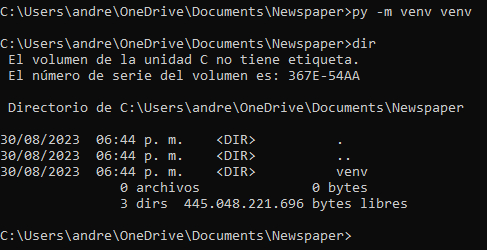
**PROYECTO: NEWSPAPER - LOGIN**

**OBJETIVO**: Crear un proyecto llamado Newspaper que utiliza un modelo de usuario personalizado que permite realizar cambios en el modelo de usuario.

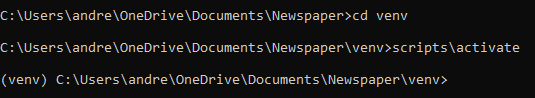
**CARACTERISTICAS**: El modelo de autenticación es bueno para la mayoría de los casos, pero es posible que tenga necesidades que no se satisfagan con los valores predeterminados; Personalizar la autenticación requiere comprender qué puntos del sistema proporcionado son extensibles o reemplazables.

Los backends de autenticación proporcionan un sistema extensible para cuando un nombre de usuario y una contraseña almacenados con el modelo de usuario deben autenticarse en un servicio diferente al predeterminado de Django; Puede otorgar a sus modelos, permisos personalizados que se pueden verificar a través del sistema de autorización de Django y puede ampliar el modelo predeterminado ***User*** o sustituirlo por un modelo completamente personalizado.

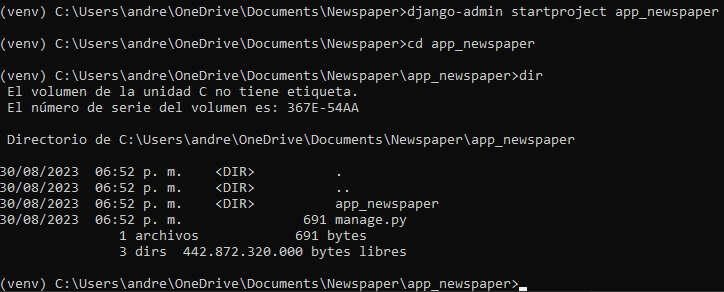
1. Ingresamos a la carpeta donde hemos creado el ambiente virtual



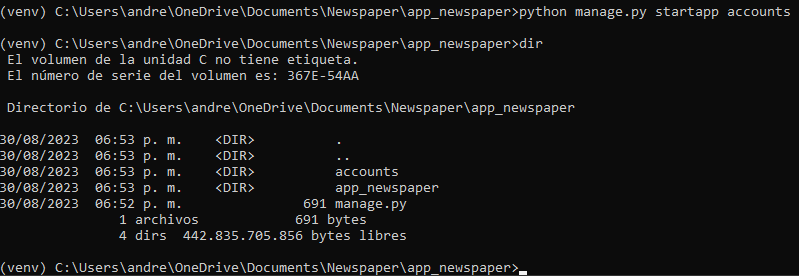
1. Activamos el ***ambiente virtual de Python***



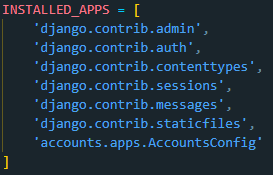
1. Creamos un nuevo ***proyecto app\_newspaper*** e ingresamos a la carpeta creada



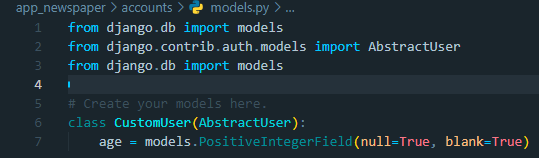
1. Creamos una ***aplicación*** ***accounts*** que gestiona un modelo de usuario personalizado



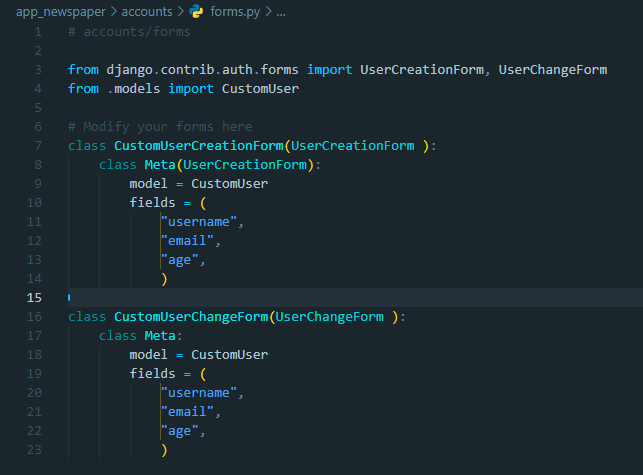
1. Actualizamos el archivo ***app\_newspaper/settings.py*** para gestionar la aplicación accounts y en la parte inferior del archivo, usar la configuración ***AUTH\_USER\_MODEL*** para decirle a Django que use nuevo modelo de usuario personalizado en lugar del modelo de usuario incorporado.



1. Actualizar ***account/models.py*** con un nuevo modelo de usuario llamado ***CustomUser*** que amplía el ***AbstractUser*** existente. Incluimos el campo personalizado para la edad (***age)***. Para crear el modelo ***User***, emplearemos la clase ***AbstractUser*** que posee los siguientes atributos: username, first\_name, last\_name, email, password, groups, user\_permissions, is\_staff, is\_active, is\_superuser, last\_login y date\_joined



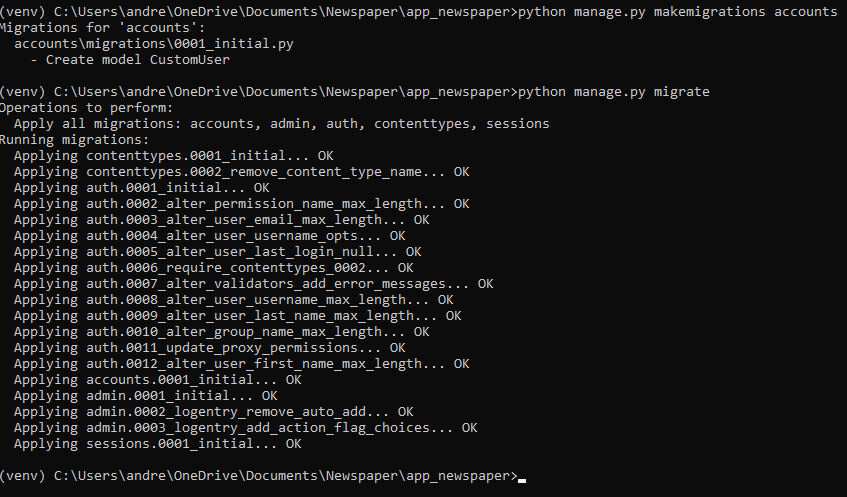
1. Creamos un nuevo archivo llamado ***accounts/forms.py*** y actualizamos el siguiente código para ampliar los formularios integrados ***UserCreationForm y UserChangeForm***



1. Actualizamos el archivo llamado accounts/admin.py

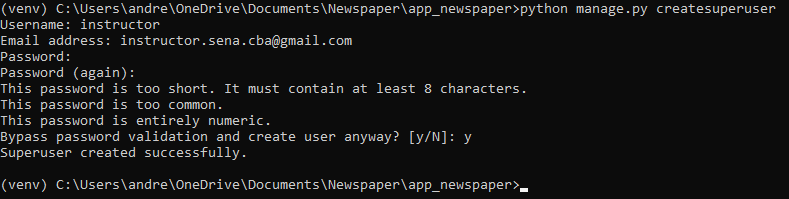


1. Ejecutamos los comandos makemigrations y migrate para crear una nueva base de datos que use el modelo de usuario personalizado:

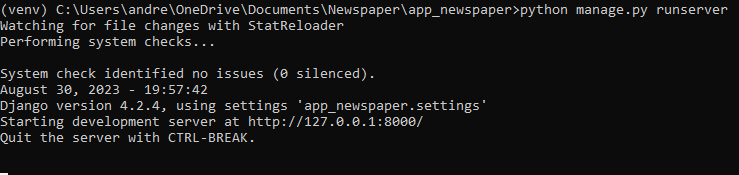


* + - * + ***python manage.py makemigrations accounts***
        + ***python manage.py migrate***

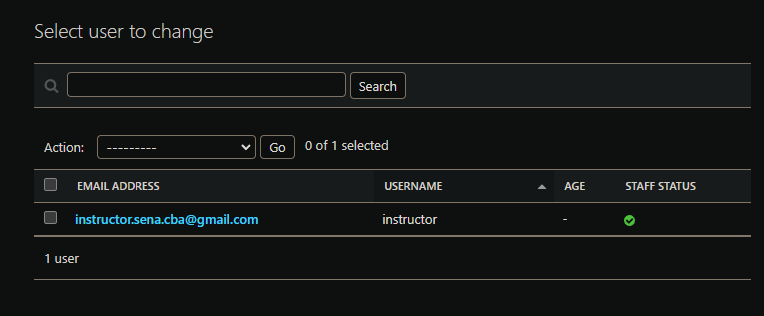
1. Creamos una cuenta de ***superuser*** para confirmar que todo funciona.



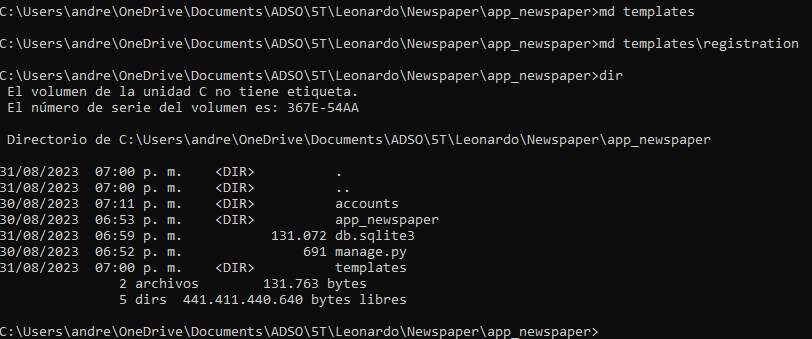
1. Iniciamos el servidor web para verificar lo realizado hasta el momento
   * 1. ***python manage.py runserver***



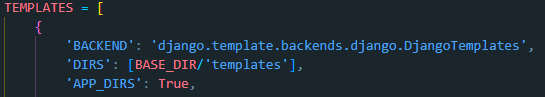
* + 1. Ingresamos en el navegador la URL ***http://127.0.0.1:8000/admin*** e iniciamos sesión. Al hacer clic en ***Users*** se debe visualizar la cuenta de ***Superuser*** creada



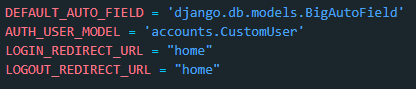
1. Creamos un nuevo directorio de plantillas y dentro un directorio de registro, ahí es donde Django buscará las plantillas relacionadas con el inicio de sesión y el registro.



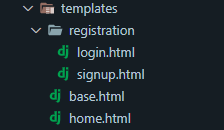
1. Actualizamos la configuración de ***TEMPLATES*** en el archivo ***app\_newspaper/settings.py***



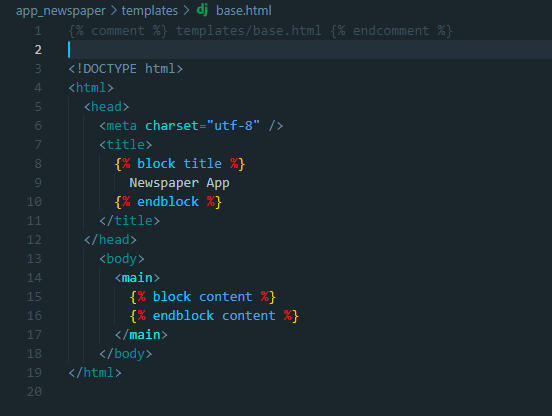
1. Agregamos dos líneas al final del archivo django\_project/settings.py con las configuraciones LOGIN\_REDIRECT\_URL y LOGOUT\_REDIRECT\_URL para informarle a Django dónde enviar a los usuarios cuando se inicia o se cierra sesión.



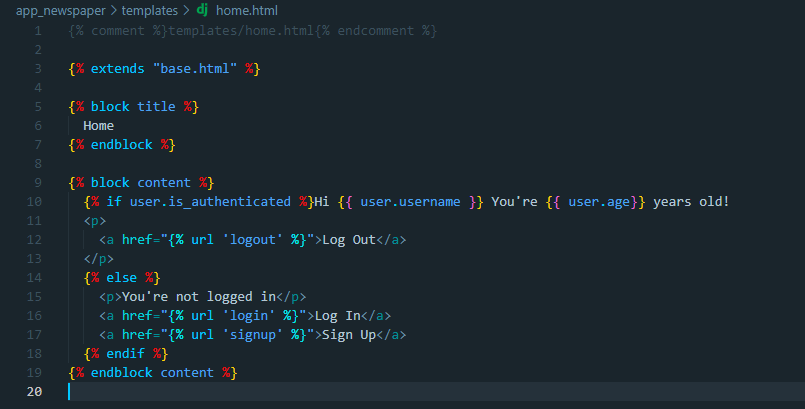
1. Crear cuatro nuevas plantillas:
   * ***templates/base.html***
   * ***templates/home.html***
   * ***templates/registration/login.html ● templates/registration/signup.html***



1. Digitamos el código dentro del archivo ***templates/base.html***



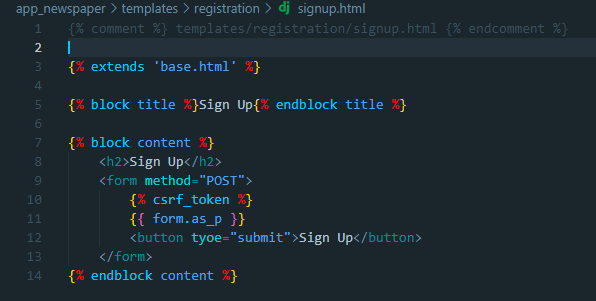
1. Digitamos el código dentro del archivo ***templates/home.html***



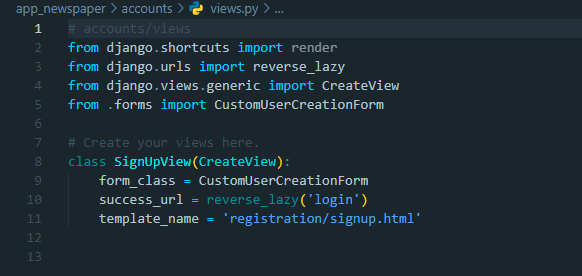
1. Digitamos el código dentro del archivo ***templates/registration/login.html***



1. Digitamos el código dentro del archivo ***templates/registration/signup.html***



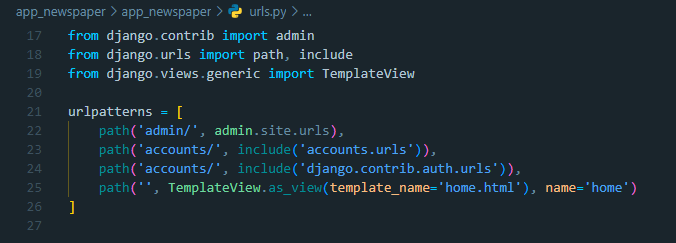
1. El archivo ***accounts/views.py*** contendrá la lógica del formulario de registro



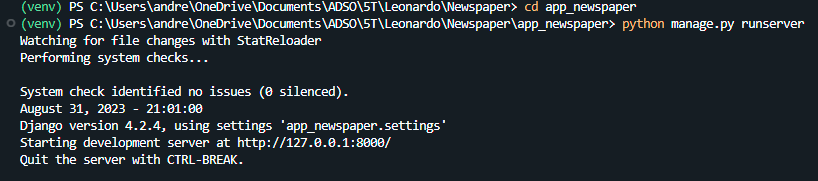
1. Creamos un archivo llamado ***accounts/urls.py***



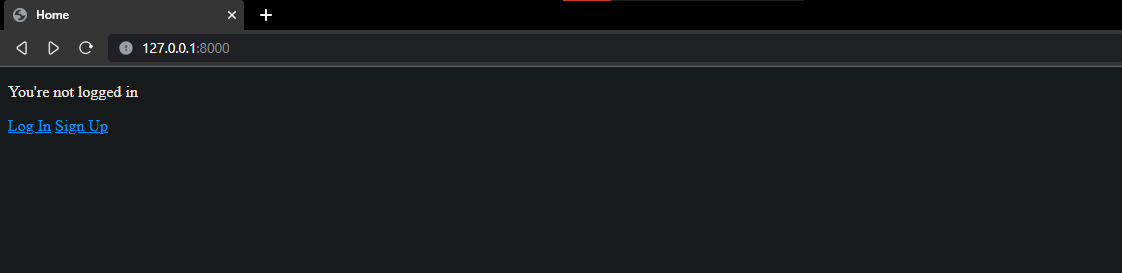
1. En el archivo ***app\_newspaper/urls.py***, se desea que la plantilla home.html aparezca como página de inicio. Para realizar el registro se debe crear una vista y URL. Para garantizar que las rutas de URL sean consistentes, deben quedar alojadas en ***account/***, por lo que las URL finales serán ***/account/login, /account/logout y /account/signup***.

