Informe sobre las modificaciones y funcionalidades del proyecto.

Introducción

Este informe detalla las modificaciones realizadas en el proyecto basado en Django, diseñado para consumir una API y gestionar imágenes, así como la funcionalidad y propósito de cada cambio.

Descripción de las Modificaciones

a. Archivo `views.py`

1. **Modificación en la función `home`:**
   - **Antes:** No implementada.
   - **Ahora:** Implementa la lógica para obtener las imágenes desde la API y los favoritos del usuario (si está autenticado), renderizando el template `home.html`.
   - **Funcionalidad:** Permite mostrar las imágenes en la galería, combinando datos de la API y favoritos.
2. **Nueva función `search`:**
   - Permite realizar búsquedas sobre las imágenes consumidas desde la API, con soporte para casos en que el input esté vacío.
3. **Funciones relacionadas con favoritos:**
   - **`getAllFavouritesByUser`:** Obtiene los favoritos del usuario y renderiza `favourites.html`.
   - **`saveFavourite` y `deleteFavourite`:** Permiten agregar o eliminar favoritos desde la interfaz gráfica.
   - **`exit`:** Lógica de cierre de sesión.

b. Archivo `services.py`

1. **`getAllImages`:**
   - **Antes:** No implementada.
   - **Ahora:** Utiliza la capa `transport.py` para consumir datos de la API y convierte cada elemento en un objeto tipo `Card` mediante un traductor.
   - **Funcionalidad:** Centraliza la lógica de negocio para obtener imágenes de forma estructurada.
2. **`saveFavourite`:**
   - Lógica para convertir datos desde el template a un formato de `Card` y guardarlos con el usuario correspondiente.
3. **`getAllFavourites`:**
   - **Antes:** No implementada.

- **Ahora:** Permite obtener los favoritos almacenados de la base de datos, mapeados como `Card`.
- **Funcionalidad:** Conecta la capa de persistencia con la lógica de presentación.

4. **`deleteFavourite`:**
   - Lógica para eliminar favoritos de la base de datos.

c. Archivo `home.html`

1. **Uso de Condicionales en Django:**
   - Modificaciones en las tarjetas (`Cards`) para cambiar el borde según el estado del personaje: verde para vivos, rojo para muertos y naranja para desconocidos.
   - Se utilizaron clases de Bootstrap personalizadas para lograr este efecto.
2. **Lógica para favoritos:**
   - Renderiza un botón que varía entre "Añadir a favoritos" y "Ya está en favoritos", dependiendo del estado del ítem en la lista de favoritos del usuario autenticado.
3. **Buscador:**
   - Se agregó un formulario que envía consultas a través del método `POST` para filtrar imágenes por texto ingresado.

d. Archivo `favourites.html`

1. **Tabla de Favoritos:**
   - Modificaciones para mostrar la lista de favoritos en una tabla con datos relevantes (nombre, estado, ubicación, episodio, etc.).
   - Se agregó un botón de eliminación para cada elemento.

e. Archivo `repositories.py`

1. **Funciones:**
   - **`saveFavourite`:** Guarda un nuevo favorito en la base de datos.
   - **`getAllFavourites`:** Recupera la lista de favoritos para un usuario específico.
   - **`deleteFavourite`:** Elimina un favorito por su ID.
   - **Modificaciones:** Implementación completa de estas funciones con manejo de excepciones.

f. Archivo `transport.py`

1. **Modificación de `getAllImages`:**
    - Filtra y verifica la presencia de la clave `image` en los datos crudos de la API, asegurando que solo se procesen objetos válidos.
    - Manejo de errores y casos en los que no hay resultados.