SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Projektni zadatak iz predmeta

VIZUALIZACIJA PODATAKA

Vizualizacija popularnosti imena novorođene djece u SAD-u kroz vrijeme i prostor

Student: Maja Varšava, 1. DRC

Mentor: Barbara Bilonić

U Osijeku, 6.2025.

SADRŽAJ

1. KV1 - Definiranje projektnog zadatka

1.1. Projektni zadatak

Naziv zadatka: Vizualizacija popularnosti imena novorođene djece u SAD-u kroz prostor i vrijeme

Opis problema: U Sjedinjenim Američkim Državama svake se godine bilježe najpopularnija imena novorođene djece. Međutim, ti podaci često ostaju samo u tabličnom formatu i nisu intuitivno predstavljeni korisnicima koji žele uvid u promjene kroz godine, usporedbe između saveznih država, razlike među spolovima te opće trendove u imenima. Nedostatak vizualne interpretacije otežava analizu i uočavanje obrazaca.

Opis zadatka:

Cilj zadatka je izraditi interaktivnu vizualizaciju podataka o imenima djece u SAD-u koja omogućuje:

- Pregled najpopularnijih imena kroz godine na nacionalnoj i državnoj razini
- Prikaz razlika između saveznih država putem interaktivne karte
- Omjer dječaka i djevojčica po godinama i državama
- Prikaz popularnosti određenog imena kroz vrijeme i prostor

Korisnik će moći:

- Odabrati godinu i pregledati najpopularnija imena te godine
- Odabrati određeno ime i pratiti njegovu popularnost kroz godine i po državama
- Prebacivati se između nacionalne i državne razine pregleda
- Usporediti statistike za više država ili imena

Cilj projekta:

Omogućiti korisnicima intuitivno i interaktivno istraživanje podataka o imenima djece u SAD-u, kako bi mogli uočiti trendove, promjene kroz vrijeme te regionalne razlike u popularnosti imena. Vizualizacija će služiti i kao edukativni alat za prikaz demografskih i kulturnih obrazaca kroz podatke o imenima.

Poveznica na git repozitorij projekta:

majavarsava/VP-projekt: Projekt iz kolegija Vizualizacija podataka

1.2. Podatci

Za projekt se koriste javno dostupni podaci s američke službene stranice Social Security Administration (SSA) te GeoJSON karta SAD-a.

- National data (nacionalni skup): https://www.ssa.gov/oact/babynames/names.zip
  → sadrži podatke o imenima djece od 1880. do danas, s brojem upisa po godini i spolu.

- State-specific data (po državama):
  https://www.ssa.gov/oact/babynames/state/namesbystate.zip
  → sadrži podatke o imenima djece po saveznim državama i godinama.

- GeoJSON karta SAD-a: https://eric.clst.org/tech/usgeojson/
  → korišteno za vizualizaciju na karti SAD-a

1.3. Obrada podataka

Podaci su očišćeni i pretvoreni u odgovarajuće CSV/JSON format za rad s D3.js bibliotekom.

Uključeno je:

- Grupiranje po godinama, imenima, spolu i državama
- Filtriranje irelevantnih zapisa i agregacija brojeva
- Pretvaranje u strukture pogodne za višedimenzionalne grafove i interaktivne prikaze – izrađen novi skup podataka vezan samo za zbroj djece u svakoj državi za lakši i brži pristup tim podacima

1.4. Relevantne vrste prikaza za korištene podatke

- Interaktivna karta SAD-a – klikom na državu se prikazuje popularnost imena te države, top 5 imena u državi određene godine
- Pie chart – udio djevojčica i dječaka u odabranoj godini i/ili državi
- Bar chart – najpopularnija imena u određenoj godini i državi
- Višelinijski graf – trend popularnosti određenog/ih imena kroz godine – dobiva se klikom na određeno ime u drugim grafovima, kada je više imena odabrano omogućava usporedbu trendova
- Višestruki koordinirani prikazi – sinkronizacija interaktivnosti između karte, grafa i odabranog imena/godine
- Kombiniranje filtera - spol, godina, početno slovo ili početna slova, država te da se u stvarnom vremenu vide ažurirani grafovi
- Lenta vremena koju se pomiče kroz godine – prikazuje promjene na svim grafovima simultano

2. KV2 - Dizajn vizualizacije podataka.

2.1. Pitanja na koja vizualizacija daje odgovor

Vizualizacija će pružiti odgovore na sljedeća pitanja, formulirana tako da budu jasna, utemeljena na podacima i usklađena s dostupnim skupovima podataka iz SSA i GeoJSON podacima karte:

2.1.1. **Koja su imena bila najpopularnija u određenoj godini na nacionalnoj razini i po državama?**
Korisnici mogu odabrati godinu kako bi vidjeli najpopularnija imena na razini cijele države ili kliknuti na državu na interaktivnoj karti da bi vidjeli 5 najpopularnijih imena novorođenih djevojčica i dječaka u toj državi za odabranu godinu.

2.1.2. **Kako se mijenjala popularnost određenog imena tijekom vremena na nacionalnoj razini i među državama?**
Odabirom imena korisnici mogu pratiti trend popularnosti kroz godine putem linijskog grafa, s mogućnošću prikaza nacionalnih ili podataka po državama

2.1.3. **Kakva je spolna raspodjela imena u određenoj godini ili državi?**
Pie chart prikazat će udio dječaka i djevojčica za odabrane godine i/ili države.

2.1.4. **Koje su države imale najveću popularnost određenog imena u nekoj godini?**
Interaktivna karta koristit će boje za označavanje popularnosti odabranog imena po državama, omogućujući korisnicima da uoče regionalne razlike.

2.1.5. **Kako se trendovi imena razlikuju među državama za određenu godinu ili ime?**
Korisnici mogu uspoređivati više država ili imena pomoću koordiniranih stupčastih i linijskih grafikona kako bi uočili regionalne razlike.

2.1.6. **Koji su vremenski trendovi imena koja pretražujemo?**
Pretraživanje imena omogućuje korisnicima da prate trendove imena koja njih zanimaciju, prikazano putem linijskog grafa.

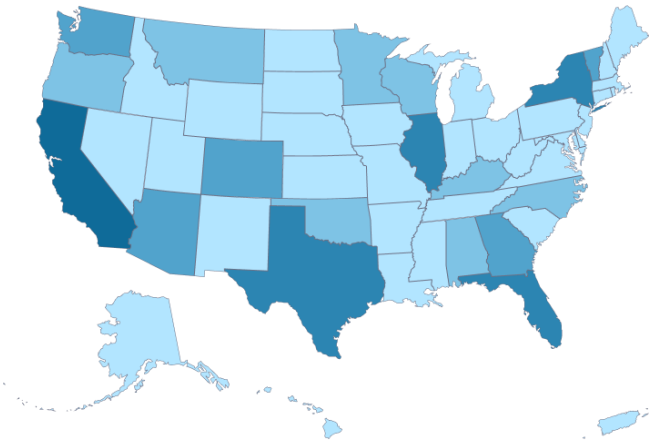
2.1.7. **Koja imena su dobivali i dječaci i djevojčice te koji je njihov trend?**
Odabirom radio gumba korisnik bira ili dječake ili djevojčice te nakon toga odabire imena koja postoje i kod dječaka i kod djevojčica, ime se dodaje na kartu te na linijski graf.

Ova pitanja osiguravaju da vizualizacija bude interaktivna, intuitivna i sposobna otkriti demografske i kulturne obrasce u imenima djece kroz vrijeme i prostor.

2.1. Skica vizualizacije podataka

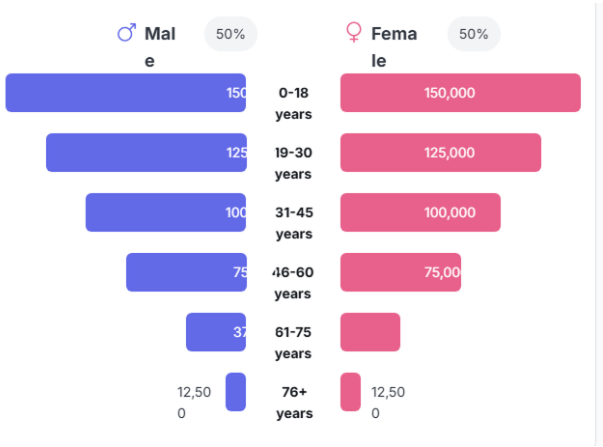
Vizualizacija će biti web-temeljena, interaktivna nadzorna ploča izrađena pomoću D3.js, s više koordiniranih prikaza. U nastavku je opis izgleda i komponenti te skice.

- **Interaktivna karta SAD-a:** prikazuje države obojene prema popularnosti odabranog imena ili zbroju djece za odabranu godinu. Klikom na državu prikazuju se grafikoni specifični za tu državu. Primjer karte je na slici ispod.

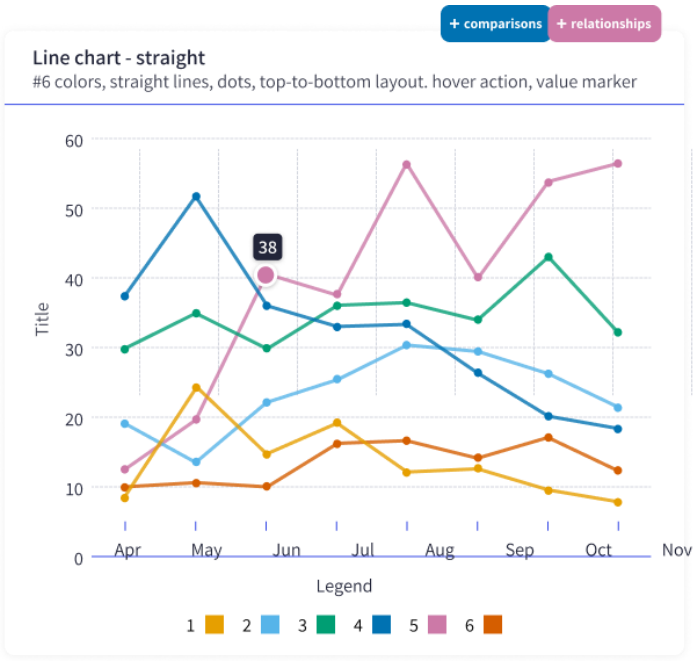


- **Pie Chart**: prikazuje spolni omjer (dječaci naspram djevojčica) za odabranu godinu i/ili državu zajedno uz tekst na kojem piše i ukupni broj djece. Primjer grafikona na slici ispod.

