Tribu

Implementa un proyecto web Django que permita gestionar una pequeña red social.

1. Puesta en marcha

Lleva a cabo los siguientes comandos para la puesta en marcha del proyecto:

```
just create-venv
source .venv/bin/activate
just setup
```

¿Qué ha ocurrido?

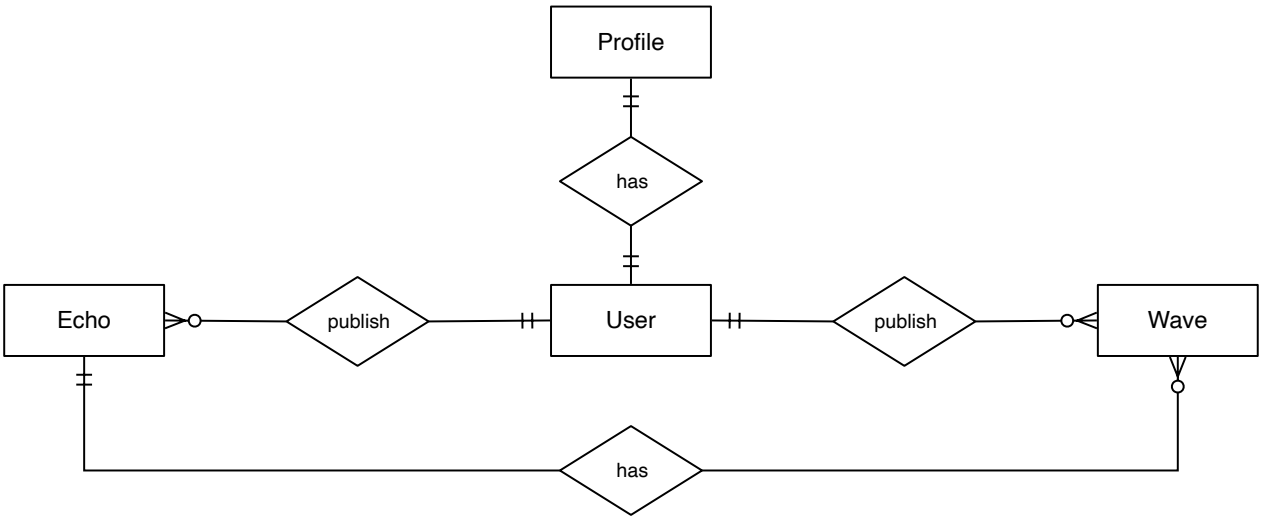
- Se ha creado un entorno virtual en la carpeta `.venv`
- Se han instalado las dependencias del proyecto.
- Se ha creado un proyecto Django en la carpeta `main`
- Se han aplicado las migraciones iniciales del proyecto.
- Se ha creado un superusuario con credenciales: `admin` - `admin`

2. Aplicaciones

Habrá que añadir las siguientes aplicaciones:

`shared`	Plantillas y utilidades compartidas.
`accounts`	Gestión de autenticación.
`echos`	Publicaciones en la red social.
`waves`	Respuestas a publicaciones en la red social.
`users`	Perfiles de usuario.

3. Modelos



3.1. `echos.Echo`

Campo	Tipo
`content`	*str*
`created_at`	*datetime*
`updated_at`	*datetime*

∗ Debes tener en cuenta las claves ajenas necesarias (*si procede*). No te olvides de `related_name`

3.2. `waves.Wave`

Campo	Tipo
`content`	*str*
`created_at`	*datetime*
`updated_at`	*datetime*

∗ Debes tener en cuenta las claves ajenas necesarias (*si procede*). No te olvides de `related_name`

3.3. `users.Profile`

Campo	Tipo	Por defecto
`avatar`	*image*	Avatar “placeholder”
`bio`	*str*	

∗ Debes tener en cuenta las claves ajenas necesarias (*si procede*).

3.4. Carga de datos

Una vez que hayas creado los modelos y aplicado las migraciones, puedes cargar datos de prueba con la siguiente receta just:

```
just load-data
```

4. URLs

`/login/`

- Inicio de sesión.
- Se debe solicitar nombre de usuario y contraseña.

