UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Proiect Inginerie Software
Event Finder

Universitatea Tehnică din Cluj-Napoca

Autori:
Napoleon Vanessa
Morar Adela
Paucean Ioana
Simion-Baciu Afrodita

Data:
Ianuarie 2026

Cuprins

1 Introducere

În contextul actual, accesul rapid la informații relevante despre evenimente culturale, sociale sau de divertisment a devenit o necesitate. Cu toate acestea, informațiile despre evenimente sunt adesea dispersate pe mai multe platforme, site-uri sau rețele sociale, ceea ce îngreunează procesul de căutare și selecție pentru utilizatori.

Proiectul **Event Finder** a fost realizat în cadrul disciplinei *Inginerie Software* și are ca scop dezvoltarea unei aplicații web care centralizează evenimentele disponibile într-o anumită zonă geografică, oferind utilizatorilor posibilitatea de a le căuta, filtra și salva în funcție de preferințele personale.

Aplicația este concepută atât pentru utilizatori obișnuiți, care pot explora și gestiona evenimentele favorite, cât și pentru administratori, care au posibilitatea de a gestiona conținutul platformei. Proiectul pune accent pe utilizabilitate, structură clară a codului și respectarea principiilor de bază ale ingineriei software, de la analiză și proiectare până la implementare și documentare.

2 Descrierea Proiectului

2.1 Tema proiectului

Tema proiectului constă în dezvoltarea unei aplicații web denumite **Event Finder**, care are rolul de a centraliza evenimente din diferite domenii (muzică, cultură, divertisment etc.) și de a le pune la dispoziția utilizatorilor într-o interfață unitară și ușor de utilizat. Aplicația permite căutarea și filtrarea evenimentelor în funcție de mai multe criterii, precum nume, categorie, locație sau interval de timp.

2.2 Context și motivație

Motivația realizării acestui proiect pornește de la dificultatea de a găsi evenimente relevante într-un singur loc. În mod obișnuit, informațiile despre evenimente sunt distribuite pe mai multe platforme online, rețele sociale sau site-uri diferite, ceea ce face procesul de selecție greoi și consumator de timp.

Prin realizarea aplicației *Event Finder*, ne-am propus să oferim o soluție care să adune aceste informații într-o platformă centralizată, unde utilizatorii pot explora rapid opțiunile disponibile și pot lua decizii mai ușor. De asemenea, proiectul a fost ales pentru a ne permite să aplicăm concepte studiate la disciplina Inginerie Software, precum proiectarea UML, lucrul cu baze de date, dezvoltarea unei aplicații full-stack și colaborarea într-o echipă folosind GitHub.

2.3 Obiective

Principalele obiective ale proiectului sunt:

- dezvoltarea unei baze de date pentru stocarea evenimentelor și a utilizatorilor;
- realizarea unei interfețe grafice moderne, intuitive și ușor de utilizat;
- implementarea unui sistem de autentificare pentru utilizatori și administratori;
- posibilitatea de creare și gestionare a unui profil de utilizator;
- adăugarea evenimentelor la lista de favorite (wishlist);
- filtrarea și căutarea evenimentelor după diferite criterii;
- separarea clară a rolurilor între utilizator și administrator;
- dezvoltarea unei aplicații scalabile, care poate fi extinsă în viitor cu funcționalități noi.

3 Cerințe funcționale

3.1 Lista de funcționalități

Aplicația **Event Finder** oferă utilizatorilor posibilitatea de a descoperi și gestiona evenimente într-un mod centralizat și intuitiv. Principalele funcționalități ale aplicației sunt:

- Crearea unui cont de utilizator și autentificarea în aplicație
- Autentificarea utilizatorilor existenți (login / logout)
- Vizualizarea listei de evenimente disponibile
- Căutarea evenimentelor după nume, categorie, oraș și interval de timp
- Adăugarea evenimentelor la lista de favorite (wishlist)
- Vizualizarea și gestionarea listei de evenimente favorite
- Accesarea detaliilor complete ale unui eveniment
- Gestionarea profilului personal (date utilizator, poze)
- Accesarea unei secțiuni dedicate administratorilor
- Adăugarea, modificarea și ștergerea evenimentelor de către administrator

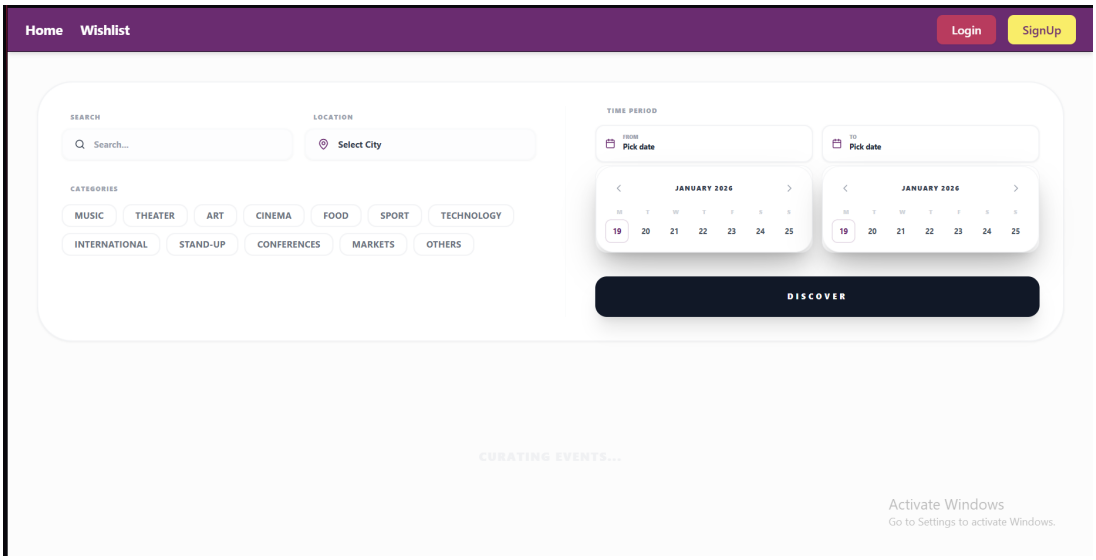

Figura 1: Pagina principală a aplicației – căutare și filtrare evenimente

3.2 Cazuri de utilizare

Principalele cazuri de utilizare identificate în cadrul aplicației sunt:

- Utilizatorul se înregistrează în aplicație
- Utilizatorul se autentifică în cont
- Utilizatorul caută evenimente folosind filtre
- Utilizatorul vizualizează detalii despre un eveniment
- Utilizatorul adaugă un eveniment la wishlist
- Utilizatorul elimină un eveniment din wishlist

- Utilizatorul își gestionează profilul personal
- Administratorul adaugă un eveniment nou
- Administratorul șterge evenimente vechi sau deja finalizate

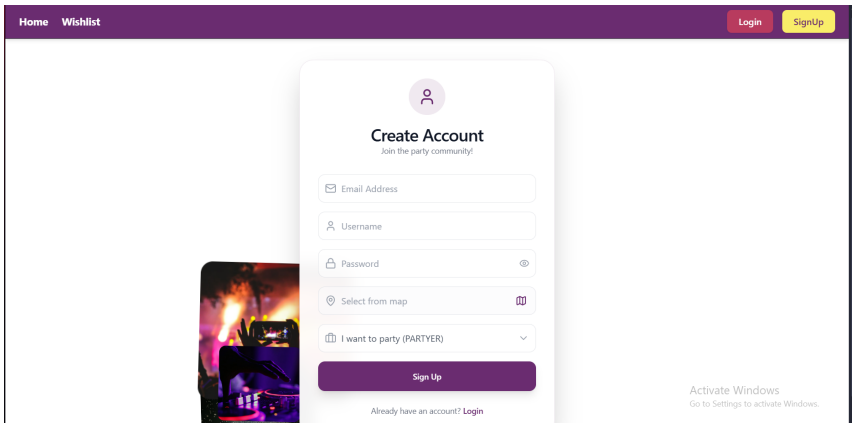

Figura 2: Autentificarea și înregistrarea utilizatorilor

3.3 Scenarii de utilizare

Scenariul 1 – Căutarea și salvarea unui eveniment:

Utilizatorul accesează aplicația și se autentifică în contul personal. Din pagina principală, acesta utilizează filtrele disponibile pentru a căuta evenimente după categorie și oraș. După identificarea unui eveniment de interes, utilizatorul accesează pagina de detalii și îl adaugă în lista de favorite pentru a-l putea accesa ulterior.

