OHSIHA 2020, Harjoitustyön vaihe 2

Arttu Saarinen
arttu.saarinen@tuni.fi

Harjoitustyön tarkoituksena on toteuttaa verkkopalvelu, joka hakee rajapinnoista erilaista finanssimarkkinadataa (mm. osakekursseja, pörssi-indeksejä, valuuttakursseja) ja koostaa niistä eräänlainen dashboard omiin informaatiotarpeisiini. Teknologioiksi valitsin alla listatut työkalut, koska olen Web-ohjelmoinnin kurssin käynyt ja halusin oppia uutta sekä uuden kielen (Javascript) että muuten uusien työkalujen avulla.

Käytetyt teknologiat

- Frontend: React
- Backend: Node.js
- Tietokanta: Atlas MongoDB
- Versionhallinta: Github
- Editori: Visual Studio Code
- Serveri/sovellusplatform: Heroku
- Hyödynnettyjä kirjastoja: Passport, Mongoose, Nodemon

Kirjautuminen ja rekisteröityminen

Harjoitustyön toisessa vaiheessa toteutin sivulleni kirjautumis- ja rekisteröitymisominaisuudet ja yleisen autentikoinnin. Autentikaatioon käytin OAuth-pohjaista toteutusta, ja kertakirjautumista toteutettuna Googlen rajapintojen avulla. Kirjautuminen toimii tällä hetkellä etusivulla, painamalla linkkiä josta käyttäjä ohjautuu ensin Googlen kirjautumisproseduuriin ja sieltä takaisin Landing-sivulle. Toteutuksen näkyvyys sivulla allaolevan näköinen.

Itse toteutus on tehty paljolti Passport-kirjaston avulla. Passportin upottaminen oli melko suoraviivaista, ja Googlen kirjautumisputki hoitaa loput: itse koodissa tuli lähinnä asettaa reitit kohdilleen sekä konfiguroida Passportin käyttämät asetukset. Alla esimerkkikoodia Passportin käyttämistä asetuksista:

```
const passport = require('passport');
const GoogleStrategy = require('passport-google-oauth20').Strategy;
const mongoose = require('mongoose')
const keys = require('../config/keys.js');


const User = mongoose.model('users');
```

```
passport.serializeUser((user, done) => {
    done(null, user.id)
});

passport.deserializeUser((id, done) => {
    User.findById(id)
    .then(user => {
        done(null, user);
    });
});

passport.use(new GoogleStrategy({
    clientID: keys.googleClientID,
    clientSecret: keys.googleClientSecret,
    callbackURL: '/auth/google/callback',
    proxy: true
    },
    async (accessToken, refreshToken, profile, done) => {
        const existingUser = await User.findOne({googleID: profile.id})

        if (existingUser) {
            return done(null, existingUser);
        }

        const user = await new User({
            googleID: profile.id,
            name: profile.name.givenName
        }).save()
        done(null, user)
    }
    )
);
```

Esimerkkidatan näyttäminen

Näytetyn esimerkkidatan hain jo API:sta, mutta kuitenkin siten että tein API-haun frontendin puolelta ko. API:n demopuolelta, eikä itse hakumerkkijonoon / urliin kosketa tässä kohtaa tarkemmin. Datan hain Alphavantage-nimiseltä verkkosivulta, jota aion käyttää myös myöhemmin tässä harjoitustyössä laajemmin. Samoin datan käsittely tehtiin tässä kohtaa myös frontendin puolella, vaikka toki tarkoituksena on siirtää datan käsittely mahdollisimman pitkälle backendiin. Koodi on tässä kohtaa melko raskasta ja uskoisin, ettei

ole tarkoituksenmukaista että frontendissä käsitellään sitä näin vahvasti, vaan ajattelisin että se olisi hyvä hoitaa mahdollisimman pitkälti backendin puolella. Alla esimerkkikoodia Stocks-nimisestä React-komponentista ja kuva toteutuksesta/esimerkkidatasta sivuilla:

```
import React, { Component } from "react";

class Stocks extends Component {
 state = {
  error: null,
  isLoaded: false,
  metadata: [],
  timeseries: []
 }

 componentDidMount() {

fetch('https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY&symbol=MSFT&interval=5min&apikey=demo')
    .then(res => res.json())
    .then((data) => {
      this.setState({
        metadata: data["Meta Data"],
        timeseries: data["Time Series (5min)"],
        isLoaded: true
      })
    },
      (error) => {
        this.setState({
          isLoaded: true,
          error
        })
      })
    .catch(console.log())
 }

 renderMetaData(){
  var metadata = this.state.metadata
  var metaArray = []

  for (var i in metadata){
    metaArray.push([i,metadata[i]])
  }

  return metaArray.map((item) => {
```

```
    var meta = item[0].substr(3)
    var info = item[1]

    return (
      <tr>
        <td>{meta}</td>
        <td>{info}</td>
      </tr>
    )
  })
}

renderTimeSeries(){
  var timeseries = this.state.timeseries
  var timeSeriesArray = []

  for (var i in timeseries){
    var closeprice = timeseries[i]["4. close"]
    timeSeriesArray.push([i, closeprice])
  }

  return timeSeriesArray.map((item) => {
    var timestamp = item[0]
    var close = item[1]

    return (
      <tr>
        <td>{timestamp}</td>
        <td>{close}</td>
      </tr>
    )
  })
}

render() {
  const { error, isLoaded } = this.state;

  if (error) {
    return <div>Error in loading</div>
  }
  else if (!isLoaded) {
    return <div>Loading...</div>
```

```
    }
    else {
      return (
        <div>
          <h2>Stocks</h2>
          <table>
            <tbody>
              {this.renderMetaData()}
            </tbody>
          </table>
          <table>
            <tbody>
              {this.renderTimeSeries()}
            </tbody>
          </table>
        </div>
      );
    }
  }
}

export default Stocks;
```

Markets Login With Go

Stocks

Information	Intraday (5min) open, high, low, close prices and volume
Symbol	MSFT
Last Refreshed	2020-03-10 12:10:00
Interval	5min
Output Size	Compact
Time Zone	US/Eastern

2020-03-10 12:10:00	155.3100
2020-03-10 12:05:00	154.7900
2020-03-10 12:00:00	153.9500
2020-03-10 11:55:00	153.6400
2020-03-10 11:50:00	154.7000
2020-03-10 11:45:00	154.7900

Haasteet ja helpot asiat

1. Uusien

Hyödyllisiä lähteitä

Passportin dokumentaatio - http://www.passportjs.org/docs/

OAuth, Passport, Google, Nodejs - https://developerhandbook.com/passport.js/how-to-add-passportjs-google-oauth-strategy/

Datan hakeminen 3:n osapuolen API:sta - https://www.techiediaries.com/react-json-fetch-rest-api-bootstrap/ ja https://howtocreateapps.com/json-html-react-tutorial/

Esimerkkidatan näyttäminen sivulla - https://howtocreateapps.com/json-html-react-tutorial/

React Appin debuggaus Chromessa ja VS Codessa - https://www.youtube.com/watch?v=PJeNReqyH88

JSON:in saaminen taulukkomuotoon Reactissa - https://dev.to/abdulbasit313/an-easy-way-to-create-a-customize-dynamic-table-in-react-js-3igg