

MINISTERUL EDUCAȚIEI



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

FESTIVALGO-PLATFORMĂ WEB PENTRU ORGANIZAREA ȘI PERSONALIZAREA PARTICIPĂRII LA FESTIVALURI MUZICALE

PROIECT DE DIPLOMĂ

Autor: **Robert Ciceu**

Conducător științific: **S. L. Dr. Ing. Dan Radu**

2025



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Vizat,

DECAN

Prof. dr. ing. Vlad MUREȘAN

DIRECTOR

AUTOMATICĂ

DEPARTAMENT

Prof. dr. ing. Honoriu VĂLEAN

Autor: **Robert Ciceu**

FestivalGo–Platformă web pentru organizarea și personalizarea participării la festivaluri muzicale

1. **Enunțul temei:** *Lucrarea dezvoltă crearea unei platforme web pentru personalizarea recomandărilor la festivaluri muzicale pe baza unui test de personalitate, achiziționarea de bilete, vizualizarea punctelor de interes cu ajutorul unei hărți interactive și crearea grupurilor de discuții.*
2. **Conținutul proiectului:** *Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Introducere, Studiu Bibliografic, Analiză, proiectare și implementare, Bibliografie*
3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca*
4. **Consultanți:** -
5. **Data emiterii temei:** 18.10.2024
6. **Data predării:** 07.07.2025

Semnătura autorului

Robert Ciceu

Semnătura conducătorului științific

SR



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

**Declarație pe proprie răspundere privind
autenticitatea proiectului de diplomă**

Subsemnatul(a) **Robert Ciceu**,
legitimat(ă) cu CI seria XM nr. 038612, CNP.
5020612245073,

autorul lucrării: FestivalGo-Platformă web pentru organizarea și personalizarea
participării la festivaluri muzicale

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la
Facultatea de Automatică și Calculatoare, specializarea **Automatică și Informatică
Aplicată**, din cadrul Universității Tehnice din Cluj-Napoca, sesiunea iulie 2025 a anului
universitar 2024-2025, 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

07.07.2025

Robert Ciceu

Robert Ciceu



SINTEZA

proiectului de diplomă cu titlul:

FestivalGo–Platformă web pentru organizarea și personalizarea participării la festivaluri muzicale

Autor: **Robert Ciceu**

Conducător științific: **S. L. Dr. Ing. Dan Radu**

1. Cerințele temei: Dezvoltarea unei aplicații full-stack care permite personalizarea participării la festivaluri prin intermediul unui chestionar pe bază de personalitate, achiziționare de bilete și vizualizarea punctelor de interes folosind hărți interactive.
2. Soluții alese: Java + Spring Boot + Spring Data JPA + Spring Security + MySQLDriver, JavaScript + Vue.js + Vite + Leaflet, Stripe, MySQL
3. Rezultate obținute: O platformă web care permite atât administrarea festivalurilor, cât și gestionarea participărilor, achiziționarea de bilete, vizualizarea hărților interactive, determinarea genului muzical recomandat pentru un utilizator și comunicarea prin grupuri de discuții.
4. Testări și verificări: Aplicația web a fost testată cu succes demonstrând funcționalitatea corectă a administrării festivalurilor, achiziționării biletelor, afișării hărților și a grupurilor de discuții. Sistemul a răspuns corect la toate interacțiunile simulate.
5. Contribuții personale: Realizarea integrală a logicii pe partea de server, implementarea paginilor web pentru administrator și client, testarea sistemului la nivel de interfață și componenta de procesare a datelor, precum și integrarea funcționalităților de vizualizare a hărților, comunicarea între utilizatori și achiziționarea online a biletelor.
6. Surse de documentare: articole științifice (ResearchGate) și documentațiile oficiale ale tehnologiilor utilizate (Spring Boot, Vue.js, Leaflet, MySQL Axios etc.)

Semnătura autorului

Robert Ciceu

Semnătura conducătorului științific

SR

Cuprins

1	INTRODUCERE	2
1.1	CONTEXT GENERAL.....	2
1.2	OBIECTIVE	3
1.3	SPECIFICAȚII.....	4
2	STUDIU BIBLIOGRAFIC.....	7
2.1	ANALIZA COMPARATIVĂ A PLATFORMELOR EXISTENTE	7
2.2	ANALIZA ARTICOLELOR ȘTIINȚIFICE RELEVANTE PENTRU CREAREA APLICAȚIEI FESTIVALGO	8
2.3	TEHNOLOGII FOLOSITE PENTRU CREAREA BACKEND-ULUI APLICAȚIEI FESTIVALGO	9
2.4	TEHNOLOGII FOLOSITE PENTRU CREAREA FRONTEND-ULUI APLICAȚIEI FESTIVALGO.....	14
2.5	TEHNOLOGII FOLOSITE PENTRU CREAREA BAZEI DE DATE A APLICAȚIEI FESTIVALGO.....	17
3	ANALIZĂ, PROIECTARE, IMPLEMENTARE	19
3.1	ARHITECTURA GENERALĂ A PLATFORMEI.....	19
3.2	STRUCTURA LOGICĂ A PLATFORMEI ȘI ORGANIZAREA MODULELOR	20
3.3	PROIECTAREA SISTEMULUI	22
3.3.1	<i>Cazuri de utilizare</i>	22
3.3.2	<i>Baza de date</i>	25
3.4	IMPLEMENTARE	26
3.4.1	<i>Descărcarea pachetelor necesare</i>	26
3.4.2	<i>Serverul aplicației</i>	27
3.4.2.1	Conectarea bazei de date la server	27
3.4.2.2	Utilizatori.....	27
3.4.2.3	Festival	30
3.4.2.4	Participări	31
3.4.2.5	Chestionar muzical	32
3.4.2.6	Mesagerie	34
3.4.2.7	Configurări	36
3.4.3	<i>Interfața aplicației</i>	36
3.4.3.1	Autentificare și Înregistrare	36
3.4.3.2	Client	37
3.4.3.3	Administrator	39
3.4.3.4	Vue Router	39
3.4.4	<i>Funcționalități externe</i>	40
3.4.4.1	Harta	40
3.4.4.2	Stripe.....	40
3.5	TESTARE ȘI VALIDARE	41
4	CONCLUZII.....	50
4.1	REZULTATE OBȚINUTE	50
4.2	DIRECȚII DE DEZVOLTARE	50
5	BIBLIOGRAFIE.....	52

1 Introducere

1.1 Context general

Industria evenimentelor live s-a dezvoltat rapid în ultimele decenii și a ajuns să joace un rol important în lumea divertismentului de azi. Chiar dacă tehnologia a schimbat radical felul în care comunicăm, festivalurile muzicale continuă să rămână relevante tocmai pentru că oferă ceva ce spațiul digital nu poate reproduce: sentimentul unic de a trăi o experiență colectivă, alături de mii de oameni care au aceleași pasiuni.

În prezent, festivalurile nu mai sunt doar despre muzică. Ele au devenit evenimente complexe, gândite în detaliu, cu o atmosferă unică, care atrag participanții, nu doar prin muzică, ci și prin decor, tematică, locație și tot ceea ce ține de organizare. Pentru mulți tineri, participarea la aceste evenimente este mai mult decât o distracție, este o evadare, o formă de exprimare, o oportunitate de a trăi câteva zile altfel decât în rutina zilnică.

Datorită acestei dezvoltări, aceste evenimente s-au transformat în stiluri de viață temporare. Oamenii vin să se simtă liberi, să se conecteze cu alții, să își creeze amintiri și să simtă că fac parte din ceva special. De aceea, festivalurile muzicale nu mai sunt văzute ca simple evenimente muzicale, ci ca experiențe personale, care rămân cu participanții mult timp după ce se încheie muzica.

Odată cu aceste schimbări, și așteptările publicului s-au transformat. Cei care participă la festivaluri doresc să fie implicați, să aibă acces la toate informațiile, să își personalizeze experiența și să interacționeze ușor cu ceilalți. În general, se dorește o aplicație care să-i ajute să descopere festivaluri potrivite gusturilor lor, să își organizeze mai bine participarea și să creeze noi conexiuni.

Din acest motiv, aplicațiile web și mobile pentru festivaluri au un potențial imens. Nu este suficient ca o aplicație să dezvăluie programul sau o listă de artiști. Publicul dorește recomandări personalizate, hărți interactive ușor de folosit, informații utile la un buton distanță și, mai ales, posibilitatea de a interacționa cu alți participanți.

Din păcate, majoritatea aplicațiilor existente se limitează la funcționalități de bază, iar acest lucru lasă loc pentru inovație. Într-o lume în care experiențele sunt mai valoroase decât obiectele materiale, avem nevoie de soluții care țin pasul cu generațiile actuale.

Motivația acestei lucrări este crearea unei platforme web full-stack, intitulată FestivalGo, menită să ajute gestionarea participărilor la festivaluri și îmbunătățirea experienței utilizatorului prin implementarea unor funcționalități practice. Aplicația integrează aspecte de organizare individuală, cât și în grup, sugestii personalizate pe baza unui set de întrebări pentru a determina genul muzical potrivit pentru utilizator și interacțiune socială, totul într-o platformă ușor de folosit și capabilă de a capta atenția oamenilor.

FestivalGo oferă posibilitatea utilizatorilor să exploreze lista festivalurilor disponibile, să-și exprime dorința de a participa, să completeze un chestionar interactiv pentru a-și identifica preferințele muzicale, să primească recomandări personalizate

bazate pe răspunsuri, să vizualizeze la fiecare festival harta cu punctele de interes (scenă, zone de mâncare, toalete, camping) și să interacționeze cu ceilalți utilizatori pentru a-și organiza cea mai bună experiență. Platforma se diferențiază față de alte aplicații prin chestionarele interactive unde utilizatorii răspund la douăzeci și unu de întrebări cu scopul de a determina genul muzical recomandat. Pe baza informațiilor primite, FestivalGo poate să ofere recomandări personalizate, selectate special pentru fiecare utilizator. Această funcționalitate ajută la îmbunătățirea experienței beneficiarului, diferențiind aplicația față de altele.

În următoarele capitole se va face un studiu bibliografic a principalelor tehnologii în domeniul dezvoltării aplicațiilor. Această analiză va forma baza teoretică a proiectului și va sprijini alegerile de design și implementare făcute în cadrul acestuia. Se vor expune tehnologiile cheie ce stau în baza creării aplicației. Se vor motiva selecțiile acestora și modul în care se utilizează. Pe parcursul analizei, se va studia în detaliu fiecare element principal al aplicației. În încheiere, se vor rezuma rezultatele obținute și se va scoate în evidență faptul că această abordare modernă, adaptabilă și interactivă este o soluție bună în contextul real al organizării festivalurilor.

1.2 Obiective

Lucrarea de față își propune dezvoltarea unei aplicații web full-stack, denumită FestivalGo, adresată utilizatorilor pasionați de festivaluri muzicale care își doresc o experiență digitală completă. Într-un peisaj în care majoritatea platformelor destinate evenimentelor pun accentul exclusiv pe vânzarea de bilete, FestivalGo vine cu o abordare centrată pe experiența utilizatorului.

Obiectivele acestei lucrări sunt:

1. Dezvoltarea unei aplicații full-stack orientate pe utilizator. Proiectarea aplicației pornește de la o analiză a nevoilor reale ale participanților la festivaluri, în special ale publicului tânăr. Scopul este crearea unei platforme moderne, cu o interfață intuitivă și funcționalități care să transforme procesul de organizare și participare la festival într-o experiență ușor captivantă.
2. Realizarea unei arhitecturi software clare și bine organizate. Aplicația va fi structurată pe mai multe niveluri, fiecare cu un rol clar, pentru a separa responsabilitățile între componente. Această abordare facilitează înțelegerea codului și oferă flexibilitate în cazul în care aplicația va fi extinsă pe viitor.
3. Implementarea autentificării și gestionarea rolurilor. Platforma va include un sistem de autentificare. Utilizatorii obișnuiți vor avea un profil personal cu participările lor, iar administratorii vor avea acces de gestiune a conținutului și moderare a întrebărilor din chestionar.
4. Integrarea unei hărți interactive personalizate pentru fiecare festival. Fiecare festival va avea o hartă interactivă unde vor fi marcate puncte de interes cum ar fi scenă, zonă de camping, zonă de mâncare, toalete. Această

- funcționalitate contribuie la o mai bună orientare în spațiu pentru participanți și îmbunătățește experiența în cadrul evenimentului.
5. Crearea unui chestionar muzical cu recomandări personalizare. FestivalGo include un chestionar cu douăzeci și unu de întrebări care analizează preferințele muzicale și trăsăturile de personalitate ale utilizatorului. Pe baza răspunsurilor, aplicația va genera genul muzical indicat pentru utilizator, apoi va afișa festivalurile recomandate. Administratorul va avea control asupra întrebărilor din chestionar.
 6. Implementarea unei componente sociale. Se dorește dezvoltarea unui sistem de comunicare între utilizatori prin formarea de grupuri de discuție în funcție de festivalurile alese de aceștia. Utilizatorii pot crea grupuri sau se pot alătura celor existente pentru a colabora în organizarea participării, de exemplu, cazare și transport.
 7. Integrarea unor funcționalități de logistică. Pentru a facilita organizarea, aplicația va permite butoane care vor redirecționa utilizatorul către pagini de căutare a transportului sau a cazării. Aceste funcții vor completa organizarea personalizată a utilizatorului.
 8. Crearea unei pagini principale interactive și dinamice. Pagina de start va conține secțiuni precum „Festivalul Lunii”, unde va fi afișat automat un festival cu data de începere cea mai apropiată de prezent. Vor fi incluse și recomandări generate pe baza chestionarului muzical complet de utilizator, oferind o experiență unică fiecărui utilizator.
 9. Implementarea unei funcționalități simple de plată pentru achiziția de bilete. Se urmărește implementarea unei metode de plată online care să permită utilizatorilor să își achiziționeze biletele direct din aplicație, într-un mod sigur și intuitiv.

1.3 Specificații

Pentru a dezvolta aplicația FestivalGo, se stabilesc un set de specificații care vor conduce la crearea unui program funcțional și ușor de utilizat. Partea de interfață se va implementa în Vue.js, iar partea de logică se va realiza în Java, folosind framework-ul Spring Boot. Baza de date pe care o vom folosi este MySQL deoarece este o soluție eficientă pentru modelarea și stocarea datelor aplicației.

Aplicația trebuie să permită înregistrarea și autentificarea utilizatorilor care trebuie să distingă utilizatorii obișnuiți și administratorii, prin atribuirea de roluri dedicate. Afișarea festivalurilor trebuie făcută sub formă categorii și să aibă posibilitatea de căutare. Utilizatorii vor avea posibilitatea de a vizualiza festivaluri, de a participa la festivaluri, de a cumpăra un bilet, de a vizualiza harta, de a răspunde la chestionar și de a crea grupuri de prieteni pentru a discuta.

Pe de altă parte, administratorii vor avea acces la funcții de gestionare a conținutului: adăugarea, editarea și ștergerea festivalurilor, editarea hărților și a punctelor de interes, gestionarea întrebărilor din chestionarul muzical.

Aplicația va include o funcționalitate de plată, care va permite utilizatorilor să achiziționeze bilete direct din platformă. Se intenționează integrarea unui serviciu de plată extern, precum Stripe, pentru a asigura o experiență de achiziție rapidă și sigură. După efectuarea acestei plăți, statusul participării utilizatorului va fi actualizat automat în baza de date.

Chestionarul muzical trebuie să fie stocat în baza de date și trebuie făcut pe baza răspunsurilor de tip „Da” sau „Nu”, fiecare răspuns având un scor distinct. Acesta va fi construit cu mare atenție pe baza unor analize profunde. Formularul va fi gestionat de administrator.

Pentru fiecare festival, se va atașa o hartă personalizată folosind biblioteca Leaflet. Utilizatorii vor putea vizualiza locațiile marcate (scenă, toalete, zone de mâncare etc.), într-un mod intuitiv. Administratorii vor fi singurii cu drept de acces pentru a adăuga, edita sau șterge punctele de interes pe hartă.

La nivelul interfeței se dorește obținerea unei interfețe cât mai simplu de utilizat, dar și plăcută vizual. Pe pagina principală se vor găsi elemente precum „Festivalul Lunii” și „Recomandările tale”. Se va utiliza un meniu pe pagina principală care va duce către celelalte pagini de interes.

Legat de performanță, se intenționează o executare rapidă a sarcinilor, spre exemplu, autentificarea, vizualizarea festivalurilor sau adăugarea de participări. Scopul este ca utilizatorul să nu perceapă timpul de așteptare, totul fiind realizat într-un interval de 1-2 secunde.

Pentru a stoca și gestiona datele, se intenționează folosirea unei baze de date relaționale, construită cu MySQL. Se va folosi pentru a modela cât mai clar aplicația, folosind entități precum User, Festival, Participare, Quiz, MapPoint, GroupChat, Message. Fiecare entitate dispune de atribute clare și sunt interconectate între ele. Spre exemplu, fiecare participare este asociată unui utilizator și unui festival.

În ceea ce privește structura codului, se optează pe utilizarea pachetelor pentru un cod mai organizat și ușor de gestionat. Fiecare pachet conține toate componentele specifice: Entitate pentru crearea datelor care se vor salva în baza de date, controller pentru gestionarea endpointurilor REST, repository pentru interacțiunea cu baza de date și service pentru a gestiona logica aplicației.

Ca element de securitate, parolele utilizatorilor vor fi stocate în baza de date și vor fi criptate folosind framework-ul Spring Security și algoritmul Bcrypt ceea ce împiedică stocarea parolelor în formă necriptată. Pentru a preveni atacurile de tip brute-force pe autentificare, se va implementa un mecanism care va bloca temporar un utilizator care a încercat de un număr prea mare de ori să se autentifice. Pentru a valida datele vom folosi adnotări specifice care previne trimiterea către baza de date a unor informații eronate. Însoțită de aceasta va fi și JPA care ajută la construirea de interogări mai sigure, reducând riscul atacuri de tip SQL Injection.

Pe partea de interfață, când un utilizator se va autentifica, acestuia i se vor salva în localStorage ID-ul, numele de utilizator și rolul. Se va evita extragerea parolei pentru a evita riscul de autentificare neautorizată. Paginile vor fi securizate în partea vizuală a aplicației, accesul către majoritatea paginilor fiind disponibilă doar dacă autentificarea a fost efectuată cu succes.

În versiunea actuală, aplicația web FestivalGo va prezenta o serie de limitări care vor putea fi abordate în versiunile viitoare. În primul rând, aplicația nu dispune de un sistem de notificări, ceea ce înseamnă că utilizatorii nu vor putea primi mesaje în timp real sau actualizări despre festivaluri. De asemenea, nu este disponibilă integrarea cu rețele sociale, astfel autentificarea cu Google, Facebook sau distribuirea rapidă a festivalurilor nu se va putea realiza. Aplicația nu oferă posibilitatea de a lăsa recenzii sau de a acorda o notă festivalurilor, ceea ce limitează interacțiunea și părerile directe între utilizatori. De asemenea, aplicația necesită conexiunea permanentă la internet.

2 Studiu bibliografic

Pentru realizarea aplicației FestivalGo s-au studiat multe articole științifice și documentații tehnice cu scopul de a identifica tehnologiile cele mai potrivite pentru dezvoltarea unei aplicații moderne și dinamice. Acest capitol detaliază principiile teoretice ce stau la baza arhitecturii client-server, analiza în detaliu a limbajelor de programare, framework-urile folosite și analiza unor articole științifice relevante care susțin din punct de vedere teoretic deciziile de proiectare a aplicației FestivalGO.

2.1 Analiza comparativă a platformelor existente

Pentru a înțelege mai bine contextul actual al aplicațiilor dedicate organizării și participării la festivaluri, a fost realizată o analiză comparativă a celor mai populare platforme internaționale din domeniu. Scopul acestei analize este de a identifica avantajele și limitările soluțiilor existente din perspectiva tehnologiilor utilizate, a experienței utilizatorului și a funcționalităților oferite. Această etapă este esențială pentru a evidenția elementele originale și creative propuse de aplicația FestivalGo.

Eventbrite este o platformă largă utilizată pentru gestionarea biletelor și promovarea evenimentelor, apreciată prin interfața sa simplă și logic organizată. Platforma permite organizatorilor să gestioneze eficient participanții și să promoveze evenimentele prin intermediul rețelelor sociale, oferind integrări solide cu aplicații precum Facebook și Salesforce [1]. Cu toate acestea, Eventbrite nu include elemente interactive avansate precum hărți live ale locațiilor sau mecanisme de tip chestionar și nici nu oferă posibilitatea comunicării directe între participanți într-un mod de chat.

Ticketmaster [2] este o platformă internațională specializată în distribuirea biletelor pentru evenimente live cum ar fi concerte, festivaluri, spectacole și evenimente sportive. Interfața principală a site-ului este organizată intuitiv, permițând utilizatorilor să filtreze rapid evenimentele în funcție de gen muzical, locație, interval de timp. În plus, sistemul oferă recomandări de evenimente considerate relevante pentru utilizator, în funcție de regiune sau preferințele anterioare. Achiziția biletului se face printr-un mecanism vizual ce permite selectarea locurilor direct de pe o hartă statică a sălii de eveniment. Cu toate acestea, Ticketmaster nu oferă o componentă de interacțiune socială între participanți și formare de grupuri, iar harta propusă este una statică.

O altă platformă de gestionare de evenimente este Eventim.ro [3], care oferă servicii complexe de cumpărare de bilete, special dedicată utilizatorilor din România. Cu o interfață bine organizată și aplicație mobilă dedicată, Eventim.ro facilitează achiziția de achiziționare a biletelor prin hărți statice ale sălilor și permite gestionarea comenzilor online. Totuși, în comparație cu FestivalGo, platforma nu integrează funcționalități moderne precum recomandări muzicale.

