

Buscador Rick & Morty



Introducción a la programación 2do semestre 2024

Grupo 4

Integrantes:

- Elian Emanuel Romero Burgos
[Telegram](http://t.me/mintandchocolate15): <http://t.me/mintandchocolate15>
[Github](https://github.com/mintandchocolate): <https://github.com/mintandchocolate>
- Mariela Acosta
[Telegram](http://t.me/Mia41312): <http://t.me/Mia41312>
[Github](https://github.com/MIA281073): <https://github.com/MIA281073>

Introducción

Este trabajo hace uso del framework web Django y de una API que contiene información sobre los personajes de la serie “Rick and Morty”. El objetivo del trabajo es crear una página web que muestre a cada uno de los personajes de la serie y que presente la siguiente información sobre cada uno de ellos:

- Una imagen del personaje (si existe)
- El nombre del personaje
- Si se encuentra vivo, muerto, o si no se conoce su estado
- El primer y último episodio en el que aparecieron

Para presentar dicha información de una forma más sencilla, prolija y agradable a la vista, se utilizan “cards”, que son plantillas de información para cada personaje. Para poder saber el estado de un personaje de una forma simple también se incluyen bordes coloreados en cada card, siendo el verde usado para los personajes vivos, rojo para los personajes muertos, y finalmente naranja para los personajes cuyo estado es desconocido.

Código y funciones implementadas

views.py

1. La lista `images` tiene la función de guardar la información de la API, ya procesada como un json, para que después pueda ser usada por `home.html`. Al principio esta lista está vacía, por lo tanto hacemos:

```
Images = services.getAllImages()
```

Entonces llenamos la lista `images` con los json de cada personaje usando la función `getAllImages()` que se encuentra en `services.py`.

services.py

1. Primero se añadió la funcionalidad para las cards y la funcionalidad que lleva a cabo la conversión de los datos crudos de la API.

```
from app.layers.transport import transport
from app.layers.utilities import card
```

2. Luego definimos la función `getAllImages`. Dentro de la función existe una lista llamada `json_collection` que está vacía, procedemos a llenarla usando

`transport.getAllImages()`. Luego tenemos otra lista en la función llamada `images`, y por cada `json` en la lista `json_collection`, lo agregamos a `images` en forma de `cards` usando `translator.fromRequestIntoCard()` y finalmente terminamos la función con un `return images`, es decir que la función entrega una lista de `cards`.

Definir la función en retrospectiva no es tan complicado pero empezando el trabajo práctico este fue uno de los mayores obstáculos ya que fue difícil deducir que funciones había que llamar para resolverlo

home.html

1. La resolución de `home.html` fue bastante sencilla a comparación, siguiendo las reglas de condicionales en `django` y la guía para `cards` fue bastante intuitivo el código. Para obtener los bordes de colores se agregaron condicionales seguidos del ciclo `{% for img in images %}` presente en la línea 38.

Los condicionales que se utilizaron fueron:

```
{% if img.status == "alive" %}
```

En la línea `<div class="card mb-3 ms-5" style="max-width: 540px;">`

Se agregó en `class=""` la clase `card border-success`, la cual coloca un borde de color verde alrededor de la `card`.

La línea quedó de esta forma:

```
<div class="card mb-3 ms-5 card border-success" style="max-width: 540px;">
```

Luego se continuó con el resto de la definición de la `card` que ya venía hecha en el trabajo. Esto se repitió 2 veces más para los personajes muertos y los personajes con estado desconocido usando:

```
{% elif img.status == "dead" %}
```

Se utilizó `card border-danger`, que agrega un borde rojo.

```
{% else %}
```

Se utilizó `card border-warning`, que agrega un borde naranja.

Finalmente siguiendo las reglas de condicionales en `django` se cerró utilizando

```
{% endif %}
```

2. Los colores mostrados dentro de la `card` al lado del status de los personajes se colocaron usando los mismos `if`, `elif` y `else` mostrados arriba, cambiando los `"true"` que venían por defecto por `img.status` (excepto el `else` que permanece igual)

Extras

Inicio de sesión

1. Añadir el extra de inicio de sesión no resultó muy difícil, el sistema para iniciar sesión ya estaba hecho y solamente hacía falta definir la función encargada del logout dentro de views.py. La función fue definida de la siguiente forma:

exit(request):

```
logout(request)
```

```
return redirect(index_page)
```

logout(request) se encarga de cerrar la sesión mientras que return redirect(index_page) regresa al usuario a la página de bienvenida.