Módulo 7, Aprendizaje Esperado 7: Ejercicio Individual

Nombre: Cristian Aranda Bórquez

Como ya se ha abordado en este bootcamp, Django es un framework de alto nivel basado en lenguaje Python destinado al desarrollo web. Este framework facilita el desarrollo de sitios y aplicaciones web de forma rápida y segura a través de un modelo MTV (Model-Template-View).

Uno de los puntos más importantes de cómo funciona este framework es a través de sus aplicaciones preinstaladas o predeterminadas. En este ensayo, se desarrollarán diversas ideas destinadas a como estas aplicaciones facilitan el desarrollo web y la importancia que tienen en el funcionamiento de Django.

Poniéndonos en el caso de un desarrollador web al que le ha encargado el desarrollo de una aplicación web utilizando Django, en donde sea necesario la autenticación de usuarios, manejo de sesiones, mensajes de notificación y una interfaz administrativa, este framework posee todo lo necesario para poder completar esto a través de sus aplicaciones preinstaladas:

- **`admin`**: Refiere al panel de administración automático para gestionar datos de la aplicación a través de una interfaz web interactiva.
- **`auth`**: Esta aplicación es el sistema de autenticación y autorización de usuarios, a través de las cuales se manejan login, permisos, grupos, etc.
- **`contenttypes`**: Permite crear relaciones genéricas entre modelos. Esto mantiene un registro de todos los modelos instalados en el proyecto Django.
- **`sessions`**: Gestiona sesiones de usuario, almacenando datos de ellos entre peticiones HTTP
- **`messages`**: Permite el uso de mensajes temporales para mostrar notificaciones a los usuarios.
- **`staticfiles`**: Gestiona archivos estáticos (CSS, JavaScript, Imágenes) en desarrollo y producción.

Todas estas aplicaciones se pueden encontrar ya instaladas dentro del archivo `settings.py` en la lista bajo el nombre de `INSTALLED_APPS` y son precedidas por un ‘`django.contrib`.’.

Al ver el listado de estas aplicaciones surge la pregunta de ¿Cuál podría ser la utilidad de ellas en un proyecto? Su función puede ser más fácil de visualizar si es que se plantean dentro de un proyecto.

Partiendo desde un punto de vista más general, se puede plantear la función de estas aplicaciones en aspectos básicos y cruciales de una aplicación web. Por ejemplo, **`auth`** es importante para la gestión de usuarios ya que facilita el registro de ellos, login y logout para acceder a diferentes partes de la app, recuperación de contraseñas y también facilita el manejo de permisos; Por otro lado, **`sessions`** administra sesiones de usuario, abriendo la puerta a funciones como carritos de compra y preferencias, guardando información del usuario; **`staticfiles`** organiza archivos estáticos del desarrollo front end en desarrollo y facilita el despliegue de ellos en producción; **`messages`** aporta a la interfaz de usuario indicando notificaciones o mensajes que guíen la experiencia del usuario; Finalmente, **`admin`** es una de las herramientas más poderosas de Django, ya que permite el manejo de modelos, usuarios, grupos, permisos y registros en la base de datos a través de una interfaz gráfica automática, lo cual simplifica el trabajo con estos elementos.

Ahora, si bien se mencionó que vienen preinstaladas al utilizar Django, existen configuraciones básicas que se deben realizar para poder habilitar y personalizar estas aplicaciones.

Ya se mencionó que deben estar en listadas en `INSTALLED_APPS` del archivo `settings.py` pero también se deben realizar las migraciones a través de `python manage.py migrate`, lo cual creará las tablas necesarias de cada aplicación en la base de datos del proyecto.

Por otro lado, en el caso de **`staticfiles`**, dentro del mismo archivo `settings.py` se debe configurar la dirección o path de la carpeta `statics`, utilizando `STATIC_URL` y `STATICFILES_DIRS`. También, se puede configurar el backend utilizado por **`sessions`** que por defecto utiliza la base de datos.

Estas aplicaciones interactúan con el sistema mediante middleware que procesa peticiones/respuestas. También, utilizan modelos ORM que se sincronizan con la base de datos mediante migraciones. Además, estas aplicaciones también se comunican entre sí, por ejemplo, **`messages`** usa **`sessions`** para almacenar notificaciones temporales. Por ejemplo, **`auth`** crea las tablas `auth_user`, `auth_group` y `auth_permission` automáticamente. **`sessions`** almacena datos en la tabla `django_session` con un identificador único (`session_key`) que se envía al navegador como cookie. El middleware

`SessionMiddleware` intercepta cada petición HTTP para cargar/guardar estos datos automáticamente.

Ahora, para darle un poco más de forma a esta visualización en un caso real, se propone un escenario de una plataforma de cursos en línea.

El desarrollador debe crear una plataforma de cursos online en donde los estudiantes se puedan registrar y acceder con usuario y contraseña. Aquí tendrán su propio perfil de usuario con su foto y preferencias guardadas. Al completar una lección, la app les entregara una notificación de progreso. Los profesores y/o el director podrán gestionar los cursos a través del panel de **`admin`**.

En este caso que se está planteando, la aplicación que permite todo lo que tenga que ver con registro y accesos de usuarios corresponde a **`auth`**. Esta app también permite la protección de vistas, utilizando un decorador de `@login_required`, como podría ser el poder ver las lecciones. En el caso de las notificaciones de progreso entraría en acción **`messages`** y para los recursos de las lecciones como pdf o videos sería **`staticfiles`**.

Dentro de lo que es el panel de **`admin`**, se les puede dar acceso a distintos usuarios a través del status de staff para que puedan acceder a esta función. A través de esto los profesores podrían gestionar los cursos, viendo los estudiantes inscritos para cada uno, agregando o eliminando alumnos inscritos, recursos de las lecciones, etc. A lo anterior se suma el uso de **`sessions`** para guardar información del alumno y su progreso dentro de la plataforma mientras la sesión está activa. Adicionalmente, **`contenttypes`** permitiría implementar un sistema de comentarios universal donde estudiantes puedan comentar tanto en cursos como en lecciones individuales usando un solo modelo, o un sistema de 'favoritos' donde puedan guardar diferentes tipos de recursos.

A modo de conclusión, parte del poderío que tiene Django en su capacidad de facilitar el desarrollo web recae dentro de estas aplicaciones por defecto. Si bien, se pueden utilizar otras librerías externas para realizar estas funciones, Django de forma nativa otorga todo lo necesario para levantar un proyecto y sus aplicaciones. Esto hace que sea un framework amigable para el desarrollador y que el producto final sea también amigable con la experiencia del usuario. Al perfeccionar el uso de estas aplicaciones se puede pasar de semanas de desarrollo a días, permitiendo a los desarrolladores enfocarse en la lógica de negocio y el planteamiento del programa más que en su infraestructura.