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);
    });
  });
}
```

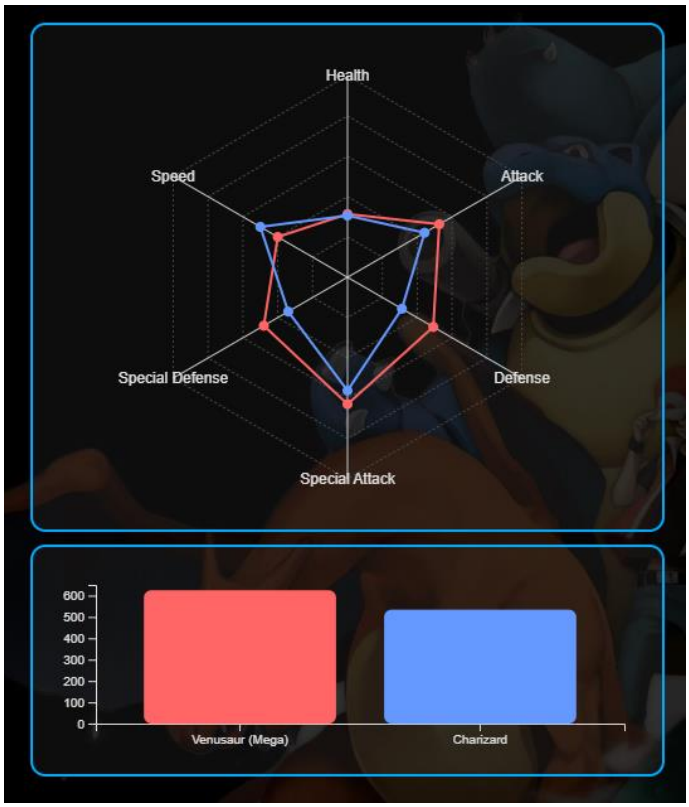
Funkcija vizualizira šest osnovnih statistika (HP, Attack, Defense, Sp. Attack, Sp. Defense, Speed) pomoću tzv. "radarskog grafa".
Prvo se nacrtaju koncentrični višekuti za referentne razine i osi za svaku metriku.
Za svakog odabranog Pokémona izračuna se položaj točaka na grafu prema njegovim vrijednostima 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.

```
function renderTotalChart(pokemons) {
const svg = d3.select("#total-chart");
const width = +svg.attr("width");
const height = +svg.attr("height");
const margin = { top: 30, right: 30, bottom: 40, left: 50 };


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


if (pokemons.length !== 2) return;

const x = d3.scaleBand()
  .domain(pokemons.map(d => d["Alternate Form Name"] ? `${d["Pokemon Name"]} (${d["Alternate Form Name"]})` : d["Pokemon Name"]))
  .range([margin.left, width - margin.right])
  .padding(0.2);

const y = d3.scaleLinear()
  .domain([0, d3.max(pokemons, d => +d["Base Stat Total"])]).nice()
  .range([height - margin.bottom, margin.top]);


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


svg.append("g")
  .attr("transform", `translate(${margin.left},0)`)
  .call(d3.axisLeft(y).ticks(5))
  .selectAll("text")
  .style("fill", "#e0e0e0");


const bars = svg.selectAll(".bar")
  .data(pokemons, d => d["Pokemon Name"]);


bars.exit().remove();
```

```
bars.transition()
  .duration(600)
  .attr("x", d => x(d["Alternate Form Name"] ? `${d["Pokemon Name"]} (${d["Alternate Form Name"]})` : d["Pokemon Name"]))
  .attr("width", x.bandwidth())
  .attr("y", d => y(+d["Base Stat Total"]))
  .attr("height", d => y(0) - y(+d["Base Stat Total"]))
  .attr("fill", (d, i) => i === 0 ? "#ff6666" : "#6699ff");


bars.enter()
  .append("rect")
  .attr("class", "bar")
  .attr("x", d => x(d["Alternate Form Name"] ? `${d["Pokemon Name"]} (${d["Alternate Form Name"]})` : d["Pokemon Name"]))
  .attr("width", x.bandwidth())
  .attr("y", y(0))
  .attr("height", 0)
  .attr("fill", (d, i) => i === 0 ? "#ff6666" : "#6699ff")
  .style("rx", 6)
  .transition()
  .duration(600)
  .attr("y", d => y(+d["Base Stat Total"]))
  .attr("height", d => y(0) - y(+d["Base Stat Total"]));


svg.selectAll(".bar")
  .data(pokemons)
  .select("title").remove();

