Aplicație pentru gestionarea controalelor la granițele din românia

PROIECT DE DIPLOMĂ

Autor: Dennis-Raul DRĂGUȘIN

Conducător științific: SL. Dr. Ing. Mihai HULEA

|  |  |
| --- | --- |
| DECAN  **Prof. dr. ing. Liviu MICLEA** | Vizat,  DIRECTOR DEPARTAMENT AUTOMATICĂ  **Prof. dr. ing. Honoriu VĂLEAN** |

Autor: Dennis-Raul DRĂGUȘIN

Aplicație pentru gestionarea controalelor la granițele din România

1. **Enunțul temei:** *O scurtă descriere a temei proiectului de diplomă*
2. **Conținutul proiectului:** *(enumerarea părților componente) Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Titlul capitolului 1, Titlul capitolului 2,… Titlul capitolului n, Bibliografie, Anexe.*
3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca, alte locuri dacă este cazul*
4. **Consultanți:** *ing. Prenume Nume (dacă este cazul)*
5. **Data emiterii temei:**
6. **Data predării:**

Semnătura autorului

Semnătura conducătorului științific

**Declaraţie pe proprie răspundere privind**

**autenticitatea proiectului de diplomă**

Subsemnatul(a) **Prenume NUME**  , legitimat(ă) cu CI/BI seria nr. , CNP ,

autorul lucrării:

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, specializareaChoose an item.**,** din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Choose an item. a anului universitar 2022-2023, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data Prenume NUME

(semnătura)

**SINTEZA**

proiectului de diplomă cu titlul:

Titlul lucrării

Autor: **Prenume NUME**

Conducător științific: **Titlu. ing. Prenume NUME**

1. Cerințele temei:

2. Soluții alese:

3. Rezultate obținute:

4. Testări și verificări:

5. Contribuții personale:

6. Surse de documentare:

Semnătura autorului

Semnătura conducătorului științific

Cuprins

[1 Introducere 3](#_Toc138253581)

[1.1 Context general 3](#_Toc138253582)

[1.2 Obiective 4](#_Toc138253583)

[1.3 Specificații 4](#_Toc138253584)

[2 Studiu bibliografic 6](#_Toc138253585)

[2.1 Controalele la granițele din România 6](#_Toc138253586)

[2.1.1 Punctul de trecere a frontierei 6](#_Toc138253587)

[2.1.2 Documentele de călătorie și înmatriculare 6](#_Toc138253588)

[2.1.3 Declarația vamală 6](#_Toc138253589)

[2.1.4 Controlul fizic al vehiculului 7](#_Toc138253590)

[2.1.5 Verificarea documentelor de însoțire a mărfurilor 7](#_Toc138253591)

[2.1.6 Controlul fiscal și vamal 7](#_Toc138253592)

[2.1.7 Controlul de securitate 7](#_Toc138253593)

[2.2 Node.js 7](#_Toc138253594)

[2.2.1 Arhitectura orientată pe evenimente 8](#_Toc138253595)

[2.2.2 Modulul NPM 8](#_Toc138253596)

[2.2.3 Suportul pentru I/O neliniar 9](#_Toc138253597)

[2.2.4 Server web încorporat 9](#_Toc138253598)

[2.2.5 Comunitatea și suportul 9](#_Toc138253599)

[2.3 React 10](#_Toc138253600)

[2.4 Google Maps API 11](#_Toc138253601)

[2.5 JavaScript 12](#_Toc138253602)

[2.6 Utilizarea aplicațiilor web în gestionarea granițelor 13](#_Toc138253603)

[3 Analiză 15](#_Toc138253604)

[3.1 Specificațiile aplicației 15](#_Toc138253605)

[3.1.1 Înregistrare și Autentificare 15](#_Toc138253606)

[3.1.2 Gestionarea controalelor 16](#_Toc138253607)

[3.1.3 Gestionarea granițelor 17](#_Toc138253608)

[3.2 Concepte de bază ale aplicației 18](#_Toc138253609)

[3.2.1 Arhitectura REST 18](#_Toc138253610)

[3.2.2 Arhitectura MVC 20](#_Toc138253611)

[3.3 Analiza riscurilor aplicației 21](#_Toc138253612)

[3.3.1 Riscul de securitate 22](#_Toc138253613)

[3.3.2 Riscul de erori de programare 22](#_Toc138253614)

[3.3.3 Riscul de scalabilitate 23](#_Toc138253615)

[3.3.4 Riscul de erori de integrare 23](#_Toc138253616)

[3.3.5 Riscul de dependențe tehnologice 23](#_Toc138253617)

[4 Proiectare 24](#_Toc138253618)

[4.1 Cazuri de utilizare 24](#_Toc138253619)

[4.1.1 Cazuri de utilizare membru 24](#_Toc138253620)

[4.1.2 Cazuri de utilizare administrator 26](#_Toc138253621)

[4.2 Diagrame de secvență 28](#_Toc138253622)

[4.2.1 Diagrama de secvență a autentificării utilizatorilor 28](#_Toc138253623)

[4.2.2 Diagrama de secvență a înregistrării unui utilizator 29](#_Toc138253624)

[4.3 Integrarea serviciilor externe în aplicație (Google Maps) 30](#_Toc138253625)

[4.4 Structura bazei de date 31](#_Toc138253626)

[4.4.1 Alegerea bazei de date 31](#_Toc138253627)

[4.4.2 Structura unei colecții în baza de date 32](#_Toc138253628)

[4.5 Designul interfeței utilizatorului 33](#_Toc138253629)

[5 Implementare 35](#_Toc138253630)

[5.1 Prezentarea componentelor principale 35](#_Toc138253631)

[5.1.1 Client 35](#_Toc138253632)

[5.1.1.1 Pagina de vizualizare a controalelor de mașini 39](#_Toc138253633)

[5.1.1.2 Pagina de introducere a unui control de mașini 40](#_Toc138253634)

[5.1.2 Server 41](#_Toc138253635)

[5.2 Testare 43](#_Toc138253636)

[5.2.1 Testarea serverului 43](#_Toc138253637)

[5.2.2 Testarea clientului 44](#_Toc138253638)

[6 Concluzii 46](#_Toc138253639)

[6.1 Rezultate obținute 46](#_Toc138253640)

[6.2 Direcții de dezvoltare 46](#_Toc138253641)

[7 Bibliografie 48](#_Toc138253642)

# Introducere

## Context general

Într-o lume in care tehnologia evoluează rapid și în care există o nevoie tot mai mare de eficiență și automatizare, dezvoltarea de aplicații web devine o preocupare importantă pentru mulți dezvoltatori de software. Aceste aplicații web pot oferi soluții pentru diverse probleme într-o multitudine de domenii, inclusiv în sectorul vamal.

Această lucrare de licență are ca scop dezvoltarea unei aplicații web pentru gestionarea controalelor de la granițele României pentru autovehiculele de transport rutier. Necesitatea de a avea un sistem automatizat pentru a monitoriza controalele la granițele terestre și pentru a oferi o mai bună gestionare a acestora, motivează crearea unei astfel de aplicații.

Prin intermediul acestei aplicații, utilizatorii au posibilitatea de a introduce controale de mașini și camioane și de a vizualiza aceste controale într-un mod simplu și intuitiv. Mai mult decât atât, aplicația oferă o interfață de hartă interactivă, cu ajutorul căreia utilizatorii pot să vizualizeze granițele României și controalele asociate cu fiecare în parte.

Importanța acestei lucrări este dată de faptul că aplicația dezvoltată poate aduce o serie de beneficii, precum reducerea timpului de așteptare la graniță și îmbunătățirea siguranței rutiere. De asemenea, poate fi folosită ca instrument pentru analiza datelor și îmbunătățirea proceselor de gestionare a controalelor la granițe.

Aplicația a fost dezvoltată utilizând tehnologiile Node.js, MongoDB și React, iar implementarea a fost aleasă din cauza flexibilității și eficienței lor. Tehnologiile menționate permit dezvoltatorilor să construiască aplicații complexe si scalabile cu ușurință.

Această lucrare de licență este structurată în cinci capitole. Primul capitol prezintă o introducere generală în lucrare și motivele pentru care aceasta a fost dezvoltată. Al doilea capitol descrie studiul bibliografic și analiza datelor din domeniul transporturilor și al controlului la granițe. Capitolul trei descrie procesul de analiză, proiectare și implementare a aplicației web dezvoltate. Al patrulea capitol oferă concluzii și direcții de dezvoltare ulterioară a aplicației. Ultimul capitol este reprezentat de bibliografia lucrării, unde se regăsesc toate sursele din care au fost extrase informații.

## Obiective

Obiectivele acestei lucrări de licență sunt următoarele:

1. Realizarea unei aplicații web moderne, folosind tehnologii actuale si populare, precum Node.js, MongoDB si React, pentru a gestiona controalele de la granițele României într-un mod ușor și eficient.
2. Crearea unei interfețe grafice intuitive și ușor de folosit, care să permită utilizatorilor să adauge și să vizualizeze controalele la granițele României în timp real.
3. Integrarea hărții Google Maps în aplicație pentru a permite vizualizarea granițelor din România și a controalelor asociate acestora. Prin intermediul acestei funcții, utilizatorii vor putea vizualiza granița pe hartă și vor putea accesa informații detaliate despre controalele din zona respectiva.
4. Posibilitatea utilizatorilor de a filtra controalele în funcție de tipul acestora, mai exact dacă sunt controale pentru mașini sau controale pentru camioane. Această funcție va oferi utilizatorilor o mai mare flexibilitate în gestionarea controalelor și o mai bună organizare a acestora.
5. Identificarea și rezolvarea problemelor legate de securitatea și integritatea datelor utilizatorilor. Prin intermediul utilizării bazei de date MongoDB și prin implementarea unor măsuri de securitate adecvate, se asigură că datele utilizatorilor sunt protejate și nu pot fi accesate de persoane neautorizate.
6. Înțelegerea și aplicarea conceptelor teoretice din domeniul programării web, bazate pe arhitectura client-server, RESTful API și baze de date de tip NoSQL.
7. Testarea și validarea aplicației, atât pe plan funcțional cât și pe plan al performanței, pentru a garanta o experiență cât mai bună pentru utilizatori.

