Dokumentacija

D – Wiz

Izrada informacijskih projekata

Autori:

Marija Kantolić – Autorizacija, Izrada „Blog“ i „Home“

Josip Tadić – Izrada Grafikona(Chart-ova)

Svrha projekta (zadatak): Izrada web platforme za vizualiziranje podataka putem grafikona

Korištene tehnologije: HTML, CSS, JavaScript (React.js), Firebase, Bootstrap, Recharts

Source code: <https://github.com/JosipTadic/IIP2021>

Kratki opis:

Home – Stranica sa obavijestima i svim važnim informacijama

Blog – Stranica na kojoj se objavljuju novosti

Charts – Stranica na kojoj se izrađuju grafikoni sa 1,2 ili 3 vrijednosti, mogućnost izrade 4 osnovne vrste grafikona: Line, Area, Bar, Scatter, te dodatni Composed chart koje sastavljen kao kombinacija osnovnih vrsta grafikona.

Opis funkcionalnosti i značajki:

Cijela aplikacija izvodi se kao „Single page application“ kroz datoteku index.js.

Index.js:

Redovi(Vrijednosti):

  const [rows, setRows] = useState([{

      id: nanoid(),

      a: '',

      b: '',

      c: '',

      labelName: '',

  }]);

Inicijalizacija redova pomoću useState „Hook-a“. Jedan red predstavlja jedan podatak na grafikonu, koji ima 1,2 ili 3 vrijednosti + Ime. Id reda se dodjeljuje putem nanoId library-a.

const increaseRows = () => {

    const newRow = {

      id: nanoid(),

      a: '',

      b: '',

      c: '',

      labelName: '',

    }

    setRows(rows.concat(newRow));

  }

Funkcija koja dodaje nove redove.

<Col md={{ offset: 2 }}><ButtonComp variant={"primary"} text='Add new row' onClick={increaseRows}/></Col>

Poziv funkcije za dodavanje redova.

const ButtonComp = (props) => {

    return(

      <Button block {...props}>{props.text}</Button>

    )

  }

Gumb je destrukturiran unutar „ButtonComp“ komponente.

  const modifyRow = (id, axis, newValue) => {

    setRows(rows.map(row => {

      if (row.id === id) {

        row[axis] = newValue;

      }

      return row;

    }));

  }

  const deleteRow = (id) => {

    setRows(rows.filter((row) => {

      return row.id !== id;

    }))

  }

  const deleteAll = () => {

    setRows([]);

  }

Funkcije za izmjenu, i brisanje (pojedinog i svih redova)

<td className="table-icons" onClick={deleteRow}><BsFillTrashFill  className="react-icons"/></td>

Poziv funkcije za brisanje pojedinog reda.

<Col md={{ offset: 0 }}><ButtonComp variant={"danger"} text='Delete Table'  onClick={deleteAll} /></Col>

Poziv funkcije za brisanje svih redova.

<Container className="fixed scroll">

   <TableComp rows={rows} deleteRow={deleteRow} modifyRow={modifyRow} chartState={chartState}/>

</Container>

Tablica redova.

const TableComp = ({rows, deleteRow, modifyRow, chartState}) => {

    if(rows < 1){

      return(

        <h2 className="center">Press "add new row" to start visualizing!</h2>

      )

    }

    switch(chartState.numberOfVariables) {

      case 1:

        return(

          <Table striped bordered hover size="sm">

            <tbody>

              <tr>

                <th>1st Value</th>

                <th>Name</th>

                <th className="table-icons"></th>

              </tr>{

                rows.map(row => <TableRow key={row.id} deleteRow={() => deleteRow(row.id)} chartState={chartState}

                modifyRow={(axis, newValue) => modifyRow(row.id, axis, newValue)} row={row}/>)

              }

            </tbody>

          </Table>

        )

„TableComp“ komponenta u kojem je fixirano zaglavlje tablice, te su redovi dodatno dekomponirani u komponentu „TableRow“, kako bi se mogli mapirati skupa sa točnim pozivima funkcija. Primjer je za red sa tri vrijednosti.

const TableRow = ({deleteRow, modifyRow, chartState, row}) => {

  switch (chartState.numberOfVariables){

    case 1:

      return(

        <tr>

          <td><input className="input-style" type="number" value={row.a} placeholder="enter 1st value" onChange={(e) => modifyRow('a', e.target.value)}/></td>

          <td><input className="input-style" type="text" value={row.labelName} placeholder="enter name" onChange={(e) => modifyRow('labelName', e.target.value)}/></td>

          <td className="table-icons" onClick={deleteRow}><BsFillTrashFill  className="react-icons"/></td>

        </tr>

      )

„TableRow“ komponenta, s pozivima funkcija.

