SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Projektni zadatak iz predmeta

VIZUALIZACIJA PODATAKA

Vizualizacija podataka sa Steam platforme

Student: Kristian Rücker

Mentor:

U Osijeku, lipanj 2024.

Sadržaj

[1. KV1 - Definiranje projektnog zadatka 1](#_Toc240799046)

[1.1. Projektni zadatak 2](#_Toc99465344)

[1.2. Podatci 2](#_Toc487953797)

[1.3. Obrada podataka 2](#_Toc417461154)

[1.4. Relevantne vrste prikaza za korištene podatke 2](#_Toc1668111988)

[2. KV2 - Dizajn vizualizacije podataka. 2](#_Toc319482197)

[2.1. Pitanja na koja vizualizacija daje odgovor 3](#_Toc1896108668)

[2.2. Skica vizualizacije podataka 3](#_Toc1613071422)

[2.3. Postojeća rješenja i primjeri 3](#_Toc1608452446)

[2.4. Prilagodba podataka 4](#_Toc34148417)

[2.5. Boje i podatci 4](#_Toc1868184388)

[3. KV3 - Izrada prototipne vizualizacije podataka 5](#_Toc1913868324)

[3.1. Osnovne funkcionalnosti i ponašanja 6](#_Toc970262159)

[3.2. Napredne funkcionalnosti i ponašanja: 6](#_Toc1907661375)

[3.3. Implementacija osnovnih funkcionalnosti 6](#_Toc550043536)

[4. KV4 - Izrada konačne vizualizacije podataka 11](#_Toc467214608)

[4.1. Implementacija osnovnih funkcionalnosti 12](#_Toc254652548)

[4.2. Implementacija osnovnog ponašanja 12](#_Toc808550057)

[4.3. Implementacija naprednih funkcionalnosti 12](#_Toc87305041)

[4.4. Implementacija naprednog ponašanja 12](#_Toc1141555374)

[5. KV5 - Dovršetak projektnog zadatka i pisanje dokumentacije 17](#_Toc1837006921)

[5.1. Izrada dokumenta - projektne dokumentacije 18](#_Toc1041051409)

[Literatura 18](#_Toc16162503)

[Prilog I 19](#_Toc734361269)

# KV1 - Definiranje projektnog zadatka

## Projektni zadatak

Naziv zadatka: Analiza podataka sa Steam platforme

Opis problema: Otkrivanje uzoraka i trendova na Steam-u na temelju podataka poput informacija o operacijskom sustavu na kojima su igre igrive, ocijene, omijer pozitivnih ocijena, količine recenzija i njihove cijene.

Opis zadatka: Analiza povijesnih trendova na Steamu kroz različite elmente, uključujući igre i operacijske sustave na kojima su one igrive, ocijene, omijer pozitivnih ocijena, količine recenzija i cijene.

Cilj projekta: Cilj ovog projekta je prikazati trendove u igrama i operacijskih sustava na kojima su one dostupnima na Steam platformi kroz prikladne vizualizacije. Cilj je razumjeti trendove te pronaći uzorke koji bi mogli pomoći developerima prilikom razvoja vlastitih igara.

Poveznica na git repozitorij projekta: <https://github.com/IamGrimbo/VP_Project>

## Podatci

* + 1. *Kao izvor podataka se koristi* <https://store.steampowered.com/charts/> *. Prikupljaju se sve osnovne informacije za svaku igricu dostupnu na platformi.*
    2. *Povjesni podaci su dostupni na:* <https://www.kaggle.com/datasets/michau96/popularity-of-games-on-steam>*Svi podaci se nalaze unutar csv datoteke shs.csv, podatke je potrebno obraditi*

## Obrada podataka

* + 1. *Podaci se obrađuju korištenjem python-a tj. Prvo se učitavaju dodatne informacije o igricama kao što su kategorije i žanrovi, te se onda novonastali dataset spaja sa datasetom koji sadrži opise i tagove. Nakon toga se dataset čisti brišući sve nepotpune stavke.*

## Relevantne vrste prikaza za korištene podatke

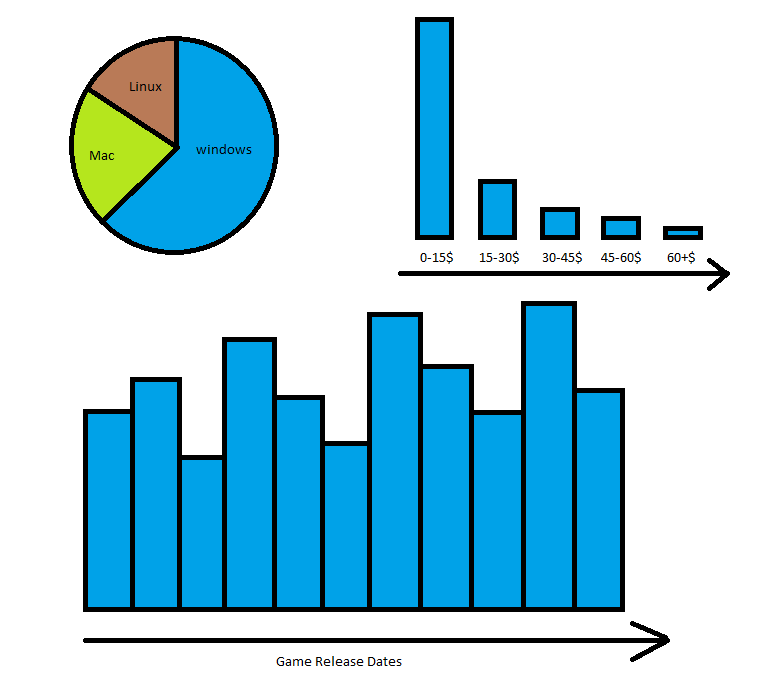
* + 1. *Koristiti će se Piechart, barchart, linechart*

