

Trabajo Práctico de Introducción a la Programación



Materia: Introducción a la Programación

Comisión: 05

Profesores: Fernando Velcic, Patricia Bagnes

Alumnos: Dutra Octavio, Piris Paz Abel

Fecha de entrega: 25/11/2024

Introducción:

El propósito principal de este trabajo es comprender el funcionamiento de la herramienta GIT, su terminal GIT Bash y sus páginas de almacenamiento de scripts en la nube, las cuales nos permiten crear/trabajar en repositorios y también colaborar en ellos con otras personas. Para realizar este trabajo debemos comprender la aplicación de tecnologías como CSS, HTML, JSON y Python con su framework de Django, aplicadas a una maqueta de aplicación web y cómo subir las actualizaciones de nuestro programa a la nube o web de la herramienta GIT, GitHub con el fin de que nuestros miembros puedan ver nuestros cambios. En este trabajo se nos brindó la consigna de realizar y agregar cambios en la maqueta de una aplicación web ya desarrollada casi en su totalidad.

CAMBIOS:

Las modificaciones del archivo services.py fueron necesarias para que views funcione correctamente, importamos la librería “transport” y luego modificamos la función getAllImages para ver en pantalla las cards de los personajes.



```
app > layers > services > services.py > ...
1  # capa de servicio/lógica de negocio
2
3  from ..persistence import repositories
4  from ..utilities import translator
5  from ..transport import transport
6  from django.contrib.auth import get_user
7
```



```
app > layers > services > services.py > ...
6  from django.contrib.auth import get_user
7
8
9  def getAllImages(input=None):
10     # obtiene un listado de datos "crudos" desde la API, usando a transport.py.
11     json_collection = transport.getAllImages(input)
12
13     # recorre cada dato crudo de la colección anterior, lo convierte en una Card y lo agrega a images.
14     images = []
15     for i in range(len(json_collection)):
16         image = translator.fromRequestIntoCard(json_collection[i])
17         images.append(image)
18
19     return images
```

En el archivo views.py modificamos la función home para que mostrara las imágenes de los personajes. De esta forma llamamos la función getAllImages desde services.py.

```
> _pycache_ 16
> config 17 def home(request):
> layers 18     images = services.getAllImages()
> migrations 19
> templates 20     favourite_list = []
> init.py 21     return render(request, 'home.html', {'images': images, 'favourite_list': favourite_list})
22
```

Para conseguir el respectivo color de los bordes agregamos código al archivo home.html y modificamos los valores para que los bordes y el estado del personaje se mostrarán correctamente. success = verde / vivo

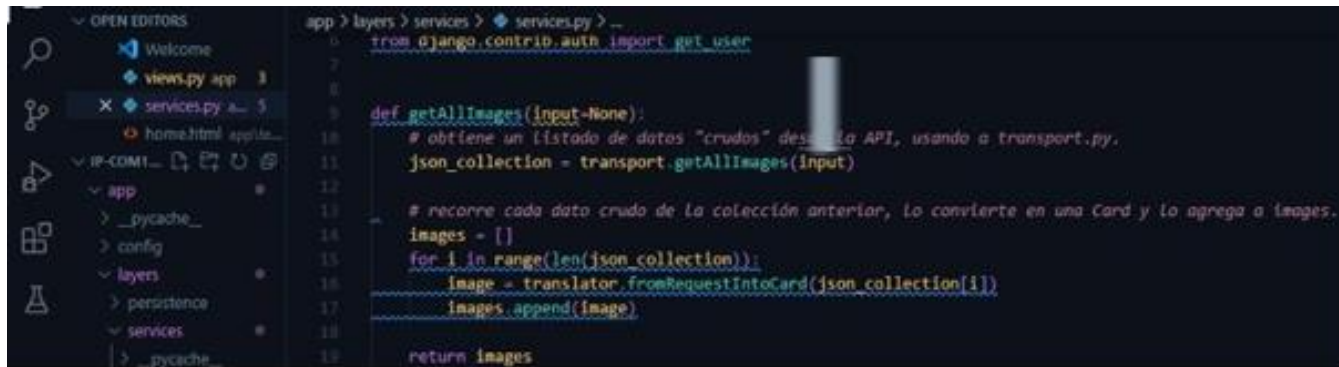
warning = amarillo / desconocido

danger = rojo / muerto

```
40 <div class="card mb-3 ms-5"
41     {% if img.status == 'Alive' %} border border -1 border-success
42     {% elif img.status == 'Dead' %} border border -1 border-danger
43     {% else %} border border -1 border-warning
44     {% endif %}
45     style="max-width: 540px;">

53 <strong>
54     {% if img.status == 'Alive' %} ● {{ img.status }}
55     {% elif img.status == 'Dead' %} ● {{ img.status }}
56     {% else %} ● {{ img.status }}
57     {% endif %}
58 </strong>
```

EDICIONES EXTRAS: Arreglamos el buscador modificando la función getAllImages del archivo services.py agregando la variable (input), la misma muestra las imágenes con la palabra ingresada en el input del buscador. en el archivo views.py arreglamos la función search, para que el buscador devuelva las cards que se están buscando.



```
app > layers > services > services.py > ...
1 from django.contrib.auth import get_user
2
3
4
5 def getAllImages(input=None):
6     # obtiene un listado de datos "crudos" desde la API, usando a transport.py.
7     json_collection = transport.getAllImages(input)
8
9     # recorre cada dato crudo de la colección anterior, lo convierte en una Card y lo agrega a images.
10    images = []
11    for i in range(len(json_collection)):
12        image = translator.fromRequestIntoCard(json_collection[i])
13        images.append(image)
14
15    return images
```



```
26 def search(request):
27     search_msg = request.POST.get('query', '')
28     # si el texto ingresado no es vacío, trae las imágenes y favoritos desde services.py,
29
30     if (search_msg != ''):
31         images = services.getAllImages(search_msg)
32         favourite_list = []
33
34         return render(request, 'home.html', {'images': images, 'favourite_list': favourite_list})
35     else:
36         return redirect('home')
```

Dentro del archivo views.py llamamos a la función “logout” dentro de la función “exit”, de esta forma redirigimos al usuario al inicio “index.html”.



```
58
59 @login_required
60 def exit(request):
61     logout(request)
62     return render(request, 'index.html')
63
```

Conclusiones:

El trabajo se nos ha hecho relativamente un poco difícil, ya que nunca hemos tenido conocimiento sobre GIT, Django, y esto sumado a la poca disponibilidad de tiempo hizo que las cosas se nos complicaran un poco, algunas dificultades que afrontamos fueron:

Conocer los comandos de GIT. La comunicación a la hora de resolver los puntos y ciertos problemas técnicos con

las PCs, lo que hizo que eventualmente un compañero se tuviera que dar de baja del grupo por no poder participar ya que su equipo no funcionaba.

A la hora de reconocer el código y saber que cambios aplicar, eventualmente tuvimos que volver a reeditar todo porque habíamos hecho un desastre.

Fuera de todas las complicaciones, fue un desafío muy divertido y desafiante para nosotros, ya que tuvimos que consultar con diversas fuentes como videos y compañeros, pero finalmente pudimos comprender y aprender las habilidades necesarias para resolver el TP. En conclusión, creemos que estas actividades fueron pilares y bases importantes para nuestro aprendizaje en el futuro a pesar de la dificultad experimentada