Aceste obiective au scopul de a îmbunătății eficiența procesului de monitorizare a traficului de vehicule și persoane la frontieră. Obiectivele vor fi atinse prin intermediul analizei și proiectării aplicației, implementării și testării acesteia, precum și prin prezentarea direcțiilor de dezvoltare ulterioară.

## Specificații

În ceea ce privește specificațiile lucrării, mi-am propus să realizez o aplicație web care să permită utilizatorilor să introducă și să vizualizeze controalele efectuate la granițele României asupra vehiculelor de transport rutier. Aplicația este destinată granițelor din România, astfel doar angajații granițelor pot fi utilizatori ai aplicației.

Aplicația prezintă un număr de funcționalități, precum posibilitatea de a introduce informații referitoare la controalele efectuate, vizualizarea și filtrarea controalelor în funcție de granița la care a fost efectuat controlul sau în funcție de tipul de vehicul (mașini sau camioane). Controalele sunt introduse sub forma unui formular și pot fi vizualizate într-un tabel. Aplicația are integrată o hartă interactivă pe care se pot vizualiza granițele din România și se pot accesa controalele de la granițele respective.

Interfața este intuitivă și ușor de utilizat, astfel încât utilizatorii sa aibă acces cât mai rapid la informațiile căutate și de asemenea securitatea datelor și confidențialitatea informațiilor introduse in aplicație să fie asigurate.

În privința nivelului de performanță al aplicației, mi-am propus ca aplicația să aibă un timp de răspuns cât mai rapid, pentru a răspunde solicitărilor utilizatorilor într-un timp cât mai scurt. Astfel informațiile vor fi puse la dispoziția utilizatorilor fără întârzieri.

În ceea ce privește structurile de date, am utilizat o bază de date non-relațională pentru stocarea informațiilor despre controale, granițe și utilizatori. Calitatea si fiabilitatea datelor introduse în baza de date este asigurată prin utilizarea unor mecanisme de verificare și validare.

O limitare importantă este legată de gestionarea și scalabilitatea datelor. Volumul de date stocate in baza de date depinde de nivelul de acumulare de informații despre controale, granițe și utilizatori ai aplicației. Astfel, performanța aplicației poate fi afectată și poate necesita optimizări suplimentare pentru a asigura o funcționare fluentă.

# Studiu bibliografic

## Controalele la granițele din România

Un control la granițele României este un proces prin care autoritățile vamale și de securitate verifică persoanele și bunurile care intră sau părăsesc teritoriul țării. Acest control are ca scop asigurarea securității, respectarea legilor și regulamentelor vamale și prevenirea infracțiunilor transfrontaliere. In subcapitolele următoare se va prezenta o descriere generală a modului în care decurge un astfel de control.

Este important de menționat că procedurile exacte și nivelul de detaliu al controlului pot varia în funcție de punctul de trecere a frontierei, de legislația în vigoare și de politica de securitate aplicată în acel moment.

Informațiile prezentate sunt bazate pe cunoștințele generale despre controlul la graniță și pot varia în funcție de circumstanțe specifice. Informații detaliate sunt prezentate în „Regulamentul de aplicare a Codului vamal al României din 07.06.2006” [1].

### Punctul de trecere a frontierei

Autovehiculele se prezintă la punctul de trecere a frontierei desemnat, unde există benzi distincte pentru controlul autovehiculelor, in funcție de tipul de autovehicul.

Echipa de control de frontieră coordonează fluxul vehiculelor și dirijează conducătorii auto către benzile de control adecvate.

### Documentele de călătorie și înmatriculare

La controlul de frontieră, conducătorul autovehiculului trebuie să prezinte documentele personale de călătorie, cum ar fi pașaportul sau cartea de identitate.

De asemenea, trebuie să prezinte documentele de înmatriculare ale vehiculului, cum ar fi certificatul de înmatriculare, asigurarea obligatorie și inspecția tehnică periodică.

### Declarația vamală

Conducătorul autovehiculului completează o declarație vamală în care declară bunurile transportate în vehicul. Aceasta include informații despre natura, cantitatea și valoarea bunurilor transportate.

Scopul declarației vamale este de a permite autorităților vamale să evalueze riscurile potențiale și să identifice mărfurile supuse controlului vamal.

Regulamentul declarației vamale face parte din „Regulamentul de aplicare a Codului vamal al României din 07.06.2006” [1].

### Controlul fizic al vehiculului

Autoritățile vamale pot efectua un control fizic al vehiculului pentru a verifica conținutul acestuia. Acesta poate implica deschiderea portbagajului, a spațiului de marfă sau a altor compartimente relevante.

Scopul controlului fizic este de a detecta și preveni contrabanda, traficul de droguri, traficul de persoane și alte activități ilegale.

### Verificarea documentelor de însoțire a mărfurilor

Dacă sunt transportate mărfuri comerciale, autoritățile vamale pot verifica documentele de însoțire a mărfurilor, cum ar fi facturile, avizele de însoțire sau alte documente necesare pentru evidența și controlul mărfurilor.

Scopul verificării documentelor de însoțire este de a asigura conformitatea cu legislația vamală și fiscală în vigoare și de a preveni evaziunea fiscală și frauda vamală.

### Controlul fiscal și vamal

Autoritățile vamale efectuează controlul fiscal și vamal pentru a verifica respectarea regulilor și regulamentelor fiscale și vamale.

Acest control include verificare plății taxelor și impozitelor corespunzătoare, verificarea declarării corecte a mărfurilor și a valorii acestora, precum și aplicarea altor prevederi fiscale și vamale relevante.

### Controlul de securitate

În funcție de nivelul de securitate și de riscurile identificate, se pot efectua verificări suplimentare pentru a preveni infracțiunile și a asigura siguranța.

Acestea pot include scanarea vehiculului cu echipamente de detectare a substanțelor interzise, verificarea identității conducătorului auto și a pasagerilor, utilizarea tehnologiilor de screening pentru depistarea armelor sau a altor obiecte periculoase.

## Node.js

Node.js, conform „Introduction to Node.js” [2] este o tehnologie open-source care permite dezvoltarea aplicațiilor de tip server-side folosind limbajul de programare JavaScript. Tehnologia a fost creată de Ryan Dahl în anul 2009 și este construită pe baza limbajului de programare JavaScript , dezvoltat de Google.

Node.js este folosit într-o varietate de aplicații, de la aplicații web, aplicații mobile,

servicii de rețea, aplicații în timp real până la microservicii și aplicații IoT (Internet of

Things). Prin abordarea sa eficientă și scalabilă, Node.js a câștigat popularitate și a devenit o alegere preferată pentru dezvoltarea aplicațiilor dinamice și performante la nivel de server.

### Arhitectura orientată pe evenimente

Arhitectura orientată pe evenimente în cadrul Node.js reprezintă o paradigmă de proiectare software în care aplicația este construită în jurul conceptului de evenimente și răspunsuri la aceste evenimente. Într-o astfel de arhitectură, Node.js reacționează la

evenimente care apar în timpul execuției aplicației, cum ar fi cereri primite de la client, finalizarea unei operațiuni de citire sau scriere a fișierelor sau alte evenimente specifice aplicației.

Un aspect important al acestei arhitecturi este evitarea blocării firului principal de execuție. În loc să aștepte sincronizat finalizarea fiecărei operațiuni, Node.js înregistrează o funcție de callback care va fi apelată atunci când evenimentul se produce. Aceasta permite aplicației să continue să execute alte operațiuni în paralel.

Această abordare asincronă și neliniară oferă o serie de beneficii. În primul rând, permite aplicației să fie foarte eficientă în gestionarea mai multor cereri și evenimente simultan. De exemplu, un server Node.js poate primi multiple cereri HTTP în același timp și poate gestiona fiecare cerere independent, fără a afecta performanța generală a aplicației.

În al doilea rând, arhitectura orientată pe evenimente facilitează dezvoltarea aplicațiilor scalabile. Deoarece operațiile de tip I/O și alte operațiuni care pot dura mai mult timp sunt gestionate în mod asincron, firul principal de execuție rămâne disponibil pentru a procesa alte cereri sau evenimente. Acest lucru permite aplicației să răspundă rapid la cereri și să evite blocarea sau îngreunarea în execuția operațiunilor costisitoare.

### Modulul NPM

Modulul NPM (Node Package Manager),conform “npm Docs” [3] reprezintă un instrument esențial în dezvoltarea aplicațiilor Node.js, oferind un sistem de gestionare a pachetelor și dependențelor. Prin intermediul NPM, dezvoltatorii au acces la un vast ecosistem de pachete și module open-source, care pot fi instalate și utilizate în aplicații Node.js.

