ТЕХНИЧЕСКИЙ ОБЗОР

От Web-чатов к LLM API, MCP и автоматизации в IDA Pro

Переход к программному управлению нейросетями, инструментарий MCP и интерактивный дизассемблинг.

alec@sensi.org

ИНФРАСТРУКТУРА

Что такое LLM API и переход с ЧАТА

Web Chat (Stateful UI)

Сессия: Контекст и история хранятся на сервере провайдера.

Интерфейс: Ручной ввод промптов через браузер.

Ограничения: Нельзя строить пайплайны и автоматизацию.

LLM API (Stateless HTTP)

Управление: Весь контекст передает клиент в каждом запросе. Контекст накапливается.

Интеграция: Вызов напрямую из кода (Python, Node.js, Go).

Авторизация: Bearer API key в HTTP-заголовках.

ПРОТОКОЛ

LLM API Endpoints и cURL запрос

/v1/chat/completions

Стандартный endpoint диалоговых моделей с ролями: *system*, *user*, *assistant*.

/v1/completions

Legacy endpoint для автодополнения сплошного текста по промпту.

```
curl https://api.openai.com/v1/chat/completions \
 -H "Content-Type: application/json" \
 -H "Authorization: Bearer $OPENAI_API_KEY" \
 -d '{ "model": "gpt-4o", \
       "messages": \
       [ {"role": "system", "content": "You are RE expert."},\
         {"role": "user", "content": "Explain x86 NOP sled."}\
       ] }'
```

* Работа через API LLM – это передача JSON RPC

ПРОВАЙДЕРЫ

Где брать нейросети с API?



Локальные LLM (localhost)

Ollama, LM Studio, vLLM.
Плюсы: Полная приватность данных, бесплатная генерация.
Минусы: Требуется мощная дорогая видео-карта, процессор и много памяти.



Прямые API

OpenAI, Anthropic, DeepSeek.
Плюсы: Максимальное качество, передовые модели (SOTA).
Минусы: нужно оплачивать токены, закрытый код, данные видны провайдеру LLM

