

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Marta Fiorencis

**USPOREDBA RAZVOJA APLIKACIJA ZA
ANDROID POMOĆU JAVE I XAMARINA**

DIPLOMSKI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Marta Fiorencis

Matični broj: 42010/13–R

Studij: Informacijsko i programsko inženjerstvo

**USPOREDBA RAZVOJA APLIKACIJA ZA ANDROID POMOĆU
JAVE I XAMARINA**

DIPLOMSKI RAD

Mentor:

lv. prof.

Prof. dr. sc. Zlatko Stapić

Varaždin, veljača 2022.

Marta Fiorencis

Izjava o izvornosti

Izjavljujem da je moj 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.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

u sažetku staviti sve u prezent i reći što jest a ne što će biti

U diplomskom radu **biti će predstavljen** i uspoređen proces razvoja Android aplikacije pomoću programskih jezika Java i C# u razvojnim okruženjima Android Studio(Java) i Visual Studio(C#). Kako bi usporedba bila što objektivnija paralelno ću kreirati i uspoređivati korake razvoja aplikacije u oba razvojna okruženja, bitno je napomenuti da će obje aplikacije imati iste funkcionalnosti kako bi krajnji zaključak mogao biti što detaljniji. Kreirane aplikacije biti će pokrenute na dostupnim emulatorima i fizičkim mobilnim uređajima. Nakon što će svi koraci razvoja biti predstavljeni definirati ću kriterije usporedbe i usporediti korištena razvojna okruženja odnosno programske jezike.

Ključne riječi: Java, C#, Android aplikacija, Visual Studio, Android Studio, Xamarin

Sadržaj

popravliti numeraciju

Uvod.....	1
Metode i tehnike rada.....	2
1.1. Baza podataka(Web server).....	2
1.2. Android aplikacije.....	2
Android Studio.....	3
Xamarin.....	4
Projekt.....	4
1.3. Instalacija.....	4
1.3.1. Android Studio.....	4
1.3.2. Xamarin.....	4
1.4. Kreiranje projekta.....	4
1.4.1. Android Studio.....	5
1.4.2. Xamarin.....	6
1.5. Prvi dojam.....	7
1.5.1. Organizacija strukture projekta.....	7
1.5.1.1. Android Studio.....	8
1.5.1.2. Xamarin.....	8
1.5.2. Osnovne informacije projekta(AndroidManifest).....	8
1.5.2.1. Android Studio.....	8
1.5.2.2. Xamarin.....	9
1.5.3. Kod.....	9
1.5.3.1. Android Studio.....	9
1.5.3.2. Xamarin.....	9
1.5.4. Resursi.....	9
1.5.4.1. Android Studio i Xamarin.....	9
1.5.5. Pokretanje aplikacije.....	10
1.5.5.1. Android studio.....	10
1.5.5.2. Xamarin.....	10
1.5.6. Notifikacije i debugiranje.....	10
1.5.6.1. Android Studio.....	10
1.5.6.2. Xamarin.....	11
1.6. Svrha aplikacije.....	12
1.7. Funkcionalnosti aplikacije.....	13
1.8. Baza podataka.....	14
1.8.1. Pristup bazi podataka.....	15
1.9. Kreiranje sučelja aplikacije.....	16
1.9.1. Android Studio.....	16
1.9.2. Xamarin.....	17

1.10.	Dizajn sučelja	18
1.10.1.	Prijava	18
1.10.1.1.	Android Studio	18
1.10.1.2.	Xamarin	22
1.10.2.	Promjena fonta	26
1.10.2.1.	Android Studio	26
1.10.2.2.	Xamarin	27
1.10.3.	Kreiranje Dana.....	27
1.10.3.1.	Android studio.....	27
1.10.3.2.	Xamarin	29
1.10.4.	Izbornik.....	30
1.10.4.1.	Android Studio	31
1.10.4.2.	Xamarin	33
1.10.5.	Dodavanje klijenta	34
1.10.5.1.	Android Studio	34
1.10.5.2.	Xamarin	35
1.10.6.	Odabir datuma za kreiranje rasporeda.....	35
1.10.6.1.	Android Studio	35
1.10.6.2.	Xamarin	36
1.11.	Funkcionalnosti.....	37
1.11.1.	Prijava	37
1.11.1.1.	Android Studio	37
1.11.1.2.	Xamarin	39
1.11.2.	Odabir datuma.....	43
1.11.2.1.	Android Studio	43
1.11.2.2.	Xamarin	43
1.11.3.	Dodavanje korisnika	44
1.11.3.1.	Android Studio	44
1.11.3.2.	Xamarin	46
1.11.4.	Brisanje Korisnika	46
1.11.4.1.	Android Studio	47
1.11.4.2.	Xamarin	47
1.11.5.	Izračun i ispis prijeđenih kilometara	47
1.11.5.1.	Android Studio	48
1.11.5.2.	Xamarin	49
1.11.6.	Spremanje podataka u bazu	50
1.11.6.1.	Android Studio	51
1.11.6.2.	Xamarin	51
1.11.7.	Generiranje izvještaja	52

1.11.7.1. Android Studio	52
1.11.7.2. Xamarin	56
Usporedba.....	60
Zaključak	62
Popis literature	64
Popis slika	65
Popis tablica	66

Uvod

Provjeriti oblikovanje (pozicije, fontove, razmake, poravnanja i drugo) u odnosu na predložak za pisanje radova i upute koje su date studentima

Tema samog rada je usporedba razvoja aplikacije u razvojnom okruženju Android

Studio koristeći programski jezik Java i u razvojnom okruženju Visual Studio koristeći

Xamarin.Android alate i C# programski jezik. U radu biti će detaljno paralelno predstavljene svi koraci razvoja u oba slučaja kako bi čitatelj mogao što jasnije razaznati sličnosti i razlike.

Motivaciju za ovaj rad dobila sam kroz diplomski studij jer prije diplomskog studija nisam bila toliko upoznata sa programskim jezikom Java i izbjegavala sam Android Studio najviše zbog negativnih stvari koje sam čula ironično od ljudi koji ga također nisu koristili ili su ga koristili vrlo malo. Nakon što sam ga bila prisiljena koristiti tokom studija shvatila sam da su moji strahovi bili neopravdani i uživala sam u edukaciji, pogotovo jer sam bila zainteresirana za razvoj mobilnih aplikacija. Tema mog završnog rada bila je kreiranje mobilnih aplikacija u Xamarin-u i nakon malo istraživanja primijetila sam da je puno ljudi uspoređivalo Xamarin i Android Studio, samo što kod tih usporedbi nisu su se uspoređivali Xamarin.Android alati već Xamarin.Forms alati što po mojem mišljenju nije baš poštena usporedba jer Android Studio je namijenjen izradi android aplikacija, a Xamarin.Forms alati su namijenjeni izradi hibridnih aplikacija i nisu se previše spominjali Xamarin.Android alati, pogotovo ako je netko postavio pitanje na forumu o usporedbi Xamarin-a i Android Studio-a. Još jedna stvar koju sam primijetila je ukoliko nije riječ o stručnim člancima, knjigama ili osobama koje su podjednako iskusne u oba programska jezika većina presuda što je bolje koristiti temelji se na programskom jeziku sa kojim je netko upoznat te je drugi izbor jako popljuvan iako ga osoba nije nikad koristila. Jednostavnije rečeno nešto ne valja jer ne znam to koristiti i nemam namjera se educirati o tome niti ikad pokušati koristiti, ali svejedno ne valja. Također se u velikoj većini članaka/rasprava čija je tema usporedba Xamarin-a i Android Studio- a uzimaju u obzir tehničke stvari razvojnih okruženja, a ne toliko osobno iskustvo u kreiranju aplikacija. Upravo zbog toga ovaj rad biti će fokusiran na paralelno kreiranje dvije android aplikacije te će na kraju biti izvršena usporedba sa kriterijima kao što su dostupna dokumentacija, emulatori, brzina pokretanja aplikacije, veličina aplikacije, pojava i rješavanje grešaka kroz proces razvoja. Svrha samog rada je ukoliko se netko ne bude mogao odlučiti što koristiti za razvoj android aplikacije, odnosno zanima ga kako izgleda proces razvoja iste aplikacije na dva različita načina iz perspektive nekog kome programski jezik i razvojno okruženje nisu ključni kriteriji odabira, moći će pročitati ovaj rad(ili samo dijelove koji ga zanimaju) te si barem malo olakšati odluku.

Dodati dispoziciju rada, to jest opisati poglavlja i način kako će biti postignuti ispred zadani cilj.

Metode i tehnike rada

1.1. Baza podataka(Web server)

- Apache
- PHP 5.6.27
- MySQL
- FileZilla(za prijenos podataka na server)

Raspisati.

Osim tehnologija, potrebno je navesti i metode koje su primijenjene u radu (npr. istraživanje literature, usporedba analiza, sinteza gradiva iz različitih izvora, možda neka statistika i slično...). Sve pojasniti.

1.2. Android aplikacije

- Java
- C#
- XML
- Visual Studio 2019
- Android Studio 3.2.1

Android Studio

Android studio je razvojno okruženje(IDE) koje se koristi za razvoj Android aplikacija. Kada upišite u google pojam „Android Studio“ i kliknete na prvu poveznicu otvara se službena stranica te prva rečenica koju korisnik vidi jest „Android Studio pruža najbrže alate za izradu aplikacija za sve vrste Android uređaja.“*[1]. Što više dodati takvoj definiciji. Povijest Android Studio započinje u prosincu 2014. godine kada je službeno izdana prva verzija. U sljedećoj tablici su prikazane sve daljnje verzije te koje su godine izdane[1].

Verzija	Godina
Arctic Fox	2021.
4.2	2021.
4.1	2020.
4.0	2020.
3.6	2020.
3.5	2019.
3.4	2019.
3.3	2019.
3.2	2018.
3.1	2018.
3.0	2017.
2.3	2017.
2.2	2016.
2.1	2016.
2.0	2016.
1.5	2015.
1.4	2015.
1.3	2015.
1.2	2015.
1.1	2015.
1.0	2014.

Tablici dodati naslov u skladu s pravilima za oblikovanje rada.

Nisu dodane posljednje BB verzije.

Jedna od stvari što je pridonijela njegovoj popularnosti jest to što je besplatan(najnovija verzija se može skinuti sa službene stranice <https://developer.android.com/studio>) ali i učinkovit. Također je dostupan za više operacijskih sustava kao što su Windows, Linux i Mac OS. Kako bi mogli razvijati aplikacije potrebno je instalirati i JDK(Java Development Kit) iliti alate pomoću kojih možemo razvijati, testirati i pokretati programe napisane u Javi. Zbog toga Java i Kotlin su programski jezici pomoću kojih razvijamo aplikacije u Android Studio-u.

O tome koji je bolji može se napisati još jedan diplomski rad. [Dopuniti/preformulirati](#)

Ovdje imamo problem sa svrhom poglavlja. Dakle, ili dopuniti poglavlje i onda prikazati/sagledati android studio detaljnije, ili ovaj tekst smjestiti u neko šire poglavlje, koje možda opisuje Android Studio i alternativnu platformu koja će biti obrađivana u radu.

Xamarin

Xamarin-ova povijest započinje u svibnju 2011. i prvotno je bila samostalna tvrtka dok je 2016. nije otkupio Microsoft[4]. Xamarin instalacija se izvršava pomoću razvojnog okruženja Visual Studio te je njegov smisao bio pružiti razvojno okruženje i alate za razvoj više platformskih (iOS, Android i Windows) aplikacija pomoću .NET, u tom slučaju korisnici koriste Xamarin.Forms alate, ali pošto hibridne aplikacije nisu tema ovog rada, dio Xamarin-a koji je bitan su Xamarin.Android alati koji koriste Android SDK (sadrži biblioteke i alate potrebne za razvoj Android aplikacija) i koriste .NET (u nastavku rada pojam Xamarin će se odnositi na Xamarin.Android alate koje ćemo koristiti u Visual Studio-u). Posljednja verzija je Xamarin.Android 12.1, a uskoro će biti i spremna 12.2 verzija[4].

Ono što je bitno napomenuti kod razvoja Android aplikacija pomoću Xamarin.Android alata jest da je programski kod napisan u C# programskom jeziku te tako omogućuje programerima koji nisu upoznati sa Java-om razvijanje mobilnih aplikacija.

I ovdje imamo problem sa svrhom poglavlja. Dakle, ili dopuniti poglavlje i onda prikazati/sagledati xamarin detaljnije, ili ovaj tekst smjestiti u neko šire poglavlje.

Projekt

Numeracija. Dodati uvod u poglavlje. Dodati tekst koji lijepi poglavlje s prethodnim poglavljima. Dodati tekst koji daje uvod u to što će biti napravljeno u ovom poglavlju.

1.3. Instalacija

Ovaj stil pisanja na žalost nije dobar za završni/diplomski rad. Poglavlje mora biti tekstualno, a u nekom dijelu se može napraviti neko pobrojenje, međutim cijelo poglavlje ne može biti pobrojenje. Jeste li razmišljali da ove stvari stavite kao Appendix u rad?

1.3.1. Android Studio

1. Skinite datoteku(.zip) koja sadrži željenu verziju sa službene stranice
2. Otpakirajte datoteku
3. Otvorite bin datoteku i pokrenite instalaciju(kliknite na studio64.exe ili studio.exe)
4. Slijedite upute čarobnjaka te instalirajte sve potrebne SDK pakete

1.3.2. Xamarin

1. Skinite datoteku koja sadrži željenu verziju Visual Studio-a sa službene stranice
2. Kliknite na skinuti datoteku i pokrenite instalaciju
3. Nakon što se instalacija pokrenula obavezno odaberite Mobile development with .NET na početnom ekranu i kliknite na install gumb

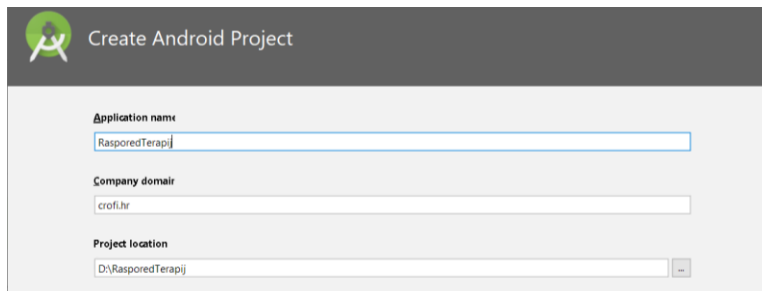
1.4. Kreiranje projekta

Isti komentari i za cijelo ovo poglavlje. Dakle, sve do 1.5 treba ili ponovno raspisati ili prebaciti u Appendix....

Nakon instalacije možemo kreirati projekt.

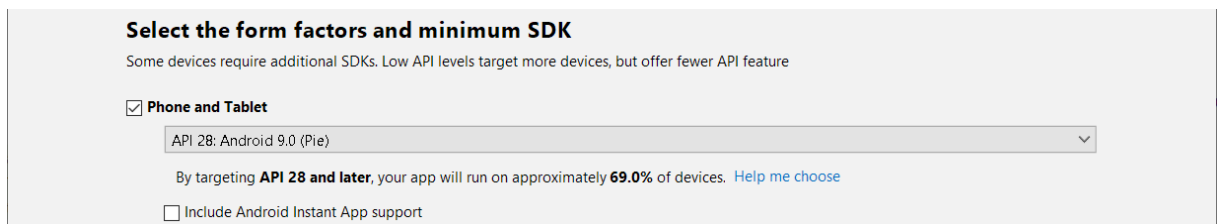
1.4.1.Android Studio

1. Odaberite File ->New->New Project te će vam se otvoriti sljedeći prozor



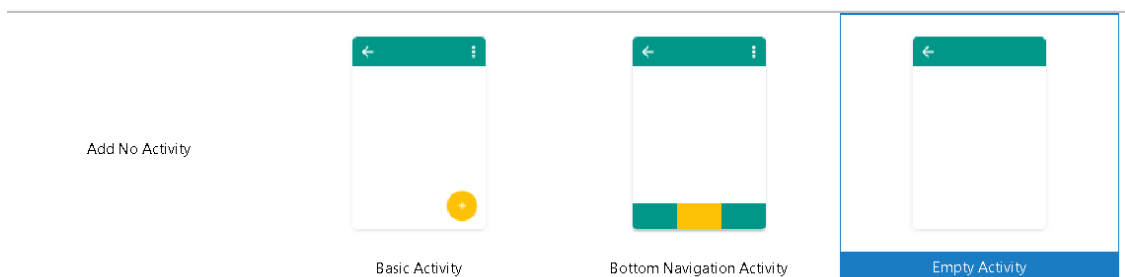
Unesite Ime aplikacije, ime paketa te lokaciju gdje će projekt biti spremljen, nakon što su podaci uneseni kliknite gumb Next

2. Otvoriti će vam se prozor u kojem odabirete uređaje za koje je aplikacija namijenjena te minimalni SDK



Nakon odabira kliknite na gumb Next

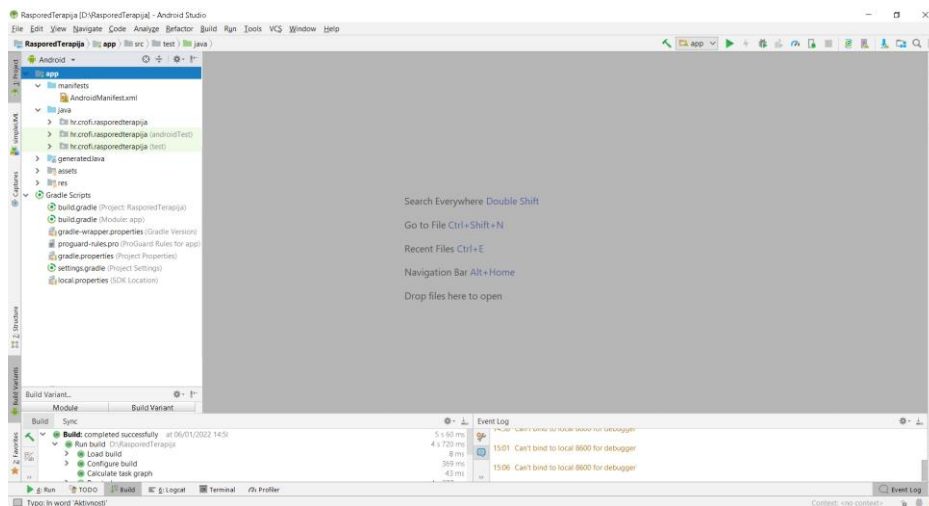
3. Potom će vam se otvoriti izbor početnih aktivnosti koje možete dodati mobilnoj aplikaciji



Kliknite na željenu aktivnost i Kliknite na gumb Next

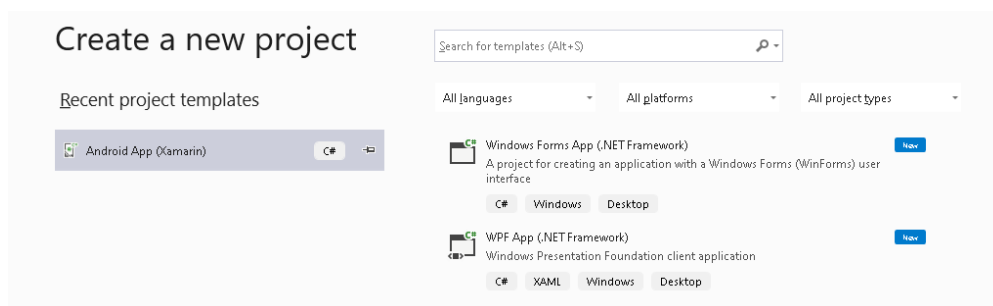
4. Ukoliko ste zadovoljni sa postavkama kliknite Finish i projekt će se generirati

Početno sučelje:



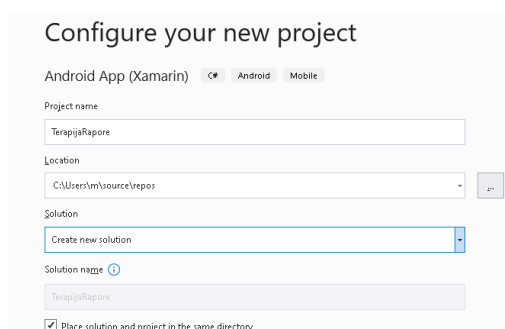
1.4.2.Xamarin

1. Odaberite File -> New Project te će vam se otvoriti sljedeći prozor



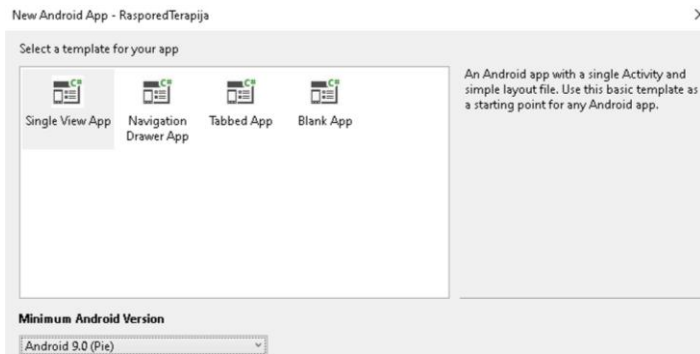
U prozoru se nalazi izbornik koji sadrži različite vrste aplikacija, u ovom slučaju odaberite ili napišite u tražilicu Android App(Xamarin) te nakon što ste izabrali kliknite na gumb Next.

2. U sljedećem prozoru definira se ime projekta, lokacija gdje će projekt biti spremljen te kontejner(Solution) projekta, ukoliko imate više povezanih projekata možete ih staviti u isti kontejner.



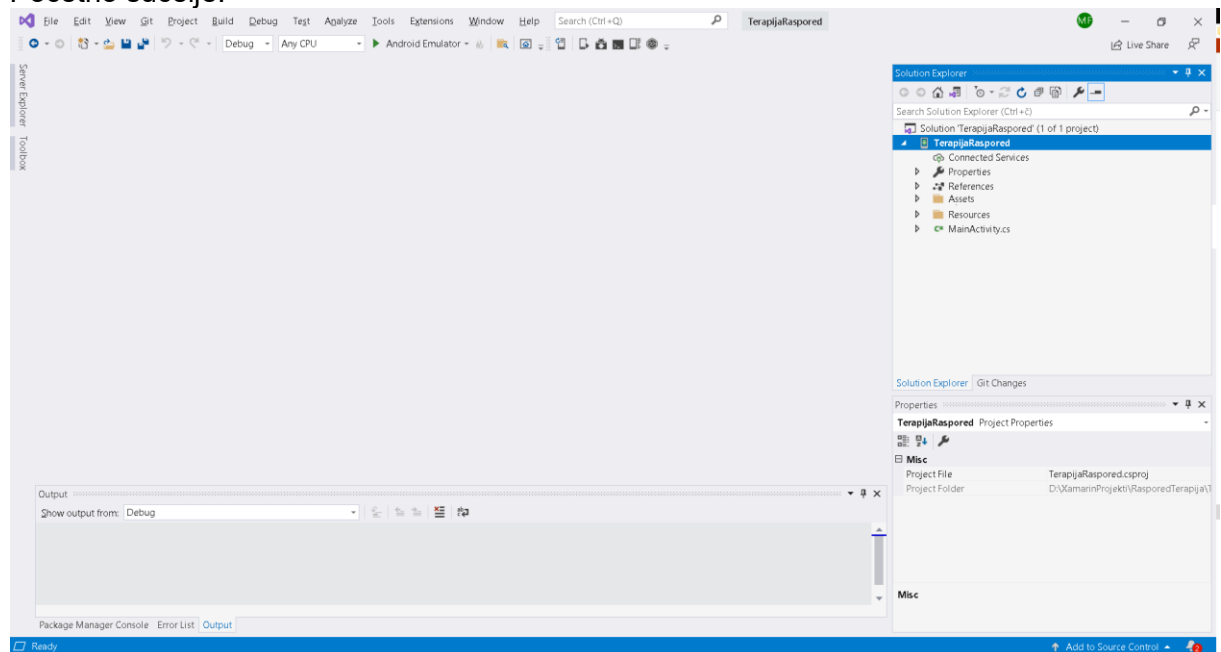
Nakon što su sve potrebne informacije unesene kliknite na gumb Create.