NPM facilitează procesul de instalare și gestionare a dependențelor într-un mod simplu și eficient. Dezvoltatorii pot specifica dependențele necesare pentru proiectul lor într-un fișier de configurare(de obicei, package.json) și apoi, prin intermediul comenzii „npm install”, aceste dependențe sunt descărcate și instalate automat în proiect.

Beneficiile utilizării NPM sunt multiple. În primul rând, prin utilizarea modulelor

existente, dezvoltatorii pot economisi timp și efort, evitând dezvoltarea de funcționalități complexe sau comune de la zero. Există o gamă largă de pachete disponibile, care acoperă o varietate de funcționalități, cum ar fi gestionarea bazei de date, manipularea imaginilor, autentificarea utilizatorilor și multe altele.

În al doilea rând, NPM facilitează gestionarea versiunilor și actualizarea pachetelor. Dezvoltatorii pot specifica versiuni specifice ale unui pachet în fișierul de configurare, asigurând astfel consistența și compatibilitatea cu dependențele existente.

De asemenea, NPM oferă un sistem robust pentru publicarea și partajarea pachetelor dezvoltate de comunitatea Node.js. Dezvoltatorii pot împărtăși propriile pachete cu restul comunității, permițându-le să fie utilizate și contribuite de alți dezvoltatori. Astfel se încurajează colaborarea și partajarea de soluții și module între membrii comunității.

### Suportul pentru I/O neliniar

Node.js este optimizat pentru operațiile de I/O neliniar, cum ar fi citirea și scrierea fișierelor, comunicarea cu baze de date sau apeluri de rețea. În loc să aștepte sincronizat finalizarea acestor operațiuni, Node.js utilizează apeluri asincrone si callback-uri pentru a continua execuția și a răspunde la alte cereri în timp ce operațiile I/O sunt în desfășurare. Acest lucru face ca aplicațiile Node.js să fie scalabile și să ofere performanță ridicată, deoarece nu blochează firul principal de execuție pentru operațiile de I/O.

### Server web încorporat

Unul dintre avantajele majore ale Node.js este disponibilitatea unui server web încorporat. Acest lucru înseamnă că dezvoltatorii pot crea și gestiona servere web direct în aplicațiile lor, din cauza funcționalităților, cum ar fi primirea și procesarea cererilor HTTP, precum și pentru generarea unor răspunsuri adecvate.

Utilizarea unui server web încorporat în Node.js are mai multe beneficii. Un prim exemplu ar fi eliminarea nevoilor de instalare și configurare a unui server web separat, reducând astfel complexitatea dezvoltării și implementării aplicațiilor web.

Un alt avantaj ar fi controlul mai mare asupra logicii aplicației. Dezvoltatorii pot configura rute personalizate, manipula cererile HTTP și genera răspunsuri adecvate în funcție de nevoile aplicației lor. Acest nivel de control permite crearea de aplicații web flexibile și personalizate.

### Comunitatea și suportul

Node.js beneficiază de o comunitate activă și implicată, deoarece dezvoltatori din întreaga lume contribuie la dezvoltarea și îmbunătățirea continuă a tehnologiei, aducând

noi idei, soluții și module care să extindă funcționalitățile și să răspundă nevoilor comunității.

Pe lângă resursele online variate disponibile, comunitatea Node.js pune la dispoziție și un ecosistem vast de pachete și module open-source. Acestea sunt create și menținute de dezvoltatori din comunitate și oferă soluții pregătite pentru utilizare în diferite aspecte ale dezvoltării de aplicații. Există pachete pentru gestionarea bazelor de date, crearea interfețelor utilizatorilor, implementarea securității, testare și multe altele. Accesul la aceste module și pachete open-source permite dezvoltatorilor să economisească timp și efort prin utilizarea unor soluții testate și validate de comunitate.

De asemenea, dezvoltatorii sunt încurajați să contribuie la comunitatea Node.js. Contribuțiile pot fi sub forma unor corecturi de bug-uri, îmbunătățiri de performanță, noi funcționalități sau simpla împărtășire de feedback și idei.

## React

Tehnologia React este o bibliotecă JavaScript utilizată pentru dezvoltarea interfețelor utilizator în aplicații web. Creată de către Meta, React se bazează pe conceptul de reprezentare virtuală a interfeței, ceea ce permite o manipulare eficientă și reactivă a elementelor interfeței. React dispune de o documentație vastă care poate fi găsită pe site-ul React [4].

Unul dintre principiile fundamentale ale React este reprezentat de conceptul de “componente”. O componentă React este o entitate independentă și reutilizabilă, care poate fi compusă din alte componente mai mici. Această abordare modulară facilitează dezvoltarea și întreținerea aplicațiilor, deoarece permite separarea logicii și a aspectului vizual în componente distincte, care pot fi reutilizate în mai multe contexte.

Un aspect cheie al React este modul în care gestionează actualizările interfeței utilizator. React utilizează o abordare numită „reconciliere virtuală”, în care reprezentarea virtuală a interfeței este actualizată și comparată cu starea anterioară a interfeței. Pe baza acestui proces de comparație, React determină diferențele și actualizează doar elementele afectate, ceea ce optimizează performanța aplicațiilor.

React utilizează, de asemenea, un sistem de “proprietăți” (props) pentru a gestiona transferul de date între componente. Prin intermediul acestui sistem, componentele pot primi date externe (proprietăți) și le pot utiliza pentru a-și actualiza starea și interfața. Acest model simplifică comunicarea între componente și facilitează dezvoltarea de componente modulare și flexibile.

Pentru a gestiona starea internă a componentelor, React introduce conceptul de “stare” (state). Starea reprezintă datele specifice unei componente și poate fi actualizată în timpul interacțiunii utilizatorului sau în forma unor evenimente. React oferă metode și mecanisme pentru a gestiona și actualiza starea componentelor într-un mod eficient și consistent.

Un alt aspect important al React este faptul că este bazat pe paradigma programării declarative. Acest lucru înseamnă că dezvoltatorii descriu cum ar trebui să arate interfața utilizator într-un anumit moment, iar React se ocupă de actualizarea și reprezentarea interfeței în mod automat. Această abordare simplifică dezvoltarea și întreținerea aplicațiilor, deoarece dezvoltatorii se pot concentra pe descrierea rezultatului dorit, fără a fi nevoie să gestioneze manual actualizările interfeței.

## Google Maps API

Google Maps API este o suită de instrumente și servicii oferite de Google, care permit dezvoltatorilor să integreze și să utilizeze funcționalitățile avansate ale hărților și locațiilor în aplicațiile lor web și mobile. API-ul Google Maps oferă un set bogat de funcționalități și opțiuni de personalizare, permițând dezvoltatorilor să creeze experiențe de hartă interactive și captivante. Google pune la dispoziție o documentație în detaliu pe pagina oficială “Google Maps Platform” [5].

Un aspect central al Google Maps API este reprezentat de hărți interactive, care permit utilizatorilor să vizualizeze și să interacționeze cu informațiile geografice într-un mod intuitiv. Dezvoltatorii pot integra hărți în aplicațiile lor și pot personaliza aspectul, nivelul de detaliu și interacțiunile disponibile. Prin intermediul API-ului, aceștia au acces la funcții de zoom, panoramare, rotire și multe altele, pentru a oferi o experiență de utilizare fluidă și precisă.

Un alt aspect important al Google Maps API este funcționalitatea de geolocație. Prin intermediul serviciilor de localizare și geocodificare, dezvoltatorii pot obține informații despre locația actuală a utilizatorilor sau pot transforma adrese în coordonate geografice și viceversa. Aceasta permite dezvoltatorilor să ofere funcționalități bazate pe locație, cum ar fi găsirea celor mai apropiate puncte de interes sau direcționarea utilizatorilor către anumite destinații.

Un alt aspect ce merită menționat este capacitatea de a adăuga markeri, etichete și alte elemente grafice pe hartă. Dezvoltatorii pot personaliza aspectul și comportamentul acestor elemente și pot adăuga informații suplimentare, cum ar fi imagini, descrieri și link-uri asociate. Aceasta permite dezvoltatorilor să ofere informații contextuale și interacțiuni suplimentare utilizatorilor lor.

Google Maps API oferă, de asemenea, funcționalități avansate, cum ar fi indicarea rutelor, calcularea distanțelor și duratelor de călătorie, precum și căutarea de locuri și adrese. Aceste funcționalități permit să creeze aplicații cu trasee personalizate, estimări ale timpului de sosire și alte caracteristici legate de navigație și planificare a călătorilor.

Pentru a utiliza Google Maps API, dezvoltatorii trebuie să obțină o cheie de autentificare, care le oferă acces la serviciile și funcționalitățile oferite de Google. Aceasta asogură securitatea și controlul asupra utilizării API-ului și permite dezvoltatorilor să monitorizeze și să gestioneze utilizarea și limitările acestuia.

## JavaScript

JavaScipt este un limbaj de programare versatil și dinamic, utilizat pe scară largă în dezvoltarea aplicațiilor web și mobile. Acest limbaj oferă posibilitatea de a crea interactivitate și funcționalitate în pagini web, permițând dezvoltatorilor să manipuleze elementele, să gestioneze evenimente și să interacționeze cu utilizatorii. Acest limbaj de programare stă la baza multor tehnologii, cum ar fi cele menționate in capitolele anterioare, React și Node.js. Aceste tehnologii folosesc majoritatea proprietăților si funcționalităților ale limbajului de programare JavaScript.

Un aspect fundamental al JavaScript este natura sa bazată pe prototipuri. În loc să folosească conceptul tradițional de clase, JavaScript utilizează obiecte și prototipuri pentru a crea structuri de date și funcționalități. Aceasta permite dezvoltatorilor să creeze obiecte și să le extindă dinamic, adăugând proprietăți și metode noi în timpul execuției.

