ManAI Vim Plugin - Documentação Completa

Índice

Introdução

O **ManAI Vim Plugin** é uma integração revolucionária que traz o poder do assistente de comandos Linux ManAI directamente para o seu editor Vim ou Neovim. Esta ferramenta permite que desenvolvedores, administradores de sistema e utilizadores de Linux acedam instantaneamente a conhecimento especializado sobre comandos Linux sem sair do ambiente de edição.

Características Principais

O plugin oferece uma experiência integrada e fluida, permitindo consultas em tempo real ao ManAI através de uma interface nativa do Vim. As funcionalidades incluem consultas contextuais baseadas no tipo de ficheiro, histórico de consultas persistente, cache inteligente para melhor performance, e suporte completo para múltiplos idiomas.

A arquitectura do plugin foi desenhada para ser não-intrusiva e altamente configurável, permitindo que cada utilizador adapte a experiência às suas necessidades específicas. O sistema de mapeamentos de teclado é flexível e pode ser completamente

personalizado, enquanto a integração com gestores de plugins populares torna a instalação simples e directa.

Benefícios para Produtividade

A integração do ManAI com o Vim elimina a necessidade de alternar entre aplicações para obter ajuda sobre comandos Linux. Isto resulta numa melhoria significativa do fluxo de trabalho, especialmente para tarefas que envolvem scripting, administração de sistema, ou desenvolvimento em ambientes Linux.

O plugin é particularmente valioso para programadores que trabalham com múltiplas linguagens e tecnologias, oferecendo consultas contextuais que consideram o tipo de ficheiro actual. Por exemplo, ao trabalhar num script bash, as consultas são automaticamente contextualizadas para scripting, enquanto que num ficheiro Python, o foco é na execução de código Python em ambientes Linux.

Instalação

Instalação Automática

A forma mais simples de instalar o ManAI Vim Plugin é através do script de instalação automática:

```
curl -fsSL https://raw.githubusercontent.com/edutech-angola/manai-vim-plugin/main/install.sh | bash
```

Este script detecta automaticamente o seu editor (Vim ou Neovim), o gestor de plugins em uso, e configura tudo adequadamente. O instalador também verifica dependências, instala o cliente Python necessário, e cria uma configuração básica funcional.

Instalação Manual

Para utilizadores que preferem controlo total sobre o processo de instalação, o plugin pode ser instalado manualmente seguindo os passos específicos para cada gestor de plugins.

Vim-Plug

Para utilizadores do vim-plug, adicione a seguinte linha ao seu `.vimrc` ou `init.vim`:

```
Plug 'edutech-angola/manai-vim-plugin'
```

Depois execute `:PlugInstall` no Vim para completar a instalação.

Vundle

Para utilizadores do Vundle, adicione ao seu `.vimrc`:

```
Plugin 'edutech-angola/manai-vim-plugin'
```

Execute `:PluginInstall` para instalar o plugin.

Pathogen

Clone o repositório directamente no directório de plugins do Pathogen:

```
cd ~/.vim/bundle
git clone https://github.com/edutech-angola/manai-vim-plugin.git
```

Instalação Nativa (Vim 8+ / Neovim)

Para Vim 8+ ou Neovim, pode usar o sistema de packages nativo:

```
# Para Vim
mkdir -p ~/.vim/pack/manai/start
cd ~/.vim/pack/manai/start
git clone https://github.com/edutech-angola/manai-vim-plugin.git

# Para Neovim
mkdir -p ~/.config/nvim/pack/manai/start
cd ~/.config/nvim/pack/manai/start
git clone https://github.com/edutech-angola/manai-vim-plugin.git
```

Verificação da Instalação

Após a instalação, reinicie o Vim e execute `:ManAIHelp` para verificar se o plugin foi carregado correctamente. Se tudo estiver configurado adequadamente, deve ver a janela de ajuda do ManAI com informações sobre comandos disponíveis e mapeamentos de teclado.

Configuração

Configuração Básica

O ManAI Vim Plugin funciona imediatamente após a instalação com configurações padrão sensatas. No entanto, pode personalizar o comportamento através de variáveis de configuração no seu `.vimrc` ou `init.vim`.

