**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Sveučilišni diplomski studij računarstva**

**Informacijske i podatkovne znanosti, DRD**

**SVJETSKA I DRŽAVNA STATISTIKA COVID-19**

**Vizualizacija podataka**

**Matej Dmitrović**

**Osijek, 2020.**

Contents

[1. OPIS PROJEKTA 4](#_Toc49196368)

[1.1 Korištene tehnologije 4](#_Toc49196369)

[1.2 Programsko rješenje 5](#_Toc49196370)

[1.2.1 HTML 5](#_Toc49196371)

[1.2.2 JavaScript/D3.js 6](#_Toc49196372)

[1.2.3 CSS 16](#_Toc49196373)

[1.3 Vizualizacija podataka 18](#_Toc49196374)

[1.3.1 Primjeri vizualizacije 18](#_Toc49196375)

[2. KORIŠTENI IZVORI 23](#_Toc49196376)

[2.1 GitHub 23](#_Toc49196377)

# 1. OPIS PROJEKTA

COVID-19 iz porodice koronavirusa je već znatno vrijeme aktualna tema cijeloga svijeta. Ovaj virus je utjecao na sve države i njihove stanovnike. Neke države su brzo i ošrto reagirale na širenje virusa, dok neke gotovo ignoriraju ovaj novonastali problem. Zbog značaja ovoga virusa, napravljena je vizualizacija njegovog utjecaja – od toga koliko je ljudi ozdravljeno, do toga koliko je ljudi umrlo. Ova vizualizacija je opisana programskim rješenjem i dizajnerskim rješenjem. Bit će moguće vidjeti kako je jedno rješenje utjecalo na drugo, te kako su ta dva rješenja uopće bila moguća.

## 1.1 Korištene tehnologije

Za vizualizaciju projekta korištene su razne tehnologije u jedinstvu. Prva tehnologija od značaja jest HTML opisni jezik. Pomoću HTML-a definirana je opća struktura stranice, a tako i raposred vizualizacije. HTML također donosi mogućnost korištenje *<svg>* elementa koji je zaslužan za oslikavanje grafova kako bi se informacije mogle u što boljem obliku predoćiti korisniku/klijentu.

Dok HTML definira podjelu stranice, skirptni jezik JavaScript služi za dohvaćanje podataka i izgradnju grafova nad zim podacima. Dok sam JavaScript može dinamično manipulirati elementima stranice, puno je teže raditi kompliciranje zadatake poput iscrtavanja samih grafova. U tu svrhu korištena je biblioteka D3.js koja je specijalizirana za prezentiranje podataka na raznim web-stranicama.

Pomoću JavaScript-a i njene biblioteke D3.js mogu se prikazati razni dijagrami, no za izgled same web-stranice se koristi CSS opisni jezik. CSS je također korišten za definiranje stilova pojedinih elemenata koji su zajednički svim elementima, tako da se dio posla D3.js biblioteke prebacuje (i bolje organizira) u CSS-u.

Zbog potrebe za povezivanjem sa vanjskim resursima, korišten je program Laragon za stvaranje virtualnih servera. Laragon sa sobom donosi PHP i MySQL koji nisu korišteni u ovom projektu.

Za verzioniranje projekta zaslužan je git-baziran sustav GitHub, dok je za dohvaćanje podataka zaslužan REST API.

## 1.2 Programsko rješenje

### 1.2.1 HTML

HTML je korišten kao temelj cijelog ovog projekta. Prije svega drugog, u *<head>* elementu definirane su poveznive na CSS dokumenta i D3.js biblioteku. U vidljivom dijelu web-stranice određenu su dvije sekcije: jedna za statistiku vezanu uz svijet, potom druga za statistiku vezanu uz pojedine države. Obje sekcije u sebi sadrže *<article>* elemente koji služe kao prostor za umetanje grafova. Kako bi bilo moguće razlikovati te elemente, svaki *<article>* ima posebnu klasu. Na kraju cijelokupne hijerarhije su paragrafi i upisni elementi.

Paragrafi ponajviše služe kao dodatan izvor informacije gdje se statistika daje izravno korisniku/klijentu. Kao i kod *<article>* elementata, paragrafi su diferencirani korištenjem klasa. Paragrafi koji ne prikazuju određenu statistiku služe za grupiranje i opisivanje upisnih elemenata. U ovom projektu su korišteni gumbovi i jedan padajući izbornik kao upisni elementi. Definiranjem tih elemenata u samom HTML-u se ne postiže nikakva dinamična funkcionalnost – ta funkcionalnost će biti objašnjena tek u JavaScript dijelu rješenja. Na Slici 1.1. dan je primjer jedna HTML sekcije. Osim već navadenih elemenata, korišten je i *<h1>* element za prikazivanje naslova tog odjeljka. Na slici se također može vijdeti kako svaki element ima svoju određenu klasu. Detalji tih klasa će biti obrađeni u CSS dijelu rješenja. Još jedan bitan element je *<span>* element. Pomoću njega, prilikom dinamičkog mijenjanja teksta, moguće je samo mijenjati jedan dio teksta – u ovom slučaju samo brojke. *<span>* element također ima semantičku ulogu zbog istog razloga jer se pregledinku može dati do znjanja da se radi o naglasku jednog broja.



Slika 1.1. Primjer HTML strukture odjeljka za prikaz statistike po državi.

Na kraju web-stranice nalazi se zaglavlje sa njezinim kratkim opisom te autorskim podacima i podacima o korištenom API-ju. Na slici 1.2. je dan prikaz HTML strukture zaglavlja. Zaglavlje je podjeljeno na više djelova korištenjem *<div>* elementima. Svaki dio sadrži jedan od navedenih opisa. Zaglavlje samo po sebi ne igra veliku ulogu u vizualizaciji podatak, ali je ključan element svih web-stranica.



Slika 1.2. Prikaz HTML rješenja zaglavlja web-stranice.

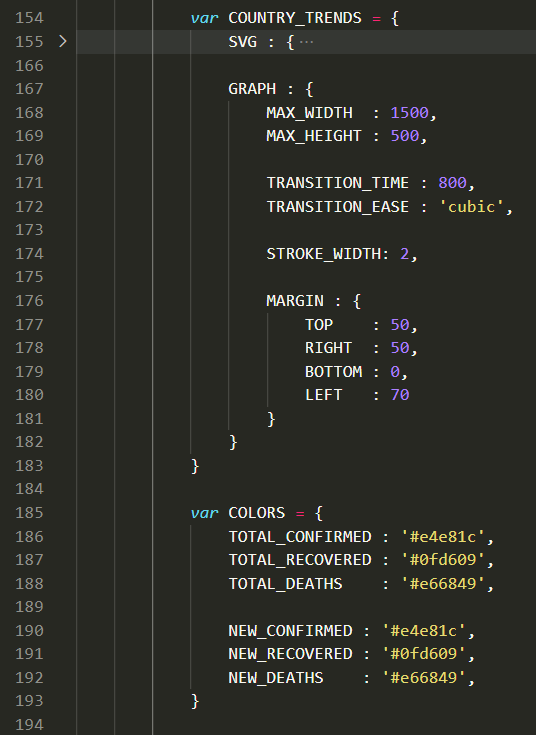
### 1.2.2 JavaScript/D3.js

Cijelokupna JavaScript skripta je napisana unutar *<script>* HTML elementa te se zbog toga kod može činiti ne urednim. Ipak, kod je moguće podjeliti u tri osnova dijela:

