# OHSIHA 2020, Lisäominaisuus: AJAX-toteutus

Arttu Saarinen
arttu.saarinen@tuni.fi

GitHub-linkki projektiin: https://github.com/saarinea/markets
Heroku-linkki projektiin: https://tranquil-hollows-95262.herokuapp.com/

Harjoitustyön tarkoituksena on toteuttaa verkkopalvelu, joka hakee rajapinnoista erilaista finanssimarkkinadataa (mm. osakekursseja, pörssi-indeksejä, valuuttakursseja) ja koostaa niistä eräänlainen dashboard omiin informaatiotarpeisiini.

Sovelsin harjoitustyössäni AJAX-teknologiaa asynkronisessa tiedonsiirrossa front- ja backendin välillä mm. hakiessa osakedataa sekä myös hakiessa käyttäjän kirjautumistietoja. Tämä tehtiin Reduxin avulla: Redux on Reactin kanssa usein käytettävä applikaation tilaa ylläpitävä kirjasto. Applikaatiokohtaista redux-storea voidaan ajatella ikään kuin applikaation tietokantana, joka pitää huolta applikaation tilasta ja joka kykenee välittämään tietoja esim. eri react-komponenttien välillä ja jossa voidaan tehdä http-requesteja backendiin, niin kuin itse tein.

## Käytetyt teknologiat

Etenkin tässä vaiheessa käytetyt teknologiat **boldattuna.**

- Frontend: React
- Backend: Node.js
- Tietokanta: Atlas MongoDB
- Versionhallinta: Github
- Editori: Visual Studio Code
- Serveri/sovellusplatform: Heroku
- Hyödynnettyjä kirjastoja: Passport, Mongoose, Nodemon, Bootstrap, **Axios**, Cors, **Redux**, **Redux-thunk,** react-stockcharts, d3.js
- Hyödynnettyjä kolmannen osapuolen rajapintoja: AlphaVantage

## Otteita AJAX-toteutuksesta

Reduxin käyttöä olen dokumentoinut paremmin toisissa harjoitustyön dokumentaatioita, joten en ala sitä selittämään alusta lähtien. Alla olevassa blogissa on myös hyviä ohjeita tähän ja siihen mitä Redux oikeastaan tarkoittaa. AJAX-tyyppinen asynkroninen tiedonhaku tehtiin actionseissa, alla kuvaus toteutuksesta:

```
// actions.js (koko tiedosto)

import axios from 'axios'
import { FETCH_USER, GET_DATA } from './types.js'

export const fetchUser = () => async dispatch => {
  const res = await axios.get('/api/current_user')
  dispatch({ type: FETCH_USER, payload: res.data })
}

export const getData = (ticker) => async dispatch => {
  const res = await axios.get('/data/stocks',
  {params: {
```

```
    ticker: ticker
  }})
  dispatch({ type: GET_DATA, payload: res.data })
}
```

Kuten koodista huomataan, asynkroninen tiedonhaku backendistä tehdään kuin mikä tahansa muu API-haku. Tässä hyödynnetään async-await-ominaisuutta sekä axios-kirjastoa.

Tämä näkyy frontendissä puolestaan siten dynaamisena tiedonhakuna, että ennen kuin tiedot ovat ladanneet, näytetään ”Loading…”-teksti kuvaajaelementin sijasta. Tästä toteutus koodina vielä alla:

```
// Singlestock.js (Osa koodista)

 render() {
    const { error, isLoaded, isSelected } = this.props

    if (error) {
      return <div>Error in loading</div>
    } else if (!isSelected) {
      return <div>Select a stock</div>
    } else if (!isLoaded) {
      return <div>Loading...</div>
    } else {
      return (
        <div>
          <div>
            <table>
              <tbody>{this.renderMetaData()}</tbody>
            </table>
            <div className="linechart-dark">
            <StockLineChart2 />
            </div>
          </div>
        </div>
      )
    }
  }
}
```

## Haasteet ja helpot asiat

1. **Haaste:** Aluksi oli melko hankalaa hahmottaa mihin väleihin tiedonhakupyynnöt API:sta tulevat: React, Redux sekä asynkroninen tiedonsiirto on melkoinen paketti opiskeltavaksi kerralla
2. **Helppoa:** Kun Reduxin ja Reactin arkkitehtuurin tajuaa, on melko helppo tehdä vain simppeleitä komponentteja toistensa päälle niiden itsenäisyyden ansiosta
3. **Haaste:** Yleisesti esim. async-await-rakenteet sekä muutenkin ei-lineaarinen koodi on Javascriptin sisälläkin melko haastavaa alkuun. Kuten yleisestikin web-koodi ja Javascript, on paljon koodia, funktioiden tyyppejä yms. joiden toimintaa ei voi pelkästään päätellä koodista vaan jotka pitää vain muistaa ulkoa. (esim. app.use()-syntaksin middlewaret)

## Hyödyllisiä lähteitä

Axios ja Node-tutoriaali -https://flaviocopes.com/node-axios/

How to Use Async/Await with Axios in React - https://medium.com/better-programming/how-to-use-async-await-with-axios-in-react-e07daac2905f

Redux ja React, asynkroninen tiedonhaku Reduxia käyttäen - https://www.valentinog.com/blog/redux/