation 'junit:junit:4.12'
26      androidTestImplementation 'com.android.support.test:runner:1.0.2'
27      androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
28      implementation 'com.android.volley:volley:1.1.1'
29  }
30  }
```

Slika 44. Dodatna linija

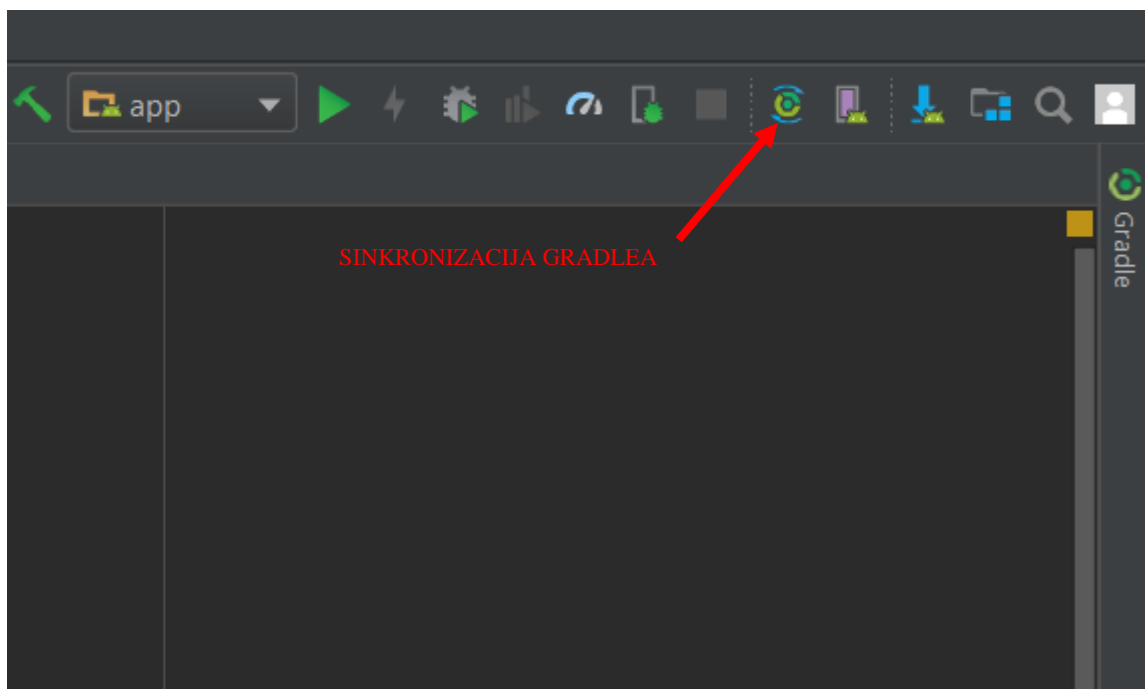
Linija koda koju treba kopirati je:

```
implementation 'com.android.volley:volley:1.1.1'
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

6.5 Sinkronizacija Gradlea

Naposljetku, potrebno je još samo sinkronizirati *Gradle* i naša aplikacija je spremna za pokretanje.



Slika 45. Sinkronizacija projekta

*AKO STE SVE ISPRAVNO NAPRAVILI, VAŠA APLIKACIJA JE SPREMNA ZA
POKRETANJE NA VAŠEM ANDROID UREĐAJU !
ČESTITAMO I HVALA VAM NA PAŽNJI ! ♡*

mRanger tim  

Kompletan projekt kreiran pomoću ove dokumentacije dostupan na:

<https://github.com/krizebcev/STEM-mRanger>

7. Logičko programiranje

Sljedeći dio ovog edukacijskog dokumenta baviti će jednim novim poglavljem, s kojim se zasigurno dosad niste imali prilike susresti. Riječ je o jednom logičkom programskom jeziku, koji je znatno drugačiji u logici rada od ostalih programskih jezika koji se koriste danas. Riječ je o Prologu, jeziku pomoću kojeg se može ostvariti nešto što će rezultirati, ili bar izgledati, kao umjetna inteligencija.

Znate ono, roboti sami donose odluke, zaključuju i slično? E pa slično tome, naša ideja je u Prolog jeziku obraditi podatke o temperaturi i vremenu, koji su izmjereni našom prethodno izrađenom aplikacijom – mRanger. Cilj je ostvariti da Prolog donese logičke zaključke, primjerice koje je godišnje doba na temelju dana i vremena te da li je temperatura viša ili niža od prosječne koja je karakteristična za to doba dana u godini.

U nastavku teksta ćemo pokazati osnove Prolog jezika, koji su njegovi sastavni elementi te kako se oni zajedno koriste.

Prolog

Prolog je programski jezik namijenjen za logičko programiranje, a pogotovo za razvoj umjetne inteligencije te od tuda njegov naziv, *PRO*graming in *LOGic*.

7.1 Uvod u Prolog

„Prolog jezik se temelji na predikatnoj logici prvog reda i koristi ograničenu verziju klauzula oblika poznatu kao Hornorov klauzalni oblik. „

Što to znači ? Pa ništa posebno vama vjerojatno, stoga ćemo i u nastavku teksta koristiti neke manje *stručne* izraze, pa ako ikada budete više učili o Prologu, oprostite nam na jednostavnosti.

Prolog bi tako bio koristan za opisivanje *objekata* i *relacija* među njima, recimo na primjer osoba je *objekt*, a ako je osoba roditelj nekoj drugoj osobi tada govorimo o *relaciji* između tih objekata.

7.1.1 Tipovi podataka

U tekstu iznad smo pričali o nekim *objektima*, oni u Prologu predstavljaju podatkovne objekte, te je pravilan naziv za njih *term*.

Tako *term* može biti konstanta ili varijabla ili složeni term.

Konstanta

- Cjelobrojne vrijednosti: 1, 0, 27000, -12, ...
- Decimalne vrijednosti 0.101, 3.14, -2.71, ...
- Atom: c, mirko, 'Tekst', [] (*lista*) odnosno sve bilo koji redoslijed alfanumeričkih (uključujući ' _ ') znakova pri ćemo prvo slovo mora biti malo.

Varijable

Varijable su bilo koji niz alfanumeričkih znakova (*čitaj brojeva i slova*) koji započinju velikim slovom ili `_`. U slučaju da varijabla započinje sa `_` tada nju nazivamo anonimnom tj. nebitnom varijablom. (*eng. don't care variable*)

Neki primjeri : **X**, **Y**, **Rezultat**, **_Rj**, **_234**, **_** i slično.

Ukoliko se neka varijabla koristi samo jednom u nekoj **rečenici** (*što je u Prolog jeziku naziv za neku funkciju*), odnosno ukoliko ona nije bitna za izvođenje programa ili upita tada se ona zapisuje u obliku `_` znaka.

Varijable u Prologu radi po istom principu kao i u ostalim jezicima, dajemo nekim imenovanim objektima vrijednost, te ih dalje koristimo pod tim imenom.

Složeni termovi

„Strukturirani podatkovni objekti su složeni termovi.“

Svaki složeni *term* sastoji se od funkcijskog simbola i sekvence od jednog ili više *term*-ova koji se nazivaju argumentima. Funkcijski simbol okarakteriziran je svojim imenom (koje je atom) i svojom arnošću (kratnošću, mjesnošću).

Što to znači ?

Primjerice imamo složeni *term* koji ima funkcijski simbol **roditelj** arnosti **2**, s argumentima **X, Y** bi se zapisao kao:

roditelj(X , Y).

Ovime bi se ostvarilo nešto poput funkcije s argumentima, te radi na principu **roditelj(majka, dijete)** ili **roditelj(otac, dijete)**, ili obrnut redoslijed argumenata, ali ukoliko počnemo na jedan način, njega se dalje moramo pridržavati.

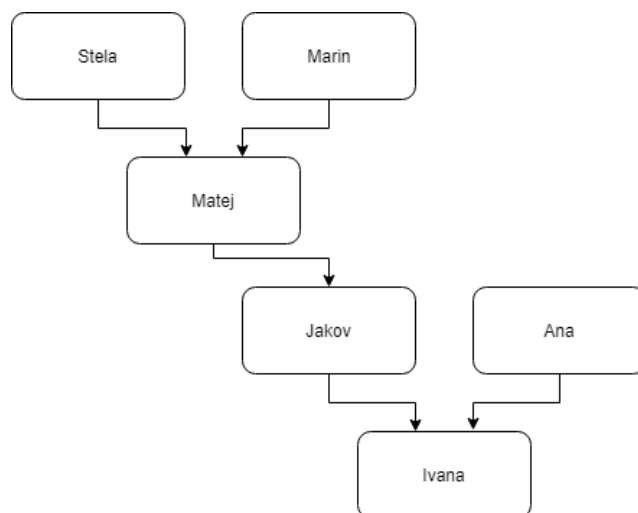
7.1.2 Činjenice i pravila

Dva koncepta koja trebamo još poznavati su *činjenice* i *pravila*. Činjenice opisuju elementarne relacije, a pravila određuju nove relacije na temelju već postojećih.

Primjer činjenice:

roditelj(stela, matej).

