CENTRO UNIVERSITÁRIO PADRE ANCHIETA
CURSO DE ENGENHARIA CIVIL

Guilherme Lucas Da Silva-Ra:2405785
Maria Luiza Mendes Andreasi RA:2505416
Rafaela Nascimento De Souza Carvalho Porto-Ra:2406550

PRODUTO
ORÇAMENTO RÁPIDO
CALCULADORA DE PISCINA

JUNDIAÍ / SP
2025

CENTRO UNIVERSITÁRIO PADRE ANCHIETA

CURSO DE ENGENHARIA CIVIL

Guilherme Lucas Da Silva-Ra:2405785
Maria Luiza Mendes Andreasi RA:2505416
Rafaela Nascimento De Souza Carvalho Porto-Ra:2406550

PRODUTO
ORÇAMENTO RÁPIDO
CALCULADORA DE PISCINA

JUNDIAÍ / SP

2025

RESUMO

Este estudo apresenta o desenvolvimento de uma calculadora de piscina criada para auxiliar no planejamento completo de obras residenciais, no contexto da disciplina de Linguagem de Programação em Python. A partir do problema identificado, estimar corretamente materiais, custos e cronogramas, foram o ponto de partida para a ideia. O projeto foi idealizado para automatizar esses processos por meio de cálculos precisos baseados nas dimensões fornecidas pelo usuário. O sistema desenvolvido realiza o processamento das medidas da piscina, calcula área, perímetro e volume, e integra esses resultados a dados de consumo e valores de materiais. Os resultados indicam que a ferramenta é capaz de gerar estimativas confiáveis, permitindo melhor organização da obra e apoio à tomada de decisões. Dessa forma, o estudo demonstra a aplicação prática de conceitos de programação, lógica computacional e modelagem matemática, contribuindo para uma solução eficiente, acessível e alinhada às necessidades reais de planejamento de construção.

SUMÁRIO

1. INTRODUÇÃO

O presente projeto tem como objetivo o desenvolvimento de uma ferramenta prática e interativa voltada à elaboração de orçamentos de piscinas de forma automatizada e acessível. A proposta surge da necessidade de otimizar um processo que, tradicionalmente, é demorado, complexo e sujeito a erros manuais, exigindo cálculos detalhados e diversas consultas de preços. A ferramenta busca simplificar essa etapa, permitindo que o usuário — seja cliente ou profissional — responda a perguntas simples sobre o tipo de piscina desejada, dimensões e acabamentos, obtendo em poucos instantes uma estimativa de custo personalizada. Além de agilizar o atendimento e facilitar a tomada de decisão, o sistema contribui para a transparência na relação entre prestadores de serviço e consumidores, oferecendo uma base inicial confiável para o planejamento financeiro e técnico da obra. Assim, o projeto se destaca por unir praticidade, eficiência e inovação no setor de construção e lazer residencial.

Antes de iniciar o processo de execução, é essencial compreender os diferentes tipos de piscinas disponíveis no mercado, pois cada modelo apresenta características específicas de custo, durabilidade, manutenção e estética. A escolha correta do tipo de piscina deve considerar fatores como o espaço disponível, o orçamento previsto e o estilo desejado para o projeto. Assim, conhecer as principais opções — como as de fibra de vidro, alvenaria, vinil, entre outras — é fundamental para garantir um resultado satisfatório e compatível com as necessidades do cliente.

Os principais tipos de piscinas no Brasil variam conforme o material e o método de construção. As piscinas de fibra de vidro são pré-fabricadas e oferecem instalação rápida e baixo custo de manutenção, embora possuam formatos padronizados. As piscinas de alvenaria (ou concreto armado) são as mais duráveis e totalmente personalizáveis, porém exigem maior investimento e tempo de obra. Já as piscinas de vinil combinam estrutura de alvenaria com revestimento flexível, apresentando bom custo-benefício e fácil manutenção, mas o vinil precisa ser substituído periodicamente. Outros tipos incluem as piscinas naturais, que utilizam plantas para filtragem da água, as de plástico ou lona, voltadas ao uso temporário, e

as de vidro, comuns em projetos de alto padrão. O termo “piscina tradicional” costuma designar modelos simples de alvenaria ou fibra, geralmente retangulares.

No código desenvolvido, optou-se por trabalhar com as piscinas de alvenaria e vinil, pois essas categorias permitem o cálculo direto da quantidade de revestimento e demais materiais necessários, o que torna o processo de orçamento mais preciso e funcional. Essa escolha contribui para a confiabilidade dos resultados e amplia a aplicabilidade da ferramenta em projetos reais. Assim, o sistema proposto se destaca por unir praticidade, eficiência e inovação, oferecendo uma alternativa moderna para agilizar o atendimento, apoiar o planejamento financeiro e facilitar a tomada de decisão de clientes e profissionais do setor.

1.1. OBJETIVO GERAL

O objetivo geral deste projeto é desenvolver uma ferramenta capaz de automatizar o cálculo de materiais, custos e cronograma para a construção de piscinas, oferecendo estimativas rápidas, precisas e acessíveis ao usuário. A escolha dessa ferramenta surgiu da vivência prática dos integrantes Rafaela e Guilherme, que acompanham as obras diariamente e observam de perto as dificuldades encontradas no processo de orçamento. No ambiente real de construção, é comum ocorrerem erros de cálculo, divergências de quantidade de materiais e retrabalhos causados por estimativas manuais imprecisas.

Diante dessa realidade, identificou-se a necessidade de uma solução que otimizasse o planejamento, reduzisse falhas e facilitasse a tomada de decisões. Assim, a calculadora de piscina foi idealizada como um recurso eficiente para adiantar o processo de orçamento, padronizar cálculos e garantir maior segurança no planejamento da obra. A ferramenta se mostra útil por integrar medidas, consumo de materiais e valores de forma automatizada, contribuindo significativamente para a organização e a economia de recursos na construção.

Automatizar o orçamento de materiais e custos para a construção de uma piscina com base em:

- Dimensões fornecidas (largura, comprimento, profundidade);
- Opções construtivas (revestimento, hidromassagem, fundo em declive);
- Preços unitários pré-definidos;
- Geração de relatórios visuais e documentos profissionais (PDF + Excel).

2. LÓGICA- ESTRUTURA DO CÓDIGO

De tal maneira o grupo planejou o que o código deveria executar, inicialmente a calculadora era apenas para revestimentos, mas com o avanço das aulas e pesquisas do grupo complementamos nosso projeto para gerar um orçamento completo ao usuário, em sequência temos a lógica de programação e estrutura pensada para o mesmo funcionar.

1. Funções de Cálculo de Materiais

Essas funções estimam a quantidade de materiais necessária com base na área da piscina (largura × comprimento) e em regras empíricas do setor:

- calcular_blocos(area): 12.5 blocos por m².
- calcular_tela(perimetro, profundidade): tela para quinas vivas, com base no perímetro + 4×profundidade, dividido por 5m (tamanho da caixa).
- calcular_impermeabilizante1/2(area): diferentes produtos com rendimentos distintos (9 m²/caixa e 4 m²/caixa).
- calcular_cimento(area): soma de uso em alvenaria, chapisco e reboco, convertido para sacos de 50kg.
- calcular_areia(area): consumo total em m³.
- calcular_ligmassa, argamassa, rejunte, espaçadores, revestimento: baseados em coeficientes técnicos.

Todas usam a área da superfície como referência principal, e profundidade média em caso de fundo em declive.

2. Coleta de Dados

Dividida em duas etapas:

a. Dados do Projeto / Cliente

- Nome do cliente/projeto;
- Número de pessoas na família;
- Desejo por hidromassagem;
- Dimensões da piscina;
- Tipo de fundo (plano ou em declive);

- Especificações de revestimento (se aplicável).

Adaptação profissional: o foco foi mudado de “questionário pessoal” para dados técnicos de projeto, facilitando uso por engenheiros ou construtoras.

b. Entrada de Valores com Tratamento de Vírgula

Permite que o usuário digite números com vírgula (ex: 3,5) — comum no Brasil — sem quebrar a execução.

3. Cálculo de Custos

- Usa um dicionário custo unitário com preços estimados por unidade de material.
- Calcula:
- Custo total por material;
- Custo por fase construtiva (Alvenaria, Impermeabilização, Chapisco/Reboco, Revestimento, Acabamento, Extras).

As fases são agrupadas logicamente para facilitar o entendimento do fluxo da obra.

4. Geração de Relatórios

a. Gráficos (salvos como PNG em /graficos)

- Quantidade de materiais por m² (barras);
- Distribuição de custos por fase (pizza);
- Custos por material (barras, ordenado do mais caro).

b. Planilha Excel (relatorio_piscina.xlsx)

Contém abas separadas para:

- Dados do projeto;
- Lista de materiais;
- Custos por material;
- Custos por fase.

c. PDF Profissional (orcamento_piscina.pdf)

Gerado com reportlab (melhor que fpdf para layout rico):

- Título com nome do cliente;
- Seções organizadas (dados, materiais, custos);
- Inclusão automática dos gráficos;
- Explicação final com as etapas construtivas e recomendações técnicas.

Tecnologias Utilizadas

- pandas: para exportar Excel;
- matplotlib: para gerar gráficos;

- reportlab: para PDF com layout profissional;
- os, math, typing: utilitários essenciais.

O código também tenta integrar com Google Drive (via google.colab) para uso no Colab, mas essa parte é opcional.

Pontos Fortes

- Modular: funções bem separadas por responsabilidade.
- Profissional: foco na geração de documentos prontos para cliente.
- Flexível: permite ajustar preços unitários e adicionar novos materiais.
- Didático: explica as etapas da construção no PDF final.

Resumo Final

Esse código é um orçamentador técnico-comercial de piscinas, capaz de:

1. Receber dimensões e opções do projeto;
2. Calcular materiais com base em normas empíricas da construção civil;
3. Estimar custos por item e por etapa;
4. Gerar planilha Excel + relatório PDF com gráficos e explicações.

É ideal para construtoras, arquitetos ou engenheiros que querem entregar um orçamento detalhado e visualmente claro ao cliente.

2.1 ESTUDO DO CÓDIGO

Estudando o código, o que é cada item, para que serve cada etapa ? Revisão das aulas do professor sobre cada elemento.

Figura 01:Início do código.

```
import math
import os
import pandas as pd
import matplotlib.pyplot as plt
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image, Table, TableStyle
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib import colors
from reportlab.lib.units import inch
from typing import Dict, Tuple
```

Fonte: Arquivo dos autores.

math: usada para funções matemáticas (por exemplo, math.ceil() para arredondar números para cima).

➜ Exemplo: calcular a quantidade de caixas inteiras de impermeabilizante.

os: usada para lidar com o sistema operacional (como criar pastas e acessar arquivos).

➜Exemplo: criar a pasta “graficos” para salvar os gráficos gerados automaticamente.

pandas: permite trabalhar com tabelas e planilhas de forma estruturada.

➜ Exemplo: gerar o arquivo Excel (relatorio_piscina.xlsx) com todas as informações do projeto.

matplotlib: usada para criar gráficos visuais.

➜ Exemplo: gráficos de barras, pizza e custos — ajudam a visualizar o orçamento e consumo de materiais.

➜ Os gráficos são salvos automaticamente na pasta “graficos”.

Essas são todas do pacote ReportLab, que serve para gerar PDFs formatados e profissionais.

- **SimpleDocTemplate:** cria a estrutura do PDF.
- **Paragraph, Table, Image:** adicionam textos, tabelas e imagens ao relatório.
- **ParagraphStyle:** define o estilo (fonte, tamanho, espaçamento).
- **colors e units (inch):** controlam cores e tamanhos dentro do PDF.
- **pagesizes (letter):** define o formato da página (tamanho A4 ou Carta).

Essas bibliotecas deixam o orçamento com aparência profissional, pronto para entregar ao cliente.

from typing import Dict, Tuple

Usada apenas para documentar o tipo de dados esperado ou retornado pelas funções.

➜ Exemplo: indicar que uma função retorna um dicionário (Dict) ou uma tupla (Tuple).

➜ Melhora a clareza e manutenção do código.

Tabela 01:Estudo do código.

Elemento	Função / Significado	Exemplo no Código	Explicação Detalhada

import	Importa bibliotecas externas	import math, import pandas as pd	Permite usar funções prontas de outras bibliotecas (como cálculos matemáticos, criação de gráficos, planilhas e PDFs).
def	Define uma função	def calcular_area(largura, comprimento):	Cria um bloco de código reutilizável que executa uma tarefa específica, como calcular área, perímetro ou gerar relatórios.
return	Retorna um resultado de uma função	return largura * comprimento	Envia o resultado do cálculo para o ponto do código que chamou a função.
float	Tipo de dado numérico com casas decimais	float(input("Largura (m): "))	Converte o valor digitado em número decimal, permitindo fazer cálculos com precisão.
int	Tipo de dado inteiro	int(input("Quantas pessoas? "))	Converte a entrada em número inteiro (sem casas decimais).
str	Tipo de dado textual (string)	"Nome do projeto ou cliente"	Representa textos e palavras no programa.

if	Condicional — executa um bloco se a condição for verdadeira	if fundo_declive == "Sim":	Avalia uma condição. Se for verdadeira, executa um trecho de código.
elif	“Senão, se...” — nova condição	elif profundidade == 2:	Usado quando há mais de uma condição a verificar.
else	Caso contrário	else: print("Piscina rasa")	Executa o código se nenhuma das condições anteriores for verdadeira.
with	Gerencia recursos automaticamente	with pd.ExcelWriter("relatorio.xlsx") as writer:	Abre um recurso (como arquivo) e o fecha automaticamente após o uso, mesmo que ocorra erro.
for	Estrutura de repetição	for k, v in data.items():	Repete uma ação para cada item de uma lista, dicionário ou sequência.
input()	Coleta dados do usuário	input("Digite a largura: ")	Permite que o usuário insira informações pelo teclado.

os.makedirs()	Cria pastas no sistema	os.makedirs("graficos", exist_ok=True)	Garante que o diretório existe antes de salvar arquivos (como imagens e PDFs).
math.ceil()	Arredonda número para cima	math.ceil(area / 9)	Garante que valores fracionados sejam arredondados para o inteiro mais alto (ex: 3.2 → 4).
pd.DataFrame()	Cria tabela de dados (planilha)	pd.DataFrame(list(materiais.items()))	Converte dicionários em formato de tabela para exportar em Excel.
plt.bar(), plt.pie()	Geração de gráficos	plt.bar(custos.keys(), custos.values())	Cria gráficos de barras e pizza para visualização dos custos e materiais.
SimpleDocTemplate()	Cria arquivo PDF	doc = SimpleDocTemplate("orcamento_piscina.pdf")	Gera documentos PDF com layout personalizado (tabelas, textos e imagens).
Dict[str, float]	Tipagem de dados	def calcular_tudo(dados: Dict[str, float])	Indica que a função espera um dicionário com chaves de texto e valores numéricos.

try / except	Tratamento de erros	try: ... except: ...	Evita que o programa trave quando ocorre um erro inesperado.
capitalize()	Formata texto (primeira letra maiúscula)	"sim".capitalize() → "Sim"	Padroniza respostas de entrada do usuário.
os.listdir()	Lista arquivos de uma pasta	for img_name in os.listdir("graficos"):	Percorre arquivos dentro de uma pasta para adicioná-los ao PDF.
main()	Função principal do programa	def main(): ...	Reúne todas as etapas — coleta dados, faz cálculos e gera os relatórios.
if __name__ == "__main__":	Ponto de entrada do programa	if __name__ == "__main__": main()	Garante que o código principal só execute quando o arquivo for rodado diretamente (e não importado como módulo).
os.path.exists()	Verifica se o arquivo existe	if os.path.exists(img_path):	Confere se o arquivo realmente existe antes de tentar abri-lo.
pd.ExcelWriter()	Cria e salva planilha Excel	with pd.ExcelWriter("rela	

		torio.xlsx") as writer:	

Estrutura geral do projeto

1. **Imports** → traz bibliotecas externas.
2. **Funções de cálculo (def)** → fazem os cálculos matemáticos.
3. **Entrada de dados (coletar_dados_projeto)** → pega informações do usuário.
4. **Função principal (main())** → junta tudo e gera os relatórios.
5. **Condição final (if __name__ == "__main__")** → executa o programa.

