TRABAJO PRÁCTICO
GALERÍA DE IMÁGENES DE LA NASA

INFORME

Introducción a la programación - COM-07

Profesores
Sergio Santa Cruz
Nazareno Avalos
Nahuel Sauma

Grupo 6
Brizuela Cristian
Vázquez Jeremías

2024

Índice Del Informe:

1. Introducción

Este trabajo práctico consistió en implementar una aplicación web fullstack usando Django Framework que permite consultar y visualizar imágenes de la API pública de la NASA. Además de mostrar las imágenes, la aplicación permite filtrarlas por categorías (como Space, moon, sun, entre otras). Y a los usuarios autenticados, guardar imágenes como favoritas, las cuales se listan en una sección especifica.

2. Funciones implementadas

2.1 views.py

Se implementó la funcionalidad para consultar y mostrar imágenes provenientes de la API de la NASA. Las imágenes se muestran en formato de cards con un título, una descripción y la imagen.

- **home()**

```python
# función principal de la galería.
def home(request):
    images, favourite_list = getAllImagesAndFavouriteList(request)
    return render(
        request, "home.html", {"images": images, "favourite_list": favourite_list}
    )

```

Esta función se ocupa de renderizar la plantilla home.html. Obtiene dos listas, una con las imágenes desde la API, y otra con las imágenes favoritas del usuario, desde la base de datos (en caso de que haya uno logueado)

- **search()**

```python
# función utilizada en el buscador.
def search(request):
    search_msg = request.POST.get("query", "")

    if search_msg:
        images, favourite_list = getAllImagesAndFavouriteList(request, search_msg)
    else:
        images, favourite_list = getAllImagesAndFavouriteList(request)

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

```

Esta función es la encargada de filtrar las imágenes mostradas según un valor ingresado por el usuario.

Obtiene el valor de búsqueda desde el campo query. Si el usuario ha ingresado un término de búsqueda, se llama a getAllImagesAndFavouriteList con el término de búsqueda como parámetro. Si no se ha ingresado un término de búsqueda, se llama a la misma función sin parámetros.

Al finalizar, renderiza la plantilla home.html, con las listas filtradas según el valor ingresado

- **getAllImagesAndFavouriteList()**

```python
def getAllImagesAndFavouriteList(request, input=None):
    favourite_list = services_nasa_image_gallery.getAllFavouritesByUser(request)

    if input is None:
        images = services_nasa_image_gallery.getAllImages()
    else:
        images = services_nasa_image_gallery.getImagesBySearchInputLike(input)

        if not images:
            images = services_nasa_image_gallery.images_not_found()

    return images, favourite_list

```

Función auxiliar, utilizada en home() y search(). Recibe una solicitud y un parámetro opcional. Primero, obtiene la lista de imágenes favoritas del usuario mediante getAllFavouritesByUser, desde la base de datos. Si no se le pasa un parámetro de entrada, obtiene todas las imágenes con getAllImages. Si se provee una entrada, realiza una búsqueda con getImagesBySearchInputLike. Si la búsqueda no arroja resultados, asigna un valor indicando que no se encontraron imágenes usando images_not_found. Por último, devuelve una lista de imágenes y la lista de favoritos del usuario.

Nota: Agregamos el parámetro de entrada input, para podes reutilizar la función, en search()

- **getAllFavouritesByUser()**

```python
@login_required
def getAllFavouritesByUser(request):
    favourite_list = services_nasa_image_gallery.getAllFavouritesByUser(request)
    return render(request, "favourites.html", {"favourite_list": favourite_list})

```

Esta función obtiene la lista de favoritos de un usuario autenticado y la muestra en la plantilla "favourites.html".

- **saveFavourite()**

```python
@login_required
def saveFavourite(request):
    services_nasa_image_gallery.saveFavourite(request)
    return search(request)

```

Guarda un favorito para el usuario autenticado basado en la solicitud recibida y luego redirige a la función de búsqueda.

Nota: En esta función intentamos que se mantenga la vista y el parámetro de búsqueda, al guardar un favorito. El problema es que, al recargar la vista, cambia la ruta. Esto no lo pudimos resolver

- **deleteFavourite()**

```python
@login_required
def deleteFavourite(request):
    services_nasa_image_gallery.deleteFavourite(request)
    return redirect("favoritos")
```

Elimina un favorito del usuario autenticado según la solicitud y redirige a la página de favoritos.

- Register()

```python
def register(request):
    data = {"form": CustomUserCreationForm()}
    if request.method == "POST":
        formulario = CustomUserCreationForm(data=request.POST)
        if formulario.is_valid():
            formulario.save()
            username = formulario.cleaned_data["username"]
            password = formulario.cleaned_data["password1"]
            user_email = formulario.cleaned_data["email"]

            user = authenticate(
                username=username,
                password=password,
            )

            # envío correo con las credenciales de inicio de sesión
            subject = "NASA Image Gallery - Registro exitoso"
            message = f"Gracias por registrarte\nTus datos de ingreso son:\nUsuario: {username}\nContaseña: {password}"
            recipient = user_email

            send_mail(
                subject,
                message,
                settings.EMAIL_HOST_USER,
                [recipient],
                fail_silently=False,
            )
            messages.success(request, "¡Te has registrado exitosamente!")
            login(request, user)

            return redirect('home')

        data["form"] = formulario

    return render(request, "registration/signup.html", data)
```

Primero, inicializa un formulario de creación de usuario personalizado (CustomUserCreationForm). Si la solicitud es de tipo POST, valida y guarda los datos del formulario. Luego, autentica al usuario recién registrado, envía un correo electrónico con las credenciales de inicio de sesión, muestra un mensaje de éxito al usuario y lo redirige a la página principal del sistema ('home'). Si hay errores en el formulario, vuelve a renderizar la página de registro con los errores mostrados.

