SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Projektni zadatak iz predmeta
VIZUALIZACIJA PODATAKA

Vizualizacija podataka pokemona od 1. do 10. generacije

Student: Mato Benković, DRC
Mentor: Denis Ivanović

U Osijeku, Lipanj 2025.

SADRŽAJ

1. KV1 - Definiranje projektnog zadatka

1.1. Projektni zadatak

[Opisati glavni cilj projekta i svrhu vizualizacije podataka. Ovaj zadatak zahtijeva jasno definiranje problema koji se pokušava riješiti. Potrebno je postaviti jasne ciljeve i granice projekta, kao i odrediti koji će se podaci prikupljati i analizirati kako bi se riješio problem.]

Naziv zadatka:
Vizualizacija podataka pokemona od 1. do 10. Generacije

Opis problema:
Podaci o Pokémonima sadrže velik broj atributa poput osnovnih statistika (HP, Attack, Defense, Sp. Atk, Sp. Def, Speed), tipova (Type 1 i Type 2), fizičkih osobina (visina, težina), te raznih varijanti (forme, regionalne razlike, mega evolucije). Zbog velikog broja zapisa i višedimenzionalnosti podataka, korisnicima je otežano brzo snalaženje, usporedba i donošenje zaključaka o odnosima među Pokémonima. Također, kombinacije tipova i njihova distribucija nisu odmah vidljive, kao ni razlike između Single-Type i Dual-Type Pokémona.

Opis zadatka:
Cilj zadatka je izraditi interaktivnu web-aplikaciju pomoću D3.js koja će omogućiti korisniku dinamički uvid u podatke o Pokémonima kroz vizualne prikaze. Implementirat će se više različitih tipova grafova.

Cilj projekta:
Krajnji cilj projekta je omogućiti korisniku brz, jasan i interaktivan uvid u osnovne karakteristike Pokémona, uz mogućnost dublje analize odnosa između tipova i statistika.

Poveznica na git repozitorij projekta:
https://github.com/mbenkovic02/Pokemon-Data-Visualization

1.2. Podatci

[Pronaći izvore podataka i opisati podatke koji će se koristiti za vizualizaciju.]

Z-1.2.1. U ovom zadatku potrebno je pronaći odgovarajuće izvore podataka koji će se koristiti za rješavanje problema definiranog u prvom zadatku. Važno je osigurati da su podaci kvalitetni i relevantni za problem koji se rješava te da su dostupni za upotrebu.

Z-1.2.2. Potrebno je opisati odabrane podatke kao i navesti pripadajuće izvore.

Za potrebe vizualizacije korišten je skup podataka Pokemon Database.csv preuzet s platforme Kaggle, (https://www.kaggle.com/datasets/mrdew25/pokemon-database/data). Dataset obuhvaća Pokémone od 1. do 10. generacije i sadrži velik broj atributa koji omogućuju sveobuhvatnu analizu njihovih karakteristika. Podaci uključuju:

- Osnovne identifikatore: ID, ime Pokémona, generacija, regionalna forma
- Statistike: HP, Attack, Defense, Sp. Atk, Sp. Def, Speed, Total
- Tipove: Type 1 i Type 2
- Fizičke karakteristike: visina, težina
- Napredni atributi: legendarni status, mitološki status, Baby Pokémon, Mega forma
- Effort Value (EV) stupci: broj EV bodova koje Pokémon daje kada bude poražen (Health EV, Attack EV, Defense EV, Special Attack EV, Special Defense EV, Speed EV, EV Yield Total)

• Pomoćni opisi: klasifikacija, spolni omjeri, sreća, grupa za uzgoj (Egg Group), evolucijske poveznice itd. Podaci su kvalitetni i detaljno strukturirani, što omogućuje složene analize.

1.3. Obrada podataka

[Opisati obavljeni postupak obrade i povezanja podataka.]

Z-1.3.1. Obraditi prikupljene podatke i povezati ih kako bi se stvorio cjelovit skup podataka. Ovo uključuje čišćenje i obradu podataka, kao i provjeru njihove konzistentnosti, aktualnosti, cjelovitosti, tj. kvalitete i ispravnosti.

Nakon preuzimanja dataset-a obavljena je obrada podataka kako bi se osigurao kvalitetan i pregledan skup za vizualizaciju. U početnoj fazi provedena je selekcija relevantnih stupaca, dok su izbačeni oni koji se ne koriste u analizi ili ne doprinose korisničkom iskustvu.

Iz skupa podataka uklonjeni su svi stupci koji nisu doprinosili analitičkoj vrijednosti u kontekstu vizualizacije. Tako je izbačen Pokemon Id jer je Pokedex Number u kombinaciji s Alternate Form Name dovoljan za identifikaciju varijanti. Stupci poput Original Pokemon ID, Special Event Ability, Special Event Ability Description, Male Ratio, Female Ratio i Base Happiness također su uklonjeni zbog rijetke prisutnosti ili irelevantnosti za prikaz. Dodatno su uklonjeni gameplay-mehanički atributi poput Catch Rate, Experience Growth, Egg Group i Egg Cycle Count, koji nemaju značajnu vrijednost za komparativnu analizu statova i tipova.

Naknadno je izbačen i stupac Evolution Details koji je stvarao probleme prilikom identifikacije pokemona u kodu zbog stvaranja duplikata.

1.4. Relevantne vrste prikaza za korištene podatke

[Predložiti nekoliko različitih načina prikaza podataka koji bi bili prikladni za ovaj projekt]

Z-1.4.1. Predložiti moguće načine prikaza podataka koji će pomoći u razumijevanju podataka i rješavanju problema koji je postavljen u prvom zadatku. Ovo može uključivati odabir najprikladnijeg načina vizualizacije podataka, ali to je zadatak iduće vježbe.

1. Radar (spider) graf – Usporedba statistika Pokémona Radar graf omogućuje usporedbu više Pokémona na temelju njihovih osnovnih statistika (HP, Attack, Defense, Sp. Atk, Sp. Def, Speed). Svaka os predstavlja jednu statistiku, a svaki Pokémon je prikazan kao višekutnik. Omogućena je selekcija 2–3 Pokémona, koji se prikazuju na istom grafikonu. Promjena odabira koristi D3.js Enter–Update–Exit obrazac s glatkim prijelazima i tooltipovima. Sa strane se prikazuju dodatni detalji o svakom Pokémonu koji nije vizualno prikazan unutar grafa (npr. tip, slika, ukupna snaga).

---

2. Boxplot – Usporedba distribucije statova po tipu Boxplot prikazuje raspodjelu jedne od osnovnih statistika unutar odabranih Type 1 tipova. Korisnik može odabrati jednu metriku (HP, Attack, Defense, Sp. Atk, Sp. Def, Speed, Total) i do šest tipova Pokémona za usporedbu. Svaki tip prikazan je kao box s prikazom medijana, kvartila, raspona i outliera. Promjene se ažuriraju

korištenjem D3.js tranzicija i Update logike, čime se osigurava glatko vizualno iskustvo bez brisanja i ponovnog crtanja grafa.

---

3. Stacked bar chart – Broj Pokémona po tipu i tipu kombinacije (Single vs Dual-Type) Ovaj prikaz prikazuje distribuciju Type 1 tipova ovisno o tome imaju li Pokémoni i Type 2 (Dual-Type) ili ne (Single-Type). Za svaki tip prikazuje se segmentirana traka s brojem Single-Type i Dual-Type Pokémona. Prikaz koristi stacked layout, sortiranje po ukupnom broju Pokémona, tooltipove s brojem i postotkom, te ikonice tipova za veću preglednost.

---

4. Histogrami – Raspodjela baznih statova među svim Pokémonima Za svaki osnovni stat (HP, Attack, Defense, Sp. Atk, Sp. Def, Speed) izrađen je histogram koji prikazuje distribuciju vrijednosti statova u populaciji Pokémona. Os X prikazuje raspon vrijednosti, dok os Y prikazuje broj (ili postotak) Pokémona unutar svakog intervala (bin-a). Grafikon omogućuje korisniku pregled gdje se većina Pokémona nalazi (npr. 50–100) te prepoznavanje rijetkih ekstremnih slučajeva.

