
Documentação Detalhada do Projeto: Chatbot Terminal com Consulta de PDFs

1. Visão Geral do Projeto

Este documento serve como um guia abrangente para entender, configurar e replicar um Chatbot de Consulta de PDFs baseado em Inteligência Artificial. O objetivo principal do chatbot é permitir que usuários interajam com documentos PDF, gerando resumos e conversando com a IA com base no conteúdo desses documentos. Além disso, ele oferece um robusto sistema de gerenciamento de sessões e a capacidade de exportar conversas para arquivos PDF, funcionando como "cadernos digitais" temáticos para organização do conhecimento.

Funcionalidades Principais:

- **Leitura e Resumo de PDFs:** Extrai texto de arquivos PDF e gera resumos inteligentes usando a API da OpenAI.
- **Interação Conversacional:** Permite ao usuário fazer perguntas e receber respostas baseadas no contexto do PDF ativo e no histórico da conversa.
- **Gerenciamento de Sessões:** Cria, carrega, lista e exclui sessões de chat, permitindo organizar conversas por tópico.
- **Gerenciamento de Resumos:** Salva, lista, carrega e exclui resumos de PDFs, oferecendo controle sobre o conteúdo processado.
- **Exportação de Chat para PDF:** Exporta o histórico completo de uma sessão de chat para um arquivo PDF, facilitando a revisão offline e o compartilhamento.
- **Controle de Tokens:** Otimiza o uso da API da OpenAI, garantindo que as interações permaneçam dentro dos limites de tokens do modelo, reduzindo custos e erros.
- **Tratamento de Erros:** Inclui mecanismos robustos para lidar com falhas de API, problemas de arquivo e entradas inesperadas do usuário.

2. Estrutura do Projeto

O chatbot é organizado de forma modular, o que facilita a manutenção, depuração e futuras expansões. A estrutura de diretórios e arquivos é a seguinte:

```
chatbot_otimizado/
├── config/
│   └── config.py
├── data/
│   ├── exports/
│   │   ├── [session_name]/
│   │   │   └── [export_name].pdf
│   ├── pdfs/
│   │   └── [your_documents].pdf
│   ├── profiles/
│   │   └── default/
│   │       ├── sessions/
│   │       │   └── [session_name].json
│   │       └── summaries/
│   │           └── [summary_id].json
│   └── utils/
│       ├── api_service.py
│       ├── pdf_exporter.py
│       ├── pdf_processor.py
│       ├── session_manager.py
│       └── token_utils.py
├── main.py
├── .env
├── requirements.txt
└── README.md
```

Explicação dos Componentes:

- **main.py:** O arquivo principal que orquestra todas as funcionalidades do chatbot. Ele gerencia o loop de interação com o usuário, processa comandos, integra os módulos e cuida do salvamento e carregamento do estado da sessão.
- **config/:**
 - **config.py:** Contém todas as configurações globais do chatbot, como a chave da API (carregada via .env), o modelo da OpenAI a ser utilizado, temperatura, limites de tokens, mensagens do sistema e instruções para resumos, e a definição de todos os comandos disponíveis.
- **data/:** Este diretório armazena os dados gerados e utilizados pelo chatbot, garantindo a persistência das informações.
 - **exports/:** Subdiretório para armazenar os PDFs exportados das conversas. Cada sessão tem seu próprio subdiretório dentro de exports/.
 - **pdfs/:** Onde os usuários devem colocar os arquivos PDF que desejam que o chatbot leia e resuma.
 - **profiles/default/:** Contém os dados persistentes das sessões e resumos.
 - **sessions/:** Armazena os arquivos JSON de cada sessão de chat, incluindo o

histórico de conversas e o resumo de PDF ativo.

