Interviul tehnic este compus din 2 cerinte separate (**backend** si **frontend**), dar care pot fi, optional, cuprinse intr-un singur proiect.

Ambele cerinte contin un set de recomandari, care ofera oportunitatea de a arata nivelul de cunostinte pe diferite concepte.

# Backend

## PHP / MySQL / Laravel / Models / Relationships

### **Cerinta 1**. Crearea unei baze de date cu urmatoarele tabele

**movies**

|  |  |  |
| --- | --- | --- |
| **Nume** | **Tip** | **Observatii** |
| id | integer | autoincrement, primary key |
| status | tinyint | Valori posibile: 1 = vizibil (**default**), 2 = ascuns |
| name | tinytext |  |
| rating | decimal (10,2) | De la 1 la 10 |
| description | text |  |
| image | string | Va retine URL-ul imaginii |
| delete\_at | datetime | pentru softdelete |
| created\_at | datetime |  |
| updated\_at | datetime |  |

**artists**

|  |  |  |
| --- | --- | --- |
| **Nume** | **Tip** | **Observatii** |
| id | integer | autoincrement, primary key |
| name | tinytext |  |
| title | tinytext | Ex: star, writer, director |
| delete\_at | datetime | pentru softdelete |
| created\_at | datetime |  |
| updated\_at | datetime |  |

**movie\_artist** (tabel pivot pentru legatura many to many intre movies si artists)

|  |  |  |
| --- | --- | --- |
| **Nume** | **Tip** | **Observatii** |
| movie\_id | integer |  |
| artist\_id | integer |  |

### **Cerinta 2.** Definirea modelelor

Definirea celor doua modele, **Movie** si **Artist**, precum si a relationship-ului dintre ele, de tip **many to many**, in ambele sensuri: **Movie::artists()** si **Artist::movies()**

## REST API

Un API cu urmatoarele 3 rute:

|  |  |  |
| --- | --- | --- |
| **Ruta** | **Verb (action)** | **Observatii** |
| **movies** | **GET** | Returneaza DOAR filmele cu **status vizibil** si cu **rating > 5** Returneaza in format JSON |
| **movies/{id}** | **DELETE** | Sterge filmul cu ID-ul indicat |
| **movies** | **POST** | Adauga un film nou cu parametri trimisi in post fields: name, rating, description, image. Daca adaugarea s-a facut cu succes, va returna un JSON cu elementul nou adaugat, cu toate coloanele lui |

## Recomandari backend (optionale)

implementarea sa fie facuta in framework-ul **Laravel** si sa se foloseasca (optional) conceptele de mai jos

* migrations / seeding pentru crearea si popularea bazei de date
* Eloquent Models pentru definirea celor doua modele
* Scopes - pentru filtrarea dupa **rating**, respectiv **status** a filmelor, dar si pentru tipurile de artisti: **star**, **writer**, **director**
* Controllers
* Modele definite cu “soft delete”

# Frontend

## CSS / HTML / JS / jQuery / Bootstrap (basic)

Stilizarea paginii din **Main.png**

1. design responsive, conform design
2. la click pe iconita de meniu, se va deschide meniul din **Menu-Opened.png**
3. la click pe iconita de back, se va inchide meniul

Assets

1. font-family: Open Sans
2. font-size:
3. 14px titluri filme si butoane,
4. 12px descriere
5. 25px continut meniu
6. imagini, icons, logo (vezi arhiva)

## Vue.js (optional)

Listarea filmelor sa fie o componenta Vue care preia datele prin API si le afiseaza conform design.

## Recomandari frontend (optionale)

* Proiectul frontend sa fie implementat pe acelasi **Laravel** de la **proiectul backend**
* Instalarea dependintelor cu npm (sau similar)
* Utilizarea **scss** sau **less** pentru scrierea css-ului
* jQuery - pentru interactiunea cu meniul (sau chiar componenta Vue.js separata)
* Bootstrap (card, button, grid) si apoi customizare conform design
* Fontawesome pentru iconite, in loc de svg
* Googlefonts
* Lazyloading pentru imagini
* Filmele listate sa fie cele din baza de date