Programación web de CS50 con Python y JavaScript

OpenCourseWare

Donar (https://cs50.harvard.edu/donate)

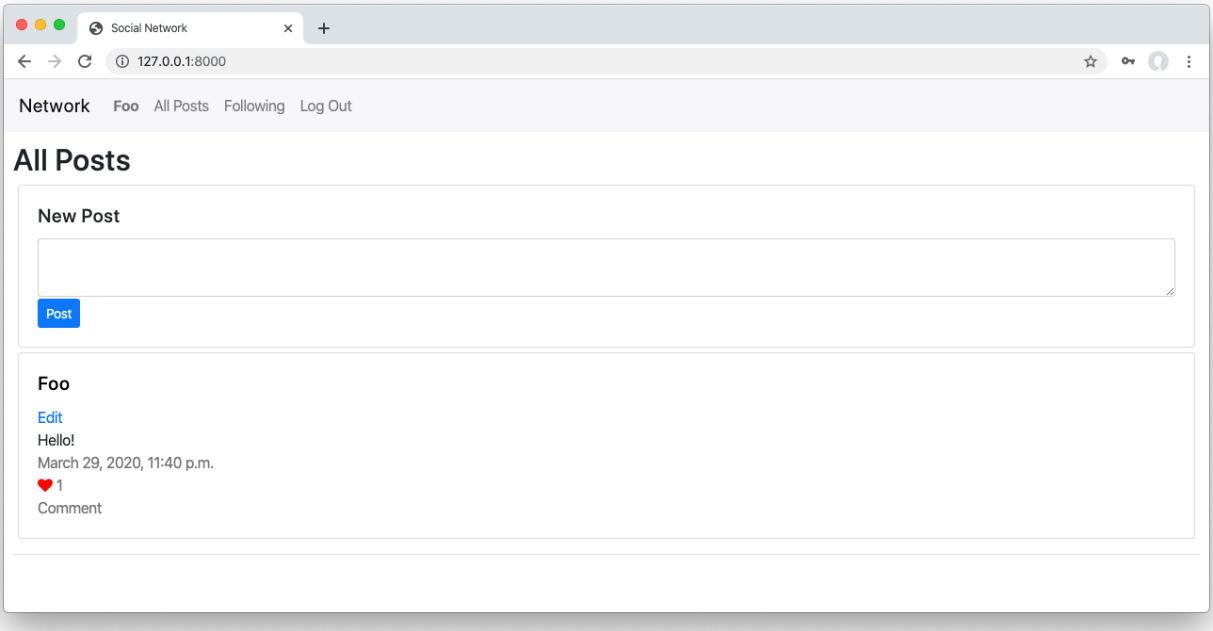
Brian Yu (https://brianyu.me)
brian@cs.harvard.edu

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Red

> CS50W no presenta una correspondencia uno a uno entre conferencias y proyectos. Si estás intentando este proyecto sin haber visto al menos la Lección 7, ¡lo estás intentando demasiado pronto!

Diseñe un sitio web de red social similar a Twitter para realizar publicaciones y seguir a usuarios.



Empezando

1. Descargue el código de distribución de https://cdn.cs50.net/web/2020/spring/projects/4/network.zip (https://cdn.cs50.net/web/2020/spring/projects/4/network.zip) y descomprímalo.
2. En tu terminal, `cd` en el `project4` directorio.
3. Ejecute `python manage.py makemigrations network` para realizar migraciones para la `network` aplicación.
4. Ejecute `python manage.py migrate` para aplicar migraciones a su base de datos.

Comprensión

En el código de distribución hay un proyecto Django llamado `project4` que contiene una única aplicación llamada `network`, estructurada de manera similar a la aplicación del Proyecto 2 `auctions`.

Primero, abra `network/urls.py`, donde se define la configuración de URL para esta aplicación. Tenga en cuenta que ya hemos escrito algunas URL para usted, incluida una ruta de índice predeterminada, una `/login` ruta, una `/logout` ruta y una `/register` ruta.

Échale un vistazo `network/views.py` para ver las vistas asociadas a cada una de estas rutas. La vista de índice por ahora devuelve una plantilla casi vacía `index.html`. La `login_view` vista muestra un formulario de inicio de sesión cuando un usuario intenta OBTENER la página. Cuando un usuario envía el formulario utilizando el método de solicitud POST, el usuario se autentica, inicia sesión y se le redirige a la página de índice. La `logout_view` vista cierra la sesión del usuario y lo redirige a la página de índice. Finalmente, la `register` ruta muestra un

formulario de registro al usuario y crea un nuevo usuario cuando se envía el formulario. Todo esto se hace por usted en el código de distribución, por lo que debería poder ejecutar la aplicación ahora para crear algunos usuarios.

