Razvoj aplikacije za vizualizaciju podataka

Juričić, Josipa

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

RAZVOJ APLIKACIJE ZA VIZUALIZACIJU PODATAKA

Josipa Juričić

Split, rujan 2020.

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za Informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

RAZVOJ APLIKACIJE ZA VIZUALIZACIJU PODATAKA

Josipa Juričić

SAŽETAK

Tema ovog završnog rada je vizualizacija podataka i razvoj aplikacije koja ju implementira. U radu je pojašnjeno što su to podaci, opisuju se veze među podacima i prikazuju se različiti načini vizualizacije podataka u obliku grafikona, dijagrama i karti. Kao dio završnog rada, izrađena je i mrežna aplikacija koja prikazuje statističke podatke o širenju Korona virusa. Programski alati koji su korišteni za razvoj aplikacije su Django, Dash i Plotly.

Ključne riječi: podaci, obrada podataka, vizualizacija, grafikon, plotly
Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 43 stranica, 15 grafičkih prikaza, 0 tablica i 7 literaturnih navoda. Izvornik je na hrvatskom jeziku.
Mentor: **Dr. sc. Divna Krpan,** *viši predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
Neposredni voditelj:
Dr. sc. Saša Mladenović, *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
Ocjenjivači: **Dr. sc. Divna Krpan,** *viši predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
Dr. sc. Saša Mladenović, *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
Dr. sc. Goran Zaharija, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
Rad prihvaćen: Rujan 2020.

Basic documentation card

Thesis

University of Split
Faculty of Science
Department of Computer Science
Ruđera Boškovića 33, 21000 Split, Croatia

DEVELOPING DATA VISUALIZATION APPLICATION

Josipa Juričić

ABSTRACT

The main topic of this paper is data visualization and the development of an application that implements it. This paper explains what data is, describes the relationships between data and shows different ways of data visualization in the forms of graphs, charts and maps. The web application, which shows statistical data of the spreading of the Coronavirus, is developed as a part of this paper. Programming tools, used to develop this application, are Django, Dash and Plotly.

Key words: data, data processing, visualization, chart, plotly
Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 43 pages, 15 figures, 0 tables and 7 references
Original language: Croatian
Mentor: **Divna Krpan, Ph.D.** *Professor of Faculty of Science, University of Split*
Supervisor: **Saša Mladenović, Ph.D.** *Associate Professor of Faculty of Science, University of Split*
Reviewers: **Divna Krpan, Ph.D.** *Professor of Faculty of Science, University of Split*
Saša Mladenović, Ph.D. *Associate Professor of Faculty of Science, University of Split*
Goran Zaharija, Ph.D. *Professor of Faculty of Science, University of Split*
Thesis accepted: September 2020.

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam završni rad s naslovom RAZVOJ APLIKACIJE ZA VIZUALIZACIJU PODATAKA izradila samostalno pod voditeljstvom dr.sc. Divne Krpan. U radu sam primijenila metodologiju znanstvenoistraživačkog rada i koristila literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući navela u završnom radu na uobičajen, standardan način citirala sam i povezala s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Studentica

Josipa Juričić

Sadržaj

Uvod

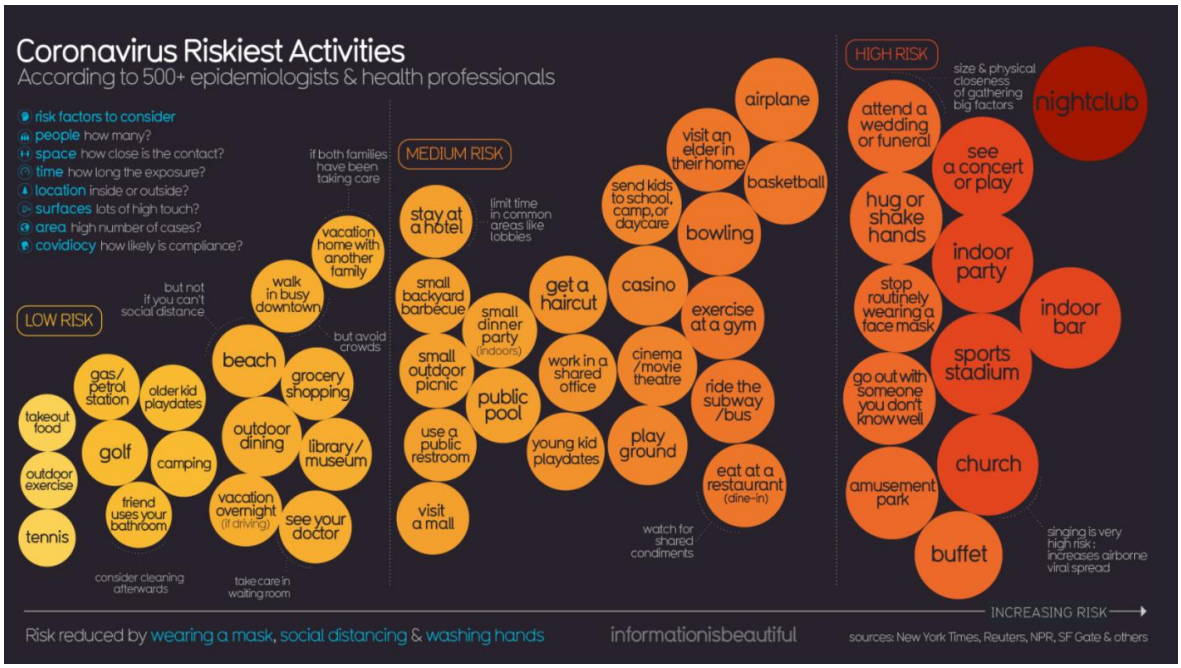
Jedan od najvećih izazova današnjice je učinkovito razumijevanje i korištenje velike količine podataka koja je sveprisutna u mnogim poslovnim okruženjima. Vizualna analiza podataka jedan je od najvažnijih alata za razumijevanje tako velikih, složenih skupova podataka. Vizualizacija olakšava proces donošenja odluka nad ogromnom količinom podataka koristeći ljudsku sposobnost opažanja i razumijevanja stvari i pojava. Razvojem tehnologije i računala omogućeno je obrađivanje velikih količina podataka velikom brzinom. U današnje vrijeme, vizualizacija podataka je postala kombinacija znanosti i umjetnosti koja je uvelike važna za poslovna tržišta. Donošenje odluka se može olakšati, ubrzati i temeljiti na pouzdanim parametrima kroz dostupne grafičke vizualizacije. Kako se svakim danom povećava količina podataka na raspolaganju tako se razvijaju i razni alati i okruženja

U ovom će se radu predstaviti načini vizualizacija podataka, pojasniti što su to podaci, u kojim ih oblicima dobivamo te na koje ih načine obrađujemo. Predstavljeni su i neki od dostupnih alata i okruženja za vizualizaciju podataka te je opisan i način izrade mrežne aplikacije za vizualizaciju podataka o širenju Korona virusa pomoću odabrane biblioteke Plotly.

1. Podaci i vizualizacija podataka

U mnogim poslovima 21. stoljeća količina podataka se svakim danom povećava nevjerojatnom brzinom. Procjenjuje se da svakog dana u svijetu količina novih podataka prelazi 2.2 milijuna terabajta. Izazov je toliki skup podataka analizirati i na osnovu tih analiza razumjeti složene odnose među podacima te dobiti nove i korisne informacije. Vizualizacija podataka se, kao proces prikazivanja podataka u obliku obojenih grafova, dijagrama ili karti, pokazala kao način na koji čovjek jednostavno i vizualno može pronaći uzorke među podacima i odrediti odnose među njima. Često gledanje u velike količine podataka u tablicama nema smisla, ali samo jednim pogledom na grafikon možemo razumjeti najvažnije odnose među tim istim podacima.

Uzmimo za primjer prikaz podataka na slici 1.1. Imamo situaciju kao što je pandemija Korona virusa i želimo javnost upozoriti na situacije u kojima se povećava rizik od zaraze virusom. Nema smisla ovakve podatke ljudima prikazati u monotonim oblicima tablice ili popisa koje većina ljudi neće ni pogledati jer im ništa ne privlači pažnju. No, ako ovakav popis situacija podijelimo u kategorije vidljivo odvojene bojama te jarkim, oku upadljivim, bojama označimo najrizičnije situacije, jednostavno je uočiti najvažnije situacije koje treba izbjegavati.



Slika 1.1 Grafički prikaz rizičnih aktivnosti tijekom pandemije Korona virusa

1.1. Što su podaci

Kroz cijelu svoju povijest čovjek opaža činjenice o svijetu, procesima i događanjima oko sebe te ih nastoji shvatiti. Takve činjenice, koje same po sebi nemaju nikakvo značenje, zovemo podacima. Riječ podatak dolazi od latinske riječi *datum* što znači dio informacije. Kada skup logički povezanih podataka ima neko značenje, nazivamo ga informacijom. Informacija, od latinske riječi *Informare* - obavještavanje, je skup podataka kojima je dano značenje u procesu koji nazivamo obrada podataka.

Primjeri podataka su: „oblak“, „plavo“, „jedan“, „nebo“. Stavljanjem podataka u kontekst dobijemo informaciju: „Na plavom nebu je jedan oblak.“.

1.2. Obrada podataka

Obrada podataka su procesi manipuliranja, klasifikacije ili računanja nad podacima kojima se dobiju informacije koje su razumljive i korisne. Kroz povijest postojali su mnoge metode obrade podataka koje su se usavršavale potaknute potrebama modernih društava.

Ciklus obrade podataka sastoji se od ulaza, obrade i izlaza. Ulaz čine podaci u obliku prikladnom s obzirom na način obrade. U procesu obrade nad podacima se vrše brojčani izračuni ili klasifikacija, tj. vrši se manipulacija nad podacima da bi se dobio neki upotrebljivi oblik.

Četiri osnovne kategorije obrade podataka su:

- ručna obrada,
- mehanička obrada,
- elektromehanička obrada,
- elektronička obrada.

Ručna obrada podataka najstariji je oblik obrade podataka koji je postojao još u samim početcima ljudske civilizacije. Ljudi su počeli primjećivati pojave kao što su smjena dana i noći, mjesečev ciklus i godišnja doba te su njihovim zapisivanjem od podataka dobili korisne informacije (sati, dani, mjeseci) koje su im pomagale u svakodnevnom životu. Jedna od najranijih metoda brojanja je metoda brojanja na prste. No, kako se civilizacija razvijala povećavala se i količina podataka koje je potrebno obraditi, pa su se razvile metode za brojanje kao što su brojanje kamenčićima, urezivanje zareza na kostima i

štapovima te urezivanje na stijene. U ručnu obradu podataka spada i prvo računalo *abakus* koje se sastojalo od kuglica čijim se pomicanjem omogućilo brzo i jednostavno računanje s malim količinama podataka.

