Pokemon Play

Andrés Salcedo Vera
Fundacion Universitaria Konrad Lorenz

***Resumen*—A continuación se presenta un resumen detallado del proyecto "PokemonPlays", una plataforma desarrollada utilizando el marco de trabajo Django y que incorpora una variedad de tecnologías, como HTML, CSS, JavaScript y Python, así como la gestión eficiente de paquetes. El objetivo central del proyecto es aprovechar los servicios proporcionados por la API de Pokémon, conocida como "PokeApi", para ofrecer una experiencia completa a los usuarios. Además, se explorará la aplicación de inteligencia artificial conocida como Ollama, que agrega una capa adicional de interactividad y funcionalidad a la plataforma.**

Dentro de la aplicación, los usuarios encontrarán una amplia gama de Pokémon disponibles, junto con un emocionante juego basado en estos personajes icónicos. Además, se implementará un buscador avanzado que permitirá a los usuarios buscar y acceder a descripciones detalladas y estadísticas de cada Pokémon.

I. Introducción

En el contexto actual, el desarrollo de aplicaciones web se ha convertido en una de las tecnologías más recurrentes. En este escenario, Django emerge como una herramienta fundamental, ofreciendo una integración eficiente entre Python y los lenguajes de front-end como HTML, CSS y JS. Este framework proporciona una sólida base para abordar diversas tareas, incluido el análisis de datos, lo que lo convierte en una opción valiosa para proyectos web.

Además de su capacidad para gestionar datos, Django destaca por su integración con SQL, lo que facilita la administración de bases de datos en entornos web. Esta característica es esencial para manejar de manera efectiva la información dentro de la plataforma web, optimizando así su rendimiento y funcionalidad.

Por otro lado, la inteligencia artificial (IA) ha adquirido un papel protagónico en la evolución de las aplicaciones modernas. Su creciente integración en una variedad de contextos empresariales no solo potencia la experiencia del usuario, sino que también automatiza procesos y genera utilidades significativas para las operaciones comerciales.

El conocimiento y comprensión de estas herramientas son cada vez más valorados en el mercado laboral. La capacidad para aprovechar eficazmente Django y las tecnologías de IA no solo enriquece las habilidades profesionales, sino que también proporciona una ventaja competitiva en un entorno donde la integración de estas competencias es cada vez más común y relevante.

II. Base de datos

Comencemos por la fase más crucial de nuestro proyecto, ya que esta establece los cimientos de nuestro trabajo. En primer lugar, procederemos a iniciar Django. Este paso se realiza en la terminal, donde crearemos nuestro proyecto. Esta acción generará las carpetas y la información necesaria para ejecutar nuestro programa de manera óptima.

Posteriormente, es esencial tener en cuenta varios factores adicionales. En primer lugar, la creación de un superusuario administrador, el cual nos otorgará acceso privilegiado a las bases de datos. Esto se realiza mediante un comando específico provisto por Django. En segundo lugar, es importante abordar la creación del modelo. En nuestro caso, esto implica la redacción de un script que definirá la estructura y relaciones de datos fundamentales para nuestro proyecto. Este modelo servirá como el esqueleto sobre el cual construiremos nuestra aplicación.

```python
from django.db import models

class Pokemon(models.Model):
    Name = models.CharField(max_length=50)
    HP = models.IntegerField(null=True)
    Attack = models.IntegerField(null=True)
    Defense = models.IntegerField(null=True)
    Special_Attack = models.IntegerField(null=True)
    Special_Defense = models.IntegerField(null=True)
    Speed = models.IntegerField(null=True)
    Element = models.CharField(max_length=50, null=True)
```

Figura 1. Modelo DB

Los campos definidos en nuestro modelo incluyen el nombre, los puntos de salud (HP), el valor de ataque, la defensa, el ataque especial, la defensa especial, la velocidad y el elemento. Esta selección se basa en los datos proporcionados por la API que vamos a consumir, ubicada en "https://pokeapi.co/api/v2/pokemon". Esta API nos suministrará toda la información relevante para cada Pokémon, lo que nos permite construir un modelo completo y detallado en nuestra aplicación.

