PUBLICLOOP (v0.3.0)

Vývoj skvělé, neuvěřitelné, úžasné a spektakulární aplikace pro vyhledávání spojení mezi bodem A a B ve veřejném prostoru krok za krokem.

Automatický restart Backend serveru

Pro otestování každé změny zdrojových souborů backendu aplikace je nutné ukončit běh node.js skriptu a skript znovu spustit. Pro automatický restart backend serveru vyvolaný změnou zdrojového kódu použijeme nodemoon.

```
{publicloop_root}/app/backend/npm install nodemon
```

Dále si do souboru {publicloop_root}/app/backend/package.json přidáme do sekce skript položky start a dev.

```
...
"scripts": {
  "start": "node index.js",
  "dev": "npx nodemon index.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
...
```

> Backend aplikace lze spustit v módu bez automatického restartu (start) nebo s automatickým restartem při změně (dev).
>
> ```
> npm run start
> npm run dev
> ```

Pro usnadnění spouštění backendu si můžeme upravit dávku {publicloop_root}/start-backend-app.bat.

```
@echo off
setlocal

set arg=%1
echo PublicLoop is trying to start - please wait  ...

if not defined arg goto :dev
if %arg%==start (goto start) else (goto unsupported)
```

```
:dev
cd app/backend
npm run dev
goto finish

:start
cd app/backend
npm run start
goto finish

:unsupported
echo Unsupported argument
goto finish

if errorlevel 0 goto finish
if errorlevel 1 goto error
goto finish

:error
echo.
echo PublicLoop could not be started

:finish
pause
```

Vytvoření relace mezi klientem a serverem

Abychom na backendu naší aplikace mohli rozlišit jednotlivé uživatele, musíme mezi klientem a serverem vytvořit relaci, která nám pomůže překonat bezestavovost protokolu HTTP (viz https://en.wikipedia.org/wiki/HTTP#HTTP_session). Vytvoření relace se skládá z následujících kroků:

1. Při zpracování každého klientského požadavku ověříme, zda klient zasílá v HTTP požadavku jedinečný identifikátor. Pokud ne, vytvoříme ho.
2. Načteme data ze souboru, jehož název je stejný jako jedinečný identifikátor. Pokud soubor neexistuje, vytvoříme ho.
3. Pokud jsme vygenerovali nový uid, tak ho klientovi zašleme.

Krok 1

Pro načtení hlaviček HTTP požadavku si vytvoříme v souboru {publicloop_root}/app/backend/index.js funkci parseCookies.

```
function parseCookies (request) {
    const list = {};
    const cookieHeader = request.headers?.cookie;
    if (!cookieHeader) return list;

    cookieHeader.split(`;`).forEach(function(cookie) {
        let [ name, ...rest] = cookie.split(`=`);
```

```
        name = name?.trim();
        if (!name) return;
        const value = rest.join(`=`).trim();
        if (!value) return;
        list[name] = decodeURIComponent(value);
    });

    return list;
}
```

Tuto funkci použijeme pro načtení Cookie session_id. Pokud neexistuje, tak uid vytvoříme.

```
const crypto = require('crypto');
const server = http.createServer((req, res) => {
  let uid = null;
  const cookies = parseCookies(req);
  if (typeof cookies.session_id != 'undefined') {
    uid = cookies.session_id;
  } else {
    uid = (crypto.randomUUID());
  }
...
}
```

> Balíček crypto si musíme do aplikace doinstalovat.
>
> ```
> npm install crypto
> ```

Krok 2

Pokud existuje uid, načteme soubor uid z úložiště. Pokud neexistuje, tak soubor vytvoříme.

```
const fs = require('fs');
const path = require('path');
const file = path.resolve(__dirname, '..', 'tmp_session/',uid);
let session_data = {session_id:uid,data: {}};
try {
  session_data = JSON.parse(fs.readFileSync(file, 'utf8'));
} catch (err) {
  fs.writeFileSync(file, JSON.stringify(session_data), { flag: 'ax' });
}
```

> Balíček fs a path si musíme do aplikace doinstalovat.
>
> ```
> npm install fs
> ```

```
npm install path
```

> Musíme si také na file systému vytvořit adresář {publicloop_root}/app/tmp_session. Z důvodu bezpečnosti si adresář tmp_session vytvoříme mimo publikační strom frondendu i backendu.

