SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Domaća zadaća iz predmeta

VIZUALIZACIJA PODATAKA

Vizualizacija Apple dionica od 2000. – 2023.

Student: Tea Krčmar, računarstvo DRD

U Osijeku, srpanj, 2023.

**SADRŽAJ**

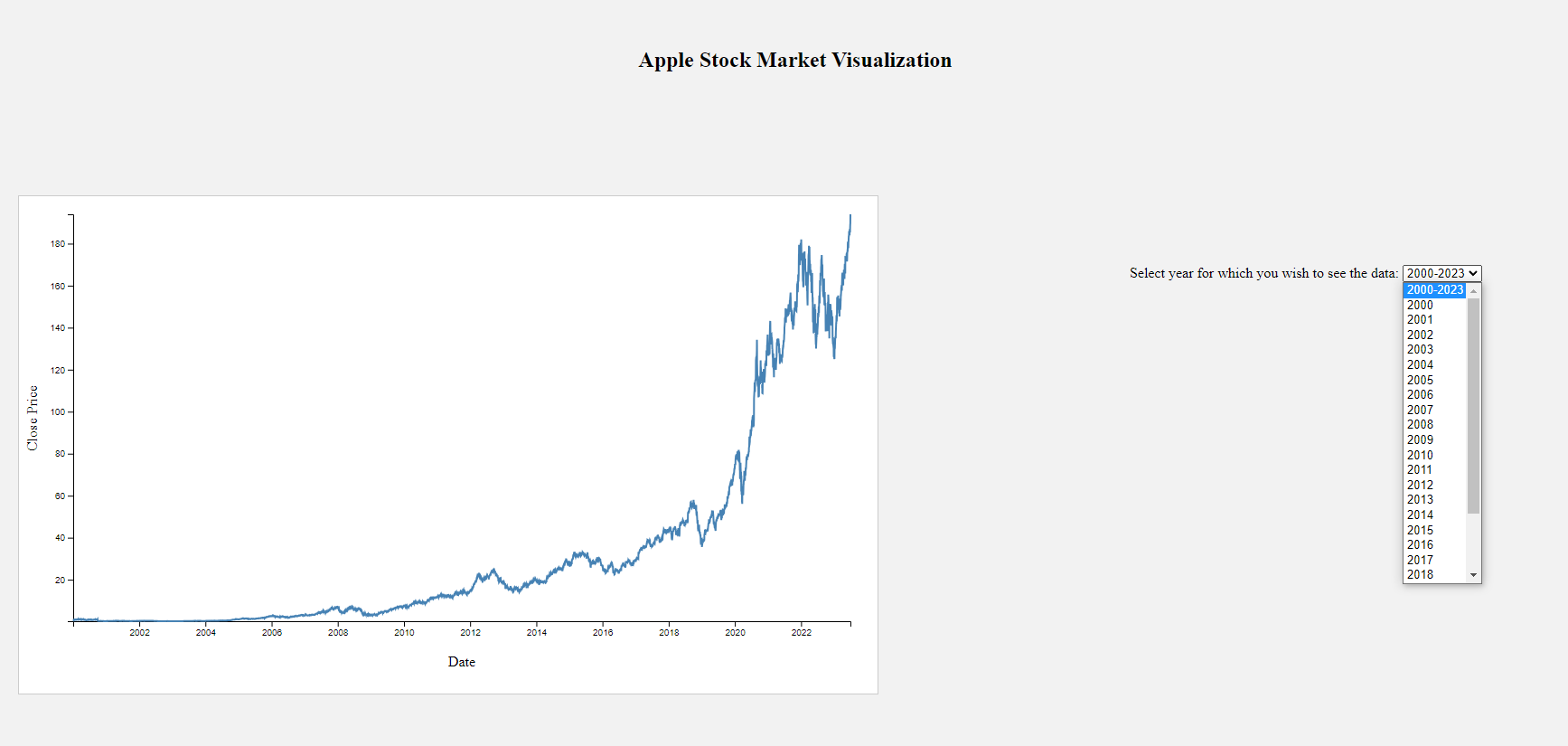
[**ZADATAK** 1](#_Toc139298797)

[**SKUP PODATAKA** 1](#_Toc139298798)

[**RJEŠENJE** 3](#_Toc139298799)

# **ZADATAK**

Zadatak ove domaće zadaće bio je napraviti jednostavnu vizualizaciju dionica tvrtke Apple u zadnje 23 godine. Korisniku je omogućen odabir pojedine godine za pregled, kao i svih zajedno. Pored toga, prelaskom miša na liniji grafa prikazuju se vrijednosti u dolarima toga dana, kao i sami datum.

