Trabajo Práctico

Página Web de Biografías de Personajes de Harry Potter

Introducción a la Programación

(Intensivo de Verano 2025)

Resumen:

Desarrollo de una aplicación web utilizando Python y Django para centralizar biografías de personajes de Harry Potter. La plataforma permite consultar datos clave (nombre, casa, actor, género, etc.), filtrar por casa de Hogwarts y visualizar información de manera interactiva.

Integrantes:

Fernández ryan

ryanfer818@gmail.com

- Ramellini Facundo

Leonardoquevedo47@gmail.com

. Introducción

El problema abordado es la falta de un repositorio centralizado para acceder a biografías detalladas de los personajes de Harry Potter. Actualmente, los usuarios deben buscar información en múltiples fuentes dispersas. Esta solución integra datos de una API externa, ofreciendo una experiencia unificada con funcionalidades como filtrado por casa y visualización de nombres alternativos.

Descripción General

La aplicación se desarrolló con Django, siguiendo la arquitectura MTV (Model-Template-View). Se implementaron:

- Modelos:No se utilizaron modelos de base de datos propios, ya que los datos se obtienen directamente de una API externa.

- Vistas:Funciones para obtener, filtrar y mostrar datos de personajes.

- Plantillas HTML/CSS: Interfaz que muestra tarjetas con información de cada personaje.

Funcionalidades Principales

a) Obtención de Datos desde la API (`getAllImages`)

>Propósito: Obtener y estructurar datos de personajes desde una API externa.

Implementación:

python

getAllImages():

try:

personajes = transport.getAllImages() # Conexión a la API

> cards = []

> for personaje in personajes:

> card = {

> "nombre": personaje.get("name", "Sin nombre"),

> "imagen": personaje.get("image", "default.jpg"),

> "casa": personaje.get("house", "Sin casa"),

> "actor": personaje.get("actor", "Sin actor"),

> "nombre_alternativo": random.choice(personaje.get("alternate_names", [])) if personaje.get("alternate_names") else None

> }

> cards.append(card)

> return cards

> except Exception as e:

> print(f"Error: {e}")

> return []

> ```

> - Retorno:Lista de diccionarios con datos de personajes.

b) Filtrado de cartas !/usr/bin/env python

```
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'main.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Dificultades y Soluciones:

- Integración con API: Se implementó manejo de excepciones para evitar fallos si la API no responde.

- Renderizado de imágenes: Configuración de `MEDIA_ROOT` en Django para mostrar imágenes desde URLs externas.

3. Conclusiones

- Django demostró ser eficaz para desarrollar aplicaciones basadas en APIs externas.

- La arquitectura MTV facilitó la separación de lógica y presentación.

Anexo: Enunciado

"Desarrollar una página web que muestre biografías de personajes de Harry Potter obtenidas desde una API externa. La aplicación debe incluir: listado general de personajes, filtrado por casa de Hogwarts, búsqueda por nombre y visualización de nombres alternativos. Tecnologías requeridas: Python, Django