PUBLICLOOP (v0.5.0)

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

Datová vrstva aplikace Publicloop

Vytvoříme si tabulky potřebné pro evidenci jízdního řádu.

```
CREATE TABLE IF NOT EXISTS `publicloop`.`lines`
(
    `line_key`  INT          NOT NULL AUTO_INCREMENT,
    `line_name` VARCHAR(255) NOT NULL,
    PRIMARY KEY (`line_key`)
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `publicloop`.`stops`
(
    `stop_key`  INT          NOT NULL AUTO_INCREMENT,
    `stop_name` VARCHAR(255) NOT NULL,
    `latitude`  DOUBLE       NOT NULL,
    `longitude` DOUBLE       NOT NULL,
    `place`     VARCHAR(255) NOT NULL,
    PRIMARY KEY (`stop_key`)
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `publicloop`.`line_stops`
(
    `linestop_key`  INT NOT NULL AUTO_INCREMENT,
    `line_key`      INT NOT NULL,
    `stop_key`      INT NOT NULL,
    `direction`     INT NOT NULL,
    `order`         INT NOT NULL,
    PRIMARY KEY (`linestop_key`),
    INDEX `fk_line_stops_lines_idx` (`line_key` ASC) VISIBLE,
    INDEX `fk_line_stops_stops1_idx` (`stop_key` ASC) VISIBLE,
    CONSTRAINT `fk_line_stops_lines`
        FOREIGN KEY (`line_key`)
            REFERENCES `publicloop`.`lines` (`line_key`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION,
    CONSTRAINT `fk_line_stops_stops1`
        FOREIGN KEY (`stop_key`)
            REFERENCES `publicloop`.`stops` (`stop_key`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
)
ENGINE = InnoDB;


CREATE TABLE IF NOT EXISTS `publicloop`.`paths`
(
    `path_key`      INT NOT NULL AUTO_INCREMENT,
    `stop_key_from` INT NOT NULL,
    `stp_key_to`    INT NOT NULL,
    PRIMARY KEY (`path_key`)
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `publicloop`.`points`
(
    `point_key` INT   NOT NULL AUTO_INCREMENT,
    `latitude`  FLOAT NOT NULL,
    `longitude` FLOAT NOT NULL,
    PRIMARY KEY (`point_key`)
)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `publicloop`.`path_points`
(
    `pathpoint_key` INT NOT NULL AUTO_INCREMENT,
    `path_key`      INT NOT NULL,
    `point_key`     INT NOT NULL,
    `order`         INT NOT NULL,
    PRIMARY KEY (`pathpoint_key`),
    INDEX `fk_path_points_paths1_idx` (`path_key` ASC) VISIBLE,
    INDEX `fk_path_points_points1_idx` (`point_key` ASC) VISIBLE,
    CONSTRAINT `fk_path_points_paths1`
        FOREIGN KEY (`path_key`)
            REFERENCES `publicloop`.`paths` (`path_key`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION,
    CONSTRAINT `fk_path_points_points1`
        FOREIGN KEY (`point_key`)
            REFERENCES `publicloop`.`points` (`point_key`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
)
ENGINE = InnoDB;
```

