Node.js: interagire con le API dei domini di Hexonet

In questo articolo vedremo come ottenere informazioni sui domini utilizzando le API di Hexonet con Node.js.

Per testare le Domain API di Hexonet abbiamo bisogno di un account di prova. Una volta ottenuto un nome utente ed impostata una password, dobbiamo ricordarci di specificare l'accesso all'area di test usando il valore `1234` nele parametro che andremo a salvare in in file `.env` insieme con le altre credenziali.

```
HEXONET_LOGIN=login
HEXONET_PW=password
HEXONET_ENTITY=1234
```

Definiamo una funzione helper per effettuare richieste HTTP POST verso le API di Hexonet.

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

const post = ({ path, data }) => {
    const query = new
URLSearchParams(data).toString();
    const options = {
        hostname: 'api.ispapi.net',
        port: 443,
        path: path,
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-
```

```
urlencoded',
            'Content-Length': query.length
        }
    };

    return new Promise((resolve, reject) => {
        const request = https.request(options,
response => {
            const status =
parseInt(response.statusCode, 10);
            if(status !== 200) {
                reject({ error: status });
            }
            response.on('data', d => {
              resolve(d.toString());
            });
        });

        request.on('error', error => {
            reject(error);
        });

        request.write(query);
        request.end();
    });
};
```

Quindi definiamo una classe per interagire con le API.

```
class Hexonet {
    constructor({ login, password, entity }) {
        this.login = login;
        this.password = password;
        this.entity = entity;
```

```
        this.apiData = {
            s_entity: this.entity,
            s_pw: this.password,
            s_login: this.login

        };
    }

    getDescription(res) {
        if(typeof res !== 'string' || res.length ===
0) {
            return '';
        }
        const parts = res.split(/\n/);
        if(parts.length === 0) {
            return '';
        }
        const descPart = parts.find(part => { return
part.includes('DESCRIPTION'); });
        if(!descPart) {
            return '';
        }
        return descPart.split('=')[1];
    }

    async checkDomain(domain) {
        this.apiData.command = 'CheckDomain';
        this.apiData.domain = domain;
        try {
            const response = await post({
                path: '/api/call.cgi',
                data: this.apiData
            });
            return this.getDescription(response);
```

```
        } catch(err) {
            return err;
        }
    }
}

module.exports = Hexonet;
```

Hexonet restituisce una risposta in formato testuale in cui le informazioni che ci interessano sono contenute nella riga con la voce `DESCRIPTION`. Volendo potremmo ampliare le funzionalità della classe per interpretare anche la riga con la voce `CODE` che restituisce il codice di stato delle API. Non abbiamo fatto altro che una ricerca in un array di righe creato a partire dal delimitatore comune `\n`.

Esempio d'uso:

```
'use strict';

require('dotenv').config();

const Hexonet = require('./lib/Hexonet');
const Hex = new Hexonet({
    login: process.env.HEXONET_LOGIN,
    password: process.env.HEXONET_PW,
    entity: process.env.HEXONET_ENTITY
});

const init = async () => {
    try {
        const response = await
Hex.checkDomain('gabrieleromanato.net');
        console.log(response);
    } catch(err) {
```

```
        console.log(err);
    }
};

init();
```