

COLEGIUL NAȚIONAL DE INFORMATICĂ "GR. MOISIL" BRAȘOV

**LUCRARE PENTRU ATESTAREA COMPETENȚELOR
PROFESIONALE**

Groovy

PROFESOR COORDONATOR

Georgescu Oana

ELEV

Dragomir Alexandra

Brașov

2023

CUPRINS

1	Argumentul lucrării.....	2
2	Descrierea aplicației din lucrare.....	3
3	Detalii tehnice de implementare.....	4
3.1	Aplicații utilizator folosite.....	4
3.2	Secvențe de program utilizate.....	4
4	Cerințe hard și soft.....	5
5	Posibilități de dezvoltare.....	6
6	Concluzii.....	7
7	Bibliografie și webgrafie.....	8

1. Argumentul lucrării

În ziua de azi tot mai mulți tineri tind să folosească dispozitive moderne, cum ar fi calculatoare, telefoane, tablete, etc, ca urmare a dezvoltării componentelor electronice și a limbajelor de programare. Domeniul muzical nu este o excepție. De la discurile de vinil la muzica digitală a fost doar un pas, iar în ziua de azi doar marii iubitori de muzică mai folosesc dispozitive analogice.

Tot ce ține de muzică, de sunet, este prezent în spațiul digital, în cloud, de la muzica clasică la ultimele aberații auditive ce se presupun a fi compoziții proprii. Muzica și-a pierdut conținutul, a pierdut profunzimea, iar mare parte din acest declin este cauzat de simplitatea, comercialitatea muzicii actuale ce pune la dispoziție în mod direct idei fără a lăsa loc de interpretare, de visare, de înțelegere a mesajului propriu-zis.

În aceste condiții, aplicația ulterior prezentată își propune să aducă în lumea digitală, măcar într-o mică măsură, o mai mare înțelegere a muzicii, reprezentând în mod direct înțelesul, explicații, mici “traduceri” a ceea ce reprezintă muzica unei ere definitorii pentru creativitate, liberă exprimare în gândire, acțiune și emoție, scoțând la lumină cultura muzicală a anilor 60’, 70’ și 80’.

2. Descrierea aplicației

Site-ul este structurat în două secțiuni:

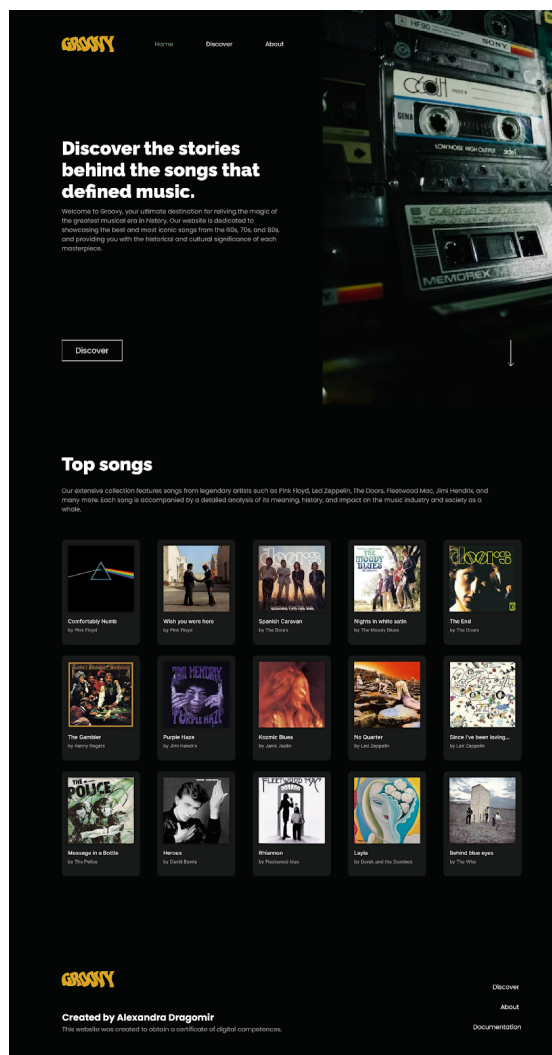
- Publică: 4 pagini, respectiv: “home”, “discover”, “about” și o pagină dedicată unei singure melodii spre prezentarea acesteia (conținutul fiind specific melodiei aleasă de utilizator).
- Privată: (doar pentru prezentare este de asemenea publică): administrarea melodiilor (inclusiv colectarea de date din Spotify), artiștilor, albumelor și a genurilor muzicale.