Vytvoříme si skript {publicloop_root}/app/backend/import.js pro naplnění tabulek ukázkovými daty.

```
const mysql = require("mysql2/promise");
const fetch = require("node-fetch");

const lines = [
    '5', '13'
];

const line_stops_position = [
    ['5', 'A',
        [
            ['Dukla,točna', 15.7587522, 50.0210234],
            ['Dukla,náměstí', 15.7586374, 50.0240244],
            ['Teplého', 15.7611141, 50.0268243],
            ['Domov mládeže', 15.7663822, 50.0270761],
            ['Jana Palacha', 15.7707138, 50.0285677],
            ['17.listopadu', 15.7703028, 50.0333627],
            ['Masarykovo náměstí', 15.7695389, 50.0379137],
            ['Náměstí Republiky', 15.7773819, 50.0374177],
            ['Krajský úřad', 15.7820969, 50.0381006],
            ['Sakařova', 15.7871599, 50.0389131],
            ['Holubova', 15.7905741, 50.0398668],
            ['Bezdíčkova', 15.7956009, 50.0412973],
            ['Židov', 15.7991791, 50.0412668],
            ['Dubina,garáže', 15.8032742, 50.0420717],
            ['Dubina,centrum', 15.8073502, 50.0448755],
            ['Dubina,sever', 15.8114271, 50.0470595]
        ]
    ],
    ['5', 'B',
        [
            ['Dubina,sever', 15.8117964, 50.0461022],
            ['Dubina,centrum', 15.8067303, 50.0446390],
            ['Dubina,garáže', 15.8031254, 50.0421709],
            ['Židov', 15.7997923, 50.0409654],
            ['Bezdíčkova', 15.7958479, 50.0414575],
            ['Holubova', 15.7913580, 50.0401910],
            ['Sakařova', 15.7863016, 50.0387529],
            ['Krajský úřad', 15.7809601, 50.0379861],
            ['Náměstí Republiky', 15.7770977, 50.0384935],
            ['Masarykovo náměstí', 15.7694921, 50.0373529],
            ['17.listopadu', 15.7700863, 50.0334772],
            ['Jana Palacha', 15.7705622, 50.0279955],
            ['Domov mládeže', 15.7666960, 50.0271867],
            ['Teplého', 15.7612953, 50.0269998],
            ['Lexova', 15.7584686, 50.0263513],
            ['Dukla,náměstí', 15.7585001, 50.0234102],
            ['Dukla,točna', 15.7585001, 50.0217627]
        ]
    ],
    ['13', 'A',
        [
            ['Ohrazenice,točna', 15.7506206, 50.0617704],
            ['Ohrazenice,Semtínská', 15.7541618, 50.0620530],
            ['Ohrazenice,škola', 15.7573872, 50.0617936],
            ['Globus', 15.7549706, 50.0590356],
            ['Trnová', 15.7570572, 50.0584672],
            ['Poděbradská', 15.7622318, 50.0547899],
            ['Polabiny,točna', 15.7595215, 50.0513223],
            ['Polabiny,Kosmonautů', 15.7600117, 50.0492700],
            ['Polabiny,hotel', 15.7630043, 50.0477518],
            ['Stavařov', 15.7666035, 50.0453943],
            ['Zimní stadion', 15.7676811, 50.0424493],
            ['Sukova', 15.7703161, 50.0390047],
            ['Náměstí Republiky', 15.7773819, 50.0374177],
            ['Krajský úřad', 15.7820969, 50.0381006],
            ['U Kostelíčka', 15.7862148, 50.0381540],
            ['Na Okrouhlíku', 15.7939396, 50.0354150],
            ['Na Drážce', 15.7986202, 50.0372575],
            ['Dubina,garáže', 15.8032742, 50.0420717],
            ['Dubina,centrum', 15.8073502, 50.0448755],
            ['Dubina,sever', 15.8112125, 50.0463361]
        ]
    ],
    ['13', 'B',
        [
            ['Dubina,sever', 15.8093181, 50.0455430],
            ['Dubina,centrum', 15.8067303, 50.0446390],
            ['Dubina,garáže', 15.8031254, 50.0421709],
            ['Na Drážce', 15.7984571, 50.0372003],
            ['Na Okrouhlíku', 15.7929726, 50.0360559],
            ['Krajský úřad', 15.7809601, 50.0379861],
            ['Náměstí Republiky', 15.7770977, 50.0384935],
            ['Zimní stadion', 15.7687788, 50.0401529],
            ['Stavařov', 15.7670937, 50.0450052],
            ['Polabiny,hotel', 15.7635155, 50.0479501],
            ['Polabiny,Kosmonautů', 15.7603168, 50.0489534],
            ['Polabiny,točna', 15.7605963, 50.0521348],
            ['Poděbradská', 15.7626057, 50.0546335],
            ['Trnová', 15.7583113, 50.0582956],
            ['Globus', 15.7548580, 50.0592111],
            ['Ohrazenice,škola', 15.7574835, 50.0620607],
            ['Ohrazenice,Semtínská', 15.7535925, 50.0620683],
            ['Ohrazenice,točna', 15.7506206, 50.0617704]
        ]
    ]
];

startSync();

async function startSync() {

    await loadLines();

    await loadStops();

    process.exit();
}

async function loadLines() {
    var connection = await getConn();
    for (let line of lines) {
        try {
            await connection.query(
            'INSERT INTO `lines` VALUES (null, ?)',
            [line]
            );
        } catch (err) {
            console.log(err);
        }
    }
```

```
}

async function loadStops() {
    var connection = await getConn();

    for (let line_stop of line_stops_position) {
        var line_name = line_stop[0];

        const [results_lines] = await connection.query(
            'SELECT line_key FROM `lines` WHERE line_name = ?',
            [line_name]
        );

        var line_key = null;
        for (const line of results_lines) {
            line_key = line.line_key;
        }

        var line_direction = line_stop[1];
        let direction = 1;
        if (line_direction == 'B') {
            direction = 2;
        }

        var index = 1;
        for (let stop of line_stop[2]) {
            try {
                const [results] = await connection.query(
                    'INSERT INTO stops VALUES (null, ?, ?, ?, ?)',
                    [stop[0], stop[1], stop[2], line_direction]
                );
                stop_key = results.insertId;

            } catch (err) {
                console.log(err);
            }
            try {
                await connection.query(
                    'INSERT INTO line_stops VALUES (null, ?, ?, ?, ?)',
                    [line_key, stop_key, direction, index]
                );
            } catch (err) {
                console.log(err);
            }
            index++;
        }
    }
}

function getConn() {
    return mysql.createConnection({
        host: 'localhost',
        user: 'publicloop',
        password: 'publiclooppassword',
        database: 'publicloop',
    });
}
```