- **Dvosmjerni stupčasti dijagram**: prikazuje 5 najpopularnijih imena za odabranu godinu i/ili državu, s bojama koje označavaju spol. Primjer grafikona na slici ispod.



- **Linijski graf**: prikazuje trend popularnosti jednog ili više odabranih imena kroz vrijeme, s podrškom za više linija radi usporedbe. Primjer grafikona na slici ispod.



- **Lenta vremena**: Na dnu, pomični klizač pokazuje promjene kroz godine i ažurira sve grafikone istovremeno. Primjer lente na slici ispod.

- **Filteri**: sadrži padajuće izbornike i polja za unos za imena.

Nadzorna ploča osigurava da su svi prikazi međusobno sinkronizirani. Na primjer, odabirom imena u stupčanom grafikonu ažurira se linijski grafikon s njegovim trendom, dok pomicanje vremenskog klizača osvježava sve prikaze za odabranu godinu.

2.2. Postojeća rješenja i primjeri

Nekoliko postojećih projekata i primjera koda korisni su za ovu vizualizaciju:

1. **D3.js primjer karte SAD-a** (D3.js Geo Documentation)
    - **Upotreba:** Predložak za prikaz interaktivne karte SAD-a pomoću GeoJSON-a. Kod će se prilagoditi za prikaz popularnosti imena.
    - **Zašto:** Omogućuje parsiranje GeoJSON-a i isticanje država – ključno za komponentu karte.
2. **U.S. baby names by year in a spreadsheet: U.S. Baby Names by Year in a Spreadsheet | Row Zero**
    - Koristi iste podatke, ali se sve prikazuje u tablicama
3. **Vizualizator popularnih imena beba u SAD-u: US Baby Name Popularity Visualizer - Engaging Data**
    - Prikazuje jedan graf: broj rođenih po 1M rođenja po godinama

    - Ima mogućnost filtriranja po djevojčicama, dječacima ili oboje te unos određenog imena

4. **Chart popularnosti određenog imena kroz godine u SAD-u: Baby Name**
    - Unosom imena dobijemo linijski graf (broj djece po godinama) gdje zajedno s linijom slova imena prate visinu linije

5. **D3.js Pie Chart primjer** (D3.js Pie Chart)
    - **Upotreba**: Kod za prikaz pie chart-a koji pokazuje spolne omjere.
    - **Zašto:** Jednostavan i učinkovit za prikaz proporcionalnih podataka poput distribucije spola.

6. **D3.js Slider primjer** (D3.js Slider)
    - **Upotreba:** Omogućuje animacije temeljene na vremenu pomoću pomičnog klizača. Kod će se prilagoditi za vremensku crtu.
    - **Zašto:** Omogućuje glatke prijelaze između godina i sinkronizirane promjene svih grafikona.

Ovi primjeru daju ideju te smjernice kako napraviti ovaj projekt.

2.3. Prilagodba podataka

Priprema podataka

SSA skupovi podataka (nacionalni i po državama) te GeoJSON karta zahtijevaju prethodnu obradu kako bi podržali D3.js vizualizacije. Koraci uključuju:

1. **Čišćenje**: Ukloniti nepotpune zapise i standardizirati formate (kodove država).
2. **Agregacija**: Grupirati podatke po godini, imenu, spolu i državi za izračun ukupnih vrijednosti i rangiranja.
3. **Transformacija**: Pretvoriti podatke u JSON/CSV formate optimizirane za D3.js, s ugniježđenim strukturama za hijerarhijski pristup.

Format podataka

Obrađeni podaci pohranit će se u dvije glavne CSS i jednoj JSON datoteci:

- **Podaci o imenima po državama te ukupno:** Sadrže broj pojavljivanja imena po godinama, spolu i državama.
- **GeoJSON:** Sadrži granice američkih saveznih država s povezanim metrikama popularnosti imena.

Kod za obradu podataka

Ispod je Python skripta za obradu SSA podataka u potreban JSON format. Kod se također nalazi na Git repozitoriju za projekt.

Sljedeći kod obrađuje zip datoteku u kojoj se nalaze nacionalni podaci i podaci po državama. Podaci se spremaju kao .csv format. Kod se nalazi unutar datoteke extractingdata.py unutar repozitorija.

```python
import zipfile
import os
import pandas as pd

# Postavi putanje do ZIP datoteka
national_zip = "names.zip"
state_zip = "namesbystate.zip"

# Stvori direktorije za ekstrakciju
os.makedirs("national_names", exist_ok=True)
os.makedirs("state_names", exist_ok=True)

# Ekstraktaj datoteke
with zipfile.ZipFile(national_zip, 'r') as zip_ref:
    zip_ref.extractall("national_names")

with zipfile.ZipFile(state_zip, 'r') as zip_ref:
    zip_ref.extractall("state_names")

# ----------------------------
# Nacionalni podaci (names.zip)
# ----------------------------

national_data = []

for filename in sorted(os.listdir("national_names")):
    if filename.startswith("yob") and filename.endswith(".txt"):
        year = int(filename[3:7])
        df = pd.read_csv(f"national_names/{filename}", names=["Name", "Gender", "Count"])
        df["Year"] = year
        national_data.append(df)

national_df = pd.concat(national_data, ignore_index=True)

# Spremi u CSV
national_df.to_csv("national_names.csv", index=False)
print("✅ Spremljeno: national_names.csv")

# ----------------------------
# Podaci po državama (namesbystate.zip)
# ----------------------------

state_data = []

for filename in sorted(os.listdir("state_names")):
    if filename.endswith(".TXT"):
        df = pd.read_csv(f"state_names/{filename}", names=["State", "Gender", "Name", "Year", "Count"])
        state_data.append(df)

state_df = pd.concat(state_data, ignore_index=True)

# Spremi u CSV
state_df.to_csv("state_names.csv", index=False)
print("✅ Spremljeno: state_names.csv")
```

Sljedeći kod računa zbroj djece za svaku godinu na nacionalnoj razini te za svaku državu. Učitava prethodno napravljene .csv datoteke te prebrojava ukupan broj djece te sprema te podakte u zaseban .csv kako bi pristup bio brži i lakši. Kod je spremljen u datoteku counts.py.

```python
import csv
from collections import defaultdict


input_file_state = "vizualizacija/state_names_full.csv"
input_file_national = "vizualizacija/national_names.csv"
output_file = "vizualizacija/counts.csv"

# Helper to accumulate counts
def accumulate_state_counts(reader):
    counts = defaultdict(lambda: {'boys': 0, 'girls': 0})
    for row in reader:
        year = int(row['Year'])
        place = row['State']
        gender = row['Gender']
        count = int(row['Count'])
        key = (place, year)
        if gender == 'M':
            counts[key]['boys'] += count
        elif gender == 'F':
            counts[key]['girls'] += count
    return counts

def accumulate_national_counts(reader):
    counts = defaultdict(lambda: {'boys': 0, 'girls': 0})
    for row in reader:
        year = int(row['Year'])
        gender = row['Gender']
        count = int(row['Count'])
```

```
            key = ('National', year)
            if gender == 'M':
                counts[key]['boys'] += count
            elif gender == 'F':
                counts[key]['girls'] += count
        return counts

    # Read and accumulate state counts
    with open(input_file_state, newline='', encoding='utf-8') as f:
        reader = csv.DictReader(f)
        state_counts = accumulate_state_counts(reader)

    # Read and accumulate national counts
    with open(input_file_national, newline='', encoding='utf-8') as f:
        reader = csv.DictReader(f)
        national_counts = accumulate_national_counts(reader)

    # Write output
    with open(output_file, 'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['Place', 'Year', 'Count', 'Boys', 'Girls'])
        # Write national
        for (place, year), data in sorted(national_counts.items()):
            total = data['boys'] + data['girls']
            writer.writerow([place, year, total, data['boys'], data['girls']])
        # Write states
        for (place, year), data in sorted(state_counts.items()):
            total = data['boys'] + data['girls']
            writer.writerow([place, year, total, data['boys'], data['girls']])
```

Sljedeći kod čisti .csv datoteku nacionalnih podataka od godina koje nemamo unutar datoteke s podacima o državama kako bi nam informacije bile konzistentne. Kod se nalazi unutar datoteke 1800to1900del.py.

```
import csv

file_path = "vizualizacija/national_names.csv"

# Read all rows except those from 1880 to 1899
with open(file_path, "r", newline="", encoding="utf-8") as csvfile:
    reader = list(csv.reader(csvfile))
    header = reader[0]
    rows = [row for row in reader[1:] if not (1880 <= int(row[3]) <= 1899)]

# Overwrite the original file with filtered rows
with open(file_path, "w", newline="", encoding="utf-8") as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(header)
    writer.writerows(rows)
```