Ejecute `python manage.py runserver` para iniciar el servidor web Django y visite el sitio web en su navegador. Haga clic en "Registrarse" y regístrese para obtener una cuenta. Debería ver que ahora ha iniciado sesión como su cuenta de usuario y que los enlaces en la parte superior de la página han cambiado. ¿Cómo cambió el HTML? Eche un vistazo `network/templates/network/layout.html` al diseño HTML de esta aplicación. Tenga en cuenta que varias partes de la plantilla están incluidas en una marca de verificación if `user.is_authenticated`, de modo que se puede representar contenido diferente dependiendo de si el usuario ha iniciado sesión o no. ¡Puedes cambiar este archivo si deseas agregar o modificar algo en el diseño!

Finalmente, eche un vistazo a `network/models.py`. Aquí es donde definirá cualquier modelo para su aplicación web, donde cada modelo representa algún tipo de datos que desea almacenar en su base de datos. Comenzamos con un `User` modelo que representa a cada usuario de la aplicación. Debido a que hereda de `AbstractUser`, ya tendrá campos para un nombre de usuario, correo electrónico, contraseña, etc., pero puede agregar nuevos campos a la `User` clase si hay información adicional sobre un usuario que desea representar. También deberá agregar modelos adicionales a este archivo para representar detalles sobre publicaciones, me gusta y seguidores. Recuerde que cada vez que cambie algo en `network/models.py`, primero deberá ejecutar `python manage.py makemigrations` y luego `python manage.py migrate` migrar esos cambios a su base de datos.

Especificación

Utilizando Python, JavaScript, HTML y CSS, complete la implementación de una red social que permita a los usuarios realizar publicaciones, seguir a otros usuarios y dar “me gusta” a las publicaciones. Debes cumplir con los siguientes requisitos:

- **Nueva publicación** : los usuarios que hayan iniciado sesión deberían poder escribir una nueva publicación basada en texto completando texto en un área de texto y luego haciendo clic en un botón para enviar la publicación.
    - La captura de pantalla en la parte superior de esta especificación muestra el cuadro "Nueva publicación" en la parte superior de la página "Todas las publicaciones". Puede optar por hacer esto también o puede hacer que la función "Nueva publicación" sea una página separada.
- **Todas las publicaciones** : el enlace "Todas las publicaciones" en la barra de navegación debe llevar al usuario a una página donde pueda ver todas las publicaciones de todos los usuarios, con las publicaciones más recientes primero.
    - Cada publicación debe incluir el nombre de usuario del autor, el contenido de la publicación en sí, la fecha y hora en que se realizó la publicación y la cantidad de

"me gusta" que tiene la publicación (esto será 0 para todas las publicaciones hasta que implemente la capacidad de "Me gusta" una publicación más tarde).

- **Página de perfil** : al hacer clic en un nombre de usuario se debe cargar la página de perfil de ese usuario. Esta página debería:
  - Muestra la cantidad de seguidores que tiene el usuario, así como la cantidad de personas que sigue.
  - Muestra todas las publicaciones de ese usuario, en orden cronológico inverso.
  - Para cualquier otro usuario que haya iniciado sesión, esta página también debería mostrar un botón "Seguir" o "Dejar de seguir" que permitirá al usuario actual alternar si sigue o no las publicaciones de este usuario. Tenga en cuenta que esto sólo se aplica a cualquier "otro" usuario: un usuario no debería poder seguirse a sí mismo.
- **Siguiente** : el enlace "Siguiente" en la barra de navegación debe llevar al usuario a una página donde vea todas las publicaciones realizadas por los usuarios que sigue el usuario actual.
  - Esta página debería comportarse igual que la página "Todas las publicaciones", solo que con un conjunto más limitado de publicaciones.
  - Esta página solo debería estar disponible para los usuarios que hayan iniciado sesión.
- **Paginación** : en cualquier página que muestre publicaciones, las publicaciones solo deben mostrarse 10 en una página. Si hay más de diez publicaciones, debería aparecer un botón "Siguiente" para llevar al usuario a la siguiente página de publicaciones (que debe ser anterior a la página de publicaciones actual). Si no está en la primera página, debería aparecer un botón "Anterior" para llevar al usuario también a la página anterior de publicaciones.
  - Consulte la sección **Sugerencias** para obtener algunas sugerencias sobre cómo implementar esto.
- **Editar publicación** : los usuarios deberían poder hacer clic en el botón "Editar" o en un enlace en cualquiera de sus propias publicaciones para editar esa publicación.
  - Cuando un usuario hace clic en "Editar" en una de sus propias publicaciones, el contenido de su publicación debe reemplazarse con un `textarea` lugar donde el usuario pueda editar el contenido de su publicación.
  - Luego, el usuario debería poder "Guardar" la publicación editada. Usando JavaScript, deberías poder lograr esto sin necesidad de recargar toda la página.
  - Por seguridad, asegúrese de que su aplicación esté diseñada de manera que no sea posible para un usuario, por cualquier ruta, editar las publicaciones de otro usuario.
- **“Me gusta” y “No me gusta”** : los usuarios deberían poder hacer clic en un botón o enlace en cualquier publicación para alternar si les “gusta” o no esa publicación.
  - Al utilizar JavaScript, debe informar al servidor de forma asincrónica que actualice el recuento de Me gusta (como mediante una llamada a `fetch`) y luego actualizar el recuento de Me gusta de la publicación que se muestra en la página, sin necesidad de volver a cargar toda la página.