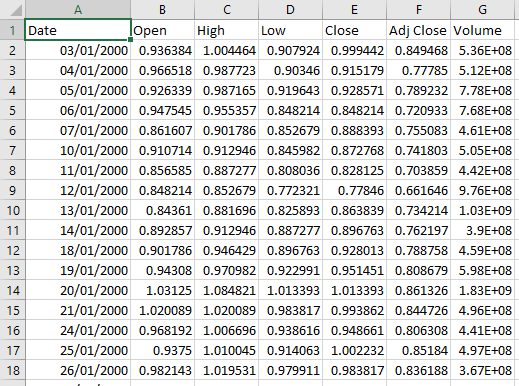


*Slika 1. Prikaz izgleda web stranice.*

Github link: <https://github.com/teakrcmar/Homework_VP.git>

# **SKUP PODATAKA**

Podaci skupljeni za prikaz povućeni su od Yahoo Finance API-ja unatrag 23 godine i spremljeni u *csv* datoteku. Set podataka sadrži 7 stupaca različitih vrijednosti. Prvi stupac predstavlja datume za koje su dostupni podaci o cijenama dionica, drugi sadrži cijene početka trgovanja za određeni datum, tj. cijene po kojima je dionica bila dostupna na početku tog dana. Treći stupac sadrži najviše cijene dostignute tijekom tog dana, četvrti najniže cijene dostignute tijekom tog dana, peti sadrži cijene zatvaranja za taj dan, tj. cijene po kojima je dionica završila trgovanje tog dana, šesti vrijednost prilagođena cijena zatvaranja koja uzima u obzir bilo kakve korekcije poput isplata dividendi ili dionica i sedmi sadrži informacije o ukupnom volumenu trgovanja, tj. koliko je dionica prodano tog dana. Za prikaz vizualizacije ovoga zadatka, korišten je peti stupac (eng. *Close*).



*Slika 2. Prikaz skupa podataka.*

Kod za prikupljanje podataka s Yahoo Finance API-ja prikazan je ispod.

import yfinance as yf

import datetime

# Stock Symbols

symbols = ['AAPL']

# Set time frame (2000-present)

end\_date = datetime.datetime.now().strftime('%Y-%m-%d')

start\_date = '2000-01-01'  # Specify the start date

# Downloading data and saving them into separate CSV files

for symbol in symbols:

    data = yf.download(symbol, start=start\_date, end=end\_date)

    filename = f"{symbol}\_Stock\_Data.csv"

    data.to\_csv(filename)

    print(f"Data for {symbol} are saved in {filename}.")

# **RJEŠENJE**

<!DOCTYPE html>

<html>

<head>

  <title>Apple Stock Market Visualization</title>

  <link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

  <div class="container">

    <h1 class="title">Apple Stock Market Visualization</h1>

    <div class="chart-container">

      <div id="stock-chart"></div>

</div>

  </div>

  <script src="https://d3js.org/d3.v7.js"></script>

  <script src="script.js"></script>

</body>

</html>

**Css kod:**

body {

  margin: 0;

  padding: 0;

  background-color: #f2f2f2;

}

.chart-container {

  display: flex;

  justify-content: flex-start;

  margin-top: 20px;

}

.container {

  display: flex;

  flex-direction: column;

}

#stock-chart {

  width: 900px;

  height: 500px;

  margin-left: 100px;

  margin-top: 100px;

  margin-right: 100px;

}

.title {

  font-size: 24px;

  font-weight: bold;

  text-align: center;

  margin-top: 60px;

}

svg {

  width: 950px;

  height: 550px;

  background-color: white;

  border: 1px solid #ccc;

}

.line {

  fill: none;

  stroke: steelblue;

  stroke-width: 2px;

