Mobilna aplikacija za praćenje i analizu COVID-19 slučajeva

Šamija, Valentin

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Valentin Šamija

APLIKACIJA ZA PRAĆENJE I ANALIZU COVID-19 SLUČAJEVA

Diplomski rad

Pula, kolovoz 2021.

Sveučilište Jurja Dobrile u Puli

Fakultet informatikeu Puli

Valentin Šamija

APLIKACIJA ZA PRAĆENJE I ANALIZU COVID-19 SLUČAJEVA

Diplomski rad

JMBAG: 0303068763, redovni student

Studijski smjer: Informatika

Predmet: Mobilne aplikacije/Primijenjena statistika

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: izv. prof. dr. sc. Siniša Sovilj

Pula, kolovoz 2021.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani **Valentin Šamija**, kandidat za **magistra informatike** ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli,_____, 2021. godine

IZJAVA

o korištenju autorskog djela

Ja, **Valentin Šamija** dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom **Aplikacija za praćenje i analizu Covid-19 slučajeva** koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Potpis

_____

U Puli,_____, 2021. godine

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Pula, 28. ožujka 2021.

DIPLOMSKI ZADATAK

Pristupnik:	**Šamija Valentin (0303068763)**
Studij:	Sveučilišni diplomski studij Informatike
Naslov (hrv.):	**Mobilna aplikacija za praćenje i analizu Covid-19 slučajeva.**
Naslov (eng.):	Mobile application for monitoring and analysis of Covid-19 cases.
Opis zadatka:	Zadatak je izraditi mobilnu aplikaciju za praćenje i analizu Covid-19 slučajeva. Aplikacija se treba temeljiti na tehnologijama: Java/Kotlin. Aplikacija treba iz više izvora automatski agregirati podatke te treba korisniku omogućiti grafički pregled trenutnog stanja, mogućnost pretraživanja, prikaz (mapu), prikaz po datumu i automatsko osvježavanje za sve nove podatke. Tablični prikaz za svaku varijablu povezanu uz virus te prikaz brzine rasta u postotcima i isto tako za brzinu smanjivanja. Omogućiti primanje personaliziranih notifikacija u slučaju iznenadnog povećanja ili smanjenja neke varijable i sl. Opisati sustav: korisničke scenarije, funkcionalnosti, izraditi potrebne UML dijagrame - klasne, use case, use sequence, opisati implementaciju te na kraju izraditi kratke korisničke upute.

Zadatak uručen pristupniku:

28.ožujka 2021.

Rok za predaju rada:

28. veljače 2022.

Mentor:

Siniša Sovilj

doc.dr.sc. Siniša Sovilj

SADRŽAJ

1. UVOD

„Korona, COVID-19 i koronavirus“ su tri fraze koje čujemo svakoga dana po nekoliko stotina puta. Osim što ih čujemo uživo, preko televizora ili radija o njima možemo i pročitati na svim portalima, društvenim mrežama i sličnim mjestima. Život cjelokupnog stanovništva se okrenuo naopačke. Prije skorog vremena ljudi su se družili, kretali, putovali, odlazili na posao i sve ostalo bez ikakve pomisli da bi im njihove svakodnevne rutine moglo nešto poremetiti, a to se baš dogodilo. Prije 2 godine, točnije tijekom prosinca 2019. nastala je zaraza uzrokovana virusom „SARS-CoV-2“ u gradu Wuhanu (Kina) koja se nekontroliranom brzinom proširila cijelim svijetom.

Isprva su svi bili ravnodušni gledajući vijesti ili videozapise preko interneta i mislili kako je to zapravo toliko udaljeno od istih te držali da se to ne može dogoditi upravo nama svima. Trenutak u kojem se sve mijenja je objava koja upućuje na to da se virus lagano približava mjestima i krajevima koji su bili bez straha. Sve veći broj zaraženih i na kraju ono najgore preminulih u ljude unosi sve veću dozu straha i panike te svi započinju s masovnom opskrbom namirnica koje su nam svakodnevno potrebne. Veći broj zaraženih donosi i neke nove mjere, a te mjere također postaju sve rigoroznije koje dovode do potpunog zatvaranja (Lockdown) koja je privremena mjera koju određuje državna vlast s ograničenjima u javnom životu radi zaustavljanja širenja epidemije ili pandemije. Kada je proglašeno zatvaranje za neke je život stao jer nije bilo moguće otići s ekipom na kavicu, neki tulum ili putovanje. Poslovi su također ograničeni kao i obrazovno-znanstvena grana te se skoro sve počinje odrađivati na daljinu (Online).

Nastavkom nekontroliranog širenja virusa nastaju i velike brojke podataka koje treba nekako sortirati, posložiti, obraditi, pratiti, vizualizirati te prikazati kako bi se moglo vidjeti u kojem se stanju trenutno nalazimo. Projektni zadatak kojim se bavi ovaj diplomski rad upravo odrađuje posao vezan uz podatke o Covid-19 slučajevima. Svi prikupljeni podaci se promatraju, obrađuju, stvaraju se tablične i grafičke analize koje svaki korisnik može pregledati u bilo kojem trenutku. U nastavku rada će sve bit

detaljno objašnjeno, prikazano na koji način sve funkcionira te sve zanimljivosti vezane uz rad.

2. PROGRAMI, PLATFORME I ALATI ZA IZRADU APLIKACIJE

Prilikom izrade aplikacije potrebno je dobro proučiti područje koje bi najbolje odgovaralo zahtjevima koje postavlja sama izrada aplikacije. S obzirom da se u ovom radu izrađuje mobilna aplikacija, istraženo je područje koje odgovara tim kriterijima. Kasnije će biti dodan i dio koji odgovara kriterijima web područja iz razloga što se obrađuje velik broj statističkih podataka koji se također mogu prikazati u tom području, odnosno na taj način.

Integrirano razvojno okruženje (IDE) softver je za izradu aplikacija koji kombinira uobičajene razvojne alate u jedno grafičko korisničko sučelje (GUI). Što se tiče „Android" svijeta najbolji programi za izradu aplikacija su:

- Visual Studio – Xamarin
- Android Studio
- IntelliJ IDEA
- DeuterIDE
- Eclipse IDE.

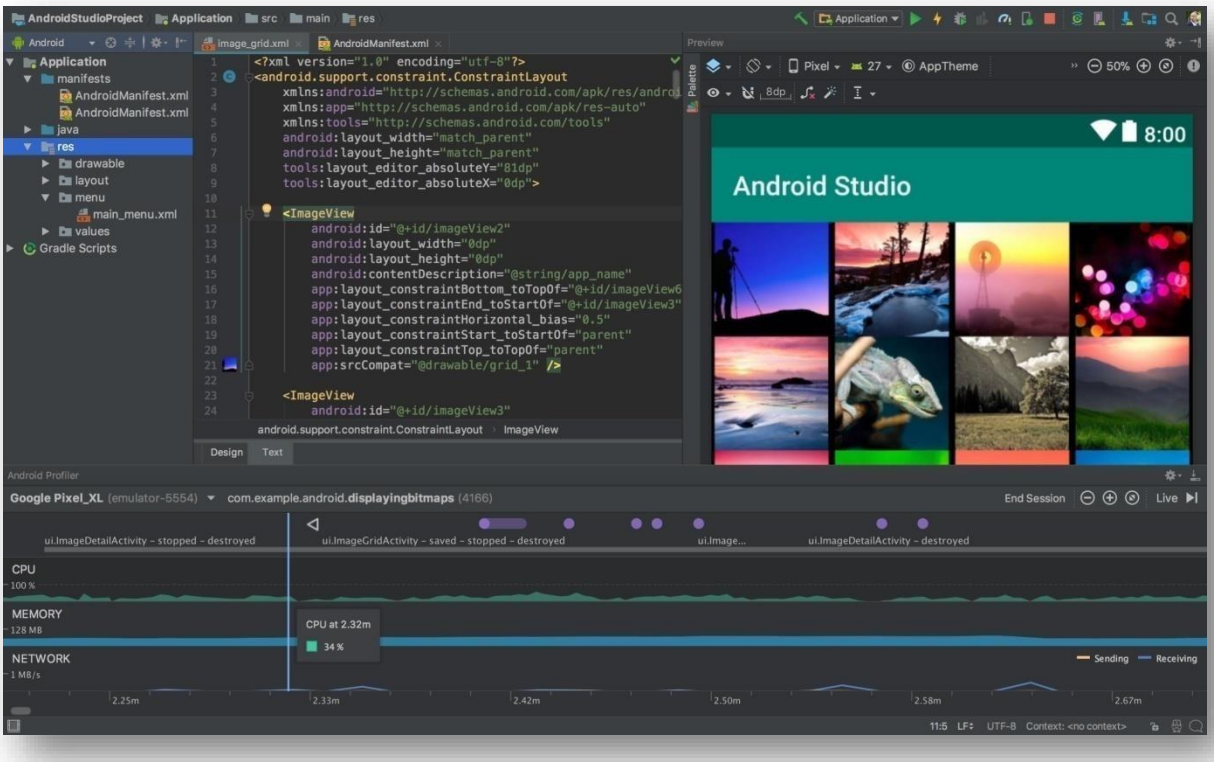
2.1. Android Studio

Android Studio službeno je integrirano razvojno okruženje (IDE) za razvoj Android aplikacija, temeljeno na IntelliJ IDEA. Povrh moćnih IntelliJ -ovih uređivača koda i alata za razvojne programere, Android Studio nudi još više značajki koje povećavaju vašu produktivnost pri izradi Android aplikacija, kao što su:

- Brz emulator bogat raznim značajkama
- Jedinstveno okruženje u kojem se može razvijati za sve Android uređaje
- Primjenjivanje izmjena u kodu i promjena resursa na pokrenutu aplikaciju bez ponovnog pokretanja aplikacije
- Predlošci koda i integracija GitHub -a koji pomažu u izgradnji uobičajenih značajki aplikacije i uvozu koda
- Opsežni alati i okviri za testiranje

- Alati za prepoznavanje performansi, upotrebljivosti, kompatibilnosti verzija i drugih problema.

Slika 1: Android Studio logo



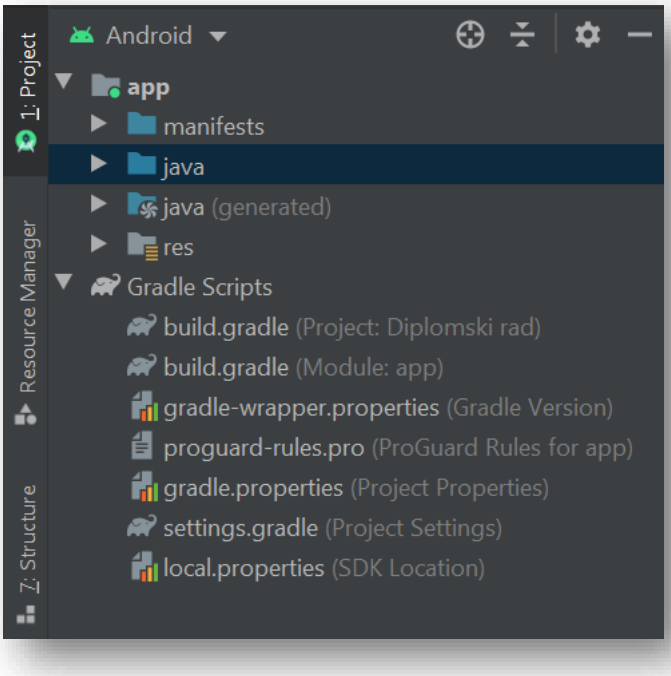
Slika 2: Android Studio sučelje

2.2. Struktura projekta

Svaki projekt u Android Studiju sadrži jedan ili više modula s datotekama izvornog koda i datotekama resursa. Vrste modula uključuju:

- Moduli aplikacija za Android
- Library moduli
- Moduli Google App Engine.

Prema zadanim postavkama, Android Studio prikazuje projektne datoteke u prikazu Android projekta, kao što je prikazano na slici 3. Ovaj je prikaz organiziran po modulima za brzi pristup ključnim izvornim datotekama projekta.



Slika 3: Android Studio projektne datoteke

Sve datoteke za izgradnju vidljive su na najvišoj razini u odjeljku Gradle Scripts, a svaki modul aplikacije sadrži sljedeće mape:

- manifests: Sadrži datoteku AndroidManifest.xml - Android Manifest je XML datoteka koja sadrži važne metapodatke o aplikaciji Android. To uključuje naziv paketa, nazive aktivnosti, glavnu aktivnost (ulaznu točku u aplikaciju), podršku

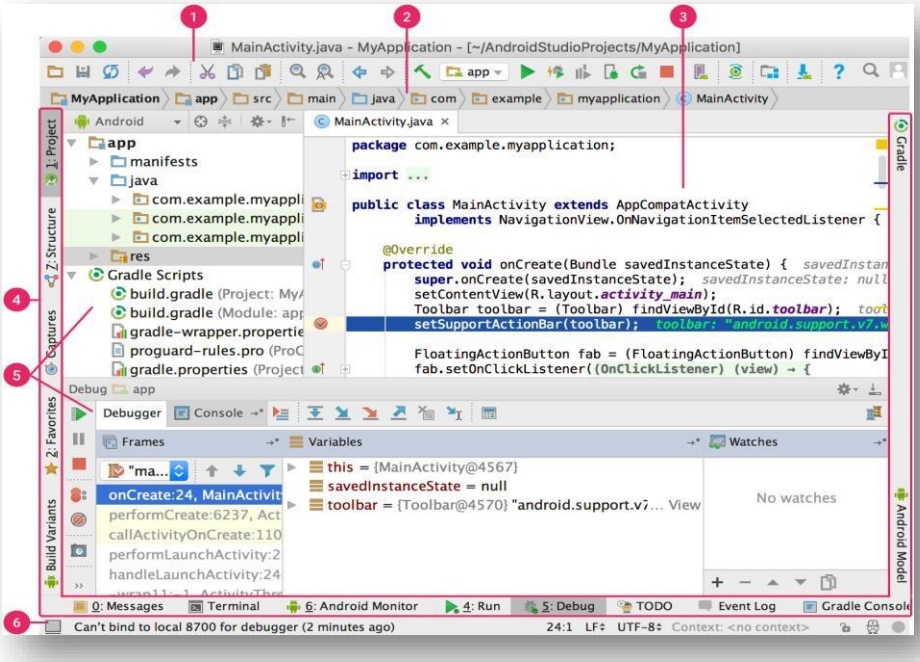
za verziju Androida, podršku za hardverske značajke, dopuštenja i druge konfiguracije.

- java: Sadrži datoteke izvornog koda Jave, uključujući testni kod JUnit.
- res: Sadrži sve resurse koji nisu kodirani, poput XML izgleda, nizova korisničkog sučelja i slika.

Android struktura projekta na disku razlikuje se od ovog „vanjskog“ prikaza. Da bi vidjeli stvarnu datotečnu strukturu projekta, potrebno je odabrati „Project“ s padajućeg izbornika Project (lijevo slika 3). Također moguće je prilagoditi prikaz projektnih datoteka tako da se usredotoči na određene aspekte razvoja aplikacije. Na primjer, odabirom prikaza „Problemi“ unutar projekta prikazuju se veze do izvornih datoteka koje sadrže prepoznate pogreške kodiranja i sintakse, poput nedostajuće oznake zatvaranja XML elementa u datoteci izgleda (Layout).

2.3. Korisničko sučelje

Glavni prozor Android Studija sastoji se od nekoliko logičkih područja što je prikazano na slici 4:



Slika 4: Korisničko sučelje

1. Alatna traka **(toolbar)** omogućuje izvršavanje širokog raspona radnji, uključujući pokretanje aplikacije i pokretanje Android alata.
2. Navigacijska traka **(navigation bar)** pomaže u kretanju kroz projekt i otvaranju datoteka za uređivanje. Omogućuje kompaktniji prikaz strukture vidljive u prozoru **Project**.
3. Prozor uređivača **(editor)** je mjesto gdje se stvara i mijenja kod. Ovisno o trenutnoj vrsti datoteke, uređivač se može promijeniti. Na primjer, prilikom pregledavanja datoteke izgleda prikazuje se uređivač izgleda.
4. Alatna traka prozora **(tool window bar)** nalazi se oko vanjske strane IDE prozora i sadrži gumbe koji omogućuju proširenje ili sužavanje pojedinih prozora alata.
5. Prozori alata **(tool windows)** omogućuju pristup određenim zadacima poput upravljanja projektima, pretraživanja, kontrole verzija i drugih. Moguće ih je proširiti i skupiti.
6. Traka statusa **(status bar)** prikazuje status vašeg projekta i samog IDE -a, kao i sva upozorenja ili poruke.

Glavni prozor moguće je organizirati kako bi omogućili više prostora na ekranu skrivanjem ili pomicanjem alatnih traka i prozora alata. Također moguće je koristiti tipkovne prečace za pristup većini IDE značajki.

2.4. Postavke strukture projekta

Da bi promijenili različite postavke za projekt unutar Android Studia, potrebno je otvoriti dijaloški okvir **Project Structure** klikom na **File -> Project Strucutre**. Sadrži sljedeće odjeljke:

- **SDK Location**: Postavlja lokaciju JDK -a, Android SDK -a i Android NDK -a koje projekt koristi.
- **Project**: Postavlja verziju za Gradle i dodatak Android za Gradle te naziv lokacije spremišta.
- **Modules**: Omogućuje uređivanje konfiguracija sastavljanja za svaki modul, uključujući ciljni i minimalni SDK, potpis aplikacije i ovisnosti o knjižnici.

2.5. RStudio

Analiza podataka ili podatkovno rudarenje (eng. Data mining) je sortiranje, organiziranje ili grupiranje velikog broja podataka i izvlačenje relevantnih informacija. Sam termin se može objasniti kao proces pronalaženja korisnog znanja ili informacija, odnosno otkrivanje znanja iz velike količine podataka. Rudarenjem i analizom se poboljšava proces donošenja odluka na strateško-poslovnoj razini pružajući uvid u „skrivene“ podatke. Rudarenjem, odnosno analizom podataka se također otkrivaju odnosi, logičnost, pravilnost te općenito bilo kakve strukture među podacima. Rudarenje podrazumijeva organiziranje baza čišćenjem podataka kako bi se pristupilo znanju i stjecanju istog na temelju postojećih podataka u bazama. Razvoj tehnologije, računala, interneta bitno doprinosi lakšem organiziranju podataka, no da bi oni postali korisni, potrebno je njihovo pretvaranje u informacije i znanje.

