Tarea 5 – Modelos en Django

Un modelo en Django es un tipo especial de objeto que se guarda en la base de datos. Una base de datos es una colección de datos.

Las aplicaciones web de Django acceden y administran los datos a través de objetos de Python a los que se hace referencia como modelos. Los modelos definen la *estructura* de los datos almacenados, incluidos los *tipos* de campo y los atributos de cada campo, como su tamaño máximo, valores predeterminados, lista de selección de opciones, texto de ayuda para la documentación, texto de etiqueta para formularios, etc. La definición del modelo es independiente de la base de datos subyacente. puede elegir una de entre varias como parte de la configuración de su proyecto. Una vez que haya elegido la base de datos que desea usar, no necesita hablar directamente con ella. Simplemente escriba la estructura de su modelo y algo de código, y Django se encargará de todo el trabajo sucio, al comunicarse con la base de datos por usted.

Los modelos están definidos, normalmente, en el archivo **models.py** de la aplicación. Son implementados como subclases de django.db.models.Model, y pueden incluir campos, métodos y metadata. El fragmento de código más abajo muestra un modelo "típico", llamado MyModelName:

```
from django.db import models

class MyModelName(models.Model):
    """
    Una clase típica definiendo un modelo, derivado desde la clase Model.
    """

    # Campos
    my_field_name   =   models.CharField(max_length=20,   help_text="Enter   field
documentation")
    ...

    # Metadata
    class Meta:
        ordering = ["-my_field_name"]

    # Métodos
    def get_absolute_url(self):
         """
         Devuelve la url para acceder a una instancia particular de MyModelName.
         """
         return reverse('model-detail-view', args=[str(self.id)])

    def __str__(self):
        """
        Cadena para representar el objeto MyModelName (en el sitio de Admin, etc.)
        """
        return self.field_name
```

Campos

Un modelo puede tener un número arbitrario de campos, de cualquier tipo. Cada uno representa una columna de datos que queremos guardar en nuestras tablas de la base de datos. Cada registro de la base de datos (fila) consistirá en uno de cada posible valor del campo. Echemos un vistazo al ejemplo visto arriba:

```
my_field_name = models.CharField(max_length=20, help_text="Enter field documentation")
```

Nuestro ejemplo de arriba tiene un único campo llamado my_field_name, de tipo models.CharField — lo que significa que este campo contendrá una cadena de caracteres alfanuméricos. Los tipos de campo son asignados usando clases específicas, que determinan el tipo de registro que se usa para guardar el dato en la base, junto con un criterio de evaluación que se usará cuando se reciban los valores de un formulario HTML (es decir, qué constituye un valor válido). Los tipos de campo pueden también tomar argumentos que especifican además cómo se guarda o cómo se puede usar. En este caso le damos a nuestro campo dos argumentos:

- max_length=20 — Establece que la longitud máxima del valor de este campo es 20 caracteres.
- help_text="Enter field documentation" — Proporciona una etiqueta de texto para mostrar que ayuda a los usuarios a saber qué valor proporcionar cuando un usuario ha de introducirlo vía un formulario HTML.

Metadatos
Puedes declarar metadatos a nivel de modelo para tu Modelo declarando class Meta, tal como se muestra.

```
class Meta:
    ordering = ["-my_field_name"]
    ...
```

Una de las características más útiles de estos metadatos es controlar el *orden por defecto* de los registros que se devuelven cuando se consulta el tipo de modelo. Se hace especificando el orden de comprobación en una lista de nombres de campo en el atributo ordering, como se muestra arriba. La ordenación dependerá del tipo de campo (los campos de caracteres de ordenan alfabéticamente, mientras que los campos de fechas están clasificados por orden cronológico). Como se muestra arriba, se puede invertir el orden de clasificación añadiendo el símbolo (-) como prefijo del nombre del campo.

Biblipgrafia

https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Models