

**PROIECT**  
**Aplicație web pentru căutarea informațiilor**  
**despre curse de Formula 1**

Student: Armășoiu Timotei-Cătălin

Profesor coordonator: Bogdan Cristian-Florea

## *1. Descrierea temei*

Tema propune proiectarea și dezvoltarea unei aplicații web interactive pentru a facilita căutarea informațiilor cu privire la cursele de formula 1 din ultimii 10 ani. Aplicația are la bază utilizarea unui API în care avem deja colectate toate datele necesare despre diferite curse, sezoane, șoferi sau competiții existente în calendarul anual de formula 1. Aceste date vor fi utilizate și prelucrate cu ajutorul tehnologiei PHP pentru a crea o interfață interactivă cu utilizatorul, dar și pentru a realiza diferitele rute necesare pentru accesarea diferitor informații. În capitolele ce urmează vor fi detaliate etapele dezvoltării teoretice și practice a unei astfel de aplicații.

## *2. Rezultatele dorite*

În urma proiectării, implementării, dar și dezvoltării software se dorește un produs final care să faciliteze accesul la informații cu privire la competițiile de Formula 1. În funcție de cunoștințele dezvoltatorului rezultatul final al aplicației poate prezenta diferențe comparativ cu rezultatele propuse în începutul lucrării.

## *3.Scopul și obiectivul temei*

Tema de față își propune realizarea unei aplicații web dinamice care să afișeze diferite informații cu privire la competițiile de Formula 1, dar totodată aceasta familiarizează cititorul cu diferitele etape de proiectare și dezvoltare a unei astfel de aplicații. Obiectivul temei este bineînțeles de a realiza o aplicație funcțională, utilizabilă și cu un aspect plăcut, dar totodată aceasta își dorește familiarizarea cu limbajul de scripting PHP.

## *4.Definirea cerințelor software*

Din punct de vedere software se va folosi mediul de dezvoltare PHPStorm, oferit gratuit de către compania JetBrains. Acesta oferă un editor pentru PHP, HTML și JavaScript cu analiză de cod din mers, prevenirea erorilor și refactorizare pentru cod

PHP și HTML. Pentru realizarea proiectului se poate utiliza orice editor de text disponibil în mediul online, însă pentru dezvoltarea aplicației curente se va folosi acest mediu de dezvoltare. Pentru a instala și utiliza acest mediu de dezvoltare trebuie urmați pași descriși în continuare.