U 17. stoljeću izgrađuju se mehanički strojevi koji omogućuju izvođenje jednostavnih aritmetičkih operacija. Blaise Pascal, francuski filozof i matematičar, je 1642. godine izumio *Pascalinu*, mehanički stroj za zbrajanje i oduzimanje sa kotačićima i zupcima koji su omogućavali automatsko zbrajanje i oduzimanje. Tridesetak godina kasnije Wilhelm von Leibniz usavršava ovaj stroj u *Leibnizov kalkulator* koji uz zbrajanje i oduzimanje može množiti i dijeliti. Sljedeći veliki pomak u povijesti obrade podataka napravio je Charles Babbage svojim *diferencijalnim strojem* kojeg zbog nedostatka financiranja nikada nije dovršio, a trebao je rješavati diferencijalne jednadžbe. Zbog istog razloga nije dovršio ni svoj *analitički stroj* koji se kao prvi programibilni kalkulator smatra pretečom današnjih modernih računala.

Doba elektromehaničke obrade podataka započelo je *strojem za sortiranje* Hermana Holleritha koji je za pogon imao električnu energiju, a za upravljanje su se koristile bušene kartice. Osnovna namjena njegovog stroja bila je svrstavanje bušenih kartica s podacima o popisu stanovništva što se pokazalo dosta učinkovitim, a od Hollerithove tvrtke kasnije je nastao današnji IBM. Sljedeći veliki izum bili su strojevi građeni od elektromagnetskih releja koji su prvi omogućili pohranjivanje podataka. Konrad Zuse i Howard Aiken su načelima logičkih operacija, pohrane u binarnom obliku te pomične decimalne točke u svojim strojevima napravili veliki korak prema računalima kakva poznajemo danas.

ENIAC (eng. *Electronic Numerical Integrator and Calculator*) se smatra prvim potpuno elektroničkim digitalnim računalom kojeg su John Mauchly i John Eckert dovršili 1946. godine čime započinje doba prve generacije elektroničkih računala temeljenih na elektroničkim cijevima. Godine 1951. izradili su UNIVAC (engl. *Universal Automatic Computer*), prvo računalo koje je moglo obrađivati i znakove, a ne samo brojke. Drugu generaciju elektroničkih računala obilježila su računala koja su umjesto elektroničkih cijevi imala tranzistore koji su omogućili pouzdaniji rad računala. Prva računala s integriranim sklopovima pojavila su se 1965. godine i nazvana su elektronička računala treće generacije. Samo desetljeće kasnije već su u upotrebi integrirani sklopovi sa 100 do nekoliko tisuća tranzistora. Računala s takvim sklopovima nazivaju se elektroničkim računalima četvrte generacije, a za njihovo se programiranje koriste viši programski jezici. Daljnji razvoj poluvodičke tehnologije nastavlja se u smjeru još većeg opsega integracije

sa stotinama tisuća tranzistora i računalima pete generacije. Sve veći napredci u razvoju računala dovode do razvoja i na području interaktivne grafike. Moderna računala, još od 1980-ih, koriste grafičko korisničko sučelje (eng. *Graphical User Interface*, skraćeno GUI) za vizualizaciju podataka. Razvojem vektorske i rasterske grafike u 21. stoljeću, vizualizacija podataka se primjenjuje u gotovo svim računalnim sustavima.

1.3. Tipovi podataka

Podatke se može podijeliti na kvantitativne (eng. *Quantitative*) i kvalitativne (eng. *Qualitative*). Glavna razlika među njima jest da su kvalitativni podaci deskriptivni, a kvantitativni podaci numerički.

Kvantitativni podatak je numerički podatak kojeg je lako izračunati i analizirati te prikazati tablicama, grafikonima i dijagramima. Kvantitativne podatke moguće je daljnje podijeliti na diskretne (podaci koji se sastoje od cijelih brojeva i mogu se prebrojiti – na primjer broj studenata) te kontinuirane (podaci koji mogu poprimiti vrijednost iz nekog intervala i mjerljivi su – na primjer visina i težina).

Kvalitativni podaci su podaci koji se mogu podijeliti u kategorije temeljene na nenumeričkim karakteristikama. Ovakvi podaci mogu, a i ne moraju imati logički poredak, a opisna priroda im ponekad otežava analizu i prikazivanje. Ovaj tip podataka dijelimo na redne (podaci koji slijede neki redoslijed – na primjer mjeseci u godini, abeceda) i kategorijalne (podaci koji ne slijede neki redoslijed – na primjer spol, nacionalnost).

1.4. Odnosi među podacima

Odnosi među podacima mogu biti jednostavni, poput napretka jednog podatka kroz vrijeme (na primjer, broj korisnika aplikacije kroz period od 3 mjeseca), ili složeni, poput uspoređivanja struktura među podacima i izvlačenja uzoraka. Izbor grafikona za prikaz podataka je uvelike temeljen na odnosu među podacima koje određeni grafikoni mogu predočiti bolje od drugih (Crooks et al., 2012). U knjizi *Show Me the Numbers: Designing Tables and Graphs to Enlighten* Stephen Few iznosi sedam odnosa među podacima koji se najčešće prikazuju grafovima:

- Nominalne usporedbe (eng. *Nominal comparisons*) – jednostavna usporedba kvantitativne vrijednosti potkategorija,

- Devijacija (eng. *Deviation*) – prikaz odstupanja pojedinog podatka od središnje vrijednosti,
- Korelacija (eng. *Correlation*) – prikaz podataka s dvije ili više varijable koje pokazuju pozitivni ili negativni odnos jedne s drugom ,najčešće na raspršenom grafikonu,
- Distribucija (eng. *Distribution*) – prikaz raspodjele podataka, često oko središnje vrijednosti,
- Djelomične i ukupne relacije (eng. *Partial and total relationships*) – uspoređuje se podskup podataka u odnosu na veći skup, za prikaz ovakvih odnosa među podacima najčešće se koriste kružni grafikoni,
- Rangiranje (eng. *Ranking*) – uspoređivanje jedne ili više vrijednosti na osnovi relativnih veličina za čiji se prikaz najčešće koriste stupčasti dijagrami,
- Podaci kroz vrijeme (eng, *Time series*) – uspoređivanje jedne varijable kroz određeno razdoblje (Crooks et al., 2012).

1.5. Skupovi podataka – prikupljanje i formatiranje

Prije vizualizacije podataka važno je pojasniti na se koje načine dolazi do skupova podataka koji se vizualiziraju te u kojim oblicima se spremaju i obrađuju. Podatke za vizualizaciju moguće je samostalno prikupiti, dobiti od stručnjaka područja čije podatke vizualiziramo ili pronaći *online*. Za bilo koji način prikupljanja važno je provjeriti podatke jer uvijek postoji mogućnost pogreške.

Najjednostavniji način pronalaska skupa podataka je upisati traženi pojam u tražilicu (poput Google-a ili Yahoo-a) te pretražiti mnogobrojne mrežne stranice koje nude skupove podataka za odabranu temu. Neki od najpopularnijih izvora podataka su *Freebase, Infochimps, Numbrary, AggData, Amazon Public Dana Sets* te *Wikipedia* (Yau, 2011). Ako postoji situacija u kojoj se traženi podaci nalaze na više mrežnih stranica, a želimo ih spremiti u jedan dokument, postoji i opcija *data scraping*-a. Način na koji *data scraping* radi je taj da se napiše neki programski kod za automatsko pretraživanje stranica koji dohvaća podatke i sprema ih u bazu podataka ili neki dokument (Yau, 2011).

Skupljene podatke moguće je spremiti u više formata, a različiti alati i okviri za vizualizaciju koriste različito formatirane skupove podataka. Najpoznatiji način spremanja velikih količina podataka je u CSV (eng. *Comma-separated values*, skraćeno CSV) formatu zarezom odvojenih vrijednosti. Podaci prikazani u stupcima i redovima se u CSV

formatu odvajaju po stupcima graničnikom koji može biti zarez, točka-zarez, razmak i sl. Ovaj način formatiranja podataka u širokoj je upotrebi, a podaci se jednostavno mogu učitati u Excel ili Google Dokument.

Neke od mrežnih aplikacija pružaju podatke preko API-a (eng. *Application Programming Interface*, skraćeno API) koji su najčešće u JSON (eng. *JavaScript Object Notation*, skraćeno JSON) obliku koji se temelji na ključ-vrijednost parovima gdje su podaci objekti kojima se pristupa preko ključa. Ovakav mrežni API se definira kao sučelje preko kojeg se odvija interakcija između poslužitelja i klijenta koji ima pristup različitim uslugama koje poslužitelj pruža različitim klijentima. Mana mrežnih API-a, a posebno onih javnih, su promjene na poslužiteljskoj strani koje mogu dovesti do pogrešaka na klijentskoj strani, kao što je dodavanje novih parametara u poziv funkcije.

Još jedan oblik formatiranja podataka je XML (eng. *Extensible Markup Language*) koji se često koristi za slanje podataka preko API-a, a zapravo je tekstni dokument s vrijednostima zatvorenima u tagove (Yau, 2011). Podaci se pohranjuje u običnu tekstualnu datoteku čiji je format intuitivno razumljiv čovjeku, a računalu jednostavan za interpretirati podatke. XML format se koristi kod razmjene podataka između čovjeka i računala, računala i računala te lokalno ili preko mreže.

Ukoliko odabrani način vizualizacije podataka zahtjeva određeni format skupa podataka, a skup je u nekom drugom formatu, postoje mnogi alati za formatiranje podataka. Primjerice, *Google Refine*, *Mr. Dana Converter*, *Mr. People* te *SpreadSheet Software* samo su neki od njih (Yau, 2011).

1.6. Pojam vizualizacije podataka

Vizualizacija podataka je grafički prikaz podataka koji ima veliku ulogu prilikom analize velike količine podataka te donošenju odluka na osnovi te analize. To je proces prevođenja podataka u vizualni kontekst, kao što je karta ili grafikon, kako bi se ljudskom mozgu olakšalo razumijevanje podataka. Glavni cilj vizualizacije podataka je olakšati prepoznavanje uzoraka, trendova i odstupanja u velikim skupovima podataka. Vizualizacija podataka omogućuje:

- brže donošenje odluka zbog poboljšanja brzine upijanja podataka,
- održavanje interesa publike podacima u njima razumljivom obliku,
- lakšu distribuciju podataka,

- povećanu sposobnost bržeg djelovanja na rezultate analize i s manje pogrešaka.