Configurações Essenciais

```
" Idioma das respostas (pt, en, es, fr, de, it, ru, zh, ja, ko)
let g:manai_language = 'pt'

" Altura da janela de resultados
let g:manai_window_height = 15

" Fechar janela automaticamente ao mover cursor
let g:manai_auto_close = 1

" Activar cache de respostas
let g:manai_cache_enabled = 1
```

Configurações de API

Para utilizadores com acesso a instâncias personalizadas do ManAI ou chaves de API específicas:

```
" URL personalizada da API
let g:manai_api_url = 'https://sua-instancia.exemplo.com/api/ManaiAgentHttpTrigger'

" Chave de função personalizada
let g:manai_function_key = 'sua-chave-personalizada'

" Timeout para consultas (em segundos)
let g:manai_timeout = 30
```

Configurações Avançadas

Personalização de Mapeamentos

Por padrão, o plugin define vários mapeamentos de teclado convenientes. Pode desabilitar os mapeamentos padrão e definir os seus próprios:

```
" Desabilitar mapeamentos padrão
let g:manai_no_mappings = 1
let g:manai_no_advanced_mappings = 1

" Definir mapeamentos personalizados
nnoremap <F1> :ManAIHelp<CR>
nnoremap <F2> :ManAIHistory<CR>
nnoremap <leader>ma :ManAIWord<CR>
nnoremap <leader>ml :ManAILine<CR>
vnoremap <leader>ms :ManAISelection<CR>
```

Configuração de Cache

O sistema de cache pode ser ajustado para optimizar performance:

```
" Tamanho máximo do cache
let g:manai_cache_size = 50

" Tempo de vida do cache (em minutos)
let g:manai_cache_ttl = 60

" Directório personalizado para cache
let g:manai_cache_dir = expand('~/.vim/manai_cache')
```

Configurações de Interface

A aparência e comportamento da interface podem ser personalizados:

```
" Tipo de janela (split, vsplit, tab, floating)
let g:manai_window_type = 'split'

" Posição da janela (top, bottom, left, right)
let g:manai_window_position = 'bottom'

" Activar sintaxe colorida nos resultados
let g:manai_syntax_highlighting = 1

" Tema de cores personalizado
let g:manai_color_scheme = 'default'
```

Comandos e Funcionalidades

Comandos Principais

O ManAI Vim Plugin oferece uma gama abrangente de comandos para diferentes cenários de uso.

`:ManAI <pergunta>`

O comando principal permite fazer qualquer pergunta ao ManAI directamente:

```
 :ManAI Como listar ficheiros ocultos?
:ManAI Qual a diferença entre chmod e chown?
:ManAI Como fazer backup de uma base de dados MySQL?
```

Este comando suporta auto-completar inteligente baseado em comandos Linux comuns e consultas frequentes.

`:ManAIWord`

Consulta informações sobre a palavra sob o cursor. Especialmente útil quando está a ler código ou documentação e encontra um comando desconhecido:

```
 " Posicione o cursor sobre 'rsync' e execute
:ManAIWord
" Resultado: Informações detalhadas sobre o comando rsync
```

`:ManAILine`

Analisa e explica a linha actual do ficheiro. Ideal para compreender comandos complexos ou scripts:

```
 " Numa linha como: find /var/log -name "*.log" -mtime +7 -delete
:ManAILine
" Resultado: Explicação detalhada do comando find com todas as opções
```

`:ManAISelection`

No modo visual, permite consultar sobre texto seleccionado. Útil para analisar blocos de código ou comandos multi-linha:

```
 " Seleccione um bloco de código bash e execute
:'<,'>ManAISelection
" Resultado: Análise completa do script seleccionado
```

Comandos de Gestão

`:ManAIHistory`

Mostra o histórico das últimas consultas realizadas:

```
:ManAIHistory
```

O histórico é persistente entre sessões e permite repetir consultas anteriores facilmente.

`:ManAIHistorySelect <número>`

Repete uma consulta específica do histórico:

```
:ManAIHistorySelect 3
" Repete a terceira consulta do histórico
```

`:ManAISnippets`

Mostra uma lista de consultas pré-definidas para cenários comuns:

```
:ManAISnippets
```

Os snippets incluem consultas sobre gestão de ficheiros, processos, rede, e outras tarefas administrativas comuns.