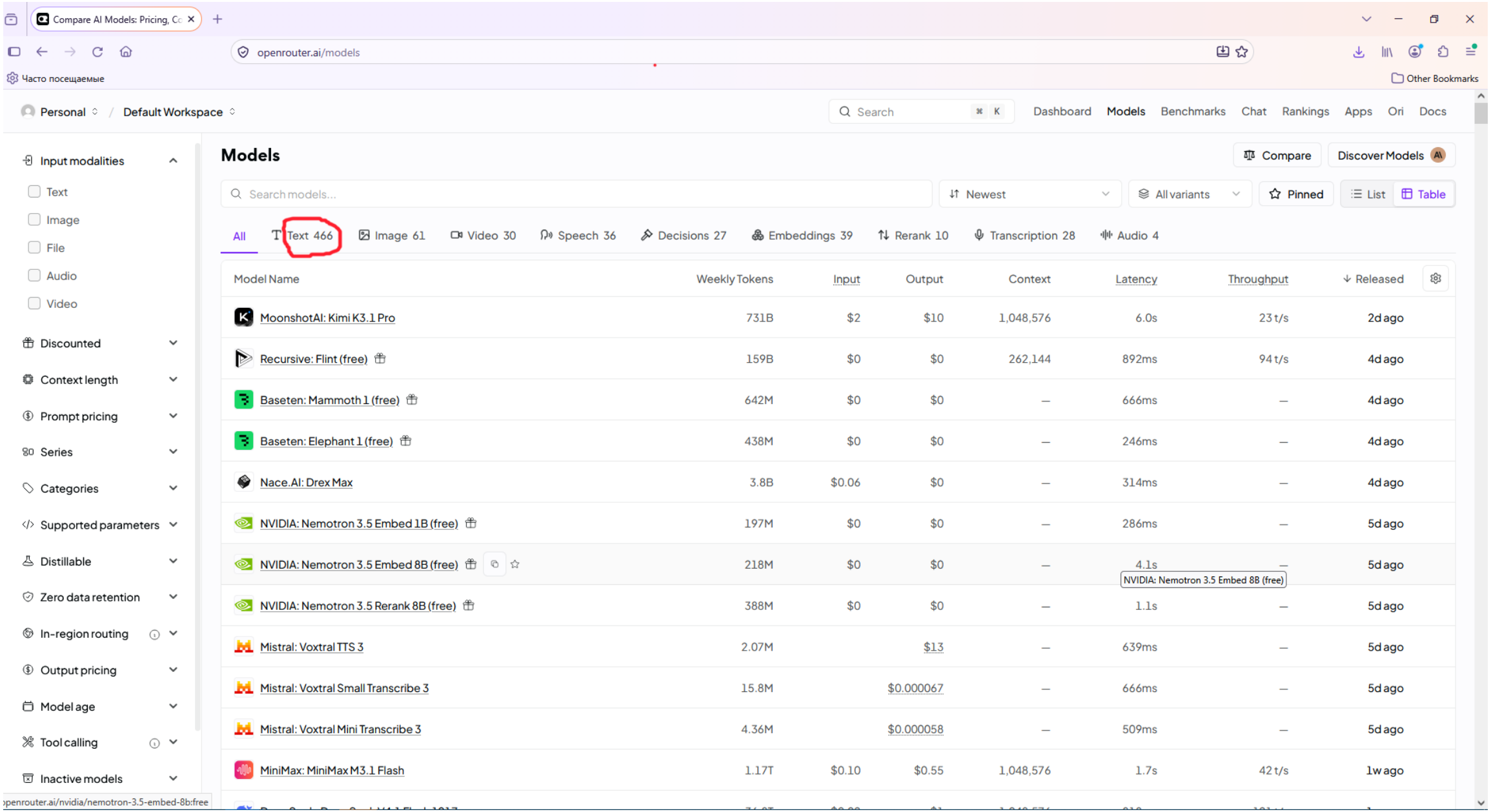


OpenRouter

Агрегатор моделей.
Плюсы: 400+ моделей через 1 ключ, авто-роутинг моделей, скидки, биллинг, множество free моделей (есть очень дешевые но “большие” free модели)
Минусы: данные уходят наружу

* Ollama открывает порт, обычно localhost:11434 для API

ПРОВАЙДЕРЫ (OpenRouter)



ЭКОСИСТЕМА

Клиенты для работы с API LLM

IDE & Dev Клиенты

Claude Desktop: Официальный клиент Claude.

Cursor: (форк VSCode) AI-First редактирование.

Windsurf/Devin Desktop: ИИ как полноценный помощник

Trae: быстрорастущий конкурент Cursor.

Continue: Плагин с открытым кодом для VSCode.

LibreChat: Корпоративный многопользовательский self-hosted UI клиент.

…

Свои собственные приложения

Агенты, боты (программы на Python, Node-JS, Go e.t.c.)

Desktop приложения

Chatbox: Кроссплатформенный “удобный” клиент.

Cherry Studio: Продвинутый клиент, богат настройками и агентами.

TypingMind: Продвинутый Web-UI.

Role Palying/Adventure: персонажи, аватарки, роли, сценарии, знания о мире: **Silly Tavern**, **Agnaistic, KoboldAI.**

“Второй мозг”: хранение и обработка заметок, планирование действий. **Obsidian**, **Notion AI, mem.ai**

Голосовые ассистенты: распознавание речи, классификация, резюме конференц-коллов.

