

# Informe del Trabajo Práctico de Introducción a la Programación



## Introducción

En el siguiente informe se detalla la resolución del trabajo práctico de programación del primer cuatrimestre del año 2025. La consigna de este trabajo es realizar una aplicación web que funcione como un buscador de Pokémons que muestre en pantalla diferentes tarjetas (o cartas) con su respectiva imagen principal y con algunos detalles específicos de cada personaje.

En las tarjetas mencionadas se visualizan las siguientes características:

- Tipo de Pokémon
- Altura
- Peso
- Nivel de experiencia base

Para que esta aplicación funcione, debe hacer uso de una API de Pokémons ya creada anteriormente llamada PokéAPI, la cual brinda todos los datos necesarios en forma de códigos para que, al usarlo en nuestros archivos y módulos, se pueda trabajar sobre ella y ejecutar sin problemas.

Las modificaciones de los archivos y códigos se harán a partir de un repositorio obtenido en GitHub, en el cual debemos estar logueados para poder trabajar sobre el mismo.

## Desarrollo

Comenzamos con la modificación que se piden en las consignas obligatorias, las cuales piden modificar los módulos “views.py”, “services.py” y “home.html”.

Dentro del módulo “views.py” modificamos algunas de las funciones que se muestran a continuación, precisamente en la función home:

```
views.py X
app > views.py > ...
1 # capa de vista/presentación
2
3 from django.shortcuts import redirect, render
4 from .layers.services import services
5 from django.contrib.auth.decorators import login_required
6 from django.contrib.auth import logout
7
8 def index_page(request):
9     return render(request, 'index.html')
10
11 # esta función obtiene 2 listados: uno de las imágenes de la API y otro de favoritos, ambos en formato Card, y los dibuja
12 # en el template 'home.html'.
13 def home(request):
14     images = services.getAllImages() #En esta línea se obtienen la lista de imagenes transformadas en card y las muestra
15     # por pantalla.
16     favourite_list = []
17
18     return render(request, 'home.html', { 'images': images, 'favourite_list': favourite_list })
19
20 # función utilizada en el buscador.
21 def search(request):
22     name = request.POST.get('query', '')
23
24     # si el usuario ingresó algo en el buscador, se deben filtrar las imágenes por dicho ingreso.
25     if (name != ''):
26         images = services.filterByCharacter(name) #en esta línea de código se obtiene la card filtrada para mostrarla en
27         # pantalla.
28         favourite_list = []
```

En esta función modificamos la variable *images*. Esta variable trae una lista de imágenes transformadas en cards y estas son mostradas en la pantalla. Para realizar esta acción, utiliza la función *getAllImages* traída desde el módulo “service.py”

A su vez, también tenemos que modificar el módulo “service.py” para que ambas funciones trabajen en conjunto:

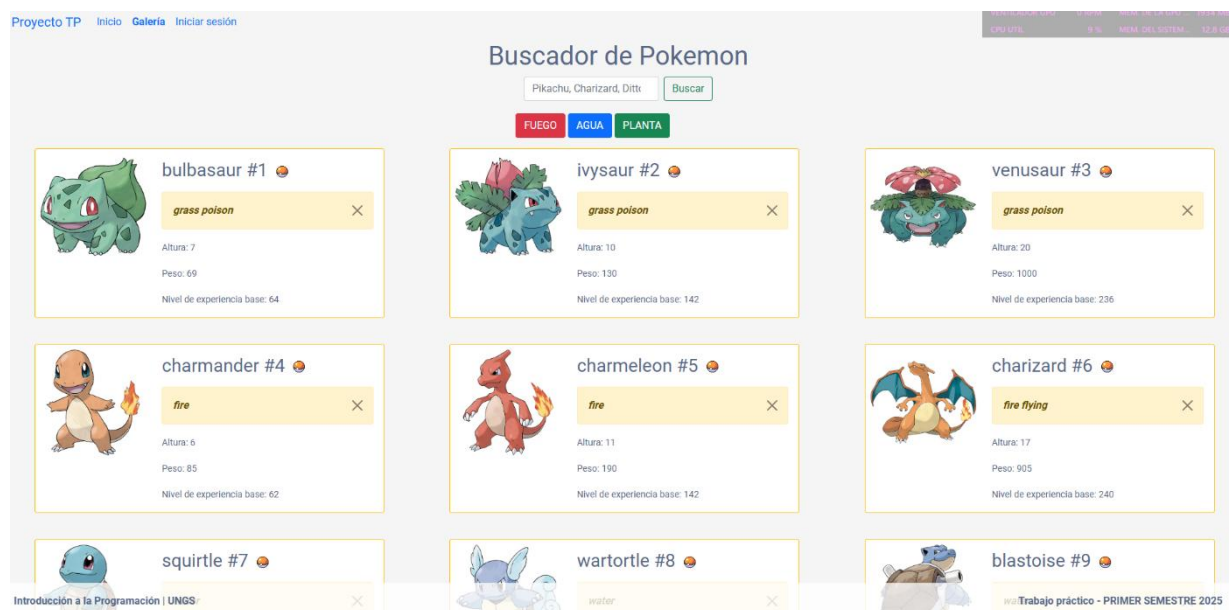
```
views.py services.py X
app > layers > services > services.py > ...
1 # capa de servicio/lógica de negocio
2
3 from ..transport import transport
4 from ...config import config
5 from ..persistence import repositories
6 from ..utilities import translator
7 from django.contrib.auth import get_user
8
9 # función que devuelve un listado de cards. Cada card representa una imagen de la
10 # API de Pokemon
11 def getAllImages():
12     # debe ejecutar los siguientes pasos:
13     # 1) traer un listado de imágenes crudas desde la API (ver transport.py)
14     # 2) convertir cada img. en una card.
15     # 3) añadirlas a un nuevo listado que, finalmente, se retornará con todas las
16     # card encontradas.
17     imagenes_crudas = transport.getAllImages() # obtiene las imágenes crudas desde
18     # la API
19     lista_cartas=[]
20     for img in imagenes_crudas:
21         # debe convertir cada imagen cruda en una Card utilizando el translator.py
22         card = translator.fromRequestIntoCard(img)
23         lista_cartas.append(card)
24     return lista_cartas # retorna un listado de cards
```

La función mostrada es la encargada de crear una lista de cards que luego son usadas en la función home. Estas cards primero son traídas como imágenes crudas desde el módulo “transport.py” y luego con el uso de un ciclo (FOR) y la capa “translator.py”, se convierte cada imagen cruda en una card y las agrega a la lista de cards.

El módulo “transport.py” cumple la importante función de comunicarse con la API que nos brinda los datos específicos de los Pokemons. Mientras que el módulo “translator.py” es la encargada de convertir los datos obtenidos a un formato específico, en este caso el mapeo de las cards.

Hasta ahora, las funciones detalladas anteriormente, nos sirven para mostrar la lista de cartas en la pagina web remota, pero las mismas aun se muestran con un formato no terminado, como son el color de los bordes según el tipo de Pokémon (“grass” o pasto, “water” o agua y “fire” o fuego).

La siguiente imagen muestra las cards con los bordes sin el cambio de los colores:



Para aplicar los cambios de colores a los bordes, debemos modificar el módulo “home.html”, para que evalúe mediante comparaciones, si la imagen del Pokémon pertenece a uno de los tipos señalados anteriormente.