Scenariul 2 – Administrarea evenimentelor:

Administratorul se autentifică în aplicație folosind un cont cu drepturi speciale. Acesta accesează panoul de administrare, unde poate adăuga evenimente noi, modifica informațiile existente sau șterge evenimentele care au avut deja loc.

4 Cerințe non-funcționale

Pe lângă funcționalitățile oferite utilizatorilor, aplicația **Event Finder** respectă o serie de cerințe non-funcționale care influențează calitatea, performanța și securitatea sistemului.

4.1 Securitate

Aplicația utilizează un sistem de autentificare bazat pe token-uri (JWT), care permite accesul securizat la resursele protejate. Parolele utilizatorilor sunt stocate în formă criptată în baza de date, iar accesul la funcționalitățile administrative este restricționat exclusiv conturilor de tip administrator.

4.2 Performanță

Sistemul este conceput astfel încât să ofere timpi de răspuns rapizi la interogările utilizatorilor. Operațiile frecvente, precum afișarea evenimentelor sau filtrarea acestora, sunt optimizate prin utilizarea unei baze de date relaționale și a unui ORM care gestionează eficient interogările.

4.3 Scalabilitate

Arhitectura aplicației este modulară, separând clar partea de frontend, backend și baza de date. Această abordare permite extinderea aplicației în viitor, prin adăugarea de noi funcționalități sau prin susținerea unui număr mai mare de utilizatori, fără modificări majore ale structurii existente.

4.4 Utilizabilitate

Interfața aplicației este intuitivă și ușor de utilizat, fiind concepută astfel încât utilizatorii să poată naviga rapid între pagini și să găsească informațiile dorite fără dificultate. Funcționalitățile principale sunt accesibile direct din meniul principal, iar mesajele de eroare sunt afișate clar în cazul unor acțiuni invalide.

4.5 Portabilitate și constrângeri tehnice

Aplicația este dezvoltată folosind tehnologii moderne web și poate fi rulată pe diferite sisteme de operare prin intermediul containerelor Docker. Totuși, anumite funcționalități depind de servicii externe sau de configurări specifice mediului de rulare, ceea ce poate impune limitări în cazul unor medii fără suport complet pentru aceste tehnologii.

5 Diagrame UML

5.1 Use Case Diagram

Diagrama de tip Use Case prezintă actorii principali ai aplicației și interacțiunile acestora cu sistemul. În cadrul aplicației Event Finder, au fost identificați doi actori principali: utilizatorul și administratorul.

Utilizatorul are posibilitatea de a se înregistra, de a se autentifica, de a căuta și filtra evenimente, precum și de a adăuga evenimentele preferate în lista de favorite. Administratorul are rolul de a gestiona conținutul aplicației, având acces la funcționalități precum adăugarea, modificarea sau ștergerea evenimentelor.

Această diagramă oferă o imagine de ansamblu asupra funcționalităților aplicației și a modului în care acestea sunt accesate de către diferitele tipuri de utilizatori.

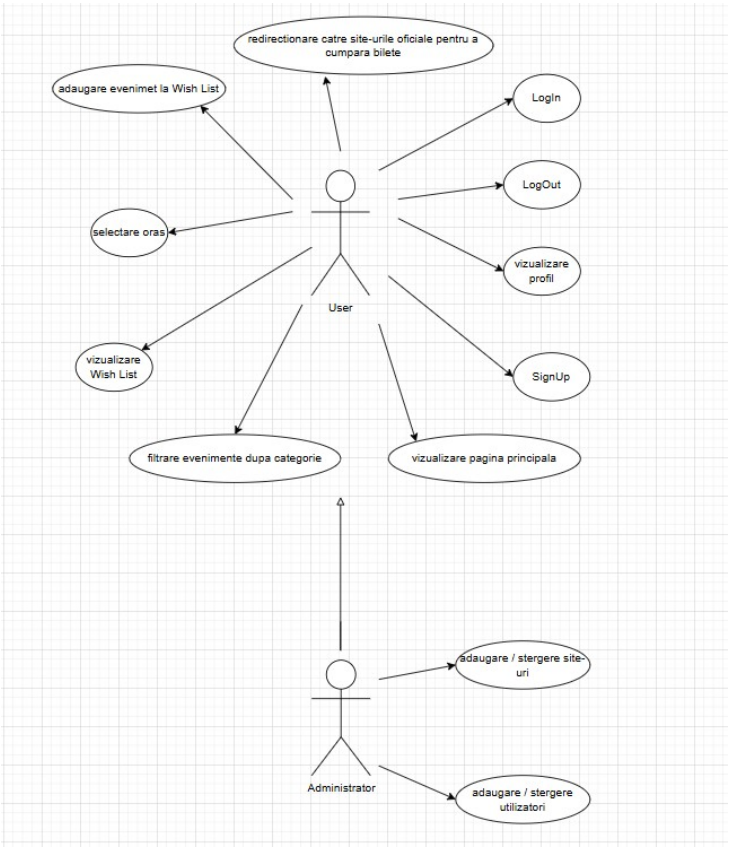

Figura 3: Use Case Diagram pentru aplicația Event Finder

5.2 Class Diagram

Diagrama de clase descrie structura statică a aplicației, evidențiind clasele principale, atributele acestora și relațiile dintre ele. Aceasta reflectă modul în care datele sunt organizate și gestionate în cadrul sistemului.

Clasele principale identificate sunt User, Event, Wishlist, Admin și Notification. Relațiile dintre clase sunt stabilite astfel încât să permită asocierea utilizatorilor cu evenimentele favorite, precum și administrarea evenimentelor de către utilizatori cu rol de administrator.

Diagrama ajută la înțelegerea arhitecturii interne a aplicației și constituie baza pentru implementarea bazei de date și a logicii aplicației.

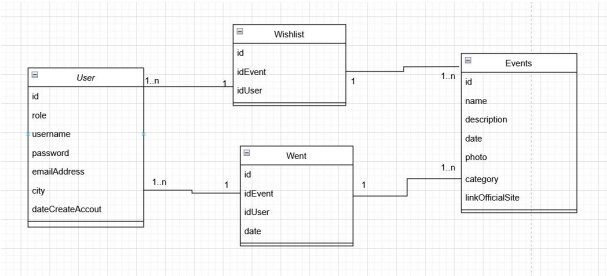

Figura 4: Class Diagram pentru aplicația Event Finder

5.3 Sequence Diagram

Diagrama de secvență ilustrează fluxul de mesaje dintre diferitele componente ale sistemului în cadrul unor scenarii concrete de utilizare. Aceasta evidențiază ordinea în care au loc interacțiunile dintre utilizator, interfața aplicației, backend și baza de date.

