Universidad Nacional de General Sarmiento
Instituto de Industria

Informe de Trabajo Practico Final;

Los Simpsons.

- ✓ Universidad Nacional de General Sarmiento.
- ✓ Materia: Introducción a la programación.
- ✓ Comisión de verano.
- ✓ Profesores: Gonzalo y Yair.
- ✓ Alumnos: Ignacio José Martin y Nicole Figueira.

¿De qué se trata este trabajo práctico?

El objetivo principal del proyecto fue construir una aplicación web inspirada en la serie animada Los Simpson; una especie de catálogo donde se pueden buscar y explorar distintos personajes de Springfield. Pero no se trata solo de mostrar imágenes, la app permite al usuario navegar, filtrar y buscar. En el mismo se pueden observar los colores de los bordes de las cartas que cambian según el estado (Vivo/Fallecido) de los personajes. Y otra característica que trae la app son las diferentes frases icónicas de los personajes que cambian al ser reiniciada la app.

Códigos implementados:

Services.py

```python
def getAllImages():

    """
    Obtiene todas las imagenes de personajes desde la API y las convierte en objetos card.

    1- Llama a transport.getAllImages() para obtener los datos crudos de la API.
    2- Recorre cada personaje recibido.
    3- Convierte cada personaje en un objeto card usando translator.fromRequestIntoCard().
    4. Guarda cada card en una lista.
    5- Retorna la lista final de Cards.
    """

    data = transport.getAllImages() # Trae los datos desde la API

    cards = [] # Lista donde se guardaran las cards convertidas

    for character in data:
        # Convierte el personaje en un objeto Card
        card = translator.fromRequestIntoCard(character)
        cards.append(card) # Agrega la card a la lista

    return cards # Retorna todas las cards
```

⬆Esta función obtiene los datos de los personajes desde la API utilizando la capa de transport. Luego convierte cada personaje en un objeto Card mediante el translator y retorna una lista con todas las cards.

```python
def filterByCharacter(name):
    """
    Filtra las cards de los personajes segun el nombre proporcionado.

    1- Obtiene todas las cards disponibles.
    2- Recorre cada card.
    3- Compara si el nombre buscado esta contenido dentro del nombre del personaje (ignorando mayúsculas).
    4- Devuelve solo las cards que coinciden.
    """
    all_cards = getAllImages() # Obtiene todas las cards

    # Filtra las cards cuyo nombre contiene el texto buscado (ignorando mayúsculas)
    filtered = [card for card in all_cards if name.lower() in card.name.lower()]

    return filtered # Retorna la lista filtrada


    pass
```

⬆ Esta función recibe un nombre como parámetro. Primero obtiene todas las cards disponibles y luego filtra solo aquellas cuyo nombre contiene el texto ingresado. Finalmente retorna la lista filtrada.

```python
def filterByStatus(status_name):

    """
    Filtra las cards de los personajes segun su estado (Alive o Deceased).

    1- Obtiene todas las cards.
    2- Recorre cada card.
    3- Compara el estado de la card con el estado recibido (ignorando mayúsculas).
    4- Devuelve solo las cards que coinciden con el estado buscado.
    """

    all_cards = getAllImages() # Obtiene todas las cards

    # Filtra por estado, comparando el estado de cada card con el estado buscado (ignorando mayúsculas)
    filtered = [card for card in all_cards if card.status.lower() == status_name.lower()]

    return filtered # Retorna la lista filtrada
```

⬆ Esta función recibe un estado (vivo o fallecido). Obtiene todas las cards y luego filtra solo las que coinciden con ese estado. Devuelve la lista filtrada.

Views.py

```python
def home(request):

    """
    Muestra la pagina principal con todas las imagenes de los personajes.

    1- Obtiene todas las imagenes desde el services.getAllImages().
    2- Crea una lista vacia para los favoritos (en este punto no se muestran los favoritos).
    3- Renderiza el template 'home.html' enviando las imagenes y favoritos.
    """
    images= services.getAllImages() # Trae todas las imagenes desde el services

    favourite_list= [] # Lista vacia de favoritos

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

🡅 Esta función se encarga de mostrar la pagina principal. Obtiene todas las imágenes desde el services y las envía al template home.html para que se muestren en pantalla.

```python
def search(request):

    """
    Busca personajes por nombre.