Secțiunea Publica

Pagina principală (landing page/home) este structurată în așa fel încât utilizatorul poate să înțeleagă scopul aplicației și să navigheze (în principal prin nav-bar), pentru a ajunge la scopul final, respectiv detalii despre o anumită melodie din colecția de melodii (inițial din cele pe care eu le consider a fi extrem de importante și cu un impact major)

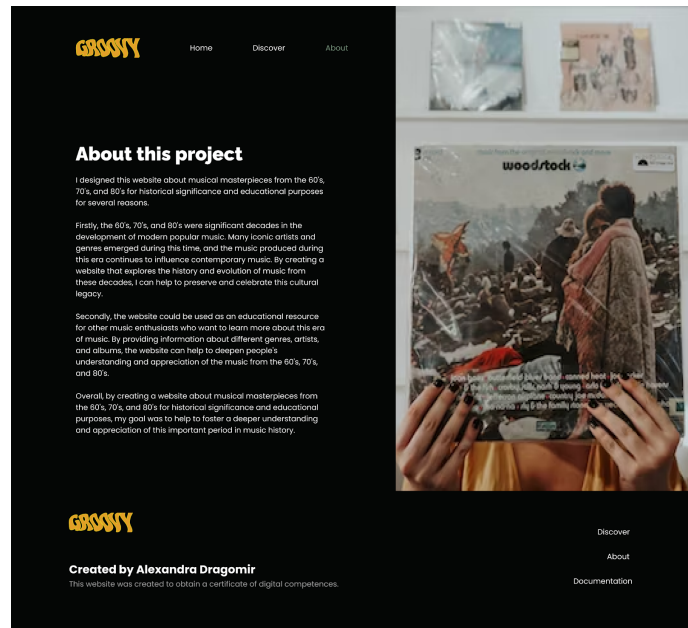
Pagina este împărțită în două secțiuni: “*Over the fold*” (OTF), respectiv ceea ce se vede pe primul ecran, la deschiderea site-ului, și “*under the fold*” (UTF), respectiv conținutul ce poate fi văzut în momentul în care se dă scroll sau se folosește butonul “*Discover*” (ce va trimite utilizatorul direct în partea de selecție)

În UTF se va regăsi colecția formată din cele mai importante 15 melodii, prezentă în aplicație cu rolul de a da click pe oricare din ele pentru a obține detaliile din pagina personală a melodiei.

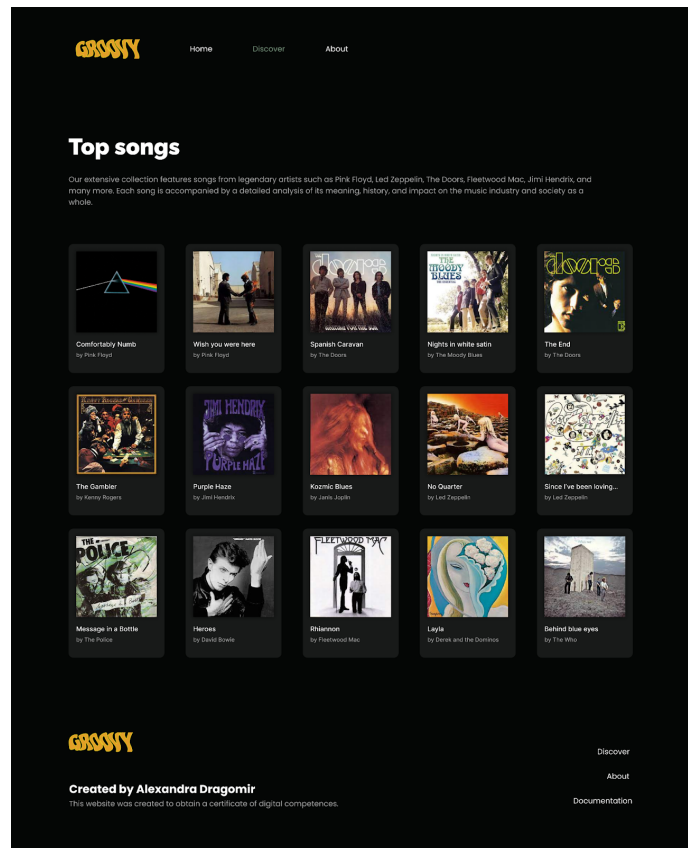


În subsolul aplicației, comun pe toate paginile, se vor regăsi detalii generale, respectiv, de copyright, și link-uri utile.

