

**UNIVERSITATEA „POLITEHNICA” din BUCUREȘTI**

**Facultatea de Electronică, Telecomunicații și Tehnologia Informației**

**Proiect2**

**Programarea Interfețelor pentru Baze de Date**

**Student: Zaharia Roxana-Florina Prof. coordonator:**

**Grupa 435B Ș.l. Dr. Ing. Pupezescu Valentin**

**București 2025**

Cuprins

[**1.** **Cerinta proiectului** 3](#_Toc157370399)

[**2.** **Tehnologii utilizare** 3](#_Toc157370400)

[**3.** **Descrierea aplicatiei** 4](#_Toc157370401)

[**4.** **Functionalitatea aplicatiei** 7](#_Toc157370402)

[**5.** **Implementarea functiilor** 9](#_Toc157370403)

[**6.** **Front-end** 14](#_Toc157370404)

[**7.** **Concluzii** 17](#_Toc157370405)

[**8.** **Bibliografie** 17](#_Toc157370406)

# **Cerinta proiectului**

Creaţi două aplicaţii care să conţină o bază de date creată în sistemul de gestiune a bazelor de date MySql şi două interfeţe la aceasta (baza de date este comună). La crearea interfeţelor se vor folosi două tehnologii(la alegere - ex.: JSP, Hibernate, JPA, .NET, Python, JavaFX, Spring Boot etc.). Baza de date va fi compusă din tabelele stabilite în lista de teme. Tot în fişierul cu teme sunt stabilite şi asocierile dintre tabele. Interfeţele vor trebui să permită utilizatorului să execute următoarele operaţii pe toate tabele(inclusiv pe cele de legătura dacă aveti asocieri de tip M:N): vizualizare, adăugare, modificare şi ştergere de date. Vizualizarea tabelelor de legătură va presupune vizualizarea datelor referite din celelalte tabele.

Scopul aplicațiilor constă în facilitarea utilizatorilor pentru efectuarea operațiunilor de bază de date, precum vizualizarea, adăugarea, modificarea și ștergerea datelor. Se presupune că interfețele vor oferi o vizualizare eficientă a datelor interconectate între diferite tabele, evidențiind relațiile dintre acestea.

# **Tehnologii utilizare**

**Descrierea sistemului de gestiune a bazelor de date MySQL**

* *MySQL* este un sistem de gestionare a bazelor de date relaționale (RDBMS) open-source, dezvoltat pentru a oferi un mediu eficient și fiabil pentru stocarea și manipularea datelor.
* *MySQL* este un sistem de gestiune a bazelor de date (SGBD) open-source, care este apreciat pentru accesibilitatea, ușurința de utilizare și costuri reduse, datorită naturii sale gratuite. Este recunoscut pentru performanțele rapide și pentru fiabilitatea în gestionarea volumelor mari de date, fiind astfel adecvat atât pentru proiecte mici, cât și pentru utilizare la scară largă. MySQL suportă o varietate de limbaje de programare și rulează pe diverse sisteme de operare, oferind scalabilitate și flexibilitate în dezvoltarea aplicațiilor web. Securitatea robustă integrată și comunitatea vastă de utilizatori contribuie la popularitatea sa continuă ca o soluție de bază de date preferată în industrie.

**Java Spring Boot** este un framework open-source pentru dezvoltarea aplicațiilor enterprise și web, conceput pentru a simplifica crearea, testarea și întreținerea aplicațiilor Java. Este renumit pentru flexibilitatea, modularitatea și capacitatea de a facilita dezvoltarea rapidă.Dintre caracteristicele sale importante, pot aminti:

* **Programare Aspect-Oriented (AOP):** Permite separarea logicii transversale (de exemplu, logare, securitate).
* **Integrare cu baze de date:** Suport pentru ORM-uri precum Hibernate și Spring Data pentru gestionarea datelor.
* **Spring MVC:** Dezvoltarea aplicațiilor web și API-uri REST.
* **Spring Boot:** Oferă configurare automată și permite crearea rapidă de aplicații standalone.
* **Spring Security:** Gestionarea autentificării și autorizării.
* **Suport pentru microservicii:** Integrare cu Spring Cloud pentru dezvoltarea arhitecturilor distribuite.

