CENTRO UNIVERSITÁRIO FAESA
UNIDADE CARIACICA
CURSO DE SISTEMAS DE INFORMAÇÃO

RAÍSSA DE AZEVEDO

SISTEMA DE CONTROLE DE GASTOS PESSOAIS COM INTERFACE EM PYTHON

UTILIZANDO STREAMLIT

CARIACICA
2025

RAÍSSA DE AZEVEDO

SISTEMA DE CONTROLE DE GASTOS PESSOAIS COM INTERFACE EM PYTHON

UTILIZANDO STREAMLIT

CARIACICA
2025

SUMÁRIO

1 INTRODUÇÃO

Muitas pessoas têm dificuldade em controlar suas finanças pessoais. Gastos pequenos do dia a dia, como lanches, transporte e lazer, acabam não sendo registrados, o que gera uma sensação de falta de controle e, muitas vezes, endividamento.

Sem um acompanhamento organizado, fica difícil saber quanto se pode gastar, quanto já foi gasto em cada categoria e quanto ainda sobra no orçamento mensal.

O gerenciamento de finanças pessoais é uma dificuldade comum para muitas pessoas. Pequenas despesas do dia a dia, como transporte, alimentação ou lazer, frequentemente deixam de ser registradas, o que compromete a clareza sobre os gastos ao final do mês. Essa falta de organização gera incertezas quanto ao orçamento disponível, dificulta o planejamento de despesas futuras e pode contribuir para situações de endividamento.

Além disso, a prática de registrar manualmente os gastos em cadernos ou planilhas pode ser trabalhosa e pouco prática, levando ao abandono do hábito de acompanhamento financeiro. A ausência de ferramentas simples e acessíveis acaba se tornando uma barreira para o controle efetivo das finanças pessoais.

Este projeto é uma aplicação web interativa desenvolvida com **Streamlit**, que permite ao usuário inserir informações sobre seu **salário**, **investimentos** e **despesas mensais**, gerando um **relatório financeiro personalizado** em **PDF** com gráficos de barras e pizza.

A aplicação visa fornecer uma visão clara da situação financeira mensal do usuário.

2 BENEFICIADOS

O sistema proposto poderá beneficiar:

- **Estudantes** que precisam administrar uma renda limitada, muitas vezes proveniente de bolsas ou mesadas;
- **Famílias** que buscam maior clareza sobre seus gastos mensais e desejam organizar melhor o orçamento doméstico;
- **Profissionais autônomos ou assalariados** que desejam acompanhar despesas cotidianas e planejar o uso da renda;
- **Qualquer pessoa** que tenha como objetivo manter um controle financeiro mais eficiente e intuitivo.

3 OBJETIVO

O objetivo do projeto é desenvolver um sistema simples e intuitivo de **controle de gastos pessoais**, utilizando **Python** em conjunto com o **framework Streamlit** para a criação de uma interface interativa e amigável.

O sistema permitirá ao usuário:

- **Registrar despesas** com valor, categoria (ex.: alimentação, transporte, lazer, contas fixas) e descrição;
- **Consultar o total de despesas** em um período de tempo determinado;
- **Visualizar relatórios** organizados por categoria de gasto;
- **Gerar gráficos automáticos**, possibilitando a análise visual e rápida da distribuição dos gastos.

4 JUSTFICATIVA

A automatização desse processo traz diversos benefícios:

- **Agilidade**: os registros podem ser feitos rapidamente, sem necessidade de cálculos manuais;
- **Organização**: todos os dados ficam centralizados em um único sistema, evitando dispersão em anotações e planilhas soltas;
- **Clareza**: gráficos e relatórios automáticos permitem visualizar padrões de gastos, identificar excessos e oportunidades de economia;
- **Acessibilidade**: a utilização do Streamlit garante uma interface de fácil uso, mesmo para pessoas com pouca experiência em tecnologia.

Dessa forma, o sistema contribui para promover o **planejamento financeiro pessoal**, auxiliando usuários a tomarem decisões mais conscientes sobre sua vida financeira e evitando problemas futuros relacionados ao descontrole de despesas.

5 REQUISITOS

Requisitos Funcionais (RF)

O sistema deve:

1. **Cadastrar salário mensal**: na primeira utilização, o usuário informa o valor do seu salário líquido mensal. Esse valor será base para cálculos de limite de gastos e investimentos.

2. **Cadastrar despesa**: permitir que o usuário registre um gasto informando:

    - Valor (numérico)
    - Categoria (ex.: Alimentação, Transporte, Lazer, Contas Fixas, Outros)
    - Descrição (texto livre)
    - Data (definida automaticamente pelo sistema, mas podendo ser ajustada pelo usuário)

3. **Listar despesas**: exibir todas as despesas cadastradas em uma tabela, organizada por data.