Un scenariu reprezentativ este cel al autentificării unui utilizator și al adăugării unui eveniment în lista de favorite. Diagrama arată pașii parcurși de la inițierea acțiunii de către utilizator până la salvarea informației în baza de date și afișarea rezultatului în interfață.

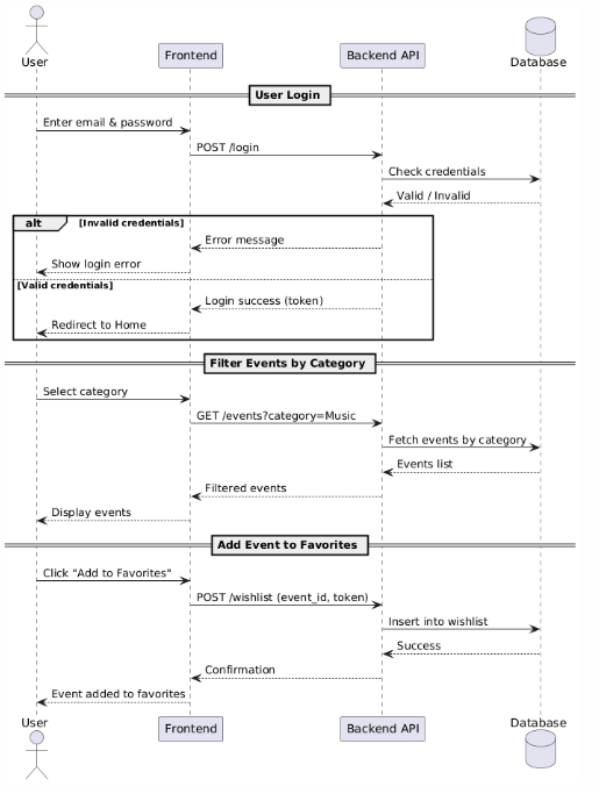

Figura 5: Sequence Diagram – autentificare și adăugare eveniment în wishlist

5.4 Activity Diagram

Diagrama de activitate descrie fluxul de activități din cadrul aplicației, de la inițierea unei acțiuni până la finalizarea acesteia. Aceasta este utilă pentru a evidenția logica proceselor și deciziile care pot apărea în timpul utilizării aplicației.

În aplicația Event Finder, diagrama de activitate poate reprezenta procesul de căutare a evenimentelor, adăugarea acestora în wishlist sau gestionarea evenimentelor de către administrator.

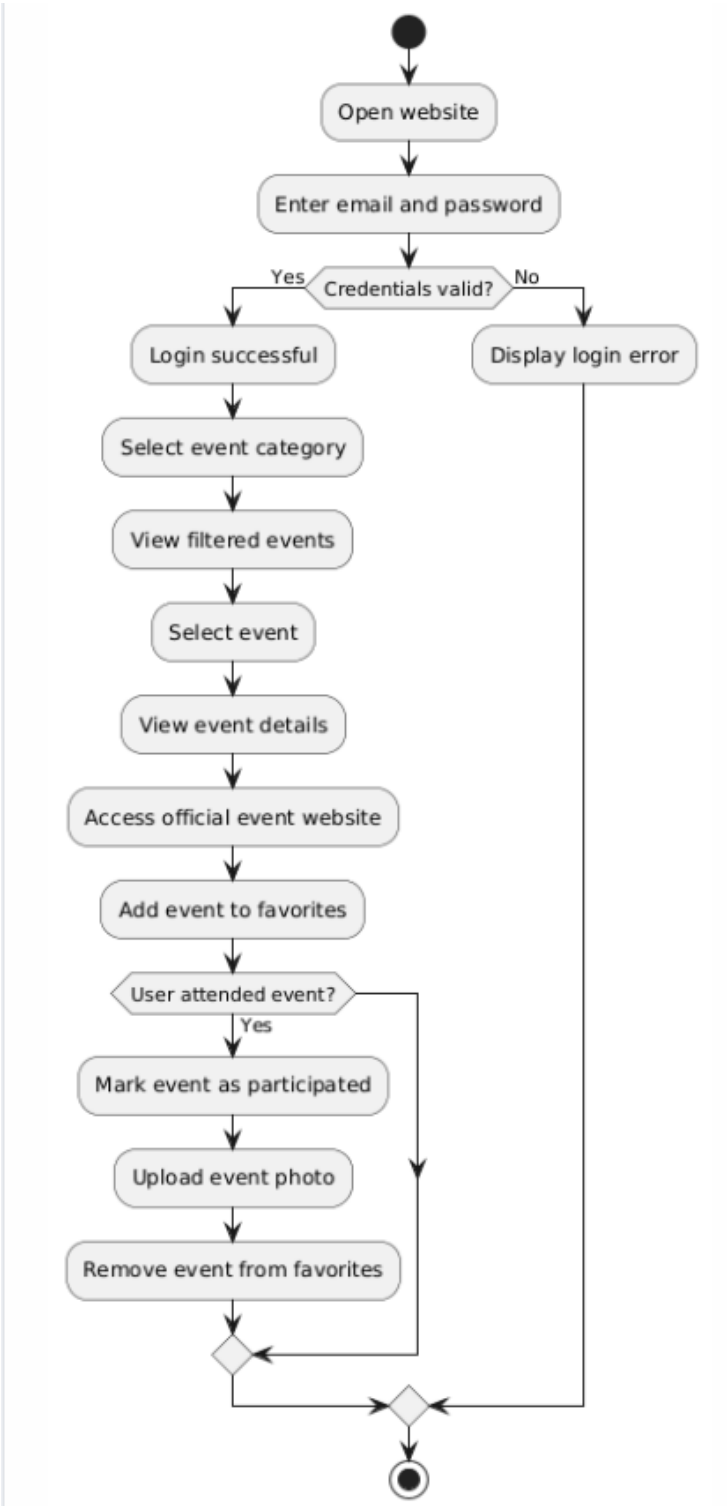

Figura 6: Activity Diagram pentru fluxul principal al aplicației

5.5 Component Diagram

Diagrama de componente prezintă arhitectura aplicației la nivel de module și evidențiază relațiile dintre frontend, backend și baza de date. Aceasta oferă o perspectivă asupra modului în care componentele software interacționează între ele.

Aplicația este structurată pe componente distincte, precum interfața utilizatorului, serverul backend, baza de date și serviciile externe utilizate pentru stocarea fișierelor sau accesarea linkurilor oficiale ale evenimentelor.

