Informe de Pokedex

Este trabajo consiste en desarrollar funciones destinadas a resolver una serie de tareas específicas en un proyecto práctico. El objetivo fue implementar soluciones eficientes y organizadas, intentando evitar complejidades innecesarias. En este documento se presenta el código de cada función junto con una breve explicación de su propósito, el desarrollo y las decisiones que se tomaron para resolverlas. La idea principal fue resolver los problemas planteados mediante un enfoque lógico, priorizando la claridad del código y las funciones dadas

1)
```
if (name != ''):
    images = getAllImagesAndFavouriteList(request)
    for img in images:
        if name.lower() in img.name.lower():
            imagenesf.append(img)
else:
            imagenesf = images


favourite_list = []


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

este código sirve para cuando el usuario ingrese algo en el buscador se filtren las cards del nombre ingresado por el usuario. Compara si el nombre de la imagen contiene lo que el usuario ingresó sin importar mayúsculas o minúsculas. Si el usuario no ingresa nada y presiona buscar saltarán todas las imágenes

2)
```
json_collection = transport.getAllImages()
images = [] # lista que contendrá las cards transformadas a partir de las imágenes crudas.
for object in json_collection:
```

card = translator.fromRequestIntoCard(object)

types_aux = []#se crea una lista auxiliar para almacenar los iconos de los tipos de cada card

for t in card.types:

types_aux.append(get_type_icon_url_by_name(t))

card.types_imgs = types_aux

images.append(card)

esto hace que el json collection sea transportado con imagenes crudas, relativamente fácil de deducir para el pensamiento lógico después de entender que hace cada función. Translator.formRequestintoCard(object) usa información de la API para transformarlo en Cards. Se recorre cada tipo de Card y se consigue el icono correspondiente a cada tipo de pokemon usando su nombre

En general este punto se nos hizo bastante complicado de entender porque fue lo primero que empezamos a tocar de este proyecto, pedimos ayuda a las profesoras y nos supieron guiar bastantes

3)

{% if "fire" in img.types %}

<div class="card border-danger border-3 mb-3 ms-5" style="max-width:

⬆️ hace que las cards con pokemon tipo fuego tengan un borde rojo(danger)

540px; ">

{% elif "water" in img.types %}

<div class="card border-primary border-3 mb-3 ms-5" style="max-width:

⬆️ hace que las cards con pokemon tipo agua tengan un borde azul(primary)

540px; ">

{% elif "grass" in img.types %}

<div class="card border-success border-3 mb-3 ms-5" style="max-width:

⬆️ hace que las cards con pokemon tipo planta tenga un borde verde(success)

540px; ">

```
        {% else %}
          <div class="card border-secondary border-3 mb-3 ms-5" style="max-width:
          ⬆️ hace que las cards con cualquier otro tipo de pokemon tenga un borde
          gris(secondary)
 540px;">
        {% endif %}
```

No hubo muchas complicaciones en este código ya que vimos en el video de Django como se usan los condicionales y estos son muy parecidos a los de python

4)

```
{% if messages %} #verifica almacenados
        <div class="mt-3"></div>
          {% for message in messages %}
            <div class="alert alert-danger" role="alert">
              {{ message }}
            </div>
          {% endfor %}
        </div>
      {% endif %}
    </form>
</div>
```

Este código lo agregamos al final de Login.html, fue el código que más nos costó hacer por no conocer mucho las funcionalidades inicios de sesión, lo que hace es informar al usuario sobre lo que pasó después de lo ingresado por este, inicio de sesión y contraseña y si hubo algún error

5)

```
def home(request):
    images = getAllImagesAndFavouriteList(request)
```

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

son dos listas, una con imágenes de la API y la otra es de favoritos en formatos de card, esto hace que se dibujen en el témplate del home.html

6)

```
def getAllImagesAndFavouriteList(request):
    images = services.getAllImages()
    return images
```

llama a la función de getAllImages() para tener todas las imágenes de la API

7) Favoritos esta en el header.html y pones solo esto "{% url 'get-favourites' %}"

Sirve para marcar las rutas(URL), lo modificamos en urls.py para poder empezar a llamar a los favoritos

```
def saveFavourite(request):
     fav = translator.fromTemplateIntoCard(request)
     fav.user = get_user(request) # le asignamos el usuario correspondiente.
    existing = repositories.get_all_favourites(fav.user) # buscamos si ya existe un
favorito con el mismo ID.
    if any(f['name'] == fav.name for f in existing):
        return None
    return repositories.save_favourite(fav) # lo guardamos en la BD.
```

Lo que hace esta funcion es que convierte al request en una Card, asigna al usuario yu verifica si ya existe un favorito con el mismo nombre, si ya existe no lo agrega pero si no existe lo guarda en la base de datos

```
def getAllFavourites(request):
```

```
    if not request.user.is_authenticated:
        return []
    else:
        user = get_user(request)
```

favourite_list = repositories.get_all_favourites(user) # buscamos desde el repositories.py TODOS Los favoritos del usuario (variable 'user').

mapped_favourites = []

for favourite in favourite_list:

card = translator.fromFavouriteIntoCard(favourite) # desde un favorito obtenemos la Card

mapped_favourites.append(card)

return mapped_favourites

Sirve para obtener todos los pokemon favoritos del usuario autenticado y devolverlo en formato de Card

```
@login_required def getAllFavouritesByUser(request):
  favourite_list = services.getAllFavourites(request)
  return render(request, "favourites.html", {"favourite_list": favourite_list})
@login_required def saveFavourite(request):
    if request.method == 'POST':
        services.saveFavourite(request)
        return redirect('home')
@login_required def deleteFavourite(request):
    if request.method == 'POST':
        services.deleteFavourite(request)
```

return redirect('get-favourites')

Este codigo sirve para manejar acciones relacionadas con la sesion y los favoritos del usuario, tanto agregar como eliminar pokemon de su lista de favoritos y redirigir al usuario a la pagina principal

def home(request):

images = getAllImagesAndFavouriteList(request)

favourite_list = services.getAllFavourites(request

favourite_list_ids = [f["id"] for f in favourite_list]

Aca obtenemos el listado de favoritos del usuario logueado y obtenemos los ids de los favoritos para poder compararlos con las imágenes.

En conclusión, el proyecto podría haber sido un poco más fácil de lo que nos costó, ya que si hubiéramos tenido un poco más de tiempo para poder entender mucho mejor los códigos y así poder hacer un mejor desarrollo del mismo, pero se logró hacer que funcione y metimos opcionales de búsqueda, tipos de pokemon y su respectivo color de la card e inicio de sesión del usuario