Universidad Nacional de General Sarmiento

Trabajo Práctico
“THE SIMPSONS API”

Introducción a la Programación

(Comisión Verano 2026)

El presente trabajo consistió en el desarrollo de una aplicación web basada en Los Simpsons, que permite visualizar una galería de personajes, en forma de cards, obtenidos desde una API externa.

Los usuarios pueden buscar a los personajes por su nombre en inglés con el buscador o filtrarlos en la galería por su estado actual en la serie, (vivos o fallecidos).

Además, se cuenta con un sistema de autenticación de usuarios y su correspondiente plantilla de registro. Esto habilita al usuario a marcar a sus personajes como “Favoritos”, acción que luego les permite visualizarlos en forma de tabla en la subsección correspondiente.

Por último, la aplicación web fue actualizada con un diseño acorde al universo de los Simpsons, con colores, imágenes y frases alusivas.

Integrantes: Florencia Gonzalez - DNI: 42.493.616

Natalia Mazzella - DNI: 34.271.838

1. Introducción

El trabajo desarrollado tuvo como objetivo hacer una aplicación web, con cuentas de usuarios, base de dato, consumo de API externa, mejoras de experiencia visual y desarrollo de una plataforma intuitiva para cualquier persona que la utilice sin necesidad de mayores explicaciones.
Al ingresar a la web podemos ver un gif animado que nos invita a ver la galería de personajes,con opciones de buscarlos por su nombre o bien para filtrar por los personajes vivos o muertos de la serie, y conocer su ficha de vida, incluyendo frases icónicas de los personajes de la serie.

2. Desarrollo

Funcionalidades principales:

Obtener el listado de imágenes convertidas en cards desde la API: En la capa services.py, dentro de la función **getAllImages**, realizamos un parámetro llamado colección, este parámetro lo que hace es conectarse a la capa transport.py y traer los datos de la API. También agregamos una lista vacía llamada cards, para ir agregando todas las cards y luego devolverlas todas juntas. Seguido creamos un ciclo for que recorre cada personaje dentro de colección, en este ciclo creamos el parámetro cartas, que se conecta al translator.py y transforma a cada iteración de personaje en una card. Para la elección de una frase aleatoria, utilizamos el condicional if y la biblioteca random para que cada frase dentro de la carta sea elegida al azar, y por último agregada al personaje dentro de la lista. Dentro de la capa views.py, en la función home conectamos la lista images a la función **getAllImages** de la capa services, para poder visualizarla en la web.

Borde de color según el estado del personaje: Se modificaron los colores a traves de la información de Bootstrap para hacer los bordes verde y grises.Se itera sobre la lista de imágenes y para cada una genera una tarjeta (card) dentro de un contenedor (col).

Buscadores por estado y nombre: Para poder buscar personajes mediante la barra de búsqueda, en la capa views se obtiene el valor ingresado y se almacena en la variable query. Si no se ingresa ningún dato, se redirige al home. En cambio, si existe un dato, desde la capa services se filtra y se devuelven aquellos que coincidan.
En la función **filterByCharacter** en el services.py, creamos una función que llama a getAllImages dentro del mismo archivo (esto lo hicimos para luego poder recorrer todas las cartas). También sumamos una lista vacía al código, llamada filtradas, para guardar los personajes encontrados por la palabra buscada por el usuario. Recorrimos cada carta dentro de la función con un for, y añadimos un condicional, que compara el nombre brindado con los de cada carta para agregarlo, si correspondiere, a la lista.

Para poder filtrar por estado del personaje, la base de nuestro código fue parecido al del filtro por personaje, lo que modificamos fue que al recorrer cada carta revisa el estado dependiendo de lo que se haya buscado, y lo guarda o no dentro de la lista para devolverlo.

En la capa views primero buscamos el valor del botón, si no se recibe ningún valor para status, se redirige al home; pero si se encuentra dentro de las cards el estado buscado, se retorna lo que se haya filtrado en la capa services.

Inicio de sesión: Ya se encontraba desarrollado en las funciones activas del trabajo práctico.