- **summaries/**: Armazena os arquivos JSON com os resumos dos PDFs gerados pela IA, permitindo que sejam carregados e reutilizados.
- **utils/**: Contém módulos utilitários que encapsulam funcionalidades específicas.
 - **api_service.py**: Responsável pela comunicação com a API da OpenAI. Ele lida com as requisições de completude, tratamento de erros da API e configuração da chave.
 - **pdf_exporter.py**: Gerencia a exportação do histórico de chat para arquivos PDF. Utiliza a biblioteca fpdf para criar os documentos.
 - **pdf_processor.py**: Extrai texto de arquivos PDF usando a biblioteca PyPDF2.
 - **session_manager.py**: Lida com o salvamento, carregamento, listagem e exclusão de sessões de chat e resumos de PDF. É a camada de persistência de dados.
 - **token_utils.py**: Contém funções para contar tokens em strings e mensagens, essencial para gerenciar os limites de contexto da API da OpenAI e otimizar o uso de recursos.
- **.env**: (Opcional, mas recomendado) Arquivo para armazenar variáveis de ambiente, como a sua chave da API da OpenAI, de forma segura e fora do controle de versão.
- **requirements.txt**: Lista de todas as bibliotecas Python necessárias para o projeto.
- **README.md**: Um arquivo de documentação inicial (que este documento complementa).

3. Configuração do Ambiente de Desenvolvimento

Para que o chatbot funcione, você precisará configurar seu ambiente.

3.1. Pré-requisitos

- **Python**: Versão 3.8 ou superior (recomenda-se Python 3.9+).
- **pip**: O gerenciador de pacotes do Python (geralmente já vem com a instalação do Python).
- **Conexão com a Internet**: Necessária para baixar pacotes e interagir com a API da OpenAI.
- **Chave da API OpenAI**: Uma chave válida é fundamental. Você pode obtê-la em platform.openai.com.

3.2. Passos de Configuração

1. Crie o Diretório do Projeto:

```
Bash
mkdir chatbot_otimizado
cd chatbot_otimizado
```

2. Crie um Ambiente Virtual (Altamente Recomendado):
Isso isola as dependências do projeto, evitando conflitos com outras instalações Python no seu sistema.

Bash

```
python3 -m venv env
```

3. **Ative o Ambiente Virtual:**

- **Linux/macOS:**

Bash

```
source env/bin/activate
```

- **Windows (CMD):**

Bash

```
env\Scripts\activate.bat
```

- **Windows (PowerShell):**

PowerShell

```
.\env\Scripts\Activate.ps1
```

- *(Você saberá que está ativo quando (env) aparecer no início da sua linha de comando).*

4. Crie o Arquivo requirements.txt:

Crie um arquivo chamado requirements.txt na raiz do seu projeto (chatbot_otimizado/) e adicione as seguintes bibliotecas:

openai

tiktoken

PyPDF2

python-dotenv

fpdf

5. Instale as Dependências:

Com o ambiente virtual ativado, instale as bibliotecas listadas no requirements.txt:

Bash

```
pip install -r requirements.txt
```

6. Configure sua Chave da API OpenAI:

Crie um arquivo chamado .env na raiz do seu projeto (chatbot_otimizado/). Dentro dele, adicione sua chave da OpenAI no seguinte formato:

```
OPENAI_API_KEY="sua_chave_secreta_aqui"
```

Substitua "sua_chave_secreta_aqui" pela sua chave real da API da OpenAI.

Recomenda-se que este arquivo .env não seja versionado em sistemas como Git para

segurança.

7. Crie os Diretórios de Dados:

Crie as pastas necessárias para o armazenamento de dados:

Bash

```
mkdir -p data/pdfs data/profiles/default/sessions data/profiles/default/summaries  
data/exports
```

- data/pdfs: Coloque aqui os arquivos PDF que você deseja processar.
- data/profiles/default/sessions: Onde as sessões de chat serão salvas.
- data/profiles/default/summaries: Onde os resumos de PDF serão armazenados.
- data/exports: Onde os históricos de chat exportados para PDF serão salvos.

4. Implementação dos Módulos Essenciais

Agora, vamos criar os arquivos Python para cada módulo. Você pode copiá-los do seu `codigos.pdf` para os locais indicados.

4.1. Módulo de Configuração (`config/config.py`)

Crie o arquivo `config.py` dentro do diretório `config/`.

Este arquivo centraliza todas as constantes e configurações do seu chatbot.

Propósito:

- Define o modelo padrão da OpenAI, a temperatura (criatividade da IA).
- Estabelece limites de tokens para entrada e saída, crucial para controle de custos e desempenho.
- Contém as mensagens do sistema e instruções para o resumo, que guiam o comportamento da IA.
- Lista todos os comandos disponíveis para o usuário, facilitando a manutenção e adição de novos comandos.