---

5. Rang-liste – Top 10 Pokémona po pojedinim statistikama Za svaku od osnovnih statistika prikazana je lista Pokémona s najvišim vrijednostima. Prikaz uključuje redni broj, ime Pokémona, vrijednost stat-a i sliku. Korisnik može selektirati metriku koja se rangira (npr. Defense) te brzo pregledati koji Pokémoni dominiraju u toj kategoriji. Ovaj prikaz služi kao dopuna distribucijskim prikazima te omogućuje fokus na najjače pojedinačne entitete u dataset-u.

---

6. Stacked bar chart – Raspodjela EV statova po Pokémonima s najvećim EV Yieldom Ova vizualizacija prikazuje koje i koliko EV bodova pojedini Pokémon daje kada bude poražen. Svaki Pokémon s ukupnim EV Yieldom većim od nule prikazan je kao vodoravna traka, segmentirana po statovima (HP, Attack, Defense, Sp. Atk, Sp. Def, Speed) koje doprinosi. Dužina i boja svakog segmenta predstavljaju broj EV bodova po stat-u, dok je ukupna dužina trake jednaka vrijednosti EV Yield Total. Pokémoni su sortirani po ukupnoj vrijednosti EV-a koje daju, čime se korisniku omogućuje brza identifikacija najefikasnijih ciljeva za EV treniranje (grindanje). Tooltip prikazuje ime Pokémona, točnu raspodjelu EV-a po statovima

2. KV2 - Dizajn vizualizacije podataka.

2.1. Pitanja na koja vizualizacija daje odgovor

[Navesti i opisati na koja pitanja će vizualizacija pružati odgovor.
Precizirati pitanja na koja se odgovara vizualizacijom podataka. Potrebno je osigurati da su pitanja jasno formulirana i da se mogu odgovoriti na temelju dostupnih podataka.]

Z-2.1.1. Popis pitanja na koja vizualizacija daje odgovor.

Kako se međusobno uspoređuju dva odabrana Pokémona prema osnovnim statističkim vrijednostima?

Koji od dva Pokémona ima veći ukupni zbroj svih statistika (Total stat)?

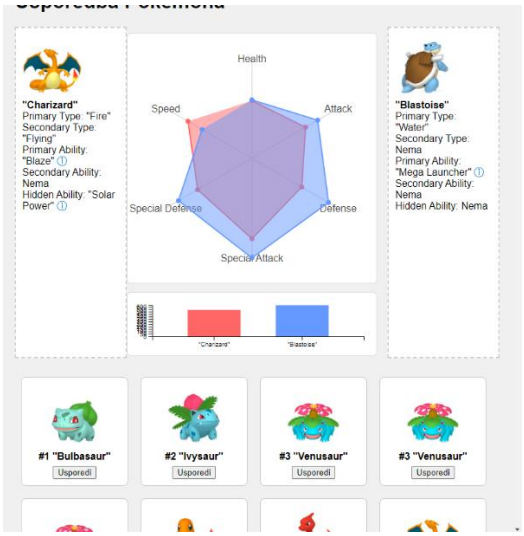
Koje su primarne i sekundarne vrste (type) Pokémona te koje su njihove primarne, sekundarne i skrivene sposobnosti (Ability, Hidden Ability), zajedno s opisima tih sposobnosti?

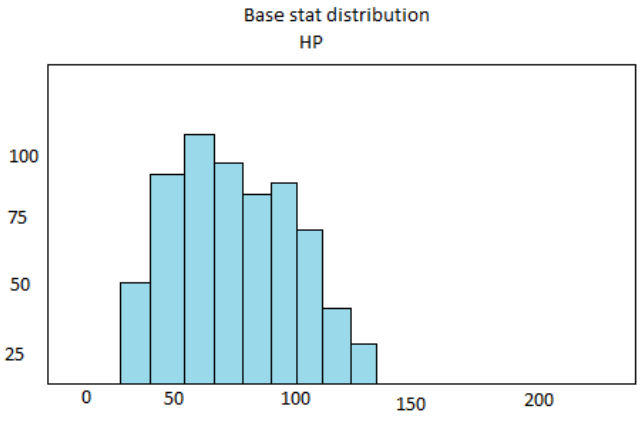
Kako su osnovne statistike (npr. HP, Attack, Defense...) raspoređene među svim Pokémonima – koliko Pokémona ima određenu vrijednost pojedinog stata?

2.2. Skica vizualizacije podataka

[Prikazati skice različitih načina prikaza podataka, uz objašnjenje njihove svrhe]

Z-2.2.1. Izraditi skice konačne vizualizacije podataka, koja će uključivati sve elemente potrebne za rješavanje problema. Ovo uključuje različite tipove grafikona, dijagrama i drugih vizualnih elemenata koji će biti uključeni u vizualizaciju podataka.





2.3. Postojeća rješenja i primjeri

[Navesti primjere sličnih projekata ili kodova koji će biti korisni za izradu ovog projekta s pripadajućim poveznicama i pojašnjenjem koji elementi/dijelovi se planitaju upotrijebiti]

- Z-2.3.1. Pretražiti dostupne stranice sa zbirkama vizualizacija podataka koje su korisne u ovom projektu.
- Z-2.3.2. Pronaći primjere koda za slične vizualizacije.
- Z-2.3.3. Analizirati primjere koda i navesti koje dijelove koda će se koristiti u projektu i objasniti zašto, tj. koji problem se rješava korištenjem pojedinog primjera koda.

Prilikom pretraživanja dostupnih alata za usporedbu Pokémona, poput , nisu pronađena rješenja koja koriste sličan pristup prikaza pomoću radar grafa. Većina dostupnih prikaza temelji se na tabličnom uspoređivanju i osnovnim stupčastim grafovima, stoga nisu pronađeni primjeri koji bi mogli poslužiti kao izravna referenca za implementaciju.

2.4. Prilagodba podataka

[Opisati potrebne prilagodbe podataka te ih prilagodili odabranom načinu prikaza]

- Z-2.4.1. Pripremiti podatke za vizualizaciju.
- Z-2.4.2. Odabrati odgovarajući oblik (engl. format) podataka.
- Z-2.4.3. Urediti podatke za vizualizaciju i prikazati ih u tablici ili drugom prikladnom obliku.
- Z-2.4.4. Pokazati slikom da su podatci uspješno prilagođeni i prikazani na grafičkom prikazu.

Za potrebe prikaza podataka korištena je CSV datoteka naziva Pokemon_Database_Filtered.csv, koja sadrži atribute poput imena Pokémona, osnovnih statistika (HP, Attack, Defense, itd.), ukupnog zbroja statistika (Base Stat Total), tipova (Primary Type, Secondary Type) te sposobnosti (Abilities) s opisima. Podaci su prethodno filtrirani i spremljeni u odgovarajućem tabličnom (CSV) formatu, pogodnom za učitavanje pomoću biblioteke D3.js. Prilikom učitavanja svaki redak predstavlja jednog Pokémona s pripadajućim atributima.

Kao dokaz uspješne pripreme i prikaza podataka koristi se slika radar grafa iz pitanja 2.2.1.. Na vizualizaciji je jasno prikazano učitavanje statistika odabranih Pokémona te njihova pravilna konverzija i raspored po osima grafa.

2.5. Boje i podatci

[Definirati boje korištene u vizualizaciji te vezu vizualnih/grafičkih elemenata i podataka]

- Z-2.5.1. Popis korištenih boja s pripadajućim obrazloženjem.

U vizualizaciji su odabrane kontrastne boje (npr. crvena i plava) kako bi se jasno razlikovali Pokemoni u usporednim prikazima. Boje su dosljedno korištene kroz više grafova kako bi korisniku olakšale praćenje istog Pokémona u različitim kontekstima. Odabir boja osigurava dobru čitljivost i estetsku preglednost.
Za vizualizacije koje prikazuju tipove Pokémona , koristit će se standardizirane boje povezane s elementarnim tipovima Pokémona (npr. Fire – crvena, Water – plava, Electric – žuta) kako bi prikaz bio intuitivan i vizualno ujednačen s postojećim konvencijama.

