

MANUAL TECNICO

- SEBASTIAN MEJIA
- MARIA JOSE RODRIGUEZ
- JUAN DAVID CARDONA
- MICHEL DAHIANA RIVERA



SENA – CALDAS
2025

CONTENIDO

Introducción	3
Objetivos	3
REQUISITOS.....	4
Estructura de repositorio	5
Instalación y configuración	5
Arquitectura y diagramas	8
DESPLIEGUE PASO A PASO	13
Conclusión	15

INTRODUCCIÓN

El presente manual técnico documenta la aplicación web desarrollada en **Django** con base de datos en **MySQL**, cuyo propósito es la **gestión integral de eventos** en un entorno académico, cultural y administrativo.

La plataforma ofrece un ecosistema digital que centraliza los procesos relacionados con la planificación, organización y evaluación de eventos.

OBJETIVOS

El objetivo de este manual técnico es **documentar de manera clara y estructurada la arquitectura, funcionamiento y componentes de la aplicación de gestión de eventos**, con el fin de facilitar su instalación, administración, mantenimiento y futura evolución.

A través de este documento se busca:

- Proporcionar a los desarrolladores y administradores del sistema una guía completa sobre la estructura del proyecto, los modelos de datos, las vistas, formularios y plantillas empleadas.
- Explicar la configuración del entorno tecnológico utilizado, incluyendo el framework **Django**, el motor de base de datos **MySQL** y las librerías externas implementadas.
- Detallar los procesos clave de la aplicación, como la gestión de **usuarios, eventos, áreas, categorías, inscripciones y evaluaciones**, garantizando una visión integral del sistema.
- Servir como base de referencia para la **ampliación de funcionalidades**, resolución de incidencias técnicas y formación de nuevos miembros en el equipo de desarrollo.
- **Docker Compose** instalado (versión 1.29 o superior recomendada).

En síntesis, este manual asegura que la aplicación pueda ser comprendida, utilizada y escalada de manera eficiente, contribuyendo a la continuidad operativa y a la mejora constante del sistema.

REQUISITOS

Requisitos de Hardware

Procesador (CPU): Intel Core i5 (10ª generación o superior) / AMD Ryzen 5 (o equivalente).

Memoria RAM: 8 GB (mínimo) – recomendado 16 GB para ejecutar servidor, base de datos y entorno de desarrollo cómodamente.

Almacenamiento: 256 GB SSD (mínimo), preferible 512 GB SSD para mejor rendimiento en consultas y manejo de archivos.

Tarjeta de red: Ethernet 1 Gbps o Wi-Fi estable.

Pantalla: Resolución mínima Full HD (1920x1080).

Otros periféricos: Teclado, mouse, acceso a internet estable.

Requisitos de Software

Sistema Operativo (SO):

- Windows 10/11 (64 bits)
- Ubuntu 20.04+ LTS o equivalente en Linux
- macOS 12+ (opcional)

Entorno de desarrollo:

- Python 3.10 o superior
- Django 4.x
- MySQL 8.x
- Pip y Virtualenv (para gestión de dependencias)

Software adicional:

- Navegador web actualizado (Chrome, Firefox, Edge)
- Git (control de versiones)
- Editor de código recomendado: VS Code o PyCharm

EXTRUCTURA DE REPOSITORIO

Para alojar nuestra aplicación, usamos GitHub este es el siguiente link

https://github.com/mejiatian98/Edu_Fest.git la cual estará alojado en la rama **main**

app_admin_eventos	Diseño aplicado	3 weeks ago
app_asistentes	Arreglos del codigo para emepzar a hacer certificaciones	last month
app_evaluadores	Diseño aplicado	3 weeks ago
app_participantes	Arreglos del codigo para emepzar a hacer certificaciones	last month
app_usuarios	Diseño aplicado	3 weeks ago
principal_eventos	Arreglos del codigo para emepzar a hacer certificaciones	3 weeks ago
static	Diseño aplicado	3 weeks ago
templates	Diseño aplicado	3 weeks ago
.gitignore	edicion btn disable por si no hay valor en calificacion del par...	2 months ago
Notas_proyectos.txt	Arreglos del codigo para emepzar a hacer certificaciones	3 weeks ago
README.md	Gestion de eventos	2 months ago
manage.py	Gestion de eventos	2 months ago
notas.txt	Arreglos del codigo para emepzar a hacer certificaciones	last month
requirements.txt	Arreglos del codigo para emepzar a hacer certificaciones	3 weeks ago
requirements1.txt	Gestion de eventos	2 months ago

INSTALACIÓN Y CONFIGURACIÓN


Requisitos previos

Python, MySQL versión, Git.

