Licenciatura en Sistemas

Trabajo Práctico
HARRY POTTER - API

INTRODUCCIÓN A LA PROGRAMACIÓN

(comisión verano 2025)

Resumen

El trabajo práctico consiste en la implementación de una aplicación web que permite a los usuarios explorar imágenes de personajes de la API de Harry Potter. Además, los usuarios autenticados pueden marcar y gestionar sus imágenes favoritas. La aplicación está desarrollada con Python y Django, siguiendo una arquitectura estructurada en diferentes capas para mantener una separación clara de responsabilidades

Integrantes

- **Rodríguez Cristian**, crisrodriguez.dev@gmail.com
- **Vargas Jorge**, Jorgevargas20202005@gmail.com

Docentes

- **Godoy Gonzalo**.
- **Ruiz Yair**.

Introducción

El trabajo práctico consiste en la implementación de una aplicación web que muestra imágenes de la API de Harry Potter y permite a los usuarios autenticados marcar y gestionar sus imágenes favoritas. La aplicación está desarrollada utilizando Python, Django y sigue una arquitectura que separa claramente las responsabilidades entre diferentes capas: presentación, servicio, transporte y acceso a datos.

A continuación, se presenta el código de las funciones implementadas, una breve explicación de cada una de ellas, las dificultades encontradas durante la implementación y las decisiones tomadas para resolverlas.

Desarrollo

Código implementado:

- views.py:

```
# esta función obtiene 2 listados: uno de las imágenes de la API y otro de favoritos, ambos en formato Card, y los dibuja en el template 'home.html'.
def getAllImagesAndFavouriteList(request):
    #obtenemos todas las imagenes de la API
    images = services.getAllImages()

    return images

def home(request):
    # Llama a la funcion auxiliar getAllImagesFavouriteList() y obtiene 2 listados: uno de las imágenes de la API y otro de favoritos por usuario
    favourite_list = []
    images = getAllImagesAndFavouriteList(request)
```

```
    return render(request, 'home.html', { 'images': images, 'favourite_list': favourite_list })
```

- services.py:

```
# función que devuelve un listado de cards. Cada card representa una imagen de la API de Harry Potter.
def getAllImages():

    json_collection = transport.getAllImages()
    images = []

    for object in json_collection:
        card = translator.fromRepositoryIntoCard(object)

        # Seleccionar un nombre alternativo al azar, si existen.
        if "alternate_names" in object and object["alternate_names"]:
            card.alternate_name = random.choice(object["alternate_names"])
        else:
            card.alternate_name = "No alternate names available."

        images.append(card)

    return images
```

- home.html:

```
<div class="row row-cols-1 row-cols-md-3 g-4">
        {% if images|length == 0 %}
        <h2 class="text-center">La búsqueda no arrojó resultados...</h2>
        {% else %} {% for img in images %}
        <div class="col">
 <!-- evaluar si la imagen pertenece a Gryffindor, Slytherin u otro -->
            <div class="card {% if img.house == 'Gryffindor' %}border-success{% elif img.house == 'Slytherin' %}border-danger{% else %}border-warning{% endif %} mb-3 ms-5" style="max-width: 540px; ">
                <div class="row g-0">
                    <div class="col-md-4">
                        <img src="{{ img.image }}" class="card-img-top" alt="imagen">
                    </div>
```

Explicación de las Funciones:

- *def home* ***(views.py)***

La función home llama a getAllImagesAndFavouriteList, que obtiene dos listados: uno con las imágenes disponibles por la API de Harry Potter, y otro con las imágenes favoritas del usuario (siempre que el usuario esté registrado; de lo contrario, retorna una lista vacía). Luego, pasa estas listas al template home.html para que se puedan visualizar.

- *def getAllImages* ***(services.py)***

Retorna todas las imágenes de la API de Harry Potter. Para eso, se usa services.getAllImages(), que obtiene los datos de la API, los convierte en objetos Card y los almacena en un listado para su uso posterior.

