Universidad Nacional de General Sarmiento

Licenciatura en Sistemas

Trabajos Práctico

PAGINA DE LA NASA

Introducción a la Programación

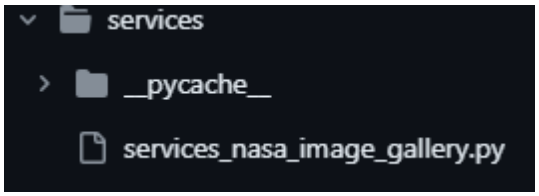
(semestre y año)

Resumen: El trabajo se trata de modificar distintas funciones con el objetivo de hacer funcionar una página web sobre la NASA, teniendo nuestros conflictos y soluciones

Integrantes: Esposito Tadeo (tadeoesposito11@gmail.com)
Kinder Gonzalo (gonkin1301@gmail.com)
Vairoli Santiago (santiagovairoli22@gmail.com)
Zarza Miguel (zarzamiguel741@gmail.com)

1. Introducción Este TP se trata de alterar líneas de código con el objetivo de lograr un buen funcionamiento y optimización de una página web. Para ello, primero realizamos una visualización de lo que ya estaba hecho. Debuggeando observamos la organización del trabajo y el concepto de multicapas. Esta complementado de una forma adecuada para la buena conexión y comunicación en el trabajo, con todas las partes que lo componen.
2. Desarrollo Luego de buscar por diversas fuentes como internet, YouTube, etc. Al principio nos enfocamos por las funciones fundamentales para que funcione la página:
-carpeta: Nasa_image_gallery
Views.py: vimos la primera función que nos pedían modificar (*home(request*) y a través los comentarios averiguamos de donde provienen las imágenes predeterminadas

```python
def home(request):
    # Llama a la función auxiliar getAllImagesAndFavouriteList() y obtiene 2 listados: uno de 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 = []
    return render(request, 'home.html', {'images': images, 'favourite_list': favourite_list} )
```



Allí dentro apreciamos la función getAllImages que investigamos y vimos que las imágenes están en transport.py

```python
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.
```

Que con ayuda de las aclaraciones pudimos comprender su funcionamiento y modificar la lógica de esta capa de la siguiente manera:

```python
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.

#TE busco y ordeno las imagenes por defecto
    json_collection = transport.getAllImages(input)

    for object in json_collection:
            nasa_card=mapper.fromRequestIntoNASACard(object)
            images.append(nasa_card)
    return images
```

Con este programa buscamos y ordenamos las imágenes predeterminadas ayudándonos con json_collection y el nasa_card

Al terminar de modificar y de verificar que estén bien hechas decidimos llamarlas a la función home(request) agregándolas en images

```
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 usuar
    # (*) este último, solo si se desarrolló el opcional de favoritos; caso contrario, será un listado vacío [].
    images = []
    favourite_list = []

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

Gracias a las dos funciones anteriores, pudimos resolver más fácil la función getAllImagesAndFavouriteList(request) completando devuelta images

```
def getAllImagesAndFavouriteList(request):
    images = []
    favourite_list = []

    ##
    images = services_nasa_image_gallery.getAllImages()
    return images, favourite_list
```

Funciones extras:

Función search:

A través de los criterios que nos indica el raedme, lo que hicimos fue configurar el buscador para que se busque imágenes relacionadas con la palabra que indica el usuario. Si el usuario no ingresa nada, retorna al home.

```
# función utilizada en el buscador.
def search(request):
    images, favourite_list = getAllImagesAndFavouriteList(request)
    search_msg = request.POST.get('query', '')

    # si el usuario no ingresó texto alguno, debe refrescar la página; caso contr
    pass
```

El código de esta condición quedaría así:

```
def search(request):
    images, favourite_list = getAllImagesAndFavouriteList(request)
    search_msg = request.POST.get('query', '')

    # si el usuario no ingresó texto alguno, debe refrescar la página; caso contrario, debe filtrar
    #pass
    #TE en caso de escribir una busqueda busco las imagenes y refresco la pantalla
    if search_msg !='':
        images = services_nasa_image_gallery.getAllImages(search_msg)
        return render(request, 'home.html', {'images': images, 'favourite_list': favourite_list} )
    elif search_msg =='':

       return home(request)
```

Con este programa cumplimos con las condiciones que nos pedía el readme.

Opcional traductor:

Primero instalamos estas dos extensiones

```
from googletrans import Translator
from langdetect import detect
```

Aprovechando la función search, modificamos la lógica para que cuando el usuario ingrese una palabra determine si es una palabra distinta del inglés, la traduzca al inglés y funcione el buscador.

