Waelai: Integração Avançada WhatsApp com ElaiRoo

1. Visão Geral

Este documento detalha os requisitos e o plano de implementação para a integração avançada do WhatsApp com o sistema ElaiRoo (Roo-Code), permitindo que o WhatsApp se torne uma interface primária para interação, gerenciamento de tarefas e configuração dentro do ElaiRoo.

2. Objetivos

- Tornar o MCP do WhatsApp uma parte integral e não removível do ElaiRoo.
- Fornecer uma interface de usuário no frontend (webview-ui) para configuração e visualização do WhatsApp.
- Implementar um sistema de comandos via chat do WhatsApp (`@elai`, `@elaifim`, `@elainow`) para criar e gerenciar tarefas no ElaiRoo.
- Estabelecer regras claras para o ciclo de vida das sessões/tarefas iniciadas via WhatsApp.

3. Atores

- Usuário do ElaiRoo/WhatsApp

4. Requisitos

4.1. Requisitos Funcionais (RF)

4.1.1. Integração do MCP WhatsApp

- **RF001:** O MCP do WhatsApp deve ser iniciado automaticamente com o ElaiRoo.
- **RF002:** A ponte Go do WhatsApp deve ser iniciada automaticamente pelo servidor MCP Python.
- **RF003:** O processo de configuração da ponte (exibição de QR Code) deve ser visível no terminal principal do ElaiRoo se a configuração for necessária.
- **RF004:** O MCP do WhatsApp deve ser configurado como um componente fixo do sistema, não passível de exclusão pela interface de gerenciamento de MCPs padrão.

4.1.2. Interface Frontend (Webview-UI)

- **RF005:** Adicionar uma seção "WhatsApp" na aba "Servidores MCP" do ElaiRoo.
- **RF006:** A seção "WhatsApp" deve ser a primeira listada.
- **RF007:** A seção deve conter um botão "Conectar/Desconectar WhatsApp".
- **RF008:** Ao clicar em "Conectar", se a ponte não estiver configurada, o fluxo de exibição do QR Code (originado na ponte Go, gerenciado pelo MCP Python) deve ser apresentado na interface do ElaiRoo (possivelmente em um modal ou no terminal integrado).
- **RF009:** Uma vez conectado, um painel estilo "terminal de chat" deve ser renderizado na interface.
- **RF010:** O painel de chat deve exibir as conversas do WhatsApp (inicialmente, talvez focar na conversa que está interagindo com `@elai` ou um resumo).

- **RF011:** O painel de chat deve ser persistente (manter estado entre sessões do VS Code, se possível).
- **RF012:** O painel de chat deve ser expansível e recolhível.

4.1.3. Comandos via Chat WhatsApp

- **RF013:** O sistema deve reconhecer o comando `@elai <mensagem>` em uma conversa do WhatsApp.
- **RF014:** Ao receber `@elai <mensagem>`, o sistema deve criar uma nova tarefa no ElaiRoo, utilizando `<mensagem>` como contexto inicial.
- **RF015:** O sistema deve reconhecer o comando `@elaifim` em uma conversa do WhatsApp.
- **RF016:** Ao receber `@elaifim`, a tarefa ativa associada àquele JID (identificador da conversa) deve ser encerrada/arquivada.
- **RF017:** O sistema deve reconhecer o comando `@elainow <mensagem>` em uma conversa do WhatsApp.
- **RF018:** Ao receber `@elainow <mensagem>`, `<mensagem>` deve ser adicionada à tarefa ativa existente para aquele JID.
- **RF019:** Se `@elainow` for usado e não houver tarefa ativa para o JID, uma nova tarefa deve ser criada (comportamento similar ao `@elai`).

4.1.4. Gerenciamento de Sessão/Tarefa

- **RF020:** Cada JID do WhatsApp pode ter no máximo uma tarefa ativa por vez.
- **RF021:** Se não houver interação com comandos (`@elai`, `@elaifim`, `@elainow`) em uma conversa (JID) por 24 horas, a tarefa ativa associada deve ser "guardada" (encerrada/arquivada).
- **RF022:** Após uma tarefa ser guardada (por tempo ou por `@elaifim`), o contexto para aquele JID é resetado. Um novo comando `@elai` iniciará uma tarefa completamente nova.

4.2. Requisitos Não Funcionais (RNF)

- **RNF001:** A inicialização da ponte e do MCP não deve impactar significativamente o tempo de startup do ElaiRoo.
- **RNF002:** A comunicação entre a ponte Go, o MCP Python e o frontend deve ser eficiente.
- **RNF003:** A interface do usuário deve ser intuitiva e responsiva.
- **RNF004:** O sistema deve ser robusto a falhas na ponte ou no MCP (ex: tentar reiniciar a ponte).
- **RNF005:** As interações devem ocorrer na instância atual do VS Code do usuário.