3. Prije nego što se projekt generira otvara se prozor u kojem se izabire početna aktivnost aplikacije i minimalna Android verzija.



Kada ste zadovoljni sa izborom kliknite na gumb ok i projekt će se generirati

Početno sučelje:



Što se tiče teksta u prethodnom poglavlju, jednostavno ga treba napisati u akademskom stilu. Nabrojavanja koraka i pobrojenja aktivnosti, bez tijela poglavlja ne mogu proći :(

1.5. Prvi dojam

Prije nisam uspio navesti jer mi komentar nije stao, ali ovo poglavlje, bez obzira što je sada pogrešno označeno kao 1., čak i kada dobije točnu oznaku, ne može biti ovako dugačko. više od 10 podpoglavlja je previše. Mislim da se ovo poglavlje mora malo restrukturirati.

Na prvi pogled Android Studio i Xamarin vrlo su slični, tako da ukoliko ste koristili jednog od njih neće vam biti problem snaći se i u drugom. U nastavku su objašnjene neke od osnovnih komponenata razvojnih okruženja i koje sličnosti odnosno razlike bi korisnici možda prvo zamijetili.

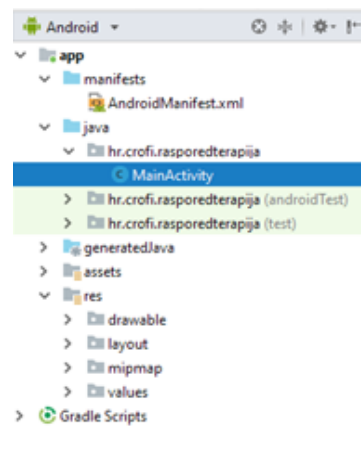
1.5.1. Organizacija strukture projekta

U tekstu poglavlja, struktura nije pojašnjena.

Kao što vidimo na slikama strukturu projekta u oba razvojna okruženja možemo pretraživati u posebnom prozoru.

1.5.1.1. Android Studio

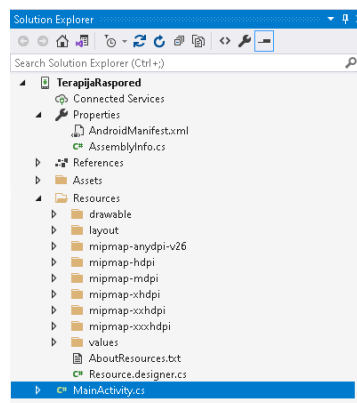
Umjesto podnaslova, staviti dvije slike jednu do druge, te slikama dodati caption.



VAŽNO!

Svaka slika/tablica/grafika mora imati caption, te se u tekstu mora spomenuti neposredno prije nego je slika/tablica/grafika prikazana...

1.5.1.2. Xamarin



1.5.2. Osnovne informacije projekta(AndroidManifest)

Android projekt mora sadržavati datoteku u kojoj će biti definiran ime projekta, verziju, aktivnosti i dopuštenja potrebna kako bi aplikacija uspješno radila(dopuštenje za korištenje Interneta, čitanje memorije, itd..).

Definitivno ove stvari ne bih stavljao u podnaslove

U prilog mi ide i činjenica da imate dva podnaslova, ali onda slijedi tekst (zaključak) koji zapravo nije dio ni jednog od tih poglavlja. Ovaj komentar se odnosi dolje ispod i na sva ostala slična poglavlja koja slijede. Ne treba stavljati podpoglavlja za Android i

1.5.2.1. Android Studio

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

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:requestLegacyExternalStorage="true"
        android:theme="@style/AppTheme">
    </application>
</manifest>
```

Android studio kod kreiranja projekta automatski generira AndroidManifest.xml u datoteci manifests.

1.5.2.2. Xamarin

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    package="com.companyname.terapijaspored">
    <uses-sdk android:minSdkVersion="28" android:targetSdkVersion="30" />
    <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">
    </application>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
</manifest>
```

Kod Xamarin-a ona se također automatski generira kod kreiranja projekta i nalazi se u datoteci Properties, te se ne razlikuje od generirane Android Studio verzije.

Kako sam napredovala u izradi aplikacije došla sam do zaključka da ukoliko je aplikacija izrađena po istoj strukturi, koristi iste servise i zahtjeva ista dopuštenja AndroidManifest.xml trebao bi biti sličnog sadržaja u oba razvojna okruženja.

1.5.3. Kod

Programski kod koji upravlja aplikacijom i njezin je najvažniji dio (upravljanje aktivnostima, dohvaćanje podataka, itd..)

1.5.3.1. Android Studio

U datoteci java generirana je datoteka sa imenom projekta u kojoj se piše programski kod aplikacije, korisnik ga može organizirati u više raznih paketa te tako osigurati jasnije vidljivu strukturu.

1.5.3.2. Xamarin

Programski kod je spremljen u datoteku pod nazivom projekta te se također može strukturirati u više raznih paketa.

Iako su programski jezici različiti, programski kod ima isti princip rada kao na primjer sučelja su vezana uz klase aktivnosti, iz aktivnosti se pokreće sučelje te ukoliko se dogodi značajna promjena za aplikaciju na sučelju aktivnost može sukladno promjeni reagirati (dohvatiti podatke, pokrenuti dretvu, promijeniti nešto na sučelju, itd..).

1.5.4. Resursi

U resurse aplikacije ubrajamo slike i ikone, bitmape (npr. fontovi), dizajn sučelja te bilo kakve animacije vezane uz dizajn, definiciju boja i string-ova ili bilo koje dodatne datoteke.

1.5.4.1. Android Studio i Xamarin

Kod kreiranja projekta generira se datoteka res u kojoj se nalaze sljedeće datoteke:

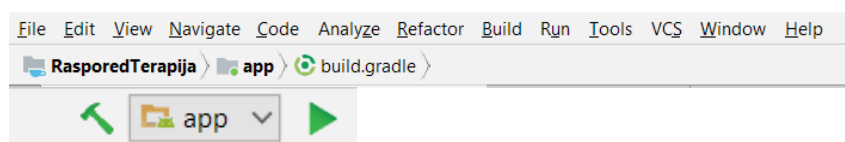
- drawable sadrži slike, ikone ili dijelovi dizajna koji se koriste u sučeljima(gumbi, tekstualna polja, izbornici, itd.)
- layout sadrži XML datoteke sučelja aplikacije
- mipmap sadrži ikonu koja će predstavljati aplikaciju na mobilnom uređaju korisnika te ostale ikone koje se koriste kod pokretanja aplikacije, jedina je razlika što Xamarin automatski generira više mipmap datoteka od kojih svaka predstavlja drugačiju gustoću piksela ikona
- values sadrži definirane boje i string-ove koje se koriste u sučeljima aplikacije

Kao što vidimo organizacijska struktura nema puno razlika ali postoji jedna bitna razlika, Android studio koristi Gradle sustav za kompajliranje pomoću kojeg uključujemo vanjske biblioteke. Pošto je Gradle sustav baziran na JVM (Java Virtual Machine) sustav u Xamarin-u se ne koristi, već većinu vanjskih biblioteka uključujemo pomoću NuGet alata(2).

1.5.5. Pokretanje aplikacije

Ukoliko je potrebno prije pokretanja projekt je potrebno očistiti i ponovno kompajlirati, u oba razvojna okruženja to činimo klikom na Build u izborniku te odaberemo Clean te potom Build. Nakon toga kliknemo na zeleni gumb za pokretanje te možemo izabrati emulator ili stvarni mobilni uređaj na kojem želimo pokrenuti aplikaciju.

1.5.5.1. Android studio



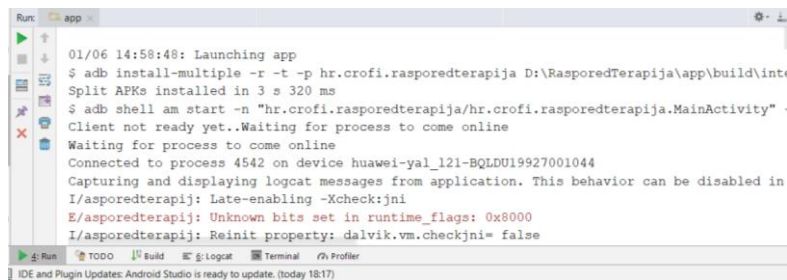
1.5.5.2. Xamarin



1.5.6. Notifikacije i debugiranje

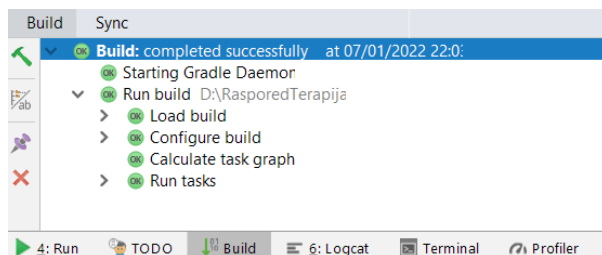
1.5.6.1. Android Studio

U donjem dijelu sučelja nalaze se razni alati koji olakšavaju praćenje i debugiranje razvoja aplikacije. Ukoliko kliknete na Run ili Debug(ovisi o postavkama), otvara se prozor u kojem će te vidjeti sve zapise sustava tokom pokretanja i rada aplikacije.

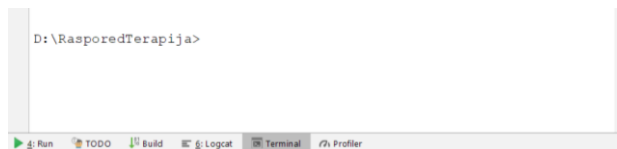


```
01/06 14:58:48: Launching app
$ adb install-multiple -r -t -p hr.crofi.rasporedterapija D:\RasporedTerapija\app\build\inter
Split APKs installed in 3 s 320 ms
$ adb shell am start -n "hr.crofi.rasporedterapija/hr.crofi.rasporedterapija.MainActivity" -
Client not ready yet..Waiting for process to come online
Waiting for process to come online
Connected to process 4542 on device huawei-yal_121-BQLDU19927001044
Capturing and displaying logcat messages from application. This behavior can be disabled in
I/asporedterapij: Late-enabling -Xcheck:jni
E/asporedterapij: Unknown bits set in runtime_flags: 0x8000
I/asporedterapij: Reinit property: dalvik.vm.checkjni= false
```

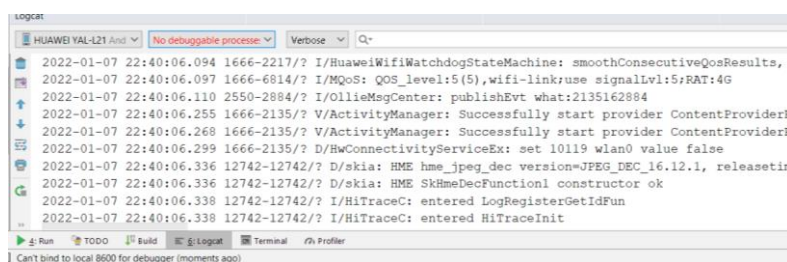
Također klikom na Build možete vidjeti kako je prošao proces kompajliranja, ukoliko postoji greška kod Gradle dijela projekta(npr. nedostaje dopuštenje ili referenca vanjsku biblioteku) ovdje će se pojaviti obavijest.



Klikom na Terminal otvara se konzolni pristup projektu, odnosno ukoliko želite dodati neki vanjsku biblioteku ili napraviti update postojećih preporučam iz osobnog iskustva da napravite to ovdje.



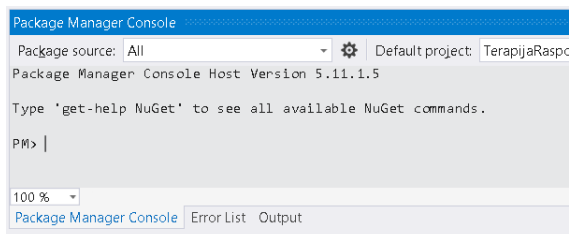
Logcat pruža uvid korisniku u tok i potrebno vrijeme procesa koji se obavljaju tokom pokretanja aplikacije.



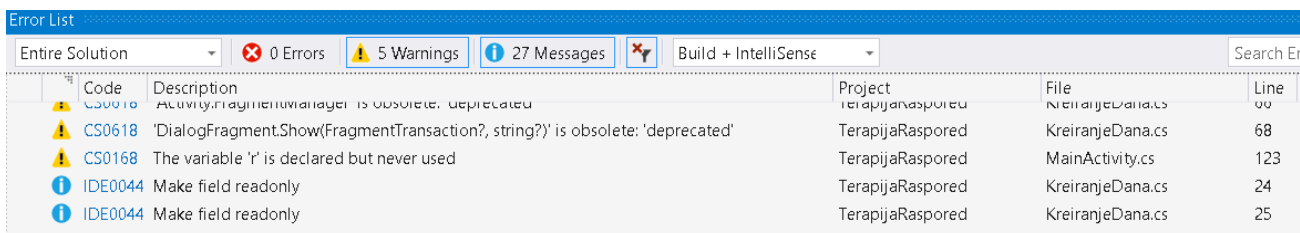
1.5.6.2. Xamarin

Xamarin pošto je dio Visual Studio-a, nudi iste alate za praćenje razvoja aplikacije, koji također olakšavaju otkrivanje i popravljjanje greška. U nastavku su spomenuti neki od najbitnijih.

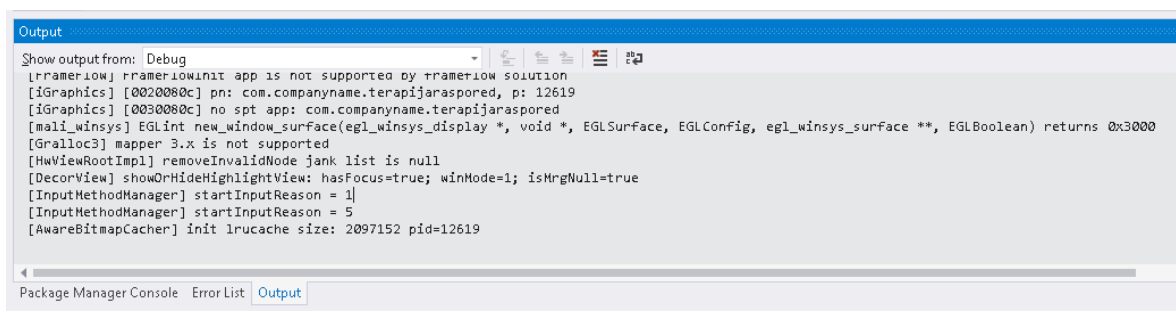
Klikom na Packager Manager Console otvara se prozor u kojem možemo ručno upravljati sa paketima i bibliotekama u projektu, ovo je dobar pristup ako imate problem sa uvozom nekog paketa ili neki paket stalno javlja grešku(najčešće su paketi vezani uz dizajn). Također moram napomenuti da većinu grešaka rješava naredba „dotnet restore“ pomoću koje možemo reinstalirati sve pakete dodane u projekt.



Sljedeći prozor kojem možemo pristupiti je “Error List” u njemu će se ispisivati sve pogreške, upozorenja ili poruke projekta. Korisnik je u mogućnosti regulirati obavijesti klikom na vrstu obavijesti koje želi prikazivati, naprimjer ako želi prikazivati samo pogreške može isključiti poruke i upozorenja.



Output prozor prikazuje sve zapise sustava tokom pokretanja i rada aplikacije(na emulatoru ili priključenom uređaju).



1.6. Svrha aplikacije

Kako bi objektivno mogla usporediti Android Studio i Xamarin izraditi ću istu aplikaciju sa istim funkcionalnostima i istim dizajnom pomoću oba alata, mislim da će se kroz proces izrade najbolje moći usporediti kvaliteta ovih alata.

[illegible]

1.7. Funkcionalnosti aplikacije

1. Logiranje - zaposlenici se logiraju pomoću dodijeljenog korisničkog imena i lozinke

- Datum dana
- Popis klijenata koje su obišli – na temelju popisa izračunava se broj prijeđenih kilometara

- Početno stanje brojila vozila
 - Završno stanje brojila vozila
3. Spremanje promjena
 4. Odabir datuma za generiranje izvještaja
 5. Generiranje izvještaja za odabrane datume u .pdf formatu

Ono što moram napomenuti jest da zaposlenici ne žele da ih se prati putem lokacije mobitela već da se izračuna broj prijeđenih kilometara na temelju unesenih klijenata, također razlika između početnog i završnog stanja brojila nije uvijek ista prijeđenim kilometrima, ali tvrtka želi da broj prijeđenih kilometara za dan predstavlja udaljenost između lokacije klijenata koji se nalaze u popisu za taj datum. Detalji svake funkcionalnosti biti će specificirani tokom razvoja.

1.8. Baza podataka

[Dodati model podataka.](#)

[Dodati tablice za opis pojedinih atributa, ako je takav opis potreban.](#)

Baza podataka koju će koristiti aplikacija nalazi se na serveru netdomena.com te se sastoji od tri tablice:

- ZaposleniciAplikacija – sadrži podatke o zaposlenicima koji će koristiti aplikaciju
 - Id(int, AUTO_INCREMENT) - id zaposlenika
 - Ime(varchar(300)) - ime zaposlenika
 - Prezime(varchar(300)) - prezime zaposlenika
 - Korime(varchar(300)) – korisničko ime kojim će se zaposlenik logirati
 - Lozinka(varchar(300)) – lozinka kojom će se korisnik logirati
 - Auto(varchar(300)) – tip automobila koji zaposlenik vozi
 - Registracija(varchar(300)) – registracija automobila zaposlenika
- KlijentiAplikacija – sadrži podatke o klijentima koje zaposlenici obilaze
 - Id(int, AUTO_INCREMENT) – id klijenta
 - Ime(varchar(300)) - ime klijenta
 - Prezime(varchar(300)) - prezime klijenta
 - Adresa(varchar(300)) – adresa klijenta
 - Lat(varchar(300)) – zemljopisna širina koordinata adrese klijenta
 - Lon(varchar(300)) - zemljopisna dužina koordinata adrese klijenta
- DaniAplikacija – sadrži podatke o danima
 - Id(int, AUTO_INCREMENT) – id dana
 - Datum(varchar(300)) - datum dana

- Poc(varchar(300)) – početno stanje brojila automobila
- Zav(varchar(300)) – završno stanje brojila automobila
- Pkm(varchar(300)) – prijeđeni kilometri
- Popis(varchar(300)) – popis običenih klijenata u danu(id klijenata)
- Mjesec(int) – mjesec dana
- Godina(int) – godina dana
- Zaposlenik(int) – id zaposlenika koji je kreirao zapis

1.8.1. Pristup bazi podataka

Aplikacije će pristupati bazi podataka pomoću php skripti koje se također nalaz na serveru.

- **Logiranje.php** – prima korisničko ime i lozinku te provjerava postoji li zaposlenik u tablici ZaposleniciAplikacija čije korisničko ime i lozinka odgovara primljenim. Ukoliko korisnik postoji vraća zaposlenikov Id.

```
$con->set_charset("utf8");
if (mysqli_connect_errno($con)) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
$username = $_GET['username'];
$password = $_GET['password'];
$result = mysqli_query($con,"SELECT * FROM ZaposleniciAplikacija WHERE Korime='$username' and Lozinka='$password'");

if($result->num_rows > 0){
    while($res = $result->fetch_assoc()) {
        echo $res["id"];
    }
}
else{
    echo "No";
}
mysqli_close($con);
```

programski kodovi moraju biti imenovani i spomenuti u tekstu kao svaka druga slika i tablica. Na kraju dokumenta se dodaje popis programskih kodova, kao i popis slika i tablica.

- **Sviklijenti.php** – vraća sve podatke klijenata unesenih u tablicu KlijentiAplikacija

```
$con->set_charset("utf8");
if (mysqli_connect_errno($con)) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
$username = $_GET['username'];
$result = mysqli_query($con,"SELECT * FROM KlijentiAplikacija");
if($result->num_rows > 0){
    while($res = $result->fetch_assoc()) {
        $red1=$res["Ime"];
        $red2=$res["Prezime"];
        $red3=$res["Adresa"];
        echo $res["id"]. "##" . $res["Ime"]. "##"
        . $res["Prezime"]. "##" . $res["Adresa"]. "##" . $res["Lat"]. "##" . $res["Lon"]. "///";
    }
}
mysqli_close($con);
```

- **SviDani.php** – prima id prijavljenog zaposlenika i vraća sve podatke dana koje je unio zaposlenik iz tablice DaniAplikacije

```
$con->set_charset("utf8");
if (mysqli_connect_errno($con)) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
$username = $_GET['id'];
$result = mysqli_query($con, "SELECT * FROM DaniApikacija WHERE Zaposlenik='$username'");

if($result->num_rows > 0){
    while($res = $result->fetch_assoc()) {
        echo $res["id"]. "##" . $res["Datum"]. "##" . $res["Poc"]. "##" . $res["Zav"]. "##"
            . $res["Pkm"]. "##" . $res["Popis"]. "##" . $res["Mjesec"]. "##" . $res["Godina"]. "///";
    }
}

mysqli_close($con);
```

- UpdateDan – prima sve podatke o danu iz tablice i Id zaposlenika koji promjenio podatke zatim u tablici DaniApikacije te briše zapis čiji podatuci o datumu i zaposleniku odgovaraju primljenim podacima, potom kreira novi zapis.

```
$con->set_charset("utf8");
if (mysqli_connect_errno($con)) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
$Datum = $_GET['Datum'];
$Poc = $_GET['Poc'];
$Zav = $_GET['Zav'];
$Pkm = $_GET['Pkm'];
$Popis = $_GET['Popis'];
$Mjesec = $_GET['Mjesec'];
$Godina = $_GET['Godina'];
$Zaposlenik = $_GET['Zaposlenik'];
$brisanje = "DELETE FROM DaniApikacija WHERE Datum='$Datum' AND Zaposlenik=$Zaposlenik";
$umetanje = "INSERT INTO DaniApikacija (Datum, Poc, Zav, Pkm, Popis, Mjesec, Godina, Zaposlenik)
VALUES ('$Datum', '$Poc', '$Zav', '$Pkm', '$Popis', '$Mjesec', '$Godina', $Zaposlenik)";
if (mysqli_query($con, $brisanje)) {
    if (mysqli_query($con, $umetanje)) {
        echo "Uspjeh";
    }
}
mysqli_close($con);
```

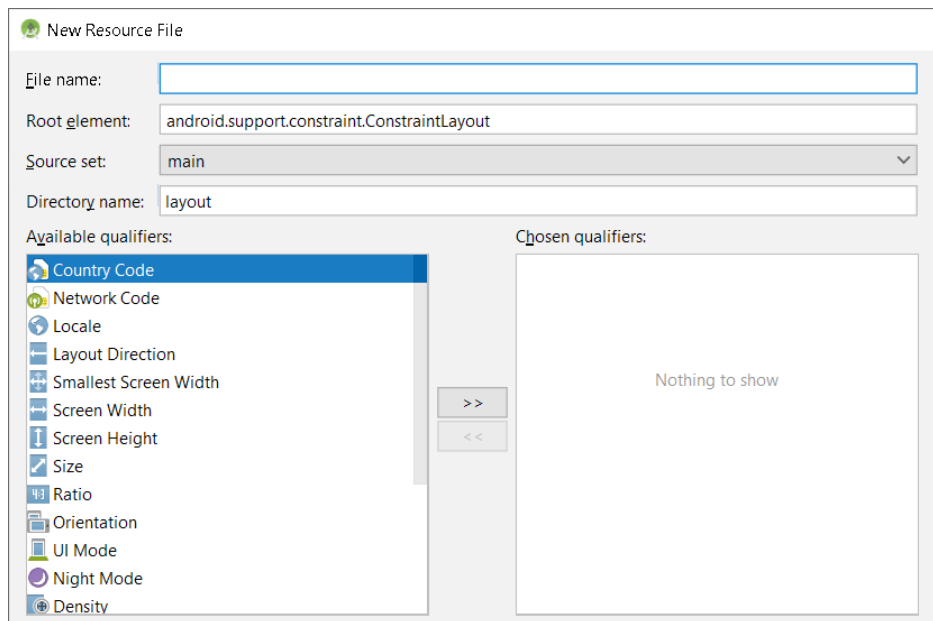
1.9. Kreiranje sučelja aplikacije

Kao što je bilo spomenuto u poglavlju o strukturi projekta u oba razvojna okruženja sučelja mogu biti povezana sa aktivnostima. Ono što je bitno zapamtiti kod dizajna bilokakvih sučelja za mobilne aplikacije jest to da korisnik može mobilni uređaj držati vertikalno i horizontalno. Zbog toga ukoliko sučelje ne sadrži samo jedan element(npr. pozdravnu poruku) preporuča se kreirati dizajn sučelja za obje verzije.