U primjeru iznad, **roditelj** nam je *relacija*, dok su nam **stela** i **matej** *term*-ovi nad kojima *relacija* **roditelj** djeluje. Time dobivamo činjenicu da je **stela** roditelj od **matej**. Napisani primjer možemo dodatno proširiti te dobijemo obiteljsko stablo. Primjer proširene relacije roditelj:



roditelj(stela, matej).

roditelj(marin, matej).

roditelj(matej, jakov).

roditelj(jakov, ivana).

roditelj(ana, ivana).

Slika 46. Obiteljsko stablo

Ovako napisane činjenice u Prologu nazivamo još bazom znanja, budući da sadrže neke konkretne informacije.

Ako želimo ispitati neku relaciju, odnosno da li je neka osoba nekome roditelj, onda iza znakova **?-** pišemo naš upit.

Npr.

?- roditelj(stela, matej).

yes

Upit vraća vrijednost **yes** ukoliko zbilja naš upit postoji kao činjenica, odnosno u nekim Prolog jezicima to će biti **true**.

U slučaju da dobijemo **no** (*false*) onda znamo da neka osoba nije roditelj drugoj osobi koju smo naveli u upitu, odnosno postavili smo upit za koji ne postoji činjenica.

Ako želimo saznati tko je roditelj od **matej** onda stavimo na mjesto roditelja, veliko slovo ili neku riječ koja počinje velikim slovom, tj postavimo *varijablu*.

Na taj način, reći ćemo Prologu da nam iz postojećih činjenica, dohvati sve roditelje koje **matej** možda ima.

Na primjer:

?- roditelj(X, matej).

X = stela

X= marin

Iz primjera možemo vidjeti da su **stela** i **marin** roditelji od **matej**. Iz navedenom primjera se dobro vidi moć zaključivanja Prologa, uz uvjet da su sve napisane činjenice ispravne.

Daljnji primjeri:

?- roditelj(stela, X), roditelj(X,Y). % Tko je Stelin unuk ?

?- roditelj(X, jakov), roditelj(X, matej). % Da li imaju istog roditelja ?

Na ovaj način smo zadali složeni upit Prologu, odvojen zarezom, te na taj način obje tvrdnje moraju biti zadovoljene kako bi se vratio neki rezultat.

7.1.3 Pravila

Pomoću pravila se u Prologu definiraju nove, proširene relacije, uz pretpostavku da imamo već prethodno zadane relacije. Tako možemo naš primjer *roditelj*-a proširiti ovako:

...

musko(marin).

musko(matej).

musko(jakov).

zensko(stela).

zensko(ivana).

zensko(ana).

majka(X,Y):-

roditelj(X,Y), zensko(X).

otac(X,Y):-

roditelj(X,Y), musko(X).

U slučaju iznad, naše nove dvije relacije su **majka** i **otac**. Tako npr. relaciju **majka** možemo ovako interpretirati.

X je majka Y ako je X roditelj Y i ako je X žensko

Pravilo za **djeda** bi ovako izgledalo:

X je djed Z-u ako je X musko i ako je X roditelj Y i ako je Y roditelj Z

tj.

djed(X, Z):-

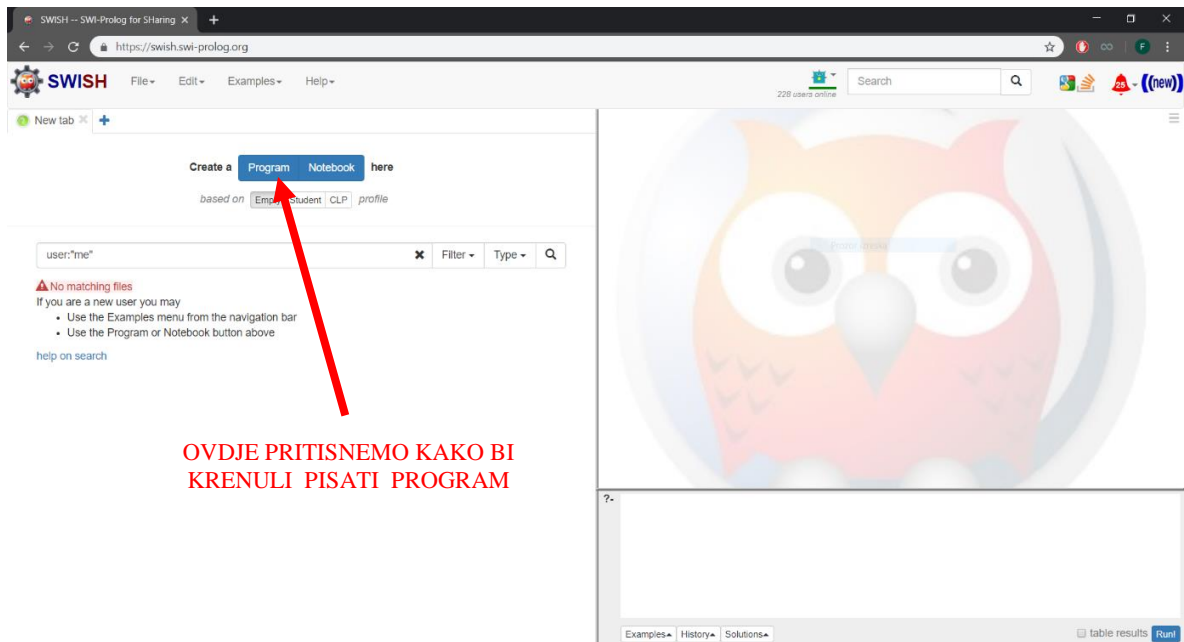
musko(X),

roditelj(X, Y),

roditelj(Y, Z).

7.2 Pisanje i izvršavanje Prolog programa

Za pisanje Prolog programa možete koristiti besplatni online alat, SWISH dostupan na <https://swish.swi-prolog.org/> lokaciji.

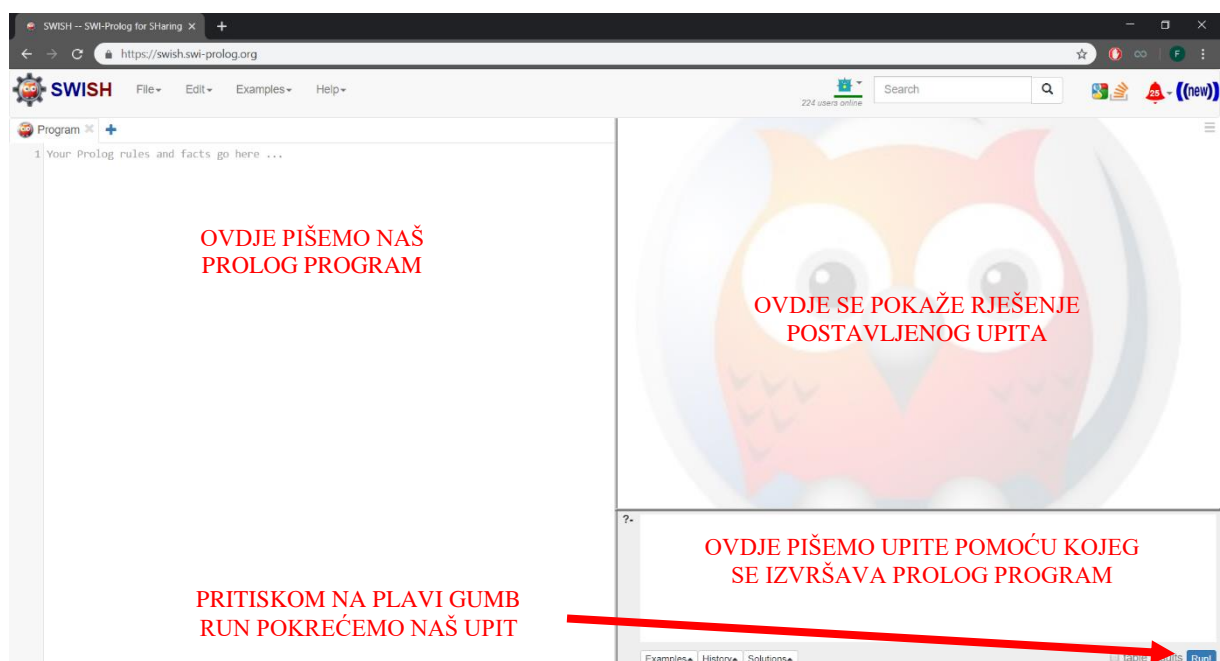


OVDJE PRITISNEMO KAKO BI
KRENULI PISATI PROGRAM

Slika 47. SWISH Prolog

Na stranici odaberemo opciju **Program** te nam se otvori sljedeći zaslón, gdje ćemo pisati naš Prolog program.

Kako bi napisani program mogli negdje drugdje koristiti, potrebno ga je preuzeti, a to obavimo na način da pritisnemo na *File* i onda na *Download*. Time smo preuzeli naš napisani Prolog program, i on bi trebao imati ekstenziju *.pl*.



