FestivalGo-Platformă web pentru organizarea si personalizarea participării la festivaluri muzicale

PROIECT DE DIPLOMĂ

Autor: **Ciceu Robert**

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

|  |  |
| --- | --- |
| DECAN  **Prof. dr. ing. Vlad MUREȘAN** | Vizat,  DIRECTOR DEPARTAMENT AUTOMATICĂ  **Prof. dr. ing. Honoriu VĂLEAN** |

Autor: **Prenume NUME**

Titlul lucrării

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

Semnătura autorului

Semnătura conducătorului științific

**Declarație pe proprie răspundere privind**

**autenticitatea proiectului de diplomă**

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

autorul lucrării:

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, 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 Prenume NUME

(semnătura)

**SINTEZA**

proiectului de diplomă cu titlul:

Titlul lucrării

Autor: **Prenume NUME**

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

1. Cerințele temei:

2. Soluții alese:

3. Rezultate obținute:

4. Testări și verificări:

5. Contribuții personale:

6. Surse de documentare:

Semnătura autorului

Semnătura conducătorului științific

Cuprins

[1 Introducere 2](#_Toc201423562)

[1.1 Context general 2](#_Toc201423563)

[1.2 Obiective 3](#_Toc201423564)

[1.3 Specificații 4](#_Toc201423565)

[2 Studiu bibliografic 7](#_Toc201423566)

[2.1 Analiza comparativă a platformelor existente 7](#_Toc201423567)

[2.2 Analiza articolelor științifice relevante pentru crearea aplicaței FestivalGo 8](#_Toc201423568)

[2.3 Technologii folosite pentru crearea backend-ului aplicației FestivalGo 9](#_Toc201423569)

[2.4 Technologii folosite pentru crearea frontend-ului aplicației FestivalGO 14](#_Toc201423570)

[2.5 Technologii folosite pentru crearea bazei de date a aplicației FestivalGo 17](#_Toc201423571)

[3 Analiză, proiectare, implementare 19](#_Toc201423572)

[3.1 Arhitectura generală a platformei 19](#_Toc201423573)

[3.2 Structura logică a platformei și organizarea modulelor 21](#_Toc201423574)

[3.3 Modelarea sistemului 22](#_Toc201423575)

[4 Concluzii 23](#_Toc201423576)

[4.1 Rezultate obținute 23](#_Toc201423577)

[4.2 Direcții de dezvoltare 23](#_Toc201423578)

[5 Reguli de formatare 24](#_Toc201423579)

[5.1 Formatarea paginii 24](#_Toc201423580)

[5.2 Titluri și stiluri 24](#_Toc201423581)

[5.3 Figuri, tabele și ecuații 25](#_Toc201423582)

[5.3.1 Figuri 25](#_Toc201423583)

[5.4 Tabele 25](#_Toc201423584)

[5.5 Ecuații 25](#_Toc201423585)

[6 Bibliografie 26](#_Toc201423586)

# Introducere

## 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. Într-o perioadă în care technoșpgia a schimbat complet modul în care comunicăm, festivalurile muzicale au reușit să rămână relevante tocmai pentru că oferă ceva ce lumea digitală nu poate reproduce: energia unei experiențe trăite împreună, în același loc, cu mii de oameni care împărtășesc aceeași pasiune.

Festivalurile nu mai sunt doar niște concerte în aer liber. Ele au devenit evenimente complexe, 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 tramsformat 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 vazute ca simple evenimente muzicale, ci ca experiențe personale, care rămân cu participanții mullt timp dupa ce se încheie muzica.

Odată cu aceste schimbări, și așteptările publicului s-au transformat. Cei care participă la cestivaluri doresc să fie implicați, să aibă acces la toate informațiile, să își personalizele experiența și să interacționeze ușor cu ceilalți. În general, se dorește o aplicație care să-i ajute să descopere festivaluri potrivite gusturilor lor, să îșoi organizele 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 dezvaluie 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 sun mai valoroase decât produsele, avem nevoie de soluții care țin pasul cu gernerațiile 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 quiz 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 participa, să completeze un chestionar interactiv pentru a-și identifica preferințele muzicale, să primească recomandări personalizate bazate pe răspunsuri, să vizualizele la fiecare festival harta cu punctele de interes (scenă, foodcourt, toalete, camping) și să interacționeze cu ceilalți utilizatori pentru a-și organiza cea mai bună experiență. Aceasta se diferențiaza față de alte aplicații prin chestionarele interactive unde userii răspund la douăzeci și unu de întrebari cu scopul de a determina genul muzical preferat. Pe baza informațiilor primite, FestivalGo poate să ofere recomandări personalizate, selectate special pentru fiecare utilizator. Această funcționalitate ajută la îmbunatațirea experienței beneficiarului, diferențiind aplicația față de altele.

