# KV3 - Izrada prototipne vizualizacije podataka

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

## Osnovne funkcionalnosti i ponašanja

[Navesti osnovne funkcionalnosti vizualizacije i njihovo ponašanje]

* + 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:

[Navesti napredne funkcionalnosti vizualizacije i njihovo ponašanje]

* + 1. Funkcionalonst uvećavanja na vremskoj osi
    2. Transformacije prikaza
    3. Klikom na neku određenu kategoriju(npr na udio po proizvodu) otvara se detaljniji prikaz njegovih točnih proizvoda

## Implementacija osnovnih funkcionalnosti

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Visualizations</title>
6. <link rel="stylesheet" href="../CSS/visualizations.css">
7. <link rel="stylesheet" href="../CSS/topbar.css">
8. <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
9. <script src="../JS/visualizations.js" defer></script>
10. <style>
11. body {
12. font-family: Arial, sans-serif;
13. margin: 0;
14. padding: 0;
15. background-color: #000000;
16. height: 100vh;
17. display: flex;
18. flex-direction: column;
19. }
20. .container {
21. display: flex;
22. flex-direction: column;
23. flex-grow: 1;
24. box-sizing: border-box;
25. overflow: hidden;
26. }
27. .chart-container {
28. flex-grow: 1;
29. position: relative;
30. width: 100%;
31. height: calc(100% - 200px);
32. }
33. #chartCanvas {
34. width: 100%;
35. height: 100%;
36. }
37. .controls {
38. display: flex;
39. flex-direction: column;
40. align-items: center;
41. }
42. .controls-row {
43. display: flex;
44. justify-content: center;
45. align-items: center;
46. gap: 20px;
47. width: 100%;
48. }
49. </style>
50. </head>
51. <body>
52. <!-- Import Topbar -->
53. <div class="topbar-container">
54. <!-- Topbar will be imported here -->
55. </div>
56. <div class="container">
57. <h1>Game Visualizations</h1>
58. <div class="controls">
59. <select id="visualizationType">
60. <option value="genres">Genres</option>
61. <option value="tags">Tags</option>
62. <option value="categories">Categories</option>
63. <option value="os">Operating Systems</option>
64. <option value="release\_dates">Release Dates</option>
65. <option value="user\_reviews\_positive\_ratio">Reviews/Ratio</option>
66. <option value="user\_reviews\_price">Reviews/Price</option>
67. </select>
68. <div id="chartTypeContainer">
69. <select id="chartType">
70. <option value="bar">Bar Chart</option>
71. <option value="pie">Pie Chart</option>
72. </select>
73. </div>
74. </div>
75. <div class="chart-container">
76. <canvas id="chartCanvas"></canvas>
77. </div>
78. </div>
79. <script>
80. document.addEventListener('DOMContentLoaded', () => {
81. let chartInstance = null;
82. document.getElementById('generateChart').addEventListener('click', generateChart);
83. document.getElementById('visualizationType').addEventListener('change', handleVisualizationTypeChange);
84. async function fetchData() {
85. const response = await fetch('../Dataset/games.json');
86. const data = await response.json();
87. return data;
88. }
89. function countOccurrences(arr) {
90. return arr.reduce((acc, val) => {
91. acc[val] = (acc[val] || 0) + 1;
92. return acc;
93. }, {});
94. }
95. function processData(data, type) {
96. let items = [];
97. data.forEach(game => {
98. if (type === 'genres') {
99. items = items.concat(game.genres);
100. } else if (type === 'tags') {
101. items = items.concat(game.tags);
102. } else if (type === 'categories') {
103. items = items.concat(game.categories);
104. } else if (type === 'os') {
105. if (game.win === "True") items.push("Windows");
106. if (game.mac === "True") items.push("Mac");
107. if (game.linux === "True") items.push("Linux");
108. if (game.steam\_deck === "True") items.push("Steam Deck");
109. }
110. });
111. let counts = countOccurrences(items);
112. const threshold = type === 'genres' ? 100 : type === 'tags' ? 1500 : 140;
113. let filteredCounts = {};
114. let othersCount = 0;
115. let othersItems = [];
116. for (let key in counts) {
117. if (counts[key] >= threshold) {
118. filteredCounts[key] = counts[key];
119. } else {
120. othersCount += counts[key];
121. othersItems.push(key);
122. }
123. }
124. let sortedCounts = Object.entries(filteredCounts).sort((a, b) => b[1] - a[1]);
125. if (othersCount > 0) {
126. sortedCounts.push(['Others', othersCount]);
127. }
128. let labels = sortedCounts.map(item => item[0]);
129. let values = sortedCounts.map(item => item[1]);
130. return { labels, values, othersItems };
131. }
132. function generateColors(numColors) {
133. const colors = [];
134. for (let i = 0; i < numColors; i++) {
135. const color = `hsl(${Math.random() \* 360}, 100%, 75%)`;
136. colors.push(color);
137. }
138. return colors;
139. }
140. function handleVisualizationTypeChange() {
141. const visualizationType = document.getElementById('visualizationType').value;
142. const chartTypeContainer = document.getElementById('chartTypeContainer');
143. chartTypeContainer.style.display = 'block';
144. }
145. function generateChart() {
146. const visualizationType = document.getElementById('visualizationType').value;
147. const chartType = document.getElementById('chartType').value;
148. if (visualizationType === 'release\_dates' && !year && !month) {
149. alert('Please select at least one of year or month.');
150. return;
151. }
152. fetchData().then(data => {
153. if (chartInstance) {
154. chartInstance.destroy();
155. }
156. const { labels, values, othersItems } = processData(data, visualizationType);
157. const ctx = document.getElementById('chartCanvas').getContext('2d');
158. const chartData = {
159. labels: labels,
160. datasets: [{
161. label: visualizationType,
162. data: values,
163. backgroundColor: generateColors(labels.length),
164. borderColor: 'rgba(54, 162, 235, 1)',
165. borderWidth: 1
166. }]
167. };
168. const options = {
169. responsive: true,
170. color: 'red',
171. maintainAspectRatio: false,
172. onClick: (event, elements) => {
173. if (elements.length > 0) {
174. const index = elements[0].index;
175. displayGameDetails(labels[index], data, visualizationType, othersItems);
176. }
177. },
178. scales: {
179. y: {
180. beginAtZero: true,
181. ticks: {
182. stepSize: 1,
183. color: 'red',
184. callback: function (value) { return value; }
185. },
186. title: {
187. display: true,
188. color: 'red',
189. text: 'Count'
190. }
191. },
192. x: {
193. title: {
194. display: true,
195. color: 'red',
196. text: visualizationType.charAt(0).toUpperCase() + visualizationType.slice(1)
197. },
198. ticks: {
199. color: 'red',
200. }
201. }
202. }
203. };
204. if (chartType === 'pie') {
205. delete options.scales;
206. }
207. chartInstance = new Chart(ctx, {
208. type: chartType,
209. data: chartData,
210. options: options
211. });
212. });
213. }
214. });
215. </script>
216. </body>
217. </html>

Gornji 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.

## Implementacija osnovnog ponašanja

