OHSIHA 2020, Lisäominaisuus: Kertakirjautuminen, Google OAuth

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.

Toteutin harjoitustyöhöni kertakirjautumisen Googlen OAuth-protokollaa / APIa käyttäen, sillä se vaikutti kaikista monikäyttöisimmältä ratkaisulta esitetyistä vaihtoehdoista. Ainakaan itselläni ei edes ole Twitteriä ym. Toteutukseen meni jonkun verran aikaa, mutta kokonaisuudessaan se oli kuitenkin melko suoraviivaista kun piti mielessä tarkalleen mitä oli tekemässä.

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, react-stockcharts, d3.js
- Hyödynnettyjä kolmannen osapuolen rajapintoja: AlphaVantage

Kertakirjautuminen: Google OAuth

Toteutus oli melko monimutkainen ja tein sen jo jonkin aikaa sitten, joten en ihan välttämättä jokaista kohtaa muista. Kokonaisuudessaan toteutus meni kuitenkin melko lailla alla olevalla tavalla:

1. Passportin sekä tietokantayhteyden konfigurointi koodissa

Projektin index.js-tiedostoon tuli importata passport ja se tuli ottaa tiedostossa myöhemmin käyttöön (app.use). Lisäksi importattiin myös User-malli tietokantaa varten (User.js) sekä erillinen asetustiedosto passport.js, jossa on määriteltynä passportin asetukset. Toteutus alla:

```
// index.js (osia)

const passport = require('passport')
require('./models/User.js')
require('./services/passport.js')
…….
app.use(passport.initialize())
app.use(passport.session())
```

Passport.js-tiedostossa puolestaan luotiin asetukset passportille ja yhdistettiin lisäksi passport tietokantaan mongoose-kirjastolla (User.js-tiedostossa määritellään käyttäjämalli):

```
// passport.js (kokonaan)

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)
    }
    )
);
```

User.js-tiedostossa puolestaan määriteltiin käyttäjän malli:

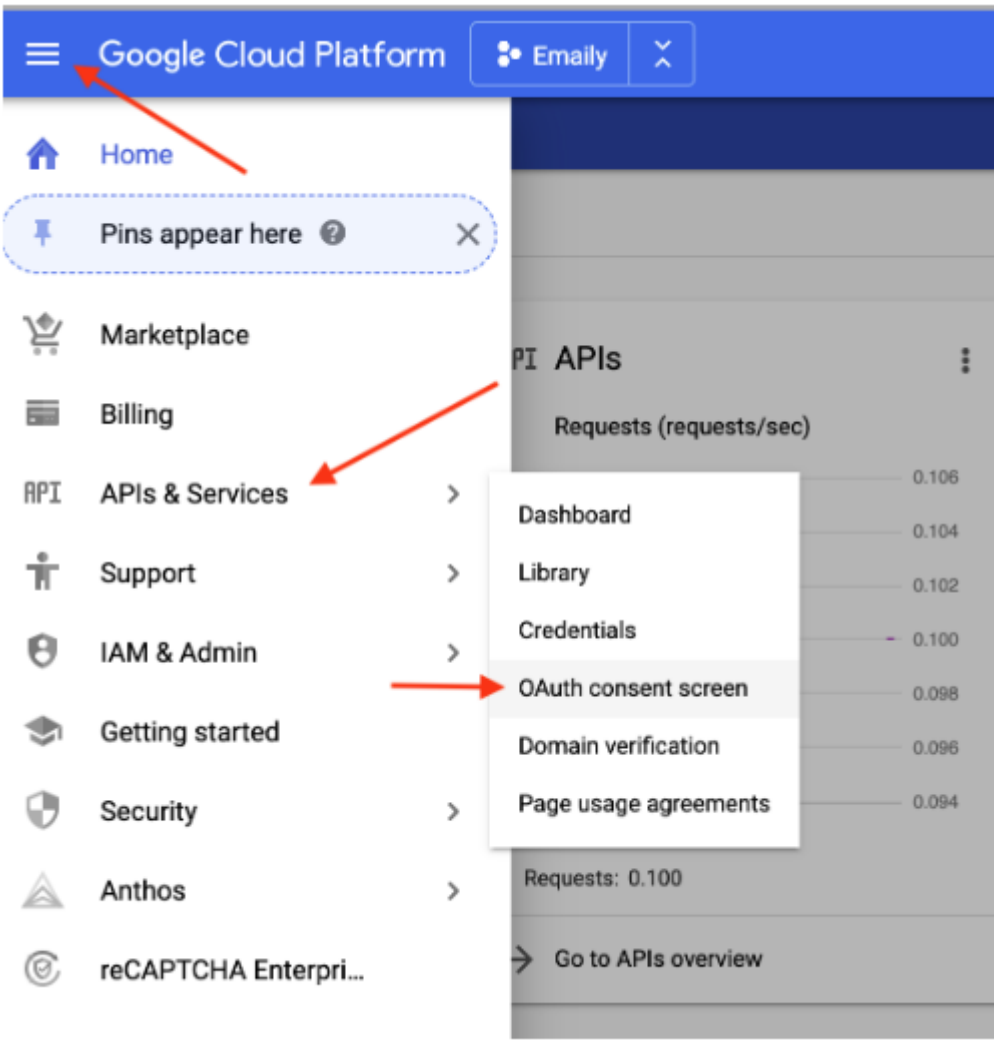
```
// User.js (kokonaan)

const mongoose = require('mongoose');
```

```
const { Schema } = mongoose;

const userSchema = new Schema({
    googleID: String,
    name: String
});

mongoose.model('users', userSchema);
```