Una vez creado el modelo, desarrollamos un script que nos permite poblar estos campos utilizando la API. Esta API contiene datos para 1025 Pokémon, incluyendo toda la información necesaria, como imágenes y otros detalles que pueden no ser utilizados en nuestra aplicación, pero que están disponibles para explorar en la página oficial de la API si es necesario. Este script automatizado nos permite integrar fácilmente la información de la API en nuestra base de datos, asegurando así que tengamos acceso a una amplia gama de Pokémon y sus atributos para nuestra aplicación.

```python
from django.core.management.base import BaseCommand
from estadisticas.models import Pokemon
import requests

class Command(BaseCommand):
    help = 'Carga datos de Pokémon en la base de datos'

    def handle(self, *args, **kwargs):
        # Obtener los nombres de los Pokémon
        pokemon_names = self.get_pokemon_names()

        # Obtener las estadísticas de los Pokémon
        for name in pokemon_names:
            stats = self.get_pokemon_stats(name)

            # Crear una instancia del modelo Pokemon con los datos obtenidos
            pokemon = Pokemon(
                Name=name,
                HP=stats['HP'],
                Attack=stats['Attack'],
                Defense=stats['Defense'],
                Special_Attack=stats['Special_Attack'],
                Special_Defense=stats['Special_Defense'],
                Speed=stats['Speed'],
                Element=stats['Element']
            )

            # Guardar el objeto Pokemon en la base de datos
            pokemon.save()

        self.stdout.write(self.style.SUCCESS('Datos de Pokémon cargados correctamente.'))

    def get_pokemon_names(self):
        url = 'https://pokeapi.co/api/v2/pokemon?limit=1025'
        response = requests.get(url)
        data = response.json()
        pokemon_names = [result['name'] for result in data['results']]
        return pokemon_names
```

Figura 2. script carga de datos 1

```python
def get_pokemon_stats(self, name):
    url = f'https://pokeapi.co/api/v2/pokemon/{name}'
    response = requests.get(url)
    data = response.json()

    stats = {
        'HP': data['stats'][0]['base_stat'],
        'Attack': data['stats'][1]['base_stat'],
        'Defense': data['stats'][2]['base_stat'],
        'Special_Attack': data['stats'][3]['base_stat'],
        'Special_Defense': data['stats'][4]['base_stat'],
        'Speed': data['stats'][5]['base_stat'],
        'Element': self.get_pokemon_element(data['types'])
    }
    return stats

def get_pokemon_element(self, types):
    url = types[0]['type']['url']
    response = requests.get(url)
    data = response.json()
    return data['name']
```

Figura 3. script carga de datos 1

Una vez completada esta fase, habremos finalizado la configuración de la base de datos. Podrás acceder a ella a través de la dirección http://localhost:3001/admin/estadisticas/pokemon/?o=-8p=1, en caso de que Django esté ejecutándose en el puerto 3001.

Por razones de practicidad, no detallaremos cada aspecto del proyecto, ya que implica múltiples elementos, como la modificación de las configuraciones (settings), las URLs, las vistas y la implementación de scripts diversos para alcanzar nuestros objetivos. Sin embargo, si estás interesado en explorar estos cambios, puedes encontrar información detallada directamente en cada una de las carpetas del proyecto. Además, hay numerosos recursos disponibles en YouTube que explican estos procesos en detalle. Esto es crucial, ya que la claridad en estos pasos nos permite acceder a cada página, obtener información y establecer conexiones entre los archivos de manera efectiva.

III. HTML y CSS

En la sección de HTML y CSS, abordaremos varios aspectos. Este proyecto tiene como objetivo consolidar conocimientos sobre estas tecnologías. Por lo tanto, la primera tarea es crear una página principal que contenga la información más relevante, la cual iremos detallando gradualmente en este documento.