4. **Calcular totais**:

    - Total de despesas do mês;
    - Saldo restante do salário (salário - total de despesas - investimento).

5. **Filtrar despesas**: permitir visualizar despesas por:

    - Período (ex.: mês atual, última semana, intervalo definido)
    - Categoria

6. **Gerar relatórios**: apresentar o total de despesas por categoria, em formato de tabela e gráfico.

7. **Visualizar gráficos interativos**: exibir gráficos (pizza e barras) que mostrem a distribuição dos gastos por categoria ou por período.

8. **Simulação de investimento**: permitir que o usuário escolha investir entre **10% e 30% do salário** mensal em uma aplicação fictícia (simulação de poupança).

    - O sistema calcula automaticamente quanto será investido.
    - O usuário poderá visualizar o acúmulo dos investimentos ao longo dos meses.

9. **Exportar dados (opcional)**: permitir que o usuário baixe os dados em formato CSV.

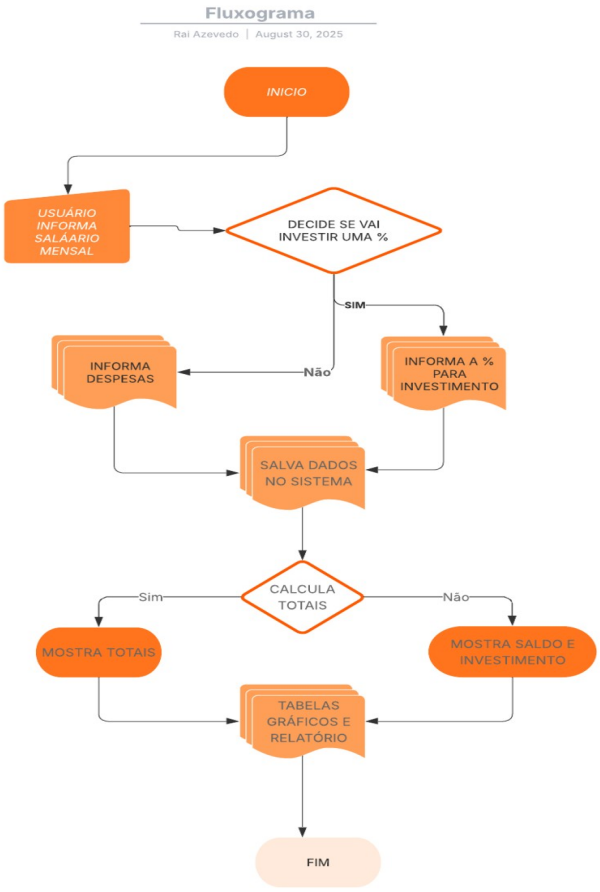
Requisitos Não Funcionais (RNF)

1. **Interface amigável** em Streamlit, com menu lateral para navegação:
    - Cadastro de salário
    - Registro de despesas
    - Relatórios e gráficos
    - Simulação de investimentos
2. **Persistência dos dados**: os dados de salário, despesas e investimentos serão armazenados em um arquivo CSV ou banco SQLite.
3. **Validação de entrada**:
    - Salário deve ser um número positivo;
    - Valor da despesa deve ser numérico e maior que zero;
    - Categoria deve ser escolhida de lista;
    - Percentual de investimento deve estar entre 10% e 30%.
4. **Portabilidade**: rodar em qualquer computador com Python instalado.
5. **Desempenho**: carregar dados e gerar relatórios rapidamente, mesmo com muitas entradas.

6 FLUXO DO USUÁRIO

Ao abrir o sistema, o usuário **informa seu salário mensal**.

- Escolhe um **percentual de investimento** (entre 10% e 30% do salário).
- O sistema calcula automaticamente quanto será destinado ao investimento todo mês.
- O usuário registra suas despesas normalmente.
- O sistema mostra:
- Total gasto
- Total investido
- Saldo restante do mês
- Em **Relatórios**, o usuário pode:
- Ver gráficos de gastos por categoria
- Ver gráfico da evolução do investimento ao longo dos meses



6 O CÓDIGO

1. **Importação de Bibliotecas**

O código utiliza as seguintes bibliotecas:

- **streamlit:** Interface web interativa.
- **pandas:** Manipulação de dados tabulares.
- **matplotlib.pyplot:** Geração de gráficos.
- **reportlab:** Geração de arquivos PDF.
- **datetime:** Obtenção da data atual.
- **io.BytesIO:** Manipulação de arquivos em memória.

