Razvoj mobilnih aplikacija za Android i iOs platformu pomoću C# programskog jezika

Fiorencis, Marta

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marta Fiorencis

Razvoj mobilnih aplikacija za Android I iOS pomoću C# programskog jezika

ZAVRŠNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marta Fiorencis

Matični broj: 42010/13–R

Studij: Informacijski sustavi

Razvoj mobilnih aplikacija za Android i iOS pomoću c# programskog jezika

ZAVRŠNI RAD

Mentor/Mentorica:

Doc. dr. sc. Zlatko Stapić

Varaždin, rujan 2018.

Marta Fiorencis

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U radu će biti predstavljeno razvojno okruženje Xamarin, proći ću kroz postupak pripreme razvojnog okruženja poput instalacije, zatim ću obraditi koncept dijeljenja koda kroz Android, iOS i Windows te nakon toga ću predstaviti postupak kreiranja jednostavne fitness Android aplikacije. Tijekom postupka kreiranja aplikacije bit će demonstrirana izrada osnovnih funkcionalnosti za koju ću koristiti c# programski jezik. U praktičnom dijelu rada neću koristiti Xamarin studio već njegovu inačicu za Visual Studio. Cilj rada je predstavljanje izrade aplikacija u C# programskom jeziku, prezentiranje novog radnog okruženja i izražavanje subjektivnog mišljenja iz perspective korisnika.

Ključne riječi: Xamarin, dijeljenje koda, C# programski jezik, Android aplikacija

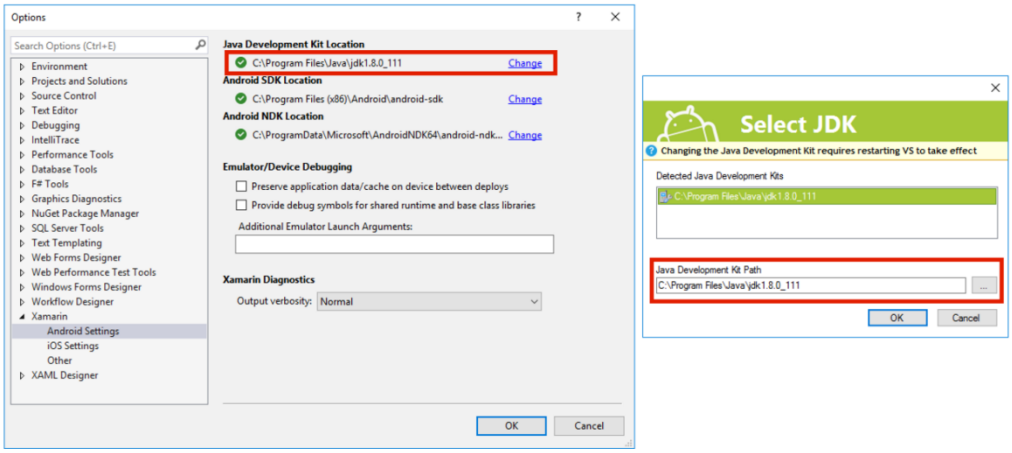
Sadržaj

1. Uvod

Što je ustvari Xamarin? Xamarin je alat koji omogućava jednostavan razvoj višeplatformskih mobilnih aplikacija za Android, IOS i Windows platformama koristeći .NET [2]. Ključna riječ u opisu je jednostavan, na Xamarin sam nabasala potpuno slučajno dok sam istraživala temu za drugi kolegij i učinio mi se vrijedan istraživanja. Xamarin je također ime tvrtke koja je osnovana 2011. i nalazi se u San Franciscu, Kaliforniji. Osnovali su ju Miguel de Icaza i Nat Friedman te je trenutno u Microsoftovom vlasništvu [1]. Proizvod koji je tvrtki donio uspjeh jest Xamarin studio koji su izdali 2013. godine, što je bio njihov prijašnji IDE Monodevelop koji su preimenovali. 2014. godine se pojavljuju Xamarin.Forms koji uvelike olakšavaju razvoj aplikacija, kao vrsta template-a. 24. Veljače Microsoft je u iznosu od 500 milijuna dolara odlučio kupiti Xamarin. 2016. godine Microsoft je objavio da će se uz Xamarin studio izdati se njegova još popularnija inačica koja se može integrirati u Visual Studio, koju ćemo u ovom radu prezentirati [1]. Nakon povijesti samog alata gdje je objašnjeno gdje i zašto je Xamarin nastao u radu bit će objašnjen postupak instalacije. Također ćemo se dotaknuti teme višeplatformskih aplikacija te kako i zašto ga koristiti. Da bi predstavila Xamarin u što stvarnijem i detaljnijem svjetlu odlučila sam ga prezentirati u svojem završnom radu iz strane korisnika (svoje) koji izrađuje jednostavnu android aplikaciju. Proći ćemo kroz postupak kreiranja nove aplikacije, pošto će ovo biti android aplikacija objasnit ću kako je pokrenuti na virtualnom te poslije i na fizičkom uređaju, odnosno što treba poduzeti kako bi aplikaciju mogli pravilno testirati. Kod same izrade bit će objašnjene osnovne funkcionalnosti poput dodavanja i izmjena sučelja, kreiranje baze podataka, unos, ispis i osvježavanje podataka, izrada klasa, funkcija i dretvi, definiranje događaja i korištenje iznimka.

2. Instalacija Xamarina u Visual Studio

U ovom poglavlju ću predstaviti postupak postavljanja radnog okruženja iliti instalaciju potrebnih alata za rad. Ako ćete dobiti volju da krenete testirati Xamarin, prvo što morate napraviti jest instalirati ga na računalo. Postoji razlog zašto sam krenula od ovog koraka, ako pročitate uputstva s njihove stranice ili slijedite video na YouTube-u kanalu, učinit će vam se da je jednostavno, skinite program i sve se instalira, to vrlo vjerojatno neće tako završiti. Općenita stvar kod Xamarina kao alata je da ima užasno puno grešaka u kodu i nekada radi nekada ne radi, jako je jednostavno u njemu programirati i snaći se te napraviti nešto, pogotovo ako ste već prije radili u Visual Studio, ali problem je što svako malo nešto ne radi, na sreću većina ih se može riješiti. Ako ste sretnici možda će vam instalacija stvarno i raditi ovako kako je napisano, ako vas tehnologija ne voli valja napomenuti par bitnih stvari, ako imate Windows 7 instalirajte novije Windowse, manje problema poslije, ako već niste skinite najnoviji Java Development Kit, iz nekog razloga najviše problema uvijek ima s tim dijelom instalacije. Kad se sve instaliralo, ako se pojavi greška u Visual Studio-u kada ga otvorite, prvo provjerite da li je dobar put da jdk, odnosno sdk-a(Software development kit, jdk je podskup sdk-a koji je skup softvera i programske podrške zaduženih za pisanje I pokretanje programa pisanih u Javi)[3]. Problem je što Visual Studio ima zadani nepostojeći put, ovo je greška ako je jdk instaliran u drugu datoteku na računalu.



Slika 1: jdk(Java development kit) instalacija

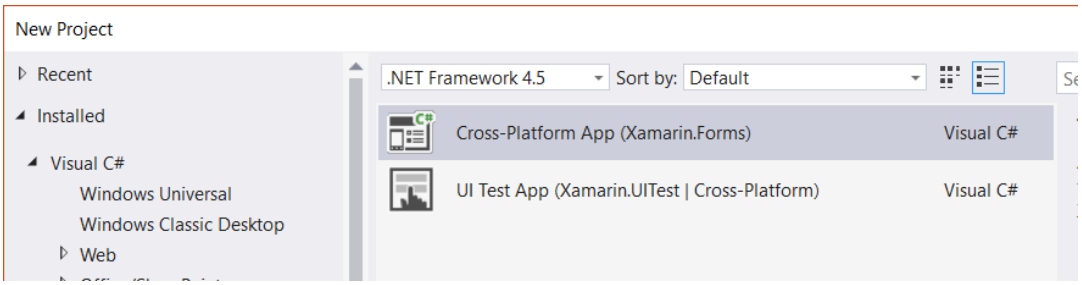
Ako ispravite put ili ste imali ispravni put a greška vam se još uvijek javlja, sljedeći korak je ažuriranje Android sdk, da ne bi previše duljili s objašnjenjima Android software development kit najlakše je zamisliti kao kutiju s alatima, različitim predlošcima, emulatorima, kodovima, itd.[3] Kod instalacije Xamarin-a učestalo se dogodi da svi alati nisu do kraja instalirani ili da neki alati nisu ažurirani. Ono što je bitno napomenuti, sve instalacije alata te njihova ažuriranje mora se raditi kao administrator[3].

Još jedna vrlo bitna stvar, morate zatvoriti Visual Studio i obrisati bin i obj (bin sadrži binarne datoteke za pokretanje aplikacije ili neke od library-ja, a obj sadrži kompajlirane binarne datoteke koji još nisu povezani) datoteke projekta kako se ne bi pomiješala stara i nova verzija obrasca, to se često događa ako imate neke zahtjevnije dijelove projekta koji su složenije nadograđivani u novijim verzijama.[2] Kada smo već kod brisanja bin i obj datoteka, po mome iskustvu u 70% slučajeva ovo rješava problem ako se aplikacija sruši na pokretanja. Nakon što smo kreirali projekt i nema nikakvih grešaka, moramo odabrati gdje ćemo pokretati aplikaciju. Ako imate malo bolje računalo, preporučujem emulator, možete ih i više odjednom pokrenuti[3]. Ja sam odabrala kompromisno rješenje prvotno sam aplikaciju pokrenula pomoću emulatora, a kasnije na uređaju[3].

3. Koncept dijeljenja koda između platforma

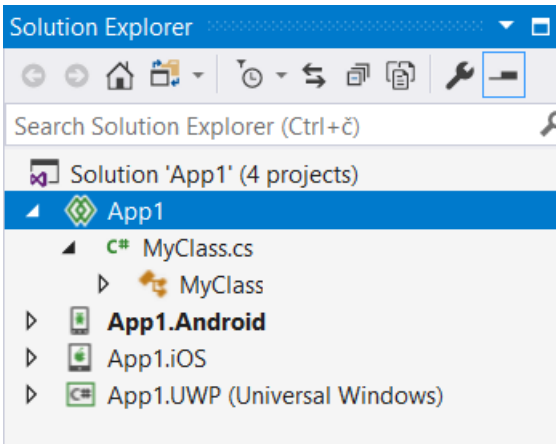
U poglavlju će se obrađivati ideja o izrađivanju aplikacije za više platformi u jednom projektu, također ćemo obraditi koncept dijeljenja koda između platforma u istom. Ideja izrade višeplatformske(Android, iOS i Windows) aplikacije u jednom projektu jest maksimalno recikliranje koda kako bi u što kraćem vremenu izradili kvalitetnu aplikaciju. Dijeljenje koda funkcionira tako da postoji jezgreni kod u kojem koristimo DOT.net biblioteke (System.Data, System.Linq, System.Xml, System.Net,..) i koji se dijeli kroz sve platforme ali također u svakoj platformi možemo koristiti aplikacijska sučelja specifična za nju(Microsoft.Phone, Windows.Storage, MapKit, UIKit, ActionBar, Renderscript, itd.) [4].

Otvorite Visual studio, odaberite izradu novog projekta i odaberite “Cross-Platform App” [2].



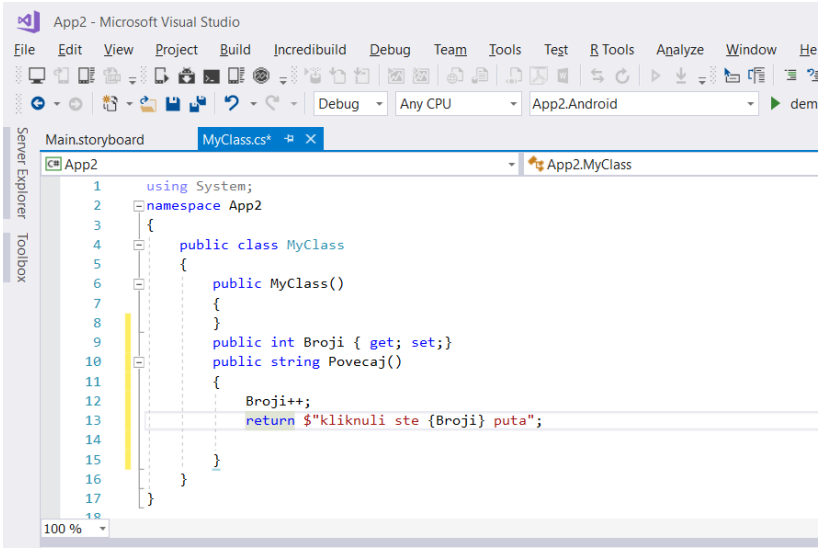
Slika 2: Kreiranje višeplatformskog projekta

Kada se kreira novi project, obratite pozornost na “Solution explorer”.



Slika 3: Struktura projekta

Projekt je strukturiran u 3 zasebna dijela, Android, iOS i UWP, te zajedničku klasu. Svaki dio sastoji se od zasebne glavne aktivnosti, resursa, reference i resursa [2]. Kako bi skratili vrijeme izrade koristimo klasu MyClass.cs.