Do projektu si doinstalujeme balíček node-fetch.

```
npm install node-fetch@2.6.1
```

Sktipt si spustíme

```
node import.js
```

Pokud chceme vymazar všechna data z tabulek, vypneme si kontrolu integritních omezení.

```
SET FOREIGN_KEY_CHECKS = 0;
TRUNCATE stops;
TRUNCATE `lines`;
TRUNCATE points;
TRUNCATE paths;
TRUNCATE path_points;
TRUNCATE line_stops;
SET FOREIGN_KEY_CHECKS = 1;
```

Do souboru {publicloop_root}/app/frondend/map.html si přidáme události pro zadání polohy FROM a TO z mapy.

```
<script>
    var point_type = null;
    document.getElementById("fromInput").addEventListener('click', function (event) {
        point_type = 'from';
        bootstrap.Modal.getInstance(document.getElementById('planJourneyModal')).hide();
    })

    document.getElementById("toInput").addEventListener('click', function (event) {
        point_type = 'to';
        bootstrap.Modal.getInstance(document.getElementById('planJourneyModal')).hide();
    })

    var vectorLayerFrom = null;
    var vectorLayerTo = null;

    map.on('click', function (event) {
        var point = map.getCoordinateFromPixel(event.pixel);
        var lonLat = ol.proj.toLonLat(point);
        if (point_type == 'from' || point_type == 'to') {
            bootstrap.Modal.getInstance(document.getElementById('planJourneyModal')).show();
        }
        if (point_type == 'from') {
            document.getElementById("fromInput").value = lonLat;

            if (vectorLayerFrom != null)
                map.removeLayer(vectorLayerFrom);

            var featureCircle = new ol.Feature({
                geometry: new ol.geom.Point(point),
                name: 'From',
            });

            var vectorCircle = new ol.source.Vector({});
            vectorCircle.addFeature(featureCircle);

            vectorLayerFrom = new ol.layer.Vector({
                source: vectorCircle,
                style: new ol.style.Style({
                    image: new ol.style.Circle({
                        radius: 6,
                        fill: new ol.style.Fill({
                            color: '#FF00CC'
                        })
                    })
                })
            });
            map.addLayer(vectorLayerFrom);
        }
        if (point_type == 'to') {
            document.getElementById("toInput").value = lonLat;

            if (vectorLayerTo != null) {
                map.removeLayer(vectorLayerTo);
            }

            var featureCircle = new ol.Feature({
                geometry: new ol.geom.Point(point),
                name: 'To',
            });

            var vectorCircle = new ol.source.Vector({});
            vectorCircle.addFeature(featureCircle);

            vectorLayerTo = new ol.layer.Vector({
                source: vectorCircle,
                style: new ol.style.Style({
                    image: new ol.style.Circle({
                        radius: 6,
                        fill: new ol.style.Fill({
                            color: '#00FFFF'
                        })
                    })
                })
            });

            map.addLayer(vectorLayerTo);
        }
        point_type = null;
    })
</script>
```

