# Serverová aplikace v NodeJS

## Zadání projektu

Vytvořte serverovou aplikaci, která umožní pomocí formuláře ukládat do .CSV souboru školní úkoly zadané v různých předmětech. Aplikace umožní vypisovat přehledný seznam úkolů, kde bude uveden datum zadání úkolu, předmět, název úkolu a termín odevzdání úkolu.

## Založení projektu

1. Vytvoříme složku pro nový projekt - např. **node-todolist**.
2. Po otevření složky ve VS Code použijeme terminál a příkazem **npm install** inicializujeme nový projekt:

**C:\skola-2021-2022\it2-pvy\node-todolist>npm init**

This utility will walk you through creating a package.json file.

It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields

and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and

save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (todolist)

version: (1.0.0)

description: Seznam skolnich ukolu

entry point: (index.js)

test command:

git repository: https://github.com/lucny/node-todolist.git

keywords: NodeJS, PUG, CSV

author: Marek Lucny

license: (ISC)

About to write to C:\skola-2021-2022\it2-pvy\node-todolist\package.json:

{

"name": "todolist",

"version": "1.0.0",

"description": "Seznam skolnich ukolu",

"main": "index.js",

"directories": {

"doc": "doc"

},

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"repository": {

"type": "git",

"url": "git+https://github.com/lucny/node-todolist.git"

},

"keywords": [

"NodeJS",

"PUG",

"CSV"

],

"author": "Marek Lucny",

"license": "ISC",

"bugs": {

"url": "https://github.com/lucny/node-todolist/issues"

},

"homepage": "https://github.com/lucny/node-todolist#readme"

}

Is this OK? (yes)

Po inicializaci projektu vznikne soubor **package.json** se záznamem inicializačních nastavení. V projektu nám bude sloužit

1. Vytvoříme základní soubor serverové aplikace - **index.js**.
2. Nainstalujeme si nástroj **nodemon** (Node monitor), pomocí něhož budeme moci spouštět a monitorovat serverovou aplikaci. Jednou z výhod je automatické restartování webového serveru v případě změny kódu. Parametr -g umožní nainstalovat modul v rámci systému, aby byl "globálně" k dispozici i pro další aplikace.

**npm install nodemon -g**

## Instalace frameworku Express a aktivace serveru

1. Nainstalujeme populární framework Express, který nám v NodeJS usnadňuje tvorbu a správu webové serverové aplikace.

**npm install express**

1. Nyní můžeme v souboru **index.js** připojit modul express do konstanty **express**, vytvořit konstantu **app** pro uložení objektu celé aplikace a vyvolat metodu **app.listen(),** která zajistí spuštění serveru naslouchajícího na zvoleném portu:

/\* Připojení modulu frameworku Express \*/

const express = require("express");

/\* Vytvoření základního objektu serverové aplikace \*/

const app = express();

/\* Nastavení portu, na němž bude spuštěný server naslouchat \*/

const port = 3000;

/\* Spuštění webového serveru \*/

app.listen(port, () => {

    console.log(`Server naslouchá na portu ${port}`);

});

1. Funkčnost serveru ověříme spuštěním:

C:\skola-2021-2022\it2-pvy\node-todolist>**nodemon index.js**

[nodemon] 2.0.7

[nodemon] to restart at any time, enter `rs`

[nodemon] watching path(s): \*.\*

[nodemon] watching extensions: js,mjs,json

[nodemon] starting `node index.js`

Server naslouchá na portu 3000

## Vytvoření git repozitáře a uložení první verze projektu na Github.com

V tuto chvíli je dobrá příležitost k založení git repozitáře pro náš nový projekt, abychom mohli s využitím verzovacího systému Git postupně ukládat jednotlivé verze projektu.

1. Vytvoříme nový lokální git repozitář a nakonfigurujeme uživatelské jméno i e-mail:

C:\skola-2021-2022\it2-pvy\node-todolist>**git init**

Initialized empty Git repository in C:/skola-2021-2022/it2-pvy/node-todolist/.git/

C:\skola-2021-2022\it2-pvy\node-todolist>**git config --local user.name "lucny"**

C:\skola-2021-2022\it2-pvy\node-todolist>**git config --local user.email** [**lucny@sspu-opava.cz**](mailto:lucny@sspu-opava.cz)

1. Vytvoříme soubor **README.md** a soubor **.gitignore**, do něhož zapíšeme název složky node\_modules, aby se do repozitáře zbytečně neukládaly instalované externí moduly.

*README.md*

# Seznam úkolů

## Serverová aplikace v Node JS

*.gitignore*

node\_modules

1. Provedeme úvodní commit.

C:\skola-2021-2022\it2-pvy\node-todolist>**git add .**

warning: LF will be replaced by CRLF in package-lock.json.

The file will have its original line endings in your working directory

warning: LF will be replaced by CRLF in package.json.

The file will have its original line endings in your working directory

C:\skola-2021-2022\it2-pvy\node-todolist>**git status**

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: .gitignore

new file: README.md

new file: index.js

new file: package-lock.json

new file: package.json

C:\skola-2021-2022\it2-pvy\node-todolist>**git commit -m "01 - zalozeni projektu, instalace Express"**