Pagina *About*: În această pagină sunt prezentate detalii despre scopul proiectului, din punct de vedere educațional și istoric, relative la muzica în sine, alături de o poză reprezentativă perioadei vizate (*festivalul “păcii” din 1969 - Woodstock ’69*).

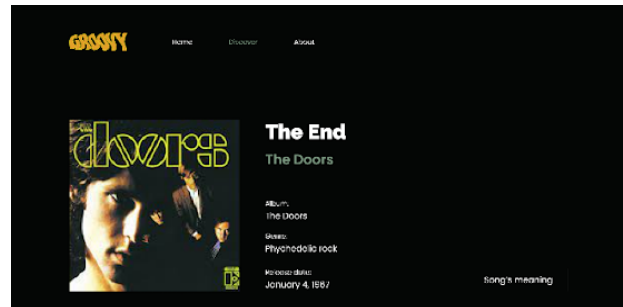


Pagina *Discover*: Această pagină este o replică minimalistă a zonei *Discovery* de pe prima pagină, pe care ulterior vor fi toate melodiile din baza de date locală.

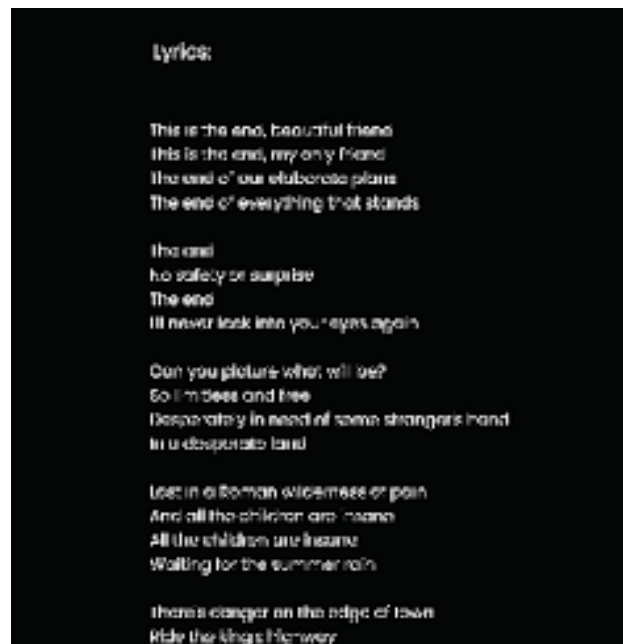


Pagina unei melodii are 3 mari secțiuni:

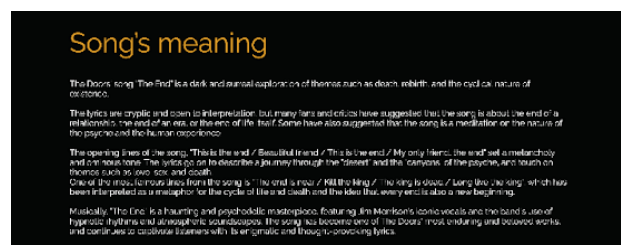
1. Prezentarea generală a melodiei, ce conține o poză a albumului de proveniență, numele albumului de proveniență, numele artistului, numele melodiei, genul muzical și data relativă a lansării albumului.



2. Prezentarea versurilor melodiei, în limba originală.



3. O interpretare (explicație), public acceptată, a înțelesului și a istoriei din spatele fiecărei melodii, cu mici păreri personale



Sectiunea Pricata (administrare)

În această secțiune se poate administra conținutul dinamic al aplicației, fiecare element al meniului ducând către o listă de componente, de unde se pot întreprinde mai multe acțiuni, respectiv, adaugare, editare, ștergere.

