SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Projektni zadatak iz predmeta

VIZUALIZACIJA PODATAKA

Ovisnost populacije države o  
Kickstarter projektima

Student: Leon Šumanovac, DRD

Mentor: Denis Ivanović

U Osijeku, lipanj 2023.

SADRŽAJ

[**1. KV1 - Definiranje projektnog zadatka 3**](#_4i6s1smptslt)

[1.1. Projektni zadatak 3](#_wgoiizki7jnh)

[1.2. Podatci 3](#_jjnsdui0w5uv)

[1.3. Obrada podataka 3](#_jfu6ybs7kwrw)

[1.4. Relevantne vrste prikaza za korištene podatke 3](#_1g4rlc1z1va6)

[**2. KV2 - Dizajn vizualizacije podataka. 4**](#_smxrkz6d6wy8)

[2.1. Pitanja na koja vizualizacija daje odgovor 4](#_cqmiylhgneug)

[2.2. Skica vizualizacije podataka 4](#_i61spu8qyxmd)

[2.3. Postojeća rješenja i primjeri 4](#_qmiyijy4jsp8)

[2.4. Prilagodba podataka 4](#_cch1vltulvr)

[2.5. Boje i podatci 4](#_w22k3frzbtb2)

[**3. KV3 - Izrada prototipne vizualizacije podataka 5**](#_k5q372z1w040)

[3.1. Osnovne funkcionalnosti i ponašanja 5](#_yaexlmqqdt0u)

[3.2. Napredne funkcionalnosti i ponašanja: 5](#_y300tbmoybm5)

[3.3. Implementacija osnovnih funkcionalnosti 5](#_g3q3h1rhqzyz)

[3.4. Implementacija osnovnog ponašanja 5](#_itz19pnh67g0)

[**4. KV4 - Izrada konačne vizualizacije podataka 6**](#_sj4kshcbtchw)

[4.1. Implementacija osnovnih funkcionalnosti 6](#_dc19rzmsn9k5)

[4.2. Implementacija osnovnog ponašanja 6](#_exvg8zl4eo3w)

[4.3. Implementacija naprednih funkcionalnosti 6](#_ackz82cvngj2)

[4.4. Implementacija naprednog ponašanja 6](#_le6sbg8zsugz)

[**5. KV5 - Dovršetak projektnog zadatka i pisanje dokumentacije 7**](#_woao7m26bf75)

[5.1. Eventualne preinake i dorade rješenja - u dogovoru s nastavnikom 7](#_116xkgvrmrqh)

[5.2. Izrada dokumenta - projektne dokumentacije 7](#_6zplla712qe7)

[**Literatura 8**](#_ahmk8vkny1f9)

[**Prilog I 9**](#_jefyptb5hazf)

# KV1 - Definiranje projektnog zadatka

## 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: Prikaz Kickstarter projekata u svijetu u odnosu na populaciju

Opis problema: Raširenost Kickstarter projekata je različita po cijelom svijetu.

Opis zadatka: Ovaj projekt će prikazati odnos količine Kickstarter projekata ovisno o populaciji država iz dataset-a o Kickstarter projektima.

Cilj projekta: Vizualizacija odnosa populacije i Kickstarter projekata pojedinih država.

Poveznica na git repozitorij projekta: <https://github.com/Malileon/vizualizacija-kickstarter>

## Podatci

Podaci koji su se koristili za ovaj projekt mogu se naći na sljedećim linkovima:

<https://www.kaggle.com/datasets/ulrikthygepedersen/kickstarter-projects><https://www.kaggle.com/datasets/iamsouravbanerjee/world-population-dataset>

Prvi link sadržava dataset od 375 tisuća kickstarter projekata s osnovnim informacijama o istima, dok drugi dataset sadrži populaciju svih država svijeta u 2022. godini.