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
from io import BytesIO
from datetime import datetime
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Spacer, Image, PageBreak
from reportlab.lib import colors
from reportlab.lib.styles import getSampleStyleSheet
```

2. **Configuração Inicial**

```
st.set_page_config(page_title="Gerenciador Financeiro", layout="centered")
```

Define o título e o layout da aplicação no navegador.

3. **Função `gerar_pdf(...)`**

Essa função é responsável por gerar o **relatório financeiro em PDF** com os seguintes elementos:

- Título e saudação personalizada.
- Resumo financeiro (salário, investimento, despesas e saldo).
- Tabela com as despesas por categoria.
- Gráfico de barras com valores por categoria.
- Gráfico de pizza com distribuição percentual das despesas.

```python
# ---------------------------
# Função para gerar PDF
# ---------------------------
def gerar_pdf(nome, salario, valor_invest, perc_invest, total_despesas, saldo, data_atual, df):
    buffer = BytesIO()
    doc = SimpleDocTemplate(buffer, pagesize=letter)
    elements = []

    styles = getSampleStyleSheet()

    # Título
    title = Paragraph(f"<b><font size=16>📊 Relatório Financeiro</font></b>", styles['Title'])
    subtitle = Paragraph(f"<font size=12>👋 Olá {nome}, sua despesa em {data_atual} é:</font>", styles['Normal'])

    elements.append(title)
    elements.append(Spacer(1, 12))
    elements.append(subtitle)
    elements.append(Spacer(1, 20))

    # Resumo
    resumo = [
        f"💵 Salário: R$ {salario:,.2f}",
        f"📈 Investimento ({perc_invest}%): R$ {valor_invest:,.2f}" if valor_invest > 0 else "",
        f"💸 Total de Despesas: R$ {total_despesas:,.2f}",
        f"✅ Saldo Final: R$ {saldo:,.2f}"
    ]
    for r in resumo:
        if r:  # não imprimir vazio
            elements.append(Paragraph(r, styles['Normal']))
            elements.append(Spacer(1, 6))

    elements.append(Spacer(1, 20))

    # ----- TABELA -----
    data = [df.columns.tolist()] + df.values.tolist()
    table = Table(data)
    table.setStyle(TableStyle([
        ('BACKGROUND', (0,0), (-1,0), colors.HexColor("#4F81BD")),
        ('TEXTCOLOR', (0,0), (-1,0), colors.white),
        ('ALIGN', (0,0), (-1,-1), 'CENTER'),
        ('FONTNAME', (0,0), (-1,0), 'Helvetica-Bold'),
        ('BOTTOMPADDING', (0,0), (-1,0), 10),
        ('BACKGROUND', (0,1), (-1,-1), colors.whitesmoke),
        ('GRID', (0,0), (-1,-1), 0.5, colors.grey),
    ]))
    elements.append(table)
    elements.append(Spacer(1, 20))

    # ----- GRÁFICO DE BARRAS -----
    fig, ax = plt.subplots(figsize=(5,3))
    df.groupby("Categoria")["Valor"].sum().plot(kind="bar", ax=ax, color="#4F81BD")
    ax.set_title("Despesas por Categoria")
    ax.set_ylabel("R$")
    img_buf = BytesIO()
    plt.savefig(img_buf, format="png", bbox_inches="tight")
    img_buf.seek(0)
    elements.append(PageBreak())
    elements.append(Paragraph("<b>📊 Gráfico de Barras</b>", styles['Heading2']))
    elements.append(Spacer(1, 12))
    elements.append(Image(img_buf, width=400, height=250))

    # ----- GRÁFICO DE PIZZA -----
    fig, ax = plt.subplots(figsize=(5,3))
    df.groupby("Categoria")["Valor"].sum().plot(kind="pie", ax=ax, autopct="%1.1f%%", startangle=90, cmap="tab20")
    ax.set_ylabel("")
    ax.set_title("Distribuição de Despesas (%)")
    img_buf2 = BytesIO()
    plt.savefig(img_buf2, format="png", bbox_inches="tight")
    img_buf2.seek(0)
    elements.append(PageBreak())
    elements.append(Paragraph("<b>🟠 Gráfico de Pizza</b>", styles['Heading2']))
    elements.append(Spacer(1, 12))
    elements.append(Image(img_buf2, width=400, height=250))

