### Definir modelos

Pensemos un momento en nuestros datos. Cada usuario tendrá que crear una serie de temas en su registro de aprendizaje. Cada entrada que hagan estará vinculada a un tema, y estas entradas se mostrarán como texto. También necesitaremos almacenar la marca de tiempo de cada entrada para poder mostrar a los usuarios cuándo hicieron cada una.

Abre el archivo *models.py* y mira el contenido existente:

**models.py**

from django.db import models  
  
# Create your models here.

Se está importando un módulo llamado models, y se nos invita a crear modelos propios. Un *model* indica a Django cómo trabajar con los datos que se almacenarán en la aplicación. Un modelo es una clase; tiene atributos y métodos, como todas las clases de las que hemos hablado. Este es el modelo para los temas que almacenarán los usuarios:

from django.db import models  
  
class Topic(models.Model):  
 """A topic the user is learning about."""  
❶ text = models.CharField(max\_length=200)  
❷ date\_added = models.DateTimeField(auto\_now\_add=True)  
  
❸ def \_\_str\_\_(self):  
 """Return a string representation of the model."""  
 return self.text

Hemos creado una clase llamada Topic, que hereda de Model-una clase padre incluida en Django que define la funcionalidad básica de un modelo. Añadimos dos atributos a la clase Topic: text y date\_added.

El atributo text es un CharField, un dato formado por caracteres o texto ❶. Utilizas CharField cuando quieres almacenar una pequeña cantidad de texto, como un nombre, un título o una ciudad. Cuando definimos un atributo CharField, tenemos que decirle a Django cuánto espacio debe reservar en la base de datos. Aquí le damos un max\_length de 200 caracteres, que debería ser suficiente para contener la mayoría de los nombres de temas.

El atributo date\_added es un DateTimeField, un dato que registrará una fecha y una hora ❷. Pasamos el argumento auto\_now\_add=True, que indica a Django que establezca automáticamente este atributo con la fecha y hora actuales cada vez que el usuario cree un nuevo tema.

Es una buena idea decirle a Django cómo quieres que represente una instancia de un modelo. Si un modelo tiene un método \_\_str\_\_(), Django llama a ese método cada vez que necesita generar una salida que haga referencia a una instancia de ese modelo. Aquí hemos escrito un método \_\_str\_\_() que devuelve el valor asignado al atributo text ❸.

Para ver los distintos tipos de campos que puedes utilizar en un modelo, consulta la página "Referencia de campos de modelo" en [https://docs.djangoproject.com/en/4.1/ref/models/fields.](https://docs.djangoproject.com/en/4.1/ref/models/fields) No necesitarás toda la información ahora mismo, pero te será muy útil cuando desarrolles tus propios proyectos Django.

[anterior](c18_16.html)[Subtema 17 de 41: (Ver todo)](c18.html)[siguiente](c18_18.html)