Programare web

Laboratorul 11

11. Controlul utilizatorilor

Cuprins

11.1. Concepte și tehnologii

În acest laborator veți continua proiectul 2 și veți lucra cu serverul web Node.js, cookies și sesiuni. De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

11.2. Structura proiectului

Structura proiectului `proiect1-numeUtilizatorGitHub` de până la acest laborator (inclusiv):

- [node_modules] - conține pachetele necesare proiectului și este creat cu utilitarul npm
- [public] - conține toate resursele accesibile direct de către client (e.g., fișiere css, javascript, imagini)
- [views]
    - autentificare.ejs - **lab 11**
    - chestionar.ejs - lab 10
    - index.ejs - **lab 11**
    - layout.ejs - lab 10
    - rezultat-chestionar.ejs - lab 10 (tema 2)
- .gitignore - lab 10 (conține doar textul node_modules)
- app.js - lab 10, actualizat **lab 11**
- intrebari.json - lab 10 (tema 3)
- package-lock.json - lab 10 (ar putea să lipsească)
- package.json - lab 10
- README.md - lab 10

Rutele definite în cadrul proiectului:

Metoda HTTP	Resursa	Nr.Lab.	Descriere
GET	/	10	Serverul răspunde cu textul 'Hello World'.
		11	Serverul răspunde cu o pagină de Bine ai venit! prin inserarea `index.ejs` în `layout.ejs` și returnarea rezultatului la client.
GET	/chestionar	10	Serverul răspunde cu formularul HTML (chestionarul) prin inserarea `chestionar.ejs` în `layout.ejs` și returnarea rezultatului la client.
POST	/rezultat-chestionar	10.2	Se cere această resursă la apăsarea butonului de submit a formularului din `/chestionar`. La server se calculează numărul de răspunsuri corecte și se returnează răspunsul.
GET	/autentificare	11	Serverul răspunde cu o pagină de Autentificare prin inserarea `autentificare.ejs` în `layout.ejs` și returnarea rezultatului la client.
POST	/verificare-autentificare	11	Se cere această resursă la apăsarea butonului de `submit` a formularului din `/autentificare`. La server se verifică dacă utilizatorul și parola sunt corecte, și se răspunde cu un redirect spre resursa `/`, dacă sunt corecte, sau spre resursa `/autentificare`, dacă nu sunt corecte.

11.3. Descriere

11.3.1. Prezentarea laboratorului

În acest laborator veți extinde aplicația din laboratorul precedent (proiect-2) și veți adăuga funcționalitatea prin care un utilizator se poate autentifica pe site (Fig. 1.)

