Estructura del Informe

1. Introducción
2. Código de las Funciones Implementadas
3. Explicación de las Funciones
4. Dificultades de Implementación y Decisiones Tomadas

Introducción

En este trabajo, hemos desarrollado una aplicación web que permite a los usuarios visualizar una galería de imágenes proporcionadas por la API de la NASA y gestionar una lista de imágenes favoritas. El proyecto se basa en el framework Django y sigue una arquitectura en capas, donde la capa de vista se comunica con servicios específicos para obtener los datos necesarios. A continuación, se presentan las funciones implementadas, su explicación y las dificultades encontradas durante el desarrollo.

Insertar código:

```
# capa de vista/presentación

# si se necesita algún dato (lista, valor, etc), esta capa SIEMPRE se comunica con services_nasa_image_gallery.py


from django.shortcuts import redirect, render

from .layers.services import services_nasa_image_gallery

from django.contrib.auth.decorators import login_required

from django.contrib.auth import logout

from .layers.transport.transport import getAllImages

from .layers.generic.mapper import fromRequestIntoNASACard


# función que invoca al template del índice de la aplicación.

def index_page(request):

    return render(request, 'index.html')


# auxiliar: retorna 2 listados -> uno de las imágenes de la API y otro de los favoritos del usuario.

def getAllImagesAndFavouriteList(request):
```

```
    api_images = getAllImages()  # obtienes todas las imágenes de la API

    favourite_list = []


    # con el mapper transformamos el json en una estructura que el html pueda leer

    images = []

    for api_image in api_images:

        image = fromRequestIntoNASACard(api_image)

        images.append(image)


    return images, favourite_list


# función principal de la galería.

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} )


# función utilizada en el buscador.

def search(request):

    # Obtén el mensaje de búsqueda del POST request

    search_msg = request.POST.get('query', None)


    if search_msg is not None and search_msg != '':

        # Si se proporcionó una palabra clave, obtén las imágenes que coinciden con esa palabra clave
```

```
        images = getAllImages(input=search_msg)
    else:
        # Si no se proporcionó una palabra clave, obtén todas las imágenes
        images = getAllImages()

    # Convierte cada imagen a un formato que tu template pueda entender
    images = [fromRequestIntoNASACard(api_image) for api_image in images]

    # Obtén la lista de imágenes favoritas del usuario
    favourite_list = []

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

# las siguientes funciones se utilizan para implementar la sección de favoritos: traer los favoritos de un usuario, guardarlos, eliminarlos y desloguearse de la app.
@login_required
def getAllFavouritesByUser(request):
    favourite_list = []
    return render(request, 'favourites.html', {'favourite_list': favourite_list})

@login_required
def saveFavourite(request):
    pass

@login_required
def deleteFavourite(request):
    pass
```

```
@login_required

def exit(request):

    pass
```

Dificultades de Implementación y Decisiones Tomadas

El desarrollo de esta aplicación presentó varios desafíos, principalmente relacionados con la integración de datos de la API y su transformación para la presentación en la vista. Utilizar un mapper para convertir los datos de la API en una estructura adecuada fue una decisión clave que facilitó el desarrollo y mantenimiento del código. Además, implementar la lógica de búsqueda y manejar diferentes casos de uso (como búsquedas vacías) fue otro desafío que se resolvió mediante una lógica condicional en la función search.

Finalmente, las funciones relacionadas con la gestión de favoritos (guardar, eliminar) y la autenticación del usuario aún no están implementadas completamente, pero se prevé que estas funcionalidades requerirán una cuidadosa gestión de la persistencia de datos y las sesiones de usuario.