TravelSafe

Nume echipă: Skepia

Studenți:
- Salomia Andrei-Gabriel, grupa 345C1
- Zdrali Andrei-George, grupa 342C5
- Dragu Nicolae, grupa 342C5

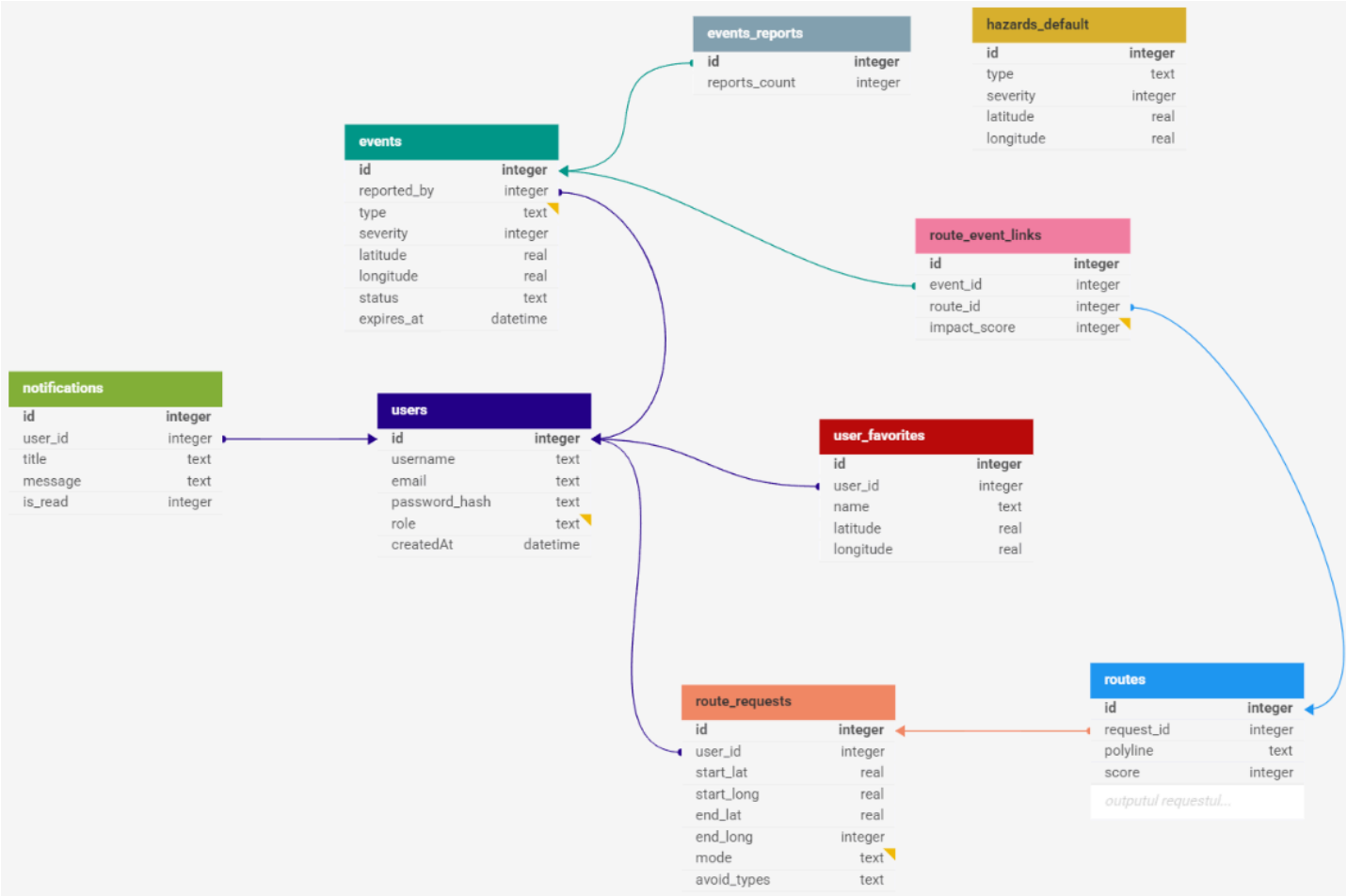
Format: Web 3 Tier (cu backend)

TravelSafe este o aplicatie web GIS pentru monitorizarea accidentelor si evenimentelor rutiere cu unelte pentru a gasi cea mai sigura ruta in orice moment de timp pentru pietoni, biciclisti dar si masini.