2.2 services_nasa_image_gallery

- **getAllImages()**

```python
def getAllImages(input=None):
    images= []
    json_collection = transport.getAllImages(input)

    for elem in json_collection:
        elem = mapper.fromRequestIntoNASACard(elem)
        images.append(elem)

    return images
```

Esta función obtiene una lista de imágenes desde el módulo transport y las guarda en una colección JSON. Luego, recorre cada elemento de la colección JSON, lo transforma en un objeto NASACard utilizando el módulo mapper, y agrega cada NASACard a una lista de imágenes que finalmente devuelve.

- **getImagesBySearchInputLike()**

```python
def getImagesBySearchInputLike(input):
    return getAllImages(input)
```

Esta función llama a getAllImages pasando un parámetro de entrada para buscar imágenes que coincidan con el valor proporcionado, devolviendo la lista de imágenes filtradas.

- **saveFavourite()**

```python
def saveFavourite(request):
    fav = mapper.fromTemplateIntoNASACard(request)
    fav.user = get_user(request)
    return repositories.saveFavourite(fav)
```

Esta función transforma una solicitud desde el template en un objeto NASACard, asigna el usuario correspondiente, y guarda el objeto NASACard en la base de datos a través del módulo repositories.

- **getAllFavouritesByUser()**

```python
def getAllFavouritesByUser(request):
    if not request.user.is_authenticated:
        return []
    else:
        user = get_user(request)
        favourite_list = repositories.getAllFavouritesByUser(user)
        mapped_favourites = []

        for favourite in favourite_list:
            nasa_card = mapper.fromRepositoryIntoNASACard(favourite)
            mapped_favourites.append(nasa_card)

        return mapped_favourites
```

Esta función revisa si el usuario está autenticado, obtiene todos los favoritos del usuario desde el repositorio, los transforma en objetos NASACard utilizando el módulo mapper, y devuelve una lista de estos objetos NASACard.

- **deleteFavourite()**

```python
def deleteFavourite(request):
    favId = request.POST.get("id")
    return repositories.deleteFavourite(favId)
```

Esta función obtiene el ID de un favorito desde la solicitud POST y lo borra del repositorio utilizando el módulo repositories.

- **images_not_found()**

```python
def images_not_found():
    return [
        {
            "image_url": "../static/img/nf.png",
            "title": "Images not found",
            "description": "Please, try again",
        }
    ]
```

Esta función devuelve una lista con una serie de datos, indicando que no se encontraron imágenes y sugiriendo intentarlo nuevamente.

3. Errores y aprendizajes

Dentro de los errores que fuimos topándonos en la prueba y error mientras hacíamos nuestro sitio, encontramos errores como:

- El hacer push sin darnos cuenta de que había un error, que investigando, nos dimos cuenta que se podía resolver utilizando git log –oneline para encontrar el identificador del commit que no queríamos pushear, luego utilizamos git revert (identificador del commit que no queríamos pushear) y esto nos hace un commit que va a deshacer todos los cambios del commit que pusheamos.
- Tuvimos que investigar mucho sobre Django, como, por ejemplo, el cómo subir imágenes, como crear formularios, como hacer el registro de usuarios, etc.
- Encontramos problemas a la hora de comprender como funcionaban todas las funciones, pero investigando las pudimos comprender y también pudimos crear nuevas como la función images_not_found() que cuando haces una búsqueda que no se relacione a las NASA cards te devuelve el mensaje junto a una foto de que no se pudo encontrar una imagen en base a los escrito previamente en el buscador.
- Tuvimos problemas para la creación de formularios, pero lo resolvimos investigando en los sitios que dejamos en la sección de material consultado, lo que nos facilitó la tarea de poder crearlos y utilizarlos.

4. Conclusión

En base al trabajo realizado, pudimos sacar distintas conclusiones, como, por ejemplo, que, al buscar por varios sitios de internet distintos, tanto videos como páginas y blogs, la información se puede comprender y utilizar de manera mucho más eficiente además de ayudarnos a resolver los problemas por los que iniciamos la búsqueda en estos sitios. Otra conclusión puede ser el cómo trabajando en equipo se pueden resolver los problemas mas rápidamente.

5. Material consultado

- Mozilla Developer Network. (n.d.). Django. Recuperado el 21/06/24, de https://developer.mozilla.org/es/docs/Learn/Server-side/Django

- YouTube. (n.d.). "Documentación de Python - Curso Completo" [Lista de reproducción]. YouTube. https://www.youtube.com/playlist?list=PL3XiwX4b6ls0Ye0IkKgZpxzXh3EGe_TOJ

- Code Band. (2024). *Django Authentication Tutorial | Part 1: Login and Logout in Django | Example By Code Band* [Video]. YouTube. https://www.youtube.com/watch?v=oKuZQ238Ncc

- Jasim, A. K. (s.f.). *cb_django-sending-emails*. GitHub. Recuperado el 26/06/24, de https://github.com/akjasim/cb_django-sending-emails