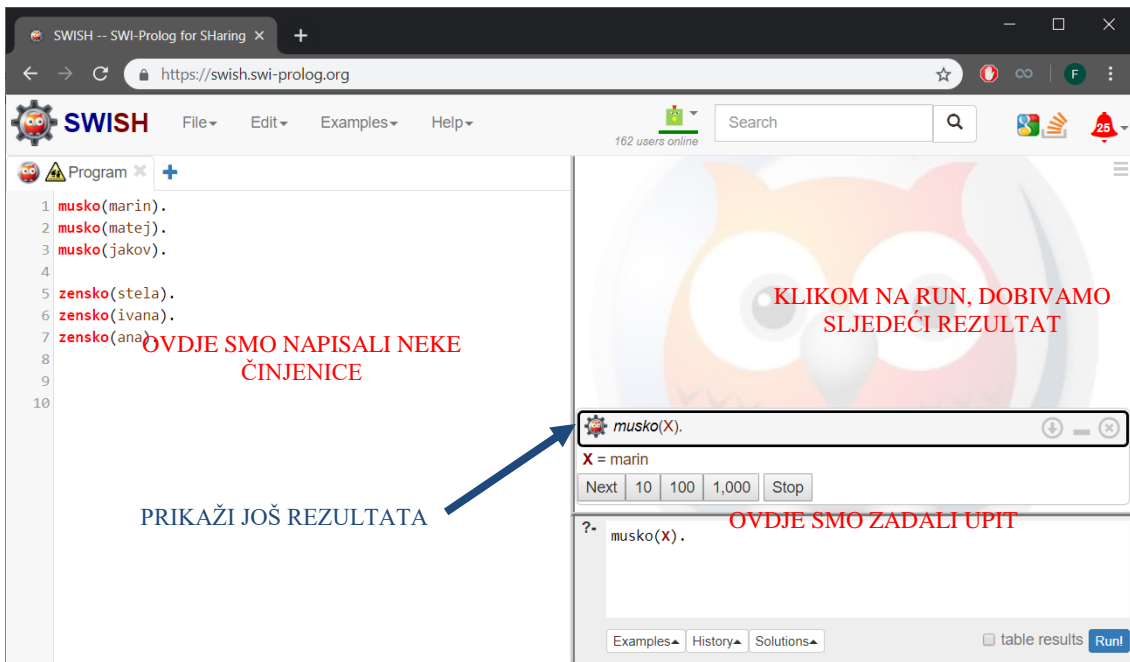
OVDJE PIŠEMO NAŠ
PROLOG PROGRAM

OVDJE SE POKAŽE RJEŠENJE
POSTAVLJENOG UPITA

PRITISKOM NA PLAVI GUMB
RUN POKREĆEMO NAŠ UPIT

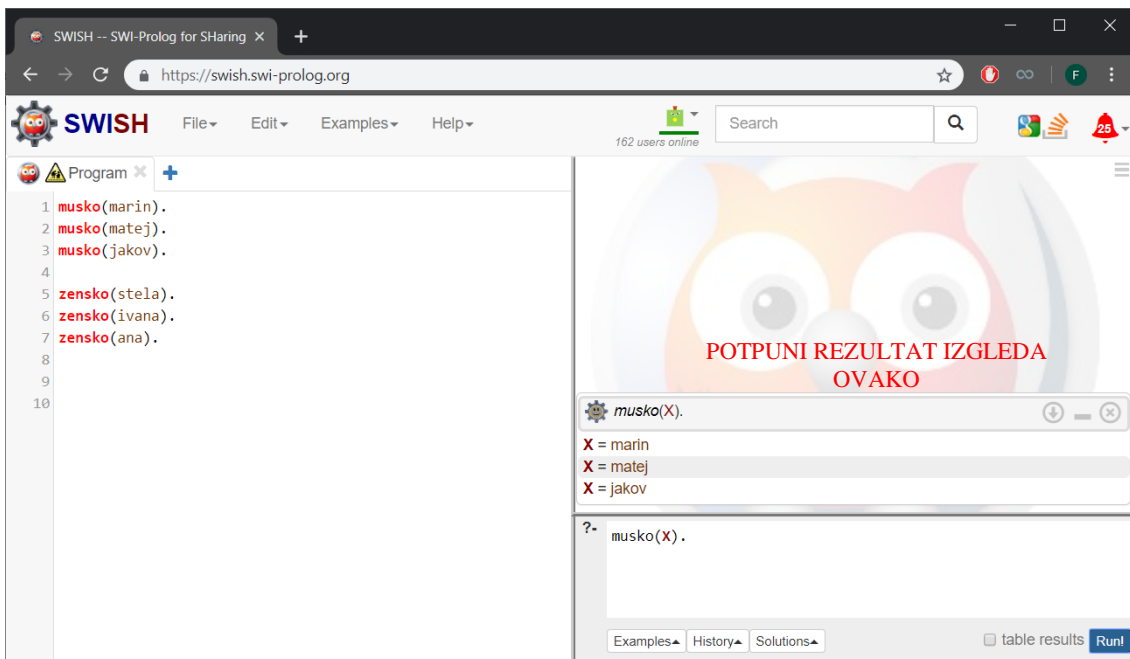
OVDJE PIŠEMO UPITE POMOĆU KOJEG
SE IZVRŠAVA PROLOG PROGRAM

Slika 48. SWISH Prolog – prozor pisanja programa



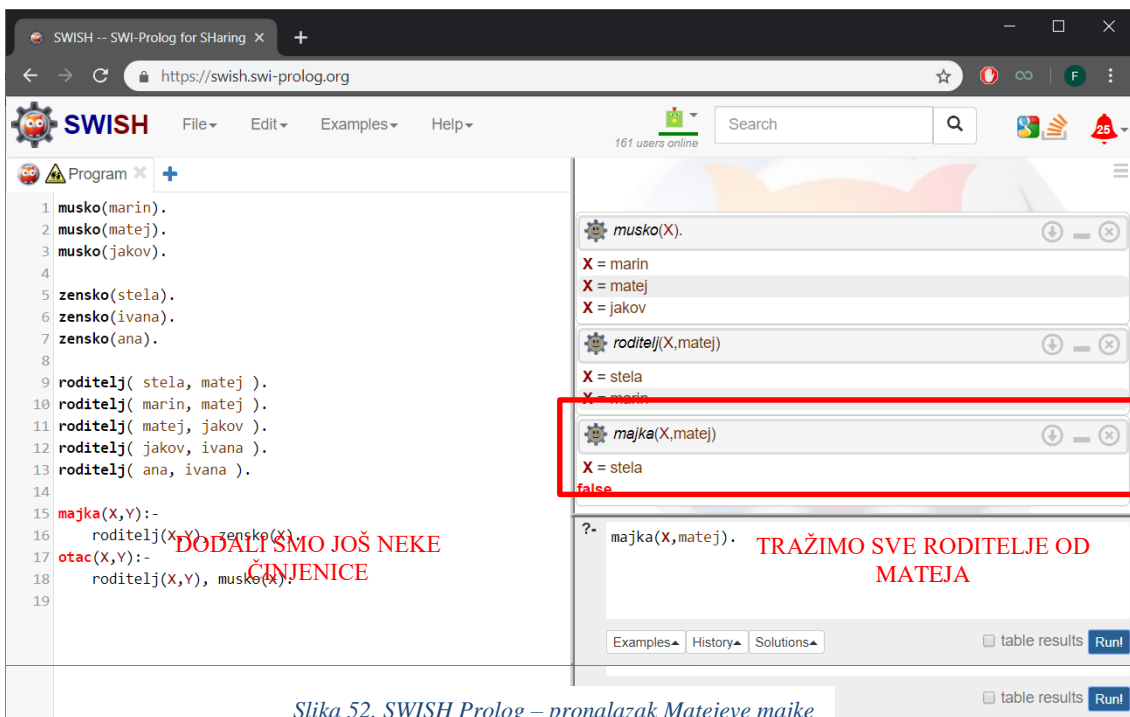
Slika 49. SWISH Prolog – pronalazak svih muških 1/2

Izvršavanjem upita, dobivamo nazad sve rezultate koji zadovoljavaju naš upit. Ukoliko ih ima