Clonar repo: git clone <repo>

Instalar entorno virtual en (Windows)


powershell

 Copiar código

```
py -m venv venv
.\venv\Scripts\activate
pip install -r requirements.txt
```

En (Mac o Linux)

bash

 Copiar código

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Variable de entorno (.env)

```
.env
1 SECRET_KEY=tu_secret_key?
2 DEBUG=True
3 ALLOWED_HOSTS=127.0.0.1
4
5 # Configuración del correo (Gmail)
6 EMAIL_HOST_USER=admin1@gmail.com
7 EMAIL_HOST_PASSWORD=xxxx xxxx xxxx xxxx
8 DEFAULT_FROM_EMAIL="Edu-Fest <admin1@gmail.com>"
9
10
11 # Clave secreta de Django
12 SECRET_KEY=tu_hiper_mega_clave_secreta_super_segura
13
14 # Activar modo debug (solo en desarrollo)
15 DEBUG=True
16
17 # Hosts permitidos (evita el error de DisallowedHost)
18
19 # Configuración de la base de datos
20 DB_NAME=bd_edufest1
21 DB_USER=root
22 DB_PASSWORD=0000
23 DB_HOST=localhost
24 DB_PORT=3306
25
26
27
28 SITE_URL = "http://127.0.0.1:8000" # o el dominio real de tu app
```

Crear base de datos llamada **bd_edufest**

Hacer las migraciones

```
bash
```

[Copiar código](#)

```
py manage.py makemigrations  
py manage.py migrate
```

Crear super usuario

```
bash
```

[Copiar código](#)

```
py manage.py createsuperuser
```

Poner nombre de usuario, correo electrónico, contraseña y confirmar contraseña.

