

PROJETO

FarmaChat

Informações do Projeto

Disciplina	Análise e Projeto de Sistemas
Professor	Juan Apolinario
Nome do projeto	FarmaChat
Equipe	Ana Beatriz Cordeiro de Oliveira Belmiro Nino Xavier Simas Raíssa Cunha de Medeiros Pinheiro Vinicius Souza Rodrigues

Documentação do Sistema

1. Visão Geral

Nome do Sistema: FarmaChat (provisório)

Objetivo: Permitir que profissionais da saúde e pacientes tirem dúvidas sobre medicamentos de forma rápida e acessível, utilizando integração com a OpenAI para fornecer respostas confiáveis.

Público-alvo: Profissionais da saúde e pacientes, especialmente idosos, que utilizam diversos medicamentos e preferem comunicação por áudio.

Plataformas: Web

Acesso: O sistema será de uso aberto, sem necessidade de login ou captação de dados do usuário.

2. Requisitos do Sistema

2.1 Requisitos Funcionais

1. O sistema deve permitir que o usuário envie perguntas via texto e áudio (interpretando o áudio através de Speech-to-Text).
2. O sistema deve processar a pergunta e buscar respostas relevantes sobre medicamentos através da OpenAI.
3. O sistema deve exibir a resposta em formato de texto e oferecer opção de resposta em áudio (convertendo por Text-to-Speech). O sistema deve utilizar uma API de Speech-to-Text para converter áudio em texto.
4. O sistema deve ter uma interface intuitiva e acessível para idosos, com botões ajustáveis (usuário escolher tamanho dos ícones e fonte) e letras legíveis.
5. O sistema deve oferecer sugestões de perguntas para facilitar o uso.

2.2 Requisitos Não Funcionais

1. **Agilidade de Resposta:** O sistema deve responder às perguntas em menos de 5 segundos.
2. **Layout Responsivo:** O layout deve ser responsivo para diferentes tamanhos de tela.
3. **Disseminação de informações e ajuda na adesão de terapias:** Explicar efeitos colaterais, interações medicamentosas e dosagens. Auxiliar na busca de medicamentos para condições específicas.

3. Backlog do Produto

3.1 Épicos

- Botão de aceite de termo

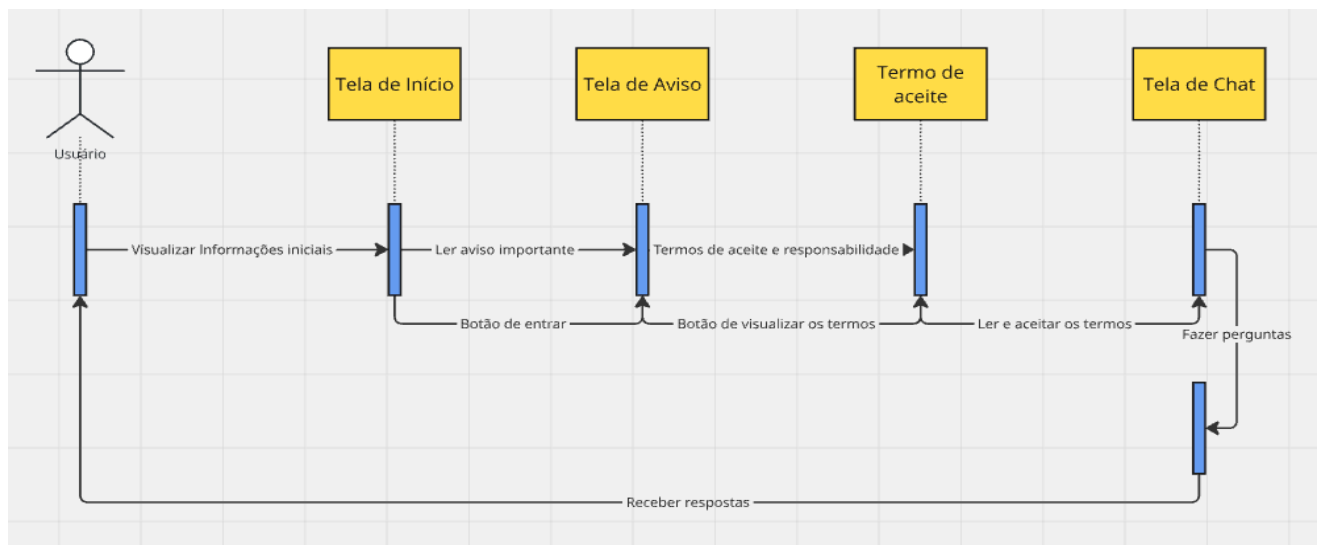
- Captura de perguntas via texto e áudio
- Processamento e geração de respostas
- Interface acessível e intuitiva