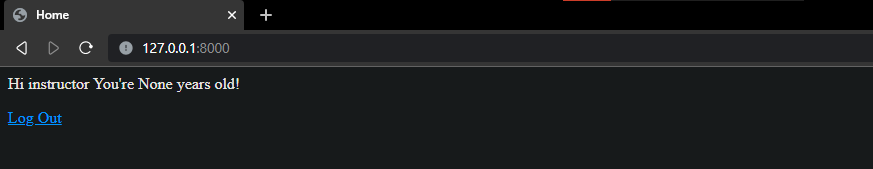


1. Iniciamos el servidor web para verificar lo realizado hasta el momento
   * 1. ***python manage.py runserver***

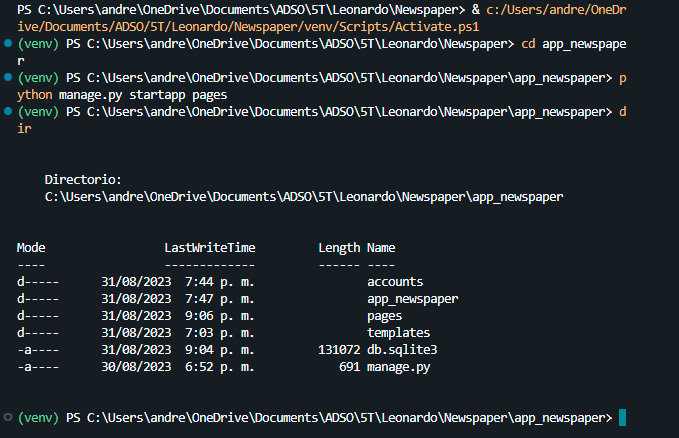


* + 1. <http://127.0.0.1:8000/>

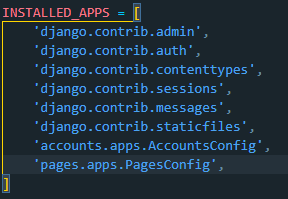




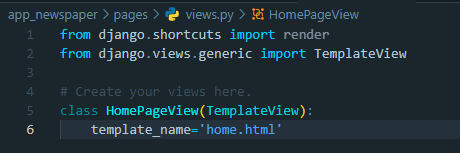
1. Creamos la aplicación ***pages*** para gestionar la carga del ***template home.html***; reemplazando la vista genérica ***TemplateView*** creada directamente en ***URLconf***, invocando el método ***as\_view().***



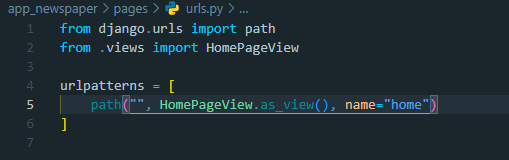
1. Agregamos la aplicación ***pages*** a nuestra lista ***INSTALLED\_APPS*** en el archivo ***app\_newspaper/settings.py***



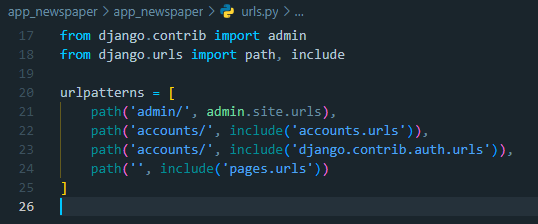
1. Modificamos el archivo ***pages/views.py*** para usar heredar la vista genérica ***TemplateView*** y poder proporcionar nuevos valores o métodos.



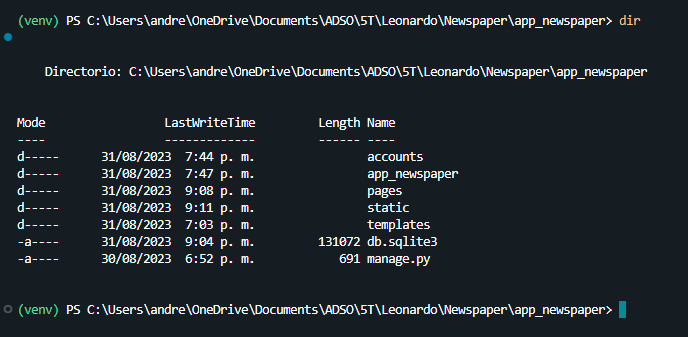
1. Creamos el archivo ***pages/urls.py*** para agregar la página de inicio.



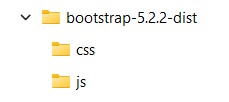
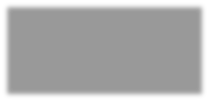
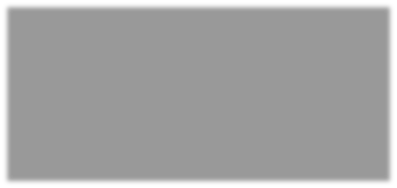
1. Actualizamos el archivo ***app\_newspaper/urls.py*** agregando el ***URLConfig*** de la aplicación ***pages***



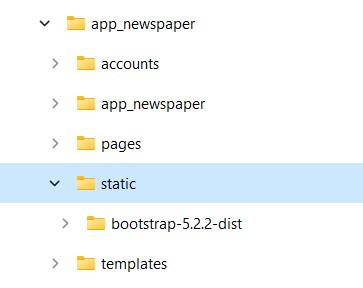
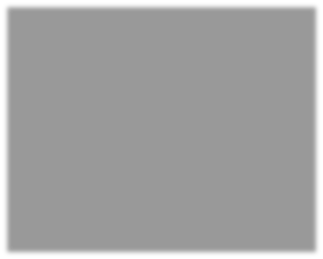
1. Creamos la carpeta ***app\_newspaper/static*** que contendrá todos los archivos estáticos (imágenes, archivos JS y CSS) incluidos los de Bootstrap.



1. Descargamos el código compilado, listo para usar, de ***Bootstrap v5.2.2*** con el fin de colocarlo fácilmente en el proyecto.



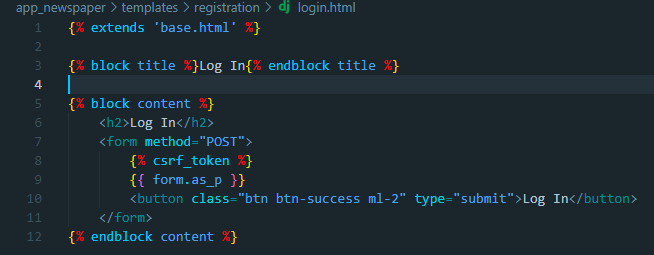
1. Ubicamos la carpeta de Bootstrap-5.2.2-dist dentro de la carpeta del proyecto static



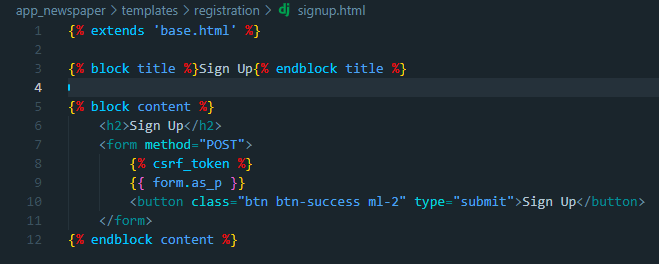
1. Modificamos el archivo ***app\_newspaper/settings.py*** para proporcionar las variables ***STATIC\_URL*** que define la URL para usar cuando se hace referencia a archivos estáticos ubicados en ***STATIC\_ROOT*** y ***STATICFILES\_DIRS*** que define las ubicaciones adicionales que atravesará la aplicación ***staticfiles*** si el buscador ***FileSystemFinder*** está habilitado.



1. Modificamos la plantilla ***templates/registration/login.html***, para dar diseño ***css***

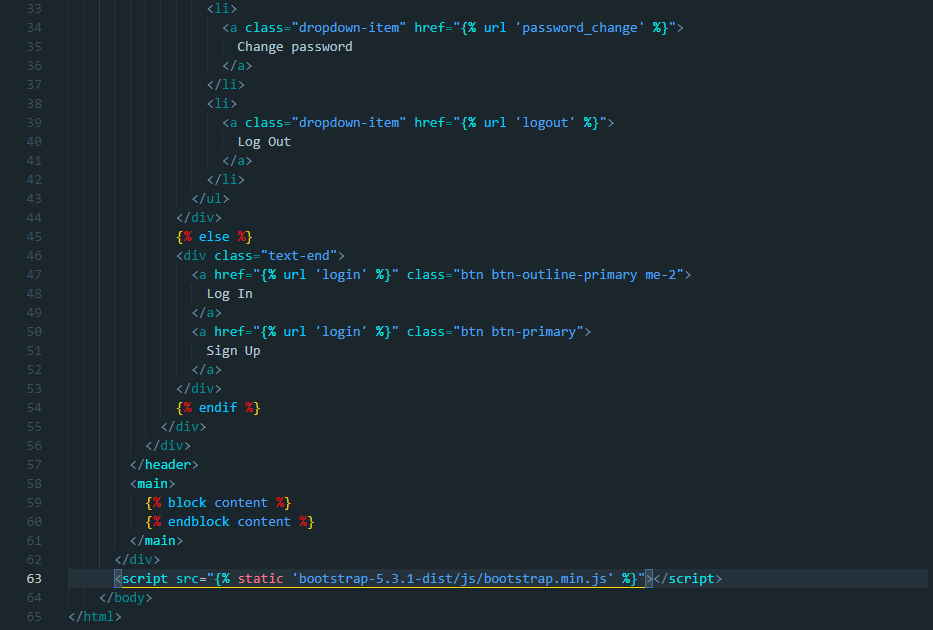


1. Modificamos la plantilla ***templates/registration/signup.html***, para dar diseño ***css***

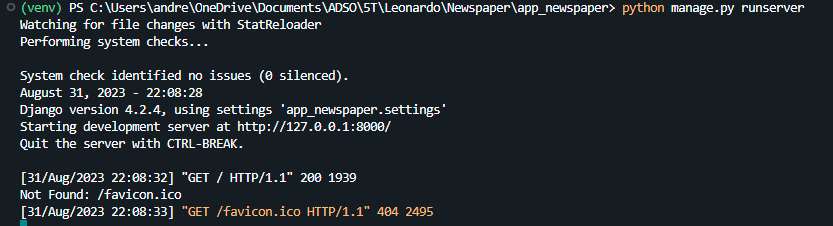


1. Modificamos el archivo ***templates/base.html*** para contener los diseños ***css*** de Bootstrap

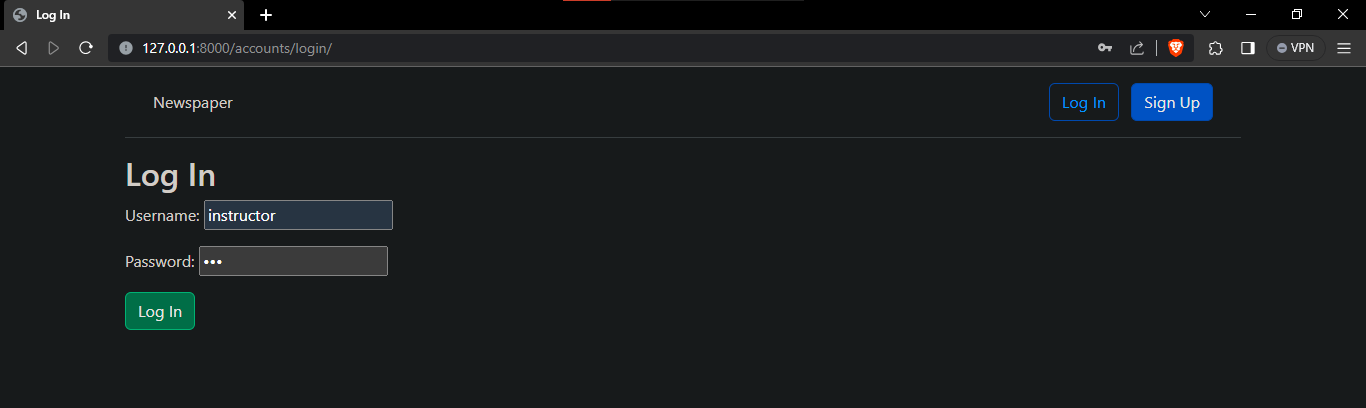


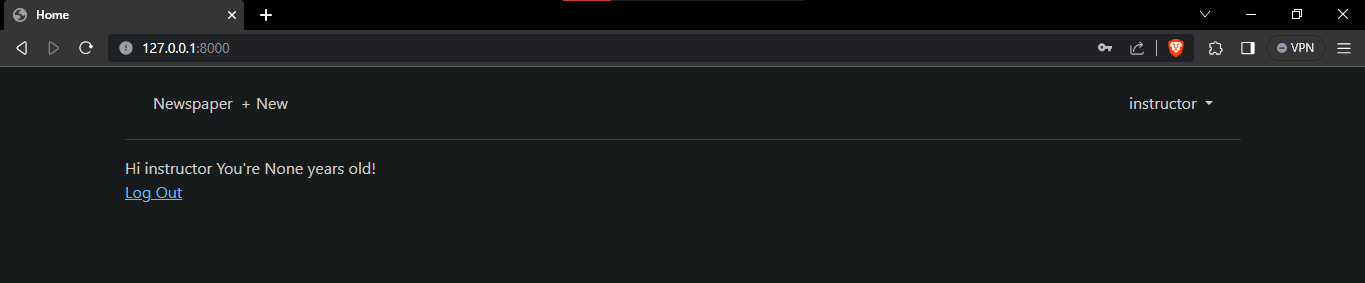


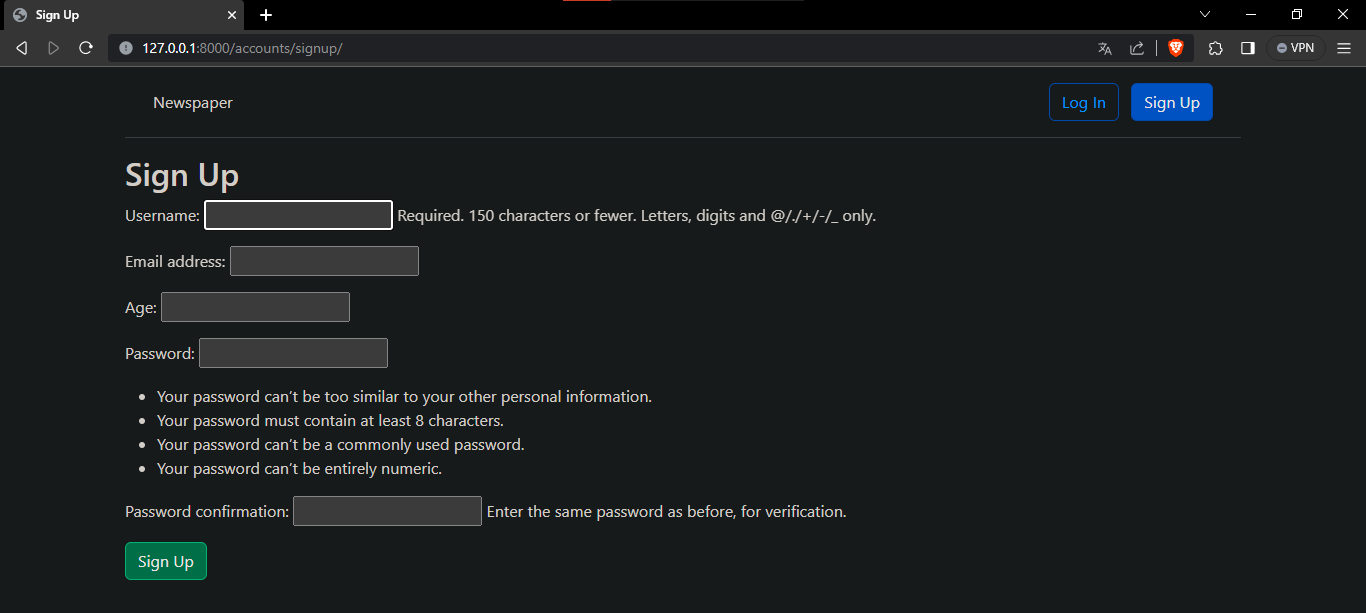
1. Iniciamos el servidor web para verificar lo realizado hasta el momento
   * 1. ***python manage.py runserver***



* + 1. <http://127.0.0.1:8000/>



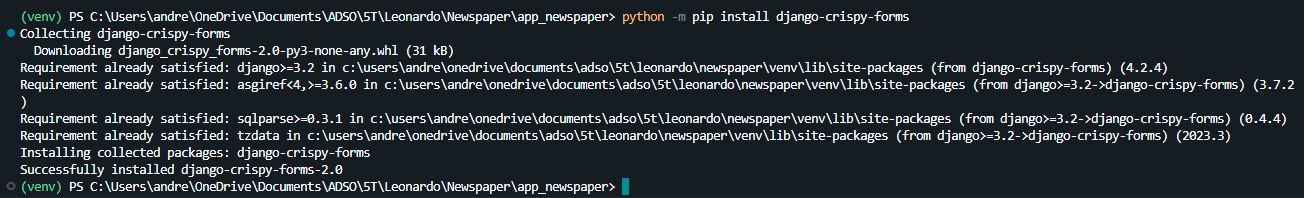




1. El paquete ***Django-crispy-forms*** es una aplicación que ayuda a administrar los formularios de Django, permitiendo ajustar las propiedades de los formularios (como el método, el botón de envío o las clases de CSS) en el backend sin tener que volver a escribirlas en la plantilla.