    1- Verifica que la solicitud sea POST.
    2- Obtiene el texto ingresado en el campo 'query'.
    3- Si el campo está vacío, redirige al home.
    4- Filtra las imagenes por nombre usando services.filterByCharacter(query).
    5- Renderiza el home con los resultados filtrados
    """
    if request.method == "POST":
        query = request.POST.get('query', '').strip() # Texto buscado


        if not query:
            return redirect('home') # Si el campo está vacío, redirige al home


        images = services.filterByCharacter(query) # Filtra por nombre


        favourite_list = [] # Lista vacia de favoritos

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


    return redirect('home')

    pass
```

⬆Esta función recibe el texto ingresado por el usuario en el buscador. Si el campo no esta vacío, llama a la función filterByCharacter() para obtener los resultados y los envía al template.

```python
def filter_by_status(request):

    """
    Filtra personajes por su estado (Alive o Deceased).

    1- Verifica que la peticion sea POST.
    2- Obtiene el estado desde el formulario.
    3- Si no hay estado, redirige al home.
    4- Filtra las imagenes por estado usando services.filterByStatus(status).
    5- Si el usuario esta logueado, obtiene su lista de favoritos, sino deja la lista vacia.
    6- Renderiza el home con los resultados.
    """

    if request.method == "POST":
        status = request.POST.get('status', '').strip()  # Alive o Deceased

        if not status:
            return redirect('home')

        images = services.filterByStatus(status) # Filtra por estado

        # Obtiene favoritos si el usuario esta autenticado, sino lista vacia
        if request.user.is_authenticated:
            favourite_list = services.getAllFavourites(request)
        else:
            favourite_list = []

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

    return redirect('home')

    pass
```

↑ Esta función recibe el estado seleccionado desde el formulario. Si el estado es válido, llama a la función filterByStatus() para obtener los personajes filtrados y los muestra en el template home.html.

Home.html

```
<!-- Recorre cada personaje de la lista de imagenes -->
{% for img in images %}
  <div class="col">
        <!-- Card de cada personaje, el borde cambia según el estado del personaje -->
        <div class="card h-100 shadow-sm
            {% if img.status == 'Alive' %}
                border-success
            {% elif img.status == 'Deceased' %}
                border-secondary
            {% else %}
                border-warning
            {% endif %}
        ">
        <div class="row g-0">
```

Se agrego una condición que cambia el color del borde según el estado del personaje:

- Border-success -> verde (si está vivo).
- Border-secondary -> gris (si está muerto).
- Boder-warning -> (cualquier otro estado).

```
<!-- Contenido -->
<div class="col-md-8">
  <div class="card-body">
    <h5 class="card-title text-center">{{ img.name }}</h5>
    <!-- Frase aleatoria del personaje, si no hay frases muestra un mensaje vacío -->
    <div class="alert alert-warning p-2 text-center">
        {% if img.phrases %}
            "{{ img.phrases|random }}"
        {% else %}
            ""
        {% endif %}
    </div>
```

↑ Acá se muestra el nombre del personaje dentro del titulo de la card. Verifica si el personaje tiene frases. Muestra una frase aleatoria usando el filtro random. Y si no hay frases, muestra unas comillas (“”).

La dificultad que tuvimos principalmente fue como trabajar sobre las funciones obligatorias que teníamos que completar ya que no sabíamos cómo empezar y desarrollarlo por lo que nos hizo perder mucho tiempo y nos preocupaba no poder resolverlo.

Para poder implementar los códigos que utilizamos, fue la ayuda de la IA muy útil para comprender como funcionaba el código y la implementación de la misma. Nos ayudó a entender mejor como conectaba todo. Fue una herramienta muy útil para poder llevar a cabo el proyecto y el resultado final. Aparte estuvimos consultando con compañeros que ya han cursado la materia e hicieron trabajos similares a este proyecto y que fue muy útil y pudimos implementarlos junto con la IA.

Lamentablemente por falta de tiempo y lo complicado que se veía nos hubiese gustado haber implementado las otras opciones para completar la app y a su vez asegurar una nota más alta.

Conclusión:

En resumen, este proyecto nos resultó bastante desafiante y a pesar de ya haber cursado más de una vez esta materia, esta vez pudimos realizar en su totalidad este proyecto y haber comprendido más cosas.