JavaScript oferă suport pentru programarea orientată pe evenimente. Dezvoltatorii pot atașa funcții de callback la evenimente, cum ar fi click-uri de mouse sau trimiterea de formulare, pentru a răspunde la acțiunile utilizatorilor. Aceasta permite crearea de interactivitate în paginile web, facilitând manipularea și actualizarea dinamică a conținutului în funcție de acțiunile utilizatorilor.

Un alt aspect important al JavaScript este gestionarea asincronă a operațiilor. Acest limbaj permite desfășurarea de operații asincrone, cum ar fi cereri HTTP sau operații de citire/scriere în fișiere, fără a bloca firul principal de execuție. Prin intermediul promisiunilor și așteptărilor, dezvoltatorii pot gestiona fluxul asincron al programului, permițând execuția de operații lungi fără a afecta interactivitatea aplicației.

JavaScript oferă, de asemenea, suport pentru manipularea și modificarea elementelor HTML și CSS. Prin intermediul Document Object Model (DOM), dezvoltatorii pot accesa și modifica elementele și proprietățile paginilor web în timpul rulării. Acest lucru permite crearea de interfețe dinamice și personalizate, adaptate nevoilor și acțiunilor utilizatorilor.

Limbajul JavaScript este interpretat de către browser și poate fi executat direct în mediul acestuia, fără a fi necesară o etapă de compilare. Aceasta face ca dezvoltarea și testarea aplicațiilor să fie mai rapidă și mai ușoară, permițând dezvoltatorilor să vadă rezultatele imediat în timpul procesului de dezvoltare.

JavaScript beneficiază de o comunitate puternică și activă de dezvoltatori, care contribuie la dezvoltarea continuă a limbajului prin intermediul bibliotecilor, framework-urilor și instrumentelor adiționale. Aceasta oferă dezvoltatorilor acces la o varietate de resurse, documentație și suport, spre exemplu “The Modern JavaScript Tutorial” [6], permițându-le să își îmbunătățească cunoștințele și să rezolve problemele întâlnite în procesul de dezvoltare.

## Utilizarea aplicațiilor web în gestionarea granițelor

Utilizarea aplicațiilor web în gestionarea granițelor reprezintă o abordare modernă și eficientă în controlul și monitorizarea fluxului de persoane și vehicule la frontiere. Aceste aplicații web, pot fi dezvoltate pe baza unor tehnologii web avansate și pot aduce numeroase beneficii în ceea ce privește eficiența operațională, securitatea și interoperabilitatea între diferitele entități implicate în gestionarea granițelor.

Una dintre principalele avantaje ale utilizării aplicațiilor web în acest context este capacitatea lor de a permite o comunicare și colaborare facilă între diferitele organizații și instituții responsabile de gestionarea granițelor. Aceste aplicații facilitează schimbul rapid și securizat de informații între poliția de frontieră, vămi, instituții de securitate și alte entități relevante, contribuind astfel la îmbunătățirea coordonării și cooperării în controlul și monitorizarea fluxului la frontiere.

Un alt aspect important al utilizării aplicațiilor web în gestionarea granițelor este capacitatea lor de a facilita procesul de colectare, stocare și gestionare a datelor relevante. Aceste aplicații pot permite înregistrarea și actualizarea informațiilor despre persoanele și vehiculele care traversează frontierele, inclusiv detalii precum nume, naționalitate, documente de identitate, scopul călătoriei și altele. Prin utilizarea unor baze de date centralizate și interconectate, aceste aplicații web pot asigura un acces rapid și eficient la informațiile necesare în procesul de control al granițelor.

Un alt beneficiu al utilizării aplicațiilor web este capacitatea lor de a oferi funcționalități de monitorizare și gestionare a fluxului de persoane și vehicule la frontiere. Prin intermediul acestor aplicații, utilizatorii pot vizualiza în timp real informații despre punctele de control, timpul de așteptare, volumul de trafic și alți indicatori relevanți. Aceste informați pot fi utilizate pentru a optimiza procesul de control și pentru a lua decizii eficiente în ceea ce privește alocarea resurselor și gestionarea fluxului la frontiere.

În ceea ce privește securitatea, aplicațiile web utilizate pot fi proiectate pentru a asigura un nivel înalt de securitate a datelor și comunicațiilor. Utilizarea protocoalelor criptografice și a mecanismelor de autentificare ș autorizare asigură protecția informațiilor sensibile și previne accesul neautorizat. De asemenea, aceste aplicații web pot fi integrate cu sisteme de securitate suplimentare, cum ar fi sisteme de supraveghere video sau sisteme biometrice, pentru a spori nivelul de securitate la frontiere.

Utilizarea aplicațiilor in gestionarea granițelor poate aduce, de asemenea, beneficii în ceea ce privește analiza și raportarea datelor. Aceste aplicații pot furniza statistici și rapoarte personalizate privind fluxul de persoane și vehicule la frontiere, precum și analize de tendințe și modele de trafic. Aceste informații pot fi utilizate pentru a îmbunătăți planificarea și luarea deciziilor strategice în cadrul autorităților responsabile de gestionarea frontierelor.

Un exemplu de aplicație web ar fi cea a Poliției de Frontieră a României „Trafic Online” [7]. Aplicația este destinată publicului larg pentru a fi ținuți la curent cu situația actuală a granițelor din România. Aceasta prezintă o hartă interactivă pe care se pot vedea

toate granițele din România si timpul de așteptare a fiecăruia și alte diferite informații care pot fi comunicate publicului.

# Analiză

## Specificațiile aplicației

### Înregistrare și Autentificare

Înregistrarea și autentificarea într-o aplicație reprezintă două aspecte esențiale ale gestionării identității utilizatorilor. Aceste funcționalități permit utilizatorilor să acceseze și să interacționeze cu aplicația într-un mod sigur și personalizat.

Procesul de înregistrare permite utilizatorilor să creeze un cont în aplicație. Acesta implică colectarea de informații personale necesare, cum ar fi:

* adresa de e-mail
* numele de utilizator
* codul unic de angajat (badge number)
* parola

După ce utilizatorul completează formularul de înregistrare, aplicația validează și verifică datele furnizate. Această etapă include verificarea dacă adresa de e-mail este unică și dacă celelalte informații introduse sunt valide. În cazul în care datele sunt valide, utilizatorul este înregistrat în baza de date a aplicației și primește o confirmare a înregistrării.

Autentificarea reprezintă procesul prin care un utilizator confirmă identitatea sa pentru a accesa contul în aplicație. Autentificarea implică furnizarea unui set de credențiale, cum ar fi adresa de e-mail și parola, pentru a demonstra că utilizatorul este deținătorul contului respectiv.

La introducerea credențialelor, aplicația va verifica dacă acestea corespund cu informațiile stocate în baza de date. Dacă verificarea este reușită, utilizatorul este considerat autentificat și i se permite accesul în aplicație. Pentru a gestiona sesiunea utilizatorului, se creează un token de autentificare JSON Web Token (JWT), care va fi utilizat pentru a identifica utilizatorul în cererile ulterioare către server.

Atât înregistrarea, cât și autentificarea într-o aplicație trebuie să fie implementate cu măsuri solide de securitate pentru a proteja datele utilizatorilor prin criptarea datelor, cum ar fi utilizarea algoritmilor robuști de hașurare (hashing) pentru stocarea parolelor.

Implementarea autentificării și înregistrării eficiente și sigure reprezintă o componentă crucială pentru asigurarea confidențialității și securității datelor utilizatorilor.

A diagram of a web

Description automatically generated with low confidence

Figura 3.1 Diagrama de autentificare

### Gestionarea controalelor

Gestionarea controalelor este principala funcționalitate a aplicației web dezvoltate. Aceasta se bazează pe introducerea, filtrarea si vizualizarea controalelor de la granițele României. Există două tipuri de controale ce pot fi introduse: controale de mașini și controale de camioane. Acestea mai apoi pot fi filtrate prin intermediul unei hărți Google Maps, și afișate sub formă de tabel.

Pentru a introduce un nou control în aplicație, utilizatorul poate accesa funcționalitatea corespunzătoare prin intermediul unui buton dedicat. Prin completarea câmpurilor disponibile, se poate introduce o nouă înregistrare a controlului în sistem. Această informație este stocată într-o bază de date și poate fi ulterior accesată și vizualizată. În Figura 3.1 sunt prezentate câmpurile ce trebuie completate pentru fiecare tip de control (mașini/camioane). Controalele efectuate asupra camioanelor necesită completarea mai multor câmpuri față de controalele de mașini, deoarece camioanele au mai multe specificații care trebuie să fie luate în vedere, cum ar fi:

* Greutatea
* Înălțimea
* Lungimea
* Lățimea

A picture containing text, screenshot, number, font

Description automatically generated

Figura 3.2 Tabel de câmpuri pentru fiecare tip de control

Pentru vizualizarea controalelor, aplicația oferă utilizatorilor posibilitatea de a filtra informațiile în funcție de granița de interes sau în funcție de tipul de control (mașină/camion). Această funcționalitate este implementată prin intermediul hărții Google Maps, unde utilizatorii pot selecta o anumită zonă geografică prin intermediul interfeței grafice și aplicația va afișa doar controalele asociate acelei zone. Aceasta facilitează navigarea și identificarea rapidă a controalelor de interes. Informațiile despre controale sunt prezentate sub formă de tabel, având pe fiecare rând câte un control specific, iar fiecare coloană afișează detaliile relevante.

