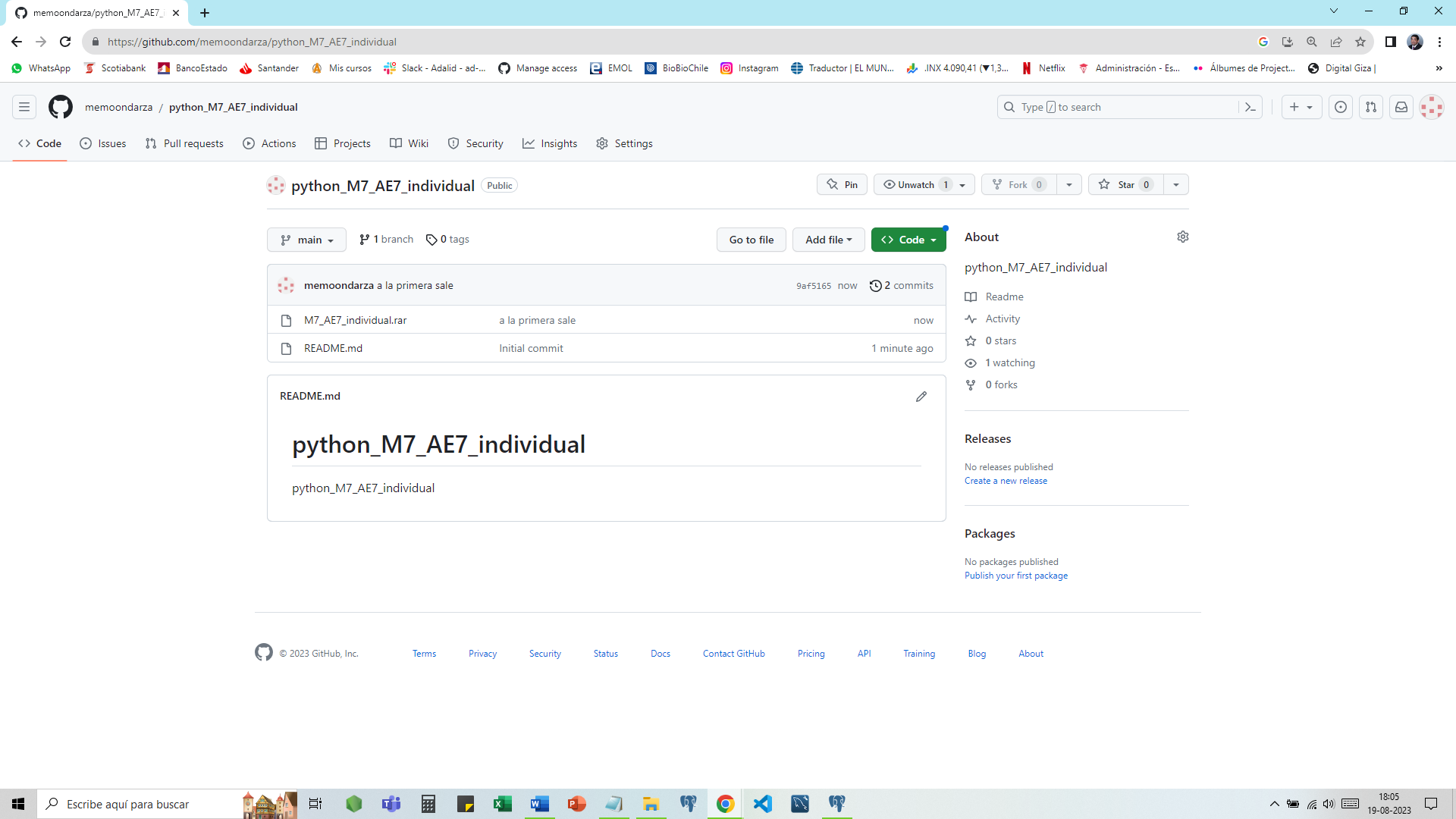
<https://github.com/memoondarza/python_M7_AE7_individual>



**DESARROLLO**

En documento Word comenta, de forma detallada, de qué forma participan las siguientes aplicaciones

preinstaladas en tu proyecto.

* **django.contrib.admin**

Una de las cosas que hacen de Django uno de los mejores frameworks (sin importar el lenguaje de otros frameworks) es el llamado Django Admin, que es un panel de control que permite administrar todos los modelos.

Es decir, el código ya está hecho por nosotros, los formularios también y lo mejor es que cada uno está dispuesto para cada tipo de dato.

La aplicación de administración de Django puede usar tus modelos para construir automáticamente un área dentro del sitio que puedes usar para crear, consultar, actualizar y borrar registros. Esto puede ahorrar mucho tiempo de desarrollo, haciendo muy fácil probar tus modelos y dar una idea de si tus datos son correctos. La aplicación de administración también puede ser útil para manejar datos en producción, dependiendo del estilo del sitio web. Desde el proyecto Django solo se recomienda para gestión de datos internos (por ejemplo, solo para uso de administradores o personas internas de tu organización), ya que como enfoque centrado en el modelo no es necesariamente la mejor interfaz posible para todos los usuarios, exponiendo una gran cantidad de detalles innecesarios de los modelos.

Toda la configuración requerida para incluir la aplicación admin en tu sitio Web fue hecha automáticamente cuando creaste el esqueleto del proyecto (para información sobre dependencias reales necesarias, vea los documentos de Django aquí). Como resultado, todo lo que debes hacer para agregar tus modelos a la aplicación admin es registrarlos. Al final de este artículo entregaremos una breve demostración sobre cómo puedes configurar aún más el área de administración para mejorar la visualización de nuestros modelos de datos.

Después de registrar los modelos te mostraremos como crear un nuevo "administrador", iniciar sesión en el sitio, y crear algunos libros, autores, instancias de libros, y géneros. Esto será útil para probar las vistas y plantillas que empezaremos a crear en el siguiente tutorial.

Es muy importante en mi aplicación, porque me permite administrar usuarios de la aplicación, otorgando permisos de administración del sitio. Esto en forma muy sencilla

* **django.contrib.auth**

Es el sistema de autenticación de Django. Se necesita de django.contrib.auth. por lo tanto, es necesario agregar estas líneas a nuestro views.py:

* from django.contrib.auth.forms import UserCreationForm
* from django.contrib.auth.forms import AuthenticationForm
* from django.contrib.auth import login, authenticate, logout
* from django.contrib.auth.decorators import login\_required

Uno de los principales apartados cuando desarrollamos un sistema es la Seguridad, en específico, la denominada Autenticación, la cual no es más que un conjunto de mecanismos para asociar la solicitud entrante con un conjunto de credenciales de identificación que puede contener el usuario del cual proviene la petición o el token perteneciente a un usuario, dependiendo los mecanismos que se hayan definido, se valida si la solicitud puede ser permitida o no.

Sabemos que la autenticación siempre se ejecuta al comienzo de la vista antes de que se produzcan las comprobaciones de permisos y limitaciones y antes de cualquier procesamiento de datos en alguna vista, por ello Django Rest Framework, luego de validar la petición, guardará la instancia de la clase User en request.user. Además de ello, existe otra propiedad llamada request.auth, la cual se utiliza para cualquier información adicional de la autenticación, por ejemplo, representar el Token para la Autenticación.

* **django.contrib.sessions**

Con todas estas limitaciones y agujeros potenciales de seguridad, es obvio que la gestión de las cookies y de las sesiones persistentes es el origen de muchos dolores de cabeza para los desarrolladores web. Por supuesto, uno de los objetivos de Django es evitar eficazmente estos dolores de cabeza, así que dispone de un entorno de sesiones diseñado para suavizar y facilitar todas estas cuestiones por vos.

El entorno de sesiones te permite almacenar y recuperar cualquier dato que quieras basándote en la sesión del usuario. Almacena la información relevante solo en el servidor y abstrae todo el problema del envío y recepción de las cookies. Estas solo almacenan una versión codificada (hash) del identificador de la sesión, y ningún otro dato, lo cual te aisla de la mayoría de los problemas asociados con las cookies.

Las sesiones se implementan mediante un middleware y un modelo Django. Para activar las sesiones, necesitas seguir los siguientes pasos:

1. Editar el valor de MIDDLEWARE\_CLASSES de forma que contenga 'django.contrib.sessions.middleware.SessionMiddleware'.
2. Comprobar que 'django.contrib.sessions' esté incluido en el valor de INSTALLED\_APPS (y ejecutar manage.py syncdb si lo tuviste que añadir).

Los valores por defecto creados por startproject ya tienes estas dos características habilitadas, así que a menos que las hayas borrado, es muy probable que no tengas que hacer nada para empezar a usar las sesiones.

Si lo que quieres en realidad es no usar sesiones, deberías quitar la referencia a SessionMiddleware de MIDDLEWARE\_CLASSES y borrar 'django.contrib.sessions' de INSTALLED\_APPS. Esto te ahorrará sólo un poco de sobrecarga, pero toda ayuda es buena.

Cuando están activadas las sesiones, los objetos HttpRequest — el primer argumento de cualquier función que actúe como una vista en Django — tendrán un atributo llamado session, que se comporta igual que un diccionario. Se puede leer y escribir en él de la misma forma en que lo harías con un diccionario normal.