Sljedeći kod nam prepravlja kratice država unutar državnim podataka u puna imena država kako bi se poklapala s podacima iz datoteke s nacionalnih podacima. Kod se nalazi unutar datoteke fullstatenames.py.

```python
import csv

# Mapping of state abbreviations to full names
state_abbrev_to_full = {
    "AL": "Alabama", "AK": "Alaska", "AZ": "Arizona", "AR": "Arkansas",
    "CA": "California", "CO": "Colorado", "CT": "Connecticut", "DE": "Delaware",
    "FL": "Florida", "GA": "Georgia", "HI": "Hawaii", "ID": "Idaho",
    "IL": "Illinois", "IN": "Indiana", "IA": "Iowa", "KS": "Kansas",
    "KY": "Kentucky", "LA": "Louisiana", "ME": "Maine", "MD": "Maryland",
    "MA": "Massachusetts", "MI": "Michigan", "MN": "Minnesota", "MS": "Mississippi",
    "MO": "Missouri", "MT": "Montana", "NE": "Nebraska", "NV": "Nevada",
    "NH": "New Hampshire", "NJ": "New Jersey", "NM": "New Mexico", "NY": "New York",
    "NC": "North Carolina", "ND": "North Dakota", "OH": "Ohio", "OK": "Oklahoma",
    "OR": "Oregon", "PA": "Pennsylvania", "RI": "Rhode Island", "SC": "South Carolina",
    "SD": "South Dakota", "TN": "Tennessee", "TX": "Texas", "UT": "Utah",
    "VT": "Vermont", "VA": "Virginia", "WA": "Washington", "WV": "West Virginia",
    "WI": "Wisconsin", "WY": "Wyoming"
}

input_file = "vizualizacija\state_names.csv"
output_file = "state_names_full.csv"

with open(input_file, newline='', encoding='utf-8') as infile, \
     open(output_file, 'w', newline='', encoding='utf-8') as outfile:
    reader = csv.DictReader(infile)
    fieldnames = reader.fieldnames
    writer = csv.DictWriter(outfile, fieldnames=fieldnames)
    writer.writeheader()
    for row in reader:
        abbrev = row["State"]
        row["State"] = state_abbrev_to_full.get(abbrev, abbrev)
        writer.writerow(row)
```

Obrađeni podaci su testirani pomoću jednostavnoj D3.js stupčanog grafikona koji je uspješno prikazao 5 najpopularnijih imena za 2020. godinu, čime je potvrđena ispravnost strukture podataka.

Podaci nakon ovih prepravaka izgledaju ovako:

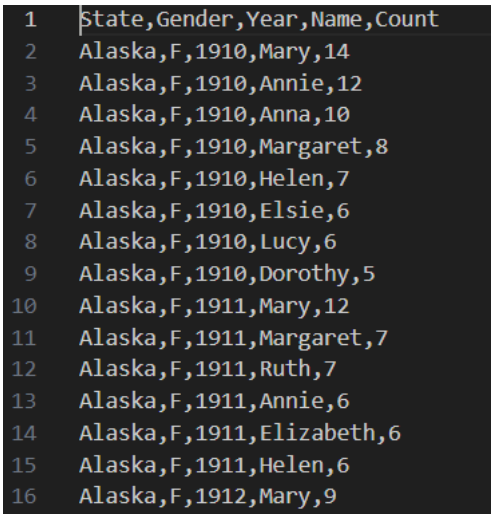
- counts.csv:

```csv
Place,Year,Count,Boys,Girls
National,1900,450258,150474,299784
National,1901,345812,106467,239345
National,1902,386733,122660,264073
National,1903,381200,119230,261970
National,1904,403485,128122,275363
National,1905,423922,132306,291616
National,1906,428451,133154,295297
National,1907,465390,146826,318564
National,1908,488649,154337,334312
National,1909,511224,163996,347228
National,1910,590711,194208,396503
National,1911,644278,225969,418309
National,1912,988062,429943,558119
National,1913,1137110,512565,624545
```

- national_names.csv:

```
Name,Gender,Count,Year
Mary,F,16705,1900
Helen,F,6342,1900
Anna,F,6114,1900
Margaret,F,5304,1900
Ruth,F,4765,1900
Elizabeth,F,4096,1900
Florence,F,3920,1900
Ethel,F,3896,1900
Marie,F,3856,1900
Lillian,F,3414,1900
Annie,F,3324,1900
Edna,F,3102,1900
Emma,F,3095,1900
```

- state_names_full.csv:



```
State,Gender,Year,Name,Count
Alaska,F,1910,Mary,14
Alaska,F,1910,Annie,12
Alaska,F,1910,Anna,10
Alaska,F,1910,Margaret,8
Alaska,F,1910,Helen,7
Alaska,F,1910,Elsie,6
Alaska,F,1910,Lucy,6
Alaska,F,1910,Dorothy,5
Alaska,F,1911,Mary,12
Alaska,F,1911,Margaret,7
Alaska,F,1911,Ruth,7
Alaska,F,1911,Annie,6
Alaska,F,1911,Elizabeth,6
Alaska,F,1911,Helen,6
Alaska,F,1912,Mary,9
```

2.4. Boje i podatci

Vizualizacija koristi usklađenu paletu boja radi jasnoće i pristupačnosti.

- **Spol**:
  - **Dječaci:** Plava boja - jasna i tradicionalno povezana s muškim spolom.
  - **Djevojčice:** Ljubičasta boja – kontrastna nijansa s plavom za jasnu razliku.
- **Intenzitet na karti** (popularnost – svih imena, imena djevojčica ili imena dječaka):
  - **Niska vrijednost:** Svijetlosmeđa – neutralna za male vrijednosti
  - **Visoka vrijednost:** Tamnije nijanse smeđe boje za visoke vrijednosti – naglašavaju visoku popularnost.
  - **Sekvencijalna ljestvica:** Prijelaz između svijetlih i tamnih za glatke gradijente.

- **Grafikoni**:
    - **Pozadina:** Bijela – čisto i pregledno.
    - **Osi / Oznake:** Crna – visok kontrast naspram pozadine radi bolje čitljivosti.
    - **Linije / Stupci (više imena):** kategorijske boje – različite za usporedbu više imena.

Mapiranje podataka

- **Karta:** Ispuna država prikazuje zbroj odabranog imena
- **Pie chart:** Plavi i ljubičasti dijelovi prikazuju udio dječaka i djevojčica.
- **Bar chart:** Stupci su obojeni prema spolu (plavi za dječake, ljubičasti za djevojčice).
- **Line chart:** Svaka linija koristi jedinstvenu boju za odabrano ime, uz legendu za jasnoću.
- **Vremenski klizač:** Neutralne boje, crno-bijelo za čisti izgled.

Ova shema boja osigurava vizualnu konzistentnost, pristupačnost za korisnike s daltonizmom i jasno mapiranje podataka na vizualne elemente.

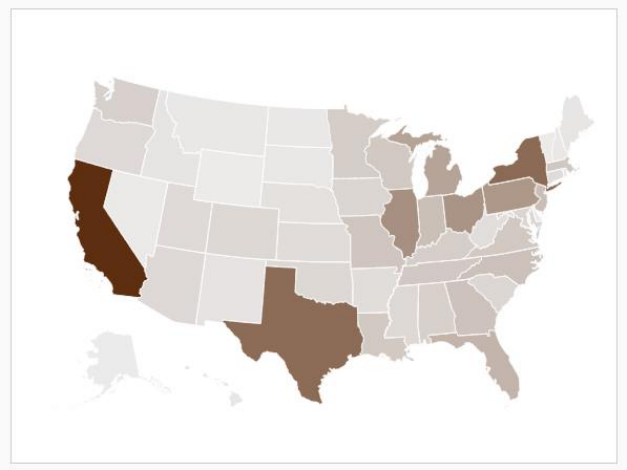
3. KV3 - Izrada prototipne vizualizacije podataka

U skladu s definiranim ciljevima projekta i dizajnom vizualizacije izrađena je prototipna web aplikacija koja omogućuje korisnicima interaktivno istraživanje popularnosti imena novorođene djece u SAD-u kroz prostor i vrijeme.

3.1. Osnovne funkcionalnosti i ponašanja

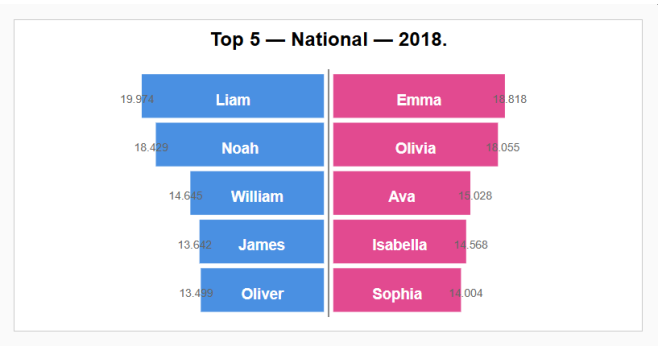
3.1.1. Interaktivna karta SAD-a

Na lijevoj strani pri vrhu se nalazi karta SAD-a izrađena pomoću GeoJSON podataka. Države su obojene prema popularnosti odabranog imena u odabranoj godini. Na karti prijelazom preko država vidimo obris države te tooltip na kojem vidimo ime te države. Države su klikabilne.



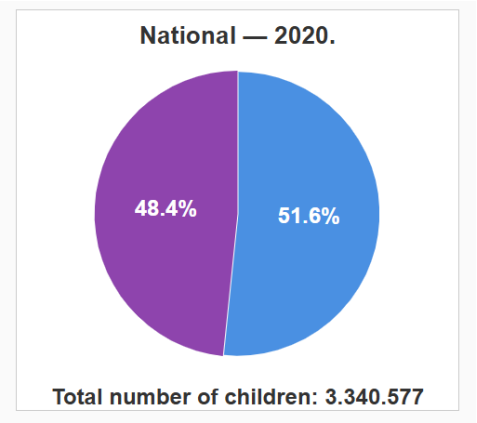
3.1.2. Dvosmjerni stupčasti dijagram – top 5 imena

Na desnoj strani pri vrhu se nalazi dvostrani stupčasti grafikon – s lijeve strane prikazuje 5 najpopularnijih imena dječaka, a s desne strane 5 najpopularnijih imena djevojčica za odabranu godinu i/ili državu. Iznad karte piše o kojoj je državi riječ i o kojoj godini je riječ. Svaki stupac dijagrama je klikabilan.