4.3. Regras de Negócio (RN)

- **RN001:** O comando `@elai` sempre inicia uma nova tarefa, potencialmente arquivando uma anterior se o limite de 24h foi atingido ou se um `@elaifim` foi usado.
- **RN002:** O JID da conversa do WhatsApp é a chave primária para o rastreamento de tarefas/sessões.
- **RN003:** A funcionalidade de "guardar" a tarefa após 24h de inatividade é baseada no último comando (`@elai`, `@elainow`) recebido para aquele JID.

5. Análise de Arquitetura e Impacto

5.1. Estrutura do Projeto ElaiRoo

- **`package.json`**: Ponto de entrada da extensão VS Code, define ativação, contribuições (views, commands).
- **`src/extension.ts`**: Lógica principal da extensão, onde a inicialização de MCPs e a comunicação com webviews provavelmente ocorrem.
- **`src/services/mcp/`**: Local atual do `whatsapp-mcp` e `whatsapp-bridge`. A lógica de carregamento e "fixação" do MCP será investigada aqui ou em `src/core/`.
- **`src/core/clineProvider.ts` (suposição)**: Provável responsável pela comunicação entre webview, MCPs e o núcleo do ElaiRoo. Interações com tarefas (`@elai`) passarão por aqui.
- **`webview-ui/`**: Frontend da aplicação (React/Vue/Svelte com Vite).
  - **`webview-ui/src/components/settings/` (suposição)**: Onde a aba "Servidores MCP" é renderizada.
- **`whatsapp-bridge/main.go`**: Ponte Go. Já modificada para ter `/api/ping`. Precisa garantir que o payload `@elai` é enviado corretamente para o MCP Python.
- **`whatsapp-mcp-server/main.py` e `whatsapp.py`**: Servidor MCP Python. Já modificado para iniciar a ponte. Precisará de lógica para:
  - Receber e interpretar comandos (`@elai`, `@elaifim`, `@elainow`) da ponte.
  - Gerenciar estado das tarefas (JID, ID da tarefa, timestamp da última atividade).
  - Comunicar com o `ClineProvider` (ou similar) para criar/atualizar tarefas no ElaiRoo.

5.2. Fluxo de Dados para Comandos `@elai`

1. Usuário envia mensagem no WhatsApp: `Amigo: @elai Crie um resumo sobre IA generativa`.
2. Ponte Go (`whatsapp-bridge`) detecta `@elai` e envia um JSON para o MCP Python.
   - Payload Exemplo: `{ "jid": "NUMERO_AMIGO@s.whatsapp.net", "command": "elai", "text": "Crie um resumo sobre IA generativa", "timestamp": "..." }`
3. MCP Python (`whatsapp.py`) recebe o payload.
4. MCP Python processa o comando:
   - Verifica se há tarefa ativa para o JID.
   - Se 24h de inatividade ou `@elaifim` anterior, arquiva a antiga.
   - Cria uma nova "intenção de tarefa" com o JID e o texto.
5. MCP Python comunica ao ElaiRoo Core (via `ClineProvider`?): "Nova tarefa solicitada via WhatsApp: JID, Texto".
6. ElaiRoo Core cria a tarefa e, opcionalmente, envia uma confirmação de volta (que pode ou não ser repassada ao WhatsApp).
7. Frontend (`webview-ui`) pode ser notificado para atualizar alguma visualização de tarefas ativas.

6. Plano de Ação (Próximos Passos)

1. **Investigar Carregamento de MCPs no ElaiRoo:**
   - Analisar `src/extension.ts`, `src/services/`, `src/core/` para entender como MCPs são registrados e carregados.
   - Identificar o local para modificar e tornar o WhatsApp MCP "fixo" e sempre ativo.
2. **Desenvolvimento Frontend (`webview-ui`):**

    - Identificar o componente da aba "Servidores MCP".
    - Adicionar a seção "WhatsApp" com botão "Conectar".
    - Implementar a lógica de comunicação para o fluxo de QR Code.
    - Projetar e implementar o painel de chat (inicialmente pode ser apenas um log de mensagens relevantes).
3. **Desenvolvimento Backend/MCP:**
    - **Ponte Go:** Confirmar/Ajustar o envio do payload `@elai` para o MCP Python.
    - **MCP Python:**
        - Implementar o recebimento e parsing dos comandos `@elai`, `@elaifim`, `@elainow`.
        - Desenvolver a lógica de gerenciamento de estado das tarefas (armazenamento em `mcp_messages.db` ou novo DB).
        - Implementar a regra de timeout de 24h.
        - Definir e implementar a interface de comunicação com o `ClineProvider` (ou equivalente) para gerenciamento de tarefas no ElaiRoo.
4. **Testes e Refinamentos.**

7. Considerações Futuras

- Envio de notificações do ElaiRoo de volta para o WhatsApp.
- Interface de chat mais rica no frontend.
- Suporte a múltiplos números/contas WhatsApp (se aplicável).

Este documento é vivo e será atualizado conforme o desenvolvimento progride.