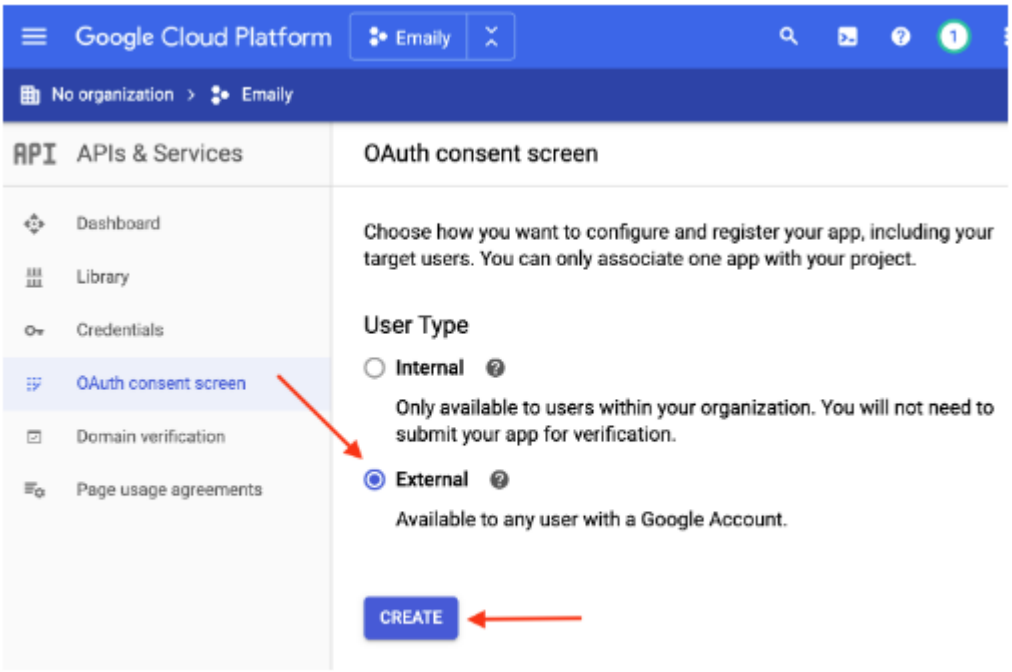
2. Uuden appin luominen Googlessa ja sen konfigurointi