}

.hover-overlay {

  fill: none;

  pointer-events: all;

}

.tooltip {

  position: absolute;

  background-color: #f9f9f9;

  border: 1px solid #ccc;

  border-radius: 4px;

  padding: 8px;

  font-size: 12px;

}

.title {

  font-size: 24px;

  font-weight: bold;

  text-align: center;

  margin-top: 60px;

}

svg {

  width: 950px;

  height: 550px;

  background-color: white;

  border: 1px solid #ccc;

}

.line {

  fill: none;

  stroke: steelblue;

  stroke-width: 2px;

}

.hover-overlay {

  fill: none;

  pointer-events: all;

}

.tooltip {

  position: absolute;

  background-color: #f9f9f9;

  border: 1px solid #ccc;

  border-radius: 4px;

  padding: 8px;

  font-size: 12px;

}

**Javascript kod:**

window.addEventListener('DOMContentLoaded', createStockChart);

function createStockChart() {

  // Load the data from the CSV file

  d3.csv("https://raw.githubusercontent.com/teakrcmar/Homework\_VP/main/AAPL\_Stock\_Data.csv").then(function(data) {

    // Convert the date strings to JavaScript Date objects

    var parseDate = d3.timeParse("%Y-%m-%d");

    data.forEach(function(d) {

      d.Date = parseDate(d['Date']);

      d.Close = +d['Close'];

    });

    // Set up the SVG container and dimensions

    var margin = { top: 20, right: 30, bottom: 30, left: 60 };

    var width = 950 - margin.left - margin.right;

    var height = 500 - margin.top - margin.bottom;

    // Create the SVG element

    var svg = d3

      .select("#stock-chart")

      .append("svg")

      .attr("width", width + margin.left + margin.right)

      .attr("height", height + margin.top + margin.bottom)

      .append("g")

      .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

    // Set the scales for the x and y axes

    var x = d3

      .scaleTime()

      .domain(d3.extent(data, function(d) { return d.Date; }))

      .range([0, width]);

    var y = d3

      .scaleLinear()

      .domain(d3.extent(data, function(d) { return d.Close; }))

      .range([height, 0]);

    // Create the x and y axes

    var xAxis = d3.axisBottom(x);

    var yAxis = d3.axisLeft(y);

    // Add the x and y axes to the SVG element

    svg.append("g")

      .attr("transform", "translate(0," + height + ")")

      .call(xAxis);

    svg.append("g")

      .call(yAxis);

    // Add x-axis label

    svg.append("text")

      .attr("class", "axis-label")

      .attr("text-anchor", "middle")

      .attr("x", width / 2)

      .attr("y", height + margin.bottom + 20)

      .text("Date");

    // Add y-axis label

    svg.append("text")

      .attr("class", "axis-label")

      .attr("text-anchor", "middle")

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

      .attr("x", -height / 2)

      .attr("y", -margin.left + 20)

      .text("Close Price");

    // Create the line for the stock prices

    var line = d3

      .line()

      .x(function(d) { return x(d.Date); })

      .y(function(d) { return y(d.Close); });

    // Add the line to the SVG element

    svg.append("path")

      .datum(data)

      .attr("class", "line")

      .attr("d", line);

    // Add the hover overlay

    var hoverOverlay = svg.append("rect")

      .attr("class", "hover-overlay")

      .attr("width", width)

      .attr("height", height)

      .style("opacity", 0);

    // Add the tooltip

    var tooltip = d3.select("#stock-chart")

      .append("div")

      .attr("class", "tooltip")

      .style("opacity", 0);

    // Add event handlers for hover interactions

    hoverOverlay.on("mousemove", handleMouseMove)

      .on("mouseout", handleMouseOut);

    // Define the mousemove event handler

    function handleMouseMove(event) {

      var bisectDate = d3.bisector(function(d) { return d.Date; }).left;

      var mouseX = d3.pointer(event)[0];

      var invertedX = x.invert(mouseX);

      var index = bisectDate(data, invertedX, 1);

      var d0 = data[index - 1];

      var d1 = data[index];

      var d = invertedX - d0.Date > d1.Date - invertedX ? d1 : d0;

      // Show tooltip with value at the corresponding position

      tooltip.style("opacity", 1)