Para comenzar, hemos optado por diseñar una página de aspecto sobrio, evitando elementos visuales excesivos que puedan distraer al usuario. En este sentido, nos enfocaremos principalmente en el uso de Bootstrap para aplicar estilos predefinidos a nuestra página. Incorporaremos elementos como una barra de navegación, un carrusel, botones y un buscador, aprovechando las funcionalidades que ofrece Bootstrap para agilizar el proceso de diseño y mantener una apariencia consistente en nuestro sitio web.

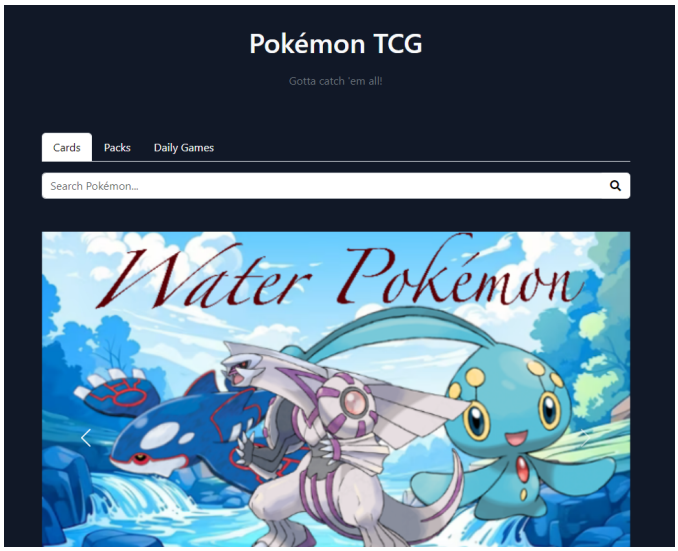

Figura 4. Enter Caption

Además, vamos a incorporar tres elementos adicionales que añadirán un toque interactivo a nuestra página. Estos elementos seleccionarán aleatoriamente a tres personajes de nuestra base de datos y los mostrarán al usuario. La idea principal es permitir a los usuarios interactuar con estos personajes, ver su información detallada, participar en juegos para obtenerlos y, en general, hacer que la experiencia en la página sea más dinámica y divertida.

Para lograr esta funcionalidad, contamos con un script de Python que nos permite obtener los detalles de tres personajes aleatorios y la información esencial asociada a cada uno. Una vez obtenida esta información mediante el script, la mostraremos dinámicamente en nuestra página HTML utilizando un ciclo For.

```python
from django.shortcuts import render
import pandas as pd
from estadisticas.models import Pokemon
from random import sample
import requests

estadisticas_csv = pd.read_csv('Docs\Estadisticas.csv')
dict_estadisticas = estadisticas_csv.set_index('Name')[['HP', 'Attack', 'Defense', 'Sp. Atk', 'Sp. Def', 'Speed', 'Total', 'Type 1']].to_dict()


import requests

def Estadisticas(request):
    # Obtener tres pokemones aleatorios de la base de datos
    random_pokemons = sample(list(Pokemon.objects.all()), 3)

    # Obtener el número de cada Pokémon y construir la URL del artwork
    for pokemon in random_pokemons:
        # Realizar solicitud a la API de Pokémon
        response = requests.get(f'https://pokeapi.co/api/v2/pokemon/{pokemon.Name.lower()}')
        if response.status_code == 200:
            data = response.json()
            # Obtener el número del Pokémon
            pokemon_number = data['id']
            # Construir la URL del artwork del Pokémon
            artwork_url = f'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/home/{pokemon_number}.png'
            # Actualizar el objeto Pokemon con la URL del artwork
            pokemon.artwork_url = artwork_url
            pokemon.save()
        else:
            # Si no se puede obtener el número del Pokémon, manejar el error de alguna manera
            pass

    return render(request, 'estadisticas/index.html', {'random_pokemons': random_pokemons})
```

Figura 5. vista para cargar 3 personajes

Estos elementos proporcionarán una experiencia envolvente al usuario, alentándolos a explorar y participar activamente en la plataforma. Esto no solo aumentará la interacción y el compromiso del usuario, sino que también agregará valor a la página al ofrecer una experiencia más rica y entretenida.