1. Definiranje konstanti.
2. Dohvačanje podataka.
3. Stvaranje grafova.

#### 1.2.2.1 Konstante

U prvom dijelu se definiraju konstante koje pomažu ako dođe do naknadnih promjena tokom izvedbe projekta. Ovo se najbolje vidi u slučaju samih grafova kada je potrebno mijenjati jedan parametar na više mjesta, a taj parametar (poput visine npr.) se može često mijenjati. Slika 1.3. je isješak konstanti. Dan je primjer konstanti vezanih uz graf za pojedine države – sama konstanta je podjeljena na dva djela radi bolje preglednosti i organizacije podataka. Prvi dio definira visinu i dužinu *<svg>* elementa te njezine margine, dok drugi dio definira svojstva vezana uz sam graf i njegove tranzicije. Također se na isoj slici mogu vidjeti konstante vezane uz boje. Definiranjem boja osigurava se jednoliko korištenje boja u svim grafovima. Boje su podjeljene za sveukupne podatke i nove podatke. Boja za nove podatke je svjetlija verzija boje za sveukupne podatke kako bi se mogla vidjeti povezanost između njih.



Slika 1.3. (155-183) JavaScript konstante vezane uz graf za pojedine države i (185-193) konstante vezane uz boje pojedinih parametara kod svih grafova.

#### 1.2.2.2 Dohvaćanje podataka

Drugi dio koda bavi se dohvaćanjem podataka iz vanjsog API-ja putem REST-a. To znači da je na određeni link (API) potrebno poslati ‘GET’ zahtjev kako bismo dobili željene podatke. U ovom projeku dolazi do spajanja na COVID-19 API. Svi podaci koji se dobiju natrag dolaze u JSON formatu. Prilikom prvog spajanja traže se općeniti podaci (engl. *summary*). Ti podaci sadrže sveukupno svjetsko stanje – koliko ima zaraženih, ozdravljenih i mrtvih – te koliko ima tih istih podataka, ali po državama. Ta dva skupa podataka služe za stvaranje prva dva grafa koji se nalaze u prvoj sekciji. Dio koda za prvo dohvaćanje se nalazi na slici 1.4. Iz nje se vidi kako je potrebno imati postban *XMLHttpRequest* objekt kako bi bilo moguće poslati zahtjev. Budući da je slanje zahtjeva asinkrona funkcija i da kod neće čekati dok ne dobije odgovor, ostatak koda nalazi se u funkciji *.onload* koja se poziva kada je odgovor poslan kako bi se program mogao slijedno izvršavati. Prije rada nad samim podacima provjerava se je li došlo do kakve greške u dohvaćanju. Funckcije *InsertGlobalSummary* i *InsertGlobalStats* služe za stvaranje već navedana prva dva grafa. Obje funkcije kao argumente primaju skup podataka nad kojim se graf stvara, klasu *<article>* elementa u kojem se nalaze te podaci za *<svg>* element i grafove. Svi od tih elementa, osim skupa podataka, su navedeni u dijelu sa konstantama.



Slika 1.4. Isječak koda za početno dohvaćanje podataka.

Prije opisivanja tih funkcija, dolazi do još dva dohvaćanja podataka. Drugo dohvaćanje je prikazano na slici 1.5. gdje se dohvaćanj *slugovi*, odnosno skraćeni zapisi imena, država, zbog toga što nemaju svi API-ji standardni zapis za države. Ovaj korak je nužan kako bi padajući izbornik, gdje se odvija treće dohvaćanje podataka, mogao funkcionirati. Padajući izbornik se inicijalizira pomoću funkcije *CountryPicker*.



Slika 1.5. Isječak koda za dohvaćanje imena/*slugova* država.

JavaScript pomoću D3.js-a dinamično kreira opcije u padajućem izborniku. Kao izvor podataka se predaju *slugovi* država te se svakom *slugu* stvara i pridjeljuje jedna opcija. Promjenom odabira pozova se funkcija *SelectChanged* prikazana na slici 1.6. Dohvaćanje podataka se vrši na isti način pomoću već kreiranog *XMLHttpRequest* objekta. Kada se podaci dohvate, stvara se graf za prikaz statistike COVID-19-a po državama.



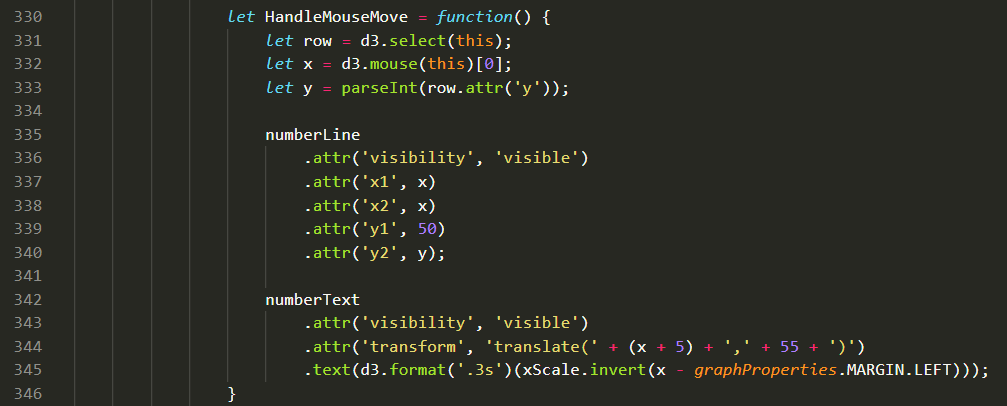
Slika 1.6. Funckija *SelectChanged* koja se poziva nakon promjene odabira u padajućem izborniku.