        .style("left", (event.pageX + 10) + "px")

        .style("top", (event.pageY - 25) + "px")

        .html("<strong>Date:</strong> " + d.Date.toDateString() + "<br><strong>Close:</strong> $" + d.Close.toFixed(2));

    }

    // Define the mouseout event handler

    function handleMouseOut() {

      tooltip.style("opacity", 0);

    }

    // Add the year selection dropdown

    var yearOptions = ["2000-2023", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019", "2020", "2021", "2022", "2023"];

    var yearSelect = d3.select("#stock-chart")

      .append("div")

      .text("Select year for which you wish to see the data: ")

      .attr("class", "year-select")

      .style("position", "absolute")

      .style("top", "300px")

      .style("right", "200px")

      .append("select")

      .on("change", function() {

        var selectedYear = this.value;

        if (selectedYear === "2000-2023") {

          updateChart(data);

        } else {

          var filteredData = data.filter(function(d) {

            return d.Date.getFullYear() === +selectedYear;

          });

          updateChart(filteredData);

        }

      });

    yearSelect.selectAll("option")

      .data(yearOptions)

      .enter()

      .append("option")

      .attr("value", function(d) { return d; })

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

    // Function to update the chart with new data

    function updateChart(data) {

      // Update the scales with the new data

      x.domain(d3.extent(data, function(d) { return d.Date; }));

      y.domain(d3.extent(data, function(d) { return d.Close; }));

      // Update the x and y axes

      svg.select(".x-axis")

        .transition()

        .duration(500)

        .call(xAxis);

      svg.select(".y-axis")

        .transition()

        .duration(500)

        .call(yAxis);

      // Update the line

      svg.select(".line")

        .datum(data)

        .transition()

        .duration(500)

        .attr("d", line);

    }

  });

}

    // Add x-axis label

    svg.append("text")

      .attr("class", "axis-label")

      .attr("text-anchor", "middle")

      .attr("x", width / 2)

      .attr("y", height + margin.bottom + 20)

      .text("Date");

    // Add y-axis label

    svg.append("text")

      .attr("class", "axis-label")

      .attr("text-anchor", "middle")

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

      .attr("x", -height / 2)

      .attr("y", -margin.left + 20)

      .text("Close Price");

    // Create the line for the stock prices

    var line = d3

      .line()

      .x(function(d) { return x(d.Date); })

      .y(function(d) { return y(d.Close); });

    // Add the line to the SVG element

    svg.append("path")

      .datum(data)

      .attr("class", "line")

      .attr("d", line);

    // Add the hover overlay

    var hoverOverlay = svg.append("rect")

      .attr("class", "hover-overlay")

      .attr("width", width)

      .attr("height", height)

      .style("opacity", 0);

    // Add the tooltip

    var tooltip = d3.select("#stock-chart")

      .append("div")

      .attr("class", "tooltip")

      .style("opacity", 0);

    // Add event handlers for hover interactions

    hoverOverlay.on("mousemove", handleMouseMove)

      .on("mouseout", handleMouseOut);

    // Define the mousemove event handler

    function handleMouseMove(event) {

      var bisectDate = d3.bisector(function(d) { return d.Date; }).left;

      var mouseX = d3.pointer(event)[0];

      var invertedX = x.invert(mouseX);

      var index = bisectDate(data, invertedX, 1);

      var d0 = data[index - 1];

      var d1 = data[index];

      var d = invertedX - d0.Date > d1.Date - invertedX ? d1 : d0;

      // Show tooltip with value at the corresponding position

      tooltip.style("opacity", 1)

        .style("left", (event.pageX + 10) + "px")

        .style("top", (event.pageY - 25) + "px")

        .html("<strong>Date:</strong> " + d.Date.toDateString() + "<br><strong>Close:</strong> $" + d.Close.toFixed(2));

    }

    // Define the mouseout event handler

    function handleMouseOut() {

      tooltip.style("opacity", 0);

    }

    // Add the year selection dropdown

    var yearOptions = ["2000-2023", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019", "2020", "2021", "2022", "2023"];

    var yearSelect = d3.select("#stock-chart")

      .append("div")

      .text("Select year for which you wish to see the data: ")