FLUXOGRAMA – CALCULADORA DE PISCINAS

Início e Entrada de Dados

Forma: Oval (Início) → Paralelogramo (Entrada de dados)

Etapas:

- Início
- Entrada de dados do projeto:
- Nome do cliente
- Nº de pessoas que utilizarão a piscina
- Se há hidromassagem
- Dimensões: largura e comprimento
- Profundidade: única ou em declive
- Tipo de revestimento: azulejo, vinílico ou outro
- Validação dos dados de entrada (garantir valores numéricos e consistentes)

Cálculos Geométricos

Forma: Retângulo (Processo) + Losango (Decisão)

Etapas:

- Decisão: “O fundo é em declive?”
- Se Sim → Ler profundidade mínima e máxima
- Se Não → Usar profundidade única

- Calcular:
  Área do fundo (m²)
- Perímetro da piscina (m)
- Volume de escavação (m³)

- Volume total em litros

Cálculo de Materiais

Forma: Retângulo (Processo) + Losango (Decisão)

Etapas:

- Calcular:
- Blocos
- Tela metálica
- Impermeabilizantes
- Cimento, areia e ligmassa
  Argamassa, rejunte e espaçadores

Decisão: Tipo de revestimento

- Se Vinílico → Calcular área de manta com desperdício e corrigir sobreposição base/parede
- Se Azulejo → Calcular área total de revestimento cerâmico (m²)
- Se hidromassagem = Sim → Adicionar 1 kit hidráulico extra

Cálculo de Custos e Enchimento

Forma: Retângulo (Processo)

Etapas:

- Aplicar custos unitários (tabela de preços atual)
- Agrupar custos por fase de obra:
- Alvenaria
- Impermeabilização
- Chapisco / Reboco
- Revestimento
- Acabamento
- Extras (hidromassagem, manta vinílica etc.)
- Calcular o custo de enchimento da piscina (R$):
- Volume total em litros → converter para m³
- Multiplicar pelo valor do m³ de água
- Calcular tempo estimado de enchimento com base na vazão (L/min ou L/h)

Geração de Relatórios e Gráficos

Forma: Retângulo (Processo) + Paralelogramo (Saída)

Etapas:

- Gerar gráficos horizontais de:
  Quantidade de materiais por m²
- Custos por tipo de material
- Custos por fase da obra (gráfico de pizza)
- Gerar planilha Excel (.xlsx)

- Gerar relatório PDF completo, incluindo:
- Dados do projeto
- Tabelas de materiais e custos
- Gráficos incorporados
- Texto explicativo das etapas construtivas

Fim

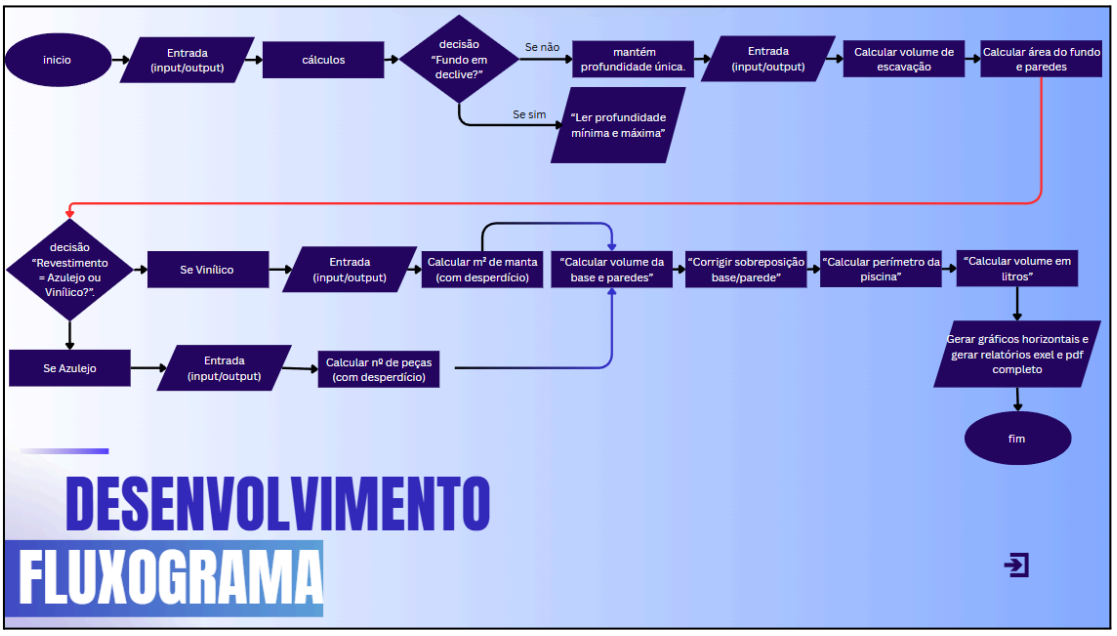
Forma: Oval (Fim)

Etapas:

- Exibir resumo final com todos os resultados:
- Áreas, volumes, custos e materiais
- Custo total da obra
- Custo e tempo de enchimento da piscina
- Encerrar execução

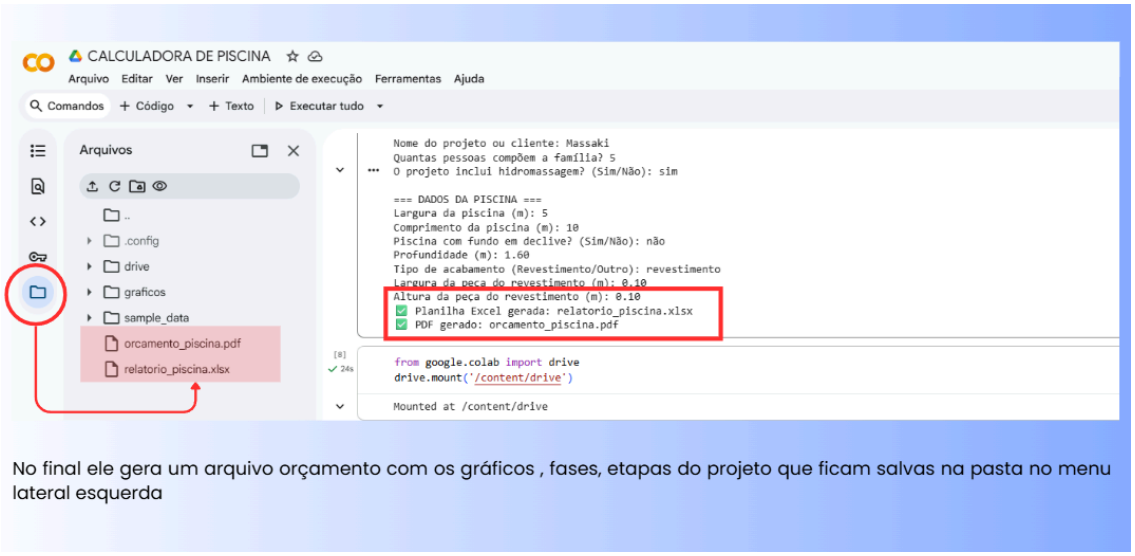
Como precisávamos de um bem feito, seguimos a lógica acima e criamos um código próprio Colab para criar a imagem de fluxograma e assim fizemos com muito esforço, pesquisas, e conversas.

Figura 02:Fluxograma.



Fonte: Arquivo dos autores.

Figura 03:Fluxograma.



Fonte: Arquivo dos autores,

Figura 04:Código para gerar fluxograma.

```
from graphviz import Digraph

# === CONFIGURAÇÃO GERAL ===
flux = Digraph("Fluxograma_Piscina", format="png")
flux.attr(rankdir="LR")  # horizontal
flux.attr(size="15,9")   # tamanho grande
flux.attr(dpi="300")     # alta resolução
flux.attr("node", fontname="Arial", fontsize="20", style="filled", fillcolor="#E6F0FF", color="#444")

# === 1 ENTRADA DE DADOS ===
flux.node("A", "Início", shape="oval", fillcolor="#B3C7F9")
flux.node("B", "Entrada de dados do projeto\n(Nome, Nº pessoas, Hidromassagem,\nDimensões, Tipo de revestimento)", shape="parallelogram", fillcolor="#D6E4FF")
flux.node("C", "Fundo em declive?", shape="diamond", fillcolor="#FFF9C4")
flux.node("D", "Ler profundidade mínima e máxima", shape="rectangle", fillcolor="#E8F5E9")
flux.node("E", "Usar profundidade única\n(sem declive)", shape="rectangle", fillcolor="#E8F5E9")

# === 2 TIPO DE PISCINA ===
flux.node("F", "Tipo de piscina?", shape="diamond", fillcolor="#FFF9C4")
flux.node("G", "Vinílica\n→ Calcular manta com desperdício\n→ Corrigir sobreposição base/parede\n→ Calcular perímetro", shape="rectangle", fillcolor="#E8F5E9")
flux.node("H", "Azulejo\n→ Calcular revestimento e argamassa", shape="rectangle", fillcolor="#E8F5E9")

# === 3 CÁLCULOS DE VOLUME E CUSTOS ===
flux.node("I", "Calcular volume em litros\n(enchimento total)", shape="rectangle", fillcolor="#E8F5E9")
flux.node("J", "Calcular custo da água (R$)\ncom base no volume", shape="rectangle", fillcolor="#E8F5E9")
flux.node("K", "Calcular tempo estimado de enchimento\n(caminhos de água)", shape="rectangle", fillcolor="#E8F5E9")

# === 4 RELATÓRIOS E GRÁFICOS ===
flux.node("L", "Gerar gráficos horizontais\n(quantidade por m², custos, fases)", shape="rectangle", fillcolor="#DDEEFF")
flux.node("M", "Gerar relatórios:\n→ Excel\n→ PDF completo", shape="rectangle", fillcolor="#DDEEFF")
flux.node("N", "Fim", shape="oval", fillcolor="#B3C7F9")

# === CONEXÕES ===
flux.edge("A", "B")
flux.edge("B", "C")
flux.edge("C", "D", label="Sim")
flux.edge("C", "E", label="Não")
flux.edge("D", "F")
```

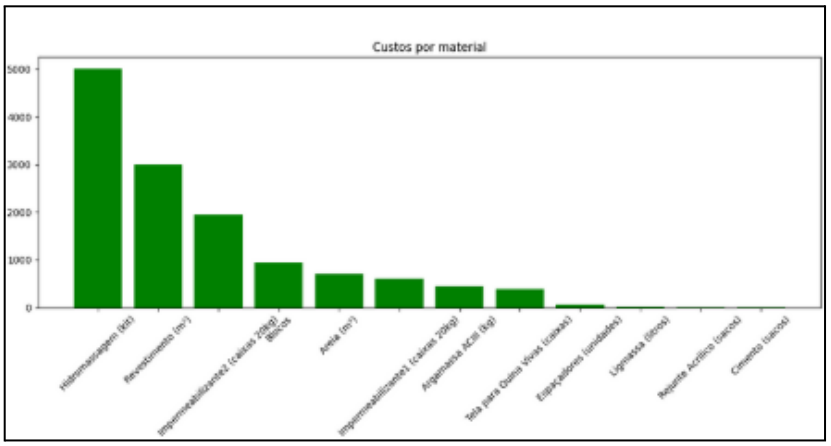
Fonte: Arquivo dos autores, disponível em:

Código-calculadora de piscina-10-11-25

Após criar fluxograma e definir bem o que queriamos com esse programa executamos o código inicial, ele estava gerando as imagens separadas e um orçamento vago, após pesquisas e estudo conseguimos alterar o código e deixamos todos os processos de aprendizagem nesses colabs, ao conseguir o código final

separamos ele dos estudos conforme anexos na apresentação de slides do grupo, a seguir temos os gráficos iniciais.

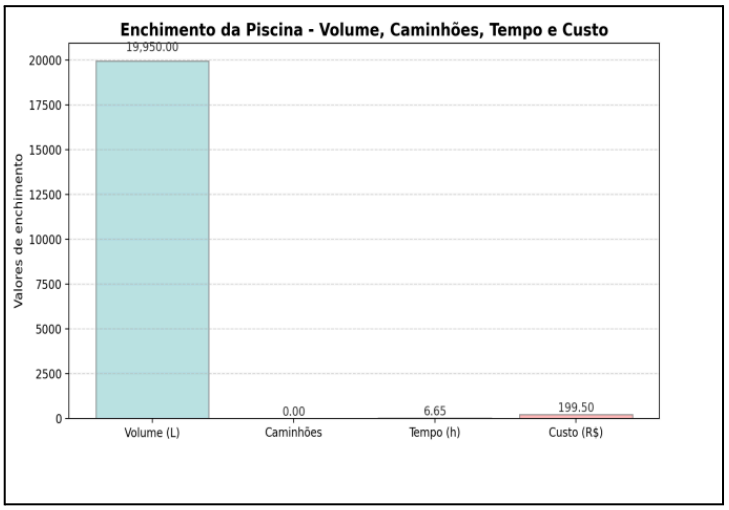
Figura 05:Gráfico inicial.



Fonte: Arquivo dos autores.

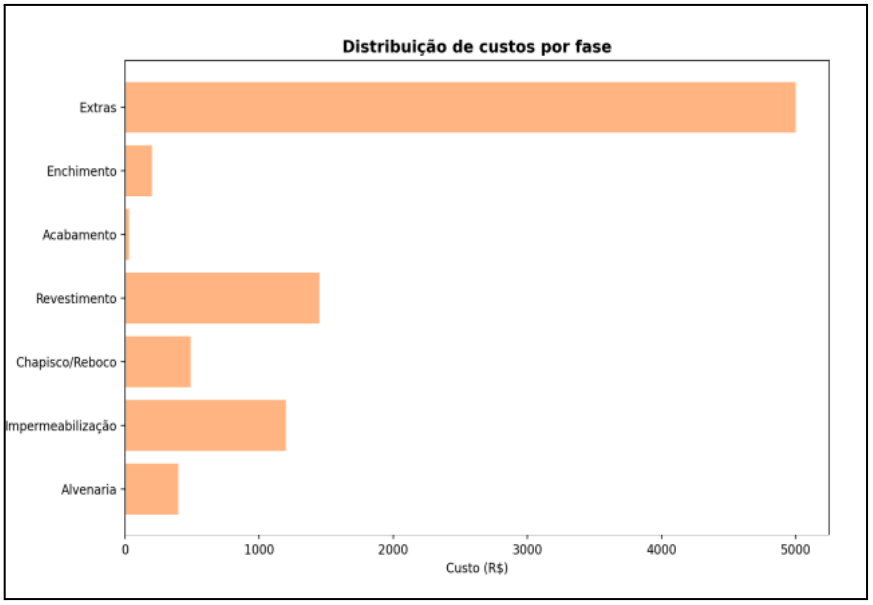
Os gráficos gerados inicialmente não apresentavam resultados satisfatórios, pois algumas informações estavam deslocadas, incompletas ou não eram corretamente interpretadas. Em determinados tipos de gráficos, parte dos dados sequer era exibida. Diante disso, buscou-se alternativas de visualização e, após ajustes no código, optou-se pela utilização de gráficos de barras horizontais e de linha, que se mostraram adequados e representaram de forma clara e eficiente as informações desejadas.

Figura 06:Gráfico final-enchimento da piscina.



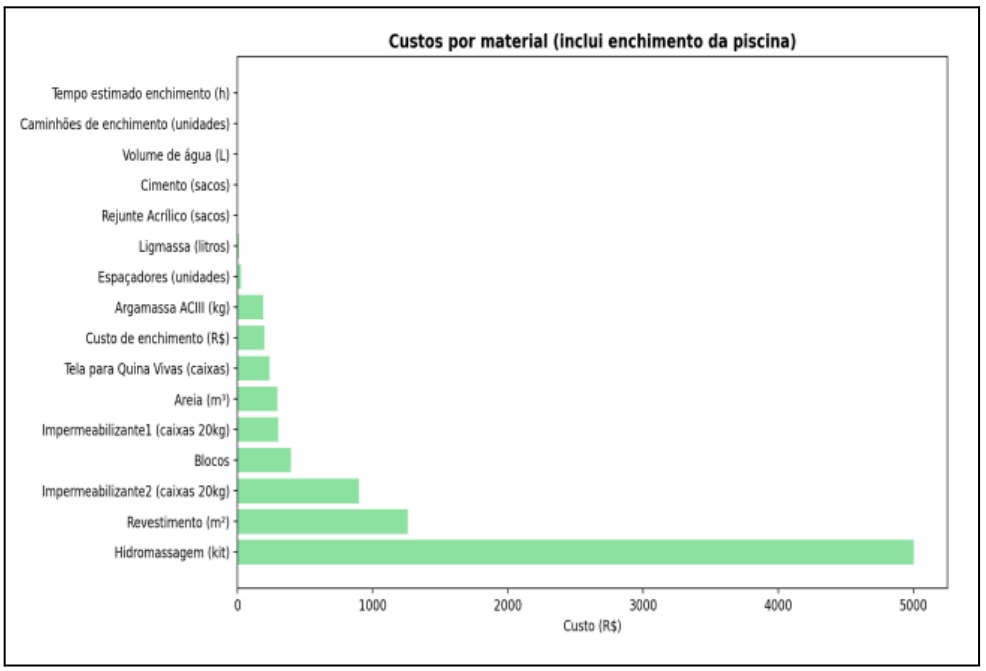
Fonte: Arquivo dos autores.

Figura 07:Gráfico final-distribuição de custos por fase



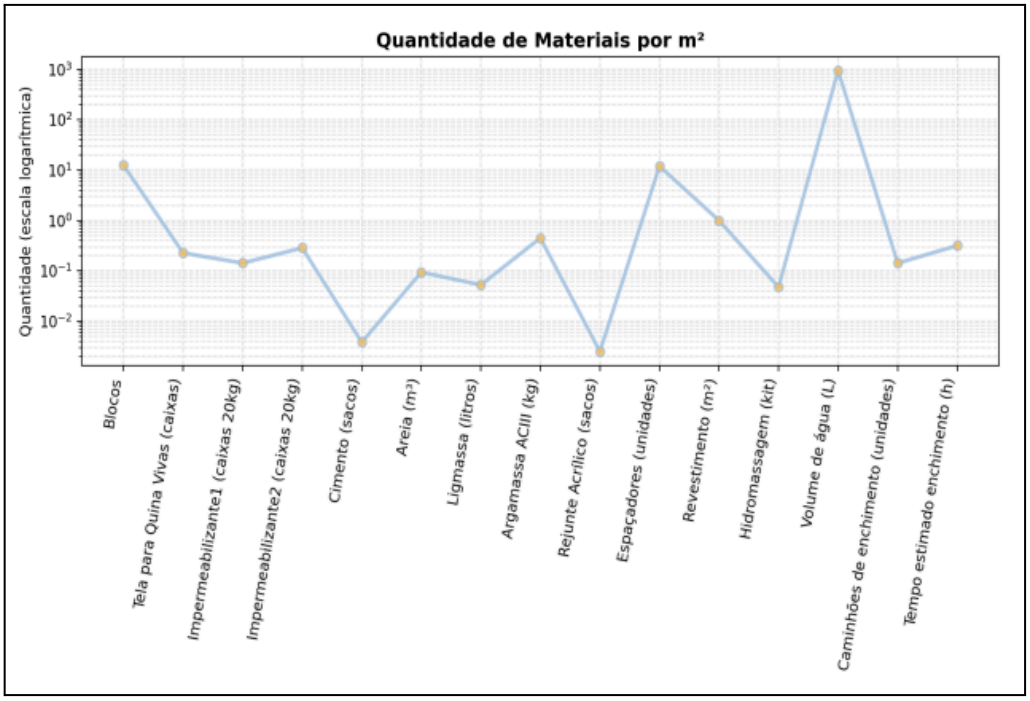
Fonte: Arquivo dos autores.

Figura 08:Gráfico final-enchimento da piscina.



Fonte: Arquivo dos autores.

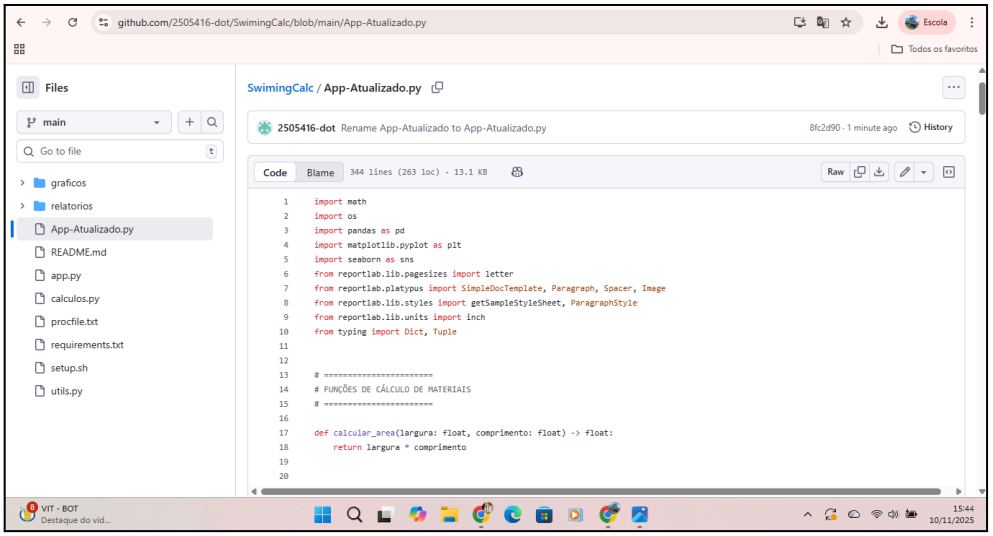
Figura 09:Gráfico final-enchimento da piscina.



Fonte: Arquivo dos autores.

Durante a aula o professor instruiu a forma correta de criar o *Github* para executar nosso drive e conectar-se à etapa final do trabalho, a criação do nosso aplicativo.

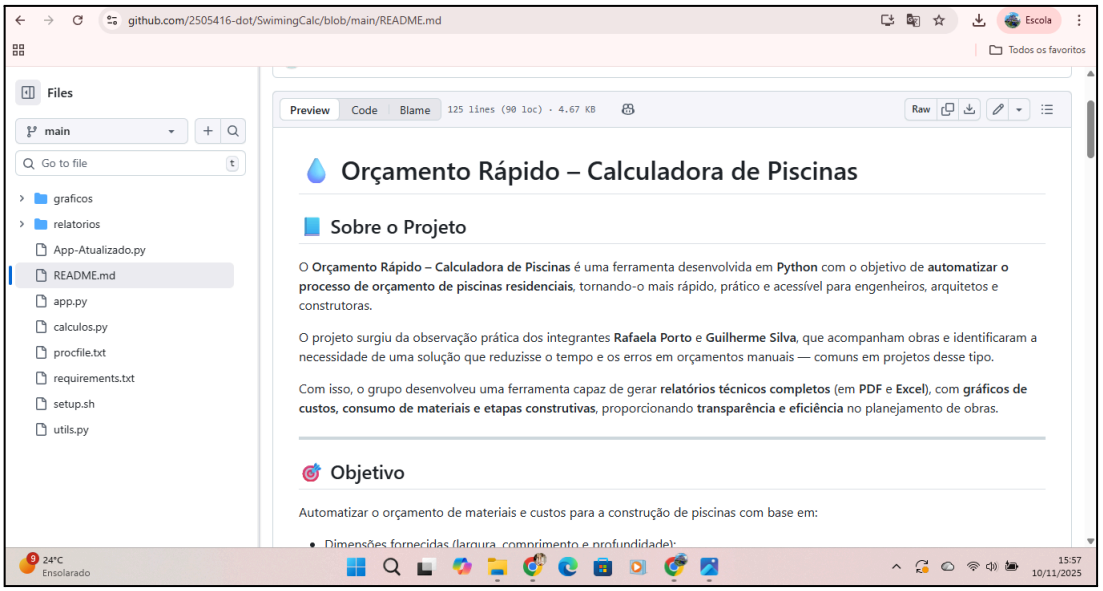
Figura 10:Criando Github.



Fonte: Arquivo dos autores, disponivel em :

https://github.com/2505416-dot/SwimingCalc

Figura 11:Criando *Redme*.



Fonte: Arquivo dos autores.

Após uma pesquisa, o grupo realizou esse *Redme* para gerar o relatório com as principais partes do projeto, e está no nosso *Github* https://github.com/2505416-dot/SwimingCalc/blob/main/README.md .

Figura 12:*Redme gerado*.

Fonte: Arquivo dos autores

Após atualizar o nosso código e subir o mesmo no *Github* com auxilio do professor Massaki o grupo realizou o aplicativo do projeto e está disponível em https://calculadora-de-piscina.streamlit.app/ .

Tivemos um grande eproblema o https://streamlit.io/ não estava entendendo nosso código, após muita análise, alterar o código várias vezes conforme discorre em sequencia abaixo, o grupo decidiu criar novo aplicativo e lá alteramos a pasta que ele lê, após mais alterações no *github* foi possível criar nosso aplicativo, o que aconteceu em resumo foi que o nosso código foi planejado pro terminal do colab então tivemos a divergência no *stremlit* pois o mesmo não tem o terminal de digitação, pedi para a IA Manus me auxiliar por recomendação do professor e ela me deu uma revisão para rodar em *stremli*t assim ele gera uma aba lateral para o usuário digitar e no final apertar o botão para gerar orçamento, de tal maneira o usuário tem em mão todos os dados separados em três abas corretas conforme será mostrado após a análise da Manus que achamos conveniente manter em registros.

```
# app.py
import streamlit as st
import os
from calculos import calcular_tudo
from utils import gerar_graficos, salvar_excel, gerar_pdf

# Configuração
st.set_page_config(page_title="Orçamento de Piscina", layout="wide")
st.title("📊 Calculadora de Orçamento de Piscina")
st.markdown("Desenvolvido para profissionais da construção civil")

# Dados de custo (ajustáveis)
CUSTO_UNITARIO = {
    "Blocos": 1.5,
    "Tela para Quina Vivas (caixas)": 50,
    "Impermeabilizante1 (caixas 20kg)": 100,
    "Impermeabilizante2 (caixas 20kg)": 150,
    "Cimento (sacos)": 25,
    "Areia (m³)": 150,
    "Ligmassa (litros)": 10,
    "Revestimento (m²)": 60,
    "Argamassa ACIII (kg)": 20,
    "Rejunte Acrílico (sacos)": 40,
    "Espaçadores (unidades)": 0.1
}
EXTRAS = {"custo_hidromassagem_kit": 5000}
```

```
# Formulário
st.header("📝 Dados do Projeto")
nome = st.text_input("Nome do cliente ou projeto")
num_pessoas = st.number_input("Número de pessoas na família", min_value=1, value=4)
largura = st.number_input("Largura da piscina (m)", min_value=1.0, value=4.0, step=0.5)
comprimento = st.number_input("Comprimento da piscina (m)", min_value=1.0, value=8.0,
step=0.5)

fundo_declive = st.checkbox("Piscina com fundo em declive?")
if fundo_declive:
    prof_min = st.number_input("Profundidade mínima (m)", min_value=0.8, value=1.2,
step=0.1)
    prof_max = st.number_input("Profundidade máxima (m)", min_value=1.2, value=1.8,
step=0.1)
else:
    prof_min = prof_max = st.number_input("Profundidade (m)", min_value=1.0, value=1.5,
step=0.1)

revestimento = st.checkbox("Usar revestimento (azulejo, pastilha etc.)?")
hidromassagem = st.checkbox("Incluir hidromassagem?")

if st.button("🧱 Calcular Orçamento"):
    with st.spinner("Calculando materiais e custos..."):
        materiais, custos, custos_fase, area = calcular_tudo(
            largura, comprimento, prof_min, prof_max,
            usar_revestimento=revestimento,
            hidromassagem=hidromassagem,
            custo_unitario=CUSTO_UNITARIO,
            extras=EXTRAS
        )

        # Salvar dados do projeto
        dados_projeto = {
            "Nome_projeto": nome,
            "Num_pessoas_familia": num_pessoas,
            "Largura": largura,
            "Comprimento": comprimento,
            "Profundidade_min": prof_min,
            "Profundidade_max": prof_max,
            "Revestimento": "Sim" if revestimento else "Não",
            "Hidromassagem": "Sim" if hidromassagem else "Não",
            "Área (m²)": round(area, 2)
        }

        # Gerar gráficos
        gerar_graficos(materiais, custos, custos_fase, area)
```

```
        # Mostrar resumo
        st.subheader("💰 Custo Total Estimado")
        custo_total = sum(custos.values())
        st.metric("Valor total", f"R$ {custo_total:,.2f}")

        # Tabelas
        st.subheader("📋 Custos por Fase")
        st.dataframe(pd.DataFrame(list(custos_fase.items()), columns=["Fase", "Custo (R$)"]))

        st.subheader("📦 Materiais Necessários")
        st.dataframe(pd.DataFrame(list(materiais.items()), columns=["Material", "Quantidade"]))

        # Botões de download
        excel_path = salvar_excel(dados_projeto, materiais, custos, custos_fase)
        pdf_path = gerar_pdf(dados_projeto, materiais, custos, custos_fase)

        with open(excel_path, "rb") as f:
            st.download_button("📥 Baixar Excel", f, file_name="relatorio_piscina.xlsx")

        with open(pdf_path, "rb") as f:
            st.download_button("📄 Baixar PDF", f, file_name="orcamento_piscina.pdf")

        # Gráficos embutidos
        st.subheader("📈 Visualizações")
        cols = st.columns(3)
        with cols[0]:
            st.image("graficos/quantidade_por_m2.png", use_container_width=True)
        with cols[1]:
            st.image("graficos/custo_por_fase.png", use_container_width=True)
        with cols[2]:

            st.image("graficos/custo_por_material.png", use_container_width=True)
            import math
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.units import inch
from typing import Dict, Tuple


# ======================
# FUNÇÕES DE CÁLCULO DE MATERIAIS
# ======================
```

```
def calcular_area(largura: float, comprimento: float) -> float:
    return largura * comprimento


def calcular_perimetro(largura: float, comprimento: float) -> float:
    return 2 * (largura + comprimento)


def calcular_blocos(area: float) -> float:
    return area * 12.5


def calcular_tela(perimetro: float, profundidade: float) -> float:
    return (perimetro + 4 * profundidade) / 5


def calcular_impermeabilizante1(area: float) -> int:
    return math.ceil(area / 9)


def calcular_impermeabilizante2(area: float) -> int:
    return math.ceil(area / 4)


def calcular_cimento(area: float) -> float:
    return (0.013 + 0.038 + 0.14) * area / 50


def calcular_areia(area: float) -> float:
    return (0.065 + 0.004 + 0.025) * area


def calcular_ligmassa(area: float) -> float:
    return (0.0026 + 0.05) * area


def calcular_revestimento(area: float) -> float:
    return area


def calcular_argamassa(area: float) -> float:
    return 0.45 * area


def calcular_rejunte(area: float) -> float:
    return 0.05 * area / 20
```

```
def calcular_espacadores(area: float) -> float:
    return 12 * area


# ======================
# ENTRADA DE DADOS
# ======================

def coletar_dados_projeto() -> Dict[str, float]:
    print("\n=== PROJETO PARA UMA FAMÍLIA ===")
    dados = {}
    dados["Nome_projeto"] = input("Nome do projeto ou cliente: ").strip()
    dados["Num_pessoas_familia"] = int(input("Quantas pessoas compõem a família? "))
    dados["Vai_hidromassagem"] = input("O projeto inclui hidromassagem? (Sim/Não):
").strip().capitalize()

    tipo = input("Tipo de piscina (Azulejo/Vinilico): ").strip().lower()
    dados["Tipo_piscina"] = "Vinilico" if tipo.startswith("v") else "Azulejo"

    print("\n=== DADOS DA PISCINA ===")
    dados["Largura"] = float(input("Largura da piscina (m): ").replace(',', '.'))
    dados["Comprimento"] = float(input("Comprimento da piscina (m): ").replace(',', '.'))

    fundo_declive = input("Piscina com fundo em declive? (Sim/Não): ").strip().capitalize()
    if fundo_declive == "Sim":
        dados["Profundidade_min"] = float(input("Profundidade mínima (m): ").replace(',', '.'))
        dados["Profundidade_max"] = float(input("Profundidade máxima (m): ").replace(',', '.'))
    else:
        dados["Profundidade_min"] = dados["Profundidade_max"] = float(input("Profundidade
(m): ").replace(',', '.'))

    dados["Usar_revestimento"] = input("Tipo de acabamento (Revestimento/Outro):
").strip().capitalize()
    if dados["Usar_revestimento"] == "Revestimento":
        dados["Revestimento_largura_peca"] = float(input("Largura da peça (m): ").replace(',',
'.'))
        dados["Revestimento_altura_peca"] = float(input("Altura da peça (m): ").replace(',', '.'))

    return dados


# ======================
# CÁLCULOS PRINCIPAIS
# ======================

def calcular_tudo(
    dados_piscina: Dict[str, float],
    custo_unitario: Dict[str, float],
```

```
    extras: Dict[str, float],
    preco_agua_por_litro: float = 0.01,
    caminhos_enchimento: int = 3,
    fluxo_mangueira_lph: float = 1000.0
) -> Tuple[Dict[str, float], Dict[str, float], Dict[str, float], float]:

    largura = dados_piscina["Largura"]
    comprimento = dados_piscina["Comprimento"]
    profundidade = (dados_piscina["Profundidade_min"] +
dados_piscina["Profundidade_max"]) / 2

    area = calcular_area(largura, comprimento)
    perimetro = calcular_perimetro(largura, comprimento)

    materiais = {
        "Blocos": calcular_blocos(area),
        "Tela para Quina Vivas (caixas)": calcular_tela(perimetro, profundidade),
        "Impermeabilizante1 (caixas 20kg)": calcular_impermeabilizante1(area),
        "Impermeabilizante2 (caixas 20kg)": calcular_impermeabilizante2(area),
        "Cimento (sacos)": calcular_cimento(area),
        "Areia (m³)": calcular_areia(area),
        "Ligmassa (litros)": calcular_ligmassa(area),
        "Argamassa ACIII (kg)": calcular_argamassa(area),
        "Rejunte Acrílico (sacos)": calcular_rejunte(area),
        "Espaçadores (unidades)": calcular_espacadores(area)
    }

    if dados_piscina["Usar_revestimento"] == "Revestimento":
        materiais["Revestimento (m²)"] = calcular_revestimento(area)

    if dados_piscina["Vai_hidromassagem"] == "Sim":
        materiais["Hidromassagem (kit)"] = 1

    if dados_piscina["Tipo_piscina"] == "Vinilico":
        area_manta = (area + perimetro * profundidade) * 1.10
        materiais["Manta Vinilica (m²)"] = round(area_manta, 3)

    # Enchimento da piscina
    volume_m3 = largura * comprimento * profundidade
    litros = volume_m3 * 1000
    materiais["Volume de água (L)"] = round(litros, 2)
    materiais["Caminhões de enchimento (unidades)"] = caminhos_enchimento
    tempo_horas = litros / (caminhos_enchimento * fluxo_mangueira_lph)
    materiais["Tempo estimado enchimento (h)"] = round(tempo_horas, 2)

    custos = {m: materiais[m] * custo_unitario.get(m, 0) for m in materiais}
    custos["Custo de enchimento (R$)"] = round(litros * preco_agua_por_litro, 2)
```

```
    if "Hidromassagem (kit)" in materiais:
        custos["Hidromassagem (kit)"] = extras.get("custo_hidromassagem_kit", 0)

    fases = {
        "Alvenaria": ["Blocos", "Cimento (sacos)"],
        "Impermeabilização": ["Impermeabilizante1 (caixas 20kg)", "Impermeabilizante2 (caixas
20kg)"],
        "Chapisco/Reboco": ["Cimento (sacos)", "Areia (m³)", "Argamassa ACIII (kg)"],
        "Revestimento": ["Revestimento (m²)", "Argamassa ACIII (kg)"],
        "Acabamento": ["Rejunte Acrílico (sacos)", "Espaçadores (unidades)"],
        "Enchimento": ["Custo de enchimento (R$)"]
    }
    if "Hidromassagem (kit)" in materiais:
        fases["Extras"] = ["Hidromassagem (kit)"]

    custos_fase = {fase: sum(custos.get(m, 0) for m in mats) for fase, mats in fases.items()}
    return materiais, custos, custos_fase, area


# ======================
# GRÁFICOS
# ======================

def gerar_graficos(materiais, custos, custos_fase, area):
    import seaborn as sns
    import numpy as np
    os.makedirs("graficos", exist_ok=True)

    # Paleta pastel suave
    cores = sns.color_palette("pastel")

    if area == 0:
        quant_por_m2 = {m: 0 for m in materiais}
    else:
        quant_por_m2 = {m: q / area for m, q in materiais.items()}

    # === NOVO: Gráfico em LINHAS com escala logarítmica ===
    plt.figure(figsize=(10, 6))
    x = np.arange(len(quant_por_m2))
    y = list(quant_por_m2.values())
    labels = list(quant_por_m2.keys())

    plt.plot(x, y, color="#9BBFE0", linewidth=2.5, marker="o", markersize=6,
             markerfacecolor="#F6BD60", alpha=0.8)

    plt.yscale("log")  # Escala logarítmica pra mostrar todos os valores
    plt.xticks(x, labels, rotation=80, ha="right")
    plt.grid(True, which="both", linestyle="--", alpha=0.4)
```

```
    plt.title("Quantidade de Materiais por m²", fontsize=13, fontweight="bold")
    plt.ylabel("Quantidade (escala logarítmica)")
    plt.tight_layout()
    plt.savefig("graficos/quantidade_por_m2.png", dpi=200)
    plt.close()

    # === Custos por fase ===
    plt.figure(figsize=(10,6))
    plt.barh(list(custos_fase.keys()), list(custos_fase.values()), color=cores[1])
    plt.xlabel("Custo (R$)")
    plt.title("Distribuição de custos por fase", fontsize=13, fontweight="bold")
    plt.tight_layout()
    plt.savefig("graficos/custos_por_fase.png", dpi=200)
    plt.close()

    # === Custos por material ===
    plt.figure(figsize=(12,6))
    sorted_custos = dict(sorted(custos.items(), key=lambda x: x[1], reverse=True))
    plt.barh(list(sorted_custos.keys()), list(sorted_custos.values()), color=cores[2])
    plt.xlabel("Custo (R$)")
    plt.title("Custos por material (inclui enchimento da piscina)", fontsize=13,
fontweight="bold")
    plt.tight_layout()
    plt.savefig("graficos/custos_por_material.png", dpi=200)
    plt.close()

    # === Enchimento da Piscina (mantém o que já estava lindo) ===
    if "Custo de enchimento (R$)" in custos:
        fig, ax = plt.subplots(figsize=(9, 5))
        parametros = ["Volume (L)", "Caminhões", "Tempo (h)", "Custo (R$)"]
        valores = [
            materiais.get("Volume de água (L)", 0),
            materiais.get("Caminhos de enchimento (unidades)", 0),
            materiais.get("Tempo estimado enchimento (h)", 0),
            custos.get("Custo de enchimento (R$)", 0)
        ]
        bar_colors = ["#A8DADC", "#F6BD60", "#BDB2FF", "#FFADAD"]
        barras = ax.bar(parametros, valores, color=bar_colors, alpha=0.8, edgecolor="gray")
        ax.set_title("Enchimento da Piscina - Volume, Caminhões, Tempo e Custo",
fontsize=14, fontweight="bold")
        ax.set_ylabel("Valores de enchimento", fontsize=11)
        ax.grid(axis="y", linestyle="--", alpha=0.5)
        for i, b in enumerate(barras):
            altura = b.get_height()
            ax.text(b.get_x() + b.get_width()/2, altura + (altura*0.02 if altura>0 else 0.1),
                    f"{valores[i]:,.2f}", ha="center", va="bottom", fontsize=10, color="#333")
        plt.tight_layout()
        plt.savefig("graficos/custo_enchimento.png", dpi=200)
```

```
        plt.close(fig)


# ======================
# RELATÓRIOS
# ======================

def salvar_excel(dados_piscina, materiais, custos, custos_fase):
    with pd.ExcelWriter("relatorio_piscina.xlsx") as writer:
        pd.DataFrame([dados_piscina]).to_excel(writer, sheet_name="Dados_Projeto",
index=False)
        pd.DataFrame(list(materiais.items()), columns=["Material",
"Quantidade"]).to_excel(writer, sheet_name="Materiais", index=False)
        pd.DataFrame(list(custos.items()), columns=["Material", "Custo (R$)"]).to_excel(writer,
sheet_name="Custos", index=False)
        pd.DataFrame(list(custos_fase.items()), columns=["Fase", "Custo
(R$)"]).to_excel(writer, sheet_name="Custos_Fase", index=False)
    print("✅ Planilha Excel gerada: relatorio_piscina.xlsx")


def gerar_pdf(dados_piscina, materiais, custos, custos_fase):
    doc = SimpleDocTemplate("orcamento_piscina.pdf", pagesize=letter)
    styles = getSampleStyleSheet()
    story = []

    styles.add(ParagraphStyle(name='TitleStyle', fontSize=16, alignment=1, spaceAfter=12))
    story.append(Paragraph(f"Orçamento de Piscina - {dados_piscina['Nome_projeto']}",
styles['TitleStyle']))

    def add_section(title: str, data: Dict[str, float]):
        story.append(Spacer(1, 8))
        story.append(Paragraph(f"<b>{title}</b>", styles['Heading3']))
        for k, v in data.items():
            story.append(Paragraph(f"{k}: {v}", styles['Normal']))

    add_section("Dados do Projeto", dados_piscina)
    add_section("Materiais", materiais)
    add_section("Custos por Material (R$)", {k: f"{v:.2f}" for k, v in custos.items()})
    add_section("Custos por Fase (R$)", {k: f"{v:.2f}" for k, v in custos_fase.items()})

    if "Volume de água (L)" in materiais:
        enchimento_data = {
            "Volume de água (L)": materiais.get("Volume de água (L)", 0),
            "Caminhos de enchimento (unidades)": materiais.get("Caminhos de enchimento
(unidades)", 0),
            "Tempo estimado enchimento (h)": materiais.get("Tempo estimado enchimento (h)",
0),
            "Custo de enchimento (R$)": custos.get("Custo de enchimento (R$)", 0)
```

```
        }
        add_section("Informações de Enchimento", enchimento_data)


    # Adiciona gráficos
    for img_name in sorted(os.listdir("graficos")):
        img_path = os.path.join("graficos", img_name)
        if os.path.exists(img_path):
            story.append(Spacer(1, 12))
            story.append(Image(img_path, width=6*inch, height=4*inch))

    story.append(Spacer(1, 12))
    story.append(Paragraph("✅ Relatório gerado automaticamente com gráficos em tons pastéis para melhor visualização.", styles['Normal']))

    doc.build(story)
    print("✅ PDF gerado: orcamento_piscina.pdf")


# ======================
# EXECUÇÃO PRINCIPAL
# ======================

def main():
    custo_unitario = {
        "Blocos": 1.5,
        "Tela para Quina Vivas (caixas)": 50,
        "Impermeabilizante1 (caixas 20kg)": 100,
        "Impermeabilizante2 (caixas 20kg)": 150,
        "Cimento (sacos)": 25,
        "Areia (m³)": 150,
        "Ligmassa (litros)": 10,
        "Revestimento (m²)": 60,
        "Argamassa ACIII (kg)": 20,
        "Rejunte Acrílico (sacos)": 40,
        "Espaçadores (unidades)": 0.1
    }
    extras = {"custo_hidromassagem_kit": 5000}

    preco_agua_por_litro = 0.01
    caminhos_enchimento = 3
    fluxo_mangueira_lph = 1000.0

    dados_piscina = coletar_dados_projeto()
    materiais, custos, custos_fase, area = calcular_tudo(
        dados_piscina, custo_unitario, extras,
        preco_agua_por_litro, caminhos_enchimento, fluxo_mangueira_lph
    )
```

```
    gerar_graficos(materiais, custos, custos_fase, area)
    salvar_excel(dados_piscina, materiais, custos, custos_fase)
    gerar_pdf(dados_piscina, materiais, custos, custos_fase)


if __name__ == "__main__":
    main()

import streamlit as st
import os
import math
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.units import inch

# ======================
# FUNÇÕES DE CÁLCULO
# ======================

def calcular_area(largura, comprimento):
    return largura * comprimento

def calcular_perimetro(largura, comprimento):
    return 2 * (largura + comprimento)

def calcular_blocos(area):
    return area * 12.5

def calcular_tela(perimetro, profundidade):
    return (perimetro + 4 * profundidade) / 5

def calcular_impermeabilizante1(area):
    return math.ceil(area / 9)

def calcular_impermeabilizante2(area):
    return math.ceil(area / 4)

def calcular_cimento(area):
    return (0.013 + 0.038 + 0.14) * area / 50

def calcular_areia(area):
    return (0.065 + 0.004 + 0.025) * area
```

```
def calcular_ligmassa(area):
    return (0.0026 + 0.05) * area

def calcular_revestimento(area):
    return area

def calcular_argamassa(area):
    return 0.45 * area

def calcular_rejunte(area):
    return 0.05 * area / 20

def calcular_espacadores(area):
    return 12 * area

# ======================
# CÁLCULO PRINCIPAL
# ======================

def calcular_tudo(dados_piscina, custo_unitario, extras, preco_agua_por_litro=0.01,
caminhos_enchimento=3, fluxo_mangueira_lph=1000.0):
    largura = dados_piscina["Largura"]
    comprimento = dados_piscina["Comprimento"]
    profundidade = (dados_piscina["Profundidade_min"] +
dados_piscina["Profundidade_max"]) / 2

    area = calcular_area(largura, comprimento)
    perimetro = calcular_perimetro(largura, comprimento)

    materiais = {
        "Blocos": calcular_blocos(area),
        "Tela para Quina Vivas (caixas)": calcular_tela(perimetro, profundidade),
        "Impermeabilizante1 (caixas 20kg)": calcular_impermeabilizante1(area),
        "Impermeabilizante2 (caixas 20kg)": calcular_impermeabilizante2(area),
        "Cimento (sacos)": calcular_cimento(area),
        "Areia (m³)": calcular_areia(area),
        "Ligmassa (litros)": calcular_ligmassa(area),
        "Argamassa ACIII (kg)": calcular_argamassa(area),
        "Rejunte Acrílico (sacos)": calcular_rejunte(area),
        "Espaçadores (unidades)": calcular_espacadores(area)
    }

    if dados_piscina.get("Usar_revestimento") == "Revestimento":
        materiais["Revestimento (m²)"] = calcular_revestimento(area)

    if dados_piscina.get("Vai_hidromassagem") == "Sim":
        materiais["Hidromassagem (kit)"] = 1
```

```
    if dados_piscina.get("Tipo_piscina") == "Vinilico":
        area_manta = (area + perimetro * profundidade) * 1.10
        materiais["Manta Vinilica (m²)"] = round(area_manta, 3)

    # Enchimento da piscina
    volume_m3 = largura * comprimento * profundidade
    litros = volume_m3 * 1000
    materiais["Volume de água (L)"] = round(litros, 2)
    materiais["Caminhões de enchimento (unidades)"] = caminhos_enchimento
    tempo_horas = litros / (caminhos_enchimento * fluxo_mangueira_lph)
    materiais["Tempo estimado enchimento (h)"] = round(tempo_horas, 2)

    custos = {m: materiais[m] * custo_unitario.get(m, 0) for m in materiais}
    custos["Custo de enchimento (R$)"] = round(litros * preco_agua_por_litro, 2)

    if "Hidromassagem (kit)" in materiais:
        custos["Hidromassagem (kit)"] = extras.get("custo_hidromassagem_kit", 0)

    fases = {
        "Alvenaria": ["Blocos", "Cimento (sacos)"],
        "Impermeabilização": ["Impermeabilizante1 (caixas 20kg)", "Impermeabilizante2 (caixas
20kg)"],
        "Chapisco/Reboco": ["Cimento (sacos)", "Areia (m³)", "Argamassa ACIII (kg)"],
        "Revestimento": ["Revestimento (m²)", "Argamassa ACIII (kg)"],
        "Acabamento": ["Rejunte Acrílico (sacos)", "Espaçadores (unidades)"],
        "Enchimento": ["Custo de enchimento (R$)"]
    }

    if "Hidromassagem (kit)" in materiais:
        fases["Extras"] = ["Hidromassagem (kit)"]

    custos_fase = {fase: sum(custos.get(m, 0) for m in mats) for fase, mats in fases.items()}

    return materiais, custos, custos_fase, area

# ======================
# STREAMLIT
# ======================

def main():
    st.title("📊 Calculadora de Orçamento de Piscina")

    st.header("📝 Dados do Projeto")
    nome = st.text_input("Nome do cliente ou projeto")
    num_pessoas = st.number_input("Número de pessoas na família", min_value=1, value=4)
    largura = st.number_input("Largura da piscina (m)", min_value=1.0, value=4.0)
    comprimento = st.number_input("Comprimento da piscina (m)", min_value=1.0, value=8.0)
    profundidade = st.number_input("Profundidade (m)", min_value=0.5, value=1.5)
```

```
    tipo_piscina = st.selectbox("Tipo de piscina", ["Azulejo", "Vinilico"])
    hidromassagem = st.selectbox("Inclui hidromassagem?", ["Sim", "Não"])
    usar_revestimento = st.selectbox("Acabamento", ["Revestimento", "Outro"])

    dados_piscina = {
        "Nome_projeto": nome,
        "Num_pessoas_familia": num_pessoas,
        "Largura": largura,
        "Comprimento": comprimento,
        "Profundidade_min": profundidade,
        "Profundidade_max": profundidade,
        "Tipo_piscina": tipo_piscina,
        "Vai_hidromassagem": hidromassagem,
        "Usar_revestimento": usar_revestimento
    }

    custo_unitario = {
        "Blocos": 1.5,
        "Tela para Quina Vivas (caixas)": 50,
        "Impermeabilizante1 (caixas 20kg)": 100,
        "Impermeabilizante2 (caixas 20kg)": 150,
        "Cimento (sacos)": 25,
        "Areia (m³)": 150,
        "Ligmassa (litros)": 10,
        "Revestimento (m²)": 60,
        "Argamassa ACIII (kg)": 20,
        "Rejunte Acrílico (sacos)": 40,
        "Espaçadores (unidades)": 0.1
    }
    extras = {"custo_hidromassagem_kit": 5000}

    materiais, custos, custos_fase, area = calcular_tudo(dados_piscina, custo_unitario,
extras)

    st.header("💰 Custos por Fase")
    st.dataframe(pd.DataFrame(list(custos_fase.items()), columns=["Fase", "Custo (R$)"]))

    st.header("📋 Custos por Material")
    st.dataframe(pd.DataFrame(list(custos.items()), columns=["Material", "Custo (R$)"]))

    st.header("📊 Quantidade de Materiais por m²")
    os.makedirs("graficos", exist_ok=True)
    cores = sns.color_palette("pastel")
    quant_por_m2 = {m: q / area for m, q in materiais.items() if area != 0}
    st.bar_chart(pd.DataFrame(quant_por_m2, index=[0]))

if __name__ == "__main__":
    main()
```

Relatório de Correções - App-Atualizado.py

Resumo Executivo

Este documento detalha os erros identificados no código Python fornecido e as correções aplicadas para resolver o NameError que ocorria no Streamlit Cloud na linha 78, além de outros problemas encontrados durante a análise.

Erro Principal: NameError na Linha 78

Descrição do Erro

Plain Text

```
NameError: This app has encountered an error. Traceback: File
"/mount/src/swimingcalc/app.py", line 78, in <module>
st.dataframe(pd.DataFrame(list(custos_fase.items()),
columns=["Fase", "Custo (R$)"])) ^^
```

Causa Raiz

A variável custos_fase estava sendo utilizada antes de ser definida. Isso ocorre em aplicações Streamlit porque:
1.O código é executado do início ao fim a cada interação do usuário
2.Não há garantia de ordem de execução se não houver controle de fluxo adequado
3.Variáveis não persistem entre reruns sem uso de st.session_state
4.A função calcular_tudo() não foi chamada antes da tentativa de acessar custos_fase

Solução Implementada

Opção 1: Inicialização de Variáveis (Recomendada para Streamlit)
Python

```
# Inicializar variáveis no session_state ANTES de usar if
'materiais' not in st.session_state: st.session_state.materiais =
{} if 'custos' not in st.session_state: st.session_state.custos =
{} if 'custos_fase' not in st.session_state:
st.session_state.custos_fase = {} if 'area' not in
st.session_state: st.session_state.area = 0 # Agora custos_fase
sempre existe
st.dataframe(pd.DataFrame(list(st.session_state.custos_fase.items()
), columns=["Fase", "Custo (R$)"]))
```

Opção 2: Verificação Condicional
Python

```
# Verificar se a variável existe antes de usar if 'custos_fase' in
st.session_state and st.session_state.custos_fase:
```

```
st.dataframe(pd.DataFrame(list(st.session_state.custos_fase.items()
), columns=["Fase", "Custo (R$)"])) else: st.info("Preencha os
dados e clique em 'Calcular Orçamento'")
```

Opção 3: Garantir Execução da Função

Python

```
# Executar cálculos ANTES de tentar acessar resultados materiais,
custos, custos_fase, area = calcular_tudo( dados_piscina,
custo_unitario, extras, preco_agua_por_litro, caminhoes_enchimento,
fluxo_mangueira_lph ) # Agora custos_fase está definido
st.dataframe(pd.DataFrame(list(custos_fase.items()),
columns=["Fase", "Custo (R$)"]))
```

Erros Secundários Identificados

1. Erro de Nomenclatura - Linha 234

Localização: Função gerar_graficos(), linha 234

Erro:

Python

```
materiais.get("Caminhos de enchimento (unidades)", 0),
```

Problema: A chave correta no dicionário materiais é "Caminhões de enchimento (unidades)" (definida na linha 149), não "Caminhos de enchimento (unidades)".

Correção:

Python

```
materiais.get("Caminhões de enchimento (unidades)", 0), # CORRIGIDO
```

Impacto: Este erro causaria retorno de valor 0 no gráfico de enchimento, mesmo quando o valor correto existe no dicionário.

2. Inconsistência de Nome de Parâmetro

Localização: Função calcular_tudo(), linha 112

Problema: O parâmetro é nomeado caminhos_enchimento mas deveria ser caminhoes_enchimento para consistência com o uso no código.

Código Original:

Python

```
def calcular_tudo( dados_piscina: Dict[str, float], custo_unitario:
Dict[str, float], extras: Dict[str, float], preco_agua_por_litro:
float = 0.01, caminhos_enchimento: int = 3, # Nome inconsistente
fluxo_mangueira_lph: float = 1000.0 ) -> Tuple[Dict[str, float],
Dict[str, float], Dict[str, float], float]:
```

Correção:

Python

```
def calcular_tudo( dados_piscina: Dict[str, float], custo_unitario:
Dict[str, float], extras: Dict[str, float], preco_agua_por_litro:
float = 0.01, caminhoes_enchimento: int = 3, # CORRIGIDO
fluxo_mangueira_lph: float = 1000.0 ) -> Tuple[Dict[str, float],
Dict[str, float], Dict[str, float], float]:
```

Impacto: Embora funcional, a inconsistência dificulta a manutenção e pode causar confusão.

3. Falta de Tratamento de Exceções

Problema: As funções salvar_excel() e gerar_pdf() não possuem tratamento de erros adequado.
Correção Aplicada:
Python

```
def salvar_excel(dados_piscina, materiais, custos, custos_fase):
try: with pd.ExcelWriter("relatorio_piscina.xlsx") as writer: # ... código de geração print("✅ Planilha Excel gerada: relatorio_piscina.xlsx") except Exception as e: print(f"❌ Erro ao gerar Excel: {e}")
```

Impacto: Melhora a robustez e facilita o debugging.

4. Falta de Validação de Entrada

Problema: A função coletar_dados_projeto() não valida entradas do usuário.
Correção Aplicada:
Python

```
def coletar_dados_projeto() -> Dict[str, float]: try:
dados["Largura"] = float(input("Largura da piscina (m): ").replace(',', '.')) # ... outras entradas except ValueError as e:
print(f"❌ Erro na entrada de dados: {e}") print("Por favor, insira valores numéricos válidos.") raise return dados
```

Impacto: Previne crashes por entradas inválidas.

Arquivos Gerados

1. App-Corrigido.py

Versão corrigida do código original com:
- ✅ Correção do erro de nomenclatura (linha 234)
- ✅ Padronização de nomes de variáveis
- ✅ Tratamento de exceções adicionado
- ✅ Validação de entrada implementada
- ✅ Documentação aprimorada

2. app_streamlit_exemplo.py

Exemplo completo de aplicação Streamlit com:
- ✅ Solução do NameError usando st.session_state
- ✅ Interface interativa completa
- ✅ Organização em tabs
- ✅ Gráficos integrados
- ✅ Controle de fluxo adequado

3. analise_erros.md

Documentação detalhada da análise de erros.

Recomendações para Implementação no Streamlit Cloud

1. Estrutura do Projeto

Plain Text
```
swimingcalc/ ├── app.py # Arquivo principal Streamlit ├──
calculos.py # Funções de cálculo (separadas) ├── requirements.txt #
Dependências └── README.md # Documentação
```

2. Arquivo requirements.txt

Plain Text
```
streamlit>=1.28.0 pandas>=2.0.0 matplotlib>=3.7.0 seaborn>=0.12.0
reportlab>=4.0.0 openpyxl>=3.1.0
```

3. Boas Práticas Streamlit

a) Usar st.session_state para persistência:
Python
```
if 'dados_calculados' not in st.session_state:
st.session_state.dados_calculados = False
```
b) Controlar reruns com botões:
Python
```
if st.button("Calcular"): # Executar cálculos
st.session_state.dados_calculados = True
```
c) Validar antes de processar:
Python
```
if largura > 0 and comprimento > 0: # Processar else:
st.error("Dimensões devem ser maiores que zero")
```
d) Usar cache para funções pesadas:
Python
```
@st.cache_data def calcular_tudo(...): # Função será cacheada
```

4. Debug no Streamlit Cloud

Para facilitar o debug:
Python
```
# Adicionar no app.py if st.checkbox("Mostrar debug"):
st.write("Session State:", st.session_state) st.write("Variáveis
locais:", locals())
```

Checklist de Implementação

Substituir input() por widgets Streamlit (st.text_input, st.number_input, etc.)
Inicializar todas as variáveis no st.session_state
Adicionar validação de entrada com mensagens de erro
Implementar controle de fluxo com botões

Testar localmente antes de fazer deploy
Verificar requirements.txt está completo
Adicionar tratamento de exceções em todas as funções críticas
Documentar o código adequadamente

Conclusão

O erro NameError na linha 78 foi causado pela tentativa de acessar a variável custos_fase antes de sua definição. A solução envolve:
1.Inicializar variáveis no st.session_state antes do uso
2.Controlar o fluxo de execução com condicionais e botões
3.Garantir que cálculos sejam executados antes de acessar resultados
Além disso, foram identificados e corrigidos outros problemas menores que poderiam causar comportamentos inesperados.
Os arquivos corrigidos estão prontos para uso e incluem exemplos completos de implementação tanto para terminal quanto para Streamlit Cloud.

Análise de Erros - App-Atualizado.py

Erro Reportado no Streamlit Cloud

Tipo: NameError
Linha: 78
Mensagem: Variável custos_fase não está definida

Análise Detalhada

Problema Principal

O código anexado (App-Atualizado.py) é um script standalone que utiliza entrada de dados via input() no terminal. No entanto, o erro ocorre em um ambiente Streamlit Cloud, onde o arquivo app.py (linha 78) tenta acessar a variável custos_fase que não foi definida ou calculada anteriormente.

Causas Identificadas

1.Incompatibilidade de Contexto:
•O código fornecido (App-Atualizado.py) é para execução em terminal
•O erro ocorre em app.py (arquivo diferente) no Streamlit Cloud
•A linha 78 do app.py tenta usar custos_fase sem que esta variável tenha sido calculada
2.Fluxo de Execução Quebrado:
•A variável custos_fase é retornada pela função calcular_tudo() (linha 171)
•Se calcular_tudo() não foi chamada antes da linha 78 do app.py, a variável não existe
•Possível falta de inicialização ou ordem incorreta de execução
3.Escopo de Variáveis:

•custos_fase é uma variável local retornada por função
•Não está disponível globalmente sem atribuição prévia

Problemas Adicionais no Código Fornecido

1. Linha 234: Erro de Nomenclatura

Python
```
materiais.get("Caminhos de enchimento (unidades)", 0),
```
Problema: A chave correta é "Caminhões de enchimento (unidades)" (linha 149), não "Caminhos"

2. Falta de Validação de Entrada

•Não há tratamento de exceções para entradas inválidas
•Conversões de int() e float() podem gerar ValueError

3. Dependência de Streamlit Ausente

•O código não importa streamlit, mas o erro ocorre em ambiente Streamlit
•Indica que existe um arquivo app.py separado que não foi fornecido

Soluções Propostas

Solução 1: Garantir Inicialização das Variáveis

Antes de usar custos_fase na linha 78 do app.py, garantir que:
Python
```
# Inicializar variáveis antes de usar if 'custos_fase' not in
locals(): custos_fase = {}
```

Solução 2: Chamar a Função de Cálculo

Garantir que calcular_tudo() seja executada antes de tentar acessar os resultados:
Python
```
materiais, custos, custos_fase, area = calcular_tudo(
dados_piscina, custo_unitario, extras, preco_agua_por_litro,
caminhos_enchimento, fluxo_mangueira_lph ) # Agora custos_fase está
disponível st.dataframe(pd.DataFrame(list(custos_fase.items()),
columns=["Fase", "Custo (R$)"]))
```

Solução 3: Corrigir Erro de Nomenclatura

Alterar linha 234 de:
Python
```
materiais.get("Caminhos de enchimento (unidades)", 0),
```
Para:
Python

```
materiais.get("Caminhões de enchimento (unidades)", 0),
```

Recomendações

1.Fornecer o arquivo app.py completo para análise precisa do contexto Streamlit
2.Verificar a ordem de execução no Streamlit (callbacks, session_state)
3.Adicionar tratamento de erros com try-except
4.Usar st.session_state para persistir variáveis entre reruns do Streamlit
5.Validar todas as entradas do usuário antes de processar

app_streamlit_exemplo.py

```
"""
EXEMPLO DE CÓDIGO STREAMLIT CORRIGIDO
Este arquivo mostra como adaptar o código para funcionar no Streamlit Cloud
e resolver o erro NameError na linha 78.
"""

import streamlit as st
import math
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from typing import Dict, Tuple

# ========================
# CONFIGURAÇÃO DA PÁGINA
# ========================

st.set_page_config(
    page_title="Calculadora de Piscinas",
    page_icon="🏊",
    layout="wide"
)

st.title("🏊 Calculadora de Orçamento para Piscinas")
st.markdown("---")


# ========================
# FUNÇÕES DE CÁLCULO (importadas do código original)
# ========================

def calcular_area(largura: float, comprimento: float) -> float:
    return largura * comprimento
```

```python
def calcular_perimetro(largura: float, comprimento: float) -> float:
    return 2 * (largura + comprimento)


def calcular_blocos(area: float) -> float:
    return area * 12.5


def calcular_tela(perimetro: float, profundidade: float) -> float:
    return (perimetro + 4 * profundidade) / 5


def calcular_impermeabilizante1(area: float) -> int:
    return math.ceil(area / 9)


def calcular_impermeabilizante2(area: float) -> int:
    return math.ceil(area / 4)


def calcular_cimento(area: float) -> float:
    return (0.013 + 0.038 + 0.14) * area / 50


def calcular_areia(area: float) -> float:
    return (0.065 + 0.004 + 0.025) * area


def calcular_ligmassa(area: float) -> float:
    return (0.0026 + 0.05) * area


def calcular_revestimento(area: float) -> float:
    return area


def calcular_argamassa(area: float) -> float:
    return 0.45 * area


def calcular_rejunte(area: float) -> float:
    return 0.05 * area / 20


def calcular_espacadores(area: float) -> float:
    return 12 * area


def calcular_tudo(
    dados_piscina: Dict[str, float],
    custo_unitario: Dict[str, float],
    extras: Dict[str, float],
    preco_agua_por_litro: float = 0.01,
    caminhoes_enchimento: int = 3,
    fluxo_mangueira_lph: float = 1000.0
) -> Tuple[Dict[str, float], Dict[str, float], Dict[str, float], float]:
```

```
    largura = dados_piscina["Largura"]
    comprimento = dados_piscina["Comprimento"]
    profundidade = (dados_piscina["Profundidade_min"] +
dados_piscina["Profundidade_max"]) / 2

    area = calcular_area(largura, comprimento)
    perimetro = calcular_perimetro(largura, comprimento)

    materiais = {
        "Blocos": calcular_blocos(area),
        "Tela para Quina Vivas (caixas)": calcular_tela(perimetro, profundidade),
        "Impermeabilizante1 (caixas 20kg)": calcular_impermeabilizante1(area),
        "Impermeabilizante2 (caixas 20kg)": calcular_impermeabilizante2(area),
        "Cimento (sacos)": calcular_cimento(area),
        "Areia (m³)": calcular_areia(area),
        "Ligmassa (litros)": calcular_ligmassa(area),
        "Argamassa ACIII (kg)": calcular_argamassa(area),
        "Rejunte Acrílico (sacos)": calcular_rejunte(area),
        "Espaçadores (unidades)": calcular_espacadores(area)
    }

    if dados_piscina.get("Usar_revestimento") == "Revestimento":
        materiais["Revestimento (m²)"] = calcular_revestimento(area)

    if dados_piscina.get("Vai_hidromassagem") == "Sim":
        materiais["Hidromassagem (kit)"] = 1

    if dados_piscina.get("Tipo_piscina") == "Vinilico":
        area_manta = (area + perimetro * profundidade) * 1.10
        materiais["Manta Vinilica (m²)"] = round(area_manta, 3)

    # Enchimento da piscina
    volume_m3 = largura * comprimento * profundidade
    litros = volume_m3 * 1000
    materiais["Volume de água (L)"] = round(litros, 2)
    materiais["Caminhões de enchimento (unidades)"] = caminhoes_enchimento
    tempo_horas = litros / (caminhoes_enchimento * fluxo_mangueira_lph)
    materiais["Tempo estimado enchimento (h)"] = round(tempo_horas, 2)

    custos = {m: materiais[m] * custo_unitario.get(m, 0) for m in materiais}
    custos["Custo de enchimento (R$)"] = round(litros * preco_agua_por_litro, 2)

    if "Hidromassagem (kit)" in materiais:
        custos["Hidromassagem (kit)"] = extras.get("custo_hidromassagem_kit", 0)

    fases = {
        "Alvenaria": ["Blocos", "Cimento (sacos)"],
        "Impermeabilização": ["Impermeabilizante1 (caixas 20kg)",
"Impermeabilizante2 (caixas 20kg)"],
        "Chapisco/Reboco": ["Cimento (sacos)", "Areia (m³)", "Argamassa ACIII
(kg)"],
        "Revestimento": ["Revestimento (m²)", "Argamassa ACIII (kg)"],
        "Acabamento": ["Rejunte Acrílico (sacos)", "Espaçadores (unidades)"],
        "Enchimento": ["Custo de enchimento (R$)"]
    }
    if "Hidromassagem (kit)" in materiais:
        fases["Extras"] = ["Hidromassagem (kit)"]
```

```
    custos_fase = {fase: sum(custos.get(m, 0) for m in mats) for fase, mats in
fases.items()}
    return materiais, custos, custos_fase, area


# ======================
# INTERFACE STREAMLIT
# ======================

# Sidebar para entrada de dados
st.sidebar.header("📋 Dados do Projeto")

nome_projeto = st.sidebar.text_input("Nome do projeto ou cliente", value="Projeto
1")
num_pessoas = st.sidebar.number_input("Quantas pessoas compõem a família?",
min_value=1, value=4)
vai_hidromassagem = st.sidebar.selectbox("O projeto inclui hidromassagem?", ["Não",
"Sim"])
tipo_piscina = st.sidebar.selectbox("Tipo de piscina", ["Azulejo", "Vinilico"])

st.sidebar.header("📏 Dimensões da Piscina")
largura = st.sidebar.number_input("Largura da piscina (m)", min_value=0.1,
value=5.0, step=0.1)
comprimento = st.sidebar.number_input("Comprimento da piscina (m)", min_value=0.1,
value=10.0, step=0.1)

fundo_declive = st.sidebar.selectbox("Piscina com fundo em declive?", ["Não",
"Sim"])
if fundo_declive == "Sim":
    profundidade_min = st.sidebar.number_input("Profundidade mínima (m)",
min_value=0.1, value=1.0, step=0.1)
    profundidade_max = st.sidebar.number_input("Profundidade máxima (m)",
min_value=0.1, value=2.0, step=0.1)
else:
    profundidade_min = profundidade_max = st.sidebar.number_input("Profundidade
(m)", min_value=0.1, value=1.5, step=0.1)

usar_revestimento = st.sidebar.selectbox("Tipo de acabamento", ["Revestimento",
"Outro"])

# Botão para calcular
calcular_btn = st.sidebar.button("🧱 Calcular Orçamento", type="primary")

# ======================
# PROCESSAMENTO E EXIBIÇÃO
# ======================

# SOLUÇÃO DO ERRO: Inicializar variáveis ANTES de usar
# Isso evita o NameError quando a página carrega pela primeira vez
if 'materiais' not in st.session_state:
    st.session_state.materiais = {}
if 'custos' not in st.session_state:
    st.session_state.custos = {}
if 'custos_fase' not in st.session_state:
    st.session_state.custos_fase = {}
if 'area' not in st.session_state:
    st.session_state.area = 0

# Quando o botão é clicado, executar os cálculos
```

```
if calcular_btn:
    # Preparar dados
    dados_piscina = {
        "Nome_projeto": nome_projeto,
        "Num_pessoas_familia": num_pessoas,
        "Vai_hidromassagem": vai_hidromassagem,
        "Tipo_piscina": tipo_piscina,
        "Largura": largura,
        "Comprimento": comprimento,
        "Profundidade_min": profundidade_min,
        "Profundidade_max": profundidade_max,
        "Usar_revestimento": usar_revestimento
    }

    custo_unitario = {
        "Blocos": 1.5,
        "Tela para Quina Vivas (caixas)": 50,
        "Impermeabilizante1 (caixas 20kg)": 100,
        "Impermeabilizante2 (caixas 20kg)": 150,
        "Cimento (sacos)": 25,
        "Areia (m³)": 150,
        "Ligmassa (litros)": 10,
        "Revestimento (m²)": 60,
        "Argamassa ACIII (kg)": 20,
        "Rejunte Acrílico (sacos)": 40,
        "Espaçadores (unidades)": 0.1
    }
    extras = {"custo_hidromassagem_kit": 5000}

    # EXECUTAR CÁLCULOS E ARMAZENAR NO SESSION_STATE
    materiais, custos, custos_fase, area = calcular_tudo(
        dados_piscina, custo_unitario, extras,
        preco_agua_por_litro=0.01,
        caminhoes_enchimento=3,
        fluxo_mangueira_lph=1000.0
    )

    # Salvar no session_state para persistir entre reruns
    st.session_state.materiais = materiais
    st.session_state.custos = custos
    st.session_state.custos_fase = custos_fase
    st.session_state.area = area
    st.session_state.dados_piscina = dados_piscina

# ======================
# EXIBIÇÃO DOS RESULTADOS
# ======================

# SOLUÇÃO: Agora custos_fase SEMPRE existe (inicializado ou calculado)
if st.session_state.custos_fase:
    st.success("✅ Orçamento calculado com sucesso!")

    # Exibir resumo
    col1, col2, col3 = st.columns(3)
    with col1:
        st.metric("Área da Piscina", f"{st.session_state.area:.2f} m²")
    with col2:
        total_custos = sum(st.session_state.custos.values())
        st.metric("Custo Total", f"R$ {total_custos:,.2f}")
```

```
    with col3:
        volume = st.session_state.materiais.get("Volume de água (L)", 0)
        st.metric("Volume de Água", f"{volume:,.0f} L")

    st.markdown("---")

    # Tabs para organizar informações
    tab1, tab2, tab3 = st.tabs(["📊 Custos por Fase", "🧱 Materiais", "💰 Custos
Detalhados"])

    with tab1:
        st.subheader("Distribuição de Custos por Fase")
        # LINHA 78 ORIGINAL DO ERRO - AGORA CORRIGIDA
        st.dataframe(
            pd.DataFrame(list(st.session_state.custos_fase.items()),
columns=["Fase", "Custo (R$)"]),
            use_container_width=True
        )

        # Gráfico de custos por fase
        fig, ax = plt.subplots(figsize=(10, 6))
        cores = sns.color_palette("pastel")
        ax.barh(list(st.session_state.custos_fase.keys()),
                list(st.session_state.custos_fase.values()),
                color=cores[1])
        ax.set_xlabel("Custo (R$)")
        ax.set_title("Distribuição de custos por fase")
        st.pyplot(fig)

    with tab2:
        st.subheader("Lista de Materiais Necessários")
        st.dataframe(
            pd.DataFrame(list(st.session_state.materiais.items()),
columns=["Material", "Quantidade"]),
            use_container_width=True
        )

    with tab3:
        st.subheader("Custos Detalhados por Material")
        st.dataframe(
            pd.DataFrame(list(st.session_state.custos.items()),
columns=["Material", "Custo (R$)"]),
            use_container_width=True
        )

else:
    st.info("👈 Preencha os dados no painel lateral e clique em 'Calcular
Orçamento'")


# ======================
# FOOTER
# ======================
st.markdown("---")
st.markdown("**Desenvolvido com Streamlit** | Calculadora de Orçamento para
Piscinas v1.0")
```

App-Corrigido.py

```python
import math
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.units import inch
from typing import Dict, Tuple


# ========================
# FUNÇÕES DE CÁLCULO DE MATERIAIS
# ========================

def calcular_area(largura: float, comprimento: float) -> float:
    return largura * comprimento


def calcular_perimetro(largura: float, comprimento: float) -> float:
    return 2 * (largura + comprimento)


def calcular_blocos(area: float) -> float:
    return area * 12.5


def calcular_tela(perimetro: float, profundidade: float) -> float:
    return (perimetro + 4 * profundidade) / 5


def calcular_impermeabilizante1(area: float) -> int:
    return math.ceil(area / 9)


def calcular_impermeabilizante2(area: float) -> int:
    return math.ceil(area / 4)


def calcular_cimento(area: float) -> float:
    return (0.013 + 0.038 + 0.14) * area / 50


def calcular_areia(area: float) -> float:
    return (0.065 + 0.004 + 0.025) * area


def calcular_ligmassa(area: float) -> float:
    return (0.0026 + 0.05) * area


def calcular_revestimento(area: float) -> float:
    return area


def calcular_argamassa(area: float) -> float:
    return 0.45 * area
```

```
def calcular_rejunte(area: float) -> float:
    return 0.05 * area / 20


def calcular_espacadores(area: float) -> float:
    return 12 * area


# =======================
# ENTRADA DE DADOS
# =======================

def coletar_dados_projeto() -> Dict[str, float]:
    """
    Coleta dados do projeto via input do terminal.
    Inclui tratamento de erros para entradas inválidas.
    """
    print("\n=== PROJETO PARA UMA FAMÍLIA ===")
    dados = {}

    try:
        dados["Nome_projeto"] = input("Nome do projeto ou cliente: ").strip()
        dados["Num_pessoas_familia"] = int(input("Quantas pessoas compõem a
família? "))
        dados["Vai_hidromassagem"] = input("O projeto inclui hidromassagem?
(Sim/Não): ").strip().capitalize()

        tipo = input("Tipo de piscina (Azulejo/Vinilico): ").strip().lower()
        dados["Tipo_piscina"] = "Vinilico" if tipo.startswith("v") else "Azulejo"

        print("\n=== DADOS DA PISCINA ===")
        dados["Largura"] = float(input("Largura da piscina (m): ").replace(',',
'.'))
        dados["Comprimento"] = float(input("Comprimento da piscina (m):
").replace(',', '.'))

        fundo_declive = input("Piscina com fundo em declive? (Sim/Não):
").strip().capitalize()
        if fundo_declive == "Sim":
            dados["Profundidade_min"] = float(input("Profundidade mínima (m):
").replace(',', '.'))
            dados["Profundidade_max"] = float(input("Profundidade máxima (m):
").replace(',', '.'))
        else:
            dados["Profundidade_min"] = dados["Profundidade_max"] =
float(input("Profundidade (m): ").replace(',', '.'))

        dados["Usar_revestimento"] = input("Tipo de acabamento
(Revestimento/Outro): ").strip().capitalize()
        if dados["Usar_revestimento"] == "Revestimento":
            dados["Revestimento_largura_peca"] = float(input("Largura da peça (m):
").replace(',', '.'))
            dados["Revestimento_altura_peca"] = float(input("Altura da peça (m):
").replace(',', '.'))

    except ValueError as e:
        print(f"❌ Erro na entrada de dados: {e}")
        print("Por favor, insira valores numéricos válidos.")
```

```
        raise

    return dados


# ======================
# CÁLCULOS PRINCIPAIS
# ======================

def calcular_tudo(
    dados_piscina: Dict[str, float],
    custo_unitario: Dict[str, float],
    extras: Dict[str, float],
    preco_agua_por_litro: float = 0.01,
    caminhoes_enchimento: int = 3,
    fluxo_mangueira_lph: float = 1000.0
) -> Tuple[Dict[str, float], Dict[str, float], Dict[str, float], float]:
    """
    Calcula todos os materiais, custos e custos por fase.

    CORREÇÃO: Parâmetro renomeado de 'caminhos_enchimento' para
'caminhoes_enchimento'
    para consistência com o uso posterior.
    """

    largura = dados_piscina["Largura"]
    comprimento = dados_piscina["Comprimento"]
    profundidade = (dados_piscina["Profundidade_min"] +
dados_piscina["Profundidade_max"]) / 2

    area = calcular_area(largura, comprimento)
    perimetro = calcular_perimetro(largura, comprimento)

    materiais = {
        "Blocos": calcular_blocos(area),
        "Tela para Quina Vivas (caixas)": calcular_tela(perimetro, profundidade),
        "Impermeabilizante1 (caixas 20kg)": calcular_impermeabilizante1(area),
        "Impermeabilizante2 (caixas 20kg)": calcular_impermeabilizante2(area),
        "Cimento (sacos)": calcular_cimento(area),
        "Areia (m³)": calcular_areia(area),
        "Ligmassa (litros)": calcular_ligmassa(area),
        "Argamassa ACIII (kg)": calcular_argamassa(area),
        "Rejunte Acrílico (sacos)": calcular_rejunte(area),
        "Espaçadores (unidades)": calcular_espacadores(area)
    }

    if dados_piscina["Usar_revestimento"] == "Revestimento":
        materiais["Revestimento (m²)"] = calcular_revestimento(area)

    if dados_piscina["Vai_hidromassagem"] == "Sim":
        materiais["Hidromassagem (kit)"] = 1

    if dados_piscina["Tipo_piscina"] == "Vinilico":
        area_manta = (area + perimetro * profundidade) * 1.10
        materiais["Manta Vinilica (m²)"] = round(area_manta, 3)

    # Enchimento da piscina
    volume_m3 = largura * comprimento * profundidade
    litros = volume_m3 * 1000
```

```
    materiais["Volume de água (L)"] = round(litros, 2)
    materiais["Caminhões de enchimento (unidades)"] = caminhoes_enchimento
    tempo_horas = litros / (caminhoes_enchimento * fluxo_mangueira_lph)
    materiais["Tempo estimado enchimento (h)"] = round(tempo_horas, 2)

    custos = {m: materiais[m] * custo_unitario.get(m, 0) for m in materiais}
    custos["Custo de enchimento (R$)"] = round(litros * preco_agua_por_litro, 2)

    if "Hidromassagem (kit)" in materiais:
        custos["Hidromassagem (kit)"] = extras.get("custo_hidromassagem_kit", 0)

    fases = {
        "Alvenaria": ["Blocos", "Cimento (sacos)"],
        "Impermeabilização": ["Impermeabilizante1 (caixas 20kg)",
"Impermeabilizante2 (caixas 20kg)"],
        "Chapisco/Reboco": ["Cimento (sacos)", "Areia (m³)", "Argamassa ACIII
(kg)"],
        "Revestimento": ["Revestimento (m²)", "Argamassa ACIII (kg)"],
        "Acabamento": ["Rejunte Acrílico (sacos)", "Espaçadores (unidades)"],
        "Enchimento": ["Custo de enchimento (R$)"]
    }
    if "Hidromassagem (kit)" in materiais:
        fases["Extras"] = ["Hidromassagem (kit)"]

    custos_fase = {fase: sum(custos.get(m, 0) for m in mats) for fase, mats in
fases.items()}
    return materiais, custos, custos_fase, area


# ========================
# GRÁFICOS
# ========================

def gerar_graficos(materiais, custos, custos_fase, area):
    """
    Gera gráficos de análise de materiais e custos.
    """
    import seaborn as sns
    import numpy as np
    os.makedirs("graficos", exist_ok=True)

    # Paleta pastel suave
    cores = sns.color_palette("pastel")

    if area == 0:
        quant_por_m2 = {m: 0 for m in materiais}
    else:
        quant_por_m2 = {m: q / area for m, q in materiais.items()}

    # === Gráfico em LINHAS com escala logarítmica ===
    plt.figure(figsize=(10, 6))
    x = np.arange(len(quant_por_m2))
    y = list(quant_por_m2.values())
    labels = list(quant_por_m2.keys())

    plt.plot(x, y, color="#9BBFE0", linewidth=2.5, marker="o", markersize=6,
             markerfacecolor="#F6BD60", alpha=0.8)

    plt.yscale("log")  # Escala logarítmica pra mostrar todos os valores
```

```
    plt.xticks(x, labels, rotation=80, ha="right")
    plt.grid(True, which="both", linestyle="--", alpha=0.4)
    plt.title("Quantidade de Materiais por m²", fontsize=13, fontweight="bold")
    plt.ylabel("Quantidade (escala logarítmica)")
    plt.tight_layout()
    plt.savefig("graficos/quantidade_por_m2.png", dpi=200)
    plt.close()

    # === Custos por fase ===
    plt.figure(figsize=(10,6))
    plt.barh(list(custos_fase.keys()), list(custos_fase.values()), color=cores[1])
    plt.xlabel("Custo (R$)")
    plt.title("Distribuição de custos por fase", fontsize=13, fontweight="bold")
    plt.tight_layout()
    plt.savefig("graficos/custos_por_fase.png", dpi=200)
    plt.close()

    # === Custos por material ===
    plt.figure(figsize=(12,6))
    sorted_custos = dict(sorted(custos.items(), key=lambda x: x[1], reverse=True))
    plt.barh(list(sorted_custos.keys()), list(sorted_custos.values()),
color=cores[2])
    plt.xlabel("Custo (R$)")
    plt.title("Custos por material (inclui enchimento da piscina)", fontsize=13,
fontweight="bold")
    plt.tight_layout()
    plt.savefig("graficos/custos_por_material.png", dpi=200)
    plt.close()

    # === Enchimento da Piscina ===
    # CORREÇÃO: Linha 234 - "Caminhos" alterado para "Caminhões"
    if "Custo de enchimento (R$)" in custos:
        fig, ax = plt.subplots(figsize=(9, 5))
        parametros = ["Volume (L)", "Caminhões", "Tempo (h)", "Custo (R$)"]
        valores = [
            materiais.get("Volume de água (L)", 0),
            materiais.get("Caminhões de enchimento (unidades)", 0),  # CORRIGIDO
            materiais.get("Tempo estimado enchimento (h)", 0),
            custos.get("Custo de enchimento (R$)", 0)
        ]
        bar_colors = ["#A8DADC", "#F6BD60", "#BDB2FF", "#FFADAD"]
        barras = ax.bar(parametros, valores, color=bar_colors, alpha=0.8,
edgecolor="gray")
        ax.set_title("Enchimento da Piscina - Volume, Caminhões, Tempo e Custo",
fontsize=14, fontweight="bold")
        ax.set_ylabel("Valores de enchimento", fontsize=11)
        ax.grid(axis="y", linestyle="--", alpha=0.5)
        for i, b in enumerate(barras):
            altura = b.get_height()
            ax.text(b.get_x() + b.get_width()/2, altura + (altura*0.02 if altura>0
else 0.1),
                    f"{valores[i]:,.2f}", ha="center", va="bottom", fontsize=10,
color="#333")
        plt.tight_layout()
        plt.savefig("graficos/custo_enchimento.png", dpi=200)
        plt.close(fig)


# ======================
```

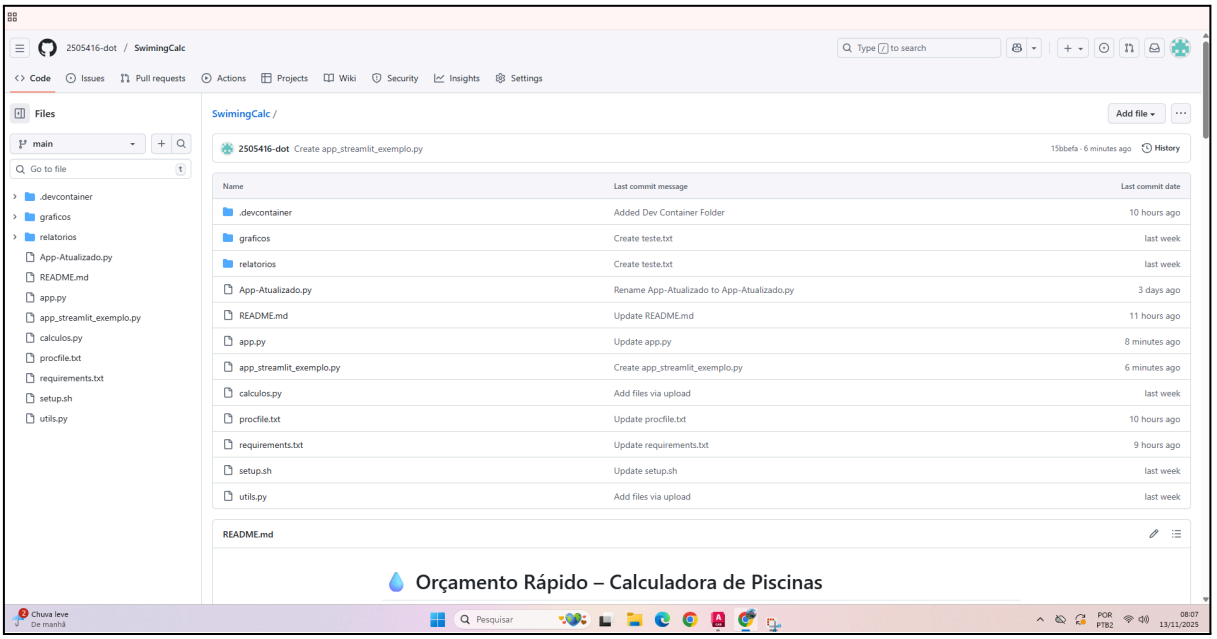
```
# RELATÓRIOS
# ========================

def salvar_excel(dados_piscina, materiais, custos, custos_fase):
    """
    Salva relatório completo em formato Excel.
    """
    try:
        with pd.ExcelWriter("relatorio_piscina.xlsx") as writer:
            pd.DataFrame([dados_piscina]).to_excel(writer,
sheet_name="Dados_Projeto", index=False)
            pd.DataFrame(list(materiais.items()), columns=["Material",
"Quantidade"]).to_excel(writer, sheet_name="Materiais", index=False)
            pd.DataFrame(list(custos.items()), columns=["Material", "Custo
(R$)"]).to_excel(writer, sheet_name="Custos", index=False)
            pd.DataFrame(list(custos_fase.items()), columns=["Fase", "Custo
(R$)"]).to_excel(writer, sheet_name="Custos_Fase", index=False)
        print("✅ Planilha Excel gerada: relatorio_piscina.xlsx")
    except Exception as e:
        print(f"❌ Erro ao gerar Excel: {e}")


def gerar_pdf(dados_piscina, materiais, custos, custos_fase):
    """
    Gera relatório em PDF com gráficos incorporados.
    """
    try:
        doc = SimpleDocTemplate("orcamento_piscina.pdf", pagesize=letter)
        styles = getSampleStyleSheet()
        story = []

        styles.add(ParagraphStyle(name='TitleStyle', fontSize=16, alignment=1,
spaceAfter=12))
        story.append(Paragraph(f"Orçamento de Piscina -
{dados_piscina['Nome_projeto']}", styles['TitleStyle']))

        def add_section(title: str, data: Dict[str, float]):
            story.append(Spacer(1, 8))
            story.append(Paragraph(f"<b>{title}</b>", styles['Heading3']))
            for k, v in data.items():
                story.append(Paragraph(f"{k}: {v}", styles['Normal']))

        add_section("Dados do Projeto", dados_piscina)
        add_section("Materiais", materiais)
        add_section("Custos por Material (R$)", {k: f"{v:.2f}" for k, v in
custos.items()})
        add_section("Custos por Fase (R$)", {k: f"{v:.2f}" for k, v in
custos_fase.items()})

        if "Volume de água (L)" in materiais:
            enchimento_data = {
                "Volume de água (L)": materiais.get("Volume de água (L)", 0),
                "Caminhões de enchimento (unidades)": materiais.get("Caminhões de
enchimento (unidades)", 0),
                "Tempo estimado enchimento (h)": materiais.get("Tempo estimado
enchimento (h)", 0),
                "Custo de enchimento (R$)": custos.get("Custo de enchimento (R$)",
0)
            }
```

```
            add_section("Informações de Enchimento", enchimento_data)

        # Adiciona gráficos
        for img_name in sorted(os.listdir("graficos")):
            img_path = os.path.join("graficos", img_name)
            if os.path.exists(img_path):
                story.append(Spacer(1, 12))
                story.append(Image(img_path, width=6*inch, height=4*inch))

        story.append(Spacer(1, 12))
        story.append(Paragraph("✅ Relatório gerado automaticamente com gráficos em
tons pastéis para melhor visualização.", styles['Normal']))

        doc.build(story)
        print("✅ PDF gerado: orcamento_piscina.pdf")
    except Exception as e:
        print(f"❌ Erro ao gerar PDF: {e}")


# ======================
# EXECUÇÃO PRINCIPAL
# ======================

def main():
    """
    Função principal que orquestra todo o fluxo de execução.
    """
    custo_unitario = {
        "Blocos": 1.5,
        "Tela para Quina Vivas (caixas)": 50,
        "Impermeabilizante1 (caixas 20kg)": 100,
        "Impermeabilizante2 (caixas 20kg)": 150,
        "Cimento (sacos)": 25,
        "Areia (m³)": 150,
        "Ligmassa (litros)": 10,
        "Revestimento (m²)": 60,
        "Argamassa ACIII (kg)": 20,
        "Rejunte Acrílico (sacos)": 40,
        "Espaçadores (unidades)": 0.1
    }
    extras = {"custo_hidromassagem_kit": 5000}

    preco_agua_por_litro = 0.01
    caminhoes_enchimento = 3  # CORRIGIDO: nome consistente
    fluxo_mangueira_lph = 1000.0

    try:
        dados_piscina = coletar_dados_projeto()
        materiais, custos, custos_fase, area = calcular_tudo(
            dados_piscina, custo_unitario, extras,
            preco_agua_por_litro, caminhoes_enchimento, fluxo_mangueira_lph
        )

        gerar_graficos(materiais, custos, custos_fase, area)
        salvar_excel(dados_piscina, materiais, custos, custos_fase)
        gerar_pdf(dados_piscina, materiais, custos, custos_fase)

        print("\n✅ Processamento concluído com sucesso!")
```

```
    except Exception as e:
        print(f"\n❌ Erro durante a execução: {e}")
        raise


if __name__ == "__main__":
    main()
```

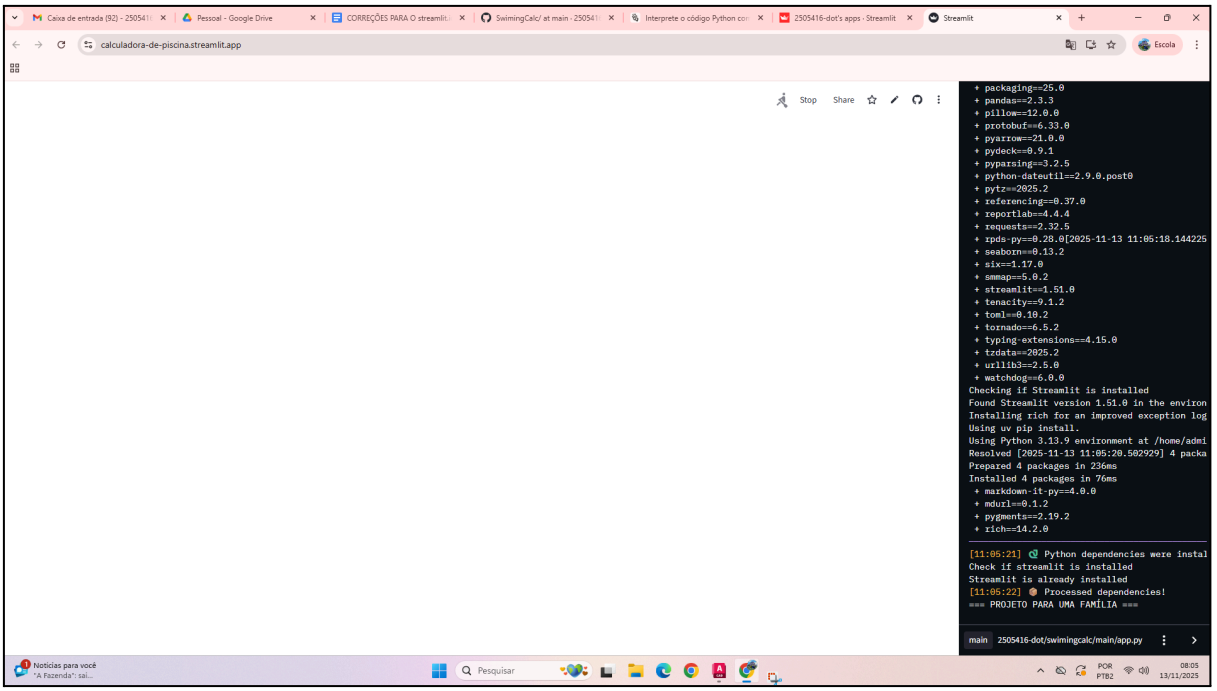
Realizei as alterações conforme a IA Manus recomendou e coloquei pra rodar novamente no streamlit.io conforme mostra tela de carregamento a seguir

Figura 13:*Github atualizado com nova pasta de código.*



Fonte: Arquivo dos autores

Figura 14:*Stremlit rodando novamente.*



Fonte: Arquivo dos autores

Aqui o erro ainda existia, não estava fazendo sentido o código rodar no colab e não funcionar aqui, então tentamos mais e mais vezes até ele funcionar, conforme registros.

Excelente! O log que você enviou revela exatamente o problema.
O Streamlit está tentando executar o seu arquivo `App-Atualizado.py` e, logo no início, ele encontra a função `coletar_dados_projeto()` que usa o comando `input()`:
Plain Text

```
[11:24:25]: # "📦 Processed dependencies!"

=== PROJETO PARA UMA FAMÍLIA ===
```

A linha `=== PROJETO PARA UMA FAMÍLIA ===` é a saída do `print()` dentro da sua função `coletar_dados_projeto()`.

🚨 O Problema

O Streamlit Cloud é um ambiente web e não possui um terminal interativo para receber comandos `input()`. Quando o Streamlit tenta executar a linha que contém `input()`, ele trava ou falha, pois não há usuário para fornecer a entrada.

Seu código original (`App-Atualizado.py`) foi escrito para ser executado em um terminal (usando `input()`), e não como um aplicativo web (usando widgets do Streamlit).

🛠️ A Solução

Você precisa remover a função `coletar_dados_projeto()` e substituir todas as chamadas a `input()` por widgets do Streamlit, como `st.text_input`, `st.number_input`, e `st.selectbox`.

Eu já havia preparado um arquivo de exemplo corrigido para o Streamlit: `app_streamlit_exemplo.py`.

Passos para Corrigir:

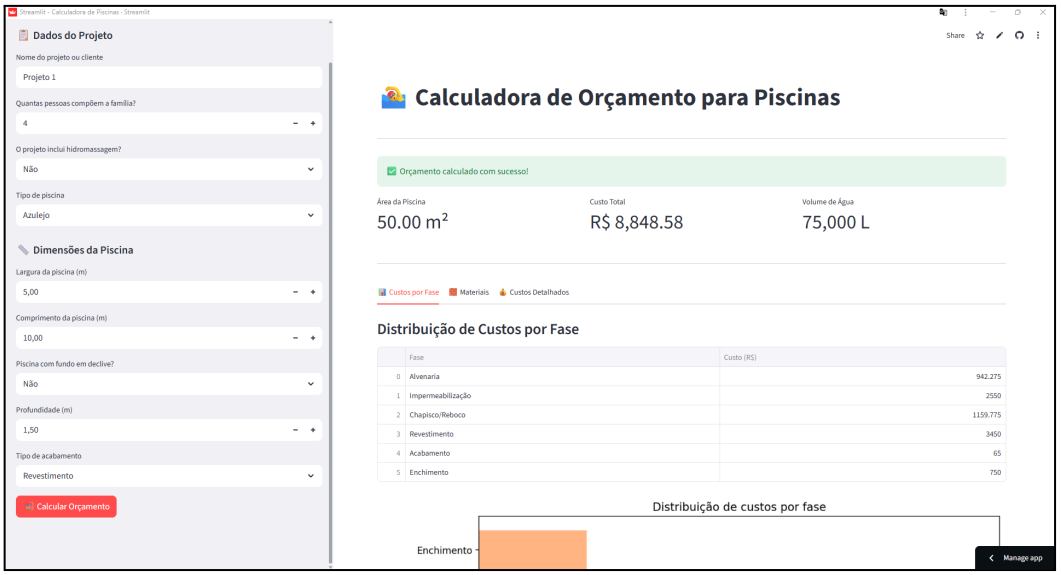
1. Renomeie o arquivo `App-Atualizado.py` no seu repositório para algo como `App-Terminal.py` (para guardar o original).
2. Crie um novo arquivo chamado `app.py` (ou use o `app_streamlit_exemplo.py` que eu forneci) e adapte-o para usar os widgets do Streamlit.
3. Certifique-se de que o Streamlit Cloud está configurado para rodar o novo arquivo (`app.py`).

Se você usar o `app_streamlit_exemplo.py` que eu forneci, o problema será resolvido, pois ele substitui todas as chamadas `input()` por widgets do Streamlit.

Por favor, confirme se você consegue substituir o conteúdo do seu `App-Atualizado.py` pelo conteúdo do `app_streamlit_exemplo.py` no seu repositório e tente um novo *reboot*.

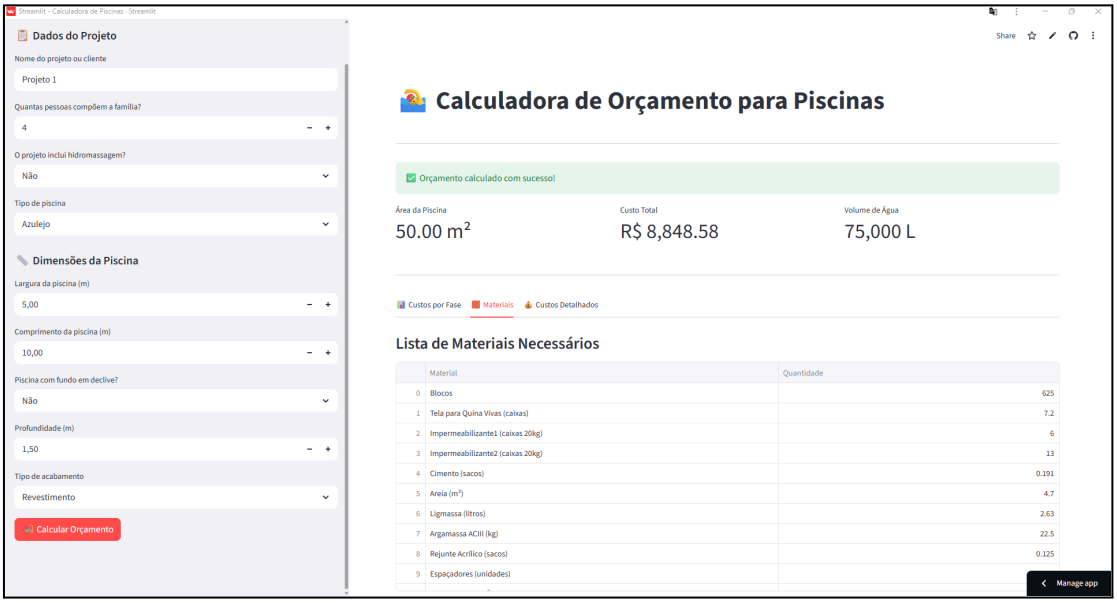
Se precisar de ajuda para fazer essa substituição no GitHub, me avise. Caso contrário, o problema é 100% o uso do `input()` em um ambiente web.

Figura 1:*Stremlit atualizado e finalizado.*



Fonte: Arquivo dos autores

Figura 16:*Stremlit aba de materiais.*



Fonte: Arquivo dos autores

Após criar novo app e pedir pra ele ler o **app_streamlit_exemplo.py** no lugar dos outros ele funcionou perfeitamente pedindo para o usuário preencher os dados ao lado e gerar orçamento https://2505416-dot-swimingcalc-app-streamlit-exemplo-vj1dyh.streamlit.app/ de tal

maneira criei um *qrcode* para o projeto podendo ser acessado por todos no dia da apresentação pelo site https://qr-code.io/pt-br/register-dpf?qr_onboarding=active_dpf&fx=QR-code&utm_source=google&utm_medium=cpc&utm_id=21666737442&utm_content=169935752871&utm_term=qr+code&network=g&matchtype=b&device=c&gaid=BR-PT-BR-C-DPF&gad_source=1&gad_campaignid=21666737442&gbraid=0AAAAA-G5uJJcK5JsRV1sTEm9kAGKZtcUC&gclid=EAIaIQobChMI_LiFj4TvkAMV8mpIAB1_OCgCEAAYASAAEgIOpPD_BwE segue nosso aplicativo funcionando.

Figura 17:*Qrcode* para aplicativo.

Fonte: Arquivo dos autores

Foi muito gratificante finalizar esse projeto e descobrir a utilidade de *Python* na vida acadêmica e no dia a dia de um engenheiro civil , será bem utilizado as aulas e ensinamentos dessa matéria e agradecemos ao professor pela ajuda e colaboração.

2.2 ATIVIDADES POR MEMBRO DO GRUPO

A seguir descrevemos o que cada membro da equipe, começamos a pensar sobre essa ferramenta para auxiliar em orçamentos específicos de piscina em setembro, assim dividimos as etapas entre cada um conforme tabela a seguir.

Divisão de atividades por membros do grupo	
Nome: Guilherme Lucas Da Silva **Ra:**2405785	Criou a lógica de programação, a apresentação de slides e trouxe dados referências dos materiais
Nome: Maria Luiza Mendes Andreasi **RA:**2505416	Realizou o código, pesquisou cada elemento do mesmo e para que servia, criou a tabela que resume cada etapa do código no relatório, juntamente da introdução.
Nome: Rafaela Nascimento De Souza Carvalho Porto **Ra:**2406550	Realizou o relatório, explicando o que queremos com esse projeto e onde ele seria aplicado.

3. CONSIDERAÇÕES FINAIS

Links importantes

Apresentação

https://www.canva.com/design/DAGyrNNmDWM/w6tO0PoC5nvvhu7UeAQb5w/edit?utm_content=DAGyrNNmDWM&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Versão final do código

https://colab.research.google.com/drive/1x-VAMGnUe91TP8otv70r3IbgplMzCqfC?usp=sharing

Github

https://github.com/2505416-dot/SwimingCalc

4. REFERÊNCIAS BIBLIOGRÁFICAS

Dados de referência de materiais retirados do banco de dados de Levinci Construtora https://www.obraprimaweb.com.br/

SODRÉ, Rafael. Tipos de Piscinas: qual o melhor modelo para sua casa? Viva Decora, 2024. Disponível em: https://www.vivadecora.com.br. Acesso em: 9 nov. 2025.