IaBilet.ro [4] este una dintre cele mai utilizate platforme din România, cu o prezență activă în domeniu muzicii live, stand-up comedy și festivaluri. Interfața site-ului este simplă și eficientă, permițând utilizatorului să filtreze evenimente în funcție de gen,

locatie, dată. Platforma oferă și o aplicație mobilă compatibilă cu IOS și Android, care permite accesarea biletelor direct din telefon. Totuși, IaBilet.ro [4] nu include funcționalități avansate precum recomandări muzicale, chestionar de identificare a preferințelor sau componente sociale.

2.2 Analiza articolelor științifice relevante pentru crearea aplicației FestivalGo

Pentru a înțelege mai bine contextul social și tehnologic al aplicației FestivalGo, au fost analizate mai multe lucrări științifice relevante din domeniul evenimentelor, tehnologii mobile și al organizării de festivaluri muzicale. Aceste studii oferă o bază teoretică care va sta la baza deciziilor de design și funcționalităților aplicației.

Primul articol analizat este realizat de Skandalis, Banister și Byrom [5], care studiază experiența participanților la festivalul Primavera Sound din Barcelona. Autorii propun conceptul de „autenticitate spațială” [5], susținând că publicul nu caută o evadare total din realitate, ci o experiență autentică în viața urbană. Festivalurile urbane devin astfel spații hibride, unde viața de zi cu zi se îmbină cu muzica. Aplicația FestivalGo, prin funcționalitățile sale de hartă interactivă și recomandări personalizate sprijină această viziune, contribuind la o integrare mai firească a evenimentelor în rutina zilnică a utilizatorilor.

Cel de-al doilea articol studiat este scris de Luxford și Dickinson [6], care oferă o perspectivă practică asupra modului în care aplicațiile moderne influențează experiența oamenilor la festivaluri de muzică. Studiul este bazat pe grupuri și analiza aplicațiilor existente în Marea Britanie. Articolul dezvăluie că participanții apreciază aplicațiile care le oferă libertate și control asupra propriei experiențe, cum ar fi program personalizat, hărți detaliate, posibilitatea de a comunica cu alți participanți. De asemenea se subliniază importanța echilibrului între funcționalitate și spontaneitate. Un punct important este evidențierea fazei de anticipare: aplicațiile sunt folosite cu mult timp înainte de festival pentru a planifica participarea, crescând entuziasmul utilizatorilor și implicarea în experiență. Aplicațiile trebuie să funcționeze rapid și să ofere acces minim, recomandându-se un design centrat pe utilizator [6].

Alt studiu relevant pentru dezvoltarea funcționalităților sociale ale aplicației FestivalGo este realizat de Rentfrow, GoldBerg și Levitin [7], care examinează modul în care preferințele muzicale pot afecta atracția interpersonală și formarea de conexiuni sociale. Studiile lor au arătat că oamenii care împărtășesc gusturile muzicale similare au tendința de a se simți compatibili, chiar dacă nu s-au cunoscut niciodată [7]. Această concluzie este foarte relevantă atunci când vine vorba de dezvoltare funcțiilor sociale în aplicația FestivalGo. Aplicația ajută la formarea de comunități și personalizează utilizatorilor prin crearea de grupuri între utilizatori cu preferințe muzicale comune.

Pentru a implementa chestionarul muzical, au fost luate în considerare câteva articole din domeniul psihologiei muzicii. Aceste informații sunt foarte utile în dezvoltarea preferințelor muzicale destinate utilizatorilor platformei web FestivalGo.

Una dintre cele mai influente lucrări din domeniu este realizată de Rentfow și Gosling [8], care explorează legătura dintre trăsăturile de personalitate din modelul Big Five și preferințele pentru anumite stiluri muzicale. După ce au analizat răspunsurile a sute de participanți, au ajuns la concluzia că răspunsurile muzicale nu sunt aleatorii, ci sunt, mai degrabă, legate de anumite caracteristici de personalitate [8].

Autorii împart genurile muzicale în patru mari categorii: cele complexe fiind genul jazz, blues și muzică clasică, cele intense fiind rock, metal sau alternativ, cele convenționale și ritmate fiind pop sau country, iar cele energetice fiind dance, hip-hop și electronic. Studiul arată că persoanele cu o deschidere mare către experiențe preferă muzica sofisticată, iar extrovertiți se orientează către muzica energetică și ritmată [8].

Prin urmare, articolele analizate oferă un cadru teoretic solid și actual, care susțin relevanța dezvoltării aplicației FestivalGo.

2.3 Tehnologii folosite pentru crearea backend-ului aplicației FestivalGo

Pentru realizarea serverului aplicației au fost analizate diverse concepte esențiale din limbajul de programare Java. Scheletul principal ales este Spring Boot, un framework moder destinat dezvoltării rapide a aplicațiilor Java.

Limbajul Java este descris formal în cartea „The Java Language Specification, Java SE 17”, scrisă de James Goslin, Bill Joy, Guy Steele, Gilad Bracha și Alex Buckley [9], acesta fiind un limbaj de programare modern, centrat pe conceptele de clasă și obiect, creat pentru a fi ușor de învățat și utilizat de către programatori [9]. Conform Capitolului 3 din documentația oficială [9], structura limbajului Java păstrează unele asemănări cu C și C++, dar oferă o structură mai clară și poate fi extins cu caractere Unicode pentru a facilita scrierea codului pe platforme și limbi diferite.

În limbajul Java, conform Capitolului 4 din documentația oficială [9], datele sunt împărțite în două mari categorii principale: cele primitive și cele referențiale. Tipurile primitive includ valori numerice întregi în reprezentare (int și long), numerele zecimale (double sau float), cele de tip caracter Unicode (char), iar cele de tip boolean (adevărat sau fals). Acestea sunt implementate pentru a fi identice pe toate platformele și implementările Java.

Pe de altă parte, în Capitolul 4 din documentația oficială [9], tipurile referențiale sunt reprezentate de clase, interfețe și tablouri. Obiectele și tablourile sunt reprezentate dinamic și sunt gestionate prin referințe, astfel încât mai multe referințe pot reprezenta același obiect.

În Capitolul 7 al lucrării [9], se descoperă faptul că limbajul Java permite organizarea programelor în pachete și module. Pachetele sunt folosite pentru gruparea claselor și interfețelor, permițând o structurare clară a numelor și accesului. Acest sistem permite aplicațiilor să fie ușor de citit și bine structurate.

Limbajul Java se diferențiază de alte limbaje de programare prin conceput de moștenire simplă. Conform Capitolului 8 din documentația oficială [9], fiecare clasă poate avea o singură superclasă directă. Prin intermediul acestui mecanism, dezvoltatorii pot crea clase noi care extind comportamentul claselor existente și pot folosi polimorfismul pentru a interacționa cu obiecte diferite într-o manieră uniformă.

Spring Boot este un framework proiectat de echipa Spring care a fost gândit pentru a ușura dezvoltarea aplicațiilor Java. Conform documentației oficiale [10], acest framework elimină setările repetitive care apar în dezvoltarea cu Spring Framework, oferind astfel un mediu de dezvoltare rapid și ușor de utilizat.

Unul dintre avantajele fundamentale ale folosirii Spring Boot este capacitatea sa de a se auto-configura. Acest mecanism detectează automat dependențele aflate în calea către clasele Java și configurează toate componentele necesare. Astfel, dezvoltatorul nu mai pierde timpul pe configurarea programului, ci se poate concentra pe logica și implementarea aplicației [10].

Un beneficiu oferit de Spring Boot este integrarea automată a unui server web precum Tomcat, ceea ce permite aplicația să fie pornită direct, fără a fi necesar un server instalat separat. Mai precis, o aplicație Spring Boot se compilează printr-un fișier .jar executabil, care conține toate dependențele și componentele necesare, inclusiv serverul web. Astfel, rularea aplicației devine simplă [10].

Împreună cu Spring, se va folosi și Apache Maven care este un instrument ce ajută la construirea și administrarea proiectelor software din mediul Java. Acesta înseamnă acumulator de cunoștințe și a fost creat inițial pentru a ușura procesele de construire a proiectelor Jakarta Turbine, unde existau multe fișiere de tip Ant diferite și metode de gestionare a dependențelor variabile. Scopul acestuia este de a oferi o metodă clară de definire a proiectului, de a partaja dependențele JAR între proiecte și de a facilita publicarea informațiilor despre proiect [11].

Un avantaj important al framework-ului Spring Boot este includerea unui server web integrat, care permite rularea aplicației fără a fi necesară instalarea sau configurarea unui server extern. Odată adăugarea dependenței spring-boot-starter-web, aplicația esre pregătită să răspundă solicitărilor HTTP, folosind în mod implicit Apache Tomcat [12].

În mod implicit, aplicațiile codate în Spring Boot și Tomcat rulează pe portul 8080 atunci când sunt pornite. Acest lucru poate fi schimbat foarte ușor prin editarea fișierului de configurare numit application.properties [12]. Această setare permite rularea pe portul specificat fiind utilă atunci când dorim rularea a mai multor aplicații. Pentru aplicația noastră vom folosi portul 8081.

În plus, Spring Boot oferă opțiunea de folosire a altor servere web în locul Tomcat-ului. Documentația oficială [12] menționează și alte alternative, cum ar fi Jetty sau Undertow.

În aplicațiile moderne se implementează adesea arhitectura pe straturi care se va folosi în aplicația platforma propusă. Aceasta presupune separarea logicii în bucăți distincte cu responsabilitate clară: Controller – Service – Repository – Database.

Acest tip de organizare respectă principiul Separation of Concerns (SoC), un concept fundamental în ingineria software [13]. Conform articolului publicat pe Medium [13], principiul SoC vine cu multiple avantaje. Mentenabilitatea crește, modificările pot deveni mai ușor de realizat. Se îmbunătățește testarea unitară, spre exemplu, testarea serviciilor se poate face fără a accesa baza de date. Scalabilitatea crește, fiecare strat poate fi refolosit pentru aplicații diferite [13].

În documentația oficială Spring Integration [14], se subliniază și aici importanța separării în straturi. Spring recomandă această arhitectură ca parte a bunelor practici în dezvoltarea aplicațiilor modulare și robuste [14]. Această structură asigură dezvoltarea aplicației pe termen lung.

Pentru realizarea unei aplicații web moderne este necesară utilizarea componentelor puse la dispoziție de modulul web. Acesta acoperă posibilitatea creării de servicii web prin arhitectura clasică Model-Vedere-Controller (MVC), cât și posibilitatea dezvoltării unor servicii web conform principiilor REST [15]. Atunci când se folosește Spring Boot, adăugarea componentelor de bază pentru web activează automat configurațiile necesare pentru funcționalitatea unui server web, precum Tomcat, care va prelua cererile utilizatorilor și le va redirecționa către zonele corespunzătoare a aplicației în funcție de codul scris [12].

Un element central în această arhitectură este partea responsabilă de gestionarea cererilor, adică codul care stabilește cum răspunde aplicația la diferite solicitări venite din interfață. În loc să fie trimise pagini vizuale, răspunsurile se vor trimite în format JSON. Această abordare este extrem de comună în aplicațiile moderne deoarece permite separarea între partea de utilizator și partea logică [15]. Platforma Spring se ocupă automat de transformarea obiectelor interne ale aplicației în structuri JSON, eliminând nevoia de configurări complicate din partea dezvoltatorului. Această funcționalitate este posibilă datorită bibliotecii de integrare specializate în manipularea acestui tip de format [12].

Pentru a organiza răspunsurile aplicației în funcție de cererea utilizatorului, Spring pune la dispoziție mecanisme de rutare. Acest sistem de legătură dintre adrese și logica aplicației este foarte flexibil și permite inclusiv filtrarea în funcție de metoda folosită de utilizator [16]. Totodată, datele venite din adresă sau din bara de căutare pot fi preluate ușor din interiorul aplicației fără cod suplimentar [16].

În proiectele actuale, unde interfața aplicației este adesea construită separat, este necesar ca acea interfață să poată trimite cereri către serverul Spring. Însă, din motive de securitate, browserele moderne nu permit schimbul automat de date care rulează pe adrese diferite. Pentru a rezolva această problemă, Spring oferă o funcționalitate nouă, CORS (Cross-Origin Resource Sharing), care permite selectiv acceptarea acestor cereri

externe. Se pot stabili reguli clare cu privire la ce aplicații au voie să trimită cereri și pentru ce tipuri de operațiuni. [17]

Spring Data JPA este un modul important din cadrul Spring, deoarece este conceput pentru a simplifica interacțiunea aplicațiilor Java cu baza de date. Aceasta este integrată direct cu Java Persistence API (JPA), simplificând astfel nevoia de a scrie cod pentru salvarea, căutarea și ștergerea entităților [18]. Prin urmare, programatorii nu trebuie să scrie cod manual pentru acces la baza de date, Spring JPA generând automat implementări pentru interfețele de acces la date și permițând focalizarea pe logica de implementare. De asemenea, acest framework standardizează gestionarea erorilor [18].

Conceputul central introdus de Spring Data JPA este repository-ul asociat unei entități din domeniul aplicației. Acesta definește o interfață specifică unei clase de domeniul indicându-se astfel framework-ului ce entitate urmează să fie administrată și cum este identificată aceasta [19]. În cazul în care se dorește disponibilitatea operațiunilor standard de bază pentru entitatea respectivă, se va extinde direct interfața care le furnizează, în locul interfeței marker minimale [19].

CRUD este acronimul pentru operațiile fundamentale de creare, citire, actualizare și ștergere. Spring Data JPA pune la dispoziție o interfață care include aceste operații de bază, pe care o putem moșteni pentru a obține setul complet de funcționalități [19]. Există, de asemenea, mai multe variante a acestei interfețe, una dintre cele mai importante fiind varianta cu liste, unde a fost introdusă o variantă a interfeței CRUD care, pentru metode ce folosesc mai multe rezultate, acestea se întorc într-o listă, ceea ce face prelucrarea mai multor rezultate într-un mod convenabil [19].

Pentru metode mai avansate, scrierea interogărilor este posibilă cu ajutorul adnotării @Query. Aceasta se folosește atunci când o interogare este mai complexă sau când se dorește folosirea JPQL (Java Persistence Query Language) [19].

Odată definite repository-urile, gestionarea tranzacțiilor este foarte simplă pentru dezvoltator deoarece Spring Data JPA aplică implicit un component tranzacțional pe modelele furnizate din aceste repository-uri. Operațiile care doar citesc date vor fi configurate automat să ruleze doar pentru citire, adică read-only, iar cele care modifică date, precum ștergerea sau salvarea, sunt configurate ca să ruleze într-o tranzacție standard [20].

Pentru securitatea aplicației FestivalGo se va folosi implementarea framework-ului Spring Security. Acesta oferă mecanisme de autentificare, autorizare și protecție împotriva atacurilor comune, fiind considerat standardul facto pentru securizarea aplicațiilor Spring [21]. Autentificarea reprezintă procesul de verificare a identității utilizatorului ce solicită acces la o resursă protejată, de regulă prin furnizarea unui nume de utilizator și a unei parole [21].

Pentru protejarea parolelor utilizatorilor, Spring Security stochează parolele folosind algoritmi de hash, astfel încât să nu fie salvate sub formă de text clar. Până la versiunea 5.0, codificatorul implicit accepta parole necodificate, care este o abordare

nesigură. În prezent, unul dintre cei mai buni algoritmi de criptare este Bcrypt, adoptat implicit de Spring Security pentru stocarea credențialelor [21].

Bcrypt face parte din categoria funcțiilor unidirecționale adaptive, ceea ce înseamnă că acesta utilizează o valoare aleatoare unică pentru fiecare parolă și permite ajustarea dimensiunii hash-ului [21]. Bcrypt generează pentru fiecare parolă o valoare unică, astfel chiar dacă doi utilizatori aleg aceeași parolă, hash-urile vor fi diferite, fiind ineficiente atacurile de tip tabel precalculat de parole [21].

De asemenea, Bcrypt este conceput pentru a fi intenționat lent în procesarea hash-urilor, ceea ce îngreunează atacurile de tip forță brută [21]. Un timp mai mare de calcul înseamnă că un atacator poate testa mult mai puține parole pe secundă. Implementarea standard din Spring Security folosește un factor de cost implicit egal cu zece, însă documentația oficială recomandă creșterea acestui factor, astfel încât verificarea parolei să dureze o secundă pe sistemul folosit [21]. Prin urmare, utilizarea algoritmului Bcrypt pentru criptarea parolelor asigură un nivel înalt de securitate al autentificării într-o aplicație Spring.

Pentru a crește securitatea aplicației FestivalGo, se va implementa un mecanism pentru limitarea ratei la autentificare. Se va alege una dintre cele mai eficiente soluții pentru implementarea limitării ratei este Bucket4j. Conform documentației oficiale [22], acest algoritm se bazează pe o „găleată virtuală” care conține un număr finit de token-uri. Fiecare cerere făcută de către un client consumă câte un token. Dacă această găleată rămâne fără token-uri atunci cererile suplimentare sunt refuzate un anumit timp până la reumplerea găleții. Această strategie permite controlul asupra numărului de cereri pe un anumit interval de timp și este recomandată pentru protecție la nivel de API [22].

Spring Boot oferă o integrare simplă cu bazele de date SQL și un mod de lucru cât mai eficient și mai rapid de configurare. Spring Boot detectează automat driverul JDBC disponibile și configurează automat sursa de date. Astfel, driverul este disponibil, iar proprietățile se pot seta din fișierul application.properties [23]. Acesta suportă, de asemenea, configurări avansate ale sursei de date, spre exemplu conexiunea la serverul Tomcat.

Spring Boot permite inițializarea automată a bazei de date folosind fișiere SQL dedicate. În documentația [23], directorul src/main/resources va avea fișierele standard schema.sql și data.sql, iar acestea vor fi executate la pornirea aplicației.

Spring Boot poate gestiona multiple conexiuni la baze de date distincte, oferind opțiunea configurării unor DataSource diferite, fiecare cu propria configurație [23].

Testarea funcționalităților backend ale unei aplicații web presupune adesea verificarea API-urilor care stau la baza comunicării dintre componente. Postman oferă o interfață prietenoasă care permite trimiterea facilă a cererilor HTTP (GET, POST, PUT DELETE) către API și inspectarea răspunsurilor primite [24]. În interfață se afișează datele esențiale ale răspunsului: cod de stare, timp de răspuns, conținutul și dimensiunea acestuia, ajutând persoana care testează codul să-i fie mai ușor [24].

2.4 Tehnologii folosite pentru crearea frontend-ului aplicației FestivalGo

După examinarea acestor componente esențiale în construirea backend-ului, se vor analiza tehnologiile principale utilizate pe partea de client. În centru acestei componente se afla JavaScript, limbajul de programare care servește drept fundament pentru interfața aplicației.

JavaScript, folosit atât pentru aplicații cu interfață grafică pentru utilizator , cât și pentru logica serverului, este printre cele mai populare limbaje de programare pentru dezvoltarea aplicațiilor web. Conform documentației oferite de Mozilla Developer Network [25], JavaScript este un limbaj multi-paradigmă, care permite abordări atât imperative, cât și funcționale sau orientate pe obiect. Sintaxa limbajului este inspirată din C, iar acesta este cel mai potrivit pentru dezvoltarea de aplicații dinamice în browser [25]. JavaScript este un limbaj interpretat și dinamic tipizat, adică variabilele nu sunt legate de un tip de date, iar acest lucru oferă o libertate mare în scrierea codului [25].

Limbajul este definit și standardizat de ECMA International prin specificația ECMAScript [26]. Standardul ECMAScript este esențial pentru uniformizarea limbajului JavaScript în diferitele medii de execuție, cum ar fi browserele și serverele Node.js, astfel încât aplicațiile să funcționeze fără probleme. ECMA a fost, de asemenea, cea care a contribuit la crearea unor standarde importante legate de limbajele de programare și arhitecturile informatice. ECMA servește ca punte de legătură între caracteristicile JavaScript și alte tehnologii cu care interacționează, cum ar fi aplicațiile Java, în care comunicarea frontend-backend se realizează de obicei prin API-uri REST sau WebSocket [26].

Aceste afirmații sunt susținute de documentația oficială oferită de MDN Web Docs [25] care descrie JavaScript ca fiind un limbaj este capabil să gestioneze operații asincrone folosind un model bazat pe event loop și call stack. Conform MDN, JavaScript este ușor de învățat pentru începători, dar suficient de complex pentru aplicații mari [25].

Un aspect esențial în JavaScript este modelul de obiecte bazat pe moștenire prin prototipuri, care diferențiază acest limbaj de programare de alte limbaje clasice orientate pe obiect. Obiectele în JavaScript pot fi create dinamic și extinse cu proprietăți noi în timpul rulării, ceea ce oferă o mare flexibilitate , dar necesită înțelegerea clară a execuției și a moștenirii [26].

Pentru construirea interfeței aplicației noastre vom folosi framework-ul Vue.js, care este bazat pe JavaScript. Acesta permite dezvoltarea de componente, favorizând astfel un design modular, reutilizabil și scalabil al aplicației.

Vue.js este un framework pentru dezvoltarea interfețelor de utilizator și este bazat pe HTML, CSS și JavaScript. Acesta este conceput astfel încât să poată să fie adaptat treptat, pornind de la o pagină HTML simplă până la dezvoltarea unor aplicații complete de tip Single Page Application (SPA), prin utilizarea unor instrumente moderne precum Vue Router și alte biblioteci suport [27].

HTML sau denumit HyperText Markup Language, reprezintă limbajul fundamental utilizat pentru a structura conținutul web. Acesta se folosește pentru organizarea elementelor în pagină prin utilizarea unor etichete [28]. Este important de menționat că HTML este un limbaj de tip declarativ, nu de programare, ceea ce îl face ușor de înțeles și de utilizat pentru dezvoltatorii la început de drum.