Java Spring este apreciat pentru performanță, scalabilitate și suportul extins pentru dezvoltarea modernă a aplicațiilor.

**HTML & CSS**

* *HTML* (HyperText Markup Language) și *CSS* (Cascading Style Sheets) sunt două tehnologii fundamentale pentru crearea paginilor web. HTML este limbajul de bază utilizat pentru a structura conținutul pe web, permițând dezvoltatorilor să definească diferite părți ale unei pagini, cum ar fi paragrafe, titluri, link-uri și imagini. Pe de altă parte, CSS este utilizat pentru a controla designul vizual al acestor elemente HTML, inclusiv layout-ul, culorile, fonturile și animațiile.

# **Descrierea aplicatiei**

Ca temă am abordat gestionarea trupelor, si a festivalelor , creând o interfață Web ce operează baza de date în care sunt stocate toate datele necesare. Funcționalitățile implementate în cadrul acestei etape a proiectului se enumeră vizualizarea datelor, adăugarea lor, modificarea și posibilitatea ștergerii acestora, într-o manieră elegantă și intuitivă.

**Implementare**

Baza de date a fost realizată cu ajutorul MySQL Workbench, fiind alcătuită din două tabele: *tabela si festival*. Pentru a realiza o asociere între aceste două tabele de tip M:N, a fost creată o tabelă intermediară numită *lista*.

**Structura coloanelor tabelelor:**

1. Tabela *trupa*
   * idtrupa ( INT, PK, NN, AI)
   * nume ( VARCHAR(45) )
   * gen\_muzical ( VARCHAR(45) )
   * nr\_membrii ( VARCHAR(45) )
   * producatorFilm ( VARCHAR(45) )
2. Tabela *festival*
   * idfestival ( INT, PK, NN, AI)
   * nume ( VARCHAR(45) )
   * data ( VARCHAR(45) )
   * pret\_bilet ( VARCHAR(45) )
3. Tabela *lista*
   * idlista ( INT, PK, NN, AI)
   * idfestival ( INT)
   * idtrupa ( INT)
   * ora\_trupa ( VARCHAR(45) )
   * dansatori ( VARCHAR(45) )
   * scena ( VARCHAR(45) )

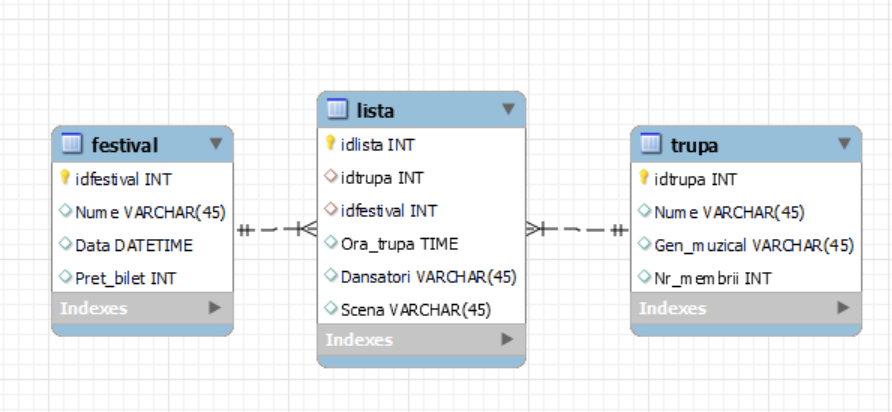


Fig. 1 – Diagrama EER a tabelelor și conținutului acestora

**Ce relatii exista intre cele trei tabele?**

* Intre **festival** si **lista** exista o asociere de tip 1:N
* Intre **trupa** si **festival** exista o asociere de tip M:N
* Intre **trupa** si **lista** exista o asociere de tip 1:N

Datorită asocierii M:N, *idtrupa* și *idfestival* sunt setate ca chei străine (foreign keys) în cadrul tabelei *lista*.

O imagine care conține text, număr, Font, linie

Descriere generată automat

# **Functionalitatea aplicatiei**

**Arhitectura proiectului**

**1. Model (Entity Layer)**

Reprezintă tabelele din baza de date utilizând clase POJO (Plain Old Java Objects), mapate prin JPA (Java Persistence API). Modificările din aceste clase se reflectă automat în baza de date.