#### 1.2.2.3 Iscrtavanje grafova

Prvi graf, koji se iscrtava funkcijom *InsertGlobalSummary*, predstavlja globalne podatke o zaraženom, odravljenom i umrlom stanovništvu. To je horizontalni stupičasti graf gdje su vrijednosti neovisne osi diskretni zapisi ovih parametara na engleskom jeziku: “CONFIRMED”, “RECOVERED” i “DEATHS”. Ovisna os predstavlja broj ljudi u jednoj od navedenih kategorija. Također, na ovom grafu se prikazuju i za dan kada korisnik/klijent se služi web-stranicom, tako da se na uvidom mogu vidjeti kolika je današnja promjena s obzirom na sveukupno stanje. Mora se naglasiti da ovisna os nije linearna, nego je korijenske ovisnosti. Korijenska ovisnost je uzeta zbog: velikih razlika između današnjih i sveukupnih vrijednosti; velike razlike između broja mrtvih spram broja oboljelih i ozdravljenih. Navedene ovisnosti su implementriane pomoću D3.js objekta *d3.scale*. Pomoću *d3.scale* objekta vizualizirane su osi za svaku od ovisnosti pomoću *d3.svg.axis()* objekta. Zbog svoje statičnosti, ovaj graf ne koristi *.data* metodu za umetanje podataka, već se umetaju pomoću funkcije *GraphInsertation()* čiji je dio koda dan na slici 1.7. Najprije se na graf dodaje HTML element *<g>* koji služi za grupiranje stupova. Svaki *<g>* element sadrži stup koji pokazuje sveukupni broj ljudi i današnji broj ljudi u jednoj kategoriji. Svojstva tih stupova već su deklarirana u konstantama dijelu koda, stoga se mijenjaju samo y vrijednost pozicije i njihove dužine. Y ovisi o kategoriji kojoj stup pripada, dok dužina o samoj vrijednosti te kategorije. Stupovi za najnovije podatke su deset piksela spušteni dolje kako bi se bolje uvidjela razlika. Kako bi se podaci mogli lakše pročitiati na korijenskoj skali, napravljene su metode *HandleMouseOver, HandeMouseMove* i *HandleMouseOut* za očitavanje podataka na pojedinom stupu kada se mišem uđe, miče, odnosno izaže iz stupa. Funkcije *HandeMouseOver* i *HandleMouseOut* stvaraju, odnosno brišu liniju, dok funkcija *HandleMouseMove* mijenja poziciju stupa kada se miš pomake. Cijela definicija navedene funkcije je dana na slici 1.8. Najbitniji dio te definicije je pozivanje *.invert()* metode nad ovisnom skalom, zbog toga što u ovom slučaju znamo vrijednost prikazanu na ekranu, a moramo dobiti vrijednost prikazanu na grafu. Ostatak koda se bavi pomicanjem pokazivačke crte.



Slika 1.7. (373-392) Dio koda funkcije *GraphInstertation()* i (394-396) njeno pozivanje.



Slika 1.8. Definincija funckije *HandleMouseMove*.

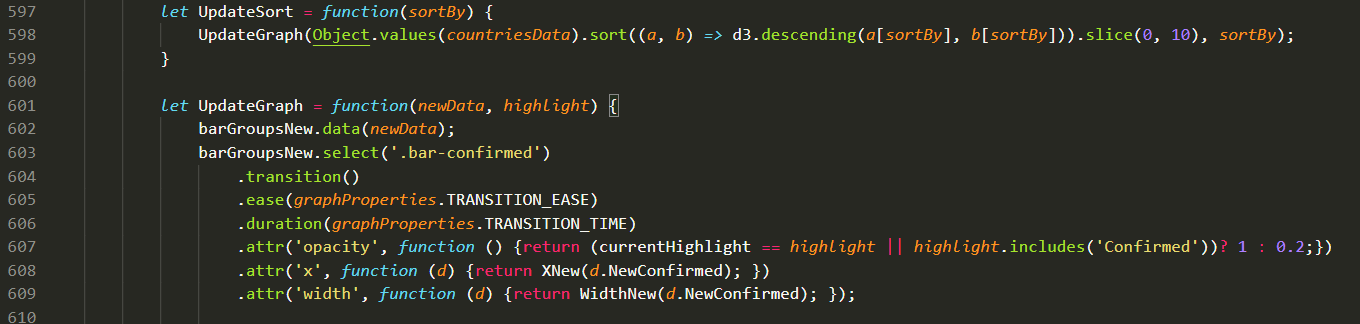
Na kraju funkcije *InsertGlobalSummary* se ispisuju točno numerirani podaci na *<p>* elemente deklarirane u HTML dijelu koda radi lakšeg očitavanja aktualnih podataka.

Drugi graf, koji se iscrtava pomoću funkcije *InsertGlobalStats*, je složeniji od prošlog grafa. Naime, ovaj graf se sastoji od dva pod-grafa koji dijele istu neovisnu os. Ovdje je neovisna os pojedina država, dok su ovisne osi današnji podaci (lijeva strana) i sveukupni podaci (desna strana) za tu državu. Podaci su, kao i u prošlom grafu, broj oboljelih, ozdravljenih i umrlih, s time da se sve tri kategorije prikazuju za pojedinu državu. Ovaj graf također nudi mogućnost sortiranja po određenoj kategoriji, tako da je potrebna mogućnost sortiranja podataka. D3.js biblioteka nudi brzo rješenja za sortiranje skupa podataka koje je prikazano na slici 1.9. Potrebno je samo odabrati po kojim se svojstvima podaci trebaju uspoređicati i na koji način (*ascending/descending)*. Metoda *.slice()* služi za odabiranje prvih deset podataka, budući da će ih se toliko samo i prikazivati. U varijabli *countries* se spremaju imena država koje se u nastavku koda koriste za definiranje neovisne osi.

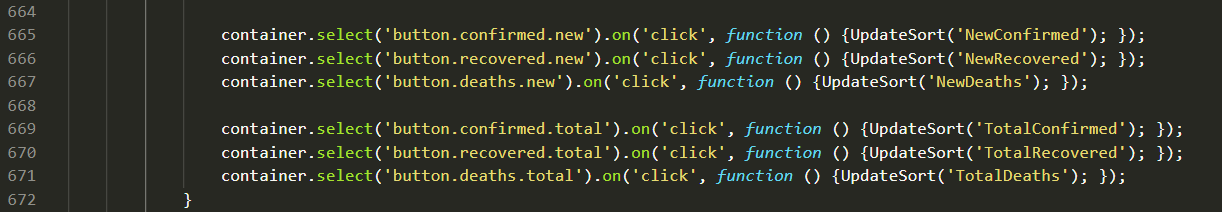


Slika 1.9. Prikaz koda za sortiranje podataka i dohvaćanje pojedinih država.

Zbog zajedničke neovisne osi, inicijalizira se samo jedna neovisna skala, te dvije ovisne skale za svaki skup podataka. Postupak pravljenja grafa je sličan kao u prvom grafu, s time da se ovdje koristi *.data* metoda za postavljanje skupa podataka. Oba pod-grafa koriste isti skup, jedina je razlika koje vrijednosti koriste. Budući da su novi podaci predstavljeni na lijevoj strani, dužina i pozicija stupova se mora računati na taj način da se doima kako stupovi proizlaze iz centra stranice. Računanje pozicije i duljina za desni pod-graf je puno lakše jer je slučaj sličan u prvom grafu, samo što su stupovi ovog pod-grafa translatirane u desnu stranu tako da stupovi počinju od centra. Interakcija ovog grafa se postiže klikom na jedan od šest gumba za svaku od pojedinih kategorija. Klikom gumba ponovno se sortiraju podaci, ali ovog puta prema novom odabranom kriteriju, te se već postojeći stupovi mijenjaju u ovisnosti o novo-sortiranim podacima. Funkcije *UpdateSort* i *UpdateGraph* su zaslučne za ovu funkcionalnost. Njihov isječak je prikazan na slici 1.10. gdje se isčitava da funkcija *UpdateSort* jednostavno već dohvaćene podatke ponovno sortira po novom kriterju te poziva funkciju *UpdateGraph* kako bi se graf prilagodio novim podacima. Najbitnija značajka funkcije *UpdateGraph* je postavljanje novog skupa podataka i isticanje samo jedna kategorije. Isitacnje se postiže mjenjanjem vidljivosti (engl. *opacity*) stupova – svi stupovi koji nisu istaknuti imaju smanjenju vidljivost. Osim stupova, na grafu se ažuriraju i države koje prate nove podatke. D3.js dolazi sa *transition()* metodom koja postupno nanosi promjenu. Bez te metode, podaci bi se ažurirali u jednom trenutku – tada bi se promjene teže primjetile. Ova funkcionalnost se dodaje na gumb pomoću *.on()* metode u kojoj se može definirati da će se funkcija pozvati kada miš klikne na jednom od gumba. Zbog već definiranih gumba u HTML dijelu koda, potrebno je pronaći gumbe koristeći *d3.select()* metodu, što je jednostavno jer svaki gumb ima svoju posebnu putanju klasa.