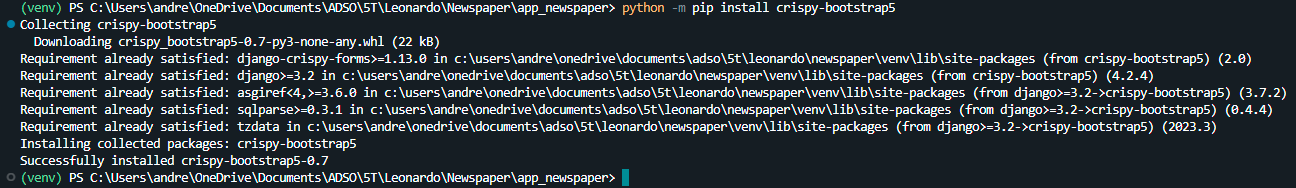
***Django-crispy-forms*** hace que la representación de la plantilla de formulario sea realmente fácil, afectando la apariencia y el estilo del formulario al agregar ***clases css*** a todo el formulario, así como a campos individuales, sus etiquetas o botones.

Instalar el paquete ejecutando el comando: ***python -m pip install django-crispy-forms***

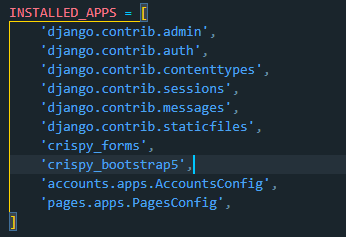


1. El paquete ***crispy-bootstrap5*** es un paquete de ***plantillas de Bootstrap 5*** para ser utilizadas por ***django-crispy-forms***.

Instalar el paquete ejecutando el comando: ***python -m pip install crispy-bootstrap5***



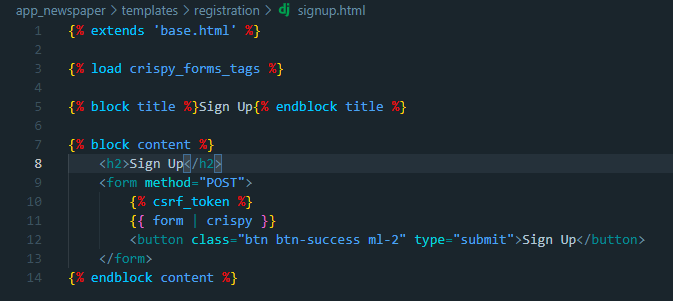
1. Agregamos los paquetes ***crispy\_forms y crispy\_bootstrap5*** a nuestra lista ***INSTALLED\_APPS*** en el archivo ***app\_newspaper/settings.py***



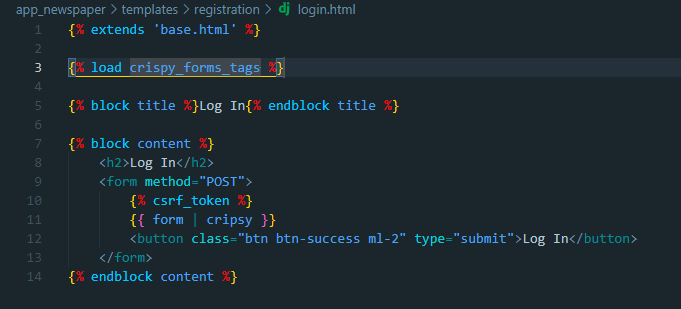
1. Agregamos dos líneas en la parte inferior del archivo ***app\_newspaper/settings.py***



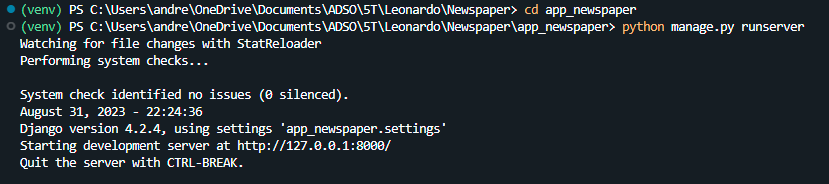
1. Modificamos la plantilla ***templates/registration/signup.html,*** para cargar la etiqueta ***crispy\_forms\_tags*** en la parte superior y luego cambiamos ***{{ form.as\_p }}*** por ***{{ form|crispy }}***.



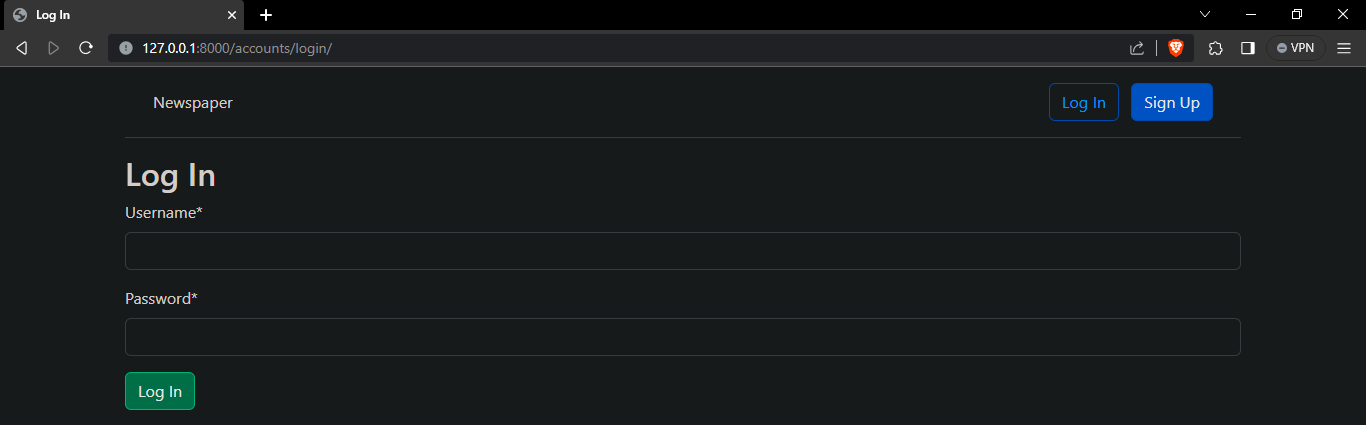
1. Modificamos la plantilla ***templates/registration/login.html,*** para cargar la etiqueta ***crispy\_forms\_tags*** en la parte superior y luego cambiamos ***{{ form.as\_p }}*** por ***{{ form|crispy }}***.

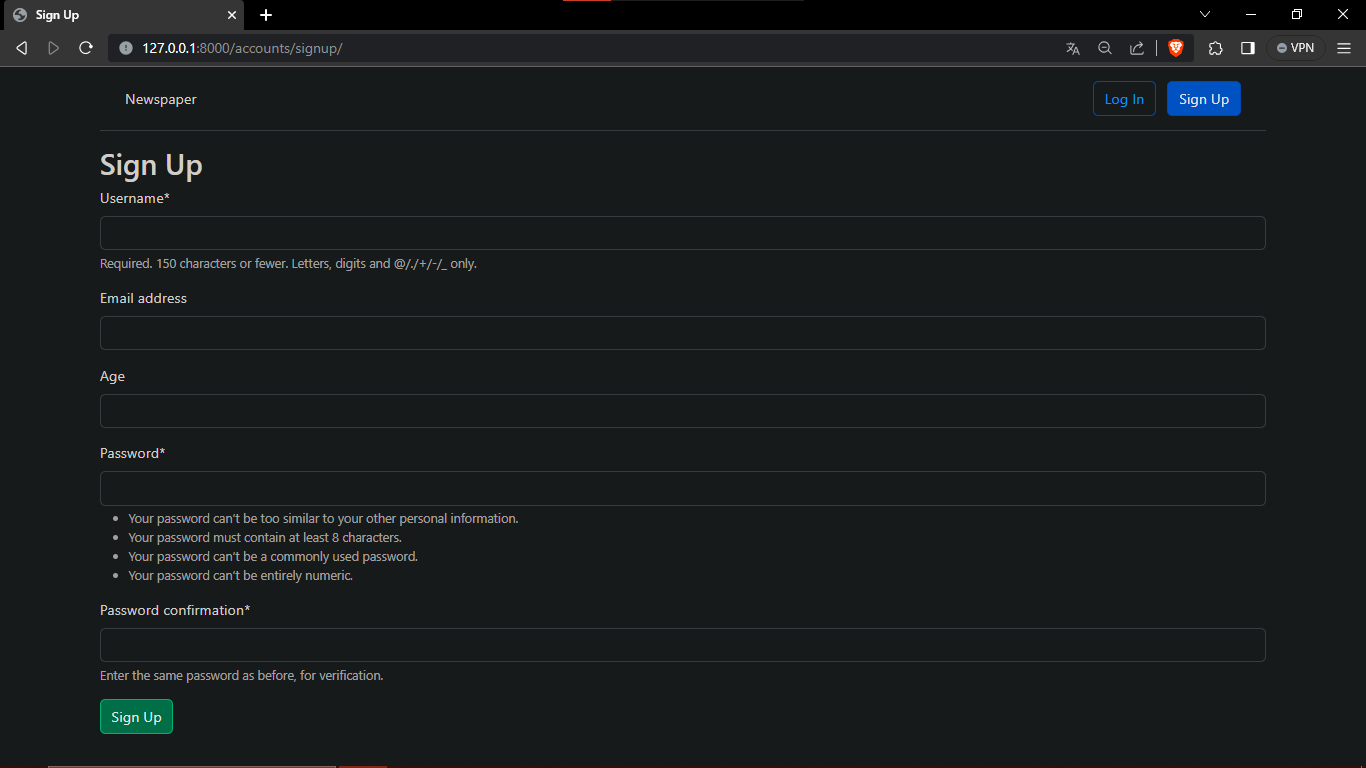


1. Iniciamos el servidor web para verificar lo realizado hasta el momento
   * + 1. ***python manage.py runserver***

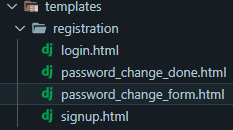


* + - 1. <http://127.0.0.1:8000/>

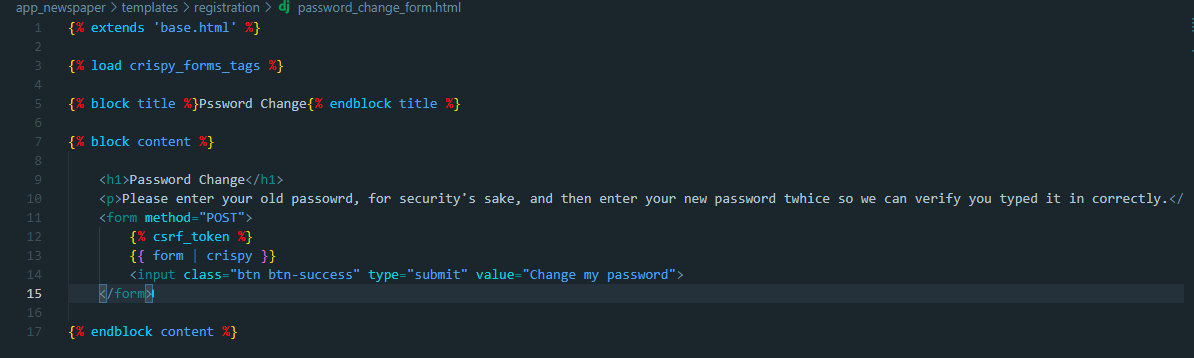




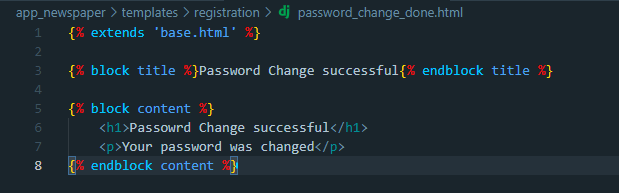
1. Para personalizar el cambio de contraseña creamos dos nuevas plantillas:
   * + 1. ***templates/registration/password\_change\_form.html***
       2. ***templates/registration/password\_change\_done.html***



1. Modificamos el archivo ***templates/registration/password\_change\_form.html***



1. Modificamos el archivo ***templates/registration/password\_change\_done.html***



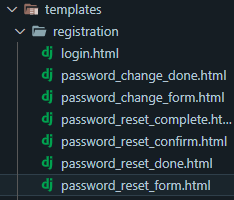
1. El restablecimiento de contraseña maneja el caso común de usuarios que olvidan sus contraseñas; los pasos son muy similares a configurar el cambio de contraseña y la única configuración requerida es decirle a Django cómo enviar correos electrónicos. Después de todo, un usuario solo puede establecer una contraseña si tiene acceso al correo electrónico vinculado a la cuenta.

En la parte inferior del archivo ***app\_newspaper/settings.py*** , agregar el siguiente cambio:

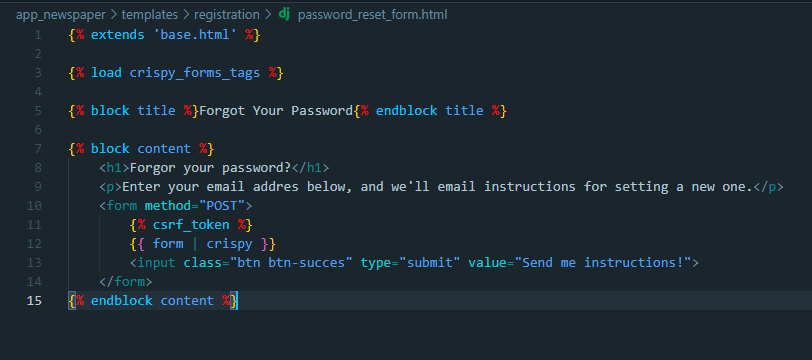
***EMAIL\_BACKEND = "django.core.mail.backends.console.EmailBackend"***



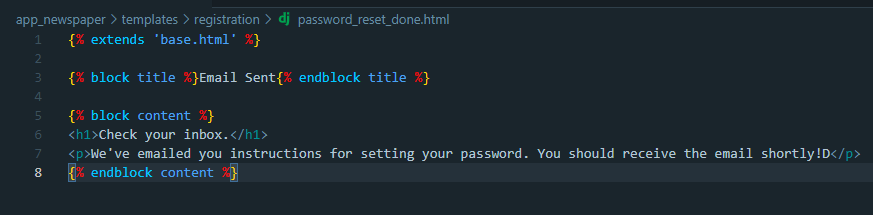
1. Para personalizar el cambio de contraseña creamos cuatro nuevas plantillas:
   * 1. ***templates/registration/password\_reset\_form.html***
     2. ***templates/registration/password\_reset\_done.html***
     3. ***templates/registration/password\_reset\_confirm.html***
     4. ***templates/registration/password\_reset\_complete.html***



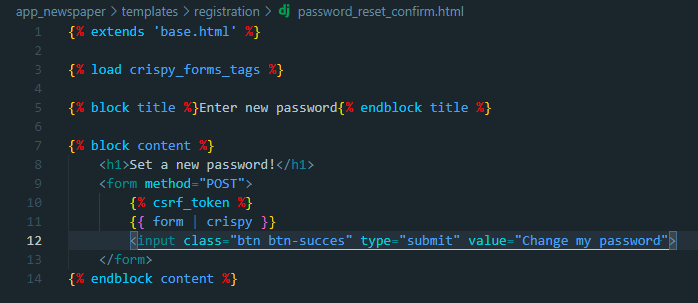
1. Modificamos el archivo ***templates/registration/password\_reset\_form.html***



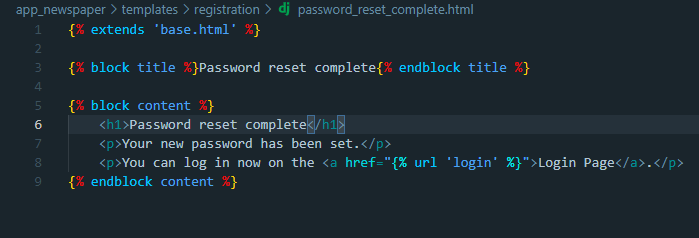
1. Modificamos el archivo ***templates/registration/password\_reset\_done.html***

