Necesito que me digas:

### \*\*1. 📝 Campos del modelo TAREA:\*\*

class Confecciones(models.Model):

pedido = models.ForeignKey('pedidos.Pedido', on\_delete=models.CASCADE, null=True, blank=True) fecha = models.DateTimeField(auto\_now=True,verbose\_name='Fecha de Registro')

fec\_entrega = models.DateTimeField(null=False,verbose\_name='Fecha de Entrega')

fec\_terminado = models.DateTimeField(null=True,blank=True,verbose\_name='Fecha de TerminaciÃƒÂ³n')

contacto = models.CharField(max\_length=20,verbose\_name='Contacto')

tel\_contacto = models.CharField(max\_length=50,verbose\_name='Tel. Contacto')

estado = models.BooleanField(default=True,verbose\_name='Activo')

obs = models.CharField(max\_length=50,null=True,blank=True,verbose\_name='Observaciones')

def save(self, \*args, \*\*kwargs):

for field in self.\_meta.fields:

if isinstance(field, models.CharField):

valor = getattr(self, field.name)

if isinstance(valor, str):

setattr(self, field.name, valor.upper())

super().save(\*args, \*\*kwargs)

class Meta:

verbose\_name = 'ConfecciÃƒÂ³n'

verbose\_name\_plural = 'Confecciones'

ordering = ['-fecha']

def \_\_str\_\_(self) -> str:

return f"{self.contacto} - cliente {self.cliente.nombre}"

@property

def deuda(self):

cuenta = self.cuentas\_set.first()

return cuenta.deuda if cuenta else 0

@property

def pagado\_total(self):

cuenta = self.cuentas\_set.first()

if cuenta:

return sum(det.monto for det in cuenta.cuentas\_detalles\_set.all())

return 0

class Refcodes(models.Model):

tipo = models.CharField(max\_length=10)

valor = models.CharField(max\_length=10)

descripcion = models.CharField(max\_length=20)

estado = models.BooleanField(default=True)

fecha = models.DateTimeField(auto\_now=True)

obs = models.CharField(max\_length=50, null=True, blank=True)

def save(self, \*args, \*\*kwargs):

for field in self.\_meta.fields:

if isinstance(field, models.CharField):

valor = getattr(self, field.name)

if isinstance(valor, str):

setattr(self, field.name, valor.upper())

super().save(\*args, \*\*kwargs)

def \_\_str\_\_(self) -> str:

return self.tipo + ' valor ' + self.valor + ' descripcion ' + self.descripcion

class Detalles(models.Model):

confecciones = models.ForeignKey(Confecciones, on\_delete=models.CASCADE)

genero = models.CharField(max\_length=10)

nombre = models.CharField(max\_length=50, null=True, blank=True)

numero = models.IntegerField(null=True,blank=True,validators=[MinValueValidator(0), MaxValueValidator(999)])

tamano = models.CharField(max\_length=10)

item = models.CharField(max\_length=10, null=True, blank=True)

item\_adic = models.BooleanField(default=False)

estado = models.BooleanField(default=True)

fecha = models.DateTimeField(auto\_now=True)

obs = models.CharField(max\_length=50, null=True, blank=True)

def save(self, \*args, \*\*kwargs):

for field in self.\_meta.fields:

if isinstance(field, models.CharField):

valor = getattr(self, field.name)

if isinstance(valor, str):

setattr(self, field.name, valor.upper())

super().save(\*args, \*\*kwargs)

def \_\_str\_\_(self) -> str:

return f"{self.confecciones.cliente.nombre} - {self.nombre} - {self.genero} - {self.tamano}"

class ItemAdicional(models.Model):

detalles = models.ForeignKey(Detalles, on\_delete=models.CASCADE, related\_name="items\_adicionales")

item = models.CharField(max\_length=15)

fecha = models.DateTimeField(auto\_now=True)

estado = models.BooleanField(default=True)

obs = models.CharField(max\_length=50, null=True, blank=True)

def save(self, \*args, \*\*kwargs):

for field in self.\_meta.fields:

if isinstance(field, models.CharField):

valor = getattr(self, field.name)

if isinstance(valor, str):

setattr(self, field.name, valor.upper())

super().save(\*args, \*\*kwargs)

class cuentas(models.Model):

confecciones = models.ForeignKey(Confecciones, on\_delete=models.CASCADE)

deuda = models.IntegerField(MinValueValidator(0))