În contextul FestivalGo, HTML va fi implementat în șabloanele componentelor Vue, formând scheletul paginii vizuale. Împreună cu JavaScript și HTML, vom folosi CSS pentru stilizarea paginilor.

Vue Router este folosită în cadrul aplicației FestivalGo pentru a gestiona navigarea între paginile aplicației prin crearea unui sistem de rute care se potrivește cu elementele vizuale ale aplicației [29]. Conform documentației oficiale, Vue Router oferă mecanisme pentru rute dinamice și protecție a rutelor [29]. Aplicațiile mari necesită acest lucru, cum ar fi aplicația FestivalGo, care are mai multe funcționalități pe roluri, ceea ce permite limitarea accesului la anumite rute în funcție de starea autentificării.

Vue.js se concentrează pe partea vizuală, această abordare fiind utilă în integrarea sa cu alte biblioteci existente. Cu toate acestea, Vue este capabil să susțină aplicații complete, menținând în același timp simplitatea de bază, atunci când este utilizat împreună cu ecosistemul complet de instrumente. Acesta utilizează un sistem reactiv intern care urmărește automat starea aplicației și actualizează DOM-ul virtual atunci când datele se modifică [27].

Arhitectura Vue.js se bazează pe componente. Acestea comunică între ele prin mecanisme precum props și emit, oferind o separare clară a responsabilităților. Vue permite organizarea de componente sub formă de fișiere, care înglobează HTML, logica JavaScript și stilurile CSS într-o singură unitate funcțională [27].

În cadrul proiectului propus se va folosi versiunea a 3-a deoarece este mai performantă și introduce o alternativă la opțiunea de API. Această abordare nouă permite o reutilizare mai avansată a logicii și o organizare mai bună prin definirea funcțiilor în blocuri clare [27]. De asemenea, Vue oferă o integrare facilă a altor biblioteci și instrumente moderne cum ar fi Axios, folosit pentru cererile HTTP sau Leaflet, pentru integrarea unor hărți interactive.

Pe partea de interfață, se alege utilizarea Vite ca tool principal pentru dezvoltare și creare, datorită performanțelor sale și integrației excelente cu Vue. Vite se distinge prin viteza sa excepțională în faza de dezvoltare, oferind un server de dezvoltare instantaneu, care permite ca modificările din cod să fie imediat observate în interfață, fără a reîncărca întreaga aplicație [30].

Acest comportament este posibil deoarece Vite separă dependențele externe de codul sursă. Dependențele sunt pre-incluse cu ajutorul unui compilator care este 10 - 100 de ori mai rapid decât alte compilatoare obișnuite [30]. În același timp, fișierele sursă sunt servite direct browser-ului ceea ce reduce semnificativ timpul de pornire al serverului [30].

Pentru producție, Vite se folosește de Rollup, un mecanism care generează pachete optimizate, folosindu-se de tehnici precum tree-shaking, code splitting și cache eficient [31]. Acest proces duce la redimensionarea aplicației și la îmbunătățirea performanței.

Vite conține un suport nativ pentru TypeScript, JSX și CSS, ceea ce elimină necesitatea configurării suplimentare [32]. Totodată, Vite este extensibil printr-un sistem de plugin-uri compatibile cu cele de la Rollup, ceea ce îl face ușor de adaptat la cerințele fiecărui proiect [32].

Pentru comunicarea dintre interfață și server se va folosi biblioteca Axios. Axios este o bibliotecă din familia JavaScript foarte populară care este folosită pentru efectuarea de cereri HTTP asincrone, în special pentru aplicații frontend care sunt construite cu framework-uri moderne precum Vue.js și React. Axios permite trimiterea de cereri GET, POST, PUT, DELETE către un server web într-un mod simplu, clar și eficient. Manipularea automată a conversiilor JSON, precum suportul pentru începători și gestionarea erorilor sunt caracteristici importante [33].

Una dintre cele mai utilizate biblioteci open-source pentru crearea de hărți interactive pe web este Leaflet. Biblioteca, care a fost concepută inițial de Vladimir Agafonkin, oferă o interfață prietenoasă, performanță ridicată și compatibilitate extinsă cu browserele și dispozitivele mobile moderne [34]. Leaflet folosește surse de date gratuite, cum ar fi OpenStreetMap, și permite interacțiuni personalizate [34]. Leaflet este utilizat în aplicația FestivalGo pentru afișarea hărților festivalurilor și pentru a adăuga marcaje specifice, cum ar fi scena, zonele de mâncare, camping sau toalete.

Această bibliotecă permite inserarea coordonatelor GPS reale și inserarea de emoji-uri sau icoane personalizate pe hartă. Leaflet oferă o experiență interactivă fără a afecta viteza de încărcare sau compatibilitatea datorită suportului pentru stiluri și evenimente dinamice. Leaflet este o opțiune foarte bună pentru proiectele care au nevoie de interacțiune vizuală [35].

Achiziționarea de bilete se va face cu ajutorul librăriei Stripe. Stripe este o platformă de procesare a plăților bine recunoscută la nivel global, disponibilă în zeci de țări, adesea fiind preferată de dezvoltatori datorită acoperirii sale extinse [36]. Sistemul de plăți oferite de Stripe permite comercianților să accepte plăți prin internet într-un mod securizat și ușor de integrat. API-ul Stripe este construit conform principiilor REST, cu URL-uri previzibile, cereri form și răspunsuri JSON, folosind coduri HTTP standard și metode de autentificare bine cunoscute [37]. De asemenea Stripe oferă o gamă largă de metode de plată, inclusiv carduri și portofele electronice cum ar fi Apple Pay, iar datele sensibile sunt protejate cu ajutorul token-ului direct în browser, astfel încât aceste informații să nu ajungă niciodată server [38].

Integrarea Stripe, în aplicațiile web, se realizează de obicei fie prin fluxul de plată integrat în pagină, fie redirectionarea către o altă pagină de plată dispusă de Stripe [36]. Varianta redirectionată propune o abordare mai simplă în care aplicația web inițiază procesul de plată prin crearea unei sesiuni de checkout folosind API-urile Stripe, apoi

clientul este direcționat către o pagină Stripe securizată care colectează datele de plată și finalizează tranzacția [36].

2.5 Tehnologii folosite pentru crearea bazei de date a aplicației FestivalGo

MySQL este un sistem de gestiune a bazelor de date relațional open-source, considerat cel mai popular SGBD de acest tip la nivel mondial [39]. Dezvoltat și susținut de Oracle Corporation, MySQL permite stocarea și administrarea eficientă a datelor folosind limbajul standardizat SQL. Bazele de date MySQL sunt de tip relațional, ceea ce înseamnă că datele sunt organizate în tabele separate, iar între acestea se pot defini relații logice [39]. Astfel de relații sunt implementate de obicei prin intermediul cheilor externe, care permit menționarea datelor dintr-o tabelă în alta și asigură consistența informațiilor între tabele folosind constrângeri [40]. Un mare avantaj al folosirii MySQL este natura open-source, ceea ce înseamnă că software-ul poate fi utilizat gratuit.

MySQL rulează pe o arhitectură de tip client-server, componenta centrală fiind serviciul MySql Server care gestionează stocarea datelor, tranzacțiile și execuția interogărilor. MySQL Server este recunoscut pentru viteza, scalabilitatea și fiabilitatea sa, putând rula confortabil atât pe calculator personal cât și pe un server dedicat [39]. Acest motor de baze de date a fost inițial creat pentru a gestiona volume mari de date mai rapid decât alte soluții existente și este folosit cu succes de ani de zile în medii de producție solicitante [39].

MySQL Workbench este o aplicație grafică unificată care oferă instrumente vizuale pentru proiectarea, dezvoltarea și administrarea bazelor de date MySQL [41]. Workbench permite atât modelarea vizuală a bazei de date, spre exemplu crearea diagramei entitate - relație și generarea schemei SQL corespunzătoare [41], cât și scrierea și optimizarea interogărilor SQL prin intermediul unui editoriu avansat. De asemenea, include o consolă administrativă, precum și monitorizarea stării serverului [41].

Baza de date relațională MySQL se potrivește mai bine pentru o aplicație de tip FestivalGo datorită naturii datelor și cerinței de integrare și consistență ridicată. Bazele de date SQL excelează în gestionarea datelor structurate și relaționale, menținând integritatea tranzacțională și permițând interogări complexe, pe când bazele NoSQL sunt optimizate pentru datele nestructurate și scalabile orizontală masivă [42]. În contextul FestivalGo, unde se pune accentul pe corectitudinea și logica datelor mai degrabă decât pe scalarea la mii de utilizatori simultan, avantajele noSQL precum scalabilitatea și disponibilitatea foarte înalte nu sunt esențiale [43].

Puterea interogărilor SQL reprezintă un avantaj major. MySQL permite realizarea interogării complexe care reunes date din mai multe tabele, ceea ce este esențial pentru a extrage informații corelate. Bazele de date noSQL, în schimb, nu oferă în mod obișnuit suport pentru JOIN-uri complexe sau interogări care implică mai multe colecții, ceea ce le face mai puțin potrivite când există necesitatea unor interogări relaționale complexe [43].

Prin urmare, datorită integrității datelor, consistenței, modelării relaționale și puterii limbajului SQL în manipularea datelor interconectate, MySQL este alegerea cea mai potrivită pentru aplicația FestivalGo.

3 Analiză, proiectare, implementare

Acest capitol reprezintă esența lucrării și detaliază procesul de dezvoltare a platformei web FestivalGo, de la faza de analiză inițială și proiectare, până la implementarea concretă și testarea funcționalităților. În timp ce primele capitole au oferit o descriere a contextului teoretic și a tehnologiilor utilizate, această secțiune oferă, în detaliu, tehnicile folosite pentru a crea această platformă web. Se vor prezenta arhitectura sistemului, metodologia utilizată, etapele de dezvoltare și deciziile tehnice care au dus la implementare.

3.1 Arhitectura generală a platformei

Aplicația FestivalGo este o platformă web dezvoltată cu scopul de a oferi utilizatorilor o modalitate intuitivă și interactivă de a descoperi festivalurile muzicale, de a interacționa cu alți participanți și de a-și gestiona participările. Arhitectura aplicației este de tip client-server, fiind structurată în componente distincte, fiecare cu rolul său bine definit.

În figura următoare (Figura 3.1.1) este ilustrată schema arhitecturală generală a aplicației FestivalGo, realizată conform principiilor de separare a responsabilităților, așa cum au fost detaliate și în studiul bibliografic. Arhitectura adoptată este de tip client-server, cu o împărțire clară între interfața utilizatorului, logica de server, stocarea datelor în baza de date și serviciile externe care completează funcționalitățile sistemului.

Interfața grafică, așa cum este menționat în studiul bibliografic, este realizată cu ajutorul framework-ului Vue.js care oferă o soluție modernă pentru dezvoltarea aplicațiilor de tip SPA (Single Page Application) [27]. În FestivalGo, frontend-ul aplicației este împărțit în două mari categorii: partea de client și partea de administrator. Partea clientului este responsabilă cu navigarea între pagini, vizualizarea festivalurilor, interacțiunea între grupuri, vizualizarea hărților interactive, vizualizarea chestionarului și crearea participărilor sau cumpărarea biletelor. Partea administratorului este responsabilă de adăugarea, ștergerea și actualizarea festivalului și gestionarea întrebărilor din chestionar. Datele sunt transmise prin cereri HTTP către server, iar aplicația folosește localStorage pentru a păstra datele esențiale precum ID-ul, username-ul și rolul său. Autentificarea și înregistrarea sunt singurele pagini comune pentru client și administrator.

Serverul aplicației este implementat în Java folosind framework-ul Spring Boot, tehnologie detaliată anterior în capitolul doi. Serverul expune o serie de endpointuri prin care gestionează principalele funcționalități: autentificare, festivaluri, participări, salvarea rezultatelor chestionarelor și conversațiile din grupuri. Securitatea este asigurată de Spring Security, iar parolele sunt criptate cu Bcrypt, oferind un nivel crescut de protecție a parolelor personale.

Conform principiilor relaționale reprezentate în studiul bibliografic, aplicația utilizează un sistem de gestiune a bazelor de date MySQL, unde sunt definite tabele

precum: user, question, answer, festival, map_point, group_chat, group_chat_member, participare, message. Legăturile dintre tabele platformei FestivalGo sunt implementate în codul serverului prin intermediul JPA și gestionate cu ajutorul repository-urilor Spring Data.

Pentru a extinde funcționalitatea aplicației, s-au implementat mai multe servicii externe precum Stripe și Leaflet. Stripe a fost folosit pentru procesarea plăților, care permite utilizatorilor să cumpere bilete pentru festivaluri. După confirmarea plății, backend-ul actualizează statusul participării în baza de date. Leaflet s-a folosit pentru integrarea hărților și marcarea punctelor de interes. Cererile de vizualizare sau editare a hărții sunt inițiate din interfață, iar serverul acționează ca intermediar între partea vizuală și baza de date.

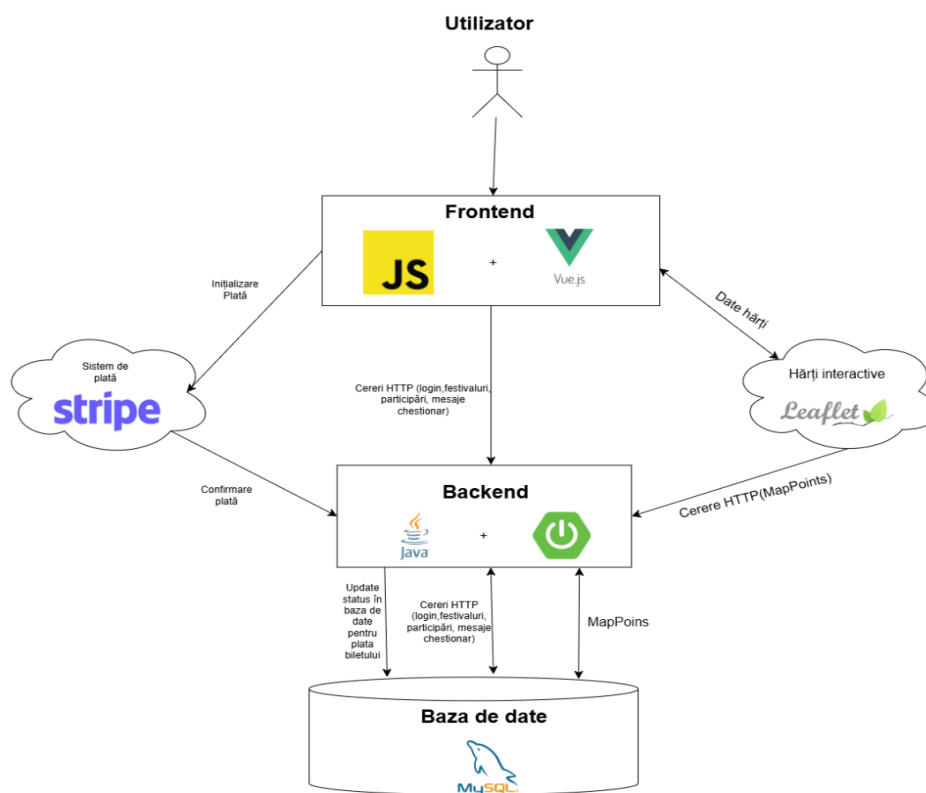


Figura 3.1.1: Schema generală

Schema generală oferă o imagine de ansamblu asupra componentelor principale și a modului în care acestea interacționează între ele. Prin această reprezentare, s-a evidențiat clar separarea responsabilităților dintre părțile componente, permițând o dezvoltare adaptabilă.

3.2 Structura logică a platformei și organizarea modulelor

Proiectarea aplicației FestivalGo este realizată pe baza unei abordări structurate și progresive, în care fiecare etapă este gândită pentru a răspunde unei nevoi reale identificate în faza de analiză. Aplicația se adresează atât utilizatorilor obișnuiți (care

doresc să exploreze și să participe la festivaluri), cât și administratorilor (care gestionează conținutul platformei), astfel a fost necesară definirea clară a funcționalităților specifice fiecărui rol, precum și a restricțiilor aferente acestora.

Primul pas constă în stabilirea cerințelor aplicației din punct de vedere a celor două tipuri principale de utilizatori: utilizator și administrator.

1. Utilizatorul are voie să își creeze cont și se autentifică, să exploreze festivaluri și hărți asociate, răspunde la chestionarul de recomandare muzicală, se poate înscrie la festivaluri și poate plăti biletul sau să creeze și să participe la grupuri de discuție.
2. Administratorul are dreptul la funcții mai limitate față de client, acesta având voie să creeze, editeze sau șterge festivaluri, să încarce poze asociate festivalurilor, să definească hărțile sau să gestioneze întrebările din quiz și punctele de pe hartă. Rolurile au fost apoi reflectate atât în baza de date, cât și în comportamentul aplicației din interfață.

Rolurile au fost selectate atât în baza de date cât și în comportamentul aplicației în componenta vizuală unde, în funcție de rolul identificat după autentificare, persoana în cauză este redirecționată automat către interfața corespunzătoare unui client sau a unui administrator.

Aplicația a fost împărțită logic în mai multe module, fiecare cu responsabilitate proprie. Această separare a fost făcută pentru a crește lizibilitatea codului și a permite o dezvoltare facilă. Modulele principale sunt:

- Autentificare și gestionarea rolurilor – include conectare, înregistrare, criptarea parolilor și limitarea numărului de încercări de autentificare.
- Festivaluri – gestionează operațiile CRUD pentru festivaluri (creare, actualizare, ștergere, afișare) precum și atribute asociate cum ar fi descriere, dată, locație, imagine, gen muzical, prețul biletului.
- Participări realizate prin asocierea utilizator-festival, cu status de PARTICIPĂ/ ANULEAZĂ/CUMPĂRAT.
- Hărți – fiecare festival are puncte marcate cu iconițe pe harta Leaflet.
- Chestionar – întrebări și răspunsuri pentru determinarea preferințelor muzicale ale utilizatorului, în scopul personalizării recomandărilor.
- Grupuri și mesaje – crearea și gestionarea grupurilor de discuție între utilizatorii care doresc să participe împreună la un festival, inclusiv trimiterea de mesaje.
- Plăți – integrarea cu Stripe, vizualizare automată a biletului cumpărat după confirmare.

În server, fiecare modul menționat anterior este implementat în propriul său pachet. Fiecare modul are în server propriile entități, servicii, controllere și repository-uri. Pachetele de autentificare și chestionar au incluse și obiecte de transfer de date (DTO), pentru a transmite datele între server și client într-un mod sigur.

Relațiile dintre entitățile identificate din baza de date relațională vor fi modelate prin adnotările @OneToMany (ex: un festival are mai multe participări), @ManyToOne

(ex: o participare aparține unui singur client) și @ManyToMany (ex: utilizatorii în grupuri).

3.3 Proiectarea sistemului

După definirea funcționalităților principale ale platformei FestivalGo și organizarea acestora pe module, a fost necesară o etapă de modelare vizuală a sistemului. Această etapă are un rol important în procesul de dezvoltare, deoarece permite înțelegerea clară a structurii aplicației și a modului în care componentele interacționează între ele.

Prin intermediul modelelor vizuale, se pot reprezenta atât interacțiunile dintre utilizator și sistem, cât și relațiile interne dintre clase, entități, tabele. Aceste diagrame oferă o imagine de ansamblu asupra aplicației și ajută la menținerea unei organizări concrete pe parcursul dezvoltării.

În paginile următoare vor fi prezentate aceste modele, ele fiind însoțite de explicații relevante care evidențiază legătura dintre partea vizuală și funcționalitatea reală a platformei FestivalGo.

3.3.1 Cazuri de utilizare

Cazurile de utilizare reprezintă modalitatea eficientă de a evidenția interacțiunile directe ale oamenilor cu sistemul, concentrându-se pe acțiuni concrete și rezultate așteptate ale acestora. Prin intermediul acestor modele, se înțelege mai ușor ce funcționalități oferă aplicația și cum sunt accesate.

În cazul aplicației FestivalGo aceste cazuri de utilizare reflectă funcționalități esențiale pentru utilizatorii platformei, printre care se numără înscrierea la festivaluri, administrarea festivalurilor, completarea chestionarelor și comunicarea în cadrul grupurilor create de participanți. Fiecare caz de utilizare descrie o funcție specifică pe care sistemul trebuie să o îndeplinească pentru un anumit tip de membru (client sau administrator), evidențiind procesul.

În Figura 3.3.1.1 se poate observa una dintre cazurile importante de utilizare: interacțiunea sistemului cu procesul de participare la un festival. Persoana conectată trebuie să se autentifice obligatoriu pe platforma FestivalGo, unde este direcționat pe pagina de start. Acesta poate să vizualizeze sau să caute festivalurile disponibile din platformă. Selectarea unui festival afișează o pagină de detalii specifice despre acel eveniment, unde participantul are acces la două opțiuni principale: participarea la festival sau achiziționarea unui bilet. În cazul în care se alege cumpărarea unui bilet, acesta este redirecționat automat către pagina de plată oferită de Stripe. După finalizarea tranzacției, beneficiarul este redirecționat înapoi în aplicație, pe pagina de start. Alternativ, dacă utilizatorul nu este sigur că dorește achiziționarea imediată a biletului, poate apăsa pe butonul „Participă”. În acest caz, sistemul înregistrează intenția sa de participare și îi afișează un mesaj de confirmare. Această funcționalitate oferă flexibilitate utilizatorilor, permițându-le să își manifeste interesul față de un festival, chiar dacă nu sunt pregătiți să efectueze o plată în momentul vizualizării festivalului.

