Trabajo Práctico

Introducción a la Programación

Integrantes:

Reynaga, Leandro

Sosa, Jonathan

Veizaga, Federico

Comisión 3

Profesores:

Bressky, Daniel

Fassio, Esteban

Velazquez, Santiago

Introducción

Se desea implementar una aplicación web que permita buscar POKEMONES, éstos se verán en distintas cards que mostrarán la imagen del Pokémon junto con su información básica.

La aplicación ya viene desarrollada casi en su totalidad utilizando Django. Lo que debemos hacer nosotros es completar algunas funciones del código dado en lenguaje Python y algo de html, para que se pueda ver el resultado final. La información utilizada será proporcionada mediante una API y luego renderizada por el framework Django.

Desarrollo

En el archivo *services.py* se realizaron las siguientes modificaciones:

# función que devuelve un listado de cards. Cada card representa una imagen de la API de Pokemon

def getAllImages():

    # debe ejecutar los siguientes pasos:

    lista\_imagenes\_crudas=transport.getAllImages()  # 1) traer un listado de imágenes crudas desde la API (ver transport.py)

    lista\_cards=[]

    for dato in lista\_imagenes\_crudas:    # 2) convertir cada img. en una card.

        card=translator.fromRequestIntoCard(dato)

        lista\_cards.append(card)

    # 3) añadirlas a un nuevo listado que, finalmente, se retornará con todas las card encontradas.

    return lista\_cards

Se modificó la función getAllImages(), le agregamos la variable lista\_imagenes\_crudas que trae un listado de imágenes crudas de la API usando la función getAllImages(),que está en el archivo *transport.py*. Creamos la variable lista\_cards=[] para guardar todas las cards y devolverla en la función. Recorrimos lista\_imagenes\_crudas para obtener cada elemento de la misma y convertimos cada imagen obtenida en una card con la función fromRequestIntoCard(dato)del archivo *translator.py*, una vez trasformado lo agregamos a la lista lista\_cards.

  card=translator.fromRequestIntoCard(dato)

        tipos=[]

        tipos\_cartas= card.types

        for tipo in tipos\_cartas:

          tipos.append(get\_type\_icon\_url\_by\_name(tipo))

        card.types\_imgs=tipos

        lista\_cards.append(card)

A la función anterior le agregamos la variable tipos=[] donde se guardara el tipo de pokémon en la información de la card. tipos\_cartas=card.types obtiene una lista de los tipos de pokemón la cual recorreremos para guardar en la card el tipo de pokemón correspondiente.

Este código agregado se encuentra luego de transformar la card y antes de agregarla a la lista de cardas ya que pensamos que ya contenía la información del tipo de pokemón cuando se transformaba pero no era así.

#A esta altura con los cambios correspondientes en views.py y home.html el programa ya funcionaba.

Se realizan los buscadores:

def filterByCharacter(name):

    filtered\_cards = []

    name=name.lower()

    for card in getAllImages():

        if name in card.name:  # debe verificar si el name está contenido en el nombre de la card, antes de agregarlo al listado de filtered\_cards.

            filtered\_cards.append(card)

    return filtered\_cards

En la función filterByCharacter(name) agregamos la variable name=name.lower() para asegurar que el nombre ingresado se pueda comparar con el de la información de la card que está en minúsculas. Luego recorremos las imágenes obtenidas de la función getAllImages() (creada previamente). Para buscar por el nombre debemos traer sólo ésa información de la card por eso escribimos card.name y lo comparamos con el nombre del pokémon que buscamos, una vez encontrado lo agregamos a la lista filtered\_cards que se encontraba vacía y luego la devuelve.

def filterByType(type\_filter):

    filtered\_cards = []

    lista\_imagenes=getAllImages()

    type\_filter=type\_filter.lower()

    for card in lista\_imagenes:

        if type\_filter in card.types:# debe verificar si la casa de la card coincide con la recibida por parámetro. Si es así, se añade al listado de filtered\_cards.

            filtered\_cards.append(card)

    return filtered\_cards

        return redirect('home')

En la función filterByType(type\_filter) agregamos la variable lista\_imagenes=getAllImages() que trae todas las cards de los pokemón usando la función anteriormente creada, agregamos la variable type\_filter=type\_filter.lower() para asegurar que el tipo ingresado se pueda comparar con el de la información de la card que está en minúsculas. Luego recorremos las imágenes para buscar las coincidencias. Como sólo queremos el tipo de pokémon de la card lo llamamos así card.types y así podemos compararlo con el tipo ingresado. En la variable filtered\_cards se guardan los pokémon ya clasificados por su tipo y es lo que devuelve a función.