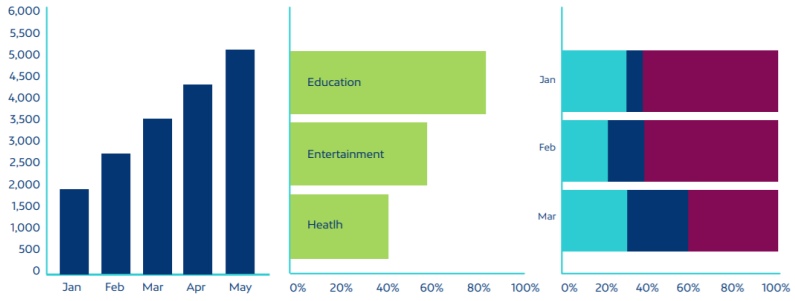
U sljedećim poglavljima navest će se osnovne i najčešće korištene vrste grafikona te pojasniti kako ih učinkovito koristiti.

1.6.1. Stupčasti grafikon (eng. *Bar chart*)

Jedan od najpopularnijih i jednostavnijih načina vizualizacije podataka su stupčasti grafovi ili grafikoni čiji prikaz podataka omogućuje brzo uočljive odnose među podacima. Svestrani su i najčešće korišteni za prikaz promjena kroz vrijeme, usporedbu različitih kategorija ili usporedbu dijelova cjeline. Postoje tri osnovna oblika: okomiti stupci, vodoravni stupci i naslagani stupci (prikaz kategorija u sumi je 100%). Okomiti se stupčasti grafikoni koriste za prikaz nominalnih odnosa među podacima, prikaz promjene podataka kroz vrijeme, rangiranje i prikaz odnosa dijelova s cjelinom (Crooks et al., 2012). Na ovakvim grafovima horizontalni stupci imaju kvalitativne podatke prikazane na X osi, a kvantitativne podatke na Y osi. Rotiranjem stupaca za 90 stupnjeva u okomiti položaj tipovi podataka na X i Y osi zamjenjuju položaje i dobije se vodoravni stupčasti grafikon. Za razliku od okomitog položaja stupaca, vodoravni položaj stupaca nije pogodan za prikaz promjene podataka kroz vrijeme. Naslagani stupčasti grafikon se najčešće koristi kada je potrebno prikazati više odnosa dijelova prema cjelini (Crooks et al., 2012).

Preporuke prilikom izrade:

- koristiti vodoravne oznake jer je okomite teže pročitati,
- odrediti pravilne razmake između stupaca (standardno razmak je ½ stupca),
- Y os započeti od 0,
- paziti na odabir boja (koristiti jednu boju za cijeli grafikon, a samo za naglašene podatke koristiti posebne boje),
- pravilno poredati podatke (po abecedi, slijedu ili vrijednosti).



Slika 1.2 Tri tipa stupčastog grafikona

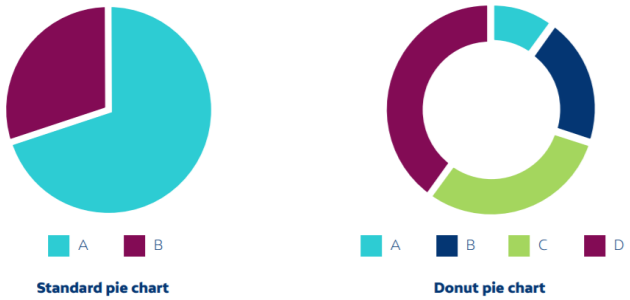
1.6.2. Strukturni krug ili kružni grafikon (eng. *Pie chart*)

Kružni grafikoni su uz stupčaste najpopularniji način jednostavne vizualizacije podataka. Najčešće ih se koristi za prikazivanje usporedbi dijelova s cjelinom diskretnih ili kontinuiranih podataka. Sastoje se od kruga podijeljenog u sektore od kojih svaki predstavlja dio cjeline. Kod slaganja dijelova kružnog grafikona, najveći kružni isječak uvijek treba smjestiti na vrh, a ostale po veličini u smjeru kazaljke na satu jer ljudi obično čitaju podatke od vrha prema dnu te će tako prvo uočiti najvažnije kategorije (Crooks et al., 2012).

Preporuke za izradu kružnog grafikona:

- ne koristiti više od pet kategorija,
- ne koristiti više kružnih grafikona za usporedbe skupova podataka (koristiti naslagane stupčaste grafikone jer ih je lakše uspoređivati nego kružne isječke),
- osigurati da svi podaci u sumi daju 100%.

Dva tipa kružnih grafikona su standardni (kružni ili tortni oblik) i prstenasti grafikon.



Slika 1.3 Dva tipa kružnog grafikona

1.6.3. Linijski grafikon (eng. *Line chart*)

Linijski se grafikoni koriste za prikaz promjena u podacima tijekom vremena, posebno prikazivanje ubrzavanja, usporavanja ili nestabilnosti u skupu podataka. Kod linijskih grafikona X os uvijek treba predstavljati vrijeme, a Y os kvantitativne podatke čiju promjenu kroz vrijeme pratimo (Crooks et al., 2012).

Kod izrade linijskih grafikona preporučljivo je držati se sljedećih smjernica:

- ako je moguće započeti osi grafa s 0,
- ne prikazivati više od četiri linije na jednom grafu¸,

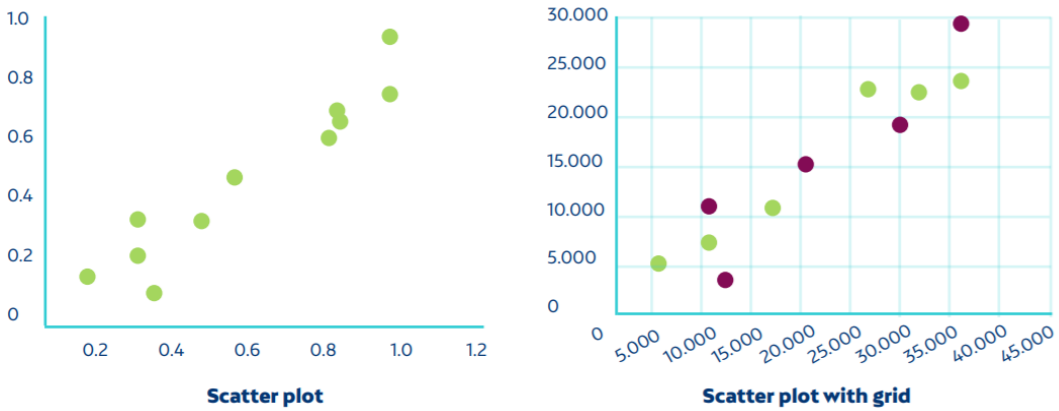
- koristiti pune linije umjesto točkastih koje mogu odvratiti pažnju prilikom analize,
- ne koristiti legendu za prikaz oznaka linija, već linije direktno označiti.

Slika 1.4 Linijski grafikon

1.6.4. Raspršeni grafikon (eng. *Scatter plot*)

Graf raspršenosti prikazuje odnos dviju varijabli, u obliku X i Y koordinata, koji predstavlja točka u Kartezijevom koordinatnom sustavu. Najčešće se koristi za prikaz odnosa dviju varijabli u velikom skupu podataka. Preporuke kod korištenja ovog načina prikaza podataka:

- Y os započeti od 0,
- koristiti veličinu i boju točke za prikaz dodatnih varijabli,
- koristiti linije smjera koje pomažu kod prikaza korelacije i smjera među varijablama,
- ne koristiti više od dvije linije smjera jer dolazi do težeg tumačenja rezultata.



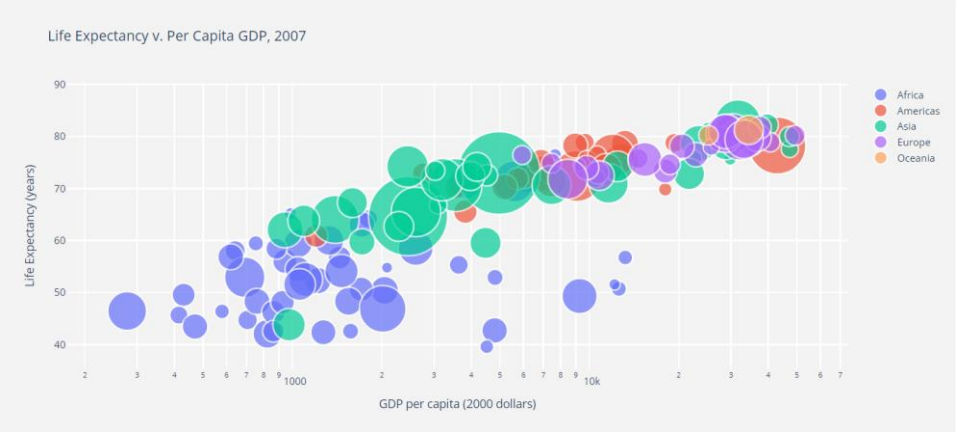
Slika 1.5 Primjeri grafikona raspršenosti

1.6.5. Grafikon s mjehurićima (eng. *Bubble chart*)

Ovi se grafovi koriste za prikazivanje trodimenzionalnih podataka i naglašavanje nominalnih usporedbi i klasifikacije odnosa. Uz usporedbu dvije glavne varijable na X i Y osi, veličina i boja mjehurića predstavlja treću dimenziju korisnu za naglašavanje dodatne vrijednosti. Dva su osnovna oblika ovakvih grafova: *bubble plot* (graf raspršenosti s mjehurićima koji predstavljaju dodatnu varijablu) i *bubble map* (prikazuju se mjehurići na određenoj geografskoj karti).

Preporuke kod izrade ovih grafova:

- osigurati vidljivost svih oznaka,
- pravilno skalirati veličine mjehurića prema području,
- izbjegavati umjesto mjehurića korištenje oblika koji nisu kružni.