Slika 1.10. Isječak koda za sortiranje postoječih podataka po novim kriterijima.



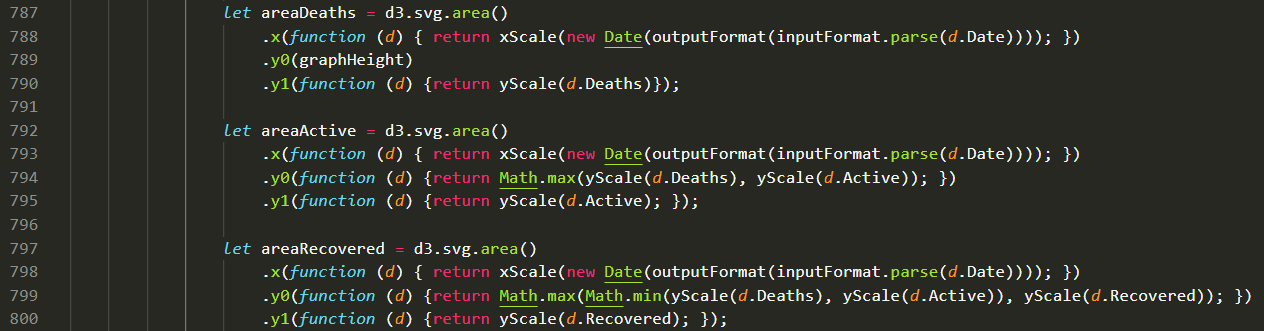
Slika 1.11. Prikaz postavljanja okidaća na pojedini gumb. Klikom na gumb se poziva funkcija *UpdateSort* sa određenim kirterijem sortiranja.

Treći graf, koji se iscrtava pomoću funkcije *CountryTrends*, ima složenije pripreme nego prijašnja dva grafa. Naime, kako bi se ovaj graf mogao iscrtiati, moraju se dohvatiti podaci o pojedinoj državi. Kako korisik/klijent ne bih morao sam upisivati države, napravljen je *<select>* element u HTML dijelu koda koji se puni *<option*> elementima čije je vrijednost *slug* pojedine države. Slika 1.12. sadrži isječak koda, dio koji je napisan u D3.js-u i dio koji je napisan u običnom JavaScript-u. Prvi dio, u funkciji *CountryPicker*, postavljaju se opcije unutar padajučeg izbornika na sličan način kako su se postavlje linije u prošlim linijskim grafovima. Razlika je ovdje što se mijenjaju drugi atributi – umjesto “x”, “y”, “width” i sl. ovdje se mijenja samo vrijednost opcije i tekst koji će imati. Kada se promjeni vrijednsot padajućeg izbornika, poziva se funkcija *SelectChanged* sa *slugom* dražve koja je odabrana. Ta funckije šalje zahtev API-ju za podatke o toj državi. Ukoliko je nastao problem, javit će se greška u konzoli. Ukoliko je sve u redu, poziva se funkcije *CountryTrends* koja iscrtava graf krivulje.



Slika 1.12. Prikaz koda funkcija *CountryPicker* i *SelectChanged* koje služe za odabiranjem pojedine države.

Postoje države koje nemaju podatke o COVID-19 koronavirusu. Zbog toga, može nastati problem ukoliko se želi iscrtati graf bez podataka. Postoje dva pristupa kako se ovaj problem može riješiti: brisanjem država u izborniku koje nemaju podatke ili ispisivanjem poruke korisniku/klijentu da odabrana država nema podataka. Prvi način se čini kao idealno rješenje, ali sa sobom bi donio dva nova problema. Jedan od njih je to što klijent koji traži podatke za neku od država bez podataka ne bih znao zašto da država nije izlistana. To je moguće rješiti dodavanjem paragrafa u kojem se iskazuje da nemaju sve države podatke. Ali, veći problem je kako u kodu vidjeti ima li ta država podatke: skup podataka koji sadrži *slugove* i imena država nikako ne ukazuje ima li ta država ili nema podatke o COVID-19-u - zbog toga bi se za svaku državu morao zvati poseban poziv API-ju kako bi se samo provjerila postojanost podataka. S obzirom na veliki broj država, svaki put kada bih netko posjetio stranicu, stranica bi morala slati 100 ili više dodatnih poziva što bi moglo uzrokovati preopterećenje API sustava. Zbog navedenih problema, odabrana je druga mogućnost koja ispisuje poruku ako država ne sadrži podatke. Ostatak koda je dosta sličan već opisanim grafovima. Razlika u ovom grafu je to što se ovdje koriste područja i linije. Korištenjem područja i linija je drugačije nego samo izrađivanje stupova. Moraju se pozvati posebne D3.js metode *d3.svg.line()* i *d3.svg.area()* za svaku kategoriju podataka. Budući da ovaj graf prikazuje ovisnost oboljelih, ozdravljenih i umrlih o datumu, postojat će tri posebne linije i područja. Zbog isticanja važnosti i boljeg pregleda, neke kategorije imaju veći prioritet iscrtavanja nego ostale. Kateogirja umrlih ima najveći prioritet – to znači ukoliko područje umrlih dijeli svoje područje s ostalim kategorijama, na grafu će samo ono biti prikazano. Drugi prioritet ima broj oboljelih, dok broj ozdravljenih ima najmanji prioritet. Slika 1.13. prikazuje implementaciju ovih prioriteta koje se definira u *.y0()* metodi. Ta metoda oznvačava mjesto na ovisnoj osi u kojoj počinje taj dio linije, dok *.y1()* označava kraj. Na taj način, područje za broj umrlih za *.y0()* ima sveukupnu veličinu grafa, dok područje za broj oboljelih mora očitati veću vrijednost između broja umrlih i oboljelih. Na sličan princip radi i područje za broj odravljenim, s time da ako postoji preklapanje između broja umrlih i oboljeih, potrebno je prvo nači koji je manji od to dvoje.

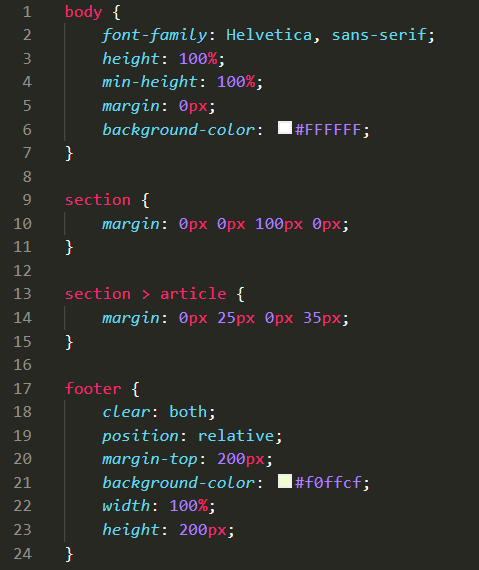


Slika 1.13. Isječak koda za deklariranje područja za svaku od tri kategorija.