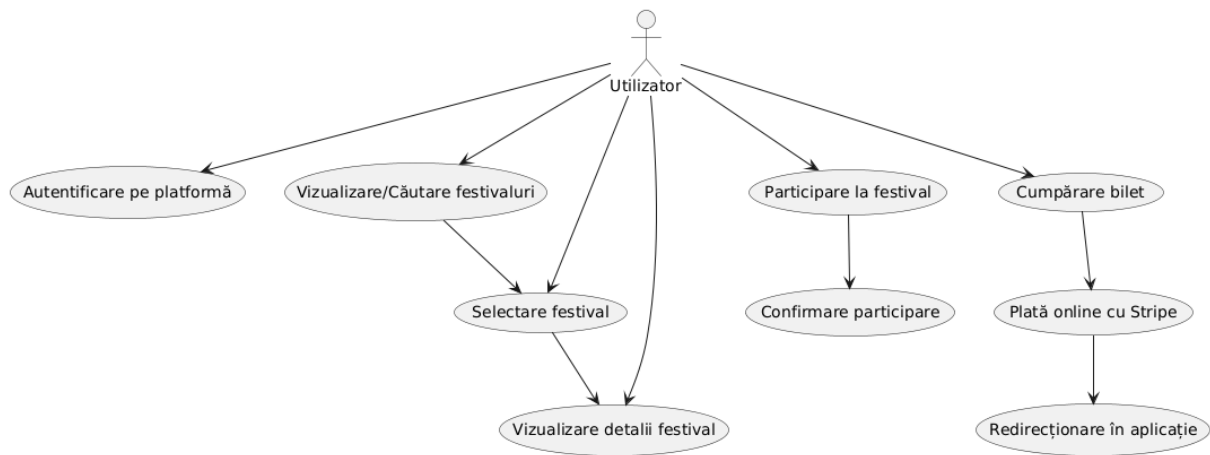


Figura 3.3.1.1: Diagrama de caz pentru participarea la un festival/cumpărare bilet

Un alt caz de utilizare important al platformei FestivalGo este reprezentat de funcționalitatea de adăugare a hărților. Această acțiune este rezervată exclusiv administratorilor deoarece presupune acces la structura și organizarea unui eveniment. În Figura 3.3.1.2 sunt evidențiați pașii parcurși de un administrator în aplicația FestivalGo pentru a adăuga un festival nou împreună cu harta sa. După autentificare, administratorul este redirecționat către o pagină principală cu meniu, unde poate naviga spre o pagină dedicată pentru adăugarea unui nou festival (nume, locație, descriere, imagine, prețul unui bilet, data începerii, data încheierii, genul muzical corespunzător). După completarea și salvarea formularului, interfața se extinde pentru a permite adăugarea unei hărți interactive. Această funcționalitate este modelată prin relația <<extend>>, deoarece această acțiune este permisă doar după salvarea cu succes a festivalului. În cadrul acestei hărți, administratorul poate marca diferite puncte relevante pentru festival. Toate aceste informații sunt ulterior salvate în baza de date.

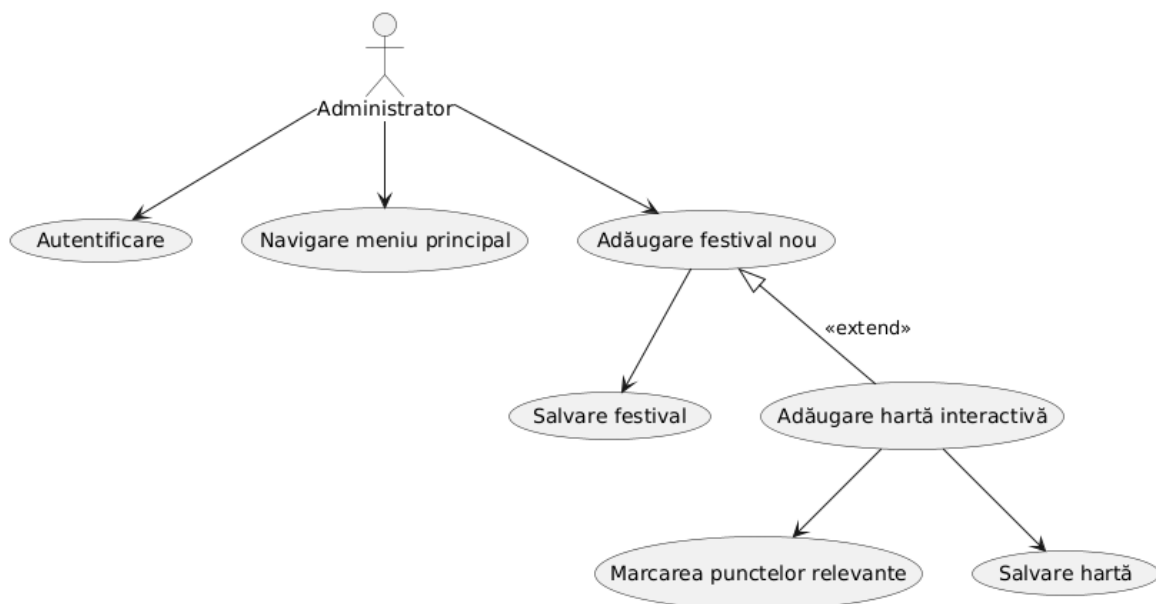


Figura 3.3.1.2: Diagrama de caz de utilizare pentru crearea unui festival și adăugarea hărții aferente

Administratorul platformei are posibilitatea de a interveni oricând asupra datelor existente în sistem, în situațiile în care informațiile introduse inițial sunt greșite sau noi detalii sunt descoperite după postarea festivalului în sistem. Pentru această situație specifică, se definește un nou caz de utilizare: modificarea datelor festivalului. Acest caz include atât posibilitatea de a actualiza informațiile generale (numele, data, descriere, locație, data începerii și a încheierii festivalului, prețul unui bilet), cât și opțiunea de a șterge complet înregistrarea acestuia din sistem. După autentificare, administratorul accesează pagina principală dedicată acestuia, unde are la dispoziție pagina de editare a unui festival. Acesta vizualizează o listă cu numele festivalurilor deja existente. La selectarea unuia dintre ele, se extinde o interfață pentru a permite editarea sau ștergerea acestuia. Acțiunea de editare este urmată de posibilitatea de modificare a hărții asociate festivalului respectiv. Astfel, administratorul poate actualiza punctele de interes marcate anterior pentru ca acestea să reflecte în mod concret realitatea. Acest caz de utilizare este reprezentat în Figura 3.3.1.3.

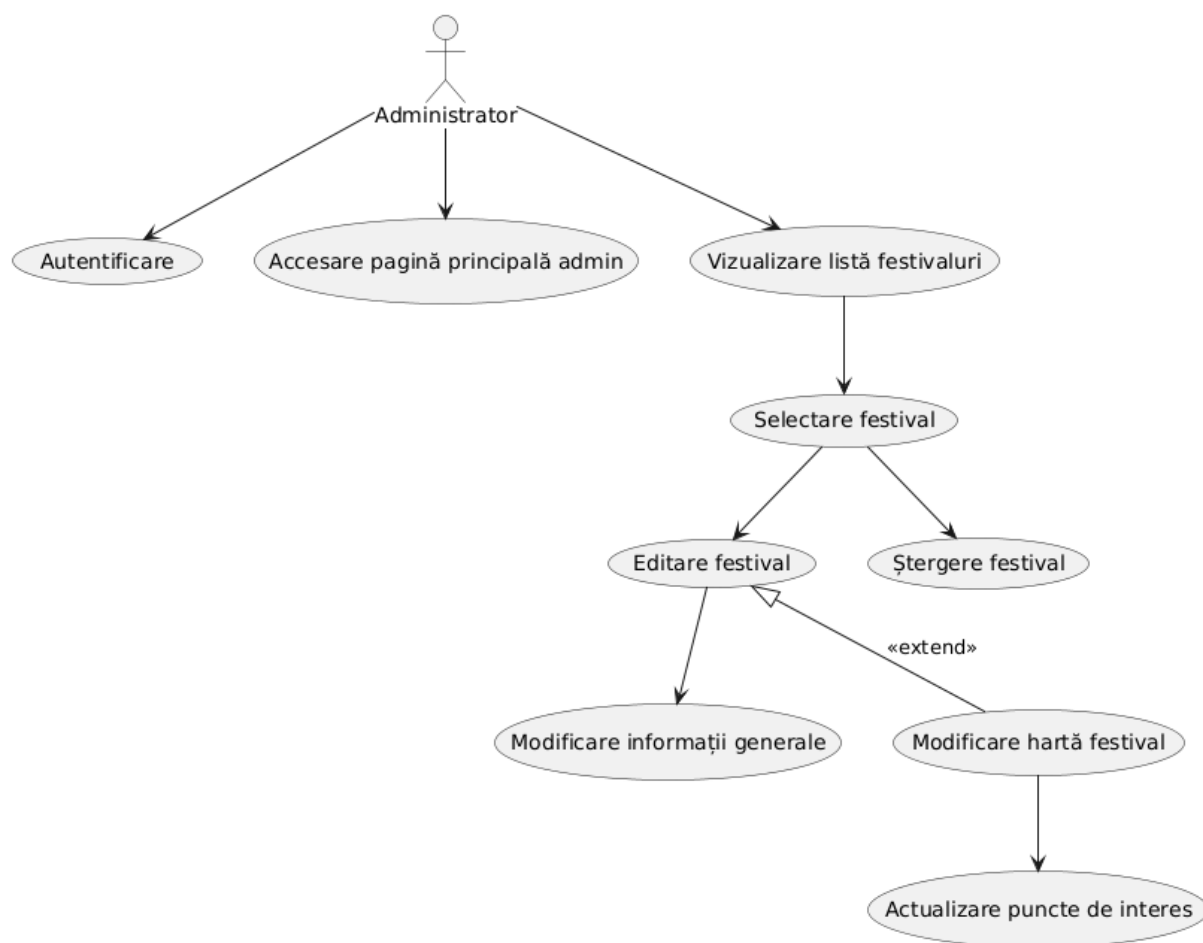


Figura 3.3.1.3: Diagrama de caz de utilizare pentru actualizarea sau ștergerea unui festival existent

Un alt caz de utilizare important al aplicației FestivalGo este completarea chestionarului de către utilizator reprezentată în Figura 3.3.1.4. După autentificare, acesta este redirecționat către pagina de start unde, dacă nu a completat încă chestionarul, va

avea la dispoziție un buton dedicat care îi trimite către acea pagină. Chestionarul este alcătuit dintr-un set de douăzeci și unu de întrebări care vizează trăsături de personalitate. După completarea acestuia și apăsarea butonului de trimite, sistemul procesează automat răspunsurile și determină genul muzical predominant asociat utilizatorului. În funcție de rezultat, platforma redirecționează utilizatorul înapoi la pagina de start, unde, în partea de jos a ecranului, sunt afișate recomandări de festivaluri care se încadrează în genul muzical preferat. Acest proces are rolul de a personaliza experiența fiecărui utilizator și de a-i oferi sugestii relevante, în funcție de persoana în sine.

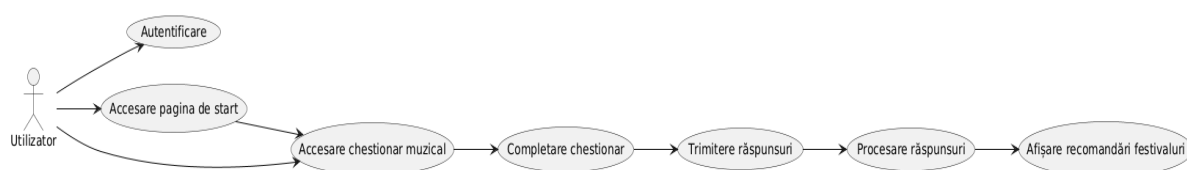


Figura 3.3.1.4: Diagrama de caz de utilizare pentru recomandarea genului muzical

3.3.2 Baza de date

Platforma web FestivalGo utilizează o bază de date relațională în care sunt definite multiple entități, fiecare cu un rol bine definit. Baza de date este structurată astfel încât să susțină toate elementele esențiale ale aplicației, cum ar fi înregistrarea utilizatorilor, participarea la festivaluri, completarea chestionarelor și comunicarea în cadrul grupurilor.

Baza de date va conține entități precum User, Festival, Participare, MapPoint, Question, Answer, ChatGroup, Message și sunt legate între ele prin relații de tip OneToMany, ManyToOne sau ManyToMany, în funcție de logica aplicației. De exemplu, un utilizator poate participa la mai multe festivaluri, iar fiecare participare este legată de un singur utilizator și un singur festival. Tot un utilizator poate trimite mai multe mesaje în grup, dar fiecare mesaj aparține unui singur grup și este trimis de o singură persoană. Grupurile sunt create astfel încât permit participarea mai multor utilizatori, iar aceasta se va face prin tabela intermediară CHAT_GROUP_MEMBER.

Pe partea de chestionar, o întrebare poate avea mai multe variante de răspuns, iar fiecare răspuns este asociat unei întrebări. Aceste date vor fi folosite pentru a genera preferințele muzicale, care va ajuta ulterior la personalizarea sugestiilor din aplicație. De asemenea, festivalurile vor avea și puncte pe hartă, iar aceste puncte vor conține informații geografice precum latitudine și longitudine, un nume și un emoticon reprezentativ. Ele sunt folosite pentru a marca locații importante în cadrul festivalului. Toate aceste relații și legături au fost gândite astfel încât să reflecte cât mai real comportamentul aplicației, oferind o structură clară.

Aceste relații dintre entități și structura completă a bazei de date pot fi observate în Figura 3.3.2.1.

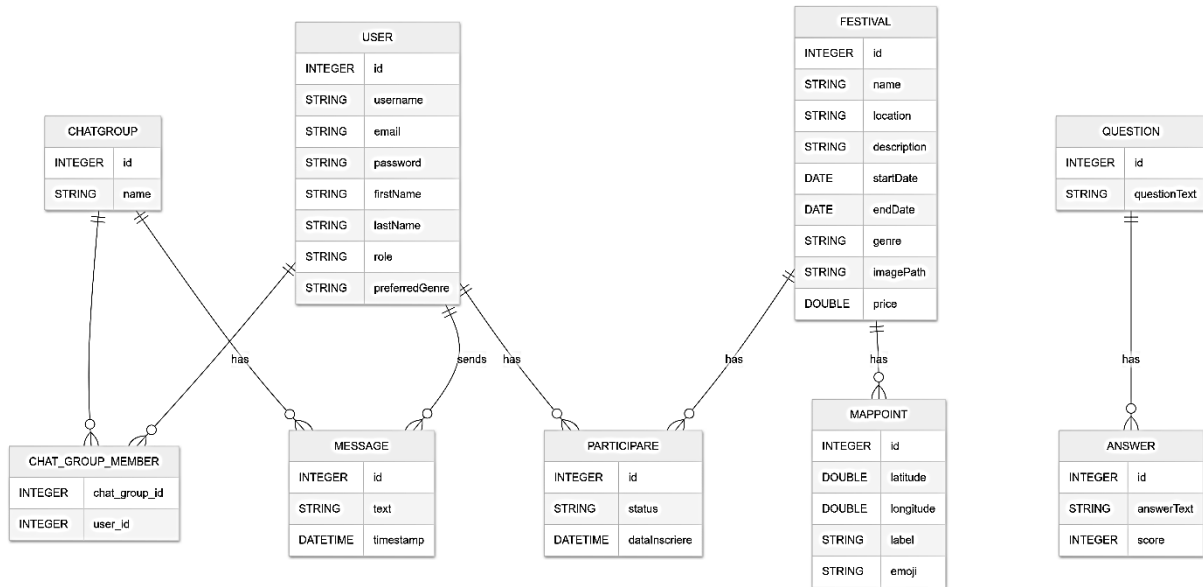


Figura 3.3.2.1: Diagrama EDR a bazei de date relaționale

3.4 Implementare

3.4.1 Descărcarea pachetelor necesare

În această secțiune este prezentată implementarea aplicației FestivalGo. Se începe prin descărcarea pachetelor necesare. Partea de logică a fost realizată în limbajul Java, utilizând framework-ul SpringBoot. Obiectivul principal al acestuia este preluarea cererilor trimise de utilizatori din interfața client, de a le procesa conform logicii aplicației și de a trimite un răspuns corespunzător. De asemenea, acesta este responsabil pentru validarea datelor și manipularea informațiilor legate de festivaluri.

Pentru dezvoltarea, testarea și rularea aplicației server s-a utilizat mediul de dezvoltare IntelliJ IDEA, datorită integrării sale excelente cu framework-ul Spring Boot și a suportului complet pentru proiecte de tip Maven. Pentru a ușura treaba, s-a folosit platforma Spring Initializr pentru configurarea rapidă a unui proiect Spring Boot. Aici au fost completate următoarele detalii:

- Proiect – Maven
- Language – Java
- Spring Boot – versiunea
- Group: com.backend
- Artifact: festivalgo
- Packaging: jar
- Java: versiunea 17

Din platforma Spring Initializr au fost selectate următoarele module: Spring Web pentru API-uri, Spring Data JPA pentru conexiunea la baza de date, MySQL Driver pentru conexiunea cu baza de date MySQL, Spring Security pentru securitatea aplicației și Validation pentru validarea datelor. Proiectul a fost descărcat și deschis în IntelliJ IDEA prin operațiunea „Open” și a fost recunoscut automat ca proiect Maven.

Clasa principală este generată automat de SpringBoot și conține adnotarea `@SpringBootApplication` care inițializează toate componentele framework-ului și pornește serverul la rulare.

Partea vizuală a aplicației FestivalGo s-a dezvoltat utilizând framework-ul Vue.js, în combinație cu Vite. Inițial, proiectul s-a realizat în linia de comandă prin rularea comenzii:

- `npm create vite@last festivalgo-frontend`

În cadrul proiectului, a fost selectată opțiunea Vue ca framework principal, iar limbajul ales este JavaScript. După finalizarea comenzii, s-a intrat în directorul proiectului apoi au fost instalate toate pachetele principale necesare pentru rularea aplicației, folosind:

- `npm install`

Această comandă a instalat toate bibliotecile de bază specificate în fișierul `package.json`. Ulterior, serverul de dezvoltare local a fost pornit folosind:

- `npm run dev`

Acest pas a permis lansarea aplicației într-un mediu local de testare la adresa `localhost`-ului 5173, care permite dezvoltatorului să observe în timp real toate modificările asupra interfeței.

3.4.2 Serverul aplicației

3.4.2.1 Conectarea bazei de date la server

Pentru a permite conexiunea la baza de date, platforma dispune de un fișier numit `application.properties`. Astfel, dezvoltatorul poate să creeze baza de date din cod sau să lege o bază de date existentă. Folosim același `localhost` ca serverul bazei de date, iar dacă baza de date există atunci se creează o legătură, iar dacă nu aceasta se va crea în acel moment. În cazul de față, baza de date este creată acum. De asemenea, în acest fișier s-a definit dialectul Hibernate compatibil cu MySQL 8, precum și strategia de sincronizare a structurii bazei de date cu entitățile Java definite în proiect. Această strategie asigură actualizarea automată a tabelelor deja existente, fără a șterge datele deja salvate. Un alt element configurat aici este portul aplicației, setat la 8081, pentru a evita suprapunerea cu alte servicii existente pe portul 8080.

Aplicația se începe prin implementarea utilizatorilor și gestionarea pe roluri a acestora. Aceasta permite înregistrarea, autentificarea sau ștergere contului precum și protejarea aplicației împotriva abuzurilor. Întregul mecanism este construit în jurul arhitecturii Spring Boot folosind concepte precum entități, controler, serviciu, repository, obiecte de transfer de date și configurări de securitate

3.4.2.2 Utilizatori

Pentru o lizibilitate mai bună a întregii aplicații, toate funcționalitățile majore vor fi separate în pachete cu nume sugestiv. Se începe prin crearea unui pachet numit `user` pentru a gestiona toată logica pentru un utilizator. La baza acestei funcționalități stă clasa `User`, care prezintă modelul de date pentru un utilizator al aplicației. Fiecare utilizator este identificat printr-un ID unic, iar pe lângă acestea sunt stocate informații esențiale precum

numele de utilizator, adresa de email, numele, prenumele, parola, rolul acestuia și genul muzical recomandat. Toate aceste informații sunt asociate către o tabelă din baza de date.

Pentru creșterea securității aplicației și evitarea unor date invalide, au fost aplicate din modulul Jakarta Validation. Această dependență se adaugă în pom.xml în secțiunea dependențe, iar modulul se aplică direct asupra câmpurilor clasei User. Se adaugă adnotarea @Email pentru verificarea formatului corect al adresei de email și @Size pentru impunerea unor limite de lungime asupra parolei și a numelui de utilizator. Se adaugă și adnotarea @NotBlank pentru câmpurile obligatorii. Astfel, datele sunt validate automat înainte de a fi procesate.

Rolul fiecărui utilizator este definit printr-o entitate separată, numită Role, care conține 2 valori de tip String: USER și ADMIN. Această structură ajută la simplificarea procesului de autorizare, deoarece fiecare rol are drepturi bine definite.

Clasa UserRepository este o interfață care extinde JpaRepository și ne oferă automat toate funcționalitățile CRUD pentru entitatea User. Aici se implementează metoda personalizată findByUsername care permite căutarea unui utilizator pe baza numelui de utilizator, necesar pentru procesul de autentificare.