3.2 Histórias de Usuário

ID	História de Usuário	Critérios de Aceitação
US01	Como paciente, quero perguntar sobre meu medicamento usando áudio para que eu não precise digitar.	O sistema deve permitir gravação e converter áudio em texto.
US02	Como profissional da saúde, quero receber rapidamente informações sobre princípios ativos, para auxiliar no atendimento ao paciente.	O sistema deve exibir resposta textual e estruturada.

4. Casos de Uso

4.1 Diagrama de Casos de Uso



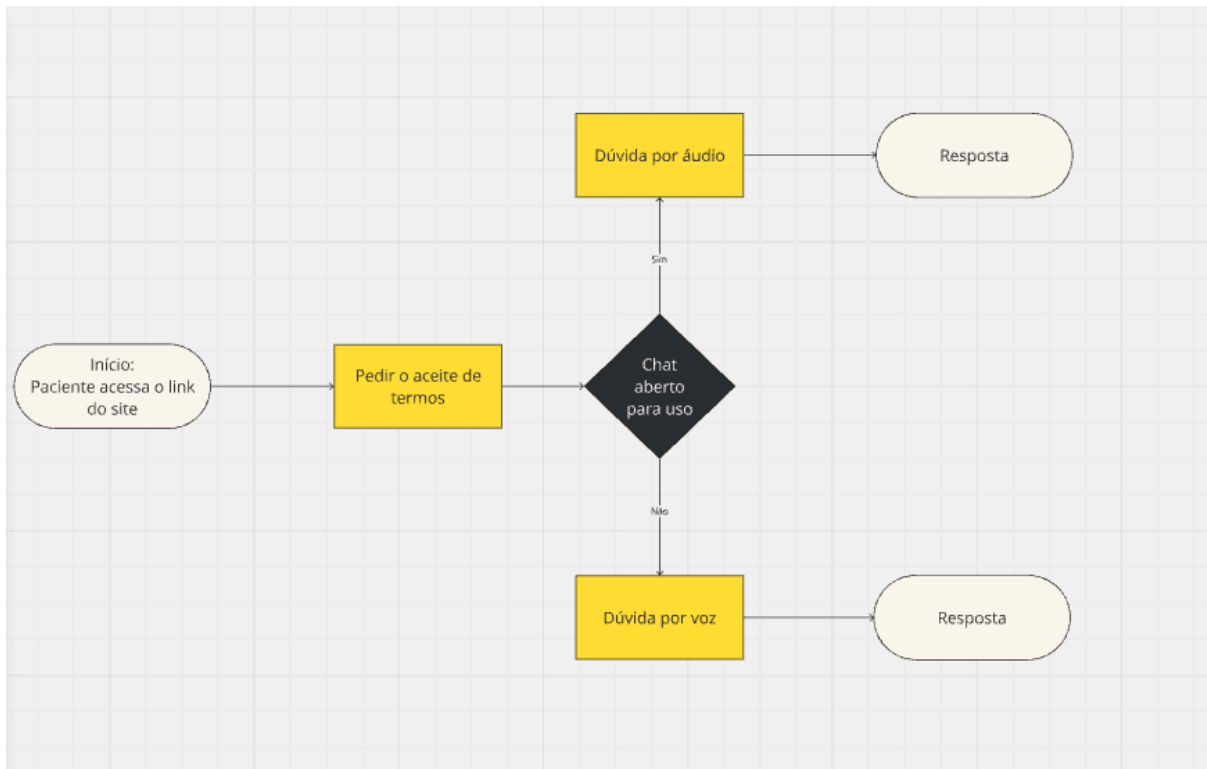


Diagrama de uso

4.2 Descrição dos Casos de Uso

Caso de Uso	Descrição	Ator	Fluxo Principal
Perguntar sobre medicamento	Usuário faz uma pergunta via texto ou áudio.	Paciente ou profissional da saúde	Usuário faz pergunta -> Sistema processa e gera resposta -> Resposta é exibida ou reproduzida em áudio.