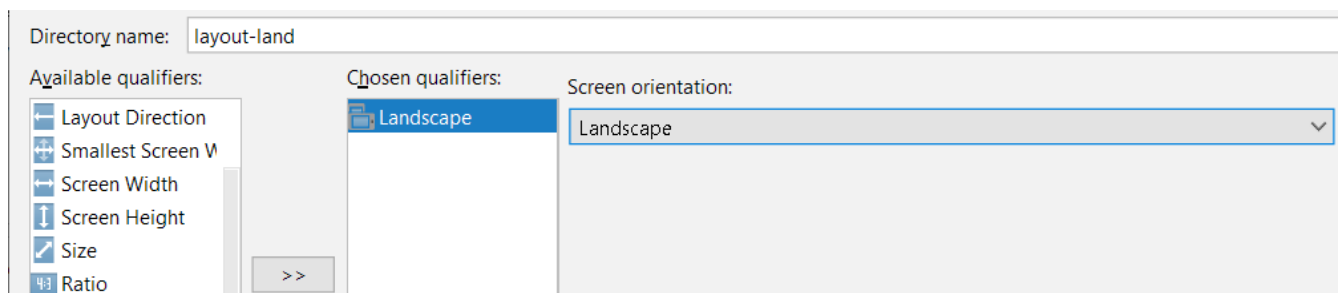
1.9.1. Android Studio

Kako bi kreirali novo korisničko sučelje u prozoru gdje je prikazana struktura projekta desnim klikom kliknemo na datoteku /res/layout i odaberemo New-> Layout resource file. Nakon toga otvara nam se sljedeći prozor.

Ovaj paragraf je dobar primjer kako možemo pisati upute za nešto. Međutim, ovo što su naredbe bi trebalo nekako istaknuti. Možda italic?



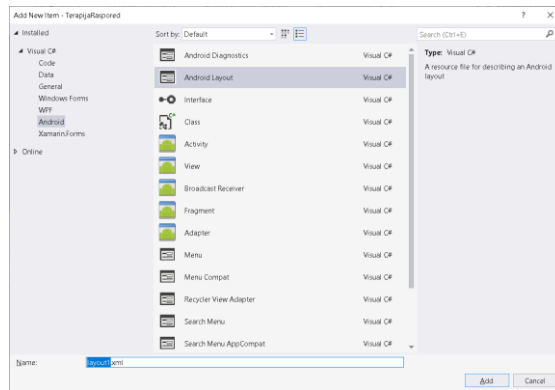
Upišite ime sučelja, odaberite kakav tip dizajna želite koristiti, iz osobnog iskustva preporučam RelativeLayout, pogotovo ako niste baš vješti sa xml-om, ali ukoliko sučelje nema previše elemenata, možete ostaviti i ConstraintLayout. Sva sučelja u ovoj aplikaciji koristiti će RelativeLayout kao bazni element dizajna. Zatim odaberite uz koju aktivnost će sučelje biti vezano i ostavite ime datoteke gdje će sučelje biti spremljeno. Uzmite u obzir da razvojno okruženje kod kreiranja sučelja automatski kreira vertikalnu verziju dizajna, tako da ako kreirate horizontalnu iz izbornika opcija odaberite Orientation i iz izbornik Screen orientation odaberite Landscape. Ime datototeke će se promjeniti u layout-land, ostavite ga tako, a ostale podatke ispunite isto kao što ste i za vertikalnu verziju.



Kada ste zadovoljni postavkama kliknite gumb Ok i sučelje će se kreirati.

1.9.2.Xamarin

Za kreiranje novog sučelje desnim klikom klinite na datoteku /Resources/layout i odaberite Add->New Item. Nakon toga otvoriti će vam se prozor sa izbornikom iz kojeg odaberete Android Layout, upišete ime sučelja i kliknete gumb Add.



Kao i Android Studio Xamarin će automatski kreirati vertikalnu verziju dizajna i kao bazni element postaviti će LinearLayout, također preporučam da ga promijenite u RelativeLayout. Ukoliko želite kreirati horizontalne verzije dizajna, prvo morate ručno kreirati datoteku layout-land u datoteci Resource i u toj datoteci kreirati sučelje(isti postupak kao i za vertikalnu verziju).

1.10. Dizajn sučelja

1.10.1. Prijava

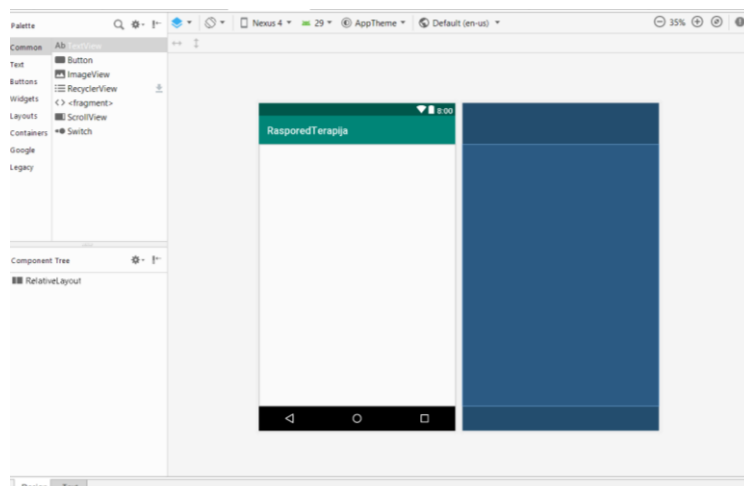
Početno sučelje aplikacije izvršavati će funkcionalnost prijave korisnika, kako bi uspješno funkcioniralo sastojati će se od sljedećih elemenata

- 2 elementa za prikaz teksta(TextView)
- 2 elementa za unos teksta(EditText) u koje će korisnik unijeti korisničko ime i lozinku
- gumb za prijavu

1.10.1.1. Android Studio

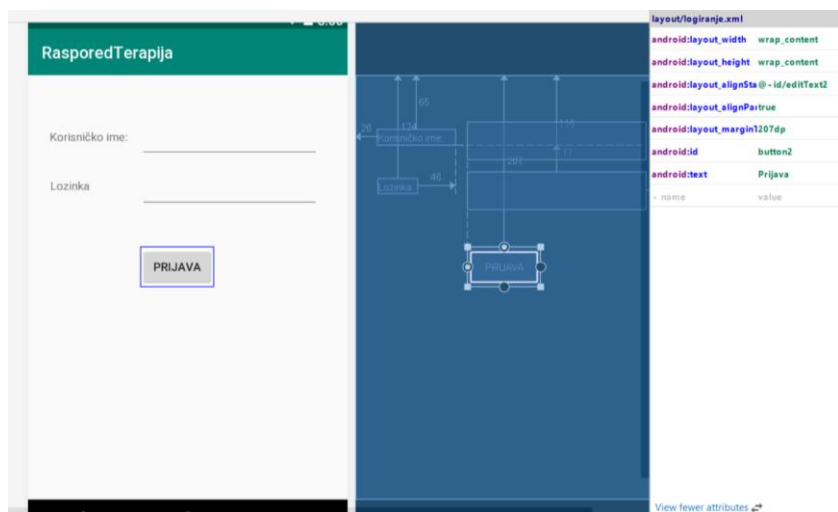
Kada je sadržaj ovih poglavlja malo veći, kao u ovom slučaju, mislim da je onda ok da imate podpoglavljja

Kada je sučelje kreirano pojaviti će se pregled početnog dizajna sučelja, odnosno prazno sučelje.



Ukoliko želimo vidjeti xml kod dizajna, kliknemo na gumb Text u donjem lijevom kutu. Sa lijeve strane nalazi se izbornik elemenata koje možemo dodati na sučelje. Osobno mislim da je najlakše kreirati zamišljeni dizajn na način da u pregledu sučelja iz izbornika dodamo željene elemente, postavimo ih na sučelje te se potom prebacimo na xml kod dizajna i tamo dodamo željene attribute elementima.

Kako bi dodali elemente na sučelje, kliknite na element iz izbornika i povucite ga na sučelje. Kada kliknete na element, otvoriti će vam se prozor sa atributima određenog elementa, možete i ovdje promijeniti attribute umjesto u xml kodu. Nakon inicijalnog dodavanja elemenata, dizajn će izgledati ovako:



Možete nastaviti uređivati u pregledniku ali osobno mi je jednostavnije prebaciti se na xml kod. Ono što preporučam kod dizajniranja strukture elemenata (ukoliko ste početnik u xml-u) jest da obratite pozornost na `layout_align` attribute jer oni omogućuju da se elementi poravnavaju na temelju drugih elemenata, a ako elemente poravnavate na temelju baznog elementa, može se dogoditi da se elementi raštrkaju ukoliko se promjeni veličina uređaja. Sljedeće na redu jest kreiranje dizajna za elemente na aplikaciji. Za svaki dizajn elementa u aplikaciji kreiramo xml datoteku u direktoriju `/res/drawable` te ga pridružujemo elementu na sučelju. Kreirati ćemo za početak dizajn gumba koji će imati 2 stanja, u stanju mirovanja i kada je kliknut. Kako bi to postigli, kreiramo 3 xml datoteke:

- `button_enabled` – predstavlja dizajn gumba u stanju mirovanja

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding="15dp">
    <solid android:color="#7ED5B3" />
    <corners
        android:bottomRightRadius="19dp"
        android:bottomLeftRadius="19dp"
        android:topLeftRadius="19dp"
        android:topRightRadius="19dp" />
    <padding android:top="14dp"
        android:bottom="10dp" />
    <stroke android:width="3dp" android:color="#64C9A1" />
</shape>
```

- button_pressed – predstavlja dizajn kliknutog gumba

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding="15dp">
    <solid android:color="#64C9A1" />
    <corners
        android:bottomRightRadius="19dp"
        android:bottomLeftRadius="19dp"
        android:topLeftRadius="19dp"
        android:topRightRadius="19dp" />
    <padding android:top="14dp"
        android:bottom="10dp" />
    <stroke android:width="3dip" android:color="#64C9A1" />
</shape>
```

- button – u ovoj datoteci definiramo da će element imati 2 stanja(normalno i kliknuto), te za svako stanje pridružujemo zasebni dizajn

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_pressed="true"
        android:state_enabled="true"
        android:drawable="@drawable/button_pressed" />
    <item
        android:state_enabled="true"
        android:drawable="@drawable/button_enabled" />
</selector>
```

Nakon toga elementu na sučelju(u ovom slučaju gumbu za prijavu) pridružujemo kreirani dizajn

```
android:background="@drawable/button"
```

Sljedeće ćemo kreirati dizajn za unos teksta, u ovom slučaju element neće imati dizajn za više stanja te će biti dovoljna samo jedna datoteka. U /res/drawable kreiramo novu xml datoteku koju ćemo nazvati edit_text i kreiramo željeni dizajn.

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding="15dp">
    <solid android:color="#64C9A1" />
    <corners
        android:bottomRightRadius="10dp"
        android:bottomLeftRadius="10dp"
        android:topLeftRadius="10dp"
        android:topRightRadius="10dp" />
    <padding
        android:top="14dp"
        android:bottom="10dp"
        android:left="10dp" />
    <stroke android:width="3dip" android:color="#7ED5B3" />
</shape>
```

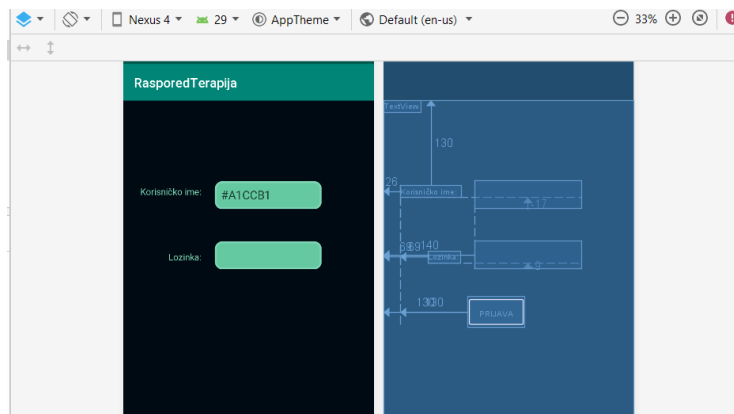
Kada je dizajn završen pridružujemo ga svim elementima za unos teksta na sučelju.

```
android:background="@drawable/edit_text"
```

Ukoliko želimo promijeniti boju pozadini aplikacije, definiramo ju u baznom elementu sučelja.

```
<RelativeLayout 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"
    android:background="#000A13">
```

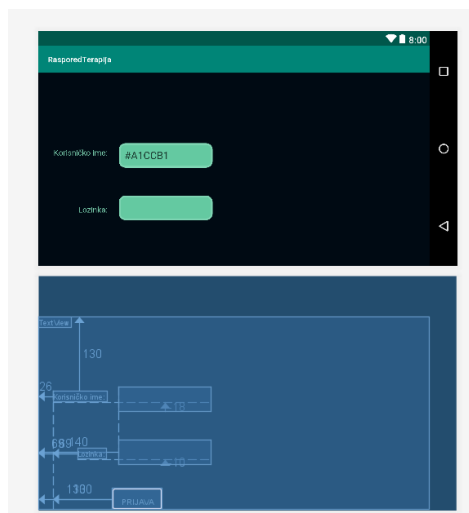
Također ćemo još promijeniti boju svim elementima za prikaz teksta na sučelju, tako da se uklapaju u dizajn. Ukoliko se sada prebacite na pregled prikaza sučelja, ono će izgledati ovako:



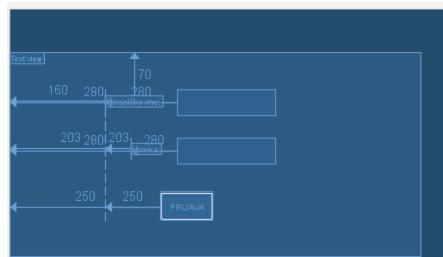
Kao što vidimo u pregledu nije prikazan odgovarajući dizajn gumba, ali nema veze jer ponekad ukoliko je u pridruženom dizajnu elementa definirano više stanja, element se neće prikazati u pregledniku, ali normalno će se prikazivati na emulatoru ili mobilnom uređaju, tako da ukoliko se dogodi ova situacija, samo pokrenite aplikaciju i pokazati će se je li stvarno u pitanju greška ili samo nemogućnost prikaza u pregledniku.

Kada smo završili sa vertikalnim dijelom dizajna, najjednostavnije je kopirati xml kod sučelja i zalijepiti u datoteku sa horizontalnim dizajnom ukoliko je već kreirana(ako nije slijedite korake iz prethodnog poglavlja). Android Studio u prozoru za pregled strukture projekta datoteke sa horizontalnim dizajnom označava sa (land).

Nakon što ste zalijepili xml kod, najbolje je pokrenuti aplikaciju na emulatoru ili uređaju i vidjeti kako izgleda u horizontalnom prikazu, odnosno koje promjene treba napraviti u dizajnu. Na primjer ukoliko ne podesimo dizajn za horizontalni dizajn, sučelje će izgledati ovako u pregledniku dizajna:



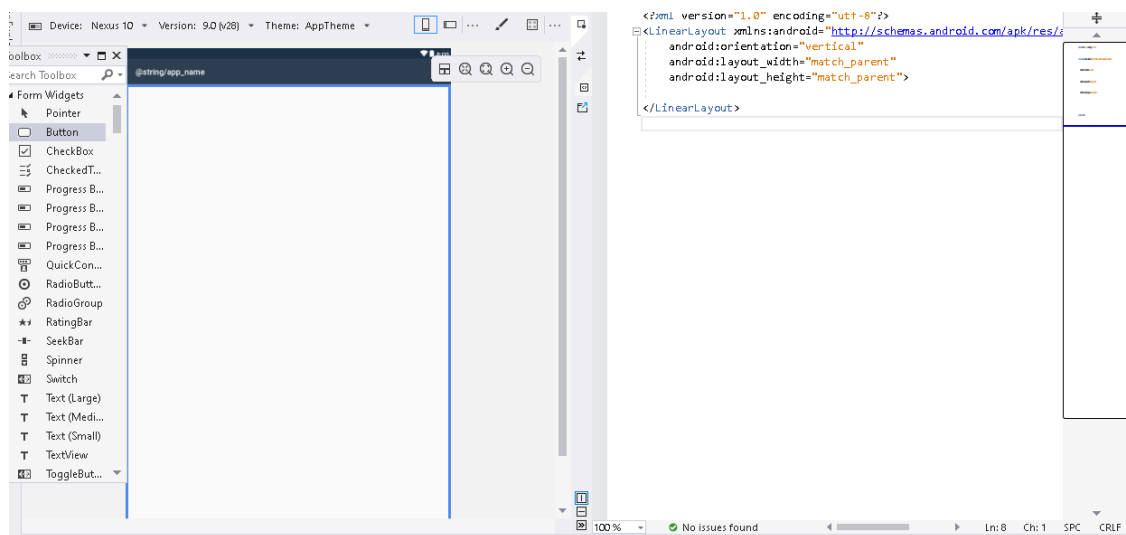
U ovom slučaju individualni dizajn elemenata neće se mijenjati tako da ću samo podesiti margine elemenata (atributi `layout_margin`). Nakon podešavanja horizontalni dizajn bi trebao izgledati ovako:



Nakon što ste završili oba dizajna pokrenite aplikaciju i testirajte prijelaz iz vertikalnog u horizontalno stanje mobilnog uređaja.

1.10.1.2. Xamarin

Nakon što je završen proces kreiranja sučelja otvara se preglednik dizajna sučelja i xml kod dizajna.

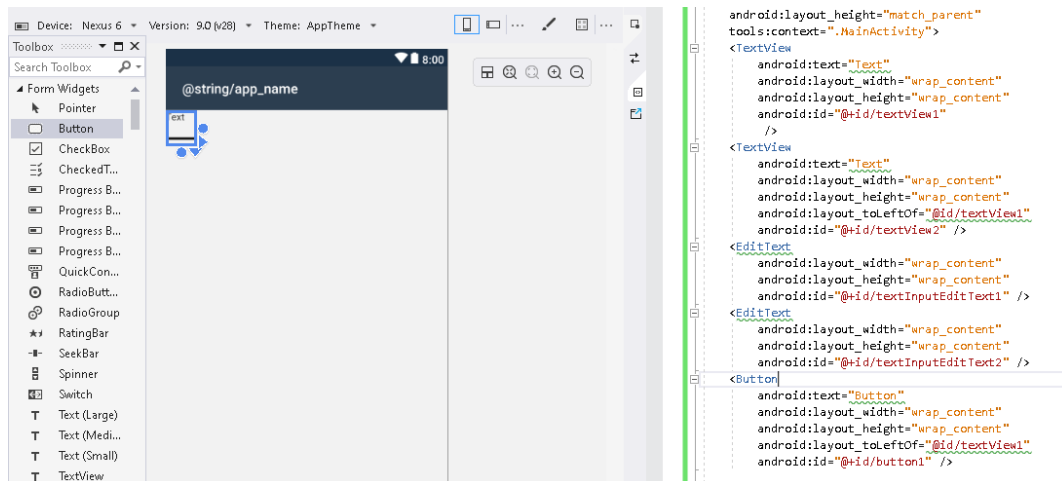


Ako mijenjamo xml kod dizajna sve promjene će se automatski prikazati u pregledniku sučelja. Sa lijeve strane nalazi se gumb Toolbox koji otvara izbornik elemenata za sučelje. Xamarin će automatski kao bazni element sučelja postaviti `LinearLayout`, koji ćemo ručno u xml kodu zamijeniti za `RelativeLayout` i povezati ćemo sučelje sa početnom aktivnosti.

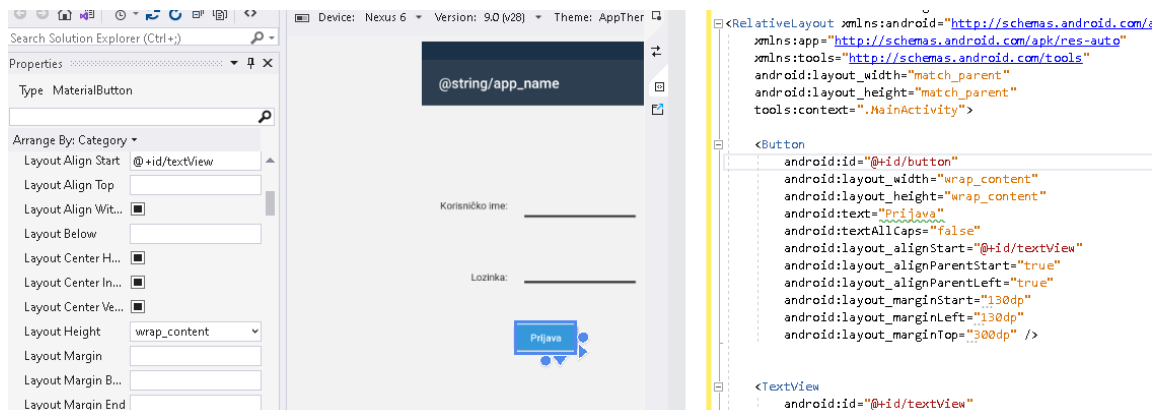
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

Kao i u prethodnom postupku elemente na sučelje dodajemo tako da iz izbornika kliknemo na element i povučemo ga na pregled sučelja (ako koristite `RelativeLayout` ponekad

se element ne želi kreirati u sučelju tako u izborniku umjesto da odvučete element samo 2 puta kliknete na njega i element će se kreirati). U pregledniku sučelja ne možete mijenjati poziciju elemenata na način da kliknete element i odvučete ga na željeno mjesto(kao što Android Studio dozvoljava) zbog toga početni dizajn nakon dodavanja elemenata izgleda ovako:



Kada su dodani svi željeni elementi, u xml kodu definiramo njihove pozicije. U ovom koraku dizajna vrijede ista pravila kao i u Android Studio-u(obraćati pozornost na layout_align atribute). Xml kod možete mijenjati ručno ili u prozoru sa atributima koji se otvara nakon što kliknete na element(kao što se vidi na slici ispod), u svakom slučaju morate biti upoznati sa funkcijama atributa.

