POKEDEX

Materia: Introducción a la Programación

Comisión: 02

Miembros: Gabriel Galarza – Lucas Alderete

Introducción:

En este trabajo práctico haremos una página web de temática de Pokémon usando Django.

El propósito del sitio web es el de mostrar una serie de imágenes que mostraran datos sobre los Pokémones (nombre, imagen, tipo, etc.), como si fuera una Pokédex, pero más simplificada. Todo esto con el objetivo de aprender acerca del funcionamiento de entornos de trabajo virtuales y el desarrollo de sitios web.

Para hacer este TP se tendrá a mano una página web casi terminada, por lo que lo único que quedaría por hacer seria completar los módulos que faltan por terminar.

Desarrollo del sitio:

- ***services.py:*** Este módulo es el más importante, pues es el que posee la función "*getAllImages()",* la cual se tiene que encargar de recibir las imágenes y convertirlas de tal modo que puedan visualizarse en el sitio.

```python
# función que devuelve un listado de cards. Cada card representa una imagen de la API de HP.
def getAllImages():
    # debe ejecutar los siguientes pasos:
    # 1) traer un listado de imágenes crudas desde la API (ver transport.py)
    # 2) convertir cada img. en una card.
    # 3) añadirlas a un nuevo listado que, finalmente, se retornará con todas las card encontradas.
    # ATENCIÓN: contemplar que los nombres alternativos, para cada personaje, deben elegirse al azar. Si no existen nombres alternativos, debe mostrar un mensaje adecuado.
    pass
```

Una forma de completar esta función seria, primero que nada, crear una lista y para ello usaremos una función ya terminada que crea dicha lista de imágenes.

```python
def getAllImages():
    json_collection = []
    for id in range(1, 30):
        response = requests.get(config.STUDENTS_REST_API_URL + str(id))

        # si la búsqueda no arroja resultados, entonces retornamos una lista vacía de elementos.
        if not response.ok:
            print(f"[transport.py]: error al obtener datos para el id {id}")
            continue

        raw_data = response.json()

        if 'detail' in raw_data and raw_data['detail'] == 'Not found.':
            print(f"[transport.py]: Pokémon con id {id} no encontrado.")
            continue

        json_collection.append(raw_data)

    return json_collection
```

Importamos esta funcion de su respectivo modulo y lo guardaremos en una variable, para ello escribiremos el codigo "*raw_images = transport.getAllImages*" (No confundir con la funcion con la que estamos trabajando), para despues crear otra variable que contenga una lista vacia, que posteriormente llenaremos con las imágenes procesadas.

Para procesar las imágenes usamos un ciclo *"for"* con un iterable *"image",* que recorrera la lista. Durante el recorrido se usara el comando *"append"* para llenar la lista, asi como tambien un diccionario, el cual tiene como utilidad crear claves que apuntan a valores, en este caso seran datos tales como el nombre (El cual posee una cadena *"capitalize()"* para que el nombre comienze con letra mayuscula), la altura, el peso, el tipo, etc.

La cosa cambia cuando nos toca trabajar con las claves *"images", "type"* y *"types"*. La clave *"image"* se asegura de buscar en los diccionarios las respectivas imágenes que se verán en la página, mientras que la clave *"type"* se encarga de buscar los tipos y escribirlos en las cards y por ultimo, la clave *"types"*, la más compleja, se encarga de acceder a la lista de todos los tipos gracias a un ciclo *"for".*

La función finalmente termina con la funcion retornando la lista con las imágenes procesadas.

```python
# función que devuelve un listado de cards. Cada card representa una imagen de la API de Pokemon
def getAllImages():
    raw_images = transport.getAllImages()
    image_cards = []
    for image in raw_images:
        image_cards.append({
            "id": image["id"],
            "name": image["name"].capitalize(),
            "height": image["height"],
            "weight": image["weight"],
            "base": image["base_experience"],
            "image": image["sprites"]["other"]["official-artwork"]["front_default"],
            "type": image["types"][0]["type"]["name"],
            "types": [t["type"]["name"] for t in image.get("types", [])]
        })
    return image_cards
```

Otra forma de hacerlo seria simplificarla con una unica lista, todo gracias a una cadena *"translator"*, la cual convierte las imágenes en una card, esto con la ayuda de una funcion ya

realizada llamada *“fromRequestIntoCard”* y un iterador *“obj”* que se usara en un ciclo *“for”* que recorrera la lista de imágenes crudas.

```
# función que devuelve un listado de cards. Cada card representa una imagen de la API de Pokemon
def getAllImages():
    raw_images = transport.getAllImages()
    images = [translator.fromRequestIntoCard(obj) for obj in raw_images]
    return images
```