`/logout/`

- Cierre de sesión.
- Mediante petición `GET`.

`/signup/`

- Registro de usuario.
- Campos a desplegar en el formulario:
  - Nombre de usuario.
  - Contraseña.
  - Nombre.
  - Apellidos.
  - Correo electrónico.
- Aquí se debe crear también el perfil del usuario registrado. Será un perfil "vacío" pero vinculado con el usuario.

`/`

- Debe redirigir a `/echos/`

`/echos/`

- Listar todos los "echos" de la red social.
- Deben estar ordenados de forma *descendente* por su *fecha de creación*.
- Truncar el contenido de cada "echo" si es mayor de **20 palabras** *(puntos suspensivos)*.

`/echos/add/`

- Añadir un nuevo "echo".
- Formulario únicamente con el contenido.

`/echos/45/`

- Detalle del "echo" con clave primaria 45.
- Se deben mostrar **sólo los 5 "waves" más recientes** ordenados de forma *descendente* por su *fecha de creación*.
- Se debe mostrar el nombre de usuario del autor/a del "echo" y del autor/a de cada "wave".
- Debe existir un enlace para ver todos los "waves" del "echo" **siempre y cuando** existan en total más de 5 "waves".

`/echos/45/waves/`

- Detalle del "echo" con clave primaria 45.
- Se deben mostrar **todos los "waves"** ordenados de forma *descendente* por su *fecha de creación*.
- Se debe mostrar el nombre de usuario del autor/a del "echo" y del autor/a de cada "wave".
- **No** debe existir un enlace para ver todos los "waves".

`/echos/45/edit/`

- Editar el "echo" con clave primaria 45.
- Formulario únicamente con el contenido.
- Devolver 403 Forbidden si el usuario logeado no es el autor del "echo".

`/echos/45/delete/`

- Borrar el "echo" con clave primaria 45.
- Devolver 403 Forbidden si el usuario logeado no es el autor del "echo".

`/echos/45/waves/add/`

- Añadir un "wave" al "echo" con clave primaria 45.
- Formulario únicamente con el contenido.

`/waves/27/edit/`

- Editar el "wave" con clave primaria 27.
- Formulario únicamente con el contenido.
- Devolver 403 Forbidden si el usuario logeado no es el autor del "wave".

`/waves/27/delete/`

- Borrar el "wave" con clave primaria 27.
- Devolver 403 Forbidden si el usuario logeado no es el autor del "wave".

`/users/`

- Mostrar listado de usuarios de la red social.
- Deben aparecer los nombres de usuario.
- Debe existir un enlace al perfil de cada usuario.

`/users/guido/`

- Perfil del usuario con nombre de usuario `guido`.
- Deben aparecer los siguientes campos de **usuario**:
  - Nombre.
  - Apellidos.
  - Nombre de usuario.
  - Correo electrónico.
- Deben aparecer los siguientes campos de **perfil**:
  - Biografía.
  - Fotografía de "avatar".
- Debe aparecer un enlace para **editar el perfil** *(sólo cuando se trata del usuario logeado)*.
- Sólo deben aparecer los **5 últimos "echos"** publicados por el usuario ordenados de forma *descendente* por su *fecha de creación*.

`/users/guido/echos/`

- Perfil del usuario con nombre de usuario `guido`.
- Deben aparecer los siguientes campos de **usuario**:
  - Nombre.
  - Apellidos.
  - Nombre de usuario.
  - Correo electrónico.
- Deben aparecer los siguientes campos de **perfil**:
  - Biografía.
  - Fotografía de "avatar".
- Debe aparecer un enlace para **editar el perfil** *(sólo cuando se trata del usuario logeado)*.
- Deben aparecer **todos los "echos"** publicados por el usuario ordenados de forma *descendente* por su *fecha de creación*.

`/users/@me/`

- Perfil del usuario logeado.
- Es una simple redirección a `/users/guido/` *(como ejemplo)*

`/users/guido/edit/`

- Editar el perfil de usuario.
- Formulario con campos de biografía y "avatar".

5. Administración

Los siguientes modelos deben estar accesibles desde la **interfaz administrativa** de Django:

- `echos.Echo`
- `waves.Wave`
- `users.Profile`