# KV2 - Dizajn vizualizacije podataka.

## Pitanja na koja vizualizacija daje odgovor

* + 1. U ovom projektu prikazujemo najpopulariniji operacijski sustav trenutno (Windows, Linux, Mac) pomoću piecharta, prikazujemo raspon cijena pomoću barcharta, prikazujemo histogram datuma izlazaka videoigrica.

## Skica vizualizacije podataka

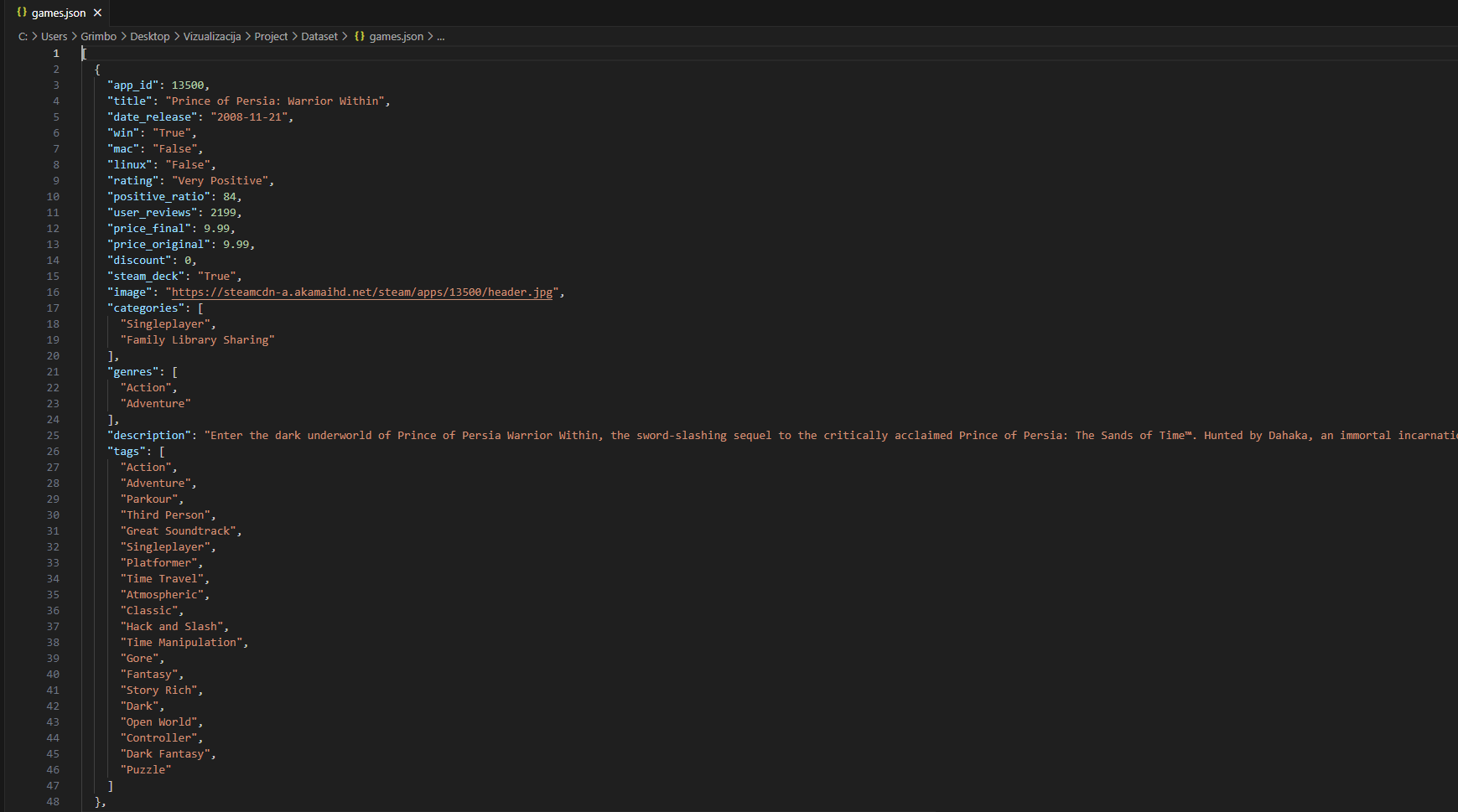
*Skica konačne vizualizacije:*  


## Postojeća rješenja i primjeri

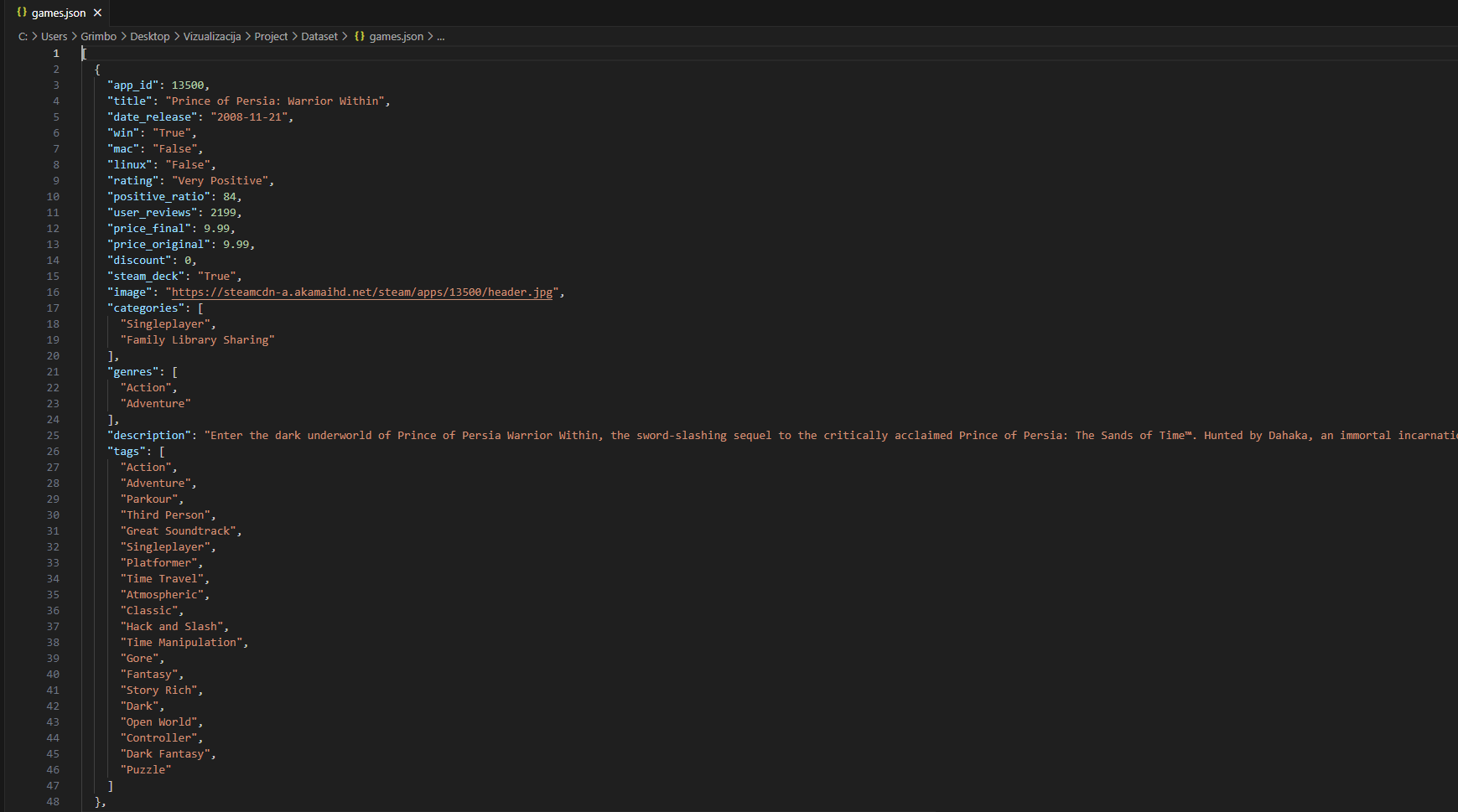
* + 1. *Korisne stranice:*
    2. [*https://observablehq.com/@d3/line-chart-missing-data*](https://observablehq.com/@d3/line-chart-missing-data)[*https://www.tutorialsteacher.com/d3js/create-bar-chart-using-d3js*](https://www.tutorialsteacher.com/d3js/create-bar-chart-using-d3js)  
       [*https://d3-graph-gallery.com/barplot.html*](https://d3-graph-gallery.com/barplot.html)  
       <https://store.steampowered.com/charts/>
    3. *Za Piechart kao osnova se koristi kod koji je dostupan u predlošku za LV*  
       *Osim ovih kodova koristi se i javascrip kako bi se odabrali podaci tj. Mjesec:*  
       <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/range>

## Prilagodba podataka

* + 1. Podaci ne trebaju daljnu obradu, već su obrađeni.
    2. Podaci ne trebaju daljnu obradu, već su obrađeni.
    3. Podaci ne trebaju daljnu obradu, već su obrađeni.
    4. *Primjer kako je prije bilo:*



* + 1. *Podaci nakon obrade:*



## Boje i podatci

* + 1. U ovom projektu boje neće biti od velike važnosti.

