## Proyecto: Red social similar a Twitter

### Descripción general

Este proyecto consiste en diseñar un sitio web de red social que permita a los usuarios realizar publicaciones, seguir a otros usuarios y dar "me gusta" a las publicaciones.

En el código proporcionado encontrarás un proyecto Django llamado `project4` que contiene una única app llamada `network`, estructurada de forma similar a la app de subastas del Proyecto 2.

---

## 1. Configuración inicial

### Archivos iniciales

- Abre el archivo `network/urls.py`. Este contiene las rutas URL configuradas para esta app, incluyendo las siguientes:
  - Ruta principal (`index`).
  - Ruta para iniciar sesión (`/login`).
  - Ruta para cerrar sesión (`/logout`).
  - Ruta para registrarse (`/register`).
- Explora el archivo `network/views.py`, que contiene las vistas asociadas a estas rutas:
  - `index`: Devuelve un template inicial `index.html`, que está vacío en gran parte.
  - `login_view`:
    - Si se accede con un método **GET**, muestra un formulario para iniciar sesión.
    - Si se envía con un método **POST**, autentica al usuario, lo inicia y redirige a la página principal.
  - `logout_view`: Cierra la sesión del usuario y redirige a la página principal.
  - `register`:
    - Muestra un formulario para registrarse.
    - Al enviar el formulario, crea un nuevo usuario.

### Ejecutar el servidor

Ejecuta `python manage.py runserver` para iniciar el servidor de desarrollo de Django y abre el sitio web en tu navegador.

1. Haz clic en "**Register**" y registra una nueva cuenta.
2. Comprueba cómo cambia el diseño del sitio según si el usuario está registrado o no.
   - El archivo `network/templates/network/layout.html` define el diseño del sitio. Verás que ciertas partes están envueltas en una comprobación `if user.is_authenticated` para mostrar contenido diferente si el usuario ha iniciado sesión.

## 2. Definición de modelos

En el archivo `network/models.py`, se encuentran los modelos para la aplicación:

- **Modelo** `User`:
  - Representa a los usuarios del sitio y hereda de `AbstractUser`, por lo que ya tiene campos como `username`, `email` y `password`.
  - Puedes añadir campos adicionales si necesitas representar más información sobre los usuarios.
- Además, deberás crear modelos adicionales para representar datos como:
  - Publicaciones (posts).
  - "Me gusta" (likes).
  - Seguidores (followers).

**Nota**:
Cada vez que realices cambios en los modelos, ejecuta:

```bash
python manage.py makemigrations python manage.py migrate
```

Copiar código

---

## 3. Especificaciones del proyecto

### Publicación nueva

- Los usuarios registrados podrán escribir publicaciones en texto mediante un área de texto y un botón para enviar.
- Puedes incluir esta funcionalidad en la parte superior de la página de "Todas las publicaciones" o en una página separada.

---

### Todas las publicaciones

- La opción "All Posts" en la barra de navegación llevará a una página que mostrará todas las publicaciones de todos los usuarios, ordenadas por las más recientes primero.
- Cada publicación debe incluir:
  - Nombre de usuario del autor.
  - Contenido de la publicación.

- Fecha y hora de creación.
- Cantidad de "me gusta" (inicialmente será 0).

---

**Página de perfil**

Al hacer clic en un nombre de usuario, se cargará su perfil, que debe mostrar:

1. Número de seguidores.
2. Número de usuarios a los que sigue.
3. Todas las publicaciones del usuario en orden cronológico inverso.
4. Si es otro usuario el que está viendo el perfil, debe mostrarse un botón para **"Seguir" o "Dejar de seguir"**.

Nota:
Un usuario **no puede seguirse a sí mismo**.

---

**Siguiendo**

- La opción "Following" en la barra de navegación llevará a una página donde se mostrarán solo las publicaciones de los usuarios que el usuario actual sigue.
- Esta página debe comportarse igual que la página de "Todas las publicaciones", pero con un conjunto más limitado de publicaciones.
- Solo estará disponible para usuarios registrados.

---

**Paginación**

En cualquier página que muestre publicaciones:

1. Muestra solo 10 publicaciones por página.
2. Si hay más de 10, debe aparecer un botón "Siguiente" para cargar la siguiente página.
3. Si no estás en la primera página, debe aparecer un botón "Anterior".

---

**Editar publicaciones**

- Los usuarios deben poder editar sus propias publicaciones mediante un botón o enlace de "Editar".
- Al hacer clic en "Editar":
    1. El contenido de la publicación debe reemplazarse por un área de texto editable.
    2. El usuario debe poder guardar los cambios.
    3. Usa **JavaScript** para que esta funcionalidad no requiera recargar toda la página.

**Nota de seguridad**:
Asegúrate de que los usuarios **no puedan editar publicaciones de otros usuarios**, ni siquiera de forma indirecta.

---

**“Me gusta” y “No me gusta”**

- Los usuarios podrán dar y quitar "me gusta" a cualquier publicación mediante un botón o enlace.
- Usa **JavaScript** para:
    1. Notificar al servidor sobre el cambio de forma asíncrona usando `fetch`.
    2. Actualizar el contador de "me gusta" en la página sin recargarla.

---

## 4. Pistas útiles

1. Revisa ejemplos de llamadas `fetch` en JavaScript en el Proyecto 3.
2. Usa la clase `Paginator` de Django para la paginación en el backend.
3. Explora las funcionalidades de paginación de **Bootstrap** para la interfaz.