Visão Geral do Código

O código implementa um **assistente acadêmico baseado em IA com RAG (Retrieval-Augmented Generation)**.
Ele **não "sabe" tudo sozinho**: ele **consulta documentos oficiais da UFBA antes de responder**, reduzindo erros e invenções.

O fluxo geral é:

Pergunta do aluno → busca nos documentos → contexto + pergunta → modelo de linguagem → resposta fundamentada

---

ETAPA 1 — Definição do problema (conceitual, não é código ainda)

Antes do código existir, o projeto parte de um problema real:

- Estudantes da UFBA têm dificuldade de entender:
  - Regimento de Graduação
  - Resoluções de Atividades Complementares
  - Textos de orientação acadêmica
- Esses documentos são:
  - Longos
  - Técnicos
  - Espalhados em PDFs

Decisão técnica:
Usar IA **não para "inventar regras"**, mas para **explicar regras existentes**, com base nos documentos oficiais.

ETAPA 2 — Coleta e preparação dos documentos (base de conhecimento)

O que o código faz aqui:

1. Recebe **PDFs oficiais da UFBA**
2. Converte esses PDFs em **texto puro**
3. Organiza esse texto para ser usado pela IA

Por que isso é importante:

Modelos de linguagem **não leem PDF diretamente**.
Eles precisam de texto limpo para poder buscar informações.

Na apresentação, você pode dizer:

“Transformamos os documentos oficiais da UFBA em texto para que eles pudessem servir como base de conhecimento do sistema.”

ETAPA 3 — Divisão do texto em partes menores (chunking)

O que o código faz:

- Divide os textos grandes em **trechos menores** (ex: parágrafos ou blocos de tamanho fixo)

Por que isso é necessário:

- Modelos têm limite de contexto (tokens)
- Buscar em um texto gigante seria impreciso
- O RAG funciona melhor com pedaços menores e bem definidos

Explicação simples:

“Em vez de entregar um documento inteiro para a IA, a gente divide o conteúdo em pequenos trechos para facilitar a busca.”

ETAPA 4 — Criação dos embeddings (representação semântica)

Aqui entra uma parte mais técnica (mas explicável):

O código:

- Transforma cada trecho de texto em um **vetor numérico** (embedding)
- Esses vetores representam o **significado** do texto, não só palavras

Por que isso é importante:

- Permite buscar por **sentido**, não só por palavras iguais
- Exemplo:
    - Pergunta: “Quantas horas de atividade complementar preciso?”
    - Documento pode usar termos diferentes, mas o sistema entende que o tema é o mesmo

Na apresentação:

“Usamos embeddings para que o sistema consiga entender o significado das perguntas e dos documentos, mesmo quando as palavras não são exatamente iguais.”

ETAPA 5 — Criação do mecanismo de busca (RAG)

Essa é a **parte central do projeto**.

O que o código faz:

1. Recebe a pergunta do usuário
2. Converte a pergunta em embedding
3. Compara com os embeddings dos documentos
4. Seleciona os **trechos mais relevantes**
5. Usa esses trechos como **contexto confiável**

Importante dizer:

"O modelo não responde direto. Antes, ele busca nos documentos oficiais da UFBA os trechos mais relevantes."

Isso é o **RAG (Retrieval-Augmented Generation)**:

- Retrieval → busca
- Generation → geração da resposta

ETAPA 6 — Escolha do modelo (GPT)

Decisão registrada no projeto:

- O GPT foi escolhido porque:
  - Entende melhor textos institucionais
  - Não exige fine-tuning pesado
  - Funciona bem com RAG + prompt

Como o código usa o GPT:

- Envia:
  - A pergunta do aluno
  - Os trechos recuperados dos documentos
- Pede:
  - Uma resposta clara
  - Baseada **apenas no contexto fornecido**

Na apresentação:

"O GPT atua como um intérprete dos documentos da UFBA, não como fonte de regras."

ETAPA 7 — Construção do prompt (engenharia de prompt)

O código monta um prompt mais ou menos assim (em lógica, não literal):

"Com base **exclusivamente** nos textos abaixo, responda à pergunta do estudante de forma clara e objetiva."

Por que isso importa:

- Reduz alucinação
- Força fidelidade ao regimento
- Mantém responsabilidade institucional

Conecta com ética e limites da IA, ponto forte do seu projeto.

ETAPA 8 — Geração da resposta final

O modelo:

- Lê os trechos
- Interpreta a regra
- Produz uma resposta em linguagem acessível ao aluno

Importante:

- Se o documento fala em **SIAC**, a resposta mantém isso
- O projeto **não "corrige" historicamente o texto**
- A explicação disso aparece na apresentação

ETAPA 9 — Avaliação com perguntas em JSON

O projeto inclui:

- 20 perguntas e respostas de referência
- Arquivo JSON para avaliação

O que o código (ou o processo) faz:

- Compara:
  - Resposta da IA
  - Resposta esperada
- Avalia:
  - Correção
  - Clareza

- Fidelidade ao documento

Avaliação humana (como o professor pediu):

“A IA não acerta tudo, e isso é intencionalmente analisado.”

ETAPA 10 — Limitações assumidas no código e no projeto

Isso é MUITO importante na apresentação:

- Não substitui a coordenação de curso
- Depende de documentos atualizados
- Pode errar se a pergunta sair do escopo dos textos
- Serve como apoio inicial ao estudante

Mostra maturidade técnica e ética.

COMO VOCÊ PODE RESUMIR TUDO EM UMA FRASE NA APRESENTAÇÃO

“O código implementa um sistema de RAG que busca informações em documentos oficiais da UFBA e usa um modelo de linguagem para explicar essas regras de forma acessível aos estudantes, mantendo fidelidade às fontes.”