Također je korišten i topojson koji se može pronaći na sljedećem linku:

[cdn.jsdelivr.net/npm/world-atlas@2/countries-110m.json](https://cdn.jsdelivr.net/npm/world-atlas@2/countries-110m.json)

## Obrada podataka

Podaci su obrađeni na naćin da su izbačeni podaci koji se ne planiraju koristit (populacija država koja nije iz 2022. godine, većina podataka iz Kickstarter CSV-a poput imena, tražene kolićine novca I sl.). Također je napravljena provjera da su imena država u oba dataseta jednake kako to ne bi pravilo probleme u budućnosti. CSV-ovi su prebaćeni u JSON radi lakšeg rukovanja.

## Relevantne vrste prikaza za korištene podatke

Predlažem korištenje D3-a za vizualizaciju topo-json filea. Za prikaz detalja o Kickstarter projektima preporućujem pie chart i bar chart.

# KV2 - Dizajn vizualizacije podataka.

## Pitanja na koja vizualizacija daje odgovor

Koliko postoji projekata po određenoj državi u svijetu? Ima li povezanosti o količini projekata i populaciji države? Koliko postoji donatora za te projekte?

## A picture containing sketch, drawing, clipart, line art Description automatically generatedSkica vizualizacije podataka

## Postojeća rješenja i primjeri

Cijeli projekt je zasnovan na znanju stećenom na prijašnjim LV-ovima i sa idejom da se LV-ovi spoje u jedan projekt, stoga se većina koda nalazi u izvještajima iz labaratorijskih vježbi.

## Prilagodba podataka

Podaci su prilagođeni kao što je navedeno u 1.3: Obrada podataka.

A black background with white text

Description automatically generated with low confidence

A screen shot of a computer program

Description automatically generated with medium confidenceA screen shot of a computer program

Description automatically generated with low confidence A screenshot of a computer program

Description automatically generated with medium confidence

## Boje i podatci

Boje koje će biti korištene su nijanse zelene skalirane u odnosu na populaciju (tamnije zelena -> veća populacija), te razne boje za prikaz Pie chart-a i bar charta, pošto je potrebno 15 različitih boja za 15 različitih kategorija.

# KV3 - Izrada prototipne vizualizacije podataka

## Osnovne funkcionalnosti i ponašanja

Osnovna funkcionalnost koja će biti napravljena u ovom KV-u je karta i zumiranje. Na hoveru pokazivaća nad državom dobije se prikaz imena države i njene populacije u 2022. godini.

## Napredne funkcionalnosti i ponašanja:

Napredna funkcionalnost koja će biti napravljena u budućnosti je prikaz raznih grafova ovisno o odabranoj državi. Ti grafovi će prikazivati koliko ima kickstarter projekata u različitim kategorijama, te koliko ljudi podržava projekte iz kojih kategorija.

## Implementacija osnovnih funkcionalnosti

<script>

        const width = 1100;

        const height = 750;

        const svg = d3.select('body').append('svg').attr('width', width).attr('height', height);

        const projection = d3.geoMercator().scale(180).translate([width / 2.1, height / 1.43]);

        const path = d3.geoPath(projection);

        const g = svg.append('g');

        var zoom = d3.zoom()

            .scaleExtent([1, 8])

            .on('zoom', function() {

                g.selectAll('path')

                .attr('transform', d3.event.transform);

        });

        var countries = [];     // 22

        var categories = [];    //15

        const kickstarterJson = fetch('data/kickstarter.json').then(response => response.json())

            .then(project => {

                for (let i = 0; i < project.length; i++) {

                    if (!countries.includes(project[i].Country)){

                        countries.push(project[i].Country);

                    }

                    if (!categories.includes(project[i].Category)) {

                        categories.push(project[i].Category);

                    }

                }

                console.log(countries);

                console.log(categories);

                getPopulation();

            });

        var maxPopulation = 0;

        var minPopulation = 0;

        var minCountry;

        var maxCountry;

        var countriesKickstarter = [];

        function getPopulation() {

            const countriesJson = fetch('data/world.json').then(response => response.json())

                .then(country => {

                    minPopulation = country[0].Population;

                    maxPopulation = country[0].Population;

                    for (let i = 0; i < country.length; i++) {

                        if (countries.includes(country[i].Country)) {

                            if (minPopulation > country[i].Population) {

                                minPopulation = country[i].Population;