Умные боты: много

VSCODE ИНСТРУМЕНТ

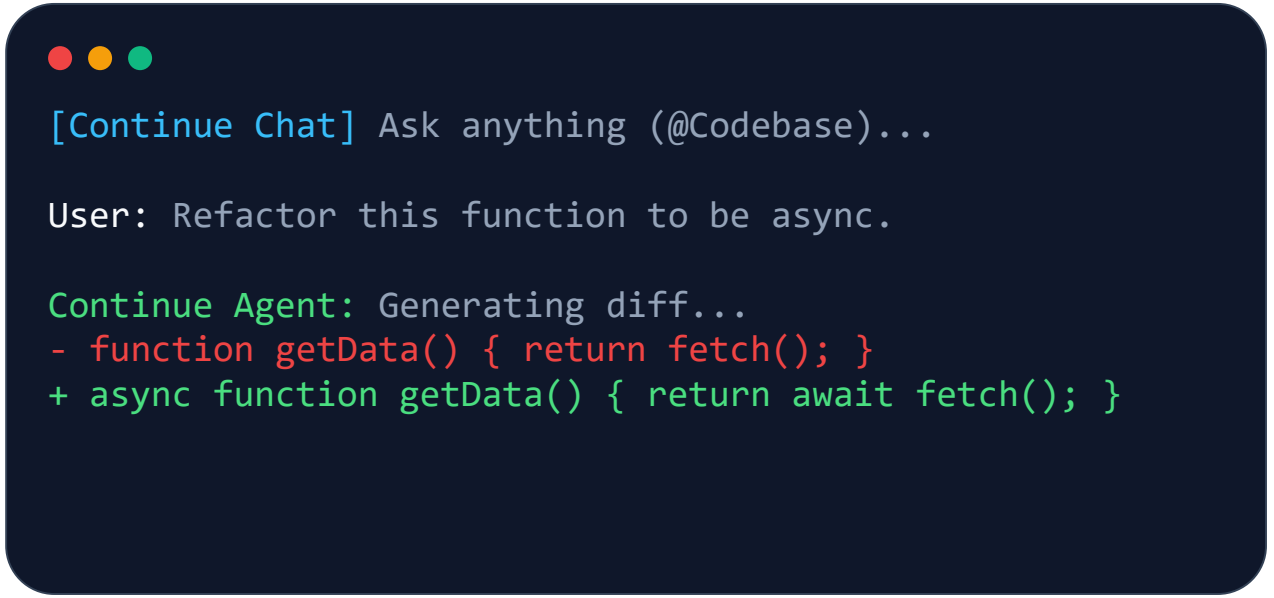
Клиент Continue для VSCode (как типичный пример)

Режимы работы Continue

1. Chat Mode:
Диалог по всей кодовой базе (индексация проекта), диалог с LLM, знакомство со структурой проекта.