Slično prvom grafu, ovdje se također mogu očitati aktualni podaci, ali za odabranu državi. Implementirano je na sličan način pomoću *<span>* elemenata gdje se mijenjaju samo vrijednosti unutar njih. Graf također podržava tranzicije ukoliko se odabere neka nova država. Nakon definiranja područja i linija, implementiranje stvaranja i ažuriranja je isto kao i u već opisanim grafovima. Manja razlike je kod implementiranja područja, gdje se umjesto metoda *.data()* koristi metoda *.datum()*. Paleta boja je konzistenta sa ostakom projekta.

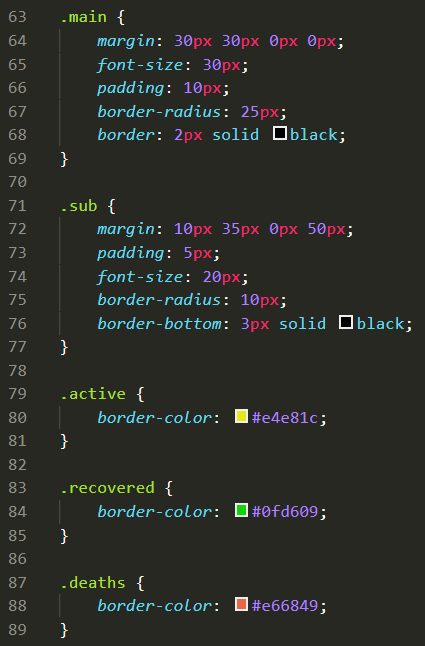
### 1.2.3 CSS

Nakon definiranje strukture i dinamičnog ponašanja stranice, CSS služi za opisivanje izgleda pojedinih klasa elemenata. Projekt sadrži dva izvora CSS-a: *webpage.css* i *graph.css*. U datoteci *graph.css* sadržana su dva manja opisa vezana isključivo uz grafove projekta, dok je većina pravila vezan uz općeniti izgled stranice opisana u *webpage.css*. Slika 1.14. prikazuje pravila za četiri najveće strukture: sekciju, članci unutar sekcije, zagljave te cijelokupnu web-stranicu. Unutar *body* elementa postavljena su pravila najmanjeg pripriteta koja vrijede za cijelu stranicu. Najznačajniji je font *Helvetica* koji se može vidjeti na osima grafova i paragrafima unutar stranice. Za sekciju i članke unutar sekcije najbitnija je margina kako bi se dobio dobar poredak elemenata na stranici, budući da samo definiranjem HTML elemenata se (najčesće) ne može dobiti željeni poredak. Za zagljavlje je najbitnije da se nalazi na samom kraju stranice, koliko god da je ta stranica visoka. To ponašanje se dobije tako što se u zaglavlju definira pravilo *clear: both*, a u tijeli stranice *min-height: 100%*. Nakon ovih osnovih pravila slijede pravila za pozicioniranje odjeljaka unutar zagljavlja.



Slika 1.14. Glavna CSS pravila iz datoteke *webpage.css*.

Za pojedine elementa i klase stranice, font je promjenjen u *Lucida Console* zbog znanstevog izgleda tog fonta koji odgovara temi projekta. Slika 1.15. prikazuje pravila za paragrafe i *<span>* elementa koje sadrže numeričke vrijednosti. Fleksibilnost klasa se dobiva kada se više klasa kombinira – tako npr. paragraf sa klasama “main active” će prikazivati istaknutiji broj sa žutom bojom ruba, dok će paragraf sa klasama “main recovered” također prikazivati istaknuti broj, ali sa zelenim rubom. Slično je kada mijenamo “main” sa “sub”, a (npr.) “active” ostavimo isto – tada se mijenja samo istaknutost broja dok je boja ruba ista. Ove klase se koriste u prvom i trećem grafu za prikazivanje akutalnih podataka.



Slika 1.15. Prikaz pravila za klase koje sadrže numeričke vrijednosti.

Pravila nakon toga samo popravljaju izgled naslovnim elemenata unutar sekcija kako bi imale veću važnost od naslovnih elemenata unutar zaglavlja. Zatim se definiraju pravila za izgled gumba. Gumbovi za sve tri kategorije su popunjeni i imaju okrugle rubove. Razlika je jedino u boji popune koja se postiže na sličan način u prošlom primjeru sa paragrafima. Bitno je napomeniti da gumb za oboljele još dodatno ima pravilo za crnu boju teksta zbog toga što je gumb žute boje, a bijeli tekst na žutoj boji se teško može pročitati. Nakon gumba su dvije opće klase koje miču element u desnu ili lijevu stranu pomoću pravila *float*.

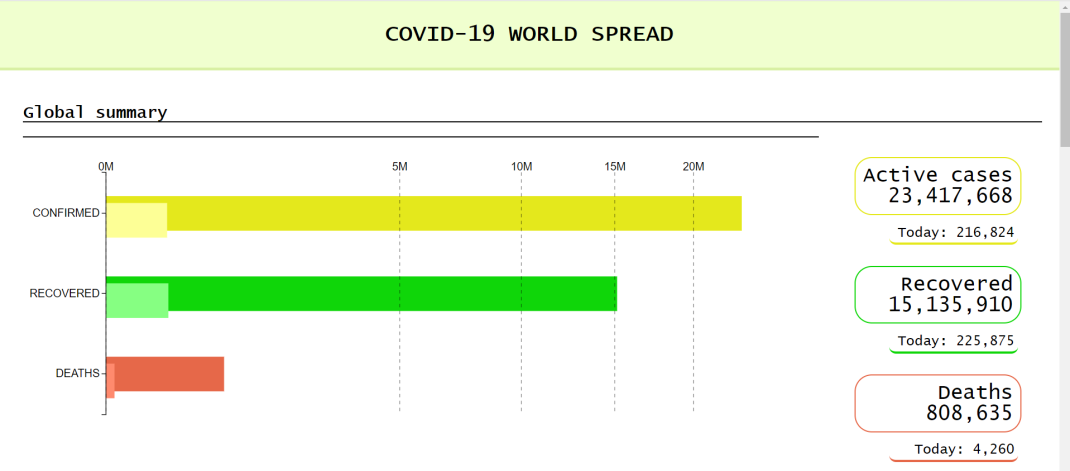
Na kraju dokumenta je pravilo za upozorenje ukoliko odabrana država u trećem grafu nema podataka. Ovo pravilo je posebije od ostalih zbog toga što ne vrijedi za klasu, nego samo za jedan jedini element sa ID-jem “no-data-warning”. Kod tog elementa je bitno da tekst bude uočljiv, a to se postiže time što je tekst crvene boje te ne nalazi na sredini grafa.