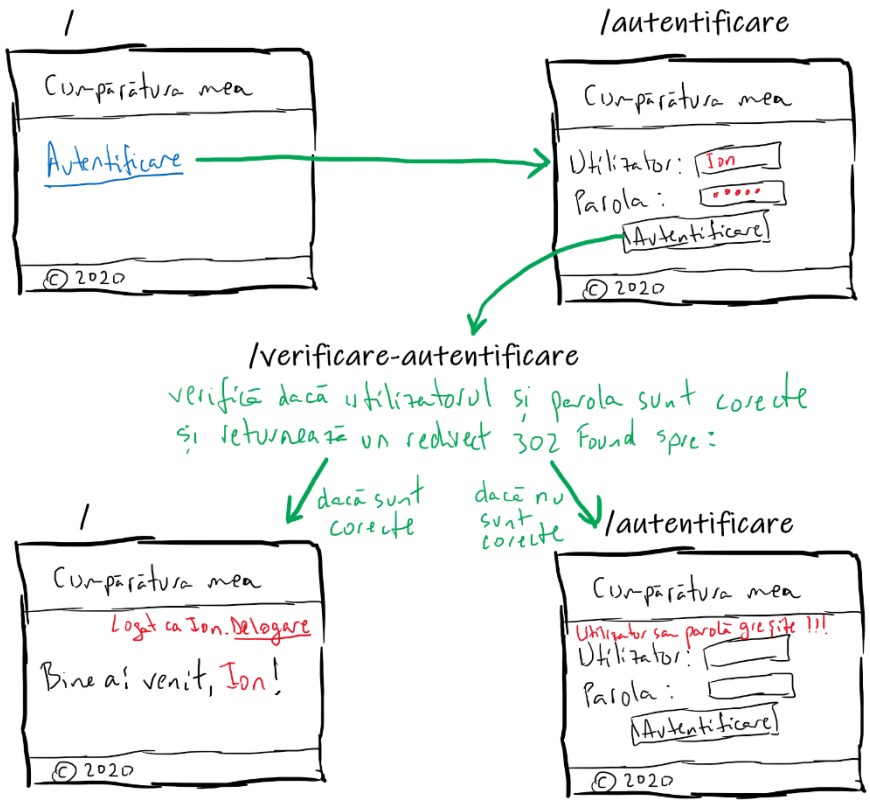

Fig. 11-1. Logica de autentificare pe site

11.3.2. Informații suplimentare

Pentru detalii privind Node.js să citiți cursul 11 și să consultați documentația următoare:

a. Introduction to Node.js - https://nodejs.dev/en/learn/introduction-to-nodejs/
b. Node.js server without a framework - https://developer.mozilla.org/en-US/docs/Learn/Server-side/Node_server_without_framework
c. Express basic routing - https://expressjs.com/en/starter/basic-routing.html
d. Serving static files in Express - https://expressjs.com/en/starter/static-files.html
e. ejs - Embedded JavaScript templates - https://www.npmjs.com/package/ejs
f. Getting started with EJS - https://medium.com/@atingenkay/getting-started-with-ejs-28fe4a693e53
g. express-ejs-layouts - https://www.npmjs.com/package/express-ejs-layouts
h. JavaScript Cookies - https://www.w3schools.com/js/js_cookies.asp
i. A sweet tale of cookies and sessions with Node.js - https://medium.com/@ratan_kumar/a-sweet-tale-of-cookies-and-sessions-with-node-js-5fe255b702ac

11.4. Teme

11.4.1. Tema 1

Continuați proiectul 2 de pe GitHub și implementați următoarele (nu uitați să testați pe măsură ce implementați și să încărcați modificările pe GitHub):

a. Creați în directorul `views` un fișier cu numele `index.ejs` care va conține un link spre pagina de autentificare.
b. În fișierul `app.js` adăugați o rută astfel încât, la accesarea `http://localhost:6789/`, să fie returnat view-ul `index.ejs`.
c. Creați în directorul `views` un fișier cu numele `autentificare.ejs` care va conține un formular cu câmpurile utilizator și parolă, și un buton cu textul Autentificare. În urma apăsării butonului se va cere, prin metoda HTTP POST, resursa: `/verificare-autentificare`.
d. În fișierul `app.js` adăugați o rută astfel încât, la accesarea `http://localhost:6789/autentificare`, să fie returnat view-ul `autentificare.ejs`.
e. În fișierul `app.js` adăugați o rută pentru resursa `/verificare-autentificare` și afișați la consolă conținutul mesajului (`req.body`[1]).

11.4.2. Tema 2

Extindeți proiectul prin implementarea următoarelor funcționalități:

a. Executați comanda `npm install cookie-parser --save` pentru a instala pachetul care facilitează lucrul cu cookie-uri.
   În fișierul app.js adăugați la început linia de cod cookie-parser `const cookieParser=require('cookie-parser');` și, după inițializarea variabilei `app`, adăugați lina de cod `app.use(cookieParser());`.
b. Modificați codul din funcția care satisface cererea pentru resursa `/verificare-autentificare` astfel încât să verificați dacă utilizatorul și parola sunt corecte (puteți să aveți un simplu `if`).
c. Dacă sunt corecte, folosiți funcția `res.cookie`[2] pentru a seta un cookie cu numele utilizator și funcția `res.redirect` [3]pentru a trimite un răspuns de redirect spre `http://localhost:6789/`.