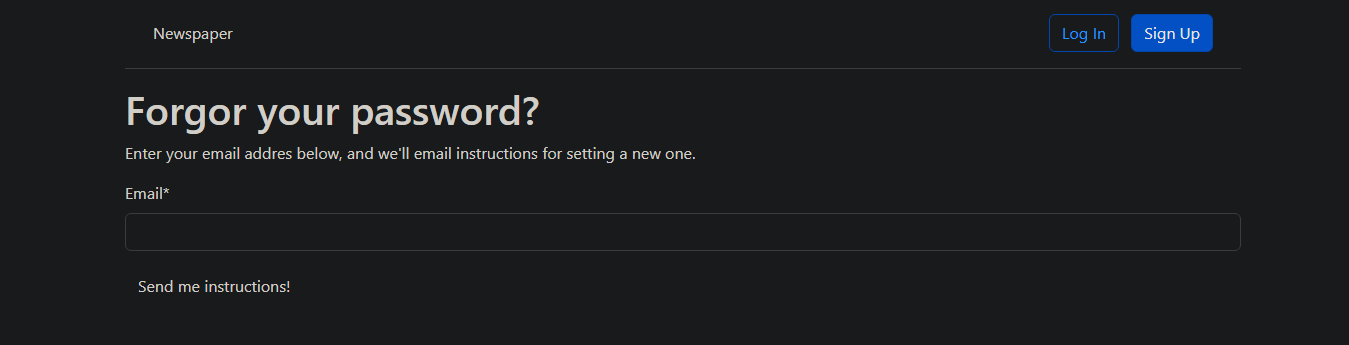
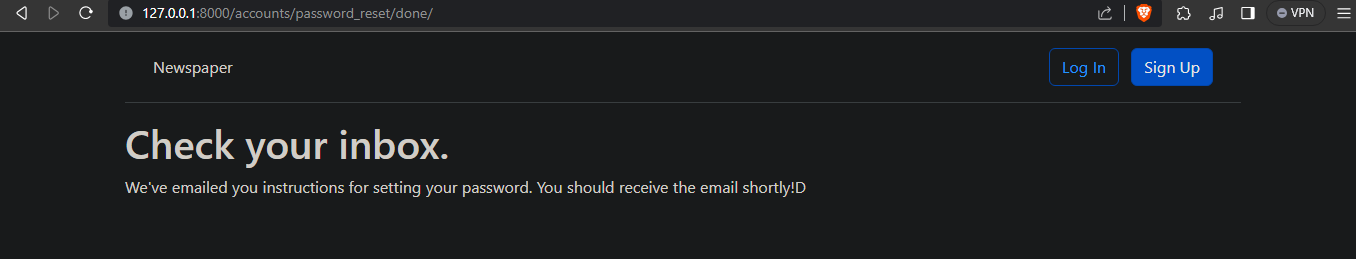
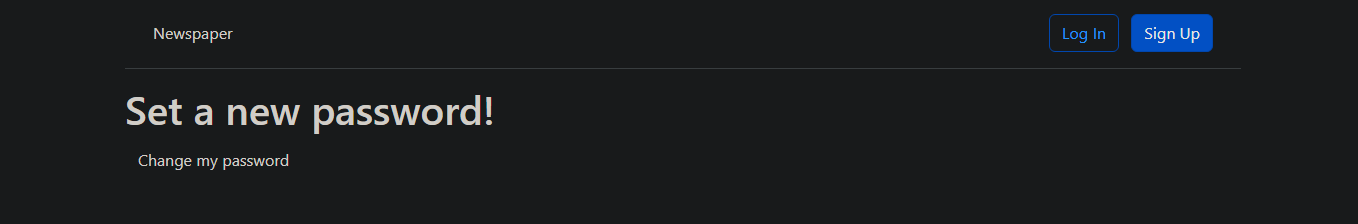
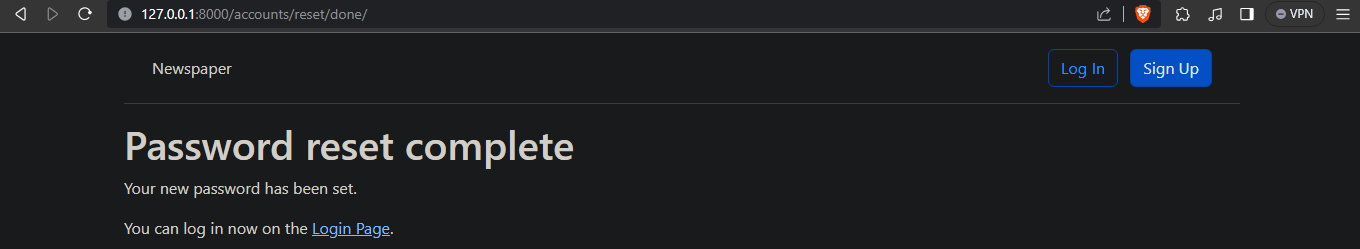


1. Modificamos el archivo ***templates/registration/password\_reset\_confirm.html***

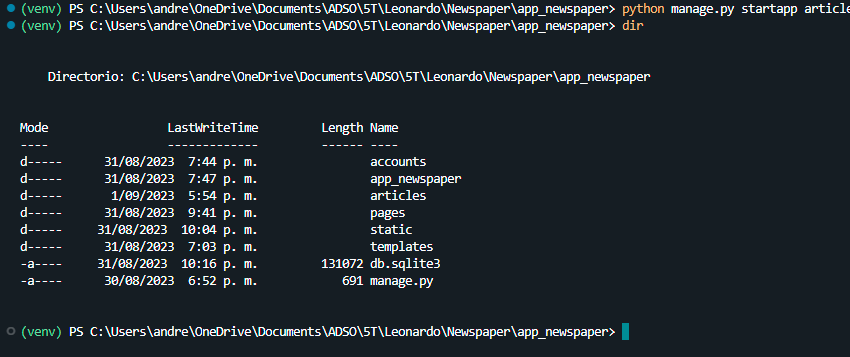


1. Modificamos el archivo ***templates/registration/password\_reset\_complete.html***

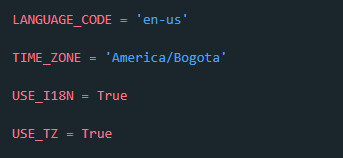
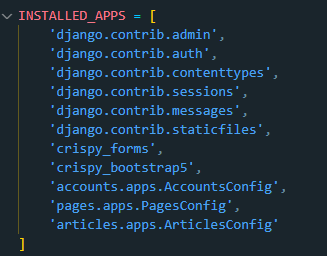


1. Iniciamos el servidor web para verificar lo realizado hasta el momento
2. 
3. 
4. 
5. 

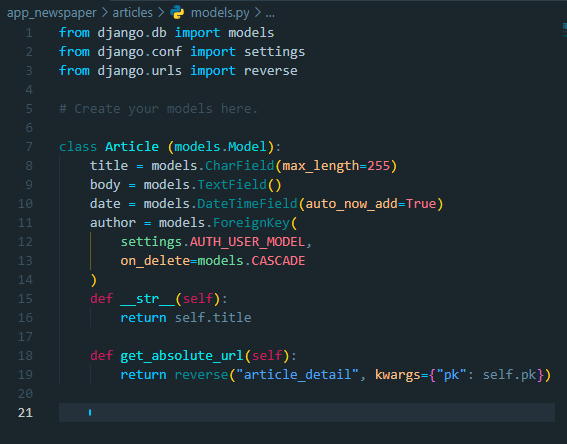
1. Creamos una aplicación ***articles*** para gestionar los artículos que se publican en el ***newspaper***



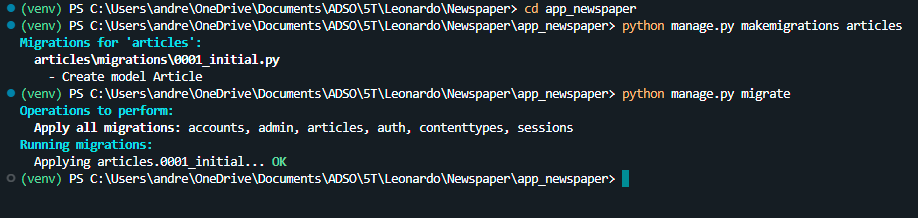
1. Modificamos el archivos ***app\_newspaper/settings.py*** para agregar la aplicación ***articles*** en ***INSTALLED\_APP***; modificamos ***TIME\_ZONE y LANGUAGE\_CODE***.



1. Definimos el modelo de ***base de datos*** que contiene cuatro campos: ***título, cuerpo, fecha y autor***. Para el campo de autor, hacer referencia al modelo de usuario personalizado ***accounts.CustomUser*** que se configuró en el archivo ***app\_newspaper/settings.py*** como ***AUTH\_USER\_MODEL***.



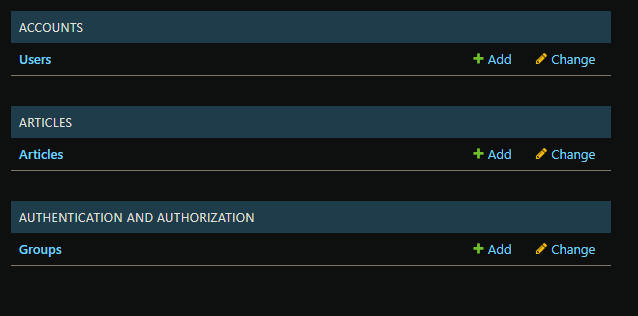
1. Como se tiene una aplicación y un modelo completamente nuevos, se debe crear un nuevo archivo de migración y luego aplicarlo a la base de datos

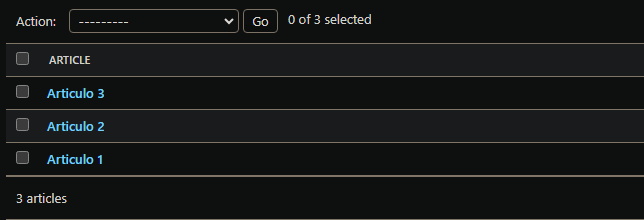


1. actualizamos ***articles/admin.py*** para que se muestre nuestra nueva tabla ***Article***



1. Iniciamos el servidor web para agregar tres (3) artículos en la tabla ***Article***; usamos la URL <http://127.0.0.1:8000/admin>





1. Creamos una nueva plantilla ***templates/article\_list.html***. Dentro de cada artículo se muestra el título, el cuerpo, el autor y la fecha. Incluso se podría proporcionar los enlaces para ***"editar" y "eliminar"***.



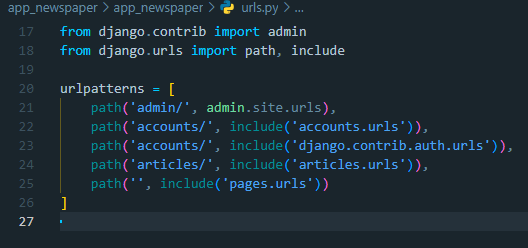
1. Creamos una vista usando el ***ListView*** genérico. Los dos campos que debemos especificar son el modelo ***Article*** y el nombre de nuestra plantilla ***article\_list.html***.



1. Creamos el archivo ***articles/urls.py***



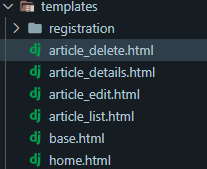
1. Agregamos un ***patrón URL*** para la aplicación ***articles*** en el archivo ***app\_newspaper/urls.py***



1. Activar el servidor con ***python manage.py runserver*** y consultar la página en [***http://127.0.0.1:8000/articles/***](http://127.0.0.1:8000/articles/)



1. El siguiente paso es agregar detalles, editar y eliminar opciones para los artículos; para lograrlo creamos tres nuevas plantillas:
   * 1. ***templates/article\_details.html***
     2. ***templates/article\_edit.html***
     3. ***templates/article\_delete.html***



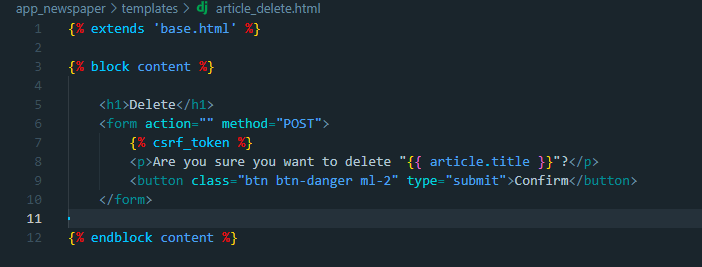
1. Modificamos el archivo ***templates/article\_detail.html*** que mostrará el título, la fecha, el cuerpo y el autor con enlaces para editar y eliminar.



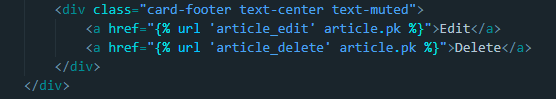
1. Modificamos el archivo ***templates/article\_edit.html*** y usamos el estilo de botón de Bootstrap.



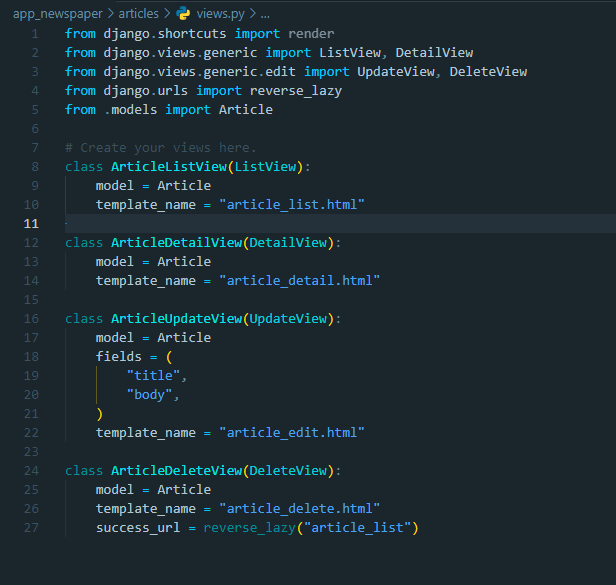
1. Modificamos el archivo ***templates/article\_delete.html***



1. Modificamos el archivo ***templates/article\_list.html*** para agregar las rutas (URL) a los atributos ***href.***



1. Modificamos el archivo ***articles/views.py*** para usar las vistas genéricas basadas en clases de Django para ***DetailView, UpdateView y DeleteView***.
   * 1. Vista detallada: listar el modelo y el nombre de la plantilla.
     2. Vista de actualización/edición: agregar los campos que se pueden cambiar.
     3. Vista de eliminación: redirección a dónde enviar después de eliminar la entrada.



Nota: Cuando se utiliza ***reverse()*** el resultado es una cadena (URL inversa), pero cuando se usa ***reverse\_lazy()***, el resultado es una función, que será llamada al final, una vez todos los ficheros de configuración se hayan cargado; para entonces, ya será conocido el nombre

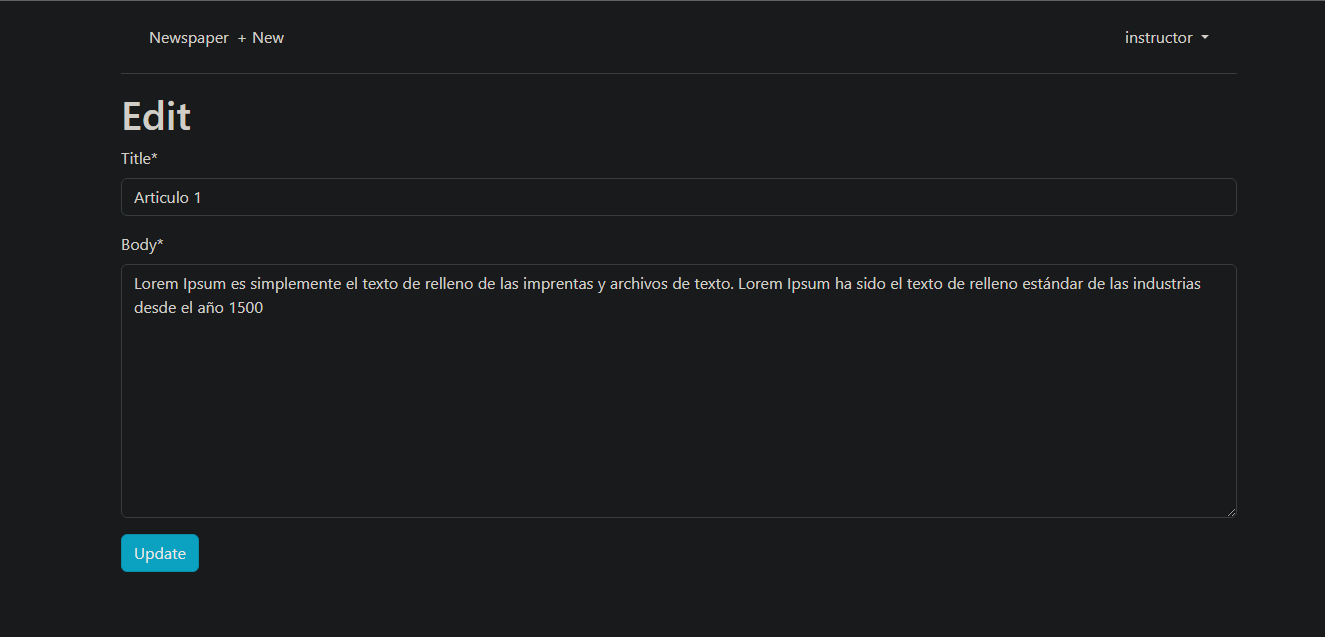
***"article\_list"*** y la función ***reverse\_lazy()*** podrá usar reverse() para generar su respuesta.

