Informe: TP 2025// PROGRAMACIÓN COM09

INTEGRANTES: LOPEZ CATERINA, ANTONELA SANTILLAN

El presente trabajo práctico consiste en el desarrollo de una aplicación web utilizando el framework Django, con el objetivo de construir una galería interactiva de Pokémon. La aplicación deberá conectarse a una API externa para obtener información de distintas criaturas Pokémon y presentarlas visualmente a través de Cards (tarjetas) que incluyan, como mínimo, su imagen, tipo, altura, peso y nivel base.

El proyecto ya cuenta con una estructura inicial que incluye los templates, parte de la lógica en las vistas (views.py) y módulos de transporte y traducción de datos. La tarea principal es completar la funcionalidad, el trabajo lo dividimos en dos partes la obligatoria y la opcional. Para completarlo realizamos lo siguiente:

**MUESTRA DE LAS CARDS**

En primer lugar, a la hora de realizar el trabajo, seguimos los pasos para la instalación de las extensiones y nos creamos una cuenta de GitHub. Luego pasamos a ver los tutoriales para copiar el proyecto en nuestro repositorio y descargarlo en nuestras computadoras. Una vez realizada las instalaciones necesarias, pasaos a ver los códigos ya implementados y comenzamos a realizar los pasos que faltaban, en views.py, services.py y home.html, para poder visualizar las imágenes en la galería. Comenzamos en views.py, en donde nos encontramos con una lista vacía de imágenes, y en services.py con una función para completar paso a paso,

1) Traer un listado de imágenes crudas desde la api (ver transport.py)

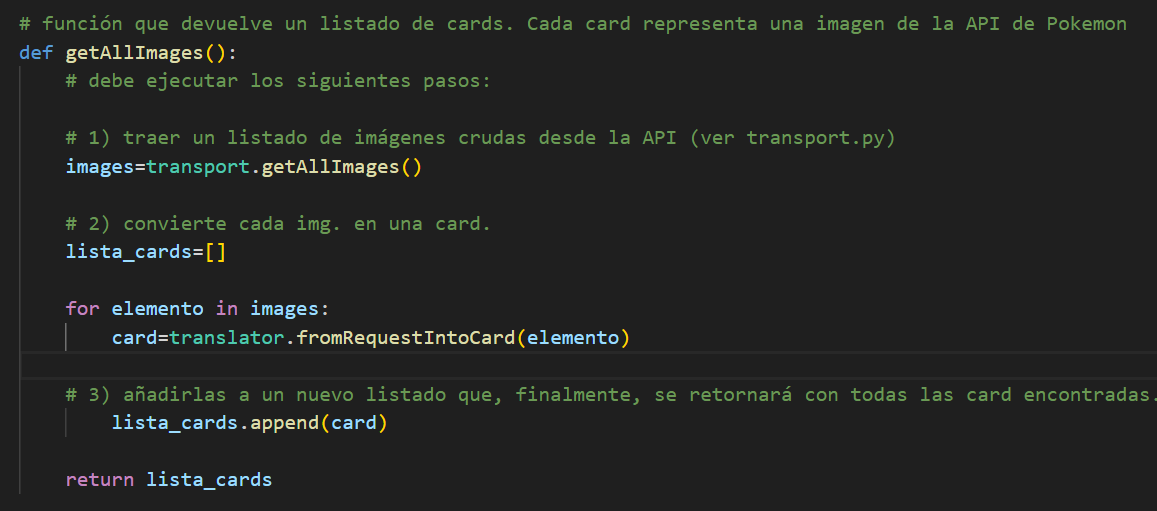
-para completar el paso uno, ingresamos al archivo transport.py donde ya existía una función que nos devuelve el listado de imágenes desde la api, la cual utilizamos para completar el paso uno asignándole el nombre images

2) convierte cada img en una Card

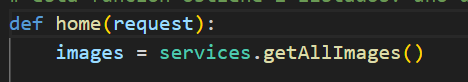
-Para este paso se encontró una función fromRequestIntoCard () en el archivo translator.py que se encarga de convertir las imágenes de la función getAllimages () en Cards. Para poder hacer la conversión, utilizamos un ciclo for para que recorra la lista de imágenes, dentro de este ciclo llamamos a la función ya creada fromRequestIntoCard en el archivo “translator.py” que es la encargada de convertir el formato crudo del elemento en un objeto de tipo "Card"