svg.selectAll(".bar")
  .data(pokemons)
  .append("title")
  .text(d => `Ukupno: ${d["Base Stat Total"]}`);
}
});
```

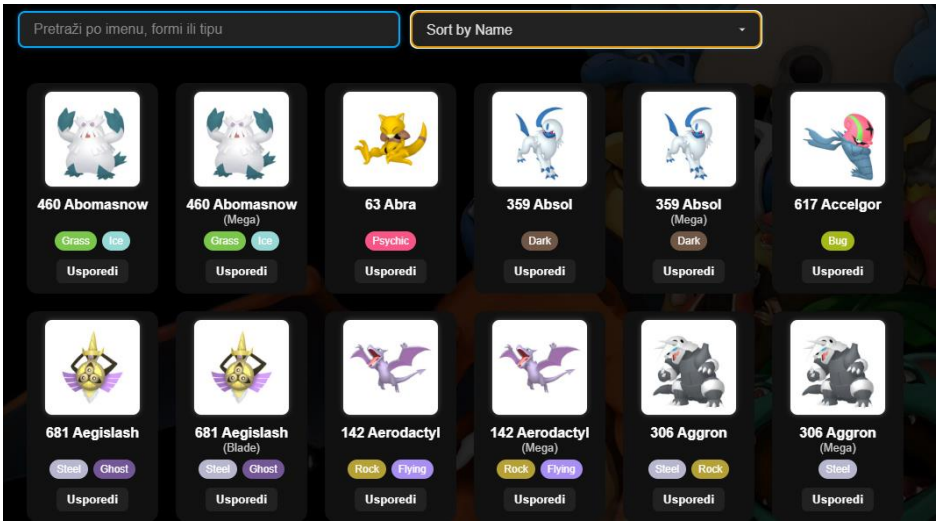
Funkcija prikazuje ukupni zbroj svih statistika ("Base Stat Total") za dva Pokémona u obliku stupčastog grafa.
Skala osi y se dinamički postavlja ovisno o većoj vrijednosti.
Stupci se prikazuju u bojama povezanim s lijevom i desnom stranom usporedbe te se animirano prikazuju njihova visina (tranzicija).

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

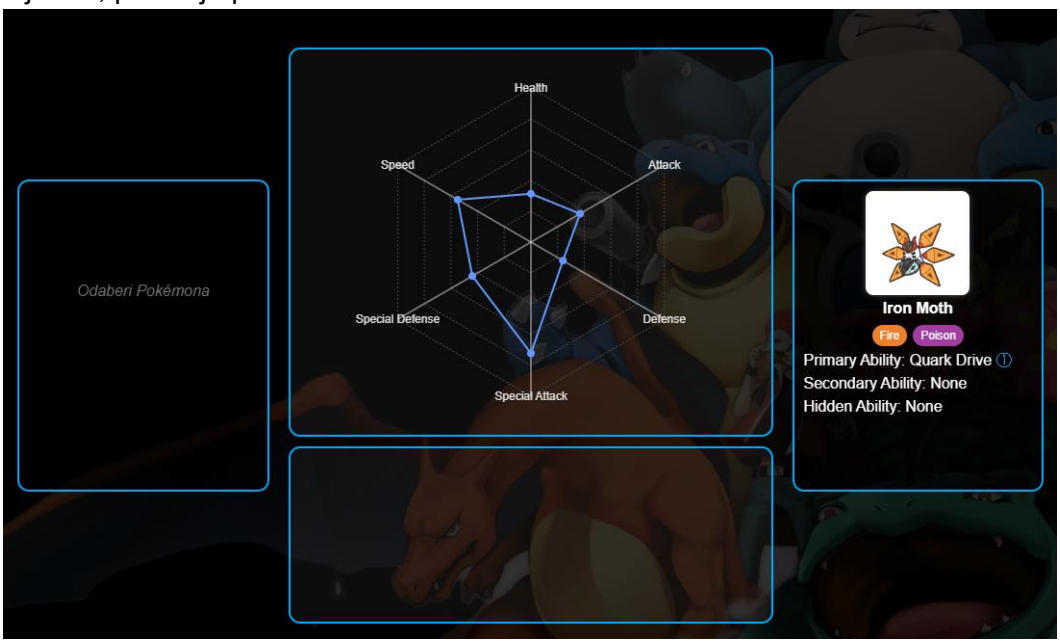
• Spider graf prikazuje šest različitih atributa svakog Pokémona unutar jednog radijalnog grafa. Omogućena je istovremena vizualna usporedba više dimenzija (Health, Attack, Defense, Special Attack, Special Defense, Speed) kroz geometrijski oblik.

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(...)