OHSIHA 2020, Lisäominaisuus: SPA-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.

Lähdin alusta asti työstämään sovellustani React-sovelluksena, jolloin siitä melkein ”pakollakin” Single Page Application-periaatteen mukainen. Applikaatio ei ole kovin monimutkainen, muuta siinä kuitenkin SPA-periaate näkyy siten, että osakkeiden tietoja tarkastellessa mennään pohjasivulle /stocks, jonka ”sisällä” puolestaan valitaan mitä osaketta halutaan tarkastella ja data haetaan ja päivitetään visualisointikomponenttiin tuon valinnan mukaan ilman, että sivu vaihtuu.

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 SPA-toteutuksesta

SPA-toteutus ei ole ihan täysin puhdas, sillä tarkoituksena on että omat alisivut toimivat ”ali-appeinaan”, vaikkei ainoastaan yksi aliapp onkin tässä vaiheessa toteutettunakaan. Reactin App-komponentin sisällä aliapp valitaan seuraavasti:

```
// App.js (osa koodista)

class App extends Component {
  componentDidMount() {
    this.props.fetchUser();
  }

  render() {
    return (
      <div>
        <BrowserRouter>
          <div>
            <Header />
            <Route exact path="/" component={Landing} />
```

```
            <Route exact path="/stocks" component={Stocks} />
            <Footer />
          </div>
        </BrowserRouter>
      </div>
    );
  }
}

export default connect(null, actions)(App);
```

Aliappissa SPA-periaate näkyy puolestaan siten, että ko. sivu ladataan ainoastaan kerran ja sen sisällä tapahtuu tarvittavat datan haut API:sta ja sitä varten tarvittavat parametrien valinnat ja välittäminen eteenpäin käyttäjän antamien komentojen mukaan.

```
// Stocks.js (kokonaan)

import React, { Component } from 'react'
import SingleStock from './SingleStock.js'
import StockDropdown from './StockDropdown.js'

class Stocks extends Component {
  render() {
    return (
      <div>
        <h2>Stocks</h2>
        <StockDropdown />
        <SingleStock />
      </div>
    )
  }
}
export default Stocks;
```

```
// StockDropDown.js (kokonaan)

import React, { Component } from 'react'
import { connect } from 'react-redux'
import { getData } from '../actions/index'

class StockDropdown extends Component {
  state = {
    isOpen: false,
    menuText: 'Choose stock'
  }

  toggleOpen = () => this.setState({ isOpen: !this.state.isOpen })
  setMenutext = text => this.setState({ menuText: text })
```

```
click = (text, ticker) => {
  this.toggleOpen()
  this.setMenutext(text)
  this.props.updateData(ticker)
}

render() {
  const menuClass = `dropdown-menu${this.state.isOpen ? ' show' : ''}`

  return (
    <div className="dropdown">
      <button
        className="btn btn-secondary dropdown-toggle"
        type="button"
        id="dropdownMenuButton"
        data-toggle="dropdown"
        aria-haspopup="true"
        aria-expanded="false"
        onClick={this.toggleOpen}
      >
        {this.state.menuText}
      </button>
      <div className={menuClass} aria-labelledby="dropdownMenuButton">
        <a
          className="dropdown-item"
          href="#"
          onClick={() => this.click('Facebook (FB)', 'FB')}
        >
          Facebook (FB)
        </a>
        <a
          className="dropdown-item"
          href="#"
          onClick={() => this.click('Amazon (AMZN)', 'AMZN')}
        >
          Amazon (AMZN)
        </a>
        <a
          className="dropdown-item"
          href="#"
          onClick={() => this.click('Apple (AAPL)', 'AAPL')}
        >
          Apple (AAPL)
        </a>
        <a
          className="dropdown-item"
          href="#"
          onClick={() => this.click('Netflix (NFLX)', 'NFLX')}
        >
          Netflix (NFLX)
        </a>
```

```
          <a
            className="dropdown-item"
            href="#"
            onClick={() => this.click('Alphabet (GOOG)', 'GOOG')}
          >
            Alphabet (GOOG)
          </a>
        </div>
      </div>
    )
  }
}

function mapDispatchToProps(dispatch) {
  return {
    updateData: ticker => dispatch(getData(ticker))
  }
}

export default connect(null, mapDispatchToProps)(StockDropdown)
```

Haasteet ja helpot asiat

1. **Haaste:** Aluksi Reactin kokonaisarkkitehtuurin tajuaminen oli melko vaikeaa: dokumentaatiossa oletetaan jo jonkunlaista osaamista Javascriptista, ja oma osaaminen sen osalta oli vielä käytännössä nollassa ennen harjoitustyötä. Täten paljon aikaa kului yksittäisten komentojen ja niiden syntaksien selvittämiseen.
2. **Helppoa:** Kun Reactin kokonaisarkkitehtuurin tajuaa ja saa yhdenkin komponentin toimimaan, on sitä helppo kopioida eteenpäin ja jalostaa edelleen
3. **Haaste:** Jonkin verran aikaa vei myös tajuta mitä eroa on App-komponentilla ja muilla Reactin komponenteilla. Olisi tärkeää saada kattotason käsitteistä vielä parempi kokonaiskuva ennen kuin asioita lähtee itse tekemään, ns. pylly edellä puuhun-metodilla.
4. **Haaste:** Kokonaisuudessaan valinta tehdä harjoitustyö Djangon sijaan Reactilla on todella työläs valinta: käytännössä kaikki tieto pitää etsiä itse.

Hyödyllisiä lähteitä

Getting Started, React - https://reactjs.org/docs/getting-started.html (muutkin Reactin sivujen tutoriaalit hyödyllisiä)

React ja Redux - https://www.valentinog.com/blog/redux/ , https://alligator.io/redux/redux-thunk/