3. KV3 - Izrada prototipne vizualizacije podataka

[Razrada koncepta, definiranje funkcionalnosti i ponašanja - izrada prototipa.]

3.1. Osnovne funkcionalnosti i ponašanja

[Navesti osnovne funkcionalnosti vizualizacije i njihovo ponašanje]

Z-3.1.1. Identificirati ključne funkcionalnosti koje će biti potrebne za prikaz podataka.

- Prikaz svih Pokémon kartica s osnovnim podacima (ime, forma, sprite, tipovi)
- Pretraživačka traka za filtriranje Pokémon kartica po imenu, formi ili tipu
- Gumb "Compare/Remove" za dodavanje/uklanjanje Pokémona iz usporedbe
- Klikom na karticu za usporedbu Pokémon se uklanja iz prikaza

Z-3.1.2. Definirati osnovne vrste ponašanja.

- Hover efekt prikazuje opise sposobnosti i konkretne vrijednosti statova.
- Klikom na gumb "Usporedi" odabire se Pokémon za lijevu ili desnu stranu usporedbe.
- Uklanjanje Pokémona iz usporedbe klikom na okvir u kojem se nalazi.

Z-3.1.3. Odabrati elemente s kojima će korisnici moći vršiti interakciju i definirati interakcije između korisnika i vizualizacije s pripadajućim opisom.

- Gumb "Usporedi/Ukloni" – omogućuje dodavanje Pokémona u usporedbu ili njegovo uklanjanje.
- Okviri za usporedbu – prikazuju osnovne podatke o odabranim Pokémonima te omogućuju njihovo uklanjanje klikom.
- Pretraživačka traka – omogućuje filtriranje prikazanih kartica po imenu, formi ili tipu Pokémona.
- Navigacijska traka – omogućuje brz prijelaz između screena za usporedbu i screena za prikaz statistika.

3.2. Napredne funkcionalnosti i ponašanja:

[Navesti napredne funkcionalnosti vizualizacije i njihovo ponašanje]

Z-3.2.1. Identificirati napredne funkcionalnosti koje će biti potrebne za analizu podataka.

- Višedimenzionalna vizualizacija: Korištenje Spider (radarskog) grafa za prikaz i usporedbu šest osnovnih statistika (HP, Attack, Defense, Special Attack, Special Defense, Speed) unutar jednog prikaza.
- Višestruka vizualna usporedba: Paralelni prikaz dvaju odabranih Pokémona, uključujući njihove osnovne informacije i ukupni Base Stat Total na dodatnom bar grafu.

- Promjena prikaza podataka po korisničkom odabiru: Omogućena je selekcija i uklanjanje Pokémona za usporedbu klikom, što direktno utječe na ažuriranje vizualizacija.

- Distribucijska analiza: Na dodatnoj stranici (statistics.html) prikazani su histogrami koji omogućuju pregled distribucije vrijednosti pojedinih statistika za cijelu populaciju Pokémona.

- Filtriranje i pretraživanje: Mogućnost filtriranja pomoću više pojmova u stvarnom vremenu, uključujući podršku za različite oblike zapisa (npr. velike i male znakove).

Z-3.2.2. Definirati napredne vrste ponašanja

- Transformacija prikaza ovisno o selekciji: Vizualni prikaz (Spider i Bar graf) se automatski ažurira pri svakom odabiru ili uklanjanju Pokémona.

- Interaktivnost između više prikaza: Odabrani Pokémoni se prikazuju i u lijevom/desnom panelu i u grafovima — promjena u jednom dijelu UI-a reflektira se u ostalim dijelovima sučelja.

- Implementacija D3.js Enter/Update/Exit obrasca: Grafovi i kartice koriste ažuriranje podataka bez ponovnog kreiranja elemenata — omogućena je efikasna manipulacija DOM-om temeljem promjena u podacima.

- Tranzicije i animacije: Histogrami na statistics.html stranici koriste glatke tranzicije pri iscrtavanju stupaca (npr. animirano povećavanje visine), čime se poboljšava razumijevanje distribucije.

- Tooltip ponašanja: Hover nad sposobnostima prikazuje dodatne opise u tooltipu, što omogućuje pristup dodatnim informacijama bez pretrpavanja osnovnog prikaza.

Z-3.2.3. Definirati interakcije koje će omogućiti korisnicima dodatnu analizu podataka.

- Hover nad vrhom radarskog grafa (spider chart) Prikaz točne vrijednosti pojedinog stat atributa pri prelasku pokazivačem preko vrhova poligona ili naziva statova, čime se omogućuje bolja interpretacija snaga i slabosti odabranih Pokémona.

- Hover nad ukupnim Base Stat Total stupcem U bar chartu koji prikazuje usporedbu ukupnih statistika, prikazuje se točna brojčana vrijednost ukupnog zbroja atributa svakog Pokémona pri hoveru.,

3.3. Implementacija osnovnih funkcionalnosti

[Opisati i dokazima potkrijepiti proces implementacije osnovnih funkcionalnosti]

Z-3.3.1. Izraditi kod koji omogućava prethodno definirane funkcionalnosti. Dokazati opisom primjera koda.

Z-3.3.2. Testirati funkcionalnosti i osigurati da su ispravne, tj. da funkcioniraju na očekivani način. Potrebno potvrditi slikom.

3.4. Implementacija osnovnog ponašanja

[Opisati i dokazima potkrijepiti proces implementacije osnovnog ponašanja]

Z-3.4.1. Izraditi kod koji omogućava ranije definirano ponašanje. Dokazati opisom primjera koda.

Z-3.4.2. Testirati ponašanje i osigurati da je ispravno, tj. da funkcionira na očekivani način. Potrebno potvrditi slikom.

Jedna od temeljnih funkcionalnosti aplikacije je dinamičko generiranje kartica za svakog Pokémona u listi. Kod koji omogućava ovu funkcionalnost nalazi se unutar funkcije renderPokemonCards(filteredData). Ova funkcija prima niz Pokémon objekata i za svakog od njih prikazuje karticu s pripadajućim informacijama.
Za svakog Pokémona generira se HTML kartica koja uključuje:

- Sprite sliku,
- Ime i forma Pokémona (baseName, altForm),
- Tipovi (Primary i Secondary Type) prikazani kao zaobljeni pillovi u boji,
- Gumb za usporedbu s mogućnošću dodavanja/uklanjanja Pokémona iz usporedbe klikom.