`:ManAISnippet <nome>`

Executa um snippet específico:

```
 :ManAISnippet find_files
:ManAISnippet permissions
:ManAISnippet network
```

Comandos de Configuração

`:ManAIConfig`

Mostra a configuração actual do plugin:

```
:ManAIConfig
```

`:ManAIClearCache`

Limpa o cache de respostas:

```
:ManAIClearCache
```

`:ManAIInteractive`

Activa/desactiva o modo interactivo, que oferece sugestões automáticas baseadas no contexto:

```
:ManAIInteractive
```

Mapeamentos de Teclado

Mapeamentos Padrão

O plugin define mapeamentos convenientes para acesso rápido às funcionalidades mais utilizadas.

Modo Normal

- `<leader>ma` - Consultar palavra sob cursor (`:ManAIWord`)
- `<leader>ml` - Consultar linha actual (`:ManAILine`)
- `<leader>mh` - Mostrar ajuda (`:ManAIHelp`)
- `<leader>mc` - Mostrar configuração (`:ManAIConfig`)
- `<leader>mq` - Consulta interactiva (`:ManAI`)

Modo Visual

- `<leader>ms` - Consultar selecção (`:ManAISelection`)

Teclas de Função

- `<F1>` - Ajuda rápida
- `<F2>` - Histórico de consultas
- `<F3>` - Lista de snippets

Mapeamentos Avançados

Para utilizadores experientes, o plugin oferece mapeamentos adicionais para funcionalidades avançadas:

- `<leader>mH` - Histórico detalhado
- `<leader>mS` - Snippets expandidos
- `<leader>mI` - Modo interactivo
- `<leader>mC` - Limpar cache
- `<leader>mx` - Consulta contextual

Personalização de Mapeamentos

Pode criar mapeamentos personalizados para consultas específicas do seu fluxo de trabalho:

```
" Consultas específicas para desenvolvimento
nnoremap <leader>mgit :ManAI Como usar git para<CR>
nnoremap <leader>mdocker :ManAI Como usar docker para<CR>
nnoremap <leader>mssh :ManAI Como configurar SSH para<CR>

" Consultas baseadas no tipo de ficheiro
autocmd FileType sh nnoremap <buffer> <leader>mbash :ManAIContext Como melhorar este script bash?<CR>
autocmd FileType python nnoremap <buffer> <leader>mpy :ManAIContext Como executar este Python no Linux?<CR>
autocmd FileType dockerfile nnoremap <buffer> <leader>mdock :ManAIContext Como optimizar este Dockerfile?<CR>
```

Funcionalidades Avançadas

Sistema de Cache Inteligente

O ManAI Vim Plugin implementa um sistema de cache sofisticado que melhora significativamente a performance e reduz a latência das consultas repetidas.

Funcionamento do Cache

O cache armazena respostas localmente usando um hash da pergunta como chave. Isto permite que consultas idênticas sejam respondidas instantaneamente sem necessidade de comunicação com a API. O sistema é inteligente o suficiente para reconhecer variações menores na formulação de perguntas similares.

O cache é persistente entre sessões do Vim, sendo armazenado no directório de configuração do utilizador. A gestão automática do tamanho do cache garante que o sistema não consuma espaço excessivo em disco, removendo automaticamente entradas mais antigas quando o limite é atingido.

Configuração do Cache

```
" Activar/desactivar cache
let g:manai_cache_enabled = 1

" Tamanho máximo do cache (número de entradas)
let g:manai_cache_size = 100

" Tempo de vida das entradas (em horas)
let g:manai_cache_ttl = 24

" Directório personalizado para cache
let g:manai_cache_dir = expand('~/.vim/manai_cache')
```

Consultas Contextuais

Uma das funcionalidades mais poderosas do plugin é a capacidade de realizar consultas contextuais baseadas no tipo de ficheiro actual e no conteúdo em edição.