Programski kod 1: Kreiranje metode u glavnoj klasi

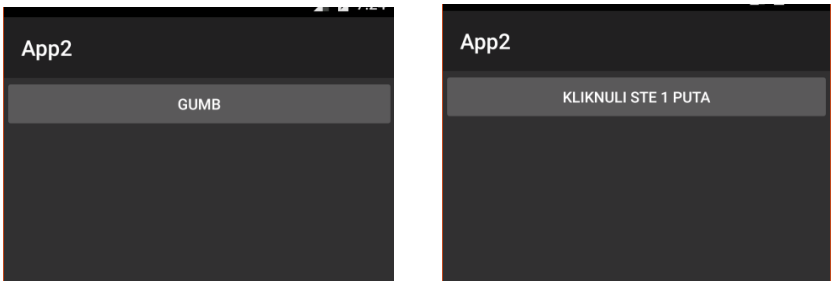
Naprimjer želimo izraditi aplikaciju koja će na svim platformama zbrajati broj klikova na gumb, što znači da će svaka platforma obavljati istu funkciju, zato ćemo tu metodu definirati u glavnoj klasi [2].

Otvaramo glavno sučelje Android dio projekta i kreiramo gumb tipa “MyButton” i definiramo događaj “Click” u kojem pozivamo funkciju Povecaj() iz glavne klase.

```
Main.axml | MainActivity.cs* | Main.storyboard | MyClass.cs
App2.Android | App2.Droid.MainActivity | OnCreate(Bundle savedInstanceState)
{
    [Activity(Label = "App2", MainLauncher = true, Icon = "@mipmap/icon")]
    public class MainActivity : Activity
    {
        MyClass dijeljen_kod = new MyClass();
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

            // Set our view from the "main" layout resource
            SetContentView(Resource.Layout.Main);

            // Get our button from the layout resource,
            // and attach an event to it
            Button button = FindViewById<Button>(Resource.Id.myButton);

            button.Click += delegate
            {
                button.Text = dijeljen_kod.Povecaj();
```

Programski kod 2: Pozivanje funkcije Povecaj(), Android platforma

Nakon pokretanja na virtualnom uređaju sučelje android aplikacije izgleda ovako:



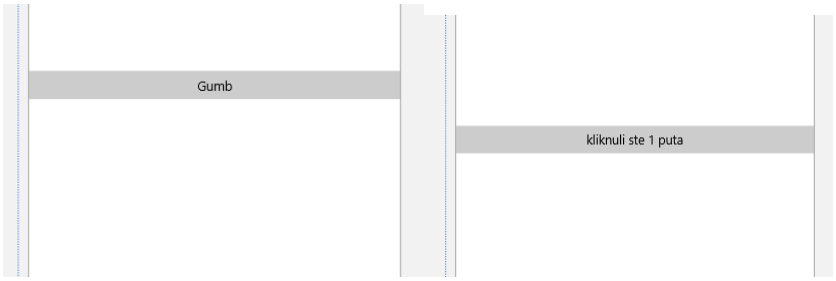
Slika 4: Sučelje Android aplikacije

Klikom na gumb prikazuje se ukupni zbroj klikova. Sada ćemo to isto ponoviti na Windows platformi, otvorit ćemo xaml.cs datoteku u windows dijelu projekta koja kontrolira sučelje(.xaml datoteka) windows platforme i ponovimo postupak od prije [4].

```
namespace App2.UWP
{
    public sealed partial class MainPage : Page
    {

        public MainPage()
        {
            this.InitializeComponent();
            MyClass dijeljeni_kod = new MyClass();
            myButton.Click += (s, e) =>
            {
                myButton.Content= dijeljeni_kod.Povecaj();
            };
        }
```

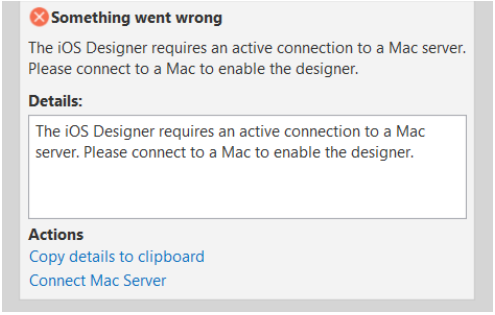
Programski kod 3: Pozivanje funkcije Povecaj(), Windows platforma

Kada pokrenemo emulator aplikacija bi trebala imati iste funkcionalnosti kao i na android platformi.



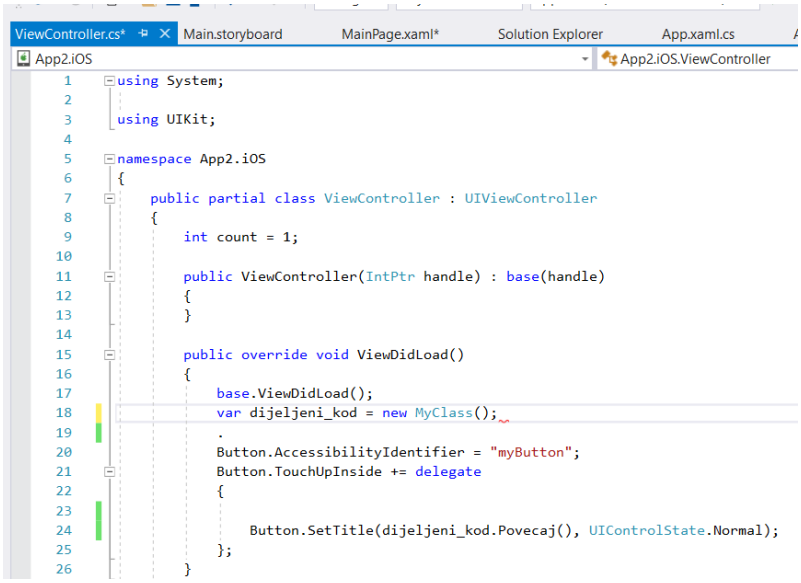
Slika 5: Prikaz sučelja windows aplikacije

Nažalost iOS aplikaciju ne mogu simulirati jer mi je za to potreban Mac računalo ili iPhone uređaj.



Slika 6: Greška kod pokretanja aplikacije, iOS platforma

Zato ću ovaj put demonstrirati samo kod. Otvorite ViewController.cs u iOS dijelu projekta, kao i kod prošlih platforma dodajte događaj gumbu i u događaju pozivamo funkciju Povecaj() iz glavne klase.

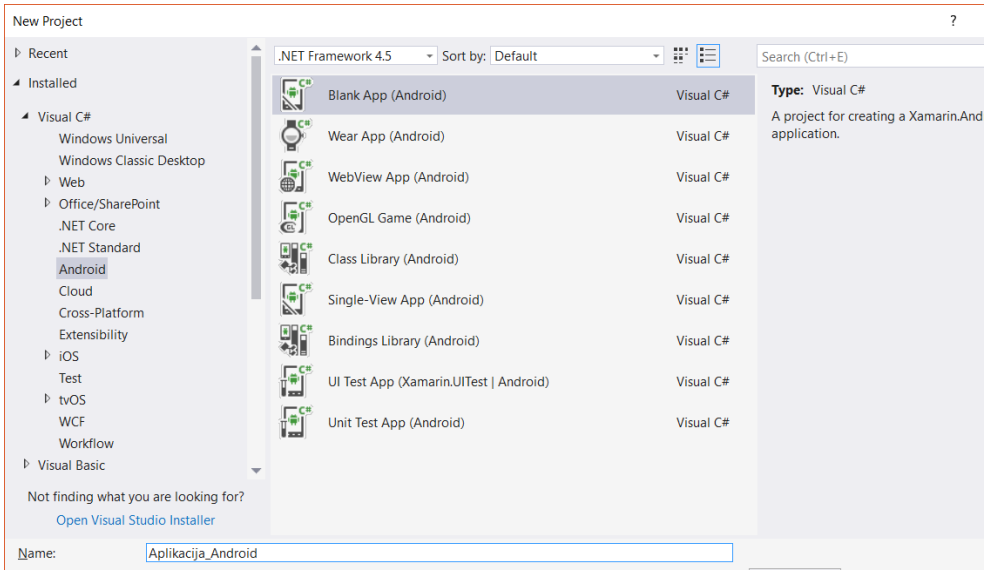


Programski kod 4: Pozivanje funkcije Povecaj(), iOS platforma

4. Xamarin Android aplikacija

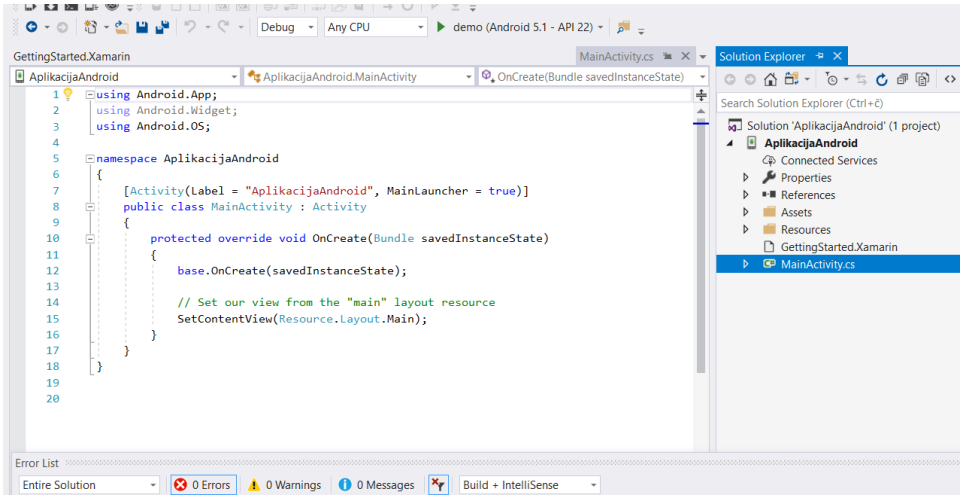
4.1. Kreiranje nove android aplikacije

U poglavlju će biti predstavljen postupak izrade jednostavne Android aplikacije. Otvorite Visual Studio i u glavnom izborniku odaberite File->New->Project. Pojavit će vam se izbornik u kojem ponuđeno koju vrstu projekta želite kreirati, za kreiranje Android aplikacije odaberite Visual C#->Office/Share point->Android->Blank App(Android). Upišite ime aplikacije i kliknite „OK“.



Slika 7: Kreiranje android aplikacije

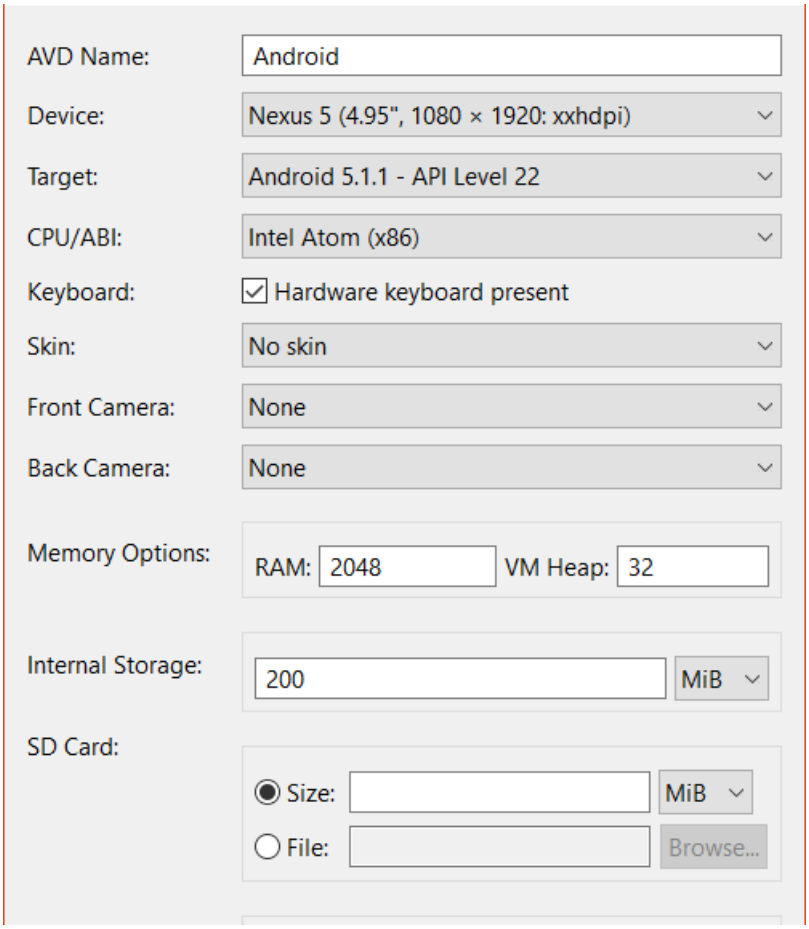
Prvobitno osnovno sučelje za izradu aplikacije izgleda ovako:



Slika 8: Prvotno sučelje aplikacije

4.2. Definiranje virtualnog uređaja

Sljedeće poglavlje obrađivati će postavljanje virtualnog uređaja. Nakon što smo kreirali projekt, moramo definirati virtualni uređaj na emulatoru kako bi mogli pokrenuti aplikaciju u okruženju u kojem će se finalni proizvod pokretati. Svrha emulatora jest da testirate aplikaciju na virtualnim uređajima koji se kreiraju raznim parametrima. Kako bi kreirali postavili virtualni uređaj kliknite Tools->Android->Android Emulator Manager->Create. Otvorit će se prozor s raznim opcijama koje trebate popuniti.