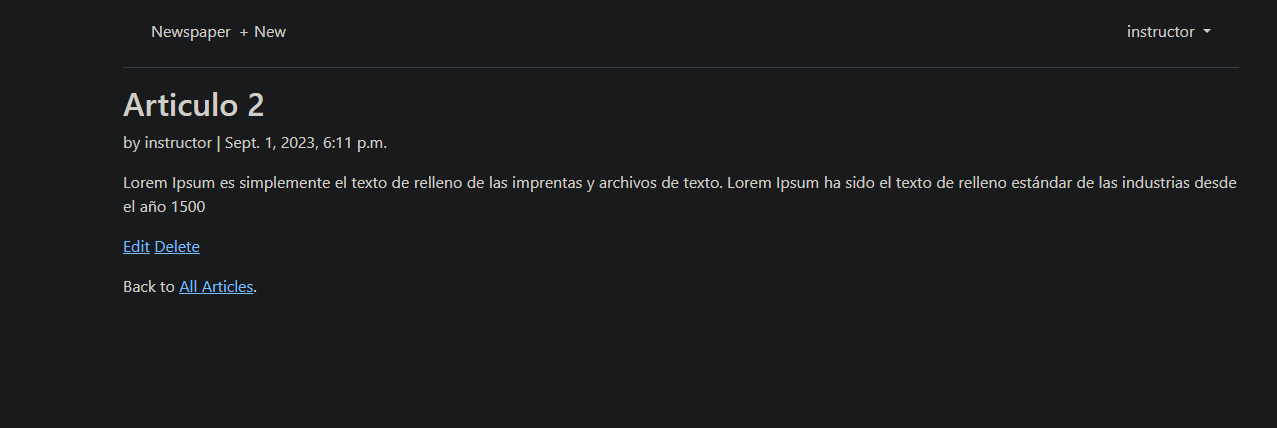
1. Modificamos el archivo ***articles/urls.py*** para agregar los ***path*** para las vistas ***ArticleDetailView, ArticleUpdateView y ArticleDeleteView.***

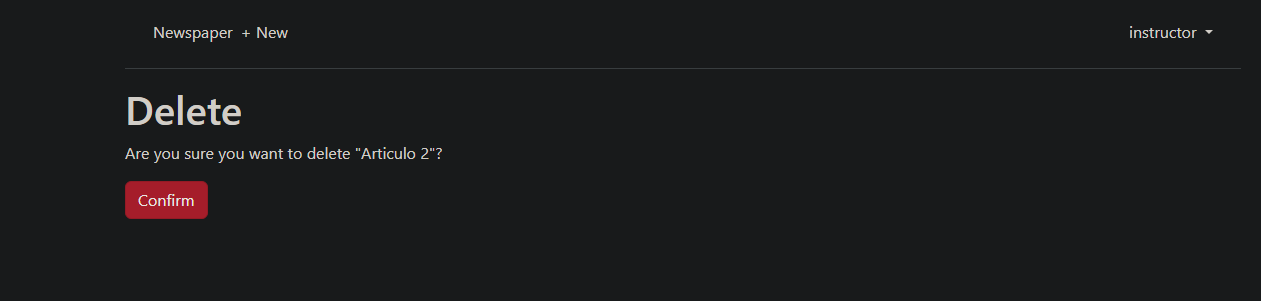


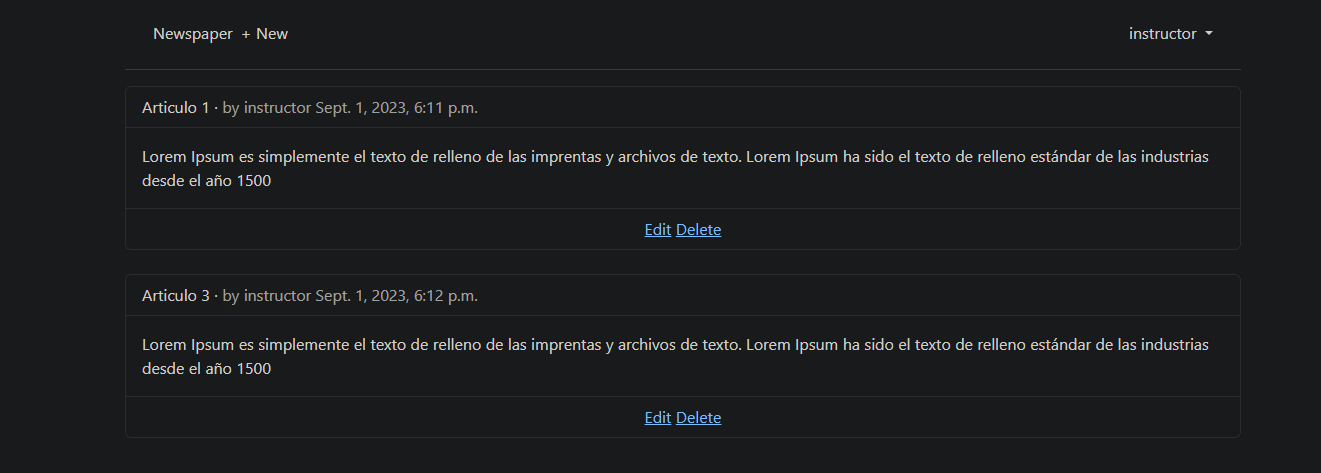
1. Activar el servidor con ***python manage.py runserver*** y consultar la página en [***http://127.0.0.1:8000/articles/***](http://127.0.0.1:8000/articles/)



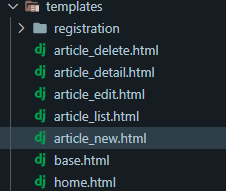








1. Para completar la nueva función de creación de nuevos artículos, agregar una nueva plantilla denominada ***templates/article\_new.html***



1. Modificamos el archivo ***templates/article\_new.html***



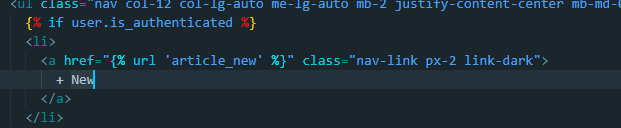
1. En el archivo ***articles/views.py*** agregamos ***CreateView*** a las importaciones y creamos una nueva clase ***ArticleCreateView*** que especifica el modelo, plantilla y los campos disponibles.



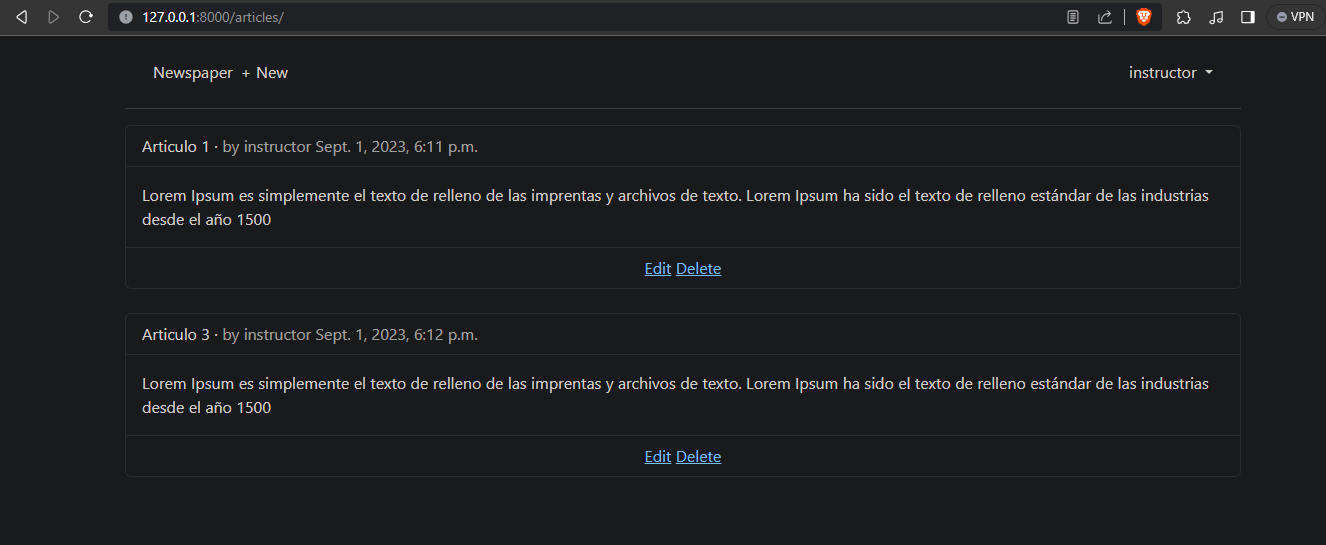
1. Modificamos el archivo ***articles/urls.py*** para agregar el ***path*** para las vista ***ArticleCreateView***

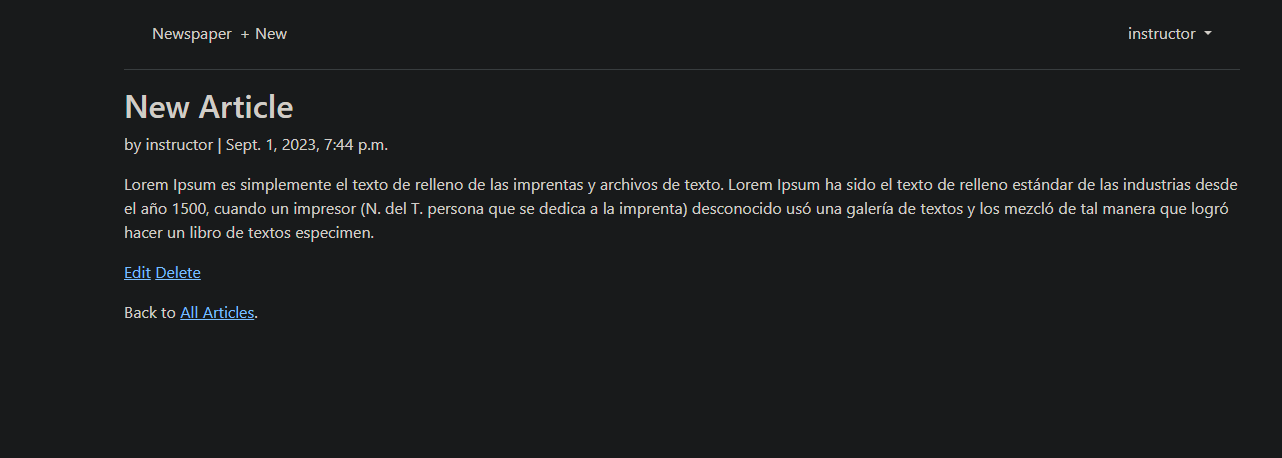


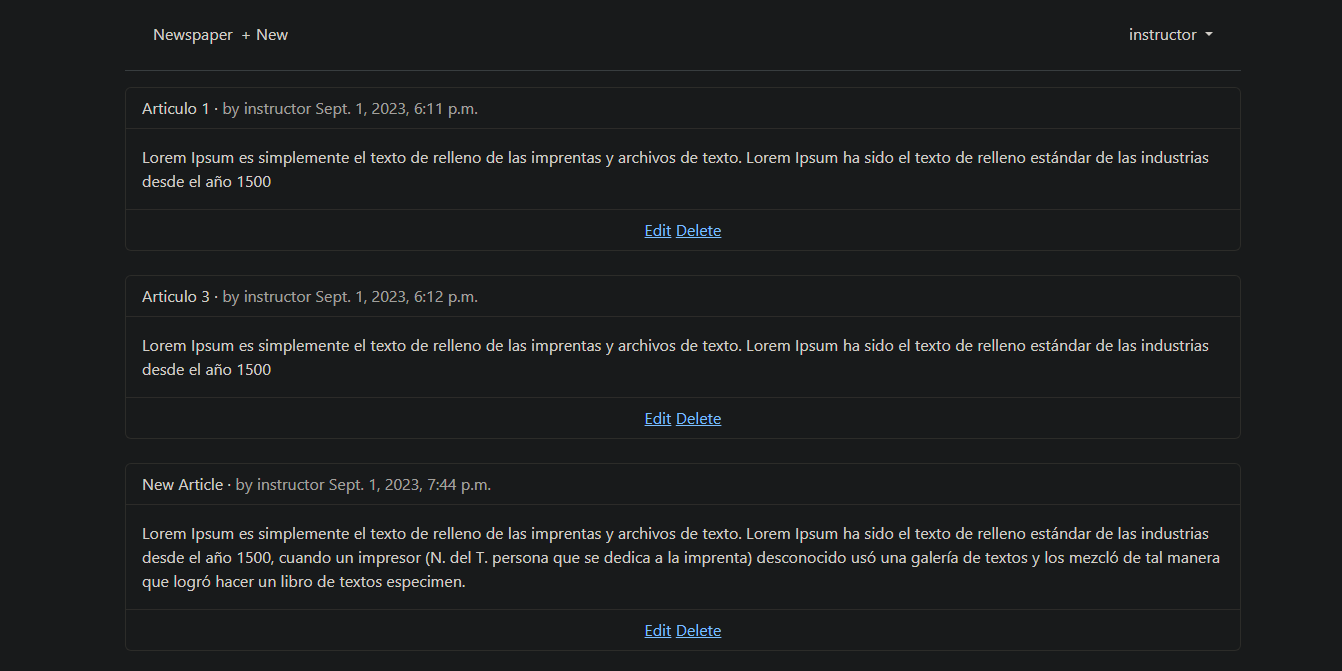
1. Modificamos el archivo ***templates/base.html***, agregando el enlace URL para crear nuevos artículos en la barra de navegación con el fin de ser accesible en cualquier parte del sitio para los usuarios registrados.



1. Activar el servidor con ***python manage.py runserver*** y consultar la página en <http://127.0.0.1:8000/articles/>







**PROYECTO: NEWSPAPER – PERMISOS -AUTORIZACIONES**

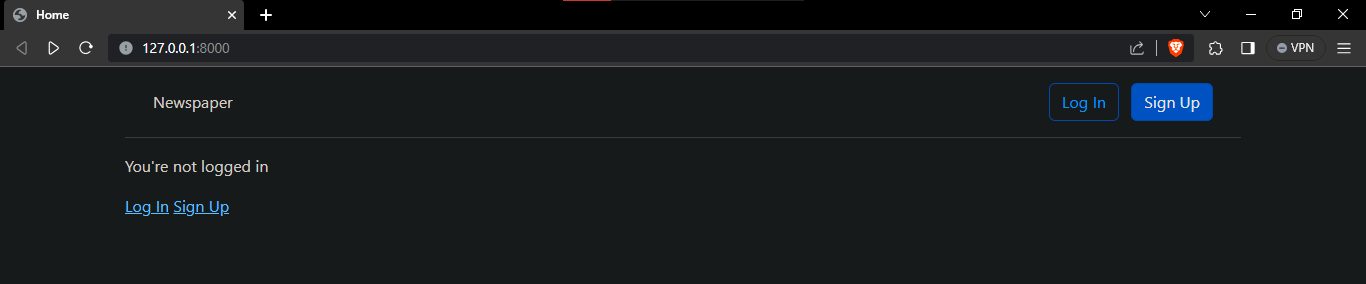
**OBJETIVO**: Configurar el proyecto llamado Newspaper para que solo el usuario propietario pueda gestionar sus artículos.

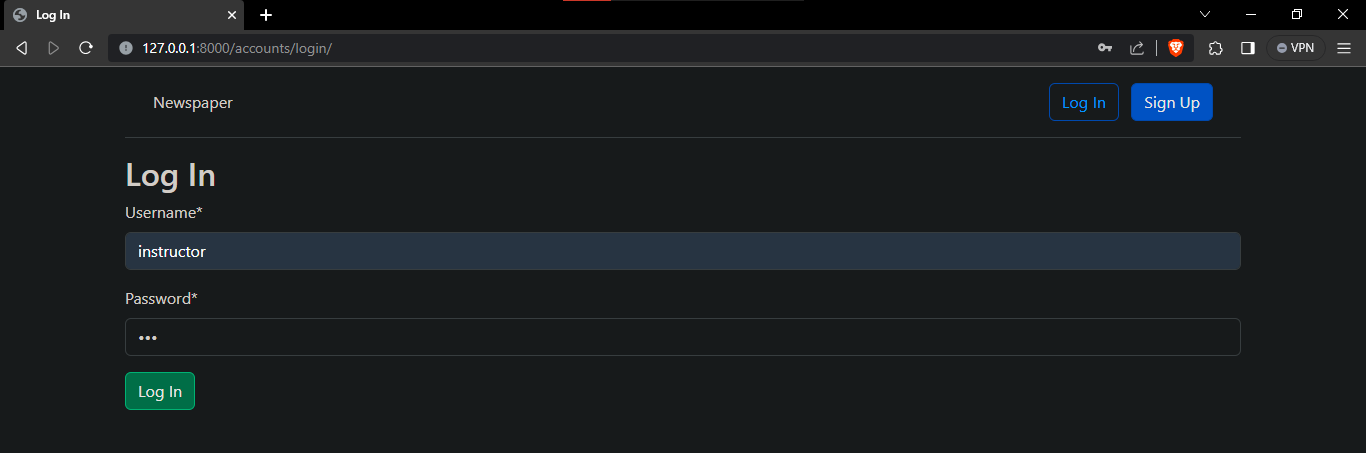
Gestionar comentarios a cada uno de los artículos

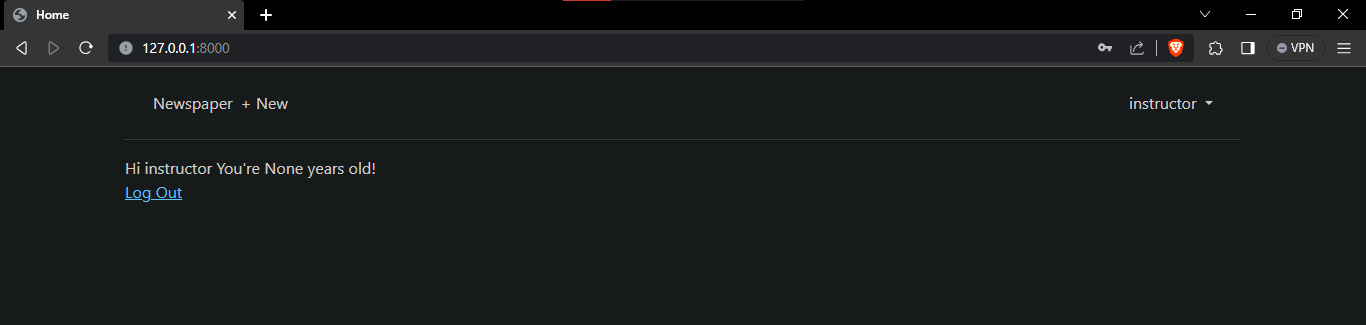
1. Hasta el momento, al crear un nuevo artículo se puede seleccionar el autor de la lista de usuarios existentes. En su lugar, debe establecerse automáticamente en el usuario que ha iniciado sesión actualmente. Se puede modificar la vista predeterminada ***CreateView*** eliminando autor de los campos y configurándolo automáticamente a través del ***método*** ***form\_valid***.

