

Sosiaalinen kirjautuminen

NativeScript + Angular

<https://github.com/SamuliRukkila/nativescript-social-login>

Samuli Rukkila

Oppimistehtävä

Huhtikuu 2019

ICT-ala

Tradenomi (AMK), tietojenkäsittelyn tutkinto-ohjelma

Sisältö

1	Johdanto.....	2
2	Versiot	2
3	Sosiaaliset APIT.....	2
3.1	Google+.....	2
3.2	Facebook.....	6
4	Frontend.....	7
4.1	Tiedostot.....	7
5	Backend	16
5.1	Google.....	17
5.2	Facebook.....	18
6	Testaus	21
6.1	Google.....	21
6.2	Facebook.....	22
	Lähteet.....	23

Kuviot

Kuvio 1.	Uuden Google-projektin luominen	2
Kuvio 2.	Googlen rajapinnan valinta	3
Kuvio 3.	Applikaation nimeäminen	3
Kuvio 4.	Androidin SHA1-sormenjäljen haku (Linux)	4
Kuvio 5.	NativeScript-projektin metatiedosto	5
Kuvio 6.	Valmis Google OAuth -lomake	5
Kuvio 7.	Facebook-applikaation asetusvalikko	6
Kuvio 8.	Facebook-applikaation App ID & App Secret -sijainti	6
Kuvio 9.	AndroidManifest.xml - Packagen nimen sijainti.....	7
Kuvio 10.	AndroidManifest.xml - LaunchModen lisääminen	8
Kuvio 11.	AndroidManifest.xml - <intent-filter> -osion lisääminen.....	8
Kuvio 12.	App.component.html -templaatti	12
Kuvio 13.	Frontend - Lopullinen kansiorakenne.....	15
Kuvio 14.	Backend - app.js.....	20
Kuvio 15.	Backend - lopullinen hakemistorakenne.....	20
Kuvio 16.	Googlen sosiaalinen kirjautuminen -näköymä.....	21
Kuvio 17.	Googlen sosiaalisen kirjautumisen tulos.....	21
Kuvio 18.	Facebookin sosiaalinen kirjautuminen -näköymä.....	22
Kuvio 19.	Facebookin sosiaalisen kirjautumisen tulos	22

1 Johdanto

Tämän raportin tarkoitus on kertoa, kuinka pystyt toteuttamaan sosiaalisen kirjautumisen NativeScriptillä. NativeScriptin kanssa tulemme käyttämään Angular-integraatiota. Palvelinpuolella teemme pienin REST-API:n, joka pystyy ottamaan vastaan HTTP-pyyntöjä käyttämällä Expressiä.

Raportti on tarkoitettu tällä hetkellä vain Android-puhelimille.

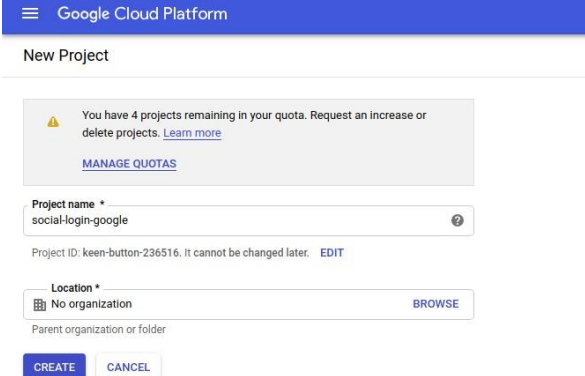
2 Versiot

Kirjasto	versio
Javac	1.8.0_201
JDK	1.8.0 191
Nativescript CLI	5.3.0-2019-02-08-12925
Nativescript	5.1.0
nativescript-oauth2	1.4.1

3 Sosiaaliset APIT

3.1 Google+

Mene osoitteeseen <https://console.cloud.google.com/apis/dashboard> ja luo uusi projekti.



Google Cloud Platform

New Project

You have 4 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)
[MANAGE QUOTAS](#)

Project name *
social-login-google

Project ID: keen-button-236516. It cannot be changed later. [EDIT](#)

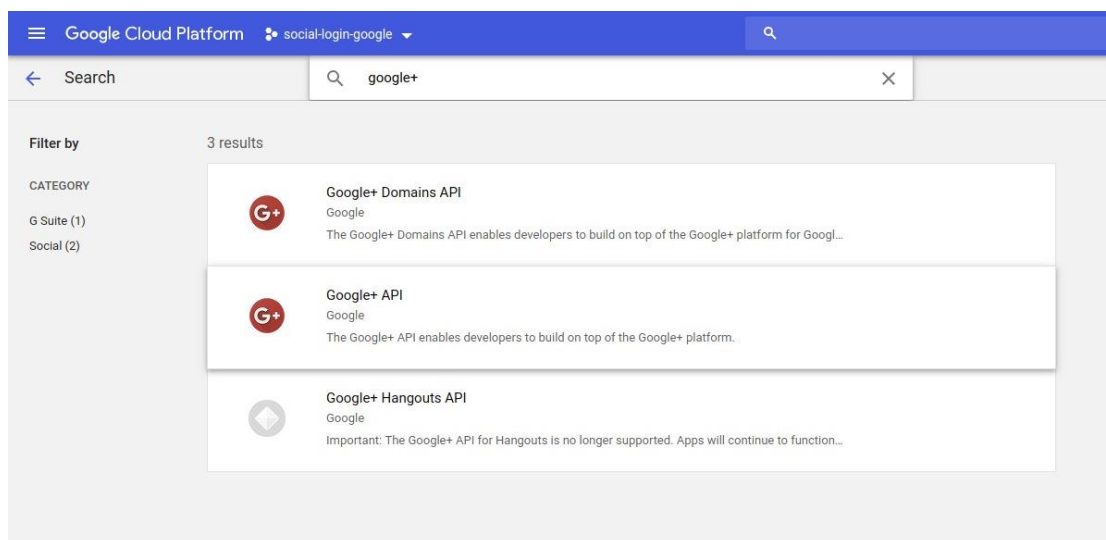
Location *
No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