      .attr("class", "year-select")

      .style("position", "absolute")

      .style("top", "300px")

      .style("right", "200px")

      .append("select")

      .on("change", function() {

        var selectedYear = this.value;

        if (selectedYear === "2000-2023") {

          updateChart(data);

        } else {

          var filteredData = data.filter(function(d) {

            return d.Date.getFullYear() === +selectedYear;

          });

          updateChart(filteredData);

        }

      });

    yearSelect.selectAll("option")

      .data(yearOptions)

      .enter()

      .append("option")

      .attr("value", function(d) { return d; })

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

    // Function to update the chart with new data

    function updateChart(data) {

      // Update the scales with the new data

      x.domain(d3.extent(data, function(d) { return d.Date; }));

      y.domain(d3.extent(data, function(d) { return d.Close; }));

      // Update the x and y axes

      svg.select(".x-axis")

        .transition()

        .duration(500)

        .call(xAxis);

      svg.select(".y-axis")

        .transition()

        .duration(500)

        .call(yAxis);

      // Update the line

      svg.select(".line")

        .datum(data)

        .transition()

        .duration(500)

        .attr("d", line);

    }

  });

}

// Define the mousemove event handler

    function handleMouseMove(event) {

      var bisectDate = d3.bisector(function(d) { return d.Date; }).left;

      var mouseX = d3.pointer(event)[0];

      var invertedX = x.invert(mouseX);

      var index = bisectDate(data, invertedX, 1);

      var d0 = data[index - 1];

      var d1 = data[index];

      var d = invertedX - d0.Date > d1.Date - invertedX ? d1 : d0;

      // Show tooltip with value at the corresponding position

      tooltip.style("opacity", 1)

        .style("left", (event.pageX + 10) + "px")

        .style("top", (event.pageY - 25) + "px")

        .html("<strong>Date:</strong> " + d.Date.toDateString() + "<br><strong>Close:</strong> $" + d.Close.toFixed(2));

    }

    // Define the mouseout event handler

    function handleMouseOut() {

      tooltip.style("opacity", 0);

    }

    // Add the year selection dropdown

    var yearOptions = ["2000-2023", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019", "2020", "2021", "2022", "2023"];

    var yearSelect = d3.select("#stock-chart")

      .append("div")

      .text("Select year for which you wish to see the data: ")

      .attr("class", "year-select")

      .style("position", "absolute")

      .style("top", "300px")

      .style("right", "200px")

      .append("select")

      .on("change", function() {

        var selectedYear = this.value;

        if (selectedYear === "2000-2023") {

          updateChart(data);

        } else {

          var filteredData = data.filter(function(d) {

            return d.Date.getFullYear() === +selectedYear;

          });

          updateChart(filteredData);

        }

      });

    yearSelect.selectAll("option")

      .data(yearOptions)

      .enter()

      .append("option")

      .attr("value", function(d) { return d; })

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

    // Function to update the chart with new data

    function updateChart(data) {

      // Update the scales with the new data

      x.domain(d3.extent(data, function(d) { return d.Date; }));

      y.domain(d3.extent(data, function(d) { return d.Close; }));

      // Update the x and y axes

      svg.select(".x-axis")

        .transition()

        .duration(500)

        .call(xAxis);

      svg.select(".y-axis")

        .transition()

        .duration(500)

        .call(yAxis);

      // Update the line

      svg.select(".line")

        .datum(data)

        .transition()

        .duration(500)

        .attr("d", line);

    }

  });

}

    yearSelect.selectAll("option")

      .data(yearOptions)

      .enter()

      .append("option")

      .attr("value", function(d) { return d; })

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

    // Function to update the chart with new data

    function updateChart(data) {

      // Update the scales with the new data

      x.domain(d3.extent(data, function(d) { return d.Date; }));

      y.domain(d3.extent(data, function(d) { return d.Close; }));

      // Update the x and y axes

      svg.select(".x-axis")

        .transition()

        .duration(500)

        .call(xAxis);

      svg.select(".y-axis")

        .transition()

        .duration(500)

        .call(yAxis);

      // Update the line

      svg.select(".line")

        .datum(data)

        .transition()

        .duration(500)

        .attr("d", line);

    }

  });

}