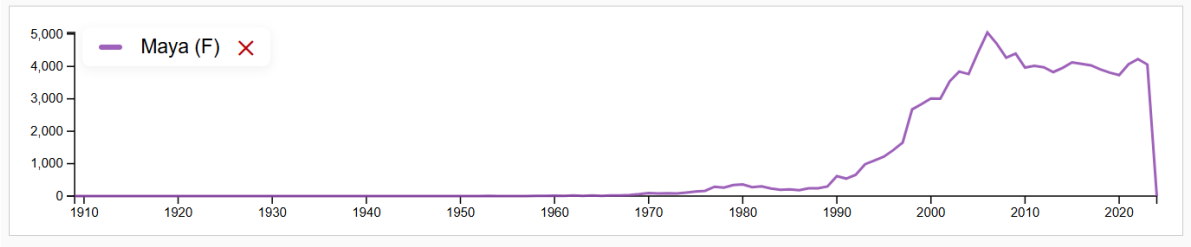
3.1.3. Pie chart – omjer dječaka i djevojčica

Između karte i dvosmjernog stupčastog dijagrama se nalazi pie chart koji prikazuje udio djevojčica i dječaka u ukupnom broju rođenih za trenutno odabranu godinu i/ili državu. Ispod grafikona prikazan je ukupan broj djece.



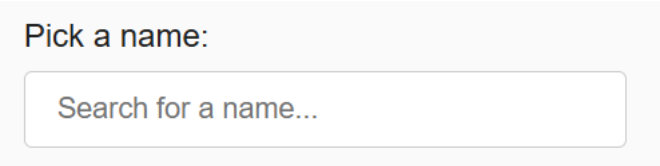
3.1.4. Linijski graf – trend imena kroz godine u različitim državama

Prikazuje se linijski graf koji pokazuje kako se popularnost jednog ili dva imena mijenjala kroz vrijeme i prostor (odabirom države). Legenda grafa se nalazi lijevo na grafu te postoji dugme za micanje imena s grafa.



3.1.5. Pretraživanje

Ispod karte se nalazi tekstualno polje za unos imena. Korisnik može upisati ime i odabrati ga kako bi se izvele druge funkcionalnosti na ostalim dijagramima (mapa i linijski graf).

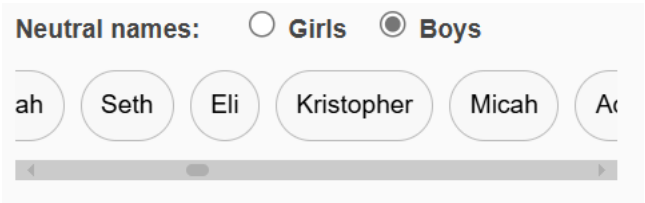


3.1.6. Klizač za promjenu godine

Na dnu stranice se nalazi klizač pomoću kojeg korisnik može odabrati godinu. Promjena godine ažurira druge prikaze te ažurira tekst iznad drške klizača koji prikazuje odabranu godinu.

3.1.7. Neutralna imena

Prikaz neutralnih imena, tj. imena koja su bila dodjeljivana i dječacima i djevojčicama. Uz prikaz imena imamo i odabir djevojčica ili dječaka koji će biti koristan kada se odabere jedno od neutralnih imena. Odabirom imena se pokreće prikaz na mapi i linijskom grafu.



3.2. Napredne funkcionalnosti i ponašanja:

Sinkronizacija svih prikaza. Svi grafovi se ažuriraju na temelju akcije (klik, unos, promjena godine klizačem) koja mijenja atribute koje ti grafovi koriste (godina, imena, država).

Kombiniranje filtera (ime+država+godina) moguće je precizno analizirati trendove kroz prostor i vrijeme.

3.2.1. Interaktivna karta SAD-a

Klikom na pojedinu saveznu državu, svi prikazi koji koriste državu kao atribut se ažuriraju s podacima specifičnima za tu državu. Ime te države je zapisano iznad grafova tokom te selekcije, te ponovnim klikom na tu državu se taj tekst mijenja u „National“ ili klikom na drugu državu se promjeni ime. Prikazi koji se mijenjaju promjenom države su: pie chart, linijski graf, dvosmjerni stupčasti dijagram.

3.2.2. Dvosmjerni stupčasti dijagram

Klikom na neko ime s dijagrama se mijenja boja mape ovisno o broju djece s tim imenom te se dodaje prikaz trenda tog imena u linijskom grafu.

3.2.3. Višestruko praćenje trendova na linijskom grafu

Odabirom imena unutar search inputa, odabirom nekog od neutralnih imena (uz odabir radio gumba za djevojčice ili dječake) ili odabirom imena preko stupčastog grafa dodajemo imena na linijski graf. Na graf možemo dodati do dva imena, sva se imena prikazuju za istu državu i za istu godinu. Klikom na dugme za uklanjanje mičemo ime s linijskog grafa, dok drugo ime ostaje. Dodavanjem trećeg imena na linijski graf se miče najranije stavljeno ime s grafa. Promjenom države ili godine se ažurira prikaz.

3.2.4. Klizač za promjenu godine

Promjenom godine preko klizača se automatski mijenjaju podaci na karti, dvosmjernom stupčastom grafu te pie chartu.

4. KV4 - Izrada konačne vizualizacije podataka

4.1. Implementacija osnovnih funkcionalnosti

4.1.1. Interaktivna karta SAD-a – prikaz boja po broju djece

```
// setting up the map
g.selectAll("path")
    .data(geojsonData.features, d => d.properties.NAME)
    .enter()
    .append("path")
    .attr("d", path)
    .attr("fill", d => {
            const stateName = d.properties.NAME;
            const row = pieCountsData.find(r => r.Place === stateName && +r.Year === +pieCurrentYear);
            const count = row ? +row.Count : 0;
            return count > 0 ? colorScale(count) : "#eee";
        })
    .on("mouseover", function(event, d) {
        let tooltipHtml = `<strong>${d.properties.NAME}</strong><br>`;
        if (selectedBarName && selectedBarGender) {
            const count =
                stateNameYearGenderCount[d.properties.NAME] &&
                stateNameYearGenderCount[d.properties.NAME][pieCurrentYear] &&
                stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender] &&
                stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender][selectedBarName]
                    ? stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender][selectedBarName]
                    : 0;
            tooltipHtml += `${selectedBarName} (${selectedBarGender}): ${count}`;
        } else {
            const row = pieCountsData.find(r => r.Place === d.properties.NAME && +r.Year === +pieCurrentYear);
            const count = row ? +row.Count : 0;
            tooltipHtml += `Total Children: ${count.toLocaleString()}`;
        }
        tooltip
            .style("display", "block")
            .html(tooltipHtml);
    })
    .on("mousemove", function(event) {
        tooltip
            .style("left", (event.pageX + 10) + "px")
            .style("top", (event.pageY - 20) + "px");
        d3.select(this).raise()
            .attr("stroke", "black")
            .attr("stroke-width", 3);
    })
```

```
// map setup
const svg = d3.select("svg.map");
const tooltip = d3.select("#tooltip");
const width = 900;
const height = 600;
const g = svg.append("g");

const projection = d3.geoAlbersUsa()
    .scale(950)
    .translate([width / 2, height / 2 ]);

const path = d3.geoPath().projection(projection);

const zoom = d3.zoom()
    .scaleExtent([1, 8])
    .on("zoom", (event) => {
    g.attr("transform", event.transform);
    });

svg.call(zoom);

const colorScale = d3.scaleSequential()
        .interpolator(d3.interpolateRgb("#eeeeee", "#5D2E0F"));
```