# KV3 - Izrada prototipne vizualizacije podataka

## Osnovne funkcionalnosti i ponašanja

* + 1. Odabir kategorije za prikaz podataka
    2. Prikazati podatke na temelju korisnikova odabira
    3. Korisnik odabire kategoriju za koju želi prikaz, na temelju odabranog, prikazuje se sadržaj

## Napredne funkcionalnosti i ponašanja:

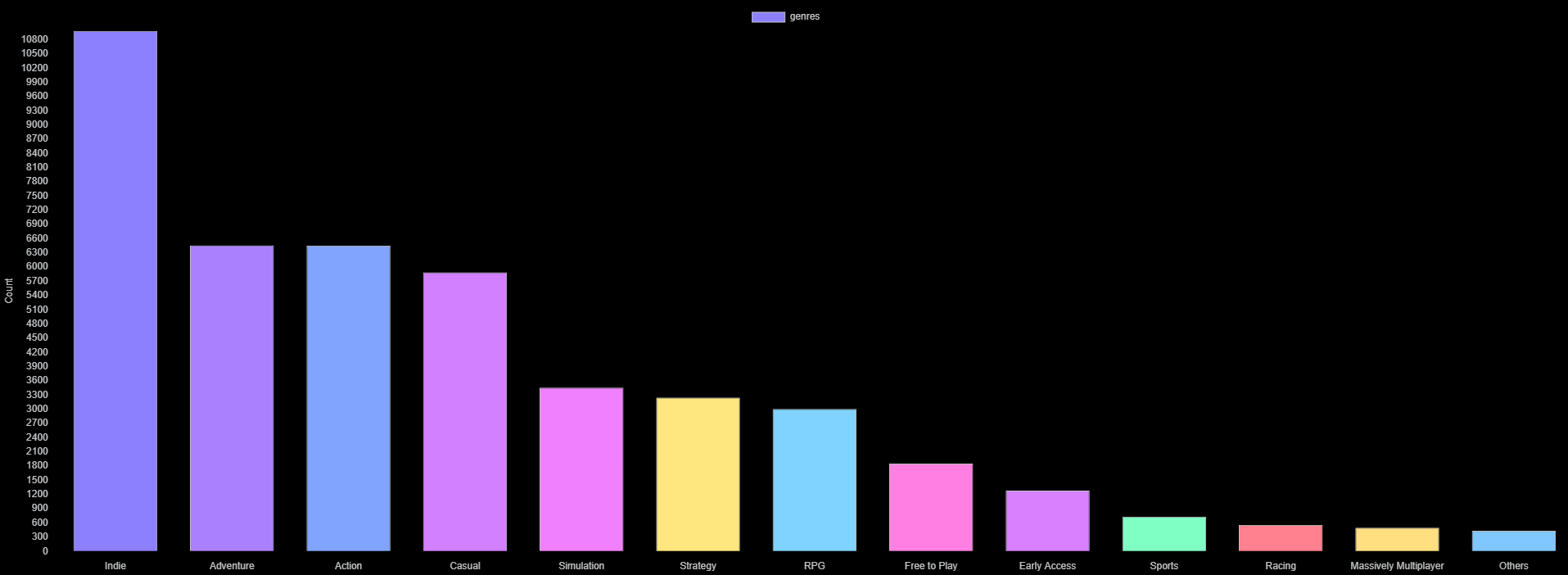
* + 1. Funkcionalonst uvećavanja na vremskoj osi
    2. Korištenje tranzicija, interaktivnost između grafova, transformacije prikaza,na temelju odabranoga prikazuje se željena interakcija
    3. Klikom na određenu kategoriju se prikazuju detaljniji podaci o njoj

## Implementacija osnovnih funkcionalnosti

* + 1. Kod omogućava prikazivanje bar i pie grafikona koristeći Chart.js biblioteku. Bar grafikon prikazuje brojnost različitih kategorija igara kao što su žanrovi, oznake i operativni sustavi, omogućujući korisnicima da vide i usporede učestalost svake kategorije. Pie grafikon prikazuje iste podatke kao dijelove cjelokupnog kruga, vizualizirajući proporcije svake kategorije u odnosu na cjelinu.