Clasa UserService este componenta centrală care conține logica pentru tot ce ține de funcționalitatea de utilizatori. În interiorul clasei se regăsesc mai multe metode importante. Una dintre cele mai esențiale este metoda responsabilă pentru înregistrarea unui nou utilizator. Ea este marcată cu adnotarea tranzacțională ceea ce înseamnă că toate operațiile sunt desfășurate într-o singură tranzacție și pot fi anulate în caz de eroare. În cadrul metodei, se setează câmpurile relevante, iar ulterior este salvat în baza de date prin intermediul userRepository.save. Alte metode relevante ar fi căutarea după numele de utilizator pentru procesul de login sau căutarea după ID care returnează un utilizator după ID-ul său.

Clasa UserController este clasa care gestionează cererile HTTP legate de operațiunile asupra utilizatorilor, apelează metodele din UserService și returnează răspunsurile către client. Aceasta este marcată cu adnotarea @RestController ceea ce înseamnă că returnează răspunsurile sub formă de răspunsuri JSON. Printre principalele funcționalități implementate se află înregistrarea unui utilizator. Prin endpointul POST („/register”) se permite înregistrarea unui nou utilizator. Înainte de salvare, parola este criptată folosind BcryptPasswordEncoder. Această criptare este esențială pentru securitatea parolelor și este posibilă datorită configurării realizate de clasa SecurityConfig. Rolul utilizatorului este determinat de următoarea gândire: utilizatorii obișnuiți se vor autentifica cu adresa personală de email, iar administratorii se vor autentifica cu un email creat special pentru această platformă. Dacă un utilizator va avea email-ul personal, spre exemplu cu terminația „@yahoo.com”, rolul acestuia va fi client obișnuit, iar dacă terminația email-ului va fi „@festivalgo.com”, utilizatorul este considerat administrator.

O altă funcționalitate principală este autentificarea unui utilizator. Aceasta se face prin endpointul POST („/login”). Această metodă primește un obiect LoginRequest, o clasă de tip transfer de date, care este folosit pentru a transfera datele de login de la frontend către backend. Aceasta conține doar două câmpuri esențiale: email și parolă,

exact ceea ce utilizatorul completează în formularul de autentificare. Acest obiect verifică dacă utilizatorul există și dacă parola introdusă corespunde cu cea criptată din baza de date. Pentru a preveni atacurile de tip forță brută, este inclus un sistem de limitare a cererilor de bază de tip IP.

Se introduce o clasă de tip service nouă care funcționează pe baza IP-ului clientului. Fiecare adresă înregistrată este corelată intern, împreună cu un obiect care conține numărul de încercări eșuate și momentul ultimei încercări. Dacă numărul de încercări eșuate depășește limita prestabilită, cinci încercări consecutive, adresa IP este blocată temporar, iar orice cerere ulterioară de autentificare este refuzată pentru o perioadă de un minut.

Această funcționalitate este folosită direct în cadrul metodei de autentificare din clasa UserController, înainte de a face verificarea efectivă a parolei. Dacă IP-ul este blocat, aplicația returnează un răspuns de eroare, informând utilizatorul că trebuie să aștepte.

Alte funcționalități principale ar fi ștergerea unui utilizator DELETE („/delete/{id}”) care permite ștergerea unui cont în funcție de ID. Pentru obținerea informațiilor unui utilizator, funcție care o sa ne fie utilă în interfața vizuală, se implementează o funcție de returnarea datelor, cum ar fi rolul, ID-ul și numele de utilizator. Procesul de autentificare este primul contact dintre utilizator și aplicație. Această secvență, ilustrată în Figura 3.4.2.2.1, pare simplă, dar implică mai multe componente care colaborează: interfața frontend, controllerul din backend, serviciul de verificare a ratei, serviciul utilizatorului și encoderul de parolă.

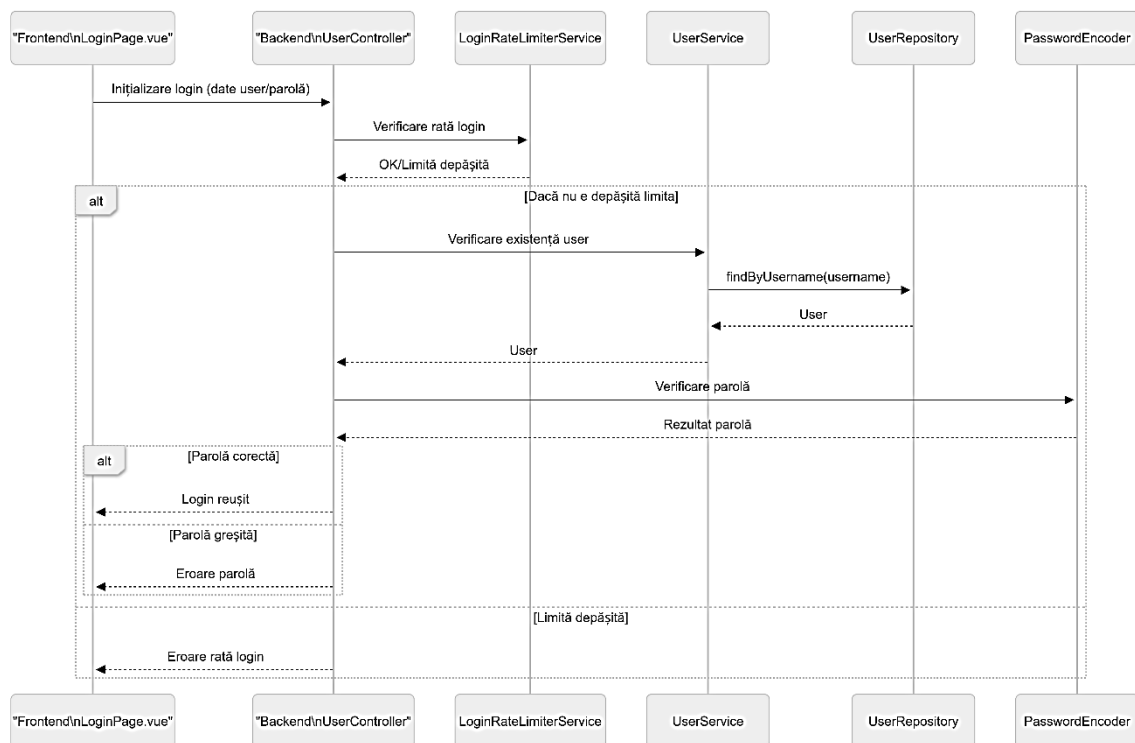


Figura 3.4.2.2.1: Fluxul logic pentru autentificarea unui utilizator în platforma FestivalGo

3.4.2.3 Festival

În cadrul aplicației FestivalGo, entitatea Festival este o componentă principală pentru gestionarea evenimentelor. Clasa Festival este o entitate JPA, adnotată cu `@Entity` și este mapată la o tabelă din baza de date. Aceasta conține câmpuri de baza cum ar fi ID-ul, generat unic, numele festivalului, descriere, data de început a festivalului și data de final, orașul în care are loc, imaginea festivalului, prețul unui bilet la festival și genul muzical. Genul muzical, la fel ca rolul unui utilizator, este definit într-o entitate separată de tip string. Genul muzical este important atât la identificarea tipului de festival cât și la influențarea sugestiilor și recomandărilor muzicale.

Interfața FestivalRepository este folosită aici strict pentru operațiile CRUD deoarece nu este nevoie de implementarea altor metode. Această abordare simplifică semnificativ codul și face logica aplicației.

Clasa FestivalService va gestiona de logica aplicației pentru partea de festival. Pe lângă metodele clasice de creare, ștergere, actualizare, citire s-a implementat o metodă de citire a imaginilor încărcate de utilizator. Această metodă a fost implementată pentru ca administratorul să poată introduce imagini din calculatorul personal pentru adăugarea unui nou festival. Imaginea ar trebui să fie salvată pe server, nu în baza de date deoarece ar fi inefficient. Aplicația salvează doar numele fișierului pentru a putea fi afișat ușor. Metoda generează un nume generic pentru fiecare imagine, evitând astfel suprascrieri accidentale, creează automat folderul de salvare, dacă acesta este inexistent, scrie fișierul pe disc și returnează numele festivalului. Această metodă este separată de logica clasică CRUD, deoarece gestionează un fișier real, nu un obiect simplu de tip Festival.

Clasa FestivalController este responsabilă pentru preluarea cererilor HTTP din frontend, asociată cu calea „/festivals” și permite manipularea resurselor de tip Festival. Unul dintre endpointurile principale ale acestei clase este adăugarea unui festival împreună cu imaginea sa POST („/post”). Față de un POST clasic, aici se primește o cerere formată din cele două componente: un obiect Festival trimis ca Json și un fișier de tip MultipartFile care reprezintă imaginea. Aceasta salvează imaginea prin apelarea metodei din FestivalService. Festivalul complet este salvat în baza de date cu tot cu imagine, aceasta returnând un status `HttpStatus.CREATED`. Un alt endpoint important este cel de tip GET („/festivals”) care returnează lista tuturor festivalurilor din baza de date. Nu necesită parametri suplimentari. Se va crea un endpoint special pentru obținerea datelor unui festival specific, identificat prin ID-ul său unic. Dacă festivalul există, răspunsul va fi `HttpStatus.OK`, dacă nu, `HttpStatus.NOT_FOUND`. Pe lângă acestea, se va implementa și o cerere pentru ștergerea unui festival și unul pentru actualizarea acestuia, PUT („/festivals/{id}”) care permite actualizarea datelor unui festival și imaginea acestuia. Dacă nu se va trimite o imagine nouă, imaginea rămâne aceeași.

Fluxul logic pentru adăugarea unui nou festival în platforma FestivalGo este demonstrat în Figura 3.4.2.3.1. Adăugarea unui festival, editarea sau ștergerea acestuia este realizată doar de către un administrator.

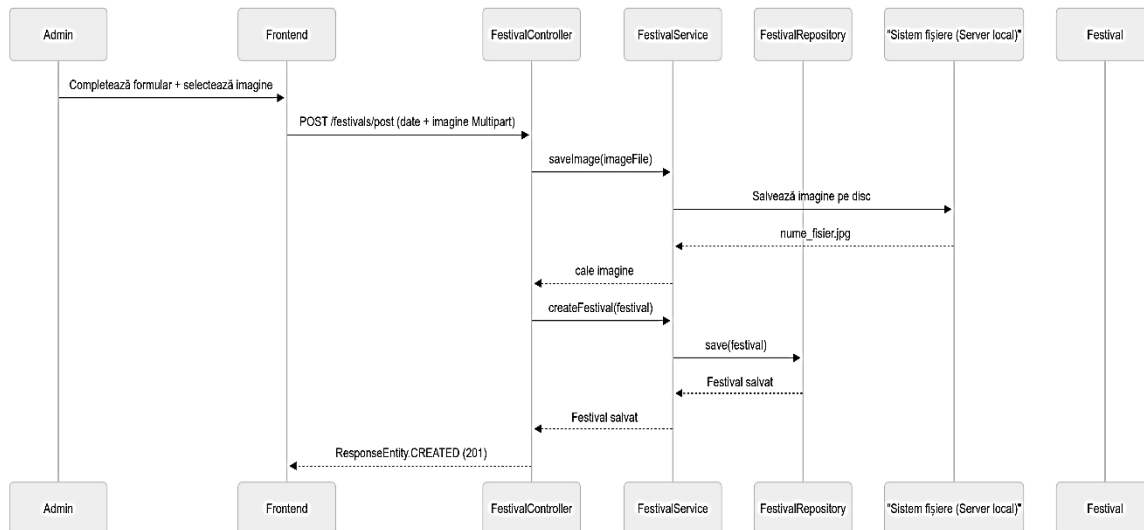


Figura 3.4.2.3.1: Fluxul logic de adăugare a unui festival în platforma FestivalGo

3.4.2.4 Participări

Pentru platforma FestivalGo, gestionarea participării utilizatorilor la festivaluri este realizată printr-un sistem simplu, dar eficient, bazat pe entitatea Participare. Clasa Participare este o entitate JPA care conține o relație dintre un utilizator și un festival. Aceasta conține câmpuri precum ID, utilizatorul care participă (relație ManyToOne cu entitatea User), festivalul la care participă (relație ManyToOne cu entitatea Festival), status, care este un câmp de tip enum care indică starea participării: PARTICIPA sau ANULAT, și data înscrierii. În acest moment al dezvoltării, platforma nu va conține plata de bilete, urmând să fie dezvoltat mai târziu.

Interfața ParticipareRepository conține metode personalizate, pe lângă cele standard, acestea având roluri bine definite. Se va crea o metodă de afișare a tuturor participărilor cu statusul PARTICIPA pentru un anumit utilizator, pe baza ID-ului acestuia folosind interogarea personalizată @Query pentru a selecta doar participările relevante pentru USER. Se mai implementează o metodă de tip adevărat sau fals pentru a verifica dacă există deja o participare un anumit status, metodă pe care se folosește pentru a preveni înscrierea multiplă a unui utilizator la un festival.

Clasa ParticipareService v-a fi legată de interfețele Repository ale entităților User și Festival. În implementarea clasei de tip Service pentru participarea la festival, s-a implementat o metodă pentru a permite utilizatorului să vadă la ce festivaluri este înscris și la care participarea este încă validă. Se creează o funcție ce apelează metoda de afișare a tuturor participărilor cu statusul PARTICIPA din clasa ParticipareRepository și filtrează participările pentru a păstra doar cele a festivalurilor cu data de sfârșit în viitorul apropiat ($endDate > azi$). Se va implementa o metodă pentru a permite unui utilizator să se înscrie la un festival folosind metoda de verificare a existenței utilizatorului din UserRepository, verifică dacă festivalul există folosind FestivalRepository, apoi folosește metoda de verificare dacă utilizatorul este sau nu înscris la acest festival unde statusul este PARTICIPA. Dacă nu există deja o astfel de participare, se creează o nouă entitate, setează utilizatorul, festivalul și statusul PARTICIPA, apoi salvează participarea în baza de date. Se

implementează și o metodă pentru anularea participării la un festival, unde se caută participarea după ID, folosind `ParticipareRepository`. Dacă participarea există, statusul acesteia se schimbă în `ANULAT` și salvează modificarea în baza de date. Această metodă oferă utilizatorului posibilitatea de a renunța participarea la un festival, fără a șterge efectiv participarea din sistem.

`ParticipareController` este clasa care primește prin cereri HTTP și apelează metodele din `ParticipareService`, această clasă fiind mapată cu „/participări”. Aici se va implementa cererea de tip GET („/user/{userId}”) pentru a returna lista participărilor active pentru un anumit utilizator, o cerere de tip POST („/adauga”) care permite înscrierea unui utilizator la un festival și o cerere de tip PUT („/anulează/{id}”) pentru anularea participării la un festival.

3.4.2.5 Chestionar muzical

Partea de chestionar muzical este compusă din mai multe etape, fiecare având un rol specific în procesul de recomandare a unui gen muzical pentru utilizatori. În prima etapă administratorul platformei introduce în sistem întrebările și răspunsurile aferente. Fiecare întrebare poate mai multe variante de răspuns având un scor asociat.

Pentru prima etapă vom avea două entități noi, `Question` și `Answer`. Entitatea `Question` reprezintă o întrebare din chestionar care are două câmpuri: ID și `questionText` (textul întrebării) și o listă de răspunsuri. `Answer` reprezintă un răspuns posibil la o întrebare și conține trei câmpuri precum ID, `answerText` (textul răspunsului) și scorul asociat răspunsului. Între `Question` și `Answer` există o relație de `OneToMany` deoarece o întrebare poate avea mai multe răspunsuri.

Cele două entități, extinse prin `JpaRepository` au fiecare câte un repository ce permit operații CRUD (creare, citire, actualizare, ștergere) pe întrebări și răspunsuri.

Se creează un singur serviciu pentru a implementa logica pachetului, `QuizService` conținând logica pentru adăugarea unei întrebări cu una sau mai multe răspunsuri. Când se adaugă o întrebare nouă, serviciul parcurge lista de răspunsuri și setează pentru fiecare răspuns referința către întrebare, apoi întrebările și răspunsurile sunt salvate în baza de date.

`QuizController`, mapată cu „/api/quiz” implementează endpointul POST („/question”) care primește o întrebare și lista de răspunsuri sub formă JSON. Controllerul apelează metoda din `QuizService` pentru a salva întrebarea și răspunsurile în baza de date.

În a doua etapă a procesului de chestionar muzical, utilizatorul parcurge întrebările afișate pe platforma `FestivalGo` și selectează răspunsurile care i se potrivesc cel mai bine. Fiecare răspuns selectat din utilizator este asociat cu un ID unic, corespunzător entității `Answer` din baza de date. Pentru început, pentru a putea completa chestionarul, aplicația trebuie să afișeze întrebările și variantele de răspuns. În acest scop se implementează o metodă de preluare a tuturor întrebărilor și a răspunsurilor în `QuizService`, apoi se creează o metodă care apelează serviciul în controller prin endpointul de tip GET („/questions”). Utilizatorul parcurge întrebările și selectează câte un răspuns pentru fiecare întrebare. După ce utilizatorul finalizează chestionarul, aplicația colectează toate răspunsurile selectate. Se creează o clasă simplă de tip transfer de obiecte DTO, folosită pentru a

transmite răspunsurile selectate de utilizator. Aceasta conține un singur câmp care reprezintă ID-ul unui răspuns selectat de un utilizator.

Etapa a treia constă în procesarea răspunsurilor și generarea recomandării muzicale. Se creează un serviciu destinat calculării scorurilor care primește lista de răspunsuri selectate sub formă de UserAnswerDTO și ID-ul utilizatorului. Se inițializează un scor pentru fiecare gen muzical, spre exemplu ROCK, EDM, POP, INDIE, URBAN, FOLK, JAZZ, care are o structură tip „hartă”, scorul inițial fiind zero. Pentru fiecare răspuns primit, se extrage din entitatea Answer valoarea scorului. Acest scor reprezintă valoarea răspunsului respectiv pentru un anumit gen muzical. Se determină genul muzical printr-o regulă de mapare, de exemplu, dacă o anumită întrebare conține cuvinte cheie, acestea sunt asociate cu un gen muzical. Se adaugă scorul răspunsului la scorul categoriei muzicale, iar după procesarea tuturor răspunsurilor, se calculează scorurile pentru a identifica genul muzical cu scorul cel mai mare. Acest gen este considerat genul muzical cu scorul cel mai mare. Se creează o metodă care caută utilizatorul în baza de date folosind UserRepository, apoi se va actualiza câmpul destinat genului muzical preferat al utilizatorului cu genul recomandat. Această metodă este apelată în clasa RecommendationController pentru a procesa răspunsurile și a determina genul muzical recomandat. Formularul muzical este destinat doar utilizatorilor de tip user și se poate completa doar odată.

Chestionarul muzical este bazat pe algoritmul de scoruri și are o procesare automată fără intervenție manuală. Fluxul complet al acestuia, de la afișarea întrebărilor până la generarea recomandării, este reprezentat în Figura 3.4.2.5.1.

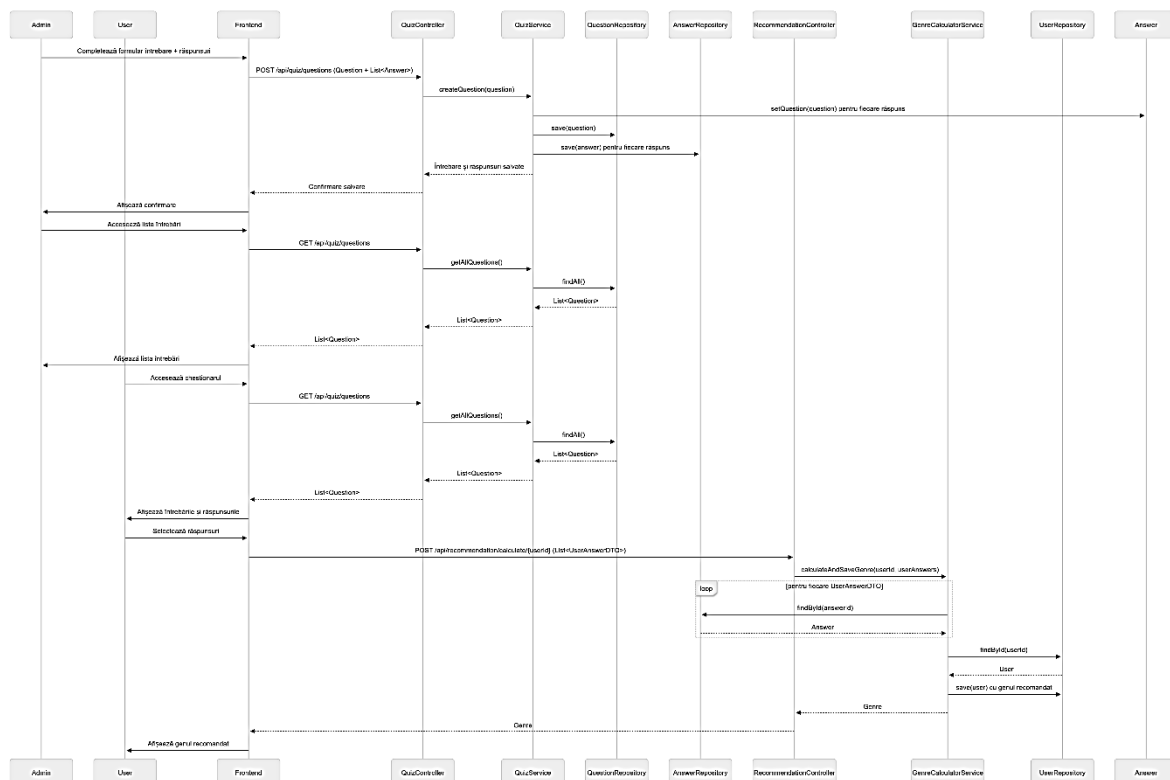


Figura 3.4.2.5.1: Fluxul logic pentru completarea chestionarului

3.4.2.6 Mesagerie

Aplicația FestivalGo conține un sistem de creare a grupurilor de chat, care permite utilizatorilor să comunice între ei. Implementarea sistemului de grupuri de chat a început cu definirea entităților ChatGroup care conține un ID unic, un nume pentru grup și o listă de membrii reprezentată ca o relație de tip ManyToMany cu entitatea User. Această relație permite ca un grup să aibă mai mulți membrii și ca un utilizator să poată face parte din mai multe grupuri. S-a creat entitatea Message, care conține un ID unic, o referință către expeditor, marcată cu relația ManyToOne cu entitatea User, textul mesajului și data trimiterii.