En el archivo *views.py* se realizaron las siguientes modificaciones:

# 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):

    lista\_cards=services.getAllImages()

    images = []

    favourite\_list = []

    for card in lista\_cards:

        images.append(card)

    return render(request, 'home.html', { 'images': images, 'favourite\_list': favourite\_list })

Se modificó la función home(request) le agregamos la variable lista\_cards=services.getAllImages() que tiene un listado de imágenes en formato card usando la función getAllImages() que está en el archivo services.py. Luego recorrimos ésta lista para obtener cada elemento de la misma y lo agregamos a la lista images que estaba vacía. De ésta forma ya se pueden ver las imágenes en el navegador.

#A esta altura con los cambios correspondientes en services.py y home.html el programa ya funcionaba.

Se realizan los buscadores:

# función utilizada en el buscador.

def search(request):

    name = request.POST.get('query', '')

    # si el usuario ingresó algo en el buscador, se deben filtrar las imágenes por dicho ingreso.

    if name != '':

        images = services.filterByCharacter(name)

        favourite\_list = []

        return render(request, 'home.html', { 'images': images, 'favourite\_list': favourite\_list })

    else:

        return redirect('home')

En la funcion search(request), que es la correspondiente al buscador por nombre del pokémon, la variable images=services.filterByCharacter(name) usa la función filterByCharacter(name) que está en el archivo services.py la cual trae las imágenes ya filtradas por el nombre que queremos buscar.

# función utilizada para filtrar por el tipo del Pokemon

def filter\_by\_type(request):

    type = request.POST.get('type', '')

    if type != '':

        images = services.filterByType(type) # debe traer un listado filtrado de imágenes, segun si es o contiene ese tipo.

        favourite\_list = []

        return render(request, 'home.html', { 'images': images, 'favourite\_list': favourite\_list })

    else:

        return redirect('home')

En la funcion filter\_by\_type(request), que es la correspondiente al buscador por tipo de pokémon, la variable images=services.filterByType(type) usa la función filterByType(type) que está en el archivo services.py la cual trae las imágenes ya filtradas por el tipo que queremos buscar.

En el archivo *home.html* se realizaron las siguientes modificaciones:

div class="card mb-3 ms-6 {% if 'fire' in img.types %}border-danger{% elif 'grass' in img.types %}border-success{% elif 'water' in img.types %}border-primary{% else %}border-warning{% endif %} " style="max-width: 600px; ">

A partir de la información del tipo de pokémon agregada a las cards previamente pudimos filtrar las imágenes con un condicional y se le asignó el color correspondiente a cada card. También se ajustó el tamaño(ancho) de la card de ms-5 a ms-6 para que se vean todas ordenadas y del mismo tamaño, ya que antes algunas cambiaban de tamaño según su información y no quedaba prolijo.

En esta parte tuvimos que ver información sobre cómo hacer un condicional en éste lenguaje que no es Python, pero fue muy similar a lo que sabíamos.

#A esta altura con los cambios correspondientes en services.py y views.py el programa ya realizaba lo deseado.

<h3 class="card-title {% if 'fire' in img.types %}text-danger{% elif 'grass' in img.types %}text-success{% elif 'water' in img.types %}text-primary{% else %}text-warning{% endif %}">{{ img.name }} #{{ img.id }} <img src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/items/poke-ball.png" alt="Master Ball"></h3>

En esta parte le agregamos color a los nombres y ids de los pokémones para que coincidan con el color del borde de su card.

Conclusión

Aprendimos nuevas tecnologías de forma repentina y en poco tiempo lo que nos representó un desafío porque no sabíamos por dónde empezar, pero luego con ayuda de internet y consultas a compañeros y profesores pudimos adaptarnos.