INFORME DE TP INTRODUCCIÓN A LA PROGRAMACIÓN

Universidad Nacional General Sarmiento

Información del TP

introducción a la Programación
Comisión – 04
Docentes: Omar Argañaras y Nadia

Giuliana Anabella
Hua

25 de junio 2025

INTRODUCCION

El trabajo práctico consiste en desarrollar una aplicación web utilizando **Django**, cuyo objetivo principal es conectarse a la **PokeAPI** (una API pública con datos sobre Pokémon) para mostrar una galería de Pokémon en formato de tarjetas (cards). Además, se implementan funcionalidades como búsqueda por nombre, filtros por tipo (agua, fuego, planta), y la posibilidad de marcar favoritos si el usuario está autenticado.

La app busca integrar el consumo de una API externa, procesamiento de datos y presentación visual dinámica, con una estructura modular y buenas prácticas de desarrollo web.

Tecnologías utilizadas

- **Python + Django**: Framework principal para el backend y manejo de vistas, templates y lógica de servidor.
- **HTML + Django Templates**: Para el diseño visual y renderizado dinámico de las tarjetas y formularios.
- **Bootstrap**: Librería de estilos CSS usada para estructurar y dar diseño moderno a la interfaz (botones, cards, alertas).
- **PokeAPI**: Fuente de datos externa para obtener la información de los Pokémon.

DESARROLLO:

Esta aplicación permite:

- Mostrar cada Pokémon en una tarjeta visual ("card") con imagen y datos relevantes.
- Buscar Pokémon por nombre.
- Filtrar por tipo (agua, fuego, planta).
- Marcar Pokémon como favoritos (solo si el usuario está autenticado).
- Mostrar cada tarjeta con un borde de color según su tipo.

Funcionamiento General

views.py: Encargada de la lógica de vista (qué se muestra al usuario).

services.py: Lógica de negocio que transforma datos crudos de la API.

El home.html: Plantilla donde se visualizan las tarjetas.

- Se consulta la API para obtener datos de Pokémon.
- Se convierten esos datos en un formato unificado (Card).
- Se pasan al template, donde se renderizan visualmente.

- Se pueden aplicar filtros y búsquedas.
- Si el usuario lo desea, puede guardar favoritos
- **home(request)** – *views.py*

```python
def home(request):
    imagenes = get_all_pokemons()
    favourite_list = []
    if request.user.is_authenticated:
        favourite_list = get_favorites_by_user(request.user)

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

- **getAllImages() – *services.py***

```python
def getAllImages():
    pokemons = transport.get_pokemon_list()
    cards = []
    for raw_pokemon in pokemons:
        card = translator.raw_to_card(raw_pokemon)
        cards.append(card)
    return cards
```

- filterByCharacter(name)

```python
def filterByCharacter(name):
    return [card for card in getAllImages() if name.lower() in card.name.lower()]
```

- filterByType(type_filter)

```python
def filterByType(type_filter):
    return [card for card in getAllImages() if type_filter.lower() in [t.lower() for t in card.types]]
```

- saveFavourite(request)

```python
def saveFavourite(request):
    fav = translator.request_to_card(request.POST)
    fav.user = get_user(request)
    return repositories.save_favourite(fav)
```

- getAllFavourites(request)

```python
def getAllFavourites(request):
    if not request.user.is_authenticated:
        return []
    user = get_user(request)
    favourites = repositories.get_favourites_by_user(user)
    return [translator.favourite_to_card(fav) for fav in favourites]
```

Dificultades encontradas

- Obtener bien la información desde la API: A veces la API no devolvía todos los datos esperados o lo hacía en un formato complicado.
- Tuvimos que hacer una función traductora (translator.py) para convertirlo a un formato más entendible.
- Manejo de favoritos: Fue complicado guardar correctamente los datos en la base y asociarlos al usuario
- Estilo dinámico de las cards: Nos costó implementar que cambien de color según el tipo

Cosas que me hubiera gustado hacer son: Agregar detalles adicionales como estadísticas o habilidades al hacer clic en una tarjeta y diseñar una interfaz más atractiva usando CSS

Conclusion

Lo más difícil fue entender cómo usar **GitHub**. Me costó bastante al principio, porque no estaba familiarizada con los comandos, los commits, ni cómo resolver errores cuando subía cambios. Pero con prueba y error fui aprendiendo y entendiendo mejor cómo se usa.

También me sirvió para entender cómo trabajar con datos externos (como los que vienen en formato JSON), cómo filtrarlos, organizarlos y mostrarlos con HTML y Bootstrap.

Me hubiera gustado tener más tiempo para mejorar el diseño y agregar más funcionalidades.