Krok 3

Klientovi zašleme session_id v cookie v hlavičce HTTP odpovědi (https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie).

```
var session_cookie = 'session_id=' + uid+'; path=/; Secure; Max-Age=9000; SameSite=None'
res.setHeader('Set-Cookie', session_cookie);
```

Řízení přístupu

Práce s daty session souboru

Nyní máme možnost ukládat na straně serveru klientova data (identifikována prostřednictvím UID). Příkladem může být uložení user_key do session souboru po úspěšném přihlášení uživatele do aplikace.

```
case '/user/signin':
    if(typeof session_data.data.user_key != 'undefined') {
        res.statusCode = 200;
        res.setHeader('Content-Type', 'application/json');
        res.setHeader('Set-Cookie', session_cookie);
        res.end(JSON.stringify({signin: true}));
        break;
    }

    var body = '';  //get request body
    req.on('data', chunk => {
        body += chunk.toString(); // convert Buffer to string
    });
    req.on('end', () => {
        let reqData = JSON.parse(body);
        let username = reqData.email;
        let password = reqData.password;
        var connection = getConn();
        connection.query(
            'SELECT *  FROM users WHERE email=? and password = ?',
            [username, password],
            function (err, results) {
                if (err) {
                    res.statusCode = 500;
                    res.setHeader('Content-Type', 'text/plain');
                    res.end('Internal Server Error');
```

```
                } else {
                    var exist = false;
                    var user_key = null;
                    results.forEach(function (item) {
                        exist = true;
                        user_key = item.user_key;
                    });
                    if (exist) {
                        session_data.data ['user_key'] = user_key;
                        fs.writeFileSync(file, JSON.stringify(session_data), {flag: ''});
                    }
                    var resData = {login: exist}
                    res.statusCode = 200;
                    res.setHeader('Content-Type', 'application/json');
                    res.setHeader('Set-Cookie', session_cookie);
                    res.end(JSON.stringify(resData));
                }
            }
        );
    });
break;
```

Ověření přihlášeného uživatele na stránce signin.html

Na stránce signin si nejdříve ověříme, jestli už není uživatel přihlášen. Vytvoříme si loader, který bude zobrazen po dobu realizace dotazu na backend. Do souboru {publicloop_root}/app/frondend/css/styles.css si vložíme následující zdrojový kód.

```
#loader {
    display: none;
    position: absolute;
    width: 100vw;
    height: 100vh;
    background:rgba(0,0,0,0.7);
    z-index: 5000;
}
.circle {
    position: absolute;
    top: calc(50vh - 58px);
    left: calc(50vw - 58px);
    border: 16px solid #f3f3f3; /* Light grey */
    border-top: 16px solid #3498db; /* Blue */
    border-radius: 50%;
    width: 120px;
    height: 120px;
    animation: spin 2s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
```

A do souboru {publicloop_root}/app/frondend/signin.html si vložíme:

```
<div id="loader">
    <div class="circle"></div>
</div>
```