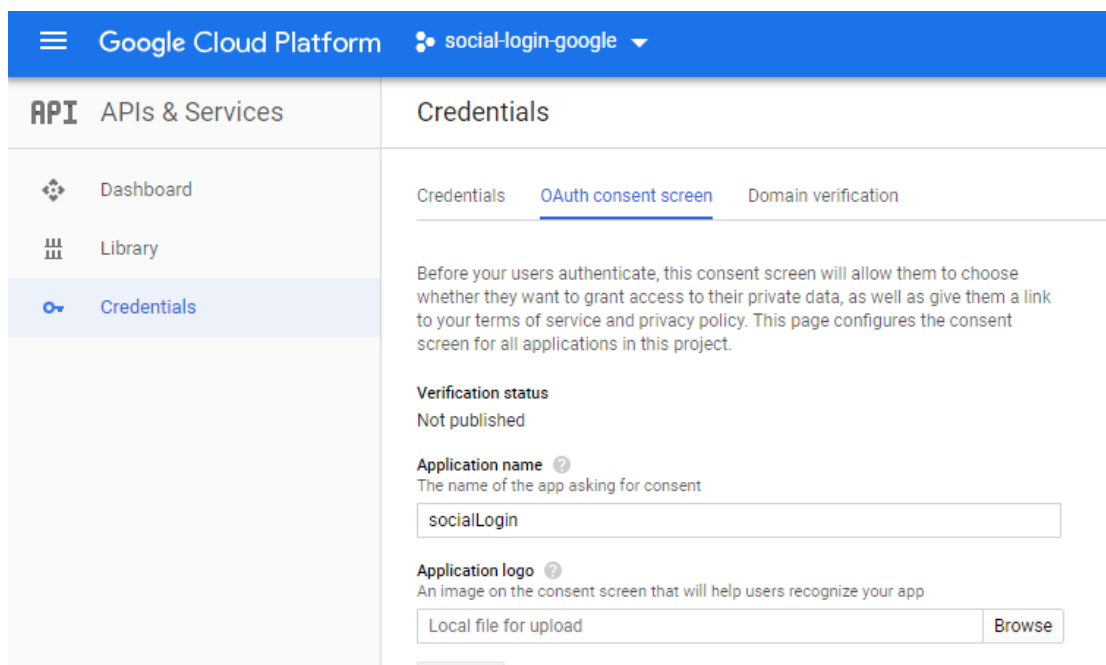
Kuvio 1. Uuden Google-projektin luominen

Kun uusi projekti on luotu, mene vasemmasta navigointipalkista *“Library”* -osioon ja valitse rajapinta *Google+ API*.



Kuvio 2. Googlen rajapinnan valinta

Paina tämän jälkeen *“Enable”*. Kun API on luotu, mene *“Credentials”* osioon ja valitse *“OAuth consent screen”* -ikkuna ja anna applikaatiolle nimi.



Kuvio 3. Applikaation nimeäminen

Kun olet nimennyt applikaation, siirry *“Credentials”* -osioon ja paina *“Create credentials”* -nappia, josta valitset kohdan *“OAuth client ID”*.

Kun pääset OAuth client ID:n sivulle, anna applikaation tyypiksi *Android*, joka avaa lomakkeen. Lomakkeesta ensimmäisen osion voit nimetä mielesi mukaan.

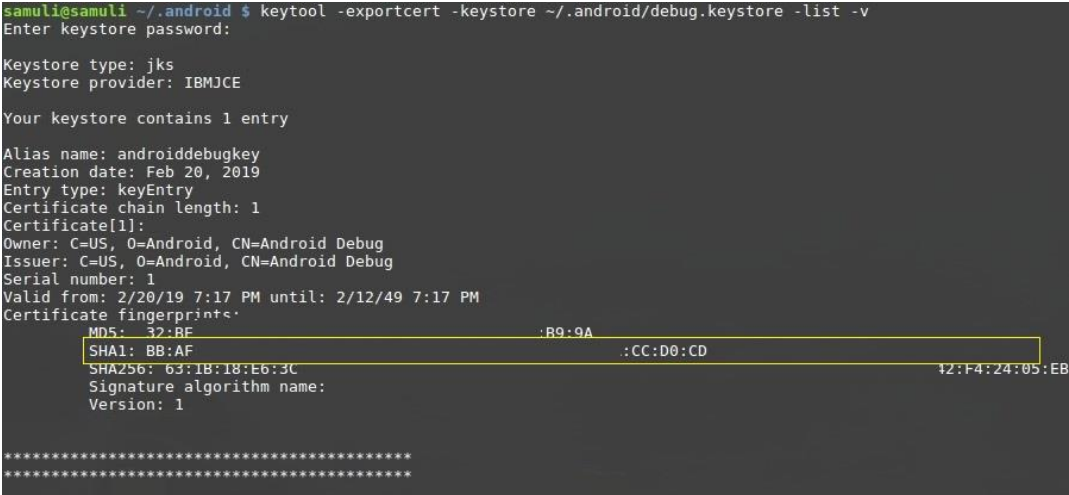
Seuraavaksi tarvitset Androidin *SHA1* sormenjäljen (*fingerprint*):

Linux/Mac

Avaa terminaali ja kirjoita komento:

```
keytool -exportcert -keystore ~/.android/debug.keystore -list -v
```

Terminaali kysyy salasanaa, joka on *"android"*. Kopioi tulostettu *SHA1* -sormenjälki ja liitä se *"Signing-certificate fingerprint"* -kohtaan.



```
samuli@samuli ~/.android $ keytool -exportcert -keystore ~/.android/debug.keystore -list -v
Enter keystore password:
Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 1 entry

Alias name: androiddebugkey
Creation date: Feb 20, 2019
Entry type: keyEntry
Certificate chain length: 1
Certificate[1]:
  Owner: C=US, O=Android, CN=Android Debug
  Issuer: C=US, O=Android, CN=Android Debug
  Serial number: 1
  Valid from: 2/20/19 7:17 PM until: 2/12/49 7:17 PM
  Certificate fingerprints:
    MD5: 32:BE :B9:9A :CC:D0:CD 42:F4:24:05:EB
    SHA1: BB:AF :CC:D0:CD 42:F4:24:05:EB
    SHA256: 63:1B:18:E6:3C
  Signature algorithm name:
  Version: 1

*****
*****
```