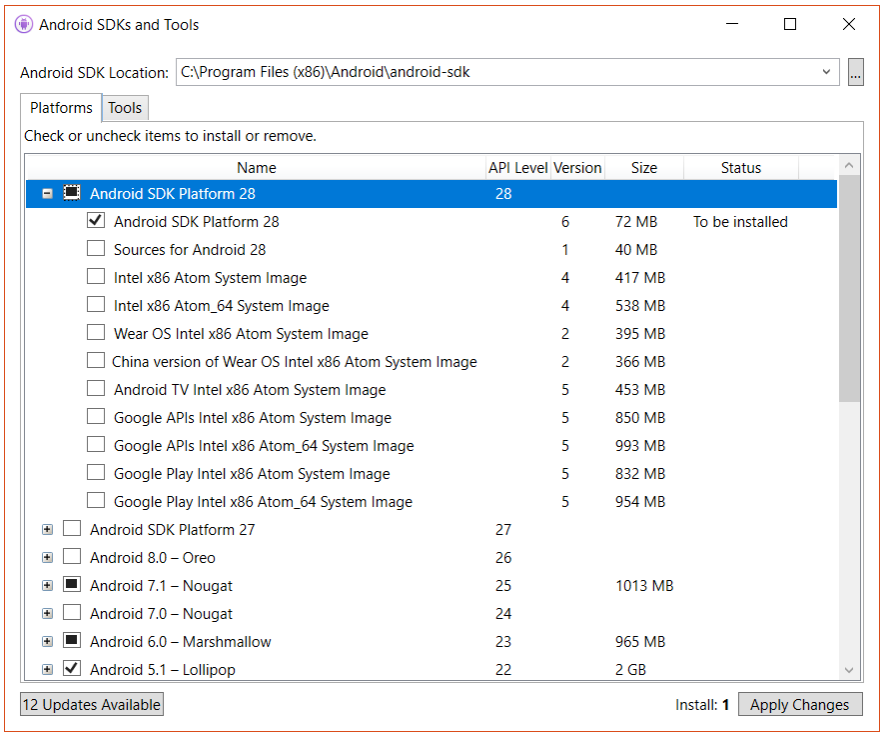


Slika 9: Postavke virtualnog uređaja

- **AVD Name**- Pripadno ime virtualnog uređaja, kako bi određeni uređaj mogli razlikovati od drugih
- **Device**- Odaberite uređaj koji želite simulirati, trenutno je odabran uređaj Nexus 5
- Target- Odabiremo inačicu android sustava, količine i vrste ponuđenih inačica ovise o tome koje inačice su instalirane ako ne vidite ponuđeno inačicu koju bi htjeli odabrati klinite na Tools(glavni izbornik) -> Android -> SDK Manager.
- **SDK**- Software development kit

- **SDK Manager**- Omogućuje korisnicima kreiranje aplikacije koristeći platformu za Android, također sadrži alate, emulator te pomoćne datoteke za pokretanje Android aplikacija

Otvorit će se prozor u kojem su ponuđeni izbornici „Platforms“ i „Tools“, automatski je odabran izbornik „Platforms“. U trenutno otvorenom izborniku su ponuđene razne inačice sustava, odaberite potrebnu i kliknite Apply changes.



Slika 10: SDK Menager

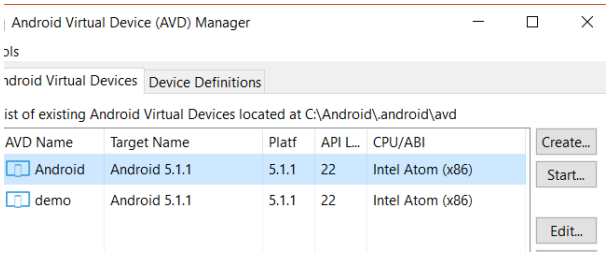
Potrebne datoteke za inačicu sustava će biti instalirane i ona će odsad biti dostupna. Ako trebate instalirate neke od alata, odaberite izbornik „tools“.

- **CPU/ABI** – Odaberite „Intel Atom(x86)“ kako bi pokrenuli ubrzanje pokretanja emulatora i simulacije virtualnog uređaja
- **Skin**- Odaberite izgled virtualnog uređaja,„Skin with dynamic hardware controls“ znači da će se na emulator sa strane pokazati dodatne kontrole, ako želite simulirati samo ekran uređaja odaberite „No skin“

Ako na virtualnom uređaju želite simulaciju kamere možete odabrati opciju Webcam() koja će koristiti kameru na vašem računalu tijekom simulacije, ako imate istu.

- **Memory options**- Virtualnom uređaju treba podesiti količinu radne memorije i VM heap- Virtual machine heap, koliko će vaš virtualni uređaj maksimalno odvojiti memorije po aplikaciji (najmanje 16MB)
- **Internal storage**- Koliko će virtualni uređaj imati memorije na raspolaganju
- **SD card**- na računalu će biti napravljena datoteka koja će simulirati SD-karticu i svim unesenim podacima može se pristupiti direktno
- **Use host GPU**- ako omogućite ovu opciju, emulator će koristiti grafičke sposobnosti računala, preporuča se ako želite ubrzati rad emulatora.

Nakon što ste podesili sve parametre kliknite „OK“ i virtualni uređaj je kreiran, ako ga želite odmah pokrenuti označite ga u izborniku i kliknite start, emulator bi se trebao pokrenuti nakon nekoliko trenutaka



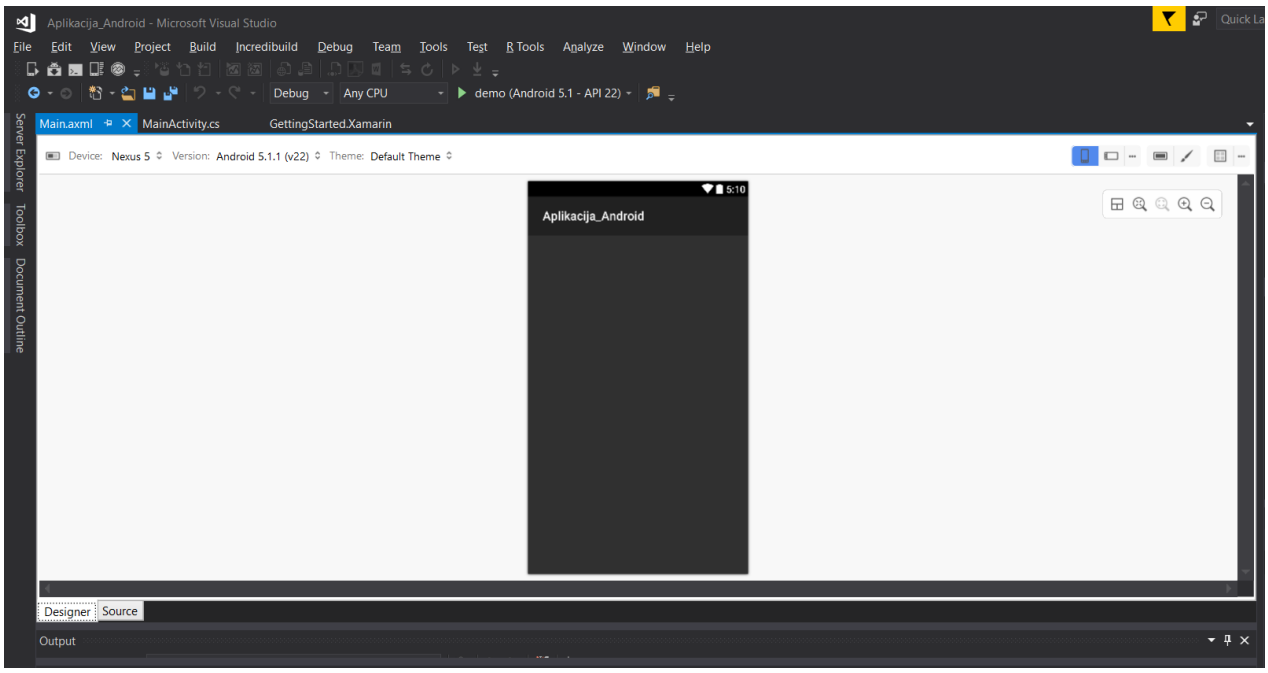
Slika 11: Android Virtual Device Menager

Slika 12: Pokrenut emulator

4.3. Pokretanje aplikacije na virtualnom uređaju

Sada ćemo demonstrirati pokretanje aplikacije na virtualnom uređaju. Prije pokretanja aplikacije na virtualnom uređaju moramo podesiti grafičko sučelje aplikacije kako bi

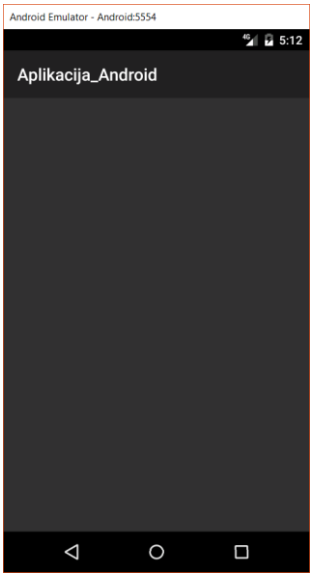
odgovaralo sučelju virtualnog uređaja. Da možemo prilagoditi grafičko sučelje aplikacije kliknemo na Resources->layouts->Main.axml.



Slika 13: Dizajn sučelja

Otvorila nam se opcije „Device“ i „Version“. Parametre postavljamo na osnovi parametara virtualnog uređaja.

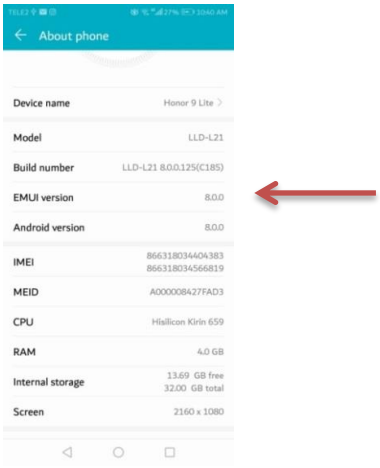
Nakon što smo podesili grafičko sučelje možemo pokrenuti aplikaciju, ako nema grešaka nakon nekoliko trenutaka aplikacija bi se trebala otvoriti u emulatoru.



Slika 14: Prvotno sučelje otvoreno u emulatoru

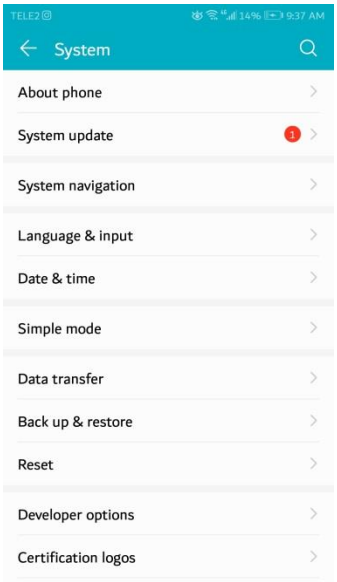
4.4. Pokretanje aplikacije na fizičkom uređaju

U ovom poglavlju bit će objašnjen postupak pokretanja aplikacije na fizičkom uređaju. Kako bi omogućili pokretanje aplikacije, moramo na uređaju aktivirati „usb debugging“, opcija je najčešće „skrivena“ da ne može biti slučajno aktivirana. Na mojem uređaju opija se uključuje postupkom Settings-> About Phone i stisnite „Build number“ 7 puta.

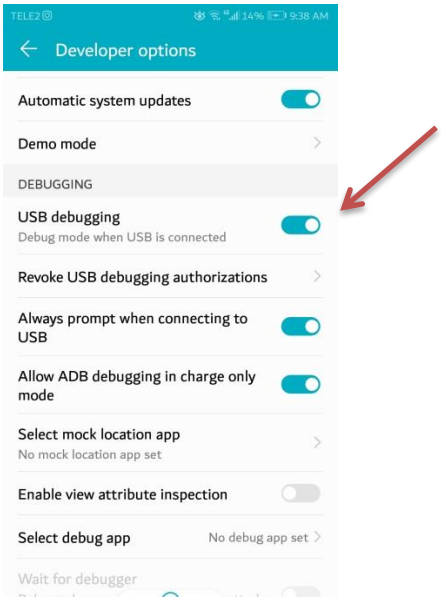


Slika 15: Postavke na fizičkom uređaju

Kada je uključen „developer mode“, vratite se u postavke i kliknite na „developer options“, bit će vam dostupne razne opcije, kliknite na „USB debugging“, sada možemo spojiti uređaj na računalo pomoću USB kabla, ako prije niste spajali uređaj, računalo će automatski instalirati program koji omogućava prepoznavanje uređaja, prijenos datoteka s uređaja/ na uređaj, punjenje i pokretanja aplikacija.

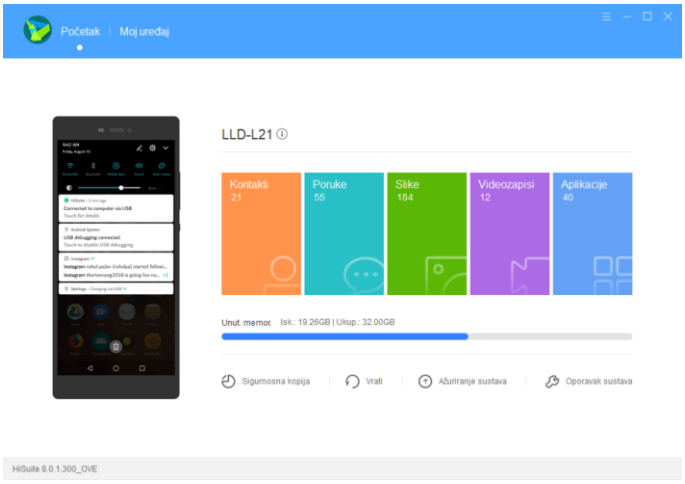


Slika 16: Klasične postavke



Slika 17: Developer postavke

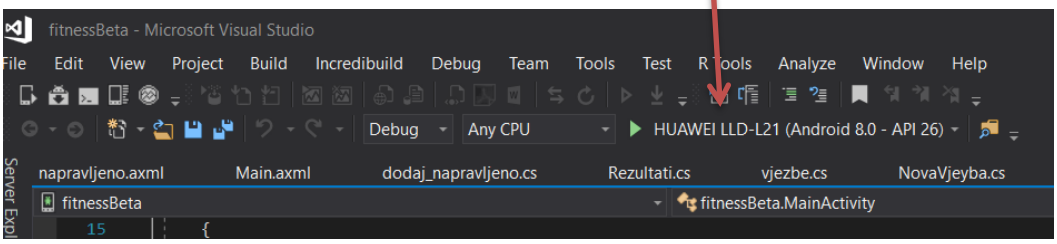
Program za moj uređaj, u desnom kutu je ime uređaja po kojem ga identificiramo



Slika 18: Program za spajanje uređaja na računalo

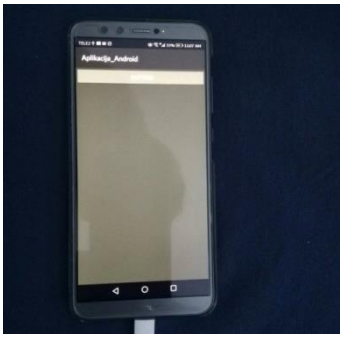
Program bi se trebao automatski otvoriti svaki puta kada se uređaj spoji na računalo.

