from django.db import models

from django.contrib.auth.models import AbstractUser

# Usuário Customizado

class User(AbstractUser):

email = models.EmailField(unique=True)

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

return self.username

# Categorias (Receitas, Despesas, Vendas)

class Category(models.Model):

TYPE\_CHOICES = [

('income', 'Receita'),

('expense', 'Despesa'),

('sale', 'Venda'),

]

user = models.ForeignKey(User, on\_delete=models.CASCADE)

name = models.CharField(max\_length=100)

type = models.CharField(max\_length=10, choices=TYPE\_CHOICES)

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

return f"{self.name} ({self.get\_type\_display()})"

# Receitas

class Income(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

category = models.ForeignKey(Category, limit\_choices\_to={'type': 'income'}, on\_delete=models.SET\_NULL, null=True, blank=True)

amount = models.DecimalField(max\_digits=12, decimal\_places=2)

description = models.CharField(max\_length=255, blank=True, null=True)

received\_at = models.DateField()

created\_at = models.DateTimeField(auto\_now\_add=True)

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

return f"R$ {self.amount} - {self.received\_at}"

# Despesas

class Expense(models.Model):

PAYMENT\_CHOICES = [

('cash', 'Dinheiro'),

('credit\_card', 'Cartão de Crédito'),

('pix', 'PIX'),

('transfer', 'Transferência'),

]

user = models.ForeignKey(User, on\_delete=models.CASCADE)

category = models.ForeignKey(Category, limit\_choices\_to={'type': 'expense'}, on\_delete=models.SET\_NULL, null=True, blank=True)

amount = models.DecimalField(max\_digits=12, decimal\_places=2)

description = models.CharField(max\_length=255, blank=True, null=True)

paid\_at = models.DateField()

payment\_method = models.CharField(max\_length=50, choices=PAYMENT\_CHOICES)

is\_installment = models.BooleanField(default=False)

installment\_number = models.PositiveIntegerField(null=True, blank=True)

total\_installments = models.PositiveIntegerField(null=True, blank=True)

created\_at = models.DateTimeField(auto\_now\_add=True)

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

return f"R$ {self.amount} - {self.paid\_at}"

# Vendas

class Sale(models.Model):

PAYMENT\_CHOICES = [

('cash', 'Dinheiro'),

('pix', 'PIX'),

('transfer', 'Transferência'),

]

user = models.ForeignKey(User, on\_delete=models.CASCADE)

product\_or\_service = models.CharField(max\_length=255)

client = models.CharField(max\_length=255, blank=True, null=True)

amount = models.DecimalField(max\_digits=12, decimal\_places=2)

received\_at = models.DateField()

payment\_method = models.CharField(max\_length=50, choices=PAYMENT\_CHOICES)

created\_at = models.DateTimeField(auto\_now\_add=True)

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

return f"Venda: {self.product\_or\_service} - R$ {self.amount}"

# Cartão de Crédito

class CreditCard(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

name = models.CharField(max\_length=100)

limit = models.DecimalField(max\_digits=12, decimal\_places=2)

closing\_day = models.PositiveSmallIntegerField()

due\_day = models.PositiveSmallIntegerField()

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

return self.name

# Transações no Cartão de Crédito

class CreditCardTransaction(models.Model):

card = models.ForeignKey(CreditCard, on\_delete=models.CASCADE, related\_name='transactions')

category = models.ForeignKey(Category, limit\_choices\_to={'type': 'expense'}, on\_delete=models.SET\_NULL, null=True, blank=True)

amount = models.DecimalField(max\_digits=12, decimal\_places=2)

description = models.CharField(max\_length=255)

transaction\_date = models.DateField()

is\_installment = models.BooleanField(default=False)

installment\_number = models.PositiveIntegerField(null=True, blank=True)

total\_installments = models.PositiveIntegerField(null=True, blank=True)

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

return f"{self.description} - R$ {self.amount}"

# Investimentos

class Investment(models.Model):

INVESTMENT\_TYPES = [

('renda\_fixa', 'Renda Fixa'),

('renda\_variavel', 'Renda Variável'),

('cripto', 'Cripto'),

('outros', 'Outros'),

]

user = models.ForeignKey(User, on\_delete=models.CASCADE)

name = models.CharField(max\_length=100)

type = models.CharField(max\_length=50, choices=INVESTMENT\_TYPES)

amount = models.DecimalField(max\_digits=12, decimal\_places=2)

applied\_at = models.DateField()

current\_value = models.DecimalField(max\_digits=12, decimal\_places=2, null=True, blank=True)

monthly\_return\_percentage = models.DecimalField(max\_digits=5, decimal\_places=2, null=True, blank=True)

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

return f"{self.name} - R$ {self.amount}"

# Orçamento

class BudgetPlan(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

total\_income\_base = models.DecimalField(max\_digits=12, decimal\_places=2)

month\_reference = models.DateField()

created\_at = models.DateTimeField(auto\_now\_add=True)

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

return f"Orçamento - {self.month\_reference}"

# Limite por Categoria no Orçamento

class BudgetCategoryLimit(models.Model):

budget\_plan = models.ForeignKey(BudgetPlan, on\_delete=models.CASCADE, related\_name='categories')

category = models.ForeignKey(Category, on\_delete=models.CASCADE)

percentage = models.DecimalField(max\_digits=5, decimal\_places=2)

amount\_limit = models.DecimalField(max\_digits=12, decimal\_places=2)

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

return f"{self.category.name}: {self.percentage}%"