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Visualizations</title>  <link rel="stylesheet" href="../CSS/visualizations.css">  <link rel="stylesheet" href="../CSS/topbar.css">  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>  <script src="../JS/visualizations.js" defer></script>  <style>  body {  font-family: Arial, sans-serif;  margin: 0;  padding: 0;  background-color: #000000;  height: 100vh;  display: flex;  flex-direction: column;  }  .container {  display: flex;  flex-direction: column;  flex-grow: 1;  box-sizing: border-box;  overflow: hidden;  }  .chart-container {  flex-grow: 1;  position: relative;  width: 100%;  height: calc(100% - 200px);  }  #chartCanvas {  width: 100%;  height: 100%;  }  .controls {  display: flex;  flex-direction: column;  align-items: center;  }  .controls-row {  display: flex;  justify-content: center;  align-items: center;  gap: 20px;  width: 100%;  }  </style>  </head>  <body>  <!-- Import Topbar -->  <div class="topbar-container">  <!-- Topbar will be imported here -->  </div>  <div class="container">  <h1>Game Visualizations</h1>  <div class="controls">  <select id="visualizationType">  <option value="genres">Genres</option>  <option value="tags">Tags</option>  <option value="categories">Categories</option>  <option value="os">Operating Systems</option>  <option value="release\_dates">Release Dates</option>  <option value="user\_reviews\_positive\_ratio">Reviews/Ratio</option>  <option value="user\_reviews\_price">Reviews/Price</option>  </select>  <div id="chartTypeContainer">  <select id="chartType">  <option value="bar">Bar Chart</option>  <option value="pie">Pie Chart</option>  </select>  </div>  </div>  <div class="chart-container">  <canvas id="chartCanvas"></canvas>  </div>  </div>  <script>  document.addEventListener('DOMContentLoaded', () => {  let chartInstance = null;  document.getElementById('generateChart').addEventListener('click', generateChart);  document.getElementById('visualizationType').addEventListener('change', handleVisualizationTypeChange);  async function fetchData() {  const response = await fetch('../Dataset/games.json');  const data = await response.json();  return data;  }  function countOccurrences(arr) {  return arr.reduce((acc, val) => {  acc[val] = (acc[val] || 0) + 1;  return acc;  }, {});  }  function processData(data, type) {  let items = [];  data.forEach(game => {  if (type === 'genres') {  items = items.concat(game.genres);  } else if (type === 'tags') {  items = items.concat(game.tags);  } else if (type === 'categories') {  items = items.concat(game.categories);  } else if (type === 'os') {  if (game.win === "True") items.push("Windows");  if (game.mac === "True") items.push("Mac");  if (game.linux === "True") items.push("Linux");  if (game.steam\_deck === "True") items.push("Steam Deck");  }  });  let counts = countOccurrences(items);  const threshold = type === 'genres' ? 100 : type === 'tags' ? 1500 : 140;  let filteredCounts = {};  let othersCount = 0;  let othersItems = [];  for (let key in counts) {  if (counts[key] >= threshold) {  filteredCounts[key] = counts[key];  } else {  othersCount += counts[key];  othersItems.push(key);  }  }  let sortedCounts = Object.entries(filteredCounts).sort((a, b) => b[1] - a[1]);  if (othersCount > 0) {  sortedCounts.push(['Others', othersCount]);  }  let labels = sortedCounts.map(item => item[0]);  let values = sortedCounts.map(item => item[1]);  return { labels, values, othersItems };  }  function generateColors(numColors) {  const colors = [];  for (let i = 0; i < numColors; i++) {  const color = `hsl(${Math.random() \* 360}, 100%, 75%)`;  colors.push(color);  }  return colors;  }  function handleVisualizationTypeChange() {  const visualizationType = document.getElementById('visualizationType').value;  const chartTypeContainer = document.getElementById('chartTypeContainer');  chartTypeContainer.style.display = 'block';  }  function generateChart() {  const visualizationType = document.getElementById('visualizationType').value;  const chartType = document.getElementById('chartType').value;  if (visualizationType === 'release\_dates' && !year && !month) {  alert('Please select at least one of year or month.');  return;  }  fetchData().then(data => {  if (chartInstance) {  chartInstance.destroy();  }  const { labels, values, othersItems } = processData(data, visualizationType);  const ctx = document.getElementById('chartCanvas').getContext('2d');  const chartData = {  labels: labels,  datasets: [{  label: visualizationType,  data: values,  backgroundColor: generateColors(labels.length),  borderColor: 'rgba(54, 162, 235, 1)',  borderWidth: 1  }]  };  const options = {  responsive: true,  color: 'red',  maintainAspectRatio: false,  onClick: (event, elements) => {  if (elements.length > 0) {  const index = elements[0].index;  displayGameDetails(labels[index], data, visualizationType, othersItems);  }  },  scales: {  y: {  beginAtZero: true,  ticks: {  stepSize: 1,  color: 'red',  callback: function (value) { return value; }  },  title: {  display: true,  color: 'red',  text: 'Count'  }  },  x: {  title: {  display: true,  color: 'red',  text: visualizationType.charAt(0).toUpperCase() + visualizationType.slice(1)  },  ticks: {  color: 'red',  }  }  }  };  if (chartType === 'pie') {  delete options.scales;  }  chartInstance = new Chart(ctx, {  type: chartType,  data: chartData,  options: options  });  });  }  });  </script>  </body>  </html> |

* + 1. Funkcionalnost je ispravna



# KV4 - Izrada konačne vizualizacije podataka

## Implementacija osnovnih funkcionalnosti

* + 1. Dovršena implementacija osnovnih funkcionalnosti u prošlom Kv-u
    2. Sav kod je dostupan na github-u: <https://github.com/IamGrimbo/Visualization>

## Implementacija osnovnog ponašanja

* + 1. Dovršena implementacija osnovng ponašanja u prošlom Kv-u
    2. Sav kod je dostupan na github-u: <https://github.com/IamGrimbo/Visualization>