Consejos

- Para ver ejemplos de `fetch` llamadas de JavaScript, puede que le resulten útiles algunas de las rutas del Proyecto 3.
- Probablemente necesitará crear uno o más modelos `network/models.py` y/o modificar el modelo existente `User` para almacenar los datos necesarios para su aplicación web.
- La clase Paginator (https://docs.djangoproject.com/en/4.0/topics/pagination/) de Django puede ser útil para implementar la paginación en el back-end (en su código Python).
- Las funciones de paginación (https://getbootstrap.com/docs/4.4/components/pagination/) de Bootstrap pueden ser útiles para mostrar páginas en el front-end (en su HTML).

Cómo enviar

1. Visite este enlace (https://submit.cs50.io/invites/89679428401548238ceb022f141b9947) , inicie sesión con su cuenta de GitHub y haga clic en **Autorizar cs50** . Luego, marque la casilla que indica que desea otorgar acceso al personal del curso a sus envíos y haga clic en **Unirse al curso** .
2. Instale Git (https://git-scm.com/downloads) y, opcionalmente, instale `submit50` (https://cs50.readthedocs.io/submit50/) .

Cuando envíe su proyecto, el contenido de su `web50/projects/2020/x/network` rama debe coincidir con la estructura de archivos del código de distribución descomprimido tal como se recibió originalmente. Es decir, sus archivos no deben estar anidados dentro de ningún otro directorio de su propia creación. Su rama tampoco debe contener ningún código de ningún otro proyecto, solo este. Si no cumple con esta estructura de archivos, es probable que su envío sea rechazado.

A modo de ejemplo, para este proyecto eso significa que si el personal de calificación visita `https://github.com/me50/USERNAME/tree/web50/projects/2020/x/network` (donde `USERNAME` está su propio nombre de usuario de GitHub como se proporciona en el formulario a continuación), deberíamos ver los dos subdirectorios (`network` , `project4`) y el `manage.py` archivo. Si no es así como está organizado su código cuando lo verifica, reorganice su repositorio necesario para que coincida con este paradigma.

3. Si lo ha instalado `submit50` , ejecute

   ```
   submit50 web50/projects/2020/x/network
   ```

   De lo contrario, usando Git, envíe su trabajo a `https://github.com/me50/USERNAME.git` , donde `USERNAME` está su nombre de usuario de GitHub, en una rama llamada `web50/projects/2020/x/network` .

4. Grabe un screencast (https://www.howtogeek.com/205742/how-to-record-your-windows-mac-linux-android-or-ios-screen/) que no exceda los 5 minutos de duración (y no lo cargue más de un mes antes de enviar este proyecto), en el que demuestre la funcionalidad de su proyecto. Asegúrese de que cada elemento de la especificación anterior se demuestre en su video. No es necesario mostrar su código en este video, solo su aplicación en acción; Revisaremos su código en GitHub. Sube ese video a YouTube (https://www.youtube.com/upload) (como no listado o público, pero no privado) o en otro lugar. En la descripción de su video, debe marcar la hora en la que su video demuestra cada uno de los siete (7) elementos de la especificación. **Esto no es opcional** , los videos sin marcas de tiempo en su descripción serán rechazados automáticamente.
5. Envíe este formulario (https://forms.cs50.io/31251163-446a-4aa0-b84a-c6f89bc24352) .

Luego puede ir a https://cs50.me/cs50w (https://cs50.me/cs50w) para ver su progreso actual.