RED LINE DOCUMENTATION

APLICACIONES WEB ORIENTADAS A SERVICIOS

TIDBIS41M

Eduardo Arellanes, Daniel Beltran

02/04/2025

INTRODUCTION.

This API mashup was developed in Django and obtains data from 3 Different API'S, one from soccer, Formula 1 and sports news, here are the links for the API'S.

Soccer: v3.football.api-sports.io

F1: https://f1connectapi.vercel.app/api/current/drivers

Sports news: https://magicloops.dev/api/loop/49bdee03-4f9e-4f3d-a146-79f6d8fbee82/run

Services.py

This File stores all the API's connections.

F1 Api:

```python
def Drivers():
    url = "https://f1connectapi.vercel.app/api/current/drivers"

    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        drivers_list = data.get('drivers',[])
        # Agregar la URL de la imagen a cada conductor

        for driver in drivers_list:
            driver['image_url'] = get_driver_image(f"{driver['name']} {driver['surname']}")
        return drivers_list
    else:
        return None
```

Line 1: define the Drivers function.

Line 2: define the variable "url" that stores our API URL.

Line 3: define the variable "response" that stores the get method and receives the url parameter.

Line 4-11: define an if statement that if the response status code is 200, the data we receive will be stored in the "data" variable and being converted to json. Then converted to a list and stored in the variable "drivers_list". After that, we iterate in drivers_list and obtain all the values. If not, we return None

Soccer API:

```
def Futbol():
    conn = http.client.HTTPSConnection("v3.football.api-sports.io")

    headers = {
        'x-rapidapi-host': "v3.football.api-sports.io",
        'x-rapidapi-key': "ef11f29081cb7e476852aa0a039679df"
        }

    conn.request("GET", "/fixtures?league=39&season=2023", headers=headers)

    res = conn.getresponse()
    data = res.read()
    i = 0
    data = json.loads(data)
    fixtures  = []
    for fixture in data['response']:
        fixtures.append({
            'local': fixture['teams']['home']['name'],
            'visit': fixture['teams']['away']['name'],
            'localImg': fixture['teams']['home']['logo'],
            'visitImg': fixture['teams']['away']['logo'],
            'date': fixture['fixture']['date'],
            'goleslocal': fixture['goals']['home'],
            'golesvisitante': fixture['goals']['away']
        })
        i += 1
        if i == 6:
            break
```

Line 1: define the Futbol function.

Line 2: establish the connection with the host.

Line 3: give the headers

Line 7: make a request to the host, including the method POST, the endpoint that will return the info from the API and the headers.

Line 8: obtain the response and store it in “res” variable.

Line 9: read the response and store it in “data” variable.

Line 10: convert the json to a dictionary.

Line 11: initialize variables

Line 12: iterate through the matches inside the response.

Line 13: append everything specified in the json to the fixtures list.

Line 21: add 1 to the iterative variable and when reaches 6 it will break because there was so much data.

Line 22: return the fixtures list.

News API:

```python
def Noticias():
    url = 'https://magicloops.dev/api/loop/49bdee03-4f9e-4f3d-a146-79f6d8fbee82/run'
    payload = {}

    response = requests.get(url, json=payload)
    responseJson = response.json()
    news= []
    i = 0
    for new in responseJson['noticias']:
        news.append({
            'titulo': new['titulo'],
            'nota': new['Nota'],
            'img': new['Foto']
        })
        i += 1
        if i == 8:
            break

    return news
```

Line 1: Define the Noticias function.

Line 2: define the “url” variable that stores our API URL.

Line 3: define the “payload” variable that stores an empty dictionary.

Line 4: make a request to the url sending the empty payload.

Line 5: converts the response to json.

Line 6: initialize iterative variable and the news list.

Line 7: iterate through every new in the news array inside the json response.

Line 8: append a dictionary into the news list.

Line 13: add 1 to the iterative variable.

Line 14: an if statement to only show 8 news, and break it

Line 15: return the news.

Views.py

```python
from django.shortcuts import render
import requests
from django.conf import settings
from .services import *

def index(request):
    drivers = Drivers()
    partidos = Futbol()
    news = Noticias()

    context = {
        'drivers': drivers,
        'partidos': partidos,
        'news': news
    }

    return render(request, "index.html", context)
```

Line 1-4: import everything we need

Line 5: define the index function that receives requests.