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

## Obiective

Lucrarea de față își propune dezvoltarea unei aplicații web-full stack, denumită FestivalGo, adresată utilizatorilor pasionași de festivaluri muzicale care își doresc o experență digitală completă. Într-un peisaj în care majoritatea platformelor destinate evenimentelor se 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. Se optează pe o arhitectură pe niveluri, cu separare clară a responsabilitățiilor pentru folosirea componentelor. Această structură contribuie la menținerea codului într- stare clară din punct de vedere al vizibilității și permite extinderea sa ulterioară.
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 patricipanți 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, cu posibilitatea de a le adăuga sau șterge.
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. 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 urmarește implementarea unei metode de plată online care să permită utilizatorilor să își achiziționeze biletele direct din aplicație, într-un mod sigur și intuitiv.

## 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 in Vue.js, iar backend-ul se va realiza in Java, folosind framework-ul Spring Boot. Baza de date pe care o vom folosi este MySQL deoarece este o soluție efientă pentru modelarea și stocarea datelor aplicației.

Aplicația trebuie să permită înregistrarea și autentificarea utilizatorilor, cu atribuirea de roluri USER și ADMIN. Afișarea festivalurilor trebuie făcută sub formă categorii și să aibă posibilitatea de căutare. Utilizatorii vor avea posibilitatea de a putea să vizualizeze festivaluri, să participe la festivaluri sau să îsi cumpere bilet, să vizualizeze harta, să răspundă la chestionar și să creeze grupuri de prieteni pentru a discuta.

Pe se 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 dar și sigură. După efectuarea acestei plăți, statusul participării utilizatorului va fi actualizat automat în baza de date.

Chestionarul muzical trebuie stocat in baza de date și trebuie făcut pe baza răspunsurilor de Da sau Nu, fiecare răspuns avand un scor distinct. Acesta va fi contruit cu mare atenție pe baza unor analize profunde. Acesta va fi gestionat de administratorr

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 sa obținem o interfață cât mai simplu de utilizat, dar și plăcută vizual. Pe pagina principală se vor găsi elemente precum Festivalul Lunii si Recomandările tale. Se va utiliza un meniu pe pagina principală care va duce catre 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, folosim o bază de date relațională, construită cu MySQL. O folosim 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 in baza de date, Controller pentru a trimite endpointurile REST, Repository pentru interacțiunea cu baza de date și Service pentru a gestiona logica aplicației.

Ca element de securitate, parolele utilizatorilor for fi stocate în baza de date și vor fi criptate folosind framework-ul Spring Security și algoritmul Bcrypt, unul dintre cele mai bune și sigure algoritme disponibile. 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 catre baza de date a unor informații eronate. Insoțită de aceasta va fi si JPA care ajută la prevenirea atacurilor de tip SQL Injection.

Pe partea de frontend, când un utilizator se va autentifica, acestuia i se vor extrage in localStorage id-ul, username-ul și rolul. Evităm extragerea parolei pentru a evita riscul de logare neautorizată. Paginile vor fi securizate în frontend-ul aplicației, accesul către majoritatea paginilor fiind disponibilă doar dacă autentificarea a fost efectuată cu succes.

În versiunea actuală, aplicația FestivalGo va prezenta o serie de limitări care vor putea di abordate în versiuniile viitoare. În primul rand, aplicația nu dispune de un sistem de notificări în timp real, ceea ce înseamnă că utilizatorii nu vor putea primi mesaje instantanee sau actualizări live 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 un rating festivalurilor, ceea ce limitează interacțiunea și feedback-ul direct între utilizatori. De asemenea, aplicația necesită conexiunea permanentă la internet.

# Studiu bibliografic

Pentru realizarea aplicației FestivalGo s-au studiat multe articole științifice și documentații technice cu scopul de a identifica technologiile cele mai potrivite pentru dezvoltarea unei aplicații moderne și interactive. 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 decizile de proiectare a aplicației FestivalGO.

## 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 technologiilor utilizate, a experienței utilizatorului și a funcționalităților oferite. Această etapă este esențială pentru a evidenția elementele originale și crative propuse de aplicația FestivalGo.

