

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 módulo 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 módulo `views.py`, se creo una función llamada "getAllImagesAndFavouriteList". La cual toma datos del buscador y de las imágenes guardadas en favoritos:

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

    return images, favourite_list
```

También se modificó en el módulo `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
```

FAVORITOS:

Para implementar la funcionalidad de agregar y eliminar de favoritos, se trabajó en los módulos views.py y services.py.

En views.py se modificaron las funciones getAllFavouritesByUser, saveFavourite y deleteFavourite:

```
@login_required
def getAllFavouritesByUser(request):
    favourite_list = services.getAllFavourites(request)
    return render(request, 'favourites.html', { 'favourite_list':
favourite_list })
```

En esta funcion, solo se modificó la variable favourite_list, donde llama a la funcion getAllFavourites alojada en services.py, que a su vez hace lo siguiente:

```
def getAllFavourites(request):
    if not request.user.is_authenticated:
        return []
    else:
        user = get_user(request)
        favourite_list = repositories.getAllFavourites(user) # buscamos desde
el repositories.py TODOS los favoritos del usuario (variable 'user').
        mapped_favourites = []

        for favourite in favourite_list:
            card = translator.fromRepositoryIntoCard(favourite) #
transformamos cada favorito en una Card, y lo almacenamos en card.
            mapped_favourites.append(card)

        return mapped_favourites
```

Continuando en views.py, se cambió lo siguiente en la función saveFavourite:

```
@login_required
def saveFavourite(request):
    services.saveFavourite(request)
    return redirect("home")
```

Lo que hace es llamar a la función saveFavourite ubicada en services.py y luego retorna "redirect("home")" que redirige automáticamente a la página home con los cambios visibles al presionar el botón Anadir a favoritos.

Por otro lado, saveFavourite en services.py hace lo siguiente:

```
def saveFavourite(request):
    fav = translator.fromTemplateIntoCard(request) # transformamos un request
del template en una Card.
    fav.user = get_user(request) # le asignamos el usuario correspondiente.
    return repositories.saveFavourite(fav) # lo guardamos en la base.
```

Ahora, la última función (deleteFavourite) en views.py:

```
@login_required
def deleteFavourite(request):
    services.deleteFavourite(request)
    return redirect("favoritos")
```

Lo que hace es llamar a la función deleteFavourite ubicada en services.py y redirige a la página favoritos de nuevo al presionar el botón de eliminar de favoritos.

A su vez, la función deleteFavourite definida en services.py, hace esto:

```
deleteFavourite(request):
    favId = request.POST.get('id')
    return repositories.deleteFavourite(favId) # borramos un favorito por su
ID. (ESTA FUNCION NO FUE MODIFICADA)
```

Lo demás relacionado con el funcionamiento de favoritos, ya estaba implementado, como es el caso del desarrollo de los botones en los módulos home.html y favourites.html y la interacción con la base de datos en el módulo repositories.py