Projeto Keila – Assistente Pessoal com IA

**Especificação Técnica – Keila**

Keila é um dispositivo físico baseado em um Raspberry Pi 3 B+, utilizando Python, que atua como uma assistente virtual inteligente. O sistema recebe comandos de voz, converte para texto (STT), envia à API da OpenAI (ChatGPT), recebe a resposta em texto e converte novamente em áudio (TTS) para resposta falada. O código-fonte estará sempre em inglês.

Todas as interações são exibidas em uma tela sensível ao toque, que também serve como interface de configuração local. O sistema pode ser configurado tanto localmente via tela embarcada quanto remotamente via navegador. Ambas são formas válidas de interagir com a API da Keila.

A tela capacitiva deve exibir um avatar da Keila, com expressões faciais simples. Essa tela é utilizada para mostrar o avatar e informações quando necessário, além de permitir o acesso às configurações como alternativa ao navegador. Deve existir um botão físico (ou acessível na tela) que inicie uma ACTION pré-definida, como a VoicePrompt.

Toda a interação com a Keila deve ocorrer via API, permitindo controle 100% via tela (menus) ou via cliente web fornecido pela própria Keila, com acesso completo aos recursos.

**Estrutura e Funcionamento**

A Keila poderá ter diferentes ACTIONS, que são capacidades/funcionalidades adicionáveis no futuro. No momento, a única ACTION é a VoicePrompt, que engloba todo o processo de gravar áudio, enviar para o GPT e tocar a resposta. No futuro, poderão ser criadas outras ACTIONS, como Tocar Música, entre outras.

Existe um estado principal que define o fluxo da aplicação: INIT, CANCELING, READY ou RUNNING. Quando uma ACTION está acontecendo, o estado global é RUNNING. Existe um endpoint capaz de cancelar todas as ações em execução.

Isso é implementado com um Thread Manager, que possui métodos como cancelar todas as ações e iniciar threads. Cada ACTION roda em sua própria thread e pode iniciar novas threads, mas o fluxo principal deve estar sempre pronto para ser interrompido por um comando do usuário.

Há um fluxo contínuo rodando no thread principal, com o objetivo de manter a Keila bem responsiva. Isso significa que, enquanto ela executa tarefas como buscar informações na internet, tocar ou gravar áudio, ela deve conseguir interromper o que está fazendo e reiniciar imediatamente o processo de escuta de áudio ao receber um comando do usuário.

Esse comportamento envolve programação de escuta contínua (non-stop), com reconhecimento de silêncio (percepção de que o usuário parou ou terminou de falar) e detecção de comandos básicos, como “Oi Keila”, entre outros.

**Requisitos Técnicos**

Todo o desenvolvimento deve ser realizado em uma máquina Windows, garantindo o funcionamento de todos os recursos da Keila, como gravação e reprodução de áudio. O sistema deve ser capaz de tocar o áudio tanto no Raspberry quanto no Windows, lidando corretamente com as diferenças entre os ambientes. Para isso, devem ser usadas bibliotecas compatíveis com ambos.

Deve existir um arquivo de configuração chamado keila\_config.ini na pasta onde o programa é executado.

Por fim, a Keila deve contar com um sensor de temperatura, que permita medir e informar a temperatura ao usuário.

**Prompt mínimo:**

Estou desenvolvendo a **Keila**, um dispositivo físico baseado em **Raspberry Pi 3 B+**, escrito em **Python**, que funciona como uma assistente virtual inteligente. A Keila usa STT (Speech to Text) para transcrição de voz, envia comandos para a API da OpenAI (ChatGPT), recebe a resposta em texto, converte em áudio (TTS) e exibe informações em uma **tela sensível ao toque** com avatar. O sistema pode ser configurado via tela embarcada ou navegador. Possui ações como **VoicePrompt**, gerenciamento de threads para escuta contínua e possibilidade de interrupção imediata, além de um **sensor de temperatura**. O desenvolvimento ocorre em **Windows**, com compatibilidade para Raspberry e bibliotecas portáveis.

# Funcionalidades

* • Reconhecimento de voz
* • Integração com OpenAI (GPT-4/4o)
* • Resposta por voz (TTS)
* • Interface touchscreen embarcada
* • Painel de administração via navegador
* • Web API local para controle e configuração
* • LEDs indicadores
* • Sensor de temperatura ambiente
* • Inicialização automática ao ligar

# Hardware Utilizado

* • Raspberry Pi 3 B+: Computador principal
* • Tela Capacitiva HDMI + USB (5~7"): Interface visual com toque
* • Fonte 5V 3A (ou 4~5A): Alimentação do sistema
* • Microfone USB ou I2S: Captação de áudio
* • Caixa de som P2 ou Bluetooth: Saída de áudio
* • Sensor de Temperatura (DHT22 ou BME280): Dados de ambiente
* • LED RGB (WS2812 ou comum): Feedback visual
* • Hub USB Alimentado: Para conectar microfone, touch etc.
* • Cartão microSD 16~32GB: Sistema operacional e dados

# Software

* • Sistema operacional: Raspberry Pi OS Lite (headless) ou com interface
* • Linguagem principal: Python 3
* • Framework web/API: FastAPI
* • Bibliotecas principais: SpeechRecognition, pyaudio, gtts, requests, pygame, gpiozero, threading ou asyncio

# Arquitetura da API

* • GET /config - Ler configurações
* • POST /config - Atualizar configurações
* • GET /status - Obter status atual
* • POST /listen - Iniciar/encerrar escuta
* • GET /history - Obter histórico

A API é usada tanto pela interface web quanto pela interface embarcada.

# Estrutura de Pastas do Projeto

keila/  
├── main.py  
├── api/  
│ ├── \_\_init\_\_.py  
│ ├── routes.py  
│ ├── config\_handler.py  
│ └── gpt\_engine.py  
├── ui/  
│ ├── tela\_embarcada.py  
│ ├── avatar.py  
│ └── status\_bar.py  
├── web/  
│ ├── static/  
│ ├── templates/  
│ └── web\_ui.py  
├── voice/  
│ ├── listener.py  
│ ├── speaker.py  
│ └── processor.py  
├── hardware/  
│ ├── leds.py  
│ ├── temperature.py  
│ └── gpio\_init.py  
├── services/  
│ └── boot\_service.sh  
├── settings.json  
├── history.log  
└── requirements.txt

# Fluxo de Funcionamento

1. Ao iniciar, main.py inicializa GPIOs, API, interface e escuta.  
2. O botão ou interface inicia a captura de áudio.  
3. O áudio é enviado à API de STT.  
4. O texto é enviado ao ChatGPT.  
5. A resposta é convertida em áudio (TTS).  
6. O áudio é reproduzido.  
7. A tela mostra a conversa.

# Interface Web Admin

Acessível via navegador na rede local, permite visualizar e alterar configurações da Keila, histórico, ativar recursos e mais.

# Interface Local Embarcada

Simples e clara, com botão de escuta, avatar animado, resposta em texto e status básicos.

# Segurança

A API pode ser protegida com senha/token, acesso remoto restrito à rede local, logs com permissões adequadas.

# Inicialização Automática

Utiliza um serviço systemd para iniciar o script automaticamente no boot.

# Simulação

O projeto pode ser simulado em um PC com Python, Flask, arquivos de áudio e mocks para sensores.