* **Clasele din com.example.demo.entity:**
  + Trupa.java:
    - Reprezintă tabela *Trupa*.
    - Atribute precum idTrupa, nume, genMuzical.
  + Festival.java:
    - Reprezintă tabela *Festival*.
    - Atribute precum idFestival, numeFestival, dataFestival.
  + Lista.java:
    - Reprezintă tabela *Lista*.
    - Relaționează cu Trupa și Festival (relații Many-to-One).

**2. View (Thymeleaf - Frontend-ul)**

Partea de vizualizare este realizată utilizând **Thymeleaf**, împreună cu HTML și Bootstrap. Aceasta permite afișarea datelor din baza de date și interacțiunea cu utilizatorul.

* **Paginile din templates:**
  + **lista.html:**
    - Afișează datele din tabela *Lista*.
    - Permite adăugarea, modificarea și ștergerea de date.
  + **create\_lista.html și edit\_lista.html:**
    - Formulare pentru crearea și editarea unei liste.
  + Paginile trupa.html și festival.html sunt similare pentru entitățile respective.

**3. Controller (Servlet Layer)**

Controller-ele gestionează interacțiunea dintre utilizatori și aplicație, primind cereri HTTP, procesând datele și oferind răspunsuri.

* **Clasele din com.example.demo.controller:**
  + ListaController.java: Controlează operațiile CRUD pentru entitatea *Lista*.
  + TrupaController.java: Controlează operațiile CRUD pentru entitatea *Trupa*.
  + FestivalController.java: Controlează operațiile CRUD pentru entitatea *Festival*.

**Exemplu: ListaController.java**

* Maparea URL-urilor pentru operațiile cu *Lista*:
  + /lista → Afișează toate listele.
  + /lista/new → Creează o listă nouă.
  + /lista/edit/{id} → Editează o listă existentă.
  + /lista/{id} → Șterge o listă.

**4. Service Layer (Business Logic)**

Gestionează logica aplicației și apelează metodele definite în DAO.

* **Interfețele din com.example.demo.service:**
  + ListaService, TrupaService, FestivalService:
    - Definirea metodelor pentru operațiile CRUD (e.g., getAllLista, saveLista).
* **Implementările din com.example.demo.service.impl:**
  + ListaServiceImpl, TrupaServiceImpl, FestivalServiceImpl:
    - Implementarea logicii de afaceri și apelarea metodelor din DAO.

**5. DAO Layer (Data Access Layer)**

Gestionează interacțiunea cu baza de date utilizând Spring Data JPA.

* **Clasele din com.example.demo.repository:**
  + ListaRepository, TrupaRepository, FestivalRepository:
    - Extind interfața JpaRepository, oferind metode predefinite precum findAll, save, deleteById.

**Fluxul datelor în aplicație:**

1. **Utilizatorul trimite o cerere HTTP (GET/POST).**
2. Cererea este procesată de un **Controller**, care:
   * Apelează metodele din Service Layer pentru logica aplicației.
   * Trimite datele către View sau le modifică în baza de date.
3. Service Layer-ul:
   * Apelează DAO-ul pentru interacțiunea cu baza de date.
   * Returnează datele către Controller.
4. Controller-ul:
   * Trimite datele către o pagină Thymeleaf, care le afișează utilizatorului.

**Funcționalități principale:**

Functionalitatea paginilor este similara pentru cele trei tabele, astfel : pentru a adauga/modifica/sterge o inregistrare noua in utilizatorul trebuie sa apese pe butonul corespunzator.

# **Implementarea functiilor**

O imagine care conține text, captură de ecran, software, afișaj

Descriere generată automat

În pachetul com.example.demo.controller se delcara atributele care vor fi mapate și utilizate în cadrul funcțiilor aplicației. Atributele reprezintă caracteristicile sau datele asociate acestor obiecte și pot include diverse informatii.