Parametri grafikona:

  const [params, setParams] = useState(

    [

        {

          id: 1,

          color: "#111222",

          legendName: 'a',

          lineType: "monotone",

          strokeDash: 0,

          strokeWidth: 1,

          strokeOpacity: 1,

        },

        {

          id: 2,

          color: "#222555",

          legendName: 'b',

          lineType: "monotone",

          strokeDash: 0,

          strokeWidth: 1,

          strokeOpacity: 1,

        },

        {

          id: 3,

          color: "#333666",

          legendName: 'c',

          lineType: "monotone",

          strokeDash: 0,

          strokeWidth: 1,

          strokeOpacity: 1,

        }

    ]

  )

Inicijalizacija Parametara i postavljanje default stanja. Parametri služe kako bi se na grafikonima mogli promijeniti: Širina, Boja, Ime na legendi, Vrstu linije, Crtice, Širinu i prozirnost.

  const modifyParams = (id, selectedParam, newValue) => {

    setParams(params.map(param => {

      if (param.id === id) {

        param[selectedParam] = newValue;

      }

      return param;

    }));

  }

Funkcija za modifikaciju parametara grafikona

  const [chartState, setChartState] = useState({

    numberOfVariables: 3,

    dash: true,

    type: true,

    width: true,

    opacity: true,

    colorChoice: true

  })

S obzirom da svaki grafikon sadrži svoje parametre(nemaju svi iste), „chartState“ čuva parametre od odgovarajućih grafikona (npr. Bar Chart nema mogućnost mijenjanja izbora linije). Nakon klika na link željenog grafikona poziva se funkcija odgovorna za svaki grafikon i broj vrijednosti. Linkovi na grafikone se nalaze u komponentama „Cards.js“ i „SubNavBar.js“, te su im proslijeđene refence na funkcije koje postavljaju „chartState“.

<Nav.Link as={Link}  to="/three/threescatter" onClick={()=>chartsSetter({nov:3,dash:false,type:false,width:true,opacity:true,color:true})}><b>Scatter Chart</b></Nav.Link>

Primjer linka na „Scatter chart“ sa 3 varijable. Poziv funkcije koja postavlja „chartState“.

const chartsSetter = ({nov,dash,type,width,opacity,color}) =>

   setChartState({

     numberOfVariables: nov,

     dash: dash,

     type: type,

     width: width,

     opacity: opacity,

     colorChoice: color

   })

Funkcija koja postavlja „chartState“.

<ParameterCustomization params={params} modifyParams={modifyParams} chartState={chartState}/>

Komponenta za modificiranje parametara.

const ParameterCustomization = ({params, modifyParams, chartState}) => {

    return(

      <Container fluid>

      <Row>

        <Col>

        {

          params.filter(param => param.id <= chartState.numberOfVariables).map(param => <InputComp key={param.id}

            modifyParams={modifyParams} param={param}/>)

        }

Komponenta „InputComp“ za mijenjanje boja, unutar komponente „ParameterCustomization“, dodatno su destrukturirane komponente za ostale parametre(WidthSelect,OpacitySelect,LegendName,DashSelect) na isti način kao i primjer „InputComp“.

const InputComp = ({modifyParams, param}) => {

    return(

      <>

        <input type="color" name="color"

        onChange={e => modifyParams(param.id, 'color', e.target.value)}/>

      </>

    )

}

Komponenta „InputComp“ omogućuje mijenjanje boja na grafikonu.