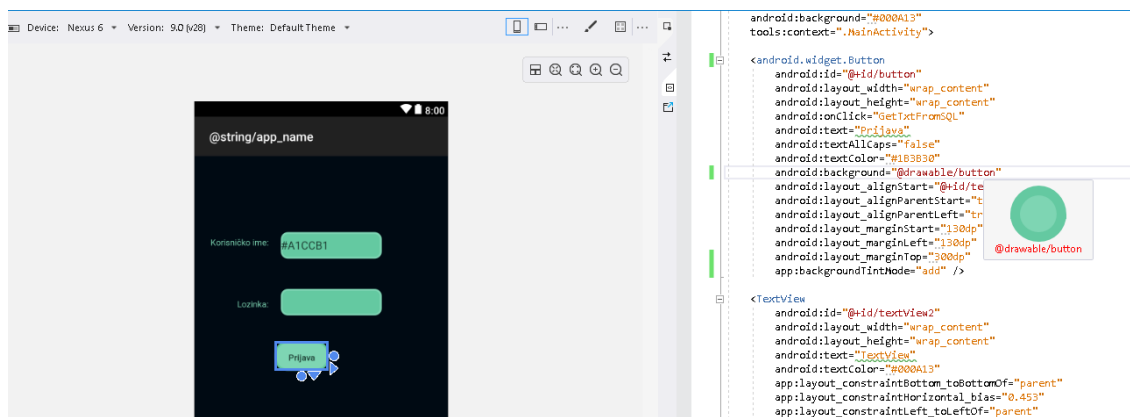


Nakon što ste zadovoljni strukturom sučelja kreiramo dizajn za svaki element. Kreiranje dizajna u Xamarin-u ima vrlo sličan princip kao i Android Studio, u direktoriju /Resources/drawable kreiramo xml datoteku, u njoj definiramo željeni dizajn te ga pridružujemo elementu na sučelju. Kao i u prethodnom primjeru kreirati ćemo sljedeće datoteke:

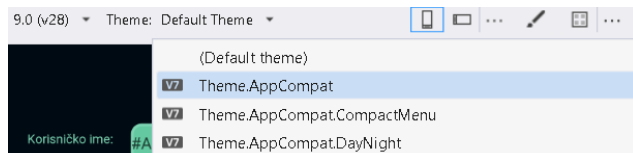
- button_enabled – predstavlja dizajn gumba u stanju mirovanja
- button_pressed – predstavlja dizajn kliknutog gumba

- button – u ovoj datoteci definiramo da će element imati 2 stanja(normalno i kliknuto), te za svako stanje pridružujemo zasebni dizajn
- edit_text - dizajn za element unosa teksta

Xml kod dizajna je isti kao i prije. Kada je dizajn završen, pridružujemo ga elementima kao i prošli puta. U ovom slučaju Xamarin ima jednu prednost, a to je kada pridružimo dizajn elementu u xml kodu automatski vidimo kako dizajn izgleda i bez da se prebacujemo na preglednik dizajna. Također preglednik nema problema sa prikazom gumba.



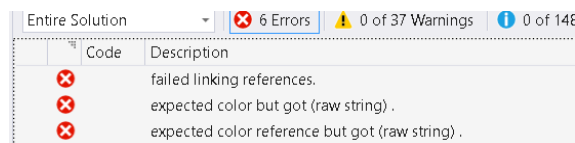
Kod kreiranja dizajna sučelja obratite pozornost na zadanu temu u pregledniku dizajna, Xamarin će automatski temu postaviti na AppCompatActivity temu, prebacite je na Default temu.



Ako preskočite taj korak ponekad neki elementi(npr.gumb) neće htjeti poprimiti zadani dizajn iako ste ga pravilno definirali.



Kada ste promijenili temu pojaviti će vam se greške kod kompajliranja projekta jer je promjena teme izbrisala definirane boje koje se koriste u baznoj temi.



Entire Solution		6 Errors	0 of 37 Warnings	0 of 146
	Code	Description		
		failed linking references.		
		expected color but got (raw string) .		
		expected color reference but got (raw string) .		

Kako bi popravili tu grešku otvorite datoteku `/Resources/values/styles.xml` koja će izgledati ovako:

```
<resources>
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.MaterialComponents.Light.DarkActionBar">
  <item name="colorPrimary"></item>
  <item name="colorPrimaryDark"></item>
  <item name="colorAccent"></item><item name="android:colorBackground" /><item name="android:windowBackground" /></style>
</resources>
```

U datoteci izbrišite sve definirane elemente unutar elementa `<style>`.

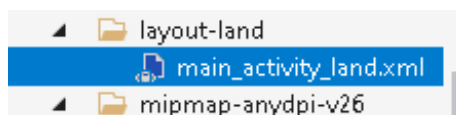
```
<resources>
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.MaterialComponents.Light.DarkActionBar">
</style>
</resources>
```

Nakon toga očistite i kompajlirate projekt, greške bi trebale nestati i elementi će poprimiti zadani dizajn. U ovoj datoteci možete i kreirati zasebnu temu.

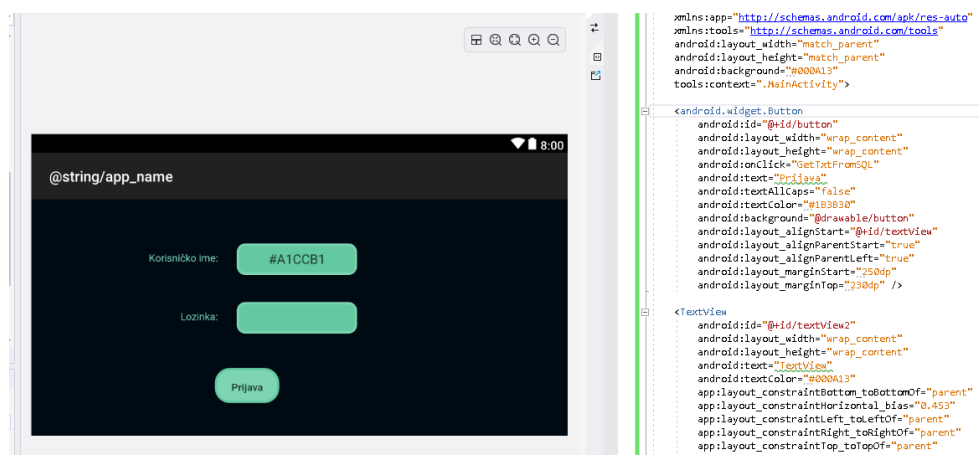
```
<android.widget.Button
  android:id="@+id/button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:onClick="GetTxtFromSQL"
  android:text="Prijava"
  android:textAllCaps="false"
  android:textColor="#1B3B30"
  android:background="@drawable/button"
```

Također bitno je napomenuti da kod kreiranja gumba element ne definirate kao samo `<Button>` već kao `<android.widget.Button>`, jer ponekad kod kompajliranja projekta može se dogoditi greška i Xamarin neće prepoznati `<Button>` kao element gumba.

Kada je završen dizajn za vertikalni prikaz, kreirajte dizajn za horizontalni prikaz.



Prvo u direktoriju `/Resources` kreirajte mapu `layout-land`, u njoj kreirajte xml datoteku pod nazivom `main_activity_land`. U njoj ćemo definirati dizajn za horizontalni prikaz. Kao i prije najlakše je kopirati dizajn za vertikalni prikaz i napraviti preinake. Također je bitno da svi elementi imaju isti id kao i u horizontalnom prikazu.



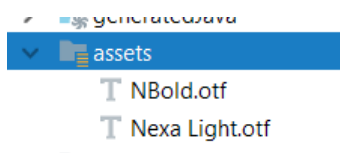
Nakon kreiranih preinaka na dizajnu, otvorite aktivnost koja je pridružena sučelju i u onCreate funkciji(funkcija koja se prva pokreće) provjerite širinu i visinu ekrana, ukoliko je širina veća od visine, otvoriti će sučelje za horizontalni prikaz inače se otvara sučelje za vertikalni prikaz.

```
protected override void OnCreate(Bundle savedInstanceState)
{
    base.OnCreate(savedInstanceState);
    con = BazaPod.getInstanca();
    Xamarin.Essentials.Platform.Init(this, savedInstanceState);
    var rezolucija = Resources.DisplayMetrics;
    var widthInDp = ConvertPixelsToDp(rezolucija.WidthPixels);
    var heightInDp = ConvertPixelsToDp(rezolucija.HeightPixels);
    if (widthInDp > heightInDp)
    {
        v1 = this.LayoutInflater.Inflate(Resource.Layout.main_activity_land, null);
    }
    else {
        v1 = this.LayoutInflater.Inflate(Resource.Layout.activity_main, null);
    }
}
```

1.10.2. Promjena fonta

Ako želite u aplikaciji postaviti specifičan font(koji nije dostupan), morate ga prvo skinuti. Na Internetu možete naći mnogo besplatnih fontova na stranicama poput freefonts. Osobno sam skinula dvije verzije Nexa fonta(Bold i Light). Promjena fonta neće se vidjeti ni u jednom pregledniku dizajna, morati će te pokrenuti aplikaciju da bi promjene došle do izražaja.

1.10.2.1. Android Studio



Kako bi mogli koristiti fontove u aplikaciji, datoteke spremite u direktorij /assets te zatim u aktivnosti koja je pridružena sučelju definirajte elemente putem njihovog id-a iz sučelja i definirajte željeni font, nakon toga upotrijebite funkciju setTypeface kako bi svim željenim elementima promijenili font.

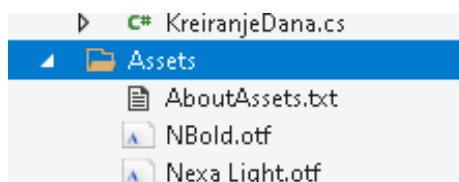
```

public class MainActivity extends AppCompatActivity{
    Button mybtn;
    TextView t1, t2, t3;
    Baza con;
    private EditText username,password;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.FLAG_FULLSCREEN);
        getSupportActionBar().hide();
        Typeface type = Typeface.createFromAsset(getAssets(), "NBold.otf");
        setContentView(R.layout.activity_main);
        con=Baza.getInstance();
        mybtn = (Button)findViewById(R.id.button);
        mybtn.setTypeface(type);
        t1= (TextView) findViewById(R.id.textView2);
        t2= (TextView) findViewById(R.id.textView);
        t2.setTypeface(type);
        t3= (TextView) findViewById(R.id.textView3);
        t3.setTypeface(type);
        username = (EditText) findViewById(R.id.korime);
        username.setTypeface(type);
        password = (EditText) findViewById(R.id.pass);
        password.setTypeface(type);
    }
}

```

1.10.2.2. Xamarin

Sličan postupak koristi se i u Xamarin-u. Prvo datoteke fontova smjestimo u direktorij / Assets.



Zatim otvaramo aktivnost pridruženu sučelju i u funkciji onCreate(ako želimo da se font primjeni odmah nakon pokretanja aplikacije) definiramo elemente sučelja i željeni font te potom elementima promijenimo font.

```

Typeface font = Typeface.CreateFromAsset(Assets, "NBold.otf");
mybtn = v1.FindViewById<Button>(Resource.Id.button);
t1 = v1.FindViewById<TextView>(Resource.Id.textView2);
t2 = v1.FindViewById<TextView>(Resource.Id.textView);
username = v1.FindViewById<EditText>(Resource.Id.korime);
password = v1.FindViewById<EditText>(Resource.Id.pass);
mybtn.Typeface = font;
t1.Typeface = font;
t2.Typeface = font;
password.Typeface = font;
username.Typeface = font;
mybtn.Typeface = font;

```

1.10.3. Kreiranje Dana

Nakon što se korisnik uspješno prijavio biti će proslijeđen na sučelje koje izvršava funkcionalnost unosa podataka za 1 dan te se sastoji od sljedećih elementa:

- 7 elementa za prikaz teskta(TextView)
- 3 elementa za unos teksta(EditText) u koje će korisnik unositi podatke
- Padajući izbornik(Spinner) u kojem će korisnik moći vidjeti popis običenih korisnika

1.10.3.1. Android studio

Kreirajte novo sučelje pod imenom activity_kreiranje_dana, zatim promijenite bazni element sučelja u RelativeLayout i postavite sve željene elemente na sučelje. Postupak je isti kao i za

sučelja logiranja, osim što u ovom sučelju postoji element padajućeg izbornika tako da za njega moramo kreirati posebni dizajn kako bi odgovarao ostatku. U direktoriju /res/drawable kreiramo novu xml datoteku koju ćemo nazvati spinner i definirati dizajn za padajući izbornik.

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <solid android:color="#EEF2F5" />
  <corners android:radius="13dp" />
  <stroke
    android:width="1dp"
    android:color="@android:color/darker_gray" />
</shape>
```

Ovaj dizajn nećemo direktno primijeniti na element već ćemo ga ugnijezditi u zasebni RelativeLayout element na koji ćemo primijeniti dizajn.

```
<RelativeLayout
  android:id="@+id/spinnerrel"
  android:layout_width="135dp"
  android:layout_height="wrap_content"
  android:layout_alignBottom="@+id/textView7"
  android:layout_alignParentStart="true"
  android:layout_alignParentLeft="true"
  android:layout_marginStart="90dp"
  android:layout_marginLeft="90dp"
  android:layout_marginBottom="-3dp"
  android:layout_weight=".28"
  android:background="@drawable/spinner"
  android:gravity="center_horizontal">
  <Spinner
    android:id="@+id/spinner"
    android:layout_width="125dp"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:layout_gravity="center"
    android:background="#EEF2F5"
    android:gravity="center"
    android:layout_marginLeft="4dp"
    android:spinnerMode="dropdown" />
```

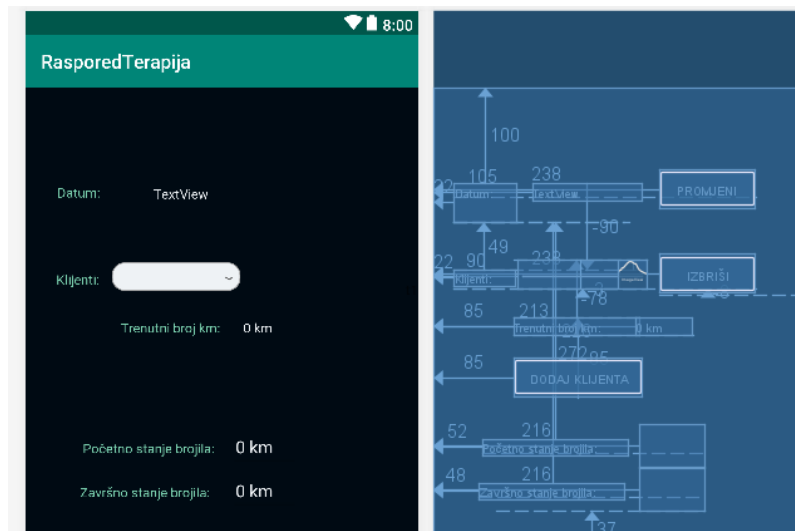
Na ovaj način primjene dizajna možemo dodati sliku strelice na kraj padajućeg izbornika.



Željene sliku spremite u direktorij /res/drawable te pod imenom strelica. Zatim u elementu RelativeLayout u kojem se nalazi padajući izbornik dodajte element <ImageView> koji će služiti za prikaz slike strelice, na element povežite sliku.

```
<ImageView
  android:layout_width="30dp"
  android:layout_height="30dp"
  android:layout_alignParentRight="true"
  android:layout_centerVertical="true"
  android:layout_gravity="center"
  android:src="@drawable/strelica" />
</RelativeLayout>
```

Trenutno će u pregledniku dizajn izgledati otprilike ovako:



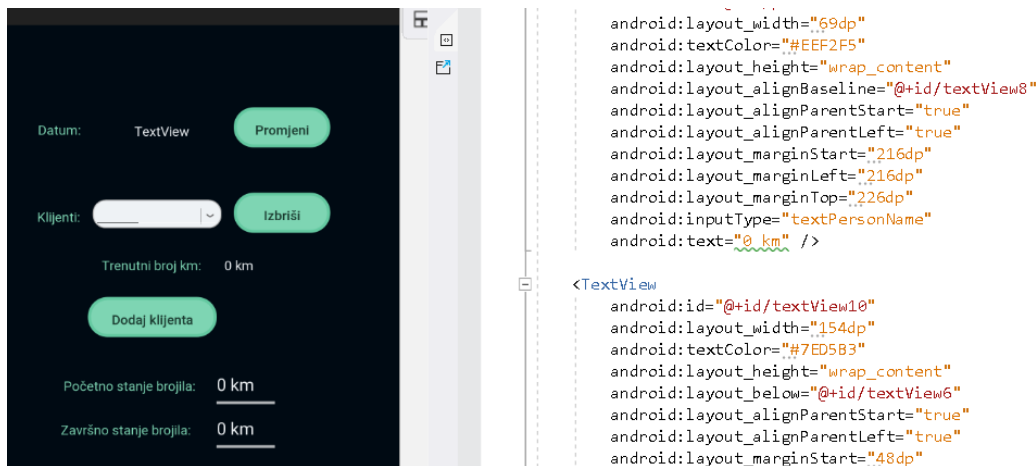
Kao i prije gumbi kojima je promijenjen dizajn se ne vide, ali normalno se pokazuju nakon pokretanja aplikacije. Sada je na redu vertikalni prikaz sučelja, postupak je isti kao i kod prethodnog sučelja(kreiramo sučelje za horizontalni prikaz sa istim nazivom kao i za vertikalni). Napravimo neke preinake tako da dizajn odgovara dimenzijama, ali u ovom slučaju bazni element moramo ugnijezditi u <ScrollView> element, kako bi korisnik mogao imati prikaz nad cijelim sučeljem i u vertikalnom prikazu.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <RelativeLayout 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"
        android:background="#000A13"
        tools:context=".Aktivnosti.KreiranjeDana">
```

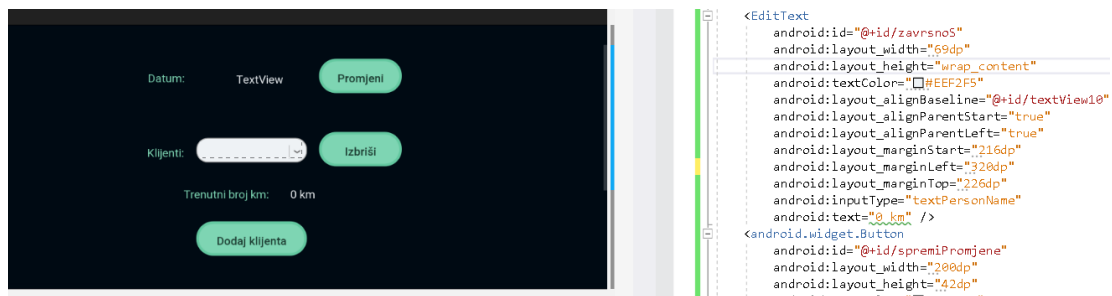
Ukoliko izostavimo ovaj element a sučelje sadrži previše elemenata korisnik će moći vidjeti samo one koji stanu na ekran te neće imati pristup ostalim.

1.10.3.2. Xamarin

U direktoriju /Resources/layout kreiramo novu xml datoteku pod nazivom activity_kreiranja_dana. Dodajte te sve željene elemente i u xml kodu definirajte strukturu sučelja(xml kod je isti kao i kod postupka za Android Studio, osim što kod definiranja elementa gumba moramo pripaziti da ga kreiramo kao <android.widget.Button>). Kako bi mogli dodati sliku strelice smještamo ju u direktorij /Resource/drawable. Nakon svih promjena u pregledniku dizajna sučelje bi trebalo izgledati otprilike ovako:



Kako bi kreirali horizontalni dizajn u direktoriju Resources/layout-land kreiramo xml datoteku pod nazivom activity_kreiranja_dana_land. U nju kopiramo vertikalni dizajn i sve ugnijezdimo u <ScrollView> element (isto kao i u Android Studio-u). Napravimo preinake u marginama elementa kako bi dizajn odgovarao horizontalnom prikazu.



U pregledniku dizajna možemo vidjeti kako će otprilike izgledati sučelje. Kao i u sučelju za logiranje moramo u pridruženoj aktivnosti provjeravati veličinu ekrana na kojoj se pokreće aplikacija i na temelju toga odabiremo sučelje.

```
var rezolucija = Resources.DisplayMetrics;
var widthInDp = ConvertPixelsToDp(rezolucija.WidthPixels);
var heightInDp = ConvertPixelsToDp(rezolucija.HeightPixels);
if (widthInDp > heightInDp)
{
    v1 = this.LayoutInflater.Inflate(Resource.Layout.activity_kreiranja_dana_land, null);
}
else
{
    v1 = this.LayoutInflater.Inflate(Resource.Layout.activity_kreiranja_dana, null);
}

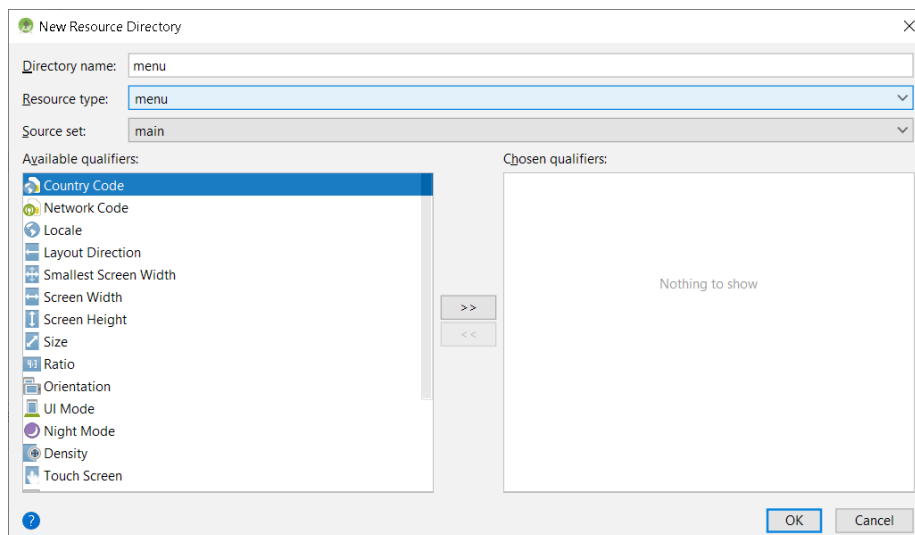
SetContentView(v1);
```

1.10.4. Izbornik

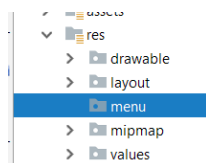
Na sučelju za kreiranje novog dana dodati ćemo još i izbornik kako bi se korisnik mogao lakše snalaziti u aplikaciji. Izbornik će se nalaziti na vrhu sučelja. Te će sadržavati poveznice na spremanje dana, kreiranje rasporeda i odjavu korisnika.