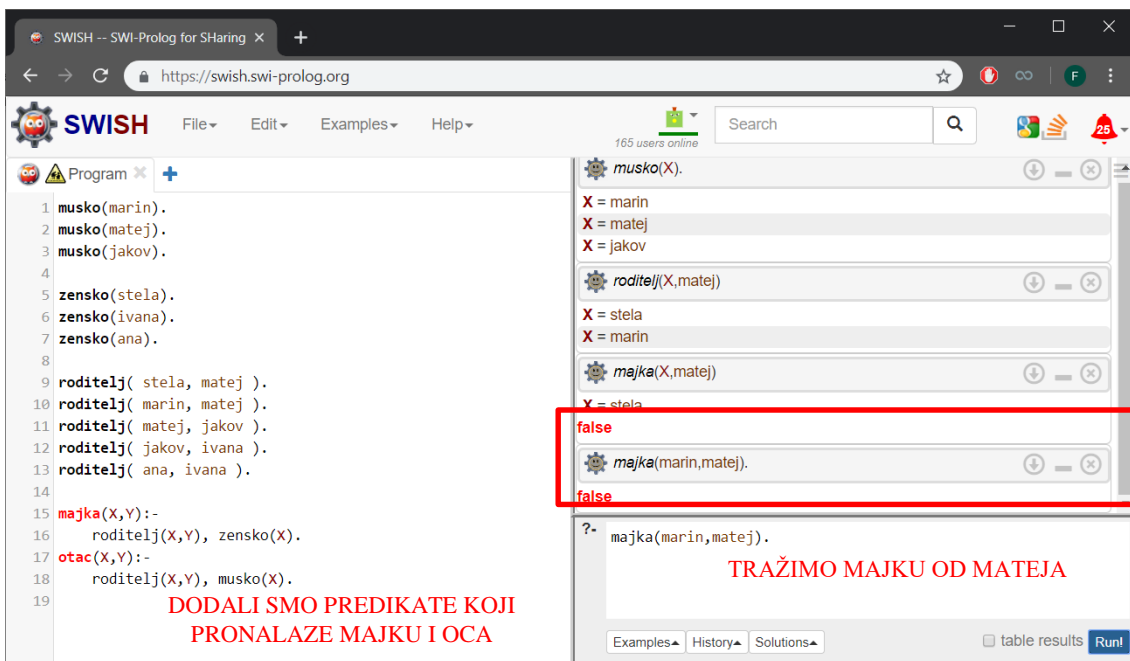
Slika 50. SWISH Prolog – pronalazak svih muških 2/2

više, neki kompajleri će odmah prikazati sva rješenja, dok ovdje će nam biti prikazano prvo te omogućeno da pomoću gumba *Next* dalje idemo kroz rezultate, odnosno prikazujemo sljedećih *10*, *100* ili *1000* sa pripadajućim gumbima.



Slika 52. SWISH Prolog – pronalazak Matejeve majke

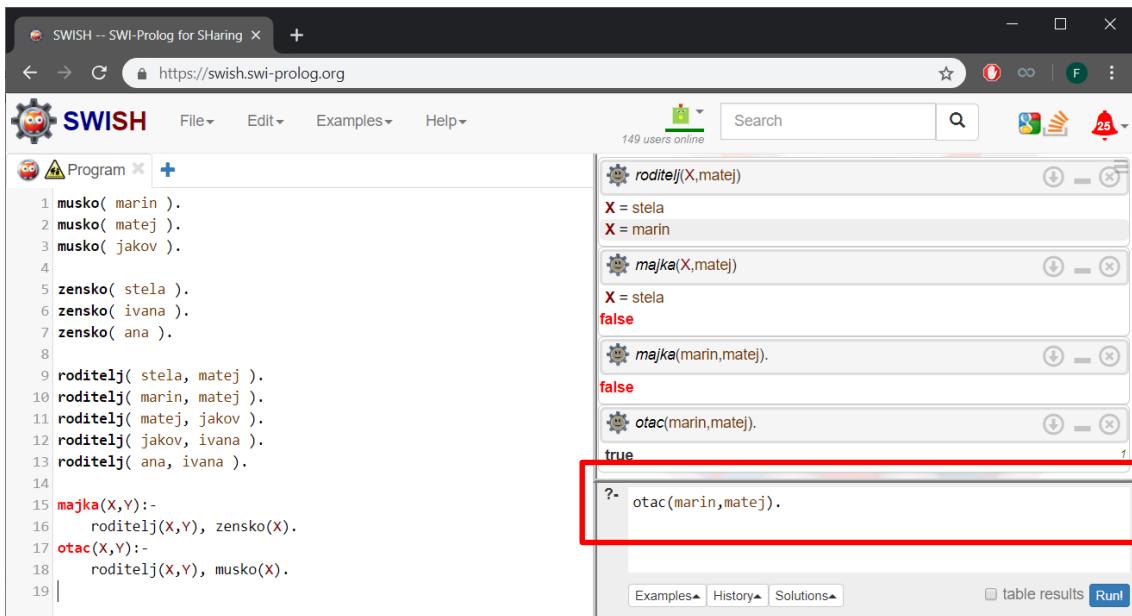
Slika 51. SWISH Prolog – pronalazak Matejevih roditelja



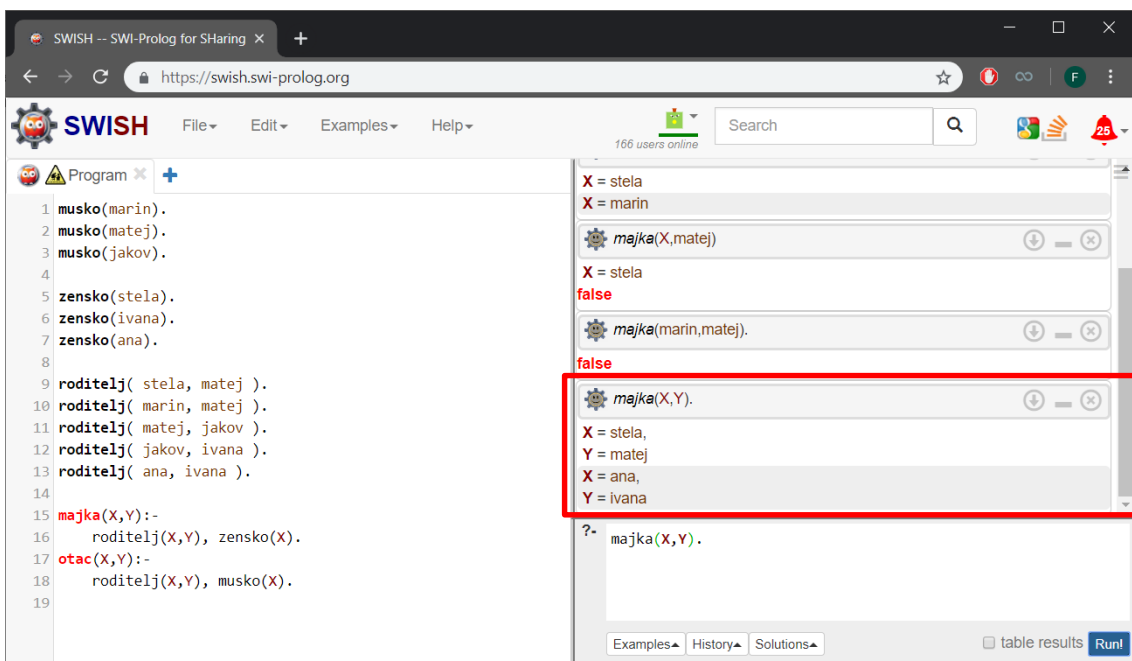
Slika 53. SWISH Prolog – provjera da li je Marin majka od Mateja

Upit *majka(marin, matej)*. vraća nazad *false*, što znači da za taj upit nije pronađeno niti jedno rješenje iz naših činjenica koje ga zadovoljava. U ovom konkretnom slučaju, kako

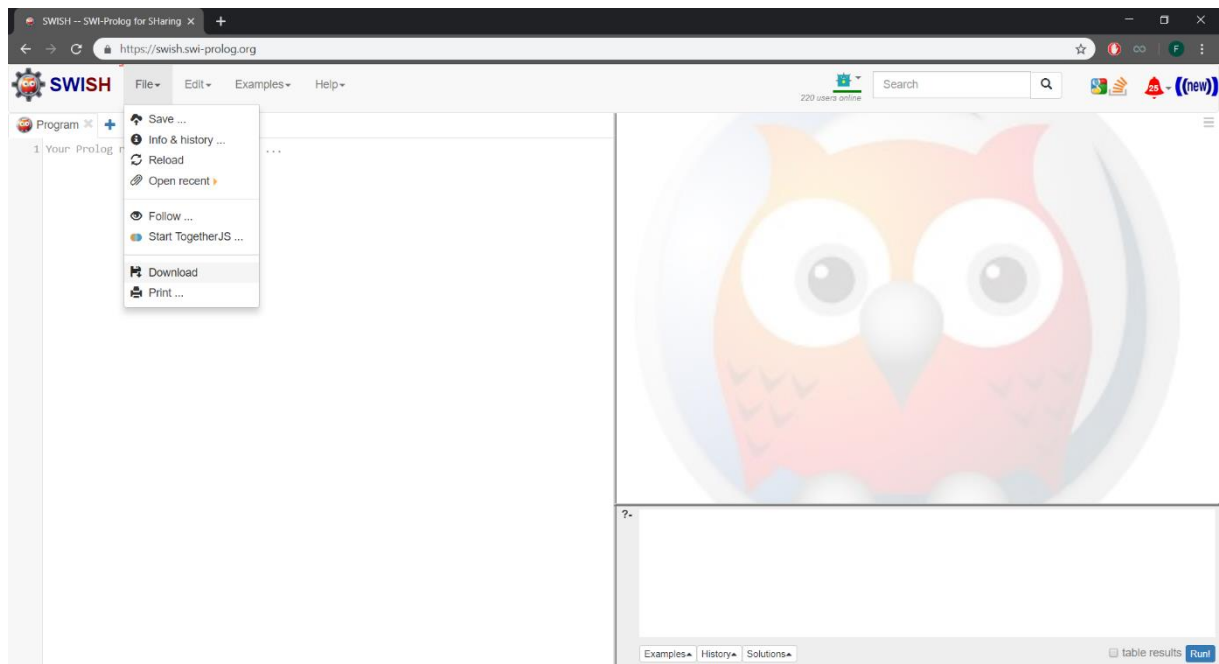
bi netko bio majka od Mateja, ta osoba mu mora biti **roditelj** te još mora biti i **žensko**. Marin je roditelj od Mateja, ali je muško.