Eventbrite este o platformă largă utilizată pentru gestionarea biletelor și promovarea evenimentelor, apreciată prin interfața sa prietenoasă și intuitivă. Platforma permite organizatorilor să gestioneze eficient participanții și să promoveze evenimentele prin intermendiul rețelelor sociale, oferind integrări solide cu aplicații precum Facebook și Salesforce [1]. Cu toate acestea, EventBrie nu include elemente interactive avansate precum hărți live ale locațiilor sau mecanisme de tip chestionar și nici nu oferă posibilitatea comunicării directe între participanți pruntr-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 formarea 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 a bilete, special dedicată utilizatorilor din Romania. Cu o interfață bine organizată și aplicație mobilă dedicată, Eventim facilitează achiziția de biletelor prin hărți statice ale sălilor și permite gestionarea comenzilor online. Totuși, în comparație cu FestivalGo, platforma nu integrează funcționalități moderne precum recomandpri muzicale.

IaBilet.ro [4] este una dintre cele mai utilizate platforme din Romania, 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, locație dată. Platforma oferă și o aplicație mobilă comăatibilă cu IOS și Android, care permite accesarea biletelor direct din telefon. Totuși, IaBilet.ro nu include funcționalități avansate precum recomandări muzicale, quiz de intentificare a preferințelor sau componente sociale.

## Analiza articolelor științifice relevante pentru crearea aplicaței FestivalGo

Pentru a înțelege mai bine contextul social și technologic al aplicației FestivalGo, au fost analizate mai multe lucrări științifice relevante din domeniul turismului, technologii 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 participanțiilor la festivalul Primavera Sound din Barcelona. Autorii propun concepul de autenticitate spațială, argumentând că participanții moderni nu caută o evadare total din cotidian, 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 contribuie la această autenticitate contextuală, facilitând integrarea evenimentelor în rutina cotidiană a utilizatorilor.

Un al doilea articol relevant este cel a lui Strand și Robertson [6],care analizează influența rețelelor sociale asupra relației dintre participanți și brandurie festivalurilor, cu accent pe generația tânără. Studiul de caz se concentrează pe festivalul TRNSMT și utilizează analiza rețelelor sociale pentru a evidenția modul în care interacțiunile digitale, cum ar fi postări, comentarii, partajări, contribuie la loialitatea față de festival și la consolidarea relației brand-utilizator. Aceste concluzii sprijină integrarea grupurilor și chat-urilor în aplicația FestivalGo.

Cel de-al treilea articol studiat este scris de Luxford și Dickinson [7], 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 și dezvăluie că participanții apreciază aplicațiile care le oferă libertate și control asupra propriei experiențe, cum ar fi program personalizat, hărți detaliate, posibilitatra de comunicare cu alți participanți. De asemenea se sublinează importanța echilibrului între funcționalitate și spontanietate.

Un punct important este evidențierea fazei de anticipare: aplicațiile sunt folosite cu mult timp înainte de festival pentru a planifica participarea , sporind entuziasmul utilizatorilor și implicarea în experență [7]. Aplicațiile trebuie să funcționeze rapid și să ofere acces minim , recomandându-se un design centrat pe utilizator [7].

Pentru a implementa chestionarul muzical, au fost luate în considerare câteva articole din domeniul psihologiei muzicii,aceste informații fiind foarte utile în dezvoltarea aplicațiie 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 constante de personalități [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 extroverții se orientează către muzica energetică și ritmată [8]. Aceste corelații pot fi folosite pentru a contrui chestionarul aplicației Festival Go.

Alt studiu relevant pentru dezvoltarea funcționalităților sociale ale aplicației FestivalGo este realizat de Rentfrow, GoldBerg și Levitin [9], 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.

Acestă 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.

Prin urmare,articolele analizate oferă un cadru teoretic coerent și actual care justifică atât necesitatea, cât și structura aplicației FestivalGo, demonstând că integrarea technologiilor moderne în experiențele de festival.

## Technologii folosite pentru crearea backend-ului aplicației FestivalGo

Pentru realizarea backend-ului aplicației au fost analizate diverse concepte cruciale din limbajul de programare Java. Scheletul principal ales este Spring Boot care este un framework destinat dezvoltării rapide a aplicațiilor Java.

Limbajul Java este descris formal în cartea „The Java Language Specification, Java SE 17” scrisă de James Gosling, Bill Joy, Guy Steele, Gilad Bracha și Alex Buckley [10].Conform documentației, Java un limbaj de programare versatil, bazat pe clase, orientat pe obiect și este structurat cât se poate de simplu pentru ca orice programator să-l înțeleagă [10]. Conform Capitolului 3 din documentația oficială [10], structura limbajului Java se face pe baza limbajelor C și C++, dar este adaptată pentru folosirea caracterelor Unicode, ceea ce permite scrierea aplicațiilor pe diverse platforme și multiple limbi.

