

TRABAJO PRÁCTICO GRUPAL

INTRODUCCIÓN A LA PROGRAMACIÓN:

Buscador Rick & Morty

Nos encontramos con un trabajo práctico ya resuelto, donde nos encargamos de realizar cambios, agregar y corregir algunas de las funcionalidades más importantes.

Iniciamos con dificultades a la hora de empezar a trabajar con un programa que no habíamos visto previamente, con ayuda de un integrante del grupo que nos brindó información para avanzar con el trabajo práctico pudimos llevarlo a cabo. Cuando llegamos al uso de Git, no estábamos al tanto de su funcionamiento a la hora de clonar un repositorio, así mismo también cuando llegamos a la parte del funcionamiento del Home.html se nos dificultó ya que no lográbamos comprenderlo. Con ayuda de videos de YouTube y también de un trabajo de investigación pudimos llevarlo adelante.




Cambios que se realizaron en el Views.py con la función del Home, llama a la función Services.GetAllImages para obtener los personajes de la API, cuyos personajes si están en favoritos pasa por un condicional que después los renderiza en el temple Home.html

```
14 def home(request):
15     # Llamar al services para obtener las imágenes, pasando el request
16     images = services.getAllImages(request)
17     # Obtener favoritos del usuario (si aplica)
18     favourite_list = []
19     return render(request, 'home.html', {
20         'images': images,
21         'favourite_list': favourite_list
22     })
```

En Services.py, específicamente en GetAllImages, agarramos los datos que están en la API de cada personaje del transport.GetAllImages y los transformamos en lo que son los datos crudos en objetos de translator.fromRequestIntoCard, les pusimos un condicional de los favoritos y después de esto nos retornan las imágenes para que se utilicen en el view.

```
9 def getAllImages(request, input=None):
10     # Obtener datos crudos desde transport.py
11     data = transport.getAllImages(input)
12     # Convertir los datos crudos en objetos Card usando el translator
13     images = [translator.fromRequestIntoCard(item) for item in data]
14     # Si el usuario está autenticado, marcar favoritos
15     if request.user.is_authenticated:
16         user = get_user(request)
17         favourites = {fav['name'] for fav in repositories.getAllFavourites(user)}
18         for img in images:
19             img.is_favourite = img.name in favourites
20     return images
```

En Home.html los cambios realizados son los estados de los personajes, ya que desde el inicio nos aparecían todos de color naranja, y a la hora de modificarlos necesitábamos que los estados de los personajes coincidieran con los colores (Vivo "Live" = color verde. Muerto "Dead" = color Rojo. Desconocido "Unknown" = color Naranja) con un borde de un pixel aproximadamente. Los estados de los personajes los sacamos desde el Service.py de las cartas de la API donde nos da como resultado el estado actual de cada uno.

```
<div class="col-md-8">
  <div class="card-body">
    <h3 class="card-title">{{ img.name }}</h3>
    <p class="card-text">
      <strong>
        {% if img.status == 'Alive' %}  {{ img.status }}
        {% elif img.status == 'Dead' %}  {{ img.status }}
        {% else %}  {{ img.status }}
        {% endif %}
      </strong>
    </p>
  </div>
</div>
```


Utilizando el sistema de “Login” de Django usando el usuario predeterminado “Admin”, que es importado como “User” utilizando la plantilla de “Login” donde se definió una variable llamada “LoginUser” que se implementó en views.py utilizando un If y Else para validar si el usuario está logueado o no retornando al usuario a la pagina de Bienvenida. También se usó el Login Required que es usado para cuando el usuario está logueado a la página, esto permite que el usuario tenga a favoritos, a las imágenes, etc. Se agrego una plantilla llamada “Register” que es otro adicional donde se agregó mediante un link de Login.py que se puede observar al final del código, esto también se implemento en los Views.py y en los Url, donde los views se los define usando el método post, esto hace que agregue el nombre, apellido, nombre de usuario, correo electrónico y contraseña.

```
2 {% block content %}
3 <div class="login-form" style="text-align: center;">
4   <form action="{% url 'login' %}" method="POST" style="display: inline-block;">
5     {% csrf_token %}
6
7     <h2 class="text-center">Inicio de sesión</h2>
8     <div class="form-group" style="margin-bottom: 5%;">
9       <input type="text" name="username" id="username" class="form-control" placeholder="Usuario" required="">
10    </div>
11    <div class="form-group" style="margin-bottom: 5%;">
12      <input type="password" name="password" id="password" class="form-control" placeholder="Contraseña" required="">
13    </div>
14    <div class="form-group">
15      <button type="submit" class="btn btn-primary btn-block">Ingresar</button>
16    </div>
17  </form>
18  <p>¿No tienes cuenta? <a href="{% url 'register' %}">Regístrate aquí</a></p>
19 </div>
20 {% endblock %}
```

Uno de los errores que observamos a la hora de hacer funcionar la página, que nos pedía una segunda contraseña, también se ve una segunda contraseña agregando un If si la contraseña 1 y la contraseña 2 coincidían y otros 2 if si el usuario esta en uso. Mientras que en el Url se agregó un views register al lado de los otros views como Home, Login, y esto hace que defina la lista de registro en el url. Para añadir el registro en el repositorio tuve que usar el código GIT ADD con el nombre de la lista del registro llamado “Register.html” y Git Commit para avisar que el registro fue añadido.

```
6 urlpatterns = [
7     path('', views.index_page, name='index-page'),
8     path('login/', LoginView.as_view(template_name='login.html'), name='login'),
9     path('home/', views.home, name='home'),
10    path('buscar/', views.search, name='buscar'),
11    path('register/', views.register, name='register'),
12
13    path('favourites/', views.getAllFavouritesByUser, name='favoritos'),
14    path('favourites/add/', views.saveFavourite, name='agregar-favorito'),
15    path('favourites/delete/', views.deleteFavourite, name='borrar-favorito'),
16
17    path('exit/', views.exit, name='exit'),]
```

Esto nos llevó a los siguientes resultados:




Beth Smith

● **Alive**

Última ubicación: Earth
(Replacement Dimension)

Episodio inicial: Earth
(Replacement Dimension)




Jerry Smith

● **Alive**

Última ubicación: Earth
(Replacement Dimension)

Episodio inicial: Earth
(Replacement Dimension)




Abradolf Lincler

● **unknown**

Última ubicación: Testicle Monster
Dimension

Episodio inicial: Earth
(Replacement Dimension)



Adjudicator Rick

● **Dead**

Última ubicación: Citadel of Ricks

Episodio inicial: unknown

[Galería](#) [Iniciar sesión](#)

Inicio de sesión

[Ingresar](#)

¿No tienes cuenta? [Regístrate aquí](#)

Registro

[Registrar](#)

¿Ya tienes cuenta? [Inicia sesión aquí](#)

Buscador Rick & Morty

1 2 3



Rick Sanchez

● Alive

Última ubicación: Citadel of Ricks

Episodio inicial: Earth (C-137)

♥ Añadir a favoritos



Morty Smith

● Alive

Última ubicación: Citadel of Ricks

Episodio inicial: unknown

♥ Añadir a favoritos



Summer Smith

● Alive

Última ubicación: Earth (Replacement Dimension)

Episodio inicial: Earth (Replacement Dimension)

♥ Añadir a favoritos

Participantes del grupo:

Lautaro Mancera (lautaromanceradaniel@gmail.com) DNI: 47.071.823

Federico Velazquez (Velazquez.agustin.federico@gmail.com) DNI: 46.414.286

Matias Benitez (MatiasBenitez95@gmail.com) DNI: 39.600.229