S-au creat interfețele Repository pentru ambele entități, care extind JpaRepository și permit operații CRUD asupra grupurilor și mesajelor. În aceste metode nu s-au dezvoltat metode noi,

S-au implementat serviciile ChatGroupService și MessageService unde s-a dezvoltat logica acestui modul. În clasa ChatGroupService s-a implementat logica pentru crearea unui grup nou. Aici vom verifica existența utilizatorului care creează grupul cu ajutorul interfeței UserRepository, se creează un obiect nou ChatGroup cu numele specific, apoi se adaugă creatorul în lista de membrii și se salvează grupul în baza de date prin ChatGroupRepository.

O altă metodă relevantă este adăugarea unui membru în grup, unde se verifică existența grupului și a utilizatorului, se adaugă utilizatorul în lista de membri ai grupului, dacă nu este deja membru și se salvează modificarea în baza de date. Ultima metodă implementată este obținerea grupurilor unui utilizator. Aceasta primește ID-ul utilizatorului și caută în baza de date toate grupurile. La final, se filtrează grupurile pentru a returna doar cele în care utilizatorul este membru.

Clasa MessageService gestionează logica pentru trimiterea și afișarea mesajelor în grupuri. Aceasta are două funcționalități de bază: trimiterea unui mesaj pe grup și afișarea mesajelor dintr-un grup. Pentru trimiterea unui mesaj pe grup, prima dată se verifică existența grupului și a utilizatorului, apoi se verifică dacă utilizatorul este membrul grupului, apoi se creează un obiect nou de tip Message cu expeditorul, grupul și textul, acesta salvându-se în baza de date folosind MessageRepository. Afișarea mesajelor se face aprins căutarea tuturor mesajelor asociate grupului, ordonate după data trimiterii. Se returnează apoi lista de mesaje pentru a putea fi afișate.

Aceste metode din servicii sunt apelate în ChatGroupController, clasă care primește cererile HTTP, mapată cu „/chat/groups”. Printre metodele principale se numără crearea unui grup prin endpointul POST („/create”) care creează un nou grup de chat cu numele dat și adaugă creatorul ca membru. O altă metodă principală este adăugarea unui utilizator într-un grup prin endpointul POST („/{groupId}/add”) care primește ID-ul grupului și a utilizatorului ca parametrii, apelează metoda din service pentru a adăuga utilizatorul în grup și returnează un mesaj de confirmare. Se va implementa și trimiterea unui mesaj într-un grup prin endpointul POST („/{groupId}/message”) care primește ID-ul grupului, ID-ul expeditorului și textul mesajului ca parametrii, apelează metoda din MessageService pentru a salva mesajul și returnează mesajul salvat. Se va implementa și afișarea tuturor

mesajelor dintr-un grup prin endpointul GET („/{groupId}/message) și afișarea tuturor grupurilor unui utilizator prin endpointul GET („user/{groupId}/message) .

În Figura 3.4.2.6.1 este realizată diagrama secvențială pentru mesajele aplicației FestivalGo.

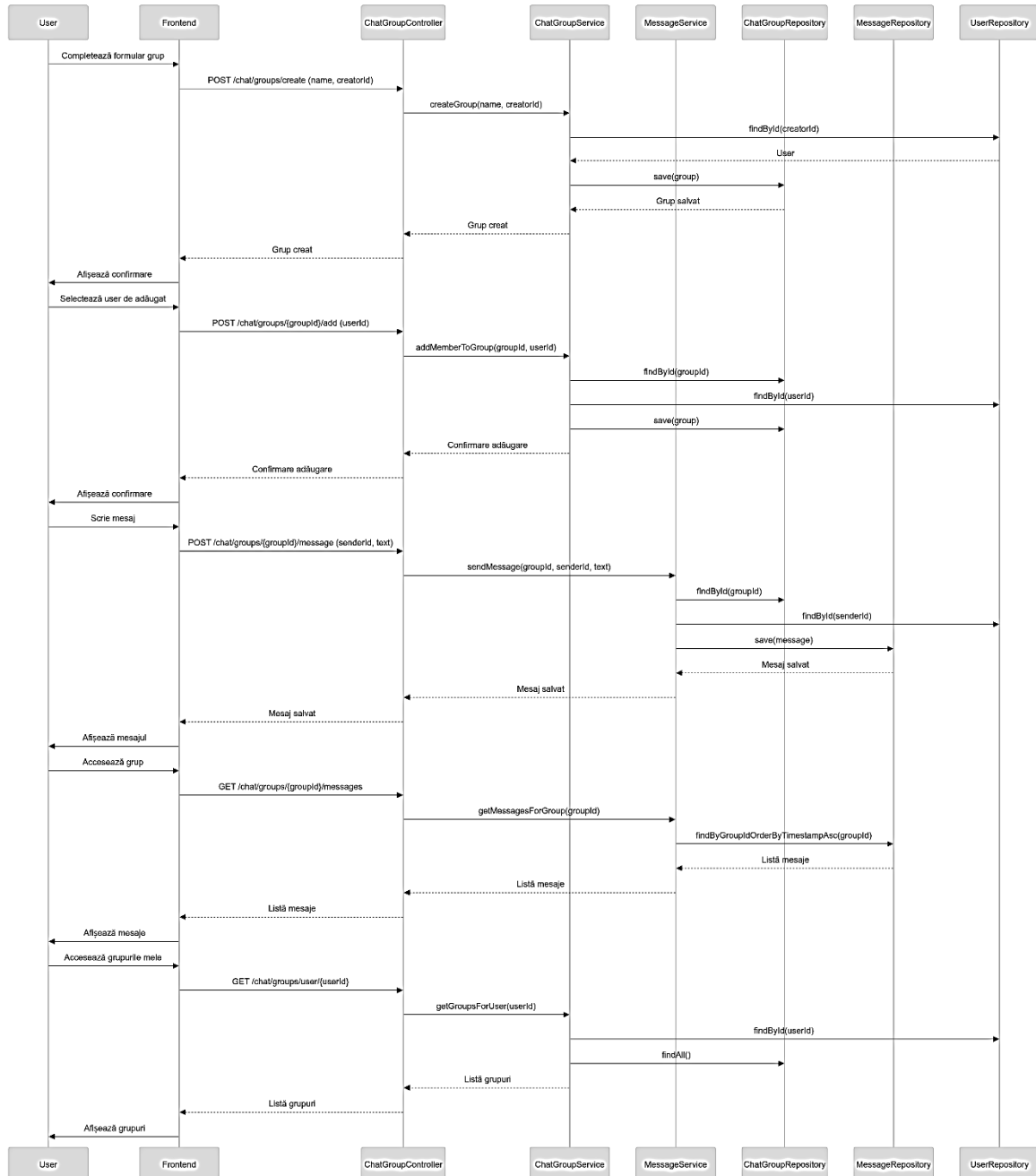


Figura 3.4.2.6.1: Fluxul logic pentru trimiterea mesajelor și crearea de grupuri de chat

Acestea au fost funcționalitățile principale aplicației FestivalGo. În cele ce urmează se vor crea interfețele aferente acestor funcționalități, apoi se vor integra serviciile externe care fac aplicația să iasă în evidență.

3.4.2.7 Configurări

În afara pachetelor pentru logica aplicației se va crea clasa WebConfig. Aceasta este o clasă de configurare SpringBoot care extinde WebMvcConfigurer. Rolul acesteia este de a configura două aspecte foarte importante: CORS și servirea fișierelor tip imagine.

Pentru configurarea CORS s-a creat un fișier de tip „coș” unde se suprascrie metoda addCorsMappings. Se adaugă o regulă care să permită accesul de pe orice endpoint dar doar de la originea localhost-ului de pe interfață. Orice cerere venită doar de la localhost-ul 5173 va fi acceptat de backend, chiar dacă rulează pe porturi diferite.

Pentru servirea fișierelor de tip imagine s-a rescris metoda addResourceHandlers din interfața de configurare a aplicației web și s-a stabilit o regulă care spune că orice cerere către adresa care începe cu „/uploads/” să fie

3.4.3 Interfața aplicației

3.4.3.1 Autentificare și Înregistrare

Se deschide partea de interfață în aplicația VisualStudio Code și se creează un folder nou care va conține toate paginile. Implementarea interfeței începe cu componenta de autentificare și înregistrare.

Componenta LoginPage.vue a permis autentificarea utilizatorilor existenți pe platformă. Fiecare componentă vizuală este construită conform structurii framework-ului Vue.js.

În secțiunea <template>, unde se scrie partea de cod HTML, se regăsește formularul de autentificare. Acesta conține două câmpuri principale: numele de utilizator și parola. Acestea sunt legate prin v-model la variabilele scrise în script, asigurându-se legătura între interfață și datele din componentă. Tot aici se afișează un mesaj de eroare, în cazul în care autentificarea eșuează sau datele introduse nu sunt corecte. Sub formular va exista un link care permite navigarea către pagina de înregistrare dacă utilizatorul nu are cont în aplicația FestivalGo. În partea de jos a formularului se află un buton care apelează metoda de autentificare din secțiunea următoare.

În secțiunea <script>, unde se scrie partea de cod JavaScript, se implementează metoda de autentificare. Aceasta preia datele introduse de utilizator și trimite o cerere de tip POST cu ajutorul bibliotecii Axios, practică standard în aplicațiile frontend pentru a comunica cu serverul, către endpointul „/users/login”. Acest endpoint este creat în server, unde există o metodă în controller care primește datele de autentificare, verifică dacă acestea sunt corecte și returnează un răspuns către interfață. Dacă răspunsul primit de server este unul de succes, datele relevante despre utilizator, cum ar fi ID-ul, rolul acestuia și numele de utilizator, iar utilizatorul este redirecționat către pagina corespunzătoare rolului său (admin sau user).

Designul paginii este gândit astfel încât formularul să fie centrat pe ecran, cu un fundal modern și elemente vizuale care asigură lizibilitate. Se definesc reguli clare de poziționare, fonturi și culori. Se folosește o imagine specifică cu logo-ul aplicației.

Componenta RegisterPage.vue a permis înregistrarea unui cont nou pe platformă. În secțiunea <template>, se introduc toate câmpurile necesare pentru înregistrare: nume de utilizator, email, parolă, nume, prenume. Aceste câmpuri sunt legate prin v-model la fel ca cele din componenta LoginPage.vue. În această secțiune sunt prezente mesajele de eroare pentru validarea locală, în cazul în care utilizatorul nu completează corect datele de înregistrare. În partea de jos a formularului se află un buton de trimitere care declanșează metoda de înregistrare.

În secțiunea <script> se verifică dacă toate câmpurile sunt completate corect. Dacă aceste condiții nu sunt îndeplinite, se afișează un mesaj de eroare specific pentru fiecare câmp. În caz contrar, se trimite o cerere POST cu ajutorul bibliotecii Axios către endpointul „users/register”. Dacă totul este conform condițiilor, utilizatorul este notificat că înregistrarea a fost realizată cu succes.

Designul paginii este asemănător al paginii de autentificare, folosind aceeași poză de fundal cu logo-ul aplicației pentru a păstra coerența vizuală a aplicației.

După autentificare, utilizatorul este trimis către pagina corespunzătoare rolului său. De aici, aplicația nu mai este una comună.

3.4.3.2 Client

Clientul aplicației este direcționat către următoarea pagină care este de întâmpinare și acces la funcționalitățile principale.

Componenta WelcomePage.vue este un panou principal pentru utilizatorul autentificat fiind realizat după structura standard Vue.js. În partea de HTML, se creează un meniu lateral și conține butoanele pentru navigarea rapidă: lista festivalurilor active, participările, chat, logout, ștergerea contului, link către formularul de recomandare. În prim plan va fi un mesaj de întâmpinare personalizat cu numele utilizatorului preluat din localStorage și sloganul aplicației. Apoi s-a creat secțiunea Festivalul Lunii unde este afișat cel mai apropiat festival și secțiunea de recomandări. În partea de JavaScript se implementează funcții specifice pentru calcularea celui mai apropiat festival și secțiunea de recomandări.

În primul rând se folosește onMounted , care la încărcarea paginii trimite cereri HTTP către backend pentru a obține datele utilizatorului și lista festivalurilor. Pentru cel mai apropiat festival se creează o funcție folosind computed care parcurge toate festivalurile disponibile și filtrează doar cele care au data de început în viitor, apoi le sortează în ordine cronologică și selectează primul din listă. Se dorește ascunderea butonului de chestionar muzical dacă utilizatorul a completat în trecut această secțiune. Dacă utilizatorul are variabila preferredGenre gol în baza de date, în meniu va apărea butonul pentru accesarea genului muzical. După ce acesta accesează chestionarul, variabila va fi completată cu genul muzical preferat. Astfel, condiția pentru afișare nu mai este îndeplinită, iar secțiunea dispare automat. Recomandările festivalului sunt generate pe baza genului muzical preferat. După ce datele utilizatorului sunt preluate din backend, variabila preferredGenre este setată cu genul preferat. Apoi, lista festivalurilor este filtrată astfel încât să fie afișate doar acele festivaluri care au genul muzical potrivit cu preferința utilizatorului. Design-ul paginii constă într-o paletă de culori închise cu accente pe violet și albastru pentru a crea senzația de lumini neon din cadrul festivalurilor.

Se creează apoi pagina de festivaluri care va putea fi accesată din pagina principală. În partea de sus se vor inițializa câmpuri precum căutarea titlului și perioada festivalului. Festivalurilor vor fi organizate în trei categorii, fiecare categorie reprezentând o categorie de vibe: Relax&Sunset Vibes, Energetic&Party Vibes și Underground Indie. În secțiunea <script>, se implementează logica de filtrare și organizare a festivalurilor. Se definesc trei categorii principale (relax, energetic, underground) și un o mapare care asociază fiecare gen muzical la categoria corespunzătoare. Se folosește o proprietate calculată festivalsByVibe care grupează festivalurile pe categorii. Design-ul paginii este același cu pagina de întâmpinare.

Se construiește componenta FestivalsDetails.vue care conține detaliile despre festival după ID-ul unic al acestuia din baza de date. În partea de <template>, se regăsește titlul festivalului, iar sub titlu sunt afișate imaginea, și detaliile festivalului. La sfârșitului paginii se regăsește butonul de participare care este apelat în partea de script. Se preiau detaliile festivalului pe baza ID-ului prin cerere HTTP către backend. Se implementează funcția participateLaFestival care va implementa cererea POST din backend pentru adăugare de participări. Designul se păstrează ca în cazul celorlalte componente.

Componenta UserParticipations.vue conține, ca un portofoliu, toate participările unui utilizator. Se inițializează titlul în partea de sus a paginii în template urmat de o verificare condiționată care afișează un mesaj informativ dacă utilizatorul nu este înscris la niciun festival. Pentru fiecare participare, se afișează un card orizontal care conține imaginea pe partea stângă și informațiile pe partea dreaptă. Dacă participarea este activă, va exista un buton de anulare al participării. Logica acestui buton este dezvoltat în următoarea secțiune, unde se preiau toat participările utilizatorului la montarea componentei onMounted prin cereri HTTP folosind ID-ul utilizatorului din localStorage. Funcția anuleazăParticipare permite utilizatorului să trimită o cerere PUT cu ajutorul bibliotecii Axios care apelează metoda din server pentru schimbarea statusului unei participări.

Pagina QuizPage.vue este implementată pentru ca utilizatorii să poată completa chestionarul muzical și să descopere genul muzical preferat. În partea de sus se va afișa titlul paginii. Pentru fiecare întrebare se afișează un box cu întrebarea și răspunsurile multiple sub formă de butoane radio. Se preiau toate întrebările din backend prin cererea HTTP către endpointul „api/quiz/questions”. Se implementează funcția submitQuiz care colectează toate răspunsurile selectate de utilizator și le trimite către backend prin cerere POST cu ajutorul bibliotecii Axios unde se calculează genul muzical preferat. După primirea rezultatului, se afișează genul muzical recomandat cu animații, iar după patru secunde utilizatorul este redirecționat către pagina de festivaluri.

Sistemul de mesaje din aplicația FestivalGo este implementat prin patru componente Vue.js care lucrează împreună. Prima componentă este ChatHub.vue care conține un buton pentru crearea de grupuri noi. Pe partea dreaptă se află panoul principal de chat, care afișează conversația fiecărui grup. În următoarea secțiune se preiau toate grupurile utilizatorului prin cerere HTTP și se implementează funcția selectGroup care permite utilizatorului să selecteze un grup pentru conversație, iar funcția reloadGroups reîncarcă lista de grupuri. Componenta GroupChat.vue conține un titlu cu numele grupuri,

o secțiune pentru afișarea mesajelor unde fiecare mesaj este afișat cu numele expeditorului și textul, o casetă de input pentru scrierea mesajelor noi și un buton de trimitere. La sfârșitul paginii se află link-uri rapide sub formă de emoticoane pentru accesarea serviciilor de cazare (Booking) și transport (CFR). Butonul de trimitere a mesajelor apelează funcția `sendMessage` care trimite un mesaj nou către backend prin cerere POST, apoi reîncarcă mesajele pentru a afișa cel mai nou trimis. Funcțiile `goToBooking` și `goToCFR` deschid link-uri externe către serviciile respective în taburi noi. Componenta `CreateGroup.vue` are un câmp de input pentru numele grupului, un buton pentru crearea grupului și un mesaj de eroare dacă se întâmpină probleme. Butonul apelează funcția `createFroup` care verifică dacă numele grupului este valid, apoi trimite o cerere POST pentru crearea grupului. Componenta `AddMember`. Vue conține un titlu, o bară de selectare care afișează toți membrii aplicației și un buton de confirmare. Butonul apelează funcția `addUserToGroup` care verifică dacă utilizatorul este selectat un utilizator, apoi trimite o cerere POST pentru adăugare.

Aceste patru componente lucrează împreună pentru a oferi un sistem complet de comunicare, asemănător cu sistemele de mesagerie populare.

3.4.3.3 Administrator

După autentificare, administratorii vor fi trimiși pe o pagină de întâmpinare care va conține un meniu de administrare și un buton de delogare care va șterge datele din `localStorage`. Designul paginii este unul simplu, cu accent pe funcționalitate și navigare rapidă.

Componenta `Add_Festival.vue` conține un formular cu toate câmpurile necesare. La apăsarea butonului de trimitere, datele sunt trimise printr-o cerere POST. Designul este centrat pe claritate și ușurință în completare. Componenta `UpdateFestival` permite administratorului să editeze sau să șteargă un festival existent din listă. Toate modificările sunt trimise către server prin cereri HTTP, designul paginii fiind unul simplu. Ultima componentă este `QuestionAdmin.vue` care este dedicat gestionării întrebărilor din formular. Administratorul poate adăuga sau șterge întrebări, operațiile fiind realizate prin cererile HTTP către backend.

3.4.3.4 Vue Router

Routerul aplicației `FestivalGo` a fost implementat folosind biblioteca oficială `Vue Router`, care permite gestionarea navigării între paginile aplicației. Acesta a fost creat într-un fișier separat de paginile aplicației, unde s-a importat funcțiile `createRouter` și `createWebHistory`, precum și toate componentele de pagină.

S-a creat un vector care conține obiecte pentru fiecare rută:

- `path`: adresa URL
- `component`: componenta Vue care trebuie afișată pentru acea rută.
- `meta`: pentru restricții de acces

Se folosește `meta` pentru restricționarea de acces pe baza rolurilor. Se implementează un mecanism care verifică la fiecare navigare dacă ruta necesară necesită autentificare și dacă utilizatorul are rolul potrivit. Dacă utilizatorul nu are drepturile necesare, este

redirecționat către pagina de autentificare. Routerul este exportat și folosit în main.js unde este integrat în instanța Vue.

3.4.4 Funcționalități externe

3.4.4.1 Harta

Harta aplicației FestivalGo a fost implementată cu ajutorul bibliotecii Leaflet și este integrată atât în paginile de administrare cât și în paginile pentru un utilizator. La montarea componentei onMounted se creează o instanță Leaflet. Astfel, harta este atașată într-un div cu ID-ul, centrată pe o locație prestabilită. Adăugarea punctelor pe hartă este destinată doar administratorului, iar la fiecare apăsare a butonului pe hartă, se preiau coordonatele, latitudine și longitudine, și se asociază un emoticon selectat. Se creează un marker personalizat cu Leaflet, folosind un divIcon care afișează emoji-ul și o etichetă. Punctele adăugate sunt stocate într-o listă pentru a putea fi trimise ulterior către server.

În componenta de server a aplicației se creează un pachet nou destinat hărților. Aici se va crea clasa MapPoint care are un ID unic, latitudine, longitudine, emoticon și o denumire specifică a emoticonului. Se creează interfața repository și serviciul unde se implementează funcții de adăugare sau actualizare a acestor date. Controllerul este responsabil de cererile HTTP aferente.

După ce administratorul a plasat toate punctele dorite, acesta apasă un buton pentru a salva aceste puncte. Pentru fiecare punct, se trimite o cerere POST către backend datele: latitudine, longitudine, emoji și etichetă. În cazul unei actualizări se trimite lista de puncte cu o cerere PUT.