[main (root-commit) e228af6] 01 - zalozeni projektu, instalace Express

5 files changed, 922 insertions(+)

create mode 100644 .gitignore

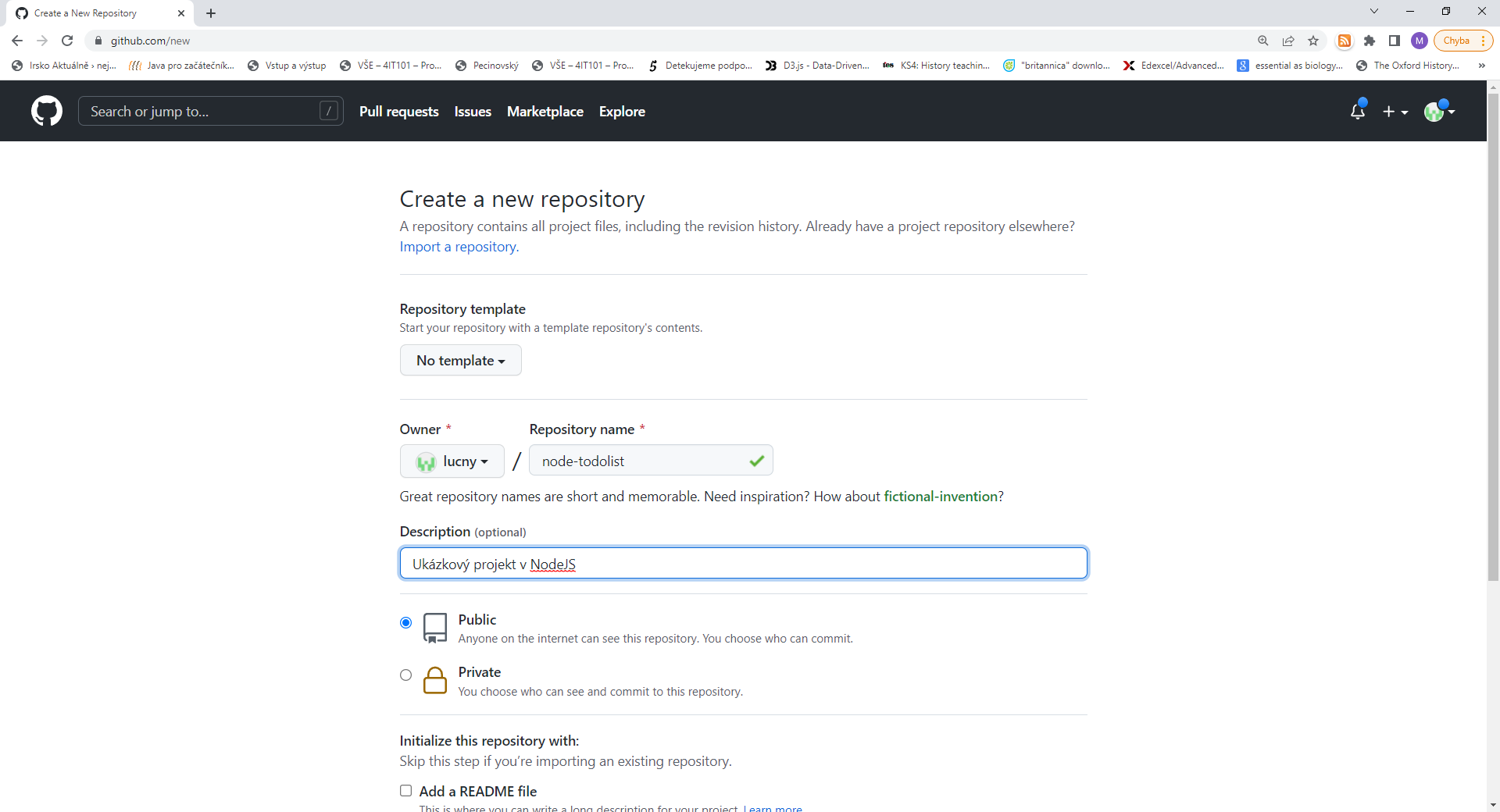
create mode 100644 README.md

create mode 100644 index.js

create mode 100644 package-lock.json

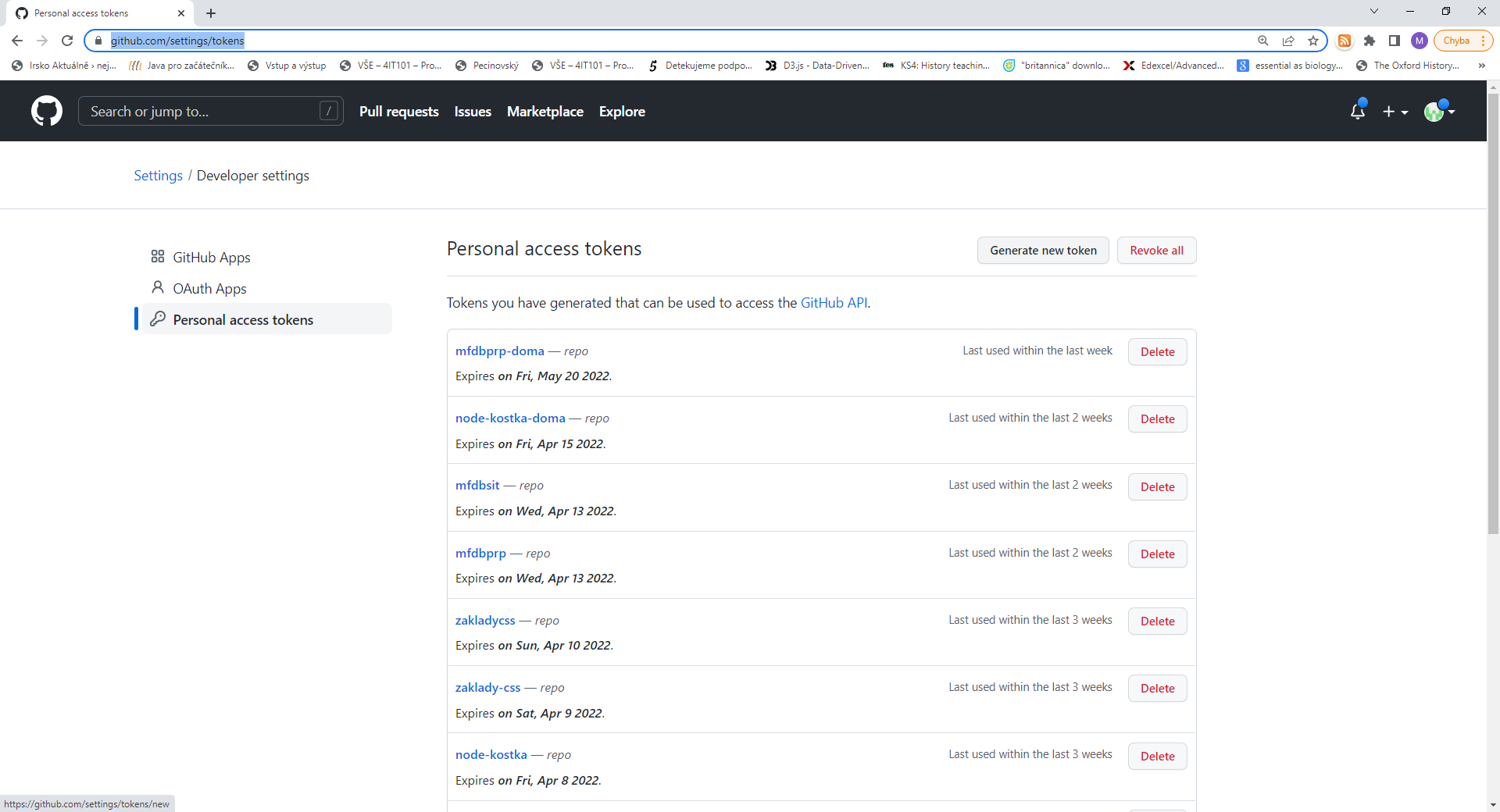
create mode 100644 package.json

1. Vytvoříme vzdálený repozitář na Gitghub.com

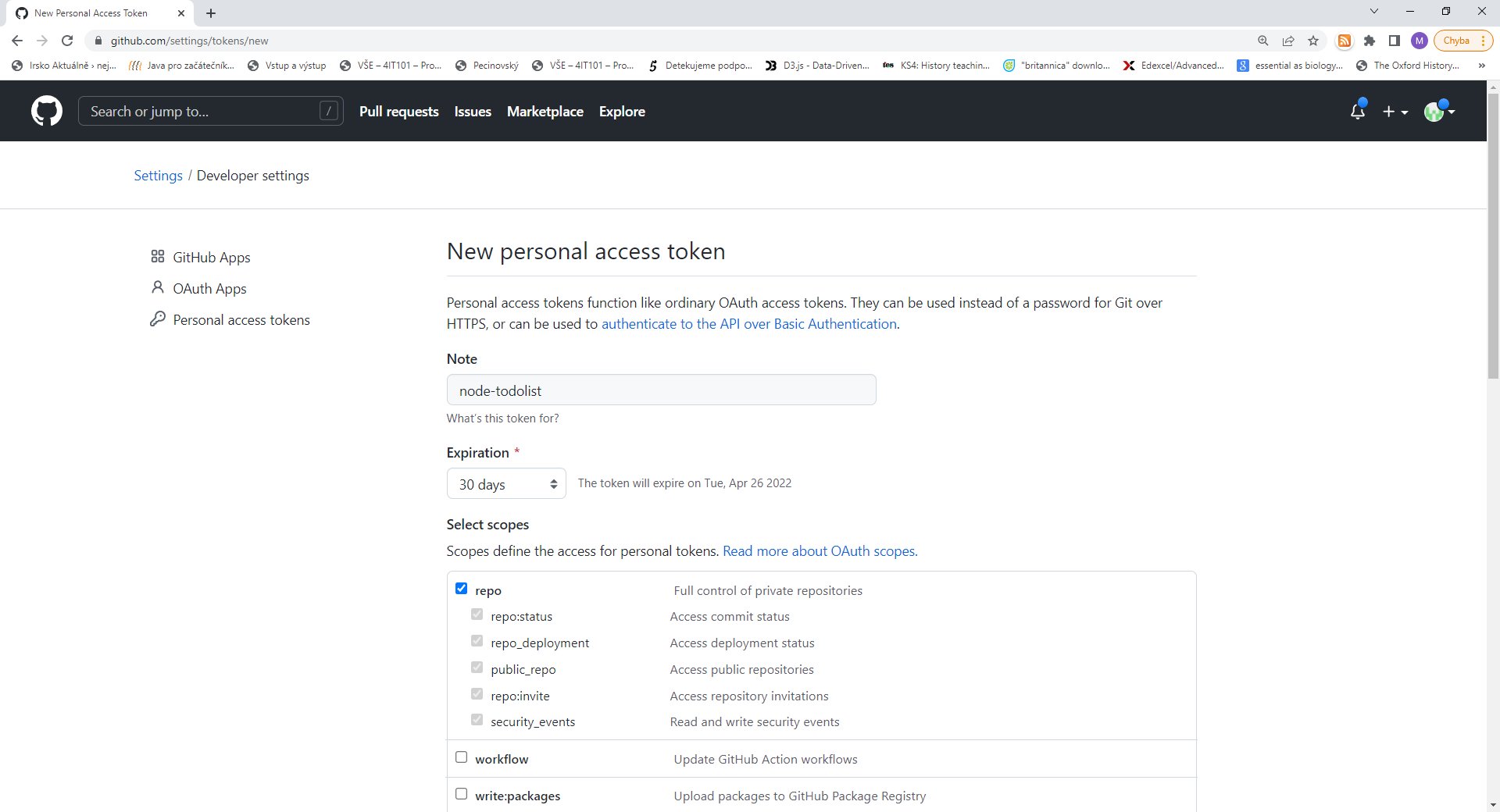


1. Vygenerujeme si přístupový token pro tento projekt na adrese <https://github.com/settings/tokens>:

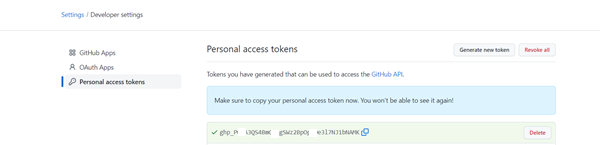
Klikneme na tlačítko Generate new token:



Pojmenujeme token a zaškrtneme alespoň volbu repo:



Vygenerovaný token si zkopírujeme, abychom ho mohli použít pro přístup do našeho repozitáře:



1. Propojíme náš lokální repozitář se vzdáleným repozitářem a příkazem **git push** provedeme synchronizaci.   
   Vygenerovaný osobní přístupový token musíme vložit do url adresy repozitáře hned za označení protokolu a oddělit od zbytku adresy pomocí znaku **@**:

C:\skola-2021-2022\it2-pvy\node-todolist>**git remote add origin https://ghp\_PO\_\_\_QS4Bm\_\_\_gSWz2BpO\_\_\_e3l7NJ1bNAMK@github.com/lucny/node-todolist.git**

C:\skola-2021-2022\it2-pvy\node-todolist>**git push -u origin main**

Enumerating objects: 7, done.