Download grafikona:

var handleSaveClick = (idOfChart) => {

    domtoimage.toBlob(document.getElementById(idOfChart))

       .then(function (blob) {

          fileDownload(blob, 'dom-to-image.png');

       });

   }

Funkcija za download grafikona. Ostvarena uz pomoć 2 dodatna library-a: domtoimage i fileDownload

<div id="blob1">

Za download potrebno je dodijeliti id grafikonu.

<Col md={{ offset: 0 }}><ButtonComp variant={"primary"} text='Download Chart' onClick={() => handleSaveClick("blob1")}/></Col>

Poziv funkcije za download „handleSaveClick“ i proslijeđivanje id-a grafikona. Bitno je da se umjesto poziva funkcije, pozove referenca, u suprotnom bi svaki re-render Button komponente pozvao funkciju, a u ovom slučaju to ne želimo.

Grafikoni:

<Container className="marginTop">

              <ResponsiveContainer className="justify-content-md-center">

                <Row>

                  <LineChart width={1400} height={350} data={rows}

                  margin={{ top: 5, right: 100, left: 5, bottom: 5 }}>

                  <CartesianGrid strokeDasharray="3 3" />

                  <XAxis dataKey="labelName"/>

                  <YAxis />

                  <Tooltip />

                  <Legend />

                  <Line strokeWidth={params[0].strokeWidth} name={params[0].legendName}

                   strokeDasharray={params[0].strokeDash} type={params[0].lineType} dataKey="a" stroke={params[0].color}/>

                  <Line strokeWidth={params[1].strokeWidth} strokeLinecap={params[1].strokeLinecap} name={params[1].legendName}

                  strokeDasharray={params[1].strokeDash} type={params[1].lineType} dataKey="b" stroke={params[1].color} />

                  <Line strokeWidth={params[2].strokeWidth} strokeLinecap={params[2].strokeLinecap} name={params[2].legendName}

                  strokeDasharray={params[2].strokeDash} type={params[2].lineType} dataKey="c" stroke={params[2].color} />

                  </LineChart>

                </Row>

              </ResponsiveContainer>

            </Container>

Komponenta „LineChart“ iz „Recharts“ library-a odgovorna je za renderiranje u ovom primjeru „Linijskog grafikona“, drugi grafikoni ostvareni su na sličan način. U „LineChart“ komponentu podaci se proslijeđuju kroz „props-e“ (data={rows}), koji se mijenjaju na prije definiran način.

„CartesianGrid“ - komponenta za renderiranje pozadinske mreže na grafikonima.

