Harjoitustyö vaihe 2

TLO-32400 Ohjelmallinen sisällönhallinta

Yleisesti

Päätin laajentaa projektia ja ottaa jo tässä vaiheessa käyttöön Reactin tietojen näyttämiseen sekä jatkaa node + express systeemillä backendin toteuttamista. Tässä vaiheessa toteutin rajapintaan yksinkertaistetut CR(U)D toiminnot sekä osittaisen tuen kirjautumiselle. Käyttöliittymän puolella toimii julkaisujen luominen, selaus ja poistaminen. Kirjautuminen ei tässä vaiheessa käyttöliittymässä toimi. Käyttöliittymä siis hakee kaiken näytettävän tiedon toiselta palvelimelta json muodossa.

Backend

Jatkoin viime vaiheessa luodun sovelluksen kehitystä. Toteutin toisen linkin mukaisesti hyvin yksinkertaisen yhteyden tietokantaan sekä yksinkertaiset rajapinnat. Toteutuksessa meni enemmän aikaa kuin kuvittelin, sillä kohtasin monia ongelmia. Tuotos on kuitenkin toimiva ja yksinkertainen.

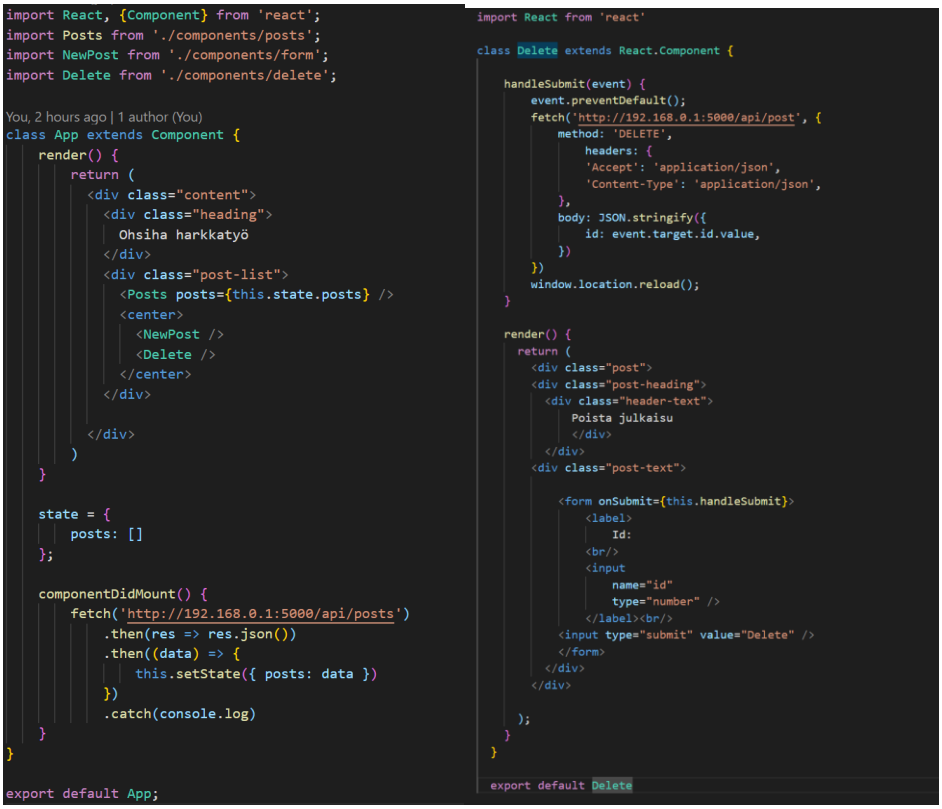
```
router.post('/api/post', (req, res, next) => {
  if (req.body.title == ""|| req.body.body == "") {
    res.status(400).send();
  } else {
    const values = [req.body.title, req.body.body];
    pool.query("INSERT INTO posts (title, body) VALUES ($1, $2)", valu
      if (q_res != null){
        if (q_res.rowCount == 0) {
          res.status(400).send();
        }
      }
      res.status(200).send();
    })
  }
});
```

Kuvassa on esitetty yksi rajapinnan methodi. Tämä vastaa osoitteessa localhost:5000/api/post ja oikeilla arvoilla se luo uuden julkaisun ja lisää sen tietokantaan. Lisättäköön tähän vaiheeseen, että tekemäni keinot eivät kuitenkaan vastaa julkisen käyttön vaatimuksia, ohjelmassa on aivan liian huono tietoturva.