5. Arquitetura do Sistema

5.1 Diagrama de Arquitetura

(Aqui pode-se inserir um diagrama representando a arquitetura do sistema)

5.2 Tecnologias Utilizadas

- Front-end: HTML, CSS, JavaScript e React
- Back-end: Node.js, Python (para integração com a OpenAI)
- Banco de Dados: Não será utilizado no MVP
- APIs: OpenAI, Speech-to-Text, Text-to-Speech

5.3 Integração com a API

POST <https://api.openai.com/v1/completions> : body {"message": "msg aq" }

POST <http://localhost:8000/text-to-speech> :body {???} , return {" " }

Descrever todos os endpoints e retornos que a API irá fornecer, usar APIs de modelos como o GPT-4 ou outros modelos de IA generativa.

- Criar um ponto de interação onde o usuário possa fazer perguntas sobre medicamentos (como um chatbot ou campo de pesquisa), que será processado pelo modelo de IA, e retornar uma resposta relevante.

Exemplo de integração com GPT-4 (Python):

```
import openai

openai.api_key = "sua-chave-de-API"

def consultar_medicamento(pergunta):
    response = openai.Completion.create(
        engine="text-davinci-003", # Ou outro modelo da OpenAI
        prompt=pergunta,
        max_tokens=150
    )
    return response.choices[0].text.strip()

resultado = consultar_medicamento("Qual o efeito colateral do medicamento Paracetamol?")
print(resultado)
```

A integração é feita com o endpoint da **API da OpenAI**, onde você envia uma solicitação POST contendo o texto de entrada e parâmetros específicos (como o modelo a ser usado, temperatura, etc.), e recebe a resposta com o texto gerado.

Para identificar perguntas sobre medicamentos ou processar áudios com um modelo GPT, o número de endpoints vai depender de como você planeja configurar a integração. Vou dividir em dois cenários: um para **perguntas sobre medicamentos** e outro para **áudios**. Nesse caso, você pode usar um único endpoint para processar as perguntas de texto. A integração com a API da OpenAI para processamento de perguntas relacionadas a medicamentos seria uma solicitação **POST** para o endpoint de **completions**. O endpoint da OpenAI para isso é algo como:

Endpoint para processamento de texto (GPT-4/3.5):

POST <https://api.openai.com/v1/completions>

- **Fluxo:**
 - Você envia um **prompt** com a pergunta sobre medicamentos.
 - O chatbot gera uma resposta baseada no modelo treinado.

- Se você deseja filtrar ou obter informações específicas sobre medicamentos, pode ajustar os parâmetros do prompt para deixar claro que o foco é nesse tipo de questão, por exemplo, "Responda a pergunta sobre medicamentos com base em evidências".

Parâmetros úteis:

- **model:** "gpt-4" ou "gpt-3.5-turbo"
- **prompt:** A pergunta sobre o medicamento, por exemplo, "Quais são os efeitos colaterais do Paracetamol?"
- **max_tokens:** Define o tamanho máximo da resposta.
- **temperature:** Controla a aleatoriedade da resposta (um valor de 0 a 1).

Se você precisar de uma API adicional para consultas específicas sobre medicamentos (como dosagens, interações, etc.), poderá usar outro endpoint de **informações especializadas** se o serviço da OpenAI ou de terceiros oferecer esse tipo de recurso. Mas, geralmente, um único endpoint para completar o prompt seria suficiente.

6. Modelagem do Banco de Dados

Não será necessário armazenar informações no MVP.