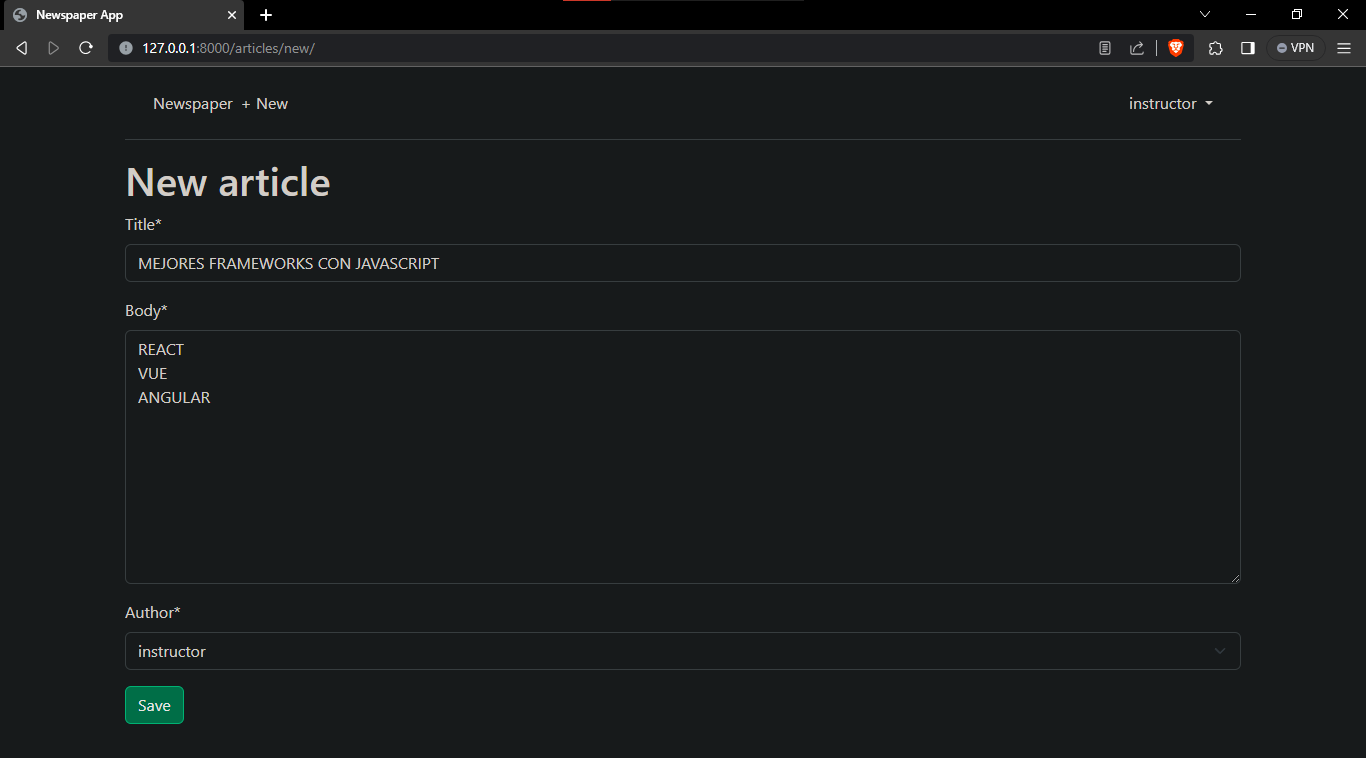


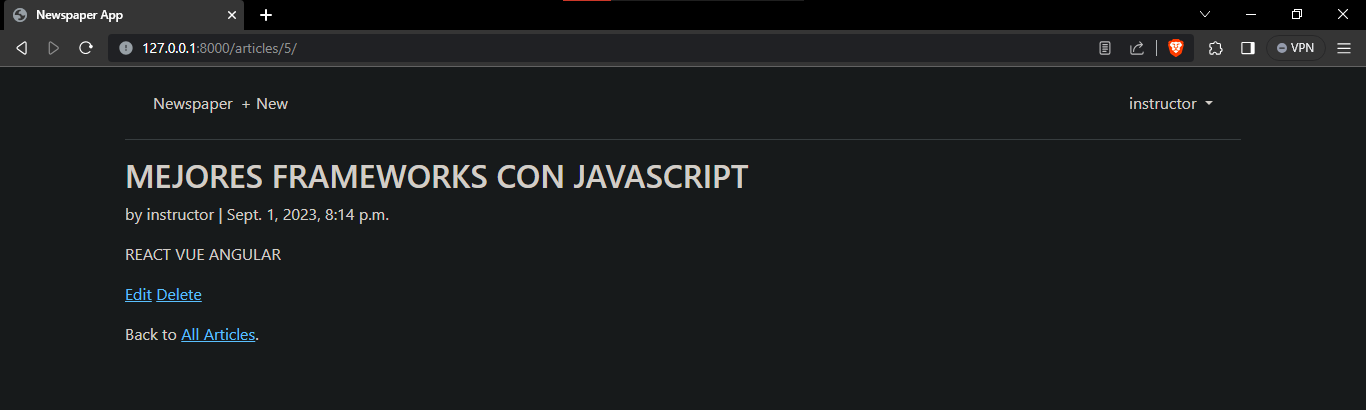
1. Activar el servidor con ***python manage.py runserver*** y consultar la página en <http://127.0.0.1:8000/articles/>



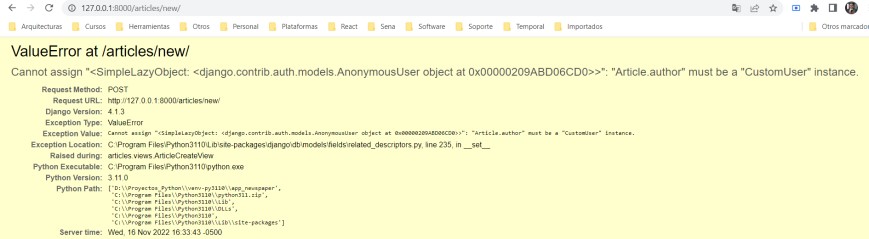
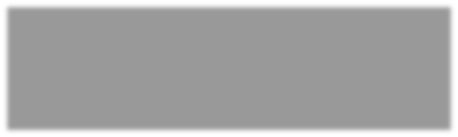








1. Si el usuario no esta autenticado y se hace una consulta a la URL [http://127.0.0.1:8000/articles/new/;](http://127.0.0.1:8000/articles/new/) al momento de grabar el artículo, se generará un error por esta causa.



Para garantizar que los usuarios eviten consultar URLS sin haber previamente iniciado sesión se implementa el ***mixin*** (es una implementación de una o varias funcionalidades para una clase) ***LoginRequiredMixin***; si una vista usa este ***mixin***, todas las solicitudes de usuarios no autenticados se redirigirán a la página de inicio de sesión o se mostrará una respuesta ***HTTP 403 Forbidden error***, según el parámetro ***raise\_exception***.

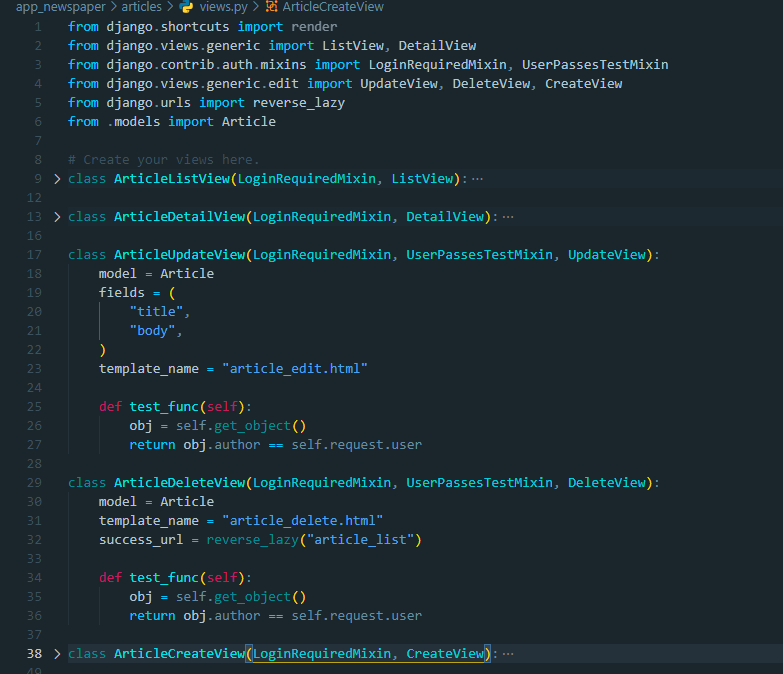
Por lo anterior en todas las vistas creadas en el archivo ***articles/views.py***, debemos incluir el ***mixin LoginRequiredMixin*** como primer parámetro.

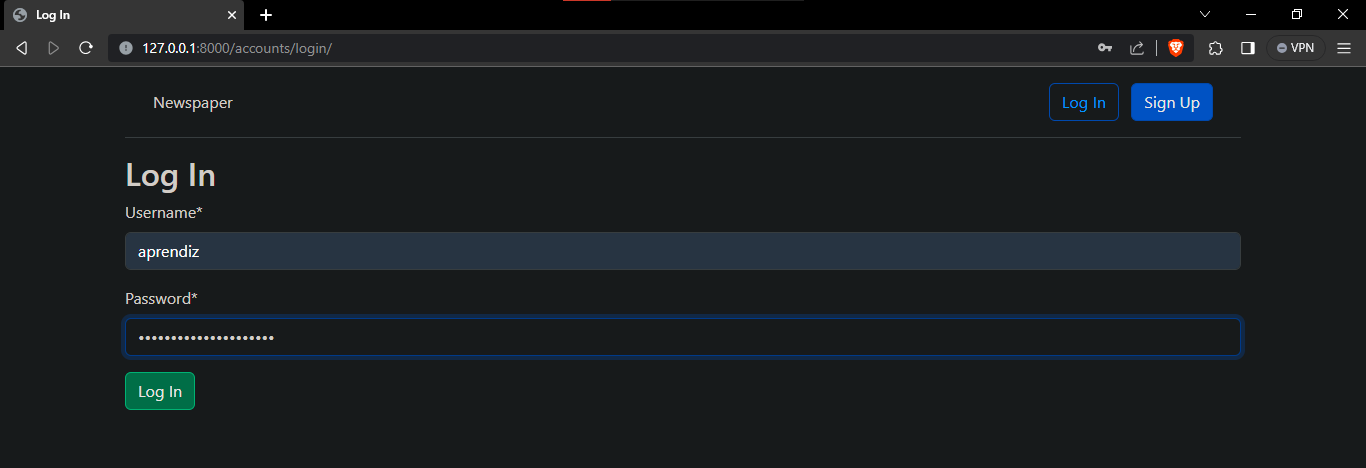
1. Cualquier usuario que haya iniciado sesión puede realizar cambios en cualquier artículo; lo ideal es restringir este acceso para que solo el autor de un artículo tenga este permiso. Django implementa el ***mixin UserPassesTestMixin*** para ese propósito.

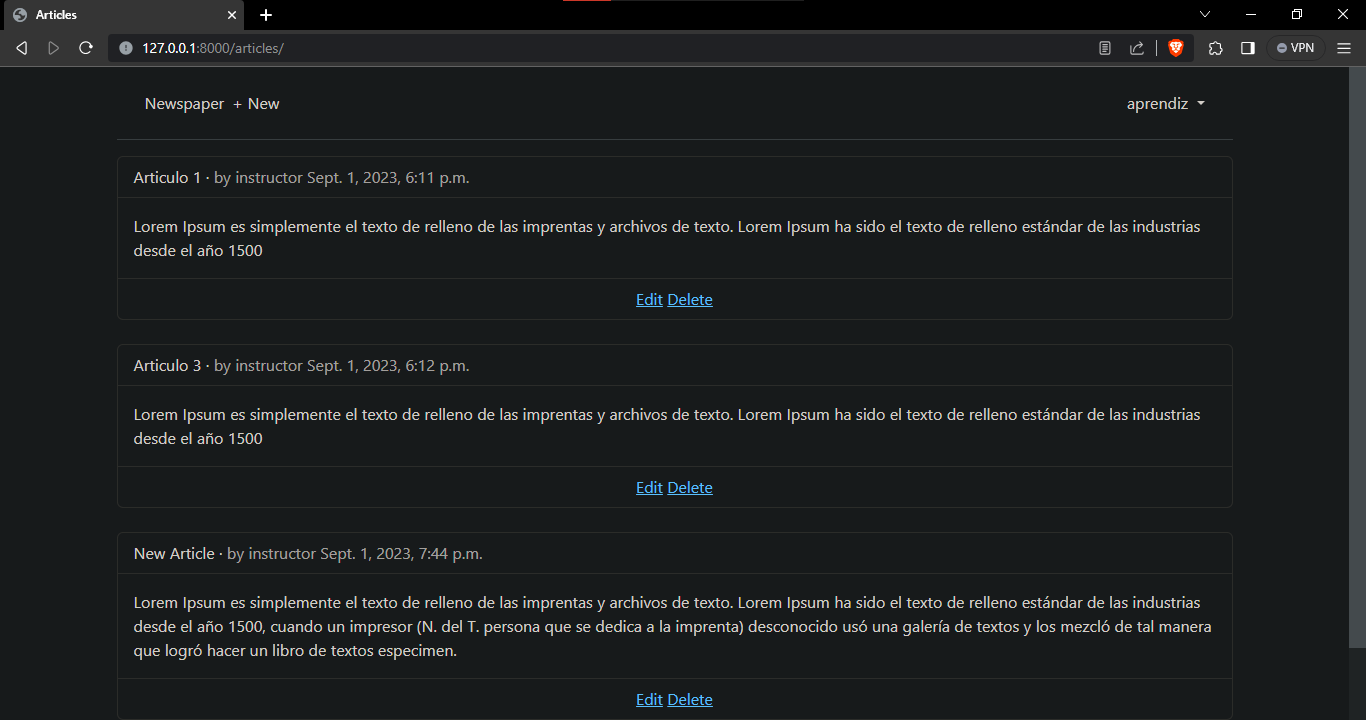
Se debe redefinir el método ***test\_func()*** para garantizar que el usuario que inicio sesión es el mismo que esta intentando editar o eliminar el articulo seleccionado.

Acorde con lo anterior en las vistas ***ArticleUpdateView y ArticleDeleteView*** definidas en el archivo ***articles/views.py***, debemos incluir el ***mixin UserPassesTestMixin*** como segundo parámetro.

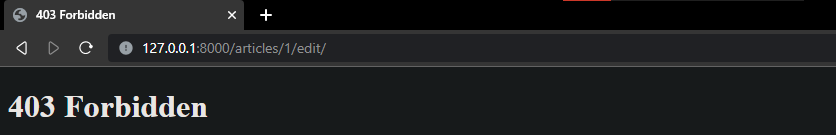


1. Activar el servidor con ***python manage.py runserver*** y consultar la página en <http://127.0.0.1:8000/articles/>

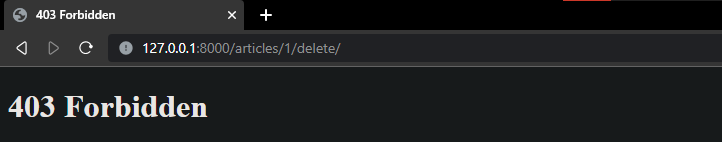




Intenta editar el articulo “Gabriel Garcia Marquez” del autor Instructor



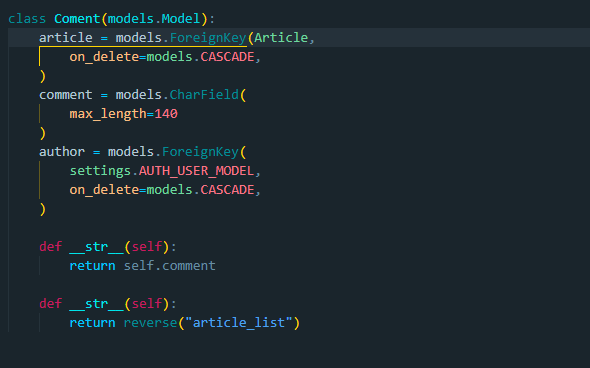
Intenta borrar el articulo “Gabriel Garcia Marquez” del autor Instructor



PROYECTO: NEWSPAPER – PERMISOS -AUTORIZACIONES

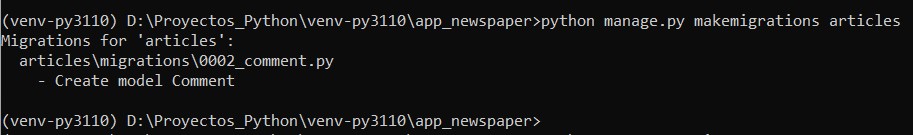
OBJETIVO: Gestionar comentarios a cada uno de los artículos

1. Para gestionar los comentarios creamos una nueva *tabla* llamada ***Comment*** que contendrá una *clave externa de muchos a uno* con la ***tabla Article***. Para realizar esta tarea modificamos el archivo ***articles/models.py*** agregando la nueva clase.