Najpoznatiji programi za analizu podataka su:

- Python
- Rstudio
- Graphpad
- SPSS
- XLSTAT.

Za web područje ovog projektnog zadatka koji je opisan u ovom diplomskom radu korišten je RStudio. Najprije treba razbiti razlike između R i RStudia. Važno je uočiti razlike između R i RStudia. R je programski jezik koji se koristi za statističko računanje, dok RStudio koristi jezik R za razvoj statističkih programa.

RStudio je integrirano razvojno okruženje (IDE) za R. Sadrži konzolu, uređivač za isticanje sintakse koji podržava izravno izvršavanje koda, kao i alate za crtanje, povijest, ispravljanje pogrešaka i upravljanje radnim prostorom. RStudio je dostupan u otvorenim komercijalnim izdanjima i radi na radnoj površini (Windows, Mac i Linux) ili u pregledniku spojenom na RStudio poslužitelj – server.

Slika 5: RStudio

R paketi - zbirke funkcija, podataka i sastavljenog koda u dobro definiranom formatu, stvorene za dodavanje specifičnih funkcija. Postoji različiti tipovi paketa:

- Za analizu i istraživanje **(tidyverse)**
- Za komunikaciju i interakciju **(shiny)**
- Za modeliranje i predviđanje
- Za povezivanje i integriranje.

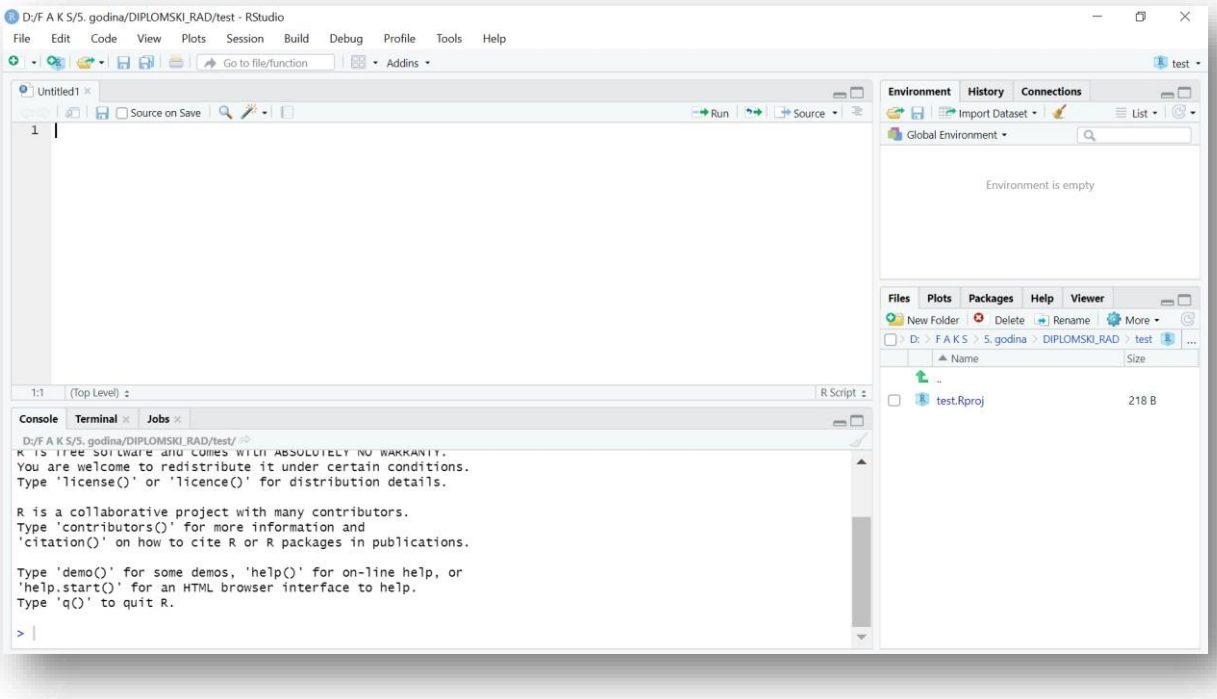
Najbitniji paket za ovaj zadatak je **Shiny**. Paket je koji olakšava izradu interaktivnih web aplikacija izravno iz R okruženja. Mogućeje postaviti samostalne aplikacije na web stranicu. Aplikacije se mogu proširiti i CSS temama, HTML widgetima i JavaScript radnjama.

Slika 6: Shiny paket

RStudio omogućuje postavljanje **Shiny** web aplikacija i interaktivnih dokumenata na internet na nekoliko načina:

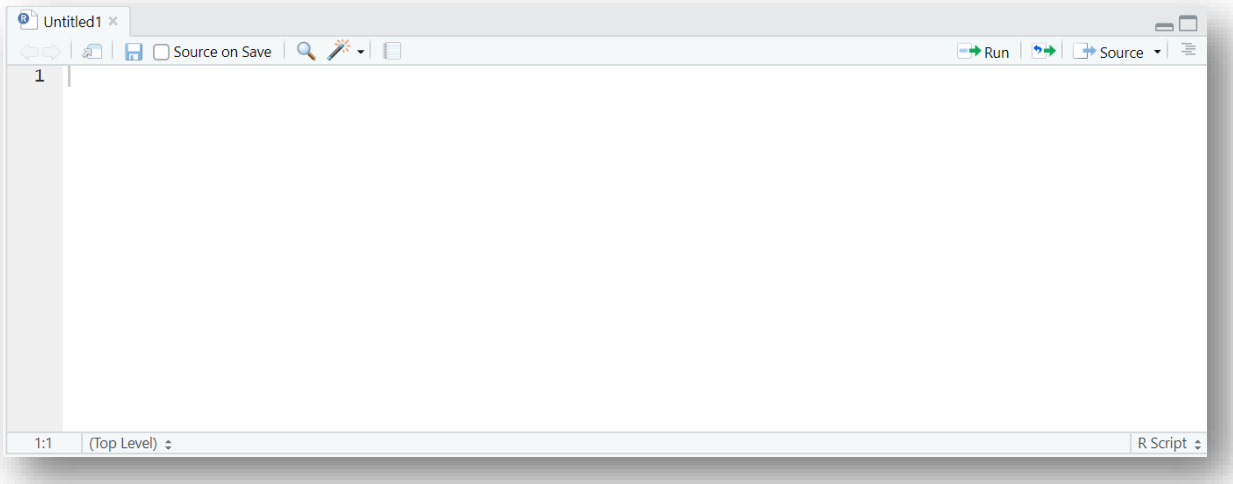
- Shinyapps.io
- Shiny Server
- RStudio Connect.

Kada se otvori RStudio trebao bi izgledati ovako:



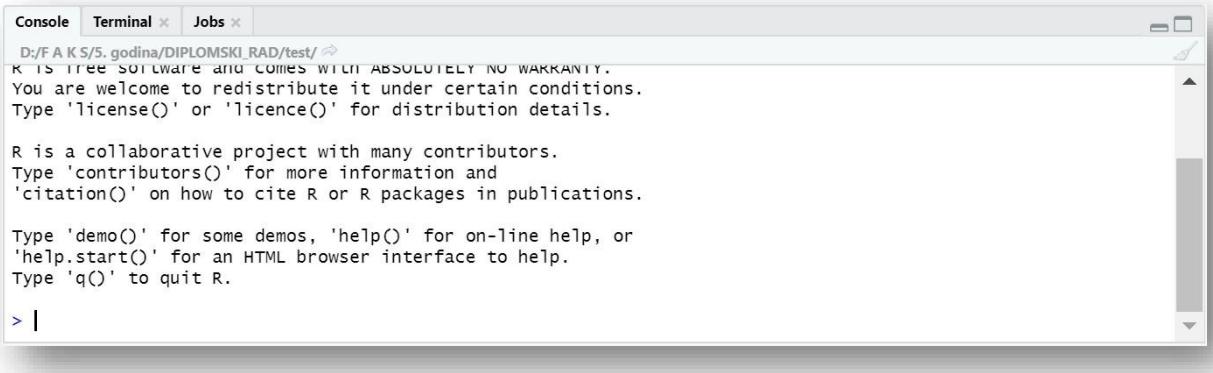
Slika 7: RStudio korisničko sučelje

Gornji lijevi kvadrant je uređivač. Unutar uređivača se piše R kod koji služi za kasnije - funkcije, klase, pakete itd. To je, za sve namjere, identično glavnom prozoru svakog drugog uređivača koda.

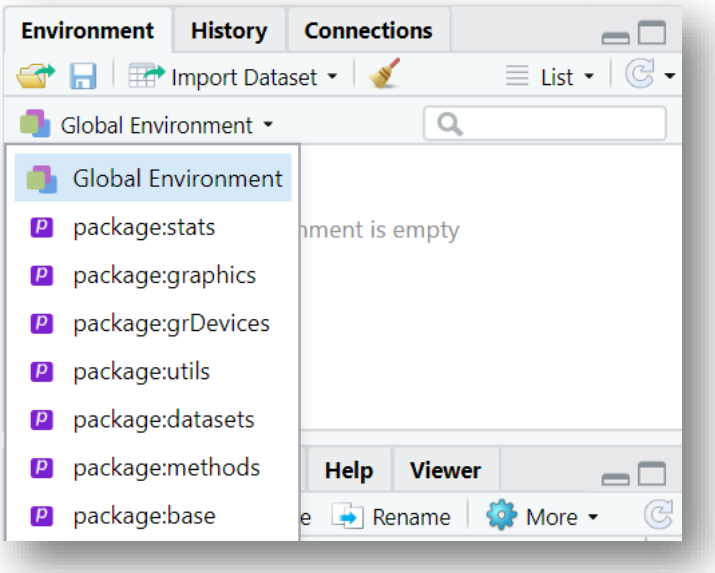


Slika 8: RStudio editor

Kvadrant ispod je zapravo konzola u kojem je moguće isprobati svoje ideje, skupove podataka, filtre i funkcije. Unutar konzole se provjera da li zamišljena ideja funkcionira prije nego se kopira u gornji uređivač (editor).



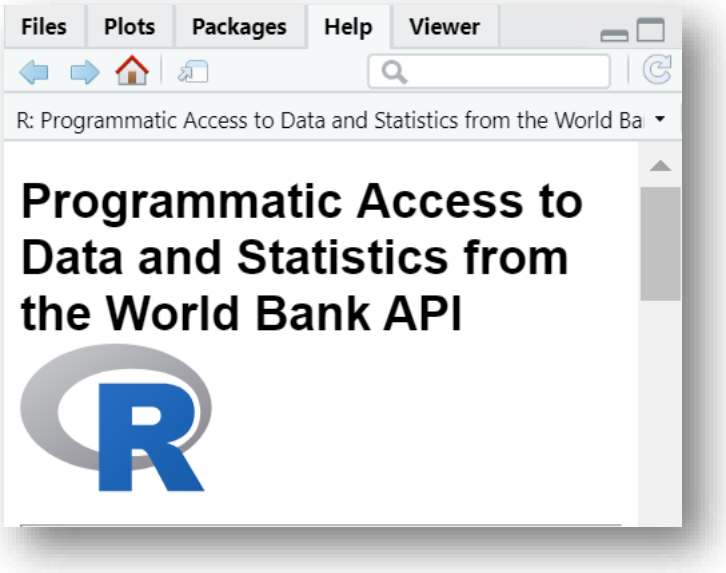
Slika 9: RStudio console



Slika 10: RStudio Environment/History

Environment prikazuje sve varijable, skupove podataka i funkcije koje su dostupne u trenutnoj sesiji. Padajući izbornik **Global Environment** prikazuje koji su paketi učitani.

Povijest sprema svaku pojedinačnu naredbu koja je izvršena od zadnjeg početka projekta. Sprema se u skrivenu datoteku **.Rhistory** u mapi unutar projekta. Ako se ne spremi okruženje nakon sesije, povijest se neće spremiti.



Slika 11: RStudio Misc

Kvadrant dolje desno prikazuje razne kartice:

- **Files**
- Kartica **Plots** sadržavat će grafikone koji su generirani. Ovdje je moguće zumirati, izvoziti, konfigurirati i pregledati grafikone
- Kartica **Packages** omogućuje instaliranje dodatnih paketa u R. Kratak opis nalazi se pored svakog dostupnog paketa, iako ih ima mnogo više od onih koji su tamo navedeni
- Kartica **Help** omogućuje pretraživanje nevjerojatno opsežnog direktorija pomoći i automatski će se otvoriti kad se god pozove pomoć putem naredbe u konzoli
- Kartica **Viewer** je zapravo ugrađen RStudio preglednik.

3. NAČIN PRIMJENE I RAZRADA FUNKCIONALNOSTI

Isprva za područje primjene je bio izabran samo mobilni dio, ali kasnije kako se projekt širio stvorena je ideja i o web dijelu aplikacije. Funkcionalnosti su približno slične, ali načine primjene i razradu je potrebno podijeliti na gore navedene dijelove radi lakšeg snalaženja i razumijevanja.

3.1. Način primjene kroz mobilni dio – Android Studio

Na samom početku ovoga rada već je spomenut Android Studio kao jedan od programa koji će biti korišteni za izradu aplikacije, ali sada je potrebno proći kroz sve korake izrade u tom području te prikazati značajnije dijelove. Izrada aplikacije temeljena je na dva sučelja – **Interfacea**, a to su: **RestAPI** i **Volley**. Najprije će biti objašnjeni svaki od njih.

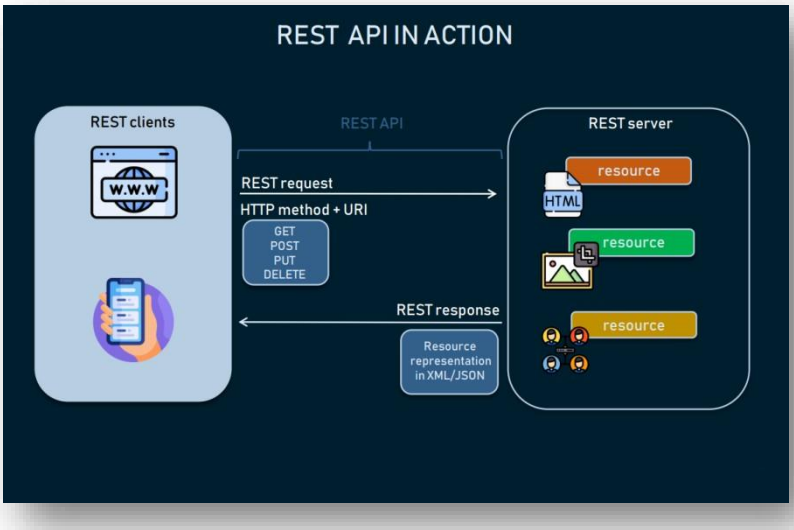
Kada vidimo riječ **RestAPI** ona se zapravo sastoji od dva dijela; **REST** i **API**. **API** je skup definicija i protokola za izradu i integraciju aplikacijskog softvera. Ponekad se naziva i ugovorom između pružatelja informacija i korisnika informacija - kojim se utvrđuje sadržaj koji se traži od potrošača (poziv) i sadržaj koji zahtijeva proizvođač (odgovor).

Drugim riječima, ako želite stupiti u interakciju s računalom ili sustavom radi dohvaćanja informacija ili obavljanja funkcije, **API** vam pomaže da prenesete sustavu ono što želite kako bi mogao razumjeti i ispuniti zahtjev. **API** možete zamisliti kao posrednika između korisnika ili klijenata i resursa ili web usluga koje žele dobiti. To je također način na koji organizacija dijeli resurse i informacije uz održavanje sigurnosti, kontrole i provjere autentičnosti - određujući tko će čemu pristupiti.

Slika 12: API

REST je skup arhitektonskih ograničenja, a ne protokol ili standard. **API** programeri mogu implementirati **REST** na različite načine. Kad je zahtjev klijenta napravljen putem RESTful API-ja, on prenosi prikaz stanja resursa na podnositelja zahtjeva ili krajnju točku. Ove se informacije ili prikazi isporučuju u jednom od nekoliko formata putem HTTP-a: **JSON** (Javascript Object Notation), HTML, XLT, Python, PHP ili običan tekst. **JSON** je općenito najpopularniji format datoteke koji se koristi jer je, unatoč svom imenu, jezično agnostičan, a čitljiv je i za ljude i za strojeve.

U ovom projektnom zadatku će **JSON** kao najpopularniji format biti upravo i korišten. Koristit će se za slanje podataka s poslužitelja klijentu kako bi se ti podaci mogli prikazati unutar aplikacije.



Slika 13: REST API

Ovo poglavlje opisuje funkcionalnosti aplikacije. Na početku će biti prikazano kako funkcionira aplikacija na temelju Use-Case dijagrama te kakva je interakcija između korisnika i aplikacije. Potom je prikazan dijagram slijeda na kojem se jasno vide funkcije ove aplikacije. U konačnici je prikazan klasni dijagram te su opisane veze između klasa. U ovoj aplikaciji definirana je zapravo samo jedna skupina korisnika, tj. korisnik, koji može pregledavati i analizirati stanje povezano uz COVID-19.

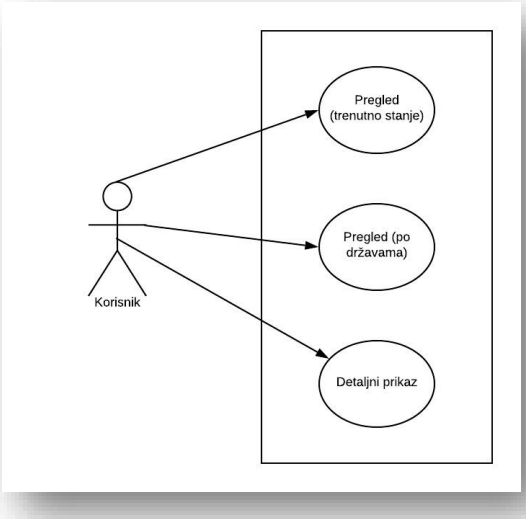
Volley je HTTP biblioteka koja čini umrežavanje za Android aplikacije lakšim i što je najvažnije, bržim. Volley nudi sljedeće prednosti:

- Automatsko zakazivanje mrežnih zahtjeva
- Više istovremenih mrežnih veza
- Podrška za određivanje prioriteta zahtjeva.