Na straně backendu si upravíme routu /journey/find. Pro jednoduchost sestavíme trasu z úseků: * FROM - první zastávka linky 5 * první zastávka linky 5 - poslední zastávka linky 5 * poslední zastávka linky 5 - TO

```
var body = '';
req.on('data', chunk => {
    body += chunk.toString();
});
req.on('end', () => {
        res.statusCode = 200;
        res.setHeader('Content-Type', 'application/json');

        let reqData = JSON.parse(body);
        let active = reqData.active;
        let arrive = reqData.arrive;
        let depart = reqData.depart;
        let from = reqData.from;
        let to = reqData.to;

        from = from.split(','); //15 50
        to = to.split(',');

        var dist = getDistance([50.0210234, 15.7587522], [from [1], from[0]]);

        var section_from = {
            type: 1,
            type_name: "walk",
            from: {latitude: parseFloat(from[0]), longitude: parseFloat(from[1]), date: depart},
            to: {latitude: 15.7587522, longitude: 50.0210234, date: depart},
            distance: dist,
            line: []
        };

        dist = getDistance([50.0470595, 15.8114271], [to [1], to[0]]);
        var section_to = {
            type: 1,
            type_name: "walk",
            from: {latitude: 15.8114271, longitude: 50.0470595, date: depart},
            to: {latitude: parseFloat(to[0]), longitude: parseFloat(to[1]), date: depart},
            distance: dist,
            line: []
        };

        var connection = getConn();
        connection.query(
            'select l.line_name, ls.direction, ls.order, s.stop_name, s.latitude, s.longitude\n' +
            'from `lines` l\n' +
            'inner join line_stops ls on l.line_key = ls.line_key\n' +
            'inner join stops s on ls.stop_key = s.stop_key\n' +
            'where l.line_name = ? and ls.direction = ?\n' +
            'order by ls.order',
            ['5', '1'],
            function (err, results) {
                if (err) {
                    res.statusCode = 500;
                    res.setHeader('Content-Type', 'text/plain');
                    res.end('Internal Server Error');
                } else {

                    var resData = {meta:[], sections:[]};
                    resData.sections.push(section_from);

                    var line_path = [];
                    results.forEach(function (item) {
                        line_path.push(
                            {
                                line_name: item.line_name,
                                direction: item.direction,
                                order: item.order,
                                stop_name: item.stop_name,
                                latitude: item.latitude,
                                longitude: item.longitude
                            });
                    });

                    dist = getDistance([50.0210234, 15.7587522], [50.0470595, 15.8114271]);
                    resData.sections.push({
                        type: 2,
                        type_name: "bus",
                        from: {latitude: 15.7587522, longitude: 50.0210234, date: depart},
                        to: {latitude: 15.8114271, longitude: 50.0470595, date: depart},
                        distance: dist,
                        line: line_path
                    });

                    resData.sections.push(section_to);
                    res.statusCode = 200;
                    res.setHeader('Content-Type', 'application/json');
                    res.setHeader('Set-Cookie', session_cookie);
                    res.end(JSON.stringify(resData));
                }
            }
        )
    }
)
```

