# UNIVERSITATEA TEHNICĂ “GH ASACHI” IAŞI FACULTATEA AUTOMATICĂ ŞI CALCULATOARE

**SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI**

**DISCIPLINA BAZE DE DATE**

**Gestiunea unui festival de muzica -XyZFestivals**

# Coordonator, Mironeanu Catalin

**Student, Baciu H. Alexandru**

**Iaşi, 2019**

**Titlu proiect : Gestiunea unui festival de muzica**

Analiza, proiectarea si implementarea unei baze de date si a aplicatiei aferente care sa modeleze activitatea de gestionare a unui festival de muzica de la modul in care poti avea acces ca si client, pana la modul de a gestiona artistii si organizarea spectacolelor.

# Descrierea cerintelor si modul de organizare al proiectului

Organizarea unui festival de muzica, desi pare ceva simplist, are in spate o organizare complexa. Volumul mare de artisti, volumul mare de spectatori, dar si spectacolele la care acestia iau parte sunt o adevarata provocare din punct de vedere al gestionarii. Activitatea de gestiune a unui festival implica o munca intensa in ceea ce priveste inregistrarea spectatorilor care pot cumpara bilete doar pe o anumita perioada(perioada care trebuie sa fie valida, adica sa fie o data in care festivalul are loc), dar si in ceea ce priveste organizarea artistilor care pot sa presteze anumite spectacole la anumite ore, in anumite locatii. Esti vital ca dupa anuntarea unui festival sa se stie locatia acestuia, artistii care sosesc, durata acestuia, dar si pretul biletelor intr-un timp destul de scurt.

**Informatiile necesare sunt:**

1. **Clientii** care sunt spectatorii ce au nevoie de acces la festival. Acestia trebuie sa ofere date personale pentru a fi usor de identificat in cazul unei anulari a spectacolului(telefon, nume, prenume, adresa e-mail) sau in cazul incidentelor neplacute ce pot avea loc(cnp).
2. **Artistii**, fara de care festivalurile nu ar exista. Acestia sunt cunoscuti printr-un nume de scena si atrag prin vastele genuri muzicale grupuri de oameni ca sa le asculte piesele live, in fata scenei.
3. **Biletele** care reprezinta accesul fizic la un anumit festival. Acestea trebuie sa contina data de acces si data de expirare astfel incat fiecare spectator sa se poate orientain ce zi poate sa participe la concertele artistilor preferati. Fiecare bilet este caracterizat printr-un anumit **tip**, astfel se determina pretul pe care un spectator il plateste pentru a fi sau a nu fi mai aproape de scena.
4. **Spectacolele**, care trebuie sa fie organizate foarte bine din punct de vedere temporal. Artistii trebuie sa se sincronizeze astfel incat sa nu existe la un moment dat 2 show-uri pe scena. Din acest punct de vedere pentru ca timpul sa fie fluidizat la maxim trebuie cunoscute pentru un spectacol ora de incepere, ora de sfarsit si pauzele pe care le ia artistul in timpul prestatiei.
5. **Detaliile festivalului** care sunt informatii despre orasul in care au loc, locatia in care se tine acesta, dar si numarul de artisti anuntat. Pe parcursul unui an un festival poate fi organizat doar o singura data in acelasi oras si este de preferat ca locatia acestuia sa fie una stabila pentru ca spectatorii sa se orienteze mai usor in cazul in care vin din alte orase.

# Descrierea functionala a aplicatiei

**Principalele funcţii care se pot intalni in gestionarea unui festival de muzica sunt:**

* + Evidenta artistilor
  + Evidenta operatiunilor referitoare la organizarea spectacolelor
  + Evidenta spectatorilor
  + Evidenta biletelor achizitionate

**Descrierea detaliata a entitatilor si a relatiilor dintre tabele:**

**Tabelele** din aceasta aplicatie sunt:

* + Festivaluri ;
  + Spectatori ;
  + Artisti ;
  + Bilete ;
  + Tip\_bilet ;
  + Spectacole;

**!!!Observatii** asupra logicii din spatele tabelelor:

* Pentru simplitatea implementarii s-a evitat crearea unei noi tabele „**Orase**”. Acest lucru ar fi trebuit realizat deoarece un festival poate fi tinut doar intr-un singur oras. (Solutie: se creeaza o tabela Orase in care avem ca primary key id-ul orasului pentru identificarea unica a acestuia).
* Relatiile many-to-many din acest proiect se solutioneaza prin existenta naturala a unor tabele(tabelele bilete si spectacole intervin in logica unui festival si rezolva in mod direct dependintele m:m intalnite).