1. Se va accesa link-ul <https://www.jetbrains.com/phpstorm/>.
2. Se va apăsa butonul din următoarea imagine:

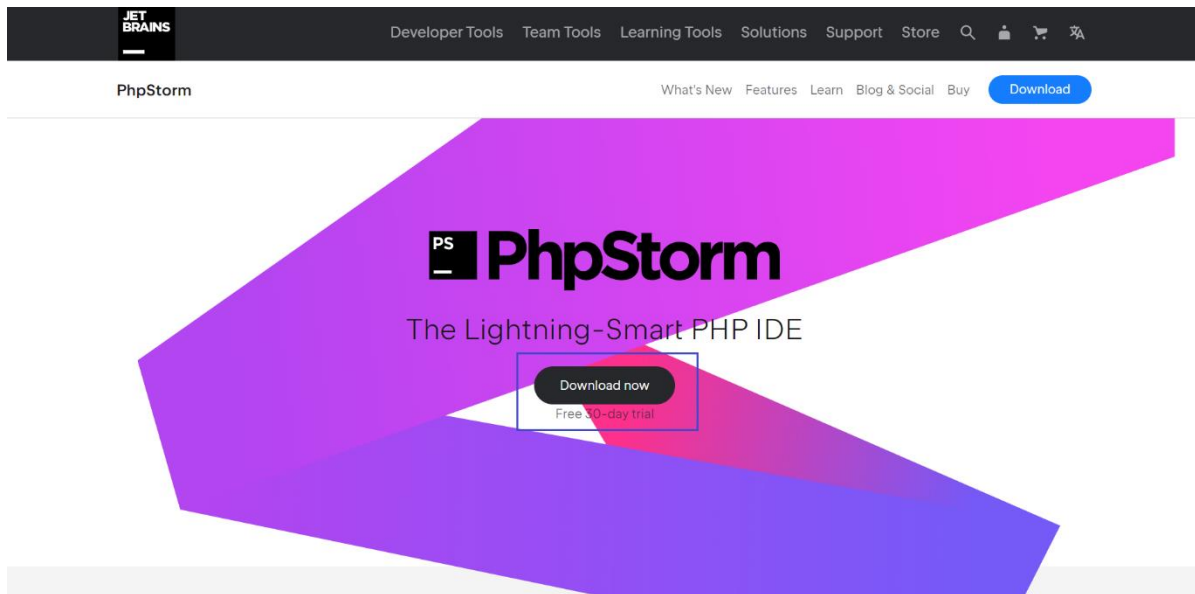


Figura 4.1

Descărcarea mediului de dezvoltare va începe automat după care se vor urma pașii de instalare de bază accesând executabilul descărcat anterior.

Totodată pentru a utiliza toate beneficiile acestui limbaj de scripting trebuie instalat, bineînțeles pachetul de PHP pentru a putea rula acest limbaj. Acesta se poate downloada de la următorul link <https://www.php.net/downloads.php>. După descărcare și instalare putem începe proiectarea și ulterior dezvoltarea aplicației.

## 4.1 De ce PHP?

Tema acestui capitol propune motivarea tehnologiei alese, deoarece există o varietate de tehnologii pentru a dezvolta o astfel de aplicație. Așa cum a fost menționat în capitolele anterioare, pentru a realiza tema propusă vom utiliza limbajul de scripting PHP datorită facilității pe care o pune la dispoziție pentru a crea o pagină web dinamică. PHP sau "hypertext preprocessor" a apărut pentru prima dată în anul 1994, perioadă în care aplicațiile de tip web aveau să ia foarte mult avânt. Acesta poate genera conținut dinamic, colecta și cripta date, poate citi scrie, șterge sau închide diferite fișiere pe server. Este totodată un limbaj versatil, rulând pe diferite platforme de operare (Windows, Linux, Unix, macOS) și compatibil cu aproape toate sistemele

server utilizate în prezent. Mai mult decât atât, PHP nu este limitat doar la a afișa cod HTML, se pot afișa imagini, fișiere cu extensia PDF, sau chiar text XHTML sau XML.

În continuare va fi prezentată o listă cu câteva avantaje ale utilizării acestui limbaj:

a. Flexibilitate și versatilitate excelentă

PHP poate fi combinat cu o varietate de limbaje de programare astfel produsul final utilizează cea mai bună tehnologie pentru fiecare funcționalitate adăugată. Acesta este un limbaj de tipul "cross-platform", adică poate rula eficient pe orice sistem de operare, iar acest lucru facilitează foarte mult procesul de dezvoltare al aplicației.

b. Foarte multă documentație disponibilă în format online

Navigând paginile de pe internet putem observa că există o varietate de videoclipuri explicative, manuale și documentații pentru a facilita procesul de învățare, dar și procesul de dezvoltare a unei aplicații web dinamice folosind tehnologia PHP.

c. Viteză de încărcare foarte mare

Utilizarea PHP poate accelera foarte mult încărcarea unei pagini web. Comparativ cu alte limbaje de programare(*exemplu: Python*) viteza de încărcare a unei pagini web poate fi și de 3 ori mai mare. Acest factor este important în condițiile în care pe piață este o nevoie constantă de servicii rapide și eficiente.

## 4.1.2 *Symfony*

Pentru realizarea proiectului au mai fost folosite și anumite unelte disponibile online ce au făcut mai facilă dezvoltarea aplicației. Una dintre aceste unelte folosite este framework-ul **Symfony**. Acest framework a fost folosit pentru arhitectura MVC foarte bine pusă la punct, având o structură organizată (*model, view, controller*). Acesta se poate descărca gratuit folosind următorul link: <https://symfony.com/download>. Mai jos se poate observa în poză chenarul verde care face referire la linia de comandă necesară pentru a descărca framework-ul symfony. Se rulează această linie de comandă în "Windows PowerShell", ulterior procesul de instalare fiind foarte facil.