Introducerea, vizualizarea și filtrarea controalelor reprezintă un instrument eficient pentru gestionarea controalelor la frontierele României. Prin intermediul tabelelor și hărții, utilizatorii pot accesa și vizualiza rapid informațiile relevante, asigurând un flux eficient al activităților de control și o gestionare mai bună a fluxului de persoane și vehicule la frontiere.

### Gestionarea granițelor

Gestionarea granițelor în aplicația dezvoltată reprezintă o componentă esențială, având ca scop principal furnizarea unei modalități intuitive și eficiente de vizualizare și gestionare a informațiilor referitoare la acestea. Aplicația permite utilizatorilor să exploreze și să interacționeze cu granițele României într-un mode flexibil și cuprinzător.

Odată ce granițele au fost introduse în baza de date, acestea devin disponibile în

cadrul aplicației și pot fi vizualizate în cadrul hărții Google Maps. Utilizatorii pot naviga în hartă și pot observa marcaje specifice care indică amplasarea fiecărei granițe. Această reprezentare grafică a granițelor oferă o perspectivă clară și intuitivă asupra distribuției geografice a acestora. De asemenea, granițele pot fi filtrate în funcție de tipul de autovehicule care pot intra sau ieși din perimetrul țării. Prin intermediul acestei filtrări pe harta Google Maps vor fi afișate granițele specifice pentru mașini sau cele specifice pentru camioane.

A diagram of a web

Description automatically generated with low confidence

Figura 3.3 Diagrama de vizualizare a granițelor

## Concepte de bază ale aplicației

### Arhitectura REST

Arhitectura REST (Representational State Transfer), conform „Code Academy” [8] este un stil arhitectural utilizat în dezvoltarea aplicațiilor web care urmărește să ofere un set de principii și restricții pentru proiectarea și implementarea serviciilor web. Această abordare se bazează pe utilizarea protocolului HTTP pentru a realiza comunicarea între client și server.

În arhitectura REST, resursele sunt considerate entități identificabile și reprezentabile prin intermediul unui identificator unic (URI). Aceste resurse pot fi diverse entități, cum ar fi utilizatori, controale, granițe etc. Fiecare resursă poate avea una sau mai multe reprezentări, care pot fi obținute în diferite formate, cum ar fi JSON sau XML.

Comunicarea între client și server se realizează prin intermediul cererilor HTTP. Cererile HTTP sunt operațiuni standardizate, cum ar fi: GET (pentru obținerea unei resurse), POST (pentru crearea unei resurse), PUT (pentru actualizarea unei resurse) și DELETE (pentru ștergerea unei resurse). Aceste cereri sunt trimise către un URI specific și sunt însoțite de antete HTTP care conțin informații suplimentare despre cerere.

Serverul, care expune resursele, oferă răspunsuri la aceste cereri, care pot conține coduri de stare HTTP, cum ar fi:

* 200 (OK)
* 201 (Created)
* 404 (Not Found)

Răspunsurile pot conține, de asemenea, corpuri de răspuns care conțin datele cerute sau informații suplimentare.

Unul dintre principiile cheie ale arhitecturii REST este independența de stare. Aceasta înseamnă că fiecare cerere trebuie să conțină toate informațiile necesare pentru a fi procesată de server, fără a se baza pe starea anterioară a comunicării. Serverul nu păstrează informații despre starea clientului și fiecare cerere este tratată independent.

Arhitectura REST promovează utilizarea unei interfețe uniforme pentru a accesa și manipula resursele. Aceasta înseamnă că resursele sunt accesate prin intermediul URI-urilor, iar operațiunile comune, cum ar fi obținerea, crearea, actualizarea și ștergerea, sunt aplicate într-un mod consistent pe resurse folosind metode HTTP.

Prin adoptarea arhitecturii REST, aplicația web devine mai scalabilă și ușor de gestionat. Această abordare facilitează dezvoltarea de servicii web independente și reutilizabile, iar utilizarea protocolului HTTP oferă suport pentru securitate, cache-uri, autentificare și alte funcționalități.

A picture containing screenshot, diagram, text, design

Description automatically generated

Figura 3.4 Diagrama arhitecturii REST

### Arhitectura MVC

Arhitectura Model-View-Controller (MVC) este un model de proiectare software utilizat în dezvoltarea aplicațiilor pentru a separa logica de prezentare (interfața cu utilizatorul) de logica de afaceri și de gestionarea datelor. Această arhitectură împarte aplicația în trei componente principale:

* Modelul (Model)
* Vizualizarea ( View)
* Controlerul (Controller)

Modelul reprezintă componenta care se ocupă de gestionarea datelor și logica de afaceri a aplicației. Acesta este responsabil pentru stocarea, manipularea și procesarea datelor în conformitate cu regulile și restricțiile specifice domeniului de aplicație. Modelul poate include și alte funcționalități, cum ar fi accesul la baze de date sau servicii externe.

Vizualizarea reprezintă componenta care se ocupă de afișarea datelor și interacțiunea cu utilizatorul. Aceasta primește informațiile necesare din Model și le prezintă într-un format adecvat, cum ar fi o interfață grafică, o pagină web sau un document text. Vizualizarea nu conține logica de afaceri sau manipularea datelor, ci doar se concentrează pe prezentarea informațiilor într-un mod coerent și atractiv.

Controlerul reprezintă componenta care gestionează interacțiunea între Model și Vizualizare. Acesta primește acțiunile utilizatorului și decide cum să le proceseze. Controlerul accesează Modelul pentru a obține sau actualiza datele necesare și apoi actualizează Vizualizarea pentru a reflecta schimbările. Controlerul se asigură că logica de afaceri este aplicată în mod corespunzător și că datele sunt manipulate în conformitate cu regulile stabilite.

Arhitectura MVC promovează separarea clară a responsabilităților între aceste tei componente, ceea ce facilitează dezvoltarea, testarea și menținerea aplicațiilor. Prin separarea logicii de afaceri de prezentare, schimbările într-o componentă nu afectează celelalte, permițând dezvoltatorilor să lucreze independent și să facă modificări într-o manieră mai eficientă și mai sigură.

Utilizarea arhitecturii MVC în aplicații oferă numeroase beneficii, precum o mai bună organizare a codului, reutilizabilitate, extensibilitate și scalabilitate. Aceasta facilitează echipelor de dezvoltare să lucreze în paralel și să își împartă sarcinile în funcție de componente MVC. De asemenea, arhitectura MVC poate fi adaptată pentru diferite platforme și tehnologii, fiind larg utilizată în dezvoltarea de aplicații web, mobile și desktop.

A picture containing text, screenshot, diagram, design

Description automatically generated

Figura 3.5 Diagrama arhitecturii MVC

## Analiza riscurilor aplicației

Analiza riscurilor în cadrul unei aplicații este esențială pentru identificarea și gestionarea potențialelor amenințări și vulnerabilități care ar putea afecta securitatea și funcționalitatea acesteia. Pentru a gestiona aceste riscuri, este recomandat să se implementeze practici și politici de securitate, să se efectueze teste și monitorizare continuă a aplicației și să se dezvolte un plan de backup și recuperare în caz de urgență. De asemenea, este important să se mențină aplicația actualizată și să se urmărească evoluția și noile amenințări de securitate. În următoarele subcapitole vor fi prezentate și analizate riscurile aplicației.

A diagram of a company

Description automatically generated with low confidence

Figura 3.6 Diagrama riscurilor aplicației

### Riscul de securitate

Aplicația poate fi expusă la atacuri cibernetice, cum ar fi atacuri de tipul:

* SQL Injection
* Cross-site scripting (XSS)
* Denegare de serviciu (DDoS)

Aceste atacuri pot compromite confidențialitatea, integritatea și disponibilitatea datelor și pot afecta funcționalitatea aplicației. Este important să se implementeze măsuri de securitate adecvate, cum ar fi autentificare și autorizare robuste, validarea și filtrarea datelor de intrare, criptarea datelor sensibile și monitorizarea activității suspecte.

### Riscul de erori de programare

Aplicația poate conține erori de programare, cum ar fi:

* Bucle infinite
* Gestionarea necorespunzătoare a excepțiilor
* Vulnerabilități de securitate

Aceste erori pot duce la comportament incorect al aplicației, blocarea sau prăbușirea acesteia. Este important să se efectueze teste riguroase și să se implementeze practici de dezvoltare sigure pentru a minimiza riscul de erori de programare.

### Riscul de scalabilitate

O creștere a numărului de utilizatori și a volumului de date poate afecta performanța și scalabilitatea aplicației. Acesta poate duce la un timp de răspuns mai mare, blocarea sau prăbușirea aplicației. Este important să se planifice și să se implementeze o arhitectură scalabilă, utilizând cache-uri, optimizarea bazei de date și soluții de încărcare echilibrată pentru a gestiona creșterea traficului și volumului de date.

### Riscul de erori de integrare

Aplicația poate interacționa cu alte sisteme sau servicii externe, cum ar fi:

* Baze de date
* API-uri
* Servicii de autentificare

Erorile de integrare pot duce la eșecuri în transmiterea datelor, incompatibilitate între formatele de date sau pierderea datelor. Este important să se testeze și să se monitorizeze integrările cu alte sisteme și să se implementeze mecanisme de recuperare în caz de erori.