Trechos Importantes (Não copiar o código completo, apenas para referência):

- `DEFAULT_MODEL`: O modelo da OpenAI a ser usado (ex: "gpt-4o-mini").
- `TEMPERATURE`: Controla a aleatoriedade da resposta (ex: 0.7).
- `MAX_TOKENS_LIMIT`: Limite máximo de tokens que o modelo pode processar (incluindo prompt e resposta).
- `SUMMARY_MAX_TOKENS`: Limite de tokens específico para as respostas de resumo.
- `HISTORY_MESSAGE_LIMIT`: Quantas mensagens anteriores manter no histórico para contexto.
- `SYSTEM_MESSAGE`: A persona inicial da IA.
- `SUMMARY_INSTRUCTION_MESSAGE`: Instruções dadas à IA ao gerar resumos.
- `DEFAULT_SESSION_NAME`: Nome da sessão padrão ao iniciar o chatbot.
- `PDFS_DIR`, `SESSIONS_DIR`, `SUMMARIES_DIR`, `EXPORTS_DIR`: Caminhos para os diretórios

de dados.

- **COMMANDS:** Um dicionário mapeando nomes de comandos internos para os comandos que o usuário digita (ex: "read_pdf": "/lerpdf").

4.2. Módulo de Serviço da API (utils/api_service.py)

Crie o arquivo api_service.py dentro do diretório utils/.

Propósito:

- Gerencia a comunicação com a API da OpenAI.
- Carrega a chave da API do .env.
- A função get_openai_completion envia as mensagens para o modelo da OpenAI e lida com possíveis erros da API (limite de taxa, conexão, etc.).

Funções Chave:

- get_openai_completion(messages: list, model: str, temperature: float, max_tokens: int = None) -> str: A função principal para interagir com a OpenAI. Recebe uma lista de mensagens (histórico de chat), o modelo, a temperatura e opcionalmente um limite de tokens para a resposta.

4.3. Módulo de Processamento de PDF (utils/pdf_processor.py)

Crie o arquivo pdf_processor.py dentro do diretório utils/.

Propósito:

- Extrai texto de arquivos PDF.
- Utiliza a biblioteca PyPDF2 para ler o conteúdo textual de cada página do PDF.

Funções Chave:

- extract_text_from_pdf(pdf_path: str) -> str: Recebe o caminho para um arquivo PDF e retorna todo o texto extraído como uma única string. Inclui tratamento para arquivos não encontrados ou erros de leitura.

4.4. Módulo de Gerenciamento de Sessões (utils/session_manager.py)

Crie o arquivo session_manager.py dentro do diretório utils/.

Propósito:

- Implementa as funcionalidades de persistência de dados para o chatbot.
- Salva e carrega o histórico de chat de sessões.
- Salva e carrega os resumos de PDF, associando-os a IDs únicos.
- Permite listar e excluir sessões e resumos salvos.

Funções Chave:

- load_session(session_name: str) -> dict: Carrega uma sessão específica.
- save_session(session_name: str, chat_history: list, active_api_summary_content: str =

- None, active_api_summary_metadata: dict = None): Salva o estado atual de uma sessão.
- list_sessions() -> list: Lista todas as sessões disponíveis.
 - delete_session(session_name: str) -> bool: Exclui uma sessão e seus dados associados.
 - save_pdf_summary(content: str, metadata: dict) -> str: Salva um resumo de PDF com metadados.
 - load_specific_pdf_summary(summary_id: str) -> dict: Carrega um resumo específico pelo ID.
 - list_summaries() -> list: Lista todos os resumos de PDF salvos.
 - delete_pdf_summary(summary_id: str) -> bool: Exclui um resumo de PDF.

4.5. Módulo de Utilitários de Tokens (utils/token_utils.py)

Crie o arquivo token_utils.py dentro do diretório utils/.

Propósito:

- Calcula o número de tokens em strings e em listas de mensagens formatadas para a API da OpenAI.
- É crucial para garantir que o prompt enviado à IA não exceda os limites de contexto do modelo, evitando erros e otimizando o custo.

Funções Chave:

- count_tokens_in_string(text: str) -> int: Conta tokens em uma string simples.
- count_tokens_in_messages(messages: list) -> int: Estima o número de tokens em uma lista de mensagens no formato da API da OpenAI, considerando a estrutura role/content.

4.6. Módulo de Exportação de PDF (utils/pdf_exporter.py)

Crie o arquivo pdf_exporter.py dentro do diretório utils/.

