# KV3 - Izrada prototipne vizualizacije podataka

## Osnovne funkcionalnosti i ponašanja

* + 1. Osnovne funkcionalnosti su:
* Učitavanje podataka
* Obrada podataka ovisno o potrebama kod izrade određenoga grafa
  + 1. Osnovne vrste ponašanja:
* Osnovni prikaz svih napravljenih vizualizacija koristeći sve podatke iz skupa
* Uvođenje raznih pomagala poput teksta, legende, osi… Radi lakšeg razumijevanja vizualizacija
  + 1. Korisnici kod osnovnih funkcionalnosti i ponašanja neće imati nikakvu mogućnost interakcije osim navigacijske trake radi lakšega dolaska do nadzorne ploče. Osnovne funkcionalnosti i ponašanja opisuju učitavanje te obradu podataka koja će ovisiti o tome koji graf se vizualizira te govore i o osnovnome, statičkome prikazu vizualiziranih grafova te njihovih pomagala.

## Napredne funkcionalnosti i ponašanja:

* + 1. Napredne funkcionalnosti su:
* Uvođenje opisnoga prozora koji ispisuje dodatne informacije pritiskom na jednog od elemenata unutar grafa
* Uvođenje ograničene mogućnosti odabira podataka koji će se vizualizirati
  + 1. Napredne vrste ponašanja:
* Transformacija prikaza odnosno iscrtavanje promjena prilikom odabira određenih podataka
* Uvođenje tranzicija kod transformacije prikaza (iscrtavanje promjena) ali i u samom prikazu opisnoga prozora
  + 1. Interakcije kod naprednih funkcionalnosti i ponašanja su sljedeći:
* Interakcija prilikom prelaska miša po elementima grafova – Prilikom prelaska miša prikazati će se do tada skriveni opisni prozor koji sadrži informacije o trenutnome elementu na koji miš pokazuje.
* Interakcija prilikom kretanja miša unutar elementa grafa – Dokle god se miš nalazi unutar elementa grafa opisni prozor će pratiti miš sve do njegovoga napuštanja elementa kada će se opisni prozor ponovno sakriti.
* Interakcija prilikom pritiska na određeni element grafa – Dolazi do transformacije prikaza svih grafova jer dolazi do promjene u podacima koji će se koristiti prilikom vizualizacije.

## Implementacija osnovnih funkcionalnosti

* + 1. Osnovne funkcionalnosti su učitavanje podataka te njihova obrada ovisno o potrebama kod izrade određenoga grafa a to je izvedeno pomoću koda:

data\_extractor.js

|  |
| --- |
| // Data initialization needed for charts  fetch('data/games\_data.json')  .then(response => response.json())  .then(data => {  filtered\_data = data;  all\_data = filtered\_data;  // Data for Pie charts  win[0].value = filtered\_data.filter( entry => entry.win === 'True').length  win[1].value = filtered\_data.filter(entry => entry.win === 'False' ).length  linux[0].value = filtered\_data.filter( entry => entry.linux === 'True').length  linux[1].value = filtered\_data.filter(entry => entry.linux === 'False' ).length  mac[0].value = filtered\_data.filter( entry => entry.mac === 'True').length  mac[1].value = filtered\_data.filter(entry => entry.mac === 'False' ).length  // Data for Histogram  years = [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023]  months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'];  years.forEach(year => {  temp\_data = filtered\_data.filter(entry => entry.year === year);  data\_histogram.push({  x: year,  y: temp\_data.length  });  // Data for line\_diagram  let sum = 0;  temp\_data.forEach( datum =>{  sum += parseFloat(datum.price\_final)  })  data\_line.push({  time: year,  price: (sum / temp\_data.length).toFixed(2)  });  });  data\_histogram.push({ // Have to add this otherwise last column would go outside of svg  x: 2024,  y: 0  });  // Data for bar\_chart  rating\_levels = ['Overwhelmingly Positive','Very Positive','Positive','Mostly Positive','Mixed','Mostly Negative','Negative','Very Negative','Overwhelmingly Negative']  rating\_levels.forEach(rating =>{  data\_ratings.push({  y: rating,  x: filtered\_data.filter(entry => entry.rating === rating).length  })  });  //Draw all graphs  Draw\_pies();  Draw\_Histogram();  Draw\_Bar\_Chart();  Draw\_line\_diagram();  })  .catch(error => console.error('Error fetching JSON:', error)); |