Por último puedes ejecutar el servidor

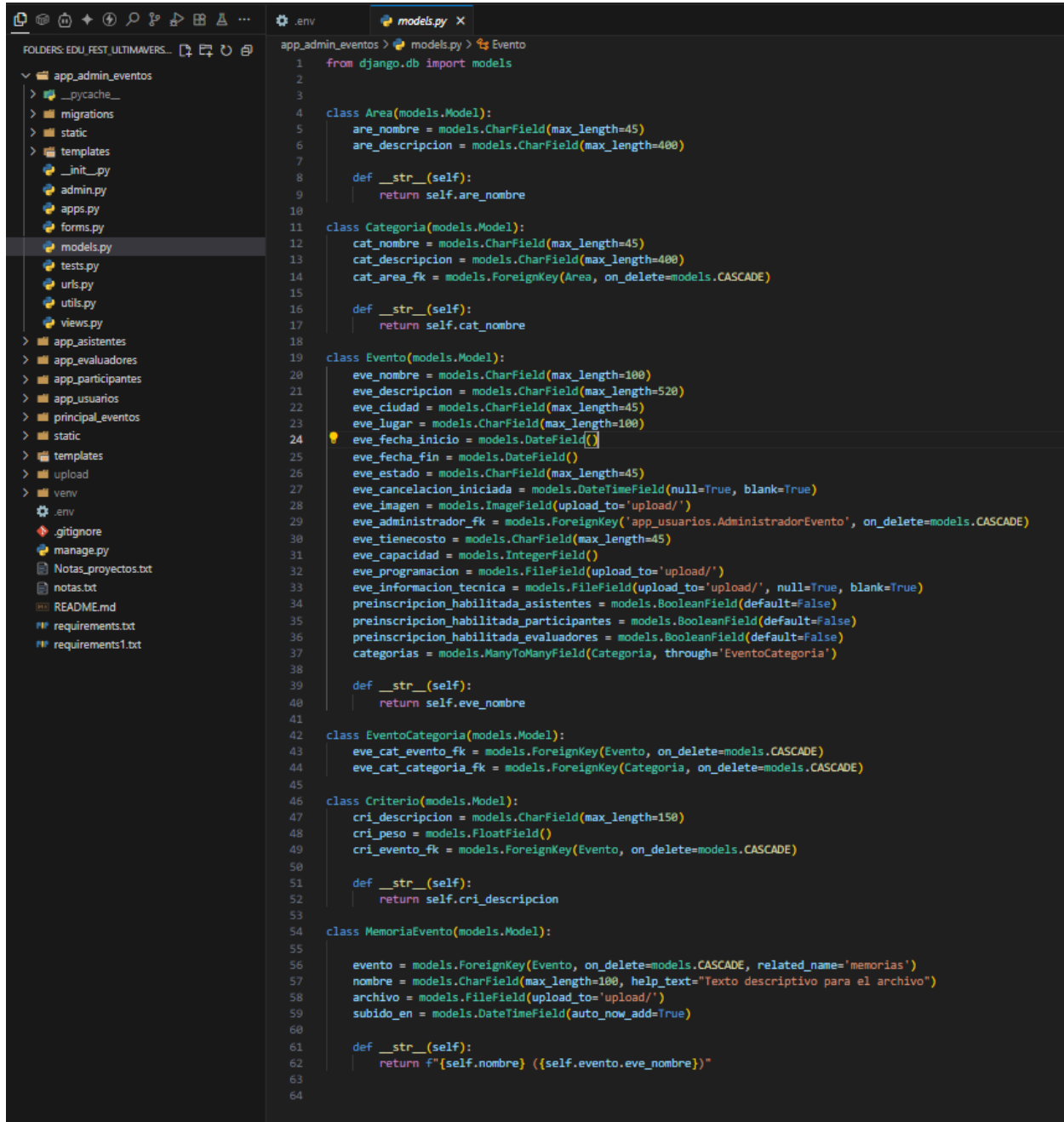
```
bash
```

[Copiar código](#)

```
py manage.py runserver
```

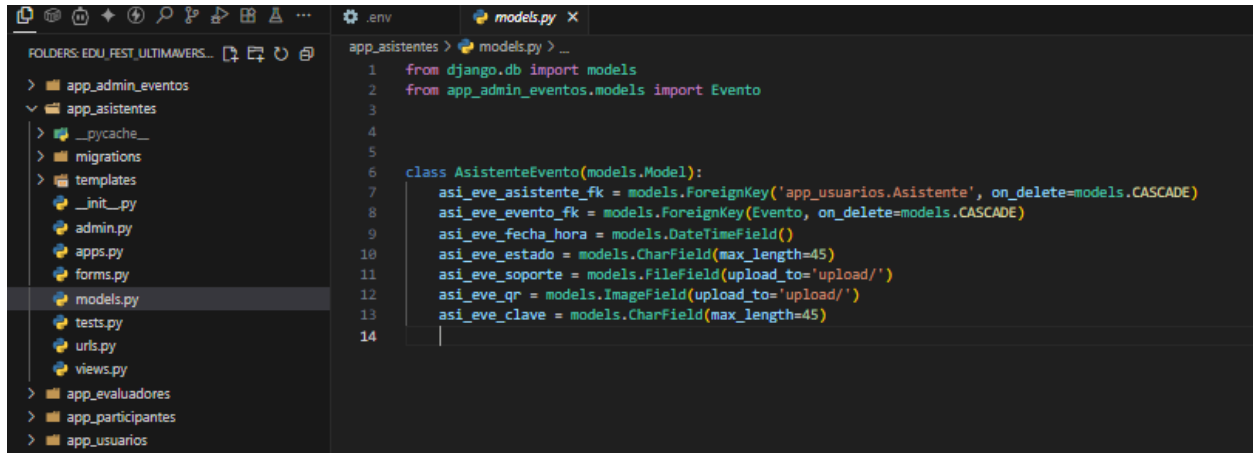

Documentación de cada modelo

App_admin_eventos



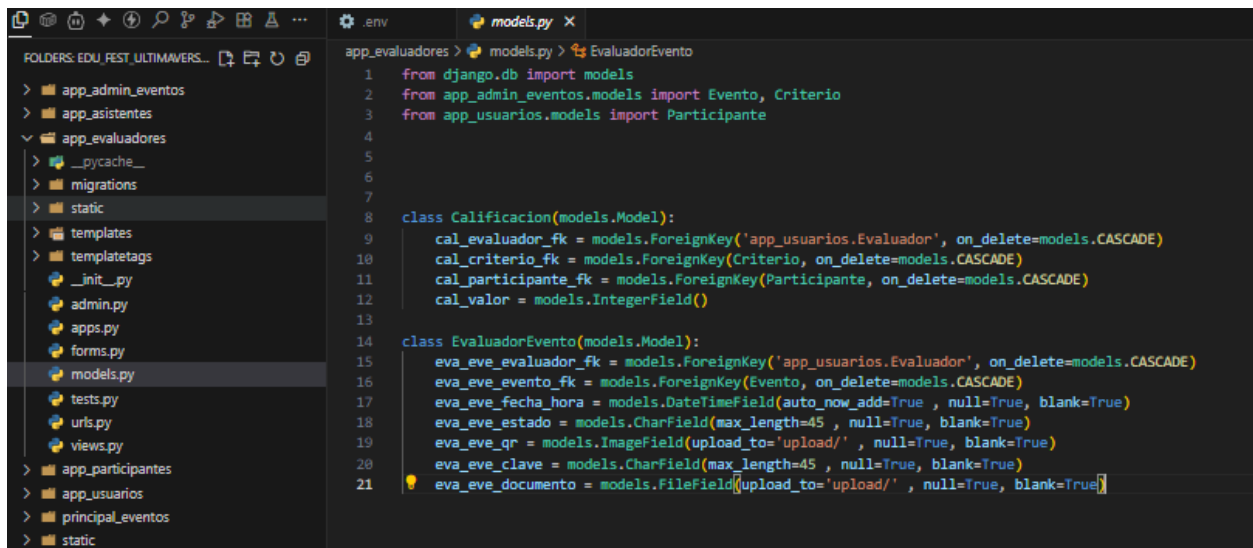
```
1 from django.db import models
2
3
4 class Area(models.Model):
5     are_nombre = models.CharField(max_length=45)
6     are_descripcion = models.CharField(max_length=400)
7
8     def __str__(self):
9         return self.are_nombre
10
11 class Categoria(models.Model):
12     cat_nombre = models.CharField(max_length=45)
13     cat_descripcion = models.CharField(max_length=400)
14     cat_area_fk = models.ForeignKey(Area, on_delete=models.CASCADE)
15
16     def __str__(self):
17         return self.cat_nombre
18
19 class Evento(models.Model):
20     eve_nombre = models.CharField(max_length=100)
21     eve_descripcion = models.CharField(max_length=520)
22     eve_ciudad = models.CharField(max_length=45)
23     eve_lugar = models.CharField(max_length=100)
24     eve_fecha_inicio = models.DateField()
25     eve_fecha_fin = models.DateField()
26     eve_estado = models.CharField(max_length=45)
27     eve_cancelacion_iniciada = models.DateTimeField(null=True, blank=True)
28     eve_imagen = models.ImageField(upload_to='upload/')
29     eve_administrador_fk = models.ForeignKey('app_usuarios.AdministradorEvento', on_delete=models.CASCADE)
30     eve_tienecosto = models.CharField(max_length=45)