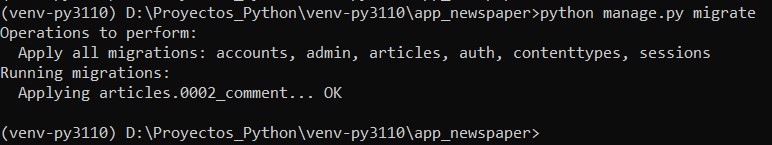


1. Al modificar el modelo de la aplicación ***Articles,*** debemos volver a ejecutar los comandos:

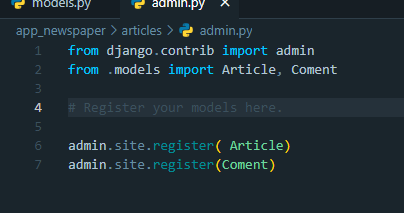
# a. Python manage.py makemigrations articles



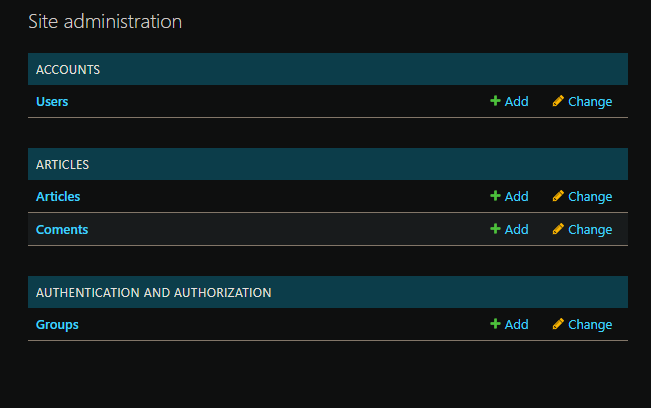
# b. Python manage.py migrate

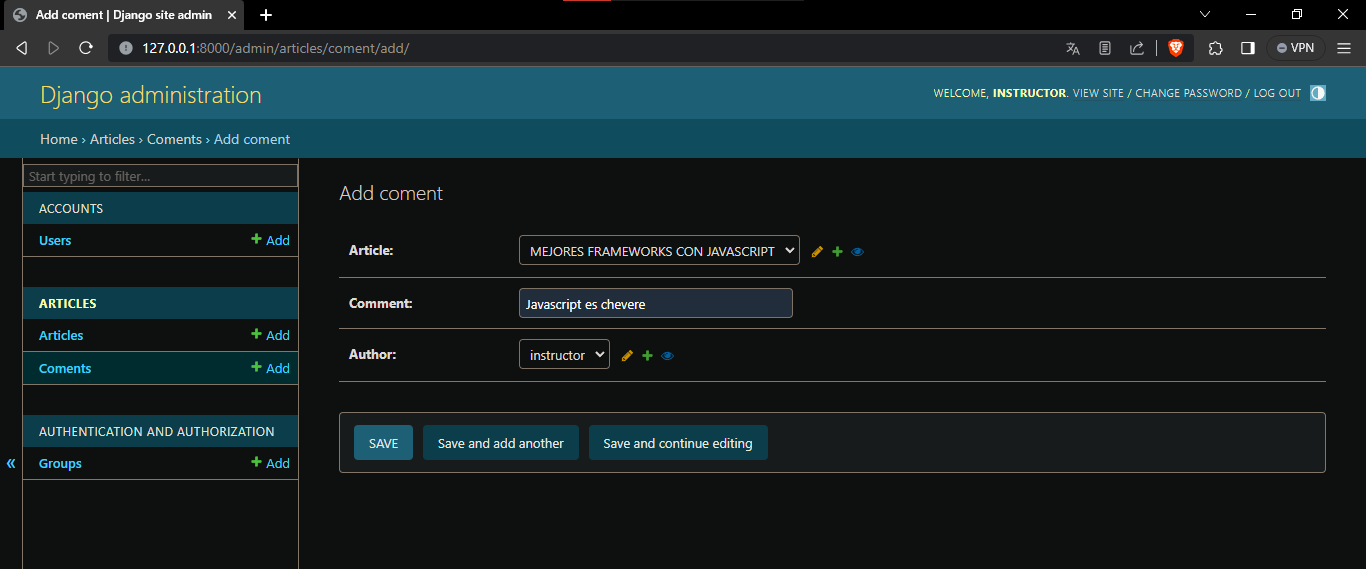


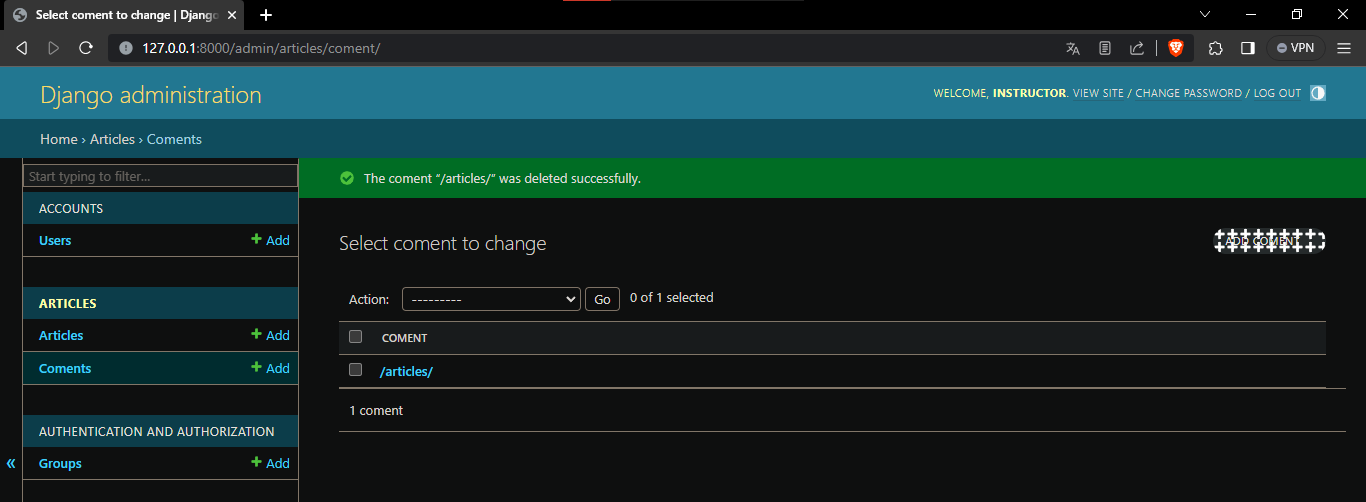
1. Agregamos el nuevo ***modelo******Comment*** al gestor de datos de Django; para ello, modificamos el archivo ***articles/admin.py***



1. Iniciar el servidor con ***python manage.py runserver***. Ir a la página principal [***http://127.0.0.1:8000/admin/***](http://127.0.0.1:8000/admin/)



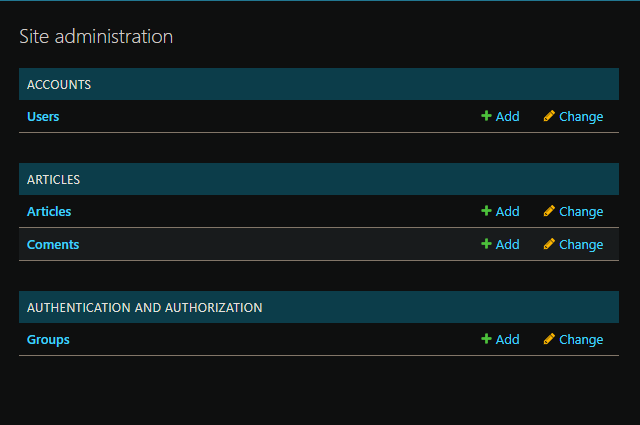


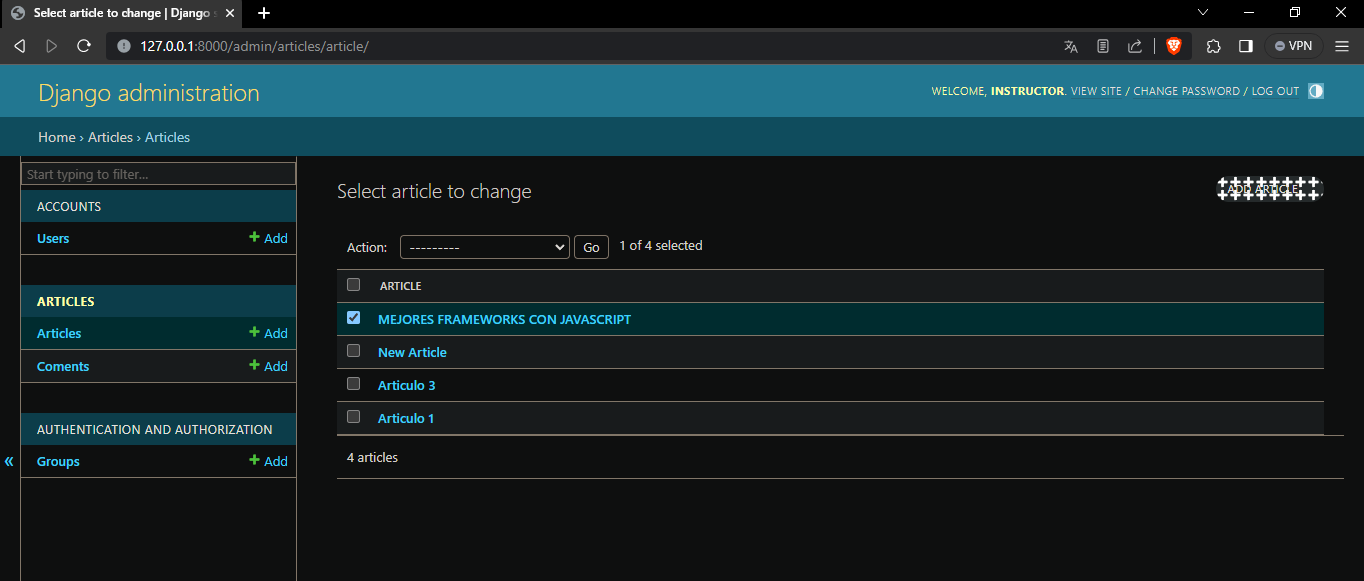


1. Actualizar ***artículos/admin.py*** de la siguiente manera para agregar la vista ***StackedInline***
   1. ***TabularInline***: todos los campos del modelo aparecen en una línea
   2. ***StackedInline***: cada campo tiene su propia línea.

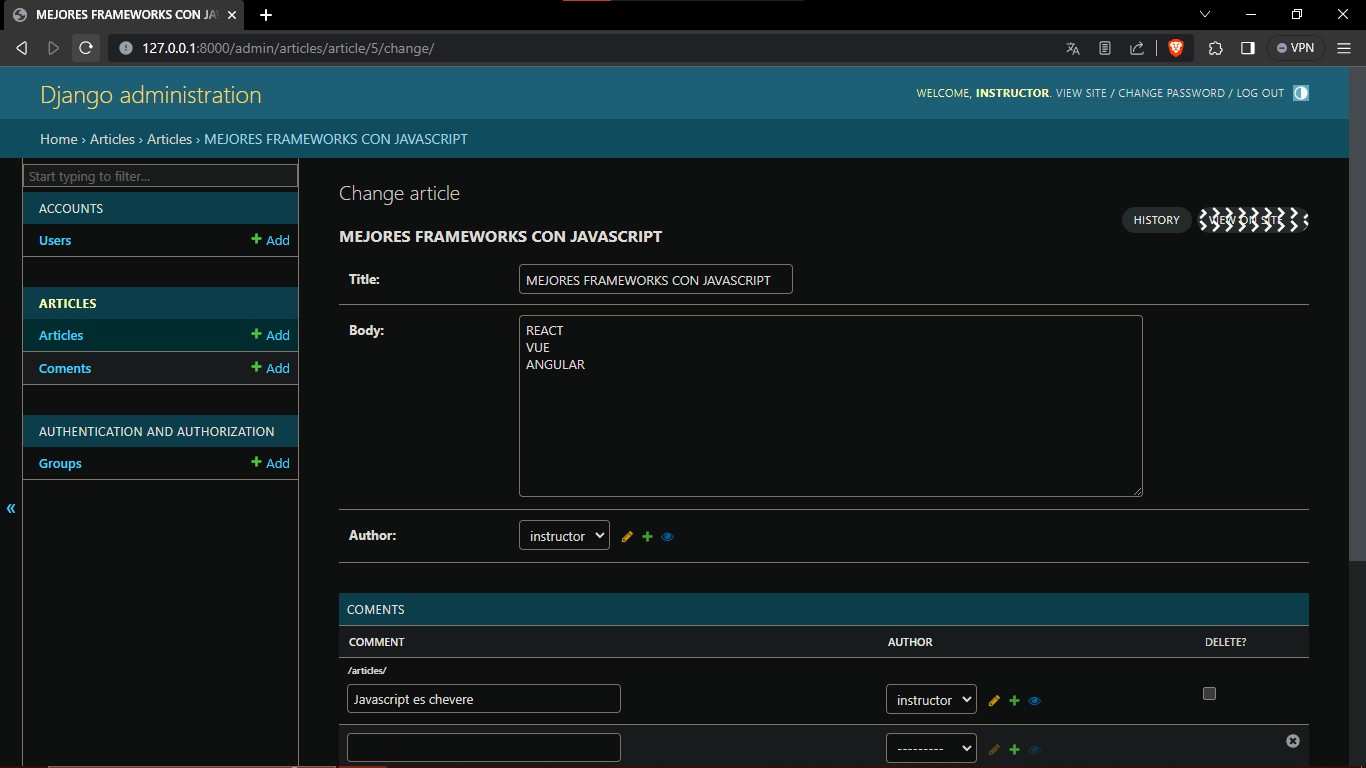


1. Iniciar el servidor con ***python manage.py runserver***. Ir a la página principal [***http://127.0.0.1:8000/admin/****;*](http://127.0.0.1:8000/admin/)hacer clic en ***articles,*** seleccionar el artículo al que se le hizo el comentario*.*