Kada ponovno pokrenemo visual studio i otvorimo aplikaciju, kod odabira uređaja trebalo bi se pojaviti ime našeg mobitela.



Slika 19: Uređaj je postavljen umjesto emulatora

Pokrenite aplikaciju, trebala bi se pokrenuti na vašem uređaju.



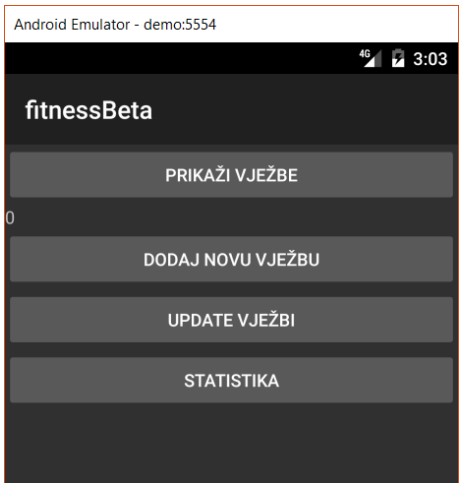
Slika 20: Aplikacija pokrenuta na fizičkom uređaju

4.5. Izrada aplikacije

U sljedećem poglavlju demonstrirati ću izradu jednostavne android aplikacije u C# programskom jeziku. Nakon što smo kreirali virtualni uređaj i istom prilagodili grafičko sučelje možemo krenuti na samu izradu. Aplikacija koju ćemo izraditi služit će kod treninga u teretani, za početak funkcije aplikacije bit će spremanje vježbi, ciljanih rezultata te ciljanih rezultata, te prikaz statistike napredovanja.

Slika 21: Prvotna arhitektura sučelja

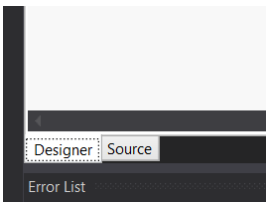
Za početak početno sučelje će sadržavati 4 gumba, za prikaz svih vježbi, dodavanje novih vježbi, update već unesenih vježbi i statistiku.



Slika 22: Prvotno sučelje u emulatoru

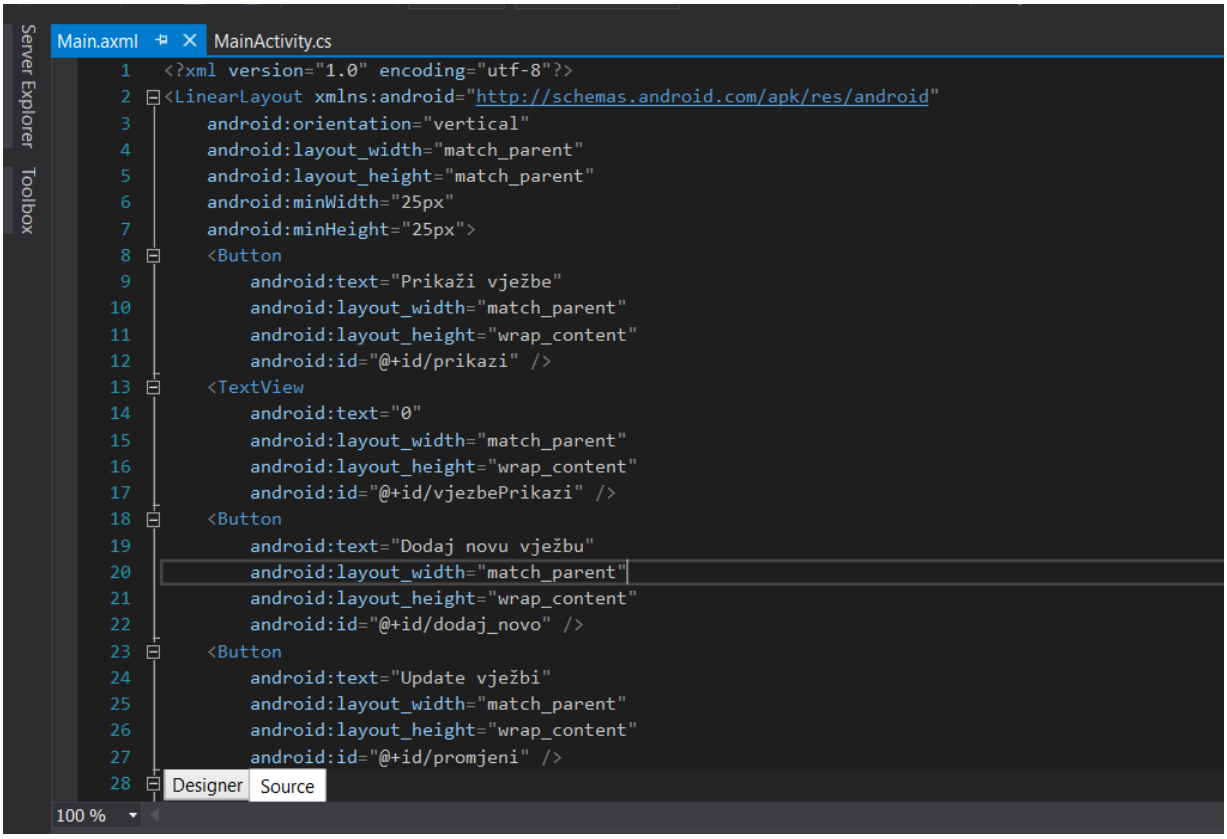
Kako bi dodali gumbe otvorite „Toolbox“ kliknite „Button“ i odvucite ga na sučelje te mu promjeni te ime u „prikazi“.

Sučelje možete prikazati u 2 oblika, „Designer“ i „source“, trenutno je sučelje u „Designer“ obliku ako želite detaljniji prikaz svojstvima dizajna, kliknite na source.



Slika 23: Odabir detaljnijeg prikaza sučelja

Otvoriti će se AXML (Android Xamarin.layout) datoteka.



Programski kod 5: Main.axml datoteka

AXML datoteka opisuje dizajn grafičkog sučelja, ovdje možemo direktno raditi promjene na sučelju kao naprimjer dodavati razne funkcije, detaljno odrediti visinu, širinu istih, ali na to ćemo se vratiti malo kasnije kada ćemo mijenjati osnovni dizajn. Svaki alat ime svoj id koji služi za identifikaciju određenog alata kako bi mu dodijelili dodatne funkcije, kao što ćemo vidjeti u nastavku [3].

4.5.1. Promjena sučelja klikom na gumb

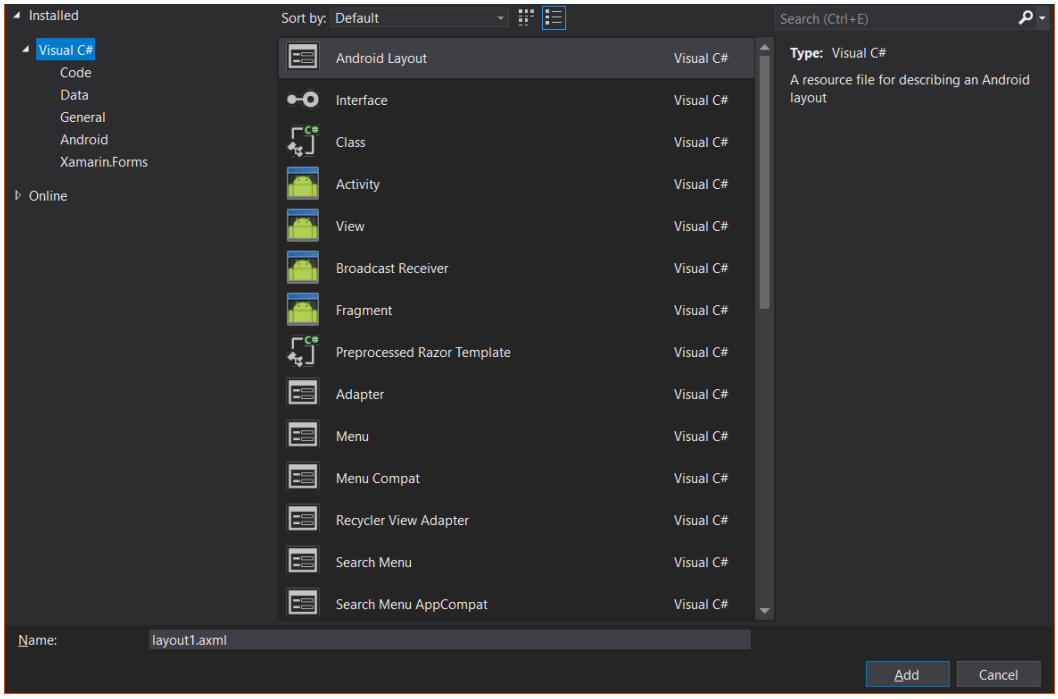
Poglavlje će opisivati postupak promjene sučelja klikom na gumb. Kada se aplikacija pokrene, prvo će otvoriti MainActivity.cs kako bi se otvorilo početno sučelje moramo ga pozvati pomoću naredbe SetContentView() u kojoj definiramo koje sučelje se otvaramo, u ovom slučaju to izgleda ovako:

```
base.OnCreate(savedInstanceState);
SetContentView(Resource.Layout.Main);
```

Programski kod 6: Postavljanje sučelja

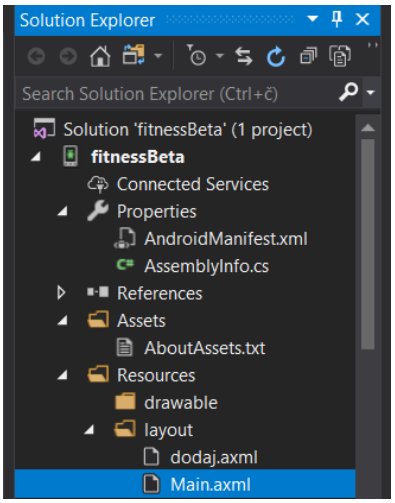
To znači da će se otvoriti sučelje Main. Kada kliknemo na gumb „dodaj novu vježbu “sučelje” bi se trebalo promijeniti [4].

Ako želimo da se klikom na gumb promjeni grafičko sučelje i dodatne funkcije aplikacije postanu dostupne, prvo moramo kreirati spomenuto sučelje. Kliknite u „Solution explorer-u“ „Resource“ i desni klik na „layout“, odaberite „Add“->“New item“, otvorit će vam se prozor s različitim vrstama sučelja.



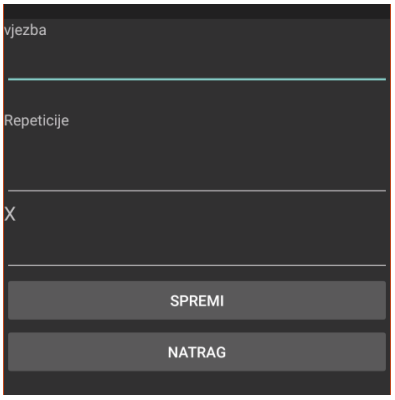
Slika 24: Dodavanje novog sučelja

Odaberite prvi „Android Layout“, promijenite ime u „dodaj“ i kliknite Add, u mapi layout trebao bi se pojaviti nova XAML datoteka.



Slika 25: Otvaranje sučelja

Kliknite 2x na novu datoteku te će se otvoriti prazno sučelje. U sučelje ćemo dodati 5 TextBox-a i 2 gumba te ćemo ih razmjestiti na sljedeći način:



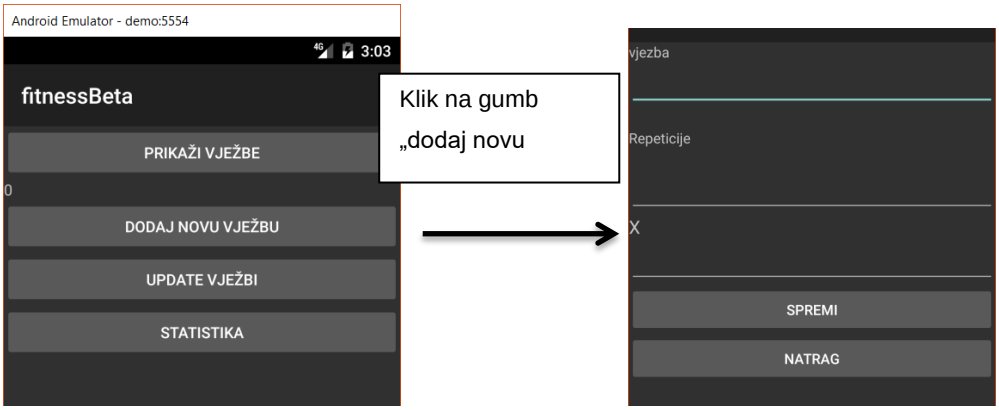
Slika 26: Dizajn novog sučelja

Nakon sučelja moramo kreirati aktivnost koja će pokretati sučelje. Kliknite u Solution Exploreru desnim klikom na projekt, odaberite „Add item“, odaberite u prozoru „Activity“, promijenite ime u „NovaVjeyba“ i kliknite „Add“. Otvorite novu aktivnost (2 klika u Solution exploreru). Kad se pokrene aktivnost „NovaVjeyba“ trebalo bi pokrenuti i otvoriti sučelje „dodaj“. Kao što sam spomenula prije to radimo s naredbom setContentView..

```
namespace fitnessBeta
{
    [Activity(Label = "NovaVjeyba")]
    public class NovaVjeyba : Activity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

            SetContentView(Resource.Layout.dodaj);
```

Programski kod 7: Kreiranje nove aktivnosti

Kada smo kreirali sučelje i aktivnost, možemo kreirati funkciju koja će klik gumba „dodaj novu aktivnost“ pokrenuti aktivnost „NovaVjeyba“ koja otvara sučelje „dodaj“. Kako bi to postigli otvorite datoteku MainActivity.cs



Slika 27: Prijelaz sučelja na klik gumba

Kako bismo to ostvarili prvo ćemo kreirati novu dretvu koja će služiti za pokretanje novog sučelja, dretvu ćemo nazvati “NovaVjezba”. Nakon što smo napravili dretvu kreiramo novu varijablu “novo” tipa “Button” kojoj pridružujemo gumb “dodaj_novo”.

```
23
24          Button novo = FindViewById<Button>(Resource.Id.dodaj_novo);
25          novo.Click += delegate
26          {
27
28              // StartActivity(typeof(NovaVjeyba));
29              NovaVjezba = new Thread(new ThreadStart(delegate
30              {
31                  try
32                  {
33                      Thread.Sleep(10);
34                  }
35                  catch(Exception e)
36                  {
37                      e.ToString();
38                  }
39                  finally
40                  {
41
42                      StartActivity(new Intent(Application.Context, typeof(NovaVjeyba)));
43                  }
44
```

Programski kod 8: Pokretanje aktivnosti klikom na gumb

Varijabli novo kreiramo događaj “Click” i nakon događaja pokrećemo dretvu, ako nema nikakvih smetnji pokreće se aktivnost “NovaVjeyba” u kojoj smo definirale otvaranje sučelja “dodaj” , odnosno kada netko klikne na gumb “dodaj novu vježbu” otvara se novo sučelje u kojem možemo definirati parametre.

4.5.2. Unos vježbi

U poglavlju bit će pojašnjeno korištenje SQLlite u svrhu izrade I Kako bi korisniku omogućili unos novih vježbi potrebno ih je negdje spremiti, a za spremanje moramo kreirati bazu podataka. Bazu ćemo kreirati tako da sve podatke sprema u korisnikovu memoriju.

```
Beta                                    fitnessBeta.MainActivity                    OnCreate(Bundle savedInstanceState)
using System.IO;
using System.Threading;
using System;
using SQLite;
namespace fitnessBeta
{
    [Activity(Label = "fitnessBeta", MainLauncher = true)]
    public class MainActivity : Activity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            Thread NovaVjezba;
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Main);
            string dbPat = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "MihaelTerten.db3");
            var db = new SQLiteConnection(dbPat);
```

Programski kod 9: Kreiranje baze podataka

Kada su baza I konekcija napravljene, kreiramo novu klasu “vjezbe”(projekt->add item->class) u kojoj ćemo definirati u kojem obliku se podaci spremaju u bazu. Također ćemo kreirati funkcije koje pomažu kod korištenja klase.

```
16      class vjezbe
17      {
18          public string ime_vjezbe { get; set; }
19          public string ciljano_r1 { get; set; }
20          public string ciljano_r2 { get; set; }
21          public string postignuto_r1 { get; set; }
22          public string postignuto_r2 { get; set; }
23
24          public vjezbe(string v, string c, string p, string s, string x)
25          {
26              ime_vjezbe = v;
27              ciljano_r1 = c;
28              ciljano_r2 = p;
29              postignuto_r1 = s;
30              postignuto_r2 = x;
31
32          }
```

Programski kod 10: Klasa vjezbe

Prvo definiramo varijable u obliku u kojem ih želimo unositi u bazu.

- ime_vjezbe- ime vježbe po kojem ćemo vježbu ubuduće identificirati
- ciljano_r1- ciljana količina repeticija
- ciljano_r2- ciljana repeticija setova

- postignuto_r1- postignuta količina repeticija
- postignuto_r2- postignuta količina setova

Naprimjer želimo unijeti vježbu čučnjevi s ciljanim rezultatom od 50 repeticija u 5 setova. Korisnik također može kasnije osvježiti parametre za postignuti rezultat, kod unošenja parametri su postavljeni na 0. Nakon toga kreiramo funkciju koja nam služi da unesene vrijednosti pridružimo varijablama

U klasi nam također treba funkcija koja će iz baze vraćati podatke o rezultatima ovisno o imenu vježbe.

```
public vjezbe VratiPlan(string vjezba)
{
    string dbPat = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "MihaelTerten.db3");
    var db = new SQLiteConnection(dbPat);
    return db.Table<vjezbe>().Where(x => x.ime_vjezbe == vjezba).SingleOrDefault();
```

Programski kod 11: Funkcija VratiPlan()

Sada možemo u aktivnosti “NovaVjeyba” kreirati konekciju za bazu i kreirati varijablu “spr” tipa Button kojoj pridružujemo gumb Spremi.

```
    SetContentView(Resource.Layout.dodaj);
    string dbPat = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "MihaelTerten.db3");
    var db = new SQLiteConnection(dbPat);

    Button spr = FindViewById<Button>(Resource.Id.spremi);
    spr.Click += delegate
    {

        var vj = FindViewById<EditText>(Resource.Id.ime_vjez);
        string vjezba = vj.Text;
        var r1 = FindViewById<EditText>(Resource.Id.r1);
        string rep1 = r1.Text;
        var r2 = FindViewById<EditText>(Resource.Id.r2);
        string rep2 = r2.Text;

        db.CreateTable<vjezbe>();
        vjezbe primjer = new vjezbe(vjezba, rep1, rep2, "0", "0");
        db.Insert(primjer);
```

Programski kod 12: Spremanje podataka u bazu nakon klika na gumb spremi

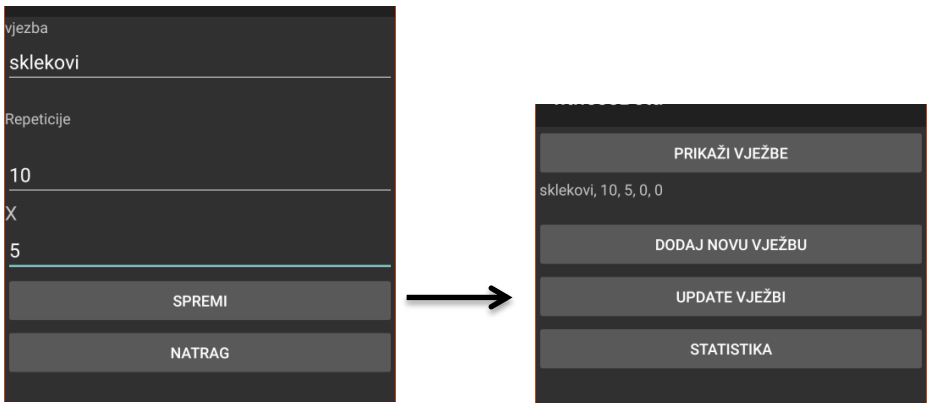
Kreiramo događaj “Click” nakon kojeg se u varijable vj, r1 i r2 unose podaci iz tekstualnih polja. Nakon toga otvaramo tablicu koja je kreirana na temelju varijabli iz klase “vjezbe”, definiramo varijablu primjer tipa “vjezbe” u koji spremamo unesene parametre I unosimo parametre u bazu. Kako bismo provjerili da li je unos ispravno napravljen, vraćamo se na aktivnost “Main” i definiramo događaj na gumbu “Prikaži vježbe” koje će prikazati sve unesene vježbe i njihove parametre.

4.5.3. Prikaz unesenih vježbi

```
Button prikaziPod = FindViewById<Button>(Resource.Id.prikazi);
prikaziPod.Click += delegate
{
    TextView prikaziText = FindViewById<TextView>(Resource.Id.vjezbePrikazi);

    prikaziText.Text = " ";
    var tablica = db.Table<vjezbe>();
    foreach(var item in tablica)
    {

        vjezbe vjezbee = new vjezbe(item.ime_vjezbe, item.ciljano_r1, item.ciljano_r2, item.postignuto_r1, item.postignuto_r2);
        prikaziText.Text += vjezbee+ "\n";
    }
```

Programski kod 13: Prikaz svih unesenih vježbi i trenutnih rezultata

Kako bismo prikazali podatke, moramo otvoriti tablicu u koju smo umetnuli vježbu, prolazimo kroz sve redove tablice i parametre spremamo u varijablu “vjezbe” te ih unosimo u tekstualno polje.



Slika 28: Prikaz unesenih vježbi i trenutnih rezultata

Kao što vidimo, vježbe su se spremljene te se prikazuju nakon što kliknemo na gumb “Prikaži”.

4.5.4. Update vježbi

Kako bi korisnik mogao osvježiti prvotno upisane parametre odnosno zabilježiti postignuti napredak u vježbama kreirat ćemo novo sučelje “napravljeno.axml” I aktivnost “dodaj_napravljeno.cs”. U sučelje ćemo ubaciti padajući izbornik(listaVjezbi), 4 polja za unošenje(PlanRepeticije, PlanRepeticije2, Postignuto, Postignuto2, gumb za spremanje i tekstualna polja za snalaženje. Korisnik će odabrati vježbu kojoj želi izmijeniti rezultate te će mu se pojaviti trenutni rezultati koje treba izmijeniti i klikom na gumb spremi ažurirati podatke u bazi.

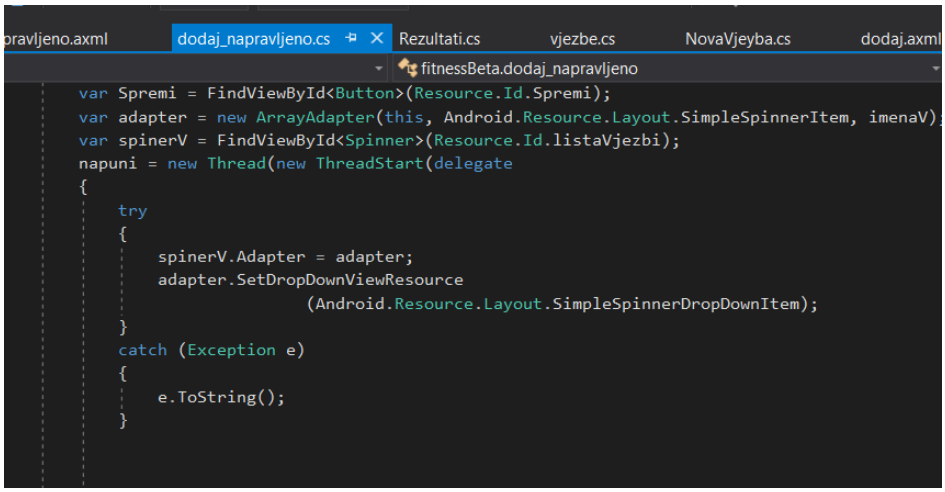
Slika 29: Sučelje za osvježavanje unesenih rezultata

Kada smo dizajnirali sučelje, otvaramo aktivnost “napravljeno”, kreiramo varijable i pridružujemo im elemente sučelja. Također kreiramo dvije dretve koje ćemo pokrenuti kada korisnik odabere željenu vježbu kako bi se omogućio prikaz trenutno unesenih parametara svake vježbe.

```
protected override void OnCreate(Bundle savedInstanceState)
{
    Thread napuni, odaberi;
    base.OnCreate(savedInstanceState);
    SetContentView(Resource.Layout.napravljeno);
    string dbPat = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal),
    var db = new SQLiteConnection(dbPat);
    var iv = db.Table<vjezbe>().GroupBy(s => s.ime_vjezbe).Select(s => s.First());
    var imenaV = iv.Select(s => s.ime_vjezbe).ToList();
    var PlanRepeticije = FindViewById<EditText>(Resource.Id.rep1);
    var PlanRepeticije2 = FindViewById<EditText>(Resource.Id.rep2);
    var Postignuto = FindViewById<EditText>(Resource.Id.pon);
    var Postignuto2 = FindViewById<EditText>(Resource.Id.pon2);
    var Spremi = FindViewById<Button>(Resource.Id.Spremi);
    var adapter = new ArrayAdapter(this, Android.Resource.Layout.SimpleSpinnerItem, imenaV);
    var spinerV = FindViewById<Spinner>(Resource.Id.listaVjezbi);
```