31     eve_capacidad = models.IntegerField()
32     eve_programacion = models.FileField(upload_to='upload/')
33     eve_informacion_tecnica = models.FileField(upload_to='upload/', null=True, blank=True)
34     preinscripcion_habilitada_asistentes = models.BooleanField(default=False)
35     preinscripcion_habilitada_participantes = models.BooleanField(default=False)
36     preinscripcion_habilitada_evaluadores = models.BooleanField(default=False)
37     categorias = models.ManyToManyField(Categoria, through='EventoCategoria')
38
39     def __str__(self):
40         return self.eve_nombre
41
42 class EventoCategoria(models.Model):
43     eve_cat_evento_fk = models.ForeignKey(Evento, on_delete=models.CASCADE)
44     eve_cat_categoria_fk = models.ForeignKey(Categoria, on_delete=models.CASCADE)
45
46 class Criterio(models.Model):
47     cri_descripcion = models.CharField(max_length=150)
48     cri_peso = models.FloatField()
49     cri_evento_fk = models.ForeignKey(Evento, on_delete=models.CASCADE)
50
51     def __str__(self):
52         return self.cri_descripcion
53
54 class MemoriaEvento(models.Model):
55
56     evento = models.ForeignKey(Evento, on_delete=models.CASCADE, related_name='memorias')
57     nombre = models.CharField(max_length=100, help_text="Texto descriptivo para el archivo")
58     archivo = models.FileField(upload_to='upload/')
59     subido_en = models.DateTimeField(auto_now_add=True)
60
61     def __str__(self):
62         return f"{self.nombre} ({self.evento.eve_nombre})"
63
64
```