- *home.html* **(template)**

La línea a la cual le damos énfasis en esta página para lograr el cambio de color dependiendo de la familia, es:

Se usa un contenedor en HTML, indicamos una clase llamada “card” (representa una tarjeta con estilos de Bootstrap), después entre los códigos {%....%}: Es una plantilla de Django que usa Python para decidir que clases de CSS agregar. También adentro de los códigos, utilizamos condicionales como if, elif y else, que dependiendo de si su casa sea Slytherin va hacer color rojo, o si su casa es Gryffindor será de color verde y si no es ninguna de las dos, será color amarillo.

Dificultades de Implementación y Decisiones Tomadas:

Inicialmente, enfrentamos dificultades al instalar los programas necesarios para ejecutar el trabajo en nuestros equipos. Una vez superada

esta fase, seguimos las instrucciones del profesor para visualizar la página en el navegador y analizamos su funcionamiento.

Después, revisamos el código completo usando Visual Studio Code, intentando comprender su lógica general antes de completar las secciones faltantes. Identificamos que las funciones debían llamarse entre sí en un flujo lógico:

1. La función en views.py recibe los datos desde services.py.
2. services.py obtiene los datos desde transport.py y los convierte en Cards.

Otra dificultad fue integrar correctamente las funciones entre las distintas capas de la arquitectura. Optamos por dividir la obtención de datos en funciones auxiliares, lo que facilitó la depuración y el mantenimiento del código.

Finalmente, resolvimos la transformación de objetos en Cards y su almacenamiento en un listado de imágenes. La decisión de centralizar esta lógica en el módulo translator.py permitió mantener el código organizado y reutilizable.

En conclusión, el proyecto implicó una serie de decisiones técnicas y desafíos que fueron abordados mediante una clara separación de responsabilidades y una estructura del código bien definida. Esto facilitó la implementación, prueba y mantenimiento de la aplicación.

Anexo

Implementar las funciones restantes de los archivos **views.py, services.py** y modificar **home.html.** Éstas son las encargadas de hacer que las imágenes de la galería se muestren/rendericen**:**

- views.py:

- *home(request):* obtiene 2 listados que corresponden a las imágenes de la API y los favoritos del usuario, y los usa para dibujar el correspondiente template. Si el opcional de favoritos no está desarrollado, devuelve un listado vacío.

```
# esta función obtiene 2 listados: uno de las imágenes de la API y otro de
favoritos, ambos en formato Card, y los dibuja en el template 'home.html'.
def home(request):
    images = []
    favourite_list = []

    return render(request, 'home.html', { 'images': images, 'favourite_list':
favourite_list })
```

- Services.py:

- *getAllImages: obtiene un listado de imágenes convertidas en cards desde la API.*

```
# función que devuelve un listado de cards. Cada card representa una imagen de
la API de HP.
def getAllImages():
    # debe ejecutar los siguientes pasos:
    # 1) traer un listado de imágenes crudas desde la API (ver transport.py)
    # 2) convertir cada img. en una card.
    # 3) añadirlas a un nuevo listado que, finalmente, se retornará con todas
las card encontradas.
    # ATENCIÓN: contemplar que los nombres alternativos, para cada personaje,
deben elegirse al azar. Si no existen nombres alternativos, debe mostrar un
mensaje adecuado.
    pass
```

- Home.html:

- *La card debe cambiar su border color dependiendo de la casa del personaje. Si pertenece a **Gryffindor**, mostrará un borde verde; si es de **Slytherin** mostrará rojo y si es de cualquier otra casa, será naranja. Se sugiere consultar la **documentación de Bootstrap sobre Cards** y **cómo generar condicionales en Django** para tener una mejor acercamiento a la solución.*

```
<div class="row row-cols-1 row-cols-md-3 g-4">
        {% if images|length == 0 %}
        <h2 class="text-center">La búsqueda no arrojó resultados...</h2>
        {% else %} {% for img in images %}
        <div class="col">
  <!-- evaluar si la imagen pertenece a Gryffindor, Slytherin u otro -->
            <div class="card border-success mb-3 ms-5" style="max-width:
540px;">
                <div class="row g-0">
                    <div class="col-md-4">
                        <img src="{{ img.image }}" class="card-img-top"
alt="imagen">
                    </div>
```

Concluido su desarrollo, deberían ver algo como lo siguiente: