Resumo do projeto

## **Visão Geral do Front-End**

O front-end será desenvolvido com **Streamlit**, focando em:

* Simplicidade e acessibilidade
* Visualização clara de dados
* Navegação intuitiva para usuários não técnicos

## **Estrutura de Telas**

### **1. Tela Inicial**

* **Nome do app**: *Amei, nota zero*
* **Descrição breve** do propósito do app
* Botão para iniciar análise
* Menu lateral com opções: “Sobre”, “Analisar Avaliações”, “Visualizar Insights”, “Exportar Dados”

### **2. Tela de Upload de Dados**

* Upload de arquivo .csv ou entrada manual de avaliações
* Campos esperados: ID, Texto da Avaliação, Nota (opcional)
* Validação de formato e preview dos dados

### **3. Tela de Análise de Sentimento**

* Visualização de sentimentos (positivo, neutro, negativo)
* Gráfico de pizza ou barras com porcentagem de cada sentimento
* Lista de avaliações agrupadas por sentimento

### **4. Tela de Extração de Tópicos**

* Palavras-chave mais frequentes
* Tópicos agrupados por elogios e críticas
* Nuvem de palavras (WordCloud)

### **5. Tela de Insights com LLM**

* Resumo automático dos principais pontos
* Sugestões de melhoria com base nos feedbacks
* Destaques de frases impactantes

### **6. Tela de Visualização de Dados**

* Gráficos interativos com Seaborn ou Plotly
* Filtros por data, sentimento, nota
* Painel de controle com métricas gerais

### **7. Tela de Acessibilidade**

* Alternância de modo alto contraste
* Textos alternativos para gráficos
* Botão para aumentar fonte

## **Tecnologias e Integrações**

| **Componente** | **Tecnologia** | **Função** |
| --- | --- | --- |
| Front-end | Streamlit | Interface interativa |
| NLP | SpaCy / Hugging Face | Análise de sentimento e tópicos |
| LLM | OpenAI / Hugging Face | Geração de resumos e insights |
| Visualização | Seaborn / Plotly | Gráficos e painéis |
| Banco de Dados | SQLite | Armazenamento local |
| Repositório | GitHub | Controle de versão e colaboração |

## **Fluxo do Usuário**

1. Usuário acessa o app e faz upload dos dados
2. O app analisa os textos com NLP e LLM
3. Resultados são exibidos em gráficos e resumos
4. Usuário pode exportar os insights ou salvar no banco local

Códigos

AmeiNotaZero

├── main.py

├── estilos

│ ├── **\_\_init\_\_**.py

│ └── visual.py

├── funcionalidades

│ ├── **\_\_init\_\_**.py

│ ├── carregamento.py

│ ├── resumo.py

│ └── visualizacao.py

para rodar o código, você deve abrir o terminal e digitar:

.venv\Scripts\activate

streamlit run main.py

Obs.: “.venv\Scripts\activate” precisa ser digitado apenas 1 vez ao abrir o terminal, para deixar o caminho salvo

main.py

## main.py

### Versão 3:

import streamlit as st

import pandas as pd

import altair as alt

from estilos.visual import aplicar\_estilos, obter\_paleta

from funcionalidades.carregamento import carregar\_arquivo, exibir\_dados

from funcionalidades.resumo import gerar\_resumo

from funcionalidades.visualizacao import gerar\_grafico\_barra

# Configuração inicial

st.set\_page\_config(page\_title="Amei, nota zero", layout="wide")

st.title("Amei, nota zero")

st.markdown("Automatização de análise de avaliações textuais em negócios online")

# Painel de Acessibilidade Visual

st.sidebar.markdown("### ♿ Acessibilidade Visual")

tamanho\_fonte = st.sidebar.radio(

"Selecione o tamanho da fonte:",

["Padrão", "Grande", "Extra Grande"]

)

# Aplicar estilo de fonte

aplicar\_estilos(tamanho\_fonte)

# Paleta de cores para gráficos

cores = obter\_paleta()

# Navegação

st.sidebar.title("Navegação")

pagina = st.sidebar.radio("Ir para:", [

"Início", "Análise de Sentimento", "Tópicos Relevantes",

"Resumo de Insights", "Visualização de Dados", "Relatório Final"

])

# Página: Início

if pagina == "Início":

st.header("Que bom ter você por aqui!")

st.markdown("""

Este aplicativo foi criado para ajudar microempreendedores a entender melhor o que seus clientes estão dizendo.

Nós vamos transformar suas avaliações textuais em \*\*insights acionáveis\*\*.

\*\*Funcionalidades:\*\*

- Análise de sentimento

- Extração de tópicos

- Resumo inteligente

- Visualização de dados

- Relatório final

""")

st.caption("Você pode enviar arquivos nos formatos PDF, CSV, Excel, TXT ou JSON.")

arquivo = st.file\_uploader("📁 Envie um arquivo com avaliações", type=["csv", "xlsx", "txt", "json", "pdf"])

if arquivo:

df = carregar\_arquivo(arquivo)

if df is not None:

st.session\_state["df\_avaliacoes"] = df

exibir\_dados(df)

# Página: Análise de Sentimento

elif pagina == "Análise de Sentimento":

st.header("Análise de Sentimento")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

sentimentos = pd.DataFrame({

"Sentimento": ["Positivo", "Neutro", "Negativo"],

"Quantidade": [len(df)//2, len(df)//4, len(df)//4]

})

else:

st.info("⚠️ Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

sentimentos = pd.DataFrame({

"Sentimento": ["Positivo", "Neutro", "Negativo"],

"Quantidade": [42, 18, 25]

})

total = sentimentos["Quantidade"].sum()

sentimentos["Percentual"] = (sentimentos["Quantidade"] / total \* 100).round(1)

# Gráfico de pizza com rótulos adaptativos

base = alt.Chart(sentimentos).encode(

theta=alt.Theta("Quantidade:Q"),

color=alt.Color("Sentimento:N", scale=alt.Scale(domain=["Positivo", "Neutro", "Negativo"],

range=[cores["Positivo"], cores["Neutro"], cores["Negativo"]]))

)

arcos = base.mark\_arc()

labels = base.mark\_text(radius=100, size=14, fontWeight="bold").encode(

text=alt.Text("label:N"),

color=alt.Color("textColor:N")

).transform\_calculate(

label="'Qtd: ' + datum.Quantidade + ' (' + datum.Percentual + '%)'",

textColor="datum.Sentimento === 'Neutro' ? 'black' : 'white'"

)

chart = (arcos + labels).properties(width=400, height=400)

col\_esq, col\_centro, col\_dir = st.columns([1, 2, 1])

with col\_centro:

st.altair\_chart(chart, use\_container\_width=False)

# Página: Tópicos Relevantes

elif pagina == "Tópicos Relevantes":

st.header("Tópicos Mais Frequentes")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

palavras = pd.Series(" ".join(df["Texto"]).lower().split())

top\_palavras = palavras.value\_counts().head(10)

st.markdown("Tópicos extraídos das avaliações:")

for palavra in top\_palavras.index:

st.markdown(f"- {palavra}")

else:

st.caption("⚠️ Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

topicos = ["atendimento", "preço", "qualidade", "tempo de espera", "ambiente", "profissionalismo"]

for item in topicos:

st.markdown(f"- {item}")

# Página: Resumo de Insights

elif pagina == "Resumo de Insights":

st.header("Resumo Inteligente")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

resumo = gerar\_resumo(df)

st.markdown(f"> {resumo}")

else:

st.caption("⚠️ Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

st.markdown("""

> “Os clientes elogiam fortemente o atendimento e a qualidade dos serviços, mas há críticas recorrentes sobre o tempo de espera.

Recomenda-se otimizar o agendamento para melhorar a experiência geral.”

""")

# Página: Visualização de Dados

elif pagina == "Visualização de Dados":

st.header("Painel de Visualização")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

dados = pd.DataFrame({

"Categoria": ["Elogios", "Sugestões", "Críticas"],

"Quantidade": [len(df)//2, len(df)//6, len(df)//3]

})

else:

st.caption("⚠️ Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

dados = pd.DataFrame({

"Categoria": ["Elogios", "Sugestões", "Críticas"],

"Quantidade": [60, 10, 30]

})

dados["Percentual"] = (dados["Quantidade"] / dados["Quantidade"].sum() \* 100).round(1)

chart = gerar\_grafico\_barra(dados, "Categoria", ["Elogios", "Sugestões", "Críticas"], cores)

st.altair\_chart(chart, use\_container\_width=True)

# Página: Relatório Final

elif pagina == "Relatório Final":

st.header("📄 Relatório Final de Avaliações")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

resumo = gerar\_resumo(df)

st.subheader("1. Visão Geral")

st.markdown(f"- \*\*Total de avaliações analisadas:\*\* {len(df)}")

st.markdown("- \*\*Formato original do arquivo:\*\* Texto Livre")

st.markdown("- \*\*Fonte:\*\* Arquivo enviado pelo usuário")

st.subheader("2. Distribuição de Sentimentos")

positivos = len(df) // 2

neutros = len(df) // 4

negativos = len(df) // 4

col1, col2, col3 = st.columns(3)

col1.metric("😊 Positivas", positivos)

col2.metric("😐 Neutras", neutros)

col3.metric("😠 Negativas", negativos)

st.subheader("3. Tópicos Mais Frequentes")

palavras = pd.Series(" ".join(df["Texto"]).lower().split())

top\_palavras = palavras.value\_counts().head(10)

st.markdown("Os termos mais recorrentes nas avaliações foram:")

for i, palavra in enumerate(top\_palavras.index, 1):

st.markdown(f"{i}. {palavra}")

st.subheader("4. Resumo Inteligente")

st.markdown(f"> {resumo}")

st.subheader("5. Recomendação Final")

st.markdown("---")

st.caption("Este relatório foi gerado automaticamente com base nas avaliações enviadas.")

st.info("💡 Dica: Para salvar este relatório, use a opção 'Imprimir como PDF' do seu navegador.")

else:

st.warning("⚠️ Nenhum arquivo foi carregado ainda. Por favor, envie um arquivo na aba 'Início' para gerar o relatório.")

# Rodapé

st.markdown("---")

st.caption("Este aplicativo foi desenvolvido com foco em acessibilidade visual, seguindo os princípios do WCAG 2.1.")

st.caption("Projeto Integrador - Univesp | Grupo DRP01-PJI240-SALA005GRUPO-006")

### 

### Versão 2:

import streamlit as st

import pandas as pd

import os

import fitz # PyMuPDF

import altair as alt

import re

from collections import Counter

# Função de estilos visuais

def aplicar\_estilos(tema, tamanho\_fonte):

estilos = "<style>"

if tema == "Claro":

estilos += """

.stApp {

background-color: #FFFFFF;

color: #31333F;

}

h1, h2, h3, h4, h5, h6, p, label, div, span {

color: #31333F !important;

}

.css-1d391kg, .css-1cpxqw2, .css-ffhzg2, .css-1v0mbdj, .css-1y4p8pa {

background-color: #F0F2F6 !important;

color: #31333F !important;

}

"""

elif tema == "Escuro":

estilos += """

.stApp {

background-color: #1E1E1E;

color: #FFFFFF;

}

h1, h2, h3, h4, h5, h6, p, label, div, span {

color: #FFFFFF !important;

}

.css-1d391kg, .css-1cpxqw2, .css-ffhzg2, .css-1v0mbdj, .css-1y4p8pa {

background-color: #2C2C2C !important;

color: #FFFFFF !important;

}

"""

if tamanho\_fonte == "Grande":

estilos += """

html, body, [class\*="css"] {

font-size: 18px !important;

}

"""

elif tamanho\_fonte == "Extra Grande":

estilos += """

html, body, [class\*="css"] {

font-size: 22px !important;

}

"""

estilos += "</style>"

st.markdown(estilos, unsafe\_allow\_html=True)

# Função para obter paleta de cores

def obter\_paleta(tema):

if tema == "Escuro":

return {

"Positivo": "#27AE60",

"Neutro": "#F1C40F",

"Negativo": "#E74C3C",

"Elogios": "#27AE60",

"Sugestões": "#F1C40F",

"Críticas": "#E74C3C"

}

else:

return {

"Positivo": "#2ECC71",

"Neutro": "#F8CAA0",

"Negativo": "#E74C3C",

"Elogios": "#2ECC71",

"Sugestões": "#F8CAA0",

"Críticas": "#E74C3C"

}

# Configuração inicial

st.set\_page\_config(page\_title="Amei, nota zero", layout="wide")

st.title("Amei, nota zero")

st.markdown("Automatização de análise de avaliações textuais em negócios online")

# Painel de Acessibilidade Visual

st.sidebar.markdown("### ♿ Acessibilidade Visual")

tema = st.sidebar.selectbox("🎨 Tema visual", ["Claro", "Escuro"])

tamanho\_fonte = st.sidebar.selectbox("🔠 Tamanho da fonte", ["Padrão", "Grande", "Extra Grande"])

aplicar\_estilos(tema, tamanho\_fonte)

# Funções auxiliares

def carregar\_arquivo(arquivo):

nome = arquivo.name

ext = os.path.splitext(nome)[1].lower()

try:

if ext == ".csv":

df = pd.read\_csv(arquivo)

elif ext == ".xlsx":

df = pd.read\_excel(arquivo)

elif ext == ".txt":

df = pd.DataFrame({"Texto": arquivo.read().decode("utf-8").splitlines()})

elif ext == ".json":

df = pd.read\_json(arquivo)

elif ext == ".pdf":

texto = ""

with fitz.open(stream=arquivo.read(), filetype="pdf") as doc:

for pagina\_pdf in doc:

texto += pagina\_pdf.get\_text()

linhas = [linha for linha in texto.split("\n") if linha.strip()]

df = pd.DataFrame({"Texto": linhas})

else:

st.error("Formato de arquivo não suportado.")

return None

st.session\_state["df\_avaliacoes"] = df

return df

except Exception as e:

st.error(f"Erro ao processar o arquivo: {e}")

return None

def exibir\_dados(df):

st.success("✅ Arquivo carregado com sucesso!")

st.write("Visualização dos dados enviados:")

st.dataframe(df.head())

def gerar\_resumo(df):

textos = df["Texto"].dropna().astype(str).tolist()

textos\_limpos = [re.sub(r"[^\w\s]", "", t.lower()) for t in textos]

elogios = ["bom", "ótimo", "excelente", "maravilhoso", "adoro", "recomendo"]

criticas = ["ruim", "péssimo", "demorado", "caro", "não gostei", "problema"]

sugestoes = ["poderia", "deveria", "sugiro", "seria melhor", "melhorar"]

contagem\_elogios = sum(any(p in t for p in elogios) for t in textos\_limpos)

contagem\_criticas = sum(any(p in t for p in criticas) for t in textos\_limpos)

contagem\_sugestoes = sum(any(p in t for p in sugestoes) for t in textos\_limpos)

palavras = " ".join(textos\_limpos).split()

top\_palavras = Counter(palavras).most\_common(5)

temas = ", ".join([p[0] for p in top\_palavras])

resumo = f"""

Foram analisadas {len(df)} avaliações.

Identificamos {contagem\_elogios} elogios, {contagem\_criticas} críticas e {contagem\_sugestoes} sugestões.

Os temas mais recorrentes incluem: {temas}.

Recomenda-se atenção especial aos pontos críticos e valorização dos aspectos positivos percebidos pelos clientes.

"""

return resumo

# Navegação

st.sidebar.title("Navegação")

pagina = st.sidebar.radio("Ir para:", [

"Início", "Análise de Sentimento", "Tópicos Relevantes",

"Resumo de Insights", "Visualização de Dados", "Relatório Final"

])

# Página: Início

if pagina == "Início":

st.header("Que bom ter você por aqui!")

st.markdown("""

Este aplicativo foi criado para ajudar microempreendedores a entender melhor o que seus clientes estão dizendo.

Nós vamos transformar suas avaliações textuais em \*\*insights acionáveis\*\*.

\*\*Funcionalidades:\*\*

- Análise de sentimento

- Extração de tópicos

- Resumo inteligente

- Visualização de dados

- Relatório final

""")

st.caption("Você pode enviar arquivos nos formatos PDF, CSV, Excel, TXT ou JSON.")

arquivo = st.file\_uploader("📁 Envie um arquivo com avaliações", type=["csv", "xlsx", "txt", "json", "pdf"])

if arquivo:

df = carregar\_arquivo(arquivo)

if df is not None:

exibir\_dados(df)

# Página: Análise de Sentimento

elif pagina == "Análise de Sentimento":

st.header("Análise de Sentimento")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

sentimentos = pd.DataFrame({

"Sentimento": ["Positivo", "Neutro", "Negativo"],

"Quantidade": [len(df)//2, len(df)//4, len(df)//4]

})

else:

st.info("⚠️ Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

sentimentos = pd.DataFrame({

"Sentimento": ["Positivo", "Neutro", "Negativo"],

"Quantidade": [42, 18, 25]

})

cores = obter\_paleta(tema)

ordem = ["Positivo", "Neutro", "Negativo"]

chart = alt.Chart(sentimentos).mark\_bar().encode(

x=alt.X("Sentimento", sort=ordem),

y="Quantidade",

color=alt.Color("Sentimento", scale=alt.Scale(domain=list(cores.keys()), range=list(cores.values())))

).properties(width=600, height=400)

st.altair\_chart(chart, use\_container\_width=True)

# Página: Tópicos Relevantes

elif pagina == "Tópicos Relevantes":

st.header("Tópicos Mais Frequentes")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

palavras = pd.Series(" ".join(df["Texto"]).lower().split())

top\_palavras = palavras.value\_counts().head(10)

st.markdown("Tópicos extraídos das avaliações:")

for palavra in top\_palavras.index:

st.markdown(f"- {palavra}")

else:

st.caption("⚠️ Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

topicos = ["atendimento", "preço", "qualidade", "tempo de espera", "ambiente", "profissionalismo"]

for item in topicos:

st.markdown(f"- {item}")

# Página: Resumo de Insights

elif pagina == "Resumo de Insights":

st.header("Resumo Inteligente")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

resumo = gerar\_resumo(df)

st.markdown(f"> {resumo}")

else:

st.caption("⚠️ Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

st.markdown("""

> “Os clientes elogiam fortemente o atendimento e a qualidade dos serviços, mas há críticas recorrentes sobre o tempo de espera.

Recomenda-se otimizar o agendamento para melhorar a experiência geral.”

""")

# Página: Visualização de Dados

elif pagina == "Visualização de Dados":

st.header("Painel de Visualização")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

dados = pd.DataFrame({

"Categoria": ["Elogios", "Sugestões", "Críticas"],

"Volume": [len(df)//2, len(df)//6, len(df)//3]

})

else:

st.caption("⚠️ Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

dados = pd.DataFrame({

"Categoria": ["Elogios", "Sugestões", "Críticas"],

"Volume": [60, 10, 30]

})

ordem = ["Elogios", "Sugestões", "Críticas"]

cores = obter\_paleta(tema)

chart = alt.Chart(dados).mark\_bar().encode(

x=alt.X("Categoria", sort=ordem),

y="Volume",

color=alt.Color("Categoria", scale=alt.Scale(domain=list(cores.keys()), range=list(cores.values())))

).properties(width=600, height=400)

st.altair\_chart(chart, use\_container\_width=True)

# Página: Relatório Final

elif pagina == "Relatório Final":

st.header("📄 Relatório Final de Avaliações")

if "df\_avaliacoes" in st.session\_state:

df = st.session\_state["df\_avaliacoes"]

resumo = gerar\_resumo(df)

st.markdown("#### 1. Visão Geral")

st.markdown(f"- Total de avaliações analisadas: \*\*{len(df)}\*\*")

st.markdown("- Formato original do arquivo: \*\*Texto Livre\*\*")

st.markdown("- Fonte: Arquivo enviado pelo usuário")

st.markdown("#### 2. Distribuição de Sentimentos (simulada)")

positivos = len(df) // 2

neutros = len(df) // 4

negativos = len(df) // 4

st.markdown(f"- Positivas: \*\*{positivos}\*\*")

st.markdown(f"- Neutras: \*\*{neutros}\*\*")

st.markdown(f"- Negativas: \*\*{negativos}\*\*")

st.markdown("#### 3. Tópicos Mais Frequentes")

palavras = pd.Series(" ".join(df["Texto"]).lower().split())

top\_palavras = palavras.value\_counts().head(10)

for i, palavra in enumerate(top\_palavras.index, 1):

st.markdown(f"{i}. {palavra}")

st.markdown("#### 4. Resumo Inteligente")

st.markdown(f"> {resumo}")

st.markdown("#### 5. Recomendação Final")

st.markdown("""

- Priorizar melhorias nos pontos críticos identificados

- Monitorar avaliações futuras para medir impacto das ações

- Repetir análise periodicamente para manter o alinhamento com a experiência do cliente

""")

st.markdown("---")

st.caption("Este relatório foi gerado automaticamente com base nas avaliações enviadas.")

st.info("💡 Dica: Para salvar este relatório, use a opção 'Imprimir como PDF' do seu navegador.")

else:

st.warning("⚠️ Nenhum arquivo foi carregado ainda. Por favor, envie um arquivo na aba 'Início' para gerar o relatório.")

# Rodapé

st.markdown("---")

st.caption("Este aplicativo foi desenvolvido com foco em acessibilidade visual, seguindo os princípios do WCAG 2.1.")

st.caption("Projeto Integrador - Univesp | Grupo DRP01-PJI240-SALA005GRUPO-006")

### Versão 1:

import streamlit as st

import pandas as pd

import os

import fitz # PyMuPDF para leitura de PDFs

import altair as alt

st.set\_page\_config(page\_title="Amei, nota zero", layout="wide")

st.title("Amei, nota zero")

st.markdown("Automatização de análise de avaliações textuais em negócios online")

try:

st.sidebar.title("Navegação")

pagina = st.sidebar.radio("Ir para:", ["Início", "Análise de Sentimento", "Tópicos Relevantes", "Resumo de Insights", "Visualização de Dados"])

except Exception as e:

st.error(f"Erro ao carregar menu lateral: {e}")

pagina = "Início"

# Página: Início

if pagina == "Início":

st.header("Que bom ter você por aqui!")

st.markdown("""

Este aplicativo foi criado para ajudar microempreendedores a entender melhor o que seus clientes estão dizendo.

Nós vamos transformar suas avaliações textuais em \*\*insights acionáveis\*\*.

\*\*Funcionalidades:\*\*

- Análise de sentimento

- Extração de tópicos

- Resumo inteligente

- Visualização de dados

""")

# Página: Análise de Sentimento

elif pagina == "Análise de Sentimento":

st.header("Análise de Sentimento")

st.caption("Você pode enviar arquivos nos formatos PDF, CSV, Excel, TXT ou JSON.")

arquivo = st.file\_uploader("📁 Envie um arquivo com avaliações", type=["csv", "xlsx", "txt", "json", "pdf"])

if arquivo is not None:

nome = arquivo.name

ext = os.path.splitext(nome)[1].lower()

try:

if ext == ".csv":

df = pd.read\_csv(arquivo)

elif ext == ".xlsx":

df = pd.read\_excel(arquivo)

elif ext == ".txt":

df = pd.DataFrame({"Texto": arquivo.read().decode("utf-8").splitlines()})

elif ext == ".json":

df = pd.read\_json(arquivo)

elif ext == ".pdf":

texto = ""

with fitz.open(stream=arquivo.read(), filetype="pdf") as doc:

for pagina\_pdf in doc:

texto += pagina\_pdf.get\_text()

linhas = [linha for linha in texto.split("\n") if linha.strip()]

df = pd.DataFrame({"Texto": linhas})

else:

st.error("Formato de arquivo não suportado.")

df = None

if df is not None:

st.success("Arquivo carregado com sucesso!")

st.write("Visualização dos dados enviados:")

st.dataframe(df.head())

# Simulação de contagem de sentimentos

st.markdown("### Distribuição simulada de sentimentos")

sentimentos = pd.DataFrame({

"Sentimento": ["Positivo", "Neutro", "Negativo"],

"Quantidade": [len(df)//2, len(df)//4, len(df)//4]

})

# Ordenar e colorir com Altair

ordem\_sentimentos = ["Positivo", "Neutro", "Negativo"]

cores\_sentimentos = {

"Positivo": "#2ECC71",

"Neutro": "#F8CAA0",

"Negativo": "#E74C3C"

}

chart = alt.Chart(sentimentos).mark\_bar().encode(

x=alt.X("Sentimento", sort=ordem\_sentimentos, axis=alt.Axis(labelAngle=0)),

y="Quantidade",

color=alt.Color("Sentimento", scale=alt.Scale(domain=list(cores\_sentimentos.keys()), range=list(cores\_sentimentos.values()))),

).properties(width=600, height=400)

st.altair\_chart(chart, use\_container\_width=True)

except Exception as e:

st.error(f"Erro ao processar o arquivo: {e}")

else:

st.info("Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

sentimentos = pd.DataFrame({

"Sentimento": ["Positivo", "Neutro", "Negativo"],

"Quantidade": [42, 18, 25]

})

ordem\_sentimentos = ["Positivo", "Neutro", "Negativo"]

cores\_sentimentos = {

"Positivo": "#2ECC71",

"Neutro": "#F8CAA0",

"Negativo": "#E74C3C"

}

chart = alt.Chart(sentimentos).mark\_bar().encode(

x=alt.X("Sentimento", sort=ordem\_sentimentos),

y="Quantidade",

color=alt.Color("Sentimento", scale=alt.Scale(domain=list(cores\_sentimentos.keys()), range=list(cores\_sentimentos.values())))

).properties(width=600, height=400)

st.altair\_chart(chart, use\_container\_width=True)

# Página: Tópicos Relevantes

elif pagina == "Tópicos Relevantes":

st.header("Tópicos Mais Frequentes")

st.caption("Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

topicos = ["atendimento", "preço", "qualidade", "tempo de espera", "ambiente", "profissionalismo"]

st.markdown("Tópicos extraídos das avaliações:")

for item in topicos:

st.markdown(f"- {item}")

# Página: Resumo de Insights

elif pagina == "Resumo de Insights":

st.header("Resumo Inteligente")

st.caption("Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

st.markdown("""

> “Os clientes elogiam fortemente o atendimento e a qualidade dos serviços, mas há críticas recorrentes sobre o tempo de espera.

Recomenda-se otimizar o agendamento para melhorar a experiência geral.”

""")

# Página: Visualização de Dados

elif pagina == "Visualização de Dados":

st.header("Painel de Visualização")

st.caption("Nenhum arquivo ainda foi enviado. Você está vendo um exemplo com dados fictícios.")

dados = pd.DataFrame({

"Categoria": ["Elogios", "Sugestões", "Críticas"],

"Volume": [60, 10, 30]

})

ordem\_categorias = ["Elogios", "Sugestões", "Críticas"]

cores\_categorias = {

"Elogios": "#2ECC71",

"Sugestões": "#F8CAA0",

"Críticas": "#E74C3C"

}

chart\_dados = alt.Chart(dados).mark\_bar().encode(

x=alt.X("Categoria", sort=ordem\_categorias, axis=alt.Axis(labelAngle=0)),

y="Volume",

color=alt.Color("Categoria", scale=alt.Scale(domain=list(cores\_categorias.keys()), range=list(cores\_categorias.values())))

).properties(width=600, height=400)

st.altair\_chart(chart\_dados, use\_container\_width=True)

# Rodapé

st.markdown("---")

st.caption("Desenvolvido para o Projeto Integrador - Univesp | Grupo DRP01-PJI240-SALA005GRUPO-006")

estilos

\_\_init\_\_.py

(vazio)

visual.py

## visual.py

### Versão 1:

import streamlit as st

def aplicar\_estilos(tamanho):

if tamanho == "Padrão":

tamanho\_css = "16px"

padding\_css = "0.5em 1em"

elif tamanho == "Grande":

tamanho\_css = "20px"

padding\_css = "0.75em 1.5em"

elif tamanho == "Extra Grande":

tamanho\_css = "24px"

padding\_css = "1em 2em"

else:

tamanho\_css = "16px"

padding\_css = "0.5em 1em"

st.markdown(f"""

<style>

html, body, [class\*="css"] {{

font-size: {tamanho\_css};

}}

button, .stButton > button {{

font-size: {tamanho\_css} !important;

padding: {padding\_css} !important;

}}

.stRadio > div {{

font-size: {tamanho\_css};

}}

</style>

""", unsafe\_allow\_html=True)

def obter\_paleta(tema="Claro"):

return {

"Positivo": "#2ECC71",

"Neutro": "#F8CAA0",

"Negativo": "#E74C3C",

"Elogios": "#2ECC71",

"Sugestões": "#F8CAA0",

"Críticas": "#E74C3C"

}

funcionalidades

\_\_init\_\_.py -

(vazio)

obs.: o hífen não faz parte do nome, coloquei para não dar erro aqui no doc.

carregamento.py

## carregamento.py

### Versão 1:

import streamlit as st

import pandas as pd

import os

import fitz # PyMuPDF

def carregar\_arquivo(arquivo):

nome = arquivo.name

ext = os.path.splitext(nome)[1].lower()

try:

if ext == ".csv":

df = pd.read\_csv(arquivo)

elif ext == ".xlsx":

df = pd.read\_excel(arquivo)

elif ext == ".txt":

df = pd.DataFrame({"Texto": arquivo.read().decode("utf-8").splitlines()})

elif ext == ".json":

df = pd.read\_json(arquivo)

elif ext == ".pdf":

texto = ""

with fitz.open(stream=arquivo.read(), filetype="pdf") as doc:

for pagina\_pdf in doc:

texto += pagina\_pdf.get\_text()

linhas = [linha for linha in texto.split("\n") if linha.strip()]

df = pd.DataFrame({"Texto": linhas})

else:

st.error("Formato de arquivo não suportado.")

return None

st.session\_state["df\_avaliacoes"] = df

return df

except Exception as e:

st.error(f"Erro ao processar o arquivo: {e}")

return None

def exibir\_dados(df):

st.success("✅ Arquivo carregado com sucesso!")

st.write("Visualização dos dados enviados:")

st.dataframe(df.head())

resumo.py

## resumo.py

### Versão 1:

import re

from collections import Counter

def gerar\_resumo(df):

textos = df["Texto"].dropna().astype(str).tolist()

textos\_limpos = [re.sub(r"[^\w\s]", "", t.lower()) for t in textos]

elogios = ["bom", "ótimo", "excelente", "maravilhoso", "adoro", "recomendo"]

criticas = ["ruim", "péssimo", "demorado", "caro", "não gostei", "problema"]

sugestoes = ["poderia", "deveria", "sugiro", "seria melhor", "melhorar"]

contagem\_elogios = sum(any(p in t for p in elogios) for t in textos\_limpos)

contagem\_criticas = sum(any(p in t for p in criticas) for t in textos\_limpos)

contagem\_sugestoes = sum(any(p in t for p in sugestoes) for t in textos\_limpos)

palavras = " ".join(textos\_limpos).split()

top\_palavras = Counter(palavras).most\_common(5)

temas = ", ".join([p[0] for p in top\_palavras])

resumo = f"""

Foram analisadas {len(df)} avaliações.

Identificamos {contagem\_elogios} elogios, {contagem\_criticas} críticas e {contagem\_sugestoes} sugestões.

Os temas mais recorrentes incluem: {temas}.

Recomenda-se atenção especial aos pontos críticos e valorização dos aspectos positivos percebidos pelos clientes.

"""

return resumo

visualizacao.py

README.md

**# Amei, nota zero**

Aplicativo interativo desenvolvido com Streamlit para análise automatizada de avaliações textuais em negócios online. Ele transforma comentários de clientes em insights acionáveis, facilitando a tomada de decisão para microempreendedores.

---

**## Funcionalidades**

- Upload de arquivos (PDF, CSV, Excel, TXT, JSON)

- Análise de sentimento (Positivo, Neutro, Negativo)

- Extração de tópicos mais frequentes

- Geração de resumo inteligente

- Visualização de dados com gráficos interativos

- Relatório final com recomendações

---

**## Estrutura do Projeto**

AmeiNotaZero/

├── main.py

├── estilos/

│ ├── **\_\_init\_\_**.py

│ └── visual.py

├── funcionalidades/

│ ├── **\_\_init\_\_**.py

│ ├── carregamento.py

│ ├── resumo.py

│ └── visualizacao.py

---

Instalação

**## 1. Clone o repositório:**

```bash

git clone https://github.com/seu-usuario/amei-nota-zero.git

cd amei-nota-zero

## 2. Crie e ative um ambiente virtual:

python -m venv .venv

source .venv/bin/activate # Linux/macOS

.venv\Scripts\activate # Windows

## 3. Instale as dependências:

pip install -r requirements.txt

## Como Executar:

streamlit run main.py

Requisitos

- Python 3.8+

- Streamlit

- pandas

- altair

- PyMuPDF (fitz)

Todas as dependências estão listadas no requirements.txt.

Autoria

Projeto Integrador — Univesp

Grupo DRP01-PJI240-SALA005GRUPO-006

Desenvolvido por Aline e equipe

Licença

Este projeto é de uso acadêmico e está disponível sob a licença MIT.

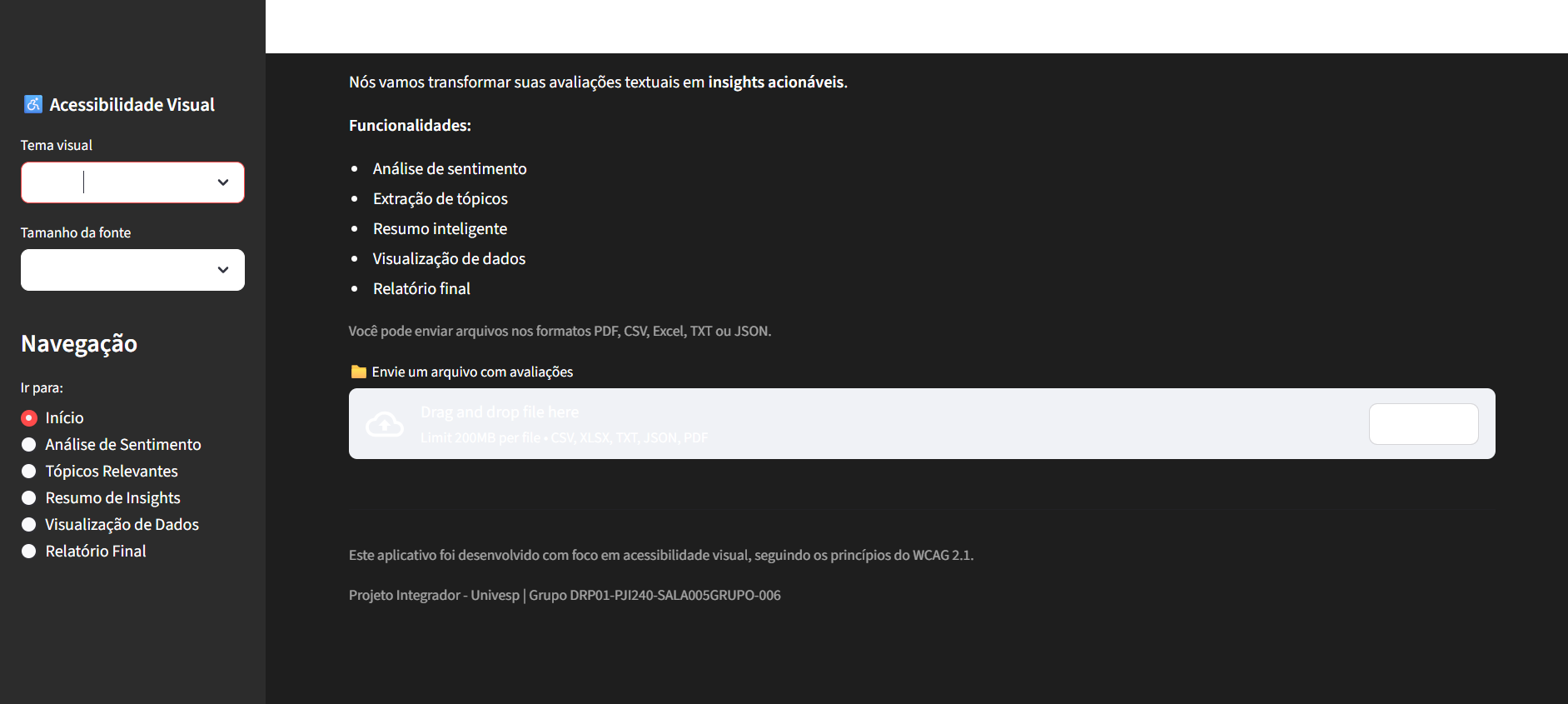
Sinta-se livre para adaptar e expandir conforme sua necessidade.

Visualização das telas

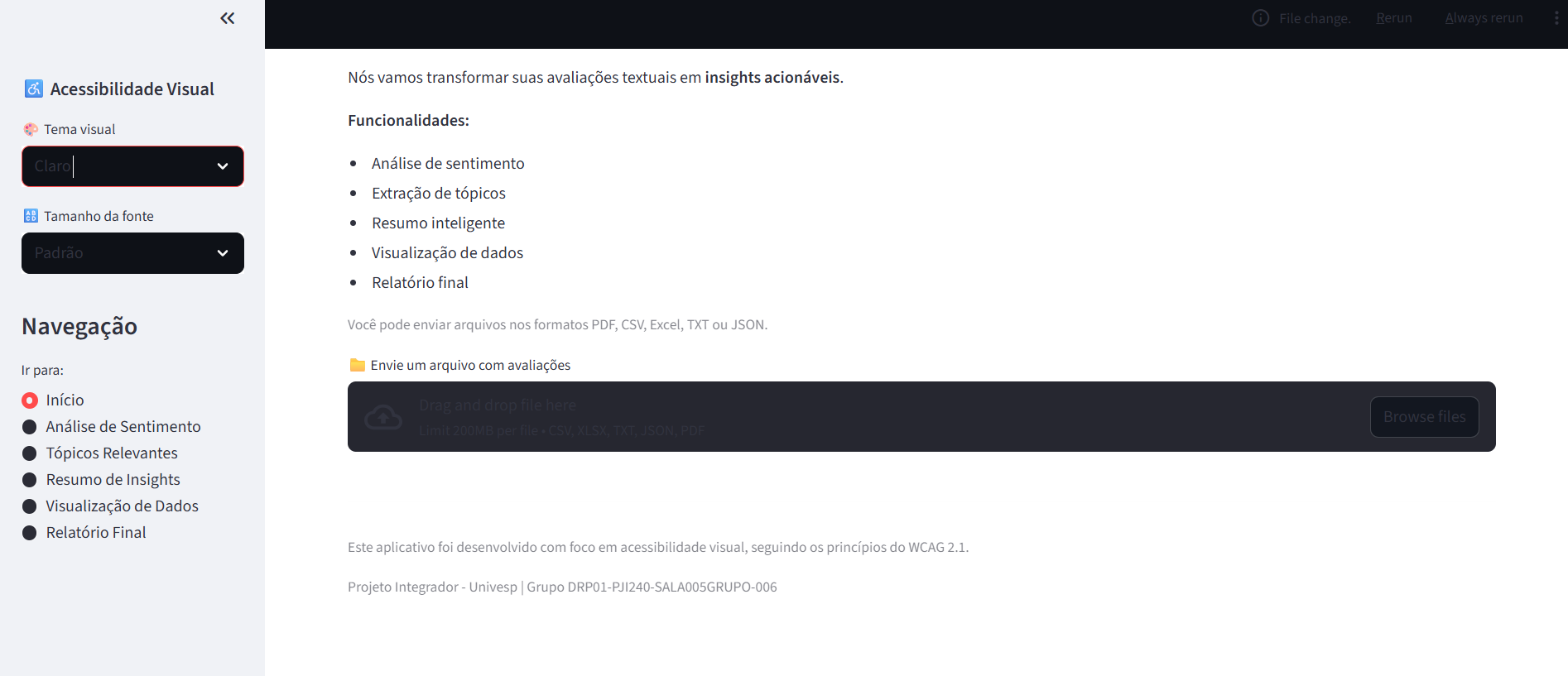
[Imagens telas app - Google Drive](https://drive.google.com/drive/u/2/folders/1UhG21hng-j-nlLHWKbbYEnriWBGx7tmC)

Erro em acessibilidade

Para navegadores configurados para o tema claro, a visualização do tema escuro ficou assim:



O inverso aconteceu em navegadores configurados para o tema claro:



Por este motivo, a função foi desativada por enquanto.