Detecção Automática de Contexto

O plugin analisa automaticamente o tipo de ficheiro (`filetype`) e adapta as consultas para fornecer respostas mais relevantes. Por exemplo, ao trabalhar num ficheiro Python, as consultas sobre comandos Linux são automaticamente contextualizadas para desenvolvimento Python em ambientes Linux.

```
" Em ficheiro .sh (bash script)
:ManAI como processar argumentos
" Resultado focado em: getopts, $1, $2, etc.

" Em ficheiro .py (Python)
:ManAI como processar argumentos
" Resultado focado em: sys.argv, argparse, subprocess
```

Comando de Contexto Explícito

Para controlo total sobre o contexto, use o comando `:ManAIContext`:

```
:ManAIContext No contexto de Docker, como optimizar imagens?
:ManAIContext Para administração de servidores, como monitorizar performance?
```

Histórico Persistente

O sistema de histórico mantém um registo completo de todas as consultas realizadas, permitindo fácil acesso a informações consultadas anteriormente.

Funcionalidades do Histórico

O histórico é armazenado de forma persistente no ficheiro `~/.vim_manai_history` e inclui timestamp, pergunta original, e metadados sobre o contexto da consulta. O sistema evita duplicados e mantém as consultas mais recentes no topo da lista.

```
" Ver histórico completo
:ManAIHistory

" Pesquisar no histórico
:ManAIHistorySearch docker

" Repetir consulta por índice
:ManAIHistorySelect 5
```

Snippets e Templates

O sistema de snippets oferece acesso rápido a consultas pré-definidas para cenários comuns de administração de sistema e desenvolvimento.

Snippets Disponíveis

- `find_files` - Como encontrar ficheiros por nome, tipo, data
- `permissions` - Gestão de permissões com chmod, chown, chgrp
- `processes` - Monitorização e gestão de processos
- `disk_space` - Verificação e gestão de espaço em disco
- `network` - Configuração e troubleshooting de rede
- `archives` - Criação e extracção de arquivos
- `text_processing` - Processamento de texto com grep, awk, sed
- `system_info` - Obtenção de informações do sistema

Criação de Snippets Personalizados

Pode criar snippets personalizados editando a configuração:

```
let g:manai_custom_snippets = {
  \ 'backup_mysql': 'Como fazer backup completo de base de dados MySQL?',
  \ 'ssl_cert': 'Como gerar e instalar certificados SSL?',
  \ 'firewall_config': 'Como configurar firewall iptables?'
\ }
```

Modo Interactivo

O modo interactivo oferece uma experiência mais dinâmica, com sugestões automáticas e detecção de comandos Linux no texto.

Activação do Modo Interactivo

```
:ManAIInteractive
```

No modo interactivo, o plugin monitora o texto em edição e oferece sugestões contextuais quando detecta comandos Linux ou padrões reconhecíveis. Isto é especialmente útil ao escrever scripts ou documentação técnica.

Funcionalidades do Modo Interactivo

- Detecção automática de comandos Linux no texto
- Sugestões contextuais baseadas no cursor
- Highlighting de comandos reconhecidos
- Consultas automáticas para texto seleccionado
- Integração com auto-completar do Vim

Integração com Outros Plugins

FZF Integration

Para utilizadores do fzf.vim, o ManAI pode ser integrado para pesquisa fuzzy no histórico:

```
function! ManAIFzfHistory()
  let l:history = readfile(expand('~/.vim_manai_history'))
  call fzf#run({
    \ 'source': l:history,
    \ 'sink': function('s:ManaiQueryFromHistory'),
```

```
      \ 'options': '--prompt="ManAI> " --preview="echo {}"'
    \ })
endfunction

command! ManAIFzf call ManAIFzfHistory()
nnoremap <leader>mf :ManAIFzf<CR>
```

Airline/Lightline Integration

Integração com plugins de statusline para mostrar informações do ManAI:

```
 " Para vim-airline
let g:airline_section_x = '%{ManAIStatus()}'

" Para lightline
let g:lightline = {
    \ 'active': {
    \   'right': [['lineinfo'], ['percent'], ['manai']]
    \ },
    \ 'component_function': {
    \   'manai': 'ManAIStatusLine'
    \ }
\ }

function! ManAIStatusLine()
    if exists('g:manai_last_query')
        return '    ' . strftime('%H:%M', g:manai_last_query_time)
    endif
    return ''
endfunction
```

Which-Key Integration (Neovim)