Slika 54. SWISH Prolog – provjera da li je Marin otac od Mateja



Slika 55. SWISH Prolog – pronalaženje svih parova majki i djeteta



Slika 56. Spremanje Prolog programa

7.3 Provjera godišnjeg doba

Prije nego što pokažemo kako se određuje je li neka temperatura prosječna za to doba godine, pokazat ćemo kako se određuje godišnje doba. Prilikom spremanja temperature s robota u bazu, također smo spremili datum i vrijeme kad smo to radili. Na temelju datuma ćemo odrediti o kojem se doba godine radi (proljeće, ljeto, jesen ili zima).

Ispod priloženi Prolog kod obavlja taj zadatak, a uz komentare možete zaključiti kako on radi.

```
provjera_godisnjeg_doba( UlazniMjesec, UlazniDan ) :-
    godisnje_doba( UlazniMjesec, UlazniDan, Ispis ),
    write( Ispis ).

%Složeniji ali kraci nacin pronalaženja godišnjeg doba.
%Kod nizanja uvjeta u složenom predikatu ',' (zarez) predstavlja
%logičko 'i', dok ';' (točka zarez) predstavlja logičko 'ili'.

godisnje_doba( Mjesec, Dan, Ispis):-

% U prvom mjesecu je sigurno godišnje doba zima, tako da dan ne moramo provjeravat.
Mjesec=1 -> Ispis="Godisnje doba je zima!";
% U drugom mjesecu je sigurno godišnje doba zima, tako da dan ne moramo provjeravat.
Mjesec=2 -> Ispis="Godisnje doba je zima!";
% U trecem mjesecu ukoliko je dan veci ili jednak 21, pocinje proljece, tako da dan moramo provjeravat.
Mjesec=3, Dan =< 20 -> Ispis="Godisnje doba je zima!";

% U trecem mjesecu ukoliko je dan veci ili jednak 21, pocinje proljece, tako da dan moramo provjeravat.
Mjesec=3, Dan >= 21 -> Ispis = "Godisnje doba je proljece!";
Mjesec=4 -> Ispis="Godisnje doba je proljece!"; % Ista logika slijedi u nastavku za sve ostale mjesece.
Mjesec=5 -> Ispis="Godisnje doba je proljece!";
Mjesec=6, Dan =< 20 -> Ispis = "Godisnje doba je proljece!";

Mjesec=6, Dan >= 21 -> Ispis = "Godisnje doba je ljeto!";
Mjesec=7 -> Ispis="Godisnje doba je ljeto!";
Mjesec=8 -> Ispis="Godisnje doba je ljeto!";
Mjesec=9, Dan =< 22 -> Ispis = "Godisnje doba je ljeto!";

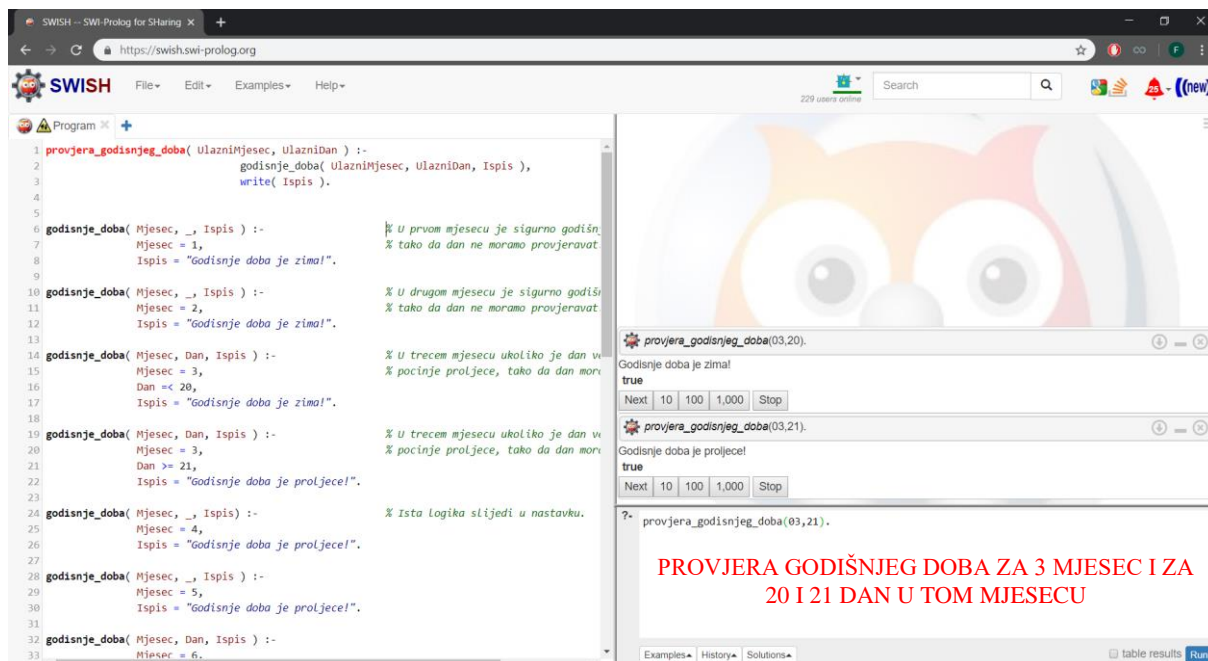
Mjesec=9, Dan >= 23 -> Ispis = "Godisnje doba je jesen!";
Mjesec=10 -> Ispis="Godisnje doba je jesen!";
Mjesec=11 -> Ispis="Godisnje doba je jesen!";
Mjesec=12, Dan =< 20 -> Ispis = "Godisnje doba je jesen!";

Mjesec=12, Dan >= 21 -> Ispis = "Godisnje doba je zima!".

/*
%Jednostavniji ali znatno duži nacin pronalaženja godišnjeg doba
godisnje_doba( Mjesec, _, Ispis ) :-          % U prvom mjesecu je sigurno godišnje doba zima,
    Mjesec = 1,                               % tako da dan ne moramo provjeravat.
    Ispis = "Godisnje doba je zima!".

godisnje_doba( Mjesec, _, Ispis ) :-          % U drugom mjesecu je sigurno godišnje doba zima,
    Mjesec = 2,                               % tako da dan ne moramo provjeravat.
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)



Slika 57. Primjer korištenja programa `provjera_godisnjeg_doba.pl`

7.4 Provjera prosječne temperature

Sad kada ste shvatili princip Prologa, pisanja programa i upita, onda možemo krenuti na zabavni dio, a to je provjera temperature koju smo dohvatili s robota. Za određivanje prosječne temperature koristimo tri podatka, a to su mjesec, dan i izmjerenu temperaturu te je cjelokupni kod za određivanje toga nešto kompleksniji od prethodnog.

U programu imamo tri doba dana (jutro, dan, noć) za koje možemo provjeriti je li u određenom mjesecu dohvaćena temperatura ispod, iznad prosjeka ili je li ona prosječna. Svako od tih doba dana za neki mjesec ima svoju pripadajuću činjenicu u kodu, koju smo mi izračunali za podatke o prosječnim temperaturama u Varaždinu za 2017. godinu.

Mjesec	Noć	Jutro	Dan	Mjesec	Noć	Jutro	Dan
1.	-8.5	-4.8	-1.1	8.	14.7	21.4	28.0
2.	-0.4	4.0	8.4	9.	9.8	14.2	18.5
3.	2.6	9.3	15.9	10.	4.8	11.7	18.5
4.	4.6	10.4	16.2	11.	1.2	5.6	10.0
5.	9.5	15.8	22.0	12.	-2.6	2.4	7.3
6.	14.4	20.7	26.9				
7.	15.1	21.8	28.4				

Graf 1. Prosječne temperature po mjesecima

U samom Prolog programu imate točne opise i komentare koja naredba što obavlja.

