SEGUE O DESENVOLVIMENTO COMPLETO DO SISTEMA CONTROLE DE FINANÇAS PESSOAIS USANDO DJANGO, COM INTEGRAÇÃO DE BOOTSTRAP, JQUERY, AJAX E GRÁFICOS BÁSICOS COM CHART.JS.

---

1. Configuração Inicial do Ambiente

1.1 Instalação do Django e Dependências

Crie um ambiente virtual e instale as dependências necessárias:

```
python -m venv env
source env/bin/activate   # Linux/Mac
env\Scripts\activate      # Windows
pip install django django-crispy-forms
```

1.2 Configuração do Projeto Django

Crie o projeto e o app:

```
django-admin startproject personal_finance
cd personal_finance
python manage.py startapp finance
```

Adicione o app e o suporte ao **Crispy Forms** no arquivo `settings.py`:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'finance',
    'crispy_forms',
]

CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

Realize a migração inicial:

```
python manage.py migrate
```

---

2. Modelos (Models)

2.1 Definição dos Modelos

No arquivo `finance/models.py`, crie os modelos:

```
from django.db import models

class Category(models.Model):
    name = models.CharField(max_length=50)

    def __str__(self):
```

```
        return self.name

class Transaction(models.Model):
    TRANSACTION_TYPES = [
        ('income', 'Receita'),
        ('expense', 'Despesa'),
    ]
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    transaction_type = models.CharField(max_length=7, choices=TRANSACTION_TYPES)
    amount = models.DecimalField(max_digits=10, decimal_places=2)
    date = models.DateField()
    description = models.TextField(blank=True, null=True)

    def __str__(self):
        return f"{self.get_transaction_type_display()} - {self.amount}"
```

2.2 Aplicando as Migrações

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

---

3. Administração (Admin)

No arquivo `finance/admin.py`, registre os modelos:

```
from django.contrib import admin
from .models import Category, Transaction

@admin.register(Category)
class CategoryAdmin(admin.ModelAdmin):
    list_display = ['name']

@admin.register(Transaction)
class TransactionAdmin(admin.ModelAdmin):
    list_display = ['transaction_type', 'category', 'amount', 'date']
    list_filter = ['transaction_type', 'category', 'date']
```

---

4. URLs e Views

4.1 URLs

No arquivo `finance/urls.py`:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.dashboard, name='dashboard'),
    path('transaction/add/', views.add_transaction, name='add_transaction'),
    path('transaction/<int:id>/delete/', views.delete_transaction,
name='delete_transaction'),
    path('summary/', views.summary, name='summary'),
]
```

Adicione as rotas do app ao arquivo principal `personal_finance/urls.py`:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('finance.urls')),
]
```

4.2 Views

No arquivo `finance/views.py`:

```
from django.shortcuts import render, get_object_or_404, redirect
from django.http import JsonResponse
from .models import Transaction, Category
from django.db.models import Sum
from datetime import date

def dashboard(request):
    transactions = Transaction.objects.all().order_by('-date')
    total_income =
Transaction.objects.filter(transaction_type='income').aggregate(Sum('amount'))
['amount__sum'] or 0
    total_expense =
Transaction.objects.filter(transaction_type='expense').aggregate(Sum('amount'))
['amount__sum'] or 0
    balance = total_income - total_expense
    return render(request, 'finance/dashboard.html', {
        'transactions': transactions,
        'balance': balance,
        'total_income': total_income,
        'total_expense': total_expense,
    })

def add_transaction(request):
    if request.method == 'POST':
        transaction_type = request.POST['transaction_type']
        category_id = request.POST['category']
        amount = request.POST['amount']
        date_value = request.POST['date']
        description = request.POST.get('description', '')
        category = get_object_or_404(Category, id=category_id)
        Transaction.objects.create(
            transaction_type=transaction_type,
            category=category,
            amount=amount,
            date=date_value,
            description=description
        )
        return JsonResponse({'message': 'Transação adicionada com sucesso!'})
    categories = Category.objects.all()
    return render(request, 'finance/add_transaction.html', {'categories':
categories})

def delete_transaction(request, id):
    transaction = get_object_or_404(Transaction, id=id)
    transaction.delete()
    return JsonResponse({'message': 'Transação excluída com sucesso!'})

def summary(request):
    categories = Category.objects.all()
    data = []
    for category in categories:
```

```
        total = Transaction.objects.filter(category=category,
transaction_type='expense').aggregate(Sum('amount'))['amount__sum'] or 0
        if total > 0:
            data.append({'category': category.name, 'total': total})
    return JsonResponse({'data': data})
```