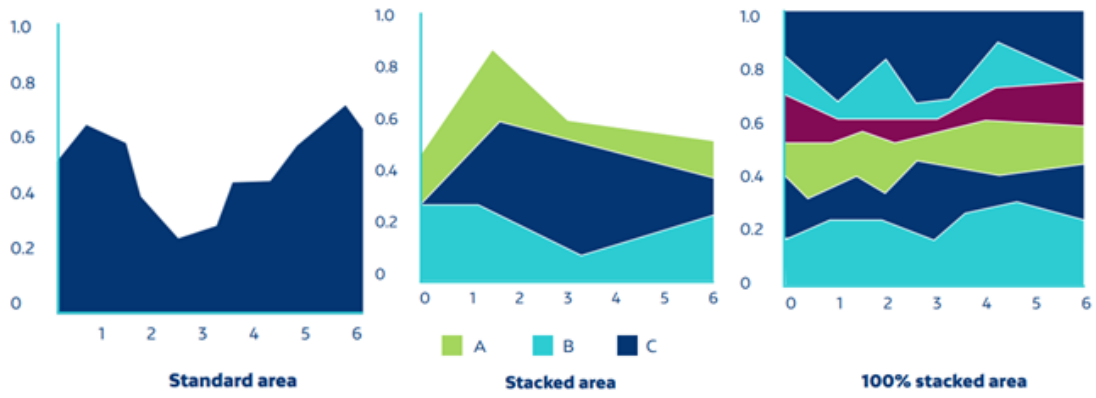
Slika 1.6 Bubble graf odnosa životnog vijeka i BDP po stanovniku

1.6.6. Površinski grafikon (eng. *Area chart*)

Slično kao i linijski grafikoni, površinski prikazuju promjene i odnose u podacima kroz vrijeme. No, za razliku od linijskih pružaju i volumen. Tri varijacije površinskih grafikona su standardno područje (prikaz podataka ili napretka kroz vrijeme), naslagano područje (prikaz odnosa među podacima kao dijelova cjeline demonstrirajući doprinos svake kategorije kumulativnoj sumi) i 100% područje (prikaz distribucije kategorija kao dijelova cjeline bez obzira na kumulativnu sumu). Za uspješnu izradu ovakvog prikaza podataka važno je:

- početi Y os s 0,

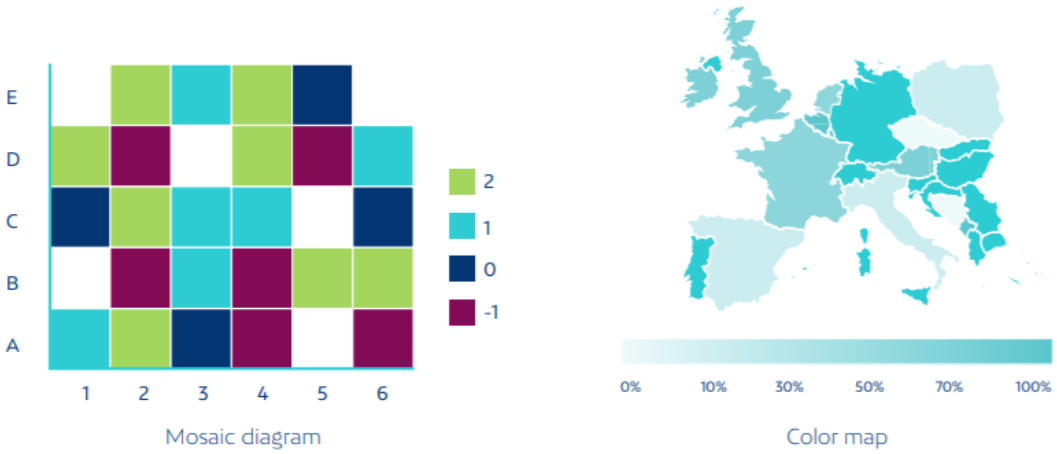
- ne prikazati više od četiri kategorije na jednom grafikonu,
- koristiti providne boje,
- ne koristiti površinske grafikone za prikaz diskretnih podataka.



Slika 1.7 Tri tipa površinskog grafa

1.6.7. Toplinske mape (eng. *Heath map*)

Toplinske karte prikazuju kategorijske podatke koristeći boje različitih jačina za prikaz vrijednosti podataka za određeno geografsko područje. Koriste se primjerice za vizualizaciju mrežne stranice gdje su područja s kojima su korisnici najčešće u interakciji prikazana toplim bojama, a područja s manje interakcije hladnim bojama. Dva tipa ovakvih grafikona su mozaički dijagram i obojena mapa.



Slika 1.8 Mozaički dijagram i obojena karta

2. Okruženja i alati za vizualizaciju

Osnova svake analize velikog skupa podataka je njegova vizualizacija radi boljeg razumijevanja. Kako se svakim danom povećava količina podataka na raspolaganju tako se razvijaju i razni alati i okruženja koji olakšavaju njihovu analizu i shvaćanje. Glavni cilj alata i okvira za vizualizaciju podataka je omogućiti korisniku značajniju interakciju sa podacima kojom se izgled grafike ažurira čime se svakom pojedinačnom korisniku pruža potencijalno drugačije iskustvo (Crooks et al., 2012). Kod odabira alata ili okvira za vizualizaciju podataka važno je da ispunjavaju sljedeće uvjete:

- jednostavna objava rezultata ciljanoj publici, bilo putem mrežnih stranica ili društvenih mreža,
- jednostavan za koristiti te pruža moderne i intuitivne kontrole,
- skalabilnost – okvir ili alat za vizualizaciju mora zadovoljiti rastuće zahtjeve poslovanja.

Većina programskih jezika i alata imaju na raspolaganju neku vrstu biblioteke za vizualizaciju podataka koja pojednostavljuje proces prikazivanja velike količine podataka pomoću grafike. Programski jezik Python nudi više biblioteka za grafički prikaz koje sadrže mnoge značajke, od biblioteka za jednostavne općenite grafove do biblioteka usko fokusiranih za određeno područje vizualizacije podataka. Kako su Python i JavaScript najpopularniji programski jezici u vizualizaciji podataka, neki od najpoznatijih i najtraženijih alata za vizualizaciju su *Tableau, FusionCharts, Highcharts, Datawrapper, D3.js,* i *Plotly.*

D3.js je vjerojatno najpopularnija JavaScript biblioteka u području vizualizacije podataka. Ovaj okvir za vizualizaciju pomaže u stvaranju dinamičnih i interaktivnih vizualizacija koristeći HTML5 (eng. *HyperText Markup Language*, skraćeno HTML), SVG (eng. *Scalable Vector Graphics*, skraćeno SVG) i CSS3 (eng. *Cascading Style Sheets*, skraćeno CSS) za manipulaciju DOM-om (eng. *Document Object Model* skraćeno DOM) u preglednicima. D3.js je u usporedbi s ostalim okvirima za vizualizaciju na nižoj razini što se tiče kompleksnosti izrade grafikona pa su mnoga rješenja izgrađen upravo na D3.js biblioteci. D3.js je iznimno brz i podržava vizualizaciju velikih skupova podataka kao i dinamično ponašanje kod interakcije i animacije.

Chart.js je popularna JavaScript biblioteka za vizualizaciju podataka u mrežnim aplikacijama stvaranjem responzivnih grafikona pomoću HTML5.

Leaflet je vodeća JavaScript biblioteka za stvaranje interaktivnih grafikona i mapa prilagodljivih mobilnim uređajima. Jednostavnog dizajna i visokih performansi prilagodljiv je svim glavnim desktop i mobilnom platformama. U području vizualizacije je cijenjen ponajviše zbog mnoštva dostupnih dodataka, dobro dokumentiranog API-a te dostupnog izvornog koda.

Pandas je alat optimiziran za vizualizaciju tabličnih podataka u obliku statičkih ploha i grafikona. Jednostavan je za upotrebu, ali sam po sebi pruža samo osnovnu funkcionalnost. Pandas se može koristiti u kombinaciji s drugim okvirima za stvaranje šireg raspona vizualizacija.

Preteča svih Python okvira za vizualizaciju, **Matplotlib** je moćan (nudi potpunu kontrolu nad svim aspektima grafikona), ali ga je kao rezultat toga teže naučiti. Sama biblioteka je ogromna, sa otprilike 70 000 linija koda. Matplotlib sadrži nekoliko različitih sučelja (načina za izradu grafa) te je sposoban za interakciju s nekolicinom *backend*-a (*backend* se bavi procesom izrade grafikona) (8).

Plotly tehnički nije okvir; to je mrežni alat za vizualizaciju koji nudi API-je za nekoliko različitih jezika, uključujući Python. Smatra se najmoćnijim alatom za izradu interaktivnih grafova. Zbog toga što je Plotly izabran za izradu praktičnog dijela ovog rada, detaljnije će se objasniti kako radi Plotly biblioteka u sljedećem poglavlju.

3. Implementacija aplikacije za vizualizaciju

Nakon definiranja svih važnih komponenti vizualizacije podataka za provesti ih u praksu odabran je projekt koji uključuje izradu mrežne aplikacije s vizualizacijom podataka o Korona virusu. Tri glavne tehnologije korištene za implementaciju ove ideje su Django, Dash i Plotly. Django i Dash su Python okviri za izradu mrežnih aplikacija, a Plotly je Python biblioteka za vizualizaciju podataka koja omogućava izradu interaktivnih vizualizacija u mrežnom okruženju.

3.1. Odabir skupa podataka

Važno je na umu imati nekoliko stvari pri odabiru skupa podataka. Prvo, je li nam važno da se pri vizualizaciji koriste najnoviji dostupni podaci ili da se koriste nepromjenjivi podaci? U ovom slučaju ne bi imalo smisla koristiti zastarjele, nepromjenjive podatke jer nam je cilj prikazivati najnovije podatke o broju oboljelih, izliječenih i umrlih od Korona virusa. Stoga, potrebno je pronaći API kojim ćemo dohvaćati redovno ažurirane podatke. Sljedeća važna stavka je provjeriti koji su podaci potrebni za vizualizaciju u odabranom okviru. Na primjer, ako želimo u Plotly-u na karti svijeta prikazati podatke oboljelih za pojedinu državu, u skupu podataka je potrebno za svaku državu imati ili zemljopisu dužinu i širinu ili standardni ISO 3166-1 kod za jedinstvenu međunarodnu identifikaciju država.

Skupove podataka odabrane za realizaciju ovog projekta može se pronaći na poveznicama:

- https://www.trackcorona.live/api/countries
- https://www.trackcorona.live/api/travel
- https://api.covid19api.com/summary
- https://opendata.ecdc.europa.eu/covid19/casedistribution/csv
- https://docs.google.com/spreadsheets/d/e/2PACX-1vTxATUFm0tR6Vqq-UAOuqQ-BoQDvYYEe-BmJ20s50yBKDHEifGofP2P1LJ4jWFIu0Pb_4kRhQeyhHmn/pub?gid=0&single=true&output=csv.

3.2. Odabir tehnologije korištene za razvoj

Tri osnovne tehnologije odabrane za izradu projekta su Django, Dash i Plotly. Navedene tehnologije su odabrane ponajprije zbog toga što koriste programski jezik Python. Razlozi odabira Pythona su jednostavna i intuitivna sintaksa, širok raspon biblioteka za rad s podacima i mrežnim aplikacijama te besplatno preuzimanje i instalacija. Python pruža mnoge okvire za izradu mrežnih aplikacija, a neki od njih su *Django*, *Bottle*, *Flask* i *TurboGear*. Razlog odabira Djanga je prethodno iskustvo koje je pokazalo da pruža izradu mrežne aplikacije uz manje koda, mnogo brže i visokih performansi. Django koristi MVT (eng. *Model-View-Template*, skraćeno MVT) princip što znači da je logika aplikacije podijeljena tako da se odvoji serverski dio aplikacije i korisnički. U ovoj situaciji jednostavniji je izbor bio Django za implementaciju glavnog dijela aplikacije, dok je Dash korišten za implementaciju korisničkog sučelja. Izboru Djanga je prethodio izbor Plotly biblioteke za vizualizaciju podataka. Plotly je odabran jer koristi Python, omogućuje implementaciju interaktivne grafike te uz Dash omogućuje elegantan dizajn više grafikona na jednom interaktivnom sučelju (eng. *dashboard*).