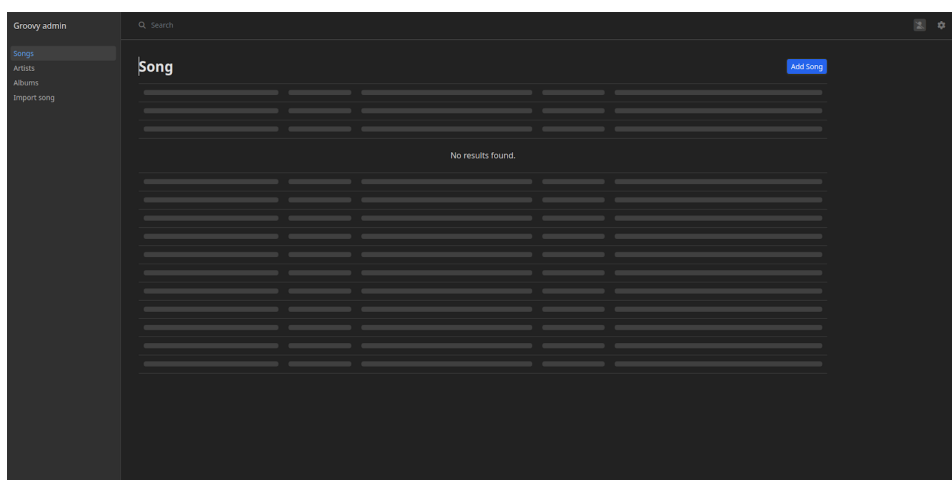
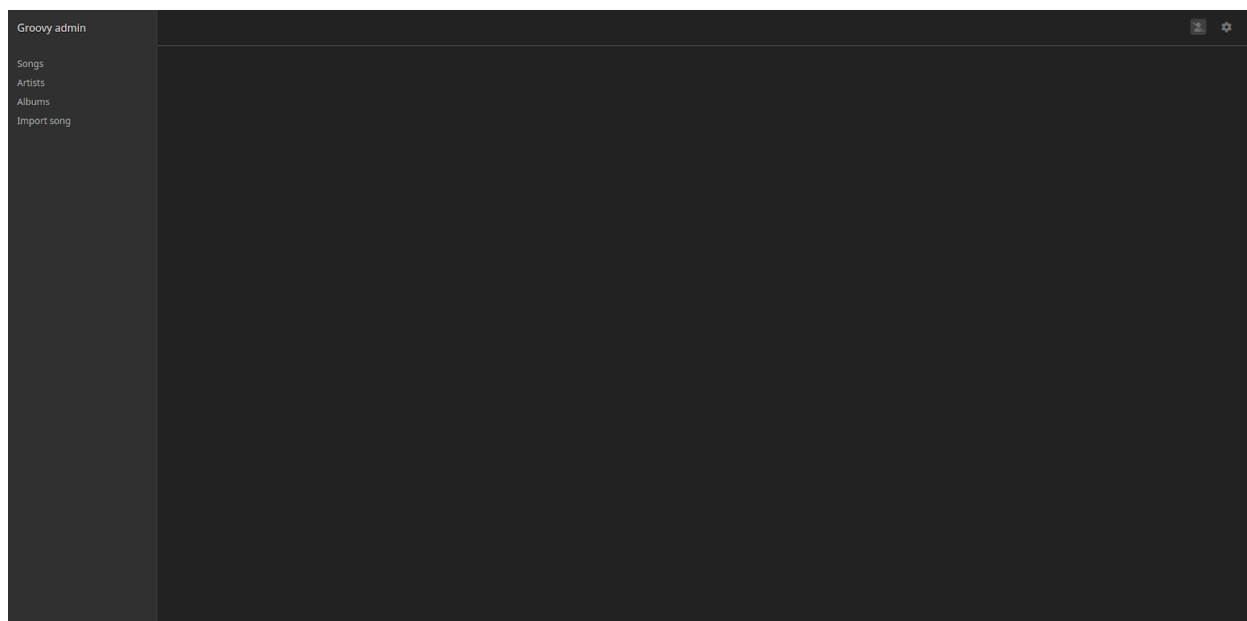


Fig. 1 Lista generala

Create Song

Create and add another **Create**

Name*

Artists

Album*

Lyrics

Lyrics meaning

Fig. 2 Ecran de adaugare/editare element

Songs
Artists
Albums
Import song

Name*
Song name

Artist*
Artist name

Submit

Select from search*

Fig. 3 Cautare detalii. Ultimul element permite importarea detaliilor unei melodii din Spotify.