[1] Express 4.x – API Reference – req.body - https://expressjs.com/en/api.html#req.body
[2] Express 4.x – API Reference – res.cookie - https://expressjs.com/en/api.html#res.cookie
[3] Express 4.x – API Reference – res.redirect - https://expressjs.com/en/api.html#res.redirect

d. Dacă nu sunt corecte, folosiți funcția `res.cookie` pentru a seta un cookie cu numele `mesajEroare` și funcția `res.redirect` pentru a trimite un răspuns de redirect spre `http://localhost:6789/autentificare`.
e. Pentru a accesa valorile cookie folosiți `req.cookies`.
f. Modificați fișierele `app.js` și `index.ejs` astfel încât cookie-ul primit de server la accesarea resursei `/` să ajungă în view-ul `index`, unde să fie afișat textul **Bine ai venit, *NumeUtilizator*!**.
g. Modificați fișierele `app.js` și `autentificare.ejs` astfel încât cookie-ul primit de server la accesarea resursei /autentificare să ajungă în view-ul autentificare, unde să fie afișat cu roșu mesajul de eroare din cookie.

11.4.3. Tema 3

Extindeți proiectul prin implementarea următoarelor funcționalități:

a. Executați comanda `npm install express-session --save` pentru a instala pachetul care facilitează lucrul cu sesiuni.
b. Utilizați sesiuni și modificați codul astfel încât pe fiecare pagină a site-ului să apară textul **Autentificat ca *NumeUtilizator*** și un buton care să permită de-logarea.
c. Copiați fișierul `utilizatori.json` din proiectul 1 și adăugați fiecărui utilizator mai multe proprietăți (cel puțin nume și prenume).
   Salvați în variabila de sesiune proprietățile corespunzătoare utilizatorului autentificat (mai puțin parola!).

Programare web

Laboratorul 12

12. Baze de date

Cuprins

12.1. Concepte și tehnologii

În acest laborator veți continua proiectul 2 și veți lucra cu serverul web Node.js și cu o bază de date la alegere.. De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

12.2. Structura proiectului

Structura proiectului `proiect2-numeUtilizatorGitHub` de până la acest laborator (inclusiv):

- [`node_modules`] - conține pachetele necesare proiectului și este creat cu utilitarul npm
- [`public`] - conține toate resursele accesibile direct de către client (e.g., fișiere css, javascript, imagini)
- [`views`]
  - `autentificare.ejs` - lab 11
  - `chestionar.ejs` - lab 10
  - `index.ejs` - lab 11, **actualizat lab 12**
  - `layout.ejs` - lab 10
  - `rezultat-chestionar.ejs` - lab 10 (tema 2) – ca alternativă se poate folosi `chestionar.ejs` cu marcarea răspunsurilor corecte/bifate
  - `vizualizare-cos.ejs` - **lab 12 (tema 3)**
- `.gitignore` - lab 10 (conține doar textul `node_modules`)
- `app.js` - lab 10, actualizat lab 11, **lab 12**
- `intrebari.json` - lab 10 (tema 3)
- `package-lock.json` - lab 10 (ar putea să lipsească)
- `package.json` - lab 10
- `README.md` - lab 10

Rutele definite în cadrul proiectului:

Met. HTTP	Resursa	View	Nr.Lab.	Descriere
GET	/	index.ejs	10	Serverul răspunde cu textul 'Hello World'.
			11	Serverul răspunde cu o pagină de Bine ai venit! prin inserarea `index.ejs` în `layout.ejs` și returnarea rezultatului la client.
			12	Serverul răspunde cu o pagină de Bine ai venit! și cu lista de produse din baza de date.
GET	/chestionar	chestionar.ejs	10	Serverul răspunde cu formularul HTML (chestionarul) prin inserarea `chestionar.ejs` în `layout.ejs` și returnarea rezultatului la client.
POST	/rezultat-chestionar	rezultat-chestionar.ejs sau chestionar.ejs	10.2	Se cere această resursă la apăsarea butonului de submit a formularului din `/chestionar`. La server se calculează numărul de răspunsuri corecte și se returnează răspunsul.
GET	/autentificare	autentificare.ejs	11	Serverul răspunde cu o pagină de Autentificare prin inserarea `autentificare.ejs` în `layout.ejs` și returnarea rezultatului la client.
POST	/verificare-autentificare	-	11	Se cere această resursă la apăsarea butonului de `submit` a formularului din `/autentificare`. La server se verifică dacă utilizatorul și parola sunt corecte, și se răspunde cu un redirect spre resursa `/`, dacă sunt corecte, sau spre resursa `/autentificare`, dacă nu sunt corecte.
GET	/creare-bd	-	12	Serverul se conectează la serverul de baze de date și, într-o bază de date cu numele `cumparaturi`, creează o tabelă `produse`, după care răspunde clientului cu un redirect spre resursa `/`.
GET	/inserare-bd	-	12	Serverul se conectează la serverul de baze de date și inserează mai multe produse în tabela `produse`, după care răspunde clientului cu un redirect spre resursa `/`.
POST sau GET	/adaugare-cos	-	12.2	Serverul adaugă id-ul produsului specificat în corpul mesajului HTTP într-un vector din variabila de sesiune (sau într-un vector global dacă nu ați implementat tema 3 din laboratorul 11).
GET	/vizualizare-cos	vizualizare-cos.ejs	12.3	Serverul răspunde cu o pagină de **Vizualizare coș** prin inserarea `vizualizare-cos.ejs` în `layout.ejs` și returnarea rezultatului la client.

12.3. Descriere

12.3.1. Prezentarea laboratorului

În acest laborator veți extinde aplicația din laboratorul precedent (proiect-2) și veți adăuga funcționalitatea prin care produsele dintr-o bază de date sunt afișate pe site.

De asemenea, dacă utilizatorul este logat, acesta trebuie să aibă posibilitatea să adauge produse într-un coș de cumpărături, și să vizualizeze produsele din coș.

Puteți să alegeți ce server de baze de date doriți.

12.3.2. Informații suplimentare

Pentru detalii privind conectarea la o bază de date din Node.js puteți să citiți cursul 12, și să consultați documentația următoare:

a. Node.js MySQL - https://www.w3schools.com/nodejs/nodejs_mysql.asp
b. Node.js MongoDB - https://www.w3schools.com/nodejs/nodejs_mongodb.asp
c. Node.js OracleDB - https://oracle.github.io/node-oracledb/
d. Node.js PostgeSQL - https://node-postgres.com/
e. Node.js Microsoft SQL Server - https://tediousjs.github.io/node-mssql/
f. Node.js Redis - https://redis.js.org/
g. Node.js SQLite - https://www.sqlitetutorial.net/sqlite-nodejs/
h. Getting Started with NodeJS SQLite - https://www.linode.com/docs/guides/getting-started-with-nodejs-sqlite/

12.4. Teme

12.4.1. Tema 1

Continuați proiectul 2 de pe GitHub și implementați următoarele (nu uitați să testați pe măsură ce implementați și să încărcați modificările pe GitHub):

a. În fișierul `views/index.ejs` adăugați un buton **Creare BD** și un buton **Încarcare BD**.
b. În fișierul `app.js` adăugați o rută astfel încât, la accesarea `http://localhost:6789/creare-bd`, serverul se conectează la serverul de baze de date și la o bază de date cu numele `cumparaturi`, în care creează o tabelă `produse`, după care răspunde clientului cu un redirect spre resursa `/`.
c. În fișierul `app.js` adăugați o rută astfel încât, la accesarea `http://localhost:6789/inserare-bd`, serverul se conectează la serverul de baze de date și inserează mai multe produse în tabela produse, după care răspunde clientului cu un redirect spre resursa `/`.
d. Verificați în baza de date dacă s-au adăugat produsele.