În limbajul Java, conform Capitolului 4 din documentația oficială [10],datele sunt împărțite în 2 mari categorii principale: cele primitive și cele referențiale. Tipurile primitive includ valori numerice întregi în reprezentare, de exemplu, int și long, numerele cu virgulă mobilă, double sau float, cele de tip caracter Unicode , char, iar cele de tip boolean vor fi true sau false. Acestea subt definite să fie identice pe toate platformele și implementările Java.

Pe de altă parte, tot in Capitolul 4 din documentația oficială [10], 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 [10], descoperim 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 [10], 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, acest framework elimină setările repetitive care apar în dezvoltarea cu Spring Framework, oferind astfel un mediu de dezvoltare rapid și ușor de utilizat [11].

Unul dintre avantajele fundamentale ale folosirii Spring Boot este capacitatea sa de a se autoconfigura. Acest mecanism detectează automat dependențele aflate în classpath și configurează toate componentele necesare. Astfel,dezvoltatorul nu mai pierde timpul pe configurarea programului, ci se poate concentra pe logica și implementarea aplicației [11].

Spring boot include servere web integrate, cum ar fi Apache Tomcat [12].Acest lucru permite rularea aplicației fără nevoia de configurare a unui server extern. 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 [11].

Împreună cu Spring, vom se va folosi și Apache Maven care este un instrument ce ajută la contruirea și administrarea proiectelor software din mediul Java. Acesta inseamnă acumulator de cunoștințe și a fost creat inițial pentu a ușura procesele de contruire 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 [13].

Unul dintre avantajele utilizării framework-ului Spring Boot este serverul web,care este integrat automat în aplicație, eliminând astfel nevoia instalării unui nou server web. Potrivit documentației oficiale [12], Spring Boot utilizează implicit serverul web Apache Tomcat atunci când este adăugată dependența spring-boot-starter-web.Atfel, dezvoltatorii nu sunt nevoiți să descarce și să configureze manual un nou server web.

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 aplication.properties [12]. Această setare permite rularea pe portul specificat fiind utilă atunci când dorim rularea a mai multor aplicații. Pentru aplicația noastră, Festival Go, 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 sai Undertow.

În aplicațiile moderne se implementează adesea arhitectura pe straturi pe care o vom folosi și noi în aplicația noastră. 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 [14]. Conform articolului publicat pe Medium [14], principiul SoC vine cu multiple avantaje. Mentenabilitatea crește, modificarile pot deveni mai ușor de realizat. Se îmbunătățește testarea unitară, spre exemplu, testarea service-urilor se poate face fără a accesa baza de date. Scalabilitatea crește, fiecare strat poate fi refolosit pentru aplicații diferite [14].

În documentația oficială Spring Integration [15], se sublinează și aici importanța separării în straturi. Spring recomandă această arhitectură ca parte a bunelor practici în dezvoltarea aplicațiilor modulare și robuste [15].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 arthitectura clasică Model-Vedere-Controler (MVC), cât și posibilitatea dezvoltări unor servicii web conform principiilor REST [16]. 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 acestă arhitectură este partea responsabilă de gestionarea cererilor, adică acel cod care stabilește cum răspunde aplicația la difeirte slicitări venite de pe frontend. În loc sa fie trimise pagini vizuale, răspunsurile se vor trimite în format JSON. Această abordare este extrem de comună în aplicațiile moderne deoarece, permite separarea logică între partea de utilizator și partea logică [16]. Platforma Spring se ocupă automat de transformarea obiectelor interne ale aplicației în structuri JSON, eliminând nevoia de configurări complicate din partea dezvoltatorului. Această funcționalite este posibilă este posibilă datorită bibliotecii de integrare specializate în manipularea acestui tip de format [12].

Pentru a organiza răspunsurile aplicației în funcție de ce cere utilizatorul, Spring pune la dispoziție mecanisme de rutare. Acest sistem de legatură dintre adrese și leogica aplicației este foarte flexibil și permite inclusiv filtrarea în funcție de metoda folosită de utilizator [17]. Totodată, datele vin din adresă sau din bara de cautare pot fi preluate ușor din interiorul aplicației fără cod suplimentar [17].

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 automat schimbul de date care rulează pe adrese diferite. Pentru a rezolva această problemă, Spring oferă o funcționalitate nouă, CORS, 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. [18]

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 entitățiilor [19]. 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 permițând focalizarea pe logica de implementare. De asemenea, acest framework standardizează gestionarea erorilor [19].

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 va administra și cum este indentificată aceasta [20]. În cazul în care dorim să avem disponibile operațiuile standard de bază pentru entitatea respectivă, vom extinde direct interfața care le furnizează, în locul interfeței marker minimale [20].

CRUD este acronimul pentru operațiile fundamentale de Creare, Retragere, Actualizare și Stergere. 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 [20]. Există, de asemenea, mai multe variante 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 [20].

Pentru metode mai avansate, scrierea interogărilor este posibilă cu ajutorul adnotării @Query. Aceasta se foloseste atunci când o interogare este mai complexă sau când se dorește folosirea JPQL (Java Persistence Query Language) [20].

Odată definite repository- urile, gestionarea tranzacțiilor este foarte simplă pentru dezvoltator deoarece Spring Data JPA aplică implicit un component tranzacțional pe nodelele furnizate din aceste repository-uri. Operațiile care doar citesc date vor fi configurate automat să ruleze doar pentru citire, adica read-only, iar cele care modifică date,precum ștergerea sau salvarea, sunt configurate ca să ruleze într-o tranzacție standard [21].

Pentru securitatea aplicației FestivalGo se va folosi implementarea framewor-ului Spring Security. Acesta oferă mecanisme de autentificare, autorizare și protecție împotriva atacurilor comune, fiind considerat standardul facto pentru securizarea aplicațiilor Spring [22]. 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 [22].

Pentru protejarea parolelor utilizatorilor, Spring Security stochează parolele sub forma unui hash criptografic, nu text clar. Până la versiunea 5.0 , codificatorul impicit 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 [22].

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 [22]. 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 [22].

De asemenea, Bcrypt este conceput pentru a fi intenționat lent în procesarea hash-urilor, ceea ce îngreunează atacurile de tip forță brută [22]. 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 10, însă documentația oficială recomandă creșterea acestui factor, astfel încât verificarea parolei să dureze o secundp pe sistemul folosit [22].

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. Una dintre cele mai eficiente soluții pentru implementarea limitării ratei este Bucket4j [23]. Conform documentației oficiale, acest algoritm se bazează pe o „găleată virtuală” care conține un numar 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 [23].

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 driverele JDBC disponibile și configurează automat sursa de date. Astfel, driverul este disponibil, iar proprietățiile se pot seta din fișierul application.properties [24]. Acesta suportă, de asemenea, configurări avansate ale sursei de date, spre exemplu conexiunea la serverul Tomcat [12].

Spring Boot permite inițializarea automată a bazei de date folosind fișsiere SQL dedicate. În documentație [24], 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 [24].

## Technologii folosite pentru crearea frontend-ului aplicației FestivalGO

După examinarea acestor compomente esențiale în construirea backendului, se vor analiza tehnologiile principale utilizate pe partea de client. În centru acestei componente se afla JavaScript [25], limbajul de programare care servește drept fundament pentru interfața aplicației.

JavaScript, folosit atât pentru aplicații frontend , cât și pentru logica backend-ului, este printre cele mai populare limbaje de programare pentru dezvoltarea aplicațiilor web. Conform documentației oferite de Mozila Developer Network, 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 arhitecurile informatice. ECMA servește ca punte de legatură între caracteristicile JavaScript și alte technologii cu care interacționează, cum ar fi aplicațiile Java [10], 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.

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 in 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 degn modular, reutilizabil și scalabil al aplicației.

Vue.js este un frameork pentru dezvoltarea interfețelor de utilizator și este bazat pe HTML, CSS și Java Script. Acesta este conceput astfel încât să poată să fie adaptat treptat, pornind de la o pagina HTML simplă până la dezvoltarea unor aplicații complete de tip Single Page Application (SPA), prin utilizarea unor instrumente moderne precum Vue Router și alte librarii 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 funcționaități 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 compleze, menținând în același timp simplitatea de bază, atunci când este utilizat împreună cu ecosistemul complet de instrumente. Acesta utilizează un sistem reactiv intern care urmarește automat starea aplicației și actualizează DOM-ul virtual atunci când datele se modifică [27].

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

În proiectul nostru vom 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 intrumemente moderne cum ar fi Axios, folosit pentru cererile HTTP sau Leaflet, pentru integrarea unor hărți interactive.

Pe partea de frontend, 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 modificarile din cod să fie imediat observate în interfață, fără a reîncărca întreaga aplicație [30].

Acest comportament este posibil deoarece Vite spară dependențele externe de codul sursă. Dependențele sunt pre-incluse cu ajutorul unui compilator care este 10-100 de ori mai rapid decât alte compilatoare obisnuite [30]. În același timp, fișierele sursă sunt servite direct browserului 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 technici 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 dintre backend și frontend vom folosi 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. Bibiloteca, care a fost concepută inițial de Vladimir Agafonkin, oferă o interfață prietenoasă, performanță ridicată și compatibilitate extinsă cu browserele și dispozitivele mobile moderne. 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 inserrarea cordonatelor GPS reale și inserarea de emoji-uri sau icoane personalizate pe hartă. Leaflet oferă o experiență interactivă fără a afecta viteza de încărcare sau compatibilitatrea 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 librariei Stripe. Stripe este o platformă de procesare a plățiilor bine recunoscută la nivel global, dispobibilă în zeci de țări, adesea fiind preferată de dezvoltatatori datorită acoperirii sale extinse [36]. Sistemul de plăți oferite de Stripe permite comercianțiilor 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 îm 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 redirecționarea către o altă pagină de plată dispusă de Stripe [36]. Varianta redirecționată 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].

## Technologii 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 modial [39]. Dezvoltat și susținut de Oracle Croporation, 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 referențierea datelor dintr-o tabelă în alta și asigură consistența informațiilor între tabele folosind constângeri [40]. Un mare avantaj al folosirii MySQL este natura open-source, ceea ce înseamnă că software-ul poate fi utilizat gratuit.

MySQL are o arhitectură de tip client-server, componenta centrală find 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 Workbanch este o aplicație grafică grafică unificată care oferă instrumente vizuale oentru proiectarea, dezvoltarea și administrarea bazelor de date MySQL [41]. Workbanch permite atâta 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 reunesc date din mai multe tabele, ceea ce este esențial pentru a extrage informații corelate. Bazele de date noSQL, în schimb, nu oferă în mod obijnuit 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.

# 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 faza de analiză inițială și proiectare, până la implementarea concretă și testarea funcționalitățiilor.În timp ce primele capitole au oferit o descriere a contextului teoretic și a technologiilor utilizate, această secțiune oferă, în detaliu, technicile 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.

Totodată, vor fi prezentate diagrame UML, schema bazei de date, configurații esențiale, precum și capturi din aplicație și exemple de cod relevante. În final, se vor testa aceste funcționalități pentru a evidenția calitatea soluției propuse.

## 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 participărike. Arhitectura aplicației este de tip client-server, fiind structurată în componente distincte, fiecare cu rolul său bine definit.

În figura urmatoare (*Figura 3.1)* este ilustrată schema arhitecturală generală a aplicației FestivalGo, realizată conform principiilor de separare a responsabilitățiilor, asa 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 funcționalitățiile 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, technologie detaliată anterior în capitolul doi. Serverul expune o serie de endpoint-uri 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 crescu de protecție a parolelor personale.

Conform principiilor relaționale rezentate în studiul bibiografic, 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 sunt implementate în codul serverului prin intermediul JPA și gestionate cu ajutorul repository-urile Spring Data.

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

A screenshot of a computer

AI-generated content may be incorrect.

Figura 3.1: Schema generală

## 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 obisnuiți (care doresc să exploreze și să participe la festivaluri), cât și administratorilor (care gestionează conținutul platformei), astfel a fost necesară definirea clară a funcționalitățiilor 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 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 hărțiile 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 frontend.

Rolurile au fost selectate atât în baza de date cât și în comportamentul aplicației în frontend unde, în funcție de rolul identificat dupa autentificare, persoana în cauză este redirecționat 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:

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

Îm backend, fiecare modul menționat anterior este implementat în propriul său pachet care va conține entități JPA, controlere REST, servicii și repository-uri.

Fiecare modul are în backend propriile entități, servicii, controlere și repository-uri.Pachetele de autentificare și chestionar au incluse și DTO-uri, pentru a transmite datele între server și client într-un mod sigur.

Relațiile dintre entitățiile identificate din baza de date relaționlă for fi modelate prin adnoturiile @OneToMany (ex: un festival are mai multe participări), @ManyToOne (ex: o participare aparține unui singur user) și @ManyToMany (ex: utilizatorii în grupuri).

## Modelarea sistemului

Pentru a întelege funcționalitatea aplicației FestivalGo și modul în care interacționează cu ea, au fost realizate mai multe diagrame UML. Acestea oferă o reprezentare vizuală a comportamentelor sistemului și a relaților dintre ele ce apar în timpul utilizării platformei.

# Concluzii

## Rezultate obținute

Evidențiați toate rezultatele pe care le-ați obținut și trageți concluzii din ele. Puteți prezenta o analiză critică a ceea ce ați realizat comparativ cu alte lucrări/studii anterioare.

Includeți o listă a contribuțiilor pe care le-ați avut în domeniul temei abordate.

## Direcții de dezvoltare

Descrieți direcțiile posibile de dezvoltare.