Line 6-8: define variables for each function.

Line 9: create a context dictionary that stores the functions.

Line 10: return the render with a request, the template and the context variable that will call all the functions.

```
drivers_results = []
if drivers:
    for driver in drivers:
        full_name = f"{driver['name']} {driver['surname']}".lower()
        if query in full_name or query in driver.get('nationality', '').lower():
            drivers_results.append({
                'type': 'driver',
                'name': f"{driver['name']} {driver['surname']}",
                'nationality': driver.get('nationality', ''),
                'image_url': driver.get('image_url', ''),
                'team': driver.get('teamId', '')
            })
```

Line 1: declare a list called drivers_results.

Line 2: if the drivers list is not empty, you will go in.

Line 3: Iterate through drivers.

Line 4: convert to lower case the result format of name and surname.

Line 5: checks if query is completed and if there is coincidence, will append the result to drivers_results.

```
football_results = []
if partidos:
    for partido in partidos:
        if query in partido['local'].lower() or query in partido['visit'].lower():
            football_results.append({
                'type': 'match',
                'local': partido['local'],
                'visit': partido['visit'],
                'localImg': partido['localImg'],
                'visitImg': partido['visitImg'],
                'date': partido['date'],
                'score': f"{partido['goleslocal']} : {partido['golesvisitante']}"
            })
```

Line 1: declare a list called football_results.

Line 2: if the partidos list is not empty, you will go in.

Line 3: Iterate through partidos.

Line 4: convert to lower case the result format of name and surname.

Line 5: checks if query is completed and if there is coincidence, will append the result to football_results like a dictionary.

```python
# Search in news
news_results = []
if news:
    for new in news:
        if query in new['titulo'].lower() or query in new['nota'].lower():
            news_results.append({
                'type': 'news',
                'title': new['titulo'],
                'content': new['nota'],
                'image': new['img']
            })
```

Line 1: declare a list called news_results

Line 2: if the news list is not empty, you will go in.

Line 3: Iterate through news.

Line 4: convert to lower case the result format of tutulo and nota.

Line 5: checks if query is completed and if there is coincidence, will append the result to news_results like a dictionary.

```python
return JsonResponse({
    'drivers': drivers_results,
    'matches': football_results,
    'news': news_results
})
```

This is the full response of the function and for each case, and we will use this response in the frontend to show information or error message.

Urls.py

```python
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index,name='index'),
    path('admin/', admin.site.urls)
]
```

Line 1-3: Do all the imports

Line 4-7: have the list of all urls, our case is the home url we import the views from the index function and give the name of index.

```html
<!-- Search model Begin -->
<div class="search-model">
    <div class="h-100 d-flex align-items-center justify-content-center">
        <div class="search-close-switch"><i class="fa fa-close"></i></div>
        <form class="search-model-form" onsubmit="return false;">
            <input type="text" id="search-input" placeholder="Search here.....">
        </form>
    </div>
</div>
```

This is the modal where we call the search function with the imput.

```
$('#search-input').on('input keypress', function(e) {
    if (e.key === 'Enter') {
        e.preventDefault();
        const query = $(this).val().trim();
        if (query.length >= 2) {
            performSearch(query);
            // Close search modal
            $('.search-model').removeClass('active');
            $('.search-close-switch').trigger('click');
        }
    }
});
```

This method call the search function when we press enter key if the length of search value is longer than 2.

```
$('#search-input').on('input', function() {
    clearTimeout(searchTimeout);
    const query = $(this).val().trim();

    if (query.length < 2) {
        $('#search-results-section').hide();
        return;
    }

    searchTimeout = setTimeout(() => {
        performSearch(query);
    }, 300);
});
```

Call the search function when we press the search input buttom or the time without type is longer tan 3 seconds

```
function performSearch(query) {
    $.get('/search/', { q: query }, function(data) {
        const hasResults = data.drivers.length > 0 || data.matches.length > 0 || data.news.length > 0;

        if (hasResults) {
            $('#search-results-section').show();

            // Scroll to search results
            $('html, body').animate({
                scrollTop: $('#search-results-section').offset().top - 100
            }, 500);
```

Line 1: we declare the function with its parameter query.