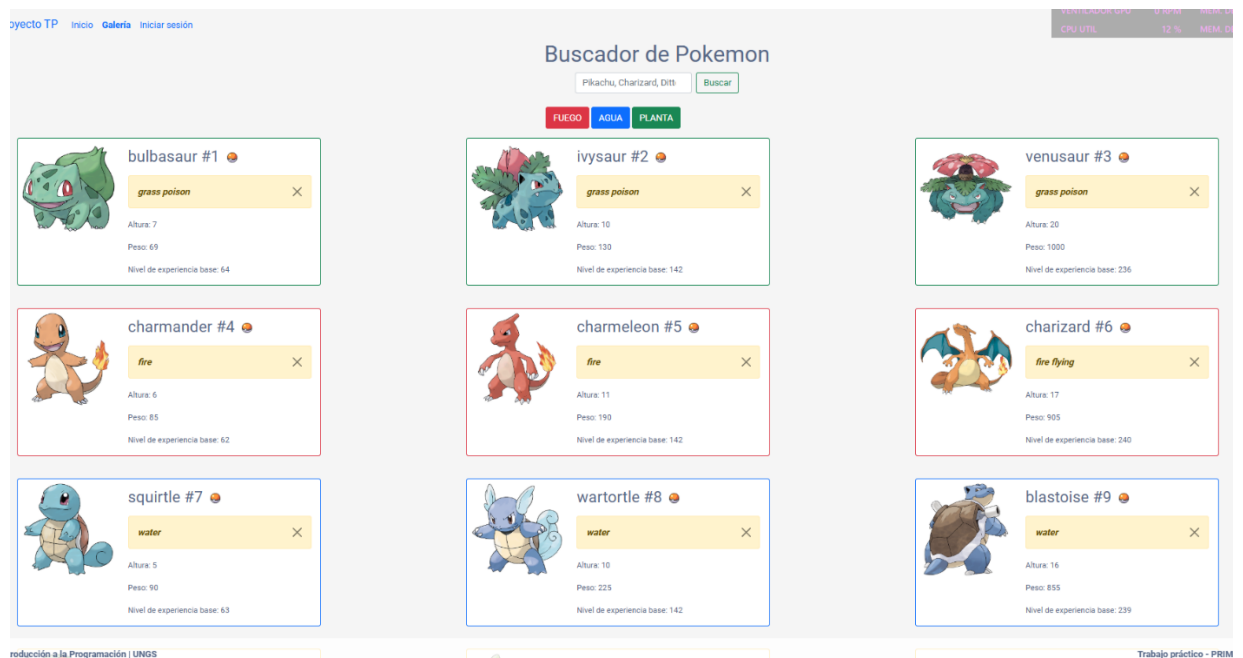
En las líneas de códigos que se marcan en la siguiente imagen, son las encargadas de darle el formato a las cards para que cambien el color del borde de cada una dependiendo del tipo de Pokémon. Esto lo hace a partir de una estructura condicional que se encarga de realizar las respectivas comparaciones mencionadas.

```

views.py services.py transport.py home.html X
app > templates > home.html > main > div.row.col-1.row.col-md-3.g-4 > div.col > div.card.border-success.mb-3.ms-5 > div.row.g-0 > div.card.border-primary.mb-3.ms-5 > div.row.g-0 > div.card.border-danger.mb-3.ms-5 > div.row.g-0 > 8
2 <main>
32 </div>
33
34 <div class="row row-cols-1 row-cols-md-3 g-4">
35   {% if images.length == 0 %}
36     <div class="text-center">La búsqueda no arrojó resultados...</div>
37   {% else %} {% for img in images %}
38     <div class="col">
39       <!-- evaluar si la imagen pertenece al tipo fuego, agua o planta -->
40       <!-- En las líneas siguientes colocamos la estructura condicional para cambiar el color del borde de cada carta dependiendo del tipo de pokemon -->
41       {% if 'grass' in img.types %}
42         <div class="card border-success mb-3 ms-5" style="max-width: 540px;">
43           <div class="row g-0">
44             <div class="col-md-4">
45               
46             </div>
47
48             {% elif 'water' in img.types %}
49               <div class="card border-primary mb-3 ms-5" style="max-width: 540px;">
50                 <div class="row g-0">
51                   <div class="col-md-4">
52                     
53                   </div>
54
55                   {% elif 'fire' in img.types %}
56                     <div class="card border-danger mb-3 ms-5" style="max-width: 540px;">
57                       <div class="row g-0">
58                         <div class="col-md-4">
59                           
60                         </div>
61
62                       {% else %}
63                         <div class="card border-warning mb-3 ms-5" style="max-width: 540px;">
64                           <div class="row g-0">
65                             <div class="col-md-4">
66                               
67                             </div>
68
69                         </div>
70
71                         <div class="col-md-8">
72                           <div class="card-body">
73                             <div class="card-title">{{ img.name }} #{{ img.id }} </div>
74                             <p class="card-text">
75                               <div class="alert alert-warning alert-dismissible fade show" role="alert">
76                                 {% for poketype in img.types %}
77                                   <strong><em>{{ poketype }}</em></strong>
78                                 </div>
79                               <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>

```

Una vez realizado el cambio y ejecutado el programa, las cartas van a adquirir el nuevo formato en sus bordes:



Hasta este punto se mostraron las resoluciones de las consignas obligatorias, con funciones que trabajan en conjunto entre diferentes capas.