3.2.1. Plotly

Plotly, koji je izgrađen na temeljima JavaScript biblioteke *plotly.js*, omogućuje izradu interaktivnih grafova koji se mogu prikazivati u *Jupyter* bilježnici¸ spremiti kao samostalne HTML datoteke ili koristiti kao dio mrežnih aplikacija koristeći Dash. Plotly stvara, manipulira i renderira grafičke figure (eng. *figures*), grafikone, mape, dijagrame, koji su predstavljeni određenom strukturom podataka. Grafičke figure se mogu predstaviti kao rječnik (eng. dictionary) ili kao instanca plotly.graoh_objects.Figure klase. Plotly Express je API vizualizacije podataka koji stvara potpune grafikone pozivom samo jedne funkcije.

Slijede primjeri koda u Pythonu koristeći Plotly za stvaranje vrsta grafikona navedenih u prvom poglavlju ovog rada.

Primjer koda za izradu jednostavnog linijskog grafikona:

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length",
color="species")
fig.show()
```

Primjer koda za izradu jednostavnog grafikona raspršenosti:

```
import plotly.express as px
df = px.data.iris()
df["e"] = df["sepal_width"]/100
fig = px.scatter(df, x="sepal_width", y="sepal_length",
color="species", error_x="e", error_y="e")
fig.show()
```

Primjer koda za izradu raspršenog grafikona na karti:

```
import plotly.express as px
df = px.data.carshare()
fig = px.scatter_mapbox(df, lat="centroid_lat",
lon="centroid_lon", color="peak_hour", size="car_hours",
color_continuous_scale=px.colors.cyclical.IceFire,
size_max=15, zoom=10, mapbox_style="carto-positron")
fig.show()
```

Sljedeći primjer koda stvara grafikon raspršenosti na geografskoj karti uz odabranu projekciju, te dostupne koordinate za države u skupu podataka:

```
import plotly.express as px
df = px.data.gapminder()
fig = px.scatter_geo(df, locations="iso_alpha",
color="continent", hover_name="country", size="pop",
animation_frame="year", projection="natural earth")
fig.show()
```

3.2.2. Dash

Dash pruža Python klase za sve vizualne komponente aplikacije. Skupovi komponenti su sadržani u *dash_core_components* biblioteci (sadrži interaktivne komponente više razine izgrađene JavaScript-om, HTML-om i CSS-om kroz React.js biblioteku) i *dash_html_components* biblioteci (sadrži komponente za svaki HTML tag). Biblioteka *dash_core_components* sadrži *Graph* komponentu koja generira interaktivnu vizualizaciju podataka koristeći plotly.js.

Dash aplikacija se sastoji od dva dijela: „app.layout“ dio određuje izgled aplikacije, dok „app.callback“ dio sadrži Python *callback* funkcije koje se automatski pozivaju kada se promijeni vrijednost ulaznih komponenti.

3.2.3. Django

Iako su Dash i Plotly sami dovoljni za izradu mrežne aplikacije s komponentama vizualizacije podataka, postoji nekoliko razloga zašto se koriste u kombinaciji s Djangom:

- omogućiti da se više Dash aplikacija koristi na jednoj web stranici,
- omogućiti više instanci Dash aplikacije,
- sve objediniti u jedan poslužiteljski proces radi pojednostavljivanja skaliranja.

django_plotly_dash djeluje tako da se omota oko dash.Dash objekta. HTTP točke koje koristi aplikacija Dash preslikavaju se na one Django aplikacije, a aplikacija se ugrađuje u mrežnu stranicu pomoću template tag-a. Na jednoj stranici može se koristiti više Dash naplikacija. Biblioteka za rad u Djangu, Dashu i Plotlyu jednostavno se instalira s `pip install django_plotly_dash`.

Važna stvar kod izrade aplikacija koristeći Django je i razumjeti kako se spremaju podaci, odnosno kako je realiziran rad s bazama podataka u Djangu. Django umjesto složenog postupka izrade baze koristeći SQL programski jezik (eng. *Structured Query Language,* skraćeno SQL) koristi objektno relacijski *mapper* koji preslikava relacijsku bazu podataka u svijet objektno orijentiranog programiranja. Ovo omogućava da se umjesto definiranja tablica i relacija u SQL jeziku, pišu Django modeli u Pythonu. Modeli su klase koje predstavljaju tablice baze podataka i koje definiraju svojstva koja predstavljaju stupce u tablici baze podataka. Migracijama nazivamo funkcije koje omogućavaju stvaranje baze podataka i tablica na osnovi Django modela. Nakon stvaranja modela prvi korak mora biti pozivanje sljedeće naredbe u naredbenom retku: `python manage.py makemigrations`. Ovo stvara datoteku za migraciju koja upućuje Django o tome kako stvoriti tablice baze podataka za modele definirane u aplikaciji. Naredbom `python manage.py dbshell` moguće je provjeriti izgled stvorene baze. Da bi se primijenile promjene na bazi potrebno je pozvati naredbu: `python manage.py migrate`.

3.3. Zahtjevi aplikacije

Općenito, prije izrade aplikacije potrebno je definirati način implementacije *backend*-a (na primjer, baze podataka) te *frontend*-a (korisničko sučelje). Mrežna aplikacija je osmišljena na način da se *backend* dio realizira u Djangu. To znači da se Django koristi za realizaciju registracije i prijave korisnika u sustav, izradu baze za spremanje podataka korisnika i

definiranje komunikacije između servera i klijenta. Dio korisničkog sučelja se također realizira u Djangu, a to je uglavnom osnovni raspored i izgled stranica definiran Bootstrap-om, okvir za izgradnju responzivnih aplikacija. Za prikaz i izgled grafikona vizualizacije podataka koristi se Plotly, a programski kod je napisan u čistom Python-u.

Dio mrežne aplikacije je osmišljen kao forum za međusobnu interakciju korisnika te dijeljenje iskustava putovanja i života tijekom pandemije Korona virusa. Za realizaciju ovog dijela aplikacije potrebno je napraviti bazu podataka za spremanje podataka korisnika prilikom registracije te spremanje detalja objava i komentara koje korisnici objavljuju na forumu. Ovaj dio aplikacije prikazuje i najnovije smjernice i obavijesti o putovanju tijekom pandemije čiji se podaci trebaju dohvatiti preko API-a.

Dio aplikacije koji prikazuje vizualizaciju podataka o širenju korona virusa implementirat će se korištenjem Plotly-a i Dash-a. Sam izgled i raspored grafikona na stranici potrebno je definirati prema smjernicama za uspješnu vizualizaciju podataka opisanima u prvom poglavlju ovoga rada. Potrebno je omogućiti interakciju korisnika s grafikonima te ažuriranje grafikona prilikom interakcije.

3.4. Realizacija

U samom početku potrebno je osigurati okruženje koje će omogućiti korištenje do sada navedenih biblioteka i paketa za rad s podacima. U ovom projektu korištena je *Conda*, sustav za upravljanje paketima i sustav upravljanja okruženjima koji radi na Windowsima, MacOS-u i Linuxu. *Conda* brzo instalira, pokreće i ažurira pakete. Prvo je u Anaconda Prompt naredbenoj ljusci (eng. *Command Shell*) naredbom `conda create env dash_plotly` stvoreno novo okruženje u koje su instalirani potrebni paketi korišteni u aplikaciji i to naredbama: `pip install django`, `pip install django_plotly_dash`, `pip install plotly`. Potom je stvoren novi Django projekt naredbom `django-admin startproject django_dash_plotly` unutar kojeg su stvorene nove aplikacije naredbama: `python manage.py startapp corona` i `python manage.py startapp users`. Ovim je postupkom stvoren Django projekt naziva **django_dash_plotly** koji sadrži aplikacije **corona** i **users**, a njegovim pokretanjem u novostvorenom okruženju **dash_plotly,** moguće je unutar projekta koristiti instalirane pakete *django, django-plotly-dash* i *plotly* (koji sadrži i dash).

Aplikacija se pokreće u Anaconda Prompt-u naredbom `python manage.py runserver`, uz uvjet da je naredbom `conda activate dash_plotly` pokrenuto dash_plotly okruženje te je trenutna pozicija naredbenog retka u direktoriju django_dash_plotly aplikacije. Ovime se pokreće Django razvojni poslužitelj (eng. *development* server) koji omogućuje lokalno pokretanje aplikacije tako da se u web preglednik upiše *http://127.0.0.1:8000*.

3.4.1. Registracija i Prijava

Registracija i prijava korisnika je implementirana koristeći Djangov ugrađeni autentifikacijski sustav koji upravlja i autentifikacijom (provjera postojanja korisnika) i autorizacijom korisnika (provjera ovlasti korisnika). Objekt *User* je temelj autentifikacijskog sustava koji predstavlja korisnike i koristi se za ograničavanje pristupa, registracije korisnika i povezivanja sadržaja s autorima. U prethodno stvorenu aplikaciju *users* dodan je sav programski kod i datoteke potrebne za implementaciju registracije i prijave. Forma za unos korisničkih podataka prilikom registracije definirana je u *forms.py* datoteci. Definirana je klasa koja predstavlja formu za registraciju, a nasljeđuje sva svojstva i funkcionalnosti klase UserCreationForm koja je ugrađena Django klasa:

```
class CreateUserForm(UserCreationForm):
    username =
forms.CharField(widget=forms.TextInput(attrs={'class':'form-
control','placeholder':"Username"}))
    email =
forms.EmailField(widget=forms.EmailInput(attrs={'class':'form
-control','placeholder':"Email"}))
    password1 =
forms.CharField(widget=forms.PasswordInput(attrs={'class':'fo
rm-control','placeholder':"Password"}))
    password2 =
forms.CharField(widget=forms.PasswordInput(attrs={'class':'fo
rm-control','placeholder':"Confirm password"}))
```

Za pisanje HTML koda za stranice registracije i prijave stvoren je novi direktorij *templates* te njemu dodane HTML datoteke: login.html, register.html i base_account.html. U *views.py* datoteku dodane su funkcije koje šalju i dohvaćaju podatke sa formi i obavljaju ostale funkcionalnosti u pozadini korisničkog sučelja. Primjer izgleda funkcije u *views.py* datoteci koja definira funkcionalnosti iza korisničkog sučelja za registraciju:

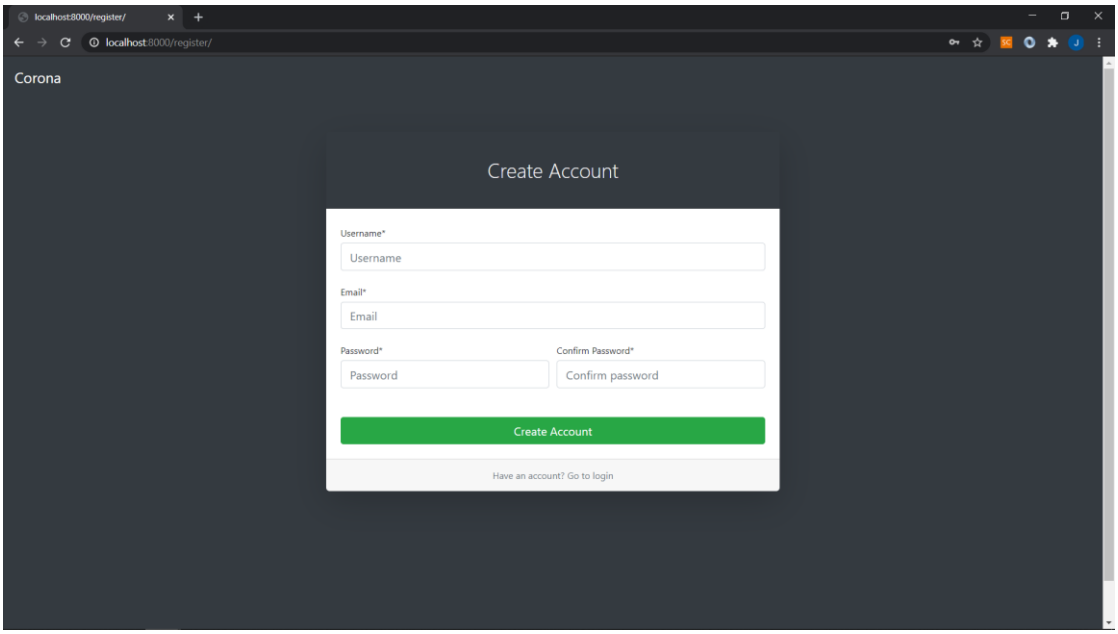
```
def registerPage(request):
      if request.user.is_authenticated:
            return redirect('home')
      else:
            form = CreateUserForm()
            if request.method == 'POST':
                  form = CreateUserForm(request.POST)
                  if form.is_valid():
                        form.save()
                        user =
form.cleaned_data.get('username')
                        messages.success(request, 'Account
was created for ' + user)

                        return redirect('login')

            context = {'form':form}
            return render(request, 'users/register.html',
context)
```

URL putanje do stranica definirane su u datoteci *urls.py*:

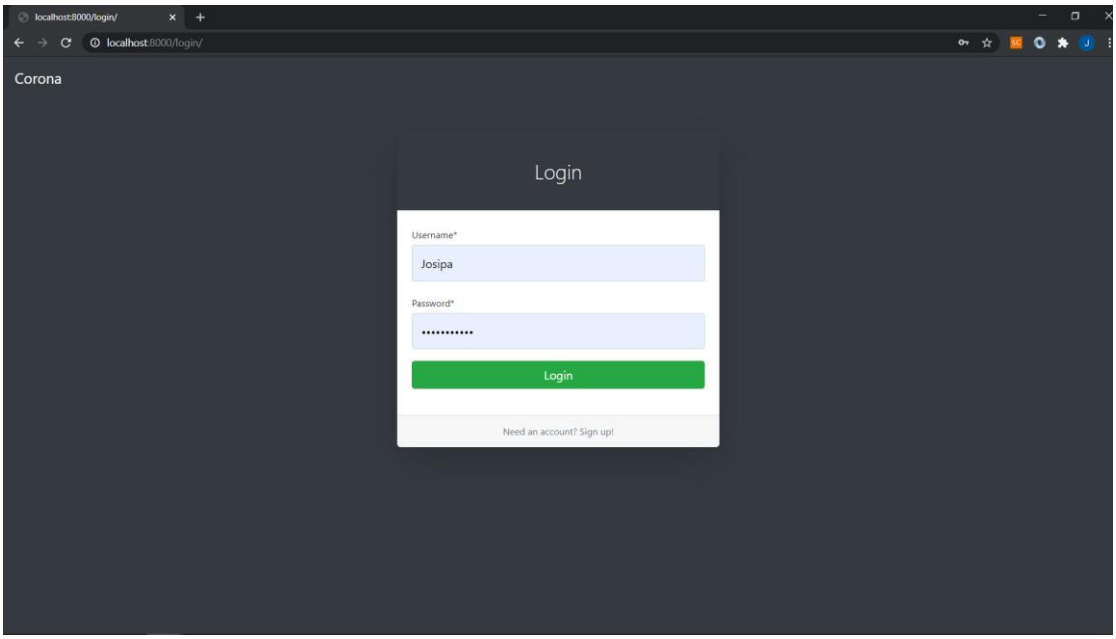
```
urlpatterns = [
      path('register/', views.registerPage, name="register"),
      path('login/', views.loginPage, name="login"),
      path('logout/', views.logoutUser, name="logout")
]
```

Općenito, u Djangu, URL putanje se definiraju navođenjem putanje (koja se prikazuje u polju za pretraživanje tražilice) i funkcije u *views.py*, koja definira logiku dohvaćanja i spremanja podataka na toj stranici. Parametar 'name' omogućuje da određenu putanju referenciramo preko imena prilikom definiranja poveznica na pripadnu stranicu. Primjerice, `<a href="{% url 'login' %}"> LogIn </a>`.

Na slikama 3.1 i 3.2 prikaza su korisnička sučelja za registraciju i prijavu.



Slika 3.1 Izgled stranice za registraciju



Slika 3.2 Izgled stranice za prijavu

3.4.2. Corona Forum

Za stvaranje foruma, uz do sada implementiranog sustava registracije i prijave, najprije su definirani modeli koji predstavljaju objave (model Post)i komentare (model Comment) koji se spremaju u bazu:

```
class Post(models.Model):
    countries = countries
    user = models.ForeignKey(User, on_delete=models.CASCADE)
```

```
    content = models.TextField()
    dateCreated = models.DateTimeField(auto_now_add=True)
    country = models.CharField(max_length=3,
choices=countries, default = 'HRV')
    class Meta:
        ordering = ['-dateCreated']

    def save(self, *args, **kwargs):
        super().save(*args, **kwargs)

    def __str__(self):
        return 'Posted by {}: {}'.format(self.user,
self.content)

    def get_absolute_url(self):
        return reverse('post-detail', kwargs={'pk': self.pk})

class Comment(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, related_name='comments',
on_delete=models.CASCADE)
    content = models.TextField()
    dateCreated = models.DateTimeField(auto_now_add=True)
    class Meta:
        ordering = ['-dateCreated']
    def save(self, *args, **kwargs):
        super().save(*args, **kwargs)
    def __str__(self):
        return 'Commented by {}: {}'.format(self.user,
self.content)
    def get_absolute_url(self):
        return reverse('post-detail', kwargs={'pk':
self.post.pk})
```

Nakon pozivanja naredbi za stvaranje novih migracija, koje će primijeniti promjene u bazi podataka, stvorene su klase u datoteci *views.py* koje barataju logikom komunikacije korisničkog sučelja i *backenda*. Primjerice, klasa PostListView definira podatke koji se u obliku konteksta šalju na HMTL stranicu *'forum.html'* :

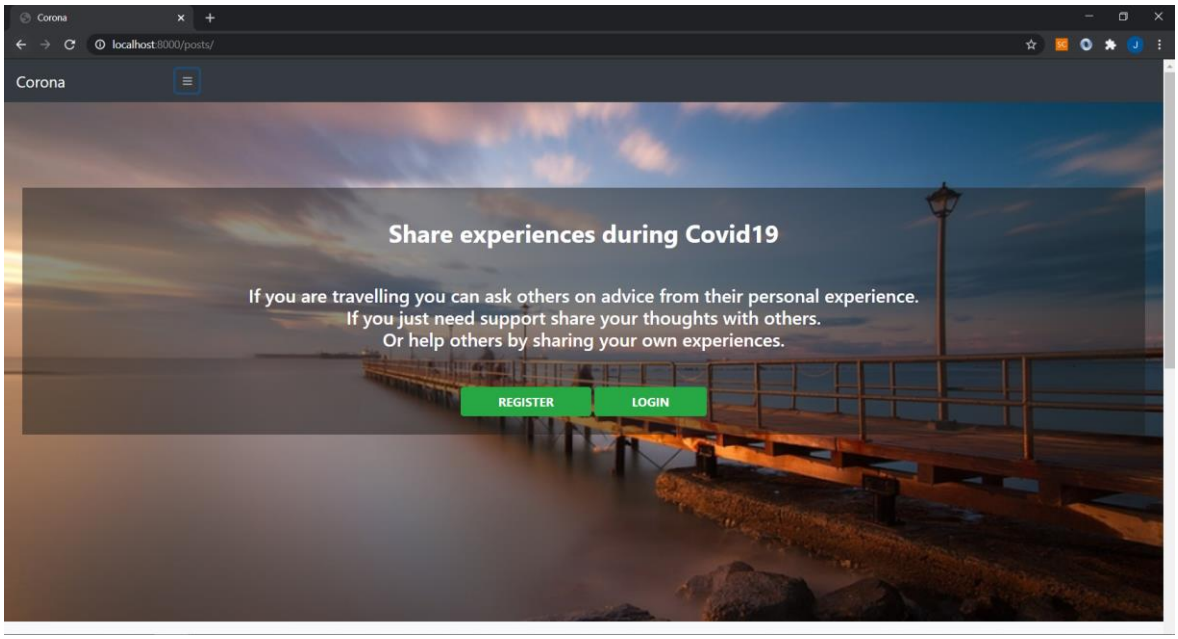
```
class PostListView(ListView):
    model = Post
    template_name = 'corona/forum.html'
```

```
    context_object_name = 'posts'
    ordering = ['-dateCreated']
    paginate_by = 7

