Documentație Inițială - Proiect RoadAlert

Echipa: Cosmin Croitoriu, Mario Parfinescu, Marius Ciochina

Tema: Aplicație de monitorizare trafic și raportare incidente (Crowdsourced Navigation)

Motivație: Deși există soluții comerciale precum Waze, acestea sunt closed-source și colectează date extensive ale utilizatorilor. Proiectul RoadAlert oferă o alternativă open-source, care are ca scop să construiască o platformă transparentă, personalizabilă pentru oricine.

Obiective: Dezvoltarea unei aplicații web crowdsourced care permite utilizatorilor să raporteze și să valideze incidente de trafic pe o hartă interactivă, creând astfel o comunitate colaborativă pentru siguranță rutieră îmbunătățită și navigare mai eficientă.

1. Structurarea Datelor

Vom utiliza o arhitectură de date hibridă pentru a maximiza performanța afișării pe hartă și securitatea datelor utilizatorilor.

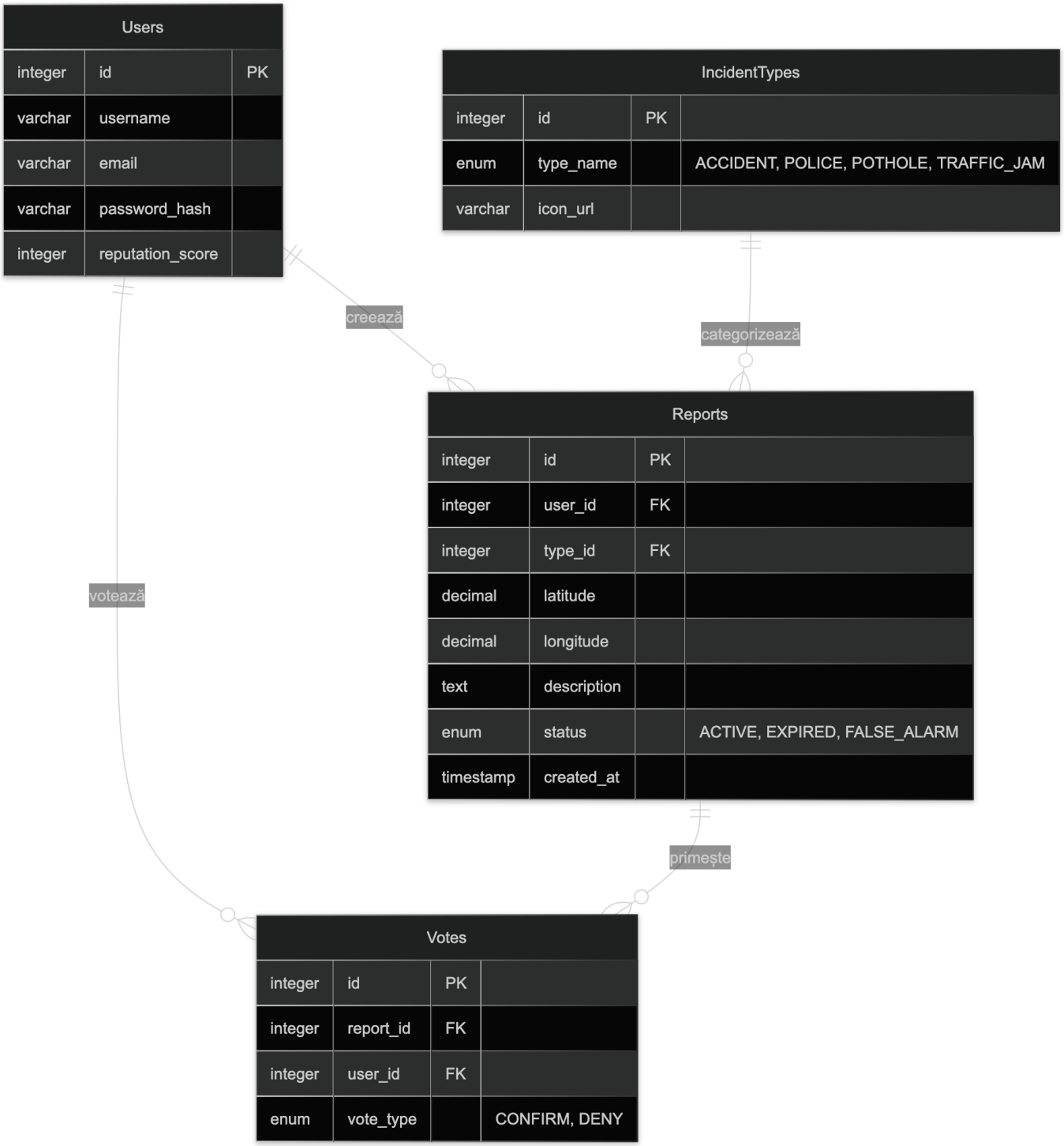
1.1. Tipul de stocare

- **PostgreSQL (Bază de date Relațională):** Va fi folosită pentru a stoca utilizatorii, istoricul raportărilor, sistemul de voturi (confirmare/infirmare eveniment), comentarii și coordonatele incidentelor (Lat/Long). Datele geografice vor fi gestionate prin câmpuri de tip POINT sau coloane separate pentru latitudine și longitudine.
- **Google Maps JavaScript API:** Va fi folosită pentru vizualizarea hărții, randarea markers pentru incidente și gestionarea interacțiunilor utilizatorului pe hartă. Acest lucru permite randarea rapidă și fluidă a punctelor pe hartă folosind infrastructura Google.