App_asistentes



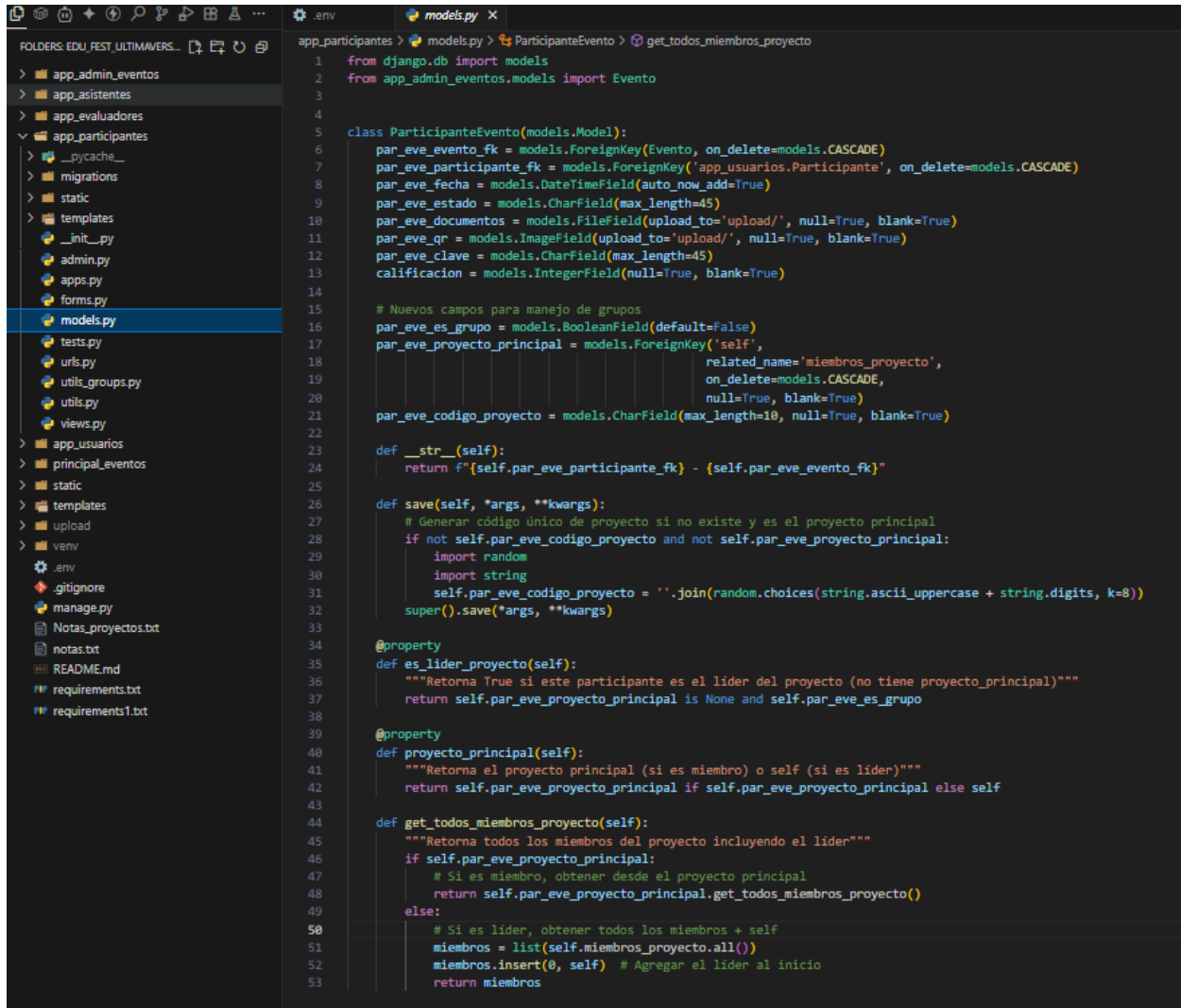
```
.env models.py X
app_asistentes > models.py > ...
1 from django.db import models
2 from app_admin_eventos.models import Evento
3
4
5
6 class AsistenteEvento(models.Model):
7     asi_eve_asistente_fk = models.ForeignKey('app_usuarios.Asistente', on_delete=models.CASCADE)
8     asi_eve_evento_fk = models.ForeignKey(Evento, on_delete=models.CASCADE)
9     asi_eve_fecha_hora = models.DateTimeField()
10    asi_eve_estado = models.CharField(max_length=45)
11    asi_eve_soporte = models.FileField(upload_to='upload/')
12    asi_eve_qr = models.ImageField(upload_to='upload/')
13    asi_eve_clave = models.CharField(max_length=45)
14
```