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        return context
```

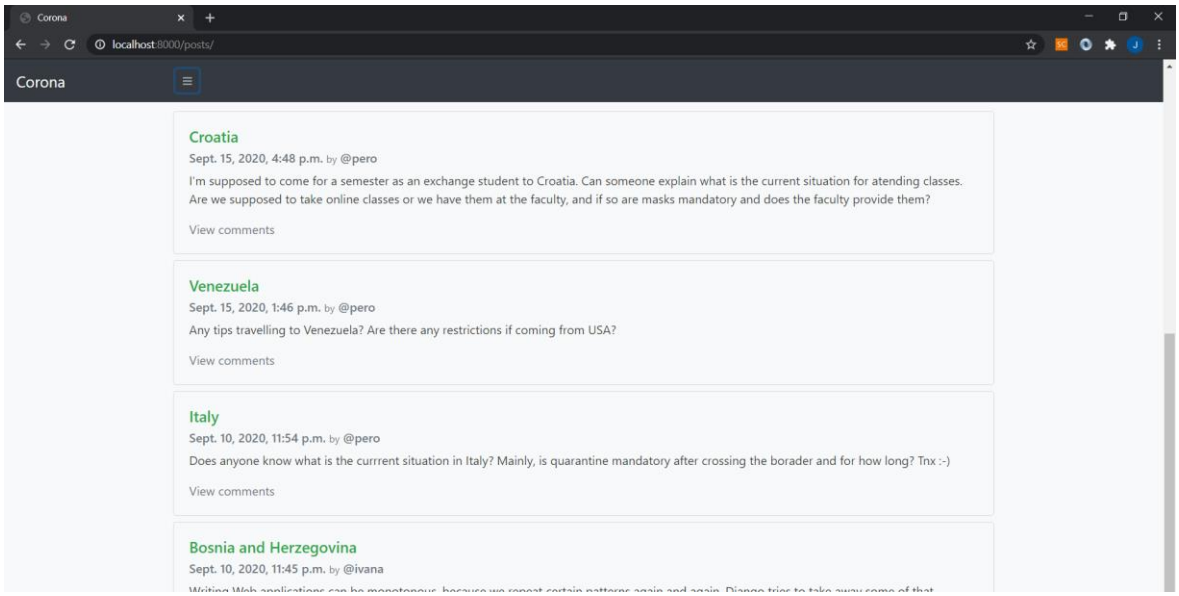
U datoteci urls.py definirane su URL putanje za pristupanje HTML stranicama:

```
urlpatterns = [
    path('', views.home, name='index'),
    path('', include('users.urls')),
    path('index/', views.index, name='index'),
    path('world', views.world, name='world'),
    path('travel', views.travel, name='travel'),
    path('countries', views.countries, name='countries'),
    # CREATE
    path('posts/new/', PostCreateView.as_view(), name='post-
create'),
    path('posts/<int:pk>/comment/new/',
CommentCreateView.as_view(), name='post-comment'),
    # READ
    path('posts/', PostListView.as_view(), name='posts'),
    path('posts/<int:pk>/', PostDetailView.as_view(),
name='post-detail'),
    # UPDATE
    path('posts/<int:pk>/update', PostUpdateView.as_view(),
name='post-update'),
    path('comments/<int:pk>/update',
CommentUpdateView.as_view(), name='comment-update'),
    # DELETE
    path('posts/<int:pk>/delete', PostDeleteView.as_view(),
name='post-delete'),
    path('comments/<int:pk>/delete/',
CommentDeleteView.as_view(), name='comment-delete'),
]
```

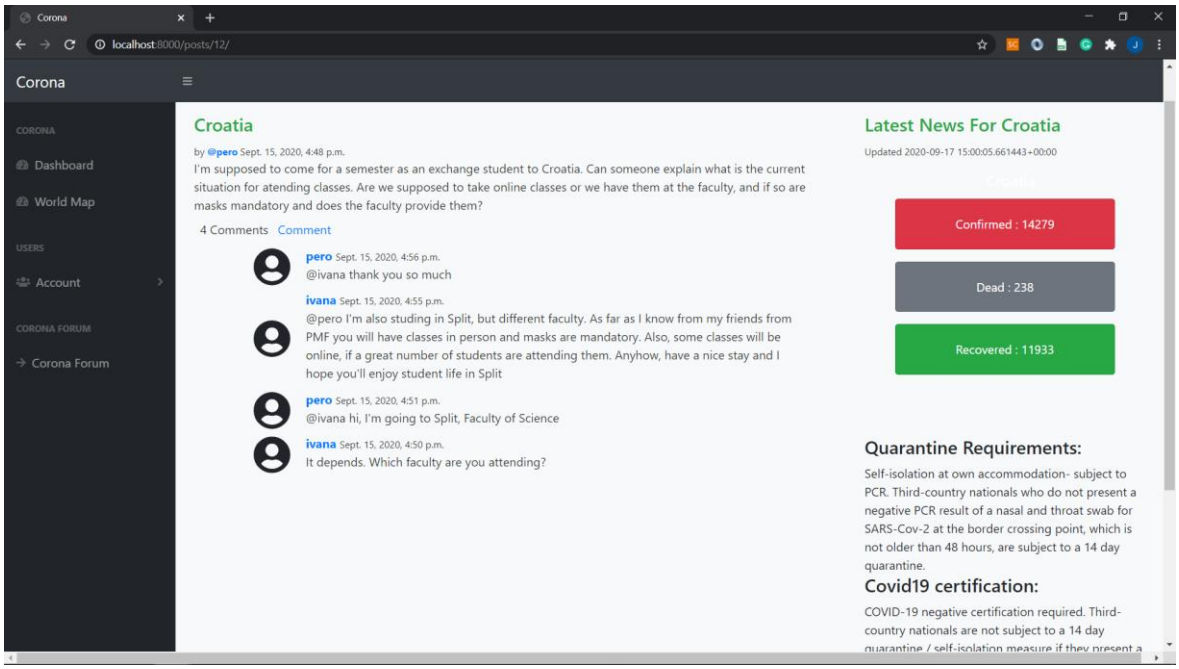
Na slikama 3.3, 3.4 i 3.5 prikazan je dio forum aplikacije sa glavnim funkcionalnostima registracije, prijave, objavljivanja, komentiranja te prikazivanja podataka sa API-a o najnovijim vijestima za putovanje u vrijeme Korone.



Slika 3.3 Izgled naslovne stranice foruma



Slika 3.4 Prikaz objava na naslovnoj stranici foruma



Slika 3.5 Prikaz komentara objave

3.4.3. Corona Dashboard

Glavni dio aplikacije koji prikazuje grafikone o podacima širenja Korona virusa implementiran je koristeći Dash i Plotly. Za svaki pojedini grafikon napravljena je Dash aplikacija, a biblioteka django_plotly_dash omogućuje da se prikazuje više Dash aplikacija na jednoj mrežnoj stranici na sljedeći način:

```
<div class="justify-content-center text-center"
style="height: 100%;width: 100%;">
      {% plotly_app name='Bar' ratio=0.6 %}
</div>
<div class="justify-content-center text-center"
style="height: 100%;width: 100%;">
      {% plotly_app name='Pie' ratio=0.6 %}
</div>

<div class="justify-content-center text-center"
style="height: 100%;width: 100%;">
          {% plotly_app name='Line' ratio=0.6 %}
</div>
```

Primjer realiziranja grafa pomoću Dash-a i Plotly-a pojasniti će se na primjeru izrade grafa raspršenosti za prikaz broja oboljelih po državama svijeta. Prvi korak je učitati podatke sa

stranica Europskog centra za sprečavanje i kontrolu širenja bolesti. U ovom slučaju korištena je Python biblioteka Pandas za otvaranje CSV dokumenta preko URL poveznice:

```
df = pd.read_csv("https://opendata.ecdc.europa.eu/covid19/casedistributio
n/csv")
```

Sljedeće, potrebno je grupirati podatke po državi i zbrojiti vrijednosti oboljelih slučajeva za pojedinu državu:

```
data = df.groupby(["countriesAndTerritories"])[["cases"]].sum()

data.reset_index(inplace=True)
```

Potom, definiramo raspored elemenata na stranici:

```
            app.layout = html.Div([
                html.H4('Choose a country to show the number of cases
            daily:'),
                dcc.Dropdown(
                    id="dropdown",
                    options=col_options,
                    value='Croatia'
                    ),
                html.H4('Choose to show number of cases or deaths:'),
                dcc.Dropdown(
                    id="dropdown2",
                    options=[
                        {'label': u'Cases', 'value': 'cases'},
                        {'label': 'Deaths', 'value': 'deaths'},
                    ],
                    value='cases'
                    ),
                dcc.Graph(
                        id="graph",
              figure=px.scatter(df.loc[df["countriesAndTerritories"] ==
"Croatia"],x="dateRep", y="cases",color="month",
hover_data=["countriesAndTerritories"],height=700))])
```

Vidljivo je da su stvorene dvije Dash komponente za padajući izbornik: jedna da korisnik može odabrati državu čiji će se podaci prikazivati na grafu, a drugi izbornik da se odabere opcija prikazivanja broja oboljelih ili broja umrlih. Stvoren je i jedan graf element koristeći Plotly Express modul, pomaže stvaranje grafa pozivom jedne funkcije. Za omogućavanje interakcije s korisnikom potrebno je u callback dijelu Dash

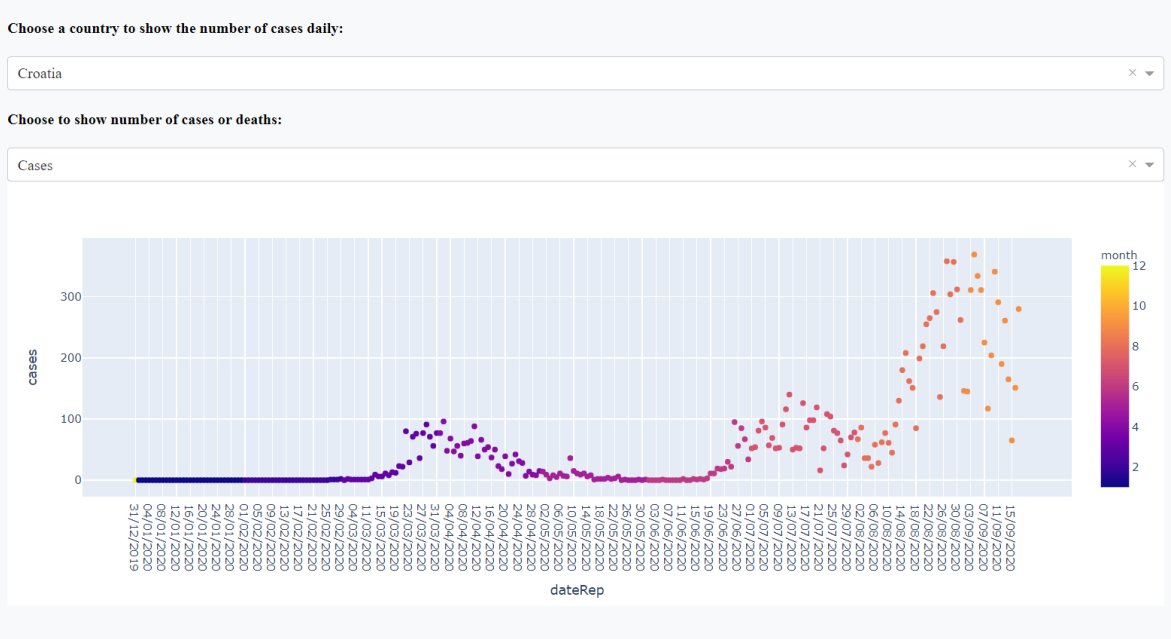
aplikacije navesti koji su ulazni elementi (čije se promjene vrijednosti prate), a koji izlazni (oni koji se ažuriraju promjenama na ulaznim elementima):