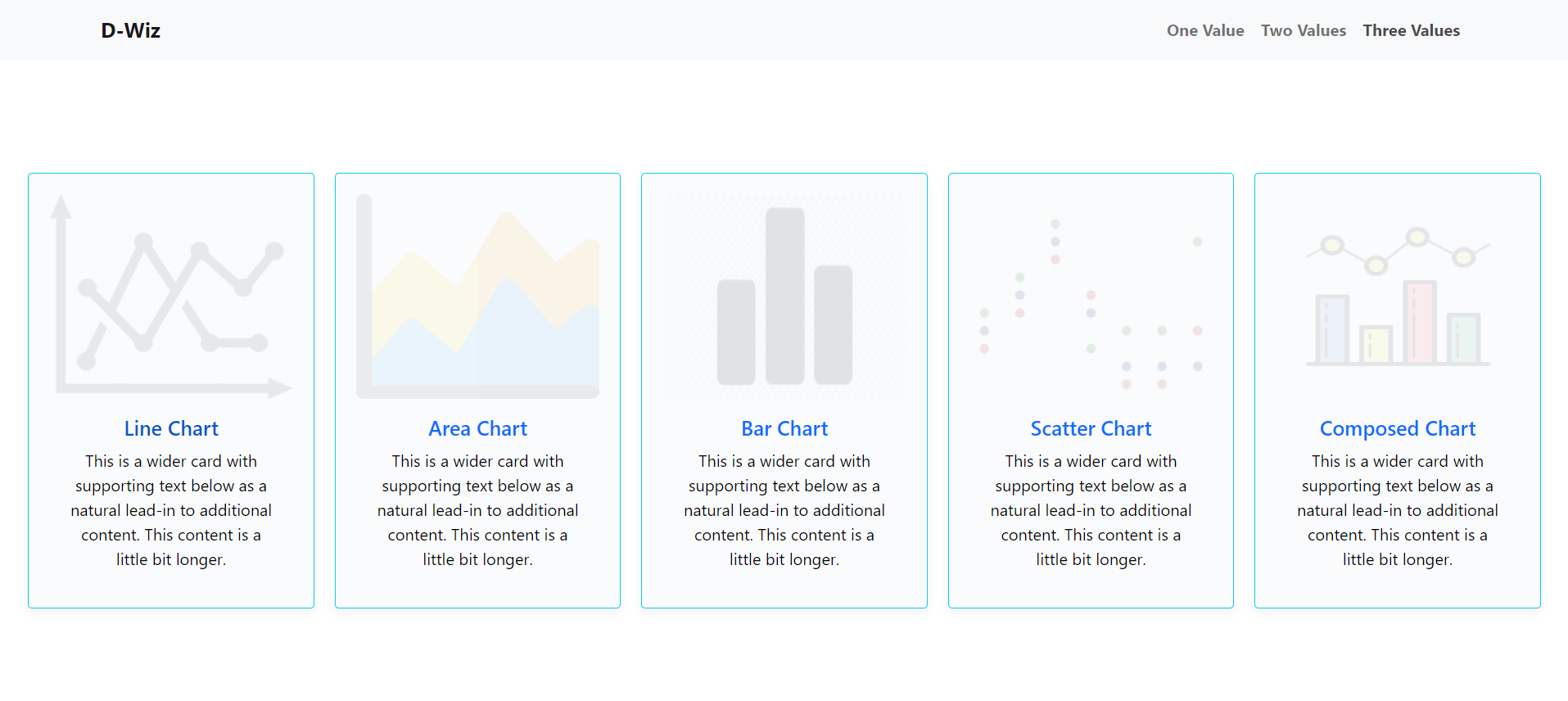
„XAxis“ i „YAxis“ - komponente za renderiranje X i Y osi.

„Legend“ – komponenta za legendu, nalazi se ispod grafikona

„Line“ – komponenta koja renderira liniju na linijskom grafikonu, u ovom slučaju radi se o linijskom grafikonu s 3 linije. Putem „dataKey“ parametra proslijeđuju se vrijednosti podataka. Ostali parametri služe za prilagođavanje grafikona po želji.

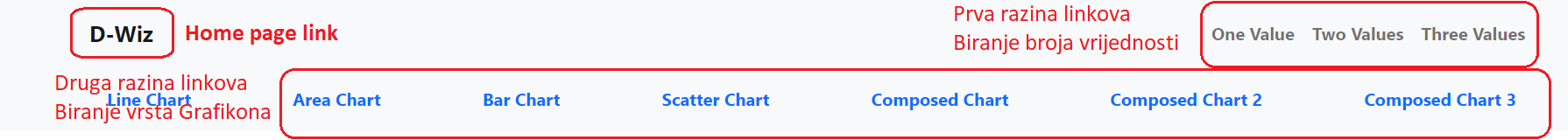
Opis UI:

Cards(Prvi linkovi na grafikone):

