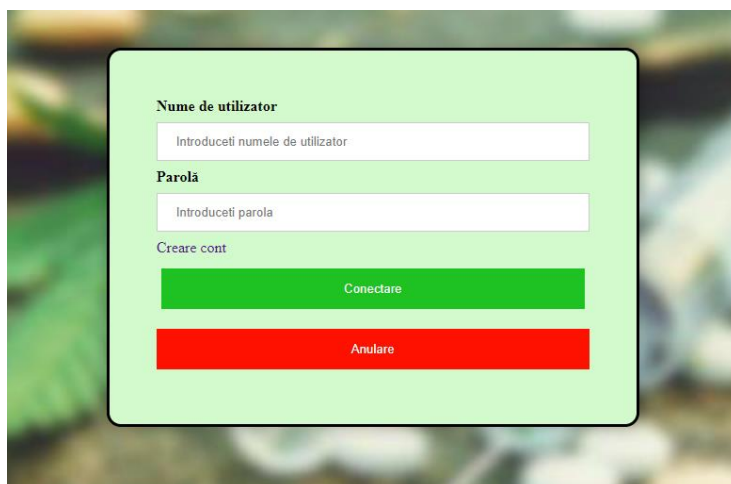


1. Introducere

RODX este o aplicatie web care permite vizualizarea de informatii cu privire la situatia drogurilor în țara noastră, prezentând statistici relevante legate de infraccionalitate, confiscări și urgente medicale. Prin analizarea datelor și informațiilor actualizate, această resursă se dorește a fi un instrument informativ și educativ, furnizând o înțelegere mai profundă a impactului drogurilor asupra societății. Prin intermediul graficelor, tabelelor și analizelor, utilizatorii pot descoperi tendințe, distribuții și implicații asociate consumului și traficului de droguri. Această pagină web își propune sa descurajeze abuzul de stupefiante, să sensibilizeze și să ofere o perspectivă obiectivă asupra problemei drogurilor în țara noastră, aducând în prim-plan nevoia de măsuri de prevenție, tratament și combatere a acestui fenomen. . Categoria corespunzatoare proiectelor si campaniilor arata eforturile si lupta comunitatii impotriva problemelor legate de droguri si creste nivelul de constientizare in societate.

2. Logare/Inregistrare

Pentru a putea naviga pe pagina web, utilizatorul trebuie sa fie logat. In cazul in care acesta nu are inca un cont, poate crea unul prin alegerea unui nume de utilizator si a unei parole. Prin apasarea butonului "Anulare", datele curente introduse sunt sterse din textbox-uri si pot fi introduse alte date. Datele introduse de utilizator în pagina de autentificare sunt trimise către backend și inserate în baza de date. Dacă datele sunt corecte și se găsesc în baza de date, un token JWT (JSON Web Token) este generat automat și are o perioadă de valabilitate de 12 ore. Acest token JWT este apoi salvat de frontend în LocalStorage. Ulterior, în cadrul fiecărei cereri trimise către backend, acest token JWT este inclus în antetul cererii în câmpul "Authorization". Backend-ul utilizează acest token pentru a valida identitatea utilizatorului după decodificarea acestuia. Această verificare asigură că doar utilizatorii autentificați și autorizați pot accesa resursele protejate. Din bara de navigare utilizatorul poate să se deconecteze dând clic pe butonul "Deconecteaza-te".



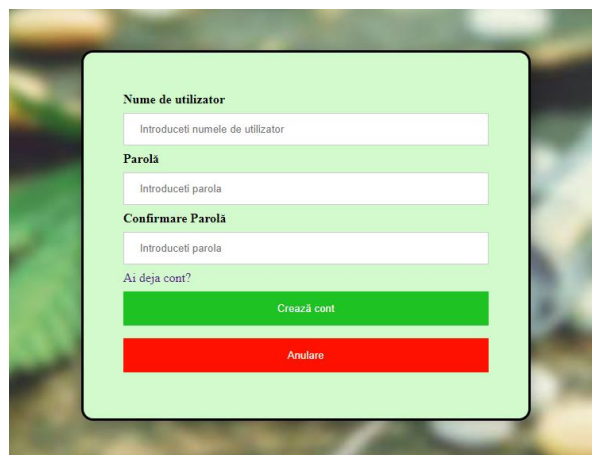
Nume de utilizator

Parolă

[Creare cont](#)