Line 2: use method get to make a request in a specific endpoint with the query param as the user input

Line 4: define has results as a Boolean in different cases to know if we have any response from the endpoint

Line 5: if we have a response from the endpoint we show the section

Line 6: we make an animation to scroll to the search result section

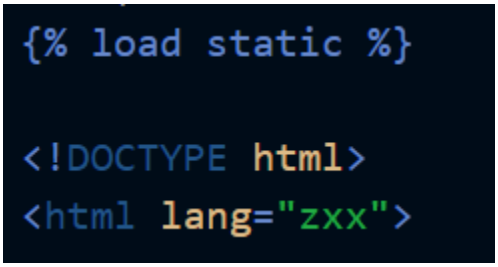
```
const driversHtml = data.drivers.map(driver => `
    <div class="col-md-4 mb-3">
        <div class="card">
            <div class="card-img-top" style="height: 200px; position: relative;">
                <img src="${driver.image_url}" class="w-100 h-100" alt="${driver.name}" style="object-fit: cont
                <div class="no-image-container" style="width: 100%; height: 100%; background-color: #f5f5f5; di
                    <div>
                        <i class="fa fa-user-circle" style="font-size: 48px; color: #e53637; margin-bottom: 10p
                        <p style="color: #666666; margin: 0;">Imagen no disponible para</p>
                        <p style="color: #e53637; font-weight: bold; margin: 5px 0;">${driver.name}</p>
                    </div>
                </div>
            </div>
            <div class="card-body">
                <h5 class="card-title">${driver.name}</h5>
                <p class="card-text">${driver.nationality}</p>
                <p class="card-text"><small class="text-muted">Team: ${driver.team}</small></p>
            </div>
        </div>
    </div>
`).join('');
$('#drivers-results').html(driversHtml || '<p>No drivers found</p>');A
```

We create the section for each case of result, in case we have results we create a card and add to the section.

For each result, if we don’t have results we show these messages, and that will work for drivers, matches and news.

Call everything in templates (index.html)

Call styles



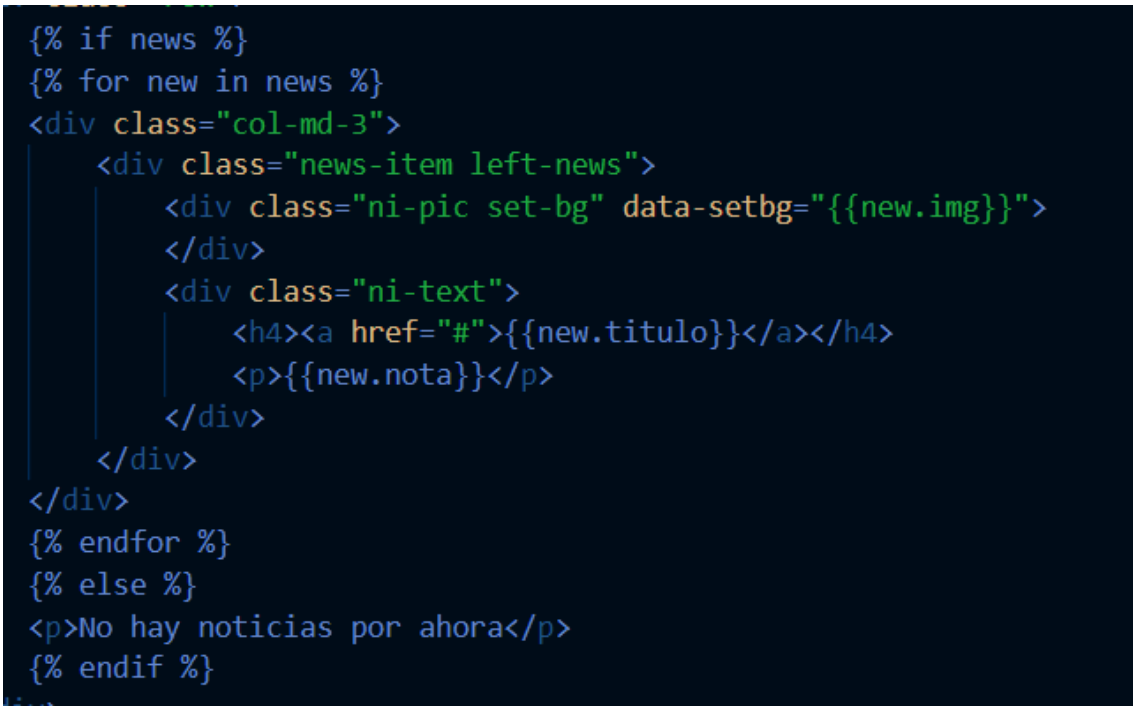
Line 1: First thing is calling the statics to all the html.