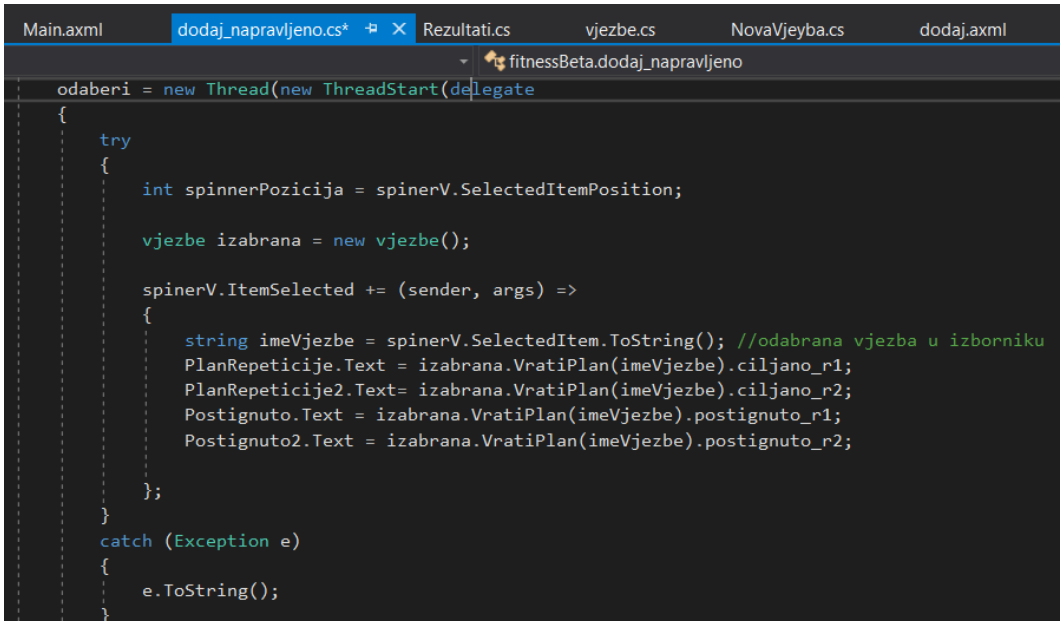
Programski kod 14: kreiranje konekcije sa bazom, pridruživanje elementa sučelja, kreiranje adaptera

Kako bi padajući izbornik prikazivao sve trenutno unesene vježbe, radimo konekciju s bazom, kreiramo varijablu “iv” u koju spremamo tablicu “vježbe”. Nakon toga kreiram varijablu “imenaV” i u njoj sva imena vježbi iz tablice “vježbe” stavljamo u listu. Kreiramo varijablu “adapter” u kojoj definiramo adapter čiji je zadatak pridruživanje liste s imenima i padajućeg izbornika. Nakon toga definiramo dretvu “napuni” u kojoj povezujemo padajući izbornik i adapter te ju pokrećemo. Ako je proces uspješan klikom na padajući izbornik korisnik dobiva prikaz imena vježbi trenutno unesenih u bazu.



Programski kod 15: Ispuna padajućeg izbornika sa podacima iz baze

Sljedeći korak je korisniku omogućiti odabir vježbe te prikaz svih trenutno unesenih parametara, koji će omogućiti korisniku željenu izmjenu određenih podataka.



Programski kod 16: Dretva odaberi

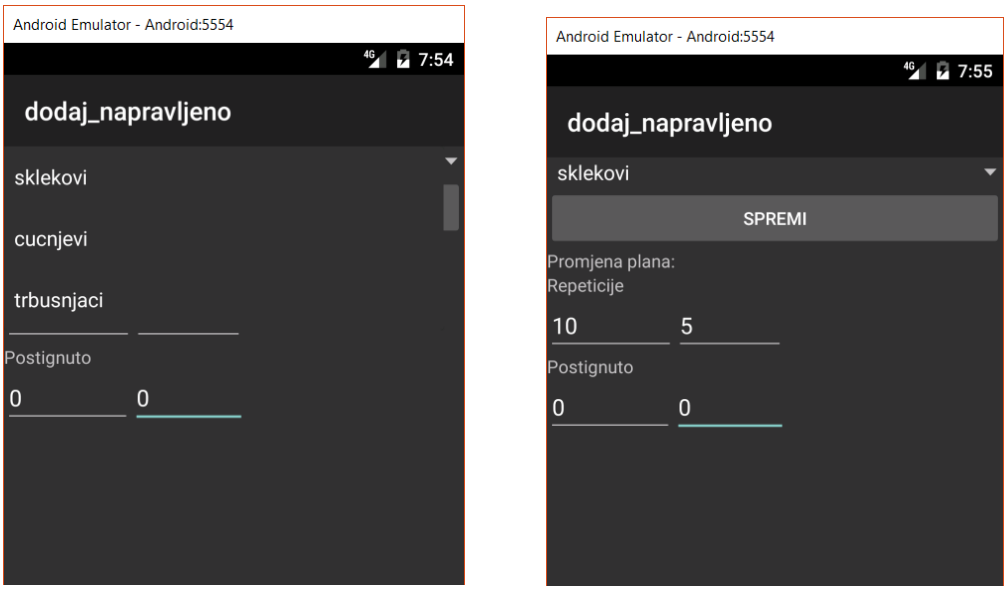
Definiramo dretvu “odaberi” u kojoj kreiramo varijablu “spinnerPozicija” i pridružujemo joj trenutnu poziciju padajućeg izbornika, također kreiramo varijablu “izabrana” tipa vježbe. Nakon toga padajućem izborniku(SpinerV) definiramo događaj “ItemSelected” u kojem se kreira varijabla ImeVjezbe i pridružujemo joj odabrani element u izborniku (ime vježbe koju je korisnik odabrao). U klasi “vjezbe” kreiramo funkciju “VratiPlan”.

```
Main.axml   dodaj_napravljeno.cs*   Rezultati.cs   vjezbe.cs   NovaVjeyba.cs   dodaj.axml   MainActivity.cs
fitnessBeta.vjezbe                                                    VratiPlan(string vjezba)
}
public override string ToString()
{
    return string.Format("{0}, {1}, {2}, {3}, {4}", ime_vjezbe, ciljano_r1, ciljano_r2, postignuto_r1, postignuto_r2);
}

public vjezbe VratiPlan(string vjezba)
{
    string dbPat = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "MihaelTerten.db3");
    var db = new SQLiteConnection(dbPat);
    return db.Table<vjezbe>().Where(x => x.ime_vjezbe == vjezba).SingleOrDefault();
```

Programski kod 17: Funkcija VratiPlan()

Njezin zadatak je da vrati red u tablici na temelju proslijeđenog imena vježba. Pomoću varijable izabrana prosljeđujemo ime vježbe i iz vraćenog reda tablice možemo izabrati atribute koje pridružujemo poljima za unos teksta.

Naprimjer, kada korisnik odabere “sklekovi”, trebali bi se pojaviti prethodno uneseni parametri.



Slika 30: Prikaz trenutno unesenog plana vježbe

Sljedeća funkcionalnost koju trebamo omogućiti korisniku je osvježavanje parametara (spremanje napretka). Prvo što moramo je gumbu “Spremi” definirati događaj “Click”, odnosno kada se gumb klikne, napredak se sprema.

```
Spremi.Click += delegate
{

    string imeVjezbe = spinerV.SelectedItem.ToString();
    db.Execute("UPDATE vjezbe SET ciljano_r1= ?, ciljano_r2= ?, postignuto_r1= ?, postignuto_r2= ? WHERE ime_vjezbe == ? "
        , PlanRepeticije.Text, PlanRepeticije2.Text, Postignuto.Text, Postignuto2.Text, imeVjezbe);
```

Programski kod 18: Spremanje novih rezultata

Kako bi spremili osvježene parametre, na bazi vršimo upit pomoću kojeg unosimo nove parametre u red tablice koji sadrži ime vježbe koju je korisnik odabrao. Prijašnje rezultate ne smijemo izgubiti jer na temelju njih kasnije ćemo izraditi statistiku napretka, zato ćemo kreirati klasu “rezultati” na temelju koje ćemo izraditi tablicu u bazi.

```
class rezultati
{
    public int redni_broj { get; set; }
    public string postignuto_r1 { get; set; }
    public string postignuto_r2 { get; set; }
    public string ime_vjezbe { get; set; }

    public rezultati(int broj, string ime, string p1, string p2)
    {
        ime_vjezbe = ime;
        redni_broj = broj;
        postignuto_r1 = p1;
        postignuto_r2 = p2;

    }
    public rezultati()
```

Programski kod 19: Klasa rezultati

U klasi ćemo kreira 4 varijable, 2 koje će spremati rezultat, ime vježbe na koju se rezultat odnosi i redni broj pomoću kojeg ćemo poredati unesene rezultate kako bi na temelju njih korisnik mogao kronološki pregledati sve rezultate.

```
Thread NovaVjezba;
base.OnCreate(savedInstanceState);
SetContentView(Resource.Layout.Main);
string dbPat = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "MihaelTerten.db3");
var db = new SQLiteConnection(dbPat);
db.CreateTable<vjezbe>();
db.CreateTable<rezultati>();
Button novo = FindViewById<Button>(Resource.Id.dodaj_novo);
novo.Click += delegate
```

Programski kod 20: Kreiranje tablice rezultati

Kreiramo tablicu “rezultati”.

```
try
{
    var upit_string = db.Query<rezultati>("SELECT * FROM rezultati  WHERE ime_vjezbe == ? ", imeVjezbe).LastOrDefault();

    int id_novi = upit_string.redni_broj;
    id_novi++;
    rezultati Update_rezultati = new rezultati(id_novi, imeVjezbe, Postignuto.Text, Postignuto2.Text);
    db.Insert(Update_rezultati);
```

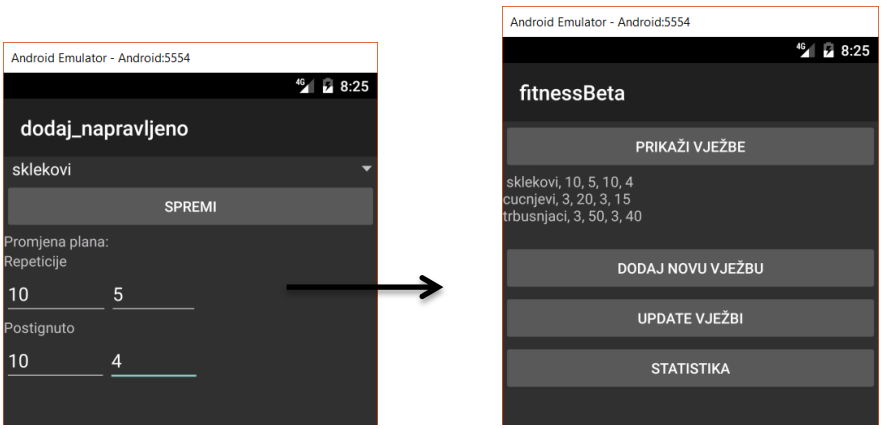
Programski kod 21: Spremanje rezultata u tablicu

Kada smo osvježili rezultate u tablici “vjezbe” nakon klika na gumb spremi, moramo spremiti stare parametre u tablicu “rezultati”. Prije spremanja moramo provjeriti da li su toj vježbi ikad mijenjani parametri, zato u varijabli “upit_string” kreiramo upit koji provjerava da li u tablici rezultati postoji ta vježba i ako postoji izabiremo zadnje unesene rezultate od vježbe. U varijablu id_novi spremamo redni broj rezultata vježbe i uvećavamo ge za jedan, potom stare parametre s uvećanim rednim brojem spremamo u tablicu rezultati.

```
    }
    catch(Exception e)
    {

        e.ToString();
        rezultati Update_rezultati = new rezultati(1, imeVjezbe, Postignuto.Text, Postignuto2.Text);
        db.Insert(Update_rezultati);

    }
```

Programski kod 22: Spremanje početnih rezultata

Ako se ispostavi da vježbi nikad nisu mijenjani rezultati i ovo je prvi puta da se rezultati spremaju u tablicu, onda rezultate spremamo u tablicu s rednim brojem 1..

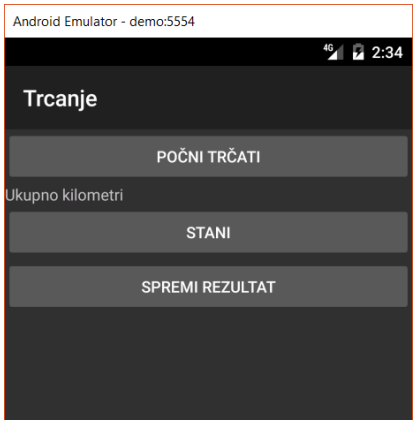


Slika 31: Prikaz vježbi nakon unosa novih podataka

Naprimjer, ako korisnik vježbi “sklekovi” promjeni parametre i klikne spremi, kod sljedećeg prikaza trebali bi se pojaviti osvježeni rezultati.

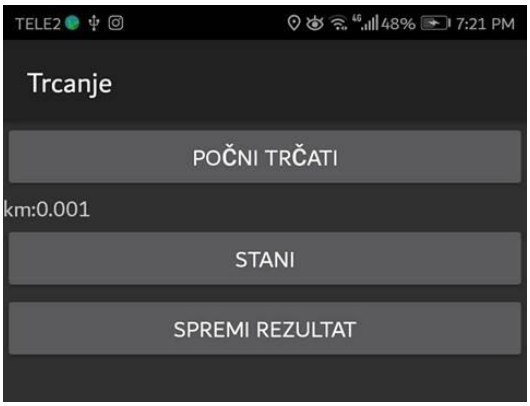
4.5.5. Trčanje

Aplikaciji ćemo dodati funkcionalnost praćenja prijeđene razdaljine, kako bi korisnik mogao pratiti količinu prijeđenih kilometara. Prvo ćemo dodati novo sučelje I nazvati ga trcanje.axml.