A přidáme si funkci getDistance().

```
function getDistance(coords1, coords2) {
    function toRad(x) {
        return x * Math.PI / 180;
    }
    var lon1 = coords1[0];
    var lat1 = coords1[1];

    var lon2 = coords2[0];
    var lat2 = coords2[1];

    var R = 6371; // km

    var x1 = lat2 - lat1;
    var dLat = toRad(x1);
    var x2 = lon2 - lon1;
    var dLon = toRad(x2)
    var a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
        Math.cos(toRad(lat1)) * Math.cos(toRad(lat2)) *
        Math.sin(dLon / 2) * Math.sin(dLon / 2);
    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    var d = R * c;

    return d;
}
```

V mapě vykreslíme vyhledanou trasu. Po obdržení dat zavoláme funkci drawSections(text).

```
fetch('http://localhost/backend/journey/find', {
    method: "POST",
    headers: {'Content-Type': 'application/json'},
    body: JSON.stringify(plan_data)
})
.then((response) => {
    if (!response.ok) {
        throw new Error(`HTTP error, status = ${response.status}`);
    }
    return response.text();
})
.then((text) => {
    document.getElementById("loader").style.display = 'none';
    drawSections(text);
})
.catch(error => {
    //TODO handle error
})
```

A funkci drawSections si vytvoříme.

```
<script>
    function drawSections(data) {
        const resdata = JSON.parse(data);
        const sections = resdata.sections;
        for (let section of resdata.sections) {
            switch (section.type) {
            case 1:
                var coords = [];
                coords.push(ol.proj.fromLonLat([section.from.latitude, section.from.longitude]));
                coords.push(ol.proj.fromLonLat([section.to.latitude, section.to.longitude]));
                drawLines(coords, '#c800ff');
                break
            case 2:
                var coords = [];
                for (let point of section.line)
                    coords.push(ol.proj.fromLonLat([point.latitude, point.longitude]));
                drawLines(coords, '#0000FF');
                break;
            }
        }
    }
    var journeyLayers = [];

    function drawLines(coords, color) {

        var featureLine = new ol.Feature({
            geometry: new ol.geom.LineString(coords),
            name: 'Line'
        });

        var vectorLine = new ol.source.Vector({});
        vectorLine.addFeature(featureLine);

        journeyLayer = new ol.layer.Vector({
            source: vectorLine,
            style: new ol.style.Style({
                fill: new ol.style.Fill({color: color, weight: 4}),
                stroke: new ol.style.Stroke({color: color, width: 4})
            })
        });
        journeyLayers.push(journeyLayer);
        map.addLayer(journeyLayer);

    }
</script>
```

Ještě si přidáme funkci pro vymazání mapy.

```
<script>
    function clearMap() {
        if (vectorLayerFrom != null) {
            map.removeLayer(vectorLayerFrom);
            document.getElementById("fromInput").value = '';
        }
        if (vectorLayerTo != null) {
            map.removeLayer(vectorLayerTo);
            document.getElementById("toInput").value = '';
        }
        if (journeyLayers != []) {
            for (let journeyLayer of journeyLayers)
            map.removeLayer(journeyLayer);
        }
    }
</script>
```

A tu si zavoláme vždy, když v menu vybereme tlačítko "Plan journey".

```
<script>
    document.getElementById("planJourney").addEventListener('click', function (event) {

        ...

        clearMap();
    })
</script>
```

Pro získání a import dat jednotlivých časů odjezdů a příjezdů autobusů si vytvoříme webscraper.

Webscraper

Pro stažení dat do našeho projektu si vytvoříme malý webscraper, kterým budeme postupně procházet stránky dopravního podniku a data si ukládat do databáze.