Kod 3.3.1: Prikaz data\_extractor.js skripte za učitavanje i obradu podataka

Gore prikazani kod učitava podatke te se to može i vidjeti početnom naredbom „fetch()“ kojoj se predaje datoteka koja predstavlja podatke. Nakon što su podaci dohvaćeni može se vidjeti kako se za svaki graf dohvaćaju samo oni podaci koji su potrebni kako bi se mogli vizualizirati. Komentarima su označeni dijelovi koda u kojima se izvodi obrada podataka za svaki od grafova. Varijable u kojima se pohranjuju različiti podaci su deklarirani u zasebnoj skripti (Kod 2.) .Na samom kraju se pozivaju funkcije koje iscrtavaju te grafove.

GlobalVariables.js

|  |
| --- |
| let win = [  { label: 'Yes', value: 0 },  { label: 'No', value: 0 },  ];  let linux = [  { label: 'Yes', value: 0 },  { label: 'No', value: 0 },  ];  let mac = [  { label: 'Yes', value: 0 },  { label: 'No', value: 0 },  ];  let data\_histogram = []  let data\_ratings = []  let data\_line = [];  let all\_data;  let filter;  let filtered\_data; |

Kod 3.3.2: Prikaz skripte koja deklarira sve potrebne globalne varijable

* + 1. Testiranjem je dokazano da navedene funkcionalnosti rade i to se može vidjeti u sljedećoj slici (Slika 3.3.1.).

Slika na kojoj se prikazuje tekst, snimka zaslona, dijagram, grafički softver

Opis je automatski generiran

Slika 3.3.1: Prikaz dobivenih vizualizacija nakon dohvaćanja, obrade i crtanja(izrade) grafova

## Implementacija osnovnog ponašanja

* + 1. Osnovne vrste ponašanja su: Osnovni prikaz svih napravljenih vizualizacija koristeći sve podatke iz skupa. Uvođenje raznih pomagala poput teksta, legende, osi… Radi lakšeg razumijevanja vizualizacija. Ta ponašanja su postignuta sljedećim kodom:

index.html

|  |
| --- |
| <div class="container-fluid" id="main">  <div class="row gx-3">  <div class="col-lg-8">  <div class="row gx-3">  <div class="col-lg-6 svg-container">  <h1 class="text-white p-3 text-center">Released games per year</h1>  <svg id="histogram" style="height: 500px;"></svg>  </div>  <div class="col-lg-6 svg-container">  <h1 class="text-white mb-4 p-3">Operating Systems</h3>  <h2 class="text-white mb-5 p-3">Which percentage of Steam games support specific OS?</h5>  <div class="row">  <div class="col-lg-4">  <h3 class="text-white text-center mb-1 p-3">Windows</h3>  </div>  <div class="col-lg-4">  <h3 class="text-white text-center mb-1 p-3">Linux</h3>  </div>  <div class="col-lg-4">  <h3 class="text-white text-center mb-1 p-3">Mac</h3>  </div>  </div>  <div class="row">  <div class="col-lg-4 text-center">  <svg id="pie\_charts1"></svg>  </div>  <div class="col-lg-4 text-center">  <svg id="pie\_charts2"></svg>  </div>  <div class="col-lg-4 text-center">  <svg id="pie\_charts3"></svg>  </div>  </div>  </div>  </div>  <div class="row gx-3">  <div class="col-lg-12 svg-container">  <svg id="line\_diagram"></svg>  </div>  </div>  </div>  <div class="col-lg-4 svg-container" style="margin-bottom: 1rem;">  <h1 class="text-white mb-4 p-3">Number of Games by Steam User Review Rating Category</h1>  <svg id="bar\_chart"></svg>  </div>  </div>  </div> |