1.10.4.1. Android Studio

Kako bi kreirali element menu- a kliknemo desnim klikom na direktorij /res te odaberemo New Resource Directory. Otvoriti će se ovakav prozor.



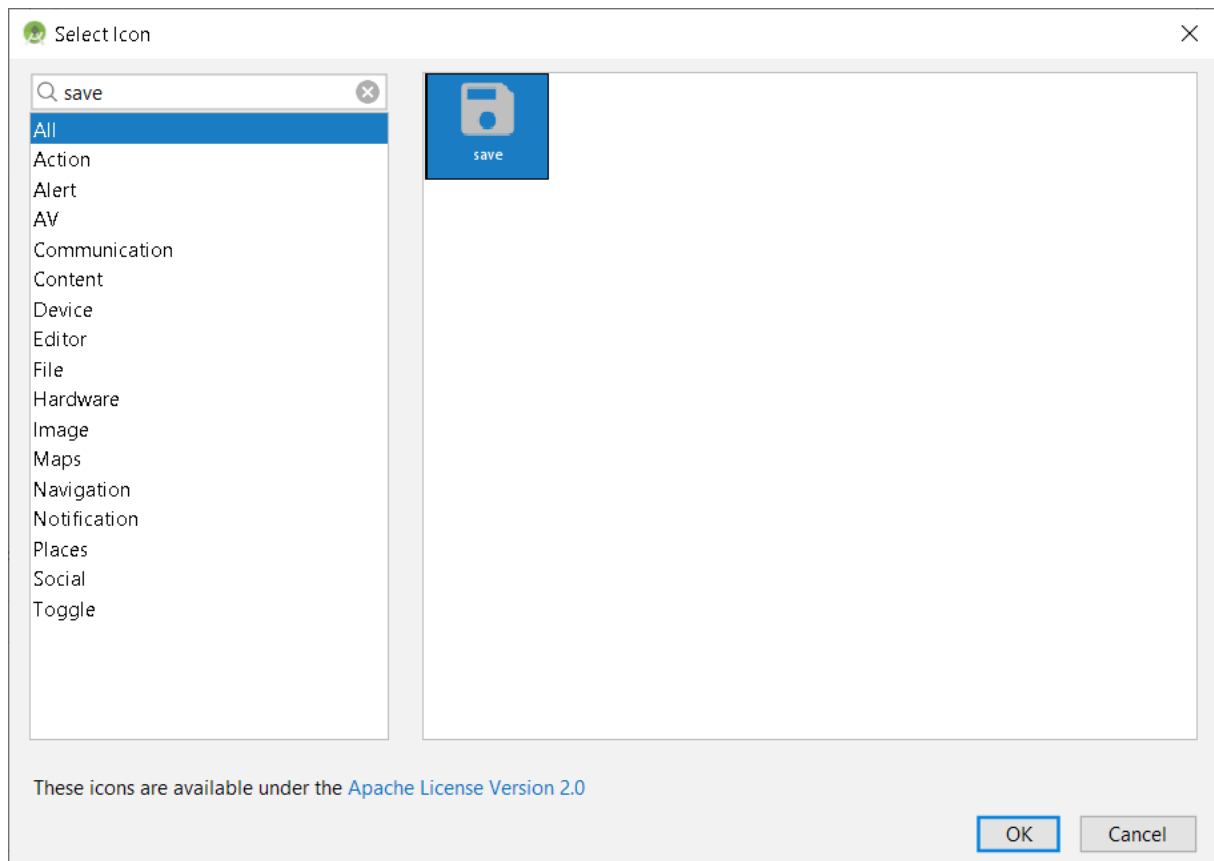
Iz padajućeg izbornika odaberite menu i kliknite gumb Ok. Trebao bi se kreirati direktorij pod nazivom menu.



Kliknite desnim klikom na direktorij i odaberite New-> Menu resource file te ga nazovite menu. U direktoriju će se kreirati xml datoteka u kojoj ćemo definirati strukturu izbornika na sljedeći način:

```
<?xml version="1.0" encoding="utf-8" ?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="#7ED5B3">
    <item
        android:id="@+id/spremi"
        android:icon="@drawable/save"
        android:title="Spremi"
        app:showAsAction="ifRoom" />
    <item
        android:id="@+id/odjavi"
        android:title="Odjava"
        app:showAsAction="never" />
    <item
        android:id="@+id/kreirajRas"
        android:title="Kreiraj Raspored"
        app:showAsAction="never" />
</menu>
```

Želimo da korisniku bude uvijek dostupna poveznica na spremanje dana, zbog toga je kreirana posebna ikona za spremanje. Ikona je pohranjena u direktoriju /res/drawable te je pridružena poveznici za spremanje.

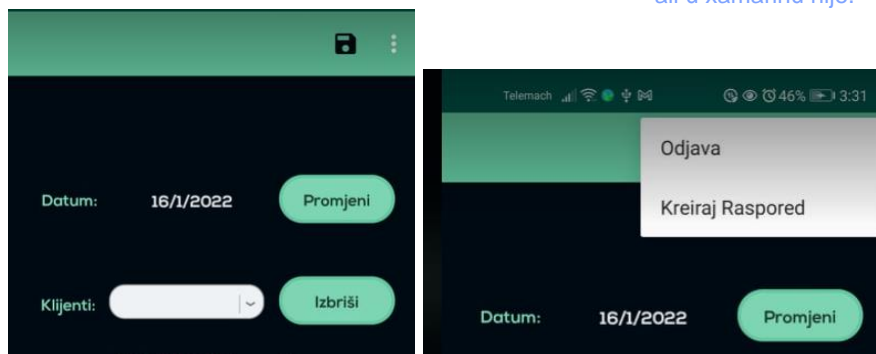


Kako bi se izbornik uspješno prikazivao u aktivnosti koja je pridružena sučelju (KreiranjeDana) definiramo `onCreateOptionsMenu()` te pomoću nje prikazujemo izbornik.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater=getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}
```

Nakon pokretanja aplikacije u sučelju za kreiranje dana trebao bi se pojaviti funkcionalan izbornik.

Ovdje je prikazano kako izbornik izgleda nakon pokretanja aplikacije, ali u xamarinu nije.

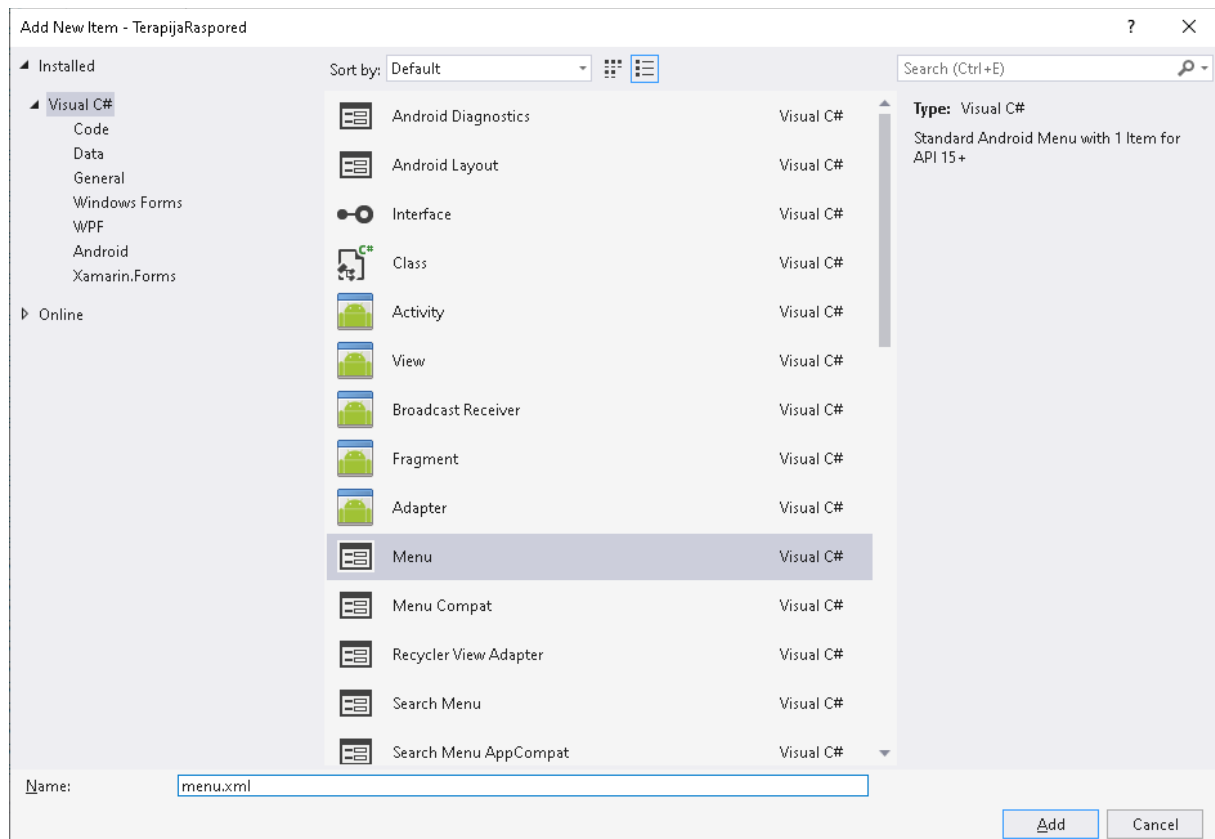


1.10.4.2. Xamarin

U Visual studio-u, morate kreirati prazni direktorij pod nazivom /resources/menu.



U tom direktoriju kreirajte xml datoteku tipa menu.



Nakon kreiranja datoteke možete upotrijebiti isti xml kod za menu kao i u Android Studio-u.

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <!--For all properties see: https://aka.ms/android-menu-resource-->
3 <menu xmlns:android="http://schemas.android.com/apk/res/android">
4
5     <item android:id="@+id/action_search"
6           android:showAsAction="always"
7           android:text="Search"
8           android:actionViewClass="android.widget.SearchView"/>
9
10 </menu>
11
```

Ukoliko je sve uspješno kompajlirano, aplikacija bi se trebalo uspješno pokrenuti(zajedno sa izbornikom na drugom sučelju).

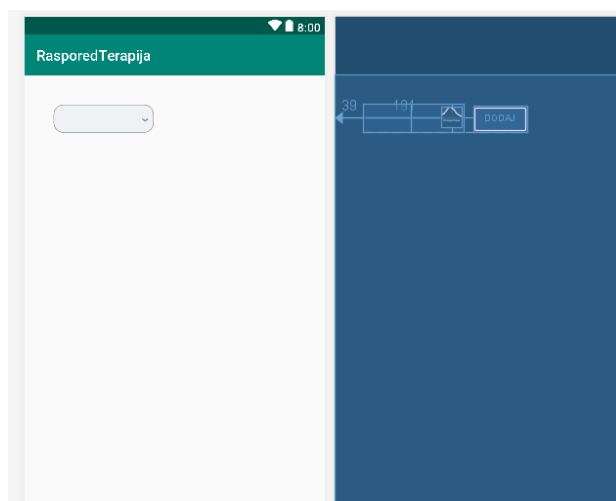
```
public override bool OnCreateOptionsMenu(IMenu menu)
{
    MenuInflater.Inflate(Resource.Menu.menu, menu);
    return true;
}
```


1.10.5. Dodavanje klijenta

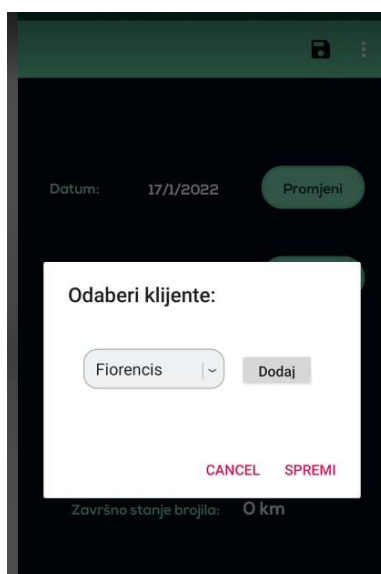
Kao što je bilo spomenuto na početku korisniku se mora omogućiti bilježenje običenih klijenata po danu. Trenutno običeni klijenti prikazivati će se u padajućem izborniku na sučelju za kreiranje dana. Kako bi korisnik mogao dodavati klijente klijente biti će kreirano sučelje koje će sadržavati popis svih klijenata te će putem njega korisnik moći izabrati i dodati klijenta.

1.10.5.1. Android Studio

U direktoriju /res/layout kreirati ćemo novu xml datoteku pod nazivom dodavanje_klijenata koja će sadržavati padajući izbornik i jedan gumb. Nakon što smo postavili sve elemente, u pregledniku dizajna sučelje bi trebalo izgledati ovako:



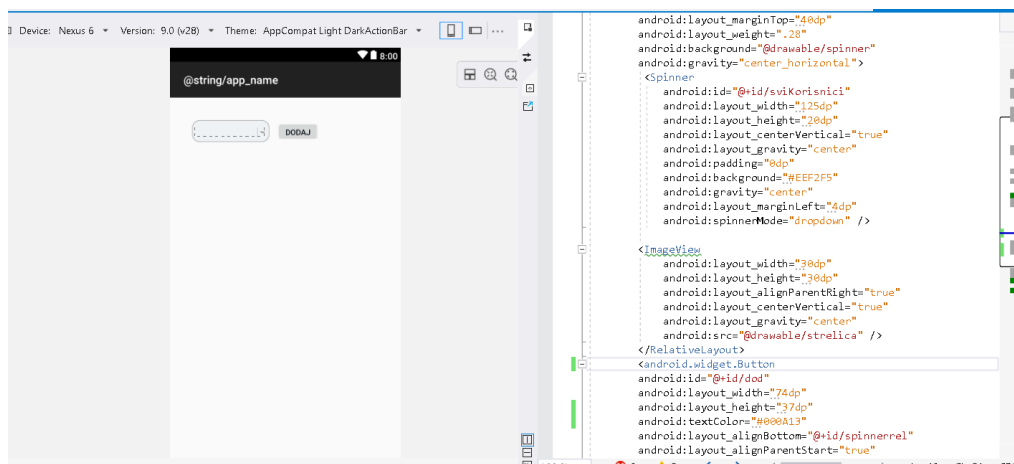
Trenutno nije bitno što elementi nisu centrirani jer će sučelje biti prikazano kao iskočni prozor nakon što korisnik klikne na gumb za dodavanje klijenta. Nakon što se pokrene aplikacija sučelje će izgledati ovako:



U ovom slučaju nema potrebe dvije vrste dizajna (vertikalni i horizontalni) pošto iskočni prozor i u jednom i u drugom obliku ne mijenja oblik.

1.10.5.2. Xamarin

U direktoriju /Resources/layout kreirati ćemo xml datoteku dodavanje_klijenta. Kao i prijašnja verzija sastojati će se od padajućeg izbornika i jednog gumba.



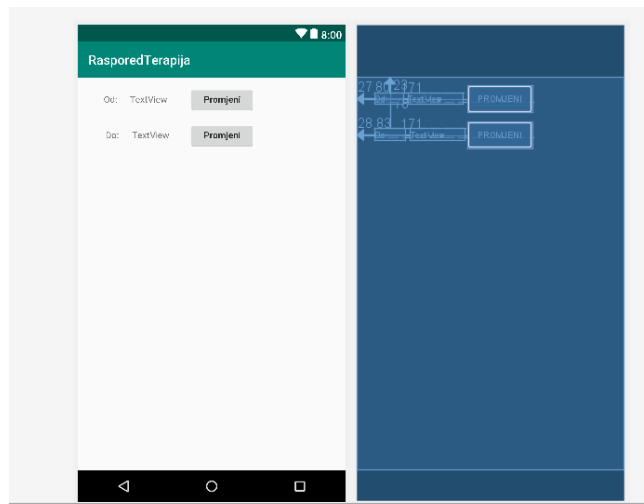
Za sada izgled sučelja nećemo puno mijenjati. Kada se aplikacija pokrene sučelje bi trebalo izgledati isto kao i u Android Studio-u.

1.10.6. Odabir datuma za kreiranje rasporeda

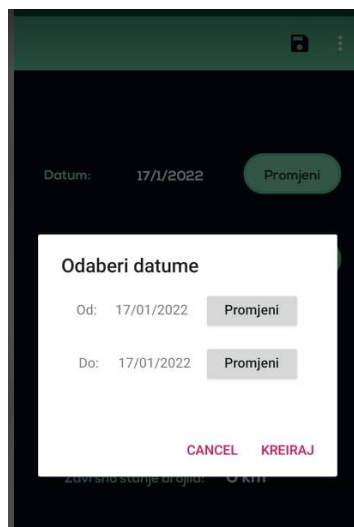
Nakon što je korisnik unio podatke za sve potrebne dane, treba mu biti omogućena funkcionalnost odabira datuma za kreiranje .pdf datoteke rasporeda. Sučelje će se također prikazivati kao iskočni prozor nakon što korisnik u izborniku odabere opciju Kreiraj raspored.

1.10.6.1. Android Studio

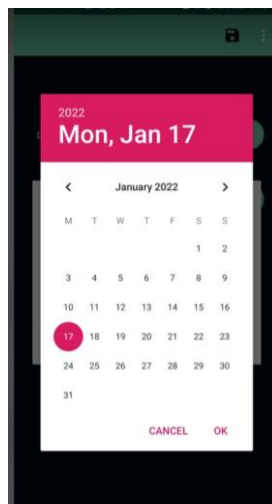
U direktoriju /res/layout/ kreirajte datoteku raspored.xml. Glavni elementi sučelja su 2 elementa za prikaz u kojima će se ispisati odabrani datumi i 2 gumba pomoću kojih će korisnik mijenjati datume. U pregledniku dizajna sučelje bi trebalo izgledati otprilike ovako:



Nakon pokretanja aplikacije trebalo bi izgledati ovako:

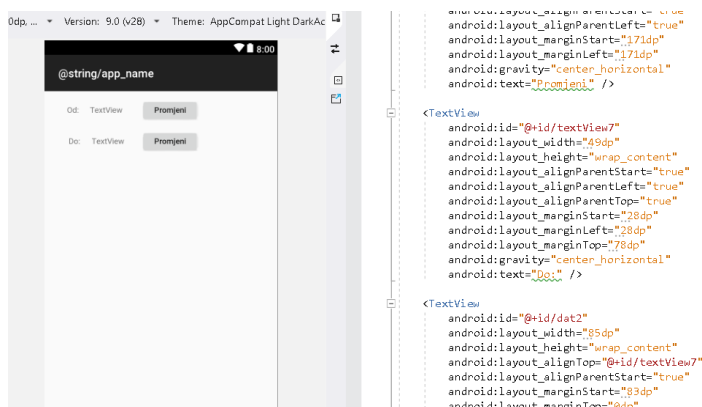


Korisnik će jednostavno moći odabrati željene datume i za te datume bit će kreiran raspored. Kasnije će se još dodati funkcionalnost otvaranja kalendara za odabir datuma.



1.10.6.2. Xamarin

U direktoriju /Resources/layout kreiramo datoteku raspored.xml. Postavimo i pozicioniramo sve željene elemente na sučelje.



Nakon pokretanja aplikacije, sučelje bi trebalo izgledati isto kao i verzija koja je kreirana u Android Studio-u.

1.11. Funkcionalnosti

1.11.1. Prijava

Kada korisnik na početnom sučelju upiše korisničko ime i lozinku te klikne na gumb za prijavu, aplikacija mora provjeriti ispravnost upisanih podataka. Ukoliko su podaci točni, iz baze se preuzimaju podaci o klijentima i danima koje je korisnik prethodno unio i korisnika se preusmjeruje na sučelje za kreiranje dana. Ako podaci nisu točni korisniku se ispisiuje obavijest.

1.11.1.1. Android Studio

Kako bi se podaci mogli spremati i provjeravati prvo ćemo kreirati opisne klase za dane i klijente.

```
public class Klijent {  
  
    private String id;  
    private String ime;  
    private String prezime;  
    private String Adresa;  
    private String Lat;  
    private String Lon;  
    public Klijent(String id, String ime, String prezime, String adresa, String Lat, String Lon) {  
        this.id = id;  
        this.ime = ime;  
        this.prezime = prezime;  
        this.Adresa = adresa;  
        this.Lat=Lat;  
        this.Lon=Lon;  
    }  
}
```

```
public class Dan {  
    private String Id;  
    private String Datum;  
    private List<Klijent> Popis=new ArrayList<>();  
    private String Km;  
    private int Mjesec;  
    private int Godina;  
    private String Pstanje;  
    private String Zstanje;  
  
    public Dan(String id, String datum, String pstanje, String zstanje, String km, List<Klijent> popis, int  
        Id=id;  
        Datum = datum;  
        Popis = popis;  
        Km = km;  
        Mjesec = mjesec;  
        Godina = godina;  
        Pstanje = pstanje;  
        Zstanje = zstanje;  
}
```

Nakon toga kreiramo singleton klasu pod nazivom Baza koja sadrži liste klijenata i dana, također će u njoj biti pohranjen id korisnika koji se uspješno prijavio.

```

public class Baza {
    private List<Klijent> sviKlijenti;
    private List<Dan> sviDani; //iz tog mjeseca
    private static Baza instance = null;
    private Context context;
    private int stavljeno=0;
    private static int UlogiraniZaposlenik;

    public static Baza getInstance() {
        if (instance == null) {
            instance = new Baza();
        }
        return instance;
    }

    private Baza() {
        sviKlijenti=new ArrayList<Klijent>();
        sviDani=new ArrayList<Dan>();
    }
}

```

U istom direktoriju kreiramo klase pod nazivom Podaci i Konekcija. U klasi Podaci ovisno o zadanom zadatku pomoću klase Konekcija dohvaćaju se i čitaju podaci iz baze.

```

public Podaci(Context context, TextView statusField, int
    this.context = context;
    this.statusField=statusField;
    this.zadatak=zadatak;
}

protected void onPreExecute() {
}

@Override
protected String doInBackground(Object[] objects) {
    String link=objects[0].toString();
    Konekcija con= new Konekcija();
    String rez="";
    switch(this.zadatak){
        case 1:
            rez=con.DohvatiPodatke(link);
            rezultat=rez;
            break;
        case 2:
            String podaci=con.DohvatiPodatke(link);
            String d=con.SviKlijenti(podaci);
            rez=d;
            break;
        case 3:
            String podaci2=con.DohvatiPodatke(link);
            String dani=con.SviDani(podaci2);
            rez=dani;
            break;
    }
}

```

Ovisno o zadatku klasi se proslijeđuju url poveznice na php skripte.

```

public String DohvatiPodatke(String link){
    try{
        URL url = new URL(link);
        HttpClient client = new DefaultHttpClient();
        HttpGet request = new HttpGet();
        request.setURI(new URI(link));
        HttpResponse response = client.execute(request);
        BufferedReader in = new BufferedReader(new
        InputStreamReader(response.getEntity().getContent()));
        StringBuffer sb = new StringBuffer("");
        String line="";
        while ((line = in.readLine()) != null) {
            sb.append(line);
            break;
        }
        in.close();
        return sb.toString();
    } catch (Exception e) {
        return "error";
    }
}

```

Ukoliko se dohvaća više podataka(svi klijenti ili svi dani korisnika) u klasi Konekcija podaci se razdvajaju i spremaju u odgovarajuće liste(u klasi Baza).

```

public String SviKlijenti(String podaci){
    List<String> razdvojeniPodaci=new ArrayList<>();
    razdvojeniPodaci=RazdvojiPodatke(podaci);
    Baza con=Baza.getInstance();
    if(con.getStavljeno()==0){
        con.StaviSveKlijente(razdvojeniPodaci);
    }
    List<Klijent> K=con.getSviKlijenti();
    String t="OK";
    return t;
}

public String SviDani(String podaci){
    List<String> razdvojeniPodaci=new ArrayList<>();
    razdvojeniPodaci=RazdvojiPodatke(podaci);
    Baza con=Baza.getInstance();
    if(con.getStavljeno()!=2){
        con.StaviSveDane(razdvojeniPodaci);
    }
    List<Dan> D=con.getSviDani();
    String t="OK";
    return t;
}

```

Ako su podaci uspješno dohvaćeni i pohranjeni, klasi Podaci vraća se potvrdni string. U klasi koja upravlja početnom aktivnošću(MainActivity) definiramo funkciju koja će se pozvati nakon što korisnik klikne na gumb za prijavu.

```

public void GetTxtFromSQL(View view){
    String u = username.getText().toString();
    String p = password.getText().toString();
    String link1 = "http://crofi.com/assets/assets/images/RasporedBaza/Logiranje.php?username="+u+"&password="+p;
    String link2 = "http://crofi.com/assets/assets/images/RasporedBaza/Svilijenti.php";
    Object result= null;
    String rezultat=null;
    Podaci pl=new Podaci(context, this, t1, zadatak 1);
    try {
        result=pl.execute(link1).get();
        rezultat=pl.result();
        if(rezultat.equals("error")||rezultat.equals("No")||rezultat==null){
            Toast.makeText(context, this, "Krivi podaci", Toast.LENGTH_LONG).show();
        }
        else{
            int idKor=Integer.parseInt(rezultat);
            con.setUlogiranizaposlenik(idKor);
            String link3 = "http://crofi.com/assets/assets/images/RasporedBaza/SviDani.php?id="+idKor;
            new Podaci(context, this, t1, zadatak 2).execute(link2).get();
            new Podaci(context, this, t1, zadatak 3).execute(link3).get();
            Intent intent = new Intent(packageContext, this, KreiranjeDana.class);
            startActivity(intent);
        }
    } catch (ExecutionException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

```

Prvo se u funkciji dohvaćaju podaci iz elementa za unos teksta. Zatim se provjerava jesu li upisani podaci ispravni te ako nisu korisniku se ispisuje obavijest. Ako korisnik postoji u bazi pomoću klase Podaci dohvaća se popis svih klijenata i podaci svih dana koje je taj korisnik upisao. Nakon što je preuzimanje podataka završilo korisnika se preusmjerava na sljedeće sučelje.