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

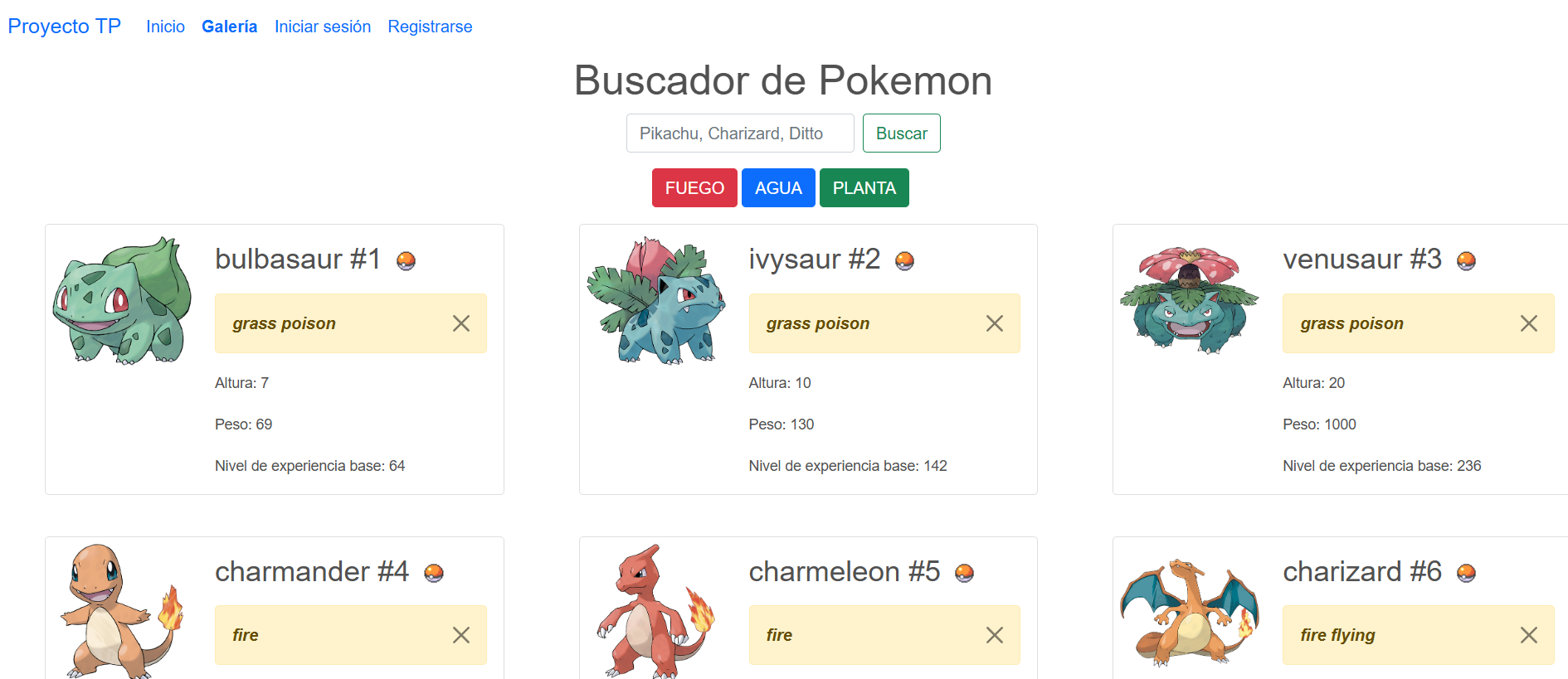
para resolver el paso 3, utilizamos el código append que va hacer el encargado de guardar todas las Cards en la lista vacia “Lista\_Cards”. Completado estos pasos la función getAllImages quedaría de la siguiente manera:



-una vez terminado la función que se encarga de devolvernos la lista de imágenes, volvemos a views.py para completar la función home, llamando a la función “getAllimages”.

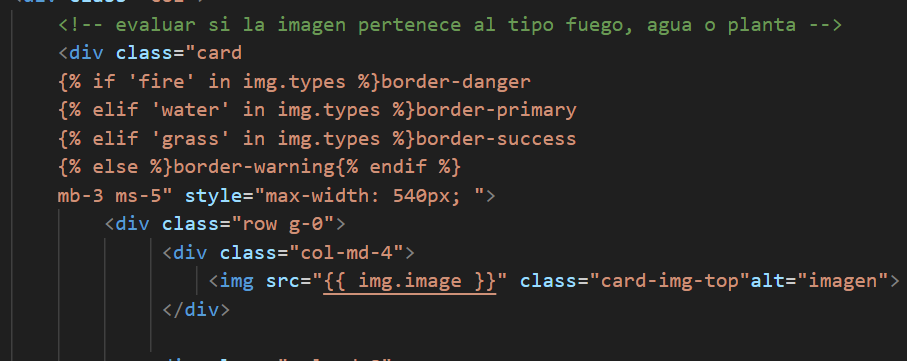


Al terminar con esto ya podremos visualizar las imágenes de los Pokémon en la galería:



**BORDES DE COLORES**

Para hacer que los bordes de las Cards cambien de color dependiendo de los tipos de Pokémon modificamos el home.html con bootstrap en un condicional.



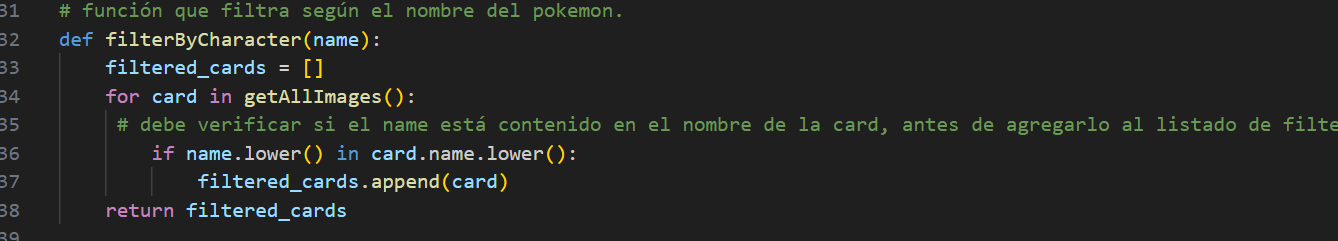
Una vez lo resolvimos pudimos ver a las Cards con los estilos que le asignamos dependiendo del tipo de Pokémon con éxito.

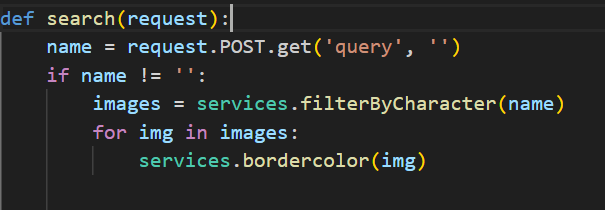


**BUSCADOR POKEMON**

Para resolver el buscador Pokémon tuvimos que investigar como convertir el texto ingresado por el usuario en minúsculas para que no haya errores al filtrar los Pokémon, por lo que decidimos utilizar “lower()”.

Una vez resuelta la función que filtra dependiendo de los que ingresa el usuario completamos la función “search” de la siguiente manera llamando a la función anteriormente completada “filterByCharacter”.

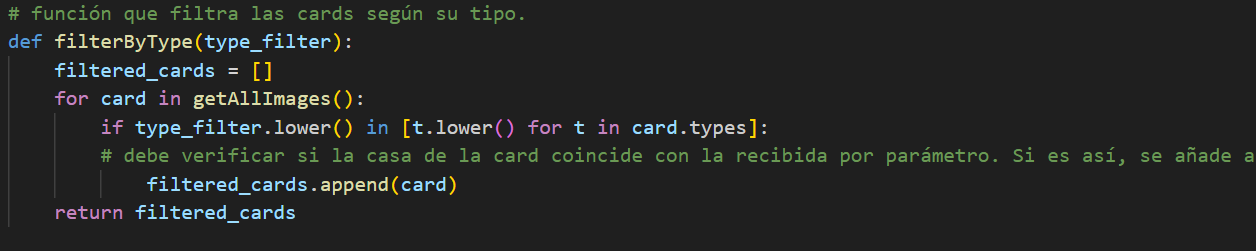




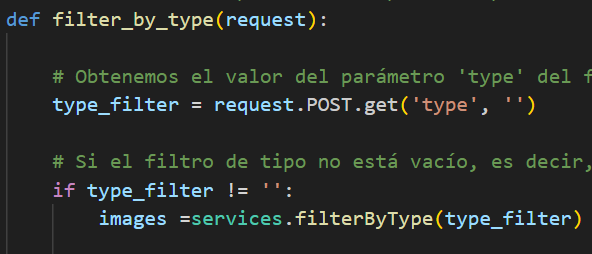
Al completar esto comprobamos que el buscador Pokémon funciona sin errores por mayúsculas.

