**Informe del trabajo realizado para la correcta ejecución del sitio web “Buscador Rick & Morty”**

**Introducción**

Para hacer que las imágenes de los personajes de la serie *Rick & Morty* aparecieran en nuestra aplicación web utilizamos Django, entre los tres hicimos una serie de pasos clave. A continuación, se describe el proceso de implementación, destacando las funciones desarrolladas y los ajustes necesarios en los archivos de código relevantes.

**Desarrollo**

**1. Configuración Inicial**

Inicialmente, nuestra aplicación contaba con varias plantillas HTML y una estructura de carpetas bien organizada, pero las funciones clave no estaban completamente implementadas. Identificamos las funciones faltantes y procedimos a completarlas.

**2. Obtención de Datos desde la API**

Para obtener los datos de los personajes, utilizamos la función ***fetch\_characters()*** desde el archivo transport.py. Esta función se encarga de llamar a la API de *Rick & Morty* y obtener la lista de personajes.

**3. Conversión de Datos Crudos a Objetos Card**

Implementamos la función ***getAllImages*** en el archivo services.py. Esta función:

* Llama a ***fetch\_characters*** para obtener los datos crudos.
* Recorre cada dato crudo y lo convierte en un objeto Card utilizando la función ***fromRequestIntoCard***.
* Filtra los resultados basándose en el parámetro de búsqueda input si está presente.
* Devuelve una lista de objetos Card que se utilizarán para mostrar en la web.

**4. Renderización de la Página Principal**

En el archivo views.py, completamos la función home, que:

* Llama a ***getAllImages*** para obtener la lista de imágenes.
* Pasa estas imágenes al template **home.html** para ser renderizadas.

**Modificaciones en el Template home.html**

Primero, se corrigieron las condiciones en el bloque **{% if %}** dentro del bucle **{% for img in images %}** del archivo home.html. Anteriormente, se intentaba comparar una variable true con los posibles estados, lo cual era incorrecto y no producía el resultado deseado. Se ajustaron las condiciones para comparar correctamente el atributo **img.status** con las cadenas 'Alive', 'Dead', y otros valores posibles.

**Modificación para mostrar un “borde” de color según el estado de personaje**

Para mejorar el estilo de las tarjetas de los personajes en la página home.html, hice una serie de modificaciones que permiten cambiar el borde de las tarjetas según el estado del personaje (vivo, muerto o desconocido). A continuación, paso a detallar un poco más lo que hice.

En primer lugar, me aseguré de identificar correctamente la sección del código donde se renderizan las tarjetas de los personajes dentro de un bucle **for**.

El siguiente paso fue añadir una clase condicional al contenedor principal de cada tarjeta. Utilicé los condicionales de Django para verificar el estado del personaje (**img.status**) y así aplicar la clase de borde correspondiente. Añadí una lógica para que, si el personaje está vivo (Alive), la tarjeta tenga un borde verde (**border-success**). Si está muerto (Dead), el borde sea rojo **(border-danger**). Y si el estado es desconocido (unknown), el borde sea naranja (**border-warning**).

**Modificaciones para el funcionamiento del buscador:**

El archivo modificado fue views.py

1. **Obtención del Parámetro de Búsqueda**

Se utiliza search\_input= request.GET.get ('search', None) para obtener el parámetro de búsqueda de la solicitud GET, el objetivo es capturar la entrada de búsqueda del usuario desde la URL.

1. **Filtrado de Imágenes**

La condición comprueba si search\_input tiene un valor. si search\_input no está vacío, se utiliza el “for” para filtrar las imágenes e incluir solo aquellas cuyo nombre contenga el texto de búsqueda (ignorando mayúsculas y minúsculas). El objetivo es reducir la lista de imágenes a solo aquellas que coinciden con el término de búsqueda del usuario.

1. Verificacion de Método de Solicitud

La función verifica que el método de la solitud sea POST. El objetivo es asegurar que la función solo procese solicitudes POST

1. Obtención del mensaje de búsqueda:

**Se utiliza** request.POST.get('query', '') para obtener el valor del campo query del formulario enviado. Si query (el valor) no está presente, se asigna una cadena vacía ''. **El objetivo es** capturar el término de búsqueda ingresado por el usuario en el formulario de búsqueda.

1. **Redirección Basada en la Búsqueda**:

Se verifica que search\_msg tenga un valor. Luego se redirige a la URL /home con el parámetro de búsqueda agregado a la query string (?search={search\_msg}).

Si search\_msg está vacío, se redirige a home y enviar al usuario a la página home con los resultados de la búsqueda, en este caso, sin parámetros de búsqueda

Si la solicitud no es POST, redirige al usuario a home

Diagnóstico y Solución del Problema

**Inserción de una lista de favoritos con su respectivo boton**

1. **Verificación y Ajuste de la Vista home**

Problema: La lista de favoritos (favourite\_list) no se estaba obteniendo correctamente, lo que causaba que siempre estuviera vacía.

Solución: Aseguramos que la vista home obtuviera los favoritos del usuario autenticado.

1. **Depuración de la Vista saveFavourite**

Problema: Necesitábamos asegurarnos de que la solicitud POST se estaba procesando correctamente.

Solución: Añadimos impresiones de depuración para confirmar que los datos se estaban recibiendo y guardando correctamente.

1. **Verificación de la Plantilla favourites.html**

Problema: Asegurarnos de que los favoritos se mostraran correctamente y permitir la eliminación.

Solución: Confirmamos que los favoritos se pasen y muestren correctamente en favourites.html, y ajustamos la acción del formulario para eliminar favoritos.

1. **Verificación de la Vista getAllFavouritesByUser**

Problema: Confirmar que se recuperan correctamente los favoritos del usuario.

Solución: Verificamos que la vista getAllFavouritesByUser obtenga los favoritos y los pase a la plantilla.

1. **Verificación de la Ruta de Eliminación**

Problema: Asegurar que la eliminación de favoritos funcione correctamente.

Solución: Ajustamos la vista y la URL para manejar correctamente el favourite\_id.

***Sitios utilizados:***

1. [***https://getbootstrap.com/docs/4.0/components/card/***](https://getbootstrap.com/docs/4.0/components/card/)*“Bootstrap’s cards”*
2. [***https://medium.com/powered-by-django/if-else-conditions-in-django-templates-5b3658cbf287***](https://medium.com/powered-by-django/if-else-conditions-in-django-templates-5b3658cbf287) *“How To Use If/Else Conditions In Django Templates”*
3. [***https://es.stackoverflow.com/questions/461609/como-hacer-una-condicional-en-plantilla-html-de-django-python***](https://es.stackoverflow.com/questions/461609/como-hacer-una-condicional-en-plantilla-html-de-django-python)*“*[*Como hacer una condicional en plantilla html de django. Python*](https://es.stackoverflow.com/questions/461609/como-hacer-una-condicional-en-plantilla-html-de-django-python)*”*