4.1.2. Dvosmjerni stupčasti dijagram s top 5 imena – prikaz

```
function updateBarchart() {
    document.getElementById("barchart-title").textContent = `Top 5 – ${pieCurrentState} – ${pieCurrentYear}.`;

    const svg = d3.select("#barchart");
    const margin = {top: 40, right: 50, bottom: 30, left: 50};
    const width = 1200 - margin.left - margin.right;
    const height = 600 - margin.top - margin.bottom;
    const centerX = width / 2;

    let chartGroup = svg.select("g.barchart-group");
    if (chartGroup.empty()) {
        chartGroup = svg.append("g")
            .attr("class", "barchart-group")
            .attr("transform", `translate(${margin.left},${margin.top})`);
    }

    // get data for current year and state/national
    let currentData;
    if (pieCurrentState === 'National') {
        currentData = nationalData.filter(d => d.year === pieCurrentYear);
    } else {
        currentData = stateData.filter(d => d.year === pieCurrentYear && d.state === pieCurrentState);
    }

    // no data text
    if (!currentData || currentData.length === 0) {
        d3.select("#barchart-no-data").style("display", "block");
        chartGroup.selectAll("*").remove();
        return;
    } else {
        d3.select("#barchart-no-data").style("display", "none");
    }

    // top 5 names for each gender
    const maleNames = currentData.filter(d => d.gender === 'M').sort((a, b) => b.count - a.count).slice(0, 5);
    const femaleNames = currentData.filter(d => d.gender === 'F').sort((a, b) => b.count - a.count).slice(0, 5);

    // scales
    const maxCount = Math.max(
        d3.max(maleNames, d => d.count) || 0,
        d3.max(femaleNames, d => d.count) || 0
    );
    const xScale = d3.scaleLinear()
        .domain([0, maxCount])
        .range([0, centerX - 10]);
    const yScale = d3.scaleBand()
        .domain([0, 1, 2, 3, 4])
        .range([0, height])
        .padding(0.1);
```

```
// center line
let centerLine = chartGroup.selectAll(".center-line").data([null]);
centerLine = centerLine.join(
    enter => enter.append("line")
        .attr("class", "center-line")
        .attr("x1", centerX).attr("y1", 0)
        .attr("x2", centerX).attr("y2", height)
        .attr("stroke", "#333")
        .attr("stroke-width", 2)
);

// --- MALE BARS ---
const maleGroups = chartGroup.selectAll("g.male-bar-group")
    .data(maleNames, d => d.name);

// ENTER
const maleGroupsEnter = maleGroups.enter()
    .append("g")
    .attr("class", "bar-group male-bar-group");

// rects
maleGroupsEnter.append("rect")
    .attr("class", "bar-male")
    .attr("x", centerX - 10)
    .attr("y", (d, i) => yScale(i))
    .attr("width", 0)
    .attr("height", yScale.bandwidth())
    .on("mouseover", function(event, d) {
        tooltip
            .style("display", "block")
            .html(`<strong>${d.name}</strong><br>${d.count.toLocaleString()}`);
    })
    .on("mousemove", function(event) {
        tooltip
            .style("left", (event.pageX + 10) + "px")
            .style("top", (event.pageY - 10) + "px");
    })
    .on("mouseout", function() {
        tooltip.style("display", "none");
    })
```

```
// name labels
maleGroupsEnter.append("text")
    .attr("class", "name-label")
    .attr("x", centerX - 10)
    .attr("y", (d, i) => yScale(i) + yScale.bandwidth() / 2)
    .attr("dy", "0.35em")
    .text(d => d.name)
    .transition()
    .duration(700)
    .attr("x", d => centerX - 10 - xScale(d.count) / 2);

// UPDATE
maleGroups.select("rect")
    .transition()
    .duration(700)
    .attr("x", (d, i) => centerX - 10 - xScale(d.count))
    .attr("y", (d, i) => yScale(i))
    .attr("width", d => xScale(d.count))
    .attr("height", yScale.bandwidth());

maleGroups.select("text")
    .transition()
    .duration(700)
    .attr("x", d => centerX - 10 - xScale(d.count) / 2)
    .attr("y", (d, i) => yScale(i) + yScale.bandwidth() / 2)
    .text(d => d.name);

// EXIT
maleGroups.exit().remove();

// --- FEMALE BARS ---
const femaleGroups = chartGroup.selectAll("g.female-bar-group")
    .data(femaleNames, d => d.name);

// ENTER
const femaleGroupsEnter = femaleGroups.enter()
    .append("g")
    .attr("class", "bar-group female-bar-group");

femaleGroupsEnter.append("rect")
    .attr("class", "bar-female")
    .attr("x", centerX + 10)
    .attr("y", (d, i) => yScale(i))
    .attr("width", 0)
    .attr("height", yScale.bandwidth())
    .on("mouseover", function(event, d) {
        tooltip
            .style("display", "block")
            .html(`<strong>${d.name}</strong><br>${d.count.toLocaleString()}`);
    })
```

```
femaleGroupsEnter.append("text")
    .attr("class", "name-label")
    .attr("x", centerX + 10)
    .attr("y", (d, i) => yScale(i) + yScale.bandwidth() / 2)
    .attr("dy", "0.35em")
    .text(d => d.name)
    .transition()
    .duration(700)
    .attr("x", d => centerX + 10 + xScale(d.count) / 2);

// UPDATE
femaleGroups.select("rect")
    .transition()
    .duration(700)
    .attr("x", centerX + 10)
    .attr("y", (d, i) => yScale(i))
    .attr("width", d => xScale(d.count))
    .attr("height", yScale.bandwidth());

femaleGroups.select("text")
    .transition()
    .duration(700)
    .attr("x", d => centerX + 10 + xScale(d.count) / 2)
    .attr("y", (d, i) => yScale(i) + yScale.bandwidth() / 2)
    .text(d => d.name);

// EXIT
femaleGroups.exit().remove();

// --- COUNT LABELS ---
// remove and redraw count labels
chartGroup.selectAll(".male-count").remove();
chartGroup.selectAll(".female-count").remove();
```

```
    chartGroup.selectAll(".male-count")
        .data(maleNames)
        .enter()
        .append("text")
        .attr("class", "count-label male-count")
        .attr("x", d => centerX - 20 - xScale(d.count))
        .attr("y", (d, i) => yScale(i) + yScale.bandwidth() / 2)
        .attr("dy", "0.35em")
        .text(d => d.count.toLocaleString());

    chartGroup.selectAll(".female-count")
        .data(femaleNames)
        .enter()
        .append("text")
        .attr("class", "count-label female-count")
        .attr("x", d => centerX + 20 + xScale(d.count))
        .attr("y", (d, i) => yScale(i) + yScale.bandwidth() / 2)
        .attr("dy", "0.35em")
        .text(d => d.count.toLocaleString());
}
```

4.1.3. Pie chart – prikaz omjera dječaka i djevojčica

```
<div id="piechart-container">
    <div id="piechart-title" class="chart-title"></div>
    <svg id="piechart" viewBox="0 0 200 200" preserveAspectRatio="xMidYMid meet"></svg>
    <div id="pie-total-text" class="total-text"></div>
    <div id="pie-no-data" style="display:none; color:#b00; font-weight: bold; text-align: center; position:absolute">
        Sorry, we have no data for this year and state.
    </div>
</div>
```

```
// pie chart
const pieSvg = d3.select("#piechart");
const pieWidth = 200;
const pieHeight = 200;
const pieRadius = Math.min(pieWidth, pieHeight) / 2 - 10;

pieSvg.selectAll("g").remove(); // clear previous content for hot reaload

const pieGroup = pieSvg
    .append("g")
    .attr("transform", `translate(${pieWidth / 2}, ${pieHeight / 2})`);

const pie = d3.pie()
    .value(d => d.count)
    .sort(null);

const arc = d3.arc()
    .innerRadius(0)
    .outerRadius(pieRadius);

const pieTooltip = d3.select("#tooltip");
const pieTotalText = d3.select("#pie-total-text");
```

```
function updatePieChart() {
    document.getElementById("piechart-title").textContent =  `${pieCurrentState} – ${pieCurrentYear}.`;
    document.getElementById("pie-no-data").style.display = "none";
    document.getElementById("pie-total-text").style.display = "block";

    pieGroup.selectAll(".pie-slice").remove();
    pieGroup.selectAll(".percentage-label").remove();

    const row = pieCountsData.find(d =>
        d.Place === pieCurrentState && d.Year === pieCurrentYear
    ) || pieCountsData.find(d =>
        d.Place === pieCurrentState && +d.Year === +pieCurrentYear
    ); // fallback for type mismatch

    let maleTotal = 0, femaleTotal = 0, grandTotal = 0;
    if (row) {
        maleTotal = +row.Boys;
        femaleTotal = +row.Girls;
        grandTotal = +row.Count;
    }

    // update total no of children text
    pieTotalText.text(`Total number of children: ${grandTotal.toLocaleString()}`);

    // prepare data for pie chart
    const pieData = [
        { gender: 'M', label: 'Boys', count: maleTotal, color: '#4a90e2' },
        { gender: 'F', label: 'Girls', count: femaleTotal, color: '#8d44add6' }
    ].filter(d => d.count > 0);

    // if no data - text for no data
    if (pieData.length === 0 || grandTotal === 0) {
        document.getElementById("pie-no-data").style.display = "block";
        document.getElementById("pie-total-text").style.display = "none";
        pieSvg.style("opacity", 0.2);
        return;
    } else {
        document.getElementById("pie-no-data").style.display = "none";
        document.getElementById("pie-total-text").style.display = "block";
        pieSvg.style("opacity", 1);
    }

    // create pie slices
    const slices = pieGroup.selectAll(".pie-slice")
    .data(pie(pieData))
    .enter()
    .append("g")
    .attr("class", "pie-slice");
```

```
// create pie slices
const slices = pieGroup.selectAll(".pie-slice")
.data(pie(pieData))
.enter()
.append("g")
.attr("class", "pie-slice");

// add path elements
slices.append("path")
    .attr("d", arc)
    .attr("class", d => d.data.gender === 'M' ? 'slice-male' : 'slice-female')
    .on("mouseover", function(event, d) {
        pieTooltip
            .style("display", "block")
            .html(`<strong>${d.data.label}:</strong> ${d.data.count.toLocaleString()}`);
        d3.select(this.parentNode)
            .raise();
        d3.select(this)
            .attr("stroke", "black")
            .attr("stroke-width", 3);
    })
    .on("mousemove", function(event) {
        pieTooltip
        .style("left", (event.pageX + 10) + "px")
        .style("top", (event.pageY - 10) + "px");
    })
    .on("mouseout", function() {
        pieTooltip.style("display", "none");
        d3.select(this)
            .attr("stroke", null)
            .attr("stroke-width", null);
    });

// add percentage labels
slices.append("text")
.attr("class", "percentage-label")
.attr("transform", d => `translate(${arc.centroid(d)})`)
.text(d => {
    const percentage = ((d.data.count / grandTotal) * 100).toFixed(1);
    return `${percentage}%`;
})
.style("display", d => {
    const percentage = (d.data.count / grandTotal) * 100;
    return percentage > 5 ? "block" : "none";
});
```