Atât în paginile de administrare, cât și în pagina de detalii a festivalului, harta afișează punctele salvate la încărcarea paginii, unde se preiau punctele de pe backend și se adaugă ca markere pe hartă, fiecare cu simbol vizual și eticheta corespunzătoare. Dacă există mai multe puncte, harta ajustează imaginea pentru a le include pe toate.

3.4.4.2 Stripe

Aplicația FestivalGo dispune de funcționalitate de plăți pentru cumpărarea biletelor la festivaluri. În primul rând, se creează un cont în platforma oficială Stripe și se vor căuta cheile secrete și publice. Cheia secretă se va copia și se va pune în server în fișierul de configurare application.properties din backend pentru a comunica în siguranță cu Stripe.

În backend, se creează un serviciu nou în pachetul de participări unde se va citi cheia și se va crea o sesiune de plată. Când un utilizator vrea să cumpere un bilet, frontend-ul apelează un endpoint din backend, iar acesta folosește cheia secretă pentru a crea o sesiune Stripe, apoi returnează către frontend un link sau un ID de sesiune Stripe. Se va implementa o metodă nouă în ParticipareService care va actualiza statusul unei participări dacă biletul a fost confirmat. Se implementează un controller nou pentru gestionarea cererilor HTTP strict pe baza plăților.

În interfața aplicației, se importă funcția loadStripe și se adaugă un buton pentru cumpărarea biletului. Când utilizatorul apasă butonul „Cumpără bilet”, este apelată funcția care preia ID-ul userului și a festivalului din localStorage, trimite o cerere POST către backend cu parametrii necesari. Se salvează temporar în localStorage datele despre utilizator și festival pentru confirmarea ulterioară a plății.

După ce backendul răspunde cu un URL de checkout Stripe, utilizatorul este redirecționat pe pagina de finalizare. Astfel, plata se face pe platforma Stripe asigurând securitatea tranzacției. După finalizarea plății, Stripe redirecționează utilizatorul către pagina de confirmare a plății.

În componenta PaymentSuccess.vue se trimite o cerere POST către backend pentru a confirma participarea și a marca biletul ca plătit. După confirmare, utilizatorul este redirecționat către lista de participări. Figura 3.4.4.2.1 ilustrează pașii principali explicați ai integrării Stripe în aplicația FestivalGo

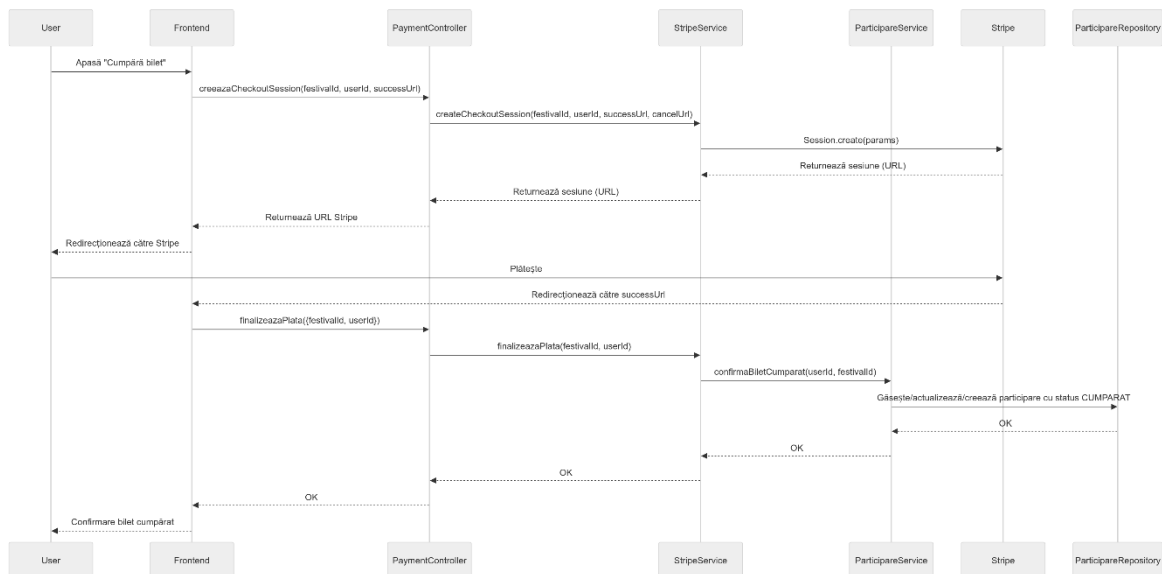


Figura 3.4.4.2.1: Fluxul logic pentru cumpărarea unui bilet în aplicația FestivalGo

3.5 Testare și validare

Această secțiune descrie procesul de testare a aplicației FestivalGo, atât din punct de vedere funcțional, cât și al performanței. Obiectivul principal a fost validarea corectitudinii implementării și verificarea comportamentului aplicației în diferite scenarii de utilizare.

Testarea serverului se va face în Postman. Prin intermediul acestui instrument, au fost verificate funcționalitățile esențiale precum vizualizarea festivalurilor, vizualizarea întrebărilor pentru chestionarul muzical și autentificarea.

Funcționalitatea de autentificare a fost testată în Postman. Scopul testării a fost verificarea comportamentului aplicației în cazul autentificării corecte sau incorecte pentru a declanșa limitarea ratei la autentificare.

În Figura 3.5.1 se poate observa o cerere de autentificare corectă pentru administratorul aplicației. Acesta s-a autentificat corect, iar sistemul răspunde cu codul 200 OK.

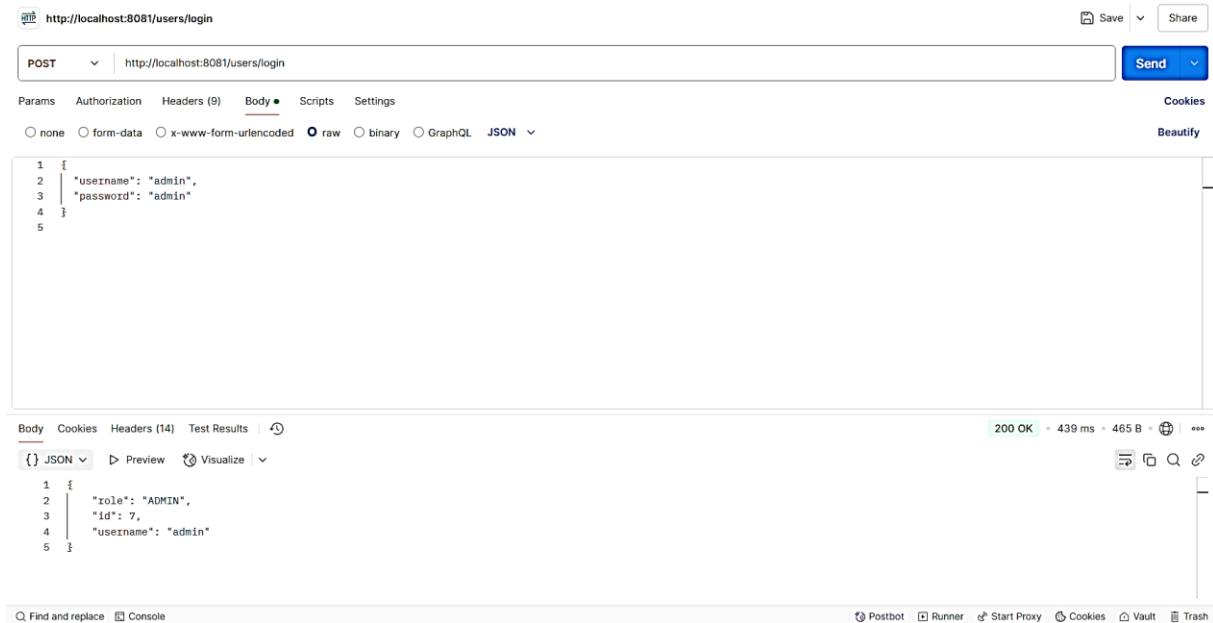


Figura 3.5.1: Testare autentificare reușită

Pentru a testa funcționalitatea limitării ratei s-a introdus un nume de utilizator și o parolă greșită de șase ori consecutive. Conform Figurii 3.5.2, primele cinci încercări sunt eșuate, iar la a șasea încercare, sistemul trimite un mesaj care blochează utilizatorul pentru un minut.

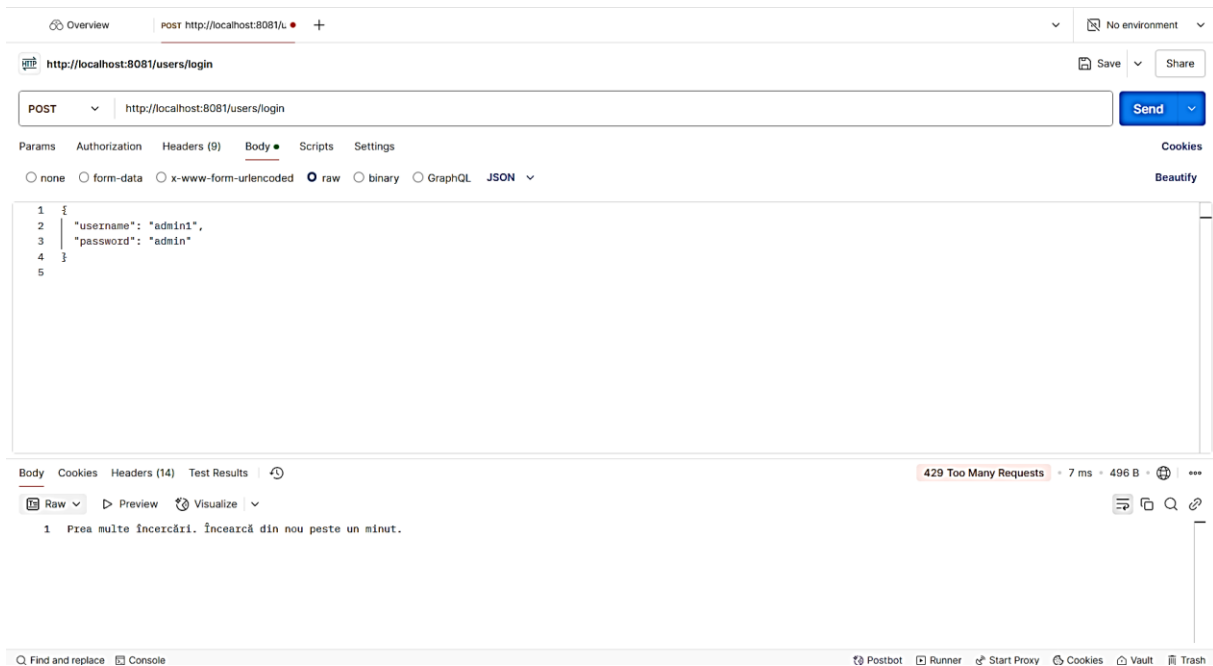


Figura 3.5.2: Serviciu de limitare a ratei

Afișarea festivalurilor a fost testată din Postman. Scopul testării a fost verificarea încărcării corecte a informațiilor din festivaluri. S-a trimis o cerere tip GET pentru vizualizarea tuturor festivalurilor complete. În Figura 3.5.3 se poate observa că metoda funcționează, contribuind semnificativ la scopul aplicației.

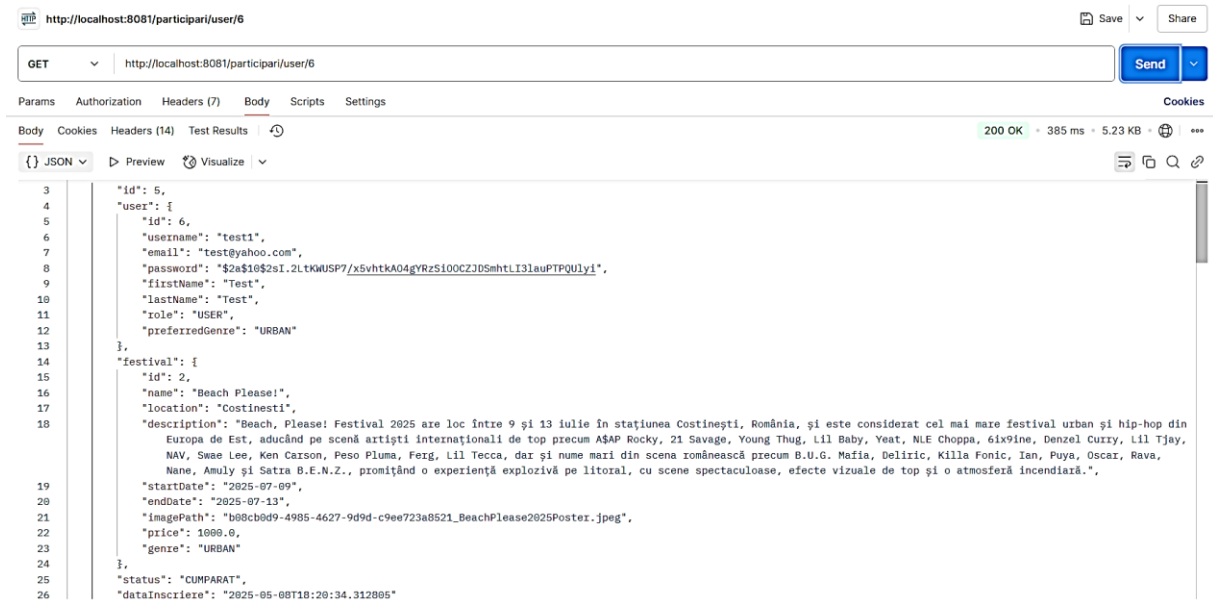


Figura 3.5.3: Vizualizare festivaluri existente

Această metodă a fost testată și din interfața aplicației pentru a se putea observa imaginile fiecărui festival așa cum este ilustrat în Figura 3.5.4.

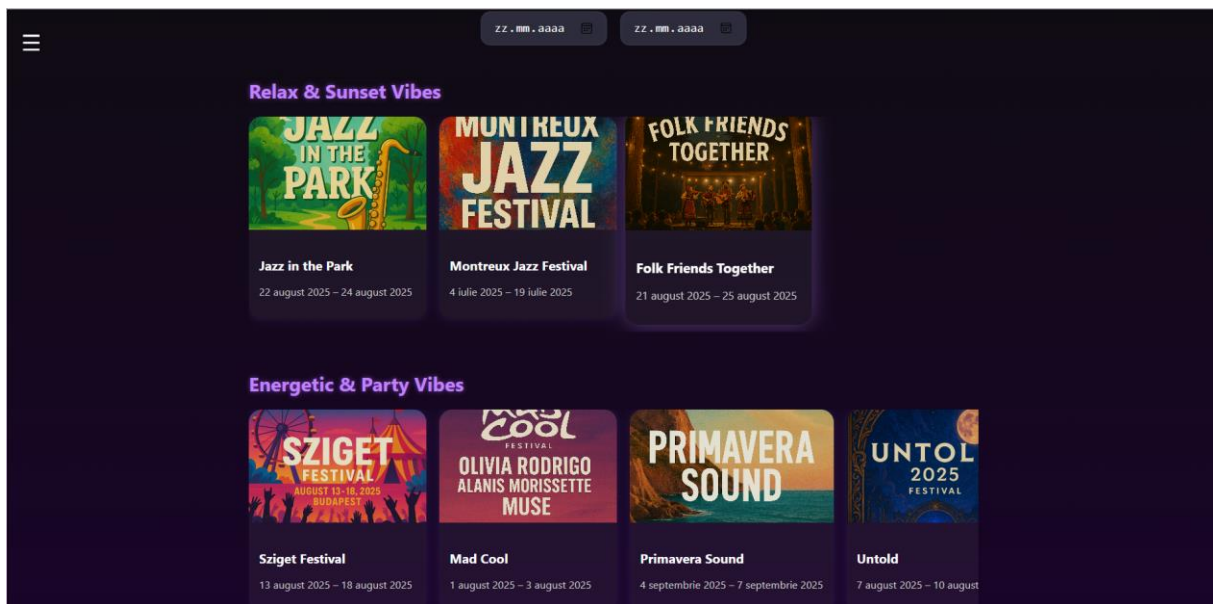


Figura 3.5.3: Vizualizare festivaluri existente din interfață

Pentru adăugarea unui cont nou pe platformă s-au testat condițiile impuse pentru crearea unui cont nou. Conform Figurii 3.5.5 se introduc în interfață date eronate și se apasă butonul de trimitere. Parola introdusă este prea scurtă, numele de utilizator este prea scurt. Implementarea validărilor sporește securitatea aplicației și ajută la prevenirea atacurilor de tip SQL Injection.

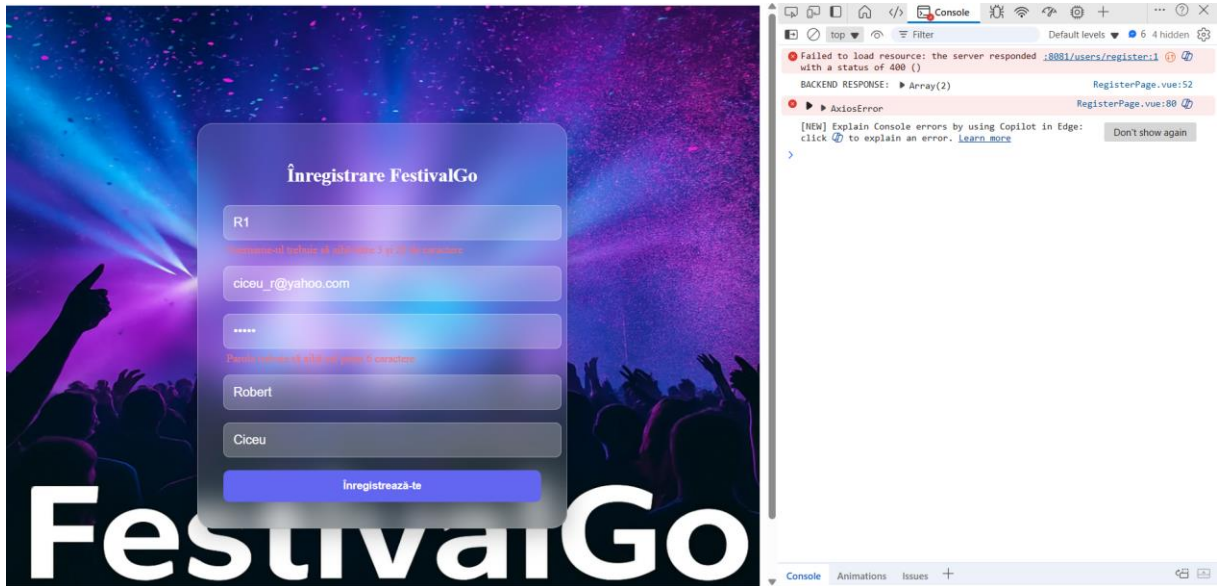


Figura 3.5.5: Autentificare greșită

Una din funcționalitățile de bază a aplicației este crearea unui nou festival împreună cu harta sa. Vom testa această funcționalitate folosind interfața aplicației. Se va completa în pagina de adăugare festival un festival nou cu toate datele specifice. În Figura 3.5.6 se poate observa că festivalul propus este procesat, cererea fiind trimisă cu succes.

localhost:5173 spune
Festival creat! Acum adaugă harta.

OK

SummerWell

Domeniul Știrbey, Buftea

08 . 08 . 2025

10 . 08 . 2025

Summer Well 2025 are loc între 8 și 10 august pe Domeniul Știrbey din Buftea, într-un cadru natu

595

Alegeți fișierul

fe13e3cb-0627-4ca4-a033-5fddaf1c239f.png

Gen muzical:

Indie

Creează festival

Figura 3.5.6: Crearea unui festival nou

După ce un festival nou este creat, pagina se extinde pentru ca administratorul să poată să pună punctele de interes pe hartă. În Figura 3.5.7 se poate observa că punctele se pot adăuga pe hartă.

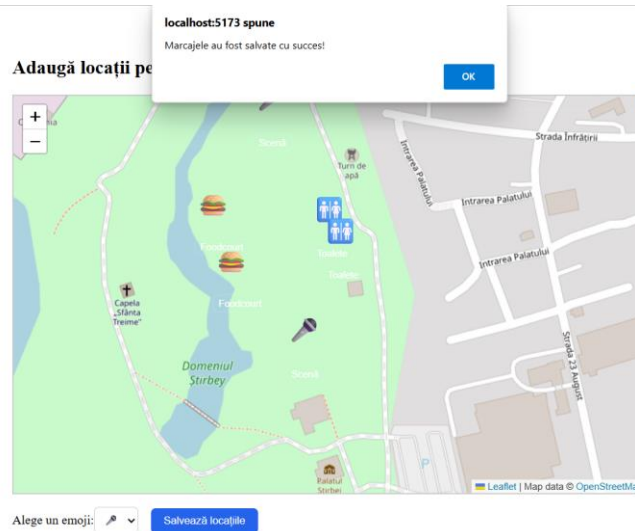


Figura 3.5.7: Crearea unui festival nou

Actualizarea unui festival este o altă funcție principală a aplicației FestivalGo. Spre exemplu, administratorul introduce greșit prețul biletului și dorește să îl actualizeze. În Figura 3.5.8 se testează această funcționalitate de actualizare a unei entități. Se poate observa mesajul de confirmare a schimbării prețului față de cel introdus inițial în Figura 3.5.6.