- ***views.py:*** En este módulo debemos completar la función “home(request)”, la cual tiene como objetivo crear dos listas de imágenes para posteriormente dibujarlas en el respectivo sitio web.

```
# esta función obtiene 2 listados: uno de las imágenes de la API y otro de favoritos, ambos en formato Card, y los dibuja en el template 'home.html'.
def home(request):
    images = []
    favourite_list = []

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

Completar esta función es bastante sencillo, ya que solo debemos importar la función anteriormente creada, para que al final envié las imágenes a un módulo *“html”* para que dibuje

```
# esta función obtiene 2 listados: uno de las imágenes de la API y otro de favoritos, ambos en formato Card, y los dibuja en el template 'home.html'.
def home(request):
    images = services.getAllImages()
    favourite_list = []
    return render(request, 'home.html', { 'images': images, 'favourite_list': favourite_list })
```

- **home.html:** Este módulo es el encargado de dibujar las imágenes y mostrar los respectivos datos en la página web. El modulo ya está casi completo, por lo que lo único que queda por hacer es modificar el modulo para que las imágenes mostradas tengas bordes que representen su tipo.
  Para hacerlo solo tenemos que usar los *“if”*, para que el modulo reconozca el tipo de Pokémon y dibuje el borde del card con su respectivo color. Por ejemplo, si el Pokémon es de tipo fuego, el borde será color rojo, esto se logrará gracias a una línea llamada *“border”* acompañado de otro comando que coloreara con el color correspondiente, en este caso *“danger”.*

La cosa cambia cuando el Pokémon no pertenece a ninguno de los tres tipos principales (fuego, agua, pasto), en cuyo caso el borde tiene que ser naranja, cosa que se consigue gracias a lo mencionado anteriormente, con una línea *“border-warning”.*

```
<!-- evaluar si la imagen pertenece al tipo fuego, agua o planta -->
<div class="card
    {% if 'fire' in img.types %}border-danger
    {% elif 'water' in img.types %}border-primary
    {% elif 'grass' in img.types %}border-success
    {% else %}border-warning
    {% endif %}
    mb-3 ms-5" style="max-width: 540px;">
```

Opcionales:

Lo siguiente no son elementos que debíamos completar obligatoriamente, pero que podíamos hacer para que la pagina este más pulida.

- **Buscador por nombre:** Como su nombre lo indica, esta herramienta se encargara de permitirle al usuario ingresar el nombre de un Pokémon y así poder ver únicamente las imágenes del susodicho.
  Para ello volveremos a los módulos *“views.py”* y *“services.py”*, donde primero añadiremos a la función *“filterByCharacter”,* donde con un *“if”* y una cadena *“name.lower()”* para que la función lea lo escrito por el usuario y con la ayuda de *“in card.name.lower()”* para que lea los nombres en las cards y devuelva lo encontrado.

```
def filterByCharacter(name):
    filtered_cards = []

    for card in getAllImages():
        # debe verificar si el name está contenido en el nombre de la card, antes de agregarlo al listado de filte
        if name.lower() in card.name.lower():
            filtered_cards.append(card)

    return filtered_cards
```

```
if (name != ''):
    images = services.filterByCharacter(name)
    favourite_list = []

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

Dificultades:

El principal problema que tuvimos fue el poco conocimiento que teníamos sobre Django, nos tocó investigar bastante para aprender a usar apropiadamente elementos como el lenguaje *“html”* y los diccionarios de Django.

Otro problema que tuvimos a la hora de desarrollar el TP fue que por alguna razón, hubo problemas con el API, lo que hacía que la página no funcionara, pero se logró solucionar gracias al uso de un VPN.

Además de eso, tuvimos el problema de que a la hora de trabajar con la función *“getAllImages”*, ya que en un principio se escribió una línea que solo pedía un *“type”* dentro del diccionario, lo que no funciono debido a la complejidad del funcionamiento de los types, para solucionar eso se modificó la misma para que esta pueda recorrer la lista de tipos y que se añadan al respectivo Pokemon.

Conclusión:

El desarrollo de este TP nos permitió entender mejor el funcionamiento de los frameworks, así como también el desarrollo de páginas web, conocimientos que nos servirán a futuro para mejorar nuestras habilidades de programador.