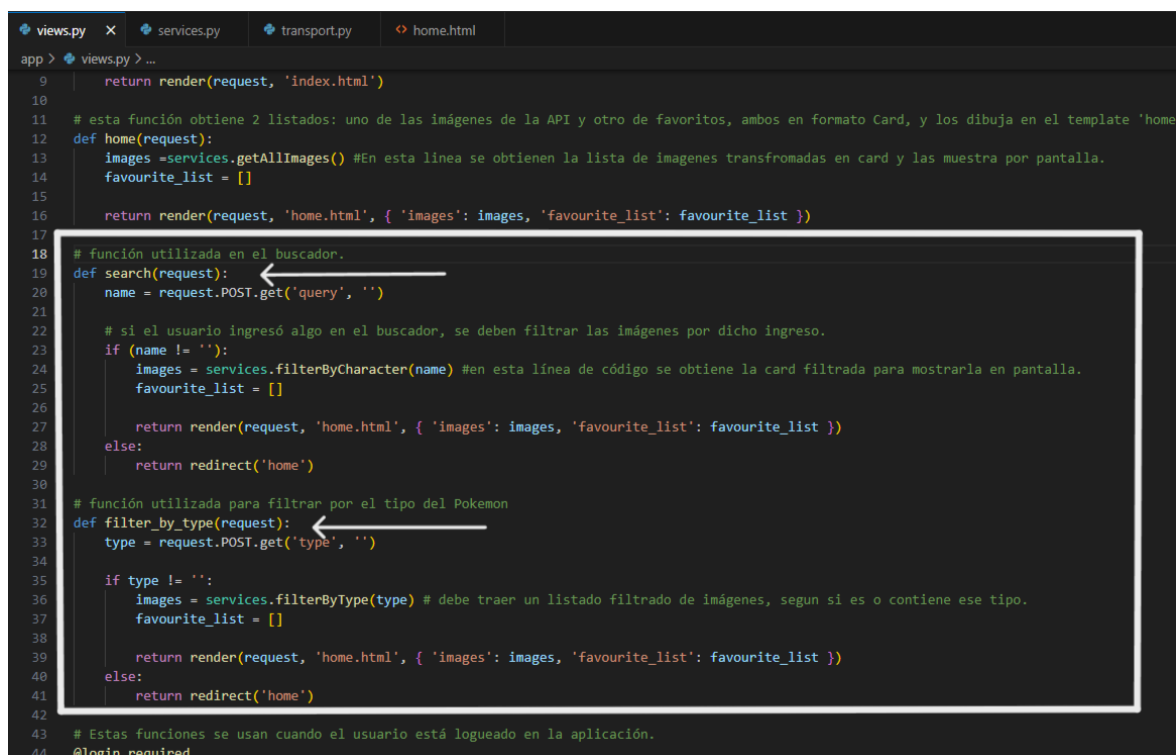
Viera, Sergio | Ramos, Alex

A continuación, también procederemos a mostrar la resolución de algunas de las consignas opcionales que brinda el trabajo práctico. Estas consignas son:

- La búsqueda de Pokémons por “nombre”.
- La búsqueda de Pokémons por “tipo”.

Al igual que las consignas anteriores, los módulos a los que les realizamos modificaciones son los mismos (“views.py” y “services.py”), pero modificamos otras funciones dentro de las mismas.