Counting objects: 100% (7/7), done.

Delta compression using up to 16 threads

Compressing objects: 100% (5/5), done.

Writing objects: 100% (7/7), 6.11 KiB | 3.05 MiB/s, done.

Total 7 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/lucny/node-todolist.git

\* [new branch] main -> main

Branch 'main' set up to track remote branch 'main' from 'origin'.

Další verze svého projektu můžeme přidávat jednoduše opakováním příkazů **git add**, **git commit** a **git push**.

## Vytvoření statických souborů klientské části webu

Framework Express umožňuje identifikovat a používat složku (nebo složky) se statickými soubory, které tvoří základ klientské části webové aplikace (grafické přílohy, soubory CSS, JS atd.). Součástí složky mohou být rovněž statické HTML soubory, které k svému zobrazení nevyžadují data ze serveru. Využijeme toho k vytvoření úvodní stránky naší aplikace, která bude obsahovat formulář pro odesílání dat na server.

1. Do souboru **index.js** přidáme řádek kódu, který serverové aplikaci oznamuje, že složka se statickými soubory se nazývá **public**.

/\* Nastavení portu, na němž bude spuštěný server naslouchat \*/

const port = 3000;

/\* Identifikace složky obsahující statické soubory klientské části webu \*/

**app.use(express.static**("public"));

/\* Spuštění webového serveru \*/

app.listen(port, () => {

    console.log(`Server naslouchá na portu ${port}`);

});

1. V základní složce projektu vytvoříme podsložku **public** a v ní soubor **index.html**, který bude představovat úvodní stránku našeho webu.

<!doctype html>

<html lang="cs">

  <head>

    <title>Školní úkoly</title>

    <!-- Required meta tags -->

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

  </head>

  <body>

    <header class="bg-primary text-center p-5">

        <h1 class="display-3 text-white">Seznam úkolů</h1>

        <p class="h3 text-warning">Ukázková aplikace v NodeJS</p>

    </header>

    <main class="container">

        <h2 class="bg-info text-light text-center p-3 mt-3">Formulář pro zadání nového úkolu</h2>

        <div class="container">

            <form>

                <div class="form-group row">

                    <label for="ukol" class="col-sm-2 col-form-label">Zadání úkolu</label>

                    <div class="col-sm-10">

                        <input type="text" class="form-control" name="ukol" id="ukol" placeholder="Napište stručné zadání úkolu">

                    </div>

                </div>

                <div class="form-group row">

                    <label for="predmet" class="col-sm-2 col-form-label">Předmět</label>

                    <div class="col-sm-10">

                      <select class="form-control" name="predmet" id="predmet">

                        <option value="Anglický jazyk">Anglický jazyk</option>

                        <option value="Český jazyk">Český jazyk</option>

                        <option value="Elektrotechnika">Elektrotechnika</option>

                        <option value="Fyzika">Fyzika</option>

                        <option value="Hardware">Hardware</option>

                        <option value="Matematika">Matematika</option>

                        <option value="Počítačová grafika">Počítačová grafika</option>

                        <option value="Počítačové sítě">Počítačové sítě</option>

                        <option value="Programování">Programování</option>

                        <option value="Programové vybavení">Programové vybavení</option>

                      </select>

                    </div>

                </div>

                <div class="form-group row">

                    <label for="odevzdani" class="col-sm-2 col-form-label">  
Datum odevzdání</label>