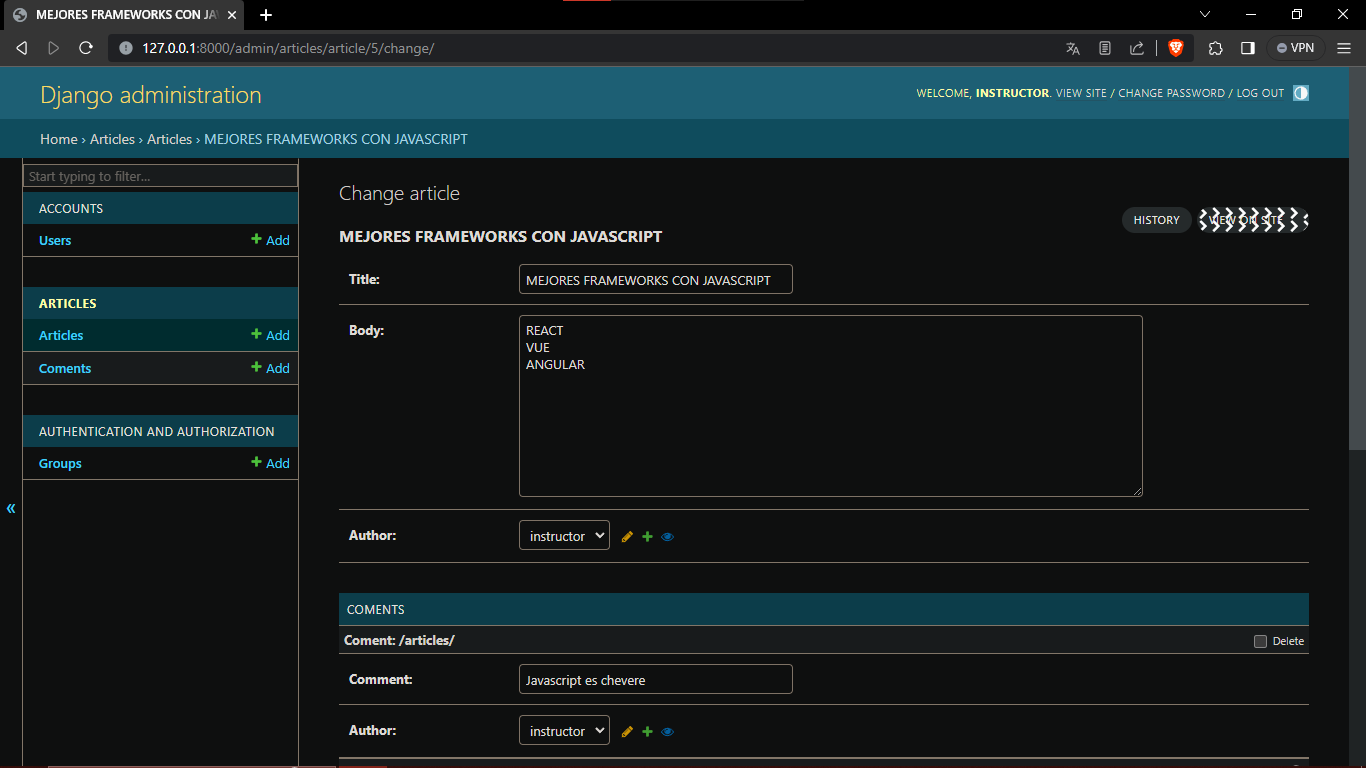




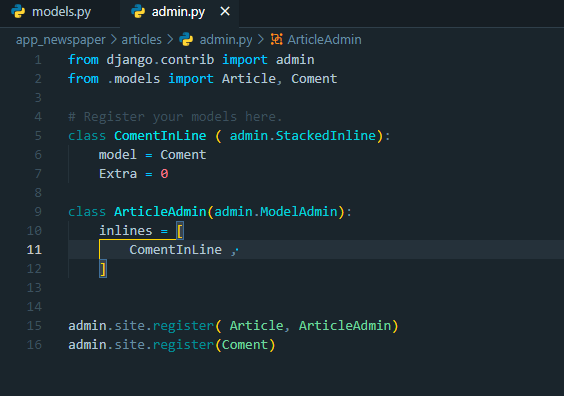
# Con TabularInline

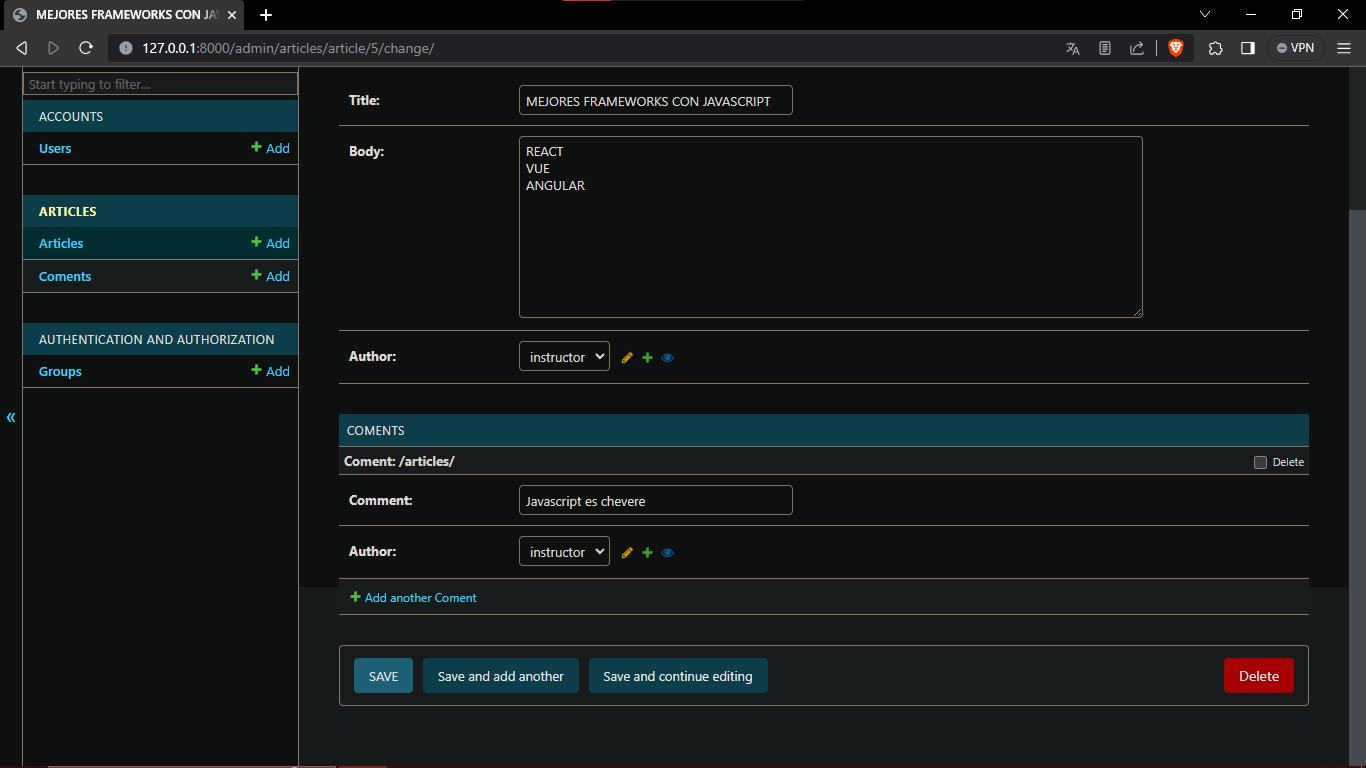


# Con StackedInline



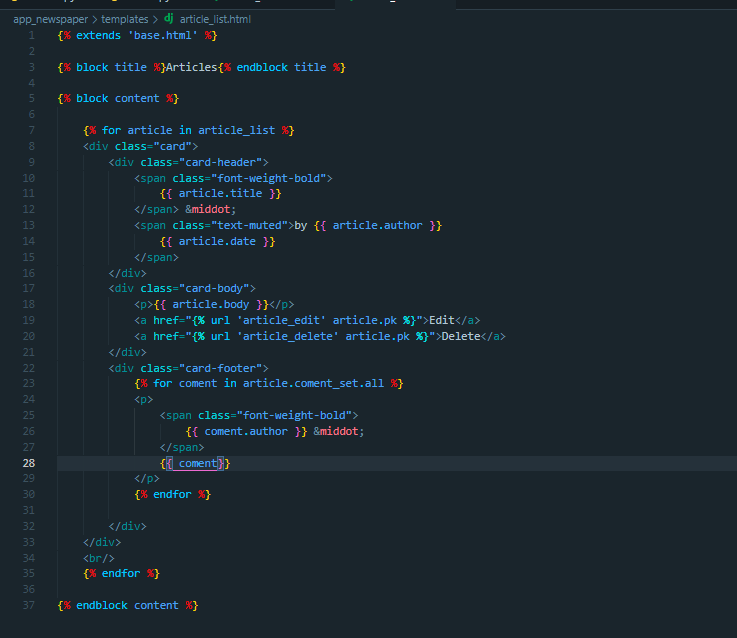
1. Modificamos ***articles/admin.py*** para no desplegar comentarios vacíos.

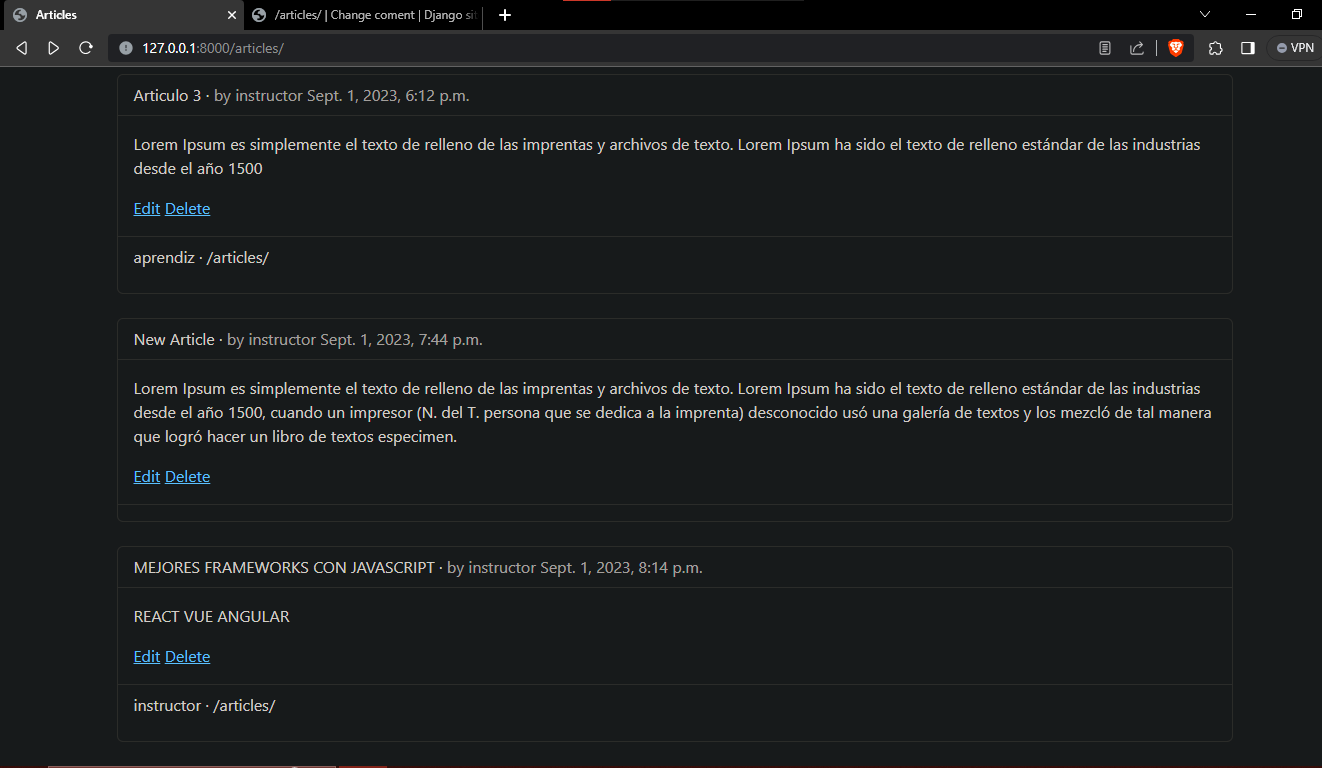




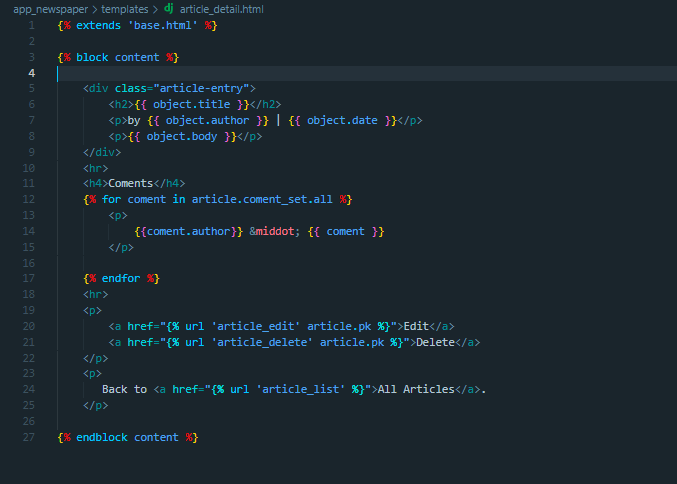
1. Para mostrar los comentarios relacionados con un artículo específico, realizar una "consulta", que es una forma de solicitar a la base de datos una información específica. Django tiene una sintaxis incorporada para esto conocida como ***FOO\_set***donde ***FOO*** es el nombre del modelo fuente en minúsculas.

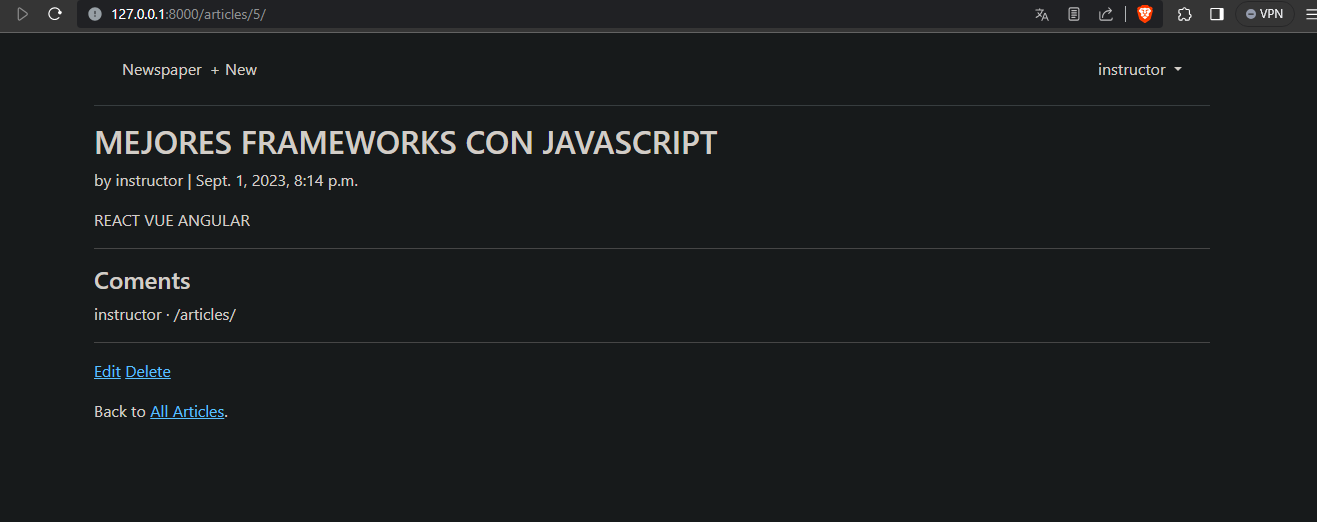
En la plantilla ***templates/article\_list.html*** se realizan los siguientes cambios:





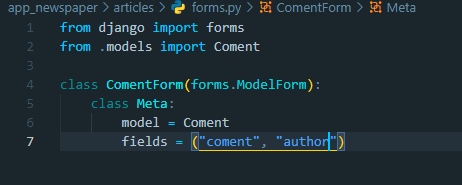
1. Agregamos los comentarios a la página de detalles de cada artículo, modificando el archivo ***templates/article\_detail.html***



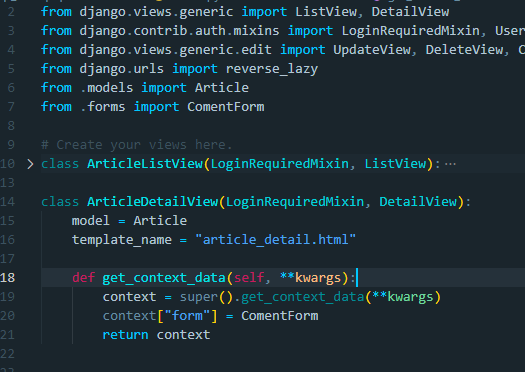


1. Agregamos un formulario para que los usuarios puedan realizar los comentarios. ***ModelForm*** es una clase auxiliar diseñada para traducir modelos de bases de datos en formularios; podemos usarlo para crear un formulario llamado ***CommentForm***.

Crear un archivo llamado ***articles/forms.py***

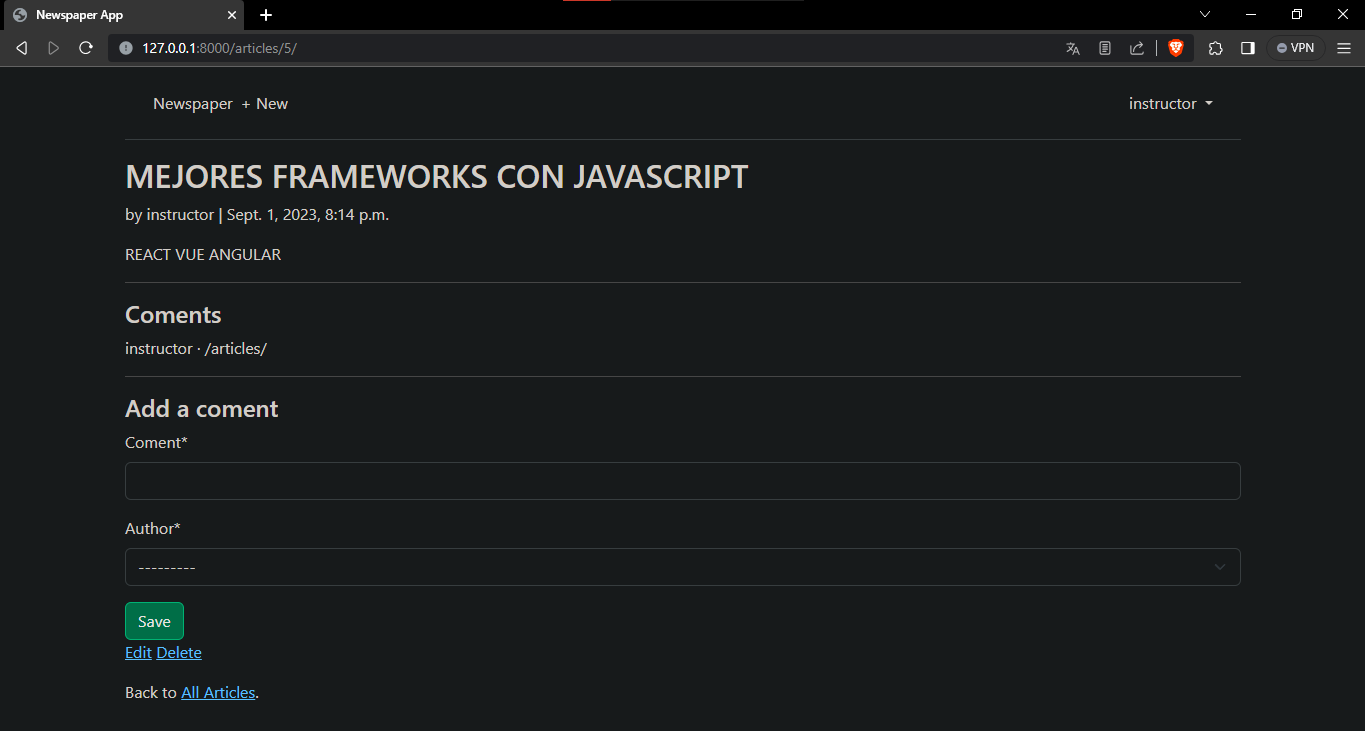


1. Modificamos ***articles/views.py*** para incluir el formulario de comentarios



1. Modificamos ***templates/article\_detail.html*** para mostrar el formulario en pantalla





1. Modificamos el archivo ***articles/views.py*** para cambiar el nombre de ***ArticleDetailView*** a ***CommentGet***, ya que maneja las solicitudes GET pero no las solicitudes POST.

Además, creamos una nueva clase ***CommentPost*** para agregar comentarios***. FormView*** es una vista integrada que muestra un formulario, cualquier error de validación y redirige a una nueva URL. Se usa en combinación con ***SingleObjectMixin***, que ayuda a asociar el artículo actual con el formulario.

Redefinimos la clase ***ArticleDetailView***