Propósito:

- Exporta o histórico de chat de uma sessão para um arquivo PDF legível.
- Permite listar, carregar e excluir esses arquivos PDF exportados.

Funções Chave:

- export_chat_to_pdf(session_name: str, export_name: str, chat_history: list) -> bool: Salva o chat_history de uma sessão como um PDF.
- list_exported_pdfs(session_name: str = None) -> list: Lista os PDFs exportados, opcionalmente filtrando por sessão.
- get_exported_pdf_path(session_name: str, export_name: str) -> str: Retorna o caminho completo para um PDF exportado específico.
- delete_exported_pdf(session_name: str, export_name: str) -> bool: Exclui um PDF de exportação.

5. O Arquivo Principal (main.py)

Crie o arquivo main.py na raiz do seu projeto (chatbot_otimizado/).

Propósito:

Este é o coração do chatbot. Ele integra todos os módulos criados, gerencia o ciclo de vida da aplicação e a interação com o usuário.

Estrutura Lógica (alto nível):

1. **Importações:** Importa todas as configurações do config.py e as funções dos módulos em utils/.
2. **Variáveis de Estado Global:** chat_history, active_api_summary_content, active_api_summary_metadata, current_session_name são definidas aqui para gerenciar o estado atual do chatbot.
3. **Funções Auxiliares de Gerenciamento de Sessão:** Funções como load_session_state e save_session_state encapsulam a lógica de interação com o session_manager.
4. **Manipuladores de Comandos:**
 - **/ajuda (display_help()):** Exibe uma lista de todos os comandos disponíveis para o usuário.
 - **/lerpdf <nome_do_arquivo.pdf> (handle_read_pdf()):**
 - Recebe o nome de um PDF da pasta data/pdfs/.
 - Extrai o texto usando pdf_processor.
 - Cria um prompt de resumo (SUMMARY_INSTRUCTION_MESSAGE + texto do PDF).
 - Usa token_utils para garantir que o prompt não exceda MAX_TOKENS_LIMIT, truncando o texto do PDF se necessário.
 - Chama api_service para gerar o resumo.
 - Salva o resumo usando session_manager e o define como active_api_summary_content para a sessão atual.
 - **/nova_sessao <nome_da_sessao> (handle_new_session()):** Cria uma nova sessão, limpando o histórico e o resumo ativo, e salvando o estado inicial.
 - **/carregar_sessao <nome_da_sessao> (handle_load_session()):** Carrega uma sessão existente, restaurando o histórico de chat e o resumo ativo.
 - **/listar_sesoes (handle_list_sessions()):** Lista todas as sessões salvas.
 - **/excluir_sessao <nome_da_sessao> (handle_delete_session()):** Exclui uma sessão específica.
 - **/listar_resumos (handle_list_summaries()):** Lista todos os resumos de PDF salvos.
 - **/carregar_resumo <id_do_resumo> (handle_load_summary()):** Carrega um resumo salvo e o ativa para a sessão atual.
 - **/excluir_resumo <id_do_resumo> (handle_delete_summary()):** Exclui um resumo de PDF.
 - **/exportar_chat <nome_do_export> (handle_export_chat()):** Exporta o histórico da sessão atual para um PDF usando pdf_exporter.

- **/listar_exports (handle_list_exports()):** Lista os PDFs de chat exportados.
 - **/carregar_export <nome_do_export> (handle_load_export()):** Tenta abrir um chat exportado (para visualização, não para carregar de volta no histórico). *Nota: O PDF não é "carregado" de volta na memória da conversa, apenas o caminho é exibido. Para interagir com ele, a IA precisaria "ler" o PDF novamente.*
 - **/excluir_export <nome_do_export> (handle_delete_export()):** Exclui um PDF de exportação.
 - **/limpar (handle_clear_context()):** Limpa o histórico de chat e o resumo ativo da sessão atual, mantendo a sessão.
 - **/sair (handle_exit()):** Encerra o chatbot.
5. **Loop Principal do Chatbot (run_chatbot()):**
- Carrega a sessão padrão ou cria uma se não existir.
 - Entra em um loop infinito que:
 - Pede a entrada do usuário.
 - Verifica se a entrada é um comando. Se for, chama o manipulador de comando apropriado.
 - Se não for um comando, prepara as mensagens para a API:
 - Inclui a SYSTEM_MESSAGE.
 - Se houver um active_api_summary_content, ele é adicionado ao contexto com a SUMMARY_INSTRUCTION_MESSAGE.
 - Adiciona as mensagens recentes do chat_history (limitado por HISTORY_MESSAGE_LIMIT).
 - Adiciona a pergunta atual do usuário.
 - Chama api_service.get_openai_completion para obter a resposta da IA.
 - Exibe a resposta e a adiciona ao chat_history.
 - Salva o estado da sessão após cada interação (save_session_state()).
 - Inclui tratamento de exceções para KeyboardInterrupt (Ctrl+C) e outros erros inesperados.