In proiectarea acestei baze de date s-au identificat tipurile de **relatii 1:m, si m:m**.

Intre tabelele **Tip\_bilet** si **Bilete** se intalneste o relatie de tip **1:m** deoarece un tip de bilet poate fi corespunde mai multor bilete: are **Data\_acces** si **Data\_expirare**. Reciproca insa nu este valabila, deoarece un bilet nu se poate asigna la mai multe tipuri. Legatura dintre cele doua tabele este realizata prin campul **Nume\_bilet**.

Intre tabelele **Artisti** si **Festivaluri** se stablieste o legatura de tip **many-to-many(m:m)** deoarece la un festival pot veni mai multi artisti, iar un artist poate merge totodata la mai multe festivaluri. Aceasta relatie se sparge in mod natural rezultand doua relatii 1:m si legatura intre cele doua tabele se va realiza cu ajutorul unei alte tabele numita **Spectacole** care va contine cheia primara a fiecarei din cele doua tabele. Atfel spus, legatura se face doua campuri **Id\_artist** si **ID\_festival** reunite intr-o tabela comuna.

Intre tabelele **Spectatori** si **Festivaluri** se stablieste o legatura de tip **many-to-many(m:m)** deoarece la un festival pot participa mai multi spectatori, iar un spectator poate participa totodata la mai multe festivaluri. Aceasta relatie se sparge in mod natural rezultand doua relatii 1:m si legatura intre cele doua tabele se va realiza cu ajutorul unei alte tabele numita **Bilete** care va contine cheia primara a fiecarei din cele doua tabele. Atfel spus, legatura se face doua campuri **cnp** si **ID\_festival** reunite intr-o tabela comuna.

# Relatiile one-to-many generate in urma spargerii naturale ale relatiilor many-to-many sunt discutate in urmatoarele paragrafe.

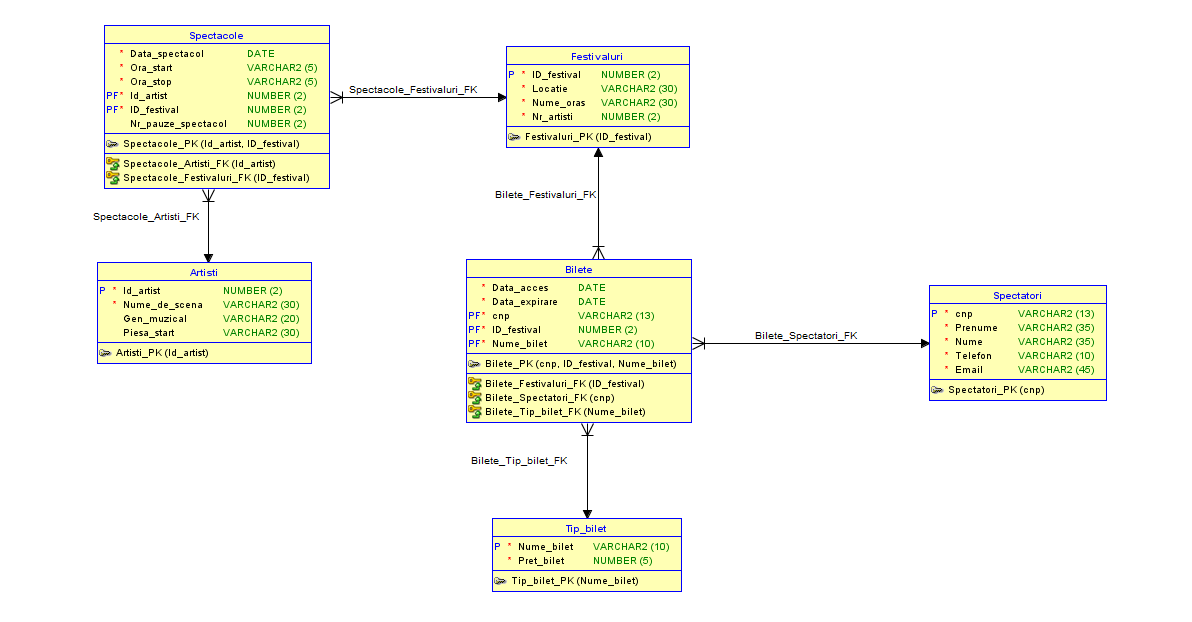
Prima relatie **1:m** apare intre tabelele **Artisti** si **Spectacole** deoarece un artist poate avea mai multe spectacole la festivaluri diferite, spectacole caracterizate prin data, ora, pauze. Reciproca insa nu este valabila, deoarece un spectacol poate fi intretinut la un moment dat de un singur artist/ o singura trupa. Legatura dintre cele doua tabele este realizata prin campul **Id\_artist**.