# Reguli de formatare

## Formatarea paginii

* + Dimensiunea paginii: A4
  + Margini: 2.5 cm (sus, jos, stânga, dreapta)
  + Antet și subsol: 1.27 cm de la marginea paginii
  + În antetul paginii (header): titlul capitolului, centrat, stil: Header\_style
  + În subsolul paginii: numărul paginii, centrat

## Titluri și stiluri

Titlurile capitolelor și subcapitolelor se marchează cu stilurile Heading 1 – 4, conform documentului model anexat în format Word. Descrierea stilurilor utilizate în document este prezentată în Tabelul 5.1.

Tabelul 5.1. Stiluri utilizate în acest document

| Nr. | Stil | Utilizat pentru | Format |
| --- | --- | --- | --- |
| 1 | Normal | Text normal | Font: (Default) Cambria, 12 pt, Justified, Line spacing: Multiple 1.1 li, Space After: 6 pt |
| 2 | Titlu | Titlul proiectului, prima pagină | Font: 24 pt, Small caps, Centered Line spacing: single, Space Before: 126pt, After: 0 pt, |
| 3 | Titlu2 | Titlul proiectului, pagina de prezentare | Font:14pt, Bold, Centered |
| 4 | Heading 1 | Titlurile capitolelor (nivel 1) | Font: 24 pt, Indent: Left: 0 cm Hanging: 0.76 cm, Space Before: 24pt, After: 12pt |
| 5 | Heading 2 | Titlurile subcapitolelor (nivel 2) | Font: 14 pt, Bold, Indent: Left: 0 cm  Hanging: 1.02 cm, Space Before: 18pt, After: 12pt |
| 6 | Heading 3 | Titlurile secțiunilor (nivel 3) | Font: Bold, Indent: Left: 0 cm Hanging: 1.27 cm, Space Before: 6 pt, After: 6pt |
| 7 | Heading 4 | Titlurile secțiunilor (nivel 4) | Font: Italic, Indent: Left: 0 cm Hanging: 1.52 cm, Space Before: 2 pt, After: 0 pt |
| 8 | Caption | Legenda figurilor și tabelelor | Font: Italic, Font color: Text 1, Line spacing: single, Space After: 10 pt, |
| 9 | Header\_style | Antetul paginii | Font: 10 pt, Italic, Centered, Border: Bottom: (Single solid line, Background 1, 0.5 pt Line width) |

## Figuri, tabele și ecuații

### Figuri

Figurile se inserează în text centrate, cu etichetă de numerotare și legendă (Caption) în partea de jos a figurii. Numărul figurii include și numărul capitolului, după exemplul prezentat în Figura 5.1.



Figura 5.1. Figură exemplu, stil: Caption

## Tabele

Tabelele se inserează în text centrate, cu etichetă și legendă (Caption) în partea de sus a tabelului, aliniată la stânga. Numărul tabelului include și numărul capitolului, după cum este prezentat, de exemplu, în Tabelul 5.1.

## Ecuații

Ecuațiile se inserează în text centrate, cu numerotare în partea dreaptă. Numărul ecuației include și numărul capitolului, conform exemplului din relația (5.1).