Para utilizadores do which-key.nvim:

```
 local wk = require("which-key")
wk.register({
  m = {
    name = "ManAI",
    a = { ":ManAIWord<CR>", "Query word under cursor" },
    l = { ":ManAILine<CR>", "Query current line" },
    s = { ":ManAISelection<CR>", "Query selection" },
    h = { ":ManAIHelp<CR>", "Show help" },
    H = { ":ManAIHistory<CR>", "Show history" },
    S = { ":ManAISnippets<CR>", "Show snippets" },
    c = { ":ManAIConfig<CR>", "Show config" },
    i = { ":ManAIInteractive<CR>", "Toggle interactive mode" },
```

```
  },
}, { prefix = "<leader>" })
```

Telescope Integration (Neovim)

Integração com telescope.nvim para pesquisa avançada:

```
 local telescope = require('telescope')
local pickers = require('telescope.pickers')
local finders = require('telescope.finders')
local conf = require('telescope.config').values

local function manai_history_picker()
  local history_file = vim.fn.expand('~/.vim_manai_history')
  local history = {}

  if vim.fn.filereadable(history_file) == 1 then
    history = vim.fn.readfile(history_file)
  end

  pickers.new({}, {
    prompt_title = "ManAI History",
    finder = finders.new_table {
      results = history
    },
    sorter = conf.generic_sorter({}),
    attach_mappings = function(prompt_bufnr, map)
      map('i', '<CR>', function()
        local selection = require('telescope.actions.state').get_selected_entry()
        require('telescope.actions').close(prompt_bufnr)
        vim.cmd('ManAI ' .. selection[1])
      end)
      return true
    end,
  }):find()
end

vim.api.nvim_create_user_command('ManAITelescope', manai_history_picker, {})
```

Casos de Uso Práticos

Desenvolvimento de Scripts Bash

O ManAI Vim Plugin é especialmente valioso no desenvolvimento de scripts bash, oferecendo ajuda contextual e verificação de sintaxe.

Cenário: Criação de Script de Backup

Imagine que está a criar um script de backup e precisa de ajuda com comandos específicos:

```
#!/bin/bash
# Script de backup - cursor na linha seguinte
rsync -avz --delete /home/user/ /backup/
```

Posicionando o cursor sobre `rsync` e executando `<leader>ma`, obtém informações detalhadas sobre o comando, incluindo explicação das opções `-avz --delete` e sugestões de melhorias.

Consultas Contextuais para Bash

```
" Ao editar ficheiro .sh
:ManAIContext Como validar argumentos de entrada neste script?
:ManAIContext Como adicionar logging a este script bash?
:ManAIContext Como tornar este script mais robusto?
```

Administração de Sistema

Para administradores de sistema, o plugin oferece acesso rápido a conhecimento especializado sobre gestão de servidores Linux.

Cenário: Troubleshooting de Performance

Durante investigação de problemas de performance:

```
" Consultar sobre comando específico
:ManAI Como usar htop para identificar processos problemáticos?

" Analisar linha de log
" Posicionar cursor numa linha de log e usar
:ManAILine

" Consultar sobre configuração
:ManAI Como optimizar configuração do Apache para alta carga?
```

Snippets para Administração

```
:ManAISnippet system_info    " Informações do sistema
:ManAISnippet disk_space     " Gestão de espaço em disco
```

```
:ManAISnippet network       " Configuração de rede
:ManAISnippet processes     " Gestão de processos
```

Desenvolvimento Python em Linux

Para programadores Python que trabalham em ambientes Linux, o plugin oferece consultas contextualizadas.

Cenário: Deploy de Aplicação Python

```
 # requirements.txt
# Cursor numa dependência específica
Django==4.2.0
```

Executando `<leader>ma` sobre "Django", obtém informações sobre instalação, configuração e melhores práticas para deploy em Linux.

Consultas Contextuais para Python

```
 " Em ficheiro .py
:ManAIContext Como configurar ambiente virtual Python no servidor?
:ManAIContext Como fazer deploy desta aplicação Django?
:ManAIContext Como configurar supervisor para esta aplicação?
```

Configuração de Serviços

O plugin é invaluável na configuração de serviços e aplicações em servidores Linux.