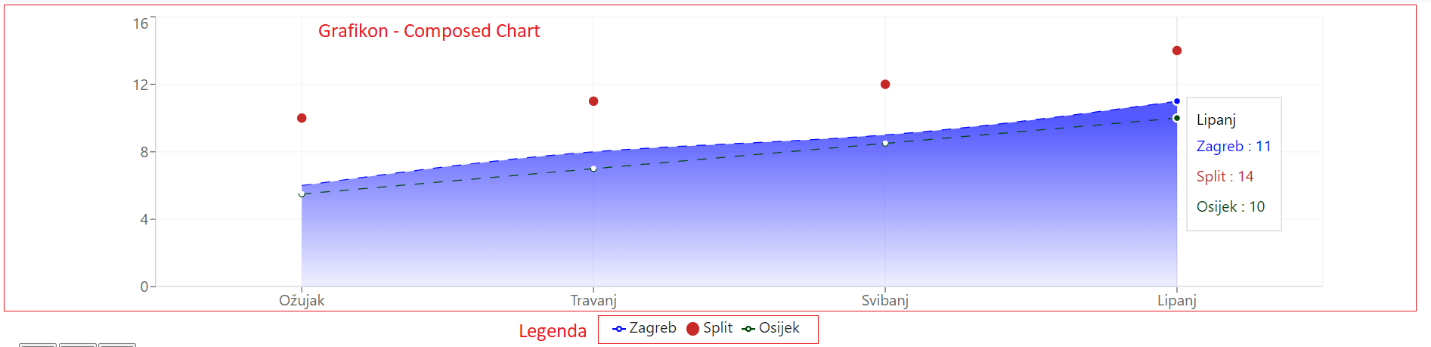


Charts:

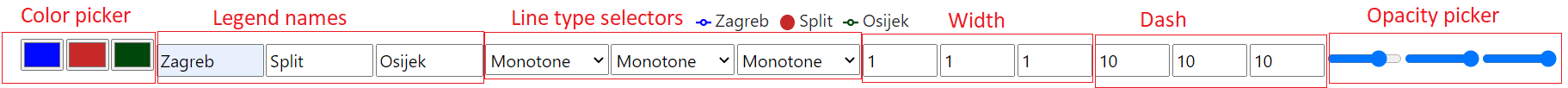
Navigacija:



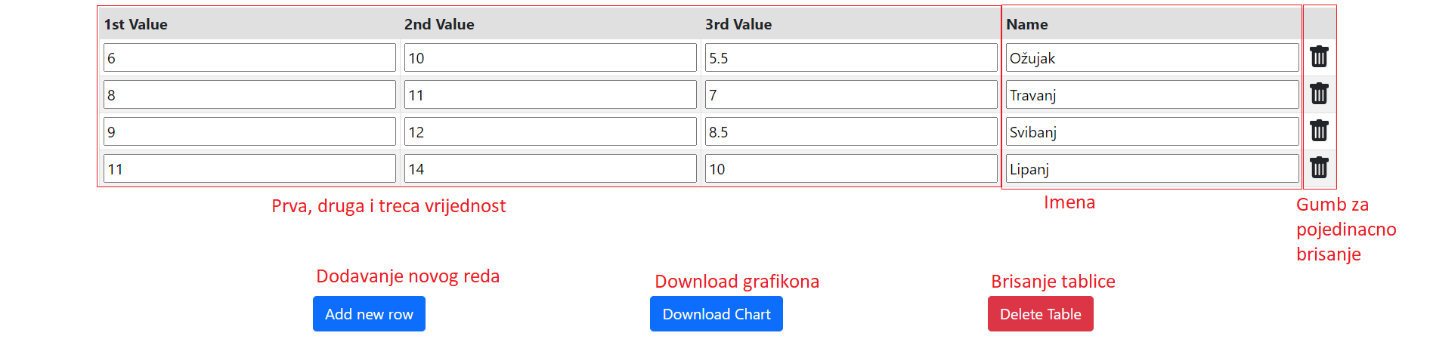
Grafikon:



Parametri prilagodbe:



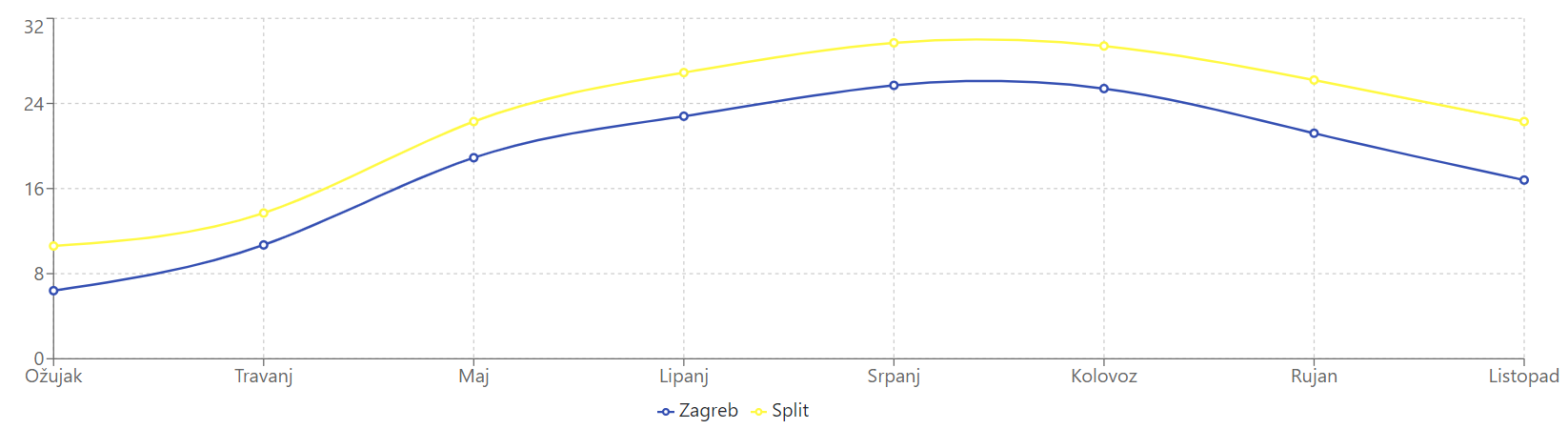
Vrijednosti (varijable):



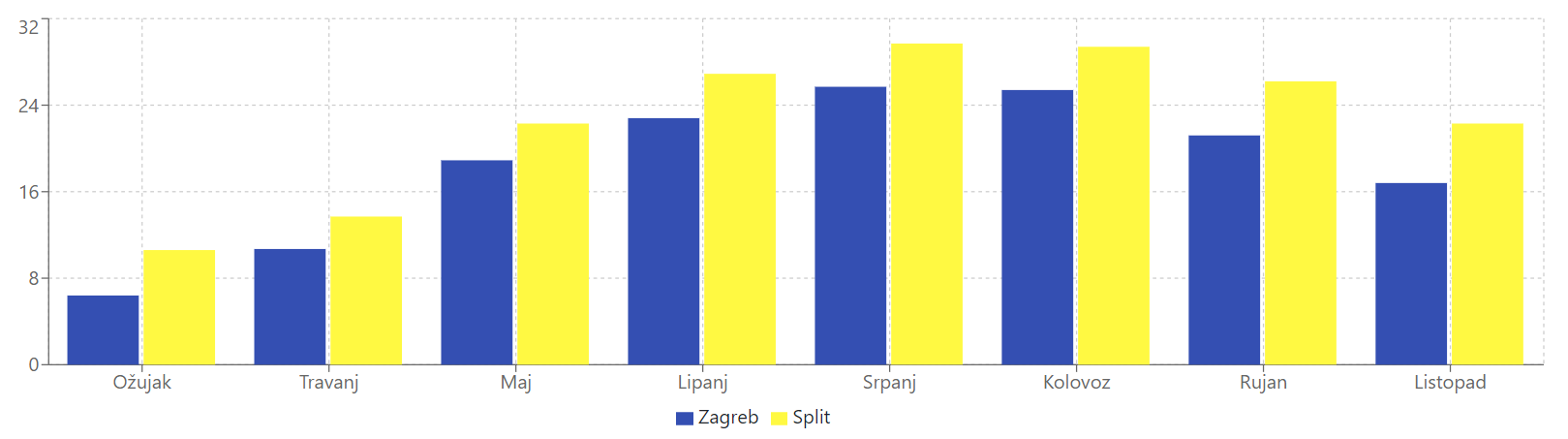
Primjer:

Usporedba prosječne mjesečne temperature Zagreb – Split (ožujak-listopad).