                                minCountry = country[i].Country;

                            }

                            if (maxPopulation < country[i].Population) {

                                maxPopulation = country[i].Population;

                                maxCountry = country[i].Country;

                            }

                            countriesKickstarter.push(country[i]);

                        }

                        if (i === country.length - 1) {

                            drawMap();

                        }

                    }

                });

        };

        function drawMap() {

            var populationColors = d3.scaleLinear()

                .domain([minPopulation, maxPopulation])

                .range(["#8afa6b", "#1e4214"]);

            d3.json('data/world\_map.json').then(data => {

                const countries = topojson.feature(data, data.objects.countries);

                g.selectAll('path').data(countries.features).enter().append('path').attr('class', 'country').attr('id', d => d.properties.name)

                .attr('d', path).style('stroke', 'darkgray').style('stroke-width', 1).style('fill', function(d) {

                    if (d.properties.kickstarter) {

                        var index = countriesKickstarter.findIndex(object => {

                                if (object.Country === d.properties.name) {

                                    return object

                                }

                            });

                        return populationColors(countriesKickstarter[index].Population)

                    }

                    return "lightgray"

                });

            })

        }

        svg.call(zoom);

    </script>

A picture containing map, text, atlas

Description automatically generated

A picture containing map, text, atlas

Description automatically generated

## Implementacija osnovnog ponašanja

Osnovno ponašanje i funkcionalnost su prikazani u prethodnom koraku.

# KV4 - Izrada konačne vizualizacije podataka

## Implementacija osnovnih funkcionalnosti

Osnovne funkcionalnosti su dovršene i funkcioniraju na očekivani način.

A map of the world with different colored circles

Description automatically generated with low confidence

## Implementacija osnovnog ponašanja

Osnovno ponašanje je testirano i radi na očekivani način.

A picture containing diagram, text, plot, sketch

Description automatically generated

## Implementacija naprednih funkcionalnosti

Prikaz Pie charta se vidi na prethodnim fotografijama. Funckija drawPie koja se poziva na click eventu neke države:

function drawPie() {

            svgPie = d3.select(".wrapper")

                .append("svg")

                .attr("width", pieWidth)

                .attr("height", pieHeight)

                .attr("class", "pieSvg");

            var arc = d3.arc()

                .innerRadius(innerRadius)

                .outerRadius(outerRadius);

            var pie = d3.pie()

                .value(function(d) { return d.value; });

            var pieArcs = svgPie.selectAll("g.pie")

                .data(pie(data))

                .enter()

                .append("g")

                .attr("class", "pie")

                .attr("transform", "translate(" + (pieWidth / 2) + ", " + (pieHeight / 2) +")");

            var projectsSum = 0;

            pieArcs.append("path")

                .attr("fill", function(d) {

                        projectsSum = projectsSum + d.value;

                        return colors[d.data.colorIndex]; })

                .attr("d", arc)

                .on('mouseover', function(d) {

                    var position = d3.event;

                    setTimeout(function () {

                        tooltipPieDiv.style('opacity', .9);

                        tooltipPieDiv.html(d.data.category + '<br/>' + d.value).style('left', position.pageX + 'px')

                            .style('top', (position.pageY - 2) + 'px');

                    }, 4)

                }).on('mouseout', function(d) {

                    tooltipPieDiv.style('opacity', 0);

                });

            var legend = svgPie.selectAll('.path')

                .data(pie(data))

                .enter().filter(function(d) {

                    if (d.index === 0)

                    return d;

                }).append('g')

                .attr('transform', 'translate(' + (pieWidth - 180) + ',' + (pieHeight - 10) + ')')

                .append('text')

                .html('Ukupno projekata: ' + projectsSum);

            drawBar();

        }

## Implementacija naprednog ponašanja

Prikaz Barcharta se vidi na prethodnim fotografijama. Kod funkcije drawBar koja se poziva na kraju prethodnog isječka koda:

d3.select(".barSvg").remove();

            svgBar = d3.select('body')