Kuvio 4. Androidin *SHA1*-sormenjäljen haku (Linux)

Windows

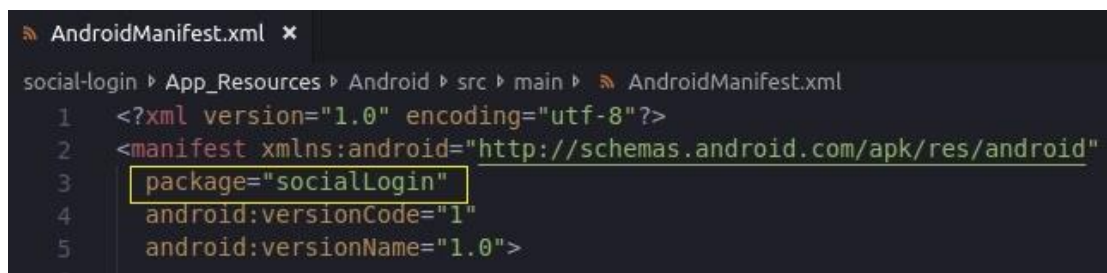
Etsi *"JDK\bin"* -hakemisto (esimerkiksi `cd C:\Program`

`Files\Java\jdkx.x.x\bin`) ja aja komento `keytool -exportcert -keystore "%USERPROFILE%\android\debug.keystore -list -v`. Salasana on *"android"*. Kopioi tulostettu *SHA1*-sormenjälki ja liitä se *"Signing-certificate fingerprint"* -kohtaan.

Lopuksi tarvitset vielä projektin nimen (*Package name*). Jos sinulla on jo valmis

Angular + NativeScript -projekti, mene projektin juuresta hakemistoon:

```
App_Resources/Android/src/main/AndroidManifest.xml
```



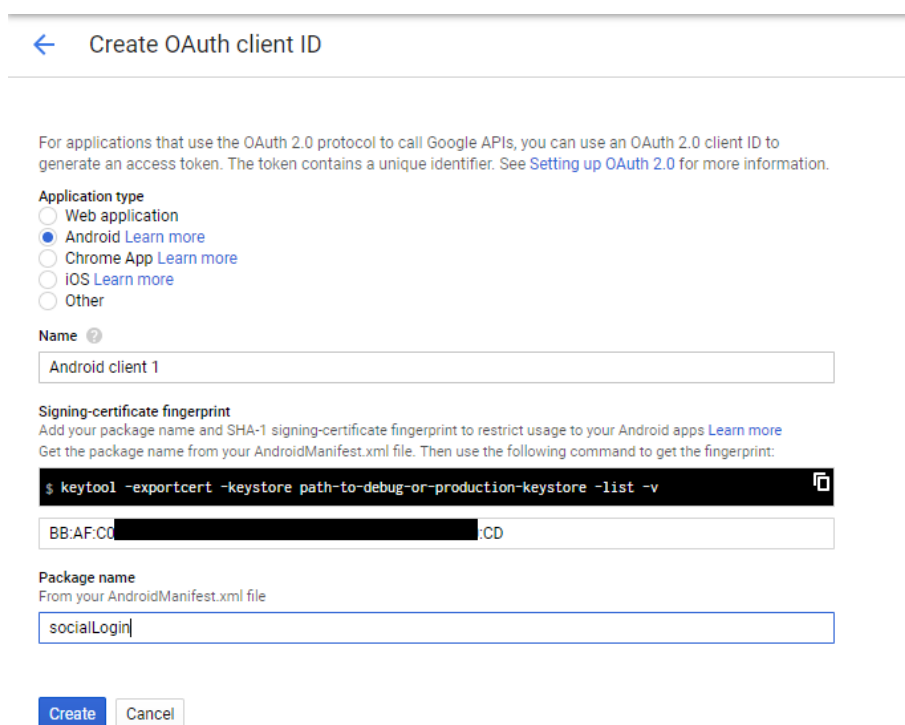
```

social-login > App_Resources > Android > src > main > AndroidManifest.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="socialLogin"
4    android:versionCode="1"
5    android:versionName="1.0">

```

Kuvio 5. NativeScript-projektin metatiedosto

Muokkaa tiedostosta rivillä 3 olevan *package* -osion nimeä (nimi on ensimmäisellä kerralla `__PACKAGE__`). Nimellä ei ole väliä, kunhan se tulee olemaan sama myös Googlen API-lomakkeessa. **Jos et ole vielä tehnyt projektia, voit liittää nimen *AndroidManifest.xml* -tiedostoon myöhemmin projektin tekovaiheessa.**



← Create OAuth client ID

For applications that use the OAuth 2.0 protocol to call Google APIs, you can use an OAuth 2.0 client ID to generate an access token. The token contains a unique identifier. See [Setting up OAuth 2.0](#) for more information.

Application type

- ☐ Web application
- ☒ Android [Learn more](#)
- ☐ Chrome App [Learn more](#)
- ☐ iOS [Learn more](#)
- ☐ Other

Name ⓘ

Android client 1

Signing-certificate fingerprint

Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps [Learn more](#)

Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:

```
$ keytool -exportcert -keystore path-to-debug-or-production-keystore -list -v
```