## 1.3 Vizualizacija podataka

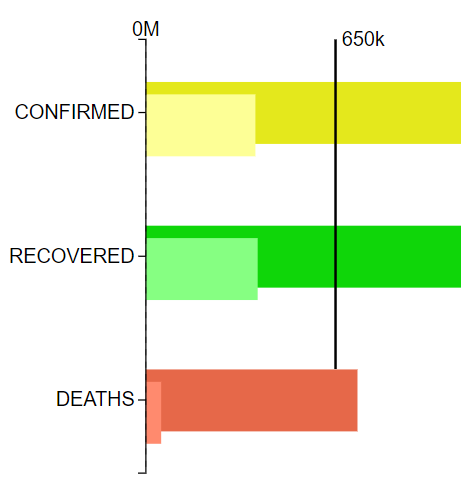
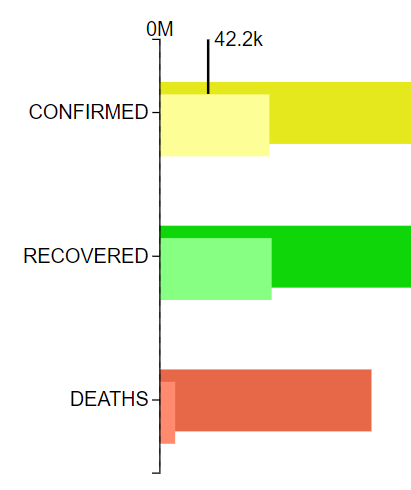
Nakon definiranje HTML-a, JavaScript-a i CSS-a, dobiva se web-stranicama sa prikazom statistika COVID-19 koronavirusa. Sveukupno stranica sadrži tri grafa (jedan graf ima dva pod-grafa) te par mjesta sa konkretnim podacima.

### 1.3.1 Primjeri vizualizacije

Prvi graf je horizontalni stupičasti graf koji prikazuje globalnu statustiku koronavirusa. Slika 1.16. prikazuje navedeni graf na web-stranici. Zato što je ovisna os korijenska, razmak među crtama za obilježavanje je sve manji. Zbog ne-linearne ovisnosti, na graf je dodana mogućnost iščitavnja podataka micanjem mišem čiji su primjeri prikazani na slika 1.17. i 1.18. na tom primjeru se također može vidjeti razlika između vrijednosti zbog korijenske ovisnosti osi. Na desnoj strani slike 1.16. prikazani su konkretni podaci koji također služe kao legenda. Aktivni slučajevi imaju žuti rub koji se podudara bojom oboljelih na grafu. Isto je za ozdravljene i umrle. Ispod ukupnih brojeva po kategoriji nalazi se broj slučajeva za dan kada klijent/korisnik otvori stranicu.



Slika 1.16. Prikaz grafa za globalnu statistiku COVID-19-a.



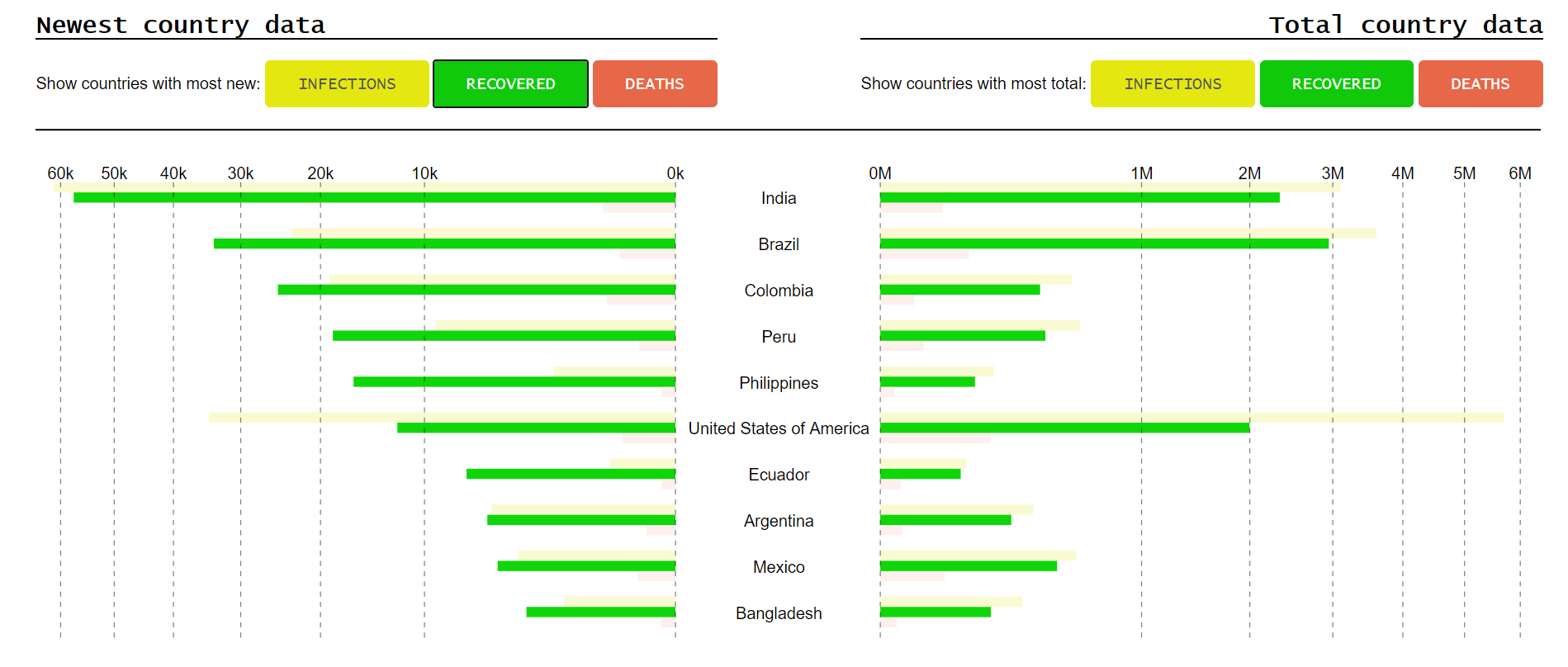
Slike 1.17. (lijeva) i 1.18. (desna) Primjer pokazivanja vrijednosti grafa micanjem miša.

Drugi graf sastoji se od dva pod-grafa: lijevi podgraf prikazuje najnoviju/današnju statistiku, dok desni prikazuje sveukupnu statistiku za pojedinu državu. Budući da oba pod-grafa dijele istu os sa državama, kada se sortira jedan graf, drugi pod-graf će se također ažurirati, ali ne će biti sortiran. Pod-grafovi se sortiraju odabirom jednog of šest gumba. Oba pod-grafa imaju gumbe za iste kategorije, ali jedan pod-graf će sortirati po najnovijim podacima dok će drugi sortirati da sveukupnim podacima.



Slika 1.19. Prikaz drugog grafa sa dva pod-grafa – (lijevi) današnji/novi slučejevi i (desni) sveukupni slučajevi po državi.