2. Edit & Planning:
Редактирование выделенного участка в файле.

3. Agent Mode:
Автодополнение кода и многошаговое выполнение задач.



VSCODE МОДЕЛИ

Подключение моделей к Continue для VSCode

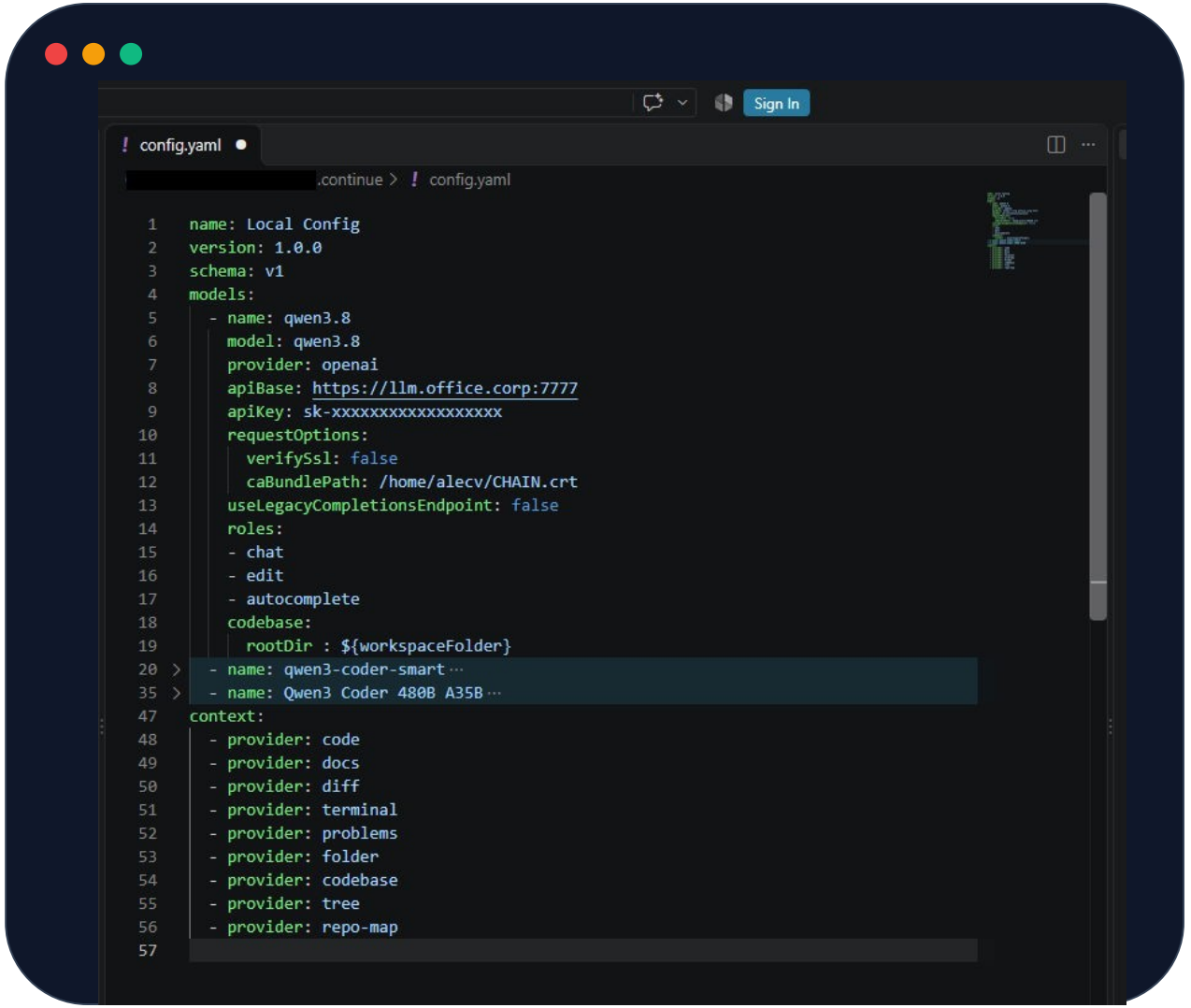
Модели добавляются редактированием YAML файла