12.4.2. Tema 2

Extindeți proiectul prin implementarea următoarelor funcționalități:

a. În fișierul `views/index.ejs`, afișați produsele din tabela produse.
b. În dreptul fiecărui produs din `views/index.ejs`, adăugați un buton cu textul **Adaugă în coș**. Dacă ați implementat logica de autentificare din laboratorul precedent, faceți astfel încât butonul de adăugare în coș să fie vizibil/activat doar dacă utilizatorul este autentificat.
c. La apăsarea butonului **Adaugă în coș** se face o cerere la server pentru resursa `/adaugare_cos` și se transmite id-ul produsului care trebuie adăugat în coș prin metoda GET sau prin POST, la alegere.
   *Prin metoda GET puteți face o cerere pentru resursa `/adaugare_cos?id=idProdus`.
   *Prin metoda POST puteți face câte un `form` pentru fiecare buton cu atributul `action` egal cu `/adaugare_cos` și cu elementul `<input type="hidden" name="id" value="idProdus">`.
d. Serverul adaugă id-ul produsului specificat în corpul mesajului HTTP într-un vector din variabila de sesiune (sau într-un vector global dacă nu ați implementat tema 3 din laboratorul 11).

12.4.3. Tema 3

Extindeți proiectul prin implementarea următoarelor funcționalități:

a. Creați în directorul `views` un fișier cu numele `vizualizare-cos.ejs` care va permite utilizatorului să vadă produsele pe care le-a adăugat în coș.
b. Faceți ca întreg site-ul să arate bine.

Programare web

Laboratorul 13

13. Securizarea aplicației

Cuprins

13.1. Concepte și tehnologii

În acest laborator veți continua proiectul 2 și veți securiza aplicația web creată. De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

13.2. Descriere

În acest laborator veți securiza aplicația web de la proiectul 2. Puteți consulta cursul 12 pentru detalii.

Temele din acest laborator conțin, în mod intenționat, puține detalii pentru a oferi studentului o flexibilitate în implementare și pentru a testa capacitatea sa de a se documenta singur pentru duce la bun sfârșit task-uri concrete.

13.3. Teme

13.3.1. Tema 1

Continuați proiectul 2 de pe GitHub și implementați următoarele (nu uitați să testați pe măsură ce implementați și să încărcați modificările pe GitHub):

a. Modificați aplicația astfel încât să suporte două tipuri de utilizatori: USER și ADMIN.
b. Creați o pagină web /admin la care să aibă acces doar utilizatorii cu rolul ADMIN.
c. Din resursa /admin trebuie să se permită adăugarea unui produs nou în baza de date.

13.3.2. Tema 2

Extindeți proiectul prin implementarea următoarelor funcționalități:

a. Detectați încercări de accesare ale unor resurse de la server inexistente și blocați temporar accesul la toate resursele pentru utilizatorul/IP-ul respectiv.

13.3.3. Tema 3

Extindeți proiectul prin implementarea următoarelor funcționalități:

a. Limitați numărul de autentificări nereușite venite de la aceeași adresă IP și/sau același nume utilizator.

b. În funcție de numărul de autentificări nereușite consecutive într-un interval scurt și într-un interval mai lung de timp, interziceți accesul la pagina de login și/sau la alte resurse de pe site.
c. Protejați-vă împotriva atacurilor de tipul injection care țintesc modul de interogare a serverului de baze de date.

Programare web

Laboratorul 10

10. Serverul web. Node.js

Cuprins

10.1. Concepte și tehnologii

În acest laborator veți începe proiectul 2 și veți lucra cu serverul web Node.js. De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

10.2. Structura proiectului

Temele din laboratoarele 10-13 vor forma un singur proiect: *proiect-2-numeUtilizatorGitHub*.

De data aceasta proiectul va avea o tematică impusă.

Astfel, veți realiza un site web care va permite utilizatorilor să cumpere produse dintr-o anumită categorie sau dintr-o sub-categorie a acesteia.

Pentru a afla categoria de produse (tema site-ului vostru) trebuie să accesați quiz-ul de la adresa https://edu.tuiasi.ro/mod/quiz/view.php?id=33658.