Autentikaatiota varten tulee luoda uusi app Googlen Cloud Platformiin ja asetella sieltä oikeat asetukset päälle. Se tapahtuu seuraavasti:

1. Mene https://console.cloud.google.com/
2. Tee uusi projekti ”Create Project” ja valitse se aktiiviseksi
3. Mene ”OAuth Consent Screen”-valikkoon



4. Aseta organisaation ulkoiset kirjautumiset sallituiksi ja paina luo



5. Aseta muut asetukset (nimi yms)
6. Aseta sallitut credentiaalit:
    a. Mene ”Credentials”
    b. Mene ”OAuth Client ID”
    c. Laita ”Authorized Javascript Origins”-kohtaan http://localhost:5000 tai muu osoite jossa paikallisen palvelimesi **frontend** pyörii. Tähän kohtaan tulee tuotantoversiossa herokussa projektin heroku-osoite.
    d. Laita ”Authorized Redirect URIs”-kohtaan http://localhost:3000 tai muu osoite jossa **backendisi** pyörii. Tuotantoversiossa tähän tulee esim. https://tranquil-hollows-95262.herokuapp.com/auth/google/callback tai muu koodin sisällä määrittämäsi callback-URL.
7. Kopioi Client ID sekä Client Secret talteen (esim. keys.js-tiedosto koodin sisällä (**MUISTA LISÄTÄ GITIGNOREEN TÄMÄ TIEDOSTO JOTTEI API:N AVAIMET PÄÄSE JULKISEEN JAKOON)** tai Herokussa paikallisiksi muuttujiksi.

Cookieiden asettaminen ja tarkistaminen sivulla

Käyttäjien autentikaatiosta pidetään huolta asettamalla kirjautumisen yhteydessä cookie selaimelle. Tätä varten pitää ensin backend-serverin index.js-tiedostoon asettaa tarvittavat konffaukset:

```
// index.js (osia)

const cookieSession = require('cookie-session')
…
app.use(
  cookieSession({
    maxAge: 30 * 24 * 60 * 60 * 1000,
    keys: [keys.cookieKey]
  })
)
```

Tämän jälkeen cookieSession pitää huolta autentikaation säilymisestä sovelluksen statessa automaattisesti.

Haasteet ja helpot asiat

1. **Haaste:** Googlen OAuth-toteutus on melko monimutkainen Reactin kanssa kytkettynä sen monimutkaisen workflown takia: tokenit ja tiedot lentelevät requestista toiseen ja prosessissa voi olla vaikea pysyä mukana. Ongelmien debugaamienn on hyvin spesifiä ja aikaavievää tämän takia.
2. **Helppo asia:** Kun Googlen Sign In:n on kerran saanut konfiguroitua oikein, on se erittäin luotettava ja helppo autentikaatiomenetelmä.
3. **Haaste:** Vaikka olen nyt kertaalleen tehnyt autentikaation ohjeita noudattamalla alusta loppuun ja ymmärrän mitä koodissa suurin piirtein tapahtuu, en osaisi mitenkään kirjoittaa tyhjästä samanlaista koodia uudelleen: kyseessä todellakin on melko monimutkainen teknologia.

Hyödyllisiä lähteitä

Googlen Sign In-esittely - https://developers.google.com/identity?csw=1

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

Oauth eri menetelmillä (en tosin käyttänyt socketia vaan cookieita) - https://codeburst.io/react-authentication-with-twitter-google-facebook-and-github-862d59583105

Erittäin hyödyllinen (joskin maksullinen, itse sain työnantajaltani pääsyn) tutoriaali, jonka OAuth-kohtien mukaan tein autentikaation - https://www.udemy.com/course/node-with-react-fullstack-web-development/