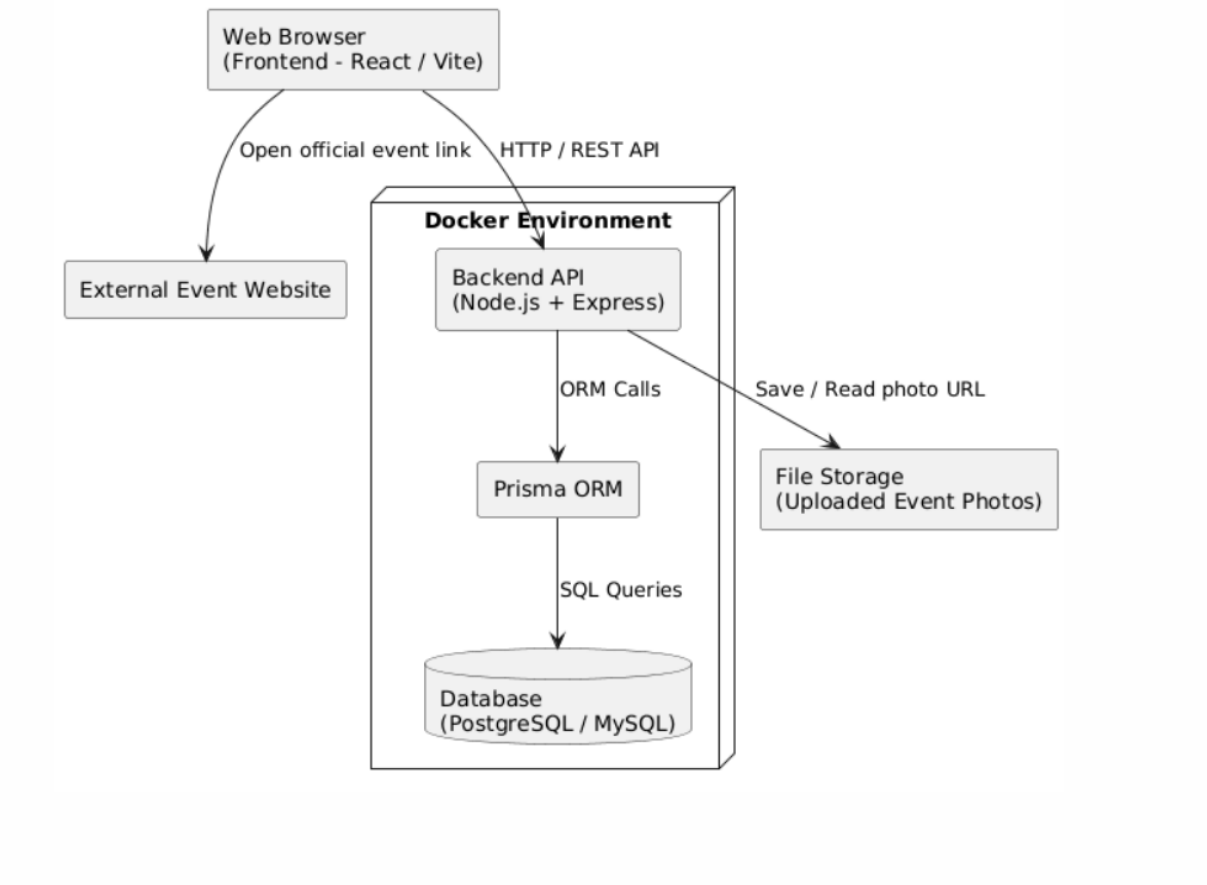

Figura 7: Component Diagram pentru arhitectura aplicației Event Finder

5.6 Package Diagram

Diagrama de tip Package ilustrează organizarea internă a aplicației pe module și pachete. Aceasta reflectă structura proiectului și separarea responsabilităților între diferitele părți ale aplicației.

Prin utilizarea acestei diagrame, este evidențiat modul în care frontend-ul și backend-ul sunt organizate în foldere și module, facilitând mentenanța și dezvoltarea ulterioară a aplicației.

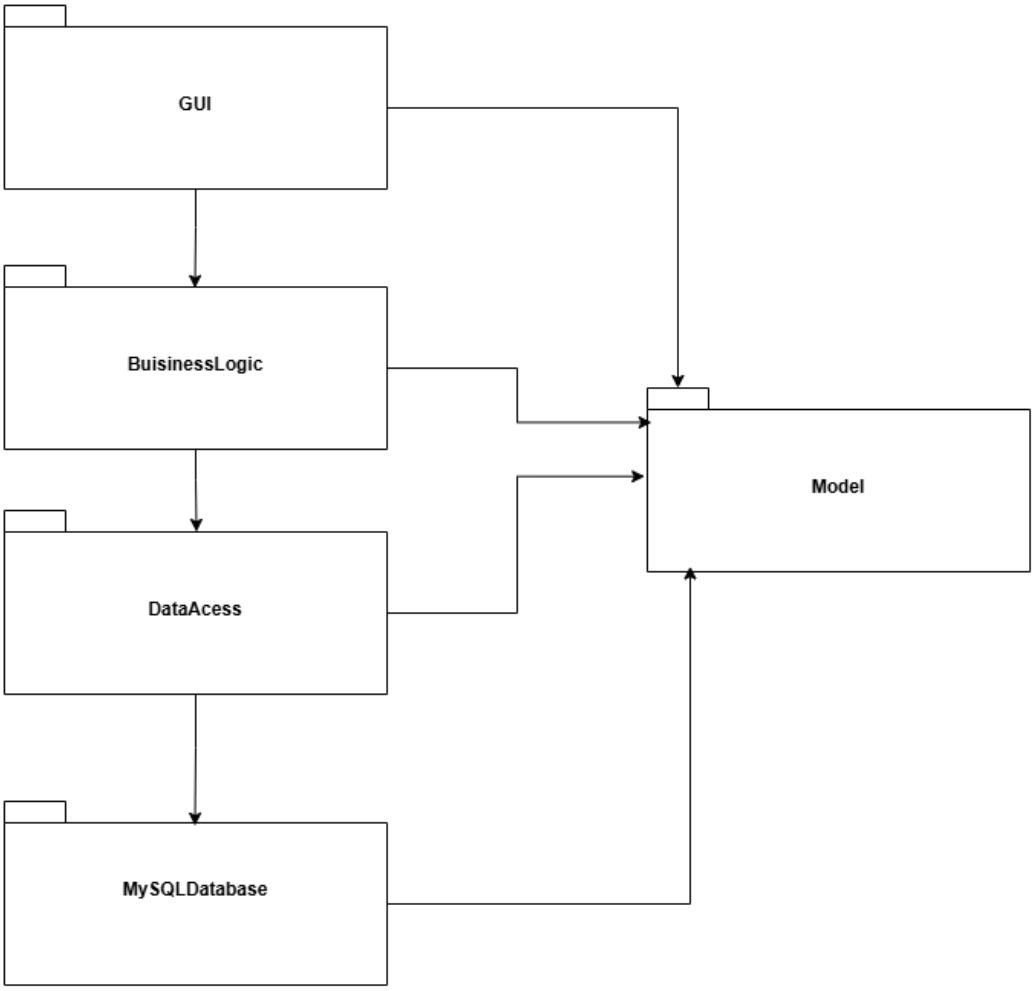

Figura 8: Package Diagram pentru organizarea internă a proiectului

5.7 State Machine Diagram

Diagrama de tip State Machine descrie stările prin care trece o entitate importantă din sistem și tranzițiile dintre aceste stări, în funcție de acțiunile utilizatorului sau de evenimentele interne ale aplicației.

În cadrul aplicației Event Finder, o entitate relevantă pentru acest tip de diagramă este relația utilizatorului cu un eveniment (de exemplu: eveniment disponibil, adăugat în wishlist, participat, respectiv șters din wishlist). Această diagramă ajută la înțelegerea logicii aplicației atunci când utilizatorul interacționează cu evenimentele (favorite/participare) și oferă o reprezentare clară a pașilor posibili și a condițiilor de trecere între stări.