BB:AF:CC [redacted] :CD

Package name

From your AndroidManifest.xml file

socialLogin

Create Cancel

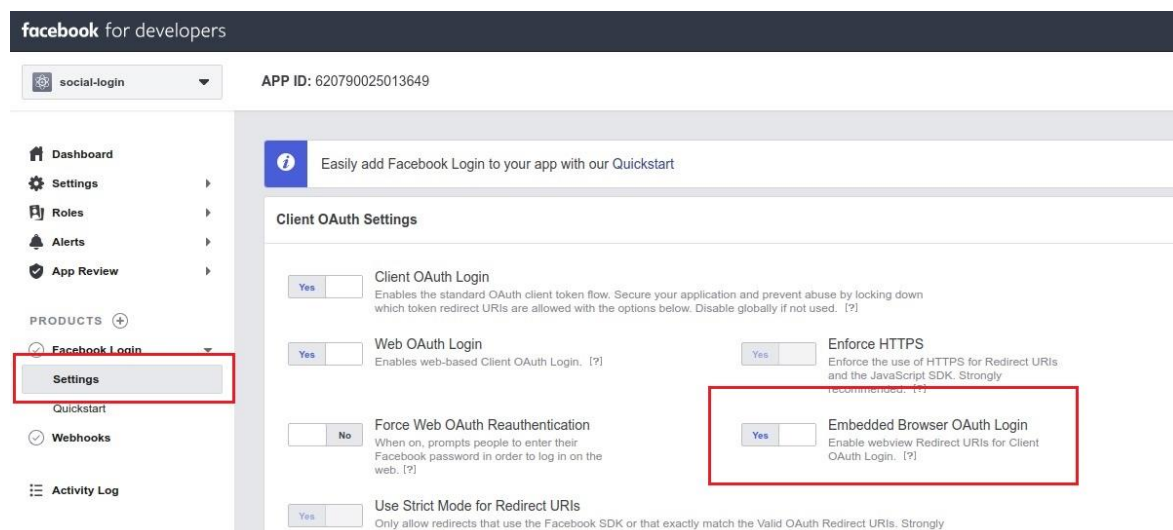
Kuvio 6. Valmis Google OAuth -lomake

Kun olet lähettänyt lomakkeen, Googlen sosiaalinen kirjautuminen on valmis. Ota **Google+ API:n Client ID** talteen.

3.2 Facebook

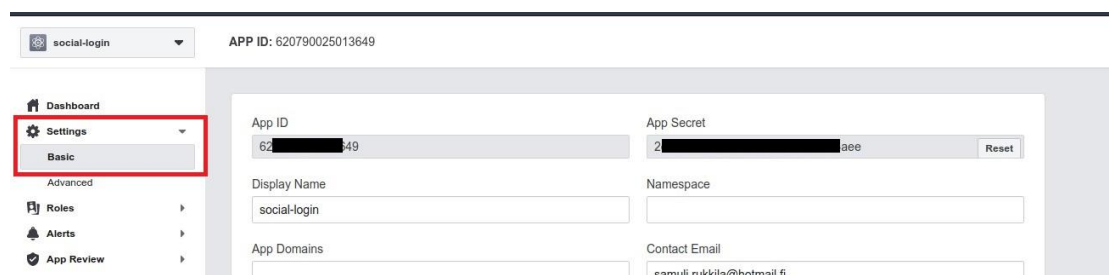
Mene hakemistoon <https://developers.facebook.com>, kirjaudu sisään ja valitse osiosta "My apps" uusi applikaatio. Anna applikaatiolle nimi ja luo uusi applikaatio. Facebook vie sinut "Select a Scenario" osioon, josta valitset "Integrate Facebook Login" -kohdan.

Luotuasi uuden applikaation mene vasemmasta navigointipalkista "Products" -otsikon alle ja valitse "Settings". Muuta asetuksista "Embedded Browser OAuth Login" todeksi.



Kuvio 7. Facebook-applikaation asetusvalikko

Mene lopuksi ylem্পään "Settings -> Basic" -osioon ja ota talteen **App ID** ja **App Secret**.



Kuvio 8. Facebook-applikaation App ID & App Secret -sijainti

4 Frontend

Luo uusi projekti: `tns create social-login --ng` (tai käytä jo valmista projektia). Jos teet uuden projektin, poista projektin luonnin jälkeen ylimääräiset komponentit (pl. *App* -komponentti).

Saadaksesi sosiaalisen kirjautumisen toimimaan *NativeScript*issä tarvitset *OAuth 2* -lisäosan. Mene hakemiston juureen ja aja komento:

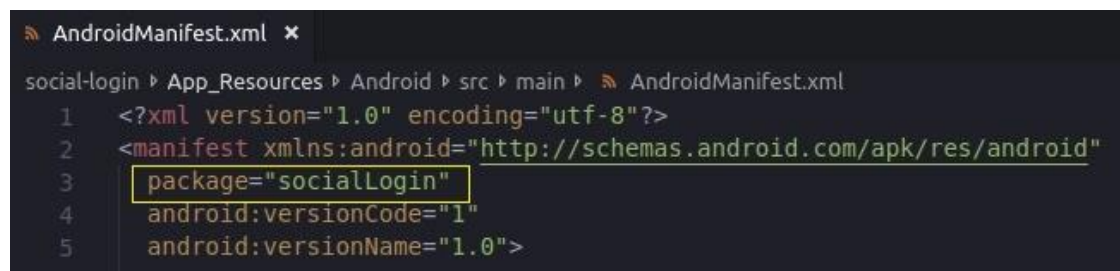
```
npm install nativescript-oauth2 --save
```

4.1 Tiedostot

AndroidManifest.xml

Googlen *social login* -API on *OpenId* sertifioitu tuottaja, joten sosiaalista loginia ei voida tehdä suoraan applikaatiossa – ainostaan selaimen kautta. Tarvitset tätä varten muutaman muokkauksen *AndroidManifest.xml* -tiedostoon.

Lisää *packagen* nimi *AndroidManifest.xml* -tiedostoon (jos et ole vielä lisännyt) hakemistossa: *App_Resources/src/main/AndroidManifest.xml*. Nimen pitää olla sama kuin mitä olet antanut *Google+ API:n credentials* -teon aikana (*package name*).



