UNGS IP COMISION 3

INFORME SOBRE LA REALIZACION DEL TP FINAL: GALERIA DE IMAGENES DE LA NASA

INTEGRANTES DEL GRUPO:

POGREBAJ MARCOS

LUNA TOMAS

En este informe se dara una explicacion de como se fue armando el tp final, cuales fueron las tareas de cada miembro y demas.

Las tareas a realizar de cada integrante fueron las siguientes:

-Pogrebaj: Terminar las funciones getAllImagesAndFavouriteList(request) y home(request) y tambien el opcional de inicio de sesion.

-Luna: Terminar la funcion de getAllImages(input=None) y tambien el opcional del buscador.

A continuacion se explicara como fuimos implementando las distintas funciones.

1) getAllImagesAndFavouriteList(request)

```
def getAllImagesAndFavouriteList(request):
    images, favourite_list=services_nasa_image_gallery.getAllImages(request),
services_nasa_image_gallery.getAllFavouritesByUser(request)
    return images, favourite_list
```

Decidimos que todas las funciones que hagan uso de dos listas siempre tengas a las mismas en una sola linea de codigo por un tema de comodidad, en esta funcion se puede ver que llamamos a las funciones escribiendo antes el modulo en el que estan porque sino nos tiraba error, no hubo mucho de investigacion en esta funcion ni en las dos que quedan de lo que seria el tp basico, sino que hubo que explorar bastante los archivos y el codigo del trabajo.

Lo que hicimos fue agrupar a las dos listas vacias y llamar a las funciones getAllImages y getAllFavouritesByUser en las mismas y luego retornarlas.

2) home(request)

```
def home(request):
    # llama a la función auxiliar getAllImagesAndFavouriteList() y obtiene 2
listados: uno de las imágenes de la API y otro de favoritos por usuario*.
    # (*) este último, solo si se desarrolló el opcional de favoritos; caso
contrario, será un listado vacío [].
    images, favourite_list= getAllImagesAndFavouriteList(request)
```

```
    return render(request, 'home.html', {'images': images, 'favourite_list': favourite_list} )
```

Esta funcion es muy parecida a la 1ra, las dos usan dos listas, una de favoritos y otra de imagenes, péro esta en vez de retornar dos listas utiliza un render.

Usamos dos listas vacias y adentro se llamo a la funcion getAllImagesAndFavouriteList la cual retorna dos listas, las mismas luego son retornadas utilizando un render.

3)getAllImages(input=None)

```
def getAllImages(input=None):
    # obtiene un listado de imágenes desde transport.py y lo guarda en un json_collection.
    # ¡OJO! el parámetro 'input' indica si se debe buscar por un valor introducido en el buscador.
    json_collection= transport.getAllImages(input=None)
    images = []
    # recorre el listado de objetos JSON, lo transforma en una NASACard y lo agrega en el listado de images. Ayuda: ver mapper.py.
    for image in json_collection:
        nasa_card= fromRequestIntoNASACard(image)
        images.append(nasa_card)
    return images
```

Esta funcion es un poco diferente de las anteriores ya que convierte archivos json en nasa_card, en si una imagen a una imagen con descripcion, titulo y fecha, se llama a getAllImages desde transport a una lista que va a guardar los archivos json y luego se recorre esa lista por medio de un for y se las convierte en nasa_card, las mismas luego son appendeadas a la lista images que sera retornada por la funcion.

Llegamos a esto explorando el codigo y las funciones del tp, una vez que llegamos a mapper.py y despues de varios errores pudimos hacer que funcione.

4)Inicio de sesion

Para poder implementar este opcional tuvimos que modificar o agregar codigo en 3 partes distintas del tp:

-MAIN/URLS.PY

```
path('accounts/', include('django.contrib.auth.urls')),
```

Esta linea de codigo la sacamos del video que se recomienda mirar en el mismo punto del trabajo donde se explica el opcional de inicio de sesion y es parte de una funcionalidad de Django llamada authentication views, la misma nos da acceso a varias urls, en este caso usamos solo accounts/login y accounts/logout.

Fue añadida en urls.py.

-TEMPLATES/REGISTRATION/LOGIN.URLS

```
<form action="{% url 'login' %}" method="POST" style="display: inline-block;">
```

En la linea 3 de login.urls en action= no habia nada y en el video del cual hablamos en el item anterior la persona que explica dice que hay que poner la url login en ese espacio, el opcional funciona perfectamente asi, pero tambien sin eso.

-TEMPLATES/HEADER.URLS

```
<a class="nav-link" href="{% url 'logout' %}">Salir</a> {% else %}
                        <a class="nav-link" href="{% url 'login' %}">Iniciar
sesión</a> {% endif %}
```

En la linea 35 y 36 de header.urls, luego de los href de cada linea habia solo un asterisco entre comillas, llegamos hasta aca porque nos dimos cuenta de que cuando se clickeaba el boton iniciar sesion aparecia un asterisco en la url de la galeria de imagenes y no nos llevaba a ningun lado, empezamos a buscar en el codigo buscando el boton de iniciar sesion y llegamos al header, ahi nos dimos cuenta de que faltaban las url de login y logout y las agregamos luego de borrar los asteriscos.

5)Buscador

```
def request(request):
    images, favourite_list = getAllImagesAndFavouriteList(request)
    request_msg = request.POST.get('query', '')

    if not request_msg.strip():
        # Si no se ingresó ningún texto de búsqueda, podrías manejarlo aquí (por ejemplo, refrescar la página)
        return render(request, 'your_template.html', {'images': images, 'favourites': favourite_list})
    else:
        # Filtra imágenes que contienen el texto de búsqueda en el título o la descripción
        filtered_images = [image for image in images if request_msg.lower() in image['title'].lower() or request_msg.lower() in image['description'].lower()]
```

return render(request, 'your_template.html', {'images': filtered_images, 'favourites': favourite_list})

Paso a paso:

1. *Importa una función*: Se trae la función render del módulo django.shortcuts. Esta función se utiliza para mostrar una página web.

2. *Define la función*: Se crea una función llamada request que se usará para manejar la solicitud de búsqueda.

3. *Intentar obtener datos*: Se intenta obtener todas las imágenes y la lista de favoritos llamando a una función getAllImagesAndFavouriteList con el objeto de solicitud.

4. *Capturar errores*: Si ocurre un error al obtener las imágenes y la lista de favoritos, se captura el error y se muestra una página con un mensaje de error.

5. *Obtener la búsqueda del usuario*: Se recoge lo que el usuario ha escrito en el campo de búsqueda del formulario. Si el usuario no ha escrito nada, se toma como una cadena vacía.

6. *Eliminar espacios en blanco*: Se eliminan los espacios en blanco al principio y al final del texto de búsqueda.

7. *Comprobar si la búsqueda está vacía*: Se verifica si el usuario no ha escrito nada en el campo de búsqueda.

8. *Mostrar todas las imágenes*: Si el usuario no escribió nada, se muestra la página con todas las imágenes y la lista de favoritos.

9. *Filtrar imágenes*: Si el usuario escribió algo, se filtran las imágenes para encontrar aquellas cuyo título o descripción contienen el texto de búsqueda, sin importar si es mayúscula o minúscula.

10. *Mostrar imágenes filtradas*: Finalmente, se muestra la página con las imágenes que coinciden con la búsqueda y la lista de favoritos.