Ističe se u operacijama koje se koriste za popunjavanje korisničkog sučelja, kao što je dohvaćanje stranice rezultata pretraživanja kod strukturiranih podataka. Lako se integrira u bilo koji protokol. Volley oslobađa od pisanja uvodnog koda i omogućuje da se programer koncentrira na logiku koja je specifična za aplikaciju koju izrađuje.

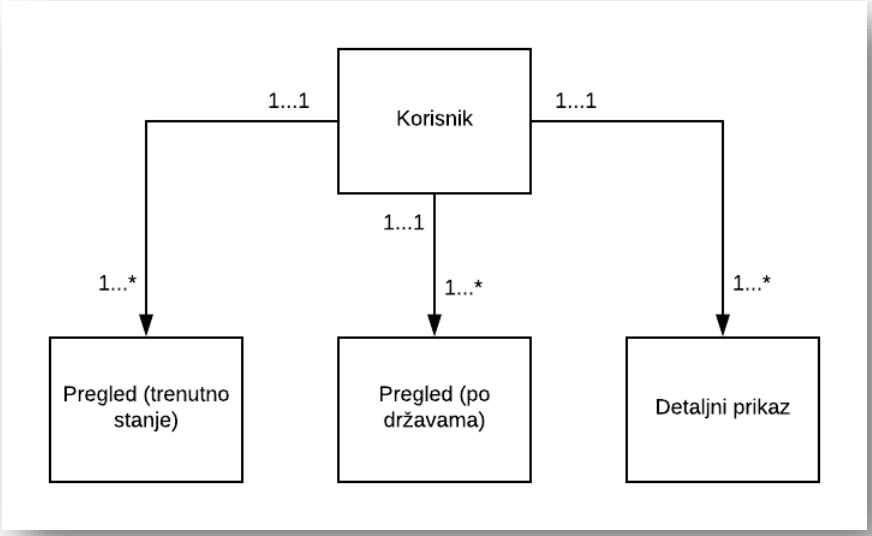
Funkcionalnosti unutar aplikacije (korisnički slučajevi):

- Pregled trenutnog stanja
- Pregled po država
- Pretraživanje.



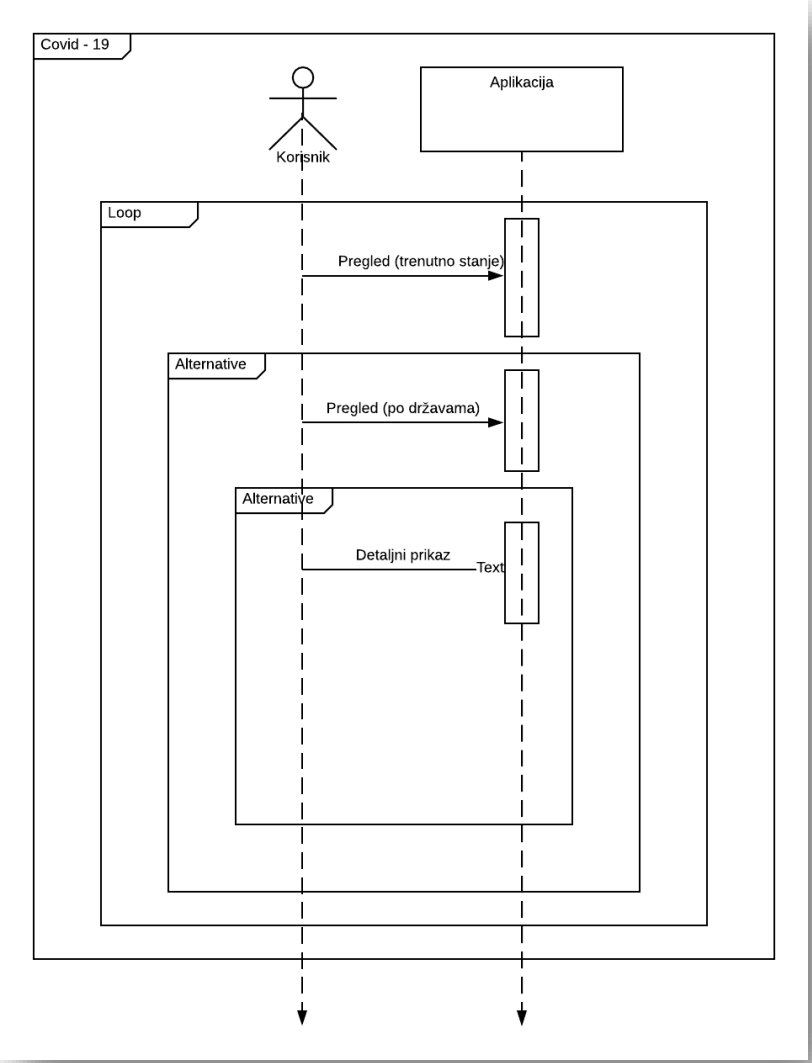
Slika 14: Use-case dijagram (mobile)

Korisnik ima tri mogućnosti: pregled trenutnog stanja,pregled po državama u kojem je integrirano i pretraživanje te detaljni prikaz.



Slika 15: Klasni dijagram (mobile)

Na slici 15 je prikazan klasni dijagram. Klasni dijagram opisuje strukturu sustava, odnosno objašnjava klase unutar sustav, njihove atribute te njihove odnose. Osim toga postoje kardinalnosti koje objašnjavaju odnose među klasama. Asocijacija prikazuje vezu između klasa.



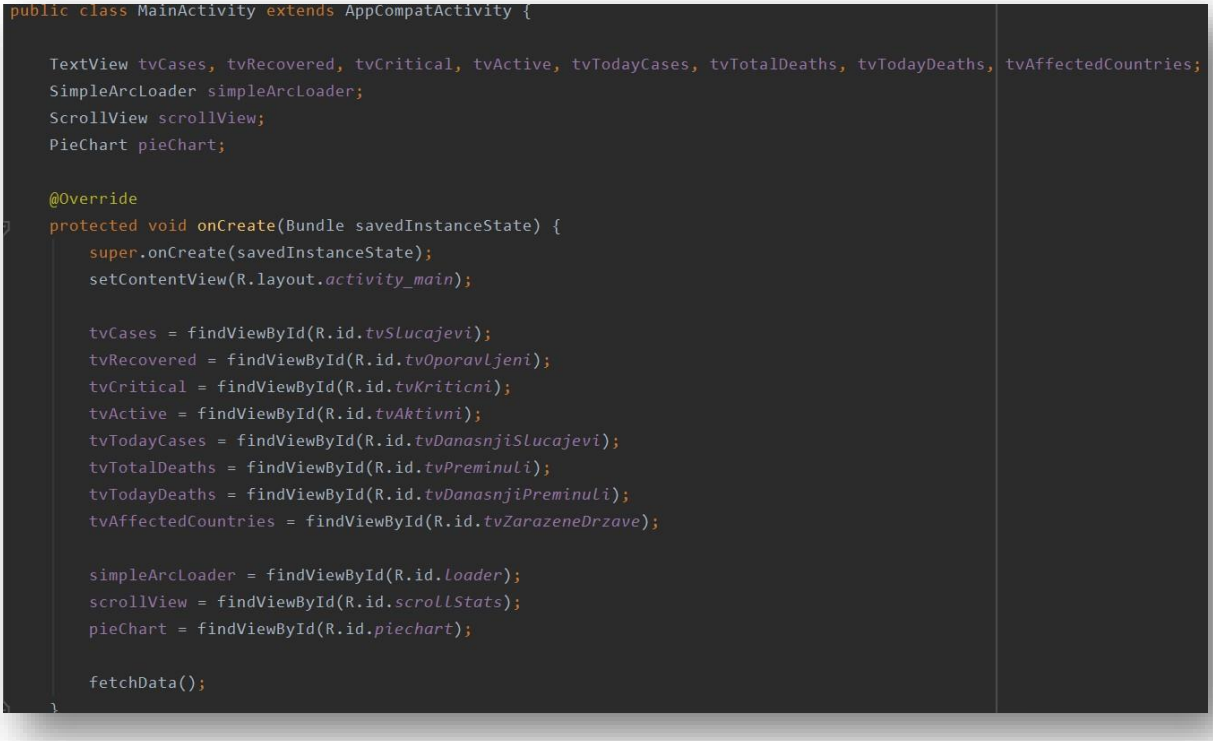
Slika 16: Use-sequence dijagram (mobile)

Korisnik bira što želi učiniti i to je prikazano “Sequence” (slijednim) dijagramom.

U nastavku će biti prikazan način izrade te prikaz i opis temeljnih funkcionalnosti:

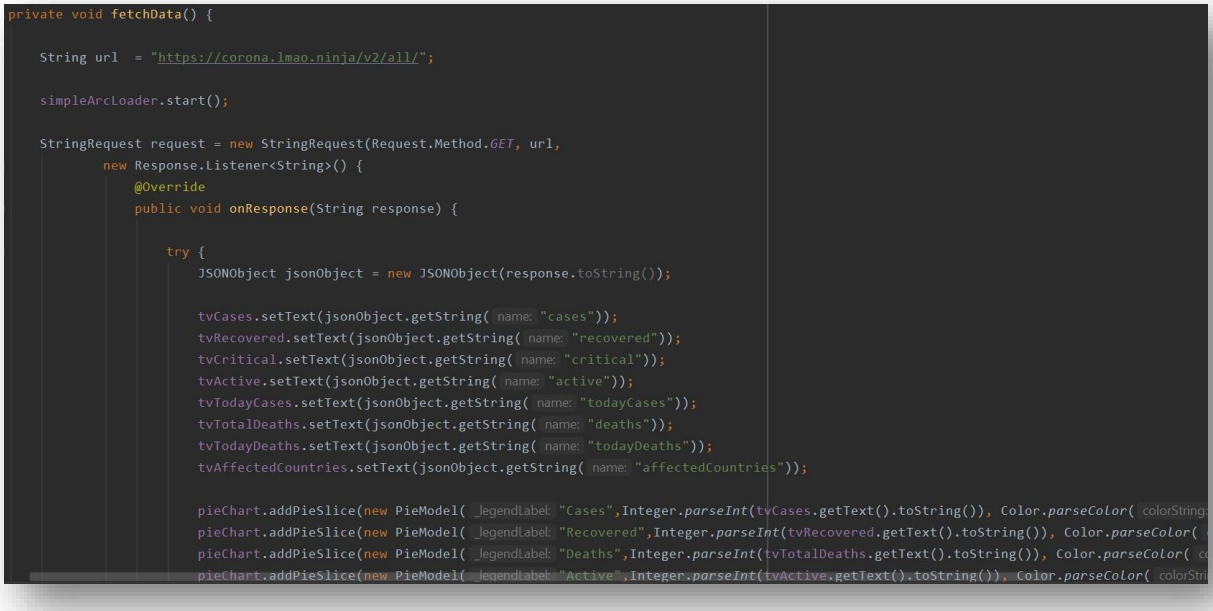
- MainActivity

Unutar MainActivity klase definirani su atributi koji se povlače s internetskog izvora u aplikaciju (https://corona.lmao.ninja/v2/all/). Bez definiranja **TextViewa** koji je odgovoran za ovu radnju podaci se nikada ne bi prikazali unutar aplikacije. Osim toga definiran je i **pieChart** odnosno tortni grafikon koji će prikazivati najrelevantnije podatke. Sve je prikazano na slici ispod.



```
public class MainActivity extends AppCompatActivity {

    TextView tvCases, tvRecovered, tvCritical, tvActive, tvTodayCases, tvTotalDeaths, tvTodayDeaths, tvAffectedCountries;
    SimpleArcLoader simpleArcLoader;
    ScrollView scrollView;
    PieChart pieChart;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tvCases = findViewById(R.id.tvSlucajevi);
        tvRecovered = findViewById(R.id.tvOporavljeni);
        tvCritical = findViewById(R.id.tvKriticni);
        tvActive = findViewById(R.id.tvAktivni);
        tvTodayCases = findViewById(R.id.tvDanasnjiSlucajevi);
        tvTotalDeaths = findViewById(R.id.tvPreminuli);
        tvTodayDeaths = findViewById(R.id.tvDanasnjiPreminuli);
        tvAffectedCountries = findViewById(R.id.tvZarazeneDrzave);

        simpleArcLoader = findViewById(R.id.loader);
        scrollView = findViewById(R.id.scrollStats);
        pieChart = findViewById(R.id.piechart);

        fetchData();
    }
```

Slika 17: MainActivity a)



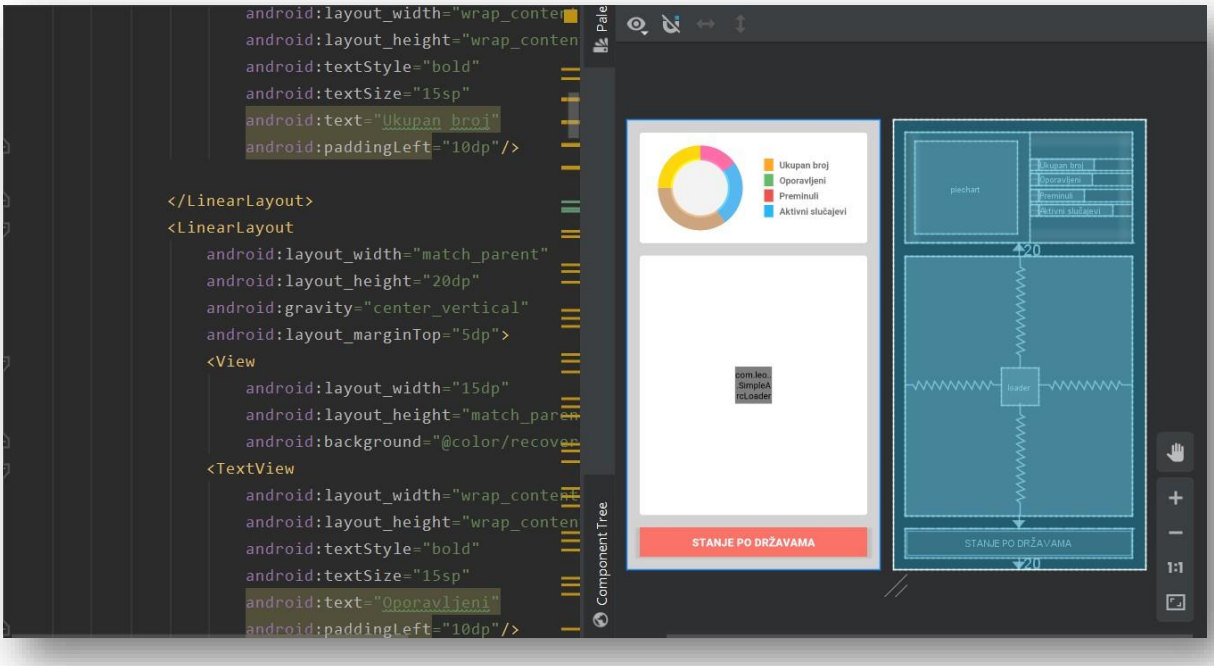
```
private void fetchData() {

    String url  = "https://corona.lmao.ninja/v2/all/";

    simpleArcLoader.start();

    StringRequest request = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {

                    try {
                        JSONObject jsonObject = new JSONObject(response.toString());

                        tvCases.setText(jsonObject.getString( name: "cases"));
                        tvRecovered.setText(jsonObject.getString( name: "recovered"));
                        tvCritical.setText(jsonObject.getString( name: "critical"));
                        tvActive.setText(jsonObject.getString( name: "active"));
                        tvTodayCases.setText(jsonObject.getString( name: "todayCases"));
                        tvTotalDeaths.setText(jsonObject.getString( name: "deaths"));
                        tvTodayDeaths.setText(jsonObject.getString( name: "todayDeaths"));
                        tvAffectedCountries.setText(jsonObject.getString( name: "affectedCountries"));

                        pieChart.addPieSlice(new PieModel( _legendLabel: "Cases",Integer.parseInt(tvCases.getText().toString()), Color.parseColor( colorString:
                        pieChart.addPieSlice(new PieModel( _legendLabel: "Recovered",Integer.parseInt(tvRecovered.getText().toString()), Color.parseColor(
                        pieChart.addPieSlice(new PieModel( _legendLabel: "Deaths",Integer.parseInt(tvTotalDeaths.getText().toString()), Color.parseColor(
                        pieChart.addPieSlice(new PieModel( _legendLabel: "Active",Integer.parseInt(tvActive.getText().toString()), Color.parseColor( colorStri
```

Slika 18: MainActivity b)

Na slici iznad prikazan je način na koji **JSON** dohvaća podatke, odnosno objekte. Osim što je prikazan način dohvaćanja podataka, prikazan je i način kreiranja tortnog grafikona. Svaki put kada se neki podatak dohvati, automatski se sprema unutar tortnog prikaza.

- activity_main.xml

Da bi se sve gore navedeno moglo vidjeti, odnosno prikazati na mobilnom uređaju ili emulatoru unutar Android Studia potpunu odgovornost snosi **XML** koji služi za onaj ljepši dio – dizajn. S obzirom da je gore kreirana klasa za nju je potreban i popratni dizajn. Unutar ove **XML** datoteke dodane su željene boje, nazivi, dimenzije, raspored izgleda i slično.