**Atributele Clasei:**

* idtrupa
* Nume
* Gen\_muzical
* Nr\_membrii

**Metode Getter și Setter:**

Pentru fiecare atribut al clasei există o metodă getter (get<atribut>()) care returnează valoarea atributului respectiv și o metodă setter (set<atribut>(valoare)) care setează valoarea atributului.

1. **Configurare mapare:** Sunt folosite adnotări JPA pentru maparea claselor și atributelor la tabelele și coloanele bazei de date.
2. **Gestionare conexiuni:** Conexiunile sunt configurate în application.properties și gestionate automat de Spring Boot.
3. **Gestionare tranzacții:** Tranzacțiile sunt gestionate prin adnotarea @Transactional oferită de Spring.

* **Adaugarea**

Pentru fiecare buton implementat, in functie de apasarea acestora se va face legatura cu functiile de adaugare din pachetul Controller.

O imagine care conține text, captură de ecran, meniu

Descriere generată automat

Fig 1 – create\_trupa.html

Fisierul adauga\_filme.jsp este utilizat pentru a permite utilizatorului să adauge informații despre filme in baza de date. Prin intermediul unui formular HTML, se preiau parametrii introdusi de utilizator, iar apoi aceștia sunt adăugați în baza de date utilizând Spring Boot.

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

Fig 2 – interfata pentru adaugarea trupelor

creare\_trupa.html

La apasarea butonului **Apasa** se va face legatura cu sevlet-ul trupaController

O imagine care conține text, captură de ecran, Font

Descriere generată automat

Fig 3 - TrupaController

Se utilizează clasa String createTrupaForm pentru a adăuga o noua trupa în baza de date folosind metoda Trupa trupa = new Trupa(); model.addAttribute("trupa", trupa);

Se utilizează return "create\_trupa"; pentru a redirecționa cererea către pagina "create\_trupa.html", astfel încât utilizatorul să poată continua să adauge noi trupe.

* **Afisarea**

O imagine care conține text, captură de ecran, software, număr

Descriere generată automat

Fig 4 – trupa.html

Dupa adaugare, ne redirectioneaza inapoi in pagina trupa.html, unde putem vedea cum trupa noua, s-a adaugat.

O imagine care conține text, captură de ecran, software

Descriere generată automat

Fig 3 - filmeController

* **Modificarea si stergerea**

Pentru a realiza modificarea sau stergerea datelor din tabela ‘trupe’ trebuie bifat apasat butonul asociat comenzii de modificare, respectiv stergere. La functia de stergere exista o particularitate, si anume ca este dezactivata completarea spatiilor, deoarece dorim doar sa stergem niste date.

O imagine care conține text, captură de ecran, Font, siglă

Descriere generată automat

* *Stergere:*

*O imagine care conține text, captură de ecran, Font, linie

Descriere generată automat*

Fig 4 - filmeController

Această metodă este responsabilă de efectuarea operației de ștergere a unui film din baza de date. După ce trupa a fost eliminat cu succes, cererea HTTP este redirecționată către o altă pagină denumită "trupa.html". Acest lucru poate indica utilizatorului că operația de ștergere s-a desfășurat cu succes sau că pagina a fost reîncărcată în urma ștergerii.