Structura proiectului `proiect-2-numeUtilizatorGitHub` de până la acest laborator (inclusiv):

- [`node_modules`] - conține pachetele necesare proiectului și este creat cu utilitarul npm (nu trebuie pus pe git)
- [`public`] - conține toate resursele accesibile direct de către client (e.g., fișiere css, javascript, imagini)
- [`views`]
  - `chestionar.ejs` - lab 10
  - `layout.ejs` - lab 10
  - `rezultat-chestionar.ejs` - lab 10 (tema 2)
- `.gitignore` - lab 10 (conține doar textul `node_modules`)

- `app.js` - lab 10
- `intrebari.json` - lab 10 (tema 3)
- `package-lock.json` - lab 10 (este generat de npm init sau npm install și ar putea să lipsească)
- `package.json` - lab 10
- `README.md` - lab 10

10.3. Descriere

10.3.1. Instalarea Node.js, pregătirea proiectului și rularea serverului

1. Descărcați și instalați ultima versiune LTS (Long-term support) a Node.js (20.12.2) de la adresa https://nodejs.org/en/download/. Nu este nevoie pe calculatoarele din laborator.
2. Dacă nu ați făcut-o deja, creați proiectul pe GitHub accesând adresa de pe tablă și urmați pașii din secțiunea Începutul laboratorului din laboratorul 0.
3. Deschideți proiectul în Visual Studio Code și creați un fișier cu numele `app.js` în care copiați următorul cod:

```
const http = require('http');
const hostname = '127.0.0.1';
const port = 6789;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Serverul rulează la adresa http://${hostname}:${port}/`);
});
```

4. În terminal executați comanda `node app.js` pentru a porni serverul.
5. Testați în browser accesând adresa http://localhost:6789/.

10.3.2. Managerul de pachete npm

1. În terminalul din Visual Studio Code, executați comanda `npm init`.
   Această comandă va crea și configura un fișier `package.json` care conține informații despre proiect și dependențele sale.
2. Pentru a accepta valoarea din paranteză pentru proprietățile respective puteți apăsa tasta ENTER. La proprietatea `test command` nu introduceți nimic.
3. Executați comanda `npm install` pentru a instala cele necesare proiectului prin procesarea fișierului `package.json`.
   Astfel, se va crea un fișier `package-lock.json` care nu ar trebui editat.
4. Executați comanda `npm install -g nodemon` pentru a instala global (-g) aplicația nodemon care permite repornirea automată a aplicației voastre de fiecare dată când se modifică un fișier. Pe calculatoarele din laborator ar trebui să fie deja instalat.
   Astfel, lansați în execuție aplicația prin comanda: `nodemon app.js`.

În Windows 10 va trebui să rulați o comandă în PowerShell: apăsați butonul Windows din taskbar ➜ scrieți PowerShell ➜ click dreapta pe Windows PowerShell ➜ Run as administrator ➜ în terminalul care se deschide executați: `Set-ExecutionPolicy -Scope LocalMachine Unrestricted`.

10.3.3. Web framework-ul Express

1. Executați comanda `npm install express --save` pentru a instala web framework-ul Express care facilitează implementarea unei aplicații web.
   Astfel, se va crea un director `node_modules` în rădăcina proiectului în care nu ar trebui să faceți modificări.
2. Executați comanda `npm install ejs --save` pentru a instala template engine-ul ejs care facilitează crearea resurselor html dinamice.
3. Executați comanda `npm install express-ejs-layouts --save` pentru a adăuga suport pentru layout-uri (template-ul folosit de site-ul vostru).
4. Executați comanda `npm install body-parser --save` pentru a interpreta corpul mesajului cererii.
5. Înlocuiți codul din fișierul `app.js` cu următorul cod:

```
const express = require('express');
const expressLayouts = require('express-ejs-layouts');
const bodyParser = require('body-parser')

const app = express();

const port = 6789;

// directorul 'views' va conține fișierele .ejs (html + js executat la server)
app.set('view engine', 'ejs');
// suport pentru layout-uri - implicit fișierul care reprezintă template-ul site-ului
este views/layout.ejs
app.use(expressLayouts);
// directorul 'public' va conține toate resursele accesibile direct de către client
(e.g., fișiere css, javascript, imagini)
app.use(express.static('public'))
// corpul mesajului poate fi interpretat ca json; datele de la formular se găsesc în
format json în req.body
app.use(bodyParser.json());
// utilizarea unui algoritm de deep parsing care suportă obiecte în obiecte
app.use(bodyParser.urlencoded({ extended: true }));

// la accesarea din browser adresei http://localhost:6789/ se va returna textul 'Hello
World'
// proprietățile obiectului Request - req - https://expressjs.com/en/api.html#req
// proprietățile obiectului Response - res - https://expressjs.com/en/api.html#res
app.get('/', (req, res) => res.send('Hello World'));

// la accesarea din browser adresei http://localhost:6789/chestionar se va apela funcția
specificată
app.get('/chestionar', (req, res) => {
    const listaIntrebari = [
        {
```

```
            intrebare: 'Întrebarea 1',
            variante: ['varianta 1', 'varianta 2', 'varianta 3', 'varianta 4'],
            corect: 0
        },
        //...
    ];
    // în fișierul views/chestionar.ejs este accesibilă variabila 'intrebari' care
conține vectorul de întrebări
    res.render('chestionar', {intrebari: listaIntrebari});
});

app.post('/rezultat-chestionar', (req, res) => {
    console.log(req.body);
    res.send("formular: " + JSON.stringify(req.body));
});

app.listen(port, () => console.log(`Serverul rulează la adresa http://localhost:`));
```