```
% Sljedeći dio koji morao napraviti je za unesene podatke
% o mjesecu i vremenu u kojem je očitana temperatura,
% odrediti da li je ta temperatura ispod ili iznad prosjeka
% za to doba dana.
```

```
provjera_prosjecne_temperature( UlazniMjesec, UlazniSat, UlaznaTemperatura ) :-
    provjera_doba_dana( UlazniSat, IzlaznoDoba),
    provjera_temperature( UlazniMjesec, IzlaznoDoba, UlaznaTemperatura, Ispis ),
    write(Ispis).
```

```
% U ovom slučaju odredili smo da jutro traje od 6 do 12 h,
% dan od 12 do 20 h, te noc od 20h do 6h ujutro.
% Sukladno tome, napravljene su sljedeće provjere koje
% će vratiti doba dana.
```

```
% Složeniji ali kraci način rješavanja problema doba_dana
```

```
provjera_doba_dana( Sati, Doba ):-
    Sati >= 20 -> Doba = noc;
    Sati < 6 -> Doba = noc;
    Sati >= 6, Sati < 12 -> Doba = jutro;
    Sati >= 12, Sati < 20 -> Doba = dan.
```

```
/*
% Jednostavniji i duži način rješavanja problema doba_dana
```

```
provjera_doba_dana( Sati, Doba ) :-
    Sati >= 20,
    Doba = noc.
```

```
provjera_doba_dana( Sati, Doba ) :-
    Sati < 6,
    Doba = noc.
```

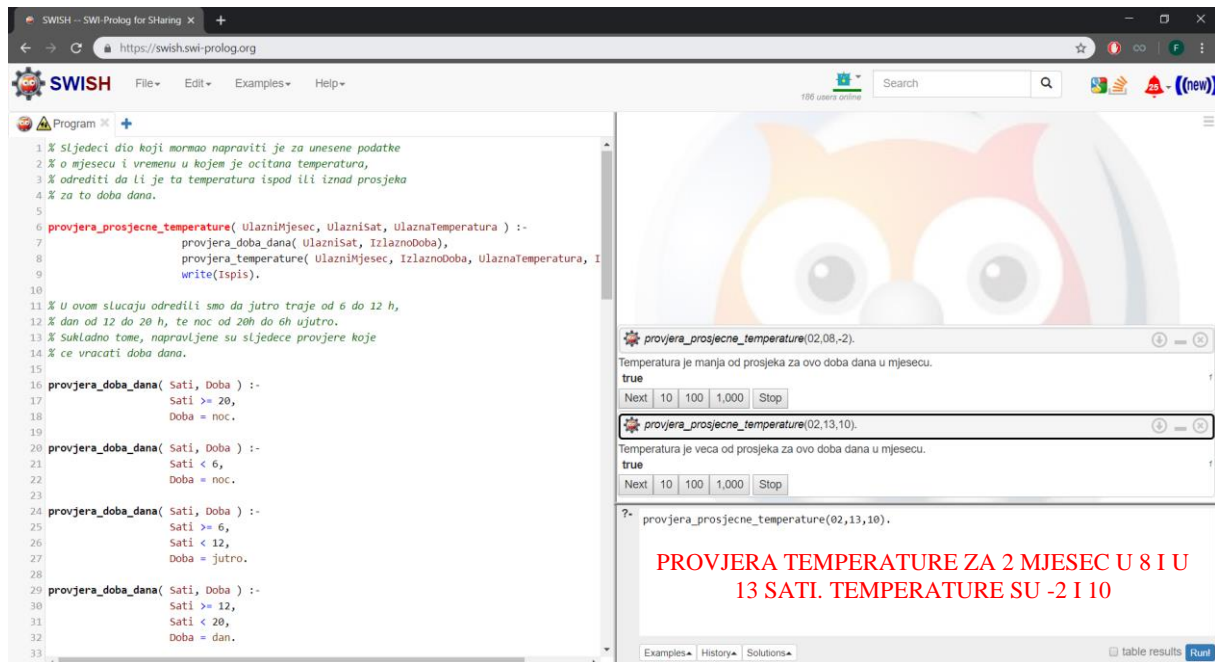
```
provjera_doba_dana( Sati, Doba ) :-
    Sati >= 6,
    Sati < 12,
    Doba = jutro.
```

```
provjera_doba_dana( Sati, Doba ) :-
    Sati >= 12,
    Sati < 20,
    Doba = dan.
```

```
*/
```

```
% Mi smo izračunali neke činjenice, u kojima samo za
% određeni mjesec i njegovo doba dana odredili
% prosječne temperature. Te činjenice su naša baza znanja,
% te ćemo ulaznu temperaturu uspoređivati s njima.
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)



Slika 58. Primjer korištenja programa `provjera_prosjecne_temperature.pl`

7.5 Pozivanje Prolog programa iz PHP-a

Ovime smo napisali dva Prolog programa koja zaključuju o kojem se godišnjem doba radi te je li dohvaćena temperatura prosječna ili ne. Sad je jedino potrebno povezati i primijeniti taj napisani Prolog kod. Mi smo se odlučili za PHP koji se koristi za obradu podataka na raznim Internetskim stranicama.

Ovdje vam nećemo objašnjavati kako i na koji način radi PHP jezik već samo kako se pozivaju naši Prolog programi.

Za Windows operacijske sustave:

```
$naredba_god_doba="C:\Users\krist\Downloads\swipl\bin\swipl.exe -f provjera_godisnjeg_doba.pl "
    ."-g provjera_godisnjeg_doba($podatakMjesec,$podatakDan)";
$naredba_temp="C:\Users\krist\Downloads\swipl\bin\swipl.exe -f provjera_prosjecne_temperature.pl "
    ."-g provjera_prosjecne_temperature($podatakMjesec,$podatakSat,$podatakTemperatura)";

$rez = shell_exec($naredba_god_doba)." ".shell_exec($naredba_temp);
echo $rez;
```

Za Linux operacijske sustave:

```
$naredba_god_doba_linux = "swipl -q -f provjera_godisnjeg_doba.pl -t "
    . "\"provjera_godisnjeg_doba($podatakMjesec,$podatakDan)\\"";
$naredba_temp_linux="swipl -q -f provjera_prosjecne_temperature.pl -t "
    . "\"provjera_prosjecne_temperature($podatakMjesec,$podatakSat,$podatakTemperatura)\\"";

$rez_linux = shell_exec($naredba_god_doba_linux)." ".shell_exec($naredba_temp_linux);
echo $rez_linux;
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

Naredbu pozivanja i izvršavanja Prolog programa dijelimo na tri djela, prvi dio pomoću kojeg definiramo putanju do .exe datoteke, odnosno samog programa koji izvršava naše naredbe (kompajler), drugi dio koji počinje s -f iza kojeg definiramo Prolog datoteku u kojoj se nalazi naš kod, a zadnji, treći dio koji počinje s -g iza kojeg pozivamo neku od naših „metoda“ iz našeg Prolog programa.

U zadnjem djelu, unutar zagrada postavljamo naše vrijednosti ili varijable koje sadrže vrijednosti koje želimo obraditi. Ovdje je bitno da nemamo nikakvih razmaka u zagradi jer se inače neće dobro pozivat Prolog program.

Sad je još jedino potrebno napisanu naredbu pozvati. Postoji nekoliko načina, a najjednostavnija je *shell_exec()*, u koju postavimo našu napisanu naredbu. Ona pozove kompajler, Prolog program koji smo zadali, te „metodu“ s postavljenim podacima, a vrati ono što *write()* naredba ispiše u programu.

Kako bi na stranici prikazali dohvaćenu poruku, u PHP-u poruku ispisujemo s naredbom *echo*.

7.6 Usporedba PHP i Prolog rješenja

Programi koji su napisani u Prolog jeziku također se mogu napisati u PHP programskom jeziku, no tada se postavlja pitanje gdje ima više smisla napisati takav program. U prethodnom dijelu dokumenta smo pokazali kako izgleda kod napisan u Prolog jeziku, a sada ćemo vam priložiti iste te programske odsječke napisane u PHP-u na temelju kojih ćemo ustanoviti gdje ima smisla koristiti Prolog jezik, a gdje ne.

7.6.1 Provjera godišnjeg doba u PHP-u

```
<?php

function provjera_godisnjeg_doba($UlazniMjesec,$UlazniDan){

    $Mjesec=$UlazniMjesec;
    $Dan=$UlazniDan;

    $Ispis;

    if ($Mjesec==1)
    {
        $Ispis="Godisnje doba je zima!"
    }
    else if ($Mjesec==2)
    {
        $Ispis="Godisnje doba je zima!"
    }
    else if ($Mjesec==3 && $Dan<=20)
    {
        $Ispis="Godisnje doba je zima!"
    }
    else if ($Mjesec==3 && $Dan>=21)
    {
        $Ispis="Godisnje doba je proljece!"
    }
    else if ($Mjesec==4)
    {
        $Ispis="Godisnje doba je proljece!"
    }
    else if ($Mjesec==5)
    {
        $Ispis="Godisnje doba je proljece!"
    }
    else if ($Mjesec==6 && $Dan<=20)
    {
        $Ispis="Godisnje doba je proljece!"
    }
    else if ($Mjesec==6 && $Dan>=21)
    {
        $Ispis="Godisnje doba je ljeto!"
    }
    else if ($Mjesec==7)
    {
        $Ispis="Godisnje doba je ljeto!"
    }
}
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

Provjera godišnjeg doba u PHP jeziku je vrlo jednostavno izvedeno te ima 75 linija programskog koda, dok kod Prolog programa imamo svega 20 linija programskog koda.

U ovakvim slučajevima, kada imamo veliki broj „*if-else if*“ provjera, Prolog program znatno smanjuje potreban broj linija programskog koda. Osim toga, programski kod napisan na taj način je više čitljiv i bolje strukturiran, a time i razumljiviji za druge.