Kod 3.4.1: Prikaz stranice koju korisnik vidi

pie\_charts.js

|  |
| --- |
| function Draw\_pies(){  const width = 150;  const height = 150;  const radius = Math.min(width, height) / 2;  color\_palette = ['#0ED354','#C0324F']  const color = d3.scaleOrdinal(color\_palette);  // First pie\_chart  const svg1 = d3.select('#pie\_charts1')  .append('g')    const pie1 = d3.pie().value(d => d.value);  const arc1 = d3.arc().innerRadius(0).outerRadius(radius);  const arcs1 = svg1.selectAll('arc')  .data(pie1(win))  .enter()  .append('g')  .attr('class', 'arc')  .attr('transform',`translate(${width / 2}, ${height / 2})`);  arcs1.append('path')  .attr('d', arc1)  .attr('fill', d => color(d.data.label))  .on("mouseover", function(event, d) {  d3.select("#tooltip")  .html(`<b>Count:</b> ${d.data.value}`) //This will need to be updated  .transition()  .duration(350)  .style("opacity", 1);  })  .on("mousemove", function(event) {  d3.select("#tooltip")  .style("left", (event.pageX + 5) + "px")  .style("top", (event.pageY - 28) + "px");  })  .on("mouseout", function() {  d3.select("#tooltip")  .transition()  .duration(350)  .style("opacity", 0);  })  .on('click',e =>{ // On click event that will be needed later  console.log(e.srcElement.\_\_data\_\_)  });  // Add labels  arcs1.append('text')  .attr('transform', d => `translate(${arc1.centroid(d)})`)  .attr('text-anchor', 'middle')  .text(d => d.data.label);  // Second pie\_chart  const svg2 = d3.select('#pie\_charts2')  .append('g')  const pie2 = d3.pie().value(d => d.value);  const arc2 = d3.arc().innerRadius(0).outerRadius(radius);  const arcs2 = svg2.selectAll('arc')  .data(pie2(linux))  .enter()  .append('g')  .attr('class', 'arc')  .attr('transform',`translate(${width / 2}, ${height / 2})`);  arcs2.append('path')  .attr('d', arc2)  .attr('fill', d => color(d.data.label))  .on("mouseover", function(event, d) {  d3.select("#tooltip")  .html(`<b>Count:</b> ${d.data.value}`) //This will need to be updated  .transition()  .duration(350)  .style("opacity", 1);  })  .on("mousemove", function(event) {  d3.select("#tooltip")  .style("left", (event.pageX + 5) + "px")  .style("top", (event.pageY - 28) + "px");  })  .on("mouseout", function() {  d3.select("#tooltip")  .transition()  .duration(350)  .style("opacity", 0);  })  .on('click',e =>{ // On click event that will be needed later  console.log(e.srcElement.\_\_data\_\_)  });  // Add labels  arcs2.append('text')  .attr('transform', d => `translate(${arc2.centroid(d)})`)  .attr('text-anchor', 'middle')  .text(d => d.data.label);  // Third pie\_chart  const svg3 = d3.select('#pie\_charts3')  .append('g')  const pie3 = d3.pie().value(d => d.value);  const arc3 = d3.arc().innerRadius(0).outerRadius(radius);  const arcs3 = svg3.selectAll('arc')  .data(pie3(mac))  .enter()  .append('g')  .attr('class', 'arc')  .attr('transform',`translate(${width / 2}, ${height / 2})`);  arcs3.append('path')  .attr('d', arc3)  .attr('fill', d => color(d.data.label))  .on("mouseover", function(event, d) {  d3.select("#tooltip")  .html(`<b>Count:</b> ${d.data.value}`) //This will need to be updated  .transition()  .duration(350)  .style("opacity", 1);  })  .on("mousemove", function(event) {  d3.select("#tooltip")  .style("left", (event.pageX + 5) + "px")  .style("top", (event.pageY - 28) + "px");  })  .on("mouseout", function() {  d3.select("#tooltip")  .transition()  .duration(350)  .style("opacity", 0);  })  .on('click',e =>{ // On click event that will be needed later  console.log(e.srcElement.\_\_data\_\_)  });  // Add labels  arcs3.append('text')  .attr('transform', d => `translate(${arc2.centroid(d)})`)  .attr('text-anchor', 'middle')  .text(d => d.data.label);  } |