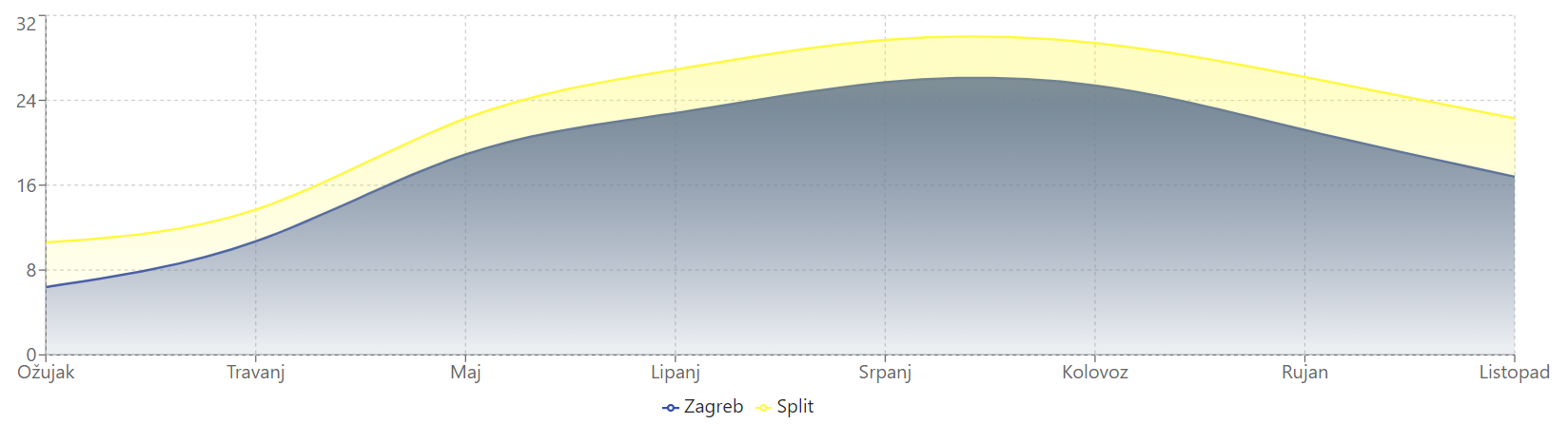
1. Line Chart



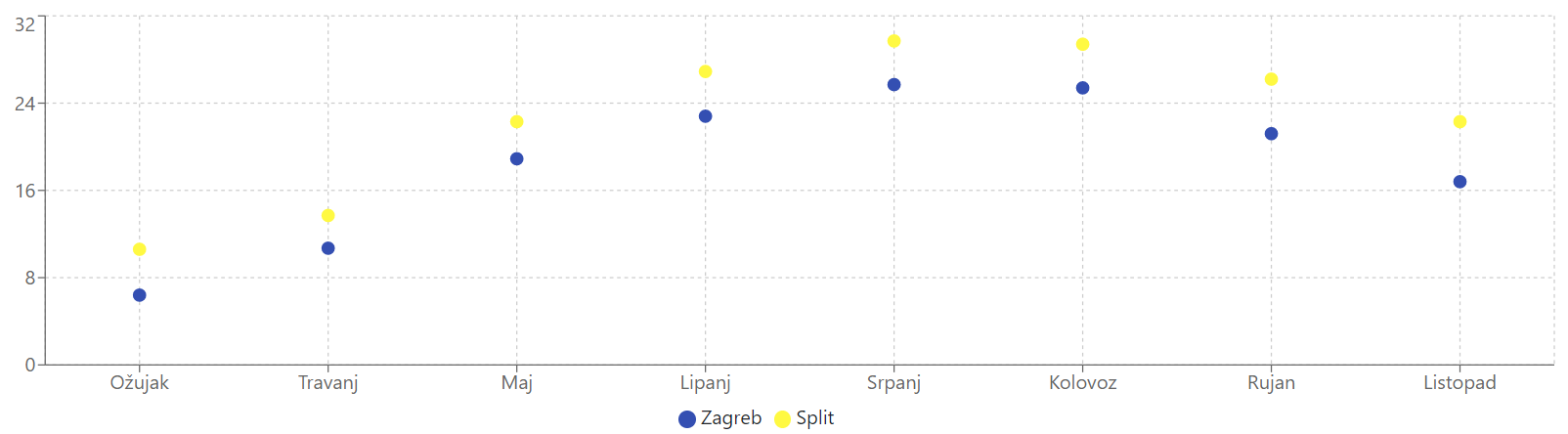
1. Bar Chart



1. Area Chart



1. Scatter Chart



1. Composed Chart