## Implementacija naprednih funkcionalnosti

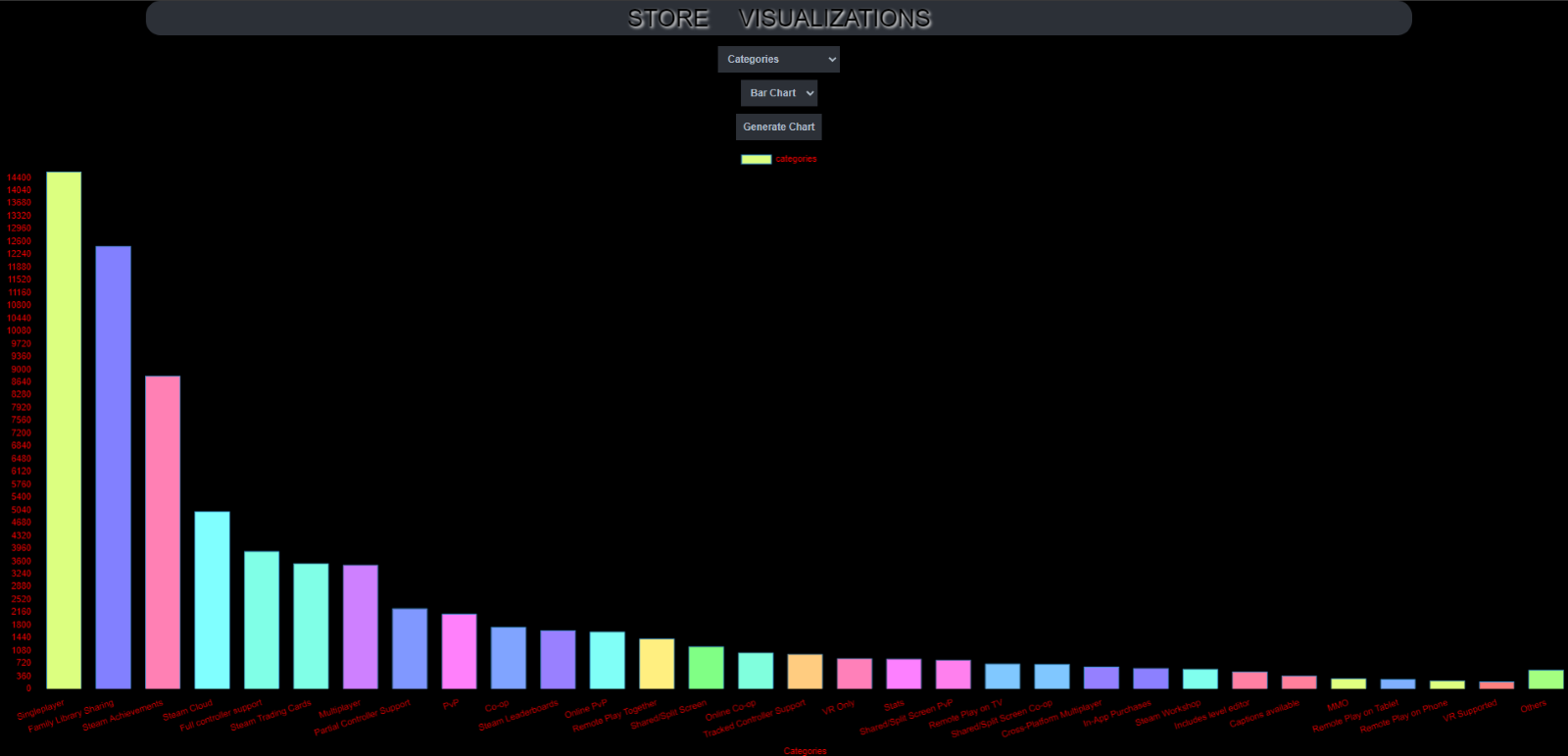
* + 1. Dovršena implementacija naprednih funkcionalnosti u prošlom Kv-u
    2. Sav kod je dostupan na github-u: <https://github.com/IamGrimbo/Visualization>