Name*
The end

Artist*
The doors

Submit

Select from search*

- The End by The Doors from album The Doors released on 1967-01-04
- The End by The Doors from album The Doors released on 1967-01-04
- Genie As Strange by The Doors from album Strange Days released on 1987-09-25
- The End by The Doors from album The Future Starts Here: The Essential Doors Hits released on 2008-01-28**
- The End - New Stereo Mix by The Doors from album The Very Best of The Doors released on 2007-09-25
- Break on Through the Blue Door by The Doors from album The Doors released on 1967-01-04

Fig. 4 Selector a 5 opțiuni preluate din Spotify.

3. Detalii tehnice de implementare

Aplicația are la bază framework-ul Symfony (PHP) care, împreună cu Twig (precompilator template PHP + HTML) pun la dispoziție conținutul dinamic HTML al site-ului.

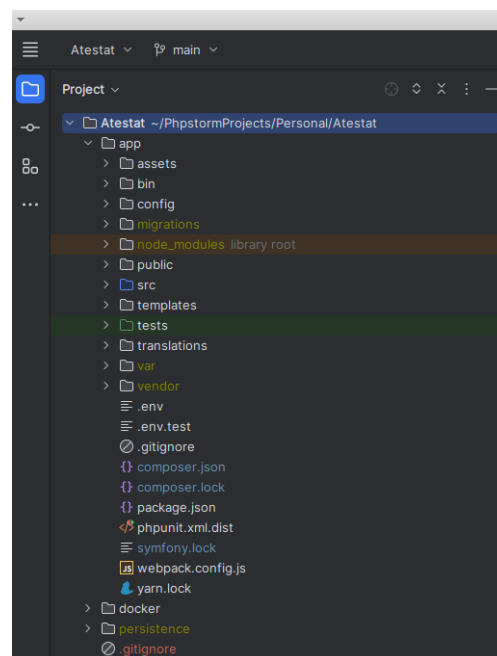
Stilizarea paginilor s-a realizat cu ajutorul pachetului Bootstrap (aliniere și adaptarea dinamică a paginii, din punct de vedere vizual, pe baza dispozitivului de afișaj), Webpack (pentru administrarea resurselor media) iar în rest, în mod direct HTML și CSS(3).

Generarea conținutului dinamic s-a realizat prin intermediul PHP (Symfony), prin procesarea cererilor și modelarea răspunsurilor în funcție de necesitățile utilizatorului, preluarea datelor s-a realizat prin intermediul Spotify API direct din binecunoscuta aplicație.

Pentru implementare am utilizat editorul PHPStorm oferit de compania JetBrains ce oferă o licență studenților pentru cei cu înclinații în programare și dezvoltare software.

3.1 Aplicații utilizator folosite

- **Symfony framework (6.2):**
 - este un framework de back-end bogat în funcții, care este folosit pentru a construi aplicații complexe. Acest framework oferă o gamă de instrumente și caracteristici concepute special pentru construirea de aplicații web PHP scalabile, inclusiv suport pentru mai multe medii și un sistem robust de stocare *cache* (stocare temporară a datelor).



- Stă la baza aplicației, generând conținutul dinamic al paginilor, gestionează intern și extern utilizarea link-urilor (rute).

```
1 Noramath
#[Route(path: '/', name: 'app_index_home')]
public function home(SongService $service): Response
{
    return $this->render('home.html.twig', [
        'active' => 'home',
        'songs' => $service->getSongsForHomePage()
    ]);
}
```