Las primeras funciones que modificamos son las que están ubicadas en el “views.py”, específicamente la llamada *search* y la llamada *filter\_by\_type*.



```
views.py x services.py transport.py home.html
app > views.py > ...
9     return render(request, 'index.html')
10
11 # 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
12 def home(request):
13     images = services.getAllImages() #En esta línea se obtienen la lista de imágenes transformadas en card y las muestra por pantalla.
14     favourite_list = []
15
16     return render(request, 'home.html', { 'images': images, 'favourite_list': favourite_list })
17
18 # función utilizada en el buscador.
19 def search(request):
20     name = request.POST.get('query', '')
21
22     # si el usuario ingresó algo en el buscador, se deben filtrar las imágenes por dicho ingreso.
23     if (name != ''):
24         images = services.filterByCharacter(name) #en esta línea de código se obtiene la card filtrada para mostrarla en pantalla.
25         favourite_list = []
26
27         return render(request, 'home.html', { 'images': images, 'favourite_list': favourite_list })
28     else:
29         return redirect('home')
30
31 # función utilizada para filtrar por el tipo del Pokemon
32 def filter_by_type(request):
33     type = request.POST.get('type', '')
34
35     if type != '':
36         images = services.filterByType(type) # debe traer un listado filtrado de imágenes, según si es o contiene ese tipo.
37         favourite_list = []
38
39         return render(request, 'home.html', { 'images': images, 'favourite_list': favourite_list })
40     else:
41         return redirect('home')
42
43 # Estas funciones se usan cuando el usuario está logueado en la aplicación.
44 @login_required
```

La primera función (*search*) es la utilizada para poder encontrar un Pokémon cuando ingresamos su nombre en el buscador. Para funcionar utiliza la capa de servicio para traer una lista de cards filtradas para mostrar la carta especificada en pantalla; esto lo hace mediante la función *filterByCharacter*.

La segunda función (*filter\_by\_type*) es la encargada de filtrar las cards por el tipo de Pokémon cuando seleccionamos uno de los botones que se muestran en pantalla.

En la siguiente imagen se muestran cómo se ven el buscador y los botones de filtro para buscar por tipo (fuego, agua y planta).



Estas funciones deben trabajar en conjunto con las funciones que modificamos en la capa de servicio. Estas son las funciones *filterByCharacter* y *filterByType* y sus respectivos códigos se muestran a continuación:

```

21 # Retorna lista_cards # Retorna un listado de cards
22
23
24 # función que filtra según el nombre del pokemon.
25 def filterByCharacter(name): ←
26     filtered_cards = []
27
28     for card in getAllImages(): #Recorre las tarjetas de la función que trae todas las imágenes
29         if name.lower() in card.name.lower(): #Verifica si el name esta contenido en el nombre de la card antes de agregarlo a listado de filtros,filtered_cards.
30
31             filtered_cards.append(card) #En caso de ser así agrega la card
32
33     return filtered_cards
34
35 # función que filtra las cards según su tipo.
36 def filterByType(type_filter): ←
37     filtered_cards = []
38
39     for card in getAllImages():
40
41         # debe verificar si la casa de la card coincide con la recibida por parámetro. Si es así, se añade al listado de filtered_cards.
42         if type_filter in card.types: # Verifica si el tipo se encuentra contenido en la card. Si es así se añade a la lista de cards filtradas.
43
44             filtered_cards.append(card)
45
46     return filtered_cards
47

```

Lo que hace la función *filterByCharacter* es recorrer con un ciclo (FOR) todas las cards de la función *getAllImages* para traer todas las imágenes y luego usa una condición para verificar si el nombre del Pokémon ingresado está contenido en el nombre de la card antes de agregarlo al listado de filtros.

De manera similar a la función anterior, *filterByType* recorre con un ciclo todas las cards y verifica si el tipo de la carta coincide con la recibida con uno de los botones de filtro. Si es así, la añade al listado de cartas filtradas.

## Conclusión

Llevamos a cabo un trabajo en equipo, donde tuvimos que aprender a dividir cada parte del programa para el correcto funcionamiento. Pudimos hacer uso de nuevos programas que facilitan el entorno para la programación y brindan muchas herramientas.

En cuanto a las dificultades que tuvimos a la hora de hacer el trabajo, podemos mencionar el poco conocimiento y entendimiento del uso de las ramas, los merges y las subidas de datos, ya que en ciertos momentos nos encontramos con algunos problemas que causaban conflictos en los códigos, lo que nos llevo a no poder avanzar como queríamos.

Como cierre de este informe podríamos decir que fue un proyecto con grandes desafíos, como conocer y poner en práctica los nuevos temas necesarios para la resolución de las consignas del trabajo práctico; por ejemplo, el uso de GitHub, repositorios, ramas y la sincronización de los archivos de forma local y remota.