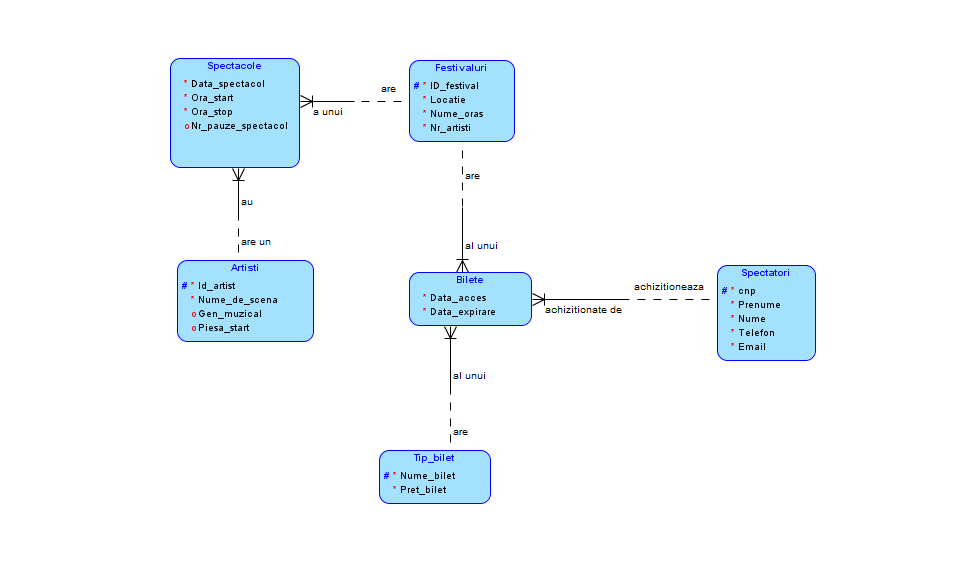
A doua relatie **1:m** apare intre tabelele **Festivaluri** si **Spectacole** deoarece la un festival pot fi desfasurate mai multe spectacole, spectacole caracterizate prin campurile deja amintite. Reciproca insa nu este valabila, deoarece un spectacol ofera o nuanta unica fiecarui festival, el neputand fi replicat. Legatura dintre cele doua tabele este realizata prin campul **ID\_festival**.

A treia relatie **1:m** apare intre tabelele **Spectatori** si **Bilete** deoarece un spectator poate achizitiona bilete pentru a participa la mai multe festivaluri, bilete identificate prin valabilitate pentru fiecare festival(Data\_acces si Data\_expirare). Reciproca insa nu este valabila, deoarece un bilet corespunde unei singure persoane(nu pot intra mai multi spectatori pe acelasi bilet, acestia trebuie sa achizitioneze fiecare cate un bilet). Legatura dintre cele doua tabele este realizata prin campul **cnp**.

Ultima relatie **1:m** apare intre tabelele **Festivaluri** si **Bilete** deoarece un festival ofera publicului diferite bilete oricarui spectator astfel incat acestia sa fie multumiti, bilete identificate prin detaliile amintite mai sus. Reciproca insa nu este valabila, deoarece un bilet iti ofera accesul la un singur festival pentru o anumita perioada(in functie de cate zile a ales spectatorul sa plateasca). Legatura dintre cele doua tabele este realizata prin campul **ID\_festival**.