App_evaluadores



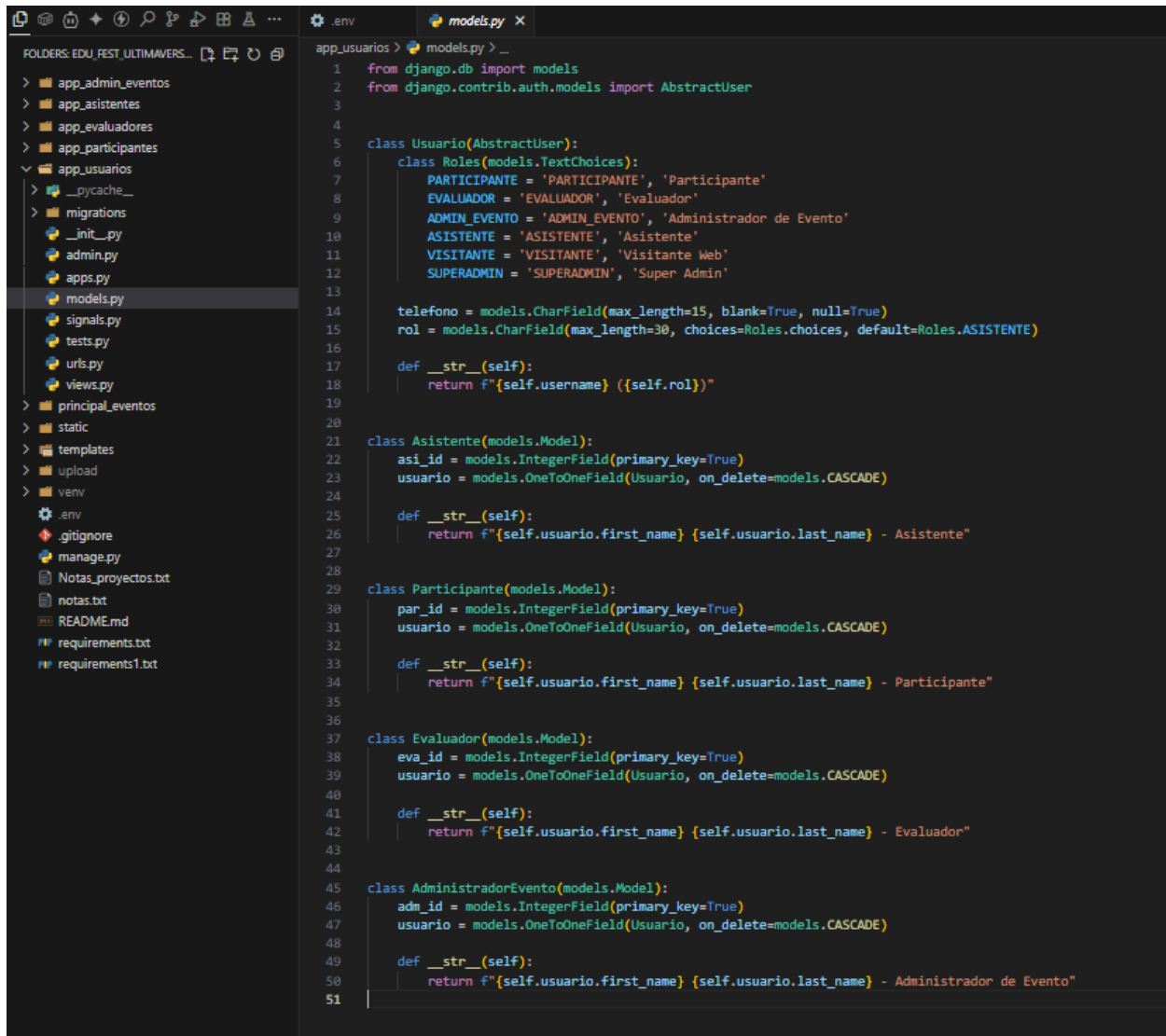
```
.env models.py X
app_evaluadores > models.py > EvaluadorEvento
1 from django.db import models
2 from app_admin_eventos.models import Evento, Criterio
3 from app_usuarios.models import Participante
4
5
6
7
8 class Calificacion(models.Model):
9     cal_evaluador_fk = models.ForeignKey('app_usuarios.Evaluador', on_delete=models.CASCADE)
10    cal_criterio_fk = models.ForeignKey(Criterio, on_delete=models.CASCADE)
11    cal_participante_fk = models.ForeignKey(Participante, on_delete=models.CASCADE)
12    cal_valor = models.IntegerField()
13
14 class EvaluadorEvento(models.Model):
15     eva_eve_evaluador_fk = models.ForeignKey('app_usuarios.Evaluador', on_delete=models.CASCADE)
16     eva_eve_evento_fk = models.ForeignKey(Evento, on_delete=models.CASCADE)
17     eva_eve_fecha_hora = models.DateTimeField(auto_now_add=True, null=True, blank=True)
18     eva_eve_estado = models.CharField(max_length=45, null=True, blank=True)
19     eva_eve_qr = models.ImageField(upload_to='upload/', null=True, blank=True)
20     eva_eve_clave = models.CharField(max_length=45, null=True, blank=True)
21     eva_eve_documento = models.FileField(upload_to='upload/', null=True, blank=True)
```