Jedan takav primjer je prikazan na slici 1.20. Odabran je gumb “RECOVERED” na lijevom pod-grafu kako bi se sortirale države ovisno o broju osoba koje su ozdravili danas. Odabirom jedne kategorije, sve ostale kategorije postanu manje vidljivi kako bi došla do izražaja odabrana kategorija. Ukoliko se ponovno žele vidjeti ostale kategorije, potrebno je odabrati isti gumb. Na slici 1.20. se također može primjetiti kako su podaci sortirani na lijevom pod-grafu dok na desnom nisu. Na slici 1.21. prikazan je slučaj kada je odabran gumb “DEATH” za sveukupne podatke. Događa se isto ponašanje kao u prošlom primjeru; ovog puta su podaci sortirani na desnoj strani dok na lijevoj nisu.

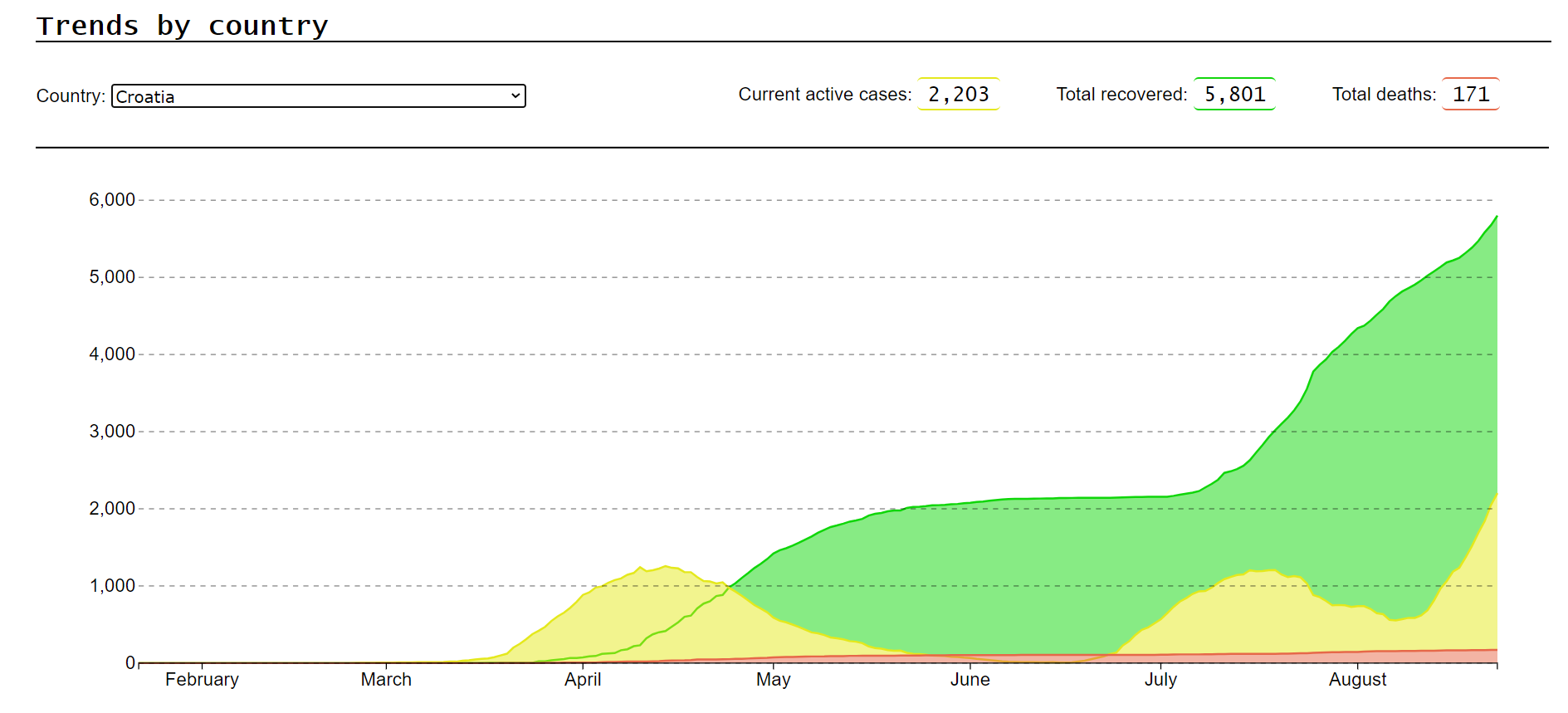


Slika 1.20. Primjer odabira “RECOVERED” gumba za najnovije podatke.



Slika 1.21. Primjer odabira “DEATHS” gumba za sveukupne podatke.

Zadnji graf prikazuje krivulje za sve tri kategorije po pojedinoj državi. Na slici 1.22. je dan primjer grafa u slučaju da je odabrana Hrvatska. U gornjem-lijevom kutu se nalazi padajući izbornik za odabir države dok se prekoputa aktualni podaci za odabranu državu. Kao u svim grafovima dosada, iste boje reprezentiraju iste kategorije, no ukoliko je potreban legenda, akutalni brojevi u gornjem-desnom kutu mogu poslužiti u svrhu legende gdje boja okvira označava jednu kategoriju. Tako je na ovom primjeru moguće vidjeti kako je Hrvatska u razdoblju od ožujka do lipnja imala “kvrgu” u broju zaraženih, a nedavni trend pokazuje da će se broj zaraženih samo rasti.

