Universidad General Sarmiento – primer semestre del 2024

Trabajo Práctico Final
Introducción a la Programación

Docentes: Fernando Velcic, Patricia Bagnes.

Integrantes del grupo: Iván Martínez, Cristian Rodriguez, Mathias Maldonado

Tarea a Realizar:

Implementar las funciones restantes de los archivos **views.py** y **services_nasa_image_gallery.py**. Éstas son las encargadas de hacer que las imágenes de la galería se muestren/rendericen:

- **views.py**:

home(request): invoca a getAllImagesAndFavouriteList(request) para obtener 2 listados que utilizará para renderizar el template.

```
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, favorite_list = []

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

getAllImagesAndFavouriteList(request): invoca al servicio correspondiente para obtener 2 listados, uno de las imágenes de la API y otro -si corresponde- de los favoritos del usuario.

```
# Auxiliar: retorna 2 listados -> uno de las imágenes de la API y otro de
los favoritos del usuario.
def getAllImagesAndFavouriteList(request):
    images = []
    favourite_list = []

    return images, favourite_list
```

- **services_nasa_image_gallery.py:**

getAllImages(input=None): obtiene un listado de imágenes de la API. El parámetro *input*, si está presente, indica sobre qué imágenes debe filtrar/traer.

```
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 = []
    images = []

    # recorre el listado de objetos JSON, lo transforma en una NASACard y
lo agrega en el listado de images. Ayuda: ver mapper.py.

    return images
```

Introducción:

El trabajo práctico consiste en la implementación de una aplicación web que muestra imágenes de la API de la NASA y permite a los usuarios autenticados marcar y gestionar sus imágenes favoritas. La aplicación está desarrollada utilizando Python, Django y sigue una arquitectura que separa claramente las responsabilidades entre diferentes capas: presentación, servicio, transporte y acceso a datos.

A continuación, se presenta el código de las funciones implementadas, una breve explicación de cada una de ellas, las dificultades encontradas durante la implementación y las decisiones tomadas para resolverlas.

Código Implementado:

views.py

def 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})
```

getAllImagesAndFavouriteList(request)

```python
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

# auxiliar: retorna 2 listados -> uno de las imágenes de la API y otro de los favoritos del usuario.
def getAllImagesAndFavouriteList(request):
    # Obtener todas las imágenes desde la API
    images = services_nasa_image_gallery.getAllImages()

    # Si el usuario está autenticado, obtener su lista de favoritos
    if request.user.is_authenticated:
        user = request.user
        favourite_list = services_nasa_image_gallery.getAllFavouritesByUser(user)
    else:
        favourite_list = []

    return images, favourite_list
```

services_nasa_image_gallery.py

getAllImages(input=None)

```python
from ..transport import transport
from ..dao import repositories
from ..generic import mapper
from django.contrib.auth import get_user

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)
    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 objeto in json_collection:
        nasa_card = mapper.fromRequestIntoNASACard(objeto)
        images.append(nasa_card)

    return images
```

Explicación de las Funciones:

- *home* ***(views.py)***

La función llama a getAllImagesAndFavouriteList, esta función obtiene dos listados, uno con las imágenes disponibles por la API de la NASA, y la otra las imágenes favoritas del usuario (siempre que el usuario este registrado, sino retorna una lista vacía). Luego pasa las dos listas obtenidas (images, favourite_list) a la plantilla home.html para que se pueda visualizar.

- *getAllImagesAndFavouriteList* ***(views.py)***

Retorna dos listados, uno de las imágenes de la API y otro de los favoritos del usuario. La función llama a services_nasa_image_gallery.getAllImages para obtener una lista de imágenes desde la API de la NASA, esta llamada se hace importando services_nasa_image_gallery la cual se encuentra en otra carpeta y ahí adentro esta la función getAllImages. Utilizamos if y else, y preguntamos si el usuario esta autenticado, nos devuelva el usuario(user) y luego a través de favourite_list llamamos a services_nasa_image_gallery.getAllFavouritesByUser(user) el cual nos devuelve la imagen favorita del usuario. Por último, la función retorna ambas listas ('images, favourite_list') para que puedan ser usadas en otra parte, como por ejemplo en la función anterior 'home'.

- *getAllImages* ***(services_nasa_image_gallery.py)***

Obtiene un listado de imágenes desde la API de la NASA.
La función llama transport.getAllImages(input) para obtener imágenes desde la API, y utilizamos transport, el cual esta importando el módulo desde otra carpeta donde se encuentra implementada la función getAllImages.
Una vez que obtenemos el listado de imágenes desde la API, utilizamos un ciclo for para recorrer el listado de objetos JSON y cada objeto JSON es transformado en un objeto NASAcard utilizando la función mapper.fromRequestIntoNASACard y estos objetos NASACard se agregan al listado de imágenes. Por último, la función retorna las imágenes para que puedan ser usadas en otra parte del código, como por ejemplo en la función anterior 'getAllImagesAndFavouriteList'.

Dificultades de Implementación y Decisiones Tomadas:

Al principio, como grupo, enfrentamos la dificultad inicial de instalar los programas necesarios para ejecutar el trabajo en nuestros computadores. Una vez superada esta primera dificultad, seguimos los pasos proporcionados por el profesor para hacer visible la página en nuestro navegador y navegamos en ella para analizar su funcionamiento en la web.

Luego, analizamos el código completo utilizando la herramienta Visual Studio Code, intentando entender la lógica general del código, no solo la parte que faltaba completar. Después de revisar el código, procedimos a completar las secciones faltantes. Primero, leímos atentamente la consigna, entendiendo que las funciones debían llamarse entre sí: la primera función proporcionaba la información obtenida de la segunda, y la segunda obtenía la información de la tercera.

Otra de las principales dificultades fue llamar a las funciones que no estaban en la misma hoja del código, sino en otra ubicación. Tuvimos que asegurar la correcta integración entre las capas de transporte, servicio y presentación. La decisión de crear

funciones auxiliares para manejar la obtención de datos y la lógica permitió una mejor separación de responsabilidades y facilitó la depuración de errores.

Asegurar que las funciones manejen correctamente los casos en los que el usuario no está autenticado fue crucial. Pudimos resolverlo utilizando lo aprendido en clase mediante un ciclo for, verificando si el usuario estaba registrado para devolver la imagen, y si no, retornando una lista vacía.

Finalmente, quedaba resolver la creación de objetos, transformándolos en una NASACard y agregándolos a un listado de imágenes. Nuestra solución a este problema fue recorrer el listado de datos json_collection, el cual importaba la API de imágenes. La decisión de centralizar esta lógica en el módulo mapper ayudó a mantener el código organizado y reutilizable.

En resumen, el proyecto implicó una serie de decisiones técnicas y desafíos que fueron abordados mediante una clara separación de responsabilidades y una estructura del código bien definida. Esto facilitó la implementación, prueba y mantenimiento de la aplicación.