1.11.1.2. Xamarin

Kao i u prethodnom primjeru prvo kreiramo opisne klase Klijent i Dan.

```

public class Klijent
{
    private String id;
    private String ime;
    private String prezime;
    private String Adresa;
    private String Lat;
    private String Lon;

    2 references
    public Klijent(string id, string ime, string prezime, string adresa, string lat, string lon)
    {
        this.Id = id;
        this.Ime = ime;
        this.Prezime = prezime;
        this.Adresa1 = adresa;
        this.Lat1 = lat;
        this.Lon1 = lon;
    }
}

```

```

14 references
public class Dan
{
    private String Id;
    private String Datum;
    private List<Klijent> Popis = new List<Klijent>();
    private String Km;
    private int Mjesec;
    private int Godina;
    private String Pstanje;
    private String Zstanje;

    2 references
    public Dan(String id, String datum, String pstanje, String zstanje, String km, List<Klijent> popis, int mjesec, int godina)
    {
        Id1 = id;
        Datum1 = datum;
        Popis1 = popis;
        Km1 = km;
        Mjesec1 = mjesec;
        Godina1 = godina;
        Pstanje1 = pstanje;
        Zstanje1 = zstanje;
    }
}

```

Nakon toga kreirati ćemo singleton klasu BazaPod u kojoj će nakon prijave biti pohranjeni podaci korisnika.

```

12 references
public sealed class BazaPod
{
    private List<Klijent> sviKlijenti;
    private List<Dan> sviDani;
    private static BazaPod instanca = null;
    private Context context;
    private int stavljeno = 0;
    private int Ulogiranizaposlenik;

    1 reference
    public int Stavljeno { get => stavljeno; }
    3 references
    public int ulogiranizaposlenik { get => Ulogiranizaposlenik; set => Ulogiranizaposlenik = value; }

    1 reference
    public List<Klijent> getSviKlijenti() { return sviKlijenti; }
    2 references
    public List<Dan> getSviDani() { return sviDani; }

    4 references
    public static BazaPod getInstanca()
    {
        if (instancja == null)
        {
            instanca = new BazaPod();
        }
    }
}

```

Kako bi dohvatili podatke iz baze kreirati ćemo klase Podaci, Konekcija i Async. U klasi Async definirati ćemo BackgroundWorker koji će se koristiti u klasi Podaci kako bi se podaci iz baze mogli istovremeno preuzimati.

```

private BackgroundWorker bw;

0 references
public Async()
{
    bw = new BackgroundWorker();
    bw.DoWork += (s, e) => { DoInBackground(); };
    bw.RunWorkerCompleted += (s, e) => { onPostExecute(); };
}

3 references
protected abstract void PreExecute(String link);
3 references
protected abstract void DoInBackground();
3 references
protected abstract string onPostExecute();

5 references
public void Execute(String link)
{
    PreExecute(link);
    bw.RunWorkerAsync();
}

```

Klasa Podaci pomoću klase Konekcija dohvaća i čita podatke iz baze.

```

21 references
public class Podaci : Async
{
    private Context context;
    private int zadatak;
    private string url;
    private List<String> rez = new List<String>();
    4 references
    public Podaci(Context context, int zadatak)
    {
        this.context = context;
        this.zadatak = zadatak;
    }
    2 references
    protected override void DoInBackground()
    {
        Konekcija con = new Konekcija();
        con.Zadatak = zadatak;
        switch (this.zadatak)
        {
            case 1:
                rez.Add(con.DohvatiPodatke(url));
                break;
            case 2:
                String podaci = con.DohvatiPodatke(url);
                String d = con.SviKlijenti(podaci);
                rez.Add(d);
                break;
            case 3:
                String podaci2 = con.DohvatiPodatke(url);

```

```

3 references
public String DohvatiPodatke(String link)
{
    String odgovor = "";
    try
    {
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(link);
        HttpWebResponse response = (HttpWebResponse)request.GetResponse();
        Stream dataStream = response.GetResponseStream();
        StreamReader reader = new StreamReader(dataStream);
        string responseFromServer = reader.ReadToEnd();
        odgovor = responseFromServer;
    }
    catch (Exception e)
    {
        odgovor = "error";
    }
    return odgovor;
}

```

U klasi Konekcija osim funkcije za dohvaćanje podataka definirane su i funkcije za razdvajanje i spremanje podataka u liste.

```

2 references
private List<String> RazdvojiPodatke(String podaci)
{
    List<String> razdvojeniPodaci = new List<String>(podaci.Split("/"));
    return razdvojeniPodaci;
}
1 reference

1 reference
public String SviKlijenti(String podaci)
{
    List<String> razdvojeniPodaci = new List<String>();
    razdvojeniPodaci = RazdvojiPodatke(podaci);
    BazaPod con = BazaPod.getInstanca();
    if (con.Stavljeno == 0)
    {
        con.StaviSveKlijente(razdvojeniPodaci);
    }
    String t = "OK";
    return t;
}

1 reference
public List<String> SviDani(String podaci)
{
    List<String> razdvojeniPodaci = new List<String>();
    razdvojeniPodaci = RazdvojiPodatke(podaci);
    return razdvojeniPodaci;
}

```

U početnoj aktivnosti(MainActivity) kada korisnik klikne na gumb za prijavu prvo se provjeravaju uneseni podaci.


```

private async void prijava(object sender, EventArgs e)
{
    String u = username.Text;
    String p = password.Text;
    String link1 = "http://crofi.com/assets/assets/images/RasporedBaza/Logiranje.php?username=" + u + "&password=" + p;
    String link2 = "http://crofi.com/assets/assets/images/RasporedBaza/Svilijenti.php";
    String result = null;
    try
    {
        Podaci logiranje= new Podaci(this, 1);
        int r = 0;
        int l = 0;
        TimeSpan vrijeme = TimeSpan.FromMilliseconds(10);
        Task t = Task.Run(() =>
        {
            logiranje.Execute(link1);
        });
        while (r == 0)
        {
            if (logiranje.getRez().Count == 0)
            {
                await Task.Delay(vrijeme);
            }
            else {
                result = logiranje.getRez()[0];
                if (result == "error" || result == "No" || result == null)
                {
                    l = 0;
                }
                else
                {
                    con.uloginizaposlenik = Int32.Parse(result);
                    l = 1;
                }
                r = 1;
            }
        }
    }
}

```

Ukoliko su podaci netočni ispisuje se obavijest korisniku, u suprotnome dohvaćaju se svi klijenti i svi prethodno uneseni dani korisnika.

```

if (l == 0)
{
    Toast.MakeText(Application.Context, "Pogrešni podaci", ToastLength.Short).Show();
}

else {
    Podaci dohvatSvihKlijenta = new Podaci(this, 2);
    Podaci dohvatSvihDana = new Podaci(this, 3);
    String link3 = "http://crofi.com/assets/assets/images/RasporedBaza/SviDani.php?id="+con.uloginizaposlenik;
    r = 0;
    Task t2 = Task.Run(() =>
    {
        dohvatSvihKlijenta.Execute(link2);
        dohvatSvihDana.Execute(link3);
    });
    while (r == 0)
    {
        if (dohvatSvihKlijenta.getRez().Count == 0 || dohvatSvihDana.getRez().Count == 0)
        {
            await Task.Delay(vrijeme);
        }
        else
        {
            List<String> Daniresult = dohvatSvihDana.getRez();
            con.StaviSveDane(Daniresult);
            List<Dan> sviKlijenti = con.getSviDani();
            r = 1;
        }
    }

    var intent = new Intent(this, typeof(KreiranjeDana));
    StartActivity(intent);
}
}

```

Aplikacija čeka da završi dohvaćanje podataka o klijentima kako bi mogla spremiti podatke o danima u listu. Nakon što je preuzimanje podataka završeno korisnika se preusmjerava na sljedeće sučelje.

Kreiranje dana

Nakon uspješne prijave korisnik je preusmjeren na sučelje za kreiranje dana. Zadatak jest dizajnu sučelja pridružiti sljedeće funkcionalnosti:

- Odabir datuma
- Kreiranje popisa običdenih klijenata(dodavanje i brisanje)
- Izračun i ispis prijeđenih kilometara
- Spremanje podataka u bazu

1.11.2. Odabir datuma

Kada korisnik klikne na gumb za promjenu datuma, otvoriti će mu se kalendar na kojem će odabrati datum te će se odabrani datum potom ispisati u elementu za prikaz teksta. Aplikacija također mora provjeriti postoji li već definirani popis običenih klijenata.

1.11.2.1. Android Studio

U aktivnosti za kreiranje dana(KreiranjeDana) definiramo funkciju odaberiDatum().

```
public void odaberiDatum(View view){
    Calendar calendar = Calendar.getInstance();
    int year = calendar.get(Calendar.YEAR);
    int month = calendar.get(Calendar.MONTH);
    int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);
    DatePickerDialog datePickerDialog = new DatePickerDialog(context KreiranjeDana.this,
        (datePicker, year, month, day) -> {
            month=month+1;
            String datumDana=day + "/" + month + "/" + year;
            t1.setText(datumDana);
            provjeriKorisnikeDatuma(datumDana);
        }, year, month, dayOfMonth);
    datePickerDialog.show();
}
```

Funkcija će se pozivati kada korisnik klikne na gumb za promjenu datuma.

```
public void provjeriKorisnikeDatuma(String dat) {
    Dan d=con.provjeriDatum(dat);
    users = new ArrayList<String>();
    users2= new ArrayList<Klijent>();
    for(int i=0;i<d.getPopis().size();i++){
        int x=i+1;
        String id=x+ ".";
        users.add(id+d.getPopis().get(i).getPrezime());
        users2.add(d.getPopis().get(i));
    }
    updateSpinner();
    String km=d.getKm();
    t2.setText(km);
}

public void updateSpinner(){
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(context this, android.R.layout.simple_spinner_item, users);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spin.setAdapter(adapter);
}
```

Prije nego se ispiše odabrani datum pomoću funkcije provjeriKorisnikeDatuma() provjeravaju se podaci u bazi te ukoliko postoji definiran popis običenih klijenata on se dohvaća i kreira se lista sa prezimenima klijenata koja se pridružuje padajućem izborniku na sučelju.

```
public Dan provjeriDatum(String datum){
    for(int i=0;i<sviDani.size();i++){
        String dat=sviDani.get(i).getDatum();
        if(datum.equals(dat)){
            return sviDani.get(i);
        }
    }
    List<Klijent> popis=new ArrayList<Klijent>();
    Dan d= new Dan( id: "", datum, pstanje: "0", zstanje: "0", km: "0", popis, mjesec: 1, godina: 2022);
    sviDani.add(d);
    return d;
}
```

Također se dohvaćaju i ispisuju prijedeni kilometri za taj datum(ako je definiran popis).

1.11.2.2. Xamarin

Kako bi se kalendar mogao uspješno prikazati prvo kreiramo klasu DatumOdabir u kojoj definiramo prikaz kalendara i vraćanje odabranog datuma.

```

public class DatumOdabir : DialogFragment, DatePickerDialog.IOnDateSetListener
{
    public static readonly string TAG = "";
    Action<DateTime> odabraniDatum = delegate { };
    1 reference
    public static DatumOdabir NewInstance(Action<DateTime> onDateSelected)
    {
        DatumOdabir frag = new DatumOdabir();
        frag.odabraniDatum = onDateSelected;
        return frag;
    }
    0 references
    public override Dialog OnCreateDialog(Bundle savedInstanceState)
    {
        DateTime currently = DateTime.Now;
        DatePickerDialog dialog = new DatePickerDialog(Activity,
                                                    this,
                                                    currently.Year,
                                                    currently.Month - 1,
                                                    currently.Day);

        return dialog;
    }
    0 references
    public void OnDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth)
    {
        DateTime selectedDate = new DateTime(year, monthOfYear + 1, dayOfMonth);
        odabraniDatum(selectedDate);
    }
}

```

Nakon što korisnik klikne na gumb za odabir datuma pozvati će se funkcija `odaberiDatum()` koja poziva klasu `DatumOdabir` te provjerava da li je taj datum već definiran.

```

public void odaberiDatum(object sender, EventArgs e)
{
    var odabirDat = DatumOdabir.NewInstance(delegate (DateTime vrijeme) {
        string datumDana = vrijeme.Day + "/" + vrijeme.Month + "/" + vrijeme.Year;
        t1.Text = datumDana;
        provjeriKorisnikeDatuma(datumDana);
    });
    odabirDat.Show(FragmentManager, DatumOdabir.TAG);
}

```

Ako je korisnik već unio podatke o danu, odnosno postoji popis o običenim klijentima, on se dohvaća i popis se ispisuje u padajući izbornik.

```

public void provjeriKorisnikeDatuma(string dat) {
    Dan d = con.provjeriDatum(dat);
    users = new List<String>();
    users2 = new List<Klijent>();
    for (int i = 0; i < d.Popis1.Count; i++)
    {
        int x = i + 1;
        String id = x + ".";
        users.Add(id + d.Popis1[i].Prezime);
        users2.Add(d.Popis1[i]);
    }
    updateSpinner();
    string km = d.Km1;
    t2.Text = km;
}

```

1.11.3. Dodavanje korisnika

Kada korisnik klikne na gumb za dodavanje klijenata aplikacija otvara prozor u kojem se nalazi padajući izbornik sa svim klijentima te korisnik može izabrati klijenta i dodati ga u popis za taj dan.

1.11.3.1. Android Studio

Kreirano sučelje (iskočnom prozoru za dodavanje korisnika) prikazivati ćemo pomoću klase `DodavanjeKlijenta`.

```

public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder= new AlertDialog.Builder(getActivity());
    LayoutInflater inflater=getActivity().getLayoutInflater();
    View v=inflater.inflate(R.layout.dodavanje_klijenta, root: null);

    con=Baza.getInstance();
    K=con.getSviKlijenti();
    for(int i=0;i<K.size();i++){
        users.add(K.get(i).getPrezime());
    }
    popis=(Spinner) v.findViewById(R.id.sviKorisnici);
    dodaj=(Button) v.findViewById(R.id.dod);
    ArrayAdapter<String> adapter = new ArrayAdapter<>(v.getContext(), android.R.layout.simple_spinner_item, users);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    popis.setAdapter(adapter);
    dodaj.setOnClickListener((v) -> { dodavanjeKorisnika(); });
    builder.setView(v)
        .setTitle("Odaberi klijente:")
        .setNegativeButton( text: "Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        })
        .setPositiveButton( text: "Spremi", (dialog, which) -> {
            listener.dodajKorisnike(DodUsers);
        });

    return builder.create();
}

```

Iz baze se dohvaćaju svi klijenti te kada korisnik odabere klijenta klikne na gumb za dodavanje i aktivira funkciju dodavanjeKorisnika() koja u zasebnu listu pohranjuje dodanog klijenta.

```

public void dodavanjeKorisnika(){
    Long prezime=popis.getSelectedItemId();
    int i=prezime.intValue();
    DodUsers.add(K.get(i));
}

```

Kada je korisnik dodao sve klijente mora kliknuti na gumb OK sa kojim završava dodavanje te se zatvara iskočni prozori i prosljeđuje se lista dodanih klijenata(ako korisnik klikne na gumb Cancel lista se ne prosljeđuje).

```

public void unesiKlijente(){
    DodavnjeKlijenta dk=new DodavnjeKlijenta();
    dk.show(getSupportFragmentManager(), tag: "Izaberi klijenta");
}

```

Iskočni prozor odnosno sučelje za dodavanje klijenata prikazuje se nakon što korisnik klikne na gumb za dodavanje koji aktivira funkciju unesiKlijente. Nakon što se prozor zatvori lista dodanih klijenata se prosljeđuje funkciji dodajKorisnike() koja osvježava padajući izbornik sa popisom klijenata.

```

public interface ExampleDialogListener {
    void dodajKorisnike(ArrayList<Klijent> Dodani);
}

} public void dodajKorisnike(ArrayList<Klijent> Dodani) {
    int pocetni=users.size();
    for(int i=0;i<Dodani.size();i++){
        pocetni=pocetni+1;
        String id=String.valueOf(pocetni);
        users.add(id+"."+Dodani.get(i).getPrezime());
        users2.add(Dodani.get(i));
    }
    updateSpinner();
}

```

1.11.3.2. Xamarin

Prvo kreiramo klasu koja će prikazivati prozor za dodavanje korisnika, u ovoj verziji nećemo dodane klijente prosljeđivati funkciji već će se nakon zatvaranja prozora dodani klijenti spremati u listu.

```
public override Dialog OnCreateDialog(Bundle savedInstanceState)
{
    AlertDialog.Builder builder = new AlertDialog.Builder(Activity);
    LayoutInflater inflater = Activity.LayoutInflater;
    con = BazaPod.getInstanca();
    K = con.getSviKlijenti();
    for (int i = 0; i < K.Count; i++)
    {
        users.Add(K[i].Prezime);
    }
    View v = inflater.Inflate(Resource.Layout.dodavanje_klijenta, null);
    dodaj = v.FindViewById<Button>(Resource.Id.dod);
    popis = v.FindViewById<Spinner>(Resource.Id.sviKorisnici);
    ArrayAdapter<string> adapter = new ArrayAdapter<string>(Activity, Android.Resource.Layout.SimpleSpinnerDropDownItem, users);
    adapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownItem);
    popis.Adapter = adapter;
    dodaj.Click += dodavanjeKorisnika;
    builder.SetView(v)
        .SetTitle("Odaberi Klijenta")
        .SetPositiveButton("Ok", (sender, args) =>
        {
            if (null != Dismissed)
                Dismissed(this, new DialogEventArgs { DodaniKlijenti = DodUsers });
        })
        .SetNegativeButton("Cancel", (sender, args) =>
        {
            Console.WriteLine("Korisnik kliknuo Cancel");
        });
    return builder.Create();
}

public void dodavanjeKorisnika(object sender, EventArgs e) {
    long index = popis.SelectedItemId;
    int i = Convert.ToInt32(index);
    DodUsers.Add(K[i]);
}
```

Prozor za dodavanje klijenata otvara se nakon što korisnik klikne na gumb za unos klijenta.

```
1 reference
public void unesiKlijente(object sender, EventArgs e)
{
    var transaction = FragmentManager.BeginTransaction();
    var dialogFragment = new DodavanjeKlijenta();
    dialogFragment.Show(transaction, "Izaberi klijenta");
    dialogFragment.Dismissed += (s, e) => {
        dodajKorisnike(e.DodaniKlijenti);
    };
}
```

Nakon što su svi klijenti dodani i korisnik je potvrdio popis, dohvaća se popis svih dodanih klijenata i prosljeđuje se funkciji dodajKorisnike() koja osvježava padajući izbornik sa popisom klijenata.

```
1 reference
public async void dodajKorisnike(List<Klijent> Dodani)
{
    int pocetni = users.Count;
    for (int i = 0; i < Dodani.Count; i++)
    {
        pocetni = pocetni + 1;
        users.Add(pocetni + "." + Dodani[i].Prezime);
        users2.Add(Dodani[i]);
    }
    updateSpinner();
}
```

1.11.4. Brisanje Korisnika

Korisnik ima opciju da iz padajućeg izbornika može odabrati korisnika i obrisati ga iz popisa.

1.11.4.1. Android Studio

Kada korisnik odabere klijenta kojeg želi maknuti iz popisa, klikne na gumb Izbriši koji aktivira funkciju izbrisiKorisnika().

```
public void izbrisiKorisnika(){
    Long broj=spin.getSelectedItemId();
    int id=broj.intValue();
    if(users.size()!=1 || users.size()!=0) {
        for (int i = 0; i < users.size(); i++) {
            if (i == id) {
                users2.remove(i);
                users.remove(i);
            }
        }
        for (int i = 0; i < users.size(); i++) {
            int noviBroj = i + 1;
            String kor = users2.get(i).getPrezime();
            String novi = noviBroj + "." + kor;
            users.set(i, novi);
        }
    }
    else{
        users = new ArrayList<String>();
        users2 = new ArrayList<Klijent>();
    }
    updateSpinner();
}
```

Cilj funkcije je izbrisati klijenta iz liste i ostale klijente pomaknuti na odgovarajuće mjesto u popisu, nakon što je klijent izbrisan i korisnici pomaknuti osvježava se padajući izbornik.

1.11.4.2. Xamarin

Kao i u prethodnoj verziji nakon što korisnik odabere klijenta klikne na gumb za brisanje koji poziva funkciju izbrisiKorisnika().

```
public async void izbrisiKorisnika(object sender, EventArgs e) {
    long broj = spin.SelectedItemId;
    int id = Convert.ToInt32(broj);
    if (users.Count != 1 || users.Count != 0)
    {
        for (int i = 0; i < users.Count; i++)
        {
            if (i == id)
            {
                users2.RemoveAt(i);
                users.RemoveAt(i);
            }
        }
        for (int i = 0; i < users.Count; i++)
        {
            int noviBroj = i + 1;
            String kor = users2[i].Prezime;
            String novi = noviBroj + "." + kor;
            users[i] = novi;
        }
    }
    else
    {
        users = new List<String>();
        users2 = new List<Klijent>();
    }
    updateSpinner();
}
```

Preostali klijenti se pomiču za jedno mjesto i padajući izbornik se osvježava.