# Diagrama Entitate Relatie si Diagrama modelului logic



**Constrangerile utilizate**

**Constrangeri pentru tabela Artisti**

* Id\_artist primary key: **pentru identificare in mod unic**
* Nume\_de\_scena check length and not null: **verificare nume real si obligare la inserare**
* Gen\_muzical check length: **verificare nume real**

**Constrangeri pentru tabela Spectacole**

* Data\_spectacol check with sysdate: **necesar ca festivalul sa aiba o data valabila**
* Ora\_start check if it has format hh:mm and not null: **verificare ora corecta**
* Ora\_stop check if it has format hh:mm and not null: **verificare ora corecta**
* Id\_artist and ID\_festival primary foreign key: **pentru identificare in mod unic si existenta foreign pentru relatii**

**Constrangeri pentru tabela Festivaluri**

* Id\_festival primary key: **pentru identificare in mod unic**
* Locatie check length and not null: **verificare nume real si obligare la inserare**
* Nume\_oras check length and not null: **verificare nume real si obligare la inserare**

**Constrangeri pentru tabela Bilete**

* Data\_acces check with sysdate: **necesar ca biletul sa aiba o data valabila**
* Data\_expirare check with Data\_acces: **necesar ca biletul sa aiba o data valabila(>Data\_acces)**
* Cnp, ID\_festival, nume\_bliet primary foreign key **pentru identificare in mod unic si existenta foreign pentru relatii**

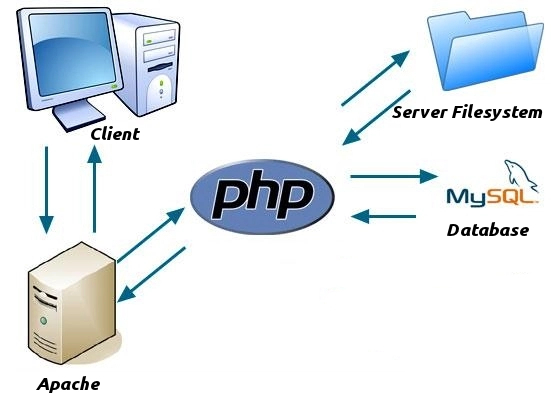
**Constrangeri pentru tabela Tip\_bilet**

* Nume\_bilet primary key: **pentru identificare in mod unic**
* Pret\_bilet not null and check if it is bigger than 0 **verificare valaore reala si obligare la inserare**

**Constrangeri pentru tabela Spectatori**

* Cnp primary key and check cnp: **pentru identificare in mod unic si verificare cnp real**
* Prenume check length and not null: **verificare nume real si obligare la inserare**
* Nume check length and not null: **verificare nume real si obligare la inserare**
* Telefon check number and not null: **verificare telefon real si obligare la inserare**
* Email check e-mail and not null: **verificare e-mail real si obligare la inserare**

**Tehnologiile folosite**



Tehnologiile folosite pentru partea de front-end sunt: **html** si **css**, iar pentru partea de back-end este folosit **php**.

Aplicatia a fost realizata cu urmatoarele versiuni:

* apache 2.4
* php 7.4
* mysql 8.0

**Conectarea la Baza de date**

Extensia **MySQLi** este un driver de bază de date relațional utilizat în limbajul de script PHP pentru a furniza o interfață cu bazele de date MySQL. Există trei opțiuni principale de API atunci când aveți în vedere conectarea la un server de baze de date MySQL:

* Extensia MySQL PHP
* Extensia MySQLi PHP
* Obiecte de date PHP

Aceasa aplicatie utilizeaza MySQLi ca API pentru lucrul cu baza de date. Pentru conectare este folosita functia „mysqli\_connect” care trebuie sa primeasca urmatorii parametri:

* $DBServer = 'server name sau IP'; de ex: 'localhost'
* $DBUser   = 'DB\_USER';
* $DBPass   = 'DB\_PASSWORD';
* $DBName   = 'DB\_NAME';

Pentru simplitate s-a ales ca paradigma de programare procedurala. Astfel, functia de conectare cu verificarea conexiunii este de forma:

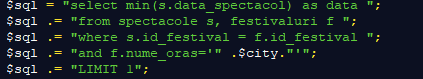
$conn = mysqli\_connect($DBServer, $DBUser, $DBPass, $DBName);