Kod koji omogućuje interaktivnu usporedbu dvaju Pokémona omogućuje korisniku da klikom na gumb "Usporedi" odabere Pokémona za usporedbu. Funkcija provjerava je li taj Pokémon već odabran s lijeve ili desne strane. Ako jest – uklanja ga. Ako nije – dodaje ga na prvo slobodno mjesto (lijevo ili desno). Nakon toga, dinamički se osvježava prikaz usporedbe (grafovi i kartice).

```
function renderPokemonCards(filteredData) {
listContainer.html("");

filteredData.forEach(d => {
  const pokedexNum = +d["Pokedex Number"];
  const baseName = d["Pokemon Name"];
  const altForm = d["Alternate Form Name"];
  const spritePath = `pokemon_sprites/${pokedexNum}.png`;

  const card = listContainer.append("div")
    .datum(d)
    .attr("class", "pokemon-card")
    .style("height", "240px")
    .style("width", "150px")
    .style("display", "inline-block")
    .style("vertical-align", "top")
    .style("margin", "10px")
    .style("text-align", "center")
    .style("box-sizing", "border-box")
    .style("padding-bottom", "10px");

  card.append("img")
    .attr("src", spritePath)
    .attr("width", 96)
    .attr("height", 96)
    .attr("alt", baseName)
    .style("background", "white")
    .style("padding", "6px")
    .style("border-radius", "8px")
    .style("box-shadow", "0 0 6px rgba(255, 255, 255, 0.3)");

  const nameContainer = card.append("div")
    .style("margin-top", "8px");

  nameContainer.append("div")
    .text(`${pokedexNum} ${baseName}`)
    .style("font-weight", "bold");

  nameContainer.append("div")
    .html(altForm ? `(${altForm})` : " ")
    .style("font-size", "13px")
    .style("color", "#ccc");

  const typeRow = card.append("div")
    .style("display", "flex")
    .style("justify-content", "center")
    .style("gap", "6px")
    .style("margin-top", "6px");

  ["Primary Type", "Secondary Type"].forEach(typeKey => {
    const type = d[typeKey];
    if (type) {
      typeRow.append("span")
        .style("padding", "4px 8px")
        .style("border-radius", "12px")
        .style("background-color", TYPE_COLORS[type] || "#777")
        .style("color", "white")
        .style("font-size", "12px")
        .text(type);
    }
  });

  const compareButton = card.append("button")
    .text("Usporedi")
    .style("margin-top", "10px")
    .style("margin-bottom", "0px")
    .attr("class", "compare-btn")
    .style("padding", "6px 12px")
    .style("border", "none")
    .style("border-radius", "6px")
    .style("cursor", "pointer")
    .style("background", "#222")
    .style("color", "white")
    .style("font-weight", "bold")
    .on("click", () => {
      const id = d["Pokedex Number"];
      const alt = d["Alternate Form Name"] || "";
      const match = (poke) =>
        poke && poke["Pokedex Number"] === id &&
        (poke["Alternate Form Name"] || "") === alt;

      if (match(selected.left)) {
        selected.left = null;
      } else if (match(selected.right)) {
        selected.right = null;
      } else if (!selected.left) {
        selected.left = d;
      } else if (!selected.right) {
        selected.right = d;
      }
      updateComparison();
      updateButtons();
```

Za prikaz odabranih Pokémona u lijevom i desnom okviru usporedbe koristi se funkcija renderPokemonCard(pokemon, container). Ova funkcija prima jednog Pokémona i HTML element u kojem se kartica treba prikazati (npr. #pokemon-left ili #pokemon-right).
Kartica sadrži sljedeće informacije:

- Sprite sliku Pokémona,
- Ime i eventualnu alternativnu formu (npr. Charizard (Mega X)),
- Tipove Pokémona prikazane kao obojani zaobljeni pillovi,
- Primarnu, sekundarnu i skrivenu sposobnost (ako postoje).

Ako neka od sposobnosti ima opis, pored naziva prikazuje se oznaka “ⓘ” koja na hover prikazuje tooltip s opisom sposobnosti. Tooltip se prikazuje pored pokazivača miša, a nestaje kada se miš makne. Na taj način korisnik može brzo doći do dodatnih informacija bez zagušivanja prikaza.

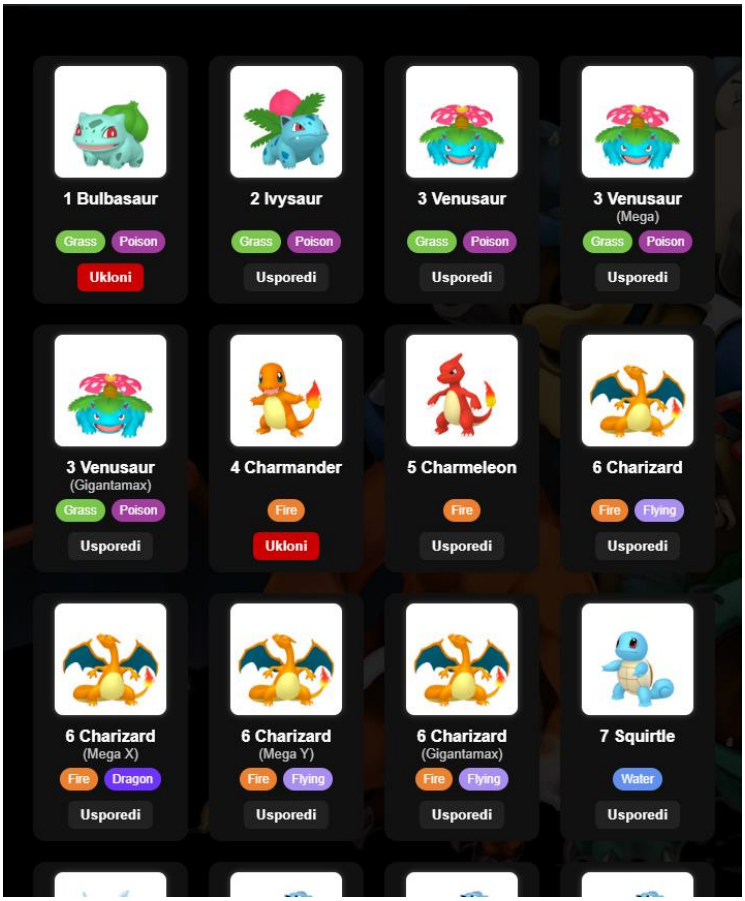
```
function renderPokemonCard(pokemon, container) {
  const card = d3.select(container).html("");
  const pokedexNum = +pokemon["Pokedex Number"];
  const baseName = pokemon["Pokemon Name"];
  const altForm = pokemon["Alternate Form Name"];
  const name = altForm ? `${baseName} (${altForm})` : baseName;
  const spritePath = `pokemon_sprites/${pokedexNum}.png`;

  card.append("img")
    .attr("src", spritePath)
    .attr("width", 96)
    .attr("height", 96)
    .style("display", "block").style("margin", "0 auto")
    .style("background", "white")
    .style("padding", "6px")
    .style("border-radius", "8px")
    .style("box-shadow", "0 0 6px rgba(255, 255, 255, 0.3)");

  card.append("div").text(name)
    .style("font-weight", "bold")
    .style("margin-top", "5px")
    .style("text-align", "center");

  const typeRow = card.append("div")
    .style("display", "flex")
    .style("justify-content", "center")
    .style("gap", "6px")
    .style("margin-top", "8px");

  ["Primary Type", "Secondary Type"].forEach(typeKey => {
    const type = pokemon[typeKey];
    if (type) {
      typeRow.append("span")
        .style("padding", "4px 8px")
        .style("border-radius", "12px")
        .style("background-color", TYPE_COLORS[type] || "#777")
        .style("color", "white")
        .style("font-size", "12px")
        .text(type);
    }
  });