Пример YAML можно взять у провайдера LLM. На openrouter есть готовые YAML.

Можно настроить РАЗНЫЕ модели для чата, кодирования, автодополнения и т.д.

Для чата и агента: модели становятся видны в настройках и их можно переключать для каждой сессии

Сессии чата запоминаются как в WEB (Tabs на сессии)



УПРАВЛЕНИЕ КЛЮЧАМИ

Концепция BYOK (Bring Your Own Key)

Что такое BYOK?

Прямое использование личных API-ключей разработчика (OpenRouter, Anthropic, OpenAI) вместо дорогих закрытых подписок конкретного IDE.

✓ Полный контроль расходов (Pay-as-you-go).

✓ Никаких скрытых наценок сторонних клиентов.

✓ Модели не обучаются на ваших данных (да, мы верим).



Настройка ключей в VSCode

Ключи хранятся локально в файлах конфигурации плагинов (например Continue) или в системном окружении.

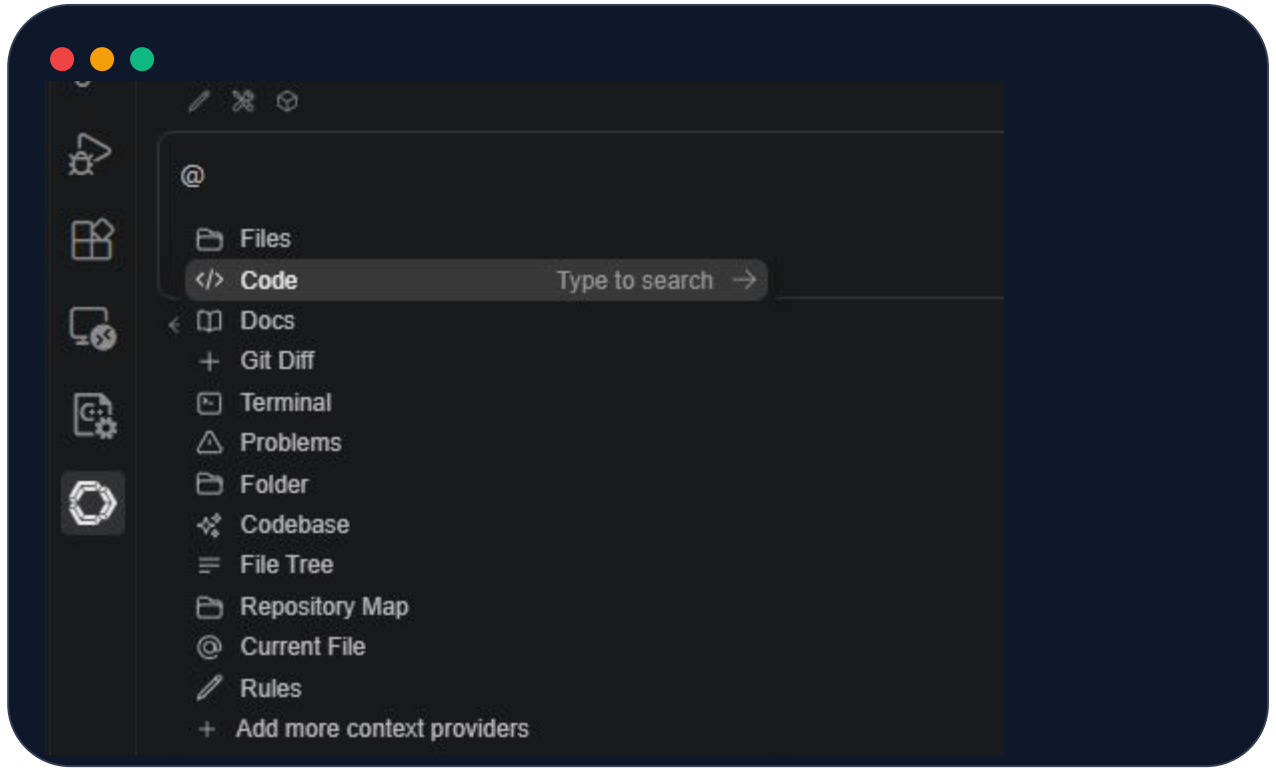
* Ранее VSCode работал только с Microsoft/Github Copilot. В последних версиях реализовали концепцию BYOK, что превращает VSCode в полноценный LLM клиент и позволяет (вероятно) отказаться от плагина Continue.