Slika 19: activity_main.xml

- ZarazeneDrzave

Gore navedena klasa zapravo prikazuje cijelu listu zaraženih država. Za prikaz svih zaraženih država opet je odgovoran **JSON** koji podatke dohvaća s druge internetske lokacije (https://corona.lmao.ninja/v2/countries/). Ova lokacija prikazuje sve detalje vezane uz pojedinu državu.

```
public class ZarazeneDrzave extends AppCompatActivity {

    EditText edtSearch;
    ListView listView;
    SimpleArcLoader simpleArcLoader;

    public static List<DrzavePrikaz> countryModelsList = new ArrayList<>();
    DrzavePrikaz countryModel;
    DrzaveLista myCustomAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_zarazene_drzave);

        edtSearch = findViewById(R.id.edtSearch);
        listView = findViewById(R.id.listView);
        simpleArcLoader = findViewById(R.id.loader);

        getSupportActionBar().setTitle("Zaražene države");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);

        fetchData();

        listView.setOnItemClickListener((parent, view, position, id) -> {
                startActivity(new Intent(getApplicationContext(), DetaljiPrikaz.class).putExtra( name: "position",position));
```

Slika 20: ZarazeneDrzave a)

Osim što je moguće pregledavati cijeli popis ili listu zaraženih država postoji i mogućnost pretraživanja. Pretraživanje je implementirano radi lakšeg i bržeg pronalaženja države koju korisnik želi analizirati.

```
edtSearch.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {

        myCustomAdapter.getFilter().filter(s);
        myCustomAdapter.notifyDataSetChanged();

    }

    @Override
    public void afterTextChanged(Editable s) {

    }
});
```

Slika 21: ZarazeneDrzave b)

- DrzaveLista

Za prikaz liste država potrebno je dohvatiti nazive država i zastavu za svaku državu koja ju obilježava:

```
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {

    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.drzave_lista, root: null, attachToRoot: true);
    TextView tvCountryName = view.findViewById(R.id.tvCountryName);
    ImageView imageView = view.findViewById(R.id.imageFlag);

    tvCountryName.setText(countryModelsListFiltered.get(position).getCountry());
    Glide.with(context).load(countryModelsListFiltered.get(position).getFlag()).into(imageView);

    return view;
}
```

Slika 22: DrzaveLista a)

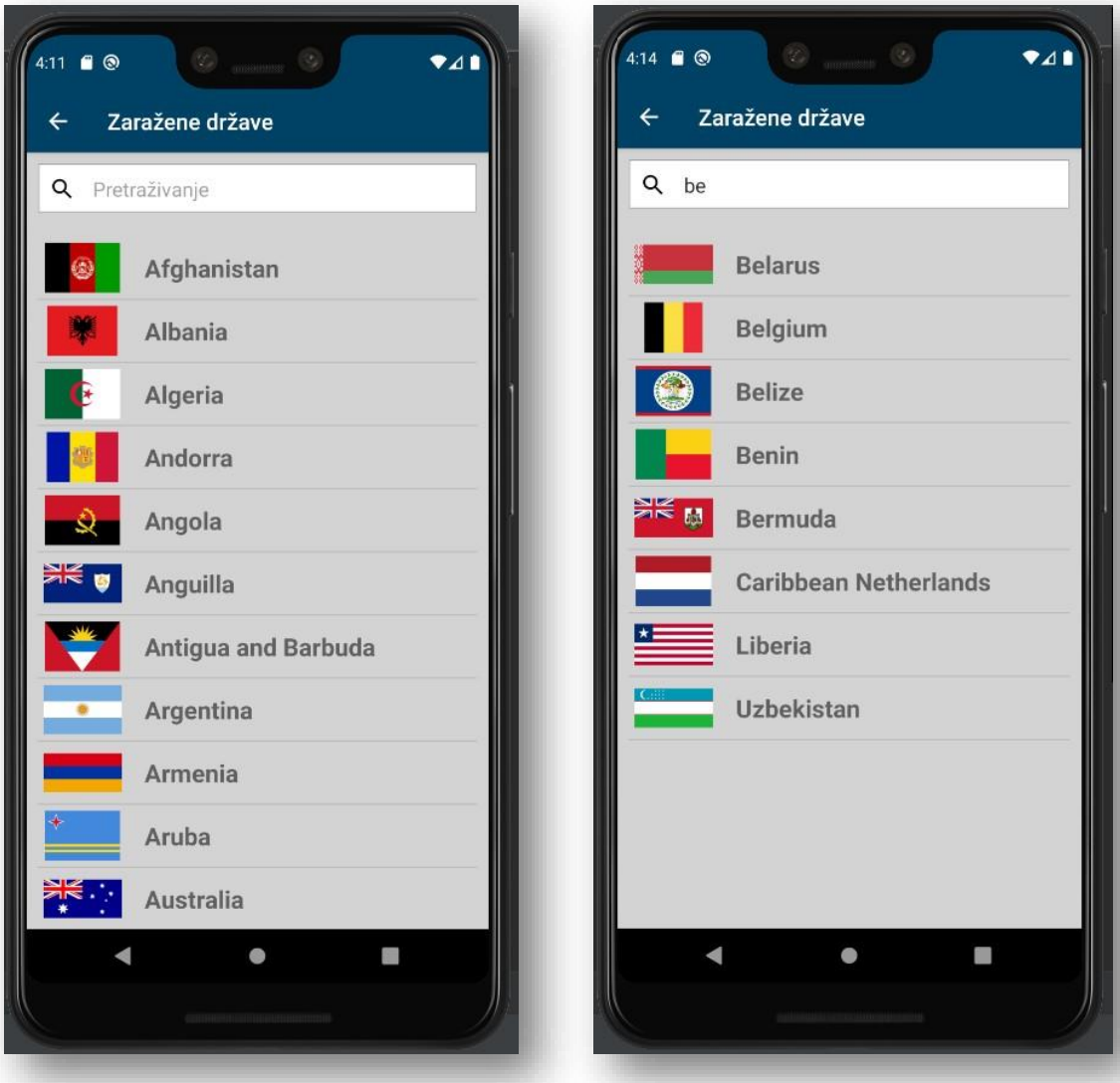
S obzirom da je broj država velik i pretraživanje treba što više ubrzati potrebno je uvesti dio koji će to izvoditi. Ovisno o tome koju državu pretražujemo, aplikacija će nam izbacivati listu država s korijenom njezinog naziva.

```
public Filter getFilter() {
        Filter filter = new Filter() {
        @Override
        protected FilterResults performFiltering(CharSequence constraint) {

            FilterResults filterResults = new FilterResults();
            if(constraint == null || constraint.length() == 0){
                filterResults.count = countryModelsList.size();
                filterResults.values = countryModelsList;

            }else{
                List<DrzavePrikaz> resultsModel = new ArrayList<>();
                String searchStr = constraint.toString().toLowerCase();

                for(DrzavePrikaz itemsModel:countryModelsList){
                    if(itemsModel.getCountry().toLowerCase().contains(searchStr)){
                        resultsModel.add(itemsModel);

                    }
                    filterResults.count = resultsModel.size();
                    filterResults.values = resultsModel;
                }


            }

            return filterResults;
```

Slika 23: DrzavaLista b)

```
protected void publishResults(CharSequence constraint, FilterResults results) {

    countryModelsListFiltered = (List<DrzavePrikaz>) results.values;
    ZarazeneDrzave.countryModelsList = (List<DrzavePrikaz>) results.values;
    notifyDataSetChanged();

}
```

Slika 24: DrzavaLista c)



Slika 25: Filtering

- DrzavePrikaz

```java
public class DrzavePrikaz {
    private String flag,country,cases,todayCases,deaths,todayDeaths,recovered,active,critical;

    public DrzavePrikaz() {
    }

    public DrzavePrikaz(String flag, String country, String cases, String todayCases, String deaths, String todayDeaths, String recovered,
                        String active, String critical) {
        this.flag = flag;
        this.country = country;
        this.cases = cases;
        this.todayCases = todayCases;
        this.deaths = deaths;
        this.todayDeaths = todayDeaths;
        this.recovered = recovered;
        this.active = active;
        this.critical = critical;
    }

    public String getFlag() { return flag; }

    public void setFlag(String flag) { this.flag = flag; }

    public String getCountry() { return country; }

    public void setCountry(String country) { this.country = country; }
```

Slika 26: DrzavePrikaz

Kako bi prikaz bio zapravo omogućen potrebno je kreirati String sa svim atributima koji se nalaze na poslužitelju (https://corona.lmao.ninja/v2/countries/).

- DetaljiPrikaz

```java
public class DetaljiPrikaz extends AppCompatActivity {

    private  int positionCountry;
    TextView tvCountry,tvCases,tvRecovered,tvCritical,tvActive,tvTodayCases,tvTotalDeaths,tvTodayDeaths;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detalji_prikaz);

        Intent intent = getIntent();
        positionCountry = intent.getIntExtra( name: "position", defaultValue: 0);

        getSupportActionBar().setTitle("Detalji za: "+ ZarazeneDrzave.countryModelsList.get(positionCountry).getCountry());
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);

        tvCountry = findViewById(R.id.tvCountry);
        tvCases = findViewById(R.id.tvCases);
        tvRecovered = findViewById(R.id.tvRecovered);
        tvCritical = findViewById(R.id.tvCritical);
        tvActive = findViewById(R.id.tvActive);
        tvTodayCases = findViewById(R.id.tvTodayCases);
        tvTotalDeaths = findViewById(R.id.tvDeaths);
        tvTodayDeaths = findViewById(R.id.tvTodayDeaths);

        tvCountry.setText(ZarazeneDrzave.countryModelsList.get(positionCountry).getCountry());
        tvCases.setText(ZarazeneDrzave.countryModelsList.get(positionCountry).getCases());
```

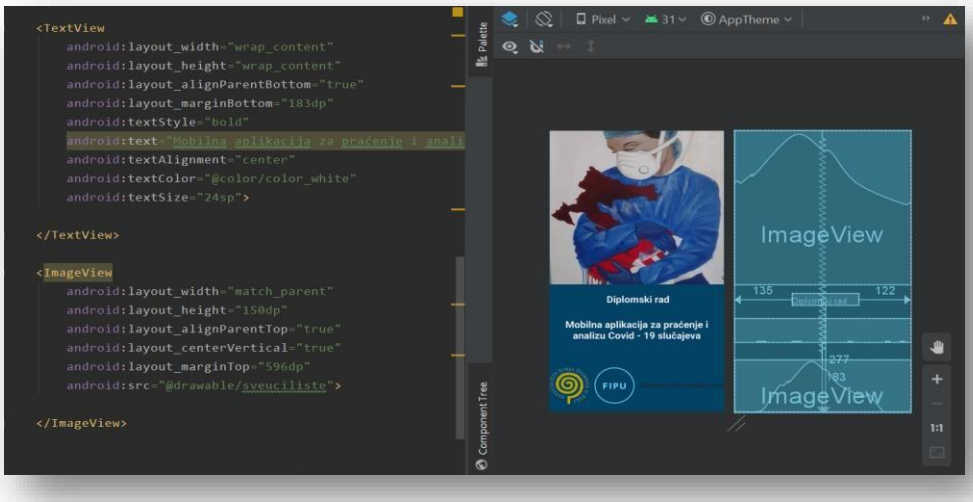
Slika 27: DetaljiPrikaz

Sam naziv klase govori o čemu se zapravo radi. U ovoj su klasi prikazani svi detalji vezani uz državu koja je izabrana za pregled i analizu. Svaka država ima svoje detalje, odnosno podatke koji su vezani za nju u trenutku pregledavanja. Svi objekti prikazani na slici 27 rezultirat će odgovarajućim podacima ovisno o tome koju smo državu izrabrali.

- Splash

Splash Screen je najčešće prvi zaslon za pokretanje koji se pojavljuje kada se aplikacija otvori. Drugim riječima, to je jednostavan stalni zaslon ograničen na određeno vrijeme koji se koristi za prikaz logotipa tvrtke, naziva, oglasnog sadržaja itd.

Slika 28: Splash Screen a)

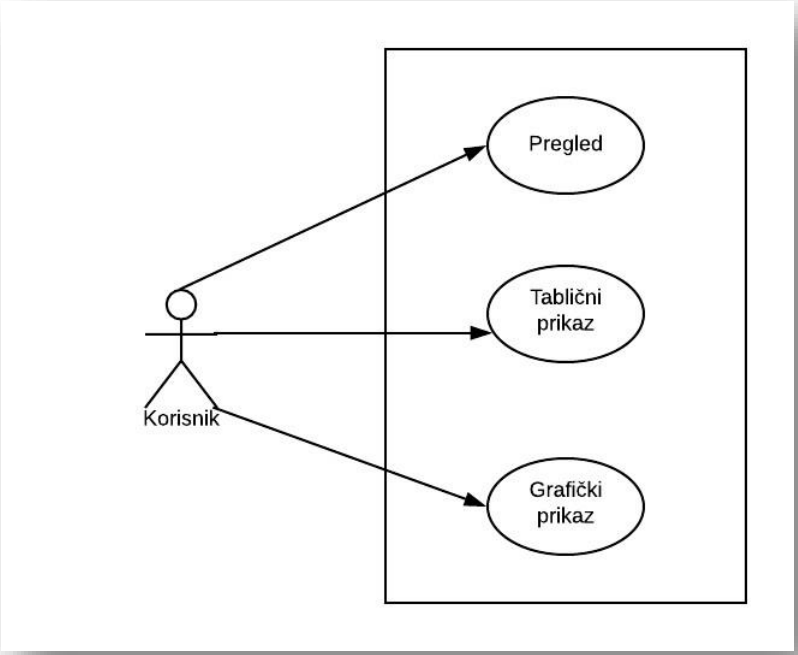


Slika 29: Splash Screen b)

3.2. Način primjene kroz web dio – RStudio

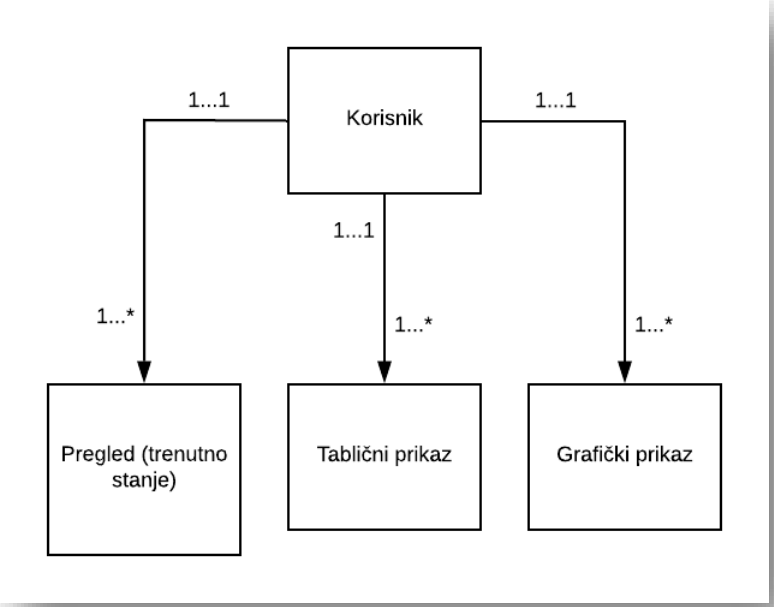
Za web dio aplikacije zaslužan je RStudio. Osim što je aplikacija izrađena uz pomoć gore navedenog programa, istovremeno omogućuje i postavljanje aplikacije na internet. U trenutku postavljanja aplikacija postaje „živa“, što znači da joj mogu pristupiti svi koji imaju poveznicu na istu. Bez ove mogućnosti aplikaciju bi bilo moguće pokrenuti samo ako korisnik posjeduje cijeli projekt. Postavljanje aplikacije će biti kasnije objašnjeno detaljno.

Slično kao i za mobilni dio potrebno je prikazati i objasniti funkcionalnosti, dijagrame te korisničke scenarije. Na slici 30 prikazan je Use-Case dijagram:

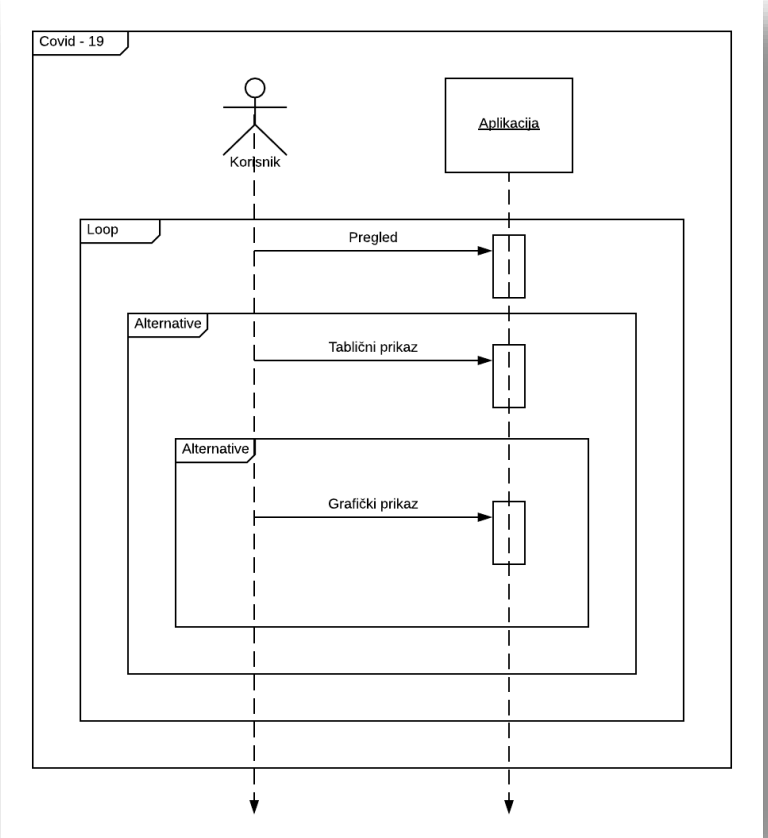


Slika 30: Use-Case dijagram (web)

Web dio prikazuje malo detaljnije razvoj podataka upravo iz razloga što je RStudio vrlo moćan program za baratanje sa statističkim podacima. U nastavku su dodani klasni i sequence dijagrami koji su opisani u prethodnom poglavlju.