Conectare

Anulare

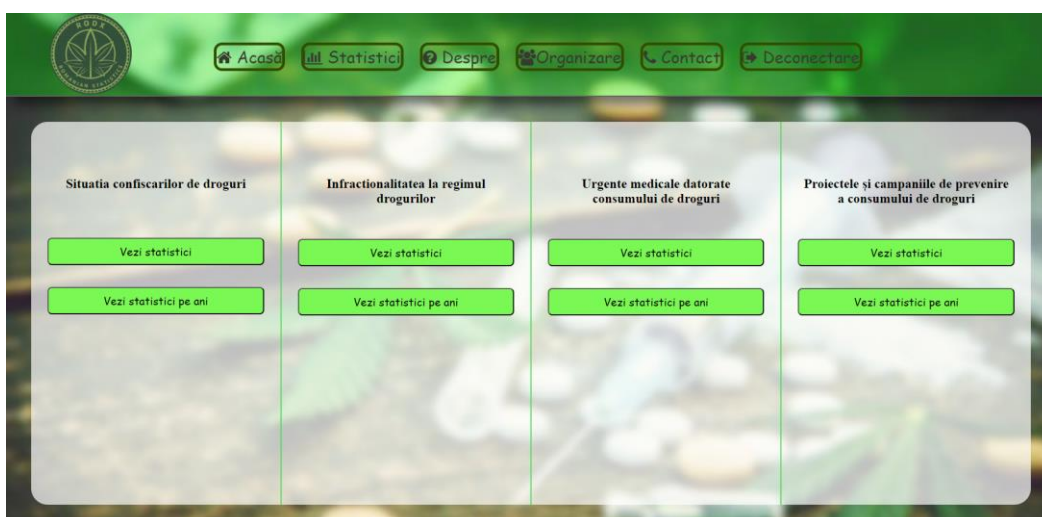


A registration form with a light green background and a black border. It contains the following fields and buttons:

- Nume de utilizator**: A text input field with the placeholder "Introduceți numele de utilizator".
- Parolă**: A text input field with the placeholder "Introduceți parola".
- Confirmare Parolă**: A text input field with the placeholder "Introduceți parola".
- Ai deja cont?**: A text label.
- Crează cont**: A green button.
- Anulare**: A red button.

3. Vizualizare statistici

Din pagina de statistici poate fi accesata categoria pentru care se doreste vizualizarea informatiilor, iar utilizatorul poate opta pentru tipul de statistici catre care sa inceapa navigarea. Apasand butonul “vezi statistici” corespunzator fiecărei categorii, se va deschide o pagina corespunzatoare pe care pot fi vizualizate charturi (linechart, barchart, piechart) ce acopera toate datele relevante, pentru a oferi userului o imagine de ansamblu asupra situatiei din ultimii ani. Optand pentru vizualizarea statisticilor pe ani, se deschide o pagina cu tabelul ce contine toate informatiile, acestea putand fi cu usurinta filtrate in functie de anumite criterii. Tabelele cu informatii filtrate pot fi descarcate in format CSV, apasand pe butonul “descarca csv”, in timp ce charturile si reprezentarile grafice pot fi descarcate in format PNG.



Infractionalitatea la regimul drogurilor: Statistici pe ani

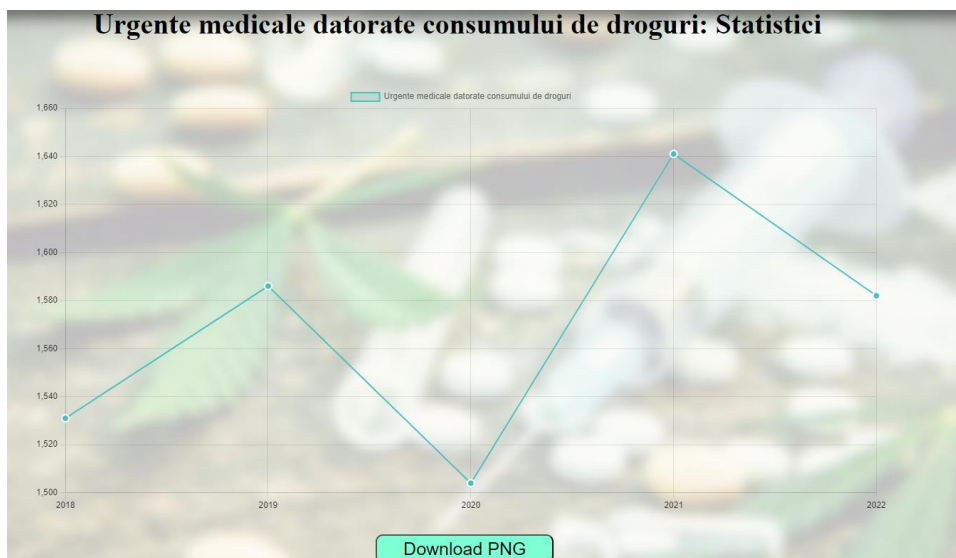
An Tip de persoane

An	Tip de persoane	Numar persoane
2022	Persoane cercetate	10102
2021	Persoane cercetate	10102
2020	Persoane cercetate	10102
2019	Persoane cercetate	10102
2018	Persoane cercetate	10102