                .append('svg')

                .attr('width', barWidth - barMargin.left - barMargin.right)

                .attr('height', barHeight - barMargin.top - barMargin.bottom)

                .attr('viewBox', [0, 0, barWidth, barHeight])

                .attr('class', 'barSvg');

            const x = d3.scaleBand()

                .domain(d3.range(data.length))

                .range([barMargin.left, barWidth - barMargin.right])

                .padding(0.1)

            const y = d3.scaleLinear()

                .domain([0, 200000])

                .range([barHeight - barMargin.bottom, barMargin.top])

            svgBar.append('g')

                .attr('fill', 'royalBlue')

                .selectAll('rect')

                .data(data.sort((a, b) => d3.descending(a.backers, b.backers)))

                .join('rect')

                    .attr('x', (d, i) => x(i))

                    .attr('y', (d) => y(d.backers))

                    .attr('height', d => y(0) - y(d.backers))

                    .attr('width', x.bandwidth())

            svgBar.append('g').call(xAxis)

            svgBar.append('g').call(yAxis)

            function xAxis(g) {

                g.attr('transform', `translate(0, ${barHeight - barMargin.bottom})`)

                .call(d3.axisBottom(x).tickFormat(i => data[i].category))

                .selectAll("text")

                .style("text-anchor", "end")

                .attr("dx", "-.8em")

                .attr("dy", ".15em")

                .attr("transform", "rotate(-65)")

                .attr('font-size', '20px');

            }

            function yAxis(g) {

                g.attr('transform', `translate(${barMargin.left}, 0)`)

                .call(d3.axisLeft(y).ticks(null, data.format))

                .attr('font-size', '20px')

            }

            svg.node();

# KV5 - Dovršetak projektnog zadatka i pisanje dokumentacije

## Eventualne preinake i dorade rješenja - u dogovoru s nastavnikom

Pod ovim zadatkom je poboljšan User Experience sa bojanjem barcharta, te je dodan naslov na stranicu, brojač kickstarter projekata po državi i označena država. Također su grafovi namješteni da se nalaze pored karte kako nebi bilo potrebe za scrollanjem.

A picture containing text, map, diagram

Description automatically generated

## Izrada dokumenta - projektne dokumentacije

Projektni zadatak prikazuje kickstarter projekte u odnosu na populaciju pojedinih država svijeta.

Korištene tehnologije se HTML, JS, CSS, D3.

Kako bi se projekt pokrenio, potrebno je imati VS Code sa ekstenzijom Live Server, te klonirati projekt iz git-a / extractat zipani file sa merlin-a. Nakon toga je potrebno otvoriti VS Code u tom folderu, te u donjem desnom kutu kliknuti na „Go Live“. Nakon toga se defaultni web browser otvara sam i vizualizacija podataka je spremna za korištenje.

Nakon što se otvori browser, user vidi naslov projekta i prikaz mape sa obojanih 20 država koje se mogu prikazati na vizualizaciji zbog dataset-a. Države su obojane u nijansama zelene boje ovisno o njihovoj populaciji (tamnija boja oznaćava veću populaciju).

Hoverom na neku od država se vide podaci o istoj, tj. vidi se naziv države i njena populacija u tooltip-u. Klikom na neku od obojanih država se nakon kratkog delay-a zbog količine podataka prikazuju 2 grafa i ime kliknute države.

Pie chart prikazuje raspodijelu Kickstarter projekata po kategorijama u kliknutoj državi. Svaka kategorija je oznaćena svojom bojom koja je jednaka kao i na Barchartu. Hoverom na segment pie charta se prikazuje tooltip imena kategorije i točnog broja projekata u toj kategoriji. Na dnu pie charta je prikazan ukupan broj Kickstarter projekata u toj državi.

Bar chart prikazuje zbroj ljudi koji su podržali projekt iz pojedinih kategorija. Hoverom na pojedini bar se vidi točan broj podržavatelja.