Kuvio 9. *AndroidManifest.xml* - Packagen nimen sijainti

Siirry alemmas ja lisää `<application>` tagien sisään `android:launchMode="singleTask"`.


```

15 <uses-permission android:name="android.permission.INTERNET"/>
16
17 <application
18     android:name="com.tns.NativeScriptApplication"
19     android:allowBackup="true"
20     android:icon="@drawable/icon"
21     android:label="@string/app_name"
22     android:theme="@style/AppTheme"
23     android:launchMode="singleTask">
24
25 <activity

```

Kuvio 10. *AndroidManifest.xml* - *LaunchModen* lisääminen

Lisää uusi `<intent-filter>` -osio ensimmäisen jälkeen. Tämä toimii Google *API:n* vaatimana *custom-URLina*.

```

// AndroidManifest.xml

<intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:path="/auth"
android:scheme="com.googleusercontent.apps.{GOOGLE_CLIENT_ID}"/>
</intent-filter>

```

```

31
32 <meta-data android:name="SET_THEME_ON_LAUNCH" android:resource="@style/AppTheme" />
33
34 <intent-filter>
35     <action android:name="android.intent.action.MAIN" />
36     <category android:name="android.intent.category.LAUNCHER" />
37 </intent-filter>
38
39 <intent-filter>
40     <action android:name="android.intent.action.VIEW"/>
41     <category android:name="android.intent.category.DEFAULT" />
42     <category android:name="android.intent.category.BROWSABLE" />
43     <data android:path="/auth" android:scheme="com.googleusercontent.apps.5087...ic5g"/>
44 </intent-filter>
45
46 </activity>

```

Kuvio 11. *AndroidManifest.xml* - `<intent-filter>` -osion lisääminen

auth-providers.ts

Luo uusi tiedosto *auth-providers.ts* *app* -hakemistoon. Tämä tiedosto sisältää kaikki sosiaaliseen kirjautumiseen liittyvien alustojen *API*-tiedot (Google, Facebook jne.). Google ei anna tietoturvasyistä *Client Secret* -avainta *Android*-applikaatioille. Voit

kopioida koodin seuraavalta sivulta ja liittää suoraan tiedostoosi. Muuta kummankin alustan client -tiedot omiksi tiedoiksesi.

```
// auth-providers.ts

// Tuo tarvittavat komponentit OAuth2-kirjastosta
import { configureTnsOAuth } from 'nativescript-oauth2';
import { TnsOaProvider, TnsOaProviderOptionsFacebook,
  TnsOaProviderFacebook, TnsOaProviderOptionsGoogle,
  TnsOaProviderGoogle
} from 'nativescript-oauth2/providers';

// Googlen tiedot sosiaaliseen kirjautumiseen
export function configureOAuthProviderGoogle(): TnsOaProvider {
  const googleProviderOptions: TnsOaProviderOptionsGoogle = {
    openIdSupport: 'oid-full',
    clientId:
      '{{GOOGLE_CLIENT_ID}}.apps.googleusercontent.com',
    redirectUri:
      'com.googleusercontent.apps.{{GOOGLE_CLIENT_ID}}:/auth',
    urlScheme:
      'com.googleusercontent.apps.{{GOOGLE_CLIENT_ID}}',
    scopes: ['email']
  };
  const googleProvider = new TnsOaProviderGoogle(googleProviderOptions);
  return googleProvider;
}

// Facebookin tiedot sosiaaliseen kirjautumiseen
export function configureOAuthProviderFacebook(): TnsOaProvider {
  const facebookProviderOptions: TnsOaProviderOptionsFacebook = {
    openIdSupport: 'oid-none',
    clientId: '{{FACEBOOK_CLIENT_ID}}',
    clientSecret: '{{FACEBOOK_CLIENT_SECRET}}',
    redirectUri: 'https://www.facebook.com/connect/login_success.html',
    scopes: ['email']
  };
  const facebookProvider = new
TnsOaProviderFacebook(facebookProviderOptions);
  return facebookProvider;
}

// Exporttaa kaikki sosiaaliset alustat
export function configureOAuthProviders() {
  const googleProvider = configureOAuthProviderGoogle();
  const facebookProvider = configureOAuthProviderFacebook();
  configureTnsOAuth([googleProvider, facebookProvider]);
}
```

main.ts

Tuo juuri tekemäsi *auth-providers.ts* -tiedosto *main.ts* -tiedostoon, ja suorita funktio ennen `platformNativeScriptDynamic()` -funktioita. Jos *NativeScript* + *Angular* -projektissasi on käytössä `<app-routing-module>`, ylläolevaa funktiota voi myös kutsua muusta tiedostosta ennen sosiaalista kirjautumista (esim. *servicestä*).

```
// main.ts

import { platformNativeScriptDynamic } from
  'nativescript-angular/platform';

// Kutsu tätä funktiota ensimmäisen importin jälkeen..
import { configureOAuthProviders } from './app/auth-providers';

import { AppModule } from './app/app.module';

// ..ja suorita se ennen platformNativeScriptDynamic() -funktioita
configureOAuthProviders();

platformNativeScriptDynamic().bootstrapModule(AppModule);
```

oauth-login.ts

Luo uusi tiedosto *app* -hakemistoon nimellä: *oauth-login.ts* . Kyseinen tiedosto tulee sisältämään *Promisen* -palauttavan funktion, joka suorittaa kirjautumisen sosiaaliseen alustaan.

Onnistuneessa kirjautumisessa Google palauttaa seuraavat tiedot:

```
{ "accessToken", "refreshToken", "idtoken" }
```

Ja Facebook:

```
{ "accessToken", "accessTokenExpiration",
  "refreshTokenExpiration", "idTokenExpiration" }
```

Nämä tiedot eivät ole sosiaalisen käyttäjä uniikkeja tietoja, **mutta** niiden avulla pystyt hakemaan käyttäjän tietoja.

```
// oauth-login.ts

import { TnsOAuthClient, ITnsOAuthTokenResult } from
  'nativescript-oauth2';

// providerType sisältää käyttäjän haluaman alustan ("Google",
// "Facebook" jne.)
export function tnsOAuthLogin(providerType: string):
  Promise<ITnsOAuthTokenResult> {

  const client = new TnsOAuthClient(providerType);

  return new Promise<ITnsOAuthTokenResult>((resolve, reject) => {
    client.loginWithCompletion(
      (tokenResult: ITnsOAuthTokenResult, error) => {
        if (error) {
          console.error(error);
          reject(error);
        } else {
          console.log(tokenResult);
          resolve(tokenResult);
        }
      }
    );
  });
}
```