```

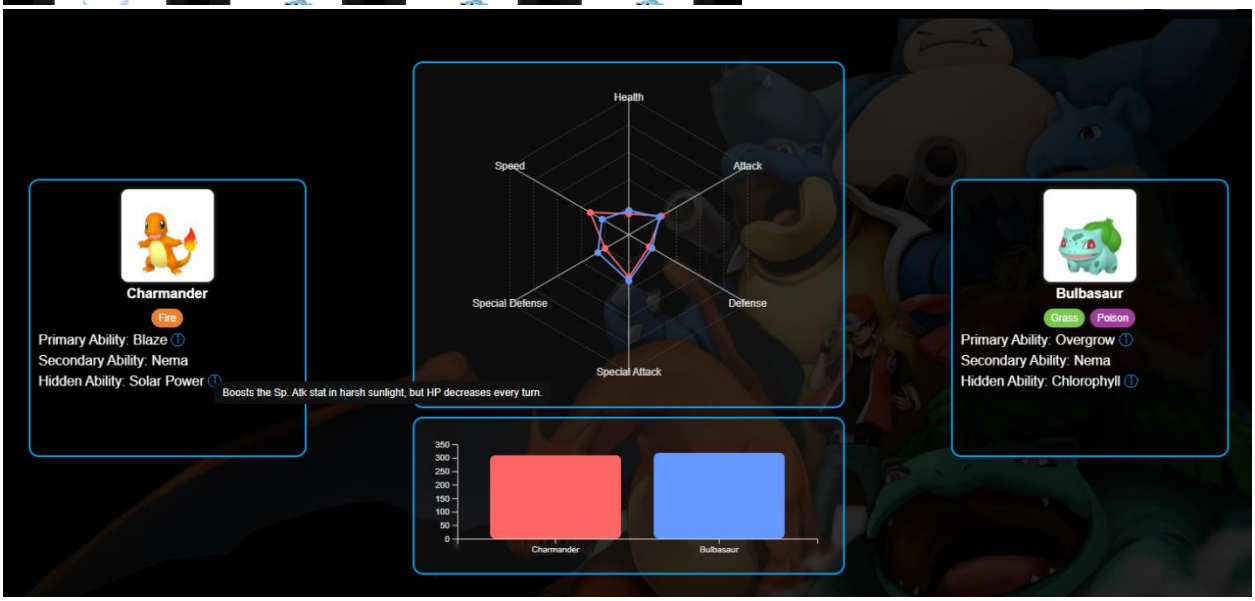
```
    const tooltip = d3.select("body").append("div")
      .attr("id", "tooltip")
      .style("position", "absolute")
      .style("background", "#111")
      .style("color", "#fff")
      .style("padding", "6px 10px")
      .style("border-radius", "6px")
      .style("font-size", "12px")
      .style("pointer-events", "none")
      .style("opacity", 0);

    [
      { key: "Primary Ability", desc: "Primary Ability Description" },
      { key: "Secondary Ability", desc: "Secondary Ability Description" },
      { key: "Hidden Ability", desc: "Hidden Ability Description" }
    ].forEach(({ key, desc }) => {
      const value = pokemon[key] || "Nema";
      const description = pokemon[desc] || "";
      const row = card.append("div").style("margin-top", "6px");
      row.append("span").text(`${key}: ${value}`);
      if (description && value !== "Nema") {
        row.append("span")
          .text(" ⓘ")
          .style("color", "#00aaff")
          .style("cursor", "pointer")
          .on("mouseover", function (event) {
            tooltip.style("opacity", 1).html(description)
              .style("left", (event.pageX + 10) + "px")
              .style("top", (event.pageY + 10) + "px");
          })
          .on("mouseout", () => tooltip.style("opacity", 0));
      }
    });
  }
```

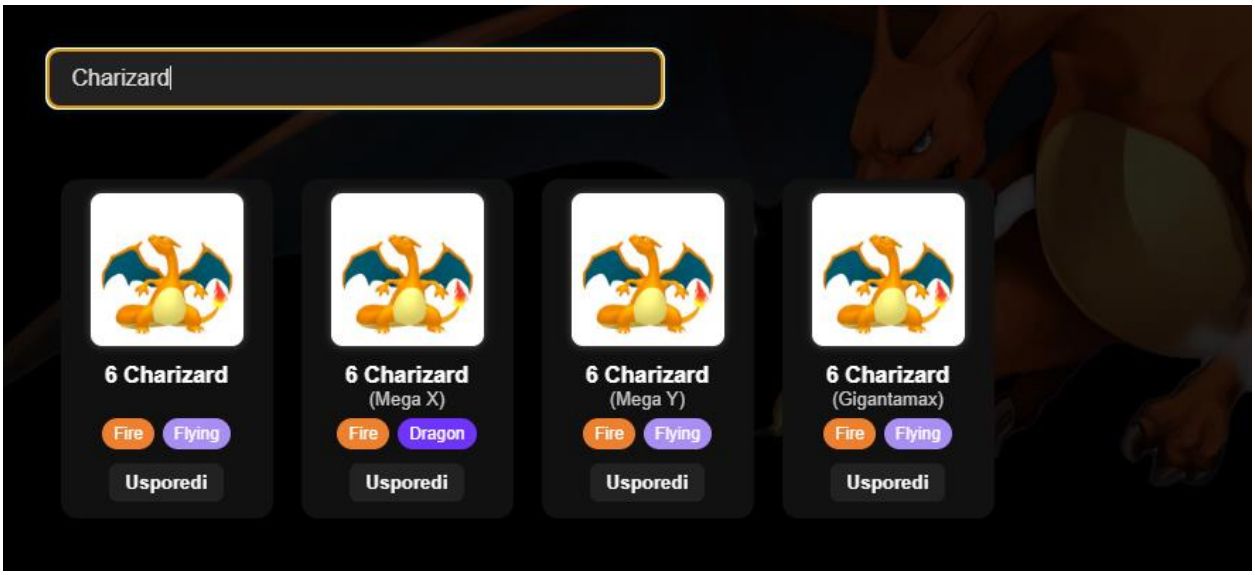
Za realizaciju mogućnosti pretraživanja Pokémona po imenu, formi ili tipu implementiran je osluškivač događaja na pretraživačkoj traci #searchbar. Funkcija se aktivira svaki put kad korisnik unese ili promijeni tekst. Uneseni tekst se pretvara u niz pojmova (razdvojenih razmakom), koji se zatim uspoređuju s kombinacijom naziva, forme i tipova svakog Pokémona. Pokemoni kod kojih su svi uneseni pojmovi pronađeni ostaju prikazani, dok se ostali filtriraju. Na taj način omogućeno je fleksibilno, case-insensitive višepojmovno pretraživanje, čime se korisniku olakšava pronalaženje željenih Pokémona.

```
d3.select("#searchbar").on("input", function () {
  const terms = this.value.toLowerCase().split(/\s+/).filter(Boolean);

  const filtered = data.filter(d => {
    const searchable = [
      d["Pokemon Name"],
      d["Alternate Form Name"] || "",
      d["Primary Type"] || "",
      d["Secondary Type"] || ""
    ].join(" ").toLowerCase();

    return terms.every(term => searchable.includes(term));
  });

  renderPokemonCards(filtered);
});
```

Z-3.1.1. Testirati ponašanje i osigurati da je ispravno, tj. da funkcionira na očekivani način. Potrebno potvrditi slikom.






Charizard
6 Charizard
Fire
Flying
Usporedi
6 Charizard
(Mega X)
Fire
Dragon
Usporedi
6 Charizard
(Mega Y)
Fire
Flying
Usporedi
6 Charizard
(Gigantamax)
Fire
Flying
Usporedi

4. KV4 - Izrada konačne vizualizacije podataka

4.1. Implementacija osnovnih funkcionalnosti

Z-4.1.1. Dovršiti implementaciju koda za nedostajuće osnovne funkcionalnosti.

Z-4.1.2. Testirati osnovne funkcionalnosti i osigurati da su ispravne, tj. da funkcioniraju na očekivani način. Potrebno potvrditi slikom.

Sve planirano je odrađeno na prethodnom KV-u

4.2. Implementacija osnovnog ponašanja

Z-4.2.1. Dovršiti implementaciju koda za nedostajuće osnovne interakcije.

Z-4.2.2. Testirati osnovno ponašanje i osigurati da je ispravno, tj. da funkcionira na očekivani način. Potrebno potvrditi slikom.

Sve planirano je odrađeno na prethodnom KV-u

4.3. Implementacija naprednih funkcionalnosti

Z-4.3.1. Identificirati napredne funkcionalnosti koje će biti implementirane.

Z-4.3.2. Implementirati napredne funkcionalnosti. Dokazati opisom primjera koda.

Z-4.3.3. Testirati napredne funkcionalnosti i osigurati da su ispravne, tj. da funkcioniraju na očekivani način. Potrebno potvrditi slikom.

Usporedba podataka za dva Pokémona

• Usporedba je omogućena putem funkcija renderSpiderChart() i renderTotalChart(). Prikazani podaci obuhvaćaju ukupne statistike (Base Stat Total) te detaljne vrijednosti za šest atributa (Health, Attack, Defense, itd.). Prikaz se ostvaruje kroz dva grafička prikaza — radar (spider) graf i bar chart.