1.2. Schema Bazei de Date (Tabele și Relații)

Baza de date relațională (PostgreSQL) va conține următoarele tabele:

Diagrama Entitate-Relație



1. Users (Utilizatori)

- **id** (PK - Integer): Identificator unic
- **username** (Varchar): Numele afișat pe hartă
- **email** (Varchar): Pentru autentificare
- **password_hash** (Varchar): Parola criptată (bcrypt)
- **reputation_score** (Integer): Scorul de încredere (crește când alții confirmă raportările)

2. IncidentTypes (Tipuri Incidente)

- **id** (PK - Integer): ID tip
- **type_name** (Enum): `'ACCIDENT'`, `'POLICE'`, `'POTHOLE'`, `'TRAFFIC_JAM'`

- **icon_url** (Varchar): Link către iconița specifică (folosită în Google Maps Markers)

3. Reports (Raportări)

- **id** (PK - Integer): Identificator unic al raportului
- **user_id** (FK): Cine a raportat
- **type_id** (FK): Ce fel de incident este
- **latitude** (Decimal): Latitudinea incidentului
- **longitude** (Decimal): Longitudinea incidentului
- **description** (Text): Detalii opționale (ex: "Groapă mare pe banda 1")
- **status** (Enum): `'ACTIVE'`, `'EXPIRED'`, `'FALSE_ALARM'`
- **created_at** (Timestamp): Ora raportării

4. Votes (Validări)

- **id** (PK): ID unic
- **report_id** (FK): Raportul votat
- **user_id** (FK): Cine a votat
- **vote_type** (Enum): `'CONFIRM'` (Există), `'DENY'` (Nu mai există)

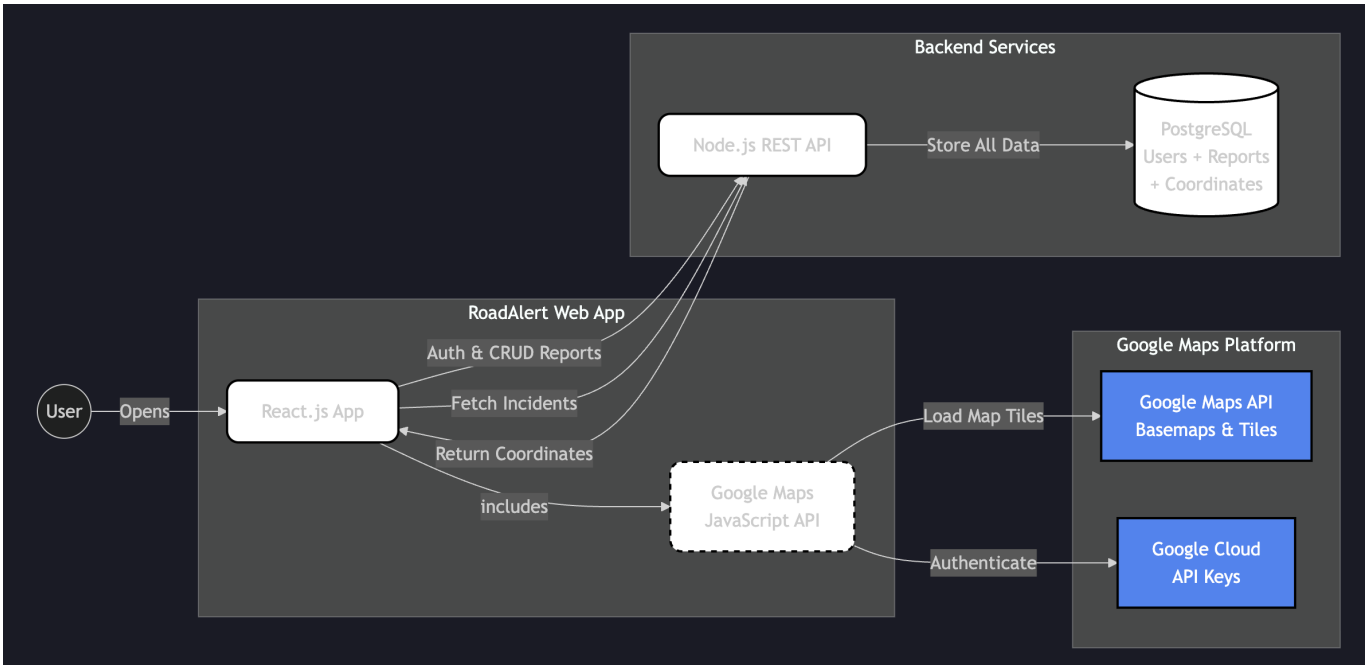
Logică: Dacă un raport primește 5 voturi de `'DENY'`, statusul devine `'FALSE_ALARM'` și dispare de pe hartă.