Favoritos: Se desarrollaron varias funciones en favoritos, para que los usuarios puedan interactuar con la plataforma:
saveFavourites(): Se transforma el request en una Card usando el translator, se asigna el usuario actual a la Card, se guarda la Card en el repositorio y se retorna el resultado. Si el personaje ya es un favorito, devuelve None y en la vista se vera el mensaje que definimos para este caso.
getAllFavourites(): Busca desde el repositorio los favoritos del usuario (User).
Se crea una lista vacía para almacenar los favoritos transformados en Cards. Se itera sobre cada favorito obtenido del repositorio. Se transforma cada favorito en una Card usando el translator. Se agrega la Card a la lista de favoritos transformados.Se retorna la lista de Cards que representan los favoritos del usuario.
deleteFavourite(): Se obtiene el ID del favorito a eliminar desde el POST del request. Se llama al repositorio para eliminar el favorito con el ID obtenido y se retorna el resultado.
En el template de favoritos de html, se agregaron los mensajes de alerta, si se realizó la acción correctamente o bien si hubo un error, cualquiera sea.
Si el user no esta autenticado no va a ver una lista, la misma va a estar vacía, pero si está autenticado ocurren dos opciones: ve la lista con sus favoritos agregados en ella, o bien la ve vacía, con una imagen de un personaje indicándole que debe ir a la vista de galería y agregar a sus personajes favoritos.

Alta de Users: Para el alta de nuevos usuarios se contaba con videos y la información técnica adjunta en el trabajo práctico.
La función lo que hace es verificar que la solicitud sea un POST (envió formulario de registro), obtiene de ese form los datos del usuario, comprueba que el nombre de usuario no exista usando **User.objects.filter(username=username).exists().**
Siel usuario ya existe, muestra un error en **register.html**.
Crea el usuario con **User.objects.create_user(...)**
Genera un correo de bienvenida con asunto, mensaje (incluye usuario y contraseña), remitente (DEFAULT_FROM_EMAIL) y destinatario (email).
Envía el correo usando send_mail(...); si falla, lo informa por consola, por no poder avanzar con esta lógica se decidió que no se envié el email y que se muestre el mensaje de falla, pero el usuario se registra de igual manera.

Qué hace:

- El usuario completa nombre, apellido, email, username y contraseña.
- El sistema verifica que el nombre de usuario no esté repetido.
- Si ya existe → muestra un mensaje de advertencia.
- Si no existe → crea el usuario en la base de datos.

Loading Spinner:En este caso lo que decidimos fue instalar un spinner de la base de bootstrap, pero queríamos darle un plus al hacer una imagen alusiva a los Simpsons con una dona girando, para esto nos sirvio de mucha ayuda la IA para poder lograrlo.
Se hizo una capa de carga que cuando el usuario hace click en algun link de la página o cambia de pestaña se ralentiza la carga adrede con el **class=showLoader()** dentro de los links de navegación para que se logre ver el efecto del spinner, se trabajó en las capas del html pero también en CSS para poder hacer el efecto de girar.

Renovar interfaz gráfica**:** Realizamos un cambio gráfico relacionado al mundo de los Simpson, utilizando los colores característicos del mismo, frases, imágenes y gifs. Para ello se usó código CSS y la librería Bootstrap. Los cambios son varios, en ocasiones se modificó directamente en el .html de la sección y en otros casos se realizaron ajustes de CSS en su plantilla correspondiente. Estas decisiones se fueron tomando acorde a como se visualizaba la implementación.
Hubo muchos problemas de ajustes, como por ejemplo, medidas de pad, tamaños de grillas, para eso se consultó con la documentación comentada, ayuda de IA para resolver los conflictos que surgían. Fue un trabajo que requirió la visualización de la web, el uso del comando “inspeccionar” como ayuda técnica para leer en qué parte del código era conveniente resolver la cuestión.
El objetivo final de esta renovación fue que la interfaz sea divertida, esté inmersa en el mundo de la serie, pero que mantenga el foco en ser intuitiva y funcional.

3. Conclusiones:

Durante este proyecto se sortearon desafíos y obstáculos respecto de funcionalidades y diseño, que llevaron a la inevitable comprensión de la lógica del desarrollo y el uso de programas/lenguajes para las tareas que se querían implementar.

Al trabajar en diferentes capas hubo mucho ejercicio de “prueba y error” para ver qué tarea estaba cumpliendo cada parte del código y tomar decisiones en base al objetivo final.
Se buscó mantener siempre clara la funcionalidad de los códigos escritos y que al pasar de pestañas y hacer clic la web no genere errores.

Trabajamos en puntos que no habíamos desarrollado durante la cursada por lo que hubo que hacer una investigación con Bootstrap, YouTube, documentación oficial de Django, Open IA y otros foros relacionados, por ejemplo para la creación de usuarios, envío de emails y el diseño general de la web.

Pudimos finalizar el proyecto con un diseño con estética “simpsoniana” y que a la vez sea funcional a lo que el trabajo requería obligatoriamente.

Consideramos que el trabajo llevó muchas horas de investigación, diversión y compromiso y gracias a eso logramos cumplir el objetivo de su correcto uso, visualización y aprendizaje.