1.11.5. Izračun i ispis prijeđenih kilometara

Prijeđeni broj kilometara će se izračunavati pomoću google api servisa distancematrix kojem će se prosljeđivati koordinate adresa klijenata i on će na temelju podataka vraćati udaljenost u kilometrima.

1.11.5.1. Android Studio

Kreiramo novu klasu pod nazivom IzracunavanjeKm koja će komunicirati sa servisom.

```
public class IzracunavanjeKm extends AsyncTask {  
  
    String pLat, Plon, zLat, Zlon;  
  
    public IzracunavanjeKm() {  
    }  
  
    protected void onPreExecute() {  
    }  
  
    @Override  
    protected String doInBackground(Object[] objects) {  
        this.pLat=objects[0].toString();  
        this.Plon=objects[1].toString();  
        this.zLat=objects[2].toString();  
        this.Zlon=objects[3].toString();  
        String link="https://maps.googleapis.com/maps/api/distancematrix/json?destinations="+zLat+"%2C"+Zlon+"&origins="+pL  
        String rez="";  
    }  
}
```

Klasa prvo prima koordinate polazišta i odredišta te ih prosljeđuje servisu. Nakon što servis primi podatke izračunava se udaljenost te izračunava udaljenost između 2 mjesta. Udaljenost se ispisuje u obliku Json objekta koji se dohvaća i čita pomoću BufferedReader-a. Nakon što je pročitana udaljenost između mjesta, klasa vraća rezultat. Ukoliko se dogodi greška kod komunikacije sa servisom vrati će se string sa tekstom koji označava grešku.

```
try {  
    URL url=new URL(link);  
    HttpURLConnection con= (HttpURLConnection) url.openConnection();  
    con.setRequestMethod("GET");  
    con.connect();  
    int statuscode=con.getResponseCode();  
    if(statuscode==HttpURLConnection.HTTP_OK)  
    {  
        BufferedReader br=new BufferedReader(new InputStreamReader(con.getInputStream()));  
        StringBuilder sb=new StringBuilder();  
        String line=br.readLine();  
        while(line!=null)  
        {  
            sb.append(line);  
            line=br.readLine();  
        }  
        String json=sb.toString();  
        JSONObject root=new JSONObject(json);  
        JSONArray array_rows=root.getJSONArray( name: "rows");  
        JSONObject object_rows=array_rows.getJSONObject( index: 0);  
        JSONArray array_elements=object_rows.getJSONArray( name: "elements");  
        JSONObject object_elements=array_elements.getJSONObject( index: 0);  
        JSONObject object_distance=object_elements.getJSONObject("distance");  
        rez= object_distance.getString( name: "text");  
    }  
} catch (MalformedURLException e) {  
    rez="erl";  
}
```

Nakon toga kreiramo klasu KorisniciLokacije koja će prolaziti kroz popis klijenata, dohvaćati njihove koordinate i pomoću klase IzracunavanjeKm izračunavati udaljenosti između klijenata koji su na popisu.

```

public class KorisniciLokacije {
    private double kilometri=0;
    public String TraziKorisnike( ArrayList<Klijent> Klijenti){
        Object duljina= new Object();
        for(int i=0;i<Klijenti.size()-1;i++){
            String pLat=Klijenti.get(i).getLat();
            String Plon=Klijenti.get(i).getLon();
            String zLat=Klijenti.get(i+1).getLat();
            String Zlon=Klijenti.get(i+1).getLon();
            try {
                duljina=new IzracunavanjeKm().execute(pLat, Plon, zLat, Zlon).get();
                String km=duljina.toString().replace( target: "km", replacement: "");
                double d=Double.parseDouble(km);
                kilometri=kilometri+d;
            } catch (ExecutionException e) {
                e.printStackTrace();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        String result = String.format("%.2f", kilometri);
        return result;
    }
}

```

Kada je funkcija prošla kroz cijeli popis klijenata vrati će ukupnu udaljenost u kilometrima. Ovu klasu ćemo ukomponirati u funkcije za dodavanje(dodajKorisnike) i brisanje klijenata(izbrisiKorisnika). Tako da svaki puta kad korisnik dodaje ili briše klijente iz popisa, klasi KorisniciLokacije se proslijeđuje nova lista klijenata i automatski se osvježava ukupno prijeđena udaljenost.

```

KorisniciLokacije kl= new KorisniciLokacije();
String km=kl.TraziKorisnike(users2);
t2.setText(km);

```

Te se ispisuje na sučelje u odgovarajući element za prikaz teksta.

1.11.5.2. Xamarin

Kreiramo klasu IzracunavanjeKm koja će primati koordinate odredišta i polazišta i pomoću google api servisa izračunavati i vraćati udaljenost u kilometrima.

```

public class IzracunavanjeKm : Async
{
    string pLat, Plon, zLat, Zlon;
    String rez = "No";
    public IzracunavanjeKm(string pLat, string Plon, string zLat, string Zlon) {
        this.pLat = pLat;
        this.Plon = Plon;
        this.zLat = zLat;
        this.Zlon = Zlon;
    }
    protected override void DoInBackground()
    {
        string link = @"https://maps.googleapis.com/maps/api/distancematrix/json?destinations=" + zLat + "%2C" + Zlon + "&origins=" + pLat + "%2C" + Plon + "&k";
        try
        {
            WebRequest request = WebRequest.Create(link);
            WebResponse response = request.GetResponse();
            Stream data = response.GetResponseStream();
            StreamReader reader = new StreamReader(data);
            string responseFromServer = reader.ReadToEnd();
            response.Close();
            JObject json = JObject.Parse(responseFromServer);
            string rezultat;
            rezultat = json.SelectToken("rows[0].elements[0].distance.text").ToString();
            rez = rezultat;
        }
        catch (Exception ex)
        {
            rez = "error";
        }
    }
}

```


Ako se dogodi greška vraća se string „error“. Ovu klasu ćemo implementirati u klasu KorisniciLokacije koja će primati listu klijenata i dohvaćati njihove koordinate.

```

        IzracunavanjeKm duljina = new IzracunavanjeKm(pLat, Plon, zLat, Zlon);

        int r = 0;
        int l = 0;
        TimeSpan vrijeme = TimeSpan.FromMilliseconds(1);
        Task t = Task.Run(() =>
        {
            duljina.Execute("");
        });
        while (r == 0)
        {
            if (duljina.getRez()=="No")
            {
                await Task.Delay(vrijeme);
            }
            else
            {
                String result= duljina.getRez();
                if (result == "error" || result == "No" || result == null)
                {
                    l = 0;
                }
                else
                {
                    string brojkm=result.Replace("km", "");
                    km = km+Convert.ToDouble(brojkm);
                    l = 1;
                }
                r = 1;
            }
        }
    }
    catch (Exception e)
    {
    }
}

kilometri = String.Format("{0:0.00}", km);

```

Kada dohvati koordinate proslijediti će ih klasi IzracunavanjeKm koja će vratiti udaljenost između dvije lokacije. Kao i u prošloj verziji klasa KorisniciLokacije koristiti će se u funkcijama izbrisiKorisnika() i u funkciji dodajKorisnike().

```

public async void dodajKorisnike(List<Klijent> Dodani)
{
    int pocetni = users.Count;
    for (int i = 0; i < Dodani.Count; i++)
    {
        pocetni = pocetni + 1;
        users.Add(pocetni + "." + Dodani[i].Prezime);
        users2.Add(Dodani[i]);
    }
    updateSpinner();
    KorisniciLokacije kl = new KorisniciLokacije();
    kl.TraziKorisnike(users2);
    string km;
    int l = 0;
    TimeSpan vrijeme = TimeSpan.FromMilliseconds(5);
    Task t = Task.Run(() => { });
    while (l == 0)
    {
        l = kl.Got;
        if (l == 0)
        {
            await Task.Delay(vrijeme);
        }
        else
        {
            km = kl.Kilometri;
            t2.Text = km;
        }
    }
}

```

1.11.6. Spremanje podataka u bazu

Podaci će se spremati ako korisnik klikne na gumb za spremanje i ako korisnik promjeni datum(automatski će se spremati na promjenu datuma.

Ovakav naziv poglavlja je dobar kada se radi komparativna analiza. Na žalost neka od prethodnih poglavlja možda treba preoblikovati ili im izmijeniti fokus.

1.11.6.1. Android Studio

U klasi KreiranjeDana kreiramo funkciju spremiDan() koja će promjeniti podatke dana u lokalnoj listi i u bazi podataka koja se nalazi na serveru.

```
public void spremiDan() {
    String datum=t1.getText().toString();
    String km=t2.getText().toString();
    con.updateDan(datum, km, users2);
    String lista="";
    for(int i=0;i<users2.size();i++){
        lista=lista+users2.get(i).getId();
        if(i!=users2.size()-1){
            lista=lista+",";
        }
    }
    String Datum = datum;
    String Poc = "0";
    String Zav = "0";
    String Pkm = km;
    String Popis = lista;
    Integer Mjesec = 10;
    Integer Godina = 2021;
    Integer Zaposlenik = con.getUlogiranzaposlenik();
    String link = "http://crofi.com/assets/assets/images/RasporedBaza/UpdateDan.php?Datum="+Datum+"&Poc="+Poc+"&Zav="+Zav+"&Pkm="+Pkm+"&Popis="+Popis+"&Mjesec="+Mjesec+"&Godina="+Godina+"&Zaposlenik="+Zaposlenik;
    Object result= null;
    try {
        result = new Podaci( context: this, statusField: null, zadatak: 4).execute(link).get();
    } catch (ExecutionException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Funkcija se poziva kada korisnik klikne na gumb za spremanje i kod odabira datuma.

```
odaberiDat.setOnClickListener ( (v) -> {
    spremiDan();
    odaberiDatum(v);
});
```

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch(item.getItemId()){
        case R.id.spremi:
            spremiDan();
            Toast.makeText( context: this, text: "Spremljene promjene", Toast.LENGTH_LONG).show();
            return true;
        case R.id.kreirajRas:
            kreirajRaspored();
            Toast.makeText( context: this, text: "Kreirana datoteka Raspored.pdf", Toast.LENGTH_LONG).show();
            return true;
        case R.id.odjavi:
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
            return true;
    }
    return true;
}
```

1.11.6.2. Xamarin

Kreirati ćemo funkciju spremanjeDatuma() koja će spremati sve promjene u lokalnim listama i u bazi podataka.

```

public async void spremanjeDana(object sender, EventArgs e)
{
    string datum = t1.Text;
    string km = t2.Text;
    con.updateDan(datum, km, users2);
    String lista = "";
    for (int i = 0; i < users2.Count; i++)
    {
        lista = lista + users2[i].Id;
        if (i != users2.Count - 1)
        {
            lista = lista + ",";
        }
    }
    string Datum = datum;
    string Poc = "0";
    string Zav = "0";
    string Pkm = km;
    string Popis = lista;
    int Mjesec = 10;
    int Godina = 2021;
    int Zaposlenik = con.uoginirizaposlenik;
    string link = "http://crofi.com/assets/images/RasporedBaza/UpdateDan.php?Datum=" + Datum + "&Poc=" + Poc + "&Zav=" + Zav + "&Pkm=" + Pkm;
    Podaci update = new Podaci(this, 1);
    int r = 0;
    int l = 0;
    TimeSpan vrijeme = TimeSpan.FromMilliseconds(2);
    Task t = Task.Run(() =>
    {
        update.Execute(link);
    });
    while (r == 0)
    {
        if (update.getRez().Count == 0)
        {
            await Task.Delay(vrijeme);
        }
    }
}

```

Funkcija se poziva kod promjene datuma i kod klika gumba za spremanje.

```

0 references
public override bool OnOptionsItemSelected(IMenuItem item)
{
    switch (item.ItemId)
    {
        case Resource.Id.spremi:
        {
            spremanjeDana();
            return true;
        }
    }
}

1 reference
public void odaberiDatum(object sender, EventArgs e)
{
    spremanjeDana();
}

```

1.11.7. Generiranje izvještaja

Kada korisnik klikne na opciju Kreiraj Raspored koja se nalazi u izborniku otvoriti će se posebni prozor u kojem korisnik odabire od kojeg do kojeg datuma će se generirati izvještaj i zatim se generira pdf izvještaj o zadanim danima. Izvještaj će se generirati u pdf obliku pomoću zasebne klase u kojoj će biti definiran dizajn izvještaja.

1.11.7.1. Android Studio

Kreiramo novu klasu pod nazivom Pdf. Prvo moramo odrediti koliko dana će izvještaj imati te koliko dana stane na jednu stranicu.

```

public PdfDocument Kreirajpdf(List<Dan> SortiraniDan){
    System.out.println("broj dana: "+SortiraniDan.size());
    Dani=SortiraniDan;
    PdfDocument pdf= new PdfDocument();
    int b=0;
    for(int i=0;i<Dani.size();i++){
        int brojLjudi=b+Dani.get(i).getPopis().size();
        System.out.println("broj ljudi u danu : "+Dani.get(i).getDatum()+" "+brojLjudi);
        if(brojLjudi<90) {
            b = b + Dani.get(i).getPopis().size();
        }
        else{
            b=0;
            System.out.println("dodani index: "+i);
            index.add(i);
        }
    }
    if(index.size()<1){
        index.add(Dani.size());
    }
    int stranice=index.size()+1;
    System.out.println("broj stranica: "+stranice);
    for(int i=0;i<index.size();i++) {
        if(i!=0) {
            KreirajStranicu( broj: i + 1, pdf, index.get(i-1), index.get(i));
        }
        else{
            KreirajStranicu( broj: i + 1, pdf, brojIndexaP: 0, index.get(i));
        }
    }
}

```

Potom kreiramo svaku stranicu u izvještaju.

```

public void KreirajStranicu(int broj, PdfDocument pdf, int brojIndexaP, int brojIndexaZ){
    PdfDocument.PageInfo pi= new PdfDocument.PageInfo.Builder( pageWidth: 1200, pageHeight: 2010,broj).create();
    PdfDocument.Page myPage=pdf.startPage(pi);
    int pageWidth=1200;
    int pageHeight=2010;
    Paint paint=new Paint();
    Paint title=new Paint();
    Canvas c= myPage.getCanvas();
    title.setTextAlign(Paint.Align.LEFT);
    title.setTypeface(Typeface.create(Typeface.DEFAULT,Typeface.BOLD));
    title.setTextSize(25);
    c.drawText( text: "USTANOVA ZA NJEGU U KUĆI DOMNIUS, JARUŠČICA 9E, ZAGREB", x: 150, y: 100, title);
    c.drawText( text: "EVIDENCIJA O KRETANJU PRIVATNOG AUTOMOBILA ZA RAZDOBLJE", x: 150, y: 160, title);
    c.drawText( text: "OD "+ Dani.get(0).getDatum()+" DO "+ Dani.get(Dani.size()-1).getDatum()+" godine U SLUŽI", x: 150, y: 220, title);
    c.drawText( text: "Korisnik: Andreja Stjepanović", x: 150, y: 250, title);
    c.drawText( text: "Marka automobila: Citroen", x: 150, y: 280, title);
    c.drawText( text: "Registarski broj automobila: 4864876584", x: 150, y: 310, title);
    //informacije u desnom kutu
    paint.setTypeface(Typeface.create(Typeface.DEFAULT,Typeface.BOLD));
    paint.setTextSize(30f);
    paint.setTextAlign(Paint.Align.RIGHT);
    c.drawText( text: "Call: 093530953", x: 1160, y: 40,paint);
    c.drawText( text: "Call: 093530953", x: 1160, y: 80,paint);
    //tablica
    title.setStyle(Paint.Style.STROKE);
    title.setStrokeWidth(3);
    c.drawRect( left: 20, top: 400, right: pageWidth-20, bottom: 480, title);
    title.setTextAlign(Paint.Align.LEFT);

    paint.setStyle(Paint.Style.FILL);
    c.drawText( text: "Datum", x: 60, y: 450,paint);
    c.drawText( text: "Naziv lokacija", x: 260, y: 450,paint);
    c.drawText( text: "Broj prijedjenih km", x: 570, y: 450,paint);
    c.drawText( text: "Izvješće o radu", x: 890, y: 450,paint);
    c.drawLine( startX: 20, startY: 400, stopX: 20, stopY: pageHeight-200,title);
    c.drawLine( startX: 180, startY: 400, stopX: 180, stopY: pageHeight-200,title);
    c.drawLine( startX: 550, startY: 400, stopX: 550, stopY: pageHeight-200,title);
    c.drawLine( startX: 830, startY: 400, stopX: 830, stopY: pageHeight-200,title);
    c.drawLine( startX: pageWidth-20, startY: 400, stopX: pageWidth-20, stopY: pageHeight-200,title);
    //podaci
    paint.setTextSize(25f);
    paint.setTypeface(Typeface.create(Typeface.DEFAULT,Typeface.NORMAL));
    int y=510;
    int pocetak=brojIndexaP;
    int zavrsetak=brojIndexaZ;
    for(int i=pocetak;i<zavrsetak;i++){
        y=y+10;
        String datumDana=Dani.get(i).getDatum();
        String kilometri=Dani.get(i).getKm();
        c.drawText(datumDana, x: 30,y,paint);
        c.drawText(kilometri, x: 580,y,paint);
        System.out.println("velicina popisa: "+Dani.get(i).getPopis().size());
    }
}

```

```

        if(Dani.get(i).getPopis().size()>3) {
            int red=0;
            String lokacije="";
            String text="";
            for (int j = 0; j < Dani.get(i).getPopis().size(); j++) {
                if(red<2&&j<Dani.get(i).getPopis().size()-1){
                    lokacije=lokacije+Dani.get(i).getPopis().get(j).getAdresa()+" ",
                    text=text+Dani.get(i).getPopis().get(j).getPrezime()+" ";
                    red++;
                }
                else{
                    red=0;
                    lokacije=lokacije+Dani.get(i).getPopis().get(j).getAdresa();
                    text=text+Dani.get(i).getPopis().get(j).getPrezime();
                    if(j!=Dani.get(i).getPopis().size()-1){
                        lokacije=lokacije+" ";
                        text=text+" ";
                    }
                    c.drawText(lokacije, x: 190, y, paint);
                    c.drawText(text, x: 840, y, paint);
                    text="";
                    lokacije="";
                    y = y + 35;
                }
            }
        }
    }

    else{
        String lokacije="";
        String text="";
        for (int j = 0; j < Dani.get(i).getPopis().size(); j++) {
            lokacije=lokacije+Dani.get(i).getPopis().get(j).getAdresa();
            text=text+Dani.get(i).getPopis().get(j).getPrezime();
            if(j!=Dani.get(i).getPopis().size()-1){
                lokacije=lokacije+" ";
                text=text+" ";
            }
        }
        c.drawText(lokacije, x: 190, y, paint);
        c.drawText(text, x: 840, y, paint);
        y = y + 35;
    }
    c.drawLine( startX: 20, y, stopX: pageWidth-20, y, title);
    y=y+20;
}

pdf.finishPage(myPage);

```

I vraćamo kreirani izveštaj. Zatim kreiramo klasu KreiranjeRasporeda koja će prikazivati i upravljati sučeljem za odabir datuma.

```

public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    LayoutInflater inflater = getActivity().getLayoutInflater();
    View v = inflater.inflate(R.layout.raspored, root: null);
    promjeni1=(Button)v.findViewById(R.id.promjenid1);
    promjeni2=(Button)v.findViewById(R.id.promjenid2);
    t1= (TextView) v.findViewById(R.id.dat1);
    t2=(TextView) v.findViewById(R.id.dat2);
    SimpleDateFormat formatter = new SimpleDateFormat( pattern: "dd/MM/yyyy");
    Date date = new Date();
    String d=formatter.format(date);
    t1.setText(d);
    t2.setText(d);
    promjeni1.setOnClickListener( (v) -> { odaberiDatum(v, dat: 1); });
    promjeni2.setOnClickListener( (v) -> { odaberiDatum(v, dat: 2); });
    builder.setView(v)
        .setTitle("Odaberi datum")
        .setNegativeButton( text: "Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }
        })
        .setPositiveButton( text: "Kreiraj", (dialog, which) -> {
            d1=t1.getText().toString();
            d2=t2.getText().toString();
            listener.proslijediDatum(d1,d2);
        });
    return builder.create();
}

```

Kada korisnik klikne na gumb za promjenu određenog datuma otvara mu se kalendar u kojem odabire željeni datum te se zapisuje u element za prikaz teksta.

```
public void odaberiDatum(View view, int dat){
    Calendar calendar = Calendar.getInstance();
    int year = calendar.get(Calendar.YEAR);
    int month = calendar.get(Calendar.MONTH);
    int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);
    final int x=dat;
    DatePickerDialog datePickerDialog = new DatePickerDialog(this.getContext(),
        (datePicker, year, month, day) -> {
            month=month+1;
            String datumDana=day + "/" + month + "/" + year;
            if(x==1){
                t1.setText(datumDana);
            }
            else{
                t2.setText(datumDana);
            }
        }, year, month, dayOfMonth);

    datePickerDialog.show();
}
```

Kada je korisnik završio sa odabirom datuma mora kliknuti na gumb Kreiraj u otvorenom prozoru(ako klikne na Cancel datumi se poništavaju). Potom se odabrani datumi proslijeđuju funkciji proslijediDatume() koja se nalazi u klasi KreiranjeDana

```
public interface ExampleDialogListener {
    void proslijediDatume(String d1, String d2);
}
```

Nakon što se odabrani datumi proslijede funkciji ona ih prvo sortira pomoću funkcije DaniRaspored() koja vraća listu sortiranih dana na temelju početnog i završnog datuma.

```
public List<Dan> DaniRaspored(Date p, Date k){
    List<Dan> sviDani=con.getSviDani();
    List<Dan> Dani=new ArrayList<Dan>();
    final SimpleDateFormat form=new SimpleDateFormat( pattern: "dd/MM/yyyy");
    for(int i=0;i<sviDani.size();i++){
        try {
            Date d=form.parse(sviDani.get(i).getDatum());
            if(d.after(p) && d.before(k)){
                Dani.add(sviDani.get(i));
            }
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
    for (int i = 0; i < Dani.size()-1; i++)
        for (int j = 0; j < Dani.size()-i-1; j++) {
            try {
                Date d=form.parse(Dani.get(j).getDatum());
                Date d2=form.parse(Dani.get(j+1).getDatum());
                if (d.after(d2))
                {
                    Dan dan=Dani.get(j);
                    Dani.set(j, Dani.get(j+1));
                    Dani.set(j+1, dan);
                }
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }
    return Dani;
}
```

```

public void proslijediDatum(String d1, String d2){
    System.out.println(d1+" "+d2);
    SimpleDateFormat form=new SimpleDateFormat( pattern: "dd/MM/yyyy");
    try {
        Date pocetak=form.parse(d1);
        Date kraj=form.parse(d2);
        List<Dan> SortiraniDani=DaniRaspored(pocetak, kraj);
        Pdf p=new Pdf();
        PdfDocument pdf= new PdfDocument();
        pdf=p.Kreirajpdf(SortiraniDani);
        File file=new File(Environment.getExternalStorageDirectory(), child: "/Raspored.pdf");
        pdf.writeTo(new FileOutputStream(file));
        pdf.close();
        Toast.makeText( context: KreiranjeDana.this, text: "Kreirana je datoteka pod imenom Raspored.pdf", Toast.LENGTH
    ) catch (IOException e) {
        e.printStackTrace();
    }
}