app.component.html

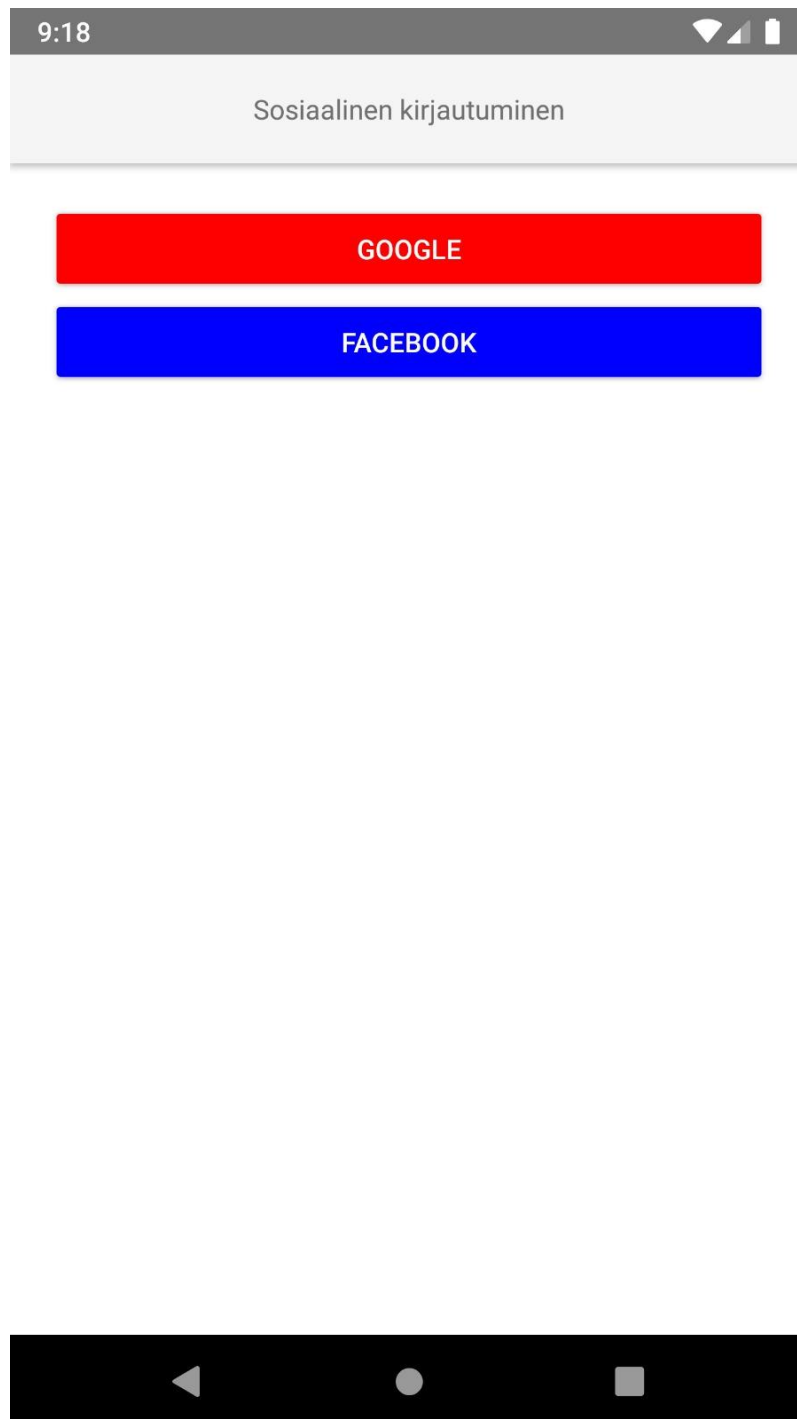
Tee templaattiin kaksi painettavaa näppäintä.

```
<!-- app.component.html -->

<ActionBar>
  <Label text='Sosiaalinen kirjautuminen'
  textAlign='center'></Label>
</ActionBar>

<StackLayout class="p-20">
  <Button text="Google" backgroundColor='red' color='white'
  (tap)="socialLogin('google')"></Button>
  <Button text="Facebook" backgroundColor='blue' color='white'
  (tap)="socialLogin('facebook')"></Button>
</StackLayout>
```

Googlen sosiaalista kirjautumista ei voi käyttää suoraan emulaattorin alkuperäisessä selaimessa. Lataa emulaattoriin Chrome/Firefox -APK, tai käytä omaa puhelintasi testaukseen.



Kuvio 12. App.component.html -templaatti

app.component.ts

Komponentin funktio kutsuu aluksi aikasemmin tehtyä `tnsOAuthLogin()` - funktiota. Jos kirjautuminen onnistuu, viedään saatu tieto serviceen, joka tekee *HTTP*-kyselyn backendiin. Onnistuneessa pyynnössä tulostetaan saatu uniikki ID-arvo.

```
// app.component.ts

import { Component } from "@angular/core";
import { tnsOAuthLogin } from "../oauth-login";
import { ITnsOAuthTokenResult } from "nativescript-oauth2";
import { AuthService } from "../auth.service";

@Component({
  selector: "ns-app",
  moduleId: module.id,
  templateUrl: "../app.component.html"
})
export class AppComponent {
  constructor(private _authService: AuthService) { }

  // Sosiaalinen kirjautumis -funktio
  public socialLogin(platform: string): void {
    // Kutsu oauth-login.ts -tiedostossa olevaa funktiota
    tnsOAuthLogin(platform)
    // Jos kirjautuminen onnistuu, palautetaan kirjautumisen tiedot
    .then((result: ITnsOAuthTokenResult) => {
      // Viedään tiedot serviceen
      this._authService.socialLogin(result, platform)
        .subscribe(res => {
          if (platform === 'google') {
            console.log('Saatiin Google-käyttäjän tietoja:');
            console.log(res);
          } else {
            console.log('Saatiin Facebook-käyttäjän tietoja:');
            console.log(res);
          }
        },
        err => {
          console.error(err);
        }
      );
    }).catch(err => {
      console.error(err);
    });
  }
}
```