### Riscul de dependențe tehnologice

Aplicația poate depinde de:

* Biblioteci
* Framework-uri
* Servicii externe

Eșecurile sau problemele cu aceste dependențe pot afecta funcționalitatea aplicației. Este important să se efectueze o analiză și o evaluare atentă a dependențelor tehnologice și să se ia în considerare alternative sau soluții de rezervă în caz de eșecuri sau probleme.

# Proiectare

## Cazuri de utilizare

Cazurile de utilizare sunt utilizate în dezvoltarea software pentru a descrie interacțiunea între utilizatori și sistem. Aceasta se concentrează pe identificarea cerințelor funcționale ale sistemului, prin analiza scenariilor de utilizare și a acțiunilor pe care utilizatorii le efectuează în cadrul aplicației.

Un caz de utilizare reprezintă o secvență de acțiuni care descrie interacțiunea între un utilizator și sistem într-un anumit context. Aceasta descrie scopul utilizării sistemului și rezultatul așteptat de la acea interacțiune.

În subcapitolele următoare vor fi prezentate cazurile de utilizare pentru utilizatorii de tip administrator si utilizatorii membri. Diferența dintre aceste două tipuri de utilizatori este accesul la diferite funcționalități ale aplicației. Utilizatorul administrator, spre deosebire de utilizatorul membru poate introduce, vizualiza și șterge utilizatori. Aceste funcționalități sunt puse la dispoziție in cadrul aplicației și pot fi folosite strict de utilizatorii de tip administrator.

### Cazuri de utilizare membru

Principalele cazuri de utilizare sunt constituite din efectuarea următoarelor activități de către utilizator. În acest caz utilizatorul este de tip membru, reprezentând orice utilizator ce folosește aplicația.

* Autentificare în aplicație
* Introducerea controalelor de camioane
* Introducerea controalelor de mașini
* Vizualizarea controalelor de camioane
* Vizualizarea controalelor de camioane filtrate în funcție de graniță
* Vizualizarea controalelor de mașini
* Vizualizarea controalelor de mașini filtrate în funcție de graniță
* Vizualizare granițe

A diagram of a person with text

Description automatically generated with low confidence

Figura 4.1 Diagrama cazurilor pentru utilizator membru

În continuare, va fi prezentat un caz de utilizare în detaliu pentru un utilizator de tip membru, și anume introducerea unui control de mașină.

Nume caz de utilizare: Introducere control de mașini

Actor: Utilizator membru

Descriere: După autentificare, utilizatorul poate introduce în baza de date un control pentru un autovehicul de tip mașină, prin intermediul aplicației.

Succesiunea evenimentelor:

* Utilizatorul autentificat navighează pe pagina de introducere a controalelor pentru mașini.
* Completează formularul afișat cu datele relevante controlului.
* După apăsarea butonului “Submit”, se apelează o cerere HTTP de tip POST, care conține un body cu datele introduse de utilizator sub format JSON.
* Dacă serverul returnează un status de tip 200 (OK), utilizatorul va fi alertat printr-o căsuță de dialog că controlul a fost introdus cu succes.
* După introducerea cu succes a controlului, utilizatorul va fi redirecționat pe pagina principală a aplicației, adică pagina Dashboard.

### Cazuri de utilizare administrator

În afară de funcționalitățile de care dispune utilizatorul de tip membru, utilizatorul de tip administrator mai are acces la câteva funcționalități adiționale. Acestea sunt:

* Înregistrarea utilizatorilor
* Vizualizarea utilizatorilor
* Ștergerea utilizatorilor

În continuare, va fi prezentat în detaliu un caz de utilizare specific utilizatorilor de tip administrator, acesta fiind vizualizarea utilizatorilor.

Nume caz de utilizare: Vizualizarea utilizatorilor

Actor: Utilizator administrator

Descriere: Fiind autentificat cu un cont de administrator, utilizatorul are opțiunea de a vizualiza, sub forma de tabel, toți utilizatorii aplicației.

Succesiunea evenimentelor:

* Administratorul navighează pe pagina destinată vizualizării utilizatorilor.
* La momentul accesării acestei pagini, se creează o cerere HTTP de tip GET către server.
* Dacă serverul returnează un status de tip 200 (OK), pe pagină vor fi afișate datele relevante ale utilizatorilor înregistrați în aplicație. Acestea fiind numele, emailul și codul unic de angajat (Badge Number).

A picture containing text, diagram, pattern

Description automatically generated

Figura 4.2 Diagrama cazurilor pentru un utilizator administrator

## Diagrame de secvență

Diagrama de secvență este o tehnică de modelare grafică utilizată în ingineria software pentru a reprezenta interacțiunile între obiecte sau entități într-un sistem sau proces. Această diagramă prezintă secvența de mesaje sau acțiuni care au loc între obiecte și ordinea în care acestea se execută. Este un instrument util pentru a înțelege și comunica modul în care obiectele interacționează în cadrul unui scenariu din cadrul aplicației.

Diagramele secvențiale sunt construite in jurul conceptului de linie de timp, care indică evoluția evenimentelor în timp. Obiectele sau entitățile sunt reprezentate ca blocuri de activitate pe linia de timp și sunt interconectate prin săgeți, care indică mesajele sau acțiunile care sunt trimise între ele. Săgețile pot fi direcționale pentru a indica fluxul de execuție al mesajelor.

Pe diagrama secvențială sunt prezentate și mesajele primite și transmise între obiecte. Aceste mesaje pot fi sincrone sau asincrone și pot conține informații despre parametrii mesajului și rezultatele returnate. De asemenea, pot fi prezentați și factorii de decizie sau condiționali, care influențează fluxul de execuție al interacțiunii.

În continuare, cu ajutorul diagramelor de secvență vor fi prezentate două dintre cele mai importante funcționalități, acestea fiind:

* Autentificarea utilizatorilor
* Înregistrarea utilizatorilor

### Diagrama de secvență a autentificării utilizatorilor

A picture containing text, diagram, receipt, parallel

Description automatically generated

Figura 4.3 Diagrama de secvență autentificare

În diagrama prezentată mai sus se poate observa logica si fluxul de desfășurare a autentificării din cadrul aplicației, prin comunicarea dintre serviciile relevante implicate în această acțiune. Un rezumat al pașilor ilustrați in diagramă ar fi:

* Utilizatorul completează formularul de autentificare.
* Dacă formularul nu este validat, o eroare va fi afișată.
* Dacă formularul este valid, se va face o cerere de autentificare care conține datele din formular către server.
* Serverul trimite o interogare către baza de date.
* Dacă există erori de existență sau corectitudine, se va afișa o eroare către utilizator
* Dacă nu există erori, se va crea un token de acces unic, care conține datele utilizatorului prin intermediul serviciului JSON Web Token (JWT).
* Tokenul va fi stocat în localstorage.
* Utilizatorul va fi alertat printr-o căsuță de dialog despre autentificarea reușită.
* Utilizatorul va fi redirecționat către pagina principală a aplicației.

### Diagrama de secvență a înregistrării unui utilizator

A picture containing text, diagram, plan, parallel

Description automatically generated

Figura 4.4 Diagramă de secvență înregistrare

În diagrama de mai sus este prezentată comunicarea dintre serviciile necesare înregistrării unui nou utilizator în cadrul aplicației și fluxul datelor implicate. O scurtă prezentare a diagramei ilustrate este:

* Utilizatorul completează formularul de înregistrare.
* Dacă nu este completat corect, va fi afișată o eroare.
* Dacă este completat corect, se va face o cerere către server.
* Serverul va face la rândul lui o cerere către baza de date.
* Dacă datele sunt invalide, o eroare va fi afișată.
* Dacă datele sunt valide, vor fi adăugate în baza de date.
* Utilizatorul va fi informat că a fost înregistrat cu succes și va fi redirecționat către pagina de Login.

## Integrarea serviciilor externe în aplicație (Google Maps)

Integrarea serviciului Google Maps într-o aplicație prezintă numeroase avantaje, oferind o serie de funcționalități și valoare adăugată pentru utilizatori. În cazul aplicației dezvoltate pentru lucrarea de licență, API-ul Google Maps ușurează interacțiunea utilizatorului cu interfața grafică a aplicației prin afișarea geolocației fiecărei granițe de la frontierele României. Acest aspect îi oferă posibilitatea utilizatorului să navigheze printre toate punctele de interes într-un mod eficient și intuitiv.

Fiecare graniță este reprezentată de către un marker, care este dispus la latitudinea și longitudinea corespunzătoare fiecărei granițe din România. În interiorul fiecărui marker se află numele graniței respective și două butoane care redirecționează utilizatorul către vizualizarea controalelor de mașini, respectiv camioane, executate la frontiera respectivă.

În figura 4.5 se poate observa conținutul unui marker din interiorul aplicației.

A screenshot of a phone

Description automatically generated with medium confidence

Figura 4.5 Detaliile afișate pe un marker Google Maps

Din punct de vedere tehnic, integrarea în aplicație a serviciului Google Maps este realizată într-un mod rapid și eficient, fiind nevoie doar de apelarea unei funcții predefinite din librăria Google Maps, care conține un parametru obligatoriu, și anume cheia API-ul (API Key). Această cheie este un parametru unic pentru fiecare utilizator al serviciului care este generată la momentul inițializării unui nou proiect.

## Structura bazei de date

Capitolul următor se concentrează asupra bazei de date a aplicației, care reprezintă un element esențial în gestionarea controalelor la granițele României. O bază de date bine concepută și eficientă joacă un rol crucial în stocarea, gestionarea și accesarea informațiilor necesare pentru funcționarea corectă a aplicației.