Call dynamic information

This is the cycle we use to print the information we want to show about the match, such as the teams, scores, dates, teams logos, etc.

```
{% if drivers %}
{% for driver in drivers %}
<div class="item">
    <div class="soccer-item">
        <div class="si-tag">F1 Driver</div>
        <div class="si-pic">
            {% if driver.image_url %}
                <img
                    src="{{ driver.image_url }}"
                    alt="{{ driver.name }} {{ driver.surname }}"
                    style="width: 100%; height: 300px; object-fit: contain; background-color: #f5f5f5;"
                    onerror="this.style.display='none'; this.nextElementSibling.style.display='flex';"
                >
                <div class="no-image-container" style="width: 100%; height: 300px; background-color: #f5f5f5; display: none; align-items: center; justify-content: center; text-align: center; padding: 20px;">
                    <div>
                        <i class="fa fa-user-circle" style="font-size: 64px; color: #e53637; margin-bottom: 15px; display: block;"></i>
                        <p style="color: #666666; margin: 0;">Imagen no disponible para</p>
                        <p style="color: #e53637; font-weight: bold; margin: 5px 0;">{{ driver.name }} {{ driver.surname }}</p>
                    </div>
                </div>
            {% else %}
                <div class="no-image-container" style="width: 100%; height: 300px; background-color: #f5f5f5; display: flex; align-items: center; justify-content: center; text-align: center; padding: 20px;">
                    <div>
                        <i class="fa fa-user-circle" style="font-size: 64px; color: #e53637; margin-bottom: 15px; display: block;"></i>
                        <p style="color: #666666; margin: 0;">Imagen no disponible para</p>
                        <p style="color: #e53637; font-weight: bold; margin: 5px 0;">{{ driver.name }} {{ driver.surname }}</p>
                    </div>
                </div>
            {% endif %}
        </div>
        <div class="si-text">
            <h5><a href="{{driver.url}}">{{driver.name}} {{driver.surname}}</a></h5>
            <ul style="background: #474747; padding: 10px; border-radius: 5px;">
                <li><i class="fa fa-flag"></i> {{ driver.nationality }}</li>
                <li><i class="fa fa-calendar"></i> {{ driver.birthday }}</li>
                <li><i class="fa fa-car"></i> Team: {{ driver.teamId }}</li>
                <li><i class="fa fa-trophy"></i> Number: {{ driver.number }}</li>
                <li><i class="fa fa-star"></i> Points: {{ driver.points }}</li>
            </ul>
        </div>
    </div>
</div>
{% endfor %}
{% else %}
<p>No hay datos disponibles en este momento.</p>
{% endif %}
```

This is the cycle that will show all the formula 1 drivers with their corresponding personal information, it also has an if conditional, so if the API doesn’t has any of the information we are requesting, will show a “No information available".



```
{% if news %}
{% for new in news %}
<div class="col-md-3">
    <div class="news-item left-news">
        <div class="ni-pic set-bg" data-setbg="{{new.img}}">
        </div>
        <div class="ni-text">
            <h4><a href="#">{{new.titulo}}</a></h4>
            <p>{{new.nota}}</p>
        </div>
    </div>
</div>
{% endfor %}
{% else %}
<p>No hay noticias por ahora</p>
{% endif %}
```

This is the same as the last one, same for loop and same if conditional, but in this case to show the news.