auth.service.ts

Auth.servicen funktio ottaa vastaan sosiaalisen kirjautumisen tiedot. Riippuen siitä, kumpaan alustaan kirjautuminen tehdään, tekee funktio *HTTP-POST* -kyselyn, vieden kirjautumisen tulokset backendiin. Käytämme emulaattorin ja backendin välillä *NgRok* (<https://ngrok.com/>) -välitystä (käytä *HTTPS*-vaihtoehtoa!).

```
// auth.service.ts

import { Injectable } from '@angular/core';
import { ITnsOAuthTokenResult } from 'nativescript-oauth2';
import { Observable } from 'rxjs';
import { HttpHeaders, HttpClient } from '@angular/common/http';

const headers = {
  headers: new HttpHeaders({
    'Content-Type': 'application/json'
  })
};

@Injectable()
export class AuthService {

  public url = 'https://8d1d694c.ngrok.io/';

  constructor(private _http: HttpClient) { }

  // Tekee HTTP-POST kyselyn backendiin, vieden mukana samalla
  // sosiaalisen kirjautumisen tiedot (token)
  public socialLogin(result: ITnsOAuthTokenResult, platform: string):
  Observable<any> {
    // POST-url riippuu siitä kumpaan alustaan teimme kyselyn
    const route = platform === 'google' ? 'googlelogin' : 'fblogin';
    return this._http.post<any>(this.url + route, result, headers)
  }
}
```



Kuvio 13. Frontend - Lopullinen kansiorakenne

5 Backend

Palvelinpuolelle teemme pienin *Node.js* -serverin *Expressillä*, joka voi ottaa vastaan *HTTP*-kyselyitä. Tee uusi hakemisto ja hakemiston sisällä aja komento

```
npm init -y
```

Tämä luo automaattisesti *package.json* -tiedoston. Aja samassa hakemistossa komento

```
npm install express google-auth-library morgan cors  
body-parser request request-promise nodemon --save
```

Lisää *package.json* -tiedoston "scripts" -osioon uusi scripti:

```
"start": "nodemon app.js"
```

Serverin pitäisi nyt lähtä päälle komennolla `npm start`.

```
// app.js  
  
const express = require('express');  
const bodyParser = require('body-parser');  
const logger = require('morgan');  
const cors = require('cors');  
  
const app = express();  
  
app.listen(3000, () => console.log('Backend kuuntelee porttia: 3000'));  
  
app.use(cors());  
app.use(bodyParser.json({ extended: true }));  
app.use(bodyParser.urlencoded({ extended: true }));  
app.use(logger('dev'));  
  
// Ottaa vastaan Googlen sosiaalisen kirjautumisen tiedot  
app.post('/googlelogin', (req, res) => {  
  
});  
// Ottaa vastaan Facebookin sosiaalisen kirjautumisen tiedot  
app.post('/fblogin', (req, res) => {  
  
});
```

5.1 Google

Luo uusi tiedosto *backend* -hakemiston juureen **validate-google-token.js** . Sisälle luodaan funktio, joka validoi *idTokenin*. Jos validointi onnistuu, funktio palauttaa Google-käyttäjän uniikin *ID:n* sekä sähköpostiosoitteen.

```
// validate-google-token.js

const { OAuth2Client } = require('google-auth-library');

// Google+ APIn Client ID
const CLIENT_ID = '{{GOOGLE_CLIENT_ID}}.apps.googleusercontent.com';
const client = new OAuth2Client(CLIENT_ID);

// Validoi Google-käyttäjän idTokenilla
async function validateGoogleToken(token) {
  const ticket = await client.verifyIdToken({
    idToken: token,
    audience: CLIENT_ID
  });
  const payload = ticket.getPayload();
  const userid = payload['sub'];
  const email = payload['email'];
  return { userid, email };
}
module.exports = validateGoogleToken;
```

Tuo yllä tekemäsi tiedosto **app.js** -tiedostoon.

```
const validateGoogleToken = require('./validate-google-token');
```

Tämän jälkeen kutsu funktiota, kun route on **/googlelogin** tiedostossa *app.js*.

```
// app.js

// Ottaa vastaan Googlen sosiaalisen kirjautumisen tiedot
app.post('/googlelogin', (req, res) => {

  // Otetaan tullut id-token muuttujaan
  const idToken = req.body.idToken;

  // Kutsutaan validate-google-token.js -tiedostossa olevaa funktiota
  validateGoogleLogin(idToken).then(creds => {
    return res.send(creds);
  }).catch(err => {
    return res.status(500).send(err);
  });
});
```

5.2 Facebook

Luo uusi tiedosto *backend* -hakemiston juureen **validate-fb-token.js**. Sisälle luodaan funktio, joka hakee Facebook-käyttäjän tietoja Facebookin *GRAPH*-osoitteesta. Tietoja ei voida niin vain hakea, vaan tarvitsen tätä varten frontendissa saamasi *access-tokenin*.

```
// validate-fb-token.js

const rp = require('request-promise');

// Funktio validoi ja etsii käyttäjän tietoja Access-tokenin perusteella
function validateFbToken(token, callback) {
  // Tehdään request Facebookin GRAPH API-osoitteeseen
  // Onnistuneessa kyselyssä saamme ID:n sekä sähköpostin.

  rp(`https://graph.facebook.com/me?fields=id,email&access_token=${token}`
    , { json: true })
    .then(creds => {
      // Validointi sekä haku onnistui
      return callback(creds, null);
    }).catch(err => {
      console.log(err);
      // Validointi ja/tai haku epäonnistui
      return callback(null, err);
    })
  }

module.exports = validateFbToken;
```

Tuo yllä tekemäsi tiedosto **app.js** -tiedostoon.

```
const validateFbToken = require('./validate-fb-token');
```