    doc.build(elements)
    pdf = buffer.getvalue()
    buffer.close()
    return pdf
```

Os gráficos são gerados com `matplotlib` e inseridos no PDF com a biblioteca `reportlab`.

4. **Interface do Usuário**

A aplicação interage com o usuário da seguinte forma:

a. Coleta de dados básicos

- **Nome do usuário (`st.text_input`)**
- **Salário mensal (`st.number_input`)**

b. Investimentos

Se o usuário optar por investir uma porcentagem do salário, ele seleciona essa porcentagem com um `st.slider`, e o valor investido é calculado automaticamente.

c. Despesas

São solicitadas as despesas mensais em cinco categorias fixas:

- Aluguel
- Alimentação
- Transporte
- Lazer
- Outros

Cada valor é armazenado em um dicionário e convertido em um DataFrame (`df_despesas`).

```python
# ----------------------------
# Início da aplicação
# ----------------------------
st.title("💰 Gerenciador de Despesas e Investimentos")

# Nome da pessoa
nome = st.text_input("Digite seu nome:")

# Entrada: salário
salario = st.number_input("Informe seu salário mensal (R$):", min_value=0.0, step=100.0)

if nome and salario > 0:
    investir = st.radio("Deseja investir uma porcentagem do salário?", ["Não", "Sim"])
    perc_invest, valor_invest = 0, 0

    if investir == "Sim":
        perc_invest = st.slider("Informe a % para investimento:", 0, 100, 10)
        valor_invest = salario * (perc_invest / 100)

    # Despesas
    st.subheader("📌 Informe suas despesas")
    despesas = {}
    categorias = ["Aluguel", "Alimentação", "Transporte", "Lazer", "Outros"]
    for cat in categorias:
        despesas[cat] = st.number_input(f"{cat} (R$):", min_value=0.0, step=50.0)

    df_despesas = pd.DataFrame(list(despesas.items()), columns=["Categoria", "Valor"])
    total_despesas = df_despesas["Valor"].sum()
    saldo = salario - total_despesas - valor_invest
    data_atual = datetime.now().strftime("%d/%m/%Y")

    # Resumo
    st.subheader("📊 Resumo Financeiro")
    st.write(f"👋 Olá **{nome}**, sua despesa em **{data_atual}** é:")
    st.write(f"💵 Salário: R$ {salario:,.2f}")
    if investir == "Sim":
        st.write(f"📈 Investimento ({perc_invest}%): R$ {valor_invest:,.2f}")
    st.write(f"💸 Total de Despesas: R$ {total_despesas:,.2f}")
    st.write(f"✅ Saldo Final: R$ {saldo:,.2f}")

    # Gráficos na interface
    st.subheader("📊 Gráficos de Despesas")
    fig_bar, ax_bar = plt.subplots()
    ax_bar.bar(df_despesas["Categoria"], df_despesas["Valor"], color="skyblue")
    ax_bar.set_title("Despesas por Categoria")
    st.pyplot(fig_bar)

    fig_pie, ax_pie = plt.subplots()
    ax_pie.pie(df_despesas["Valor"], labels=df_despesas["Categoria"], autopct="%1.1f%%", startangle=90)
    ax_pie.set_title("Distribuição das Despesas")
    st.pyplot(fig_pie)

    # Botão para baixar PDF
    pdf = gerar_pdf(nome, salario, valor_invest, perc_invest, total_despesas, saldo, data_atual, df_despesas)
    st.download_button("📥 Baixar relatório em PDF", data=pdf, file_name=f"relatorio_{nome}.pdf", mime="application/pdf")
```

5. **Cálculos Financeiros**

Com base nas entradas do usuário, os seguintes valores são calculados:

- **Total de Despesas**
- **Valor investido** (caso aplicável)
- **Saldo final** = Salário - Despesas - Investimento

6. **Exibição dos Resultados na Interface**

O resumo é exibido na tela com:

- Textos formatados
- Gráfico de barras (`bar`)]
- Gráfico de pizza (`pie`)

7. **Geração do PDF**

Após o preenchimento dos dados, o usuário pode clicar em:

```
# Botão para baixar PDF
pdf = gerar_pdf(nome, salario, valor_invest, perc_invest, total_despesas, saldo, data_atual, df_despesas)
st.download_button("📥 Baixar relatório em PDF", data=pdf, file_name=f"relatorio_{nome}.pdf", mime="application/pdf")
```

Isso gera e permite o download de um relatório com todos os dados, incluindo os gráficos.

7 **EXECUÇÃO**

A interface pode ser ativa de 2 formas:

1. Via terminal
   Através do código de ativação local do stramlit

```
streamlit run system.py
```

2. Através do link de Deploy do strealit
   CLIQUE AQUI