Slika 31: Klasni dijagram (web)



Slika 32: Use-sequence dijagram (web)

U nastavku će biti prikazan način izrade te prikaz i opis temeljnih funkcionalnosti:

- global.R (main)

Unutar **global.R** izvršava se uvoz svih potrebnih biblioteka i paketa, učitavanje datoteka iz željenog direktorija, učitavanje i osvježavanje podataka s Githuba te razvoj podataka ovisno o državi.

```r
# Učitavanje potrebnih biblioteka
library("shiny")
library("shinydashboard")
library("tidyverse")
library("leaflet")
library("plotly")
library("DT")
library("fs")
library("wbstats")
```

Slika 33: Učitavanje biblioteka/paketa

```r
# Preuzimanje podataka (Johns Hopkins)
downloadGithubData <- function() {
  download.file(
    url      = "https://github.com/CSSEGISandData/COVID-19/archive/master.zip",
    destfile = "data/covid19_data.zip"
  )

  data_path <- "COVID-19-master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_"
  unzip(
    zipfile   = "data/covid19_data.zip",
    files     = paste0(data_path, c("confirmed_global.csv", "deaths_global.csv", "recovered_global.csv")),
    exdir     = "data",
    junkpaths = T
  )
}
```

Slika 34: Preuzimanje podataka

```r
updateData <- function() {
  # Preuzimanje podataka (Johns Hopkins) (https://github.com/CSSEGISandData/COVID-19) ako su stariji od pola sata. Dolje je prikazan kao "hours".
  if (!dir_exists("data")) {
    dir.create('data')
    downloadGithubData()
  } else if ((!file.exists("data/covid19_data.zip")) || (as.double(Sys.time()
                                                                   - file_info("data/covid19_data.zip")$change_time, units = "hours") > 0.5)) {
    downloadGithubData()
  }
}

# Ažuriranje prilikom pokretanja aplikacije
updateData()
```

Slika 35: Ažuriranje podataka

Ovo je jedna od najbitnijih funkcija jer se svaki dan podaci o slučajevima vezanim uz Covid – 19 virus mijenjaju. Ako bi se ovaj dio izostavio, aplikacija bi prikazivala samo one podatke koji su preuzeti prvi put, odnosno koji su učitani prvi put u projekt.

```
# Razvoj slučajeva po državama
data_confirmed_sub <- data_confirmed %>%
  pivot_longer(names_to = "date", cols = 5:ncol(data_confirmed)) %>%
  group_by(`Province/State`, `Country/Region`, date, Lat, Long) %>%
  summarise("confirmed" = sum(value, na.rm = T))

data_deceased_sub <- data_deceased %>%
  pivot_longer(names_to = "date", cols = 5:ncol(data_deceased)) %>%
  group_by(`Province/State`, `Country/Region`, date, Lat, Long) %>%
  summarise("deceased" = sum(value, na.rm = T))

data_evolution <- data_confirmed_sub %>%
  full_join(data_deceased_sub) %>%
  ungroup() %>%
  mutate(date = as.Date(date, "%m/%d/%y")) %>%
  arrange(date) %>%
  group_by(`Province/State`, `Country/Region`, Lat, Long) %>%
  mutate(
    recovered = lag(confirmed, 14, default = 0) - deceased,
    recovered = ifelse(recovered > 0, recovered, 0),
    active = confirmed - recovered - deceased
  ) %>%
  pivot_longer(names_to = "var", cols = c(confirmed, recovered, deceased, active)) %>%
  ungroup()
```

Slika 36: Razvoj po državama

Nakon što su potvrđeni i preminuli slučajevi grupirani, potrebno je prikazati daljni razvoj. Pomoću **data_evolution** prikazivat će se razvoj slučajeva od početka pandemije sve do trenutnog stanja.

```
# Prikaz novih slučajeva
data_evolution <- data_evolution %>%
  group_by(`Province/State`, `Country/Region`) %>%
  mutate(value_new = value - lag(value, 4, default = 0)) %>%
  ungroup()
```

Slika 37: Prikaz novih slučajeva

```
# Prikaz podataka od početka pandemije do trenutnog stanja
data_evolution <- data_evolution %>%
  left_join(population, by = c("Country/Region" = "country"))
rm(population, countryNamesPop, countryNamesDat, noDataCountries)

data_atDate <- function(inputDate) {
  data_evolution[which(data_evolution$date == inputDate),] %>%
    distinct() %>%
    pivot_wider(id_cols = c("Province/State", "Country/Region", "date", "Lat", "Long", "population"), names_from = var, values_from = value) %>%
    filter(confirmed > 0 |
             recovered > 0 |
             deceased > 0 |
             active > 0)
}
```

Slika 38: Početno/trenutno stanje

Country/Region	date	Lat	Long	var	value	value_new	population
China	2020-01-22	31.8257	117.2264	confirmed	1	1	1402112000

Slika 39: Početno stanje - Rstudio

```
# Najnoviji podaci
data_latest <- data_atDate(max(data_evolution$date))
```

Slika 40: Najnoviji podaci

Country/Region	date	Lat	Long	population	confirmed	recovered	deceased	active
Cote d'Ivoire	2021-08-22	7.54000	-5.54710	25069226	53645	50985	395	2265
Croatia	2021-08-22	45.10000	15.20000	4047200	369765	356744	8301	4720

Slika 41: Najnoviji podaci - RStudio

```
# Najzaraženije države (s najvećim brojem slučajeva)
top5_countries <- data_evolution %>%
  filter(var == "active", date == current_date) %>%
  group_by(`Country/Region`) %>%
  summarise(value = sum(value, na.rm = T)) %>%
  arrange(desc(value)) %>%
  top_n(5) %>%
  select(`Country/Region`) %>%
  pull()

# Provjera
top5_countries
```

Slika 42: Najzaraženije države

Funkcija iznad prikazuje najzaraženije države na temelju najnovijih podataka. Ispod će biti prikazano rješenje unutar programa i izvješće s interneta radi usporedbe:

United States	37294141
India	32449306
Brazil	20494212
UK	6747087
Iran	4715771

```
> # Provjera
> top5_countries
[1] "US"             "Iran"           "India"          "United Kingdom" "Brazil"
```

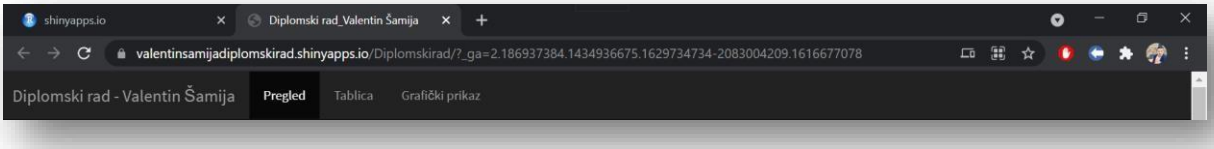
Slika 43: Prikaz 5 najzaraženijih država

- ui.R (korisničko sučelje)

Za korisničko sučelje kreirano je više dijelova; pregled, tablični prikaz te grafički prikaz, odnosno pregled. Za početni dio korišten je **fluid_page**. Fluid stranice skaliraju svoje komponente u stvarnom vremenu kako bi ispunile svu dostupnu širinu preglednika. Na slici 45 se to najbolje vidi.

```
source("ui/ui_pregled.R", local = TRUE)
source("ui/ui_tablica.R", local = TRUE)
source("ui/ui_graf_prikaz.R", local = TRUE)

ui <- fluidPage(
  title = "Diplomski rad_Valentin Šamija",
  tags$head(
  ),
  tags$style(type = "text/css", ".container-fluid {padding-left: 0px; padding-right: 0px !important;}"),
  tags$style(type = "text/css", ".navbar {margin-bottom: 0px;}"),
  tags$style(type = "text/css", ".content {padding: 0px;}"),
  tags$style(type = "text/css", ".row {margin-left: 0px; margin-right: 0px;}"),
  tags$style(HTML(".col-sm-12 { padding: 5px; margin-bottom: -15px; }")),
  tags$style(HTML(".col-sm-6 { padding: 5px; margin-bottom: -15px; }")),
  navbarPage(
    title       = div("Diplomski rad - Valentin Šamija", style = "padding-left: 10px"),
    inverse = TRUE,
    collapsible = TRUE,
    fluid       = TRUE,
    tabPanel("Pregled", page_overview, value = "page-overview"),
    tabPanel("Tablica", page_fullTable, value = "page-fullTable"),
    tabPanel("Grafički prikaz", page_plots, value =  "page_plots"),
    tags$script(HTML("var header = $('.navbar > .container-fluid');
    console.log(header)")
    )
  )
)
```

Slika 44: UI – FluidPage a)



Slika 45: UI – FluidPage b)

- kljucni_podaci.R

Ključni podaci prikazuju stanje na trenutni datum (današnji). Podaci se automatski ažuriraju te se prikazuje broj aktivnih, oporavljenih, umrlih te broj zaraženih država. Vrijeme ažuriranja je jasno prikazano.

```
key_figures <- reactive({
  data           <- sumData(input$timeSlider)
  data_yesterday <- sumData(input$timeSlider - 1)

  data_new <- list(
    new_confirmed = (data$confirmed - data_yesterday$confirmed) / data_yesterday$confirmed * 100,
    new_recovered = (data$recovered - data_yesterday$recovered) / data_yesterday$recovered * 100,
    new_deceased  = (data$deceased - data_yesterday$deceased) / data_yesterday$deceased * 100,
    new_countries = data$countries - data_yesterday$countries
  )

  keyFigures <- list(
    "confirmed" = HTML(paste(format(data$confirmed, big.mark = " "), sprintf("<h4>(%+.1f %%)</h4>", data_new$new_confirmed))),
    "recovered" = HTML(paste(format(data$recovered, big.mark = " "), sprintf("<h4>(%+.1f %%)</h4>", data_new$new_recovered))),
    "deceased"  = HTML(paste(format(data$deceased, big.mark = " "), sprintf("<h4>(%+.1f %%)</h4>", data_new$new_deceased))),
    "countries" = HTML(paste(format(data$countries, big.mark = " "), "", sprintf("<h4>(%+d)</h4>", data_new$new_countries)))
  )
  return(keyFigures)
})
```

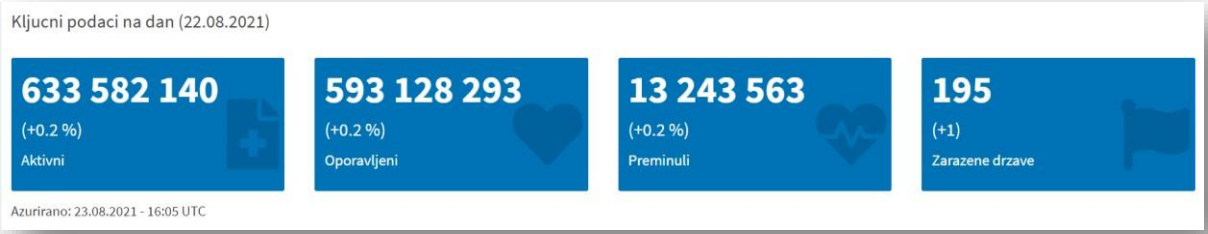
Slika 46: Ključni podaci a)

```
output$valueBox_confirmed <- renderValueBox({
  valueBox(
    key_figures()$confirmed,
    subtitle = "Aktivni",
    icon     = icon("file-medical"),
    color    = "blue",
    width    = NULL
  )
})


output$valueBox_recovered <- renderValueBox({
  valueBox(
    key_figures()$recovered,
    subtitle = "Oporavljeni",
    icon     = icon("heart"),
    color    = "blue"
  )
})

output$valueBox_deceased <- renderValueBox({
  valueBox(
    key_figures()$deceased,
    subtitle = "Preminuli",
    icon     = icon("heartbeat"),
    color    = "blue"
  )
})

output$valueBox_countries <- renderValueBox({
  valueBox(
    key_figures()$countries,
    subtitle = "Zarazene drzave",
    icon     = icon("flag"),
    color    = "blue"
  )
})
```

Slika 47: Ključni podaci b)

```
output$box_keyFigures <- renderUI(box(
  title = paste0("Kljucni podaci na dan (", strftime(input$timeSlider, format = "%d.%m.%Y"), ")"),
  fluidRow(
    column(
      valueBoxOutput("valueBox_confirmed", width = 3),
      valueBoxOutput("valueBox_recovered", width = 3),
      valueBoxOutput("valueBox_deceased", width = 3),
      valueBoxOutput("valueBox_countries", width = 3),
      width = 12,
      style = "margin-left: -20px"
    )
  ),
  div("Azurirano: ", strftime(changed_date, format = "%d.%m.%Y - %R %Z")),
  width = 12
))
```

Slika 48: Ključni podaci c)



Slika 49: Ključni podaci d)

- map.R

Osim tabličnog i grafičkog prikaza dodana je i karta za detaljni prikaz. Pomoću karte najbolje se vidi stanje u kojem se cijeli svijet nalazi. Svaka vrsta slučaja je prikazana drugom bojom. Npr. potvrđeni slučajevi su označeni crvenom bojom, a oporavljeni zelenom.

```
map <- leaflet(addLabel(data_latest)) %>%
  setMaxBounds(-180, -90, 180, 90) %>%
  setView(0, 20, zoom = 2) %>%
  addTiles() %>%
  addProviderTiles(providers$CartoDB.Positron, group = "Karta") %>%
  addProviderTiles(providers$HERE.satelliteDay, group = "Satelit") %>%
  addLayersControl(
    baseGroups    = c("Karta", "Satelit"),
    overlayGroups = c("Confirmed", "Confirmed (per capita)", "Estimated Recoveries", "Deceased", "Active", "Active (per capita)")
  ) %>%
  hideGroup("Confirmed (per capita)") %>%
  hideGroup("Estimated Recoveries") %>%
  hideGroup("Deceased") %>%
  hideGroup("Active") %>%
  hideGroup("Active (per capita)") %>%
  addEasyButton(easyButton(
    icon    = "glyphicon glyphicon-globe", title = "Reset zoom",
    onClick = JS("function(btn, map){ map.setView([20, 0], 2); }"))) %>%
  addEasyButton(easyButton(
    icon    = "glyphicon glyphicon-map-marker", title = "Locate Me",
    onClick = JS("function(btn, map){ map.locate({setView: true, maxZoom: 6}); }")))

observe({
  req(input$timeSlider, input$overview_map_zoom)
  zoomLevel              <- input$overview_map_zoom
  data                   <- data_atDate(input$timeSlider) %>% addLabel()
  data$confirmedPerCapita <- data$confirmed / data$population * 100000
  data$activePerCapita    <- data$active / data$population * 100000

  leafletProxy("overview_map", data = data) %>%
    clearMarkers() %>%
    addCircleMarkers(
      lng          = ~Long,
      lat          = ~Lat,
      radius       = ~log(confirmed^(zoomLevel / 2)),
      stroke       = FALSE,
      fillOpacity  = 0.5,
      color        = "#FF0000",
      label        = ~label,
      labelOptions = labelOptions(textsize = 15),
      group        = "Confirmed"
    ) %>%
```

Slika 50: Karta a)

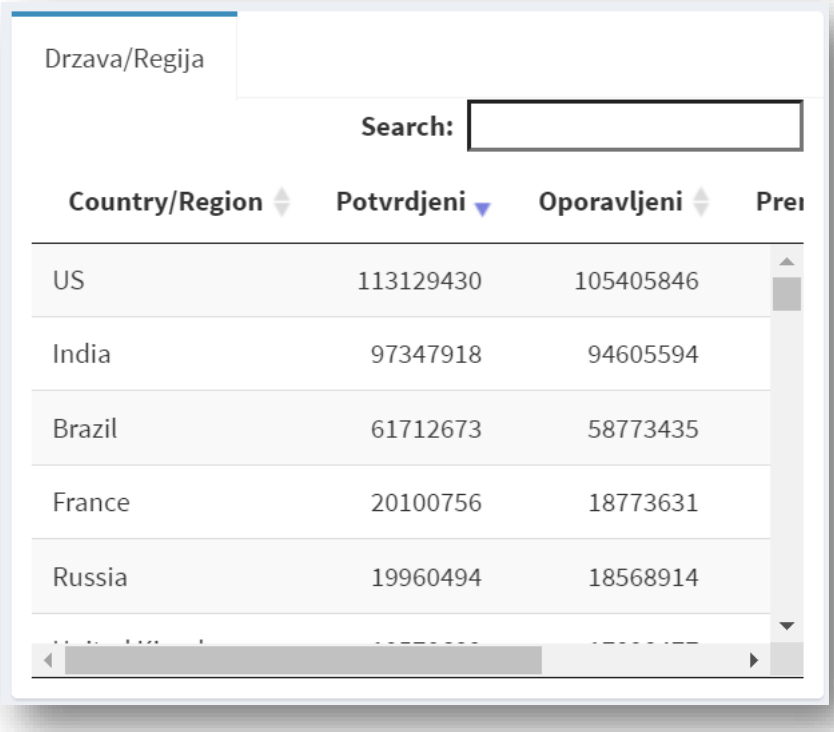
Slika 51: Karta b)

- sazeti_prikaz.R

Sažeti prikaz prikazuje tablicu pokraj karte koja zapravo prikazuje podatke koji su povezani uz države koje imaju najveću koncentraciju slučajeva.

```
output$summaryTables <- renderUI({
  tabBox(
    tabPanel("Drzava/Regija",
      div(
        dataTableOutput("summaryDT_country"),
        style = "margin-top: -10px")
    ),
    width = 12
  )
})

output$summaryDT_country <- renderDataTable(getSummaryDT(data_atDate(current_date), "Country/Region", selectable = TRUE))
proxy_summaryDT_country  <- dataTableProxy("summaryDT_country")

observeEvent(input$timeSlider, {
  data <- data_atDate(input$timeSlider)
  replaceData(proxy_summaryDT_country, summariseData(data, "Country/Region"), rownames = FALSE)
}, ignoreInit = TRUE, ignoreNULL = TRUE)

observeEvent(input$summaryDT_country_row_last_clicked, {
  selectedRow     <- input$summaryDT_country_row_last_clicked
  selectedCountry <- summariseData(data_atDate(input$timeSlider), "Country/Region")[selectedRow, "Country/Region"]
  location        <- data_evolution %>%
    distinct(`Country/Region`, Lat, Long) %>%
    filter(`Country/Region` == selectedCountry) %>%
    summarise(
      Lat  = mean(Lat),
      Long = mean(Long)
    )
  leafletProxy("overview_map") %>%
    setView(lng = location$Long, lat = location$Lat, zoom = 4)
})

summariseData <- function(df, groupBy) {
  df %>%
    group_by(!!sym(groupBy)) %>%
    summarise(
      "Potvrdjeni"            = sum(confirmed, na.rm = T),
      "Oporavljeni" = sum(recovered, na.rm = T),
      "Preminuli"             = sum(deceased, na.rm = T),
      "Aktivni"               = sum(active, na.rm = T)
    ) %>%
    as.data.frame()
}
```

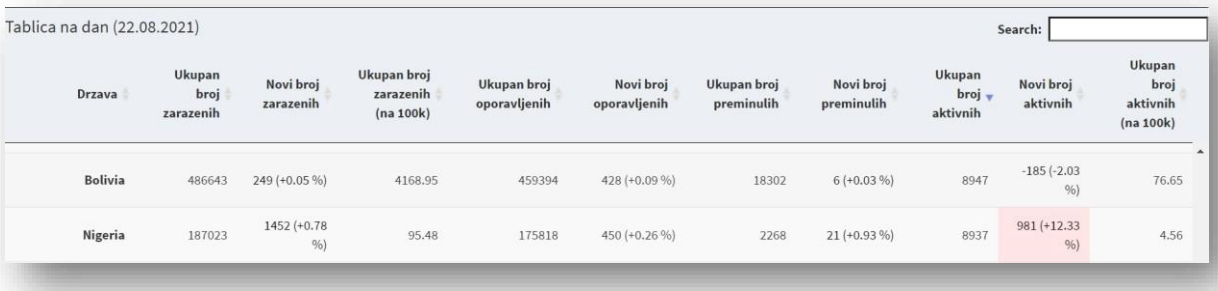
Slika 52: Sažeti prikaz a)



Slika 53: Sažeti prikaz b)

- tablica.R

Tablica prikazuje najdetaljnije podatke vezane uz pandemiju, ažurira sve podatke koji su prikazani u tablici te prikazuje postotke povećanja broja novih/aktivnih slučajeva, odnosno postotak broja oporavljenih. Osim prikaza podataka korisniku je omogućeno i pretraživanje. Ako se dogodi naglo povećanje broja novih slučajeva, podaci će unutar tablice postati „crveni“. Obrnuti slučaj će prikazati naglo padanje, odnosno podaci će postati „zeleni“.