```
def search(request):

    # si el usuario no ingresó texto alguno, debe refrescar la página; caso contrario, debe filtrar aque
    # pass


    if search_msg !='':
        # Detectar el idioma del cuadro de texto que ingreso el
        current_lang  = detect(search_msg)

        # Traducir la consulta de búsqueda solo si no está en inglés
        if current_lang  != 'en':
                translator = Translator()
                translated_msg = translator.translate(search_msg, src='es', dest='en').text
        else:
            translated_msg = search_msg

        images = services_nasa_image_gallery.getAllImages(translated_msg)
        return render(request, 'home.html', {'images': images, 'favourite_list': favourite_list} )
    else:
        return redirect('home')
```

Con esta función tuvimos el problema que al utilizar palabras muy cortas (por ejemplo: sol), no detecta el idioma correcto y no la traduce.

Función iniciar sesión y favoritos:
Modificamos para hacer una lista de favoritos, guardando los favoritos del usuario dentro de esta o dando la opción de eliminarlos.

```python
@login_required
def getAllFavouritesByUser(request):
    ##
    favourite_list = Favourite.objects.filter(user=request.user)
    return render(request, 'favourites.html', {'favourite_list': favourite_list})


@login_required
def saveFavourite(request):
    ##pass
    if request.method == 'POST':
        try:
            saved_favourite = services_nasa_image_gallery.saveFavourite(request)
            if saved_favourite:
                return redirect('favoritos')  # Redirige a la vista de favoritos si e
            else:
                messages.error(request, 'Error al guardar el favorito.')
        except Exception as e:
            messages.error(request, f'Error: {str(e)}')
```

```python
@login_required
def deleteFavourite(request):
    ##pass
    if request.method == "POST":
        success = services_nasa_image_gallery.deleteFavourite(request)
        if success:
            return redirect('favoritos')  # Redirigir a la lista de favor
        else:
            messages.error(request, 'Error al guardar el favorito.')
            return redirect('favoritos')  # Aquí redirigimos de todos mod
    else:
        return redirect('favoritos')
```

```python
def getAllImages(input=None):
    # obtiene un listado de imágenes desde transport.py y lo
    # ¡OJO! el parámetro 'input' indica si se debe buscar por
    json_collection = []

    images = []

    # recorre el listado de objetos JSON, lo transforma en un

#TE busco y ordeno las imagenes por defecto
    json_collection = transport.getAllImages(input)

    for object in json_collection:
            nasa_card=mapper.fromRequestIntoNASACard(object)
            images.append(nasa_card)
    return images
```

Si el usuario esta logueado, se muestra el apartado favoritos. En caso contrario, no aparece.

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

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

        return mapped_favourites
```

Función iniciar sesión:
Cambiamos el header.html para poder iniciar sesion.

```
{% if request.user.is_authenticated %}
<a class="nav-link" href="{% url 'logout' %}">Salir</a> {% else %}
<a class="nav-link" href="{% url 'login' %}">Iniciar sesión</a> {% endif %}
```

```
path('accounts/', include('django.contrib.auth.urls'))
```

3. Conclusiones Como conclusión, podemos afirmar que el proyecto tuvo su dificultad por el manejo de códigos que no tenemos conocimientos, pero por medio de investigación y tutoriales, encontramos la forma de resolverlo. También, nos familiarizamos con el concepto, ya mencionado, de la arquitectura de multicapas. A la hora de seleccionar los opcionales, elegimos los que consideramos más fundamentales para que funcione una página.

Anexo: https://github.com/ungs-ip/ip-public-repo