Datová vrstva pro import dat

Vytvoříme si vytvoříme potřebné tabulky pro import dat v datovém modeleru MySQL Workbench. SQL dotazy si následně spustíme na MariaDB.

```
CREATE TABLE IF NOT EXISTS `publicloop`.`imp_batch` (
    `impbatch_key` INT NOT NULL AUTO_INCREMENT,
    `impbatch_start` DATETIME NOT NULL,
    `impbatch_finish` DATETIME,
    `impbatch_log` LONGTEXT NULL,
    PRIMARY KEY (`impbatch_key`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `publicloop`.`imp_lines` (
    `impline_key` INT NOT NULL AUTO_INCREMENT,
    `impline_name` VARCHAR(255) NOT NULL,
    `impline_url` VARCHAR(255) NOT NULL,
    `impbatch_key` INT NOT NULL,
    PRIMARY KEY (`impline_key`),
    INDEX `fk_imp_lines_imp_batch1_idx` (`impbatch_key` ASC) VISIBLE,
    CONSTRAINT `fk_imp_lines_imp_batch1`
    FOREIGN KEY (`impbatch_key`)
        REFERENCES `publicloop`.`imp_batch` (`impbatch_key`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `publicloop`.`imp_stops` (
    `impstop_key` INT NOT NULL  AUTO_INCREMENT,
    `impstop_name` VARCHAR(255) NOT NULL,
    `impstop_url` VARCHAR(255) NOT NULL,
    `impstop_direction` INT NULL,
    `impstop_order` VARCHAR(255) NOT NULL,
    `impline_key` INT NOT NULL,
    PRIMARY KEY (`impstop_key`),
    INDEX `fk_imp_stops_imp_lines1_idx` (`impline_key` ASC) VISIBLE,
    CONSTRAINT `fk_imp_stops_imp_lines1`
    FOREIGN KEY (`impline_key`)
        REFERENCES `publicloop`.`imp_lines` (`impline_key`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `publicloop`.`imp_departures` (
    `impdeparture_key` INT NOT NULL AUTO_INCREMENT,
    `impstop_key` INT NOT NULL,
    `departure` TIME NOT NULL,
    `valid_from` DATE NULL,
    `valid_to` DATE NULL,
    `impdeparture_type` VARCHAR(255) NULL,
    PRIMARY KEY (`impdeparture_key`),
    INDEX `fk_imp_departures_imp_stops1_idx` (`impstop_key` ASC) VISIBLE,
    CONSTRAINT `fk_imp_departures_imp_stops1`
    FOREIGN KEY (`impstop_key`)
        REFERENCES `publicloop`.`imp_stops` (`impstop_key`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `publicloop`.`imp_departure_params` (
    `impdepartureparam_key` INT NOT NULL AUTO_INCREMENT,
    `impdeparture_key` INT NULL,
    `param_code` VARCHAR(255) NOT NULL,
    `param_name` VARCHAR(255) NOT NULL,
    PRIMARY KEY (`impdepartureparam_key`),
    INDEX `fk_imp_departure_params_imp_departures1_idx` (`impdeparture_key` ASC) VISIBLE,
    CONSTRAINT `fk_imp_departure_params_imp_departures1`
    FOREIGN KEY (`impdeparture_key`)
        REFERENCES `publicloop`.`imp_departures` (`impdeparture_key`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `publicloop`.`imp_departure_exclude`
(
    `impdepartureexclude_key` INT  NOT NULL AUTO_INCREMENT,
    `impdeparture_key`        INT  NOT NULL,
    `exclude_from`            DATE NOT NULL,
    `exclude_to`              DATE NOT NULL,
    PRIMARY KEY (`impdepartureexclude_key`),
    INDEX `fk_imp_departure_exclude_imp_departures1_idx` (`impdeparture_key` ASC) VISIBLE,
    CONSTRAINT `fk_imp_departure_exclude_imp_departures1`
    FOREIGN KEY (`impdeparture_key`)
```

```
        REFERENCES `publicloop`.`imp_departures` (`impdeparture_key`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Zdrojový kód webscraperu

Do našeho projektu si doinstalujeme balíček jsdom.

```
npm install jsdom
```

Následně si vytvoříme nový soubor {publicloop_root}/app/backend/scraper.js s níže uvedeným zdrojovým kódem.

```
zdrojový kód viz soubor scraper.js
```