Cenário: Configuração de Nginx

```
server {
  listen 80;
  server_name example.com;
  # Cursor na linha seguinte
  location / {
     proxy_pass http://localhost:3000;
  }
}
```

Seleccionando o bloco `location` e executando `<leader>ms`, obtém explicações detalhadas sobre configuração de proxy reverso e sugestões de optimização.

Análise de Logs

Para análise de ficheiros de log, o plugin oferece interpretação contextual.

Cenário: Análise de Log do Apache

```
192.168.1.100 - - [25/Dec/2023:10:15:30 +0000] "GET /api/users HTTP/1.1" 500 1234
```

Posicionando o cursor na linha e executando `:ManAILine`, obtém explicação do formato de log, significado do código de erro 500, e sugestões para investigação.

Troubleshooting

Problemas Comuns

Plugin Não Carrega

Sintoma: Comandos ManAI não são reconhecidos após instalação.

Soluções:
1. Verificar se o plugin está no directório correcto
2. Confirmar que o gestor de plugins executou a instalação
3. Verificar se há erros no `:messages`
4. Tentar recarregar com `:source ~/.vimrc`

```
" Verificar se plugin está carregado
:echo exists('g:loaded_manai')
" Deve retornar 1 se carregado

" Verificar localização do plugin
:echo globpath(&rtp, 'plugin/manai.vim')
```

Erro de Conexão com API

Sintoma: "Erro de conexão" ou timeouts nas consultas.

Soluções:
1. Verificar conectividade à internet
2. Confirmar URL da API na configuração
3. Validar chave de função
4. Aumentar timeout se necessário

```
" Testar configuração
:ManAIConfig

" Testar conexão manualmente
:ManAI teste de conexão
```

Respostas em Idioma Errado

Sintoma: Respostas em inglês quando configurado para português.

Soluções:
1. Verificar configuração de idioma
2. Confirmar que a API suporta o idioma
3. Recarregar configuração

```
" Verificar idioma configurado
:echo g:manai_language

" Alterar idioma
:let g:manai_language = 'pt'
```

Performance Lenta

Sintoma: Consultas demoram muito tempo para responder.

Soluções:
1. Activar cache se desabilitado
2. Verificar conectividade de rede
3. Aumentar tamanho do cache
4. Usar modo offline para testes

```
" Activar cache
:let g:manai_cache_enabled = 1

" Aumentar tamanho do cache
:let g:manai_cache_size = 100

" Limpar cache corrompido
:ManAIClearCache
```

Problemas de Compatibilidade

Conflitos com Outros Plugins

Alguns plugins podem interferir com o funcionamento do ManAI:

Soluções:
1. Carregar ManAI após outros plugins
2. Usar mapeamentos alternativos
3. Desabilitar funcionalidades conflituosas

```
" Carregar ManAI por último
autocmd VimEnter * runtime plugin/manai.vim

" Usar mapeamentos alternativos
let g:manai_no_mappings = 1
" Definir mapeamentos personalizados
```

Problemas com Python

Sintoma: Erros relacionados com Python ou módulos em falta.

Soluções:
1. Verificar versão do Python
2. Instalar módulos necessários
3. Configurar caminho do Python

```
" Verificar Python no Vim
:echo has('python3')

" Configurar caminho personalizado
:let g:manai_python_path = '/usr/bin/python3'
```

Debugging Avançado

Activar Logging Detalhado

Para problemas complexos, active logging detalhado:

```
" Activar debug
:let g:manai_debug = 1

" Ver logs
:ManAIDebugLog
```

```
" Limpar logs
:ManAIDebugClear
```

Testar Componentes Individualmente

```
" Testar apenas cache
:ManAITestCache

" Testar apenas API
:ManAITestAPI

" Testar configuração
:ManAITestConfig
```

API e Desenvolvimento

Arquitectura do Plugin

O ManAI Vim Plugin é estruturado em módulos independentes que podem ser estendidos e personalizados.