[Download CSV](#)



Urgente medicale datorate consumului de droguri: Statistici



În fișierul corespunzător prelucrării informațiilor, sunt efectuate cereri de tip AJAX, utilizând funcția '\$.ajax()'. Scopul este de a pregăti și inițializa datele necesare pentru grafice și filter, astfel încât utilizatorul să poată interacționa cu acestea din interfață. După ce datele sunt obținute, acestea sunt procesate și ulterior afișate. În plus, valorile selectate anterior sunt recuperate din sessionStorage și setate în filtrele corespunzătoare, asigurând consistența selecțiilor utilizatorului.

```
682 function reloadFirstBarChartConfiscari() {
683   let query = '/api/statistics/confiscari_bar?';
684   query = `${query}&an=${filtersFirstBarChartConfiscari['anFirstBarChartConfiscari']}`;
685
686   $.ajax({
687     url: query,
688     type: "GET",
689     dataType: "json",
690     success: function(data) {
691       var labels = [];
692       var values = [];
693       data.elements.forEach(function(item) {
694         labels.push(item.nume_drog);
695         values.push(item.nr_capturi);
696       });
697       console.log(labels);
698       console.log(values);
699       redrawFirstBarChartConfiscari(labels, values);
700     }
701   });
702 }
```

4. Prelucrarea datelor

În urma descărcării manuale a fișierelor cu informații de pe site-ul guvernului, datele sunt introduse într-un folder specific, apoi sunt prelucrate prin parsare și inserate într-o bază de date SQLite, care conține tabele corespunzătoare tuturor categoriilor importante: activități, proiecte sau campanii de prevenire, situația confiscărilor, infracționalitatea rezultată în urma consumului de droguri, dar și urgențe medicale. Tot în baza de date, există un tabel specific utilizatorilor, care conține username-ul și parola.

```
1 import fs from 'fs';
2 import sqlite3 from 'sqlite3';
3 const dbFilePath = "../database.db";
4
5 function addQuotesToCSV(csvString) {
6   const rows = csvString.split('\n');
7   const quotedRows = rows.map(row => {
8     const values = row.split(',');
9     const quotedValues = values.map(value => `"${value.trim()}"`);
10    return quotedValues.join(',');
11  });
12
13  return quotedRows.join('\n');
14 }
15
16 export const runAll = (manual = false, callback = ()=>{}) => {
17   try { fs.unlinkSync(dbFilePath); } catch(e) { }
18
19   const db = connectToDatabase();
20   db.exec(`
21     CREATE TABLE confiscari
22     (
23       an NUMERIC,
24       nume_drog TEXT,
25       grame NUMERIC,
26       nr_capturi NUMERIC
27     );
28     CREATE TABLE persoane_cercetate
29     (
30       an NUMERIC,
31       tip_persoane TEXT,
32       nr_persoane NUMERIC
33     );
34   `);
35 }
```

