

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. Índice del informe.

2. Introducción de nuestro trabajo, Funciones implementadas:
views.py

- home()
- search()

3. Funciones de views.py

- getAllImagesAndFavouriteList()
- getAllFavouritesByUser()

4. Funciones de views.py

- saveFavourite()
- deleteFavourite()

5. Función de views.py:

- Register()

6. Funciones de services_nasa_image_gallery:

- GetAllImages()
- GetImagesBySearchInputList()
- SaveFavourite()

7. Funciones de services_nasa_images_gallery:

- GetAllFavouriteByUser()
- DeleteFavourite()
- images_not_found()

8. Conclusión del trabajo y material utilizado para el trabajo.

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 específica.

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()**

```
1 # función principal de la galería.
2 def home(request):
3     images, favourite_list = getAllImagesAndFavouriteList(request)
4     return render(
5         request, "home.html", {"images": images, "favourite_list": favourite_list}
6     )
7
```

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()**

```
1 # función utilizada en el buscador.
2 def search(request):
3     search_msg = request.POST.get("query", "")
4
5     if search_msg:
6         images, favourite_list = getAllImagesAndFavouriteList(request, search_msg)
7     else:
8         images, favourite_list = getAllImagesAndFavouriteList(request)
9
10    return render(
11        request,
12        "home.html",
13        {"images": images, "favourite_list": favourite_list, "search_msg":
14        search_msg},
15    )
15
```

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()`**

```
1 def getAllImagesAndFavouriteList(request, input=None):
2     favourite_list = services_nasa_image_gallery.getAllFavouritesByUser(request)
3
4     if input is None:
5         images = services_nasa_image_gallery.getAllImages()
6     else:
7         images = services_nasa_image_gallery.getImagesBySearchInputLike(input)
8
9         if not images:
10             images = services_nasa_image_gallery.images_not_found()
11
12     return images, favourite_list
13
```

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 poder reutilizar la función, en `search()`

- **`getAllFavouritesByUser()`**

```
1 @login_required
2 def getAllFavouritesByUser(request):
3     favourite_list = services_nasa_image_gallery.getAllFavouritesByUser(request)
4     return render(request, "favourites.html", {"favourite_list": favourite_list})
5
```

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

- **saveFavourite()**

```
1 @login_required
2 def saveFavourite(request):
3     services_nasa_image_gallery.saveFavourite(request)
4     return search(request)
5
```

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()**

```
1 @login_required
2 def deleteFavourite(request):
3     services_nasa_image_gallery.deleteFavourite(request)
4     return redirect("favoritos")
```

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

- Register()

```
1 def register(request):
2     data = {"form": CustomUserCreationForm()}
3     if request.method == "POST":
4         formulario = CustomUserCreationForm(data=request.POST)
5         if formulario.is_valid():
6             formulario.save()
7             username = formulario.cleaned_data["username"]
8             password = formulario.cleaned_data["password1"]
9             user_email = formulario.cleaned_data["email"]
10
11             user = authenticate(
12                 username=username,
13                 password=password,
14             )
15
16             # envío correo con las credenciales de inicio de sesión
17             subject = "NASA Image Gallery - Registro exitoso"
18             message = f"Gracias por registrarte\nTus datos de ingreso son:\nUsuario:
19 {username}\nContraseña: {password}"
20             recipient = user_email
21
22             send_mail(
23                 subject,
24                 message,
25                 settings.EMAIL_HOST_USER,
26                 [recipient],
27                 fail_silently=False,
28             )
29             messages.success(request, "¡Te has registrado exitosamente!")
30             login(request, user)
31
32             return redirect('home')
33
34     data["form"] = formulario
35
36     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()**

```
1 def getAllImages(input=None):
2     images= []
3     json_collection = transport.getAllImages(input)
4
5     for elem in json_collection:
6         elem = mapper.fromRequestIntoNASACard(elem)
7         images.append(elem)
8
9     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()**

```
1 def getImagesBySearchInputLike(input):
2     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()**

```
1 def saveFavourite(request):
2     fav = mapper.fromTemplateIntoNASACard(request)
3     fav.user = get_user(request)
4     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()**

```
1 def getAllFavouritesByUser(request):
2     if not request.user.is_authenticated:
3         return []
4     else:
5         user = get_user(request)
6         favourite_list = repositories.getAllFavouritesByUser(user)
7         mapped_favourites = []
8
9         for favourite in favourite_list:
10             nasa_card = mapper.fromRepositoryIntoNASACard(favourite)
11             mapped_favourites.append(nasa_card)
12
13     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()**

```
1 def deleteFavourite(request):
2     favId = request.POST.get("id")
3     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()**

```
1 def images_not_found():
2     return [
3         {
4             "image_url": "../static/img/nf.png",
5             "title": "Images not found",
6             "description": "Please, try again",
7         }
8     ]
```

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

3. Conclusiones

En este trabajo practico concluimos, que, nos hemos encontrado con diversos problemas a medida que avanzamos en la programación de la página web, como por ejemplo a la hora de usar distintos programas para poder hacer funcionar el sitio, programas de los cuales nos hemos tenido que informar desde qué función cumplen hasta cómo poder utilizarlos para poder avanzar y mejorar el funcionamiento de nuestro sitio web. Entre ellos Django Framework, Git, GitHub, Visual Studio code entre otros. Algo que facilito encarar los problemas que se nos presentaron fue también el darnos cuenta como facilita el trabajo el poder desarrollar nuestro sitio web en equipo manteniendo una buena comunicación.

4. 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