Tämän jälkeen kutsu funktiota, kun route on **/fblogin** tiedostossa *app.js*.

```
// app.js

// Ottaa vastaan Facebookin sosiaalisen kirjautumisen tiedot
app.post('/fblogin', (req, res) => {
  // Kutsutaan validate-fb-token.js -tiedostossa olevaa funktiota
  validateFbToken(req.body.accessToken, (creds, err) => {
    if (err) {
      return res.status(500).send(err);
    }
    // Lähetään Facebookin GRAPH-APIsta saamat tiedot
    return res.send(creds);
  })
});
```

```

JS app.js x
backend ▸ JS app.js ▸ ...
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const logger = require('morgan');
4  const cors = require('cors');
5  const validateGoogleToken = require('./validate-google-token');
6  const validateFbToken = require('./validate-fb-token');
7
8  const app = express();
9
10 app.listen(3000, () => console.log('Backend kuuntelee porttia: 3000'));
11
12 app.use(cors());
13 app.use(bodyParser.json({ extended: true }));
14 app.use(bodyParser.urlencoded({ extended: true }));
15 app.use(logger('dev'));
16
17
18 // Ottaa vastaan Googlen sosiaalisen kirjautumisen tiedot
19 app.post('/googlelogin', (req, res) => {
20   // Otetaan tullut id-token muuttujaan
21   const idToken = req.body.idToken;
22
23   // Kutsutaan validate-google-token.js -tiedostossa olevaa funktiota
24   validateGoogleToken(idToken)
25     .then(creds => {
26       console.log(creds);
27       return res.send(creds);
28     })
29     .catch(err => {
30       console.log(err);
31       res.status(500).send(err)
32     });
33 });
34
35
36 // Ottaa vastaan Facebookin sosiaalisen kirjautumisen tiedot
37 app.post('/fblogin', (req, res) => {
38   // Kutsutaan validate-fb-token.js -tiedostossa olevaa funktiota
39   validateFbToken(req.body.accessToken, (creds, err) => {
40     if (err) {
41       return res.status(500).send(err);
42     }
43     // Lähetään Facebookin GRAPH-APIsta saamat tiedot
44     return res.send(creds);
45   })
46 });
47

```

Kuvio 14. Backend - app.js

```

SOCIAL-LOGIN
├─ backend
│  └─ node_modules
│     JS app.js
│     {} package-lock.json
│     {} package.json
│     JS validate-fb-token.js
│     JS validate-google-token.js

```

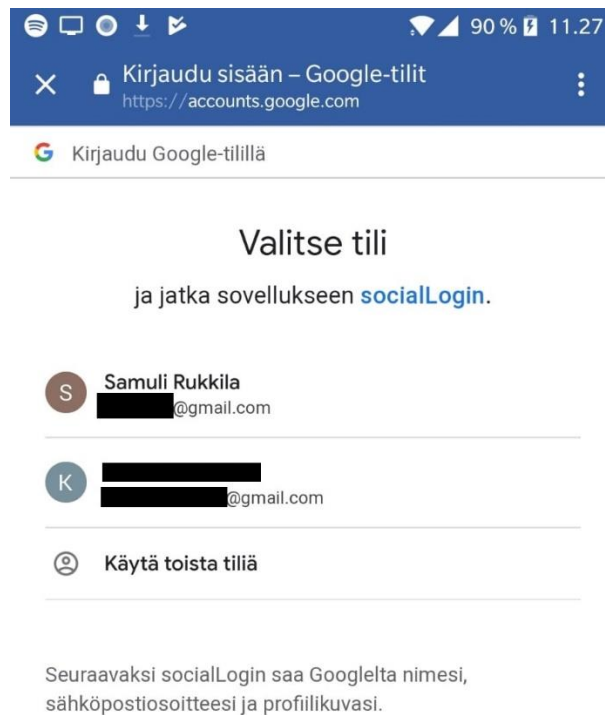
Kuvio 15. Backend - lopullinen hakemistorakenne

6 Testaus

Laita backend päälle (`npm start`) ja avaa frontend joko emulaattorissa tai omassa puhelimessa.

6.1 Google

Kokeile kirjautua frontendin templaatissa sisään painamalla Googlen nappia:



Kuvio 16. Googlen sosiaalinen kirjautuminen -näkymä

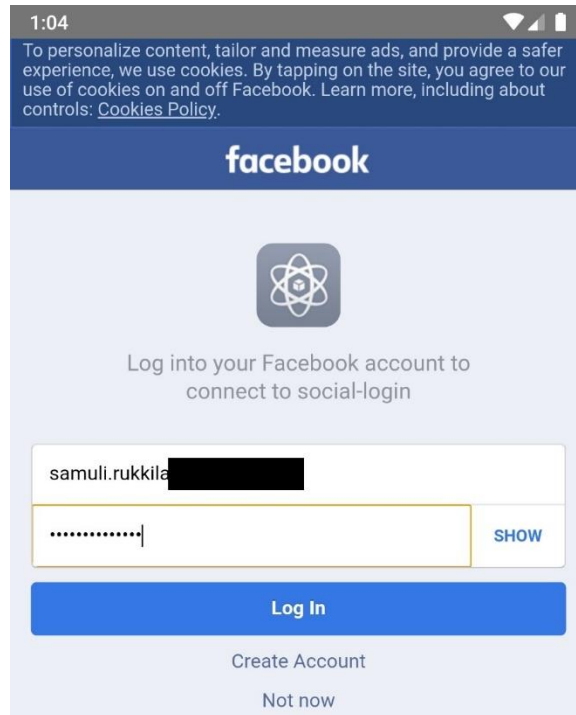
Jos kirjautuminen onnistuu, pitäisi frontendin konsoliin tulla seuraavanlainen viesti:

```
JS: Saatiin Google-käyttäjän tietoja:
JS: {
JS:   "userid": "10[REDACTED]44",
JS:   "email": "[REDACTED]@gmail.com"
JS: }
```

Kuvio 17. Googlen sosiaalisen kirjautumisen tulos

6.2 Facebook

Kokeile kirjautua frontendin templaatissa sisään painamalla Facebookin nappia:



Kuvio 18. Facebookin sosiaalinen kirjautuminen -näkymä

Jos kirjautuminen onnistuu, pitäisi frontendin konsoliin tulla seuraavanlainen viesti:

```
JS: Saatiin Facebook-käyttäjän tietoja:
JS: {
JS:   "id": "10[REDACTED]5",
JS:   "email": "samuli.rukkila[REDACTED]"
JS: }
```

Kuvio 19. Facebookin sosiaalinen kirjautumisen tulos

Lähteet

NativeScript OAuth 2 Lisäosa

<https://www.npmjs.com/package/nativescript-oauth2>

Google Developers API

<https://console.developers.google.com/apis>

Facebook Developers API

<https://developers.facebook.com/>