montoPago = models.IntegerField()

estado = models.BooleanField(default=True)

fecha = models.DateTimeField(auto\_now=True)

obs = models.CharField(max\_length=50, null=True, blank=True)

def save(self, \*args, \*\*kwargs):

for field in self.\_meta.fields:

if isinstance(field, models.CharField):

valor = getattr(self, field.name)

if isinstance(valor, str):

setattr(self, field.name, valor.upper())

super().save(\*args, \*\*kwargs)

class cuentas\_detalles(models.Model):

cuentas = models.ForeignKey(cuentas, on\_delete=models.CASCADE)

monto = models.IntegerField(MinValueValidator(1000))

estado = models.BooleanField(default=True)

fecha = models.DateTimeField(auto\_now=True)

obs = models.CharField(max\_length=50, null=True, blank=True)

def save(self, \*args, \*\*kwargs):

for field in self.\_meta.fields:

if isinstance(field, models.CharField):

valor = getattr(self, field.name)

if isinstance(valor, str):

setattr(self, field.name, valor.upper())

super().save(\*args, \*\*kwargs)

- ¿Qué información debe tener cada tarea?

- ¿Estado, prioridad, asignado a, fecha límite?

- ¿Descripción de la tarea específica?

### \*\*2. 🎯 Tipos de tareas:\*\*

- ¿Solo tareas de confección de equipos?

También puede ser la confección de prendas individuales, como camiseta, short, medias(no confeccionamos pero podemos tener en stock), musculosas, canguros (también puede ser con el pantalón), gorras, etc.

- ¿O también otros tipos de tareas internas?

### \*\*3. 👥 Asignación:\*\*

- ¿Las tareas se asignan a empleados específicos?

Cualquier empleado puede administrar los pedidos, para eso necesito la implementación de auditoría, para registrar las actualizaciones de los datos de los pedidos y las tareas.

- ¿O son para un departamento en general?

Por ahora no.

### \*\*4. 📊 Estados de tarea:\*\*

- ¿Pendiente, en progreso, completada?

Están bien estos estados.

- ¿Otros estados específicos?

## 🚀 \*\*Mientras tanto, puedo preparar la base:\*\*

\*\*Modelo base de Tarea\*\* (que puedes modificar después):

```python

# tareas/models.py (base)

from django.db import models

from usuarios.models import Usuario

class Tarea(models.Model):

ESTADOS\_TAREA = (

('pendiente', 'Pendiente'),

('en\_progreso', 'En Progreso'),

('completada', 'Completada'),

('cancelada', 'Cancelada'),

)

PRIORIDADES = (

('baja', 'Baja'),

('media', 'Media'),

('alta', 'Alta'),

('urgente', 'Urgente'),

)

titulo = models.CharField(max\_length=200)

descripcion = models.TextField()

estado = models.CharField(max\_length=20, choices=ESTADOS\_TAREA, default='pendiente')

prioridad = models.CharField(max\_length=20, choices=PRIORIDADES, default='media')

asignado\_a = models.ForeignKey(Usuario, on\_delete=models.SET\_NULL, null=True, blank=True)

fecha\_creacion = models.DateTimeField(auto\_now\_add=True)

fecha\_limite = models.DateTimeField(null=True, blank=True)

fecha\_completada = models.DateTimeField(null=True, blank=True)

# Relación con el pedido que generó esta tarea

pedido = models.ForeignKey('pedidos.Pedido', on\_delete=models.CASCADE, null=True, blank=True)

class Meta:

ordering = ['-prioridad', 'fecha\_limite']

def \_\_str\_\_(self):

return f"{self.titulo} - {self.get\_estado\_display()}"

```

## 🔄 \*\*Integración automática:\*\*

Cuando se cree un pedido de equipo, automáticamente se creará una tarea:

```python

# pedidos/models.py (ejemplo)

def crear\_tarea\_desde\_pedido(pedido):

if pedido.es\_pedido\_equipo: # Tu lógica específica

Tarea.objects.create(

titulo=f"Confección de equipo - Pedido #{pedido.id}",

descripcion=f"Pedido de equipo para {pedido.cliente}",

prioridad='alta',

pedido=pedido,

# asignado\_a=... (si tienes lógica de asignación)

)

```

\*\*¿Este enfoque va por buen camino?\*\* Por favor, compárteme los campos específicos que necesitas para que modele exactamente lo que requieres. 🛠️