**FILTROS**

Lo que hicimos para que los filtros “fuego”, “agua” y “planta” funcionaran fue convertir los tipos de Pokémon en minúsculas junto con los tipos ingresados para guardarlos y retornarlos una lista.



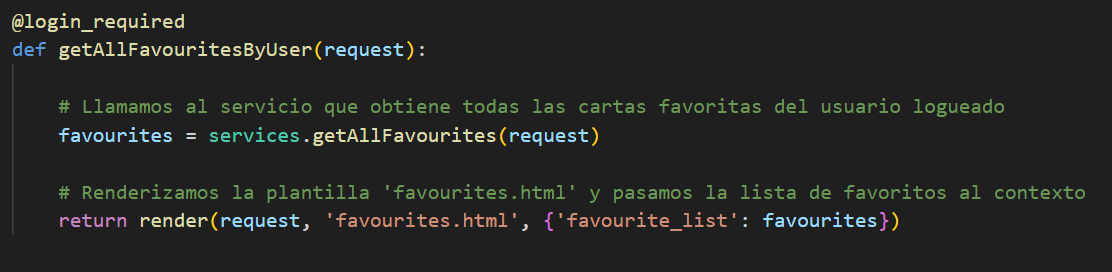
Al igual que search llamamos a la función filterByType en nuestra función filter\_by\_type en el archivo views.py de la siguiente manera:

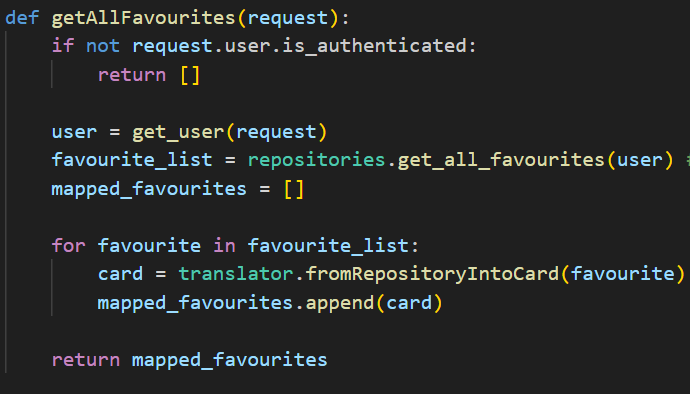


**FAVORITOS**

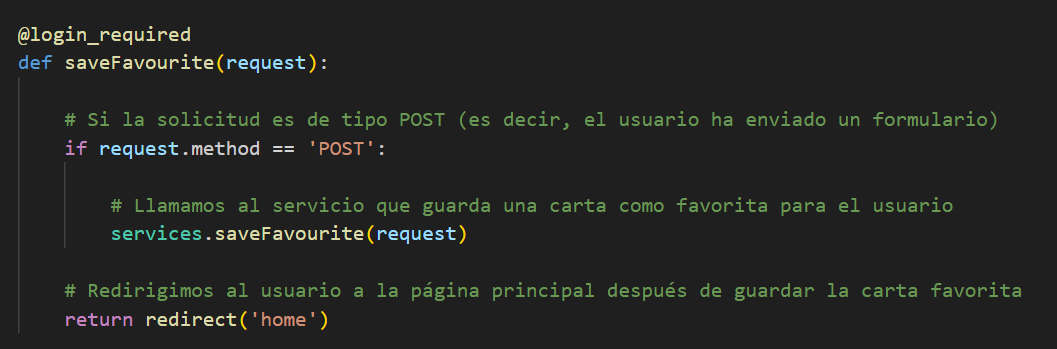
Para completar las funciones de favoritos simplemente seguimos los comentarios que estaban escritos en ellas y reutilizar los códigos que ya creados:

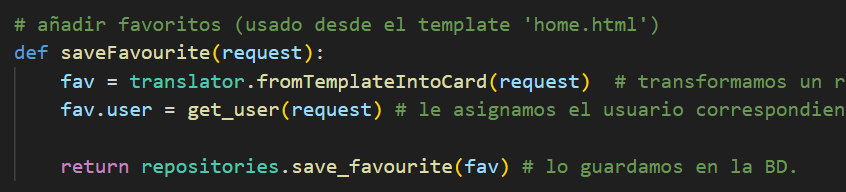
**Mostrar favoritos:**





**Guardar favoritos:**





**Eliminar favoritos:**