# Literatura

Wikipedia, HTML, dostupno na: [HTML - Wikipedia](https://en.wikipedia.org/wiki/HTML)  
Wikipedia, JS, dostupno na: [JavaScript - Wikipedia](https://en.wikipedia.org/wiki/JavaScript)  
D3.js, Data-Driven Document, dostupno na: [D3.js - Data-Driven Documents (d3js.org)](https://d3js.org/)  
Github, World Atlas, dostupno na: [GitHub - topojson/world-atlas: Pre-built TopoJSON from Natural Earth.](https://github.com/topojson/world-atlas)

# Prilog I

Poveznica na git repozitorij projekta: [Malileon/vizualizacija-kickstarter (github.com)](https://github.com/Malileon/vizualizacija-kickstarter)

Programski kod

<!DOCTYPE html>

<html lang='en'>

<head>

    <meta charset='UTF-8'>

    <meta http-equiv='X-UA-Compatible' content='IE=edge'>

    <meta name='viewport' content='width=device-width, initial-scale=1.0'>

    <script src='http://d3js.org/d3.v5.min.js' charset='utf-8'></script>

    <script src='http://d3js.org/topojson.v1.min.js'></script>

    <link rel="stylesheet" href="style.css">

    <title>Document</title>

</head>

<body>

    <script>

        var title = d3.select('body').append('div').attr('class', 'title').html('Prikaz kickstarter projekata u odnosu na populaciju')

        var countryName = d3.select('body').append('div').attr('class', 'country')

        const width = 1100;

        const height = 750;

        const svg = d3.select('body').append('svg').attr('width', width).attr('height', height)

            .attr('class', 'worldMap');

        const projection = d3.geoMercator().scale(180).translate([width / 2.1, height / 1.43]);

        const path = d3.geoPath(projection);

        const g = svg.append('g');

        var zoom = d3.zoom()

            .scaleExtent([1, 8])

            .on('zoom', function() {

                g.selectAll('path')

                .attr('transform', d3.event.transform);

        });

        var countries = [];     // 22

        var categories = [];    //15

        var populations = [];

        var data = [];

        var colors = ["#FF6384", "#36A2EB ", "#4444FF ",

                      "#4BC0C0 ", "#FF9F40 ", "#FF3333 ",

                      "#FDCB6E ", "#27AE60 ", "#E84A5F ",

                      "#2ECC71 ", "#964B00 ", "#F9690E ",

                      "#9B59B6 ", "#F39C12 ", "#1ABC9C "];

        const kickstarterJson = fetch('data/kickstarter.json').then(response => response.json())

            .then(project => {

                for (let i = 0; i < project.length; i++) {

                    if (!countries.includes(project[i].Country)){

                        countries.push(project[i].Country);

                    }

                    if (!categories.includes(project[i].Category)) {

                        categories.push(project[i].Category);

                        var object = { category: project[i].Category, value: 0, backers: 0, colorIndex: data.length}

                        data.push(object);

                    }

                }

                console.log(countries);

                console.log(categories);

                getPopulation();

            });

        var maxPopulation = 0;

        var minPopulation = 0;

        var minCountry;

        var maxCountry;

        var countriesKickstarter = [];

        var selectedPopulation;

        function getPopulation() {

            const countriesJson = fetch('data/world.json').then(response => response.json())

                .then(country => {

                    minPopulation = country[0].Population;

                    maxPopulation = country[0].Population;

                    for (let i = 0; i < country.length; i++) {

                        if (countries.includes(country[i].Country)) {

                            if (minPopulation > country[i].Population) {

                                minPopulation = country[i].Population;

                                minCountry = country[i].Country;

                            }

                            if (maxPopulation < country[i].Population) {

                                maxPopulation = country[i].Population;

                                maxCountry = country[i].Country;

                            }

                            countriesKickstarter.push(country[i]);

                        }

                        if (i === country.length - 1) {

                            drawMap();

                        }

                    }

                });

        };

        function drawMap() {

            var tooltipDiv = d3.select("body").append("div")

                .attr("class", "tooltip")

                .style("opacity", 0);

            var populationColors = d3.scaleLinear()