                    <div class="col-sm-5">

                        <input type="date" class="form-control" name="odevzdani" id="odevzdani">

                    </div>

                </div>

                <div class="form-group row">

                    <div class="offset-sm-2 col-sm-10">

                        <button type="submit" class="btn btn-primary">Odeslat údaje</button>

                    </div>

                </div>

            </form>

        </div>

    </main>

    <footer class="bg-light text-center p-3">

        <p><img src="./img/skola-logo.png" alt="Školní logo"></p>

        <p>&copy; 2022 - Marek Lučný, IT2, SŠPU Opava</p>

    </footer>

    <!-- Optional JavaScript -->

    <!-- jQuery first, then Popper.js, then Bootstrap JS -->

    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>

  </body>

</html>

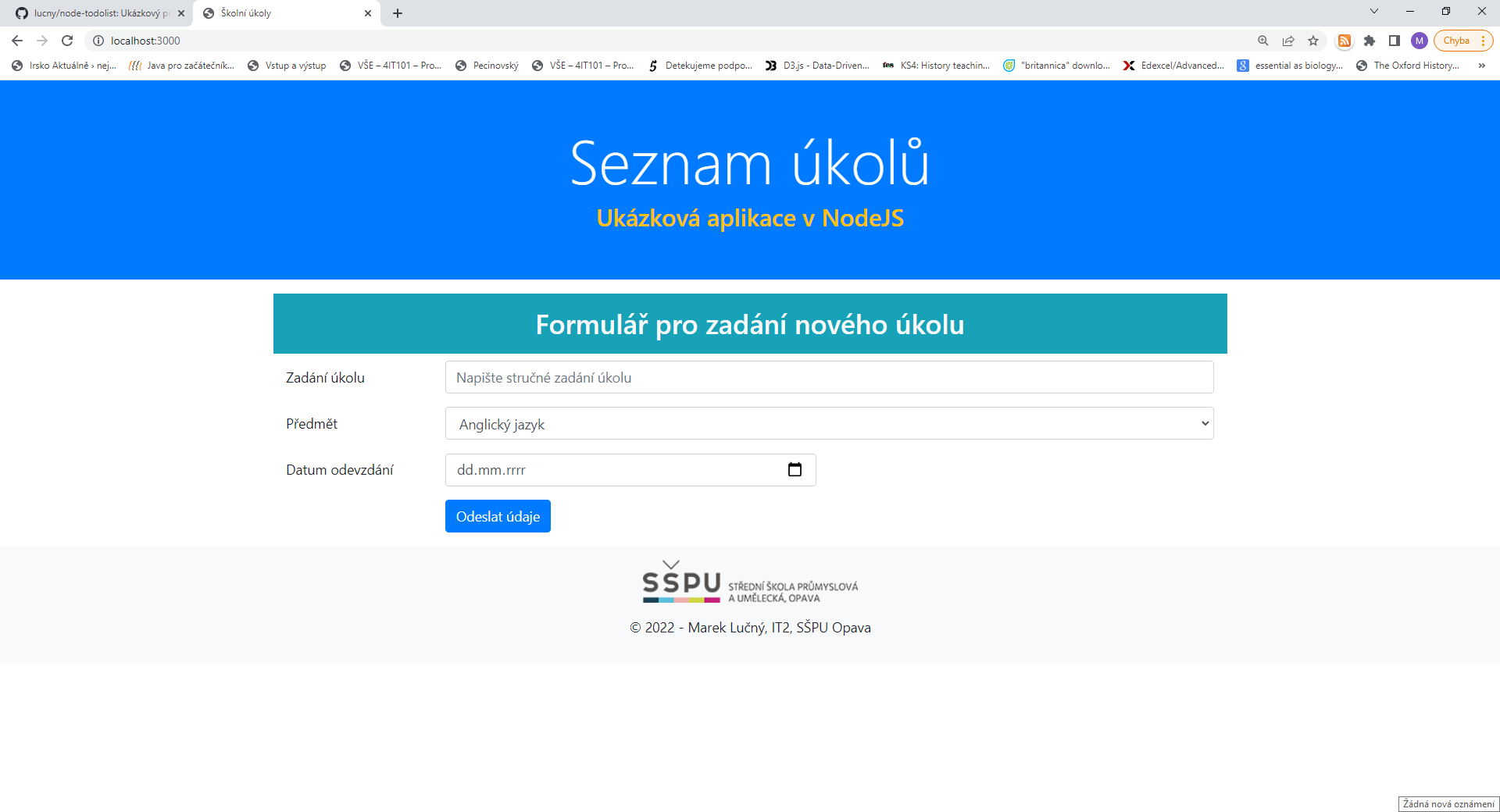
Klíčovou částí webové stránky je formulář, který obsahuje tři vstupní prvky:

* prvek pojmenovaný **ukol** (name="ukol") typu *text*, do něhož by měl být zadán stručný popis úkolu,
* výběrový seznam (*select*) pojmenovaný **predmet** (name="predmet"), který obsahuje předvolby s názvy školních předmětů,
* prvek pojmenovaný **odevzdani** (name="odevzdani") typu *date*, který umožňuje vybrat požadované datum odevzdání úkolu.

K odeslání formuláře slouží tlačítko **Odeslat údaje**, které je typu *submit* (tj. odeslat).

Stránka je upravena s využitím frameworku Bootstrap. Do zápatí stránky je vložen obrázek se školním logem - příslušný soubor **skola-logo.png** se nachází v podsložce **img**, která musí být také součástí složky **public**.

1. Po spuštění serveru - nodemon index.js - si můžeme vytvořenou úvodní stránku prohlédnout v prohlížeči.



## Odesílání dat z formuláře a jejich uložení na serveru

V této fázi tvorby naší webové aplikace se musíme postarat o správné odeslání dat z formuláře na server a jejich zpracování v podobě uložení do souboru typu **CSV** (*Comma Separated Values* - data oddělená čárkou).

1. **Úprava atributů formuláře**. Do úvodní značky formuláře přidáme atribut **action** (určuje adresu na serveru, kam jsou posílána data z formuláře) a **method** (určuje typ použité metody pro odeslání dat). Výchozí metodou je sice metoda **GET**, ale ta data z formuláře přidává přímo do URL adresy jako parametry tzv. dotazu (**query**). V případě odesílání dat na server je proto mnohem lepší použít metodu **POST**, která data odesílá v těle požadavku (**body**). Kromě toho, že data nejsou přímo viditelná v URL adrese, mohou být i mnohem obsáhlejší.

<form action="/savedata" method="post">

1. **Instalace a připojení potřebných modulů na straně serveru**.

Nejprve naistalujeme dvě užitečné externí knihovny:

npm install body-parser

npm install moment

Nyní můžeme obě uvedené knihovny a další dvě vestavěné knihovny (fs a path) připojit na začátku souboru index.js:

/\* Připojení modulu frameworku Express (https://expressjs.com/) \*/

const express = require("express");

/\* Připojení externího modulu body-parser (https://www.npmjs.com/package/body-parser) - middleware pro parsování těla požadavku \*/

const bodyParser = require("body-parser");

/\* Připojení externího modulu moment (https://momentjs.com/) - knihovna pro formátování datových a časových údajů \*/

const moment = require("moment");

/\* Připojení vestavěných  modulů fs (práce se soubory) a path (cesty v adresářové struktuře) \*/

const fs = require("fs");

const path = require("path");

1. **Vytvoření datového souboru CSV**.

Data z formuláře budeme ukládat do souboru CSV, který si připravíme v nově založené složce **data**. Souboru dáme název **ukoly.csv** a vypíšeme do něj první řádek představující záhlaví (na konci zadáme Enter):

"Zadání úkolu","Předmět","Datum zadání","Datum odevzdání"

1. **Vytvoření metody pro zpracování požadavku na straně serveru a uložení dat do souboru CSV**.

Do souboru **index.js** přidáme metodu **app.post(),** pomocí níž můžeme zpracovat data poslaná z formuláře. Funkce zpracovává požadavky odeslané metodou POST na adresu **savedata**.

Abychom mohli snadno získat údaje vložené do těla požadavku (req.body), použijeme jako zprostředkující software (tzv. *middleware*) připojenou knihovnu body-parser. Pro účely frameworku Express si parser nakonfigurujeme do konstanty **urlencodedParser**, kterou pak předáme jako druhý parametr metodě app.post().