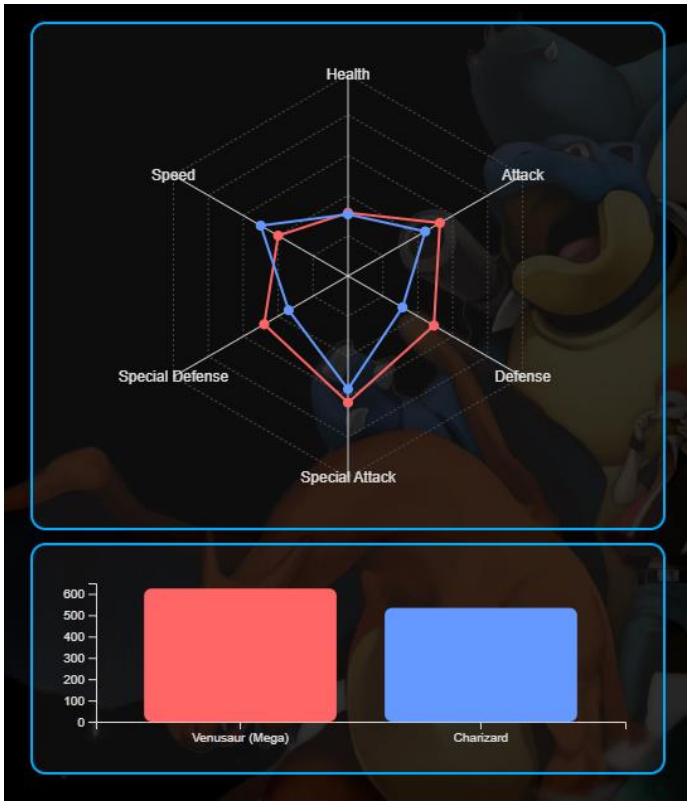
```
function renderSpiderChart(pokemons) {
  const svg = d3.select("#spider-chart");
  const width = +svg.attr("width");
  const height = +svg.attr("height");
  const centerX = width / 2;
  const centerY = height / 2;
  const radius = Math.min(width, height) / 2 - 40;

  const stats = [
    "Health Stat",
    "Attack Stat",
    "Defense Stat",
    "Special Attack Stat",
    "Special Defense Stat",
    "Speed Stat"
  ];
  const angleSlice = (2 * Math.PI) / stats.length;

  const statMaxValues = {
    "Health Stat": 255,
    "Attack Stat": 190,
    "Defense Stat": 250,
    "Special Attack Stat": 194,
    "Special Defense Stat": 250,
    "Speed Stat": 200
  };

  svg.selectAll("*").remove();

  for (let level = 0.2; level <= 1; level += 0.2) {
    const points = stats.map((stat, i) => {
      const angle = i * angleSlice - Math.PI / 2;
      const r = radius * level;
      return [centerX + Math.cos(angle) * r, centerY + Math.sin(angle) * r];
    });
    svg.append("polygon")
      .attr("points", points.map(p => p.join(",")).join(" "))
      .attr("stroke", "#444")
      .attr("fill", "none")
      .attr("stroke-dasharray", "2,2");
  }

  stats.forEach((stat, i) => {
    const angle = i * angleSlice - Math.PI / 2;
    const x = centerX + Math.cos(angle) * radius;
    const y = centerY + Math.sin(angle) * radius;
```

```

    svg.append("line")
      .attr("x1", centerX)
      .attr("y1", centerY)
      .attr("x2", x)
      .attr("y2", y)
      .attr("stroke", "#ccc")
      .attr("stroke-width", 1);

    const statName = stat.replace(" Stat", "");
    const left = pokemons[0] ? `${pokemons[0]["Pokemon Name"]}: ${pokemons[0][stat]}` : "";
    const right = pokemons[1] ? `${pokemons[1]["Pokemon Name"]}: ${pokemons[1][stat]}` : "";
    const tooltipText = `${statName}\n${left}\n${right}`.trim();

    svg.append("text")
      .attr("x", x).attr("y", y)
      .attr("dy", "0.35em")
      .attr("text-anchor", "middle")
      .attr("fill", "#e0e0e0")
      .style("font-size", "12px")
      .text(statName)
      .append("title")
      .text(tooltipText);
  });

  pokemons.forEach((pokemon) => {
    const isLeft = selected.left && pokemon["Pokedex Number"] === selected.left["Pokedex Number"]
      && (pokemon["Alternate Form Name"] || "") === (selected.left["Alternate Form Name"] || "");
    const color = isLeft ? "#ff6666" : "#6699ff";

    const line = d3.lineRadial()
      .radius((_, i) => {
        const stat = stats[i];
        const value = +pokemon[stat];
        const max = statMaxValues[stat];
        return (value / max) * radius;
      })
      .angle((_, i) => i * angleSlice)
      .curve(d3.curveLinearClosed); // ensure polygon closes

    svg.append("path")
      .datum(stats)
      .attr("transform", `translate(${centerX},${centerY})`)
      .attr("fill", "none") // remove fill
      .attr("stroke", color)
```

```
      .attr("stroke-width", 2)
      .attr("d", line);

    stats.forEach((stat, i) => {
      const angle = i * angleSlice - Math.PI / 2;
      const value = +pokemon[stat];
      const max = statMaxValues[stat];
      const scaled = (value / max) * radius;
      const x = centerX + Math.cos(angle) * scaled;
      const y = centerY + Math.sin(angle) * scaled;

      const statName = stat.replace(" Stat", "");
      const left = pokemons[0] ? `${pokemons[0]["Pokemon Name"]}: ${pokemons[0][stat]}` : "";
      const right = pokemons[1] ? `${pokemons[1]["Pokemon Name"]}: ${pokemons[1][stat]}` : "";
      const tooltipText = `${statName}\n${left}\n${right}`.trim();

      svg.append("circle")
        .attr("cx", x).attr("cy", y)
        .attr("r", 4)
        .attr("fill", color)
        .append("title")
        .text(tooltipText);
    });
  });
}
```

Funkcija vizualizira šest osnovnih statistika (HP, Attack, Defense, Sp. Attack, Sp. Defense, Speed) pomoću tzv. "radarskog grafa". Prvo se nacrtaju koncentrični višekuti za referentne razine i osi za svaku metriku. Za svakog odabranog Pokémona izračuna se položaj točaka na grafu prema njegovim vrijednostima statistika, koje se zatim povezuju u zatvoreni poligon. Boje označavaju lijevu i desnu stranu usporedbe. Tooltip prikazuje vrijednosti statova po Pokémonu. Graf se automatski ažurira u skladu s trenutnim odabirom.

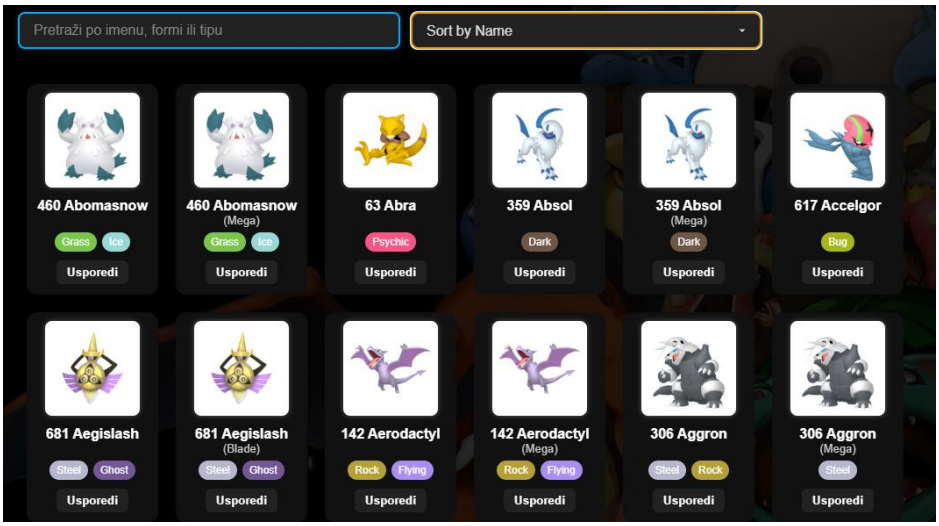
Funkcija prikazuje ukupni zbroj svih statistika ("Base Stat Total") za dva Pokémona u obliku stupčastog grafa. Skala osi y se dinamički postavlja ovisno o većoj vrijednosti. Stupci se prikazuju u bojama povezanim s lijevom i desnom stranom usporedbe te se animirano prikazuju njihova visina (tranzicija). Na hover prikazuju tooltip s točnom vrijednošću. Za generiranje stupaca koristi se D3 enter–update–exit obrazac, čime se omogućuje ažuriranje podataka bez ponovnog iscrtavanja cijelog grafa.