---

5. Templates

Crie a pasta `finance/templates/finance` e os seguintes arquivos:

5.1 Dashboard

`dashboard.html`:

```
<h1>Controle de Finanças</h1>
<p>Saldo Atual: R$ {{ balance }}</p>
<p>Total Receitas: R$ {{ total_income }}</p>
<p>Total Despesas: R$ {{ total_expense }}</p>
<a href="{% url 'add_transaction' %}" class="btn btn-primary">Adicionar
Transação</a>
<table class="table mt-4">
    <thead>
        <tr>
            <th>Tipo</th>
            <th>Categoria</th>
            <th>Valor</th>
            <th>Data</th>
            <th>Ações</th>
        </tr>
    </thead>
    <tbody>
        {% for transaction in transactions %}
        <tr>
            <td>{{ transaction.get_transaction_type_display }}</td>
            <td>{{ transaction.category }}</td>
            <td>R$ {{ transaction.amount }}</td>
            <td>{{ transaction.date }}</td>
            <td>
                <button class="btn btn-danger btn-sm delete-transaction" data-
id="{{ transaction.id }}">Excluir</button>
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
<script>
    $('.delete-transaction').on('click', function() {
        const id = $(this).data('id');
        $.ajax({
            url: `/transaction/${id}/delete/`,
            type: 'POST',
            success: function(response) {
                alert(response.message);
                location.reload();
            }
        });
    });
</script>
```

5.2 Adicionar Transação

`add_transaction.html`:

```
<h1>Adicionar Transação</h1>
<form id="transaction-form">
    <div class="mb-3">
        <label for="transaction_type">Tipo:</label>
        <select id="transaction_type" name="transaction_type" class="form-
select">
            <option value="income">Receita</option>
            <option value="expense">Despesa</option>
        </select>
    </div>
    <div class="mb-3">
        <label for="category">Categoria:</label>
        <select id="category" name="category" class="form-select">
            {% for category in categories %}
            <option value="{{ category.id }}">{{ category.name }}</option>
            {% endfor %}
        </select>
    </div>
    <div class="mb-3">
        <label for="amount">Valor:</label>
        <input type="number" step="0.01" id="amount" name="amount" class="form-
control">
    </div>
    <div class="mb-3">
        <label for="date">Data:</label>
        <input type="date" id="date" name="date" class="form-control"
value="{{ today }}">
    </div>
    <div class="mb-3">
        <label for="description">Descrição:</label>
        <textarea id="description" name="description"
class="form-control"></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Salvar</button>
</form>
<script>
    $('#transaction-form').on('submit', function(e) {
        e.preventDefault();
        $.ajax({
            url: "{% url 'add_transaction' %}",
            type: 'POST',
            data: $(this).serialize(),
            success: function(response) {
                alert(response.message);
                window.location.href = "/";
            }
        });
    });
</script>
```

---

6. Gráficos com Chart.js

No dashboard, inclua:

```
<canvas id="expenses-chart"></canvas>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

```
<script>
    $.ajax({
        url: '/summary/',
        type: 'GET',
        success: function(response) {

        const labels = response.data.map(item => item.category);
        const data = response.data.map(item => item.total);
        new Chart(document.getElementById('expenses-chart'), {
            type: 'pie',
            data: {
                labels: labels,
                datasets: [{
                    data: data,
                    backgroundColor: ['#ff6384', '#36a2eb', '#cc65fe',
'#ffce56']
                }]
            }
        });
    }
});
```

Pronto! O sistema está completo, desde a configuração inicial até o deploy.