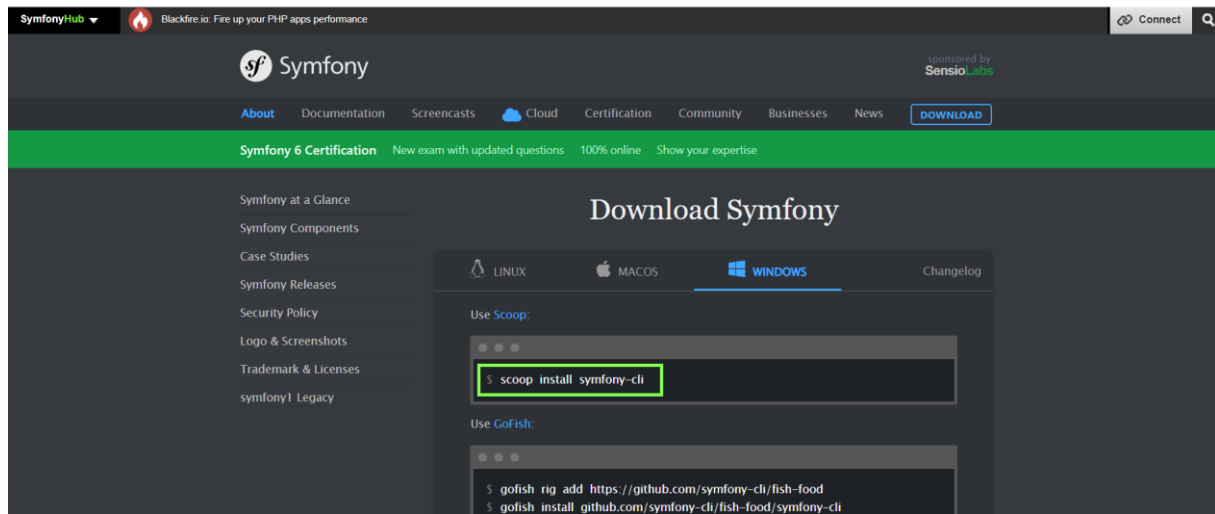


Figura 4.2

### 4.1.3 Twig

O altă unealtă folosită pentru realizarea acestui proiect este template-ul **Twig**. Acesta este un motor de șabloane pentru limbajul de programare PHP. Sintaxa sa provine de la alte două șabloane numite Jinja și Django. La fel ca și framework-ul prezentat anterior și această unealtă este open-source sub licență BSD și se poate descărca de la următorul link: <https://twig.symfony.com/>.