Componentes Principais

1. **Core Engine** (`manai-core.vim`) - Funcionalidades básicas
2. **Advanced Features** (`manai-advanced.vim`) - Funcionalidades avançadas
3. **API Client** (`manai-client.py`) - Cliente Python para API
4. **Cache System** - Sistema de cache inteligente
5. **UI Components** - Interface de utilizador

Estrutura de Ficheiros

```
manai-vim-plugin/
├── plugin/
│   ├── manai.vim           # Plugin principal
│   └── manai-advanced.vim  # Funcionalidades avançadas
├── autoload/
│   └── manai.vim           # Funções carregadas sob demanda
├── doc/
│   └── manai.txt           # Documentação do Vim
├── syntax/
│   └── manai_result.vim    # Sintaxe para resultados
└── python/
    └── manai_client.py     # Cliente Python
```

API Interna

Funções Principais

```
" Função principal de consulta
function! manai#query(question, options = {})

" Gestão de cache
function! manai#cache#get(key)
function! manai#cache#set(key, value)

" Gestão de histórico
function! manai#history#add(query)
function! manai#history#get()

" Interface de utilizador
function! manai#ui#show_result(content, title)
function! manai#ui#show_floating(content, options)
```

Eventos e Hooks

O plugin emite eventos personalizados que podem ser interceptados:

```
" Antes de fazer consulta
autocmd User ManAIPreQuery echo "Fazendo consulta..."

" Após receber resposta
autocmd User ManAIPostQuery echo "Resposta recebida"

" Erro na consulta
autocmd User ManAIError echo "Erro na consulta"
```

Extensões e Personalizações

Criar Comandos Personalizados

```
" Comando personalizado para consultas específicas
command! -nargs=1 ManAIGit call manai#query('Como usar git para ' . <q-args>)

" Função personalizada
function! MyManAIWrapper(context, question)
    let l:full_question = a:context . ': ' . a:question
    call manai#query(l:full_question)
endfunction
```

Integração com APIs Externas

```
" Integrar com API personalizada
function! manai#custom_api#query(question)
    " Implementação personalizada
    return system('curl -s "https://minha-api.com/query?q=' . shellescape(a:question) .
'"')
endfunction
```

Filtros de Resposta

```
" Filtro para processar respostas
function! manai#filter#code_only(response)
    " Extrair apenas código das respostas
    return substitute(a:response, '```\(.\{-}\)```', '\1', 'g')
endfunction
```

Contribuição para o Projeto

Estrutura de Contribuição

Para contribuir para o desenvolvimento do plugin:

1. **Fork** do repositório oficial
2. **Clone** do seu fork localmente
3. **Branch** para nova funcionalidade
4. **Desenvolvimento** com testes
5. **Pull Request** com documentação

Guidelines de Desenvolvimento

```
" Convenções de nomenclatura
" Funções públicas: manai#module#function()
" Funções privadas: s:function_name()
" Variáveis globais: g:manai_setting
" Variáveis locais: l:variable_name

" Exemplo de função bem documentada
function! manai#example#new_feature(param1, param2) abort
    " Descrição da função
    "
    " Parâmetros:
    "   param1 (string): Descrição do parâmetro
    "   param2 (dict): Opções adicionais
    "
    " Retorna:
```

```
    "   dict: Resultado da operação

    " Implementação...
endfunction
```

Testes

O plugin inclui suite de testes automatizados:

```
 " Executar todos os testes
:ManAIRunTests

" Executar testes específicos
:ManAITestModule cache
:ManAITestModule api
:ManAITestModule ui
```

Licença

O ManAI Vim Plugin é distribuído sob a licença MIT, permitindo uso livre em projectos comerciais e não-comerciais.

Termos da Licença

```
 MIT License

Copyright (c) 2024 EduTech Angola

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
```

```
OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE
SOFTWARE.
```

Contribuições

Ao contribuir para este projeto, concorda em licenciar as suas contribuições sob os mesmos termos da licença MIT.

Suporte e Comunidade

Canais de Suporte

- **GitHub Issues**: Para bugs e solicitações de funcionalidades
- **Documentação**: Wiki oficial no GitHub
- **Comunidade**: Fórum de discussão no GitHub Discussions

Roadmap

Versão 1.1 (Próxima)

- Suporte para LSP integration
- Melhorias na interface flutuante
- Novos snippets e templates
- Optimizações de performance

Versão 1.2 (Futuro)

- Integração com AI local
- Suporte para múltiplas APIs
- Interface gráfica opcional
- Sincronização entre dispositivos

Agradecimentos

Agradecemos a todos os contribuidores e à comunidade Vim/Neovim pelo feedback e sugestões que tornaram este plugin possível.