4.1.4. Linijski graf – prikaz linijskog grafa i dodavanja imena

```
function updateLineChart() {
    const svg = d3.select("#linechart");
    const legendDiv = document.getElementById("linechart-legend");
    const tooltip = d3.select("#linechart-tooltip");
    svg.selectAll("*").remove();
    legendDiv.innerHTML = '';

    const margin = {top: 20, right: 20, bottom: 30, left: 50};
    const width = svg.node().clientWidth;
    const height = svg.node().clientHeight;
    const chartW = width - margin.left - margin.right;
    const chartH = Math.max(1, height - margin.top - margin.bottom);

    const years = d3.range(1909, 2025);
    const x = d3.scaleLinear().domain([1909, 2024]).range([0, chartW]);
    const y = d3.scaleLinear().range([chartH, 0]);

    // prepare data for each selected name
    let allCounts = [];
    const lineData = selectedLineNames.map(sel => {
        let data;
        if (pieCurrentState === "National") {
            data = years.map(year => {
                let total = 0;
                for (const state in stateNameYearGenderCount) {
                    if (
                        stateNameYearGenderCount[state][year] &&
                        stateNameYearGenderCount[state][year][sel.gender] &&
                        stateNameYearGenderCount[state][year][sel.gender][sel.name]
                    ) {
                        total += stateNameYearGenderCount[state][year][sel.gender][sel.name];
                    }
                }
                allCounts.push(total);
                return { year, count: total };
            });
        } else {
            data = years.map(year => {
            let count = 0;
            if (
                stateNameYearGenderCount[pieCurrentState] &&
                stateNameYearGenderCount[pieCurrentState][year] &&
                stateNameYearGenderCount[pieCurrentState][year][sel.gender] &&
                stateNameYearGenderCount[pieCurrentState][year][sel.gender][sel.name]
            ) {
                count = stateNameYearGenderCount[pieCurrentState][year][sel.gender][sel.name];
            }
            allCounts.push(count);
            return { year, count };
        });
        }
```

```
    });
    }
    return { ...sel, data };
});

// set y domain
const maxY = Math.max(1, d3.max(allCounts));
y.domain([0, maxY]);

// draw axes
const g = svg.append("g").attr("transform", `translate(${margin.left},${margin.top})`);
g.append("g")
    .attr("transform", `translate(0,${chartH})`)
    .call(d3.axisBottom(x).tickFormat(d3.format("d")).ticks(10))
    .selectAll("text")
    .style("font-size", "10px");
g.append("g")
    .call(d3.axisLeft(y).ticks(5))
    .selectAll("text")
    .style("font-size", "10px");

// get SVG position for tooltip positioning
const svgRect = svg.node().getBoundingClientRect();
// draw lines
const line = d3.line()
    .x(d => x(d.year))
    .y(d => y(d.count));

lineData.forEach((d, i) => {
    // draw the line
    const path = g.append("path")
        .datum(d.data)
        .attr("fill", "none")
        .attr("stroke", d.color)
        .attr("stroke-width", 2)
        .attr("class", "linechart-line")
        .attr("d", line);

    // animate the line drawing
    const totalLength = path.node().getTotalLength();
    path
        .attr("stroke-dasharray", totalLength + " " + totalLength)
        .attr("stroke-dashoffset", totalLength)
        .transition()
        .duration(1200)
        .ease(d3.easeLinear)
        .attr("stroke-dashoffset", 0);
```

```
// draw legend
lineData.forEach((d, i) => {
    const legendItem = document.createElement("div");
    legendItem.style.display = "flex";
    legendItem.style.alignItems = "center";
    legendItem.style.gap = "0.5rem";
    legendItem.innerHTML = `<span style="display:inline-block;width:18px;height:4px;background:${d.color};margin-right:6px;border-radius:2px;"></span>
        <span>${d.name} (${d.gender})</span>
        <span class="legend-x" style="color:#b00;cursor:pointer;font-weight:bold;margin-left:6px;">&#10005;</span>`;
```

4.1.5. Pretraživanje – prikaz prijedloga imena

```
<div id="searchbar-container">
    <input type="text" id="name-search" placeholder="Search for a name..." autocomplete="off"/>
    <div id="search-suggestions"></div>
```

```
// search bar functionality
const searchInput = document.getElementById('name-search');
const suggestionsDiv = document.getElementById('search-suggestions');

searchInput.addEventListener('input', function() {
    const value = this.value.trim().toLowerCase();
    if (!value) {
    suggestionsDiv.style.display = 'none';
    return;
    }
    const matches = allNames
        .filter(d => d.name.toLowerCase().startsWith(value));

    if (matches.length === 0) {
        suggestionsDiv.style.display = 'none';
        return;
    }
    suggestionsDiv.innerHTML = '';
    matches.forEach(d => {
        const div = document.createElement('div');
        div.className = 'suggestion-item';
        div.textContent = `${d.name} (${d.gender})`;
```

4.1.6. Klizač za promjenu vremena

```
<div id="slider-container">
  <svg class="slider-svg"></svg>
</div>
```

```
// slider
const sliderSvg = d3.select(".slider-svg");
const sliderMargin = {left: 15, right: 10};
const sliderWidth = +sliderSvg.node().clientWidth - sliderMargin.left - sliderMargin.right - 20;
const sliderHeight = 140;

const x = d3.scaleLinear()
    .domain([1900, 2024])
    .range([0, sliderWidth])
    .clamp(true);

const slider = sliderSvg.append("g")
    .attr("transform", `translate(${sliderMargin.left},${sliderHeight / 2 + 10})`);

slider.append("line")
    .attr("class", "track")
    .attr("x1", x.range()[0])
    .attr("x2", x.range()[1])
    .clone(true).attr("class", "track-inset")
    .clone(true).attr("class", "track-overlay")
```

```
slider.insert("g", ".track-overlay")
    .attr("class", "ticks")
    .attr("transform", "translate(0, 18)")
    .selectAll("text")
    .data(x.ticks((2024 - 1900) / 10))
    .join("text")
    .attr("x", d => x(d))
    .attr("text-anchor", "middle")
    .text(d => d);

const handle = slider.insert("circle", ".track-overlay")
    .attr("class", "handle")
    .attr("r", 9)
    .attr("cx", x(1900));

    const handleLabel = slider.append("text")
    .attr("class", "year-label")
    .attr("text-anchor", "middle")
    .attr("x", x(1900))
    .attr("y", -18)
    .text(1900);
```

4.2. Implementacija osnovnog ponašanja

4.2.1. Interaktivna karta SAD-a – prikaz boja po broju djece

```
    .on("mouseover", function(event, d) {
        let tooltipHtml = `<strong>${d.properties.NAME}</strong><br>`;
        if (selectedBarName && selectedBarGender) {
            const count =
                stateNameYearGenderCount[d.properties.NAME] &&
                stateNameYearGenderCount[d.properties.NAME][pieCurrentYear] &&
                stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender] &&
                stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender][selectedBarName]
                    ? stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender][selectedBarName]
                    : 0;
            tooltipHtml += `${selectedBarName} (${selectedBarGender}): ${count}`;
        } else {
            const row = pieCountsData.find(r => r.Place === d.properties.NAME && +r.Year === +pieCurrentYear);
            const count = row ? +row.Count : 0;
            tooltipHtml += `Total Children: ${count.toLocaleString()}`;
        }
        tooltip
            .style("display", "block")
            .html(tooltipHtml);
    })
    .on("mousemove", function(event) {
        tooltip
            .style("left", (event.pageX + 10) + "px")
            .style("top", (event.pageY - 20) + "px");
        d3.select(this).raise()
            .attr("stroke", "black")
            .attr("stroke-width", 3);
    })
```

```
.on("mouseout", function() {
    tooltip.style("display", "none");
    d3.select(this).attr("stroke-width", 1);
})
.on("click", function(event, d) {
    pieCurrentState = d.properties.NAME;
    // updating other charts based on selected state
    updatePieChart();
    updateBarchart();
    updateLineChart();
});
```

4.2.2. Dvosmjerni stupčasti dijagram s top 5 imena – prikaz

```
    .on("mouseover", function(event, d) {
        tooltip
            .style("display", "block")
            .html(`<strong>${d.name}</strong><br>${d.count.toLocaleString()}`);
    })
    .on("mousemove", function(event) {
        tooltip
            .style("left", (event.pageX + 10) + "px")
            .style("top", (event.pageY - 10) + "px");
    })
    .on("mouseout", function() {
        tooltip.style("display", "none");
    })
    .on("click", function(event, d) {
        selectedBarName = d.name;
        selectedBarGender = d.gender;
        selectName(d.name, d.gender);
    })
    .transition()
    .duration(700)
    .attr("x", d => centerX - 10 - xScale(d.count))
    .attr("width", d => xScale(d.count));
```

4.2.3. Pie chart – prikaz omjera dječaka i djevojčica

```
    .on("mouseover", function(event, d) {
        pieTooltip
            .style("display", "block")
            .html(`<strong>${d.data.label}:</strong> ${d.data.count.toLocaleString()}`);
        d3.select(this.parentNode)
            .raise();
        d3.select(this)
            .attr("stroke", "black")
            .attr("stroke-width", 3);
    })
    .on("mousemove", function(event) {
        pieTooltip
        .style("left", (event.pageX + 10) + "px")
        .style("top", (event.pageY - 10) + "px");
    })
    .on("mouseout", function() {
        pieTooltip.style("display", "none");
        d3.select(this)
            .attr("stroke", null)
            .attr("stroke-width", null);
    });
```

4.2.4. Linijski graf – prikaz linijskog grafa i dodavanja imena

```
        <span class="legend-x" style="color:#b00;cursor:pointer;font-weight:bold;margin-left:6px;">&#10005;</span>`;
    legendItem.querySelector(".legend-x").onclick = () => {
        selectedLineNames = selectedLineNames.filter(sel => !(sel.name === d.name && sel.gender === d.gender));
        updateLineChart();
    };
    legendDiv.appendChild(legendItem);
});
}
```