Jednotlivé části kódu jsou detailněji popsány v komentářích:

/\* Využití modulu body-parser pro parsování těla požadavku \*/

const urlencodedParser = bodyParser.urlencoded({extended: false});

/\* Ošetření požadavku poslaného metodou POST na adresu <server>/savedata

   Ukládá data poslaná z webového formuláře do souboru CSV \*/

app.post('/savedata', urlencodedParser, (req, res) => {

    /\* Do proměnné date bude pomocí knihovny MomentJS uloženo aktuální datum v podobě YYYY-MM-DD (rok-měsíc-den) \*/

    let date = moment().format('YYYY-MM-DD');

    /\* Vytvoření řetězce z dat odeslaných z formuláře v těle požadavku (req.body) a obsahu proměnné date.

       Data jsou obalena uvozovkami a oddělená čárkou. Escape sekvence \n provede ukončení řádku. \*/

    let str = `"${req.body.ukol}","${req.body.predmet}","${date}","${req.body.odevzdani}"\n`;

    /\* Pomocí modulu fs a metody appendFile dojde k přidání připraveného řádku (proměnná str) do uvedeného souboru \*/

    fs.appendFile(path.join(\_\_dirname, 'data/ukoly.csv'), str, function (err) {

        /\* Když byla zaznamenána chyba při práci se souborem \*/

        if (err) {

            /\* Vypsání chyby do konzole NodeJS (na serveru). \*/

            console.error(err);

            /\* Odpovědí serveru bude stavová zpráva 400 a v hlavičce odpovědi budou odeslány upřesňující informace. \*/

            return res.status(400).json({

                success: false,

                message: "Nastala chyba během ukládání souboru"

            });

        }

    });

    /\* Přesměrování na úvodní stránku serverové aplikace včetně odeslání stavové zprávy 301. \*/

    res.redirect(301, '/');

});

1. **Spuštění aplikace a ověření funkčnosti**.  
   Teď již můžeme opakovat zadání dat a jejich odeslání z formuláře na server. Jestliže vše funguje správně, můžeme postupně sledovat přibývající řádky v souboru **ukoly.csv**.

"Zadání úkolu","Předmět","Datum zadání","Datum odevzdání"

"Goniometrické funkce","Matematika","2022-03-27","2022-04-03"

"Slovíčka k tématu My family","Anglický jazyk","2022-03-27","2022-04-07"

"Tvorba aplikace v NodeJS","Programové vybavení","2022-03-27","2022-04-20"

## Výpis dat pomocí šablonovacího systému PUG

Pro výpis dat, která jsou uložena na webovém serveru (v datových souborech, nebo v databázích) využívají webové frameworky tzv. šablonovací systémy.

**Šablonovací systém** (anglicky **template engine**) je část webové aplikace, která se stará o tzv. prezentační vrstvu aplikace. Ta uživateli vykresluje danou stránku webu s využitím předem připravených šablon. Do nich při vykreslování plní data, která předtím získala jiná část aplikace a šablonovacímu systému je předala.

Pro vykreslování výstupních stránek v naší aplikaci využijeme populární šablonovací systém PUG, který je nástupcem staršího JADE.

1. **Instalace a propojení šablonovacího systému PUG s frameworkem Express.**

Nejprve nainstalujeme modul PUG (v podstatě interpret jazyka pug) - **npm install pug**

Poté provedeme nastavení aplikace Express tak, aby dokázala pracovat se šablonami **.pug** umístěnými do nově vytvořené složky **views**. Do souboru **index.js** zapíšeme oba příkazy s nastavením pod příkaz nastavující složku se statickými soubory webu:

/\* Identifikace složky obsahující statické soubory klientské části webu \*/

app.use(express.static("public"));

/\* Nastavení typu šablonovacího engine na pug\*/

app.set("view engine", "pug");

/\* Nastavení složky, kde budou umístěny šablony pug \*/

app.set("views", path.join(\_\_dirname, "views"));

1. **Vytvoření šablony index.pug ve složce views**.

V základní složce projektu vytvoříme podsložku views, která může obsahovat soubory šablonovacího systému .pug. Jazyk PUG umožňuje vytvářet přehlednou strukturu stránky zjednodušeným zápisem značek i atributů v kombinaci s makry, do kterých je možné zapisovat předané proměnné (např. #{nadpis}). Lze rovněž využít i některé řídicí struktury včetně podmínek nebo cyklů (zde např. each ukol in ukoly).   
Podrobnou referenci o používání jazyka PUG lze najít zde - <https://pugjs.org/api/getting-started.html>

Pro správnou interpretaci kódu tohoto jazyka je nezbytné důsledně a přesně dodržovat hierarchickou strukturu kódu s využitím odsazování pomocí tabulátoru.

doctype html

html

    head

        title Seznam úkolů

        link(rel='stylesheet' href='https://getbootstrap.com/docs/4.4/dist/css/bootstrap.min.css')

    body

        header.jumbotron-fluid.bg-success.p-3

            h1.text-center.text-white #{nadpis}

        main.container

            table(class="table table-striped")

                thead

                    tr.bg-dark.text-light

                        th  Zadání úkolu

                        th  Předmět

                        th  Datum zadání

                        th  Datum odevzdání

                tbody

                    each ukol in ukoly

                        tr

                            td  #{ukol.ukol}

                            td  #{ukol.predmet}

                            td  #{ukol.zadani}

                            td  #{ukol.odevzdani}

        footer.bg-dark.p-2

            .container.text-white

                p.float-right

                    a(href="/") Zpět na formulář

                p &copy; 2022 - IT2 - NodeJS

        script(src='https://code.jquery.com/jquery-3.4.1.slim.min.js')

        script(src='https://getbootstrap.com/docs/4.4/dist/js/bootstrap.bundle.min.js')

1. **Instalace knihovny csvtojson**.