Scopul acestui capitol este de a analiza în detaliu structura și funcționalitatea bazei de date, precum și modul în care aceasta este utilizată în cadrul aplicației noastre. Vom explora diferitele entități și relații, precum și modalitățile de organizare a datelor pentru a asigura performanța și integritatea acestora.

### Alegerea bazei de date

Alegerea unei baze de date adecvate pentru aplicația dezvoltată reprezintă un aspect esențial în dezvoltarea și funcționarea corectă a sistemului. O bază de date bine aleasă poate contribui la performanța, scalabilitatea și securitatea aplicației, oferind o soluție eficientă pentru stocarea și gestionarea datelor. Pentru aplicația din cadrul lucrării de licență, am ales să folosesc baza de date MongoDB.

MongoDB este o bază de date document-orientată, non-relațională, care se bazează pe conceptul de documente flexibile și dinamice. A fost dezvoltată pentru a oferi scalabilitate și performanță în aplicații moderne, care manipulează volume mari de date și necesită flexibilitate în structura datelor.

Un aspect distinctiv al MongoDB este modelul său de date bazat pe documente JSON-like, numite colecții. Un document reprezintă o entitate independentă și conține perechi cheie-valoare, unde valorile pot fi de tipuri variate, cum ar fi string-uri, numere, obiecte, tablouri sau chiar alte documente. Această flexibilitate permite adaptarea structurii datelor în timp, fără a necesita modificări de schema.

Arhitectura MongoDB este distribuită și se bazează pe un set de servere care colaborează pentru a asigura redundanță și scalabilitate. Replicarea permite replicarea datelor între servere, asigurând astfel o disponibilitate ridicată și rezistență la eșecuri. De asemenea, funcționalitatea de partajare permite distribuirea datelor în mai multe noduri, pentru a gestiona volumul mare de date și a crește performanța.

Pentru securitate, MongoDB oferă autentificare și autorizare bazate pe roluri, permisii fine-grained și criptarea datelor în tranzit și în repaus. Administrarea bazei de

date este ușurată prin intermediul instrumentelor de gestionare grafică și linie de comandă, care facilitează crearea, modificarea și monitorizarea bazei de date.

Un alt avantaj al MongoDB este integrarea sa cu limbajul de programare JavaScript, ceea ce facilitează dezvoltarea aplicațiilor complete, de la nivelul bazei de date până la interfața utilizator. API-urile MongoDB sunt disponibile pentru diferite limbaje de programare, oferind dezvoltatorilor o interacțiune convenabilă cu baza de date.

### Structura unei colecții în baza de date

În acest subcapitol va fi prezentată structura unei colecții a bazei de date. În figura de mai jos este prezentată colecția bordersData, care conține toate datele relevante despre o graniță. Aceste date sunt structurate sub forma cheie-valoare, unde cheile sunt:

* \_id
* name
* latitude
* longitude
* waitTime
* areCarsAllowed
* areTrucksAllowed

A picture containing text, screenshot, font, number

Description automatically generated

Figura 4.6 Diagrama unei colecții

Un exemplu de document din colecția bordersData poate fi vizualizat în figura de mai jos. În acest document sunt introduse datele despre granița de la Albita.

A screenshot of a computer code

Description automatically generated with low confidence

Figura 4.7 Document din colecția bordersData

## Designul interfeței utilizatorului

Designul interfeței utilizatorului (UI) reprezintă un aspect esențial al dezvoltării aplicațiilor software, concentrându-se pe modul în care utilizatorii interacționează cu produsul. Interfața utilizatorului este puntea de legătură între utilizator și funcționalitățile aplicației, având un impact semnificativ asupra experienței utilizatorului și succesului aplicației.

Designul unei interfețe utilizatorului eficiente și plăcute implică o serie de aspecte, cum ar fi arhitectura informației, structura și organizarea elementelor, utilizarea culorilor, fonturilor și a elementelor vizuale, navigarea și fluxul interacțiunilor. Scopul principal al unui design de interfață reușit este de a facilita utilizatorului să realizeze sarcinile dorite într-un mod intuitiv, eficient și plăcut.

Designul interfeței a fost împărțit în două mari componente, și anume:

* Bara de navigație: Această componentă conține elementele necesare pentru navigarea în interiorul aplicației, cât și informații relevante pentru utilizator, cum ar fi: numele aplicației, logo-ul, butoanele pentru navigarea pe diferitele pagini ale aplicației și numele utilizatorului. Aceste elemente sunt vizibile doar după autentificarea utilizatorului în aplicație, din motive de securitate a datelor.
* Conținutul: Această componentă prezintă conținutul specific paginii pe care se află.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 4.8 Infrastructura Interfeței

Pentru a avea un efect plăcut asupra utilizatorului, au fost alese două culori care să creeze un contrast între elementele paginilor. Folosind doar două culori, acestea fiind nuanțe ale culorilor turcoaz și vișiniu, interfața oferă un aspect curat si simplist. În figura de mai jos se poate observa pagina Dashboard a aplicației, care conține toate elementele de design folosite în interiorul aplicației.

A picture containing map, screenshot, text, software

Description automatically generated

Figura 4.9 Interfața paginii Dashboard

# Implementare

## Prezentarea componentelor principale

Componentele reprezintă mijloacele principale de implementare a unei aplicații, în cazul de față platforma este compusă din două componente principale, și anume:

* Client
* Server

În următoarele capitole vor fi prezentate cele două componente în detaliu, acestea fiind la rândul lor formate din alte componente pentru a crea o structura ușor de extins și de modificat pe viitor.

### Client

Componenta client reprezintă partea aplicației cu care interacționează utilizatorul în mod direct. Rolul clientului este de a furniza o interfață utilizator intuitivă și prietenoasă, astfel încât utilizatorii să aibă acces și să utilizeze funcționalitățile aplicației într-un mod eficient și convenabil.

Implementarea componentei client este realizată prin utilizarea unor tehnologii de actualitate, cum ar fi JavaScript, HTML și SCSS, acestea fiind înglobate prin folosirea librăriei ReactJS.

În figura de mai jos se poate observa structura componentei client, aceasta fiind bazată pe fișierele pages și components. Platforma conține 11 pagini prin care utilizatorul poate naviga, acestea conținând date specifice și relevante. Paginile au fost construite cu ajutorul componentelor și sunt conectate între ele folosind rutarea realizată de React Router DOM.

A screenshot of a black screen

Description automatically generated with low confidence

Figura 5.1 Structura componentei Client

Componentele care facilitează structurarea și stilizarea paginilor sunt:

* Navbar: Bara de navigare a aplicației reprezintă o componentă grafică, care conține un set de elemente de navigare, cum ar fi butoane, meniuri și iconițe, care facilitează accesul rapid la diferite secțiuni sau funcționalități ale aplicației. În figura de mai jos se poate observa implementarea componentei de navigație. Această componentă se regăsește pe toate paginile din interiorul aplicației, acest lucru fiind necesar pentru a genera o interfață eficientă si prietenoasă pentru utilizator.

A screen shot of a computer program

Description automatically generated with medium confidence

Figura 5.2 Componenta Navbar

* Tabel: Tabelul reprezintă o structură grafică formată din rânduri și coloane, utilizată pentru organizarea și afișarea datelor într-o formă tabulară. Acesta este utilizat pentru a prezenta și gestiona date structurate într-un format clar și ușor de înțeles pentru utilizatori. Această componentă se regăsește pe paginile care se ocupă cu vizualizarea datelor, cum ar fi vizualizarea controalelor de mașini, camioane sau vizualizarea utilizatorilor înregistrați în aplicație. În figura de mai jos se poate observa structura componentei Table.

A screen shot of a computer program

Description automatically generated with low confidence

Figura 5.3 Componenta Tabel

* Butoane: Butoanele reprezintă un element interactiv vizual, care permite utilizatorilor să inițieze anumite acțiuni sau să interacționeze cu funcționalitățile aplicației. Acestea sunt utilizate pentru a crea o interfață intuitivă și prietenoasă, permițând utilizatorilor să realizeze acțiuni specifice cu un singur click. Butoanele au diferite aspecte și stiluri, adaptate designului și temei vizuale al aplicației. Acestea se regăsesc în majoritatea paginilor din interiorul aplicației. Un exemplu de implementare al unui buton se regăsește în figura de mai jos.

A picture containing text, screenshot, font

Description automatically generated

Figura 5.4 Exemplu implementare Buton

* Formulare: Formularele reprezintă o componentă interactivă prin intermediul căreia utilizatorii pot introduce și trimite date către server. Formularele sunt utilizate pentru a colecta informații sau pentru a permite utilizatorilor să efectueze anumite acțiuni specifice, cum ar fi înregistrarea, autentificarea sau introducerea unui control. Un formular este format dintr-o combinație de elemente precum câmpuri de introducere, opțiuni de selecție, butoane de acțiune și alte elemente interactive. Aceste elemente permit utilizatorului să completeze și să valideze informațiile necesare într-un mod structurat și organizat.

În continuare vor fi prezentate câteva pagini din interiorul aplicației, pentru a explora felul în care componentele au fost folosite împreuna pentru a crea o interfață de comunicare optimă și eficientă pentru utilizator.

#### Pagina de vizualizare a controalelor de mașini

A screenshot of a computer

Description automatically generated with medium confidence

Figura 5.5 Pagina de vizualizare a controalelor de mașini

În figura de mai sus este prezentată pagina All Car Controls, în care utilizatorul are acces la informațiile legate de toate controalele efectuate asupra mașinilor la granițele țării. Pagina conține componenta de navigare (Navbar), și o componentă de tip tabel, care facilitează afișarea datelor relevante.