Kod 3.4.2: Prikaz skripte za crtanje kružnih grafova sa svojim oznakama

histogram.js

|  |
| --- |
| function Draw\_Histogram(){  // Set dimensions for the histogram  const width = 600;  const height = 500;  const margin = { top: 20, right: 30, bottom: 50, left: 50 };  const svg = d3.select('#histogram');  // Create scales  const x = d3.scaleLinear()  .domain(d3.extent(data\_histogram, d => d.x))  .range([margin.left, width-margin.right]);  const y = d3.scaleLinear()  .domain([0, d3.max(data\_histogram, d => d.y)])  .nice()  .range([height - margin.bottom, margin.top]);  // Draw bars  svg.selectAll('rect')  .data(data\_histogram)  .enter()  .append('rect')  .attr('x', d => x(d.x))  .attr('width', 40)  .attr('y', d => y(d.y))  .attr('height', d =>(height - margin.bottom)-y(d.y))  .attr('fill', '#01D1FF')  .on("mouseover", function(event, d) {  d3.select("#tooltip")  .html(`<b>Year:</b> ${d.x}<br><b>Released Games:</b> ${d.y}`) //This will need to be updated  .transition()  .duration(350)  .style("opacity", 1);  })  .on("mousemove", function(event) {  d3.select("#tooltip")  .style("left", (event.pageX + 5) + "px")  .style("top", (event.pageY - 28) + "px");  })  .on("mouseout", function() {  d3.select("#tooltip")  .transition()  .duration(350)  .style("opacity", 0);  })  .on('click',e =>{ // On click event that will be needed later  console.log(e.srcElement.\_\_data\_\_)  });  // Add x-axis  svg.append('g')  .attr('transform', `translate(0,${height - margin.bottom})`)  .attr('class','Apply-white')  .call(d3.axisBottom(x));  // Add y-axis  svg.append('g')  .attr('transform', `translate(${margin.left},0)`)  .attr('class','Apply-white')  .call(d3.axisLeft(y).ticks(10));  // Add x-axis label  svg.append('text')  .attr('transform', `translate(${width/2},${height - margin.bottom/4})`)  .style('text-anchor', 'middle')  .attr('class','Apply-white')  .text('Year');  // Add y-axis label  svg.append('text')  .attr('class','Apply-white')  .attr('x', 0 -height/2 )  .attr('y', 0 + margin.left)  .attr('transform', 'rotate(-90)')  .attr('dy', '1em')  .style('text-anchor', 'middle')  .text('Released games');  } |

Kod 3.4.3: Prikaz skripte za crtanje histograma sa pripadajućim osima

bar\_chart.js

|  |
| --- |
| function Draw\_Bar\_Chart(){  // Set dimensions for the bar chart  const width = 600;  const height = 750;  const margin = { top: 20, right: 30, bottom: 50, left: 50 };  const innerWidth = width - margin.left - margin.right;  const innerHeight = height - margin.top - margin.bottom;  const svg = d3.select("#bar\_chart");  // Create a group element for the inner chart area  const g = svg.append("g")  .attr("transform", `translate(${margin.left},${margin.top})`);  // Create scales  const x = d3.scaleLinear()  .domain([0, d3.max(data\_ratings, d => d.x)])  .nice()  .range([0, innerWidth]);  const y = d3.scaleBand()  .domain(data\_ratings.map(d => d.y))  .range([0, innerHeight])  .padding(0.1);  // Create and append bars  g.selectAll("rect")  .data(data\_ratings)  .enter().append("rect")  .attr("x", 0)  .attr("y", d => y(d.y))  .attr("width", d => x(d.x))  .attr("height", y.bandwidth())  .attr('fill', '#4E8CF6')  .on("mouseover", function(event, d) {  d3.select("#tooltip")  .html(`<b>Category:</b> ${d.y}<br><b>Number of Games:</b> ${d.x}`) //This will need to be updated  .transition()  .duration(350)  .style("opacity", 1);  })  .on("mousemove", function(event) {  d3.select("#tooltip")  .style("left", (event.pageX + 5) + "px")  .style("top", (event.pageY - 28) + "px");  })  .on("mouseout", function() {  d3.select("#tooltip")  .transition()  .duration(350)  .style("opacity", 0);  });  // Create and append axes  g.append("g")  .attr("class", "x-axis")  .attr('class','Apply-white')  .attr("transform", `translate(0,${innerHeight})`)  .call(d3.axisBottom(x));    g.append("g")  .attr("class", "y-axis")  .attr('class','Apply-white')  .style('font-size','15px')  .call(d3.axisLeft(y).tickSize(0).tickPadding(10))  .selectAll("text")  .style("text-anchor", "start");  // Add labels for the axes  svg.append("text")  .attr("class", "x-label")  .attr("text-anchor", "end")  .attr("x", width / 2 + margin.left)  .attr("y", height - 10)  .text("Number of Games");  svg.append("text")  .attr("class", "y-label")  .attr("text-anchor", "middle")  .attr("x", -height / 2 + margin.top)  .attr("y", 15)  .attr("transform", "rotate(-90)")  .style("font-size",'20px')  .text("Steam User Review Rating Category");  } |