7. Fluxo de Telas (Wireframes)

Figma

8. Plano de Testes

8.1 Testes Funcionais

ID	Descrição	Entrada	Saída Esperada
T01	Testar envio de pergunta por texto	Texto digitado pelo usuário	Resposta gerada e exibida em texto
T02	Testar envio de pergunta por áudio	Áudio gravado pelo usuário	Sistema converte áudio para texto e exibe resposta
T03	Testar reprodução de resposta em áudio	Texto da resposta	Sistema converte texto para áudio e reproduz

9. Cronograma

Atividade	Responsável	Prazo
-----------	-------------	-------

Definição de Requisitos	[Nome]	[Data]
-------------------------	--------	--------

Desenvolvimento	[Nome]	[Data]
-----------------	--------	--------

Testes	[Nome]	[Data]
--------	--------	--------

Implantação	[Nome]	[Data]
-------------	--------	--------

10. Considerações Finais

O sistema será um MVP funcional para testar a adesão do público-alvo, garantindo acessibilidade e respostas rápidas. Futuras melhorias podem incluir personalização das respostas e autenticação de usuários.

Pergunta: Como integrar uma IA generativa a um programa de consulta de medicamentos?

Resposta:

Integrar uma IA generativa a um programa de consulta de medicamentos pode ser uma excelente maneira de melhorar a experiência do usuário, fornecer informações personalizadas e até mesmo otimizar o processo de pesquisa de medicamentos com base em perguntas específicas. Aqui estão os passos gerais para fazer essa integração:

1. Definir o Objetivo e a Funcionalidade da IA

Primeiro, determine quais funcionalidades você espera que a IA desempenhe. Exemplos incluem:

- Responder perguntas sobre medicamentos.
- Sugerir medicamentos com base em sintomas ou condições de saúde.
- Explicar efeitos colaterais, interações medicamentosas e dosagens.
- Auxiliar na busca de medicamentos para condições específicas.

2. Escolher a Tecnologia de IA Generativa

A IA generativa pode ser implementada de várias formas, dependendo do tipo de interação desejada:

- **Modelos de Linguagem:** Modelos como GPT-4 ou GPT-3 podem ser usados para responder a perguntas complexas e gerar informações detalhadas de maneira conversacional.
- **Modelos Específicos para Medicina:** Existem IAs treinadas especificamente para fornecer informações médicas, como o BioGPT ou modelos de NLP (Processamento de Linguagem Natural) especializados em saúde.

3. Fontes de Dados para Treinamento ou Consulta

Se você optar por criar um modelo personalizado ou usar um pré-existente, é importante garantir que a IA tenha acesso a fontes de dados confiáveis. Isso inclui:

- **Bases de dados médicas:** Como PubMed, FDA, ANVISA ou outras fontes de informação sobre medicamentos e tratamentos.
- **Listas de medicamentos:** Como a Bula dos medicamentos (se disponível em formato acessível via API).
- **Documentos e Manuais médicos:** Que podem ser usados para treinar ou alimentar a IA.

4. Escolher a Plataforma e a Linguagem de Programação

A escolha da plataforma e linguagem depende de como você deseja integrar a IA. Algumas opções incluem:

- **Backend em Python:** Usando bibliotecas como [OpenAI](#) (para GPT), [Transformers](#) (para modelos de NLP) ou APIs como [Dialogflow](#) para integrar o chatbot de IA.
- **Plataforma Web:** Para integração em um site ou aplicativo, você pode usar frameworks como Flask ou Django (em Python) para construir o backend, e JavaScript ou React no frontend.

5. Desenvolver a Integração de IA ao Sistema

Dependendo da sua necessidade, existem duas formas principais de integrar a IA ao seu sistema de consulta de medicamentos:

A. Integração via API:

- Usar APIs de modelos como o GPT-4 ou outros modelos de IA generativa.
- Criar um ponto de interação onde o usuário possa fazer perguntas sobre medicamentos (como um chatbot ou campo de pesquisa), que será processado pelo modelo de IA, e retornar uma resposta relevante.

Exemplo de integração com GPT-4 (Python):

```
import openai

openai.api_key = "sua-chave-de-API"

def consultar_medicamento(pergunta):
    response = openai.Completion.create(
        engine="text-davinci-003", # Ou outro modelo da OpenAI
        prompt=pergunta,
        max_tokens=150
    )
    return response.choices[0].text.strip()

resultado = consultar_medicamento("Qual o efeito colateral do medicamento Paracetamol?")
print(resultado)
```

B. Integração com Banco de Dados de Medicamentos:

- Se seu sistema tiver um banco de dados de medicamentos, você pode usar consultas SQL para buscar informações sobre medicamentos e então passar esses dados para o modelo de IA gerar respostas mais específicas.