---

2. Arhitectura Generală a Aplicației

Arhitectura este distribuită, separând interfața vizuală de logica de validare a datelor.

Diagrama Arhitecturii



2.1. Componente Principale

- **Frontend (Web App):** Aplicație React.js care încarcă harta de bază și suprapune incidentele. Permite utilizatorului să pună un "pin" pe hartă.
- **Backend (API Server):** Node.js + Express. Primește datele de la Frontend, validează utilizatorul și decide dacă un incident trebuie șters sau actualizat. Gestionează toate operațiunile CRUD pentru incidente și stochează coordonatele în PostgreSQL.
- **Google Maps Services:**
  - **Maps JavaScript API:** Randarea hărții interactive în browser
  - **Geocoding API:** Convertirea adreselor în coordonate (opțional)
  - **Places API:** Informații suplimentare despre locații (opțional)

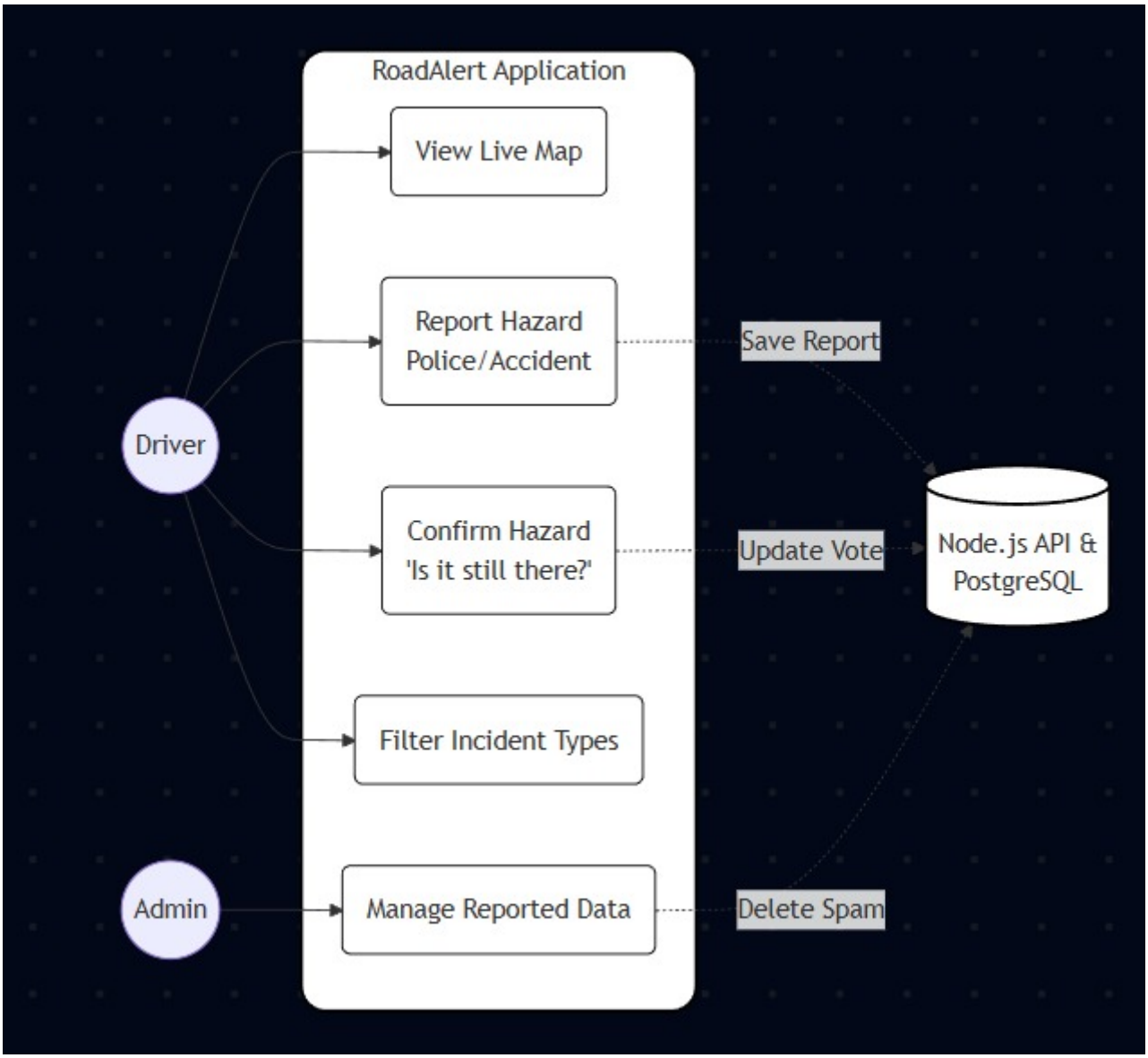
2.2. Fluxul de Date (Exemplu: Raportare Accident)