Kod 3.4.4: Prikaz skripte za crtanje stupčastoga grafa sa pripadajućim osima

line\_diagram.js

|  |
| --- |
| function Draw\_line\_diagram(){  const margin = { top: 10, right: 30, bottom: 15, left: 40 },  width = 1200 - margin.left - margin.right,  height = 300 - margin.top - margin.bottom;    const innerWidth = width - margin.left - margin.right;  const innerHeight = height - margin.top - margin.bottom;  const svg = d3.select("#line\_diagram").append("g")  .attr("transform", `translate(${margin.left},${margin.top})`);  const x = d3.scaleTime()  .domain(d3.extent(data\_line, d => d.time))  .range([0, innerWidth]);  const y = d3.scaleLinear()  .domain([0, Math.max(...data\_line.map(o => o.price))]) //DO NOT USE d3.max() for some reason it doesn't return correctly. ONLY ON THIS!  .nice()  .range([innerHeight, 0]);  const xAxis = d3.axisBottom(x);  const yAxis = d3.axisLeft(y);  svg.append("g")  .attr("transform", `translate(0,${innerHeight})`)  .attr('class','Apply-white')  .call(xAxis);  svg.append("g")  .attr("class", "y-axis")  .attr('class','Apply-white')  .call(yAxis);  const line = d3.line()  .defined(d => !isNaN(d.price))  .x(d => x(d.time))  .y(d => y(d.price));  svg.append("path")  .datum(data\_line)  .attr("class", "line")  .attr("d", line)  .attr('fill','none')  .style("stroke", "#01D1FF");  // Adding circles to each data point  svg.selectAll("circle")  .data(data\_line)  .enter().append("circle")  .attr("class", "dot")  .attr("cx", d => x(d.time))  .attr("cy", d => y(d.price))  .attr("r", 5)  .attr('fill','#E5C852')  .on("mouseover", function(event, d) {  d3.select("#tooltip")  .html(`<b>Year:</b> ${d.time}<br><b>Average Price:</b> ${d.price}`) //This will need to be updated  .transition()  .duration(350)  .style("opacity", 1);  })  .on("mousemove", function(event) {  d3.select("#tooltip")  .style("left", (event.pageX + 5) + "px")  .style("top", (event.pageY - 28) + "px");  })  .on("mouseout", function() {  d3.select("#tooltip")  .transition()  .duration(350)  .style("opacity", 0);  })  .on('click',e =>{ // On click event that will be needed later  console.log(e.srcElement.\_\_data\_\_)  });    // Add labels for the axes  svg.append("text")  .attr("class", "x-label")  .attr("x", width / 2 + margin.left)  .attr("y", height - 10)  .text("Year");  svg.append("text")  .attr("class", "y-label")  .attr("text-anchor", "middle")  .attr("x", -height / 2)  .attr("y", -22)  .attr("transform", "rotate(-90)")  .text("Average Price");  } |

Kod 3.4.5: Prikaz skripte za crtanje linijskoga dijagrama sa pripadajućim osima

* + 1. Testiranjem je dokazano da navedena ponašanja rade te se to moglo vidjeti u već ranije prikazanoj slici (Slika 3.3.1.). Ali zbog jasnijeg dokaza izdvojene su sljedeće slike:

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, dijagram

Opis je automatski generiran

Slika 3.4.1: Prikaz rezultata skripte za crtanje kružnih grafova (pie\_charts.js)

Slika na kojoj se prikazuje tekst, snimka zaslona, dijagram, Font

Opis je automatski generiran

Slika 3.4.2: Prikaz rezultata skripte za crtanje histograma (histogram.js)

Slika na kojoj se prikazuje snimka zaslona, crta, radnja, dijagram

Opis je automatski generiran

Slika 3.4.3: Prikaz rezultata skripte za crtanje linijskoga dijagrama (line\_diagram.js)

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, dizajn

Opis je automatski generiran

Slika 3.4.4: Prikaz rezultata skripte za crtanje stupčastoga grafa (bar\_chart.js)