Exemplo de integração com banco de dados SQL (Python + SQLite):

```
import sqlite3

def obter_informacao_medicamento(nome):
    conn = sqlite3.connect('medicamentos.db')
    cursor = conn.cursor()
    cursor.execute("SELECT descricao FROM medicamentos WHERE nome = ?", (nome,))
    resultado = cursor.fetchone()
    conn.close()
    return resultado[0] if resultado else "Medicamento não encontrado."

# Exemplo de consulta
medicamento = "Paracetamol"
info = obter_informacao_medicamento(medicamento)
print(info)
```

6. Treinamento (Caso necessário)

Se você decidir criar uma IA personalizada, o modelo pode precisar de treinamento específico. Isso envolve fornecer dados sobre medicamentos, condições, efeitos colaterais, interações, etc., e ensinar o modelo a gerar respostas precisas para essas consultas. Isso pode ser feito com técnicas de aprendizado supervisionado ou fine-tuning de um modelo pré-existente.

7. Testar e Melhorar o Sistema

Teste a integração com um grupo de usuários para verificar se a IA está respondendo de maneira adequada e precisa. Isso inclui garantir que as informações sobre medicamentos sejam corretas e que a IA forneça respostas confiáveis.

8. Considerar Aspectos Legais e Éticos

Como se trata de um sistema de consulta de medicamentos, é essencial garantir que as informações fornecidas pela IA estejam dentro dos regulamentos de saúde. Dependendo do país, isso pode incluir conformidade com normas como HIPAA (nos EUA) ou GDPR (na Europa), além de garantir que os dados médicos sejam tratados de forma segura e confidencial.

Ferramentas que podem ser úteis:

- **OpenAI API** (para GPT-4 e modelos generativos)
- **Dialogflow** (para criar assistentes conversacionais)
- **MedlinePlus, PubMed, API da FDA** (para fontes confiáveis de informações sobre medicamentos)
- **Pandas e SQLAlchemy** (para gerenciar e acessar bancos de dados de medicamentos)

Com esses passos, você poderá criar uma integração eficaz de IA generativa em um sistema de consulta de medicamentos, proporcionando respostas úteis e personalizadas aos usuários. Se precisar de ajuda com alguma parte específica do processo, estou à disposição!

Sim, integrar uma IA geralmente é feito por meio de endpoints, especialmente quando você está utilizando APIs. Quando você conecta uma aplicação a um modelo de IA como o GPT, você interage com a IA por meio de uma **API** que expõe endpoints. Esses endpoints são URLs que você acessa para enviar dados para a IA (geralmente no formato JSON) e, em troca, a IA responde com os resultados.

No caso do GPT, por exemplo, a integração é feita com o endpoint da **API da OpenAI**, onde você envia uma solicitação POST contendo o texto de entrada e parâmetros específicos (como o modelo a ser usado, temperatura, etc.), e recebe a resposta com o texto gerado.

Se você estiver trabalhando com outra IA, como um modelo de visão computacional ou reconhecimento de voz, o princípio de integração via endpoints também pode ser seguido. A comunicação se dá por chamadas HTTP, onde você manda dados e recebe os resultados.

Quer um exemplo prático de como isso funciona ou precisa de mais detalhes sobre a implementação?

Dito por si:

quantos endpoints serão utilizados GPT? para identificar perguntas sobre medicamento ou áudios.

Dito pelo ChatGPT:

```
/* Exercício de Fixação

Cenário: Um aluno entra no sistema acadêmico para:

• Ver suas notas

• Atualizar seus dados

• Fazer matrícula

Tarefa:

1. Identifique os atores.

2. Liste os casos de uso.

3. Desenhe a classe "Aluno" com atributos e métodos.

4. Represente a sequência do processo "Ver notas". */

1. Aluno

2. [Aluno] ----> (ver nota)

           ----> (Atualizar seus dados)

           ----> (fazer matrícula)

3. classe: Aluno

+ nome: String
```

```
+ CPF: String
- notas: Double
-
```