**💡 Proposta de Integração de IA no WACS**

**🧠 1. Assistente Virtual Inteligente**

Interface conversacional que entende comandos de voz e texto.

**Funcionalidades:**

* **Controle da cadeira de rodas por voz** (ex: “Leve-me para o banheiro acessível mais próximo”).
* **Navegação assistida** com base na localização e nos filtros de acessibilidade.
* **Interação com o app**: “Quero adicionar um lugar”, “Mostre locais acessíveis com rampas”, etc.
* **Lembretes inteligentes**: "Você costuma ir ao mercado às sextas, quer uma rota acessível?"

**Ferramentas:**

* speech\_recognition ou whisper para reconhecer voz.
* transformers da Hugging Face ou OpenAI API para entendimento da linguagem.
* TTS (Text-to-Speech): pyttsx3, gTTS ou Edge-TTS pra resposta falada.

**🗺️ 2. IA de Mapeamento Inteligente**

Análise e sugestões com base nos dados locais salvos.

**Funcionalidades:**

* **Sugestão automática de locais acessíveis** com base em histórico e localização.
* **Análise de acessibilidade** com IA: por exemplo, avaliar uma imagem de entrada e detectar presença de rampa.
* **Rota otimizada com IA**: evitar ruas com ladeiras íngremes ou calçadas sem acessibilidade.
* **Validação automatizada de dados**: detectar se um local é suspeito ou inválido com base em comportamento do usuário.

**Ferramentas:**

* scikit-learn ou XGBoost para modelos preditivos.
* OpenCV + CNN para análise de imagem (ex: entrada do local).
* NetworkX ou GraphHopper (com IA) para rotas inteligentes.

**🔄 3. Gerenciamento inteligente do sistema**

Backend com IA cuidando de tudo em tempo real.

**Funcionalidades:**

* **Recomendações personalizadas** com base no perfil do usuário (ex: preferências de acessibilidade).
* **Detecção de padrões de uso** para melhorias automáticas no app.
* **Notificações preditivas** (ex: “Choveu muito ontem, a rota X pode estar ruim”).
* **Gerenciamento de usuários, denúncias e validações** com modelos de detecção de anomalias.

**Ferramentas:**

* Python com FastAPI para servir a IA como API.
* Firebase ou SQLite para receber os dados do app.
* Modelos de clustering e classificação para segmentação.

**🔊 Comando de Voz e Conversa Natural**

**Como integrar:**

1. **Gravação no app React Native** → envia áudio pro backend em Python.
2. **Python processa áudio**:
   * Usa reconhecimento de voz (speech\_recognition / whisper)
   * Passa para um modelo LLM (tipo GPT-4, mistral, ou um Rasa local) para entender a intenção.
   * Executa ação correspondente (abrir mapa, adicionar local, etc.)
3. **Responde com áudio** (TTS) e/ou ação no app.

**Alternativas:**

* Rasa (open source, treinável, conversa contextual).
* OpenAI GPT (via API, simples de usar, precisa conexão).
* Whisper (para voz), TTS com ElevenLabs ou Edge-TTS (para fala natural).

**🚀 Exemplo prático de uso**

Usuário fala:  
“Quero encontrar um restaurante acessível com banheiro adaptado perto de mim.”

O sistema:

1. Reconhece a fala.
2. Converte pra texto e identifica a intenção.
3. Usa a API do Google Maps + dados locais do WACS.
4. Filtra os locais com banheiro adaptado e acessibilidade.
5. Traça a rota mais acessível.
6. Fala: “Encontrei 3 restaurantes acessíveis. O mais próximo é o [nome], a 300 metros. Deseja iniciar a rota?”

**🛠️ Infraestrutura sugerida**

| **Componente** | **Tecnologia** |
| --- | --- |
| Backend IA | Python + FastAPI |
| Reconhecimento de Voz | Whisper ou speech\_recognition |
| NLP / Conversa | transformers (HuggingFace) ou Rasa |
| TTS | pyttsx3, gTTS, Edge-TTS, ElevenLabs |
| Integração com App | API REST via Axios ou fetch em React Native |