6. Como Executar o Chatbot

Após configurar o ambiente e colocar todos os arquivos nos seus respectivos lugares:

1. **Abra o Terminal/Prompt de Comando.**
2. **Navegue até o diretório raiz do seu projeto (chatbot_otimizado).**

Bash

```
cd /caminho/para/chatbot_otimizado
```

3. **Ative o Ambiente Virtual:**

- **Linux/macOS:**

Bash

```
source env/bin/activate
```

- **Windows (CMD):**
Bash
env\Scripts\activate.bat
- **Windows (PowerShell):**
PowerShell
.\env\Scripts\Activate.ps1

4. **Execute o main.py:**

Bash
python3 main.py

- **Dica:** Se você encontrar problemas de cache (ModuleNotFoundError ou comportamento inesperado após modificações), pode tentar executar com a flag -B:
Bash
python3 -B main.py

Isso impede que Python escreva arquivos .pyc (bytecode) e força a recarga do código-fonte.

7. Comandos do Chatbot

Uma vez que o chatbot esteja em execução, você pode interagir com ele usando os seguintes comandos (ou simplesmente digitando perguntas para a IA):

- /ajuda: Exibe a lista de todos os comandos disponíveis.
- /lerpdf <nome_do_arquivo.pdf>: Lê um arquivo PDF da pasta data/pdfs/, gera um resumo e o ativa como contexto para a conversa.
- /nova_sessao <nome_da_sessao>: Inicia uma nova sessão de chat com o nome especificado, limpando o histórico e o resumo ativo.
- /carregar_sessao <nome_da_sessao>: Carrega uma sessão de chat existente.
- /listar_sesoes: Lista todas as sessões de chat salvas.
- /excluir_sessao <nome_da_sessao>: Exclui uma sessão de chat e seu histórico.
- /listar_resumos: Lista todos os resumos de PDF salvos.
- /carregar_resumo <id_do_resumo>: Carrega um resumo de PDF salvo pelo seu ID, tornando-o o contexto ativo.
- /excluir_resumo <id_do_resumo>: Exclui um resumo de PDF salvo.
- /exportar_chat <nome_do_export>: Exporta o histórico da sessão atual para um arquivo PDF em data/exports/<session_name>/. O nome do arquivo será <nome_do_export>.pdf.
- /listar_exports: Lista todos os chats exportados para PDF (organizado por sessão).
- /carregar_export <nome_do_export>: Exibe o caminho para um chat exportado. (Note: Este comando *não* recarrega a conversa para o chatbot, apenas informa onde o PDF

está.)

- `/excluir_export <nome_do_export>`: Exclui um chat exportado para PDF.
- `/limpar`: Limpa o histórico de chat e o resumo ativo da sessão atual, mas mantém a sessão ativa.
- `/sair`: Encerra o chatbot.

8. Fluxo de Interação Exemplo

1. **Inicie o Chatbot:** `python3 main.py`
2. **Leia um PDF:** `> /lerpdf meu_documento.pdf`
 - (Certifique-se de que `meu_documento.pdf` está em `data/pdfs/`)
 - O chatbot irá processar o PDF e gerar um resumo.
3. **Faça Perguntas:** `> Qual o tópico principal do documento?`
 - A IA responderá com base no resumo do PDF.
4. **Crie uma Nova Sessão:** `> /nova_sessao ConcursosPublicos`
 - Isso limpará o contexto atual e iniciará uma nova conversa.
5. **Carregue a Sessão Anterior:** `> /carregar_sessao default_session`
 - O histórico e o resumo ativo da sessão padrão serão restaurados.
6. **Exporte a Conversa:** `> /exportar_chat minha_conversa_pdf`
 - Um PDF com o histórico do chat será criado em `data/exports/default_session/minha_conversa_pdf.pdf`.
7. **Saia do Chatbot:** `> /sair`