|  |  |
| --- | --- |
|  | (5.1) |

# 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. |
| [3] | Eventim.ro, „Bilete la concerte, festivaluri și spectacole în Romania,” [Interactiv]. Available: https://www.eventim.ro/ro/. |
| [4] | IaBilet.ro, „Evenimente și bilete online,” [Interactiv]. Available: https://www.iabilet.ro/. |
| [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. Strand și M. Robertson, „An Attitudinal Impacts Analysis of Social Media Platforms and Brand Relationship Quality at Music Festivals,” *Event Management,* vol. 24, p. 769–788, 2020. |
| [7] | 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. |
| [8] | S. D. G. P. J. Rentfrow, „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] | L. R. G. a. D. J. L. P. J. Rentfrow, „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. |
| [10] | 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]. |
| [11] | Spring, „Spring Boot Reference Documentation,” 2024. [Interactiv]. Available: https://docs.spring.io/spring-boot/. [Accesat 06 2025]. |
| [12] | Spring, „Web MVC Framework,” Spring Boot Reference Documentation, 2024. [Interactiv]. Available: https://docs.spring.io/spring-framework/reference/web.html. [Accesat 06 2025]. |
| [13] | Apache, „What is Maven,” 2024. [Interactiv]. Available: https://maven.apache.org/what-is-maven.html. [Accesat 06 2025]. |
| [14] | T. Nguyen, „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 06 2025]. |
| [15] | Spring, „Spring Integration Reference Documentation – Overview,” 2025. [Interactiv]. Available: https://docs.spring.io/spring-integration/reference/overview.html. [Accesat 06 2025]. |
| [16] | Spring, „Building a RESTful Web Service,” Spring.io Guides, 2024. [Interactiv]. Available: https://spring.io/guides/gs/rest-service. [Accesat 06 2025]. |
| [17] | Spring, „Mapping Requests,” Spring Framework Reference Documentation, [Interactiv]. Available: https://docs.spring.io/spring-framework/reference/web/webmvc/mvc-controller/ann-requestmapping.html. |
| [18] | 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 06 2025]. |
| [19] | Spring, „Spring Data JPA Reference Documentation,” 2024. [Interactiv]. Available: https://docs.spring.io/spring-data/jpa/reference/index.html. [Accesat 06 2025]. |
| [20] | Spring, „Spring Data JPA Reference Documentation - Defining Repository Interfaces,” 2025. [Interactiv]. Available: https://docs.spring.io/spring-data/jpa/reference/repositories/definition.html. [Accesat 06 2025]. |
| [21] | Spring, „Spring Data JPA Reference Documentation - Transactionality,” 2025. [Interactiv]. Available: https://docs.spring.io/spring-data/jpa/reference/jpa/transactions.html. [Accesat 06 2025]. |
| [22] | Spring, „Spring Security- Reference Documentation,” 2025. [Interactiv]. Available: https://docs.spring.io/spring-security/reference/. [Accesat 06 2025]. |
| [23] | V. Kalashnikov, „Bucket4j – Java rate-limiting library,” Bucket4j Documentation, [Interactiv]. Available: https://bucket4j.com/8.14.0/toc.html#what-is-bucket4j. |
| [24] | Spring, „Spring Boot Reference Documentation- SQL Databases,” 2025. [Interactiv]. Available: https://docs.spring.io/spring-boot/reference/data/sql.html. [Accesat 06 2025]. |
| [25] | Mozila Contributors, „JavaScript,” MDN Web Docs, [Interactiv]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accesat 16 06 2025]. |
| [26] | ECMA International, „ECMAScript® 2024 Language Specification – 15th Edition,” Jun 2024. [Interactiv]. Available: https://262.ecma-international.org/15.0/. [Accesat 16 06 2025]. |
| [27] | Vue.js Team, „Introduction-Vue.js Guide,” Vue.js Official Documentation, 2024. [Interactiv]. Available: https://vuejs.org/guide/introduction.html. [Accesat 06 2025]. |
| [28] | Mozilla Contributors, „HTML: HyperText Markup Language,” MDN Web Docs, [Interactiv]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML. |
| [29] | Vue.js Team, „Introduction – Vue Router,” Vue.js Official Documentation, 2025. [Interactiv]. Available: https://router.vuejs.org/guide/. |
| [30] | Vite Team, „Why Vite?,” Vite Official Guide , 2024. [Interactiv]. Available: https://vite.dev/guide/why. |
| [31] | Vite Team, „Buid and Deploy,” Vite Official Guide, 2024. [Interactiv]. Available: https://vite.dev/guide/build.html. |
| [32] | Vite Team, „Features,” Vite Official Guide, 2024. [Interactiv]. Available: https://v3.vite.dev/guide/features.html. |
| [33] | Axios, „Axios - Promise based HTTP client for the browser and Node.js,” Axios GitHub Documentation, [Interactiv]. Available: https://axios-http.com/. |
| [34] | LeafletJS, „Leaflet: an open-source Javascript library for interactive maps,” Official Documentation, 2024. [Interactiv]. Available: https://leafletjs.com/. |
| [35] | LeafletJS, „Leaflet Quick Start Guide,” Official Documentatation, 2024. [Interactiv]. Available: https://leafletjs.com/examples/quick-start/. |
| [36] | K. Harsh, „A Guide to Stripe Integration in Spring Boot Application,” Kinsta Blog, 5 feb 2025. [Interactiv]. Available: https://kinsta.com/blog/stripe-java-api/. |
| [37] | Stripe, „API Reference – Introduction,” Stripe Docs, 2024. [Interactiv]. Available: https://stripe.com/docs/api. |
| [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. |
| [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. |
| [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. |
| [41] | Corporation Oracle, „MySQL Workbench Product Page,” [Interactiv]. Available: https://www.mysql.com/products/workbench/. |
| [42] | MongoDB, „Understanding SQL vs NoSQL Databases,” [Interactiv]. Available: https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql. |
| [43] | Oracle, „What Is NoSQL?,” Oracle.com, [Interactiv]. Available: https://www.oracle.com/database/nosql/what-is-nosql. |