Slika 54: Tablica (naglo povećanje broja slučajeva)

Tablica na dan (22.08.2021)

Search:

Drzava	Ukupan broj zarazenih	Novi broj zarazenih	Ukupan broj zarazenih (na 100k)	Ukupan broj oporavljenih	Novi broj oporavljenih	Ukupan broj preminulih	Novi broj preminulih	Ukupan broj aktivnih	Novi broj aktivnih	Ukupan broj aktivnih (na 100k)
Jamaica	61282	0	2069.53	53488	379 (+0.71 %)	1371	0	6423	-379 (-5.57 %)	216.91
Ghana	114584	627 (+0.55 %)	384.93	107258	1469 (+1.39 %)	968	7 (+0.73 %)	6358	-849 (-11.78 %)	21.36

Slika 55: Tablica (naglo smanjenje broja slučajeva)

```
getFullTableData <- function(groupBy) {
  padding_left <- max(str_length(data_evolution$value_new), na.rm = TRUE)
  data         <- data_evolution %>%
    filter(date == current_date) %>%
    pivot_wider(names_from = var, values_from = c(value, value_new)) %>%
    select(-date, -Lat, -Long) %>%
    add_row(
      "Province/State"      = "World",
      "Country/Region"      = "World",
      "population"          = 7800000000,
      "value_confirmed"     = sum(.$value_confirmed, na.rm = T),
      "value_new_confirmed" = sum(.$value_new_confirmed, na.rm = T),
      "value_recovered"     = sum(.$value_recovered, na.rm = T),
      "value_new_recovered" = sum(.$value_new_recovered, na.rm = T),
      "value_deceased"      = sum(.$value_deceased, na.rm = T),
      "value_new_deceased"  = sum(.$value_new_deceased, na.rm = T),
      "value_active"        = sum(.$value_active, na.rm = T),
      "value_new_active"    = sum(.$value_new_active, na.rm = T)
    ) %>%
    group_by(!!sym(groupBy), population) %>%
    summarise(
      confirmed_total     = sum(value_confirmed, na.rm = T),
      confirmed_new       = sum(value_new_confirmed, na.rm = T),
      confirmed_totalNorm = round(sum(value_confirmed, na.rm = T) / max(population, na.rm = T) * 100000, 2),
      recovered_total     = sum(value_recovered, na.rm = T),
      recovered_new       = sum(value_new_recovered, na.rm = T),
      deceased_total      = sum(value_deceased, na.rm = T),
      deceased_new        = sum(value_new_deceased, na.rm = T),
      active_total        = sum(value_active, na.rm = T),
      active_new          = sum(value_new_active, na.rm = T),
      active_totalNorm    = round(sum(value_active, na.rm = T) / max(population, na.rm = T) * 100000, 2)
    ) %>%
    mutate(
      "confirmed_newPer" = confirmed_new / (confirmed_total - confirmed_new) * 100,
      "recovered_newPer" = recovered_new / (recovered_total - recovered_new) * 100,
      "deceased_newPer"  = deceased_new / (deceased_total - deceased_new) * 100,
      "active_newPer"    = active_new / (active_total - active_new) * 100
    ) %>%
    mutate_at(vars(contains('_newPer')), list(~na_if(., Inf))) %>%
    mutate_at(vars(contains('_newPer')), list(~na_if(., 0))) %>%
    mutate(
      confirmed_new = str_c(str_pad(confirmed_new, width = padding_left, side = "left", pad = "0"), "|",
        confirmed_new, if_else(!is.na(confirmed_newPer), sprintf(" (%+.2f %%)", confirmed_newPer), "")),
      recovered_new = str_c(str_pad(recovered_new, width = padding_left, side = "left", pad = "0"), "|",
        recovered_new, if_else(!is.na(recovered_newPer), sprintf(" (%+.2f %%)", recovered_newPer), "")),
      deceased_new  = str_c(str_pad(deceased_new, width = padding_left, side = "left", pad = "0"), "|",
        deceased_new, if_else(!is.na(deceased_newPer), sprintf(" (%+.2f %%)", deceased_newPer), "")),
      active_new    = str_c(str_pad(active_new, width = padding_left, side = "left", pad = "0"), "|",
        active_new, if_else(!is.na(active_newPer), sprintf(" (%+.2f %%)", active_newPer), ""))
    ) %>%
    select(-population) %>%
    as.data.frame()
}
```

Slika 56: Prikaz podataka unutar tablice

- razvoj_slucajeva.R

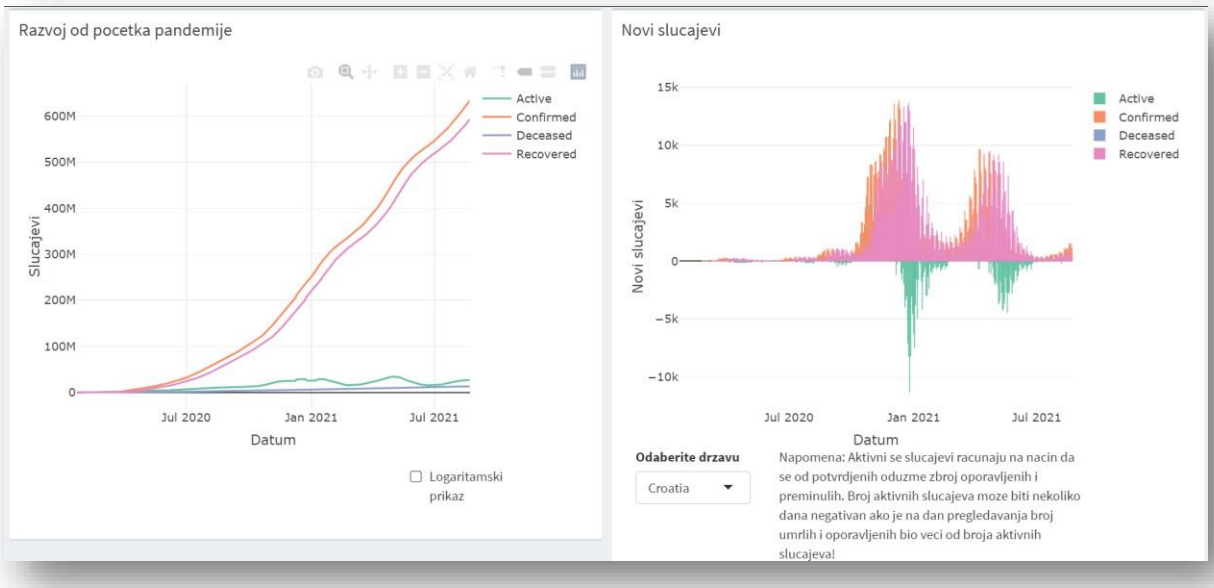
```
getDataByCountry <- function(countries, normalizeByPopulation) {
  req(countries)
  data_confirmed <- data_evolution %>%
    select(`Country/Region`, date, var, value, population) %>%
    filter(`Country/Region` %in% countries &
      var == "confirmed" &
      value > 0) %>%
    group_by(`Country/Region`, date, population) %>%
    summarise("Confirmed" = sum(value, na.rm = T)) %>%
    arrange(date)
  if (nrow(data_confirmed) > 0) {
    data_confirmed <- data_confirmed %>%
      mutate(Confirmed = if_else(normalizeByPopulation, round(Confirmed / population * 100000, 2), Confirmed))
  }
  data_confirmed <- data_confirmed %>% as.data.frame()

  data_recovered <- data_evolution %>%
    select(`Country/Region`, date, var, value, population) %>%
    filter(`Country/Region` %in% countries &
      var == "recovered" &
      value > 0) %>%
    group_by(`Country/Region`, date, population) %>%
    summarise("Estimated Recoveries" = sum(value, na.rm = T)) %>%
    arrange(date)
  if (nrow(data_recovered) > 0) {
    data_recovered <- data_recovered %>%
      mutate(`Estimated Recoveries` = if_else(normalizeByPopulation, round(`Estimated Recoveries` / population * 100000, 2), `Estimated Recoveries`))
  }
  data_recovered <- data_recovered %>% as.data.frame()

  data_deceased <- data_evolution %>%
    select(`Country/Region`, date, var, value, population) %>%
    filter(`Country/Region` %in% countries &
      var == "deceased" &
      value > 0) %>%
    group_by(`Country/Region`, date, population) %>%
    summarise("Deceased" = sum(value, na.rm = T)) %>%
    arrange(date)
  if (nrow(data_deceased) > 0) {
    data_deceased <- data_deceased %>%
      mutate(Deceased = if_else(normalizeByPopulation, round(Deceased / population * 100000, 2), Deceased))
  }
  data_deceased <- data_deceased %>% as.data.frame()

  return(list(
    "confirmed" = data_confirmed,
    "recovered" = data_recovered,
    "deceased"  = data_deceased
  ))
}
```

Slika 57: Razvoj slučajeva

```
output$selectize_casesByCountries_new <- renderUI({
  selectizeInput(
    "selectize_casesByCountries_new",
    label    = "Odaberite drzavu",
    choices  = c("All", unique(data_evolution$`Country/Region`)),
    selected = "All"
  )
})

output$case_evolution_new <- renderPlotly({
  req(input$selectize_casesByCountries_new)
  data <- data_evolution %>%
    mutate(var = sapply(var, capFirst)) %>%
    filter(if (input$selectize_casesByCountries_new == "All") TRUE else `Country/Region` %in% input$selectize_casesByCountries_new) %>%
    group_by(date, var, `Country/Region`) %>%
    summarise(new_cases = sum(value_new, na.rm = T))

  if (input$selectize_casesByCountries_new == "All") {
    data <- data %>%
      group_by(date, var) %>%
      summarise(new_cases = sum(new_cases, na.rm = T))
  }

  p <- plot_ly(data = data, x = ~date, y = ~new_cases, color = ~var, type = 'bar') %>%
    layout(
      yaxis = list(title = "Novi slucajevi"),
      xaxis = list(title = "Datum")
    )
})

output$box_caseEvolution <- renderUI({
  tagList(
    fluidRow(
      box(
        title = "Razvoj od pocetka pandemije",
        plotlyOutput("case_evolution"),
        column(
          checkboxInput("checkbox_logCaseEvolution", label = "Logaritamski prikaz", value = FALSE),
          width = 3,
          style = "float: right; padding: 10px; margin-right: 50px"
        ),
        width = 6
      ),
      box(
        title = "Novi slucajevi",
        plotlyOutput("case_evolution_new"),
        column(
          uiOutput("selectize_casesByCountries_new"),
          width = 3,
        ),
        column(
          HTML("Napomena: Aktivni se slucajevi racunaju na nacin da se od potvrdjenih oduzme zbroj oporavljenih i preminulih.
               Broj aktivnih slucajeva moze biti nekoliko dana negativan ako je na dan pregledavanja broj umrlih i oporavljenih
               bio veci od broja aktivnih slucajeva!"),
```

Slika 58: Grafički prikaz razvoja slučajeva

Razvoj slučajeva prikazan je i tabličnom, ali i u grafičkom pregledu što je prikazano na slici 59:

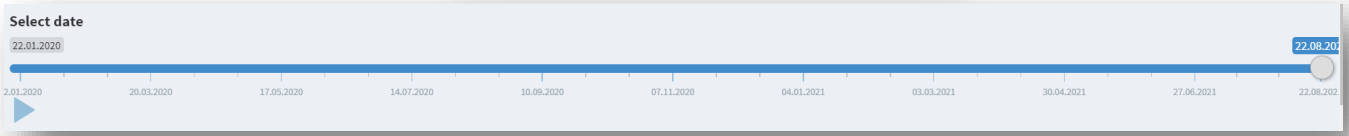


Slika 59: Grafički prikaz razvoja slučajeva

- Slider

Ispod karte, implementirana je i vremenska crta koja prikazuje razvoj pandemije od početka do trenutnog stanja.

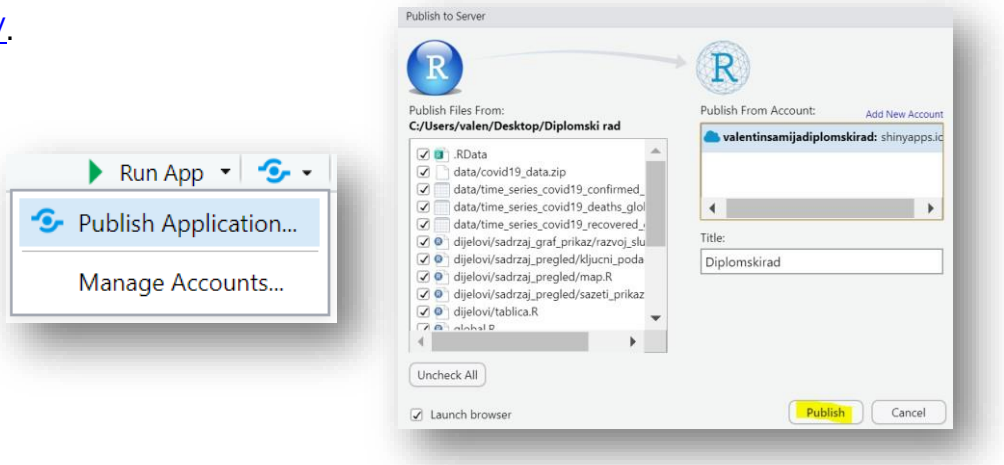
```
column(
  sliderInput(
    "timeSlider",
    label      = "Select Date",
    min        = min(data_evolution$date),
    max        = max(data_evolution$date),
    value      = max(data_evolution$date),
    width      = "100%",
    timeFormat = "%d.%m.%Y",
    animate    = animationOptions(loop = TRUE)
  ),
  class = "slider",
  width = 12
)
```



Slika 60: Vremenska crta (Slider)

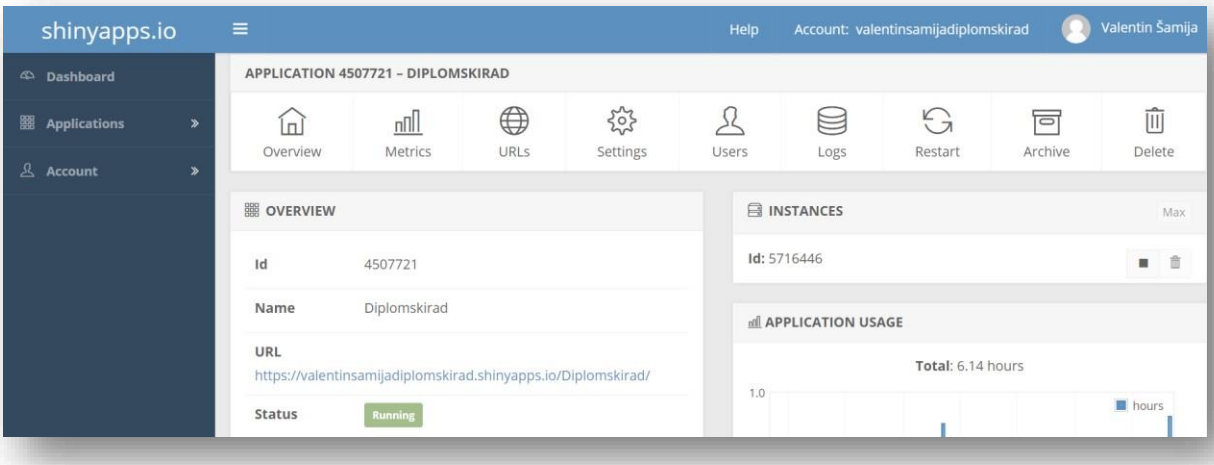
3.3 Postavljanje aplikacije online pomoću shinyapps.io

Da bi aplikacija postala „živa“ za to je odgovoran **Shiny**. Nakon što je aplikacija kreirana moguće ju je pokrenuti unutar RStudia. Osim toga, RStudio nudi i mogućnost poptuno besplatnog postavljanja aplikacije online. U nastavku će biti prikazani koraci za ovu radnju. Najprije je potrebno kreirati korisnički račun na: https://www.shinyapps.io/. Koraci koje je potrebno pratiti nalaze se na sljedećoj poveznici: https://statsandr.com/blog/how-to-publish-shiny-app-example-with-shinyapps-io/.