```

1.11.7.2. Xamarin

Kreiramo klasu Pdf u kojoj se nalaze funkcije Kreirajpdf() i KreirajStranicu().

```

public PdfDocument Kreirajpdf(List<Dan> SortiraniDan)
{
    Console.WriteLine("broj dana: " + SortiraniDan.Count);
    Dani = SortiraniDan;
    PdfDocument pdf = new PdfDocument();
    int b = 0;
    for (int i = 0; i < Dani.Count; i++)
    {
        int brojLjudi = b + Dani[i].Popis1.Count;
        Console.WriteLine("broj ljudi u danu : " + Dani[i].Datum1 + " " + brojLjudi);
        if (brojLjudi < 90)
        {
            b = b + Dani[i].Popis1.Count;
        }
        else
        {
            b = 0;
            Console.WriteLine("dodani index: " + i);
            index.Add(i);
        }
    }
    if (index.Count < 1)
    {
        index.Add(Dani.Count);
    }
    int stranice = index.Count + 1;
    for (int i = 0; i < index.Count; i++)
    {
        if (i != 0)
        {
            KreirajStranicu(i + 1, pdf, index[i - 1], index[i]);
        }
        else
        {
            KreirajStranicu(i + 1, pdf, 0, index[i]);
        }
    }
    KreirajStranicu(stranice, pdf, index[index.Count - 1], Dani.Count);
    return pdf;
}

```

Funkciji Kreirajpdf() proslijeđuje se lista sortiranih dana od čijih podataka će se kreirati izvještaj. S obzirom na broj dana kreiraju se stranice izvještaja te svaka stranica ima određen broj podataka, svaka stranica kreira se pomoću funkcije KreirajStranicu().

```

public void KreirajStranicu(int broj, PdfDocument pdf, int brojIndexaP, int brojIndexaZ)
{
    PdfDocument.PageInfo pi = new PdfDocument.PageInfo.Builder(1200, 2010, broj).Create();
    PdfDocument.Page myPage = pdf.StartPage(pi);
    int pageWidth = 1200;
    int pageHeight = 2010;
    //naslov
    Paint paint = new Paint();
    Paint title = new Paint();
    Canvas c = myPage.Canvas;
    title.TextAlign = Paint.Align.Left;
    title.SetTypeface(Typeface.DefaultBold);
    title.TextSize = 25;
    c.DrawText("USTANOVA ZA NJEGU U KUĆI DOMINIUS, JARUŠČICA 9E, ZAGREB", 150, 100, title);
    c.DrawText("EVIDENCIJA O KRETANJU PRIVATNOG AUTOMOBILA ZA RAZDOBLJE", 150, 160, title);
    c.DrawText("OD " + Dani[0].Datum1 + " DO " + Dani[Dani.Count - 1].Datum1 + " godine U SLUŽBENE SVRHE", 150, 190, title);
    c.DrawText("Korisnik: Andreja Stjepanović", 150, 250, title);
    c.DrawText("Marka automobila: Citroen", 150, 280, title);
    c.DrawText("Registarski broj automobila: 4864876584", 150, 310, title);
    //informacije u desnom kutu
    paint.SetTypeface(Typeface.DefaultBold);
    paint.TextSize = 30f;
    paint.TextAlign = Paint.Align.Right;
    c.DrawText("Call: 093530953", 1160, 40, paint);
    c.DrawText("Call: 093530953", 1160, 80, paint);
}

```

Na svakoj stranici prvo se moraju ispisati osnovne informacije o izvještaju te se potom podaci o danima ispisuju u tablici.

```

//tablica
title.SetStyle(Paint.Style.Stroke);
title.StrokeWidth = 3;
c.DrawRect(20, 400, pageWidth - 20, 480, title);
title.TextAlign = Paint.Align.Left;
title.SetStyle(Paint.Style.Fill);
paint.TextAlign = Paint.Align.Left;
paint.SetStyle(Paint.Style.Fill);
c.DrawText("Datum", 60, 450, paint);
c.DrawText("Naziv lokacija", 260, 450, paint);
c.DrawText("Broj prijedanih km", 570, 450, paint);
c.DrawText("Izvjješće o radu", 890, 450, paint);
c.DrawLine(20, 400, 20, pageHeight - 200, title);
c.DrawLine(180, 400, 180, pageHeight - 200, title);
c.DrawLine(550, 400, 550, pageHeight - 200, title);
c.DrawLine(830, 400, 830, pageHeight - 200, title);
c.DrawLine(pageWidth - 20, 400, pageWidth - 20, pageHeight - 200, title);

for (int i = pocetak; i < zavrsetak; i++)
{
    y = y + 10;
    String datumDana = Dani[i].Datum1;
    String kilometri = Dani[i].Km1;
    c.DrawText(datumDana, 30, y, paint);
    c.DrawText(kilometri, 580, y, paint);
    Console.WriteLine("veličina popisa: " + Dani[i].Popis1.Count);
    if (Dani[i].Popis1.Count > 3)
    {
        int red = 0;
        String lokacije = "";
        String text = "";
        for (int j = 0; j < Dani[i].Popis1.Count; j++)
        {
            if (red < 2 && j < Dani[i].Popis1.Count - 1)
            {
                lokacije = lokacije + Dani[i].Popis1[j].Adresa1 + ", ";
                text = text + Dani[i].Popis1[j].Prezime + ", ";
                red++;
            }
            else
            {
                red = 0;
                lokacije = lokacije + Dani[i].Popis1[j].Adresa1;
                text = text + Dani[i].Popis1[j].Prezime;
                if (j != Dani[i].Popis1.Count - 1)
                {
                    lokacije = lokacije + ", ";
                    text = text + ", ";
                }
                c.DrawText(lokacije, 190, y, paint);
                c.DrawText(text, 840, y, paint);
                text = "";
                lokacije = "";
                y = y + 35;
            }
        }
    }
    else
    {
        String lokacije = "";
        String text = "";
        for (int j = 0; j < Dani[i].Popis1.Count; j++)
        {
            lokacije = lokacije + Dani[i].Popis1[j].Adresa1;
            text = text + Dani[i].Popis1[j].Prezime;
            if (j != Dani[i].Popis1.Count - 1)
            {
                lokacije = lokacije + ", ";
                text = text + ", ";
            }
        }
        c.DrawText(lokacije, 190, y, paint);
        c.DrawText(text, 840, y, paint);
        y = y + 35;
    }
    c.DrawLine(20, y, pageWidth - 20, y, title);
    y = y + 20;
}

pdf.FinishPage(myPage);

```

Kada su sve stranice završene vraća se pdf dokument. Zatim kreiramo novu klasu KreiranjeRasporeda sa kojom ćemo upravljati sučeljem za odabir datuma koji će se koristiti u kreiranju izvještaja.


```

public class KreiranjeRasporeda : DialogFragment
{
    2 references
    public class DialogEventArgs : EventArgs
    {
        2 references
        public String pocetni { get; set; }
        2 references
        public String zavrzni { get; set; }
    }
    public delegate void DialogEventHandler(object sender, DialogEventArgs args);
    Button promjeni1;
    Button promjeni2;
    TextView t1;
    TextView t2;
    public event DialogEventHandler Dismissed;
}

```

Klasa će imati dvije varijable pocetni i zavrzni u koje će se spremati odabrani datumi.

```

0 references
public override Dialog OnCreateDialog(Bundle savedInstanceState)
{
    AlertDialog.Builder builder = new AlertDialog.Builder(Activity);
    LayoutInflater inflater = Activity.LayoutInflater;
    View v = inflater.Inflate(Resource.Layout.kreiranje_rasporeda, null);
    promjeni1 = v.FindViewById<Button>(Resource.Id.promjeni1);
    promjeni2 = v.FindViewById<Button>(Resource.Id.promjeni2);
    t1 = v.FindViewById<TextView>(Resource.Id.dat1);
    t2 = v.FindViewById<TextView>(Resource.Id.dat2);
    DateTime trenutnoVrijeme = DateTime.Now;
    String datumTrenutno = trenutnoVrijeme.Day + "/" + trenutnoVrijeme.Month + "/" + trenutnoVrijeme.Year;
    t1.Text = datumTrenutno;
    t2.Text = datumTrenutno;
    promjeni1.Click += odaberiDatum1;
    promjeni2.Click += odaberiDatum2;
    builder.SetView(v)
        .SetTitle("Odaberi datum")
        .SetPositiveButton("Kreiraj", (sender, args) =>
        {
            if (null != Dismissed)
                Dismissed(this, new DialogEventArgs { pocetni = t1.Text, zavrzni = t2.Text });
        })
        .SetNegativeButton("Cancel", (sender, args) =>
        {
            Console.WriteLine("Korisnik kliknuo Cancel");
        });
    return builder.Create();
}

1 reference
public void odaberiDatum1(object sender, EventArgs e) {
    var odabirDat = DatumOdabir.NewInstance(delegate (DateTime vrijeme) {
        string datumDana = vrijeme.Day + "/" + vrijeme.Month + "/" + vrijeme.Year;
        t1.Text = datumDana;
    });
    odabirDat.Show(FragmentManager, DatumOdabir.TAG);
}

```

Svaki puta kad korisnik klikne na gumb za promjenu datuma, otvoriti će mu se kalendar te kada odabere datum on će se spremiti u jednu od varijabli(ovisno koji datum korisnik odabire). Zatim u klasi kreiranjeDana kreiramo funkciju KreirajRaspored() u kojoj će se kreirati objekt klase KreiranjeRasporeda. Funkcija će se pozivati kada korisnik klikne na gumb Kreiraj raspored u izborniku.

```

public void KreirajRaspored()
{
    var transaction = FragmentManager.BeginTransaction();
    var dialogFragment = new KreiranjeRasporeda();
    dialogFragment.Show(transaction, "Izaberi datum");
    dialogFragment.Dismissed += (s, e) => {
        proslijediDatum(e.pocetni, e.zavrzni);
    };
}

```

Nakon što je korisnik potvrdio odabir datuma na sučelju, sučelje se zatvara i datumi se proslijeđuju se funkcije proslijediDatum().

```

1 reference
public void prosljediDatum(String d1, String d2)
{
    Toast.makeText(Application.Context, "Kreirana datoteka Raspored2.pdf", ToastLength.Short).Show();
    DateTime pocetak = DateTime.Parse(d1, new System.Globalization.CultureInfo("pt-BR"));
    DateTime zavrsetak = DateTime.Parse(d2, new System.Globalization.CultureInfo("pt-BR"));
    List<Dan> SortiraniDani = DaniRaspored(pocetak, zavrsetak);
    Pdf p = new Pdf();
    PdfDocument pdf = new PdfDocument();
    pdf = p.Kreirajpdf(SortiraniDani);
    var IerapijaRaspored = pdf;
    Java.IO.File file = Android.OS.Environment.GetExternalStoragePublicDirectory(Android.OS.Environment.DirectoryDownloads);
    FileStream fileStream = new FileStream(file.AbsolutePath + "/Raspored2.pdf", FileMode.Create);
    pdf.WriteTo(fileStream);
    fileStream.Flush();
    fileStream.Close();
    pdf.Close();
}
1 reference

```

Funkcija `prosljediDatum()` prvo prosleđuje početni i završni datum funkciji `DaniRaspored()` koja na temelju odabranih datuma dohvaća i sortira sve dane koji spadaju u zadani vremenski okvir.

```

public List<Dan> DaniRaspored(DateTime p, DateTime k)
{
    List<Dan> sviDani = con.getSviDani();
    List<Dan> Dani = new List<Dan>();
    for (int i = 0; i < sviDani.Count; i++)
    {
        try
        {
            DateTime d = DateTime.Parse(sviDani[i].Datum1, new System.Globalization.CultureInfo("pt-BR"));
            int poslije = DateTime.Compare(d, p);
            int prije = DateTime.Compare(d, k);
            if (poslije > 0 && prije < 0)
            {
                Dani.Add(sviDani[i]);
            }
        }
        catch (Exception e)
        {
        }
    }
    for (int i = 0; i < Dani.Count - 1; i++)
        for (int j = 0; j < Dani.Count - i - 1; j++)
        {
            try
            {
                DateTime d = DateTime.Parse(Dani[j].Datum1, new System.Globalization.CultureInfo("pt-BR"));
                DateTime d2 = DateTime.Parse(Dani[j + 1].Datum1, new System.Globalization.CultureInfo("pt-BR"));
                int poslije = DateTime.Compare(d, d2);
                if (poslije > 0)
                {
                    Dan dan = Dani[j];
                    Dani[j] = Dani[j + 1];
                    Dani[j + 1] = dan;
                }
            }
            catch (Exception e)
            {
            }
        }
    return Dani;
}

```

I zatim vraća listu sortiranih dana. Potom se vraćena lista prosleđuje u funkciju `Kreirajpdf()` te kada funkcija vrati kreirani pdf izvještaj sprema se pod nazivom `Raspored2.pdf`.

Usporedba

Trebalo bi detaljnije opisati način kako su kriteriji nastali. Idealno bi bilo da su to kriteriji koji se uobičajeno koriste pri usporedbi. Bilo bi dobro imati autora koji pobrojava kriterije koje ste kreirali.

Usporedbu ću napraviti pomoću **nekoliko kriterija** te će usporedba biti iz osobnog iskustva paralelnog kreiranja android aplikacije u oba razvojna okruženja, još jednom napominjem da u ovom slučaju naziv Xamarin se odnosi na korištenje Xamarin.Android alata u Visual Studio-u.

Kriteriji	Xamarin	Android Studio	Pojašnjenje
Programski jezik	C#	Java	U ovom projektu osobno mi nije predstavljao problem korištenja jednog od ovih jezika jer sam relativno upoznata sa oba, ali nekome tko nije vjerojatno će ovaj kriterij biti ključan kod izbora.
Dokumentacija	10/10	9/10	Bitno je napomenuti da oba alata imaju vrlo detaljno razrađenu dokumentaciju, ali ono što sam primijetila kod korištenja dokumentacije Android Studio-a je da više puta neke stvari su objašnjene korak po korak ali nedostaje neki ključan korak kako bi funkcionalnost proradila pa je potrebno potražiti dodatnu dokumentaciju u knjigama, člancima ili forumima, a kod korištenja Xamarin-a to se događalo puno rjeđe.
Emulatori	5/10	8/10	Pretpostavljam da korisnici koji imaju računala sa više RAM-a, nemaju ovakvih problema ali nažalost pošto ja nemam preporučam svima koji su u istoj situaciji da aplikaciju testiraju na fizičkom mobilnom uređaju(ili više njih) jer iako je Android Studio u ovom slučaju odnio pobjedu svejedno se može reći da je emulator spor, a u slučaju Xamarin-a može se reći da je bolno spor. Tehnički oba emulatora rade i aplikacija se može testirati na više uređaja ali mojem slučaju polako i strašno polako.

Veličina aplikacije	6/10	10/10	Kada su aplikacije instalirane na istom uređaju(ponavljam, koriste iste resurse i imaju iste funkcionalnosti), Android Studio verzija aplikacije zauzima 2.86 MB interne memorije uređaja, a Xamarin verzija zauzima 99.48MB. Je li to u današnje vrijeme uistinu problem je za raspravu pošto prosječni mobilni uređaj ima minimalno 32GB(u većini slučajeva i više) interne memorije. Osobno na mobilnom uređaju(128GB) imam instalirano 42 aplikacije, 2508 slika, 230 videa, 481 ostalih datoteka i tek imam potrošeno 34GB memorije.
Brzina pokretanja	8/10	10/10	Ukoliko aplikacija nije instalirana na mobilnom uređaju, i kliknete na gumb za pokretanje, Android Studio verziji trebati će otprilike 10-12 sekundi za pokretanje ili ukoliko je već instalirana 3-4 sekunde, a Xamarin verziji 16-19 sekundi ili 4-6 ako je prethodno instalirana.
Pojava grešaka	9/10	7/10	Xamarin je u ovom slučaju (barem što se tiče mog iskustva) pobijedio. Tijekom izrade Xamarin verzije aplikacije svi vanjski dodaci aplikaciji(većinom NuGet paketi) integrirali su se sa puno manje grešaka dok je u Android Studio verziji znalo doći do puno više problema, pogotovo kod usklađivanja Android verzija aplikacije. Također dosta puta mi se dogodilo da nije prepoznata neka metoda ili klasa iz dodanih biblioteka(ponekad i klase i metode iz samog projekta) te je trebalo par puta isključiti te ponovno pokrenuti Android Studio kako bi funkcionalnost proradila .

Trebalo bi dodati zaključak nakon usporedbe. Možda zbroj bodova?
 Opis prednosti jednoga i drugoga, nedostatke jednoga i drugoga.
 Preporuka?

Zaključak

Objektivno gledajući(po kriterijima usporedbe) u ovom slučaju pobjeđuje Android Studio, ali moram uzeti u obzir da je ovaj slučaj jako specifičan. Programer(ja) poznaje i zna se koristiti sa oba programska jezika(java i C#) i ima iskustva sa oba razvojna okruženja, također ovdje se radilo o android aplikaciji koja trenutno nije namijenjena niti će vjerojatno biti drugim platformama(npr. iOS). Ali ako razmotrimo i druge slučajeve onda gore navedeni kriteriji padaju u vodu. Na primjer ako tvrtka posjeduje licencu za Visual Studio te u njoj rade .NET programeri i primi narudžbu za kreiranje mobilne aplikacije(može biti kao dodatni zahtjev originalne narudžbe) logičnije bi bilo koristiti Xamarin u takvoj situaciji, jer čak i ukoliko programeri nisu upoznati sa strukturom mobilne aplikacije poznaju programski jezik i razvojno okruženje te uz edukaciju ne bi im trebao biti problem kreirati mobilnu aplikaciju, a ukoliko se odluče koristiti Android Studio morati će zaposliti nove programere ili uložiti puno više vremena i financija u edukaciju trenutnih. Također još jedan mogući scenarij je da tvrtka dobije narudžba za Android aplikaciju, ali nije sigurno hoće li ta aplikacija u budućnosti trebati funkcionirati i na drugim platformama. Ovo je malo kompliciraniji slučaj jer ukoliko se tvrtka specijalizirala za izradu mobilnih aplikacija i imaiskusne timove od kojih je svaki specijaliziran za neku od platformi, neće im biti problem u budućnosti kreirati aplikaciju za drugu platformu i trenutno će vrlo vjerojatno odabrati Android Studio. Ukoliko tvrtka nije specijalizirana za izradu mobilnih aplikacija na drugim platformama onda bi po mojem mišljenju bilo bolje odabrati Xamarin jer tim programera koji će raditi na toj aplikaciji neće imati problema prilagoditi je i za druge platforme(90% koda možda i više koji je napisan pomoću Xamarin.Android alata može se bez problema koristiti i u Xamarin.Forms alatima pomoću koji služe za kreiranje aplikacija za više platformi ili koristiti druge alate kao Xamarin.iOS za izradu aplikacije namijenjene specifičnoj platformi) i nije potrebno imati različite programere odnosno timove programera za druge platforme jer će se koristiti isti programski jezik i isto razvojno okruženje. Što se tiče scenarija u kojem programer koji je upoznat sa samo jednim od navedenih programskih jezika dobije zadatak za kreirati Android aplikaciju a prije nikad nije radio mobilne aplikacije, preporučam da odabere alat u kojem se koristi programski jezik sa kojim je upoznat. I sada dolazimo do konačnog scenarija u kojem programer nije upoznat sa niti jednim od navedenih jezika, nema iskustva sa razvojnim okruženjima ni korištenim alatima i nije upoznat sa strukturom mobilne aplikacije ali ima želju u budućnosti kreirati mobilne aplikacije te se postavlja pitanje što mu preporučiti kao početniku? Odgovor koji bi osobno dala na to pitanje jest da nauči oba, odnosno da krene sa izradom aplikacije upravo kako je predstavljeno u ovom radu, paralelno kreirajući dvije

aplikacije i na taj način upoznati će se sa oba programska jezika, razvojnim okruženjima i u budućnosti moći će odabrati način razvoja aplikacije koji će najbolje odgovarati zahtjevima. Zaključak ovog rada jest da nema konačnog pobjednika već odabir ovisi isključivo o okolnostima.

Popis literature

Nisam primijetio u tekstu da se spominju sve reference. Možda sam pogriješio. Svaka referenca ovdje spomenuta mora biti korištena barem jednom u tekstu. Predlažem korištenje APA stila referenciranja (Autor, godina).

- [1] Anroid Developers, „Developer Guides“ list. 2021. [Na internetu] Dostupno: <https://developer.android.google.cn/guide/>, [pristupano 2.10.2021]
- [2] Microsoft, „Xamarin.Android Documentation“ kol.2020. [Na internetu] Dostupno: <https://docs.microsoft.com/en-us/xamarin/android/>, [pristupano 5.11.2021]
- [3] Raghu Ramakrishna, „Building Xamarin.Android application in C#“ ožuj. 2021. [Na internetu] Dostupno: <https://www.ecanarys.com/Blogs/ArticleID/75/Building-Xamarin-Android-application-in-C> [pristupano 1.11.2021]
- [4] Altexsoft, „ The Good and The Bad of Xamarin Mobile Development“ stud. 2020. [Na internetu] Dostupno: <https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/> [pristupano 6.12.2021]
- [5] Serve an answer, „Change font Xamarin.android“ velj. 2022. [Na internetu] Dostupno: <https://serveanswer.com/questions/change-font-xamarin-android-c> [pristupano 14.02.2022]
- [6] Charles Petzol, “Creating Mobile Apps with Xamarin.Forms“ , izd. 1, str. 1-1122, studeni-2021
- [7] Xamarin.Forms, “Notes for proffesionals””, izd. 1, str. 1-174, listopad-2021
- [8] Android, “Notes for proffesionals”, izd. 1, str. 1-1281, listopad-2021
- [9] Google Developer Training Team, „Android Developer Fundamentals Course, izd 1, str. 1-491, pros-2021
- [10] Android arsenal, „CPicker“ sij. 2020. [Na internetu] Dostupno: <https://android-arsenal.com/details/1/8025> [pristupano 13.12.2021]
- [11] Fred Porciuncula „Rendering PDFs on Android the easy way“ sij. 2020. [Na internetu] Dostupno: <https://proandroiddev.com/rendering-pdfs-on-android-the-easy-way-c05635b2c3a8> [pristupano 11.12.2021]
- [12] Joe Hindy „15 best Android emulators for PC and Mac of 2022.“ Velj.2022 [Na internetu] Dostupno: <https://www.androidauthority.com/best-android-emulators-for-pc-655308/> [pristupano 1.2.2022]
- [13] Tutorials point „Java“ [Na internetu] Dostupno: https://www.tutorialspoint.com/java/java_documentation [pristupano 1.10.2021]
- [14] C# in Depth „Implementing the Sinleton Pattern in C#“ velj. 2011. [Na internetu] Dostupno: <https://csharpindepth.com/articles/singleton> [pristupano 20.11.2021]
- [15] Journal Dev „Java Singleton Design Pattern“ ož. 2021. [Na internetu] Dostupno: <https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples> [pristupano 20.11.2021]

Popis slika

Popis tablica