Slika 32: Početno sučelje aktivnosti "Trcanje"

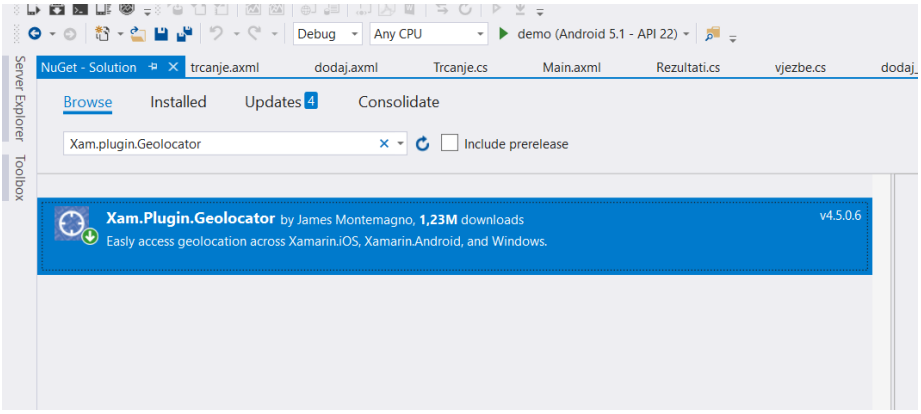
Na sučelje ćemo dodati 3 gumba I nazvati ih start, stop i spremi. Kada korisnik stisne start aplikacija će početi mjeriti prijeđenu udaljenost.



Slika 33: Prikaz prijeđene udaljenosti

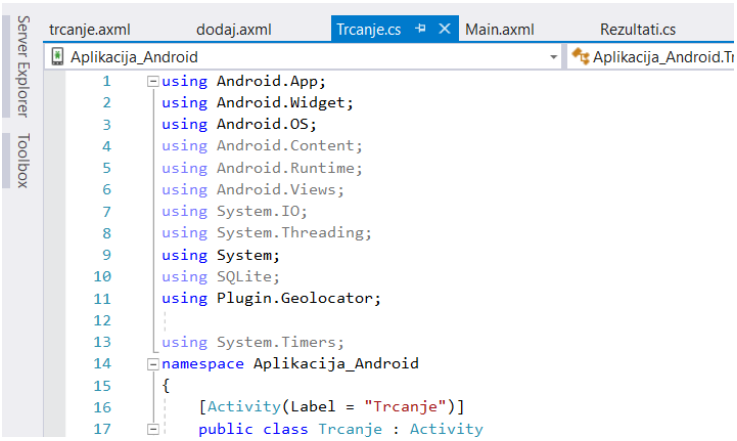
Gumb stani zaustavlja mjerenje ako korisnik zbog nekog razloga želi zaustaviti mjerenje ukopne udaljenosti te se na gumb start ponovno mjerenje pokreće(od udaljenosti gdje je stao). Nakon što je korisnik završio mjerenje klikne na gumb spremi te se prijeđena udaljenost sprema u bazu. Kada smo kreirali sučelje, kreirat ćemo i aktivnost Trcanje.cs kako bi korisniku omogućili korištenje gore spomenutih opcija. Da bi aplikacija mogla mjeriti prijeđenu udaljenost potrebno je omogućiti praćenje pomoću gps-a, u ovoj aplikaciji koristit ćemo Geolocator. Geolocator je skupina Nuget(upravitelj svih dodatnih i osnovnih paketa) biblioteka koja omogućuje jednostavno integriranje opcije trenutne geolokacije [5]. Prvi korak je skidanje I instaliranje plug-ina, pošto je to Nuget paket, otvaramo opcije za Nuget pakete i

u izbornik upisujemo Xam.Plugin.Geolocator, odabiremo najnoviju verziju i kliknemo na Install [6].



Slika 34: Instalacija geolocator plug- ina

Nakon što je Geolocator instalira dodajemo možemo ge uključiti u naš projekt tako da dodamo Plugin [5].Geolocator biblioteku u potrebnu aktivnost.



Programski kod 23: Uključivanje geolocator-a u aktivnost

U ovom slučaju to je aktivnost Trcanje.cs. Korisnikovo kretanje ćemo pratiti tako da prvo lociramo njegovu lokaciju I spremimo ju u varijablu, svakih 10 sekundi lokacija se osvježava I uspoređuje s prijašnjom lokacijom i između njih se izračunava udaljenost, udaljenost se nadodaje na ukupni zbroj prijašnjih udaljenosti koji se u svakoj iteraciji ispisuje na ekranu.

Prvo ćemo kreirati varijablu get_location tipa button kojoj pridružujemo gumb start, također kreiramo dvije varijable tipa double ukupnokm i km u koje ćemo spremati ukupno prijeđenu razdaljinu i trenutno prijeđenu razdaljinu. Kako bi mogli pokrenuti/zaustaviti iteraciju definiramo varijablu a tipa int te ju postavljamo na vrijednost 1. Za pokretanja mjerenja korisnik pritišće gumb start, što znači da varijabli get_location pridružujemo događaj Click.

```
Button get_Location= FindViewById<Button>(Resource.Id.start);

double ukupnokm = 0;
double km = 0;


int a = 1;
get_Location.Click += async delegate
{
    a = 1;
    var locator = CrossGeolocator.Current;
    var position = await locator.GetPositionAsync(TimeSpan.FromSeconds(10), null, true);
    locator.DesiredAccuracy = 20;
    double lat = position.Latitude;
    double lon = position.Longitude;
```

Programski kod 24: Dohvaćanje koordinata nakon klika na gumb

U događaju kreiramo varijablu u kojoj ćemo definirati geolocator i varijablu pomoću koje ćemo locirati poziciju, u varijabli također definiramo vrijeme kroz koje želimo dobiti podatke. Nakon toga lokatoru postavljamo željenu preciznost vraćenih podataka [5]. Kreiramo dvije double varijable lat(latitude) i lon(longitude) u koje se spremaju podaci o geografskoj širini I dužini, odnosno početna lokacija korisnika. Pokrećemo while petlju u kojoj ponovo lociramo korisnika(svakih 10sec).

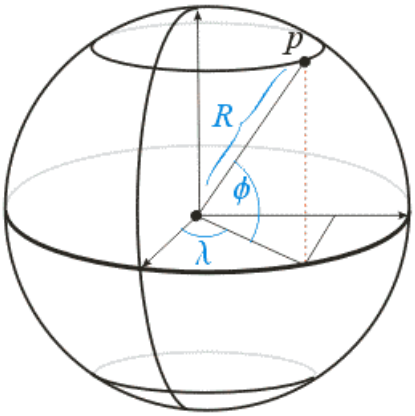
```
while (a == 1)
{
    locator = CrossGeolocator.Current;

    position = await locator.GetPositionAsync(TimeSpan.FromSeconds(10), null, true);
    locator.DesiredAccuracy = 20;
    km = razdaljina(lat, lon, position.Latitude, position.Longitude);

        ukupnokm = ukupnokm + km;
        lat = position.Latitude;
        lon = position.Longitude;


    kilometri.Text = "km:" + ukupnokm.ToString();
}
```

Programski kod 25: Praćenje prijeđene razdaljina

```
trcanje.axml   dodaj.axml   Trcanje.cs*   Main.axml   Rezultati.cs   vjezbe.cs   dodaj_na
lroid                                   Aplikacija_Android.Trcanje
    double razdaljina(double lat1, double lon1, double lat2, double lon2)
    {
        double circumference = 4000;
        double distance = 0.0;
        double latitude1Rad = DegreesToRadians(lat1);
        double longitude1Rad = DegreesToRadians(lon1);
        double latititude2Rad = DegreesToRadians(lat2);
        double longitude2Rad = DegreesToRadians(lon2);
        double logitudeDiff = Math.Abs(longitude1Rad - longitude2Rad);
        if (logitudeDiff > Math.PI)
        {
            logitudeDiff = 2.0 * Math.PI - logitudeDiff;
        }

        double angleCalculation =
            Math.Acos(
                Math.Sin(latititude2Rad) * Math.Sin(latitude1Rad) +
                Math.Cos(latititude2Rad) * Math.Cos(latitude1Rad) * Math.Cos(logitudeDiff));

        distance = circumference * angleCalculation / (2.0 * Math.PI);

        return distance;
```

Programski kod 26: Funkcija razdaljina()

Prijašnje informacije o lokaciji koje su spremljene u varijablama lat i lon uspoređujemo s novom lokacijom u funkciji razdaljina. Funkcija je kreirana na temelju Harversinove formula koja određuje najkraću udaljenost na površini sfere između dvije točke [7].



Slika 35: Prikaz udaljenosti točaka na sferi(Prof. David Bernstein**)**

Ako se točke nalaze na suprotnim krajevima sfere, njihova udaljenost će uvijek biti pola opsega sfere. U funkciji sve dužine I širine prebacujemo iz stupnjeva u radijane pomoću funkcije DegreesToRadians() koja stupnjeve množi s PI te rezultat dijeli sa 180.

```
    double DegreesToRadians(double degrees)
    {
        return degrees * Math.PI / 180.0;
```

Programski kod 27: Funkcija za pretvaranje stupnjeva u radijane

Nakon toga spremamo apsolutnu vrijednost razlike početne dužine i nove dužine, ako je vrijednost veća od PI(3.14159265…) udvostručujemo PI i od rezultata oduzimamo razliku. Nakon toga zbrajamo umnožak sinusa širina i umnožak kosinusa širina pomnoženog s kosinusom apsolutne vrijednosti razlike između dužina. Iz dobivenog rezultata računamo arkuskosinus. Konačnu udaljenost izračunavamo da zemljin opseg pomnožimo s razlikom dobivenog arkuskosinus i udvostručene PI vrijednosti. Kada smo izračunali udaljenost, nadodajemo ju na zbroj ukupno prijeđene razdaljine i stare koordinate zamjenjujemo novima te ispisujemo novu ukupno prijeđenu razdaljinu [7] [8].

```
Button zaustavi = FindViewById<Button>(Resource.Id.stop);
zaustavi.Click += delegate
{
    a = 3;
```

Programski kod 28: Zaustavljanje mjerenja udaljenosti

Kako bi korisniku omogućili opciju pauziranja mjerenja gumbu stop pridružujemo događaj zaustavi u kojem varijabli a mijenjamo vrijednost u 3 i tako zaustavljamo petlju.

4.5.6. Spremanje rezultata trčanja

Kada je korisnik gotov s trčanjem i želi spremiti rezultat kako bi ga mogao kasnije usporediti s prijašnjim, kliknuti će na gumb “Spremi rezultat”. Kako bi mogli spremati rezultate trčanja kreirat ćemo klasu Rezultati_trcanja i na temelju klase bit će kreirana tablica u bazi.

```
12
13  namespace Aplikacija_Android
14  {
15      class Rezultati_trcanja
16      {
17          public string redni_broj{ get; set; }
18          public string rezultat_km { get; set; }
19
20
21          public Rezultati_trcanja(string broj, string rez)
22          {
23              redni_broj = broj;
24              rezultat_km = rez;
25
26
27          }
```

Programski kod 29: Klasa Rezultati_trcanja

U klasi ćemo definirati 2 varijable redni_broj i rezultat_km. U prvu varijablu ćemo stavljati redni broj rezultata kako bi kasnije sve rezultate mogli kronološki ispisati te tako omogućiti korisniku uvid u svoj napredak. Prikaz svih korisnikovih rezultata kasnije ćemo omogućiti u aktivnosti statistika koja će sadržavati kronološki poredane rezultate od svih korisnikovih vježbi i trčanja.

```
Button spremi = FindViewById<Button>(Resource.Id.spremi);
spremi.Click += delegate
{
    string zaokruženo = Math.Round(ukupnokm, 3).ToString();
    a = 3;
    try
    {
        var upit = db.Query<Rezultati_trcanja>("SELECT redni_broj FROM Rezultati_trcanja").LastOrDefault();
        int zadnji_broj = Int32.Parse(upit.redni_broj);
        zadnji_broj++;
        Rezultati_trcanja trenutni_rezultat = new Rezultati_trcanja(zadnji_broj.ToString(), zaokruženo.ToString());
        db.Insert(trenutni_rezultat);
    }
```

Programski kod 30: Spremanje rezultata trcanja

Kada smo kreirali klasu Rezultati_trcanja, gumbu spremi pridružujemo događaj .Click, te zaokružujemo ukupno prijeđenu razdaljinu koju je korisnik prešao na 3 decimale. Nakon toga kreiramo upit koji u tablici Rezultati_trcanja pronalazi zadnje uneseni redni broj, odnosno zadnji korisnikov rezultat. Zadnji broj uvećavamo za 1 i kreiramo varijablu tipa Rezultati_trcanja u koju spremamo postignuti rezultat i uvećani redni broj te ih umećemo u bazu. Gore spomenuti proces stavljamo u funkciju try ako je tablica prazna i nema zadnje unesenog rednog broja zbog čega bi se mogla dogoditi iznimka koja će blokirati cijelu aplikaciju.

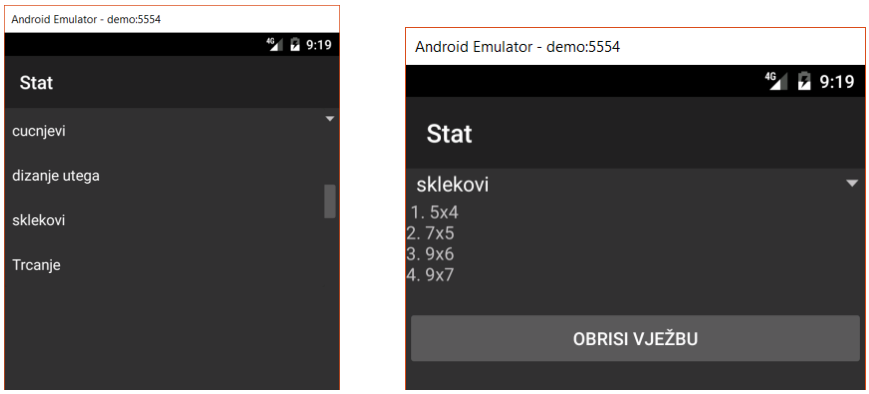
```
catch (Exception e)
{

    Rezultati_trcanja trenutni_rezultat = new Rezultati_trcanja(1.ToString(), zaokruženo);
    db.Insert(trenutni_rezultat);


}
```

Programski kod 31: Spremanje prvog rezultata

Ako se aktivira iznimka znači da moramo unijeti prvi rezultat odnosno da redni broj unesenog rezultata mora biti 1. Ako su uspješno izvršene aktivnosti definirane u događaju .Click, rezultat je spremljen u bazu.