Slika 61: Publish Application

Nakon postavljanja aplikacije, postoji i kontrolna ploča koja nudi razne mogućnosti:

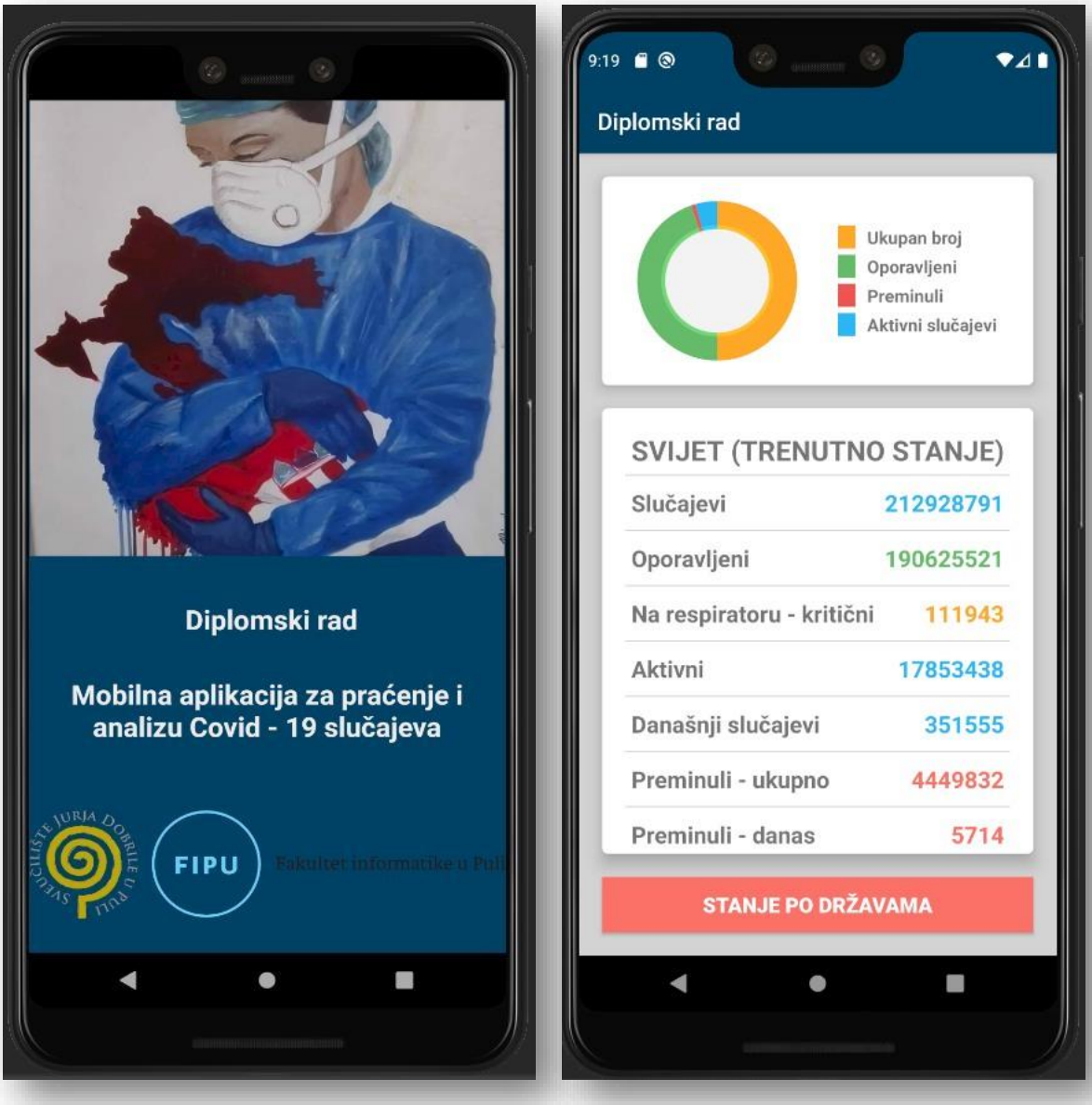


Slika 62: Shinyapps kontrolna ploča

4. KORISNIČKE UPUTE

4.1. Mobile

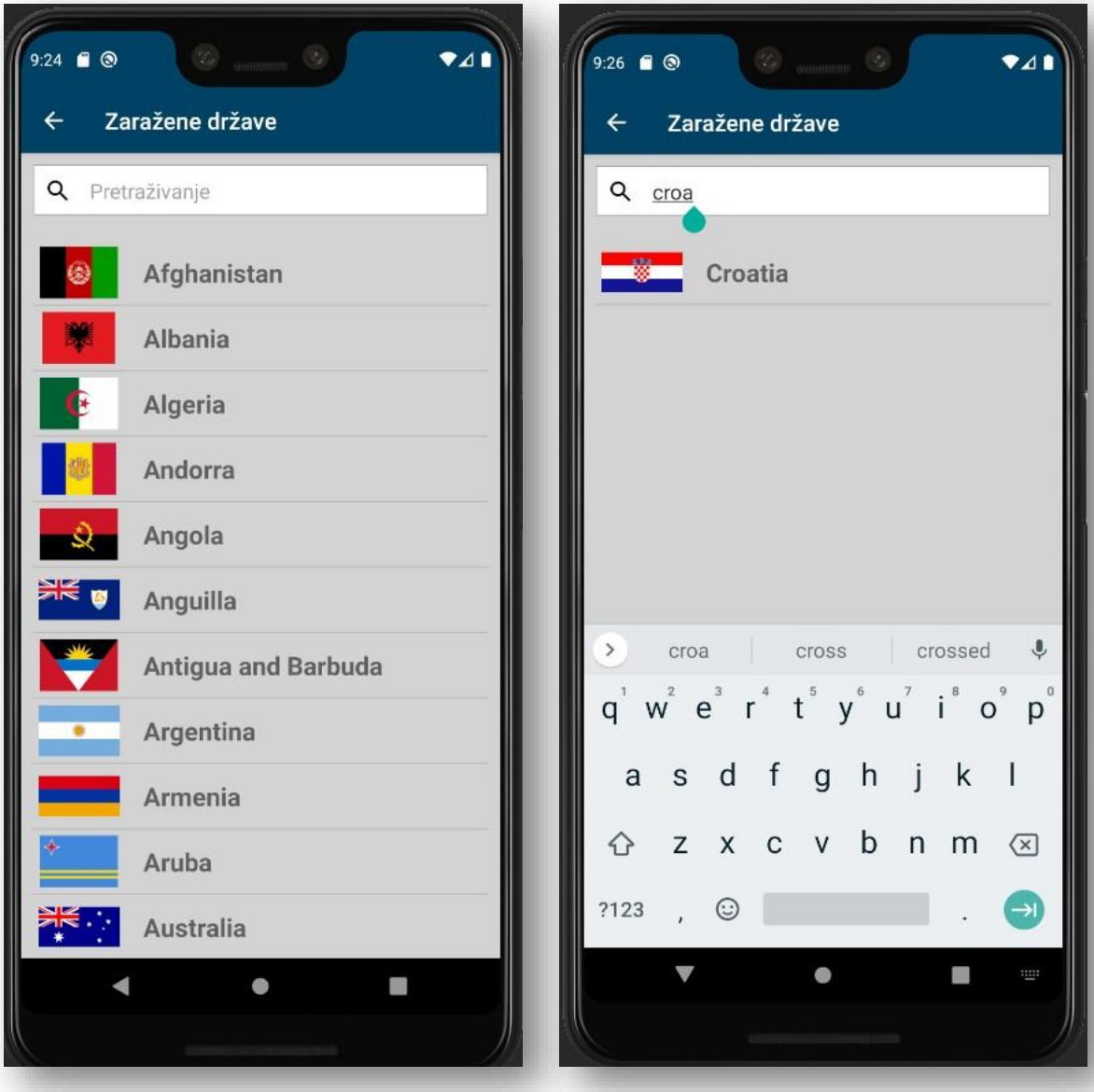
Prilikom pokretanja aplikacije otvara se početni zaslon te nakon kratkog učitavanja prikazuje pregled trenutnog stanja:



Slika 63: Pokretanje aplikacije/trenutno stanje

Kao što se može vidjeti desno na slici 63, prikazano je trenutno stanje. Svi podaci vezani uz trenutno stanje kao što su akitvni slučajevi, oporavljeni i drugi ubrajaju se u tortni grafikon gore iznad. Tortni grafikon zapravo je prikaz analize svih podataka vezanih uz

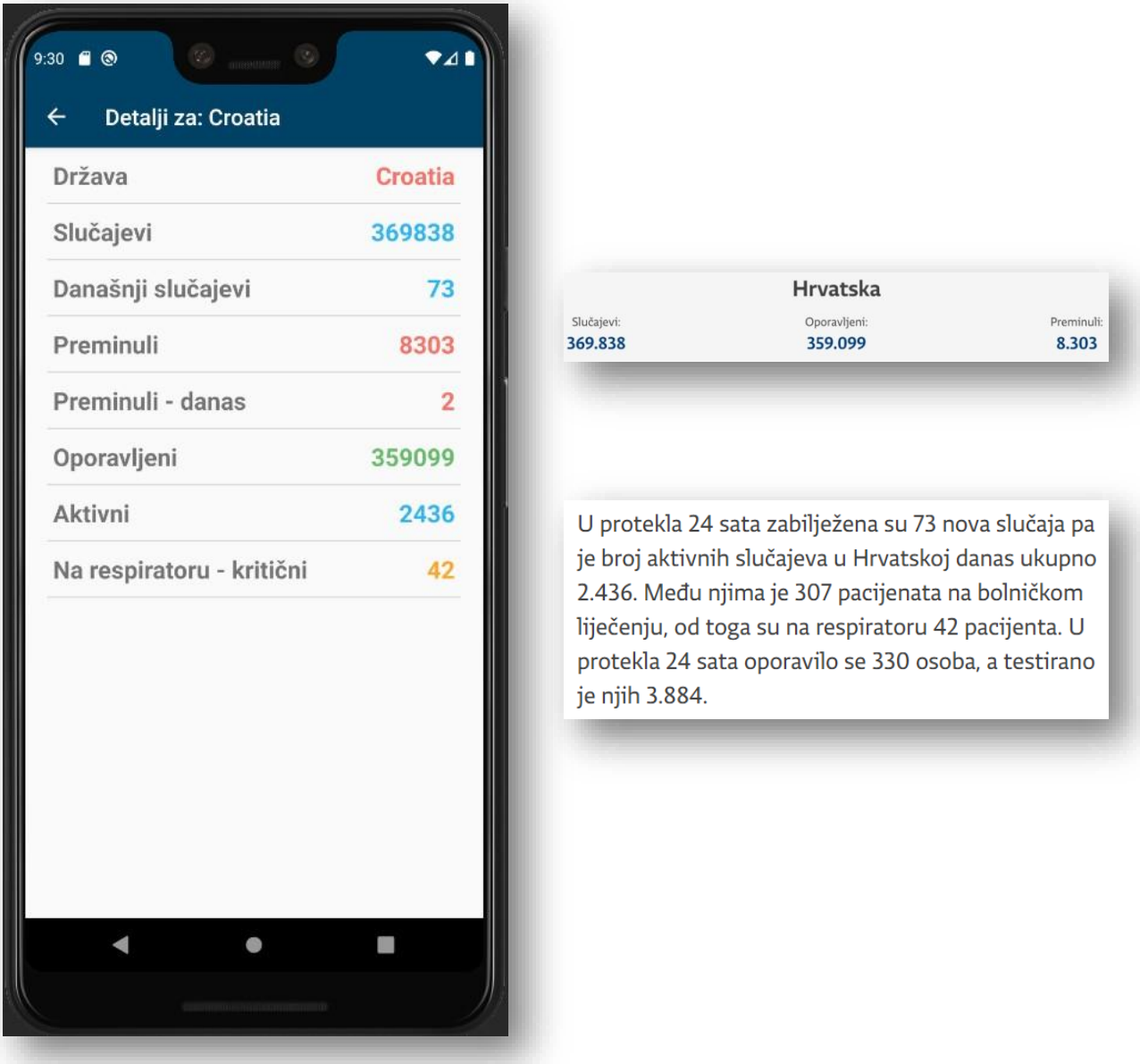
pandemiju. Nakon što je korisnik iščitao podatke koji su ga zanimali, može dodirnuti „STANJE PO DRŽAVAMA“ nakon čega se otvara abecedni popis država koje su zaražene. Korisnik može ručno pregledavati popis država ili jednostavno u tražilicu upisati naziv države za koju želi izvršiti pregled:



Slika 64: Pretraživanje liste država

U trenutku kada korisnik odluči koju državu želi analizirati jednostavno klikne na nju nakon čega se otvara detaljni pregled podataka vezanih uz pandemiju kao što je

prikazano na slici 65. Ujedno će biti prikazana usporedba za trenutnu situaciju s podacima koji se generiraju na sljedećoj poveznici: https://www.koronavirus.hr/.

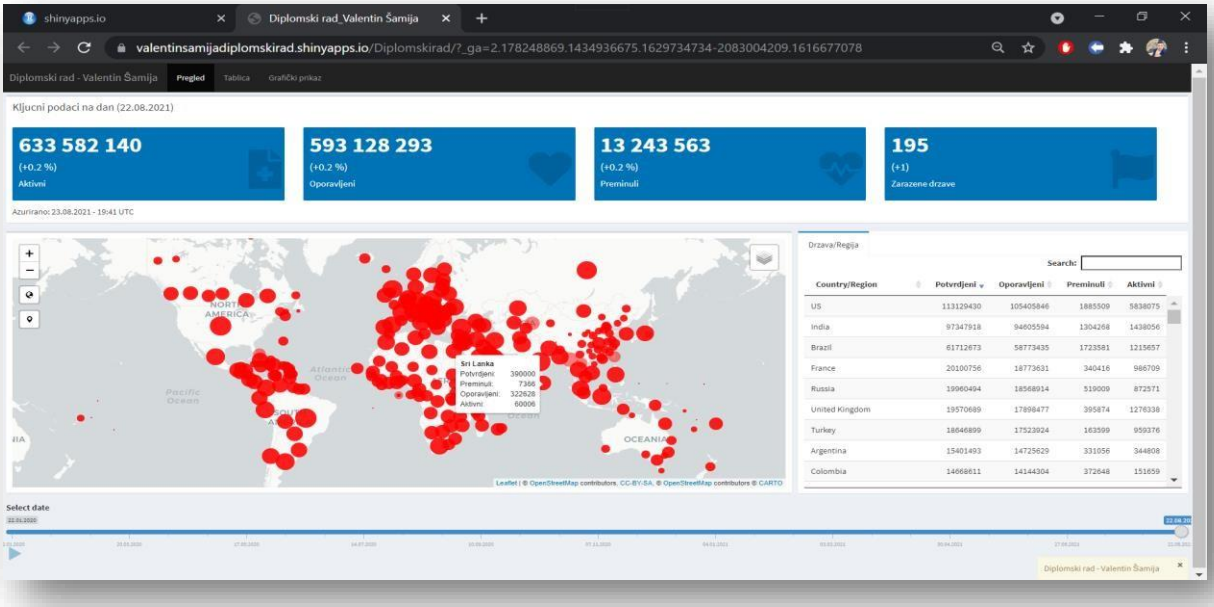


Slika 65: Detaljni prikaz za Hrvatsku/Usporedba

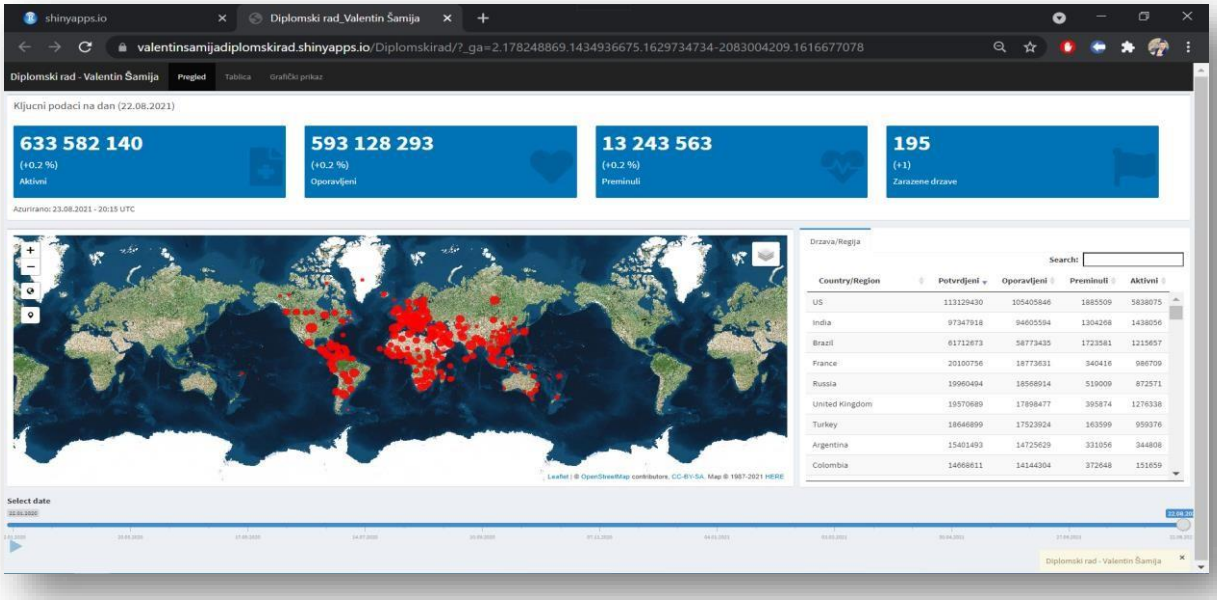
Prilikom usporedbe rada aplikacije i podataka koje generira Hrvatski zavod za javno zdravstvo vidljivo je da su brojke slučajeva identične.

4.2. Web

U trenutku kada korisnik pristupi aplikaciji preko poveznice - https://valentinsamijadiplomskirad.shinyapps.io/Diplomskirad/?_ga=2.246906628.1434936675.1629734734-2083004209.1616677078 otvara se početna stranica (Pregled). Karta ima dvije opcije prikaza; 2D i satelitski prikaz.