Structurarea datelor

Pentru persistenta si organizarea informatiilor, proiectul utilizeaza o baza de date relationala gestionata prin SQLite. Din punct de vedere fizic, toate datele sunt stocate local, intr-un singur fisier pe disc, structura care asigura portabilitatea si simplifica procesul de backup.

Datele stocate in baza de date sunt impartite in tabele interconectate.



Arhitectura generala a aplicatiei

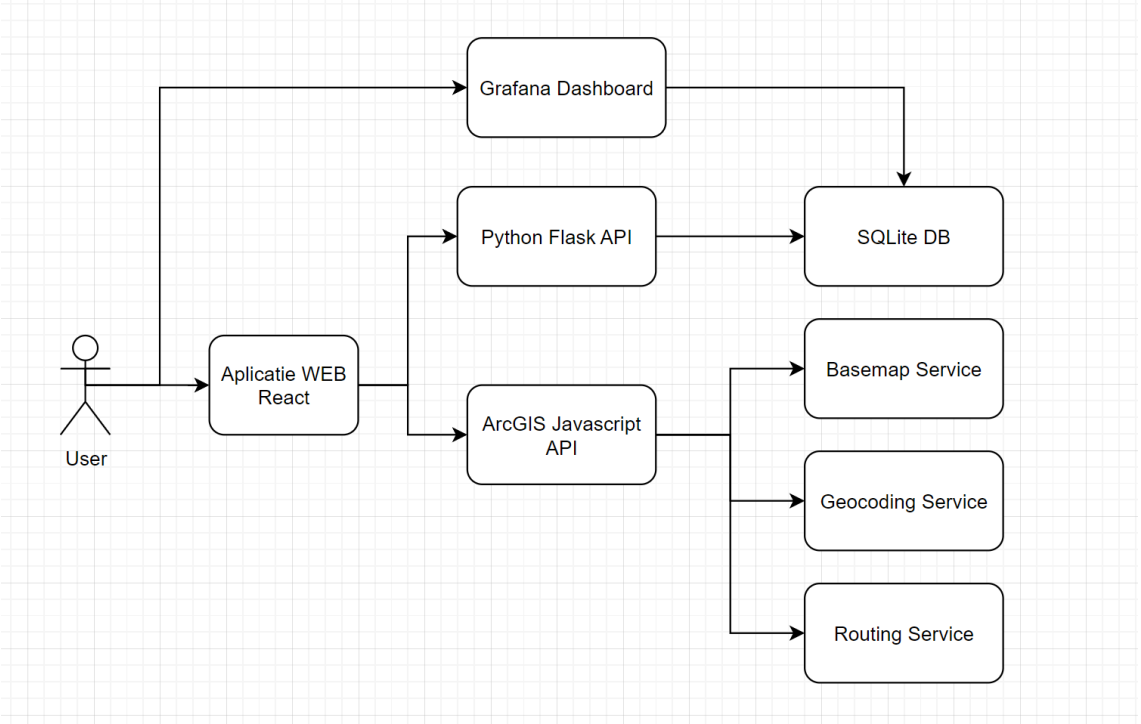
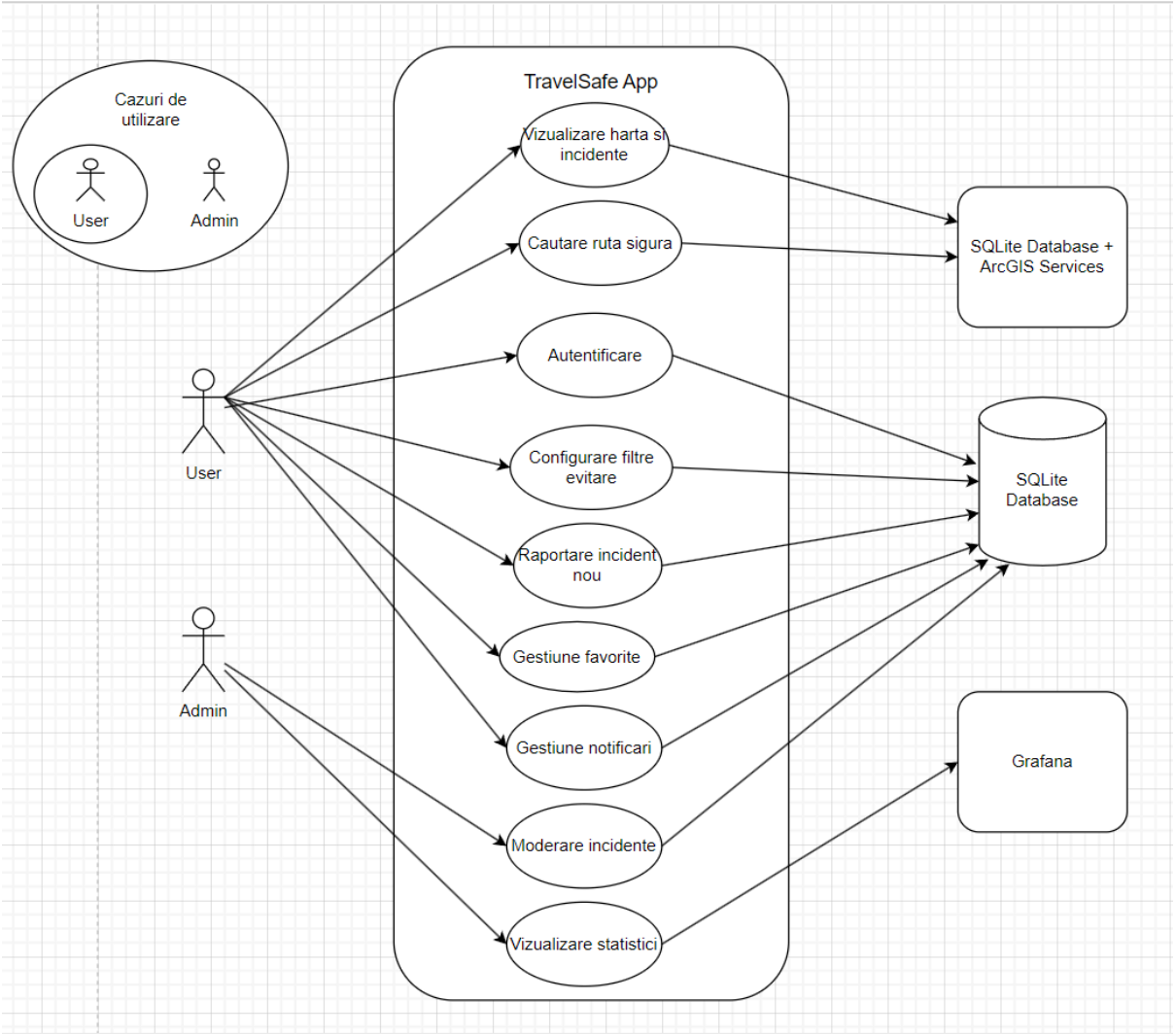


