# 🤖 LocalAI Assistant ### Premium AI Assistant with Local LLM Support [![License: MIT](https://img.shields.io/badge/License-MIT-blue.svg)](https://opensource.org/licenses/MIT) [![Python](https://img.shields.io/badge/Python-3.11+-green.svg)](https://python.org) [![React](https://img.shields.io/badge/React-19-61DAFB.svg)](https://reactjs.org) [![TypeScript](https://img.shields.io/badge/TypeScript-5.3-3178C6.svg)](https://typescriptlang.org) [![FastAPI](https://img.shields.io/badge/FastAPI-0.109-009688.svg)](https://fastapi.tiangolo.com) [![TailwindCSS](https://img.shields.io/badge/TailwindCSS-3.4-38B2AC.svg)](https://tailwindcss.com) [![Docker](https://img.shields.io/badge/Docker-Ready-2496ED.svg)](https://docker.com)

**Um assistente de IA moderno e de nível enterprise com interface premium e funcionalidades poderosas.**

[Funcionalidades](#-funcionalidades) • [Screenshots](#-screenshots) • [Instalação](#-instalação-passo-a-passo) • [Como Usar](#-como-usar) • [API](#-documentação-da-api) • [Troubleshooting](#-troubleshooting)

---

## ✨ Funcionalidades

### 🎨 Interface Premium

- **Dashboard Moderno** - Analytics bonitos com estatísticas em tempo real
- **Interface estilo ChatGPT** - Experiência de chat familiar e intuitiva
- **Dark/Light Mode** - Alternância elegante de temas
- **Design Glassmorphism** - Tendências modernas de design $^{2024}/_{2025}$
- **Animações Suaves** - Interações com Framer Motion
- **Design Responsivo** - Mobile-first, funciona em todos os dispositivos

### 🤖 Capacidades de IA

- **Suporte a LLM Local** - Execute modelos localmente com Ollama
- **Múltiplos Modelos** - Alterne entre diferentes modelos de IA

- **Streaming de Respostas** - Streaming de tokens em tempo real
- **Parâmetros Customizáveis** - Controles de Temperature, Top-P, Top-K
- **System Prompts** - Templates de prompts pré-construídos e customizados

### 💬 Funcionalidades do Chat

- **Gerenciamento de Conversas** - Criar, editar, deletar conversas
- **Histórico de Mensagens** - Persistência completa das conversas
- **Syntax Highlighting** - Destaque de código para 100+ linguagens
- **Renderização Markdown** - Formatação de texto rica
- **Copiar para Clipboard** - Cópia de código com um clique

### 📊 Dashboard de Analytics

- **Estatísticas de Uso** - Acompanhe conversas, mensagens, tokens
- **Uso de Modelos** - Veja quais modelos você mais usa
- **Métricas de Performance** - Analytics de tempo de resposta
- **Gráficos de Atividade** - Padrões visuais de uso

---

## 📸 Screenshots

### Dashboard ![Dashboard](/home/ubuntu/localai-assistant/screenshots/dashboard.jpg) *Dashboard premium de analytics com estatísticas em tempo real* ### Interface de Chat ![Chat](/home/ubuntu/localai-assistant/screenshots/chat.jpg) *Interface de chat moderna com streaming de respostas* ### Configurações ![Settings](/home/ubuntu/localai-assistant/screenshots/settings.jpg) *Configurações completas com configuração de modelos*

---

# 🚀 Instalação Passo a Passo

## Pré-requisitos

Antes de começar, você precisa ter instalado:

| Software | Versão | Link para Download |
|---|---|---|
| **Python** | 3.11 ou superior | python.org/downloads |
| **Node.js** | 20 ou superior | nodejs.org |
| **Ollama** | Última versão | ollama.ai |
| **Git** | Qualquer versão | git-scm.com |
| **Docker** | (Opcional) | docker.com |

---

## 📥 Passo 1: Instalar o Ollama

O Ollama é o software que roda os modelos de IA no seu computador.