9. Solução de Problemas Comuns

- **ModuleNotFoundError:**
 - **Causa:** Python não consegue encontrar um módulo.
 - **Solução:** Verifique se o ambiente virtual está ativado e se todas as dependências em `requirements.txt` foram instaladas (`pip install -r requirements.txt`). Confirme se a estrutura de diretórios está correta e se os caminhos no `config.py` estão apontando para os locais certos.
- **TypeError: missing required positional arguments:**
 - **Causa:** Uma função está sendo chamada sem todos os argumentos necessários. Isso pode acontecer se houver uma incompatibilidade entre a definição da função e sua chamada.
 - **Solução:** Revise o `main.py` e os módulos `utils/` para garantir que as assinaturas das funções correspondam às chamadas (número e tipo de argumentos). Erros de cache de Python também podem causar isso (tente `python3 -B main.py`).
- **OpenAIError: max_tokens is too large ou context window exceeded:**
 - **Causa:** O texto de entrada (prompt + histórico + resumo) excede o limite de tokens do modelo da OpenAI.

- **Solução:** A lógica de truncamento em `handle_read_pdf` no `main.py` já lida com isso para resumos de PDF. Para conversas gerais, você pode ajustar `HISTORY_MESSAGE_LIMIT` no `config.py` para reduzir o histórico enviado, ou garantir que `MAX_TOKENS_LIMIT` esteja correto para o modelo escolhido.
- **FileNotFoundError:**
 - **Causa:** O chatbot não consegue encontrar um arquivo (PDF, sessão JSON, resumo JSON).
 - **Solução:** Verifique se o nome do arquivo está correto e se ele está no diretório esperado (ex: PDFs em `data/pdfs/`, sessões em `data/profiles/default/sessions/`). Confirme as permissões de arquivo/diretório.
- **Arquivo .json corrompido:**
 - **Causa:** O arquivo JSON de uma sessão ou resumo foi salvo incorretamente, causando erros de decodificação.
 - **Solução:** Pode ser necessário inspecionar o arquivo JSON manualmente para corrigir a sintaxe (se for um problema pequeno) ou, em casos mais graves, deletar o arquivo corrompido e iniciar uma nova sessão. A implementação atual do `session_manager` e `main.py` já inclui tratamento para salvar de forma robusta.
- **Comandos não reconhecidos:**
 - **Causa:** O comando digitado não corresponde aos comandos definidos ou há um erro de digitação.
 - **Solução:** Digite `/ajuda` para ver a lista de comandos corretos. Verifique a seção `COMMANDS` no `config.py` e a lógica de `handle_command` no `main.py`.

10. Próximos Passos e Potenciais Otimizações Futuras

Este chatbot é uma base sólida e funcional. Para futuras melhorias e para aproximá-lo de funcionalidades avançadas, você pode considerar:

- **Gerenciamento Avançado de Memória e Contexto:**
 - Implementar estratégias para "condensar" ou resumir partes mais antigas do `chat_history` para otimizar o uso de tokens e manter o contexto mais relevante para o modelo.
 - Explorar técnicas de Retrieval Augmented Generation (RAG) mais sofisticadas para buscar e injetar dinamicamente informações do PDF/resumo com base na consulta do usuário, em vez de enviar o resumo inteiro.
- **Engenharia de Prompt Refinada:**
 - Experimentar com diferentes `SYSTEM_MESSAGES` e `SUMMARY_INSTRUCTION_MESSAGES` para moldar o comportamento do modelo de forma mais precisa e conversacional.
- **Interface do Usuário (UI) Aprimorada:**
 - Considerar o desenvolvimento de uma interface gráfica (GUI) com bibliotecas como Tkinter, PyQt, ou uma interface web (Flask, Django) para uma experiência mais rica e amigável.

- **Adição de Novas Funcionalidades:**

- Implementar a capacidade de editar resumos.
 - Suporte a múltiplos resumos ativos ou a capacidade de mesclar informações de vários documentos.
 - Integração com outros tipos de documentos (TXT, DOCX, etc.).
 - Funcionalidades de anotação ou marcação em PDFs.
-