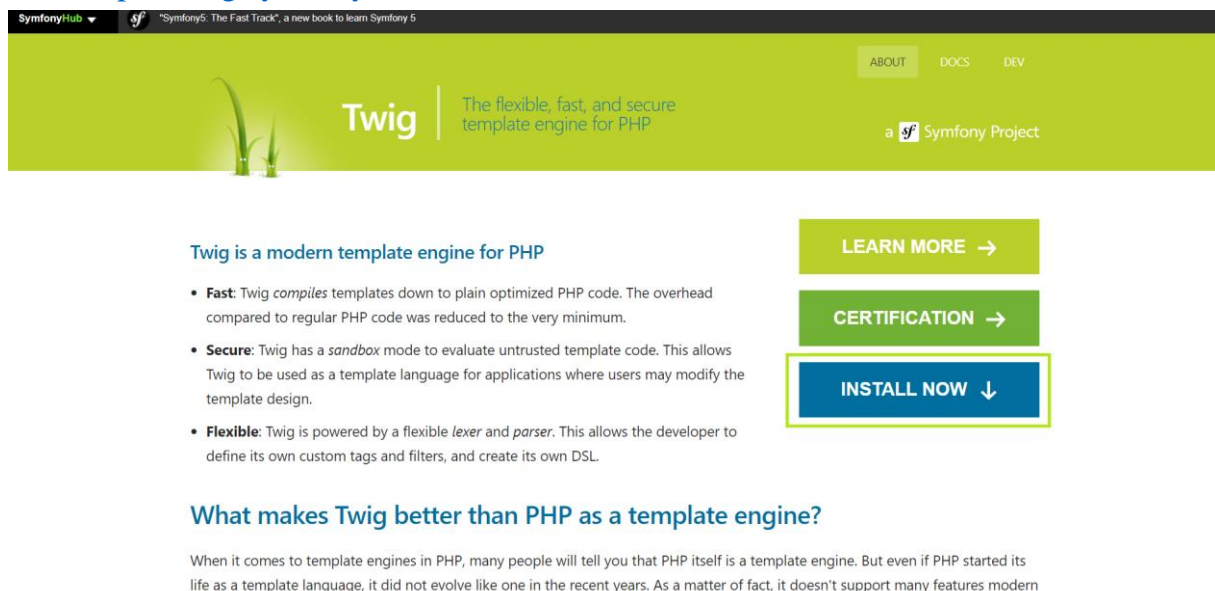


Figura 4.3

## 4.1.4 API

Termenul API provine din limba engleză și se traduce ca ”Interfață de programare a aplicațiilor”. Acesta pune la dispoziție dezvoltatorului un număr de funcții pe care el din urmă să le folosească pentru a construi diferite aplicații. Pentru a simplifica modul de lucru, API-ul oferă un răspuns al utilizatorului la un sistem și trimite acest răspuns sistemului înapoi unui utilizator. În acest proiect a fost folosit un API pus la dispoziție de către site-ul <https://rapidapi.com/>. Pentru a putea accesa API-urile folosind acest site este nevoie ca utilizatorul să își creeze un cont. Numărul de cereri care se pot realiza pe parcursul unei zile este de 100.

## 5. Elemente de analiză și proiectarea sistemului

În acest capitol va fi descrisă o diagramă funcțională a proiectului, pe baza căreia se poate implementa logica din spatele aplicației web și anume implementarea ”back-end”. Pentru a explica mai ușor va fi prezentată o imagine cu scopul de a explica logica din spatele aplicației.

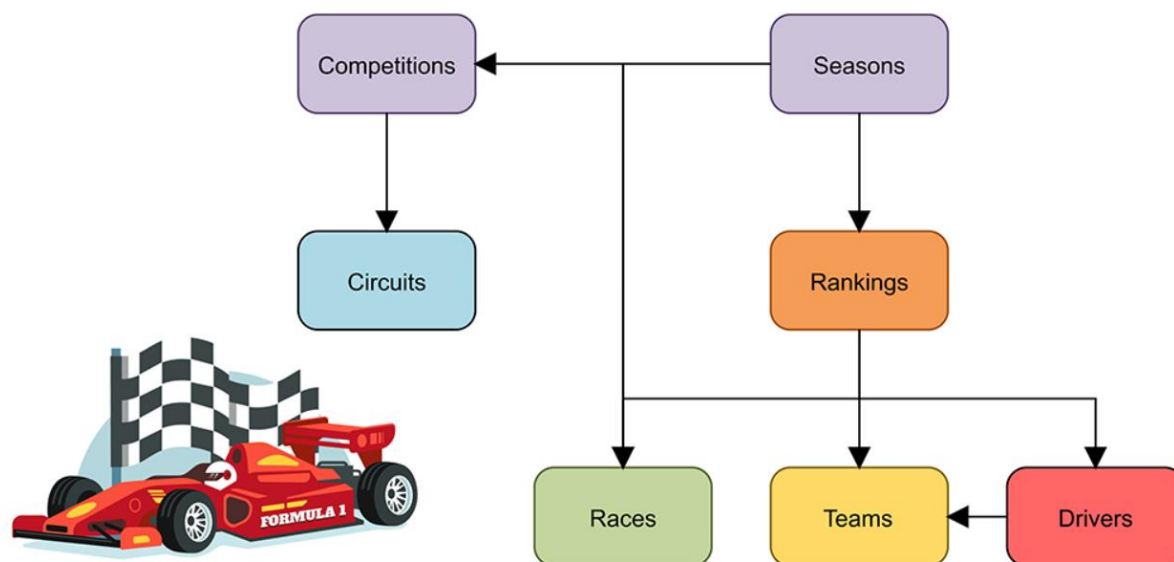


Figura 5.1

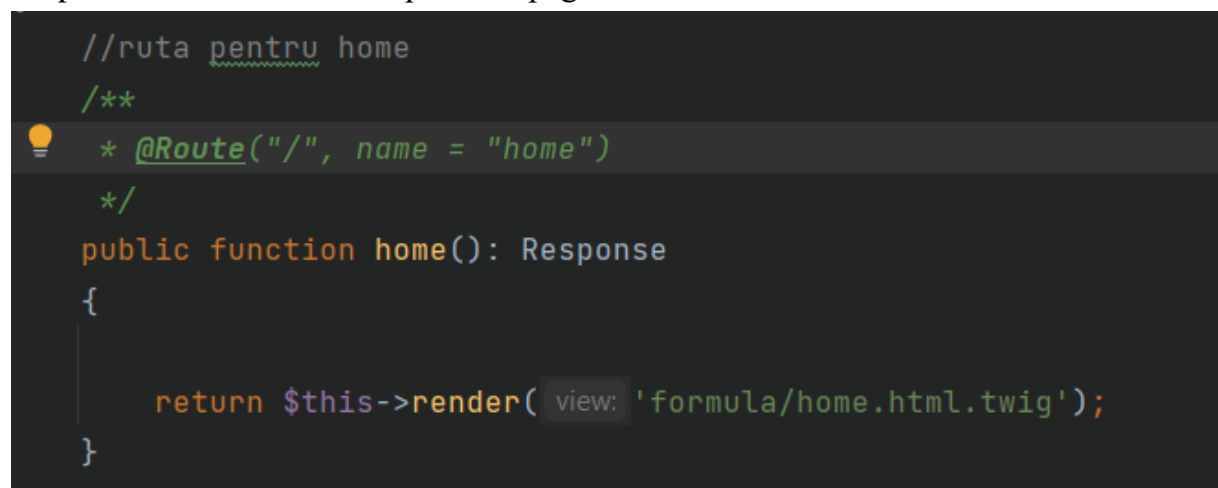
În figura 5.1 se poate vedea modul de proiectare al aplicației. În momentul în care este selectat un anumit sezon de formula 1, trebuie să existe posibilitatea de a vedea

clasamentul sezonului selectat, dar dacă se dorește totodată se pot vedea și competițiile din sezonul respectiv. De asemenea au fost implementate toate celelalte funcționalități, posibilitatea de a vedea diferitele circuite, curse, echipe sau informații despre șoferii participanți în această competiție. Toată această implementare reprezintă ”logica back-end” a aplicației și constituie scheletul acesteia, partea funcțională cu care utilizatorul nu interacționează.

## 6. Scurta descriere a aplicației

În acest capitol, pentru o mai bună înțelegere a modului de funcționare al aplicației vor fi descrise etapele cheie ce au constituit realizarea aplicației

Etapa 1: Realizare unei rute pentru o pagină de ”home”.



```
//ruta pentru home
/**
 * @Route("/", name = "home")
 */
public function home(): Response
{
    return $this->render( view: 'formula/home.html.twig');
}
```

Figura 6.1

Această funcție va returna conținutul său către fișierul ”home.html.twig”, iar la accesare URL-ului ”127.0.0.1:8000”, vom fi întâmpinați de această pagină