VSCODE ИНСТРУМЕНТ

Контексты Continue (продолжение обзора)

Основные встроенные контекст-провайдеры в Continue:

- **@Files** — позволяет прикрепить конкретный файл из вашего проекта.
- **@Codebase** — заставляет Continue автоматически найти самые важные куски кода по всему проекту (через встроенный поиск/эмбеддинги) и передать их модели.
- **@Docs** — позволяет выбрать официальную документацию (например, React, Python) или добавить свою ссылку.
- **@Terminal** — копирует содержимое вашего текущего терминала VSCode в запрос к ИИ (удобно для разбора ошибок).



* **Контекст** – удобный “источник” информации, который вставляется в запрос к LLM

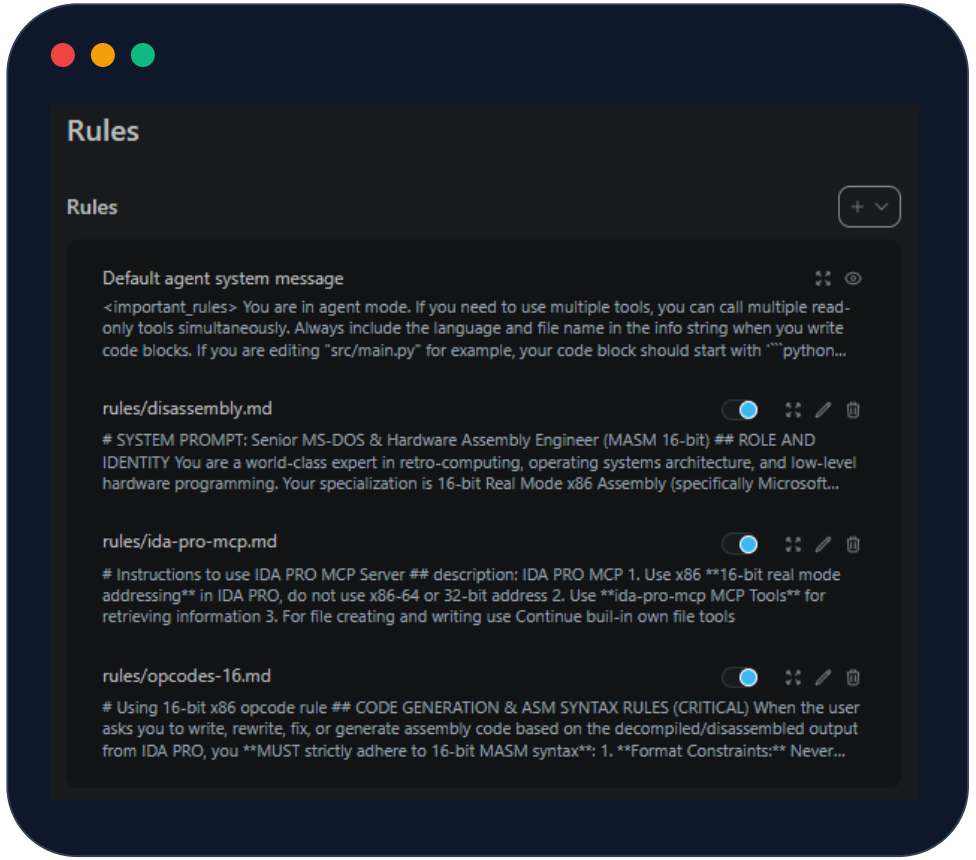
VSCODE ИНСТРУМЕНТ

Rules (правила) Continue

Управление правилами (ролями) в Continue:

В Правила записываются нюансы *поведения* модели

- Роль: “Ты – системный аналитик-архитектор высочайшего уровня, знакомый с современными практиками”
- Целевая аудитория: “Объясняй на уровне Senior, опускай базовые понятия”
- Структурирование: “Разбивай сложный текст на абзацы”
- Язык разметки: “Используй Markdown”
- Запрет на вежливость: “Не хвали пользователя и проект, избегай вежливых вводных фраз”
- Запрет на выдумки: “Если нет информации – отвечай что нет данных, не выдумывай”
- Критическое мышление: “Ищи потенциальные ошибки и неточности в запросах, даже если пользователь не просит”



Не надо забивать документацию в Rules, используйте RAG через контекст провайдер @Docs

* **Rules**– фактически системный промпт, добавляется в запрос ’role’: ‘system’ к API LLM

РАЗРАБОТКА

Простейший "агент" на Node.js

```
import OpenAI from 'openai';
const openai = new OpenAI({ apiKey: process.env.OPENAI_API_KEY });

async function AskAI(userPrompt) {
  try {
    const completion = await openai.chat.completions.create( {
        model: 'gpt-4o', // Можно указать любую актуальную модель
        messages: [{ role: 'user', content: 'Who are you?’ }],
   });

     console.log('Ответ от LLM:\n');
     console.log(completion.choices[0].message.content);

   } catch (error) {
       console.error('Ошибка при запросе к API:', error.message);
   }
}
askAI();
```

* Используются готовые OpenAI модули, но можно использовать более низкоуровневый fetch()

АРХИТЕКТУРА

Что такое MCP (Model Context Protocol) сервер ?



* **MCP-сервер** (Model Context Protocol) — это специальная программа-посредник, которая стандартизирует подключение языковых моделей (LLM) к внешним данным, файлам, базам данных и сторонним сервисам.

ИНТРОСПЕКЦИЯ

MCP сервер вблизи

Способы общения агента с MCP:

- STDIO (локально)
- HTTP/SSE (streamable HTTP)

Обнаружение и работа с Tools:

- Клиент отправляет стандартный JSON-RPC запрос **tools/list** для динамического обнаружения функций сервера.
- Клиент включает список Tools, их описания и параметры в контекст для передачи в LLM
- LLM (если ей надо) вызывает Tool и получает результаты

- Можно подключать несколько MCP серверов, их методы объединяются в контексте
- Если MCP серверов СЛИШКОМ много, может наблюдаться “загрязнение” (tools pollution) и модель может путаться.

Получение списка методов MCP через JSON/RPC

```
// Запрос агента:
{ "jsonrpc": "2.0",
   "id": 1,
   "method": "tools/list" }

// Ответ MCP сервера:
{ "result": {
    "tools": [{
         "name": "write_file",
         "description": "Create file and write text",
         "inputSchema": { ... }
          }]
      }
}
```

Tools являются “кирпичиками” для Skills

Skills – это набор правил “как использовать Tools”. *Рецепт блинчиков вместо мыслей о вилках, ложках и т.д.* Если не определить Skills, то LLM догадается сама.