10.3.4. Informații suplimentare

Pentru detalii privind Node.js să citiți cursul 11 și să consultați documentația următoare:

a. Introduction to Node.js - https://nodejs.dev/en/learn/introduction-to-nodejs/
b. Node.js server without a framework - https://developer.mozilla.org/en-US/docs/Learn/Server-side/Node_server_without_framework
c. Express basic routing - https://expressjs.com/en/starter/basic-routing.html
d. Serving static files in Express - https://expressjs.com/en/starter/static-files.html
e. ejs - Embedded JavaScript templates - https://www.npmjs.com/package/ejs
f. Getting started with EJS - https://medium.com/@atingenkay/getting-started-with-ejs-28fe4a693e53
g. express-ejs-layouts - https://www.npmjs.com/package/express-ejs-layouts

10.4. Teme

10.4.1. Tema 1

Urmați pașii prezentați în acest laborator și citiți cu atenție comentariile din cod.

De abia la ultima temă va trebui să faceți ca site-ul să arate frumos. Accentul este acum pe funcționalitate.

Implementați următoarele funcționalități.

a. Creați în directorul `views` un fișier cu numele `layout.ejs` în care puneți scheletul site-ului (template-ul).
   În momentul în care se va apela `res.render`, se va returna conținutul fișierului `layout.ejs` și, unde apare `<%- body %>`, se va insera conținutul view-ului specificat ca argument, (e.g., `chestionar.ejs`).
b. Creați în directorul `views` un fișier cu numele `chestionar.ejs` în care să fie formularul cu quiz-ul respectiv.
   În acest fișier aveți acces la variabila `intrebari` specificată ca argument la apelul `res.render`. Pentru a crea formularul trebuie să parcurgeți vectorul `intrebari` și să adăugați întrebările și variantele de răspuns (va trebui să creați elemente de tipul `<input type="radio" .../>`).
c. Testați în browser accesând adresa http://localhost:6789/chestionar.

10.4.2. Tema 2

Extindeți proiectul prin implementarea următoarelor funcționalități:

a. Adăugați mai multe întrebări cu variantele lor de răspuns referitoare la tema impusă.
b. Adăugați logica de determinare a numărului de răspunsuri corecte. După completarea chestionarului se va face o cerere pentru resursa `/rezultat-chestionar` (vezi atributul `action` a formularului), unde se va afișa informația.
c. Faceți ca resursa `/rezultat-chestionar` să aibă același layout (template) ca resursa `/chestionar`.

10.4.3. Tema 3

Extindeți proiectul astfel încât întrebările afișate în chestionar să fie stocate într-un fișier `intrebari.json`.

Pentru a citi fișierul utilizați varianta **asincronă** prezentată în articolul Reading and Writing JSON Files with Node.js - https://stackabuse.com/reading-and-writing-json-files-with-node-js/.

De asemenea, faceți ca site-ul să arate frumos.