Abychom si usnadnili načítání dat ze souboru CSV, využijeme externí knihovnu csvtojson, která umožňuje načíst CSV data do formátu JSON (JavaScript Object Notation), tedy v podobě pole javaskriptových objektů.

Nejprve tedy nainstalujeme potřebný modul (**npm install csvtojson**) a poté ho připojíme do souboru index.js:

/\* Připojení externího modulu moment (https://momentjs.com/) - knihovna pro formátování datových a časových údajů \*/

const moment = require("moment");

/\* Připojení externího modulu csvtojson (https://www.npmjs.com/package/csvtojson) - knihovna usnadňující načtení dat z CSV do formátu JSON \*/

const csvtojson = require('csvtojson');

/\* Připojení vestavěných  modulů fs (práce se soubory) a path (cesty v adresářové struktuře) \*/

const fs = require("fs");

1. Vytvoření metody pro přípravu a vykreslení stránky

Samotné vykreslení stránky je provedeno funkcí, která reaguje na odeslání požadavku metodou **get** na URL adresu **<server>/todolist**. Je samozřejmě součástí souboru **index.js**.

/\* Reakce na požadavek odeslaný metodou get na adresu <server>/todolist \*/

app.get("/todolist", (req, res) => {

    /\* Použití knihovny csvtojson k načtení dat ze souboru ukoly.csv.

       Atribut headers zjednodušuje pojmenování jednotlivých datových sloupců. \*/

    /\* Pro zpracování je použito tzv. promises, které pracují s částí .then (úspěšný průběh operace) a .catch (zachycení možných chyb) \*/

    csvtojson({headers:['ukol','predmet','zadani','odevzdani']}).fromFile(path.join(\_\_dirname, 'data/ukoly.csv'))

    .**then**(data => {

        /\* Vypsání získaných dat ve formátu JSON do konzole \*/

        console.log(data);

        /\* Vykreslení šablony index.pug i s předanými daty (objekt v druhém parametru) \*/

        res.render('index', {nadpis: "Seznam úkolů", ukoly: data});

    })

.**catch**(err => {

        /\* Vypsání případné chyby do konzole \*/

        console.log(err);

        /\* Vykreslení šablony error.pug s předanými údaji o chybě \*/

        res.render('error', {nadpis: "Chyba v aplikaci", chyba: err});

    });

});

Zelenou barvou jsou v kódu vyznačeny části, které tvoří strukturu tzv. promises. Jedná se o speciální mechanismus používaný v moderních verzích JS k ošetření určitých operací (často datových) pomocí zřetězených callback funkcí. V části **then** jsou řešeny reakce na úspěšně naplněný "příslib" (data byla správně načtena), v části **catch** jsou zachyceny a ošetřeny chybové stavy.

1. **Vytvoření šablony error.pug a vypsání případné chyby.**

V našem případě by případná chyba zpracování CSV souboru měla být oznámena uživateli speciální chybovou stránkou - očekává se vyrendrování šablony error.pug. Ta zatím neexistuje, musíme ji proto vytvořit ve složce views.

doctype html

html

    head

        title Chybová stránka

        link(rel='stylesheet' href='https://getbootstrap.com/docs/4.4/dist/css/bootstrap.min.css')

    body

        header.jumbotron-fluid.bg-danger.p-3

            h1.text-center.text-white #{nadpis}

        main.container

            .alert.alert-danger #{chyba}

        footer.bg-dark.p-2

            .container.text-white

                p.float-right

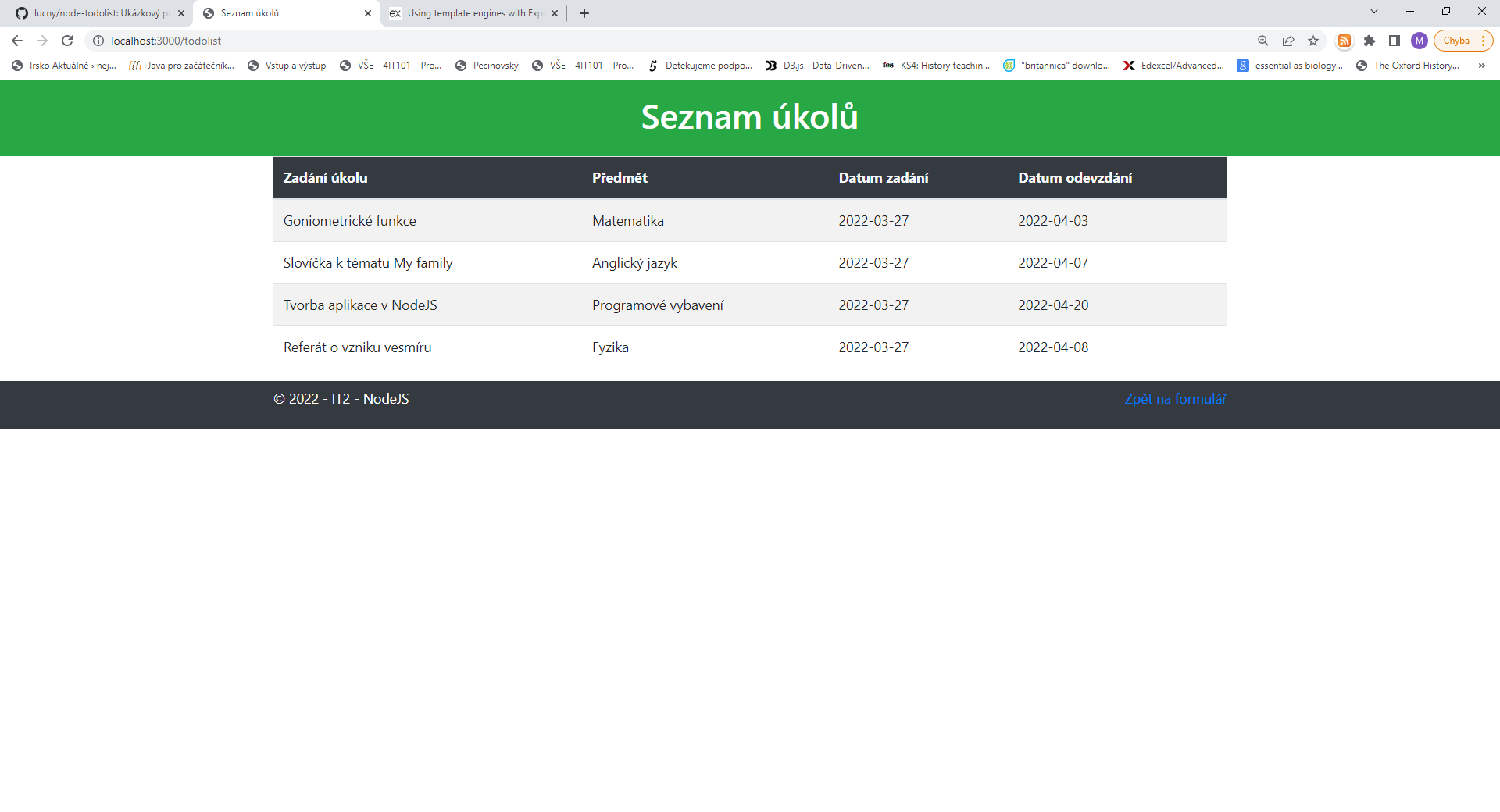
                    a(href="/") Zpět na formulář