Diagrama Cazurilor de Utilizare



Tehnologii

Avand in vedere tehnologiile generale, arhitectura sistemului se bazeaza pe urmatoarele componente principale:

- Frontend-ul utilizeaza **React** pentru a construi interfata web
- Baza de date folosita este **SQLite**
- Panourile de administrare pentru analiza si administrarea datelor sunt realizate folosind **Grafana**
- Logica din backend este implementata in **Python** cu ajutorul framework-ului **Flask**
- Pentru design-ul aplicatiei vom folosi **Bootstrap CSS**

Implementarile in detaliu a componentelor individuale vor folosi urmatoarele solutii specifice:

- Autentificarea utilizatorilor se va realiza prin **JSON Web Tokens**
- Pentru stocarea in siguranta a parolelor vom folosi algoritmul **bcrypt**
- Interactiunea backend-ului cu baza de date va fi prin intermediul **SQLAlchemy**
- **ArcGIS MapView** se va folosi pentru afisarea hartii in aplicatie
- Pentru a interactiona cu harta, vom folosi **ArcGIS Maps SDK** for JavaScript.
- Pentru a transforma adresele în coordonate GPS, vom folosi **ArcGIS World Geocoding Service**
- Incidentele preluate in backend sunt redate pe harta printr-un **GraphicsLayer**
- Interactiunea cu evenimentele de pe harta se face prin **pop-ups** din ArcGIS SDK
- Organizarea endpoint-urilor API se va face **Flask Blueprints** pentru modularitate
- Comunicarea dintre frontend si backend se va securiza folosind **Flask CORS**
- Pentru efectuarea cererilor HTTP asincrone catre backend vom utiliza biblioteca **Axios**
- Notificarile de tip Toast vor fi afisate utilizand biblioteca **React-Toastify**
- Calculul rutei se va obtine prin **ArcGIS Routing Service**
- Pentru a permite citirea datelor de catre Grafana concomitent cu scrierea lor de catre Flask baza de date va fi configiurata in modul **WAL**

Organizarea Activitatilor

Task	Responsabil	LAB9	LAB10	LAB11	LAB12
Initializare Backend	**Salo**				
Definire tabele in SQLite	**Zdrali**				
Setup React si Integrare ArcGIS MapView	**Nic**				
Implementare sistem de auth pe backend si a endpointurilor de login/register cu hashing si generare token JWT	**Salo**				
Implementarea interfetei pentru login/register	**Nic**				

Creare API pentru managementul incidentelor si validare de input	**Zdrali**				
Seeding cu accidente default pentru orasul Bucuresti	**Nic**				
Adaugare layer pentru afisarea accidentelor pe harta si configurare heat map	**Salo**				
Interfata Raportare, click pe harta pentru adaugare de incident nou	**Zdrali**				
Implementare de Pop-Ups pentru incidente deja existente	**Salo**				
Integrare Routing Service pentru a calcula o ruta standard fara scor inca	**Nic**				
Implementarea unui algoritm care preia ruta default si identifica incidentele din ruta	**Nic**				
Integrare Filtre utilizator, modificarea endpointului de rutare pentru a accepta avoid_types a filtra incidentele	**Zdrali**				
Algoritm pentru calcularea unui scor de siguranta si integrarea acestuia intr-o ruta finala	**Salo**				
UI Cautare Ruta + Alegere tipul rutei: bicicleta, masina, pieton	**Zdrali**				
API Favorite: implementare CRUD pentru locatii favorite	**Salo**				
Logica notificari in caz ca se intampla ceva in apropierea unei locatii favorite	**Zdrali**				
API Notificari pentru a trimite notificarile catre frontend	**Nic**				
Interfata favorite si interfata notificari	**Zdrali**				
Logging rute in route_history	**Zdrali**				

pentru a salva rutele calculate deja pentru utilizare cu Grafana					
Conectare si setup Grafana	**Nic**				
Grafice statistici preluate din SQLite	**Salo**				

Identificarea Riscurilor

Impact / **Probabilitate**	**Scazut**	**Moderat**	**Ridicat**
Mica	Documentatie insuficienta	Probleme de compatibilitate intre versiuni ale aplicatiilor	Probleme la serviciile ArcGIS
Medie	Stiluri de lucru diferite intre membrii echipei	Indisponibilitatea membrilor echipei	Complexitatea functiilor de implementare
Mare	Cauze naturale, vreme, evenimente personale	Subestimarea timpului necesar implementarii	Neobtinerea de suficiente date coerente

Masuri de contracarare a riscurilor

- In lipsa datelor reale suficiente, vom putea genera date sintetice pentru testare
- Erorile serviciilor ArcGIS vor fi tratate prin mesaje clare, fara a bloca aplicatia
- Vom aloca o marja de timp aditionala de ~25% fata de timpul estimat initial pentru functionalitatile complexe
- Codul va fi centralizat pe GitHub pentru a fi accesibil oricand si se va folosi un coding style standardizat, alaturi de comentarii si documentatie detaliata