4.5.6. Prikaz rezultata

Kako bi korisnik mogao pregledavati svoje rezultate i vidjeti napredak, kreirat ćemo sučelje koje će omogućiti da u padajućem izborniku odabere vježbu i da se pojave svi prethodno postignuti rezultati.



Slika 36: Prikaz prethodnih rezultata vježbe

Kako bi napunili izbornik s vježbama, spajamo se na bazu, kreiramo adapter pomoću kojeg povezujemo bazu i izbornik, potom kreiramo funkciju nap() u kojoj povezujemo izbornik i adapter i punimo izbornik s podacima.

```
SetContentView(Resource.Layout.Statistika);
string dbPat = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "MihaelTerten.d
var db = new SQLiteConnection(dbPat);
var iv = db.Table<vjezbe>().GroupBy(s => s.ime_vjezbe).Select(s => s.First());
var imenaV = iv.Select(s => s.ime_vjezbe).ToList();
imenaV.Add("Trcanje");
var adapter = new ArrayAdapter(this, Android.Resource.Layout.SimpleSpinnerItem, imenaV);
var spinerV = FindViewById<Spinner>(Resource.Id.odaberi);

void nap()
{
    spinerV.Adapter = adapter;
    adapter.SetDropDownViewResource
                (Android.Resource.Layout.SimpleSpinnerDropDownItem);


}
```

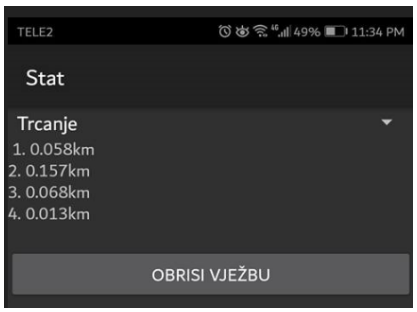
Programski kod 32: Punjenje izbornika

Kada korisnik odabere vježbu iz padajućeg izbornika, prvo se provjerava da li je izabrao trčanje jer su rezultati trčanja spremljeni u tablici „Rezultati_trcanja“.

```
prikazi_rezultate = new Thread(new ThreadStart(delegate
{
    try
    {
        TextView prikaziText = FindViewById<TextView>(Resource.Id.rezutati);
        spinerV.ItemSelected += (sender, args) =>
        {
            string vjezba = spinerV.SelectedItem.ToString();
            prikaziText.Text = " ";

            if (vjezba == "Trcanje")
            {
                var tablica = db.Table<Rezultati_trcanja>();

                foreach (var item in tablica)
                {
                        prikaziText.Text += item.redni_broj + ". " + item.rezultat_km + "km" +"\n";
                }
```

Programski kod 33: Dohvaćanje rezultata trčanja

Ako je korisnik izabrao trčanje, dohvaćaju se svi podaci spremljeni u tablici i prikazuju se kronološkim redom.



Slika 37: Prikaz rezultata trčanja

Ako je korisnik izabrao neku od vježbi, iz tablice rezultati dohvaćaju se kronološkim redom svi rezultati koje je korisnik unio za odabranu vježbu.

```
}
else
{
    var tablica = db.Table<rezultati>();

    foreach (var item in tablica)
    {
        if (item.ime_vjezbe == vjezba)
        {
            prikaziText.Text += item.redni_broj + ". " + item.postignuto_r1 + "x" + item.postignuto_r2 + "\n";
        }
```

Programski kod 34: Dohvaćanje rezultata vježbe

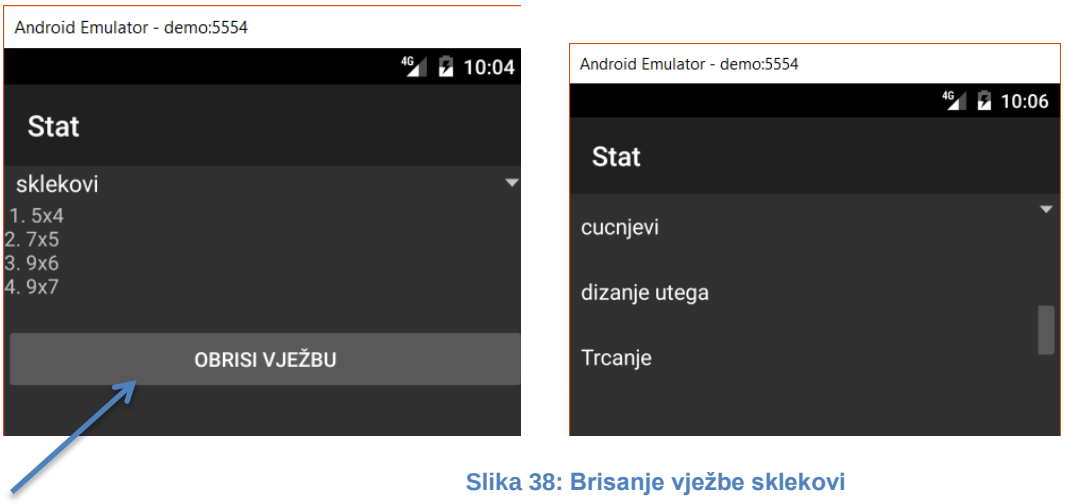
4.5.7. Brisanje vježbi

Korisniku je također pružena opcija brisanja vježbe i svih njezinih rezultata iz baze, ako korisnik doista želi izbrisati vježbu, kada ju odabere iz padajućeg izbornika, ispod prikazanih rezultata klikne na gumb „Obriši vježbu“.

```
Button obrisi = FindViewById<Button>(Resource.Id.brisi);
obrisi.Click += delegate
{



        try
        {
            string imeVjezbe = spinerV.SelectedItem.ToString();
            var upit_string_rez = db.Query<rezultati>("DELETE  FROM rezultati  WHERE ime_vjezbe == ?", imeVjezbe);
            var upit_string_vj = db.Query<vjezbe>("DELETE  FROM vjezbe  WHERE ime_vjezbe == ?", imeVjezbe);
            db.Delete(upit_string_rez);
            db.Delete(upit_string_vj);

        }
        catch (Exception e)
        {
            e.ToString();

        }



    nap();
```

Programski kod 35: Brisanje vježbe

Vježbu brišemo tako da u upite proslijedimo ime vježbe koje je odabrano iz izbornika. U prvom upitu brišemo sve rezultate iz tablice „rezultati“ u drugom brišemo samu vježbu iz tablice „vježbe“. Oba upita izvršavamo nad bazom. Kada je vježba i njezini rezultati izbrisani, osvježavamo padajući izbornik s funkcijom nap().

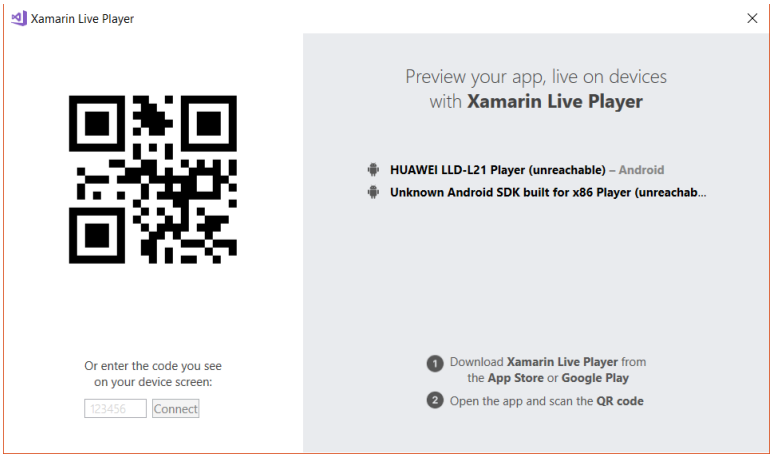


Slika 38: Brisanje vježbe sklekovi

Nakon što je korisnik kliknuo na gumb, vježba je nastala iz izbornika jer je izbrisana iz baze.

5. Zaključak

Nakon što sam izradila aplikaciju, mogu zaključiti da Xamarin ima i dobre i loše strane. Kao prvo jedna od boljih stvari(meni osobno) jest to da se u razvoju aplikacija koristi C#, jezik s kojim sam upoznata, također je pozitivno što je sama arhitektura projekta i princip izrade slična prijašnjim projektima u Visual Studio- u, moje je mišljenje da ako je korisnik prije radio s Visual Studio-m ili je koristio C# ne bi se trebao susretati s većim poteškoćama. Osobno tijekom projekta što se tiče izrade funkcionalnosti aplikacije nisam imala nekih problema je sam prije radila sa Windows.Forms što mi je uvelike olakšalo pristup radu i rješavanje zadataka. Što se tiče izrade Android aplikacija nemam puno prigovora osim što ponekad kod skidanja i instaliranja novih Nuget paketa javljaju se tehnički problemi kao naprimjer projekt ne registrira instalaciju pa ga treba očistiti, zatvoriti i ponovo pokrenuti u Visual Studio- u, osim kad se skine paket koji nije ispravan pa kod instalacija može uništiti cijeli projekt, zato preporučujem uvijek imati back- up. Izrada aplikacije za više platforma odjednom je vrlo dobro zamišljena, ali ne toliko dobro izvršena. Prva negativna stvar jest to da se korisnik mora imati ili Mac ili iPhone uređaj za bilo kakvu iOS aplikaciju, ali nije ni bilo za očekivati drugačije. Kod izrade višeplatformskih aplikacija, najbolje funkcionira Android dio, odmah prepoznaje promjene u glavnoj klasi i bez problema dohvaća kod namijenjen za dijeljenje, naime kod Windows i iOS platformi morala sam izbrisati referencu na glavnu klasu te ju ponovno dodati kako bi bile prepoznate nove metode. Kod skidanja Nuget paketa se stalno javljaju greške, ili je jdk krivo referenciran, ili se na jednoj od platforma počnu javljati greške koje se tiču nepoznatih i neprikladnih datoteka. Kako bi olakšali pokretanje na fizičke uređaje, Xamarin je predstavio Live player, aplikaciju koju korisnik instalira na uređaj I na računalo, kada ga pokrene u Visual Studio-u, program generira barkod kojeg korisnik skenira s mobitelom i zatim bi se trebao spojiti s Visual Studijom(preko WiFI-ja) [9].



Slika 39: Live player

Pokušala sam se spojiti s dva različita uređaja i na svakom bi se spojilo i nakon 5 sekundi prekinulo vezu, jednostavnije je samo ukopčati usb kabel, uglavnom poanta je da razvojni tim Xamarina ima dobre ideje koje su malo manje dobro izvedene [10]. Možda je razlog tome što se previse ljudi ne koristi u profesionalne svrhe sa Xamarinom nego radije bira Android Studio, Swift ili Xcode pa ima premalo korisnika koji prijavljuju greške koje bi se trebale ispraviti. Smatram da Xamarin nije trenutno najbolja opcija za profesionalnu izradu aplikacija ali za 5 ili 7 godina mislim da ne samo da će biti više korišten nego će sigurno biti u top 3 razvojna okruženja. Također bi svakom tko ima volje i vremena preporučila da se upozna sa Xamarinom.

6. Literatura

[1] Versluis G. A Brief History of Xamarin.”Xamarin.Forms Essentials”, izd. 1 , str. 1-12, 2017.

[2] Jim Bennet, Xamarin in Action “Creating native cross platform application”, izd. 1, str. 1-65, ruj.-2018

[3] Xamarin.Forms, “Notes for proffesionals””, izd. 1, str. 1-52, jan.-2018

[4] Mark McLemore, Tom Opgenorth, Craig Dunn, Brad Umbaugh "Xamarin.Android” kol. 2018. [Na internetu]. Dostupno: Xamarin, https://docs.microsoft.com/en-us/xamarin/android/ [pristupano 1.09.2018].

[5] Pierce Boggan "Geolocation for iOS, Android and Windows made easy” jan. 2016. [Na internetu]. Dostupno: https://blog.xamarin.com/ [pristupano 3.08.2018].

[6] James Montemagno “Xam.Plugin.Geolocator” svib.2018 Dostupno: https://www.nuget.org/

[7] Akshay Updhya “Harvesin Formula” lip. 2017 [Na internetu] Dostupno: https://www.igismap.com/haversine-formula-calculate-geographic-distance-earth/ [pristupano 1.9.2018]

[8] Frank Van Puffelen “How to find my distance to a known location in JavaScript” pros.2012 [Na internetu] https://stackoverflow.com/questions/13840516/how-to-find-my-distance-to-a-known-location-in-javascript [pristupano 1.9.2018]

[9] James Montemagno “Getting Started with the Xamarin Live player” Dostupno svib. 2017 [Na internetu] https://montemagno.com/xamarin-live-player-getting-started/ [pristupano 5.9.2018]

[10] Xamarin.Forums “ Live Player does not pair with Visual Studio” Dostupno pros. 2017 [Na internetu] “https://forums.xamarin.com/discussion/113731/live-player-does-not-pair-with-visual-studio-community”

7. Popis slika

8. Popis programskih kodova