1. Utilizatorul apasă pe butonul "Raportează" în Frontend
2. Click pe hartă → Frontend-ul preia coordonatele [Lat, Long] prin Google Maps API
3. Frontend-ul trimite datele (coordonate + tip incident) la Backend (Node.js)
4. Backend-ul validează datele și salvează raportul în PostgreSQL (inclusiv latitudine și longitudine)
5. Backend-ul returnează confirmarea către Frontend
6. Frontend-ul adaugă un nou Marker pe Google Maps și notifică ceilalți utilizatori prin WebSocket/Polling

3. Diagrama Cazurilor de Utilizare (Use Cases)

Actorii principali sunt: **Șoferul (Utilizatorul)** și **Moderatorul (Admin)**.

Diagrama Cazurilor de Utilizare



3.1. Descrierea funcționalităților

Actor: Șofer (Driver)

- **Vizualizare Hartă (View Map):** Vede propria locație și incidentele din jur
- **Raportare Incident (Report Hazard):** Adaugă un marker nou (Poliție, Accident, Lucrări)
- **Confirmare Incident (Vote on Report):** Când trece pe lângă un incident raportat de altcineva, poate apăsa "Încă este acolo" (Like) sau "Nu este" (Dislike)
- **Filtrare (Filter Map):** Poate alege să vadă doar Poliție sau doar Accidente

Actor: Moderator

- **Ștergere Rapoarte:** Poate elimina manual rapoartele spam/vulgare
- **Gestionare Categorii:** Poate adăuga tipuri noi de alerte (ex: "Ceață")

---

4. Tehnologii

4.1. Tehnologii Generale

- **Limbaj de programare:** JavaScript / TypeScript
- **Frontend Framework:** React.js (pentru interfață rapidă tip Single Page App)
- **Backend Framework:** Node.js (pentru gestionarea API-urilor asincrone)
- **Bază de date:** PostgreSQL (stocare persistentă)

4.2. Tehnologii Specifice GIS

Google Maps JavaScript API:

- **Map:** Pentru randarea hărții interactive 2D
- **Geolocation API:** Pentru a găsi locația GPS a utilizatorului în browser
- **Marker:** Pentru a plasa și gestiona punctele de interes (incidente) pe hartă
- **InfoWindow:** Pentru a afișa informații despre incidente la click pe marker
- **MarkerClusterer:** Pentru gruparea marker-elor când sunt prea multe într-o zonă
- **Custom Overlays:** Pentru a desena temporar punctul înainte de confirmarea salvării

Google Cloud Platform:

- **API Key Management:** Securizarea și gestionarea cheilor API
- **Maps JavaScript API:** Hărți de bază cu styling personalizat
- **Custom Icons:** Configurarea iconițelor diferite pentru fiecare tip de incident (Poliție, Accident, Groapi, etc.)