Izmjena/odabir načina sortiranja podataka

• Implementiran je padajući izbornik pomoću kojeg korisnik može sortirati prikaz Pokémona po broju u Pokédexu, imenu te ukupnim statistikama (uzlazno i silazno). Promjenom izbornika poziva se funkcija koja ažurira prikaz prema odabranom kriteriju.

```
d3.select("#sort-select").on("change", function () {
  const sortType = this.value;

  const terms = d3.select("#searchbar").property("value").toLowerCase().split(/\s+/).filter(Boolean);

  let filtered = data.filter(d => {
    const searchable = [
      d["Pokemon Name"],
      d["Alternate Form Name"] || "",
      d["Primary Type"] || "",
      d["Secondary Type"] || ""
    ].join(" ").toLowerCase();

    return terms.every(term => searchable.includes(term));
  });

  switch (sortType) {
    case "name":
      filtered.sort((a, b) => a["Pokemon Name"].localeCompare(b["Pokemon Name"]));
      break;
    case "total-asc":
      filtered.sort((a, b) => +a["Base Stat Total"] - +b["Base Stat Total"]);
      break;
    case "total-desc":
      filtered.sort((a, b) => +b["Base Stat Total"] - +a["Base Stat Total"]);
      break;
    case "pokedex":
    default:
      filtered.sort((a, b) => +a["Pokedex Number"] - +b["Pokedex Number"]);
      break;
  }

  renderPokemonCards(filtered);
});
```



Uvid u višedimenzionalne podatke

• Spider graf prikazuje šest različitih atributa svakog Pokémona unutar jednog radijalnog grafa. Omogućena je istovremena vizualna usporedba više dimenzija (Health, Attack, Defense, Special Attack, Special Defense, Speed) kroz geometrijski oblik.

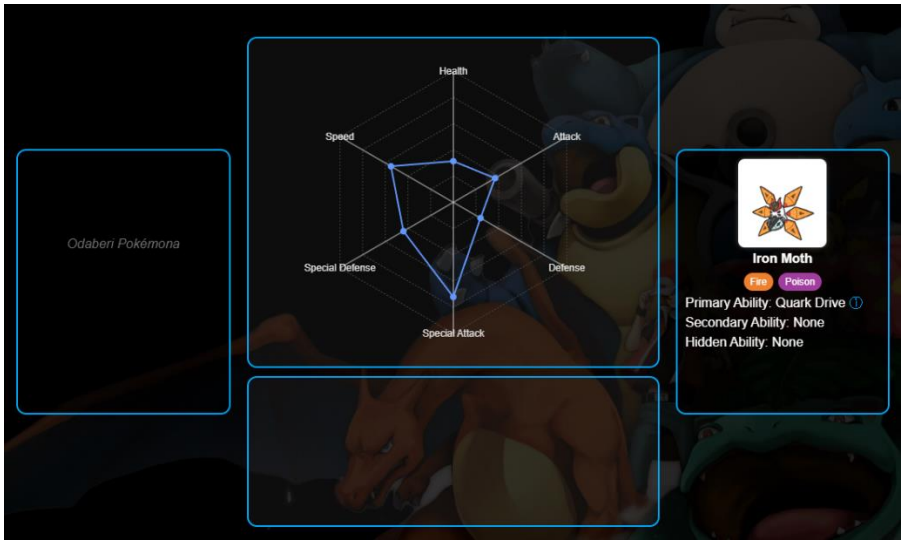
4.4. Implementacija naprednog ponašanja

Z-4.4.1. Identificirati napredna ponašanja koje će biti implementirana.

Z-4.4.2. Implementirati napredna ponašanja. Dokazati opisom primjera koda.

Z-4.4.3. Testirati napredno ponašanje i osigurati da je ispravno, tj. da funkcionira na očekivani način. Potrebno potvrditi slikom.

Transformacija prikaza u ovisnosti o odabranim podatcima
• Spider graf prilagođava prikaz ovisno o broju odabranih Pokémona — npr. ako je odabran samo jedan Pokémon, prikazuje se samo njegova statistika; ako nisu odabrani nijedan, prikaz je prazan.



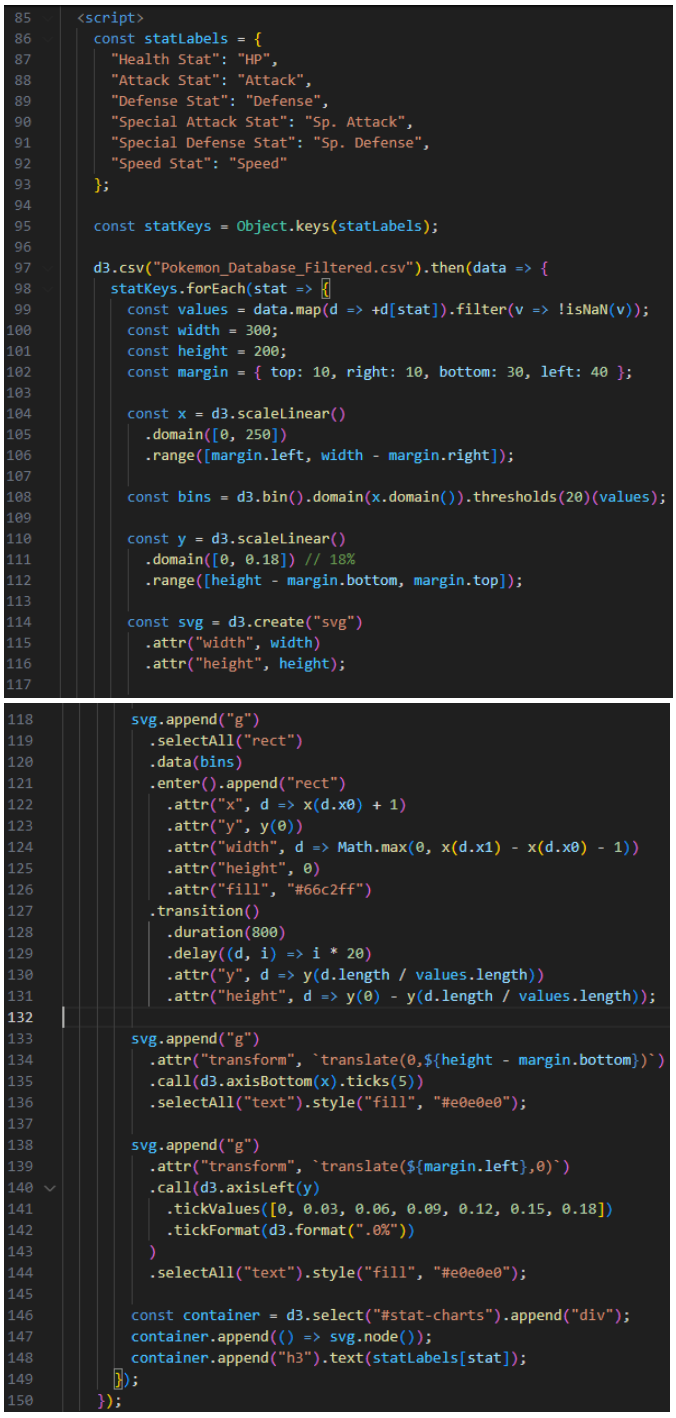
Interaktivnost između više grafova
• Spider graf i bar graf su međusobno povezani funckijom updateComparison() te se istovremeno ažuriraju kada korisnik promijeni izbor Pokémona. Promjena selekcije reflektira se na oba grafa, čime se omogućuje konzistentna i paralelna vizualizacija podataka.

```
function updateComparison() {
  d3.select("#pokemon-left").html('<div class="placeholder-text">Odaberi Pokémona</div>');
  d3.select("#pokemon-right").html('<div class="placeholder-text">Odaberi Pokémona</div>');
  d3.select("#spider-chart").html("");
  d3.select("#total-chart").html("");

  if (selected.left) {
    renderPokemonCard(selected.left, "#pokemon-left");
    d3.select("#pokemon-left").on("click", () => {
      selected.left = null;
      updateComparison();
      updateButtons();
    });
  }
  if (selected.right) {
    renderPokemonCard(selected.right, "#pokemon-right");
    d3.select("#pokemon-right").on("click", () => {
      selected.right = null;
      updateComparison();
      updateButtons();
    });
  }

  const chosen = [selected.left, selected.right].filter(Boolean);
  renderSpiderChart(chosen);
  if (chosen.length === 2) renderTotalChart(chosen);
  updateButtons();
}
```

Glavna funkcija koja sinkronizira sve elemente usporedbe. Uklanja prethodne prikaze lijevog i desnog Pokémona te prikazuje trenutno odabrane. Postavlja event listener za brisanje odabira klikom na prikaz. Poziva renderSpiderChart i renderTotalChart te ponovno ažurira stanje tipki putem updateButtons(). Omogućuje pravilno funkcioniranje usporedbe u stvarnom vremenu.