5. Rute

Toate rutele API utilizate in constructia site-ului sunt definite in fisierul `RequestRouter` . Acestea sunt utilizate pentru autentificare, inregistrare, obtinerea de statistici, generarea de fisiere CSV si alte operatiuni relevante pentru aplicatia web. La inceputul fisierului, exista o sectiune pentru importuri, in care sunt incluse modulele si functiile necesare pentru functionalitatea rutelor, dupa care este definita functia principala “`routeRequest`”, care gestioneaza toate cererile primite de la client. Aceasta verifica metoda HTTP impreuna cu URL-ul fiecărei cereri si apeleaza functiile corespunzatoare pentru a trata aceste cereri. Sunt utilizate metode HTTP specific REST, pentru manipularea resurselor si comunicarea intre server si client.

```
1 import { GetAni, GetDroguri, GetStatisticsConfiscari, GetStatisticsConfiscariAni, GetStatisticsCercetariAni, GetStatisticsPedepseAni,
2 GetPedepse, GetPersoane, GetStatisticsCercetari, GetStatisticsUrgente_gen, GetStatisticsPedepse, GetStatisticsUrgente_drog, GetUrgente_drog, GetStatisticsUrgente,
3 GetUrgente, GetLocatii, GetStatisticsUrgenteAni, GetStatisticsProiecte, GetStatisticsCampanii, GetNumeCampanii, GetStatisticsCondamnari, GetStatisticsActivitatiAni,
4 GetStatisticsCampaniiAni, GetStatisticsActivitati, GetUsers, getDate } from './statistics.mjs';
5 import { getParams } from './utilities.mjs';
6 import { decodeToken, loginUser, registerUser } from './login.mjs';
7 import fs from 'fs';
8 import crypto from 'crypto';
9 import sqlite3 from 'sqlite3';
10 const dbFilePath = '../database.db';
11
12
13 export const routeRequest = (req, response) => {
14
15 >   if(req.method === 'POST' && req.url === '/api/authenticate_user') { ...
36 >   }
37 >   else if(req.method === 'POST' && req.url === '/api/register_user') { ...
55 >   }
56 >   else if(req.method === 'GET' && req.url === '/api/check_auth') { ...
69 >   }
70 >   else if (req.method === 'GET' && req.url.includes('/api/statistics/droguri')) { ...
80 >   }
81 >   else if(req.method === 'GET' && req.url === '/api/statistics/ani') { ...
90 >   }
91 >   else if(req.method === 'GET' && req.url === '/api/statistics/persoane') { ...
100 >   }
101 >   else if(req.method === 'GET' && req.url === '/api/statistics/pedepse') { ...
110 >   }
111 >   else if(req.method === 'GET' && req.url.includes('/api/statistics/confiscari_tabel')) { ...
127 >   }
128 >   else if(req.method === 'GET' && req.url.includes('/api/statistics/confiscari_bar')) { ...
144 >   }
145 >   else if(req.method === 'POST' && req.url === '/api/statistics/confiscari_generate_csv') { ...
172 >   }
173 >   else if(req.method === 'GET' && req.url.includes('/api/statistics/confiscari_line')) { ...
186 >   }
187 >   else if(req.method === 'GET' && req.url.includes('/api/statistics/infractiuni_line')) { ...
200 >   }
208 >   }
209 }
```

În fișierul `statistics.mjs` sunt definite toate metodele responsabile de obținerea statisticilor, acestea oferind modularitate și flexibilitate în manipularea datelor prin utilizarea unor conexiuni SQLite. Folosind pachetul `SQLite3`, se realizează conexiunea cu baza de date specificată în variabila `'dbFilePath'` și se execută interogări pentru a obține informațiile dorite.

```
93 export const GetStatisticsCercetariAni = (callback) => {
94   let query = "select an, sum(nr_persoane) as nr_persoane from persoane_cercetate group by an"
95   const db = new sqlite3.Database(dbFilePath);
96
97   console.log("Executing '${query}'");
98   db.all(
99     query,
100     {},
101     (err, rows) => {
102       if(err) {
103         {
104           console.log(err);
105           callback(err);
106         }
107         db.close();
108         callback(rows);
109       }
110     });
111   return query;
112 }
113
114 export const GetStatisticsPedepseAni = (callback) => {
115   let query = "select an, sum(nr_pedepse) as nr_pedepse from situatia_pedepselor group by an"
116   const db = new sqlite3.Database(dbFilePath);
117
118   console.log("Executing '${query}'");
119   db.all(
120     query,
121     {},
122     (err, rows) => {
123       if(err) {
124         {
125           console.log(err);
126           callback(err);
127         }
128         db.close();
129         callback(rows);
130       }
131     });
132 }
```

6. Server

`Server.mjs` instantiază un server HTTP static și API care servește fișiere statice și gestionează cererile către ruta `'/api/'` într-o aplicație web. Scopul este de a furniza resursele statice (fișiere HTML, CSS, imagini etc.) către client și de a permite comunicarea cu serverul prin intermediul API-ului definit.

Serverul este creat folosind modulul `'http'` din Node.js și este configurat pentru a asculta pe un port specific (`'PORT'`). În cadrul serverului, este definit un obiect `'MIME_TYPES'` care conține tipurile de conținut asociate diferitelor extensii de fișiere. Acesta este utilizat pentru a seta antetul `'Content-Type'` în răspunsurile HTTP.