                .domain([minPopulation, maxPopulation])

                .range(["#8afa6b", "#1e4214"]);

            d3.json('data/world\_map.json').then(data => {

                const countries = topojson.feature(data, data.objects.countries);

                g.selectAll('path').data(countries.features).enter().append('path').attr('class', 'country').attr('id', d => d.properties.name)

                .attr('d', path).style('stroke', 'darkgray').style('stroke-width', 1).style('fill', function(d) {

                    if (d.properties.kickstarter) {

                        var index = countriesKickstarter.findIndex(object => {

                                if (object.Country === d.properties.name) {

                                    return object

                                }

                            });

                        return populationColors(countriesKickstarter[index].Population)

                    }

                    return "lightgray"

                }).on('mouseover', function(d) {

                    getCountryPopulation(d.properties.name)

                    var position = d3.event;

                    setTimeout(function () {

                        tooltipDiv.style('opacity', .9);

                        tooltipDiv.html(d.properties.name + '<br/> Population: ' + selectedPopulation).style('left', position.pageX + 'px')

                            .style('top', (position.pageY - 28) + 'px');

                    }, 10)

                }).on('mouseout', function(d) {

                    tooltipDiv.style('opacity', 0);

                }).on('click', function(d) {

                    clearData();

                    if (countriesKickstarter.some(e => e.Country === d.properties.name)) {

                        countCategories(d.properties.name);

                        setTimeout(function() {

                            drawPie();

                            countryName.html(d.properties.name)

                        }, 200)

                    }

                    else {

                        countryName.html('');

                    }

                });

            })

        }

        function countCategories(name) {

            var json = fetch('data/kickstarter.json').then(response => response.json())

            .then(project => {

                for (let i = 0; i < project.length; i++) {

                    if (project[i].Country === name) {

                        for (let j = 0; j < data.length; j++) {

                            if (data[j].category === project[i].Category) {

                                data[j].value += 1;

                                data[j].backers += project[i].Backers

                            }

                        }

                    }

                }

            });

        }

        function getCountryPopulation(name) {

            const countriesJson = fetch('data/world.json').then(response => response.json())

                .then(country => {

                    for (let i = 0; i < country.length; i++) {

                        if (country[i].Country === name) {

                            selectedPopulation = country[i].Population;

                        }

                    };

                });

        }

        var pieWidth = 600;

        var pieHeight = 320;

        var outerRadius = 160;

        var innerRadius = 0;

        var tooltipPieDiv = d3.select("body").append("div")

                .attr("class", "tooltipPie")

                .style("opacity", 0);

        var svgPie = d3.select("body")

                .append("div")

                .attr("class", "wrapper")

                .append("svg")

                .attr("width", pieWidth)

                .attr("height", pieHeight)

                .attr("class", "pieSvg");

        function drawPie() {

            svgPie = d3.select(".wrapper")

                .append("svg")

                .attr("width", pieWidth)

                .attr("height", pieHeight)

                .attr("class", "pieSvg");

            var arc = d3.arc()

                .innerRadius(innerRadius)

                .outerRadius(outerRadius);

            var pie = d3.pie()

                .value(function(d) { return d.value; });

            var pieArcs = svgPie.selectAll("g.pie")

                .data(pie(data))

                .enter()

                .append("g")

                .attr("class", "pie")

                .attr("transform", "translate(" + (pieWidth / 2) + ", " + (pieHeight / 2) +")");

            var projectsSum = 0;

            pieArcs.append("path")

                .attr("fill", function(d) {

                        projectsSum = projectsSum + d.value;

                        return colors[d.data.colorIndex]; })

                .attr("d", arc)

                .on('mouseover', function(d) {

                    var position = d3.event;

                    setTimeout(function () {

                        tooltipPieDiv.style('opacity', .9);

                        tooltipPieDiv.html(d.data.category + '<br/>' + d.value).style('left', position.pageX + 'px')

                            .style('top', (position.pageY - 2) + 'px');

                    }, 4)