Korištenje D3.js UPDATE obrasca (Enter–Update–Exit)
• U bar grafu implementiran je Enter–Update–Exit obrazac pomoću D3.js za ažuriranje. Time se omogućava ažuriranje postojećih elemenata bez potrebe za njihovim brisanjem i ponovnim crtanjem, što rezultira efikasnijim i skalabilnijim rješenjem.

Korištenje tranzicija i animacija
• U datoteci statistics.html, histogram koristi glatke animacije stupaca prilikom prikaza visine. Također, renderTotalChart() koristi D3 tranziciju za animaciju visine stupaca, čime se postiže vizualna uglađenost i poboljšava korisničko iskustvo.



```
<script>
  const statLabels = {
    "Health Stat": "HP",
    "Attack Stat": "Attack",
    "Defense Stat": "Defense",
    "Special Attack Stat": "Sp. Attack",
    "Special Defense Stat": "Sp. Defense",
    "Speed Stat": "Speed"
  };

  const statKeys = Object.keys(statLabels);

  d3.csv("Pokemon_Database_Filtered.csv").then(data => {
    statKeys.forEach(stat => {
      const values = data.map(d => +d[stat]).filter(v => !isNaN(v));
      const width = 300;
      const height = 200;
      const margin = { top: 10, right: 10, bottom: 30, left: 40 };

      const x = d3.scaleLinear()
        .domain([0, 250])
        .range([margin.left, width - margin.right]);

      const bins = d3.bin().domain(x.domain()).thresholds(20)(values);

      const y = d3.scaleLinear()
        .domain([0, 0.18]) // 18%
        .range([height - margin.bottom, margin.top]);

      const svg = d3.create("svg")
        .attr("width", width)
        .attr("height", height);
```

```
      svg.append("g")
        .selectAll("rect")
        .data(bins)
        .enter().append("rect")
          .attr("x", d => x(d.x0) + 1)
          .attr("y", y(0))
          .attr("width", d => Math.max(0, x(d.x1) - x(d.x0) - 1))
          .attr("height", 0)
          .attr("fill", "#66c2ff")
        .transition()
          .duration(800)
          .delay((d, i) => i * 20)
          .attr("y", d => y(d.length / values.length))
          .attr("height", d => y(0) - y(d.length / values.length));

      svg.append("g")
        .attr("transform", `translate(0,${height - margin.bottom})`)
        .call(d3.axisBottom(x).ticks(5))
        .selectAll("text").style("fill", "#e0e0e0");

      svg.append("g")
        .attr("transform", `translate(${margin.left},0)`)
        .call(d3.axisLeft(y)
          .tickValues([0, 0.03, 0.06, 0.09, 0.12, 0.15, 0.18])
          .tickFormat(d3.format(".0%"))
        )
        .selectAll("text").style("fill", "#e0e0e0");

      const container = d3.select("#stat-charts").append("div");
      container.append(() => svg.node());
      container.append("h3").text(statLabels[stat]);
    });
  });
```

Kod iscrtavanja stupaca (rect) u histogramu primjenjuje se animacija visine, tako da svaki stupac "naraste" iz vrijednosti y=0 prema svojoj stvarnoj visini. Ova animacija koristi .transition().duration(...)

5. KV5 - Dovršetak projektnog zadatka i pisanje dokumentacije

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

[Ovaj zadatak odnosi se na potencijalne izmjene i dorade koje je potrebno napraviti na rješenju projektnog zadatka, a koje se dogovore s nastavnikom. Moguće je da je potrebno promijeniti neke funkcionalnosti, korekcije u kodu ili bilo koju drugu doradu kako bi se osiguralo kvalitetno i potpuno rješenje.]

5.2. Izrada dokumenta - projektne dokumentacije

[U ovom zadatku potrebno je izraditi projektnu dokumentaciju koja će opisati proces i rezultate projektnog zadatka. Projektna dokumentacija obično uključuje opis projektnog zadatka, potrebne alate, proces rada, opis dizajna i vizualizacije podataka, izvješća o provedenim testovima i rezultatima, zaključak i slično. Cilj je da dokumentacija bude jasna, detaljna i potpuna kako bi drugi mogli razumjeti i koristiti vaše rješenje.]

Z-5.2.1. Hijerarhija projekta.

index.html – Glavna ulazna točka aplikacije
/js – Svi modularni JavaScript fajlovi:
app.js
cards.js – Prikaz kartica i logika usporedbe
charts.js – Prikaz spider i total bar grafa
filters.js – Filtracija i sortiranje kartica
layout.js – Kreiranje osnovnog layouta
state.js – Globalne varijable i stanje selekcije
/data
Pokemon Database.csv – Izvorni skup podataka
Pokemon_Database_Filtered.csv – Filtrirani dataset generiran pomoću `filter.py`
/css
style.css – Stilizacija stranice
/images
bg.jpg – Pozadinska slika aplikacije
/pokemon_sprites
[sprite slike] – Slike svih Pokemona
/scripts
filter.py – Skripta za čišćenje i filtriranje podataka
image_scalper.py – Skripta za dohvat i spremanje sprite slika
List of Pokémon (sprites gallery).html – Korišteno u skripti za dohvat spriteova
/pages
statistics.html – Stranica za dodatne statističke vizualizacije

Z-5.2.2. Popis korištenih tehnologija, bez opisa.

HTML5

CSS3

JavaScript (ES6)

D3.js (v7)

Python 3 (za pripremu podataka)

Visual Studio Code

Git & GitHub

Live Server (VS Code ekstenzija)

Z-5.2.3. Upute za postavljanje.

Klonirajte repozitorij ili preuzmite ZIP arhivu projekta.

Otvorite projekt u Visual Studio Code ili drugom uređivaču.

Pokrenite lokalni server(npr. pomoću Live Server ekstenzije u VS Code-u) i otvorite index.html.

Z-5.2.4. Upute za korištenje.

- Uz pomoć searchbara i opcija sortiranja pronađite željene Pokémone za usporedbu.
- Klikom na gumb "Compare" odaberite do dva Pokémona.
- Odabrani Pokémoni i njihovi ključni podaci prikazuju se lijevo i desno, dok se njihove statistike uspoređuju u Spider i Bar grafu.
- Ponovnim klikom na odabranu karticu uklanja se selekcija.
- Uz pomoć navigacijske trake moguće je prijeći na prikaz distribucije Pokémona po vrijednostima pojedinih statistika.

Literatura

Mike Bostock, “D3.js — Data-Driven Documents”, https://d3js.org/

Pokémon Database, “Complete Pokémon List”, https://pokemondb.net/

MDN Web Docs, “JavaScript Guide”, https://developer.mozilla.org/en-US/docs/Web/JavaScript

W3Schools, “HTML, CSS, JS Reference”, https://www.w3schools.com/

Prilog I

Poveznica na git repozitorij projekta:
https://github.com/mbenkovic02/Pokemon-Data-Visualization

Programski kod