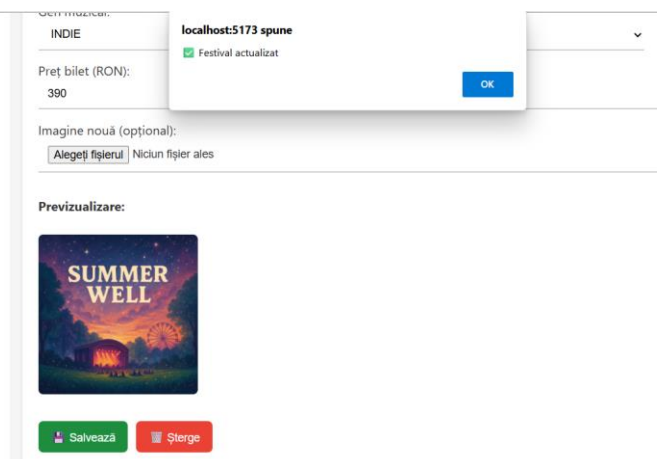


Figura 3.5.8: Actualizarea unui festival

Ștergerea unui festival este o acțiune posibilă în aplicația FestivalGo, implementată și testată. Dacă un festival se anulează, acesta nu ar mai trebui să existe în sistemul platformei. În Figura 3.5.9 se poate observa cum un festival anulat se poate șterge.



19. Ți place rafinamentul, detaliile subtile și eleganța?

☒ Da
☐ Nu

20. Ai o fire contemplativă, ți place să observi mai mult decât să vorbești?

☒ Da
☐ Nu

21. Ți place flexibilitatea, nu ți place să fii constrâns de reguli rigide?

☒ Da
☐ Nu

Trimite quiz-ul

Figura 3.5.11: Formularul de întrebări

După trimiterea chestionarului, genul muzical a fost calculat, iar pentru utilizator a fost vizibilă partea de recomandări muzicale. În cazul testului, genul muzical calculat a fost pop, ilustrat în Figura 3.5.12. Chestionarul muzical nu mai este valabil pentru un utilizator care a completat chestionarul

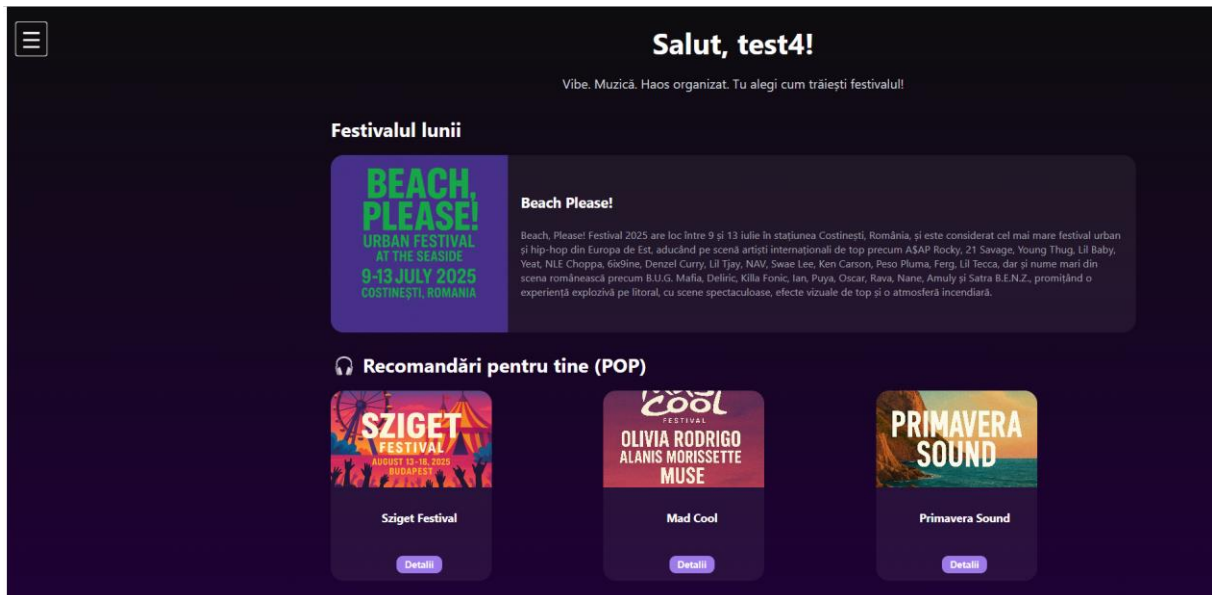
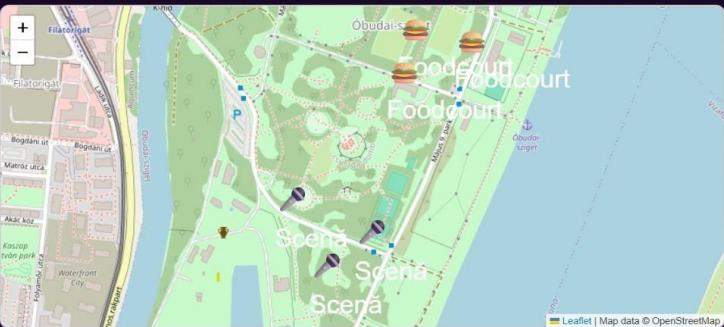


Figura 3.5.12: Recomandări

Ultimul test a fost pentru participarea la un festival și cumpărarea unui bilet. Utilizatorul poate intra pe detaliile festivalului recomandat, unde o să se afișeze harta și butoanele de participare și cumpărare bilet. Inițial se va crea o participare nouă. Această acțiune este ilustrată în Figura 3.5.13.

Descriere: Sziget Festival 2025 va avea loc în perioada 6-11 august 2025 pe celebra Óbudai-sziget (Insula Óbuda) din Budapesta, Ungaria. Cunoscut ca „The Island of Freedom”, Sziget este unul dintre cele mai mari și diverse festivaluri de muzică și cultură din Europa, atrăgând anual peste 400.000 de participanți din peste 100 de țări. Pentru ediția din 2025, organizatorii au anunțat deja o serie de headlineri impresionanți: Post Malone, Charli XCX, A\$AP Rocky, Shawn Mendes, Chappell Roan, Anyma.

Preț bilet: 799 RON



ID	data_inregistrarii	status	ticket_id	user_id
1	2025-03-10 12:04:44.810286	ANULAT	1	3
2	2025-04-07 18:18:12.712140	ANULAT	1	6
4	2025-04-09 13:35:46.750548	ANULAT	1	6
5	2025-05-08 18:30:34.312805	CUMPARAT	2	6
6	2025-05-10 11:24:31.270753	CUMPARAT	8	6
7	2025-05-17 22:15:57.303838	PARTICIPA	9	8
8	2025-06-18 13:27:17.512300	CUMPARAT	9	6
9	2025-06-18 15:18:38.174621	CUMPARAT	10	6
10	2025-06-18 15:35:31.982230	CUMPARAT	1	6
11	2025-06-18 15:46:58.983287	CUMPARAT	1	8
12	2025-06-18 15:46:58.983332	CUMPARAT	10	8
13	2025-06-18 20:23:15.036838	PARTICIPA	8	11


[Participă la acest festival](#)

[Cumpără bilet](#)

Figura 3.5.13: Participarea la un festival

După ce un utilizator a creat o participare, poate intra pe pagina „Participările mele” pentru a o vizualiza. Utilizatorul poate accesa butonul de cumpărare a unui bilet, așa cum se poate observa în Figura 3.5.14.

Participările mele



Sziget Festival
2025-08-13 – 2025-08-18

[Te-ai răzgândit?](#)

[Cumpără bilet](#)

Figura 3.5.14: Vizualizare participările mele

Prin apăsarea butonului de cumpărare, utilizatorul este trimis pe pagina de Stripe ilustrată în Figura 3.5.15 unde utilizatorul poate să completeze datele cardului.

Figura 3.5.15: Stripe

După finalizarea tranzacției, baza de date se actualizează cu statusul „CUMPARAT”, iar participarea se actualizează în interfață ca în Figura 3.5.16.

id	data_microne	status	festival_id	user_id
1	2025-03-10 12:04:44.933796	ANULAT	1	5
2	2025-04-07 18:18:12.753340	ANULAT	1	6
4	2025-04-09 13:35:46.730548	ANULAT	1	6
5	2025-05-08 18:20:24.320865	CUMPARAT	2	6
6	2025-05-10 11:24:31.270753	CUMPARAT	8	6
7	2025-05-17 22:15:57.301838	PARTICIPA	9	6
8	2025-06-18 13:27:17.632360	CUMPARAT	9	6
9	2025-06-18 15:16:38.274621	CUMPARAT	10	6
10	2025-06-18 15:16:31.982220	CUMPARAT	1	6
11	2025-06-18 15:16:56.963787	CUMPARAT	1	6
12	2025-06-18 15:46:56.388332	CUMPARAT	10	6
13	2025-06-18 20:22:15.930636	CUMPARAT	9	11

Figura 3.5.16: Plată efectuată

S-a evaluat performanța aplicației prin măsurarea timpului de răspuns al principalelor endpointuri REST, folosind Postman și consola browser-ului. Timpul de răspuns a fost calculat pentru operații GET, POST și DELETE, relevante pentru un utilizator obișnuit.

Pentru endpointul de logare, timpul de răspuns variază între 95 și 429 milisecunde, pentru afișarea tuturor festivalurilor 329 milisecunde, pentru ștergerea unui festival 180 milisecunde, pentru înregistrarea unui nou cont 394 milisecunde, pentru afișarea chestionarelor 120 milisecunde. Fișierele imagine pentru festivaluri sunt gestionate eficient, iar dimensiunea maximă admisă a fost extinsă la 10 MB pentru a preveni erorile de mărime a fișierelor.

4 Concluzii

4.1 Rezultate obținute

Lucrarea de față a avut ca scop dezvoltarea unei aplicații web full-stack moderne, orientate pe utilizator, dedicată participanților la festivaluri. Obiectivele lucrării au fost atinse, iar rezultatele obținute confirmă eficiența și utilitatea soluției implementate.

În urma implementării, s-au obținut rezultate semnificative din punct de vedere tehnic și funcțional. S-a realizat o platformă modernă, cu o interfață intuitivă, adaptată nevoilor reale ale publicului tânăr interesat de festivaluri. Utilizatorii beneficiază de o experiență captivantă, de la înscriere până la participarea efectivă la eveniment.

Aplicația permite autentificarea diferențiată în funcție de rol, o interfață dedicată pentru administratori și una pentru utilizatori, afișarea festivalurilor cu informații vizuale relevante, precum și participarea sau cumpărarea biletelor la evenimente. Un element distinctiv este integrarea hărților interactive, unde locațiile importante sunt marcate cu simboluri sugestive contribuind la o experiență de utilizare clară și ușor de utilizat.

Totodată, aplicația oferă un sistem simplificat de recomandări și posibilitatea de a crea grupuri de discuții între utilizatori, care diferențiază platforma web de alte platforme similare.

Printre avantajele aplicației web se numără flexibilitatea arhitecturii, care permite adăugarea ușoară de noi funcționalități. Un alt beneficiu major este dat de utilizarea Leaflet pentru integrarea hărților, ceea ce permite marcarea locațiilor reale cu un grad mare de personalizare.

În concluzie, realizarea platformei web FestivalGo a reprezentat un demers complet de proiectare și implementare a unei soluții online actuale, cu aplicabilitate concretă în domeniul evenimentelor muzicale. Proiectul a îmbinat partea tehnică cu cea creativă, reușind să ofere o aplicație ușor de folosit, adaptabilă și relevantă pentru utilizatorii de astăzi. Prin fiecare funcționalitate implementată s-a urmărit crearea unui sistem logic, accesibil și pregătit să evolueze în funcție de cerințele utilizatorilor și de posibilele direcții de dezvoltare ale aplicației.

4.2 Direcții de dezvoltare

Deși aplicația FestivalGo este complet funcțională și acoperă toate scenariile de utilizare propuse, există mai multe direcții de extindere. Într-o etapă viitoare de dezvoltare, ar putea fi implementat un sistem de convorbiri în timp real, care permite utilizatorilor să comunice direct, fără a fi necesară reîmprospătarea paginii. Această funcționalitate ar îmbunătăți semnificativ experiența de interacțiune.

O altă direcție de dezvoltare ar fi dezvoltarea unui algoritm de recomandare mai avansat, care să analizeze atât chestionarul muzical cât și comportamentul utilizatorului în aplicație, cum ar fi festivalurile vizualizate sau cele la care participarea este activă.

De asemenea, integrarea unui sistem de notificări în timp real, prin care utilizatorii să fie informați instant despre evenimente noi, mesaje primite sau modificări importante legate de participările lor.

Funcționalitatea de plată poate fi și ea extinsă, printr-un sistem complet de achiziționare a biletelor care să includă selectarea tipului de abonament pentru festival, trimiterea de confirmări pe e-mail și generarea automată a facturilor.

În plus, o secțiune dedicată recenziilor și evaluărilor ar permite utilizatorilor să își exprime opinia despre festivalurile la care au participat, contribuind astfel la îmbunătățirea recomandărilor.

Nu în ultimul rând, se intenționează dezvoltarea unei aplicații mobile dedicate FestivalGo, pentru a oferi utilizatorilor o experiență optimizată pe dispozitivele Android sau IOS.

5 Bibliografie

- [1] M. Bramhe, A. A. B. Waghmare, K. Rao, A. Kadu și D. T, „Online Event Management System: A critical Review of Reserch Findings and Methodologies,” *International Journal of Innovations in Engineering and Science*, vol. 9, nr. 5, pp. 11-13, 2024.
- [2] Ticketmaster, „Concerts and Tours,” [Interactiv]. Available: <https://www.ticketmaster.com/discover/concerts>. [Accesat Iun 2025].
- [3] Eventim.ro, „Bilete la concerte, festivaluri și spectacole în Romania,” [Interactiv]. Available: <https://www.eventim.ro/ro/>. [Accesat Iun 2025].
- [4] IaBilet.ro, „Evenimente și bilete online,” [Interactiv]. Available: <https://www.iabilet.ro/>. [Accesat Iun 2025].
- [5] A. Skandalis, E. Banister și J. Byrom, „Spatial Authenticity and Extraordinary Experiences: Music Festivals and the Everyday Nature of Tourism Destinations,” *Journal of Travel Research*, vol. 63, nr. 2, p. 357–370, 2024.
- [6] A. Luxford și J. E. Dickinson, „The Role of Mobile Applications in the Consumer Experience at Music Festivals,” *Event Management*, vol. 19, p. 33–46, 2015.
- [7] P. J. Rentfrow, L. R. Goldberg și D. J. Levitin, „The music of our lives: The role of musical preferences in predicting romantic attraction,” *Psychology of Music*, vol. 44, nr. 3, p. 573–587, 2016.
- [8] P. J. Rentfrow și S. D. Gosling, „The Do Re Mi’s of Everyday Life: The Structure and Personality Correlates of Music Preferences,” *Journal of Personality and Social Psychology*, vol. 84, nr. 6, p. 1236–1256, 2003.
- [9] J. Goslin, B. Joy, G. Stelle, G. Bracha și A. Buckley, „The Java Language Specification, Java SE 17,” 2021. [Interactiv]. Available: <https://docs.oracle.com/javase/specs/jls/se17/html/index.html>. [Accesat 06 2025].
- [10] Spring, „Spring Boot Reference Documentation,” 2024. [Interactiv]. Available: <https://docs.spring.io/spring-boot/>. [Accesat Iun 2025].
- [11] Apache, „What is Maven,” Apache Maven Project, 2024. [Interactiv]. Available: <https://maven.apache.org/what-is-maven>. [Accesat Iun 2025].
- [12] Spring, „Web MVC Framework,” Spring Boot Reference Documentation, 2024. [Interactiv]. Available: <https://docs.spring.io/spring-framework/reference/web/webmvc.html>. [Accesat Iun 2025].

- [13] A. T. Tuấn, „Reasons why Separation of Concerns (SoC) is essential in software development,” Medium, 2021. [Interactiv]. Available: <https://medium.com/tuanhdotnet/reasons-why-separation-of-concerns-soc-is-essential-in-software-development-5073de9d3c8f>. [Accesat Iun 2025].
- [14] Spring, „Overview,” Spring Integration Reference Documentation, 2025. [Interactiv]. Available: <https://docs.spring.io/spring-integration/reference/overview.html>. [Accesat Iun 2025].
- [15] Spring, „Building a RESTful Web Service,” Spring.io Guides, 2024. [Interactiv]. Available: <https://spring.io/guides/gs/rest-service/>. [Accesat Iun 2025].
- [16] Spring, „Mapping Requests,” Spring Framework Reference Documentation, [Interactiv]. Available: <https://docs.spring.io/spring-framework/reference/web/webmvc/mvc-controller/ann-requestmapping.html>. [Accesat Iun 2025].
- [17] Spring, „Enabling Cross-Origin Requests for a RESTful Web Service,” Spring.io Guides, 2025. [Interactiv]. Available: <https://spring.io/guides/gs/rest-service-cors/>. [Accesat Iun 2025].
- [18] Spring, „Spring Data JPA Reference Documentation,” 2024. [Interactiv]. Available: <https://docs.spring.io/spring-data/jpa/reference/index.html>. [Accesat Iun 2025].
- [19] Spring, „Defining Repository Interfaces,” Spring Data JPA Reference Documentation, 2025. [Interactiv]. Available: <https://docs.spring.io/spring-data/jpa/reference/repositories/definition.html>. [Accesat Iun 2025].
- [20] Spring, „Transactionality,” Spring Data JPA Reference Documentation, 2025. [Interactiv]. Available: <https://docs.spring.io/spring-data/jpa/reference/jpa/transactions.html>. [Accesat Iun 2025].
- [21] Spring, „Spring Security,” Spring Boot Reference Documentation, 2025. [Interactiv]. Available: <https://docs.spring.io/spring-security/reference/>. [Accesat Iun 2025].
- [22] V. Kalashnikov, „Bucket4j – Java rate-limiting library,” Bucket4j Documentation, [Interactiv]. Available: <https://bucket4j.com/8.14.0/toc.html#what-is-bucket4j/>. [Accesat Iun 2025].
- [23] Spring, „SQL Databases,” Spring Boot Reference Documentation-, 2025. [Interactiv]. Available: <https://docs.spring.io/spring-boot/reference/data/sql.html>. [Accesat Iun 2025].
- [24] C. Nair și R. R. Asha, „Postman for API Testing: A Comprehensive Guide for QA Testers,” *Data & Learn. Mach. Intell. Artif. J.*, vol. 1, nr. 1, pp. 14-18, 2022.

- [25] Mozilla Contributors, „JavaScript,” MDN Web Docs, [Interactiv]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/>. [Accesat 16 Iun 2025].
- [26] ECMA International, „ECMAScript® 2024 Language Specification – 15th Edition,” 2024. [Interactiv]. Available: <https://262.ecma-international.org/15.0/>. [Accesat 16 Iun 2025].
- [27] Vue.js Team, „Introduction-Vue.js Guide,” Vue.js Official Documentation, 2024. [Interactiv]. Available: <https://vuejs.org/guide/introduction.html>. [Accesat Iun 2025].
- [28] Mozilla Contributors, „HTML: HyperText Markup Language,” MDN Web Docs, [Interactiv]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTML/>. [Accesat Iun 2025].
- [29] Vue.js Team, „Introduction – Vue Router,” Vue.js Official Documentation, 2025. [Interactiv]. Available: <https://router.vuejs.org/guide/>. [Accesat Iun 2025].
- [30] Vite Team, „Why Vite?,” Vite Official Guide, 2024. [Interactiv]. Available: <https://vite.dev/guide/why/>. [Accesat Iun 2025].
- [31] Vite Team, „Build and Deploy,” Vite Official Guide, 2024. [Interactiv]. Available: <https://vite.dev/guide/build.html>. [Accesat Iun 2025].
- [32] Vite Team, „Features,” Vite Official Guide, 2024. [Online]. Available: <https://v3.vite.dev/guide/features.html>. [Accesat Iun 2025].
- [33] Axios, „Promise based HTTP client for the browser and Node.js,” Axios GitHub Documentation, [Interactiv]. Available: <https://axios-http.com/>. [Accesat Iun 2025].
- [34] LeafletJS, „Leaflet: an open-source Javascript library for interactive maps,” Official Documentation, 2024. [Interactiv]. Available: <https://leafletjs.com>. [Accesat Iun 2025].
- [35] LeafletJS, „Leaflet Quick Start Guide,” Official Documentatation, 2024. [Interactiv]. Available: <https://leafletjs.com/examples/quick-start/>. [Accesat Iun 2025].
- [36] K. Harsh, „A Guide to Stripe Integration in Spring Boot Application,” Kinsta Blog, 2025. [Interactiv]. Available: <https://kinsta.com/blog/stripe-java-api/>. [Accesat Iun 2025].
- [37] Stripe, „API Reference – Introduction,” Stripe Docs, 2024. [Interactiv]. Available: <https://stripe.com/docs/api/>. [Accesat Iun 2025].

- [38] Stripe, „Stripe Web Elements – Create your own checkout flows with prebuilt UI components,” Stripe Docs, 2025. [Interactiv]. Available: <https://docs.stripe.com/payments/elements/>. [Accesat Iun 2025].
- [39] Oracle Corporation, „What is MySQL?,” MySQL 8.4 Reference Manual, [Interactiv]. Available: <https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.html>. [Accesat Iun 2025].
- [40] Oracle Corporation, „FOREIGN KEY Constraints,” MySQL 8.4 Reference Manual, [Interactiv]. Available: <https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html>. [Accesat Iun 2025].
- [41] Corporation Oracle, „MySQL Workbench Product Page,” [Interactiv]. Available: <https://www.mysql.com/products/workbench/>. [Accesat Iun 2025].
- [42] MongoDB, „Understanding SQL vs NoSQL Databases,” © 2025 MongoDB, Inc, [Interactiv]. Available: <https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql/>. [Accesat Iun 2025].
- [43] Oracle, „What Is NoSQL?,” Oracle.com, [Interactiv]. Available: <https://www.oracle.com/database/nosql/what-is-nosql/>. [Accesat Iun 2025].