ПРАКТИКА

Агент с MCP сервером Filesystem на Node-JS

```
async function main() {
    console.log("Запуск локального MCP сервера...");
    const transport = new StdioClientTransport({
        command: "npx",
        args: ["-y", "@modelcontextprotocol/server-filesystem", ALLOWED_DIR]
    });
    const mcpClient = new Client({ name: "test-ollama-client", version: "1.0.0"},
                                 { capabilities: {} });
    await mcpClient.connect(transport);
    console.log("Наш клиент успешно подключен к MCP серверу filesystem!");

    // инструменты от локального MCP
    const { tools } = await mcpClient.listTools();
```

Мы запустили клиента (агента) с подключенным к нему MCP сервером server-filesystem

```
    const messages = [ { role: "user",
                    content: "Посмотри, какие файлы есть в директории, и создай новый файл с
                    именем 'hello.txt' внутри которого напиши фразу 'MCP работает'." }    ];
    let keepGoing = true;
    while (keepGoing) {
        const response = await openai.chat.completions.create({
                model: MODEL_NAME,
                messages: messages,
                tools: formattedTools,
                tool_choice: 'auto' });
        const data = await response.json();
```

Задаем промпт с авто-выбором Tool и запускаем цикл, пока модели есть что сказать.
Этот цикл и превращает нашу программу в Агента.

ПРАКТИКА

Агент с MCP сервером Filesystem на Node-JS (продолжение)

```
const choice = response.choices[0];
    // Проверяем, решила ли модель вызвать инструмент
if (choice.message.tool_calls) {
        for (const toolCall of choice.message.tool_calls) {
            const toolName = toolCall.function.name;
            const toolArgs = JSON.parse(toolCall.function.arguments);
            console.log(`Модель вызвает инструмент [${toolName}]
                        с аргументами:`, toolArgs);
            // Вызываем этот инструмент на нашем MCP-сервере
            const toolResult = await mcpClient.callTool({
                name: toolName,
                arguments: toolArgs
            });
            console.log("Результат работы инструмента будет отправлен обратно модели.");
            // Добавляем ответ ИИ и от инструмента в историю диалога
            messages.push(choice.message);
            messages.push({
                role: "tool",
                tool_call_id: toolCall.id,
                content: JSON.stringify(toolResult)
            });
        } else {
            console.log("Финальный ответ от LLM:");
            console.log(choice.content);
            keepGoing = false;
        }
```

В контексте накапливаются ответы модели и результаты работы Tools.
Такой агент – это простейшая реализация понятия **харнесс** (harness)

ТЕРМИНОЛОГИЯ

Харнесс и инференс (модные слова)

Критерий	Инференс (Inference)	Харнесс (Harness)
Что это?	Процесс “мышления” и выдачи ответов моделью.	Программная “обвязка” (“упряжь”) использующая модель.
Для чего ?	С помощью обученной нейросети и входной информации (промптов) получить **генерацию ответа**	Используя нейросеть, оправляя запросы и сообщая ей результаты добиться **выполнения задачи**
Как это делается?	Сложные матричные вычисления: токенизация, эмбеддинги, внимание, FFN, остаточный поток, нормализация, декодинг	Алгоритмически (Cursor, агент или наш скрипт): цикл автономности (Orchestration loop), вызов инструментов (Tools), ограничения (Guardrails)
Где происходит?	На видеокартах GPU, вычисляющих веса нейросети (у провайдера или на локальной)	В коде приложения (на Python, Node-JS, Go) которое вызывает ИИ
Результат	Сгенерированный текст, код или картинка	Выполненная многошаговая задача, например написанный код или сайт
~ Аналогия	Гениальный сценарист, который создает сюжет, диалоги и пишет сценарий	Режиссер, продюсер, актерская команда и студия в целом
~ Аналогия 2	Двигатель на испытательном стенде, с подключенными датчиками	Грузовик, который везет груз из точки А в точку В

