

Trabajo Practico Introducción a la Programación

El trabajo consiste en desarrollar una aplicación web que permita buscar imágenes de la serie "Rick y Morty".

Primer punto, views.py:

```
def home(request):
    images = services.getAllImages()
    favourite_list = []
    return render(request, 'home.html', { 'images': images, 'favourite_list':
favourite_list })
```

En el modulo views.py vemos que la función “home(request)” contiene 2 listados (uno de las imágenes de la API y otro de los favoritos del usuario). Para obtener los listados usamos la función “getAllImages”(función definida en el módulo service.py), la cual transforma un listado de los datos “crudos” de la API en cards y los agrega a “images”. Entonces, tenemos una lista con todas las imágenes y otra llamada “favourite_list” que contiene las imágenes guardadas por el usuario.

Services.py:

En este módulo se define la función “getAllImages” explicada anteriormente. Primero, se importa el módulo transport (que ya está desarrollado) para convertir las imágenes. json_collection contiene los datos “crudos” de la API :

```
json_collection = transport.getAllImages()
```

Luego se recorre cada uno de estos datos en json_collection y se los convierte en cards mediante translator.fromRequestIntoCard(image) (con translator también importado previamente) y se los agrega “en images”:


```
images = [translator.fromRequestIntoCard(image) for image in json_collection]
```




En el módulo home.html, para que se muestren los bordes de colores según el estado del personaje (alive, dead o unknown) se agregó la siguiente secuencia de condicionales:

```
<div
    class="card mb-3 ms-5 {% if img.status == 'Alive'%} border-success {%
elif img.status == 'Dead'%} border-danger {% else %} border-warning {%endif%}"
    style="max-width: 540px">
```

Lo que se hace es evaluar img.status, es decir evaluar el estado del personaje. Si img.status es “Alive” se aplica border-success (para bordes verdes). Si el status es “Dead” se aplica border-

danger (bordes rojos) y si no es ninguno de los otros dos estados, se aplica border-warning (bordes amarillos). Estos border-success, danger y warning son propiedades de Bootstrap.

De forma parecida, para adaptar los círculos de colores al lado del nombre del personaje según su estado se utilizaron condicionales de Django. Por ejemplo, si el personaje está vivo, se muestra  seguido de “Alive” y así dependiendo de que condición se cumple.

```
<strong>
    {% if img.status == 'Alive'%}
    <p> {{img.status}}</p>
    {% elif img.status == 'Dead'%}
    <p> {{img.status}}</p>
    {%else%}
    <p> {{img.status}}</p>
    {%endif%}
</strong>
```

Para realizar el punto del buscador, en el modulo views.py, se creo una funcion llamada “getAllImagesAndFavouriteList”. La cual toma datos del buscador y de las imagenes guardadas en favoritos:

```
def getAllImagesAndFavouriteList(search_msg, request):
    images = services.getAllImages(search_msg)
    favourite_list = services.getAllFavourites(request)

    return images, favourite_list
```

Tambien se modifico en el modulo services.py y se le agrego un input a json_collection:

```
def getAllImages(input=None):
    # obtiene un listado de datos "crudos" desde La API, usando a
    transport.py.
    json_collection = transport.getAllImages(input)

    # recorre cada dato crudo de La colección anterior, Lo convierte en una
    Card y lo agrega a images.
    images = [translator.fromRequestIntoCard(image) for image in
    json_collection]
    return images
```