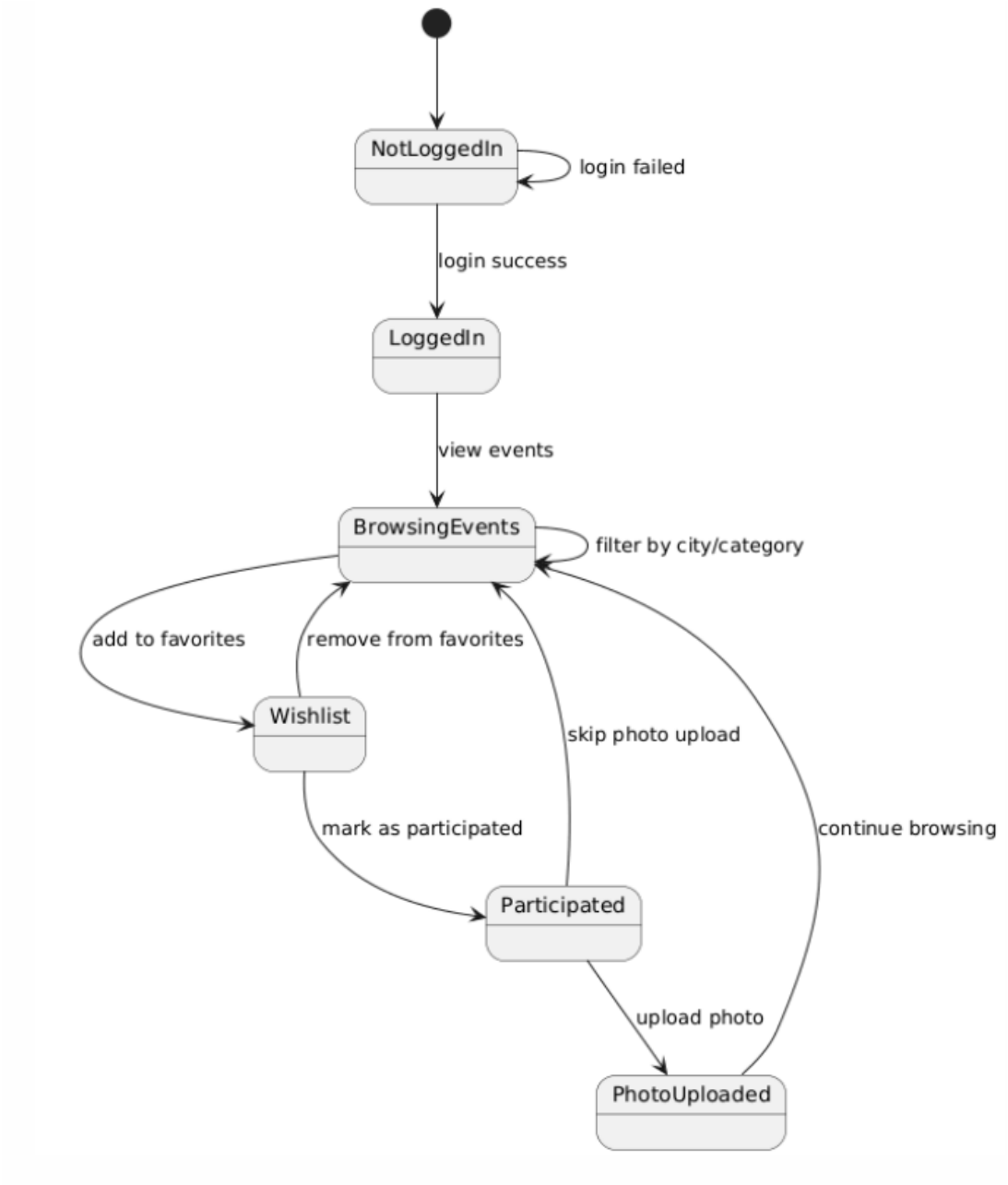

Figura 9: State Machine Diagram – stările unui eveniment pentru un utilizator

6 Arhitectura Sistemului

6.1 Prezentare generală

Aplicația **Event Finder** este realizată folosind o arhitectură de tip *client–server*, separând clar interfața utilizatorului de logica aplicației și de gestionarea datelor. Această abordare permite o dezvoltare modulară, o întreținere mai ușoară a codului și o bună organizare a responsabilităților fiecărei componente.

Sistemul este împărțit în trei părți principale: frontend, backend și baza de date, care comunică între ele prin intermediul unor interfețe bine definite.

6.2 Arhitectură pe straturi

Arhitectura aplicației este una pe straturi (*layered architecture*), fiecare strat având un rol clar definit:

- **Stratul de prezentare (Frontend)** – responsabil de afișarea interfeței grafice și de interacțiunea

directă cu utilizatorul. Acesta este realizat folosind tehnologii web moderne și comunică cu backend-ul prin apeluri HTTP.

- **Stratul de logică (Backend)** – gestionează logica aplicației, autentificarea utilizatorilor, validarea datelor și procesarea cererilor primite de la frontend.
- **Stratul de date (Baza de date)** – este responsabil de stocarea persistentă a informațiilor despre utilizatori, evenimente, liste de favorite și alte date relevante ale aplicației.

Această separare permite modificarea unui strat fără a afecta semnificativ celelalte componente ale sistemului.

6.3 Module și interacțiuni

Frontend-ul comunică cu backend-ul prin intermediul unor endpoint-uri REST, trimițând cereri pentru operații precum autentificare, căutare evenimente sau gestionarea wishlist-ului. Backend-ul procesează aceste cereri și interacționează cu baza de date pentru a returna informațiile necesare.

Pentru accesarea unor resurse externe, precum linkurile oficiale ale evenimentelor sau încărcarea pozelor, aplicația poate apela servicii externe, fără a afecta funcționalitatea de bază a sistemului.

6.4 Justificarea alegerii tehnologiilor

Tehnologiile utilizate au fost alese astfel încât să asigure flexibilitate, performanță și ușurință în dezvoltare. Utilizarea unui framework modern pentru frontend permite realizarea unei interfețe dinamice, iar backend-ul oferă suport pentru logica aplicației și comunicarea securizată cu baza de date.

De asemenea, utilizarea containerizării prin Docker facilitează rularea aplicației în medii diferite și simplifică procesul de configurare și testare a sistemului.

7 Design Patterns Utilizate

În cadrul aplicației **Event Finder** au fost aplicate mai multe tipare și principii de proiectare care au ajutat la organizarea codului, separarea responsabilităților și menținerea unei structuri ușor de extins. Deși proiectul este unul academic, s-a urmărit implementarea într-un mod cât mai apropiat de practică, folosind pattern-uri frecvent întâlnite în aplicațiile web.

7.1 Model–View–Controller (MVC)

Backend-ul aplicației urmează principiile MVC prin separarea clară a responsabilităților:

- **Model** – reprezentat de schema Prisma (de exemplu: *User, Event, City, Wishlist, AdminInviteCode*), care definește structura datelor și relațiile dintre entități;
- **Controller** – reprezentat de endpoint-urile din Express, care primesc cereri HTTP, aplică logica de business și trimit răspunsuri către client;
- **View** – reprezentată de frontend-ul React, care afișează datele utilizatorului și consumă informațiile oferite de backend în format JSON.

Această organizare face proiectul mai ușor de întreținut și permite extinderea ulterioară fără modificări majore în codul existent.

7.2 Repository Pattern (prin Prisma Client)

Accesul la baza de date este realizat prin Prisma Client, care funcționează ca un strat de abstractizare peste PostgreSQL. În loc să lucrăm direct cu query-uri SQL în cod, operațiile CRUD sunt realizate prin metodele oferite de ORM. Beneficiile acestui pattern sunt:

- decuplarea logicii aplicației de detaliile bazei de date;
- posibilitatea de a modifica structura datelor prin migrații fără a rescrie masiv codul;
- cod mai curat și mai ușor de testat.

7.3 Middleware Pattern

Backend-ul utilizează middleware-uri Express pentru prelucrarea cererilor înainte de a ajunge la endpoint-urile principale. Exemple relevante sunt:

- middleware de **autentificare** – verifică existența și validitatea token-ului JWT;
- middleware de **autorizare** – verifică rolul utilizatorului (de exemplu, accesul la funcționalități de admin);
- validări ale datelor de intrare (câmpuri obligatorii, formate).

Acest pattern crește reutilizarea codului și menține endpoint-urile mai simple și mai ușor de urmărit.

7.4 REST Architectural Pattern

Comunicarea dintre frontend și backend respectă principiile REST:

- folosirea metodelor HTTP standard (GET, POST etc.);
- endpoint-uri clare pentru resurse (ex.: *users, events, cities, wishlist*);
- schimb de date în format JSON;
- comunicare de tip stateless (token-ul JWT este trimis la fiecare cerere protejată).

Această abordare facilitează integrarea dintre componente și permite scalarea aplicației.

7.5 Factory Pattern (utilizare implicită)

În anumite fluxuri ale aplicației, crearea obiectelor este realizată controlat, prin funcții dedicate (de exemplu: înregistrarea utilizatorilor, generarea codurilor de invitație pentru administrator, inserarea automată de evenimente din scriptul Python). Chiar dacă nu există o clasă „Factory” în mod explicit, comportamentul reflectă acest pattern prin faptul că obiectele sunt create într-un mod standardizat și validat.

7.6 Singleton Pattern

Instanța Prisma Client este tratată ca o resursă unică (o singură instanță reutilizată) în cadrul aplicației backend. Astfel se evită inițializări repetate și consum inutil de resurse, iar conexiunea la baza de date rămâne stabilă pe durata rulării aplicației.

7.7 Separation of Concerns (SoC)

De-a lungul proiectului a fost respectat principiul separării responsabilităților:

- **frontend** – interfață și interacțiune utilizator;
- **backend** – logică de business, securitate, API;
- **script Python** – import automat de evenimente;
- **Docker** – configurarea și rularea infrastructurii.

Această separare crește lizibilitatea proiectului și permite ca fiecare componentă să fie dezvoltată și testată independent.

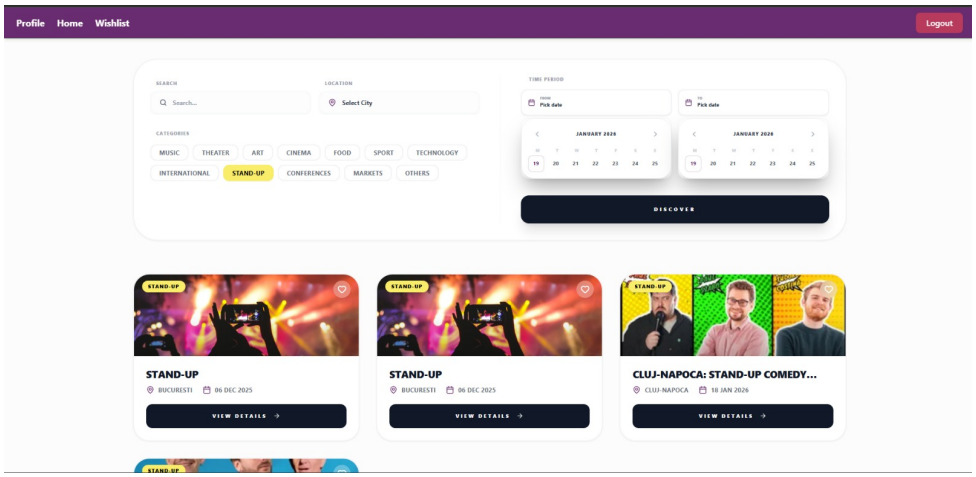

Figura 10: Pagina Wishlist – evenimente favorite

8 Implementare

8.1 Structura proiectului

Proiectul Event Finder este organizat într-o structură clară, separând responsabilitățile între partea de frontend și cea de backend. Această organizare facilitează dezvoltarea, testarea și mentenanța aplicației.

Structura principală a proiectului este împărțită în următoarele componente:

- **Frontend** – responsabil de interfața grafică și interacțiunea cu utilizatorul;
- **Backend** – responsabil de logica aplicației, autentificare și gestionarea datelor;
- **Baza de date** – utilizată pentru stocarea persistentă a informațiilor despre utilizatori, evenimente și listele de favorite.

8.2 Funcționalități principale implementate

Printre funcționalitățile principale implementate în cadrul aplicației se numără autentificarea utilizatorilor, afișarea și filtrarea evenimentelor, gestionarea listei de favorite (wishlist), precum și administrarea evenimentelor de către utilizatori cu rol de administrator.

Aplicația permite utilizatorilor să navigheze rapid între pagini, să vizualizeze detalii despre evenimente și să își personalizeze experiența prin salvarea evenimentelor preferate.

8.3 Căutarea și filtrarea evenimentelor

Aplicația permite selectarea orașului direct de pe harta României, oferind o metodă vizuală și intuitivă de filtrare a evenimentelor. Utilizatorul poate face click pe zona corespunzătoare unui județ sau oraș de pe hartă, iar în urma acestei acțiuni, lista de evenimente este actualizată automat pentru locația selectată.

La interacțiunea cu harta, este afișat numele orașului ales, iar utilizatorul poate vedea imediat evenimentele disponibile în acea zonă, fără a fi necesară introducerea manuală a orașului într-un câmp de căutare. Această funcționalitate îmbunătățește experiența de utilizare și facilitează explorarea rapidă a evenimentelor la nivel național.

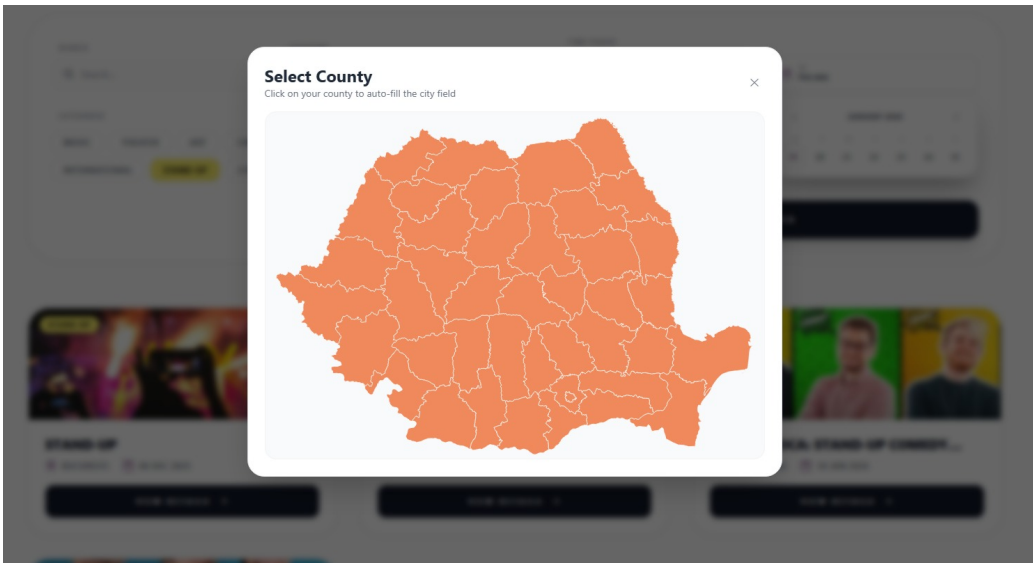

Figura 11: Selectarea orașului prin interacțiunea cu harta României

8.4 Încărcarea imaginilor în profilul utilizatorului

Aplicația permite utilizatorilor autentificați să încarce imagini din dispozitivul personal, care sunt asociate profilului acestora și afișate sub forma unui carusel. Această funcționalitate oferă posibilitatea utilizatorilor de a salva și vizualiza momente sau amintiri de la evenimentele la care au participat.

Încărcarea imaginilor implică o comunicare directă între frontend și backend, fiind necesară configurarea corespunzătoare a permisiunilor de acces pentru a permite trimiterea fișierelor către server. După upload, imaginile sunt procesate și afișate în interfața aplicației.

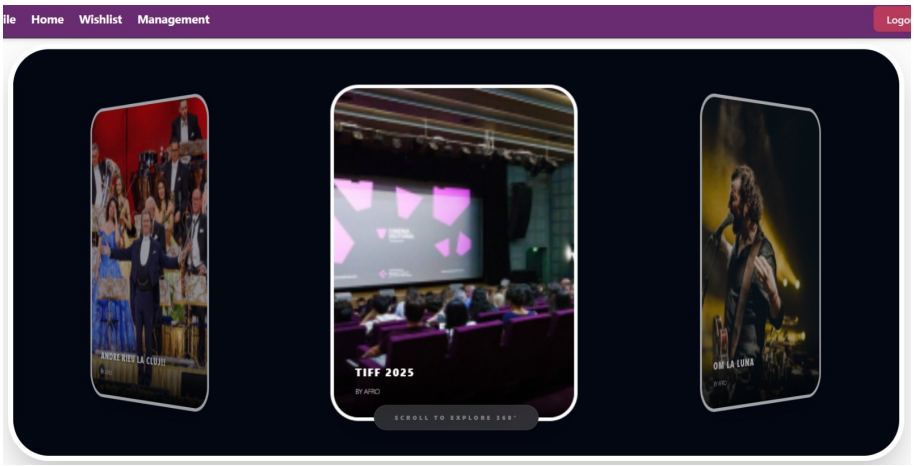

Figura 12: Încărcarea imaginilor în profilul utilizatorului

8.5 Adăugarea de evenimente și limitări de stocare

Aplicația permite administratorilor să adauge evenimente noi în platformă, prin intermediul unei interfețe dedicate de management. Pentru fiecare eveniment pot fi introduse informații precum titlul, descrierea, locația, data, categoria și sursa evenimentului.

În ceea ce privește gestionarea fișierelor media (imagini asociate evenimentelor sau încărcate de utilizatori), aplicația este configurată pentru a funcționa optim cu fișiere de dimensiuni reduse. Această limitare este impusă din considerente de memorie și performanță, fiind specifică mediului de dezvoltare și scopului academic al proiectului.

Imaginile de dimensiuni mici sunt încărcate și afișate corect în aplicație, însă fișierele foarte mari pot necesita soluții suplimentare de stocare externă, care pot fi integrate în versiuni viitoare ale aplicației.

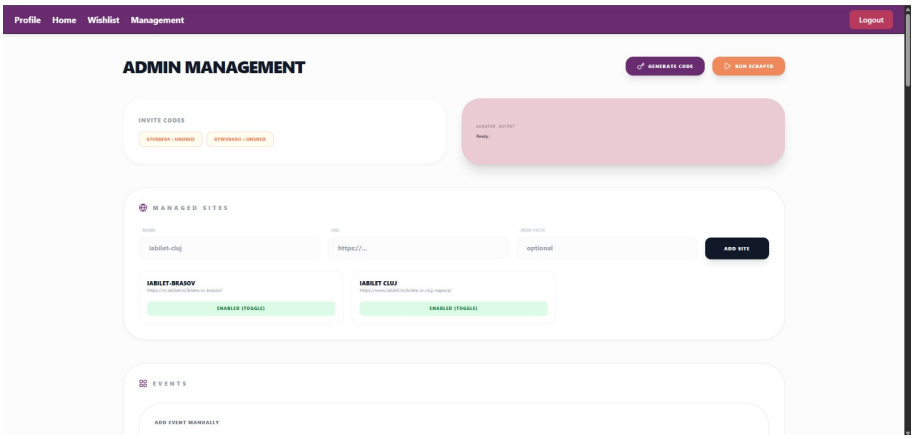

Figura 13: pagina Admin management

8.6 Ștergerea evenimentelor

Aplicația **Event Finder** permite administratorilor să șteargă evenimentele existente din sistem, în special în situațiile în care acestea au avut deja loc sau nu mai sunt relevante pentru utilizatori.

Funcționalitatea de ștergere este disponibilă exclusiv în interfața de administrare și este protejată prin mecanisme de autentificare și autorizare, astfel încât doar utilizatorii cu rol de administrator pot efectua această acțiune. La inițierea comenzii de ștergere, aplicația verifică drepturile utilizatorului și elimină evenimentul selectat din baza de date.

Ștergerea unui eveniment determină, de asemenea, eliminarea automată a legăturilor asociate acestuia (de exemplu, din lista de evenimente favorite ale utilizatorilor), asigurând consistența datelor și evitarea apariției informațiilor invalide în aplicație.