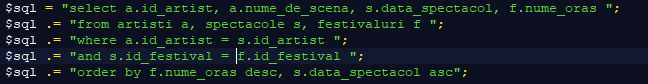
if (mysqli\_connect\_errno()) {

  exit ('Probleme de conectare: '  . mysqli\_connect\_error(), E\_USER\_ERROR);}

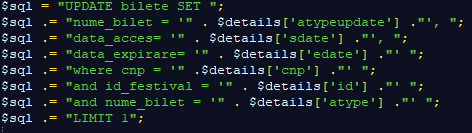
**Instructiunile SQL utilizate sunt:**

* + - **Select**

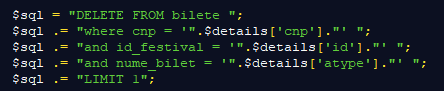
****

****

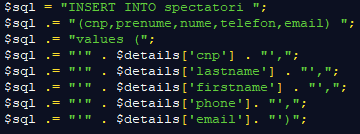
* + - **Update**

****

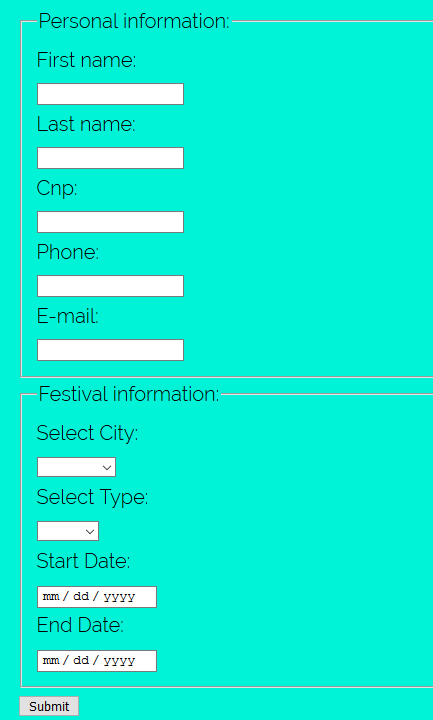
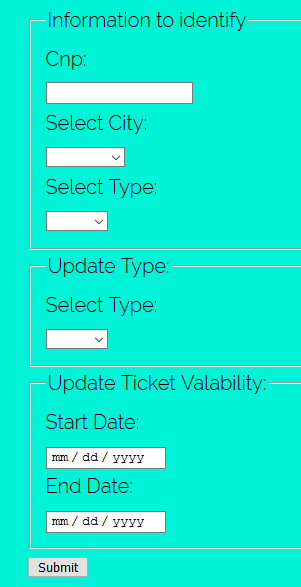
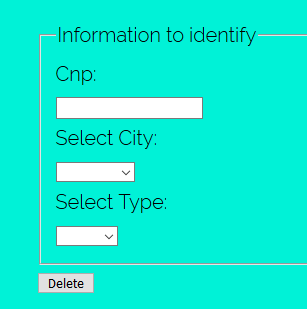
* + - **Delete**

****

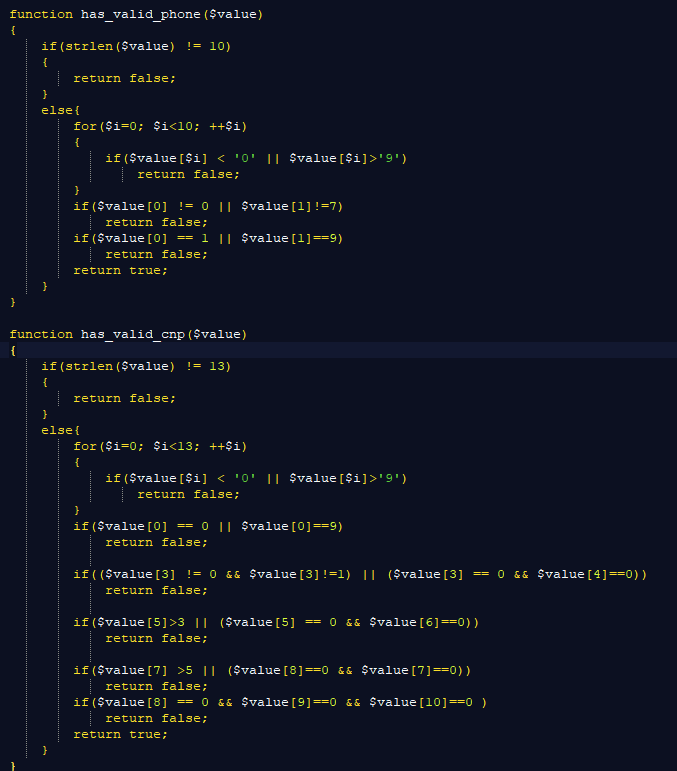
* + - **Insert**

****

Instructiunile sunt utilizate cu scopul de a permite utilizatorului sa achizitioneze, sa modifice sau sa anuleze un bilet la festival.

Imagini din interfata -^



Exemplu de functii de validare din interfata -^