• Twig

- Twig este un motor de șabloane pentru aplicații web construite folosind framework-ul Symfony. Acesta oferă o sintaxă simplă și intuitivă pentru a defini șabloane HTML și CSS. Twig este bazat pe PHP și este ușor de învățat și de folosit pentru dezvoltatorii web de toate nivelele. Prin utilizarea Twig, am putut separa logica aplicației de prezentare, ceea ce face mai ușoară întreținerea și dezvoltarea ulterioară a aplicației.
- Fiecare șablon poate include alte șabloane (prin extindere sau incluziune directă - *embedding*) astfel încât să se permită refolosirea codului existent.

```
▼ templates
  > security
    about.html.twig
    base.html.twig
    discover.html.twig
    home.html.twig
    onside_top_base.html.twig
    search_song.html.twig
    spotify_song_list.html.twig
    twosides_top.base.html.twig
```

```
{% extends 'base.html.twig' %}
{% block page_content %}
    <body>
        <header>
```

- **Webpack (Encore)**

- Webpack este un *module bundler* (creator/manager de colecții de resurse cod) pentru aplicații web. Acesta preia diferite tipuri de fișiere (cum ar fi fișiere JavaScript, fișiere CSS și imagini) și le transformă în module pe care le poate încărca în browser. Webpack este proiectat să fie utilizat în aplicații web complexe și încurajează modularitatea, ceea ce înseamnă că poți împărți codul tău în module mici și ușor de gestionat. Acesta oferă, de asemenea, o serie de funcționalități utile, cum ar fi încărcarea leneșă a modulelor, generarea automată de fișiere sursă și integrarea cu alte unelte populare de dezvoltare web. Folosirea Webpack poate îmbunătăți semnificativ performanța și eficiența dezvoltării aplicațiilor web.

```
webpack.config.js x
1 const Encore = require('@symfony/webpack-encore');
2
3 if (!Encore.isRuntimeEnvironmentConfigured()) {
4     Encore.configureRuntimeEnvironment({ environment: process.env.NODE_ENV || 'dev' });
5 }
6
7 Encore
8     .setOutputPath('public/build/')
9     .setPublicPath('/build')
10    .copyFiles({
11        from: './assets/images',
12        to: 'images/[path][name].[hash:8].[ext]',
13        pattern: /\.?(png|jpg|jpeg|avif|webp|svg)$/
14    })
15    .addEntry('app', './assets/app.js')
16    .addEntry('main', './assets/main.js')
17    .addEntry('two-sided', './assets/two-sided.js')
18    .addEntry('home', './assets/home.js')
19    .splitEntryChunks()
20    .enableSingleRuntimeChunk()
21    .cleanupOutputBeforeBuild()
22    .enableBuildNotifications()
23    .enableSourceMaps(!Encore.isProduction())
24    .enableVersioning(Encore.isProduction())
25    .configureBabelPresetEnv((callback) => {
26        callback.config.useBuiltIns = 'usage';
27        callback.config.corejs = '3.23';
28    })
29    .enableSassLoader()
30    ;
31
32 module.exports = Encore.getWebpackConfig();
```

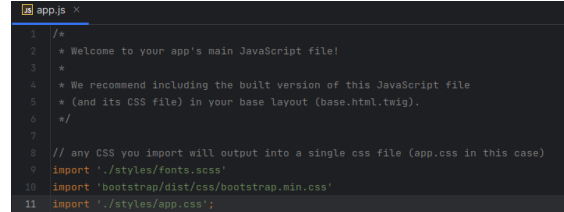
- Cu ajutorul webpack se creaza colecții (CSS, Javascript, Font-uri, Imagini - *assets*) ce pot fi ulterior folosite optim în fiecare pagina separat.

```
<head>
  <meta charset="UTF-8">
  <title>{% block title %}Welcome!{% endblock %}</title>
  <link rel="icon" href="{{ asset('build/images/fav_icon.svg') }}">
  {{ encore_entry_link_tags('app') }}
  {{ encore_entry_link_tags('main') }}
  {% block stylesheets %}
  {% endblock %}
</head>
```

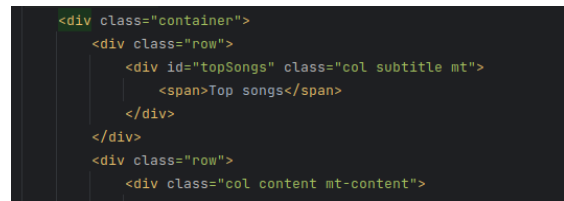
- **Bootstrap**

- Bootstrap este un cadru front-end popular pentru construirea de site-uri web și aplicații web receptive și mobile. A fost creat inițial de Twitter și acum este întreținut de o comunitate de dezvoltatori.