4.2.5. Pretraživanje – prikaz prijedloga imena

```
            div.addEventListener('click', () => {
                selectName(d.name, d.gender);
                suggestionsDiv.style.display = 'none';
                searchInput.value = d.name;
            });
            suggestionsDiv.appendChild(div);
        });
        suggestionsDiv.style.display = 'block';
    });

    // hide suggestions on blur
    searchInput.addEventListener('blur', function() {
        setTimeout(() => { suggestionsDiv.style.display = 'none'; }, 150);
    });
```

4.2.6. Klizač za promjenu vremena

```
    .call(d3.drag()
        .on("start drag", (event) => {
        const year = Math.round(x.invert(event.x));
        updateYear(year);
        }));
```

4.3. Implementacija naprednih funkcionalnosti

4.3.1. Interaktivna karta SAD-a – klik na državu, van država te bojanje ovisno o imenu

```
.on("click", function(event, d) {
    pieCurrentState = d.properties.NAME;

    updatePieChart();
    updateBarchart();
});
```

```
// updating map coloring based on state/national and year and if name is selected
function updateMapColors() {
    if(!geojsonData) return; // ensure geojson data is loaded

    // update color scale domain based on current data
    let counts;
    if (selectedBarName && selectedBarGender) {
        counts = geojsonData.features.map(d => {
            const count =
                stateNameYearGenderCount[d.properties.NAME] &&
                stateNameYearGenderCount[d.properties.NAME][pieCurrentYear] &&
                stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender] &&
                stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender][selectedBarName]
                    ? stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender][selectedBarName]
                    : 0;
            return count;
        });
        const maxCount = Math.max(...counts);
        colorScale.domain([0, maxCount > 0 ? maxCount: 1]);
    } else {
        // Use total children counts
        counts = geojsonData.features.map(d => {
            const row = pieCountsData.find(r => r.Place === d.properties.NAME && +r.Year === +pieCurrentYear);
            return row ? +row.Count : 0;
        });
        //colorScale.domain(d3.extent(counts));
        const extent = d3.extent(counts);
        colorScale.domain(extent[0] === extent[1] ? [0, extent[1] || 1] : extent);
    }

    g.selectAll("path")
        .attr("fill", d => {
            if (selectedBarName && selectedBarGender) {
                const count =
                    stateNameYearGenderCount[d.properties.NAME] &&
                    stateNameYearGenderCount[d.properties.NAME][pieCurrentYear] &&
                    stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender] &&
                    stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender][selectedBarName]
                        ? stateNameYearGenderCount[d.properties.NAME][pieCurrentYear][selectedBarGender][selectedBarName]
                        : 0;
                // Use a color scale for the count (adjust as needed)
                return count > 0 ? colorScale(count) : "#eee";
            } else {
                const row = pieCountsData.find(r => r.Place === d.properties.NAME && +r.Year === +pieCurrentYear);
                const count = row ? +row.Count : 0;
                return count > 0 ? colorScale(count) : "#eee";
            }
        });
}
```

```
// clicking outside of the states resets other chart (and the map) to national
svg.on("click", function(event) {
    if (event.target.tagName !== "path") {
        pieCurrentState = "National";
        updatePieChart();
        updateBarchart();
        updateMapColors();
        updateLineChart();
    }
});
```

4.3.2. Dvosmjerni stupčasti dijagram – bojanje mape i prikaz imena odabirom imena

```
    })
    .on("click", function(event, d) {
        selectedBarName = d.name;
        selectedBarGender = d.gender;
        selectName(d.name, d.gender);
    })
    .transition()
    .duration(700)
    .attr("width", d => xScale(d.count));
```

4.3.3. Višestruko praćenje trendova na linijskom grafu

```
    <span class="legend-x" style="color:#b00;cursor:pointer;font-weight:bold;margin-left:6px;">&#10005;</span>`;
legendItem.querySelector(".legend-x").onclick = () => {
    selectedLineNames = selectedLineNames.filter(sel => !(sel.name === d.name && sel.gender === d.gender));
    updateLineChart();
};
legendDiv.appendChild(legendItem);
```

```
selectedLineNames = selectedLineNames.filter(d => (d.name + "|" + d.gender) !== key);
selectedLineNames.push({ name, gender });
if (selectedLineNames.length > 2) selectedLineNames = selectedLineNames.slice(-2); // keep only last 2
selectedLineNames.forEach(d => {
    d.color = d.gender === 'F' ? '#8d44add6' : '#4a90e2';
});
```

4.3.4. Neutralna imena – sortiranje gledamo li dječake ili djevojčice nakon klika

```
// radio buttons for neutral names
document.querySelectorAll('input[name="neutral-gender"]').forEach(radio => {
    radio.addEventListener('change', function() {
        if(selectedBarName && getNeutralNames().includes(selectedBarName)) {
            selectName(selectedBarName, this.value);
        }
    });
});
```

4.4. Implementacija naprednog ponašanja

4.4.1. Pretraživanje – promjena podataka prikaza ovisno o odabiru podataka

```
// handle Enter key
searchInput.addEventListener('keydown', function(e) {
    if (e.key === 'Enter') {
    const value = this.value.trim();
    if (!value) return;
    // Try to find the first matching name
    const match = allNames.find(d => d.name.toLowerCase() === value.toLowerCase());
    if (match) {
        selectName(match.name, match.gender);
        suggestionsDiv.style.display = 'none';
    }
    }
});
```

4.4.2. Odabiranje imena – mijenja druge prikaze promjenom

```
// set selected name/gender and update map and line chart
function selectName(name, gender) {
    selectedBarName = name;
    selectedBarGender = gender;

    const mapLabel = document.getElementById('map-picked-label');
    if (name && gender) {
        mapLabel.textContent = `Selected: ${name} (${gender})`;
    } else {
        mapLabel.textContent = '';
    }

    const key = name + "|" + gender;
    selectedLineNames = selectedLineNames.filter(d => (d.name + "|" + d.gender) !== key);
    selectedLineNames.push({ name, gender });
    if (selectedLineNames.length > 2) selectedLineNames = selectedLineNames.slice(-2); // keep only last 2
    selectedLineNames.forEach(d => {
        d.color = d.gender === 'F' ? '#8d44add6' : '#4a90e2';
    });
    const infoDiv = document.getElementById('picked-name-info');
    if (name && gender) {
        infoDiv.innerHTML = `Name picked: ${name} (${gender}) <span id="picked-name-x" style="color:#b00;cu
        document.getElementById('picked-name-x').onclick = function() {
            selectedBarName = null;
            selectedBarGender = null;
            infoDiv.textContent = 'Pick a name:';
            document.getElementById('map-picked-label').textContent = '';
            updateMapColors();
            updateLineChart();
        };
    } else {
        infoDiv.textContent = 'Pick a name:';
    }

    updateMapColors();
    updateLineChart();
}
```

4.4.3. Tranzicije umjesto instantne promjene – crtanje linija za linijski graf

```
// animate the line drawing
const totalLength = path.node().getTotalLength();
path
    .attr("stroke-dasharray", totalLength + " " + totalLength)
    .attr("stroke-dashoffset", totalLength)
    .transition()
    .duration(1200)
    .ease(d3.easeLinear)
    .attr("stroke-dashoffset", 0);
```

4.4.4. Implementacije update umjesto delete – u dvosmjernom stupčastom grafu se imena izmijenjuju, dodavaju ona koja nisu postojala u grafu te miču se nepotrebna – ako je jedno ime bilo na 4. mjestu a sada je na 1. će se samo pomaknuti gore umjesto da se briše i ponovno crta

```
// --- FEMALE BARS ---
const femaleGroups = chartGroup.selectAll("g.female-bar-group")
    .data(femaleNames, d => d.name);

// ENTER
const femaleGroupsEnter = femaleGroups.enter()
    .append("g")
    .attr("class", "bar-group female-bar-group");
```

```
// UPDATE
femaleGroups.select("rect")
    .transition()
    .duration(700)
    .attr("x", centerX + 10)
    .attr("y", (d, i) => yScale(i))
    .attr("width", d => xScale(d.count))
    .attr("height", yScale.bandwidth());

femaleGroups.select("text")
    .transition()
    .duration(700)
    .attr("x", d => centerX + 10 + xScale(d.count) / 2)
    .attr("y", (d, i) => yScale(i) + yScale.bandwidth() / 2)
    .text(d => d.name);

// EXIT
femaleGroups.exit().remove();
```

4.4.5. Interaktivnost između više grafova

- prikazano kod ostalih kodova gdje nakon jedne promjene se zovu funkcije više različitih grafova

5. KV5 - Dovršetak projektnog zadatka i pisanje dokumentacije

5.1. Eventualne preinake i dorade rješenja - u dogovoru s nastavnikom

Tijekom završne faze projekta provedene su manje dorade u vizualnom prikazu elemenata, posebice u cilju poboljšanja korisničkog iskustva. Dodan je tekst iznad mape kada je neko ime odabrano. Legenda je pomaknuta na drugu stranu kako nebi smetala linijama te je napravljena da bude transparentan cijeli dio osim gumba za ukloniti ime. Poboljšan je način učitavanja podataka kako bi se sve izvodilo brže (sve je opisano unutar obrade podataka). Poboljšan je vizualni izgled stranice te je dodan naslov.