* *Modificare:*

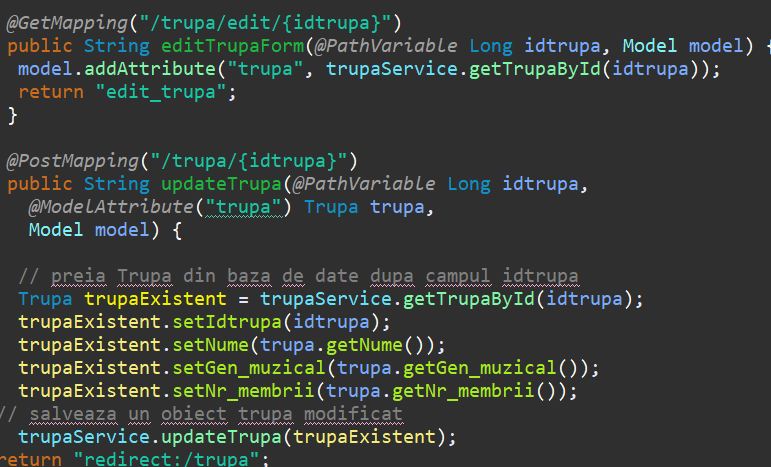


Fig 5- TrupaController

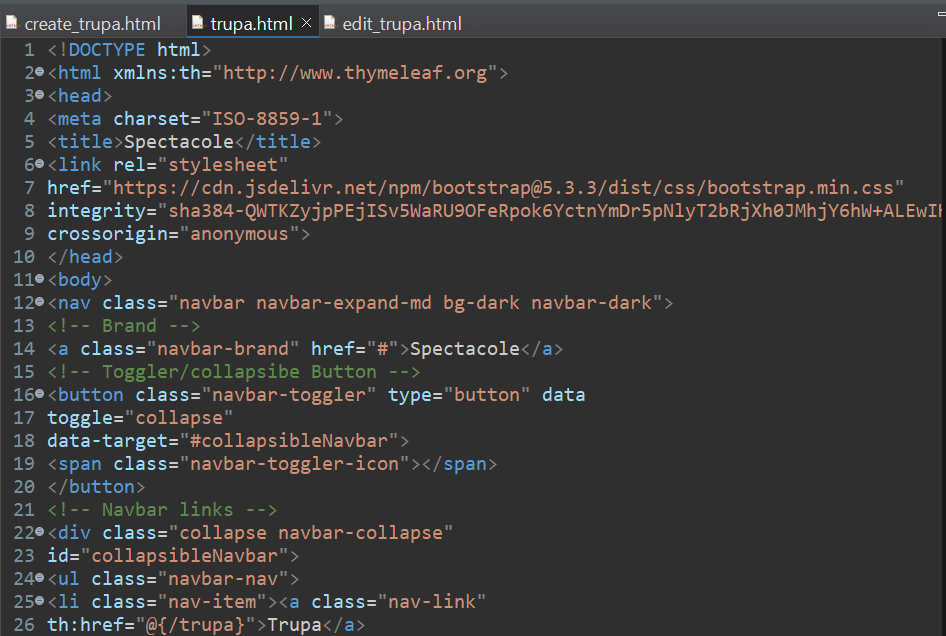
Dacă parametrul este prezent, se extrag alte parametri din cerere, cum ar fi "idtrupa", "Nume", "Gen\_muzical", "Nr\_membrii”. Acești parametri sunt extrasi din cerere utilizând metoda getParameter() și sunt convertiți la tipurile de date corespunzătoare (de exemplu, Long și Integer).

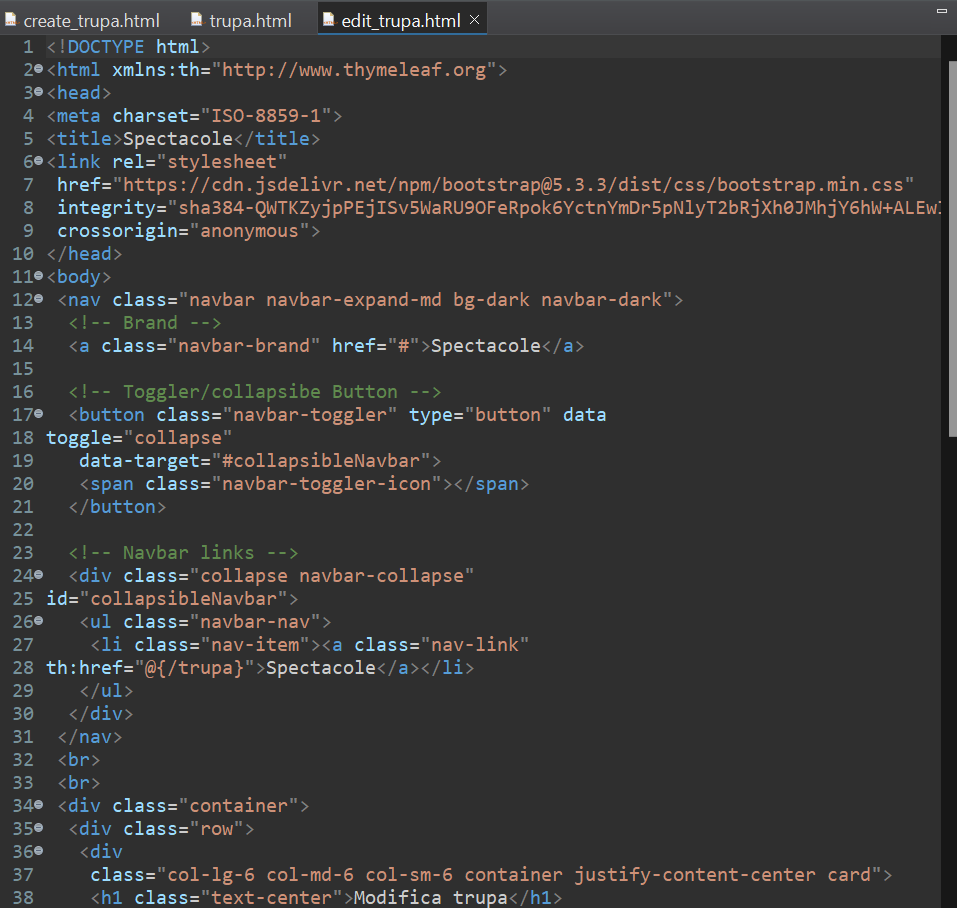
Se apelează metoda edit\_trupa a obiectului , furnizând parametrii extrasi anterior. Aceasta are rolul de a actualiza informațiile despre o trupa în baza de date cu noile valori.