Bootstrap oferă un set de clase CSS, pluginuri JavaScript și șabloane HTML care facilitează proiectarea și dezvoltarea paginilor web cu stil și aspect consistent. Cadrul include un sistem de grilă care permite dezvoltatorilor să creeze cu ușurință machete receptive care se adaptează la diferite dimensiuni de ecran și dispozitive.



```
1 /*
2  * Welcome to your app's main JavaScript file!
3  *
4  * We recommend including the built version of this JavaScript file
5  * (and its CSS file) in your base layout (base.html.twig).
6  */
7
8 // any CSS you import will output into a single css file (app.css in this case)
9 import './styles/fonts.scss'
10 import 'bootstrap/dist/css/bootstrap.min.css'
11 import './styles/app.css'
```



```
<div class="container">
  <div class="row">
    <div id="topSongs" class="col subtitle mt">
      <span>Top songs</span>
    </div>
  </div>
  <div class="row">
    <div class="col content mt-content">
```

- **SpotifyAPI client**

- Pentru conectarea cu Spotify am folosit pachetul (*composer*) “calliostro/spotify-web-api-bundle”

3.2 Secvențe de program utilizate

Symfony pune la dispoziție un model MVC (Model-View-Controller) pe care l-am folosit după cum urmează:

Fișierul principal, ce guvernează aplicația este IndexController (App\Controller\IndexController) unde sunt definite toate rutele (destinații ce pot fi văzute de către utilizatori), se preia informația de la browser.

```
#[Route(path: '/', name: 'app_index_home')]
public function home(SongService $service): Response
{
    return $this->render('home.html.twig', [
        'active' => 'home',
        'songs' => $service->getSongsForHomePage()
    ]);
}
```

Toate acestea vor fi trimise către un “serviciu” ce aduce datele din baza de date (prin intermediul unor conectori numiți *repositories* (*Model* în MVC), le procesează și va returna înapoi către controller datele ne-formatate.

```
public function getLastFifteenSongs() {
    return $this->createQueryBuilder('s')
        ->setMaxResults(15)
        ->orderBy('s.id', 'DESC')
        ->getQuery()
        ->getResult();
}
```

De aici, Controllerul va formata informația și o va pregăti pentru trimiterea în formatul final către browser print intermediul TWIG ceea ce reprezintă partea *View*.

```
<div class="container-fluid">
    {% for song in songs %}
        {% embed 'songPanel.html.twig' %}
        {% endembed %}
    {% endfor %}
</div>
```

Folosirea serviciilor nu este întotdeauna necesară, atât timp cât nu apare nevoia de a interacționa cu un serviciu de rezistență (stocare de date).

Spre exemplu, pentru a cauta date din Spotify:

```
#[Route(path: '/search', name: 'app_index_search', methods: ['GET',
'POST'])]
public function search(Request $request): Response
{
    $items = [];
    $form = $this->getForm(null, null, $items);
    $form->handleRequest($request);
    if ($form->isSubmitted()) {
        ['name' => $name, 'artist' => $artist] = $form->getData();
        $items = $this->getItems($name, $artist);
        $form = $this->getForm($name, $artist, $items);
    }
    return $this->render('spotify_song_list.html.twig', [
        'form' => $form,
        'items' => $items
    ]);
}
```

Aici folosim datele transmise prin FORM de către browser (respectiv numele melodiei căutate și numele artistului) si prin intermediul metodei *getItems()* preluam datele din Spotify.

```
private function getItems(string $name, string $artist) {
    $key = str_replace(' ', '_', strtolower($name) . '_' .
    strtolower($artist));
    return $this->cache->get($key, function () use ($name, $artist) {
        $result = $this->spotifyWebAPI->search(
            $artist . ' ' . $name,
            ['track'],
            [
                'limit' => 5,
                'offset' => 0,
                'market' => 'RO'
            ]
        );
        return $result->tracks->items;
    });
}
```

Este important sa nu facem cereri ce nu sunt necesare către Spotify API, de aceea, înainte de a trimite cererea verificăm dacă nu cumva pentru aceeași combinație de melodie + artist verificam sa nu existe deja informatia in cache-ul local.