                p &copy; 2022 - IT2 - NodeJS

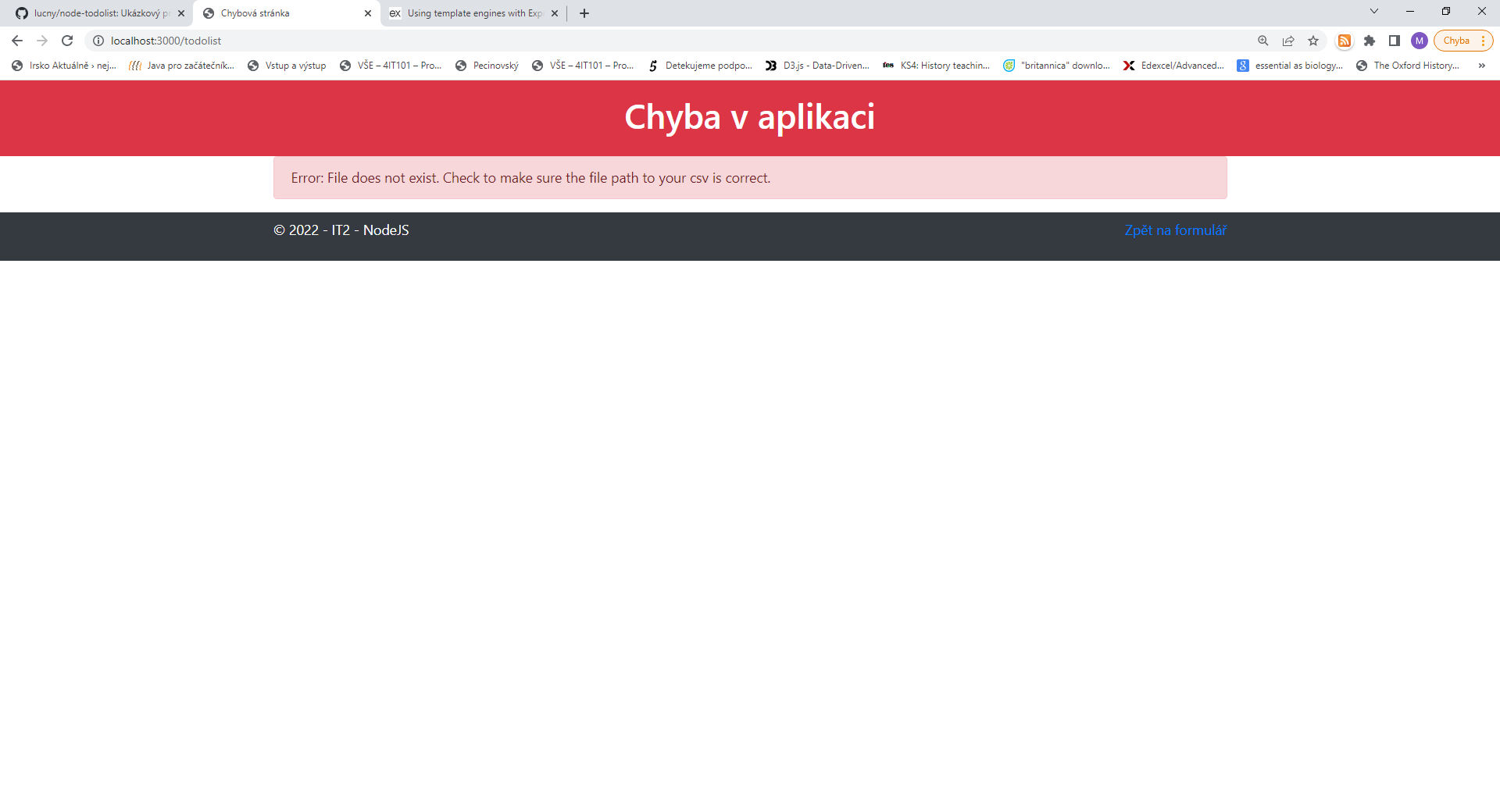
        script(src='https://code.jquery.com/jquery-3.4.1.slim.min.js')

        script(src='https://getbootstrap.com/docs/4.4/dist/js/bootstrap.bundle.min.js')

Po zadání adresy **<server>/todolist** by se měla objevit stránka se seznamem úkolů:



V případě nějaké chyby (např. nesprávný název souboru CSV) zase chybová stránka:



## Náměty pro vylepšení aplikace

* Validace dat na straně klienta i serveru.
* Vypsání data zadání i odevzdání v podobě typické pro české prostředí (den. měsíc. rok).
* Seřazení úkolů podle data odevzdání.
* Zvýraznění úkolů, které jsou již po termínu odevzdání.
* Rozdělení výstupních stránek na šablony pro jednotlivé obsahové bloky (např. header, main, footer).