**Windows:**

1. Acesse ollama.ai
2. Clique em “Download for Windows”
3. Execute o instalador e siga as instruções
4. Após instalar, o Ollama iniciará automaticamente

**macOS:**

```
# Via Homebrew
brew install ollama

# Ou baixe diretamente de ollama.ai
```

**Linux:**

```
curl -fsSL https://ollama.ai/install.sh | sh
```

---

## 📥 Passo 2: Baixar um Modelo de IA

Abra o terminal/prompt de comando e execute:

```
# Modelo recomendado (leve e rápido)
ollama pull dolphin-mistral

# Modelo para código (opcional)
ollama pull codellama

# Modelo sem censura (opcional)
ollama pull wizardlm-uncensored
```

> ***Nota:*** *O download pode demorar alguns minutos dependendo da sua internet. Os modelos têm entre 4GB e 8GB.*

---

## 📥 Passo 3: Clonar o Repositório

```
# Clone o projeto
git clone https://github.com/lucasandre16112000-png/localai-assistant.git

# Entre na pasta do projeto
cd localai-assistant
```

---

## 📥 Passo 4: Configurar Variáveis de Ambiente

```
# Copie o arquivo de exemplo
cp .env.example .env
```

Edite o arquivo `.env` se necessário (os valores padrão funcionam para a maioria dos casos).

---

## 📥 Passo 5: Iniciar o Backend

### Opção A - Com Docker (Recomendado):

```
docker-compose up -d
```

### Opção B - Manualmente:

Abra um terminal e execute:

```
# Entre na pasta do backend
cd backend

# Crie um ambiente virtual
python -m venv venv

# Ative o ambiente virtual
# Windows:
venv\Scripts\activate
# Linux/macOS:
source venv/bin/activate

# Instale as dependências
pip install -r requirements.txt

# Inicie o servidor
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

Você verá uma mensagem como:

```
INFO:     Uvicorn running on http://0.0.0.0:8000
```

---

### 📥 Passo 6: Iniciar o Frontend

Abra **outro terminal** (mantenha o backend rodando) e execute:

```
# Entre na pasta do frontend
cd frontend

# Instale as dependências
npm install
# ou se preferir pnpm:
pnpm install

# Inicie o servidor de desenvolvimento
npm run dev
# ou:
pnpm dev
```

Você verá uma mensagem como:

```
VITE v5.x.x  ready in xxx ms
➜  Local:   http://localhost:3000/
```

---

### 📥 Passo 7: Acessar a Aplicação

1. Abra seu navegador
2. Acesse: **http://localhost:3000**
3. Pronto! Você verá a interface do LocalAI Assistant

---

## 🎮 Como Usar

### Iniciando uma Conversa

1. Clique em **"+ New Chat"** na sidebar
2. Digite sua pergunta no campo de texto

3. Pressione **Enter** ou clique em **Send**
4. Aguarde a resposta da IA (aparece em tempo real!)

## Exemplos de Perguntas

