Harjoitustyö vaihe 4

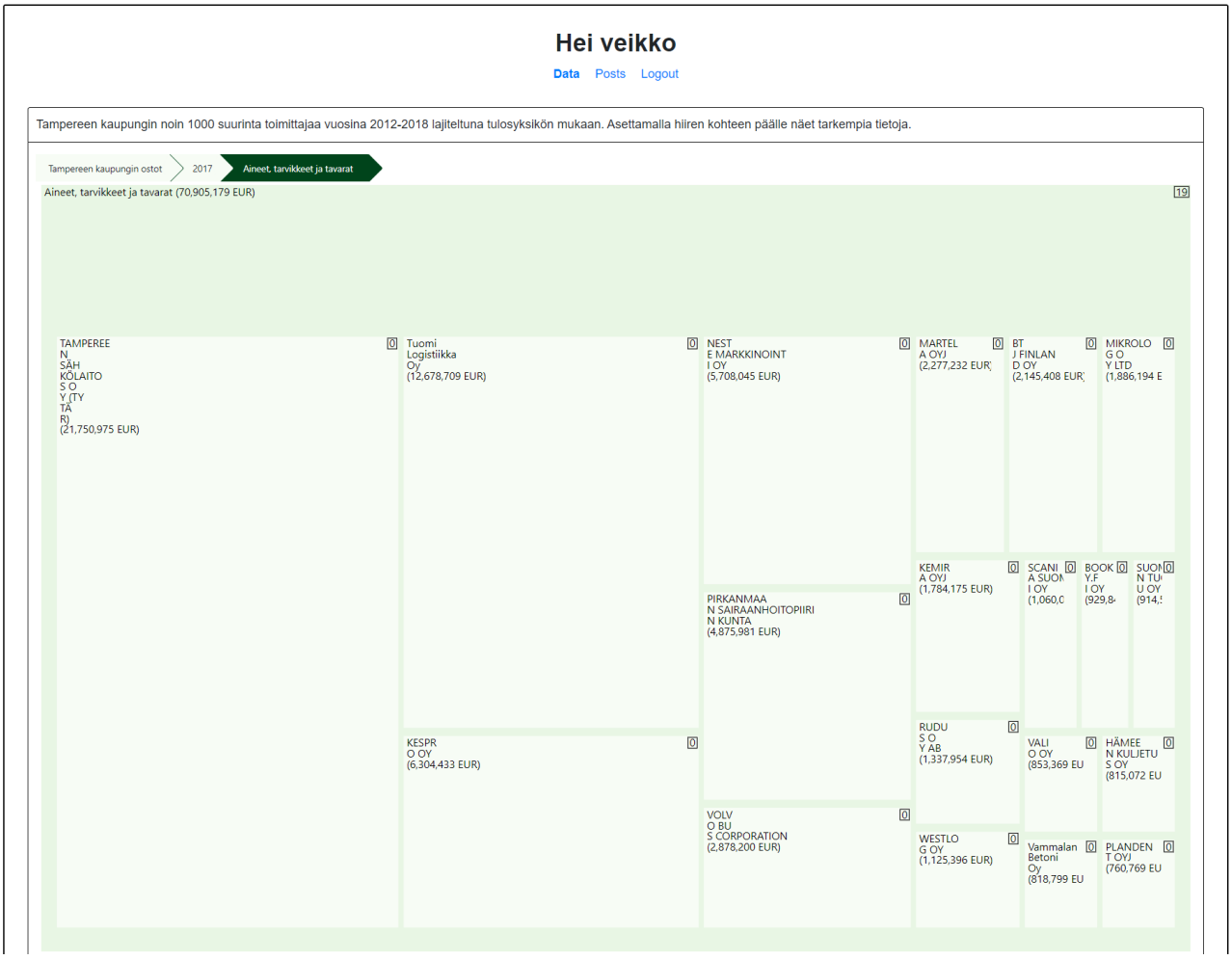
TLO-32400 Ohjelmallinen sisällönhallinta

Yleisesti

Päätin visualisoida dataa Treemap graafin kanssa, jolloin jokaisen alueen koko riippuu sen arvosta (euromäärästä). Tarkoituksenani oli käyttää vain vuoden 2018 tietoja mutta kuvaajasta sai mielenkiintoisemman useammalla vuodella. Vuosien lisäys oli yllättävän helppoa.