5.2. Izrada dokumenta - projektne dokumentacije

5.2.1. Hijerarhija projekta.

VP-projekt/

├── babynames.html

├── state_names.csv

├── state_names_full.csv

├── national_names.csv

├── us-states.json

├── counts.py

├── counts.csv

├── fullstatenames.py

├── extracting_data.py

├── 1800to1900del.py

├── README.md

5.2.2. Popis korištenih tehnologija

- HTML5
- CSS
- JavaScript
- D3.js v7 (vizualizacija)
- Python (obrada)
- Pandas

- Podaci – SSA dataset, GeoJSON format za kartografske podatke, CSV format za strukturirane podatke
- Git, Visual Studio Code

5.2.3. Upute za postavljanje.

Preduvjeti:

- Web preglednik
- Git (za kloniranje repozitorija)

Korak 1: Kloniranje repozitorija

bash
git clone https://github.com/majavarsava/VP-projekt.git
cd VP-projekt

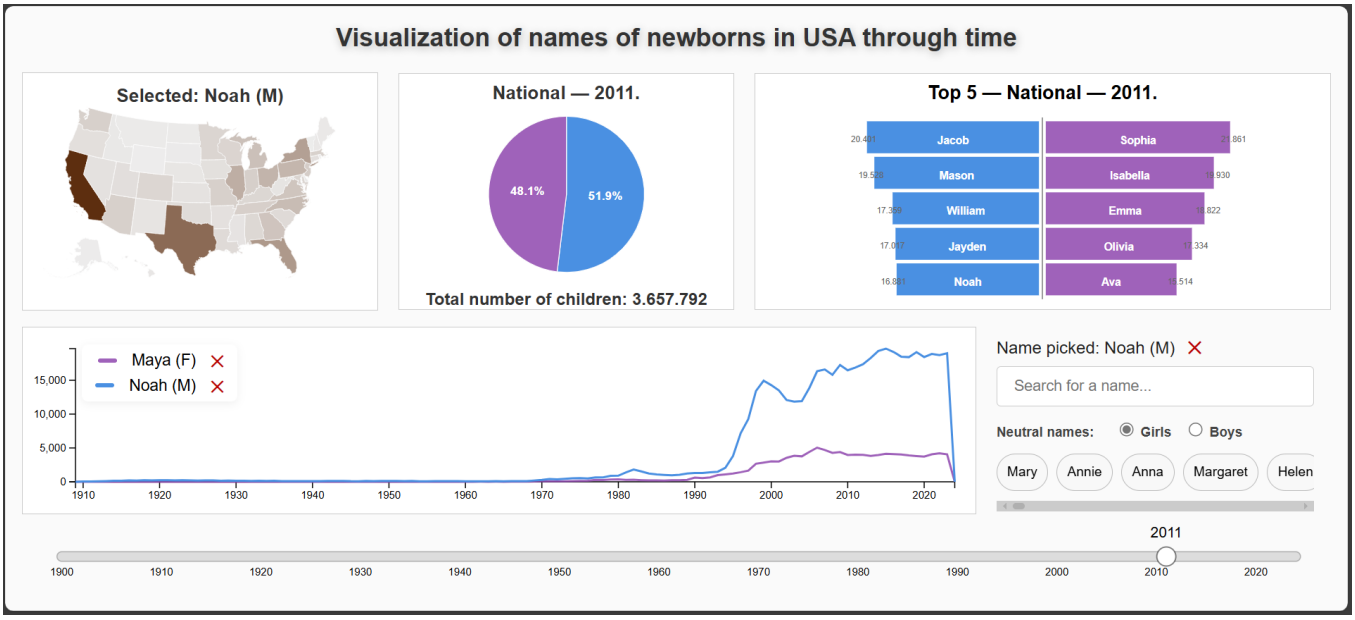
Korak 2: Pokretanje aplikacije

bash

python -m http.server 8000
zatim otvoriti http://localhost:8000 i odabrati babynames.html

Alternativni korak 2: otvaranje pomoću Live Server ekstenije u VS Code-u

5.2.4. Upute za korištenje.



Osnovne funkcionalnosti

1. Pregled po godinama
   i. Koristite klizač na dnu stranice za odabir godine (1900-2024)
   ii. Svi grafovi (osim linijskog) se automatski ažuriraju prema odabranoj godini

2. Interaktivna karta SAD-a
    i. Nalazi se pri vrhu na lijevom dijelu stranice
    ii. Kliknite na bilo koju državu za prikaz podataka specifičnih za tu državu, te van država za deselektiranje država (postavljanje na nacionalnu razinu)
    iii. Kartu možete zumirati i pomicati
    iv. Boje na karti predstavljaju broj djece rođene u toj državi, ili broj djece s tim imenom ako je ime odabrano (iznad karte se pojavljuje tekst ukoliko je odabrano ime)
    v. Tooltip pokazuje naziv države i točan broj djece pri hover-u
3. Top 5 imena na dvosmjernom stupčastom graf
    i. Nalazi se pri vrhu na desnom dijelu stranice
    ii. Dvosmjerni stupčasti graf prikazuje 5 najpopularnijih imena odabrane godine u odabranoj državi ili na nacionalnoj razini
    iii. Lijeva strana: 5 najpopularnijih imena dječaka
    iv. Desna strana: 5 najpopularnijih imena djevojčica
    v. Kliknite na bilo koje ime za dodavanje na linijski graf te ažuriranje boja karte
4. Omjer spolova
    i. Nalazi se između karte i dvosmjernog stupčastog grafa
    ii. Pie chart prikazuje postotak dječaka i djevojčica
    iii. Iznad grafa piše koja godina i koja država (ili nacionalna razina) je odabrana
    iv. Broj ispod grafa pokazuje ukupan broj rođene djece
    v. Tooltip prikazuje točan broj dječaka, tj. djevojčica na hover preko tog odjeljka pie chart-a
5. Trend imena kroz vrijeme
    i. Nalazi se ispod karte i pie charta, iznad klizača
    ii. Linijski graf prati popularnost imena kroz godine u određenoj državi ili na nacionalnoj razini
    iii. Možete dodati do 2 imena istovremenu za usporedbu
    iv. Legenda se nalazi lijevo gore kod grafa
    v. Legenda ima gumb X koji služi za micanje imena s grafa
    vi. Tooltip pokazuje godinu i točni broj djece za godinu iznad kojeg se nalazi miš na hover preko linije

Napredne funkcionalnosti

1. Pretraživanje imena
    i. Nalazi se ispod dvosmjernog stupčastog grafa, a iznad klizača, s desne strane stranice
    ii. Upišite ime u polje za pretraživanje
    iii. Aplikacija će predložiti postojeća imena i spol
    iv. Odaberite ime za prikaz trenda
2. Neutralna imena
    i. Nalazi se ispod pretraživanja imena
    ii. Odaberite radio gumb za dječake ili djevojčice
    iii. Kliknite na neutralno ime (ponuđeni u obliku gumba) za dodavanje u analizu (linijski graf te prikaz boja na karti)
    iv. Neutralna imena su ona koja koriste oba spola, no ovdje biramo želimo li vidjeti podatke o djevojčicama ili dječacima kojima su dodjeljena ta imena
3. Kombiniranje filtera
    i. Možete kombinirati godinu + državu + ime
    ii. Svi grafovi se sinkroniziraju automatski
    iii. Promjene se reflektiraju u stvarnom vremenu
4. Uklanjanje imena iz analize
    i. Koristite X gumb pored imena u linijskom grafu za micanje iz linijskog grafa
    ii. Možete ukloniti jedno ime iz linijskog grafa dok drugo ostane aktivno

iii. Koristite X gumb pored imena iznad tražilice za micanje iz mape

Savjeti za korištenje

Analiza trendova:

1. Odaberite zanimljivo ime klikom na stupčasti graf
2. Promijenite godine pomoću klizača za praćenje trendova
3. Kliknite na različite države za regionalne usporedbe

Usporedba imena:

1. Dodajte prvo ime bilo kojim načinom
2. Dodajte drugo ime pretraživanjem ili klikom
3. Pratite kako se linije kreću kroz vrijeme

Regionalna analiza:

1. Odaberite određenu godinu
2. Kliknite na različite države
3. Usporedite top 5 imena između država

Ograničenja i napomene

- Podaci pokrivaju period od 1900. do 2024.
- Uključene su samo američke savezne države
- Minimum 5 djece po imenu je potrebno za uključivanje u statistike (zbog privatnosti)

Literatura

Podaci i izvori podataka

1. Social Security Administration. *Popular Baby Names*. [Online]. Dostupno na: https://www.ssa.gov/oact/babynames/names.zip [Pristupljeno: 6.2025.]

2. Social Security Administration. *Baby Names by State*. [Online]. Dostupno na: https://www.ssa.gov/oact/babynames/state/namesbystate.zip [Pristupljeno: 6.2025.]

3. Eric Celeste. *US GeoJSON*. [Online]. Dostupno na: https://eric.clst.org/tech/usgeojson/ [Pristupljeno: 6.2025.]

Dokumentacija i tehnička literatura

4. D3.js. *D3.js Geo Documentation*. [Online]. Dostupno na: https://d3js.org/geo [Pristupljeno: 6.2025.]

5. D3.js. *D3.js Pie Chart Documentation*. [Online]. Dostupno na: https://d3js.org/d3-shape/pie [Pristupljeno: 6.2025.]

6. Mike Bostock. *D3.js Slider Example*. [Online]. Dostupno na: https://bl.ocks.org/mbostock/6452972 [Pristupljeno: 6.2025.]

Postojeći primjeri i inspiracija

7. Row Zero. *U.S. Baby Names by Year in a Spreadsheet*. [Online]. Dostupno na: https://rowzero.io/datasets/us-baby-names-by-year [Pristupljeno: 6.2025.]

8. Engaging Data. *US Baby Name Popularity Visualizer*. [Online]. Dostupno na: https://engaging-data.com/baby-name-visualizer/ [Pristupljeno: 6.2025.]

9. Jetpack AI. *Baby Name Popularity Chart*. [Online]. Dostupno na: https://baby-names.jetpack.ai/ [Pristupljeno: 6.2025.]

Projektni repozitorij

10. Varšava, M. *VP-projekt: Projekt iz kolegija Vizualizacija podataka*. GitHub repozitorij. [Online]. Dostupno na: https://github.com/majavarsava/VP-projekt

Prilog I

Poveznica na git repozitorij projekta: majavarsava/VP-projekt: Projekt iz kolegija Vizualizacija podataka