---

6. Organizare Activități (4 Săptămâni)

6.1. Repartizarea Sarcinilor

Student 1 (Frontend & UX)

- **T1.1:** Configurare React și integrare harta full-screen
- **T1.2:** Implementare buton "Locația Mea" (Geolocation API)
- **T1.3:** Creare meniu plutitor (Floating Action Button) pentru tipurile de incidente
- **T1.4:** Implementare ferestre de tip Pop-up la click pe incident (arată cine a postat și ora)

Student 2 (Backend & Database)

- **T2.1:** Structurare bază de date PostgreSQL (tabele Users, Votes)
- **T2.2:** API pentru Login/Register (JWT)
- **T2.3:** Logică de "Trust Score" (dacă un user are multe rapoarte false, este blocat)
- **T2.4:** Job automat (Cron Job) care șterge incidentele mai vechi de 2 ore

Student 3 (Maps & GIS Integrator)

- **T3.1:** Configurare Google Maps API și obținere API Key din Google Cloud Console
- **T3.2:** Implementare funcții de adăugare/ștergere Markers prin Google Maps JavaScript API
- **T3.3:** Configurare iconițe custom (imagini SVG/PNG pentru poliție/gropi/accidente)
- **T3.4:** Implementare filtrare vizuală (ascundere/afișare markers după tip la cerere)
- **T3.5:** Implementare MarkerClusterer pentru gruparea incidentelor

6.2. Planificare Temporală (Timeline)

Task	Responsabil	L7	L8	L10	L11	L12	L13
T1.1 - Configurare React și integrare hartă	Student 1 (Cosmin)	-	-				
T1.2 - Implementare buton "Locația Mea"	Student 1 (Cosmin)		-	-			
T1.3 - Creare meniu plutitor FAB	Student 1 (Cosmin)			-	-		
T1.4 - Implementare Pop-up pentru incidente	Student 1 (Cosmin)				-	-	
T2.1 - Structurare bază de date PostgreSQL	Student 2 (Mario)	-	-				
T2.2 - API Login/Register (JWT)	Student 2 (Mario)		-	-			
T2.3 - Logică "Trust Score"	Student 2 (Mario)			-	-		
T2.4 - Cron Job ștergere incidente vechi	Student 2 (Mario)				-	-	
T3.1 - Configurare Google Maps API	Student 3 (Marius)	-	-				
T3.2 - Funcții adăugare/ștergere Markers	Student 3 (Marius)		-	-			
T3.3 - Configurare iconițe custom	Student 3 (Marius)			-	-		
T3.4 - Implementare filtrare vizuală	Student 3 (Marius)				-	-	
T3.5 - Implementare MarkerClusterer	Student 3 (Marius)					-	-

Legendă:

- **L7-L13:** Săptămânile de lucru (Laboratoare 7-13)
- **-:** Task activ în acea săptămână

---

7. Identificarea Riscurilor

7.1. Tabel Risc Aplicație

Functionalitate	Risc	Probabilitate	Impact	Prioritate	Măsuri de Reducere
Raportare polițiști	Raport fals	Medie	Moderat	Medie	Verificare captcha și rate limiting
Raportare accidente	Lipsa acuratețe locație	Mare	Ridicat	Ridicată	GPS auto-corecție, validare cu alte rapoarte

Functionalitate	Risc	Probabilitate	Impact	Prioritate	Măsuri de Reducere
Căutări utilizatori	API/Sistem	Medie	Ridicat	Ridicată	Timeout și fallback
Hartă în timp real	Server suprasolicitat	Mică	Ridicat	Medie	Scalare server și cache
Puncte/Utilizator	Fraudă puncte	Medie	Moderat	Medie	Detectare anomalii și limitări

7.2. Riscuri Generale și Măsuri de Contracarare

Risc	Probabilitate	Impact	Măsuri de Contracarare
Raportări False (Spam)	Mare	Ridicat	Implementarea sistemului de votare. Un raport nou apare ca "Neverificat" până primește 2 confirmări. Captcha la raportare.
Acuratețe GPS slabă	Mare	Ridicat	Permiterea utilizatorului să "tragă" (drag & drop) pinul pe hartă pentru a corecta poziția automată. Validare încrucișată cu alte rapoarte.
Supraîncărcare vizuală	Medie	Moderat	Clustering (gruparea mai multor puncte apropiate într-unul singur dacă se face zoom out).
Costuri Google Maps API	Medie	Ridicat	Monitorizare utilizare API, optimizare cereri (caching), folosire limitei gratuite ($200/lună), restricții API Key la domeniu.
Fraudă sistem puncte	Medie	Moderat	Algoritmi de detectare anomalii, limitare rapoarte per utilizator/zi, ban automat pentru comportament suspect.
Indisponibilitate server	Mică	Ridicat	Load balancing, auto-scaling, timeout și fallback pentru API-uri externe.