Frontend

Tiesin haluavani Treemapin joten tehtäväni oli löytää sopiva kirjasto käytettäväksi Reactin kanssa. Ongelma osoittautui yllättävän hankalaksi. Erilaisia kirjastoja on saatavilla runsaasti mutta harva oli kohdallani sopiva. Päädyin käyttämään kolmannen linkin kirjastoa, jonka avulla onnistuin luomaan hienon graafin. Muuntelin myös hiemna ulkoasua data sivulla ja lisäsin logiikan, jolla sivu hakee tiedot toiselta serveriltä. Liitteenä kuva valmiista käyttöliittymästä.



Kuvaajan lisääminen sivulle ei vaatinut kuin muutaman importin lisäämisen ja alla olevan koodin lisäämisen oikeaan kohtaan. Kuvaaja on dynaaminen, eli klikkaamalla aluetta se avaa alueen ja näyttää mistä se koostuu.

```
<TreeMap
  height={1000}
  width={1500}
  data={this.state.data}
  valueUnit={"EUR"}
/>
```

```
pool.query("SELECT * FROM data ORDER BY summa DESC LIMIT 1000", (q_err, q_res) => {
  var data = [];
  for (var i = 0; i < q_res.rows.length ; i++) {
    let tmp = q_res.rows[i].osio;
    let vuosi = q_res.rows[i].vuosi;

    if (data.filter(e => e.name == vuosi).length != 1){
      data.push({name: vuosi, children: []});
    };

    for (var l = 0 ; l < data.length ; l++){
      if (data[l].name == vuosi) {
        if (data[l].children.filter(e => e.name === tmp).length != 1){
          data[l].children.push({name: tmp, children: []});
        };
      };
    };

    for (var k = 0 ; k < data.length ; k++){
      if (data[k].name == vuosi) {
        for (var j = 0 ; j < data[k].children.length ; j++){
          if (data[k].children[j].name === tmp){
            data[k].children[j].children.push({name: q_res.rows[i].nimi, value: q_res.rows[i].summa});
          };
        ;}
      };
    };
  };
  var rdata = {name: "Tampereen kaupungin ostot", children : data}
  res.setHeader('Content-Type', 'application/json');
  res.json(rdata)
  })
```

Backend

Toisella serverillä tosin riitti hankaluuksia. Kaikki datan muokkaus oikeaan muotoon tehtiin tällä puolella. Käyttämäni kirjasto vaati tiedot tarkassa muodossa. Tiedon muokkaaminen tähän muotoon vaati paljon aikaa ja monta eri yritystä. Yllä oleva kuva oikealla toimivasta koodista.

Päätin myös lisätä lisää vuosia ja ottaa talteen myös toimiryhmän, joten myös datan jalostus vaati päivitystä. Jalostus perustuu viime vaiheessa luotuun python skriptin, joka ajetaan käsin. Päivitettynä skripti käy kaikki vuosien 2012-2018 vuosien tiedot läpi. Prosessissa kestää noin 20 minuuttia. Tämä skripti puskee tiedot psql-kantaan, josta backend serveri hakee tiedot. Alla oleva kuva kuvaa csv tiedoston purkamista.

```
print("Vuosi " + str(year) + " rivillä " + str(line))
if str(row[toimiala]) not in data:
    data[str(row[toimiala])] = {}
if  str(row[firma]) in data[str(row[toimiala])]:
    data[str(row[toimiala])][str(row[firma])] += float(row[summa].replace(',','.'))
else:
    data[str(row[toimiala])][str(row[firma])] = float(row[summa].replace(',','.'))
```

3 helppoa/hankalaa tekijää

1. Datan muokkaaminen oikeaan muotoon oli haastavaa.
2. Sopivan kirjaston etsimisessä kului paljon aikaan.
3. Itse kirjaston käyttäminen oli helppoa.

Hyödyllisiä linkkejä

- https://www.smashingmagazine.com/2018/02/react-d3-ecosystem/
- https://github.com/hshoff/vx
- https://github.com/jquintozamora/react-d3-treemap