Frontend

En ole aikaisemmin tehnyt Reactin kanssa mitään, joten homma alkoi lukemalla ensimmäisen linkin mukainen ohjeistus kokonaan läpi. Itse toteutuksessa en suoranaisesti käyttänyt mitään ohjetta, sillä koin monet esimerkit liian monimutkaisiksi tarpeisiini. Tämän takia piti sooloilla ja etsiä tietoa monesti tietoa yksittäisiin ongelmiin.

Itse toteutus on hyvin yksinkertainen. Vasemmalla ylhäällä kuvassa näkyy, miten ohjelman keskeisin tiedosto on koostettu. Tässä toteutetaan julkaisujen haku sekä muiden komponenttien kutsuminen. Oikealla ylhäällä näkyy poistamisessa käytetty luokka. Se kysyy poistettavan julkaisun id:n ja lähettää oikean muotoisen pyynnön palvelimelle. Julkaisun luominen on toteutettu samaan tapaan. Alhaalla on esitetty julkaisujen näyttämisessä käytettävä komponentti.



```
import React, {Component} from 'react';
import Posts from './components/posts';
import NewPost from './components/form';
import Delete from './components/delete';

You, 2 hours ago | 1 author (You)
class App extends Component {
    render() {
        return (
          <div class="content">
            <div class="heading">
              Ohsiha harkkatyö
            </div>
            <div class="post-list">
              <Posts posts={this.state.posts} />
              <center>
                <NewPost />
                <Delete />
              </center>
            </div>

          </div>
        )
    }

    state = {
        posts: []
    };

    componentDidMount() {
        fetch('http://192.168.0.1:5000/api/posts')
            .then(res => res.json())
            .then((data) => {
                this.setState({ posts: data })
            })
            .catch(console.log)
    }
}

export default App;
```

```
import React from 'react'

class Delete extends React.Component {

    handleSubmit(event) {
        event.preventDefault();
        fetch('http://192.168.0.1:5000/api/post', {
            method: 'DELETE',
                headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                id: event.target.id.value,
            })
        })
        window.location.reload();
    }

    render() {
      return (
        <div class="post">
        <div class="post-heading">
          <div class="header-text">
              Poista julkaisu
              </div>
          </div>
        <div class="post-text">

            <form onSubmit={this.handleSubmit}>
                <label>
                    Id:
                <br/>
                <input
                    name="id"
                    type="number" />
                </label><br/>
            <input type="submit" value="Delete" />
            </form>
          </div>
        </div>

      );
    }
  }

export default Delete
```

```
import React from 'react'

const Posts = ({ posts }) => {
  return (
    <div>
      {posts.map((post) => (
        <div class="post">
          <div class="post-heading">
            <div class=" header-text">{post.title}</div>
            <div class="date">Id: {post.id}</div>
            </div>
          <div class="post-text">{post.body}</div>
        </div>
      ))}
    </div>
  )
};

export default Posts
```

Vaiheen toteuttamisessa meni huomattavasti enemmän aikaa kuin kuvittelin, joten käyttöliittymään en kirjautumista ehtinyt toteuttamaan. Toisaalta minulla on nyt hyvin tehty pohja lähteä jatkamaan harjoitustyötä seuraavissa vaiheissa.

3 helppoa/hankalaa tekijää

1. Reacti on oma kokonaisuutensa, jonka opiskeluun ja sisäistämiseen menee aikaa.
2. Omaan tilaanteeseen sopivia ohjeita ei juurikaan löytynyt, joten piti soveltaa.
3. Rajapinta on tutun oloista koodia jota oli helppo toteuttaa alun ongelmien jälkeen.

Hyödyllisiä linkkejä

- https://www.w3schools.com/REACT/default.asp
- https://medium.com/@olinations/build-a-crud-template-using-react-bootstrap-express-postgres-9f84cc444438
- https://www.freecodecamp.org/news/fullstack-react-blog-app-with-express-and-psql/
- https://pusher.com/tutorials/consume-restful-api-react