# Esqueleto `models.py` (Django) — pronto para colar

Observação: usei convenções compatíveis com o projeto (apps com prefixo `app_`).
Ajuste imports e `app_label` se necessário.

```
# app_raffles/models.py
from django.db import models
from django.utils import timezone

class Rifa(models.Model):
    STATUS_CHOICES = [
        ('draft', 'Rascunho'),
        ('ativa', 'Ativa'),
        ('encerrada', 'Encerrada'),
        ('sorteada', 'Sorteada'),
    ]

    titulo = models.CharField(max_length=200)
    descricao = models.TextField(blank=True)
    item = models.CharField(max_length=200)  # ex: "Carro Fiat Uno"
    preco_bilhete = models.DecimalField(max_digits=10, decimal_places=2)
    quantidade_bilhetes = models.PositiveIntegerField()
    data_sorteio = models.DateTimeField()
    concurso = models.CharField(max_length=50, blank=True, null=True)  # número da
Loteria (opcional)
    usar_api_loteria = models.BooleanField(default=False)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default='draft')
    criado_em = models.DateTimeField(auto_now_add=True)
    atualizado_em = models.DateTimeField(auto_now=True)

    class Meta:
        ordering = ['-criado_em']

    def __str__(self):
        return f"{self.titulo} ({self.id})"

class Participante(models.Model):
    nome = models.CharField(max_length=200)
    cpf = models.CharField(max_length=14, unique=False)  # validação em
forms/serializers
    email = models.EmailField(blank=True, null=True)
    telefone = models.CharField(max_length=30, blank=True, null=True)
    endereco = models.TextField(blank=True, null=True)
    criado_em = models.DateTimeField(auto_now_add=True)

    class Meta:
```

```
        ordering = ['nome']

    def __str__(self):
        return f"{self.nome} - {self.cpf}"

class Bilhete(models.Model):
    ESTADO_CHOICES = [
        ('disponivel', 'Disponível'),
        ('reservado', 'Reservado'),
        ('vendido', 'Vendido'),
    ]

    rifa = models.ForeignKey(Rifa, on_delete=models.CASCADE, related_name='bilhetes')
    numero = models.PositiveIntegerField()
    estado = models.CharField(max_length=20, choices=ESTADO_CHOICES,
default='disponivel')
    participante = models.ForeignKey(Participante, on_delete=models.SET_NULL,
null=True, blank=True, related_name='bilhetes')
    reservado_em = models.DateTimeField(null=True, blank=True)
    vendido_em = models.DateTimeField(null=True, blank=True)

    class Meta:
        unique_together = ('rifa', 'numero')
        ordering = ['numero']

    def __str__(self):
        return f"{self.rifa.titulo} - #{self.numero}"

class Transacao(models.Model):
    STATUS_CHOICES = [
        ('pendente', 'Pendente'),
        ('pago', 'Pago'),
        ('cancelado', 'Cancelado'),
        ('estornado', 'Estornado'),
    ]

    rifa = models.ForeignKey(Rifa, on_delete=models.CASCADE, related_name='transacoes')
    participante = models.ForeignKey(Participante, on_delete=models.CASCADE,
related_name='transacoes')
    bilhetes = models.ManyToManyField(Bilhete, related_name='transacoes')
    valor_total = models.DecimalField(max_digits=12, decimal_places=2)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default='pendente')
    gateway_id = models.CharField(max_length=200, blank=True, null=True)  # id do
provedor de pagamento
    criado_em = models.DateTimeField(auto_now_add=True)
    atualizado_em = models.DateTimeField(auto_now=True)

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

```
    def __str__(self):
        return f"Transacao {self.id} - {self.participante.nome} - {self.valor_total}"

class Apuracao(models.Model):
    FONTE_CHOICES = [
        ('api', 'API Loteria Federal'),
        ('manual', 'Manual'),
    ]

    rifa = models.OneToOneField(Rifa, on_delete=models.CASCADE,
related_name='apuracao')
    fonte = models.CharField(max_length=20, choices=FONTE_CHOICES)
    concurso = models.CharField(max_length=50, blank=True, null=True)
    numero_sorteado = models.CharField(max_length=50)  # pode ser numérico ou
alfanumérico
    bilhete_vencedor = models.ForeignKey(Bilhete, on_delete=models.SET_NULL, null=True,
related_name='+')
    participante_vencedor = models.ForeignKey(Participante, on_delete=models.SET_NULL,
null=True, related_name='+')
    data_registro = models.DateTimeField(default=timezone.now)
    registro_prova = models.JSONField(blank=True, null=True)  # JSON com evidências
(ex: payload da API)

    def __str__(self):
        return f"Apuração Rifa {self.rifa.id} - {self.numero_sorteado}"
```

---

## Observações

- Ao colar o `models.py`, mantenha o arquivo dentro de `app_raffles/models.py` (ou o nome do app que preferir, lembrando do prefixo `app_`).
- Validações adicionais (CPF, integridade, limites de compra) implementá-las em `forms`, `serializers` ou `domain services`.
- Para gerar os bilhetes, crie um comando management `python manage.py gerar_bilhetes <rifa_id>` que popula `Bilhete(numero=1..N)` respeitando `unique_together`.
- Para concorrência ao vender bilhetes: use transação atômica + `SELECT ... FOR UPDATE` ou locking via Redis (ex.: setnx) + TTL para reservas.