Datele afișate au fost apelate cu ajutorul unei cereri HTTP de tip GET, care a fost construită pentru a aduce toate informațiile despre controalele de mașini din colecția carControlsData din interiorul baze de date.

Stilizarea paginii a fost realizată cu ajutorul tehnologiei SCSS, care facilitează stilizarea componentelor din cadrul paginii.

#### Pagina de introducere a unui control de mașini

A screenshot of a computer

Description automatically generated with medium confidence

Figura 5.6 Pagina de introducere a unui control de mașini

În figura de mai sus este prezentată pagina Car Control, în care utilizatorul are posibilitatea de a introduce datele necesare unui control pentru mașini, prin intermediul unui formular. Pagina este formată din componenta de navigare (Navbar) și un formular, care are ca rol ușurarea și eficientizarea introducerii unui control.

Informațiile introduse în cadrul formularului sunt procesate cu ajutorul unui endpoint HTTP, de tipul POST, care are ca scop transmiterea datelor și stocarea acestora în colecția carControlsData din interiorul bazei de date.

### Server

Componenta server din aplicație reprezintă partea care gestionează cererile și furnizează răspunsuri către client. Serverul este responsabil pentru procesarea și manipularea datelor, gestionarea logicii de afaceri și comunicarea cu alte componente sau servicii externe. Componenta server a fost implementată cu ajutorul framework-ului NodeJS. În figura de mai jos se poate observa organizarea componentei server.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 5.7 Structura componentei server

După cum se poate observa și în figura de mai sus, componenta server este compusă din două mari elemente, și anume:

* Fișierul models:

Acest fișier conține structura de implementare a colecțiilor din interiorul bazei de date. În primul rând, se vor importa modulele necesare pentru a interacționa cu baza de date și a defini structura modelelor.

Fiecare model va avea propriile sale proprietăți, care vor reflecta structura datelor stocate în colecție. În figura de mai jos se poate observa implementarea modelului colecției truckControlsData.

A screen shot of a computer program

Description automatically generated with low confidence

Figura 5.8 Modelul colecției truckControlsData

* Fișierul index:

Acest fișier poate fi considerat punctul central al serviciilor oferite de aplicație. Acest fișier conține rutele și logica asociată pentru crearea, citirea, actualizarea și ștergerea informațiilor din baza de date.

În interiorul indexului sunt definite rutele pentru diferitele operații CRUD (Create, Read, Update, Delete). Fiecare rută are un URL asociat și va fi legată de un anumit tip de cerere HTTP ( GET, POST, PUT, DELETE). Aceste rute, în esență, specifică ce acțiuni pot fi efectuate asupra datelor din baza de date.

În figura de mai jos este prezentat un API din cadrul fișierului index, care se ocupă cu stocarea datelor unui utilizator nou înregistrat în baza de date. Se poate observa că parola introdusă de utilizator trece printr-un proces de criptarea, pentru a spori securitatea datelor din interiorul aplicației. Mai apoi, datele vor fi stocate în baza de date cu ajutorul unei cereri HTTP de tip POST.

A picture containing text, screenshot, software, font

Description automatically generated

Figura 5.9 API pentru înregistrarea unui utilizator

## Testare

Testarea este o parte esențială a dezvoltării unei aplicații, deoarece are ca scop verificarea și validarea funcționalității acesteia înainte de a fi pusă în producție. Testarea asigură că aplicația funcționează corect, este fiabilă și îndeplinește cerințele și așteptările utilizatorilor.

În cadrul aplicației, am folosit diferite instrumente și metode pentru testarea acesteia. În capitolele următoare vor fi prezentate exemple de testare atât pe partea de server, cât și pe partea de client.

### Testarea serverului

Pentru a asigura buna funcționalitate și fiabilitatea componentei server a aplicației, am implementat o serie de teste folosind aplicația Postman, care se ocupă cu simularea de cereri HTTP.

Fiecare cerere în parte testează o anumită funcționalitate a aplicației, așadar în imaginea de mai jos este ilustrat un tip de cerere, de tip POST, care execută introducerea unui document în baza de date, acesta conținând datele relevante despre o graniță a României.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 5.10 Testarea unei funcționalități cu ajutorul Postman

În figura de mai sus se poate observa mesajul afișat după executarea cererii, acesta reflectând reușita introducerii unei granițe în baza de date.

### Testarea clientului

Testarea componentei client din cadrul aplicației a fost implementată folosind conceptul de testare manuală.

Conceptul de testare manuală a clientului implică efectuarea de teste în modul interactiv, introducând diferite variabile în câmpurile formularelor și acționând butoanele pentru a observa reacția aplicației. Acest tip de testare se concentrează pe evaluarea interfeței utilizatorului, funcționalităților vizibile și experienței generale a utilizatorului în timpul interacțiunii cu aplicația.

Un exemplu de test manual executat asupra aplicației ar fi testarea formularului de autentificare a utilizatorului. În figura de mai jos se poate observa simularea unei încercări de autentificare, prin completarea formularului cu informații greșite pentru a vedea comportamentul interfeței.

A screenshot of a login screen

Description automatically generated with medium confidence

Figura 5.11 Testarea componentei client

În imaginea de mai sus se poate observa comportamentul optim al aplicației în acest caz, afișând o căsuță de dialog care atenționează utilizatorul asupra credențialelor introduse.

# Concluzii

## Rezultate obținute

În urma dezvoltării aplicației descrise anterior, am obținut rezultatele pozitive și funcționalități esențiale pentru gestionarea controalelor la frontierele României. Câteva dintre rezultatele notabile obținute sunt:

* Simplificarea controalelor la frontiere: Prin intermediul aplicației, utilizatorii pot introduce informații despre controalele desfășurate la granițele României. Aceste informații includ locația, tipul controlului, data efectuării, și alte detalii relevante. Prin intermediul unei interfețe intuitive, utilizatorii pot completa formulare și trimite informațiile către baza de date, astfel centralizând și simplificând accesul acestor informații.
* Vizualizarea granițelor pe hartă: Prin integrarea serviciului Google Maps în aplicație, utilizatorii au posibilitatea de a vizualiza granițele și de a accesa detalii despre controalele executate la granițele respective într-o manieră interactivă.
* Filtrarea controalelor: Utilizatorii pot alege să afișeze doar controalele pentru camioane, sau doar controalele pentru mașini, selectând opțiunile corespunzătoare în cadrul interfeței. Aceasta permite utilizatorilor să se concentreze doar pe informațiile relevante pentru nevoile lor și să obțină o imagine mai precisă a situației la frontiere.

În concluzie, dezvoltarea aplicației pentru gestionarea controalelor la granițele României a adus rezultate semnificative și funcționalități esențiale. Utilizatorii pot introduce, vizualiza și gestiona informațiile despre controalele la frontiere într-un mod intuitiv și eficient. Această aplicație contribuie la optimizarea și eficientizarea procesului de gestionare a controalelor și poate fi adaptată și extinsă în funcție de nevoile specifice ale utilizatorilor și contextului operațional.

## Direcții de dezvoltare

Există mai multe direcții posibile de dezvoltare pentru viitoare versiuni ale aplicației de gestionare a controalelor la granițele României, cum ar fi:

* Extinderea funcționalităților: Aplicația poate fi extinsă prin adăugarea de funcționalități suplimentare. De exemplu, se pot include opțiuni pentru generarea automată a rapoartelor, notificări în timp real despre controalele noi sau actualizate, sau posibilitatea de a adăuga comentarii sau evaluări.
* Integrarea cu sisteme externe: Aplicația poate fi integrată cu alte sisteme sau servicii relevante. De exemplu, se poate realiza o integrare cu servicii de monitorizare a traficului sau cu platforme de comunicare pentru a permite utilizatorilor să primească actualizări în timp real despre evenimente

relevante din cadrul granițelor. Integrarea cu alte sisteme va spori capacitatea aplicației de a furniza informații în timp util și de a se adapta la schimbările din mediul înconjurător.

* Optimizarea performanței și scalabilității: Pe măsură ce numărul de utilizatori și volumul de date cresc, este important să se optimizeze performanța și scalabilitatea aplicației. Aceasta poate include optimizarea interogărilor la baza de date, implementarea cache-ului pentru a reduce timpul de răspuns, utilizarea tehnologiilor de distribuire a sarcinilor pentru a gestiona traficul sporit și scalarea infrastructurii pentru a face față cererilor crescute.

# Bibliografie

|  |  |
| --- | --- |
| [1] | „Regulamentul de aplicare a Codului vamal al României din 07.06.2006,” [Interactiv]. Available: https://legislatie.just.ro/Public/DetaliiDocument/84287. |
| [2] | „Introduction to Node.js,” [Interactiv]. Available: https://nodejs.dev/en/learn/. |
| [3] | „npm Docs,” [Interactiv]. Available: https://docs.npmjs.com/about-npm. |
| [4] | „React,” [Interactiv]. Available: https://react.dev/learn. |
| [5] | „Google Maps Platform,” [Interactiv]. Available: https://developers.google.com/maps/documentation. |
| [6] | „The Modern JavaScript Tutorial,” [Interactiv]. Available: https://javascript.info/. |
| [7] | „Poliția de Frontieră,” [Interactiv]. Available: https://www.politiadefrontiera.ro/ro/traficonline/?vt=1&vw=1. |
| [8] | „What is REST?,” [Interactiv]. Available: https://www.codecademy.com/article/what-is-rest. |