7.6.1 Provjera prosječne temperature u PHP-u

```
<?php

function provjera_prosjecne_temperature($UlazniMjesec,$UlazniSat,$UlaznaTemperatura)
{
    $Mjesec=$UlazniMjesec;
    $Sati=$UlazniSat;
    $Temperatura=$UlaznaTemperatura

    $Doba;

    $Ispis;

    if ($Sati>=20)
        $Doba="noc";
    else if ($Sati<6)
        $Doba="noc";
    else if ($Sati>=6 && $Sati<12)
        $Doba="jutro";
    else if ($Sati>=12 && $Sati<20)
        $Doba="dan";

    if ($Mjesec==1)
    {
        if ($Doba=="jutro")
        {
            $ProsjecnaTemperatura=-4.8;
            $Ispis=provjera_temperature($Temperatura,$ProsjecnaTemperatura);
        }
        else if ($Doba=="dan")
        {
            $ProsjecnaTemperatura=-1.1;
            $Ispis=provjera_temperature($Temperatura,$ProsjecnaTemperatura);
        }
        else if ($Doba=="noc")
        {
            $ProsjecnaTemperatura=-8.5;
            $Ispis=provjera_temperature($Temperatura,$ProsjecnaTemperatura);
        }
    }

    else if ($Mjesec==2)
    {
        if ($Doba=="jutro")
        {
            $ProsjecnaTemperatura=4;
            $Ispis=provjera_temperature($Temperatura,$ProsjecnaTemperatura);
        }
    }
}
```

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

U takvom primjeru, kada imamo veliki broj „*if-else if*“ provjera, Prolog program znatno smanji broj napisanih linija, tj. s nekakvih 275 linija na tek 70 linija. Osim toga tako napisan program je znatno čitljiviji te je bolje strukturiran, a time i razumljiviji.

Tako velika prednost postignuta je na napisanim činjenicama za temperature po dobu dana u određenom mjesecu, odnosno našoj bazi znanja. U PHP-u je potrebno za takvu provjeru napisati nekoliko „ *if* “ provjera, dok u Prologu napišemo činjenicu u jednom redu pa program sam zaključi što treba učiniti. U prijevodu, kada imamo veliki broj provjera, tada se s jednostavnim činjenicama kod može znatno smanjiti te nam na neki način zamijeni „ *if* “ naredbu.

To je samo jedan dobar primjer gdje je Prolog koristan, ali ima znatno veći broj takvih primjera, potrebno je samo malo vremena i truda uložiti kako bi se takvo nešto postiglo.

7.7 Dodatni Prolog materijali

Dosad napisani sadržaj je dovoljan za izradu osnovnih Prolog programa, ali za nešto više potrebna su dodatna znanja i vještine, a ovime ćemo vam pružati nekakav dodatni sadržaj čime si možete proširiti već postojeće znanje iz Prologa. Osim toga, priloženi sadržav vam možda dodatno pojašni nekakve nedoumice koje ste dosad imali.

7.7.1 Liste

Liste kao i u svakom programskom jeziku su jedni od glavnih podatkovno strukturnih elemenata te je bitno da se one savladaju. U Prologu su liste atomi označene s „ [] “ simbolima, a mogu biti i složeni term-ovi s funkcijskim simbolom „ . “. U većini slučajeva se koriste atomi s oznaka „ [] “.

Npr.

[1 , 2 , 3]

Kod lista je bitno znati da se njen sadržaj dijeli na Glavu (G) i na Rep (R). Glava je uvijek prvi element iz liste, a Rep je cijeli onaj ostatak bez prvog elementa. Na primjeru iznad, Glava je 1, a Rep [2 , 3]. Iz Repa [2 , 3] Glava je 2, a Rep [3]. Kao zadnje, iz liste [3] Glava je 3, a Rep [].

7.7.2 Rekurzije

U programskom svijetu **rekurzije** su programi koji sami sebe pozivaju, i to neki određeni broj puta jer mora postojati neki **granični uvjet** gdje će se zaustaviti. Međutim u Prologu nemamo takve postojeće konstrukcije koja to omogućuje.

Kako bi takvo nešto postigli, koristimo rekurzivne relacije, a da bi vam to bilo jasnije koristit ćemo primjer *roditelj*.

```
% predak je roditelj...
predak( X, Y ) :-
    roditelj( X, Y ).
% ... ali i roditelj od bilo kojeg drugog pretka
predak( X, Z ) :-
    roditelj( X, Y ),
    predak( Y, Z ).
```

Relacija *predak* je opisana sa dva stavka što znači da vrijedi ako su zadovoljeni svi od njih. Više stavaka koji definiraju jednu relaciju čine *proceduru* te je redoslijed stavki u proceduri bitan.

Budući da znamo kako liste funkcioniraju, mogli bi napraviti ispis pojedinih elemenata iz liste, a za takvo nešto potrebna nam je rekurzija. Primjer možete sami isprobati.

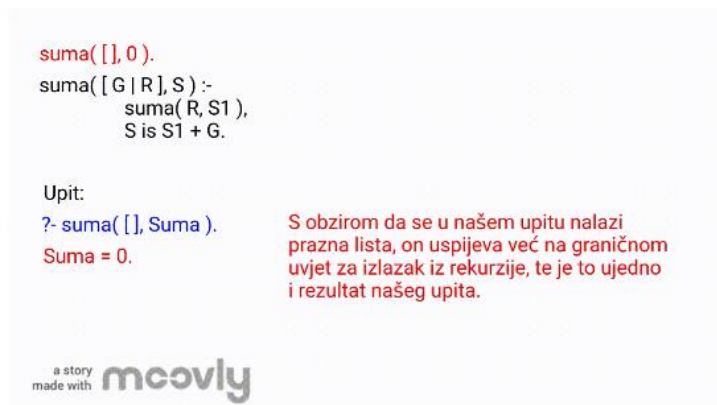
```
elementi( [ G | R ] ):-           %cjelokupna lista gdje se izdvaja prvi element od ostatka
    writeln( G ),                 %ispis glave (writeln - idući ispis ide u novi red)
    elementi( R ).                %ponovno pozivanje predikata elementi s listom bez glave tako
                                   dugo dok ima nešto za ispisat
```

Postoje slučajevi u kojima se rekurzija neće niti početi izvršavati te nam je stoga potreban uvjet pomoću kojeg ispisujemo rješenje. U ostalim slučajevima potrebno je izvršavanje rekurzije zaustaviti u točno određenom trenutku gdje je također potreban izlazni uvjet.

Kako bi vam to bilo jasnije ispod smo vam priložili kratke animacije gdje je vizualno prikazano korištenje takvog uvjeta. U animacijama se međusobno zbrajaju brojevi iz liste.

Granični uvjet je istoimeni predikat čiji argumenti čine logičku granicu izvođenja. Odnosno ukoliko tijekom izvođenja programa dođe do zadovoljenja tog predikata, to će omogućiti programu izlazak iz rekurzije.

Primjer s praznom listom []:



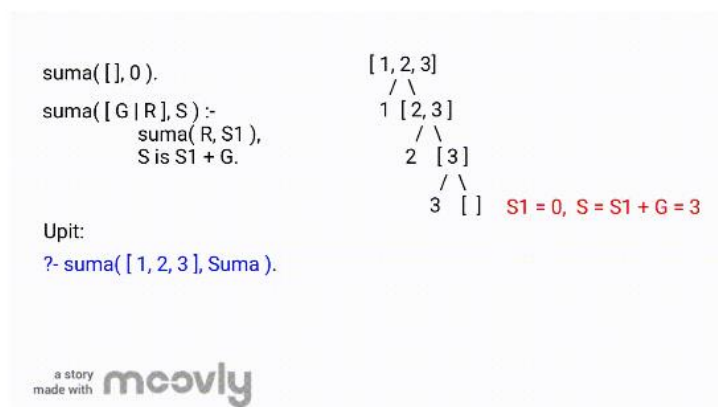
Slika 59. Veza na animaciju s primjerom prazne liste

Primjer s listom [1]:



Slika 60. Veza na animaciju s primjerom liste s jednim elementom

Primjer s listom [1 , 2 , 3]:



Slika 61. Veza na animaciju s primjerom liste s tri elementa

(**Ctrl** + **klik** na sliku kako biste otvorili animacije u pregledniku)

U prethodnim primjerima imali smo predikat koji je služio zbrajanju elemenata liste. Granični uvjet u svakom od tih primjera je bio **suma([], 0)**. Taj predikat je značio da ukoliko tijekom izvođenja programa, dođemo do toga da imamo praznu listu, suma te liste će biti **0 (nula)**.

```
suma([], 0).  
suma([G|R], Suma) :-  
    suma(R, S1),  
    Suma is G + S1.
```

Primjerice ako imamo listu sa jednim elementom, **[1]**, granični uvjet nije zadovoljen te ulazimo u sljedeći predikat, te dijelimo tu listu na glavu od jednom elementa – **1** i preostalog repa, koji će biti **prazna lista []**.