                }).on('mouseout', function(d) {

                    tooltipPieDiv.style('opacity', 0);

                });

            var legend = svgPie.selectAll('.path')

                .data(pie(data))

                .enter().filter(function(d) {

                    if (d.index === 0)

                    return d;

                }).append('g')

                .attr('transform', 'translate(' + (pieWidth - 180) + ',' + (pieHeight - 10) + ')')

                .append('text')

                .html('Ukupno projekata: ' + projectsSum);

            drawBar();

        }

        var barWidth = 700;

        var barHeight = 700;

        var barMargin = { top: 20, bottom: 160, left: 100, right: 50 };

        var svgBar = d3.select('.wrapper')

                .append('svg')

                .attr('width', barWidth - barMargin.left - barMargin.right)

                .attr('height', barHeight - barMargin.top - barMargin.bottom)

                .attr('viewBox', [0, 0, barWidth, barHeight])

                .attr('class', 'barSvg');

        function drawBar() {

            svgBar = d3.select('.wrapper')

                .append('svg')

                .attr('width', barWidth - barMargin.left - barMargin.right)

                .attr('height', barHeight - barMargin.top - barMargin.bottom)

                .attr('viewBox', [0, 0, barWidth, barHeight])

                .attr('class', 'barSvg');

            const x = d3.scaleBand()

                .domain(d3.range(data.length))

                .range([barMargin.left, barWidth - barMargin.right])

                .padding(0.1)

            const y = d3.scaleLinear()

                .domain([0, getMaxBackers()])

                .range([barHeight - barMargin.bottom, barMargin.top])

            svgBar.append('g')

                .selectAll('rect')

                .data(data.sort((a, b) => d3.descending(a.backers, b.backers)))

                .join('rect')

                .attr('x', (d, i) => x(i))

                .attr('y', (d) => y(d.backers))

                .attr('height', d => y(0) - y(d.backers))

                .attr('width', x.bandwidth())

                .attr('fill', function(d) { return colors[d.colorIndex]; })

                .on('mouseover', function(d) {

                    var position = d3.event;

                    setTimeout(function () {

                        tooltipPieDiv.style('opacity', .9);

                        tooltipPieDiv.html(d.category + '<br/>' + d.backers).style('left', position.pageX + 'px')

                            .style('top', (position.pageY - 2) + 'px');

                    }, 4)

                }).on('mouseout', function(d) {

                    tooltipPieDiv.style('opacity', 0);

                });

                svgBar.append('text')

                    .attr('class', 'yos')

                    .attr('text-anchor', 'end')

                    .attr('transform', 'rotate(-90)')

                    .text('Broj podržavaća (donatora)')

            svgBar.append('g').call(xAxis)

            svgBar.append('g').call(yAxis)

            function xAxis(g) {

                g.attr('transform', `translate(0, ${barHeight - barMargin.bottom})`)

                .call(d3.axisBottom(x).tickFormat(i => data[i].category))

                .selectAll("text")

                .style("text-anchor", "end")

                .attr("dx", "-.8em")

                .attr("dy", ".15em")

                .attr("transform", "rotate(-65)")

                .attr('font-size', '20px');

            }

            function yAxis(g) {

                g.attr('transform', `translate(${barMargin.left}, 0)`)

                .call(d3.axisLeft(y).ticks(null, data.format))

                .attr('font-size', '20px')

            }

            function getMaxBackers() {

                var firstNumber, maxBackers = data[0].backers;

                data.forEach(element => {

                    if (element.backers > maxBackers) maxBackers = element.backers;

                })

                firstNumber = parseInt(maxBackers.toString()[0]) + 1;

                for (var i = 1; i < maxBackers.toString().length; i++) {

                    firstNumber \*= 10;

                }

                return firstNumber;

            }

            svg.node();

        }

        function clearData() {

            d3.select(".pieSvg").remove();

            d3.select(".barSvg").remove();

            for (let i = 0; i < data.length; i++) {

                data[i].value = 0;

                data[i].backers = 0;

            }

        }

        svg.call(zoom);

    </script>

</body>

</html>