Node.js: come pubblicare contenuti su Linkedin tramite API

In questo articolo vedremo come pubblicare un post su Linkedin tramite API con ExpressJS.

La procedura da seguire è piuttosto articolata. Come primo step dobbiamo creare un'applicazione nella sezione sviluppatori di Linkedin. Una volta che l'applicazione è stata verificata (verified), ossia associata ad una pagina di una società, possiamo ottenere le due credenziali fondamentali, ossia il client ID ed il client secret.

Sempre nella nostra app, dobbiamo impostare un callback URL in modo che gli utenti vengano reindirizzati sul nostro sito dopo aver acconsentito ad usare la nostra applicazione.

Salviamo la configurazione in un file dedicato.

```
'use strict';

module.exports = {
    clientId: '',
    clientSecret: '',
    authorizationURL:
'https://www.linkedin.com/oauth/v2/authorization',
    accessTokenURL:
'https://www.linkedin.com/oauth/v2/accessToken',
    redirectURI: 'http://localhost:3000/callback',
    sessionName: '',
    sessionKeys: ['', '']
};
```

Creiamo quindi una classe per gestire l'interazione con le API di Linkedin.

```
'use strict';

const request = require('request');
const { clientId, clientSecret, authorizationURL,
redirectURI, accessTokenURL } = require('../config');

class API {

}

module.exports = API;
```

Il secondo step è quello di far effettuare il login agli utenti su Linkedin creando un URL specifico.

```
static getAuthorizationUrl() {
        const state = Buffer.from(Math.round(
Math.random() * Date.now() ).toString()
).toString('hex');
        const scope =
encodeURIComponent('r_liteprofile r_emailaddress
w_member_social');
        const url = `${authorizationURL}?
response_type=code&client_id=${clientId}&redirect_uri
=${encodeURIComponent(redirectURI)}&state=${state}&sc
ope=${scope}`;
        return url;
    }
```

Per renderlo effettivo, creiamo una route specifica per il redirect.

```
app.get('/auth', (req, res) => {
    res.redirect(API.getAuthorizationUrl());
});
```

Quando l'utente effettua l'autenticazione viene reindirizzato sull'URL di callback che abbiamo specificato. La query string di questo URL contiene il parametro `code` che è fondamentale per passare al terzo step, ossia la richiesta di un access token.

Nella nostra classe avremo questo metodo:

```
static getAccessToken(req) {
        const { code } = req.query;
        const body = {
            grant_type: 'authorization_code',
            code,
            redirect_uri: redirectURI,
            client_id: clientId,
            client_secret: clientSecret
        };
        return new Promise((resolve, reject) => {
            request.post({url: accessTokenURL, form:
body }, (err, response, body) =>
        {
            if(err) {
                reject(err);
            }
            resolve(JSON.parse(body));
        }
        );
        });
    }
```

La route di callback otterrà l'access token e lo salverà nella sessione corrente.

```
app.get('/callback', async (req, res) => {
    if(!req.query.code) {
        res.redirect('/');
        return;
    }
    try {
        const data = await API.getAccessToken(req);
        if(data.access_token) {
            req.session.token = data.access_token;
            req.session.authorized = true;
        }
        res.redirect('/');
    } catch(err) {
        res.json(err);
    }
});
```

L'access token da solo non basta per pubblicare un post su Linkedin. È necessario infatti ottenere anche l'ID dell'utente per collegare il post ad un profilo specifico. Nella nostra classe dobbiamo quindi definire un metodo per il reperimento di tale ID.

```
static getLinkedinId(req) {
        return new Promise((resolve, reject) => {
            const url =
'https://api.linkedin.com/v2/me';
            const headers = {
                'Authorization': 'Bearer ' +
req.session.token,
                'cache-control': 'no-cache',
                'X-Restli-Protocol-Version': '2.0.0'
```

```
            };

            request.get({ url: url, headers: headers
}, (err, response, body) => {
                if(err) {
                    reject(err);
                }
                resolve(JSON.parse(body).id);
            });
        });
    }
```

La view principale mostrerà il pulsante di login se l'utente non è autenticato o il form per la creazione del post. Tale form avrà come campi il titolo del post, il testo del post, il link da condividere e l'URL dell'immagine associata al post. Avrà come campo nascosto l'ID dell'utente.

```
app.get('/', async (req, res) => {
    const isAuthorized = (req.session.authorized);
    if(!isAuthorized) {
        res.render('index', { isAuthorized, id: ''
});
    } else {
        try {
            const id = await API.getLinkedinId(req);
            res.render('index', { isAuthorized, id
});
        } catch(err) {
            res.send(err);
        }
    }
});
```

Infine, per pubblicare un post dobbiamo combinare insieme l'access token e l'ID dell'utente unitamente ai dati raccolti dal form.

```
static publishContent(req, linkedinId, content) {
        const url =
'https://api.linkedin.com/v2/shares';
        const { title, text, shareUrl,
shareThumbnailUrl } = content;
        const body = {
            owner: 'urn:li:person:' + linkedinId,
            subject: title,
            text: {
                text: text
            },
            content: {
                contentEntities: [{
                    entityLocation: shareUrl,
                    thumbnails: [{
                        resolvedUrl:
shareThumbnailUrl
                    }]
                }],
                title: title
            },
            distribution: {
                linkedInDistributionTarget: {}
            }
        };
        const headers = {
            'Authorization': 'Bearer ' +
req.session.token,
            'cache-control': 'no-cache',
            'X-Restli-Protocol-Version': '2.0.0',
            'x-li-format': 'json'
```

```
        };

        return new Promise((resolve, reject) => {
            request.post({ url: url, json: body,
headers: headers}, (err, response, body) => {
                if(err) {
                    reject(err);
                }
                resolve(body);
            });
        });

    }
```

La nostra route pubblicherà i contenuti solo dopo che i valori del form supereranno la procedura di validazione.

```
app.post('/publish', async (req, res) => {
    const { title, text, url, thumb, id } = req.body;
    const errors = [];

    if(validator.isEmpty(title)) {
        errors.push({ param: 'title', msg: 'Invalid
value.'});
    }
    if(validator.isEmpty(text)) {
        errors.push({ param: 'text', msg: 'Invalid
value.'});
    }
    if(!validator.isURL(url)) {
        errors.push({ param: 'url', msg: 'Invalid
value.'});
    }
```

```
    if(!validator.isURL(thumb)) {
        errors.push({ param: 'thumb', msg: 'Invalid
value.'});
    }

    if(errors.length > 0) {
        res.json({ errors });
    } else {
        const content = {
            title: title,
            text: text,
            shareUrl: url,
            shareThumbnailUrl: thumb
        };

        try {
            const response = await
API.publishContent(req, id, content);
            res.json({ success: 'Post published
successfully.' });
        } catch(err) {
            res.json({ error: 'Unable to publish your
post.' });
        }
    }
});
```

Se non si sono verificati errori, il post verrà pubblicato sul profilo dell'utente.

Codice sorgente

GitHub