App_participantes



```
1 from django.db import models
2 from app_admin_eventos.models import Evento
3
4
5 class ParticipanteEvento(models.Model):
6     par_eve_evento_fk = models.ForeignKey(Evento, on_delete=models.CASCADE)
7     par_eve_participante_fk = models.ForeignKey('app_usuarios.Participante', on_delete=models.CASCADE)
8     par_eve_fecha = models.DateTimeField(auto_now_add=True)
9     par_eve_estado = models.CharField(max_length=45)
10    par_eve_documentos = models.FileField(upload_to='upload/', null=True, blank=True)
11    par_eve_gr = models.ImageField(upload_to='upload/', null=True, blank=True)
12    par_eve_clave = models.CharField(max_length=45)
13    calificacion = models.IntegerField(null=True, blank=True)
14
15    # Nuevos campos para manejo de grupos
16    par_eve_es_grupo = models.BooleanField(default=False)
17    par_eve_proyecto_principal = models.ForeignKey('self',
18                                                  related_name='miembros_proyecto',
19                                                  on_delete=models.CASCADE,
20                                                  null=True, blank=True)
21    par_eve_codigo_proyecto = models.CharField(max_length=10, null=True, blank=True)
22
23    def __str__(self):
24        return f"{self.par_eve_participante_fk} - {self.par_eve_evento_fk}"
25
26    def save(self, *args, **kwargs):
27        # Generar código único de proyecto si no existe y es el proyecto principal
28        if not self.par_eve_codigo_proyecto and not self.par_eve_proyecto_principal:
29            import random
30            import string
31            self.par_eve_codigo_proyecto = ''.join(random.choices(string.ascii_uppercase + string.digits, k=8))
32            super().save(*args, **kwargs)
33
34    @property
35    def es_lider_proyecto(self):
36        """Retorna True si este participante es el lider del proyecto (no tiene proyecto_principal)"""
37        return self.par_eve_proyecto_principal is None and self.par_eve_es_grupo
38
39    @property
40    def proyecto_principal(self):
41        """Retorna el proyecto principal (si es miembro) o self (si es lider)"""
42        return self.par_eve_proyecto_principal if self.par_eve_proyecto_principal else self
43
44    def get_todos_miembros_proyecto(self):
45        """Retorna todos los miembros del proyecto incluyendo el lider"""
46        if self.par_eve_proyecto_principal:
47            # Si es miembro, obtener desde el proyecto principal
48            return self.par_eve_proyecto_principal.get_todos_miembros_proyecto()
49        else:
50            # Si es lider, obtener todos los miembros + self
51            miembros = list(self.miembros_proyecto.all())
52            miembros.insert(0, self) # Agregar el lider al inicio
53            return miembros
```