```
@app.callback(
    Output('graph','figure'),
    [Input(component_id='dropdown',
component_property='value'),Input(component_id='dropdown2',
component_property='value')]
    )
```

Promjenom vrijednosti na jednom od padajućih izbornika, automatski se poziva callback funkcija:

```
def update_graph(option_slctd,option_slctd2):
    return px.scatter(df.loc[df["countriesAndTerritories"] ==
option_slctd],
     x="dateRep", y=option_slctd2,color="month",
hover_data=["countriesAndTerritories"])
```

Rezultat prethodno navedenog koda prikazan je na slici 3.6.



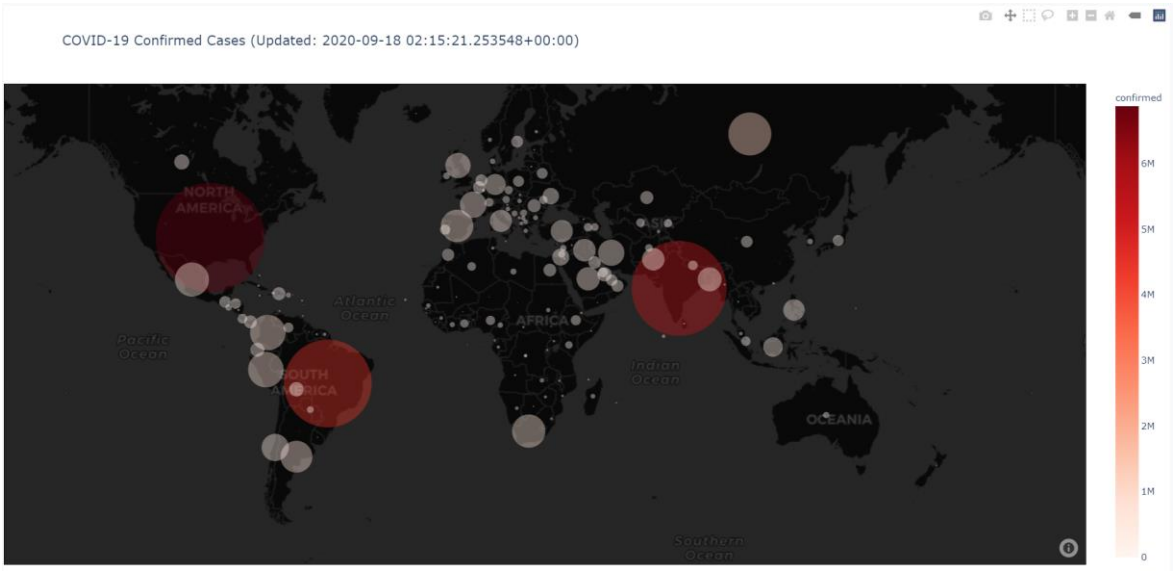
Slika 3.6 Prikaz interaktivnog grafikona raspršenosti

Osim prikaza podataka kao točaka na grafikonu, moguć je i prikaz podataka na geografskoj karti. Naravno, postoji uvjet da u skupu podataka treba postojati podatak o geografskoj dužini i širini za podatak koji se iscrtava na karti. Stoga, prilikom prikaza podataka o širenju Korona virusa grupiranih po državama i smještenih na geografsku kartu, korišten je

API preko URL putanje https://www.trackcorona.live/api/countries koji pruža za svaku državu broj oboljelih, umrlih, ozdravljenih te geografske koordinate države. Kod kojim je realiziran ovakav prikaz:

```
fig2 = px.scatter_mapbox(df, lat="latitude", lon="longitude",
                         hover_name="location",

hover_data={"location":False,"latitude":False,"longitude":Fal
se,"confirmed":True,"recovered":True,"dead":True,"updated":Fa
lse},

color_continuous_scale=px.colors.sequential.Reds,

color="confirmed",size="confirmed",size_max=100,
                         zoom=0.75, height=800,
                         opacity=0.4,
                         title = 'COVID-19 Confirmed Cases
(Updated: ' + df["updated"][0] +')'
                         )
fig2.update_layout(mapbox_style="carto-darkmatter",
mapbox_accesstoken=token)
fig2.update_layout(margin={"r":0,"l":0})
```

Prikaz podataka kao rezultat prethodno navedenog koda nalazi se na slici



Slika 3.7 Raspršeni graf na geografskoj karti

4. Zaključak

U svijetu u kojem se svaki dan dobiva ogromna količina podataka, čovjeku je gotovo nemoguće pronaći smisao i dobiti korisne informacije analizirajući ih u tabličnom prikazu. Zadatak vizualizacije podataka je prikazati tablične podatke u obliku grafikona te primjenjujući čovjekove sposobnosti prepoznavanja uzoraka i boja omogućiti brzo donošenje odluka na osnovi njihovih analiza.

Kroz ovaj je rad pojašnjeno što su to podaci te na koje ih se načine može vizualizirati. Navedeni su neki od najpoznatijih okvira i biblioteka za rad s podacima i njihova vizualizacija. Kroz projekt, korištenjem Python biblioteke Plotly, prikazan je primjer vizualizacije podataka o širenju Korona virusa. Cilj projekta je bio napraviti jednostavne grafikone, lako razumljive i intuitivne, te ih ukomponirati u mrežnu aplikaciju, a sve da se podigne svijest o ozbiljnosti situacije pandemije Korona virusa. Sve to je i realizirano, a veliku ulogu imao je i Dash, Python okvir za mrežne aplikacije, koji je omogućio interakciju korisnika s grafikonima.

Područje vizualizacije podataka je mnogo veće od onog prikazanog u ovom radu, čija je ideja bila upoznati se s osnovama vizualizacije podataka i dostupnim alatima. Plotly je samo jedna od mnogih dostupnih biblioteka za rad s podacima, ali za početnike u području vizualizacije je odlična jer pruža jednostavnost i intuitivnost izrade grafikona te se jednostavno ukomponira u mrežnu aplikaciju kao što je Django, ili Flask.

Literatura

[1] CROOKS, R.; Lankow, J.; Ritchie, J. *Infographics: The Power of Visual Storytelling*. Hoboken: John Wiley, 2012.

[2] YAU, N. *Visualize This: The Flowing Data Guide to Design, Visualization, and Statistics*. Indianapolis: John Wiley, 2011.

[3] HUBSPOT, Data visualization 101: how to design charts and graphs, https://msucreativecomp.files.wordpress.com/2016/08/data_visualization_101_how_to_design_charts_and_graphs.pdf , 14.9.2020.

[4] MATIAS, M. Visualize It! A Comprehensive Guide to Data Visualization. https://www.netquest.com/hubfs/docs/ebook-data-visualization-EN.pdf?hsCtaTracking=522c24a0-0231-40c5-9fdd-c449a5b64b92%7C174f96ae-1e66-4aec-9aa3-c7478eb1390d, 15.9.2020.

[5] MICROSTRATEGY, Data Visualization. https://www.microstrategy.com/us/resources/introductory-guides/data-visualization-what-it-is-and-why-we-use-it, 15.9.2020.

[6] ENFORY, A. 15+ Best Data Visualization Tools of 2020 (with Examples), https://www.adamenfroy.com/data-visualization-tools, 15.9.2020.

[7] SOLOMON, B. Python Plotting With Matplotlib, https://realpython.com/python-matplotlib-guide/, 14.9.2020.

Sažetak

U radu je definirano što su to podaci, navedeni su tipovi podataka te načini obrade podataka. Vizualizacija i načini vizualizacije podataka su prikazani i kroz teoriju i kroz izradu mrežne aplikacija u kojoj je implementirana vizualizacija podataka o širenju Korona virusa. Navedeni su i uspoređeni najpoznatiji okviri i alati za vizualizaciju podataka. Odabrana je biblioteka Plotly za izradu grafikona prikazanih u aplikaciji. Korištenjem Python okvira za razvoj mrežnih aplikacija naziva Django, realizirani su temeljni dijelovi aplikacije kao što su baza podataka, mogućnost registracije i prijave u aplikaciju, izrada stranica aplikacije i definiranje komunikacije lokalnog servera i klijentskog računala. U kombinaciji s Plotly-em korišten je Dash za omogućavanje interakcije korisnika sa grafikonima. Korištenjem API-a dohvaćeni su podaci u CSV i JSON obliku te su prikazani u obliku linijskog grafikona, stupčastog dijagrama, kružnog grafikona, raspršenog grafikona i mape.

Ključne riječi: podaci, vizualizacija, plotly, django, dash, interakcija, grafikon

Summary

This paper explains what data is, states different types of data, and different ways of data processing. Data visualization and ways to visualize data have both been explained through theory and through the development of the web application which visualizes data about the spreading of the Coronavirus. The most well-known frameworks and tools of data visualization have been listed and compared. Python library Plotly has been chosen to visualize data in the application. Django, the Python framework for developing web applications, was used to implement database, registration, login, building web pages, and to implement the communication between the local server and the client computer. Dash is used with Plotly to enable interaction between users and graphs. Data in CSV and JSON form has been fetched from web using API and shown in a form of line chart, bar chart, pie chart and scatter chart and map.

Keywords: data, visualization, plotly, django, dash, interaction, chart

Skraćenice

GUI	*Graphical User Interface*	grafičko korisničko sučelje
CSV	*Comma-separated values*	vrijednosti odvojene zarezom
API	*Application Programming Interface*	aplikacijsko programsko sučelje
JSON	*JavaScript Object Notation*	JavaScript objektno zapisivanje
XML	*Extensible Markup Language*	jezik za označavanje podataka
HTML	*HyperText Markup Language*	prezentacijski jezik za izradu mrežnih stranica
SVG	*Scalable Vector Graphics*	jezik za prikazivanje vektorske grafike
CSS3	*Cascading Style Sheets*	stilski jezik za mrežne stranice
DOM	*Document Object Model*	objektni model dokumenta
MVT	*Model-View-Template*	
SQL	*Structured Query Language*	strukturni upitni jezik

Privitak

Instalacija programske podrške

Za pokrenuti aplikaciju potrebno je osigurati dostupnost Python biblioteka koje su navedene u *requirements.txt* dokumentu koji je dio projekta. Preporuča se stvaranje Conda okruženja te instaliranje navedenih biblioteka pomoću naredbe `pip install -r requirements.txt`. Potom se u direktoriju projekta django_dash_plotly, uz uvjet da je pokrenuto novostvoreno okruženje, pokrene aplikacija naredbom u Anaconda Prompt-u `python manage.py runserver`. Nakon toga, potrebno je u web pregledniku upisati *http://localhost:8000* i aplikacija će se pokrenuti. Rad s aplikacijom je jednostavan i prethodno opisan u samom radu.