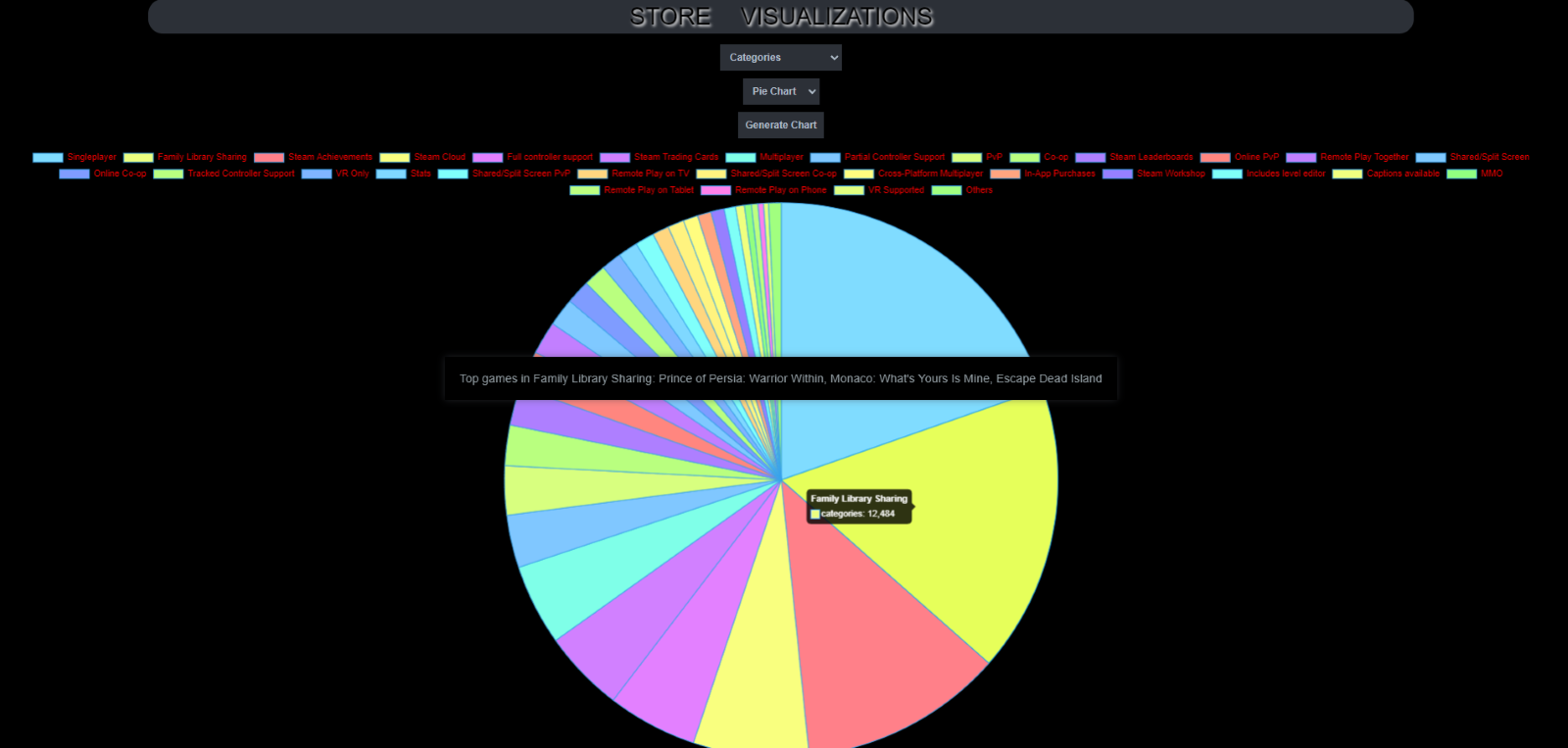
## Implementacija naprednog ponašanja

* + 1. Dovršena implementacija naprednog ponašanja u prošlom Kv-u
    2. Sav kod je dostupan na github-u: <https://github.com/IamGrimbo/Visualization>