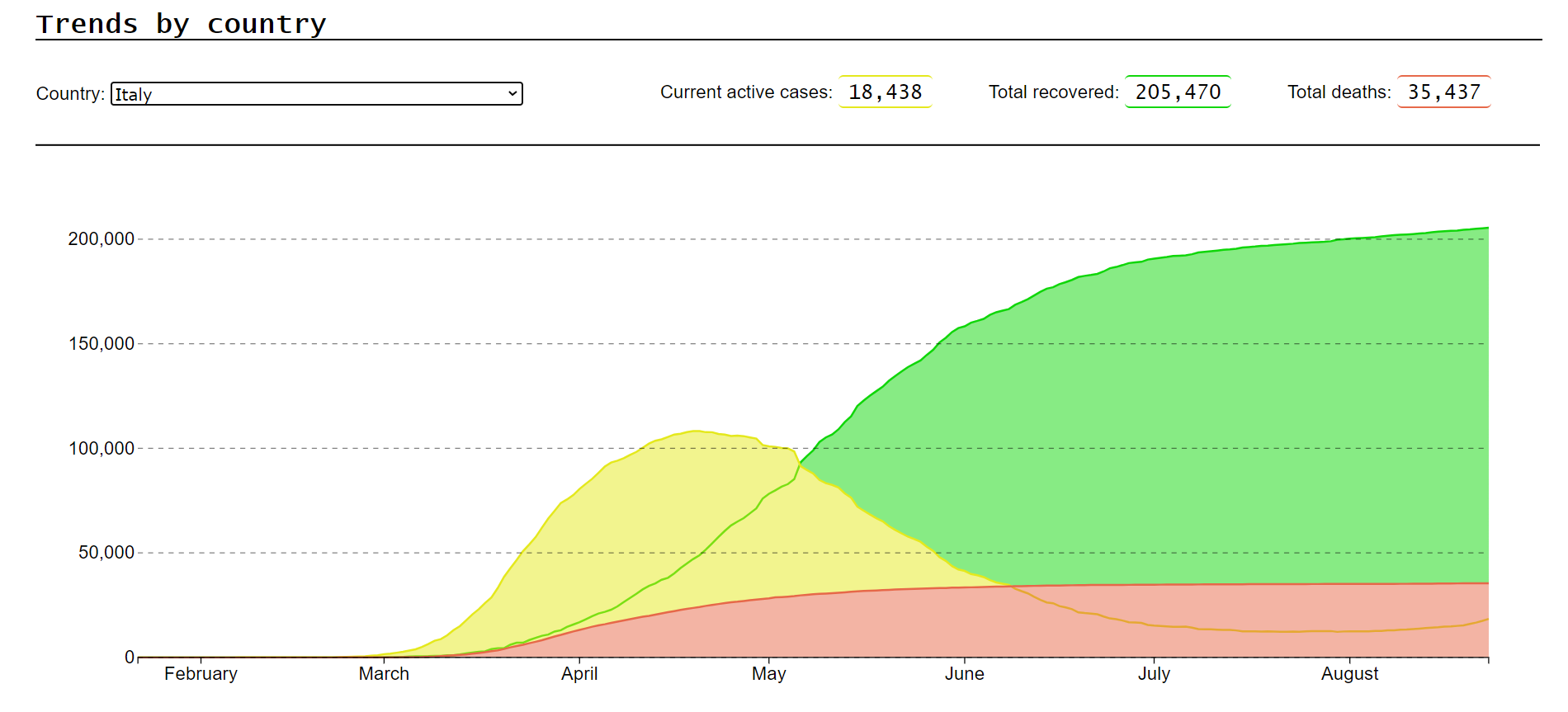


Slika 1.22. Primjer grafa krivulje za Hrvatsku.

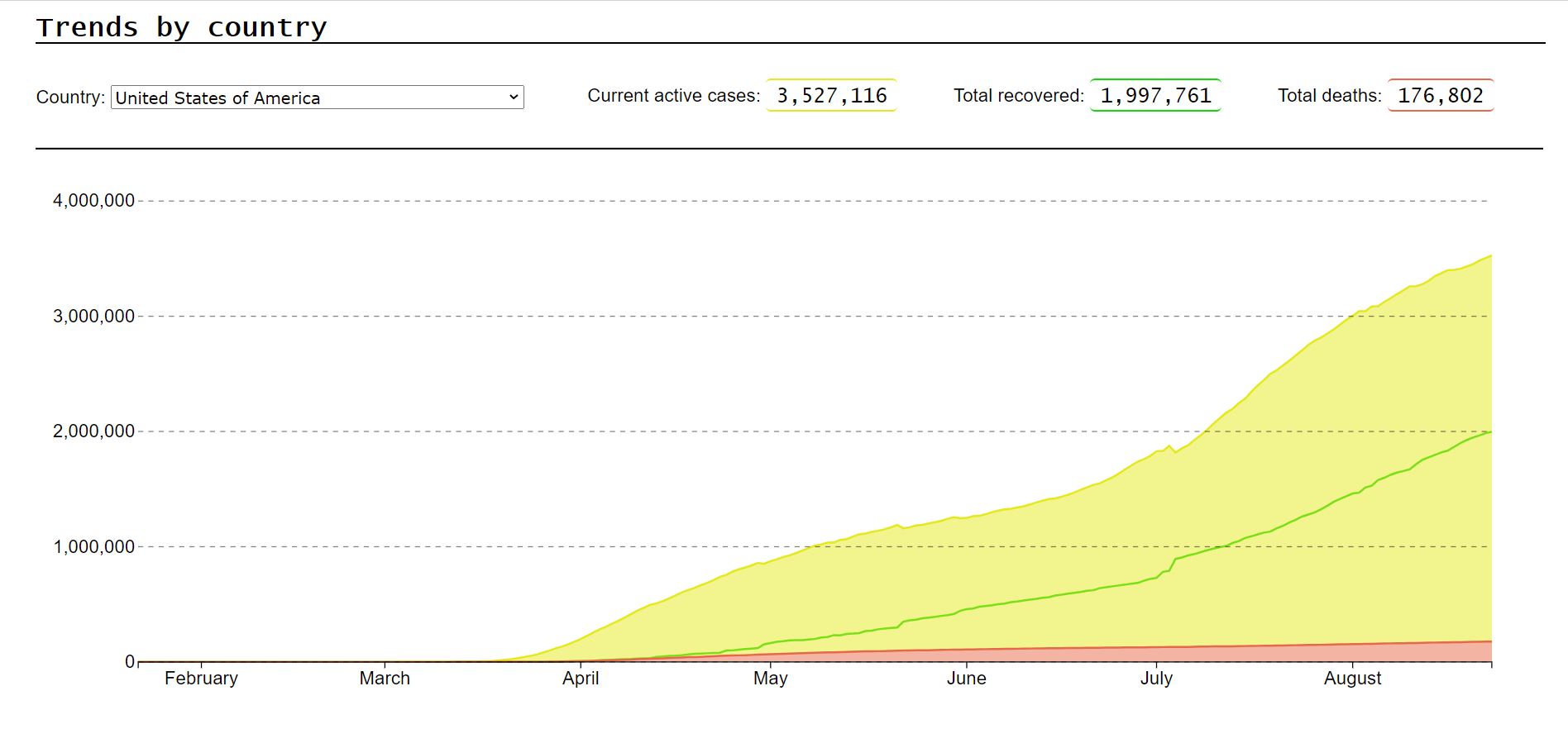
Drugi primjer je na slici 1.23. gdje je predložen graf krivulje za Italiju. Ovdje se mogu jasno vidjeti prioriteti područja opisanim u pogljavu 1.2.2.3 gdje područje za smrt ima najveći prioritet te se najprije iscrtava, dok područje za odravljene ima najmanji prioritet te se zbog toga zadnje iscrtava. Sa grafa se može iščitati kako je Italija imala istu “kvrgu” i razdoblju od ožujka do lipnja, ali i to da nakon te “kvrge” više nemaju porast novozaraženih.

Zadnji primjer je dan na slici 1.24. za SAD. Ovaj graf prikazuje jedan od ektremnih slučajeva kada broj zaraženih je uvijek veći od broja odravljenih.

Ukoliko se odabere država o kojoj ne postoje podaci, tada se prikazuje paragraf sa crvenim slovima u kojem piše: “No data for selected country.”



Slika 1.23. Primjer grafa krivulje za Italiju.



Slika 1.24. Primjer grafa krivulje za SAD.

# 2. KORIŠTENI IZVORI

Korišten je COVID-19 API sa stranie <https://covid19api.com/>. Upute za korištenje i primjeri korištenja se nalaze u njihovoj Postman dokumentaciji. Specifično su korišteni linkovi u programu:

* (GET) <https://api.covid19api.com/summary> - dohvaćanje globalne statistike
* (GET) <https://api.covid19api.com/countries> - dohvaćanje *slugova* i imena država
* (GET) <https://api.covid19api.com/total/country/(country_slug)> – dohvaćanje dnevnih podataka za pojedinu državu

Ukoliko je potrebno naći ove veze u programskom kodu, potrebno je tražiti komentare koje u sebi sadrže “SOURCE”.

## 2.1 GitHub

Projekt se nalazi na GitHub repozitoriju <https://github.com/Matt-Tyrantin/VP_project>.