### **Informe de Desarrollo y Funcionamiento del Sistema de Galería de Imágenes de la NASA**

#### **Introducción**

Este informe describe el desarrollo y funcionamiento de un sistema de galería de imágenes de la NASA, implementado con el framework Django. Se detallan las funcionalidades principales, la estructura del código y la implementación de la funcionalidad de búsqueda y gestión de favoritos.

#### **Funcionalidades Principales**

* **Visualización de la Galería de Imágenes**: Los usuarios pueden ver una colección de imágenes obtenidas de una API.
* **Búsqueda de Imágenes**: Los usuarios pueden buscar imágenes en la galería mediante un formulario de búsqueda.
* **Gestión de Favoritos**: Los usuarios autenticados pueden añadir y eliminar imágenes de su lista de favoritos.
* **Autenticación de Usuarios**: Los usuarios pueden registrarse, iniciar sesión y cerrar sesión.

#### **Estructura del Código**

El código se organiza en diferentes capas para separar las responsabilidades:

* **Views**: Maneja las solicitudes HTTP y devuelve respuestas HTTP.
* **Services**: Contiene la lógica de negocio, como la obtención de imágenes de la API y la gestión de favoritos.
* **Models**: Define las estructuras de datos y las relaciones entre ellas.

**Historial de Modificaciones:**

**getAllImages**

Importaciones y dependencias:

-from ..generic import mapper: Importa el módulo “mapper” desde el paquete “generic”, que contiene funciones de mapeo de datos.

Enfocándonos en la función, está diseñada para obtener un listado de imágenes desde un servicio externo (transport.py) y almacenarlas en un json\_collection. Utiliza un bucle “for” para iterar sobre json\_collection, aplicando una función de mapeo (mapper.fromRequestIntoNASACard(x)) a cada elemento para transformarlo en un formato adecuado para una "NASACard". Por último las imágenes transformadas se agregan a la lista “images”.

**saveFavourite**

Esta función se encarga de guardar un favorito en la base de datos. mapper.fromTemplateIntoNASACard(request): Utiliza una función de mapeo (fromTemplateIntoNASACard) del módulo “mapper” para transformar los datos recibidos en el request del template en un objeto NASACard.

user = get\_user: Utiliza la función “get\_user” de Django para obtener el usuario actual que realizó la solicitud (request).

fav.user = user: Asigna el usuario obtenido al campo user del objeto fav.

repositories.saveFavourite(fav): Llama a una función “saveFavourite” del módulo “repositories” para guardar el favorito (fav) en la base de datos.

**getAllFavouritesByUser**

El proposito de esta función es obtener los favoritos de un usuario en específico y transformarlos en una “NASACard”.

get\_user(request): Obtiene el usuario actual de la solicitud.

repositories.getAllFavouritesByUser(user): Llama a una función “getAllFavouritesByUser” del módulo “repositories” para obtener todos los favoritos almacenados en la base de datos para el usuario específico.

mapper.fromRepositoryIntoNASACard(favourite): Utiliza una función de mapeo (fromRepositoryIntoNASACard) del módulo “mapper” para transformar cada favorito recuperado de la base de datos en un objeto “NASACard”. Los objetos “NASACard” transformados se agregan a la lista mapped\_favourites, que luego se devuelve como resultado de la función.

**Paginado**

En la paginacion de la tabla se le aplicaron márgenes, alineación y más ajustes visuales.

float: right;: Hace que los elementos de paginación se alineen a la derecha.

margin: 0 0 5px;:Establece márgenes de 0 en la parte superior e izquierda, y de 5px en la parte inferior.

padding-bottom: 4%;: Agrega un relleno en la parte inferior equivalente al 4% del tamaño del contenedor padre.

.pagination li a: Establece estilos para los enlaces dentro de cada ítem de la paginación, aquí podrían definirse propiedades como el color, tamaño de fuente, bordes, etc.

**search**

El propósito de la función es realizar búsquedas de imágenes basadas en un mensaje de búsqueda (query) ingresado por el usuario.

images = [] y favourite\_list = []: Inicializa dos listas vacías para almacenar las imágenes y la lista de favoritos.

images, favourite\_list = getAllImagesAndFavouriteList(request): Llama a la función getAllImagesAndFavouriteList(request) para obtener todas las imágenes disponibles y la lista de favoritos relacionados con el usuario actual.

search\_msg = request.POST.get('query', ''): Obtiene el mensaje de búsqueda ingresado por el usuario desde la solicitud POST.

if search\_msg == '': search\_msg = 'space': Si el mensaje de búsqueda está vacío, se establece como 'space'.

filtered\_images = []: Inicializa una lista vacía para almacenar las imágenes filtradas que coinciden con los criterios de búsqueda.

Bucle for image in images: Itera sobre todas las imágenes obtenidas.

* if search\_msg.lower() in image.title.lower() or search\_msg in image.description.lower(): Comprueba si el mensaje de búsqueda está contenido en el título o la descripción de cada imagen.
* Si una imagen cumple con los criterios de búsqueda, se agrega a la lista filtered\_images.

return render(request, 'home.html', {'page\_obj': filtered\_images, 'favourite\_list': favourite\_list}): Devuelve una respuesta renderizando la plantilla home.html, pasando como contexto las imágenes filtradas (filtered\_images) y la lista de favoritos (favourite\_list).

**views.py**

Primero se importaron varias librerías y módulos necesarios para el funcionamiento del script.

-django.contrib.auth: Proporciona funciones para la autenticación de usuarios.

-django.shortcuts: Contiene funciones útiles para renderizar plantillas y redirigir solicitudes.

-passlib.hash: Se utiliza para el hashing seguro de contraseñas.

-sqlite3: Para interactuar con la base de datos SQLite.

**Función “index\_page”:**

Esta función renderiza la página principal del sitio web (index.html) cuando se solicita la ruta raíz (/).

**getAllImagesAndFavouriteList**

Esta función es utilizada para obtener los datos(imágenes) necesarios desde la API de la nasa y la lista de favoritos del usuario.

**login\_user**

La función maneja el proceso de inicio de sesión de un usuario. Cuando se envía un formulario POST con los campos “username” y “pasword” se autentica al usuario utilizando “authenticate()” de Django. Si las credenciales son válidas, el usuario se autentica usando “login()” y se redirige a la página principal (index). Si las credenciales son incorrectas, se muestra un mensaje de error en la plantilla de inicio de sesión (login.html). Tambien se importa la librería passlib.hash para realizar la comparación con la contraseña encriptada.

**logout\_user**

Esta función se encarga del cierre de sesión de un usuario autenticado. Utiliza “logout()” de django para cerrar la sesión actual del usuario y luego redirige a la página de inicio de sesión “login”.

**Alta Usuarios**

Se agrega la funcionalidad de registro para nuevos usuarios que se guardan en la base de datos con los datos necesarios (nombre,apellido,usuario,mail y contraseña) mientras no haya un duplicado ya en la base, para despues poder usarse normalmente en el sistema

#### **Conclusión**

El sistema de galería de imágenes de la NASA implementado en Django permite a los usuarios visualizar, buscar y gestionar sus imágenes favoritas de manera eficiente. Se solucionaron problemas relacionados con la búsqueda y se verificó que todas las funcionalidades principales estuvieran operativas, brindando una experiencia de usuario fluida y agradable.