```
- "Explique como funciona o algoritmo QuickSort em Python"
- "Escreva uma função para calcular o fatorial de um número"
- "Crie um componente React para um formulário de login"
- "Me ajude a debugar este código: [cole seu código]"
```

## Usando o Dashboard

1. Clique em **“Dashboard”** na sidebar
2. Veja estatísticas de uso:
    - Total de conversas
    - Mensagens enviadas
    - Tokens utilizados
    - Tempo médio de resposta

## Configurando o Modelo

1. Clique em **“Settings”** na sidebar
2. Vá em **“Models”**
3. Selecione o modelo desejado
4. Ajuste parâmetros como temperatura (criatividade)

---

# 📚 Documentação da API

## Endpoints Principais

| Método | Endpoint | Descrição |
|---|---|---|
| `GET` | `/api/v1/conversations` | Lista todas as conversas |
| `POST` | `/api/v1/conversations` | Cria nova conversa |
| `GET` | `/api/v1/conversations/{uuid}` | Obtém conversa com mensagens |
| `DELETE` | `/api/v1/conversations/{uuid}` | Deleta conversa |
| `POST` | `/api/v1/chat/completions` | Envia mensagem e obtém resposta |
| `GET` | `/api/v1/models` | Lista modelos disponíveis |

## Documentação Interativa

Quando o backend estiver rodando, acesse:

- **Swagger UI**: http://localhost:8000/docs
- **ReDoc**: http://localhost:8000/redoc

---

# 🔧 Configuração

## Variáveis de Ambiente

| Variável | Descrição | Padrão |
|---|---|---|
| `OLLAMA_BASE_URL` | URL da API do Ollama | `http://localhost:11434` |
| `DEFAULT_MODEL` | Modelo LLM padrão | `dolphin-mistral` |
| `DEFAULT_TEMPERATURE` | Temperatura de sampling | `0.7` |
| `DATABASE_URL` | Conexão do banco de dados | `sqlite:///./localai.db` |
| `DEBUG` | Ativar modo debug | `false` |

---

# ❓ Troubleshooting

## Problema: “Ollama não está respondendo”

**Solução:**

1. Verifique se o Ollama está rodando:

```
ollama list
```

2. Se não estiver, inicie-o:

```
ollama serve
```

## Problema: “Modelo não encontrado”

**Solução:**

```
# Baixe o modelo
ollama pull dolphin-mistral
```

### Problema: “Erro de conexão com o backend”

**Solução:**

1. Verifique se o backend está rodando na porta 8000
2. Acesse http://localhost:8000/health para verificar

### Problema: “npm/pnpm não encontrado”

**Solução:**

1. Instale o Node.js de nodejs.org
2. Reinicie o terminal após a instalação

### Problema: “Python não encontrado”

**Solução:**

1. Instale o Python de python.org
2. Marque a opção “Add Python to PATH” durante a instalação
3. Reinicie o terminal

---

## 🏗️ Estrutura do Projeto

```
localai-assistant/
├── backend/                 # Servidor FastAPI
│   ├── app/
│   │   ├── main.py          # Aplicação FastAPI
│   │   ├── routers/         # Endpoints da API
│   │   ├── services/        # Lógica de negócio
│   │   ├── models/          # Modelos do banco de dados
│   │   ├── schemas/         # Schemas Pydantic
│   │   └── core/            # Configurações
│   ├── tests/               # Testes
│   └── requirements.txt     # Dependências Python
├── frontend/                # Aplicação React
│   ├── src/
│   │   ├── components/      # Componentes React
│   │   ├── lib/             # Utilitários & API
│   │   ├── styles/          # Estilos globais
│   │   └── App.tsx          # Componente principal
│   └── package.json         # Dependências Node.js
├── screenshots/             # Screenshots do projeto
├── docker-compose.yml       # Configuração Docker
├── .env.example             # Exemplo de variáveis de ambiente
└── README.md                # Este arquivo
```

---

## 🤝 Contribuindo

Contribuições são bem-vindas! Sinta-se à vontade para enviar um Pull Request.

1. Fork o repositório
2. Crie sua branch de feature (`git checkout -b feature/NovaFuncionalidade`)
3. Commit suas mudanças (`git commit -m 'Adiciona nova funcionalidade'`)
4. Push para a branch (`git push origin feature/NovaFuncionalidade`)
5. Abra um Pull Request

---

## 📄 Licença

Este projeto está licenciado sob a Licença MIT - veja o arquivo LICENSE para detalhes.

---

## 👨‍💻 Autor

**Lucas Andre S**

- GitHub: @lucasandre16112000-png

---

### ⭐ Dê uma estrela neste repo se você achou útil! Feito com ❤️ por Lucas Andre S