Primjer za barchart/piechart:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Visualizations</title>  <link rel="stylesheet" href="../CSS/visualizations.css">  <link rel="stylesheet" href="../CSS/topbar.css">  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>  <script src="../JS/visualizations.js" defer></script>  <style>  body {  font-family: Arial, sans-serif;  margin: 0;  padding: 0;  background-color: #000000;  height: 100vh;  display: flex;  flex-direction: column;  }  .container {  display: flex;  flex-direction: column;  flex-grow: 1;  box-sizing: border-box;  overflow: hidden;  }  .chart-container {  flex-grow: 1;  position: relative;  width: 100%;  height: calc(100% - 200px);  }  #chartCanvas {  width: 100%;  height: 100%;  }  .controls {  display: flex;  flex-direction: column;  align-items: center;  }  .controls-row {  display: flex;  justify-content: center;  align-items: center;  gap: 20px;  width: 100%;  }  </style>  </head>  <body>  <!-- Import Topbar -->  <div class="topbar-container">  <!-- Topbar will be imported here -->  </div>  <div class="container">  <h1>Game Visualizations</h1>  <div class="controls">  <select id="visualizationType">  <option value="genres">Genres</option>  <option value="tags">Tags</option>  <option value="categories">Categories</option>  <option value="os">Operating Systems</option>  <option value="release\_dates">Release Dates</option>  <option value="user\_reviews\_positive\_ratio">Reviews/Ratio</option>  <option value="user\_reviews\_price">Reviews/Price</option>  </select>  <div id="chartTypeContainer">  <select id="chartType">  <option value="bar">Bar Chart</option>  <option value="pie">Pie Chart</option>  </select>  </div>  </div>  <div class="chart-container">  <canvas id="chartCanvas"></canvas>  </div>  </div>  <script>  document.addEventListener('DOMContentLoaded', () => {  let chartInstance = null;  document.getElementById('generateChart').addEventListener('click', generateChart);  document.getElementById('visualizationType').addEventListener('change', handleVisualizationTypeChange);  async function fetchData() {  const response = await fetch('../Dataset/games.json');  const data = await response.json();  return data;  }  function countOccurrences(arr) {  return arr.reduce((acc, val) => {  acc[val] = (acc[val] || 0) + 1;  return acc;  }, {});  }  function processData(data, type) {  let items = [];  data.forEach(game => {  if (type === 'genres') {  items = items.concat(game.genres);  } else if (type === 'tags') {  items = items.concat(game.tags);  } else if (type === 'categories') {  items = items.concat(game.categories);  } else if (type === 'os') {  if (game.win === "True") items.push("Windows");  if (game.mac === "True") items.push("Mac");  if (game.linux === "True") items.push("Linux");  if (game.steam\_deck === "True") items.push("Steam Deck");  }  });  let counts = countOccurrences(items);  const threshold = type === 'genres' ? 100 : type === 'tags' ? 1500 : 140;  let filteredCounts = {};  let othersCount = 0;  let othersItems = [];  for (let key in counts) {  if (counts[key] >= threshold) {  filteredCounts[key] = counts[key];  } else {  othersCount += counts[key];  othersItems.push(key);  }  }  let sortedCounts = Object.entries(filteredCounts).sort((a, b) => b[1] - a[1]);  if (othersCount > 0) {  sortedCounts.push(['Others', othersCount]);  }  let labels = sortedCounts.map(item => item[0]);  let values = sortedCounts.map(item => item[1]);  return { labels, values, othersItems };  }  function generateColors(numColors) {  const colors = [];  for (let i = 0; i < numColors; i++) {  const color = `hsl(${Math.random() \* 360}, 100%, 75%)`;  colors.push(color);  }  return colors;  }  function handleVisualizationTypeChange() {  const visualizationType = document.getElementById('visualizationType').value;  const chartTypeContainer = document.getElementById('chartTypeContainer');  chartTypeContainer.style.display = 'block';  }  function generateChart() {  const visualizationType = document.getElementById('visualizationType').value;  const chartType = document.getElementById('chartType').value;  if (visualizationType === 'release\_dates' && !year && !month) {  alert('Please select at least one of year or month.');  return;  }  fetchData().then(data => {  if (chartInstance) {  chartInstance.destroy();  }  const { labels, values, othersItems } = processData(data, visualizationType);  const ctx = document.getElementById('chartCanvas').getContext('2d');  const chartData = {  labels: labels,  datasets: [{  label: visualizationType,  data: values,  backgroundColor: generateColors(labels.length),  borderColor: 'rgba(54, 162, 235, 1)',  borderWidth: 1  }]  };  const options = {  responsive: true,  color: 'red',  maintainAspectRatio: false,  onClick: (event, elements) => {  if (elements.length > 0) {  const index = elements[0].index;  displayGameDetails(labels[index], data, visualizationType, othersItems);  }  },  scales: {  y: {  beginAtZero: true,  ticks: {  stepSize: 1,  color: 'red',  callback: function (value) { return value; }  },  title: {  display: true,  color: 'red',  text: 'Count'  }  },  x: {  title: {  display: true,  color: 'red',  text: visualizationType.charAt(0).toUpperCase() + visualizationType.slice(1)  },  ticks: {  color: 'red',  }  }  }  };  if (chartType === 'pie') {  delete options.scales;  }  chartInstance = new Chart(ctx, {  type: chartType,  data: chartData,  options: options  });  });  }  });  </script>  </body>  </html> |

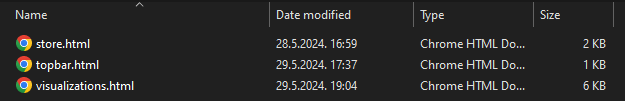
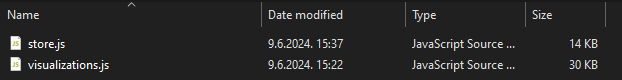
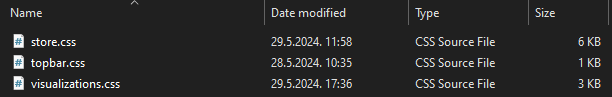
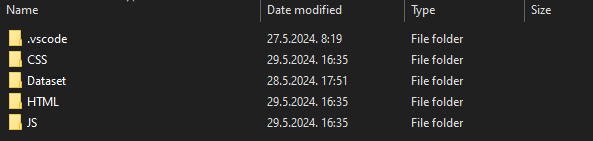




# KV5 - Dovršetak projektnog zadatka i pisanje dokumentacije

## Izrada dokumenta - projektne dokumentacije

* + 1. Projekt se sastoji od više html datoteka unutar kojih je i javascript potreban za rad sa svg elementima. Sav stil je sadržan unutar style.css.
    2. d3.js i javascript
    3. Kako se projekt sastoji samo od nekoliko povezanih .html datoteka, postavljanje je vrlo jednostavno. Početna stranica je store.html, sav stil je dan unutar store.css, visualization.css i topbar.css, pristup drugim stranicama se vrši preko izbornika te je samo potrebno sve datoteke staviti na host.



* + 1. Potrebno je samo skinuti datoteke i sve ih držati zajedno u jednom folderu, poželjno je krenuti od store.html, ali nije nužno. Potreban je WebServer.

# Literatura

Priručnik za LV

<https://www.tutorialsteacher.com/d3js>

<https://d3-graph-gallery.com/graph/barplot_horizontal.html>

<https://d3-graph-gallery.com/pie.html>

<https://www.w3schools.com/js/default.asp>

https://store.steampowered.com/charts

# Prilog I

Poveznica na git repozitorij projekta: <https://github.com/IamGrimbo/VP_Project>

Programski kod: Kod se sastoji od više datoteka, dostupan na linku gore