Aula 03: Modelos Avançados em Django

Introdução

Nesta aula, vamos aprofundar nossos conhecimentos em Django, focando em funcionalidades avançadas dos modelos. Abordaremos relacionamentos entre modelos, métodos personalizados, signals e validações.

1. Relacionamentos entre Modelos

1.1. Relacionamento One-to-Many

Vamos criar um relacionamento One-to-Many entre `Aluno` e `Post`. No arquivo `models.py` do `meu_app`, adicione o seguinte código:

```
from django.contrib.auth.models import AbstractBaseUser, PermissionsMixin,
BaseUserManager
from django.db import models
from django.utils.translation import gettext_lazy as _

class AlunoManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError(_('O e-mail é obrigatório'))
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        if password:
            user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, password=None, **extra_fields):
        extra_fields.setdefault('is_superuser', True)
        extra_fields.setdefault('is_staff', True)

        if not password:
            raise ValueError(_('Superusuários devem ter uma senha.'))
        return self.create_user(email, password, **extra_fields)

class Aluno(AbstractBaseUser, PermissionsMixin):
    email = models.EmailField(unique=True, verbose_name=_('Email'))
    nome = models.CharField(max_length=100, verbose_name=_('Nome Completo'))
    cpf = models.CharField(max_length=14, unique=True, verbose_name=_('CPF'))
    foto = models.ImageField(upload_to='fotos/%Y/%m/%d/', blank=True, null=True,
verbose_name=_('Foto'))
    data_de_cadastro = models.DateTimeField(auto_now_add=True,
verbose_name=_('Data de Cadastro'))
    ip_de_cadastro = models.GenericIPAddressField(default='0.0.0.0',
verbose_name=_('IP de Cadastro'))
    is_active = models.BooleanField(default=True, verbose_name=_('Ativo'))
    is_staff = models.BooleanField(default=False, verbose_name=_('Equipe'))
```

```
    is_superuser = models.BooleanField(default=False,
verbose_name=_('Superusuário'))

    groups = models.ManyToManyField(
        'auth.Group',
        related_name='grupos_aluno',
        blank=True,
        help_text=_('Os grupos aos quais este usuário pertence. Um grupo
representa uma coleção de permissões.'),
        verbose_name=_('Grupos')
    )
    user_permissions = models.ManyToManyField(
        'auth.Permission',
        related_name='permissoes_usuario_aluno',
        blank=True,
        help_text=_('Permissões específicas para este usuário.'),
        verbose_name=_('Permissões de Usuário')
    )

    objects = AlunoManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['nome', 'cpf']

    @property
    def is_professor(self):
        return False

    def __str__(self):
        return self.email

    class Meta:
        verbose_name = _('Aluno')
        verbose_name_plural = _('Alunos')

class Post(models.Model):
    author = models.ForeignKey(Aluno, on_delete=models.CASCADE)
    title = models.CharField(max_length=255)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    approved = models.BooleanField(default=False)
    image = models.ImageField(upload_to='post/%Y/%m/%d/', blank=True, null=True)
    links = models.TextField(blank=True)

    def __str__(self):
        return f"{self.title} by {self.author.email}"

    class Meta:
        ordering = ['-created_at']
```

Flashcards

AlunoManager

- **create_user**: Método para criar um usuário normal.
- **create_superuser**: Método para criar um superusuário.

Aluno

- **email**: Campo de email único.
- **nome**: Campo de nome completo.
- **cpf**: Campo de CPF único.
- **foto**: Campo de imagem para foto do aluno.
- **data_de_cadastro**: Data de cadastro do aluno.
- **ip_de_cadastro**: IP de cadastro do aluno.
- **is_active**: Indica se o aluno está ativo.
- **is_staff**: Indica se o aluno é parte da equipe.
- **is_superuser**: Indica se o aluno é superusuário.
- **groups**: Grupos aos quais o aluno pertence.
- **user_permissions**: Permissões específicas do aluno.
- **USERNAME_FIELD**: Campo usado para login (email).
- **REQUIRED_FIELDS**: Campos obrigatórios além do email.
- **is_professor**: Propriedade que indica se o usuário é professor.
- **Meta**: Metadados do modelo.

Post

- **author**: Relacionamento com o modelo Aluno.
- **title**: Título do post.
- **content**: Conteúdo do post.
- **created_at**: Data de criação do post.
- **updated_at**: Data de atualização do post.
- **approved**: Indica se o post foi aprovado.
- **image**: Imagem associada ao post.
- **links**: Links adicionais no post.
- **Meta**: Metadados do modelo, ordenando por data de criação.

Conclusão

Nesta aula, exploramos funcionalidades avançadas dos modelos em Django, incluindo relacionamentos, métodos personalizados, signals e validações. Com esses conhecimentos, você está mais preparado para desenvolver aplicações Django robustas e complexas.