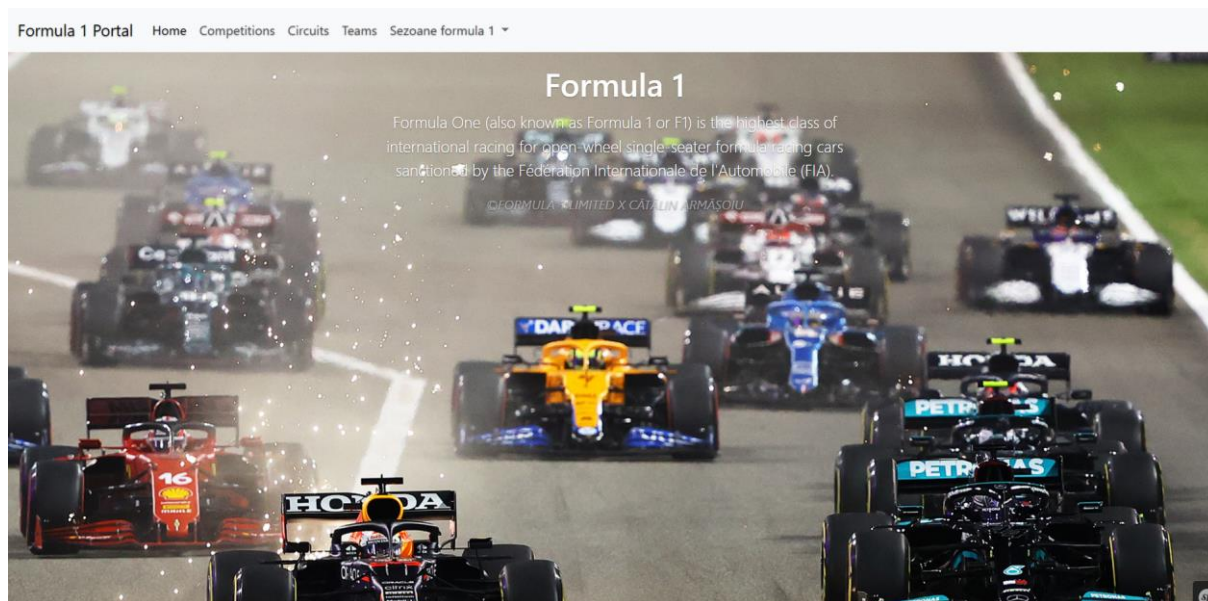


Figura 6.2

Etapă 2:

În această etapă vom studia funcționalitatea proiectului și legătura cu API-ul pentru a accesa diferite informații. Pentru acesta vom intra în fișierul de tip Controller din proiect și vom realiza o ruta.

```
/**
 * @Route("/seasons", name = "seasons")
 */
public function seasons(): Response
{
    $headers = [
        'x-rapidapi-host' => 'api-formula-1.p.rapidapi.com',
        'x-rapidapi-key' => '5a6f44fa10msh40be2be8d20bc5bp18a190jsnb4478dcbc8f1'
    ];
    $client = new GuzzleClient([
        'headers' => $headers
    ]);

    $r = $client->request( method: 'GET', uri: 'https://api-formula-1.p.rapidapi.com/seasons');
    $response = $r->getBody()->getContents();

    return $this->render( view: 'formula/season.html.twig', [
        'response' => json_decode($response),
    ]);
}
```

Figura 6.3

În figura 6.3 se observă faptul că facem un call către un API extern. Acesta poate fi accesat doar folosind cheia pe care o procurăm în momentul în care ne facem un cont pe site-ul descris la capitolul 4, pentru a putea consuma API-ul. Pentru a realiza cererile HTTP către API-ul respectiv, s-a folosit "Guzzle", care este de fapt un client care ușurează modalitatea de a face cereri (în cazul acesta de tip GET) către un API. După ce



a fost făcută cererea se va decoda(folosind funcția `json_decode`) și returna răspunsul API-ului(un fișier de tip JSON) către interfața un fișier HTML. Mai departe prelucrarea datelor va fi realizată în respectivul fișier.

Tot în această figură se observă că a fost creată ruta `"/seasons"`, care deși nu este accesată direct din URL, ci prin intermediul unui buton, aceasta oferă utilizatorului posibilitatea de a vedea o listă cu diferite sezoane de formula 1 în funcție de an.

Observație: În acest caz, datorită limitării informațiilor oferite de API, se pot vedea diferite statistici doar pe ultimii 10 ani

```
{% extends "base.html.twig" %}

{% block title %}Seasons{% endblock %}

{% block body %}

    <br><br>
    <div class="container">
        <div class="dropdown">
            <div class="col-md-12 text-center">

                <button type="button" class="btn btn-primary dropdown-toggle" data-bs-toggle="dropdown">

                    Detalii sofer in functie de sezon
                </button>
                <ul class="dropdown-menu">
                    {% set seasons = response.response %}
                    {% for season in seasons %}
                        <li><a class="dropdown-item" href="{{ path('seasons_year', {'year':season}) }}">{{ season }}</a></li>
                    {% endfor %}
                </ul>
            </div>
        </div>
    <br><br>
{% endblock %}
```

Figura 6.4

În interfață vor fi prelucrate datele din fișierul JSON. Pentru a putea naviga cu ușurință între sezoanele de Formula 1, a fost creată o buclă de tipul `for`, care parcurge fiecare index din câmpul `response` pentru a extrage informația care mai apoi va fi afișată în pagina web ca în Figura 6.5