Do souboru {publicloop_root}/app/frondend/js/app.js si přidáme obslužné funkce.

```
function isUserSignin(session_id) {
    return new Promise((resolve) => {
        if (session_id != null) {
            fetch('http://localhost/backend/user/signin', {
                method: "POST",
                headers: {'Content-Type': 'application/json'},
                body: JSON.stringify({})
            })
                .then((response) => {
                    if (!response.ok) {
                        throw new Error(`HTTP error, status = ${response.status}`);
                    }
                    return response.text();
                })
                .then((text) => {
                    resolve(text);
                })
                .catch(error => {
                    //TODO handle error
                })
        } else {
            resolve(false);
        }
    });
}

function getCookie(name) {
    var dc = document.cookie;
    var prefix = name + "=";
    var begin = dc.indexOf("; " + prefix);
    if (begin == -1) {
        begin = dc.indexOf(prefix);
        if (begin != 0) return null;
    } else {
        begin += 2;
        var end = document.cookie.indexOf(";", begin);
        if (end == -1) {
            end = dc.length;
        }
    }
    return decodeURI(dc.substring(begin + prefix.length, end));
}
```

A do souboru {publicloop_root}/app/frondend/signin.html si vložíme skript pro volání funkce isUserSignin(session_id).

```
<script>
    window.addEventListener('DOMContentLoaded', function () {
        var session_id = getCookie('session_id');
        document.getElementById("loader").style.display = 'block';
        isUserSignin(session_id).then((response) => {
            var res = JSON.parse(response);
            if (res.signin) {
                window.location.href = 'http://localhost/map.html'
            } else {
                document.getElementById("loader").style.display = 'none';
            }
        })
    })
</script>
```

Ověření přihlášeného uživatele na stránce map.html

Na stránce map.html si ověříme, zda je uživatel přihlášen. Pokud uživatel nebude přihlášen, přesměrujeme ho na stránku {publicloop_root}/app/frondend/signin.html.
Toto ověření provedeme naprosto stejně jako v předcházejícím příkladu. Na stránce {publicloop_root}/app/frondend/map.html si přidáme loader a skript pro dotaz na server.

```
<div id="loader">
    <div class="circle"></div>
</div>
<script>
    window.addEventListener('DOMContentLoaded', function () {
        var session_id = getCookie('session_id');
        if (session_id != null) {
            document.getElementById("loader").style.display = 'block';
            isUserSignin(session_id).then((response) => {
                var res = JSON.parse(response);
                if (res.signin) {
                    document.getElementById("loader").style.display = 'none';
                } else {
                    window.location.href = 'http://localhost/signin.html'
                }
            })
        } else {
            window.location.href = 'http://localhost/signin.html'
        }
    })
</script>
```

Odhlášení uživatele

Na stránku {publicloop_root}/app/frondend/map.html si přidáme tlačítko pro odhlášení uživatele.

```
<a id="signOut" href="#">Sign out</a>
<script>
    document.getElementById("signOut").addEventListener('click', function (event) {
        event.preventDefault();

        fetch('http://localhost/backend/user/signout', {
            Method: 'GET'
        })
            .then((response) => {
                if (!response.ok) {
                    throw new Error(`HTTP error, status = ${response.status}`);
                }
                return response.text();
            })
            .then((text) => {
                let retObj = JSON.parse(text);
                if (retObj.signout) {
                    window.location.href = 'http://localhost/signin.html'
                } else {
                    //TODO something wrong
                }
            })
            .catch((error) => {
                var p = document.createElement("p");
                p.appendChild(document.createTextNode(`Error: ${error.message}`));
                document.body.innerHTML = "";
                document.body.appendChild(p);
            });
    })
</script>
```

Na straně backendu si přidáme routu /user/signout.

```
case '/user/signout':
    if(typeof session_data.data.user_key != 'undefined') {
        try {
            session_data.data = {};
            fs.writeFileSync(file, JSON.stringify(session_data), {flag: ''});

            res.statusCode = 200;
            res.setHeader('Content-Type', 'application/json');
            res.end(JSON.stringify({signout: true}));
        } catch (e) {
            res.statusCode = 200;
            res.setHeader('Content-Type', 'application/json');
            res.end(JSON.stringify({signout: false}));
        }
    } else {
        res.statusCode = 200;
        res.setHeader('Content-Type', 'application/json');
        res.end(JSON.stringify({signout: false}));
```

```
    }
break;
```