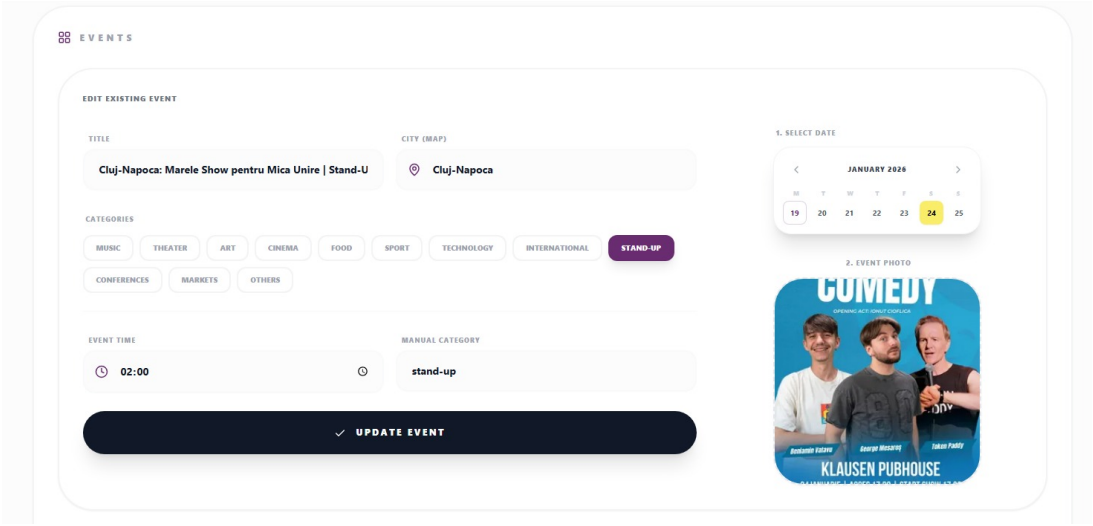

Figura 14: editere eveniment

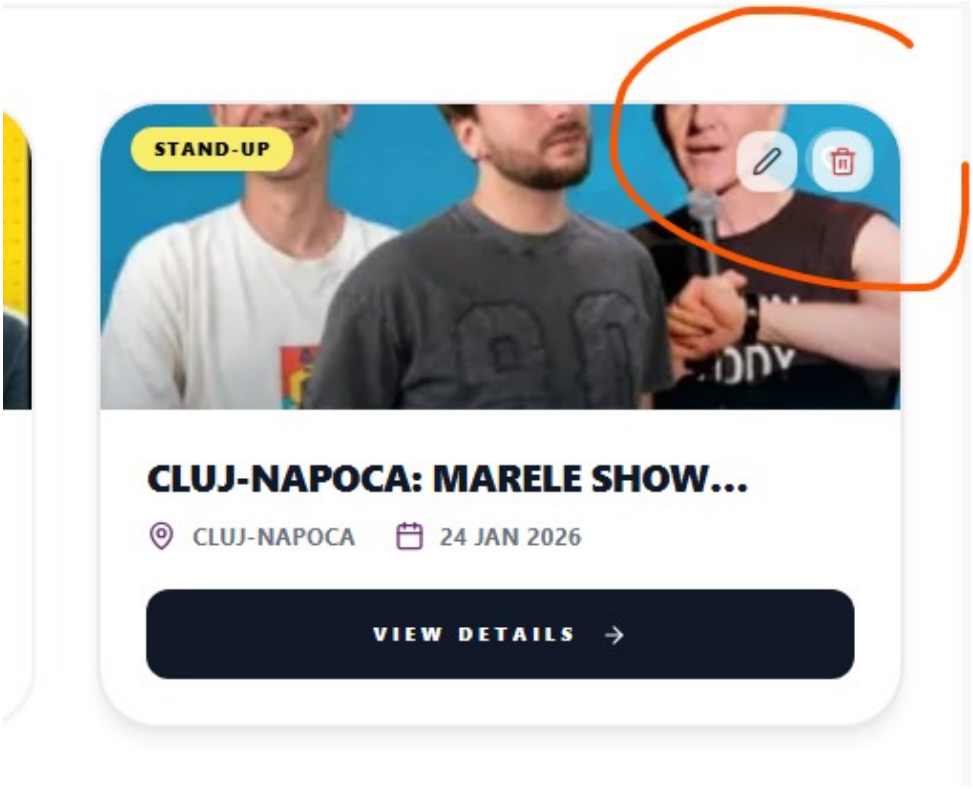

Figura 15: Ștergerea unui eveniment din interfața de administrare

8.7 Interfața API

Comunicarea dintre frontend și backend se realizează prin intermediul unor endpoint-uri REST. Acestea permit efectuarea operațiilor de bază, precum autentificarea utilizatorilor, obținerea listei de evenimente, filtrarea acestora și gestionarea listei de favorite.

Această abordare asigură o separare clară între interfață și logica aplicației și permite extinderea ulterioară a funcționalităților fără modificări majore ale codului existent.

8.8 Fluxul aplicației

Fluxul general al aplicației începe cu accesarea paginii principale, unde utilizatorul poate vizualiza evenimentele disponibile. Pentru a accesa funcționalitățile avansate, precum adăugarea în wishlist, utilizatorul trebuie să se autentifice.

După autentificare, utilizatorul poate naviga prin aplicație, poate filtra evenimentele și le poate salva pe cele de interes. Administratorul are acces la funcționalități suplimentare, care permit gestionarea conținutului aplicației.

9 Planificare și Management

9.1 Organizarea echipei

Proiectul Event Finder a fost realizat în echipă, fiecare membru având responsabilități clar definite. Împărțirea sarcinilor a fost realizată astfel încât să valorifice competențele fiecărui membru și să asigure o bună colaborare pe parcursul dezvoltării aplicației.

Rolurile principale în cadrul echipei au fost:

- **Napoleon Vanessa** – proiectarea bazei de date la nivel conceptual, inițial definită în MySQL (tabele, chei primare și relații), realizarea niste diagrame UML și documentație;
- **Morar Adela** –dezvoltarea interfeței grafice (frontend) a aplicației folosind React, prin utilizarea și adaptarea unor componente moderne din librării precum React Bits și 21st.dev. A contribuit la

structurarea paginilor principale ale aplicației, realizarea componentelor reutilizabile și integrarea acestora cu backend-ul prin apeluri API, asigurând o experiență de utilizare fluentă și coerentă. De asemenea, a participat la testarea interfeței , si contribuția la documentație.;

- **Paucean Ioana** – implementarea backend-ului (endpoint-uri API), autentificare și autorizare cu JWT (inclusiv middleware pentru verificarea rolurilor), definirea și actualizarea schemei bazei de date cu Prisma (migrații), gestionarea evenimentelor (listare/filtrare/ordonare), sistem de coduri de invitație pentru admin, precum și scriptul Python pentru import automat de evenimente în baza de date.
- **Simion-Baciu Afrodita** – integrarea unor funcționalități între frontend și backend și rezolvarea problemelor de comunicare între componente (în special configurări CORS). A contribuit la legarea acțiunilor din interfață cu endpoint-urile backend, inclusiv pentru rularea scriptului de import (scraper) din zona de Management și pentru încărcarea imaginilor în profilul utilizatorului.

9.2 Managementul versiunilor

Pentru gestionarea codului sursă și colaborarea între membrii echipei, a fost utilizată platforma GitHub. Fiecare membru a lucrat pe branch-uri dedicate, iar integrarea modificărilor a fost realizată prin mecanismele oferite de Git.

Această abordare a permis urmărirea modificărilor, evitarea conflictelor majore și o mai bună organizare a contribuțiilor fiecărui membru.

9.3 Link repository

Codul sursă al proiectului este disponibil pe platforma GitHub, la următorul link:

https://github.com/ioana333/proiect_IS

10 Concluzii

În urma realizării proiectului **Event Finder**, echipa a reușit să dezvolte o aplicație web funcțională, care îndeplinește obiectivele stabilite inițial. Aplicația oferă utilizatorilor posibilitatea de a descoperi și gestiona evenimente într-un mod centralizat, simplu și intuitiv, răspunzând unei nevoi reale întâlnite în viața de zi cu zi.

Pe parcursul dezvoltării proiectului, au fost aplicate concepte studiate la disciplina Inginerie Software, precum analiza cerințelor, proiectarea UML, organizarea arhitecturală pe straturi și colaborarea eficientă într-o echipă folosind sisteme de control al versiunilor. De asemenea, proiectul a contribuit la consolidarea cunoștințelor practice legate de dezvoltarea aplicațiilor web full-stack.

Deși aplicația este funcțională în forma actuală, aceasta poate fi extinsă în viitor prin adăugarea unor funcționalități suplimentare, precum recomandări personalizate de evenimente, integrarea unor servicii externe pentru achiziționarea biletelor sau îmbunătățirea interfeței grafice. Astfel, Event Finder reprezintă o bază solidă pentru dezvoltări ulterioare și un exemplu relevant de aplicare a principiilor ingineriei software într-un proiect practic.