Slika 66: Početna web – 2D

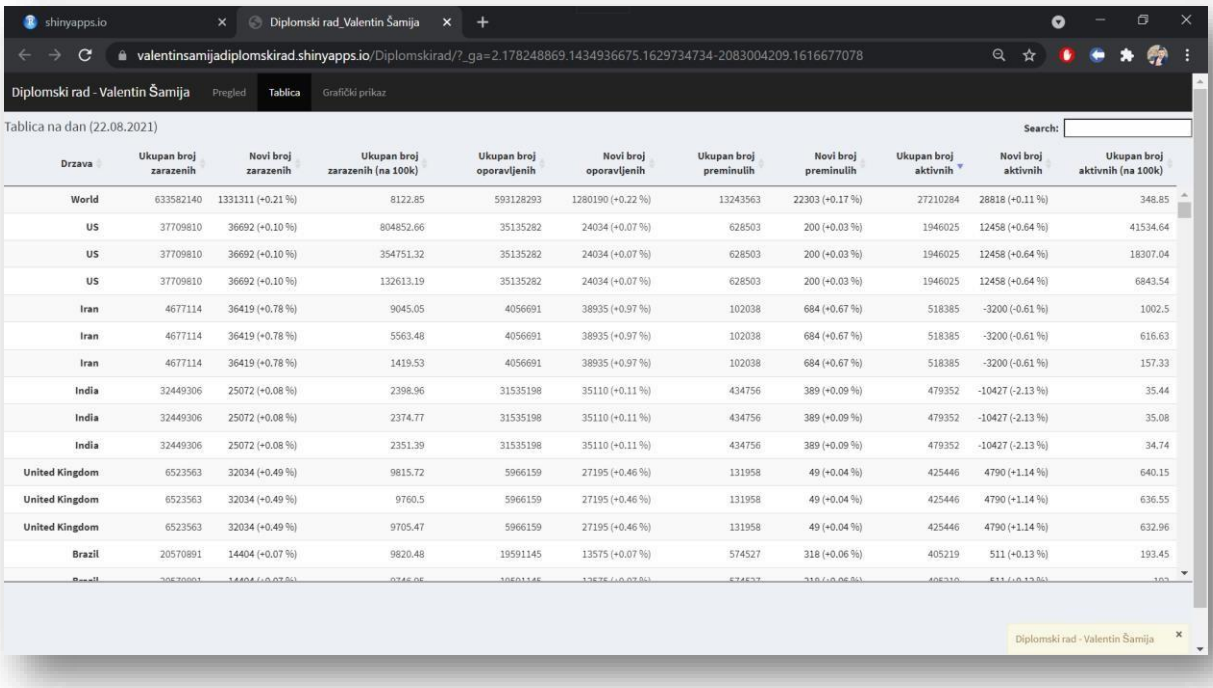


Slika 67: Početna web - satelit

Na početnoj stranici (Pregled) korisniku se pruža mogućnost pregleda detalja na mapi jednostavnim zumiranjem pomoću miša te prelaskom preko „točkica“ kako bi se prikazali detalji. Slika iznad prikazuje podatke za Sri Lanku. Iznad mape prikazani su ključni podaci, označeni plavim pravokutnikom koji se mijenjaju svakoga dana prilikom

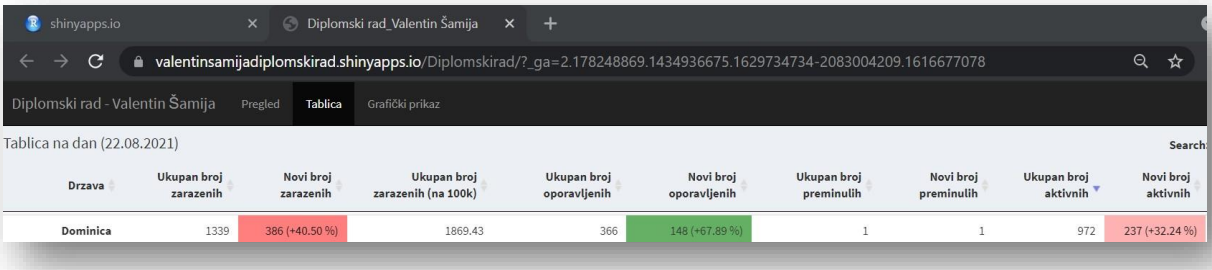
ažuriranja. Osim što se pruža mogućnost pregleda na mapi, na desnoj strani je vidljiv sažeti prikaz, odnosno države s najvećim brojem slučajeva. Podaci se mogu lako iščitati ili pretraživati (unos naziva države ili unos brojke). Ispod mape nalazi se vremenska crta koja omogućuje prikaz od početnog stanja do trenutnog stanja. Razvoj slučajeva prikazan je animacijom. Sve dok animacija traje, desno u sažetom prikazu podaci se mijenjaju, variraju te sortiraju od one države koja ima najveći broj zaraženih prema državi s najmanjim brojem zaraženih.

U samom vrhu početnog zaslona vidljiva su tri izbora: **Pregled**, **Tablica** i **Grafički prikaz**. Slika iznad prikazuje **Pregled**. Klikom na **Tablica**, otvara se sljedeća kartica:



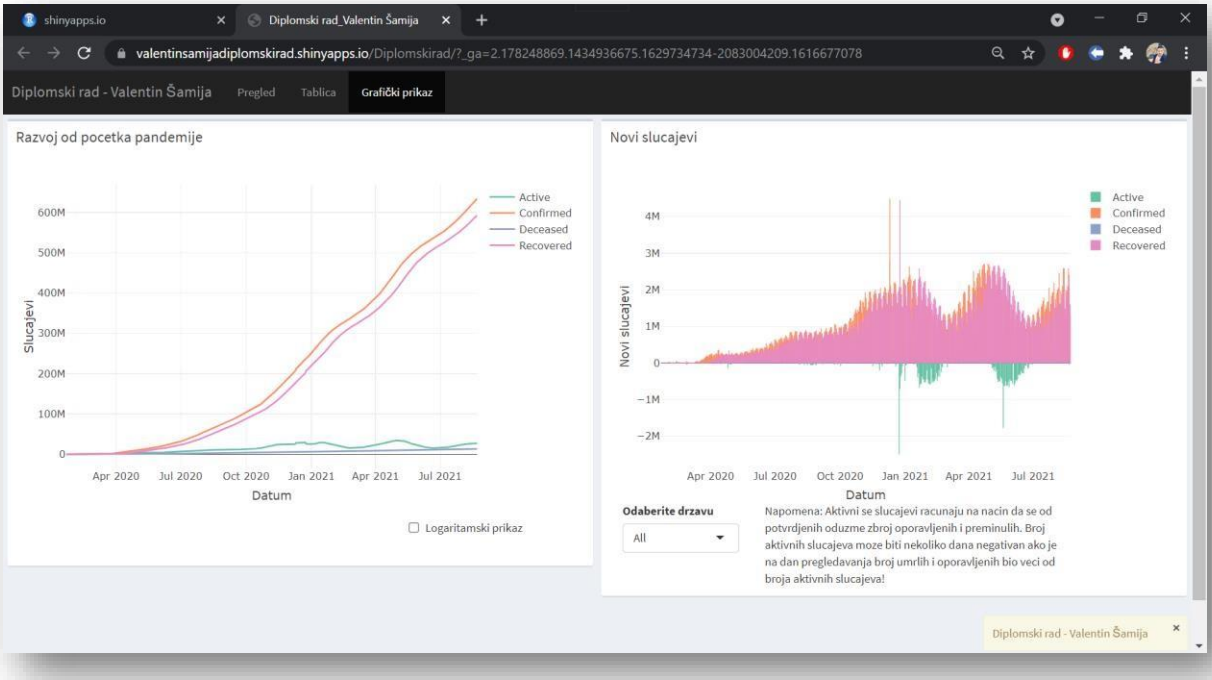
Slika 68: Tablični prikaz a)

Unutar tablice vidljivi su svi podaci vezani uz pandemiju. Omogućeno je ručno pretraživanje pomoću miša ili jednostavno pretraživanje unosom naziva države ili brojke koja označava stupanj zaraženih, oporavljenih, preminulih, aktivnih te unos postotka. Ako se u nekoj državi naglo povećao broj zaraženih, odjeljak u tablici će biti crvene boje, a ako se dogodi suprotno, odnosno nagli porast oporavljenih odjeljak će biti zelene boje. Dominikanska Republika je najbolji primjer za prije navedeno jer se naglo povećao broj zaraženih ujedno i aktivnih, ali istovremeno i broj oporavljenih:



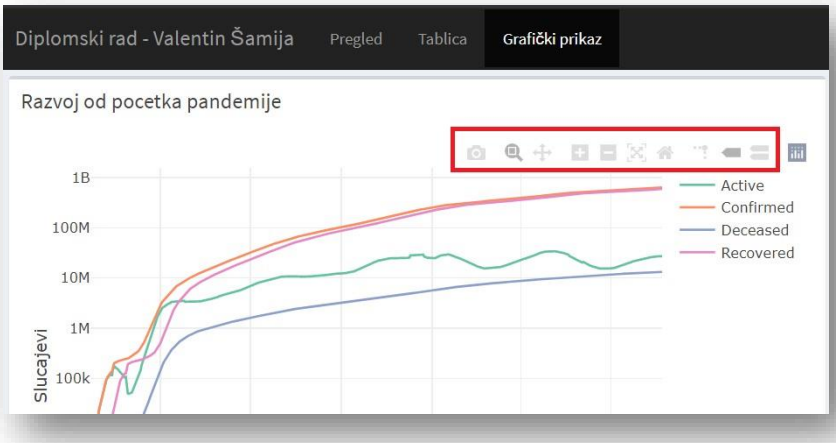
Slika 69: Tablični prikaz b)

Na samom kraju preostaje **Grafički prikaz**. Klikom na navedeni odjeljak otvara se sljedeća kartica:

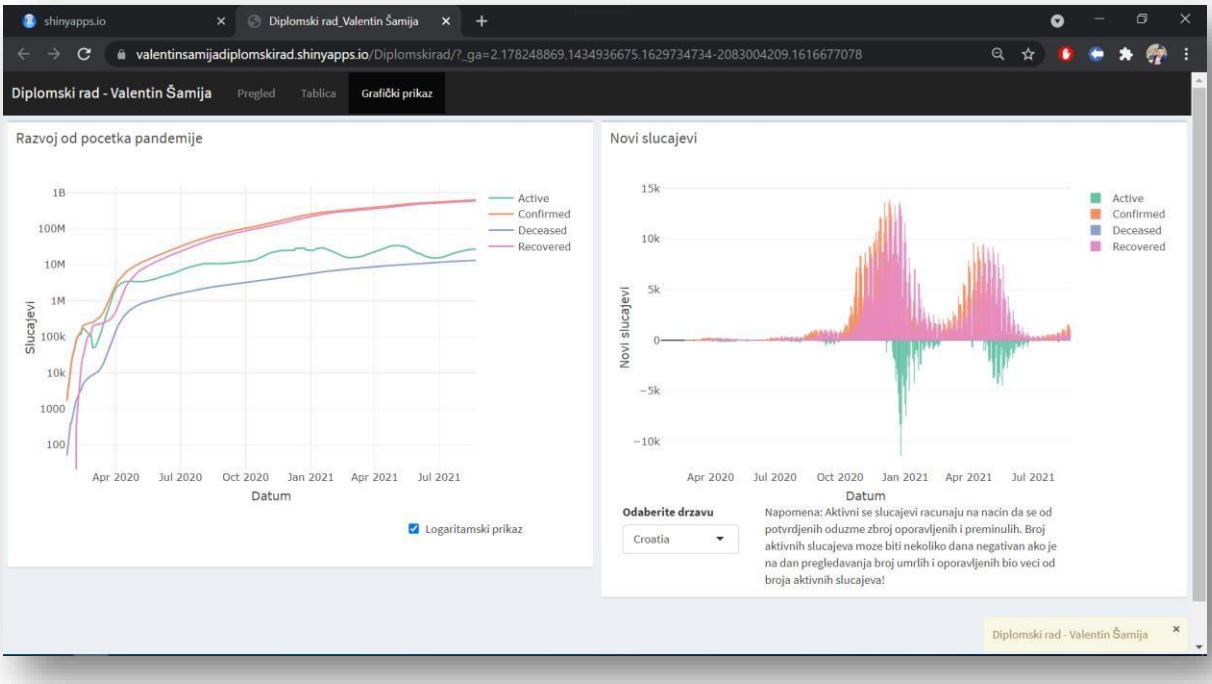


Slika 70: Grafički prikaz a)

Lijeva strana prikazuje razvoj slučajeva od početka pandemije, a desna strana prikazuje nove slučajeve. Razvoj se može prikazati eksponencijalno ili logaritamski. Novi slučajevi prikazuju najrelevantnije podatke za sve države ili samo za državu koju korisnik izabere klikom na „Odaberite drzavu“. Prelaskom miša iznad grafova nude se dodatne mogućnosti kao što su preuzimanje grafa u .png formatu, zumiranje i slično.



Slika 71: Dodatne mogućnosti



Slika 72: Grafički prikaz b)

5. ZAKLJUČAK

Projektni zadatak koji je ujedno i diplomski rad naslanja se na najaktualniju temu današnjice – koronavirus. Upravo zbog toga nastala je i ova aplikacija koja prikuplja, pregledava i analizira sve podatke vezane uz pandemiju. Veličina podataka je iz dana u dan sve veća i veća. Osim ogromnog broja podataka pojavljuju se i novi indikatori koji dodatno kompliciraju sve.

Aplikacija korisniku prikazuje najnovije podatke nakon svakodnevnog ažuriranja. Mobilni dio je u jednu ruku precizniji jer ima manji broj varijabli koje treba obraditi za razliku od web dijela koji obrađuje masu podataka. Korisniku su sve mogućnosti i funkcionalnosti potpuno dostupne i jednostavne, što je zapravo i želja svakoga koji koristi aplikaciju. Svaka dodatna komplikacija će korisnika udaljiti od aplikacije. Prilikom izrade aplikacije odabrani su programi koji daju najbrže i najjednostavnije smjernice do željenog rezutata.

Ovaj diplomski rad, tema, zadatak i alati korišteni pri izradi povezani su sa kolegijima koji su se našli „na putu“ prilikom studiranja. Cjelokupno vrijeme izrade diplomskog rada zahtjevalo je dodatno istraživanje koje je ponekad prelazilo neke granice i kapacitete koje je trebalo dostići i srušiti, ali to je donijelo neka nova znanja i iskustva koja će se zasigurno pokazati u najboljem svijetlu u skoroj budućnosti.

6. LITERATURA

Knjige:

[1]. A. Gerber, C. Clifton (2015.), Learn Android Studio, Build Android Apps Quickly and Effectively, 08.05.2021.

[2]. J. Paul Cardle (2017.), Android App Development in Android Studio, 08.05.2021.

[3]. H.Wickham, G. Grolemund (2016.), R for Data Science, 26.06.2021.

[4]. H. Wickham, (2015.), R Packages, 26.07.2021.

Internet:

[1].https://developer.android.com/studio?gclid=Cj0KCQjwjo2JBhCRARIsAFG667Vqji9noQpjgua38ragNJxl_8w3NdMglJ7Rj8hEu4aZWzJA7AhnA3IaAnibEALw_wcB&gclsrc=aw.ds , 14.05.2021.

[2]. https://developer.android.com/studio/intro , 14.05.2021.

[3]. https://www.redhat.com/en/topics/api/what-is-a-rest-api , 16.05.2021.

[4]. https://developer.android.com/training/volley, 16.05.2021.

[5]. https://stackoverflow.com/questions/tagged/android-studio , 10.05.2021.

[6].https://stackoverflow.com/questions/5486789/how-do-i-make-a-splash-screen , 23.05.2021.

[7]. https://stackoverflow.com/questions/2169294/how-to-add-manifest-permission-to-an-application, 24.05.2021.

[8]. https://developer.android.com/studio/build/dependencies?hl=sk, 24.05.2021.

[9]. https://developer.android.com/reference/android/widget/Filter , 30.05.2021.

[10]. https://corona.lmao.ninja/ , 06.06.2021.

[11]. https://corona.lmao.ninja/v2/all, 06.06.2021.

[12]. https://corona.lmao.ninja/v2/countries/ ,06.06.2021.

[13]. https://github.com/generic-leo/SimpleArcLoader ,06.06.2021.

[14]. https://github.com/blackfizz/EazeGraph, 06.06.2021.

[15]. https://www.koronavirus.hr/ , 06.06.2021.

[16].https://www.koronavirus.hr/podaci/otvoreni-strojno-citljivi-podaci/526, 06.06.2021.

[17]. https://www.jhu.edu/ , 06.06.2021.

[18]. https://www.rstudio.com/, 25.06.2021.

[19]. https://shiny.rstudio.com/ , 25.06.2021.

[20]. https://rstudio.github.io/shinydashboard/ , 25.06.2021.

[21]. https://rstudio.github.io/DT/ , 10.07.2021.

[22]. https://cran.r-project.org/web/packages/fs/index.html , 11.07.2021.

[23]. https://cran.r-project.org/web/packages/wbstats/index.html, 11.07.2021.

[24]. https://plotly.com/ , 11.07.2021.

[25]. https://www.tidyverse.org/, 16.07.2021.

[26]. https://github.com/CSSEGISandData/, 30.07.2021.

[27]. https://github.com/CSSEGISandData/COVID-19 , 30.07.2021.

[28]. https://coronavirus.jhu.edu/map.html , 30.07.2021.

[29]. https://www.shinyapps.io/, 18.08.2021.

[30]. https://sdk.finance/how-to-spot-a-good-api/ , 21.08.2021.

[31]. https://www.altexsoft.com/blog/rest-api-design/, 21.08.2021.

[32]. https://statsandr.com/blog/how-to-publish-shiny-app-example-with-shinyapps-io/, 22.08.2021.

[33]. https://github.com/vsamija , 24.08.2021

7.POPIS SLIKA

8.SAŽETAK

Ovaj diplomski rad se sastoji od teorijskog dijela i praktičnog dijela. U teorijskom dijelu ukratko je objašnjen problem koji je povezan uz pandemiju koronavirusa. Nakon toga objašnjeni su programi i alati koji su korišteni pri izradi projektnog zadatka. Struktura projekta unutar aplikacije opisuje glavne dijelove koji su korišteni pri izradi. Slijedi razrada funkcionalnosti te načini primjene. Da bi se korisnik znao služiti aplikacijom dodane su i korisničke upute koje su vrlo jednostavne.

KLJUČNE RIJEČI: *Diplomski rad*, *Diplomski zadatak, Projekt, Android Studio, Android Emulator, App, Project, Structure, Gradle, Manifest, Layout, Java, Res, Activity, Dependencies, MainActivity, ZarazeneDrzave, DrzavePrikaz, DrzaveLista, DetaljiPrikaz, Splash, RStudio, R, Shiny, Packages, Plotly, DataTables, API, TidyVerse, Johns Hopkins University, UI, Server, Mobile, Web*

ABSTRACT

This thesis consists of a theoretical part and a practical part. The theoretical part briefly explains the problem associated with the coronavirus pandemic. After that, the programs and tools used in creating the project task were explained. The structure of the project within the application describes the main parts used in the development. The following is an elaboration of the functionality and methods of application. In order for the user to know how to use the application, user instructions have been added, which are very simple.

KEYWORDS: *Graduation thesis, Android Studio, Android Emulator, App, Project, Structure, Gradle, Manifest, Layout, Java, Res, Activity, Dependencies, MainActivity, Infected States, CountriesView, CountriesList, DetailsView, Splash, RStudio, R, Shiny, Packages, Plotly, DataTables, API, TidyVerse, Johns Hopkins University, UI, Server, Mobile, Web*