Directorul static în care se află fișierele serverului este specificat prin calea `'STATIC_PATH'`. Acesta este combinat cu URL-ul cererii pentru a obține calea către fișierul solicitat. Funcția `'prepareFile()'` este responsabilă de pregătirea fișierului pentru a fi trimis către client. Ea verifică existența fișierului, gestionează situația traversării ilegale a căilor și returnează un obiect care conține informații despre găsirea fișierului, extensia și fluxul de citire a fișierului.

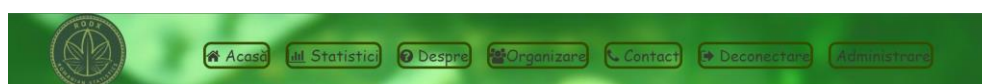
Serverul gestionează cererile primite prin rutarea acestora către API sau prin servirea de fișiere statice. Dacă URL-ul începe cu `/api/`, cererea este rutată către funcția `routeRequest()` din modulul `requestRouter.mjs`, care se ocupă de gestionarea logicii de la nivelul API-ului. În caz contrar, serverul pregătește și trimite fișierul static către client, inclusiv anteturile corespunzătoare.

În final, serverul este pornit prin apelul funcției `listen()` pe portul specificat. Se afișează un mesaj în consolă pentru a indica că serverul rulează și pe ce adresă este disponibil.

```
1 import * as fs from 'node:fs';
2 import * as http from 'node:http';
3 import * as path from 'node:path';
4 import { routeRequest } from './api/requestRouter.mjs';
5
6 const PORT = process.env.port || 8027;
7
8 const MIME_TYPES = {
9   default: 'application/octet-stream',
10  html: 'text/html, charset=UTF-8',
11  js: 'application/javascript',
12  css: 'text/css',
13  png: 'image/png',
14  jpg: 'image/jpeg',
15  gif: 'image/gif',
16  ico: 'image/x-icon',
17  svg: 'image/svg+xml',
18 };
19
20 const STATIC_PATH = path.join(process.cwd(), './static');
21
22 const toBool = (() => true, () => false);
23
24 const prepareFile = async (url) => {
25   const paths = [STATIC_PATH, url];
26   if (url.endsWith('/')) paths.push('login.html');
27   const filePath = path.join(...paths);
28   const pathTraversal = !filePath.startsWith(STATIC_PATH);
29   const exists = await fs.promises.access(filePath).then(...toBool);
30   const found = !pathTraversal && exists;
31   const streamPath = found ? filePath : STATIC_PATH + '/404.html';
32   const ext = path.extname(streamPath).substring(1).toLowerCase();
33   const stream = fs.createReadStream(streamPath);
34   return { found, ext, stream };
35 };
36
37 http.createServer(async (req, res) => {
38   if (req.url.startsWith('/api/')) {
39     routeRequest(req, res);
40   }
41   else {
42     const file = await prepareFile(req.url);
43     const statusCode = file.found ? 200 : 404;
44     const mimeType = MIME_TYPES[file.ext] || MIME_TYPES.default;
45     res.writeHead(statusCode, { 'Content-Type': mimeType });
46     file.stream.pipe(res);
47     console.log(`${req.method} ${req.url} ${statusCode}`);
48   }
49 }).listen(PORT);
50
```

7. Administrare

La logarea cu date de administrare (username: admin, parola: admin) se ofera acces catre o pagina care permite vizualizarea utilizatorilor din baza de date, impreuna cu data ultimei logari in aplicatie. La fiecare logare, in LocalStorage este stocat rolul pe care il are utilizatorul (normal user/admin), iar in cazul in care acesta are rol de administrator, programul permite aparitia, in bara de navigare, a unui nou buton corespunzator acesti pagini de administrare.



Administrare	
Nume utilizator	Ultima conectare
admin	22-06-2023
test	22-06-2023
test2	22-06-2023
tu	22-06-2023

8. Design

Designul aplicației pune accent pe ușurința în utilizare și accesibilitate, asigurând că utilizatorii pot accesa rapid și eficient toate informațiile relevante într-un mod plăcut și intuitiv. O bară de navigare intuitivă permite accesul rapid către secțiunea de statistici, oferindu-le o privire panoramică asupra datelor relevante privind infraccționalitatea, confiscările și urgențele medicale asociate consumului de droguri. Nuanțele de verde creează o atmosferă caldă și reconfortantă, evocând natura și evidențiind importanța protejării sănătății și a conștientizării cu privire la problemele legate de droguri.