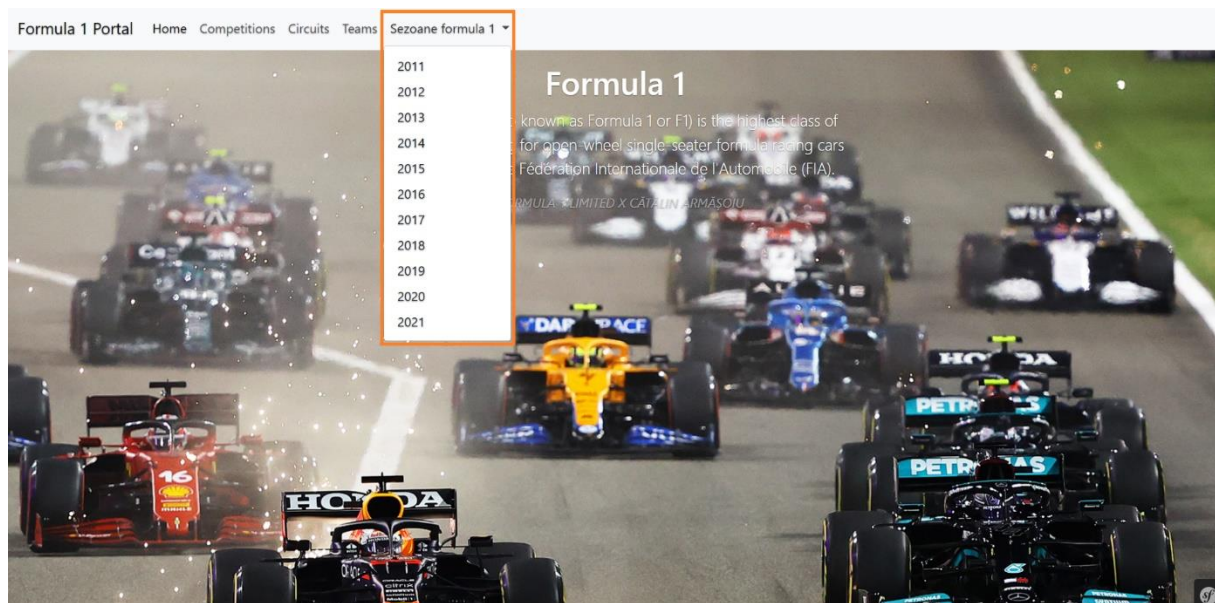


Figura 6.5

Pentru a consuma în continuare conținutul API-ului se vor reitiera cele 2 etape pentru diferite informații cu privire la campionatul de Formula 1. De asemenea pentru interfața aplicației WEB s-a utilizat un template existent din arhiva Bootstrap 5. Acest template se poate descărca de la următorul link <https://getbootstrap.com/docs/5.0/examples/>. Template-ul folosit se numește Cover și poate fi descărcat de la link-ul anterior.

## *7. Identificare a riscurilor și posibile soluții de rezolvare a riscurilor*

În realizarea proiectului au apărut probleme de natură funcțională, precum o legătură greșită între logica back-end și interfața front-end a aplicației. Pentru a rezolva aceste probleme, la momentul apariției am utilizat funcția `{{dump(response)}}`. Această funcție oferă utilizatorului o privire de ansamblu asupra datelor consumate de la API și totodată folosind metodă incrementării în pași mici se pot rezolva problemele de această natură. Odată cu extinderea funcțională a aplicației pot apărea bineînțeles multe alte probleme, însă pentru aplicația curentă problema predominantă era reprezentată de o slabă legătură între logica aplicației și interfața acesteia.

## 8. *Bibliografie*

Acest capitol include o listă cu toate materialele care au fost folosite în redactare documentației, dar și în realizarea propriu zisă a aplicației:

1. <https://light-it.net/blog/why-use-php-main-advantages-and-disadvantages/>
2. [https://www.w3schools.com/php/php\\_intro.asp](https://www.w3schools.com/php/php_intro.asp)
3. <https://www.w3schools.com/bootstrap/>
4. <https://symfony.com/>
5. <https://twig.symfony.com/>
6. <https://rapidapi.com/api-sports/api/api-formula-1/details>
7. <https://getbootstrap.com/docs/5.0/components/navbar/>
8. <https://docs.guzzlephp.org/en/stable/>
9. <https://api-sports.io/documentation/formula-1/v1>