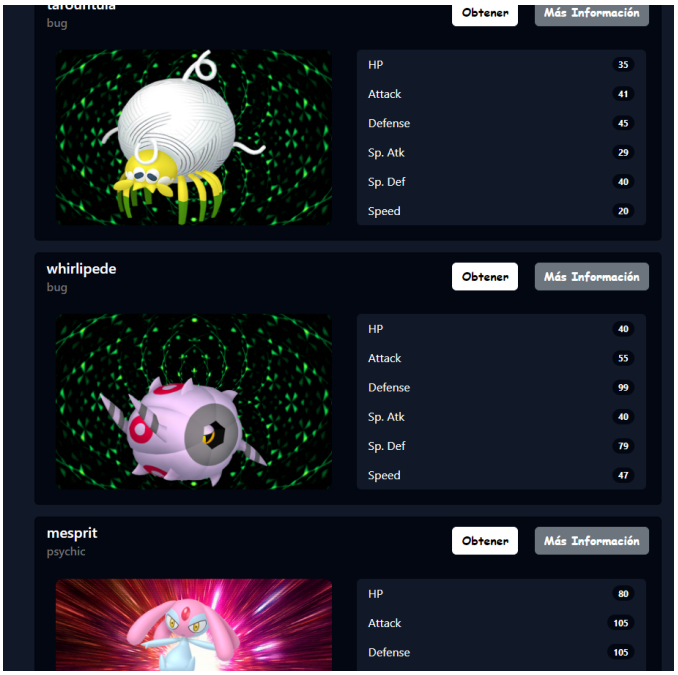

Figura 6. Interactividad en la pagina principal

IV. Buscador pokemon y API interactiva con IA

Ahora nos adentramos en uno de los aspectos más intrigantes del proyecto: el buscador de Pokémon. Esta función nos permitirá realizar búsquedas en la API de Pokémon, obteniendo así información detallada sobre cada criatura. Lo más destacado aquí son dos aspectos principales: en primer lugar, desde nuestra página web utilizaremos JavaScript para obtener el nombre del Pokémon seleccionado, el cual luego pasaremos a un script de Python encargado de recuperar la información necesaria y organizarla adecuadamente. En segundo lugar, en este punto encontraremos nuestra interacción con la inteligencia artificial Ollama. Si bien no entraremos en detalles exhaustivos en este documento debido a la extensión del tema, es crucial comprender los scripts que permiten este trabajo y su integración en nuestra aplicación.

```python
import requests
from django.shortcuts import render
from ..models import Pokemon

def buscar_pokemon(request):
    if 'pokemon_buscado' in request.GET:
        nombre_pokemon = request.GET['pokemon_buscado']
        # Buscar el Pokémon en la base de datos
        try:
            # Cambia 'nombre' a 'Name' en la consulta para que coincida con el campo de la base de datos
            pokemon = Pokemon.objects.get(Name=nombre_pokemon)

            # Realizar solicitud a la API de Pokémon para obtener más información, incluido el artwork
            response = requests.get(f'https://pokeapi.co/api/v2/pokemon/{nombre_pokemon.lower()}')
            if response.status_code == 200:
                data = response.json()
                # Obtener la URL del artwork del Pokémon
                artwork_url = data['sprites']['other']['home']['front_default']
            else:
                # Si no se puede obtener la URL del artwork, establecerla como None
                artwork_url = None

            # Renderizar la página con los detalles del Pokémon y la URL del artwork
            return render(request, 'estadisticas/pokemon_base.html', {'pokemon': pokemon, 'artwork_url': artwork_url})
        except Pokemon.DoesNotExist:
            mensaje = 'El Pokémon no fue encontrado.'
            return render(request, 'estadisticas/error.html', {'mensaje': mensaje})
    else:
        mensaje = 'Por favor, introduce un nombre de Pokémon.'
        return render(request, 'estadisticas/error.html', {'mensaje': mensaje})
```

Figura 7. Busqueda pokemon en python

este seria el resultado:

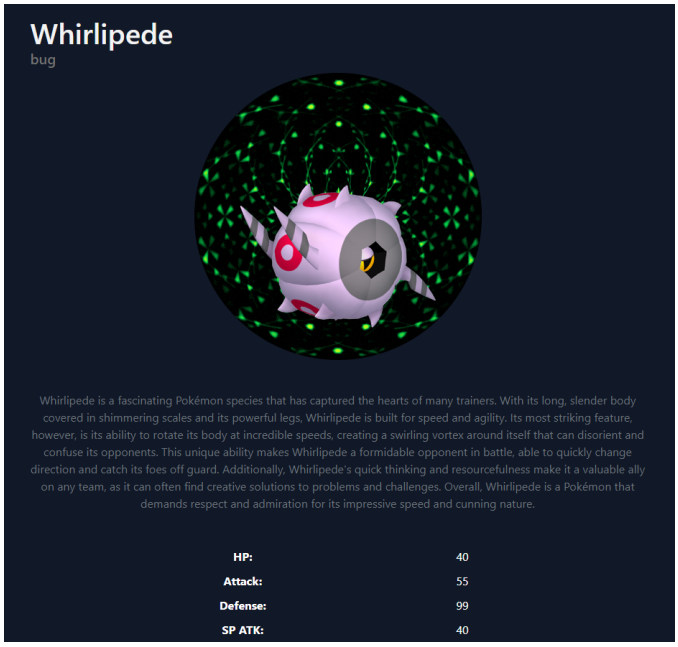

Figura 8. Enter Caption

V. Juego

Para concluir nuestra primera versión del programa, agregaremos una característica emocionante: un juego. La idea es crear un juego similar a "Smash Ants", donde los usuarios podrán aplastar hormigas, pero con la variante de utilizar Pokémon. En la pantalla, se mostrará un espacio donde los usuarios podrán presionar los Pokémon que deseen capturar. Además, habrá señuelos que restarán puntos, lo que añadirá un desafío adicional y complejizará el juego. Se visualizará un puntaje y habrá un límite de tiempo para agregarle un componente de urgencia y emoción.

En futuras versiones del programa, planeamos permitir que cada usuario tenga su propio Pokémon en su propia base,

donde podrán recolectarlos todos jugando de esta manera. Esta funcionalidad añadirá una capa adicional de personalización y compromiso para los usuarios, convirtiendo la experiencia en algo más personal y gratificante.

Figura 9. Juego

VI. Final Version I

En la versión uno de nuestro proyecto, hemos logrado implementar varias características emocionantes, como el buscador de Pokémon, el juego “Smash Ants” con Pokémon y una interfaz de usuario funcional. Sin embargo, tenemos planes ambiciosos para futuras versiones donde queremos agregar más elementos y mejorar la experiencia del usuario.

En las próximas versiones, planeamos incorporar las siguientes mejoras:

1. **Visualización de todas las cartas:** Queremos permitir a los usuarios ver todas las cartas disponibles en nuestro sistema, brindando una experiencia completa de exploración y descubrimiento.
2. **Mejoras en los packs:** Los packs de Pokémon tendrán un espacio dedicado para que los usuarios puedan acceder fácilmente a cada uno de los elementos de los Pokémon incluidos en ellos. Esto mejorará la accesibilidad y la experiencia de navegación.
3. **Adición de más juegos:** Ampliaremos la variedad de juegos disponibles en nuestra plataforma para ofrecer a los usuarios más opciones de entretenimiento y diversión.
4. **Mejoras en el diseño:** Trabajaremos en hacer la página más vistosa y atractiva visualmente. Esto incluirá mejoras en el diseño de la interfaz de usuario, el diseño de los elementos visuales y la experiencia general del usuario.

Es importante destacar que, aunque en la versión actual el repositorio puede carecer de una organización estable, esto se optimizará en las próximas versiones. La integración correcta de paquetes en un sistema como Django puede ser compleja, pero estamos comprometidos a mejorar la estructura y la organización del código para tener un enfoque más claro y organizado en futuras iteraciones del proyecto.

VII. Bibliografia

OpenAI. (2021). ChatGPT: A large language model.
Andés Salcedo Vera. (2024).