4. Cerințe hard și soft

Hardware

Cerinta	Minim	Optim
Procesor	-	-
Memorie (RAM)	512MB	1GB
Spatiu de stocare	500MB	2GB (depinde de dimensiunile bazei de date)
Rezolutie monitor	1024x786	1920x1080

Software

- PHP >= 8.1
- Symfony >= 6.2
- Librarii
 - Pachete composer (meta pachete)
 - @symfony
 - calliostro/spotify-web-api-bundle
 - easycorp/easyadmin-bundle
 - twig/twig
 - symfony/webpack-encore-bundle
 - Pachete npm (meta pachete)
 - @babel
 - file-loader
 - webpack
 - sass-loader
 - Bootstrap
- NodeJs

O lista completa de dependinte este prezentă și administrată de cele doua managere de pachete, respectiv composer și yarn și se regaseste in fișiere de configurare pentru fiecare dintre cele doua, respectiv composer.json si package.json

5. Posibilități de dezvoltare

Aplicație este modulară, codul este decuplat ceea ce face mentenanța și dezvoltarea viitoare a aplicației extrem de facilă. În momentul de față aplicația permite deja adăugarea unui număr nelimitat de melodii prin intermediul panoului de administrare.

O îmbunătățire semnificativă ar fi implementarea unei conexiuni API cu musixmatch, ceea ce ar permite preluarea automată de texte ale melodiilor și posibil chiar o interfatare cu Chat GPT (Open AI) pentru a prelua opiniile de ordin public despre text și înțelesuri, în acest fel reducând imens de mult munca administratorului.

6. Concluzii

Prin utilizarea unui framework precum Symfony, am câștigat experiență în dezvoltarea site-urilor web folosind o abordare structurată. Acum sunt familiarizată cu beneficiile framework-urilor, cum ar fi reutilizarea codului, scalabilitatea și menținerea acestuia.

Utilizarea twig, un motor de șabloane, împreună cu randarea de tip server a Symfony, m-a ajutat să creez pagini web dinamice cu date de la Spotify API. Acest lucru îmi oferă o înțelegere bună a modului în care funcționează randarea de tip server și cum poate fi utilizată pentru a crea pagini web dinamice.

Utilizarea webpack (encore) și Bootstrap mi-au oferit experiență în utilizarea bibliotecilor client-side pentru a îmbunătăți experiența utilizatorului. Acum sunt familiarizată cu modul în care pot utiliza aceste biblioteci pentru a îmbunătăți timpul de încărcare al paginilor, a crea design-uri responsive și a adăuga interactivitate.

Utilizarea clientului API Spotify m-a ajutat să dobândesc experiență în utilizarea API-urilor pentru a prelua date din surse externe. Acest lucru mi-a oferit o bună înțelegere a modului în care funcționează API-urile și cum le pot integra în aplicațiile web.

Crearea unui site web funcțional și estetic plăcut mi-a oferit o înțelegere a celor mai bune practici în dezvoltarea web, cum ar fi organizarea adecvată a fișierelor, optimizarea codului pentru performanță și crearea unui design responsiv pentru diferite dimensiuni de ecran.

În concluzie, cunoștințele mele de programare dobândite la școală mi-au oferit o bună fundație pentru experiența în a crea acest website utilizând framework-ul Symfony, twig, webpack, Bootstrap și clientul API Spotify. Experiența dobândită din crearea acestui website mi-a oferit cunoștințe și abilități valoroase pe care le pot utiliza pentru a îmi dezvolta în continuare abilitățile de dezvoltare web.

7. Bibliografie și webgrafie

- Documentatii:
 - <https://symfony.com/doc/current/index.html>
 - <https://twig.symfony.com/doc/>
 - <https://developer.spotify.com/documentation/web-api>
 - <https://github.com/callioastro/spotify-web-api-bundle>
 - <https://getbootstrap.com/docs/5.1/getting-started/contents/>
 - <https://developer.mozilla.org/en-US/>
 - <https://symfony.com/bundles/EasyAdminBundle/4.x/index.html>
 - <https://www.w3schools.com>
 - <https://git-scm.com/doc>
- Resurse grafice:
 - <https://fontawesome.com/icons>
 - <https://unsplash.com/>
 - <https://www.figma.com/>
 - <https://coolors.co/>