După efectuarea modificărilor, cererea HTTP este redirecționată către pagina "trupe.html". Acest lucru indică utilizatorului că operația de modificare a fost realizată cu succes sau că pagina a fost reîncărcată pentru a reflecta actualizările.

# **Front-end**

Pentru interfata am ales un aspect simplu dar ingrijit, usor de inteles si utilizat folosind limbajul HTML combinat cu CSS.





O imagine care conține text, captură de ecran, software, calculator

Descriere generată automat

O imagine care conține text, captură de ecran, software, număr

Descriere generată automat

O imagine care conține text, captură de ecran, număr, Font

Descriere generată automat

O imagine care conține text, captură de ecran, software, Pictogramă computer

Descriere generată automat

O imagine care conține text, captură de ecran, software, Pictogramă computer

Descriere generată automat

O imagine care conține text, captură de ecran, software, Sistem de operare

Descriere generată automat

# **Concluzii**

Cursul a condus la dezvoltarea unei interfețe prietenoase pentru administrarea eficientă a datelor într-o bază de date MySQL, accesibilă inclusiv persoanelor fără cunoștințe avansate în SQL sau limbaje de programare. Implementarea tehnologiei Spring Boot a adus numeroase beneficii, cum ar fi persistența datelor după închiderea aplicației, facilitând interacțiunea între obiectele Java și tabelele MySQL.

Utilizarea Spring Boot a simplificat operațiunile flexibile asupra datelor, precum modificarea stării unui dosar în funcție de termene sau actualizarea informațiilor despre o trupa. Această abordare a adus avantaje notabile, inclusiv performanță superioară comparativ cu conectarea directă la baza de date, stabilitate și calitate dovedite, precum și scalabilitate pentru aplicații de diverse dimensiuni, inclusiv cele cu sute de mii de utilizatori.

Interfața dezvoltată a facilitat gestionarea eficientă a afacerilor pentru o firmă de avocatură, inclusiv adăugarea de clienți, avocați și dosare noi. Un aspect crucial este că orice angajat poate utiliza interfața fără a fi nevoie de cunoștințe tehnice specializate. Implementarea tehnologiilor Spring Boot și MySQL a permis accesul la aplicație de la distanță, prin intermediul internetului, furnizând astfel flexibilitate și disponibilitate globală.

# **Bibliografie**

1. Cursurile PIBD
2. "JavaTpoint - Learn Tutorials Point with Examples," - https://www.javatpoint.com
3. "FreeCodeCamp Forum," [Online]. - https://forum.freecodecamp.org
4. "Stack Overflow - Where Developers Learn, Share, & Build Careers," -https://stackoverflow.com
5. “W3school” - https://www.w3schools.com/html/default.asp