Daljnijim izvođenjem koda izvršava se ponovno taj predikat gdje je lista zapravo preostali rep, koji je u ovom slučaju prazna lista, te se time zadovoljava **granični uvjet** i program zna da je kraj pozivanja rekurzija, te se vraća istim putem nazad. Time će **S1 postati 0**, te na samom kraju **Suma** će biti **G (1) + S1 (0) = 1**.

Ovo gradivo treba malo vježbe i strpljenja da sjedne, te ukoliko Vam još uvijek nije jasno, najbolje što možete je pregledati ponovno drugu animaciju (Slika 53.). Istom logikom kao i kod Slika 54. se program izvršava za bilo koji broj elemenata liste.

7.7.3 Posebni ugrađeni predikati

```
avenger(ironman, muski).
avenger(hulk, muski).
avenger(blackWidow, zenska).
avenger(captainMarvel, zenska).
avenger(spiderman, muski).
avenger(spiderman, muski).
```

/* DEFINICIJE

1) findall(Varijabla, Cilj, Lista).

-> Pronadi sve vrijednosti Varijabla koje zadovoljavaju Cilj i skupi ih u listi Lista.
Ako nema niti jedne takve vrijednosti Lista ce se vezati u praznu listu [].

2) bagof(Varijabla, Cilj, Lista).

-> Isto kao i findall/3 osim sto ako ne postoji niti jedna vrijednost koja bi zadovoljila cilj, tada cijeli cilj ne uspijeva.

3) setof(Varijabla, Cilj, Lista).

-> Isto kao i bagof/3 osim sto u konacnoj listi nema ponavljanja elemenata te su oni sortirani.

*** PRIMJERI

% Ovdje pronalazimo sve Avengere koji su muški.

?- findall(X, avenger(X, muski), L).

L = [ironman, hulk, spiderman, spiderman].

% Ovdje nas ne zanima kojeg su spola, već prolazni sve.

?- findall(X, avenger(X, _), L).

L = [ironman, hulk, blackWidow, captainMarvel, spiderman, spiderman].

% Ovdje pronalazimo sve Avengere koji su životinje, u ovom

% slučaju ih nema te je rezultat prazna lista.

?- findall(X, avenger(X, zivotinja), L).

L = [].

?- bagof(X, avenger(X, zivotinja), L).

false.

?- bagof(X, avenger(X, muski), L).

L = [ironman, hulk, spiderman, spiderman].

?- setof(X, avenger(X, muski), L).

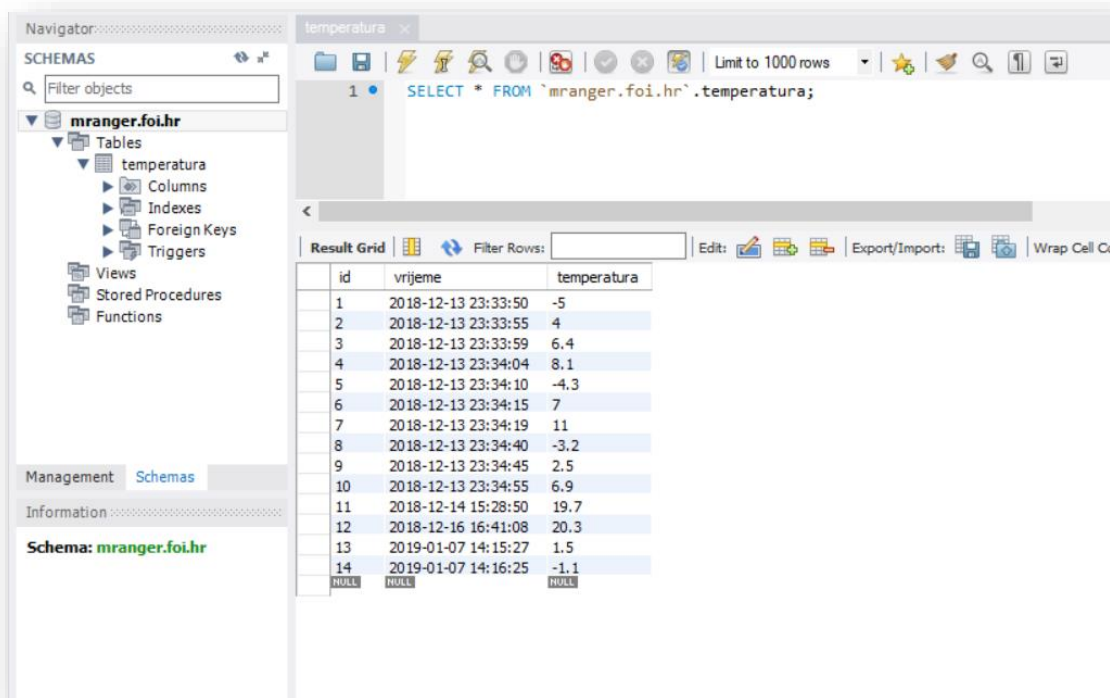
L = [hulk, ironman, spiderman].

*/

(Potrebno je kliknuti dva puta na okvir koda te će se cjelokupni kod prikazati za kopiranje.)

7.8 Prikaz baze podataka i web stranice

Sljedeća slika prikazuje tablicu *temperatura* koja se nalazi u našoj **mranger.foi.hr** bazi podataka. U toj tablici se bilježe sve temperature koje smo spremili u bazu putem naše **mRanger** android mobilne aplikacije. Svaki zapis ima svoj redni broj, vrijeme spremanja tog zapisa u bazu (sastoji se od datuma i lokalnog vremena) te sami iznos izmjerene temperature u stupnjevima Celzijusa.



The screenshot shows a database management interface with a 'Navigator' on the left and a 'Result Grid' on the right. The 'Navigator' shows the schema 'mranger.foi.hr' with a table 'temperatura'. The 'Result Grid' displays the following data:

id	vrijeme	temperatura
1	2018-12-13 23:33:50	-5
2	2018-12-13 23:33:55	4
3	2018-12-13 23:33:59	6.4
4	2018-12-13 23:34:04	8.1
5	2018-12-13 23:34:10	-4.3
6	2018-12-13 23:34:15	7
7	2018-12-13 23:34:19	11
8	2018-12-13 23:34:40	-3.2
9	2018-12-13 23:34:45	2.5
10	2018-12-13 23:34:55	6.9
11	2018-12-14 15:28:50	19.7
12	2018-12-16 16:41:08	20.3
13	2019-01-07 14:15:27	1.5
14	2019-01-07 14:16:25	-1.1
NULL	NULL	NULL

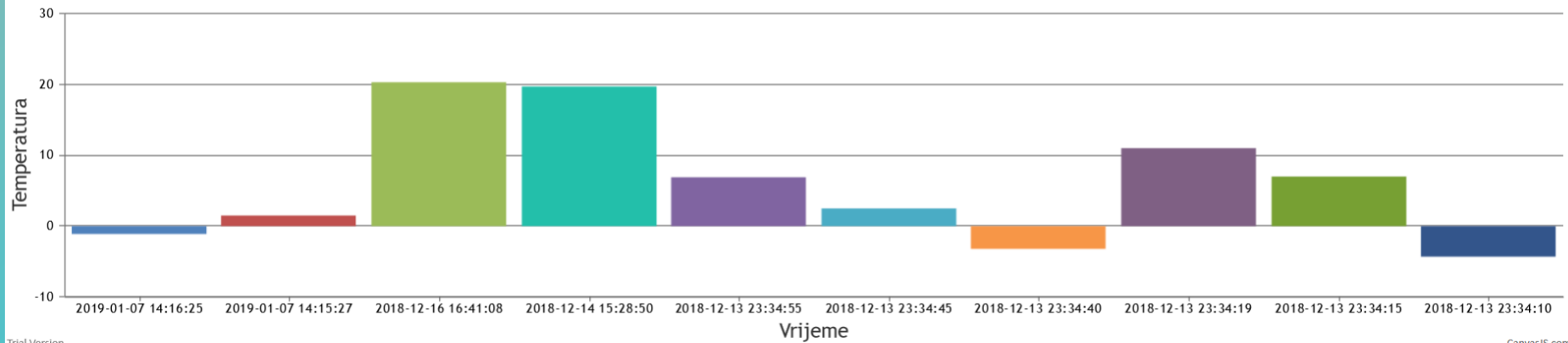
Slika 62. Prikaz tablice *temperatura* u bazi podataka

mRanger temperatura

Posljedna očitana temperatura na robotiču je bila 2019-01-07 14:16:25 i iznosi -1.1 °C.

Godisnje doba je zima! Temperatura je prosjecna za ovo doba dana u mjesecu.

Statistika zadnjih deset temperatura



Slika 63. Prikaz *mranger.foi.hr* stranice

Na slici poviše prikazana je naša web stranica koja se nalazi na domeni **mranger.foi.hr**.

Stranica prikazuje posljednjih 10 zapisa iz baze podataka, te na temelju njih izrađuje se graf temperatura. Također, posljednji podatak iz tablice se dohvaća i obrađuje pomoću Prologa na serveru, potom zaključujući **o kojem godišnjem dobu se radi** te da li je **temperatura veća, manja ili prosječna** za to doba dana u tom mjesecu.

PLEASE LIKE, SHARE, SUBSCRIBE  