App_usuarios, lo cual el usuario hereda de **AbstractUser** del modelo Django sacando provecho de este modelo.



```
1 from django.db import models
2 from django.contrib.auth.models import AbstractUser
3
4
5 class Usuario(AbstractUser):
6     class Roles(models.TextChoices):
7         PARTICIPANTE = 'PARTICIPANTE', 'Participante'
8         EVALUADOR = 'EVALUADOR', 'Evaluador'
9         ADMIN_EVENTO = 'ADMIN_EVENTO', 'Administrador de Evento'
10        ASISTENTE = 'ASISTENTE', 'Asistente'
11        VISITANTE = 'VISITANTE', 'Visitante Web'
12        SUPERADMIN = 'SUPERADMIN', 'Super Admin'
13
14        telefono = models.CharField(max_length=15, blank=True, null=True)
15        rol = models.CharField(max_length=30, choices=Roles.choices, default=Roles.ASISTENTE)
16
17        def __str__(self):
18            return f"{self.username} ({self.rol})"
19
20
21 class Asistente(models.Model):
22     asi_id = models.IntegerField(primary_key=True)
23     usuario = models.OneToOneField(Usuario, on_delete=models.CASCADE)
24
25     def __str__(self):
26         return f"{self.usuario.first_name} {self.usuario.last_name} - Asistente"
27
28
29 class Participante(models.Model):
30     par_id = models.IntegerField(primary_key=True)
31     usuario = models.OneToOneField(Usuario, on_delete=models.CASCADE)
32
33     def __str__(self):
34         return f"{self.usuario.first_name} {self.usuario.last_name} - Participante"
35
36
37 class Evaluador(models.Model):
38     eva_id = models.IntegerField(primary_key=True)
39     usuario = models.OneToOneField(Usuario, on_delete=models.CASCADE)
40
41     def __str__(self):
42         return f"{self.usuario.first_name} {self.usuario.last_name} - Evaluador"
43
44
45 class AdministradorEvento(models.Model):
46     adm_id = models.IntegerField(primary_key=True)
47     usuario = models.OneToOneField(Usuario, on_delete=models.CASCADE)
48
49     def __str__(self):
50         return f"{self.usuario.first_name} {self.usuario.last_name} - Administrador de Evento"
51
```

DESPLIEGUE PASO A PASO

Despliegue con Docker

Crear un archivo en la carpeta raíz de proyecto llamada **Dockerfile**

```
dockerfile

# Imagen base de Python
FROM python:3.11-slim

# Configuración del directorio de trabajo
WORKDIR /app

# Instalar dependencias del sistema
RUN apt-get update && apt-get install -y libpq-dev gcc

# Copiar los archivos de requerimientos
COPY requirements.txt /app/

# Instalar dependencias de Python
RUN pip install --no-cache-dir -r requirements.txt

# Copiar todo el proyecto al contenedor
COPY . /app/

# Exponer el puerto
EXPOSE 8000

# Comando para ejecutar la aplicación
CMD ["gunicorn", "tu_proyecto.wsgi:application", "--bind", "0.0.0.0:8000"]
```

Luego crear el **docker-compose.yml**

```
yaml Copiar código

version: '3.9'

services:
  db:
    image: postgres:15
    environment:
      POSTGRES_DB: bd_edufest
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: admin123
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

  web:
    build: .
    command: gunicorn tu_proyecto.wsgi:application --bind 0.0.0.0:8000
    volumes:
      - ./app
    ports:
      - "8000:8000"
    depends_on:
      - db

volumes:
  postgres_data:
```

Tener en cuenta el archivo .env donde se sacara los datos de la base de datos

Luego levantar contenedores con Docker

```
bash
```

```
# Construir la imagen
```

```
docker-compose build
```

```
# Levantar los contenedores
```

```
docker-compose up -d
```

Luego escribe en tu navegador

```
arduino
```

```
http://localhost:8000
```

CONCLUSIÓN

El presente manual técnico documenta de manera estructurada los componentes, la arquitectura y los procedimientos necesarios para la instalación, configuración, despliegue y mantenimiento de la aplicación. Gracias a esta guía, se facilita el trabajo de los administradores del sistema y desarrolladores, asegurando la correcta comprensión del funcionamiento interno y de las dependencias que requiere la solución.

La aplicación fue diseñada siguiendo principios de modularidad, escalabilidad y mantenibilidad, lo que permite su evolución en el tiempo y su adaptación a nuevos requerimientos sin comprometer la estabilidad. El uso de tecnologías modernas como Django, MySQL/PostgreSQL y Docker garantiza un entorno seguro, estandarizado y fácilmente replicable en distintas fases del ciclo de vida del software (desarrollo, pruebas y producción).

Finalmente, este manual busca servir como un documento de referencia para futuros desarrolladores y equipos de soporte, aportando lineamientos claros sobre cómo extender funcionalidades, gestionar cambios y garantizar la continuidad operativa del sistema.