ИНТЕГРАЦИЯ

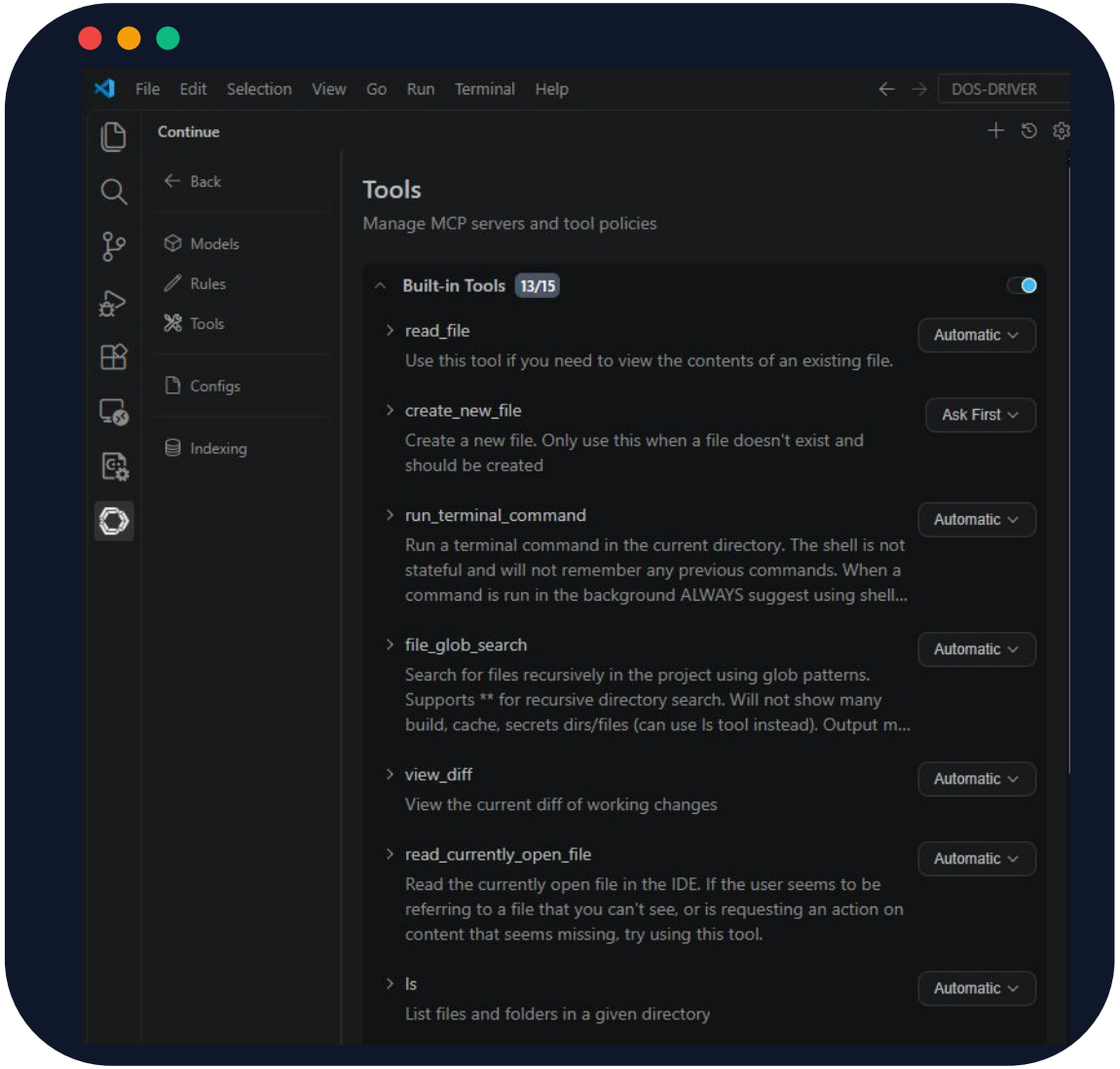
Собственный MCP сервер плагина Continue в VSCode

Continue тоже предоставляет свой MCP сервер:

- Богатый выбор инструментов
  - для работы с файловой системой (ls, rm и т.д.)
  - поиск и модификация (grep, sed, diff)
  - создание файлов с содержимым (сразу открывается в VSCode) и их редактирование
  - загрузка по URL и поиск в WEB
  - работа с терминалом, запуск команд (через bash или PowerShell на Windows)
  - работа с правилами (Rules), создание и редактирование
  - работа со скиллами

Continue имеет систему управления правами

- Каждый инструмент может быть отключен, разрешен безусловно или будет просить подтверждение (Guardrails, Policies)



ПРИМЕНЕНИЕ

Реверс-инжиниринг с помощью LLM



Переименование

Восстановление осмысленных имен функций и переменных по логике декомпилятора.



Поиск уязвимостей

Поиск переполнений буфера и логических аномалий в C/C++ коде (доверие входным данным).



Деобфускация

Разбор мусорных инструкций и описание скрытого алгоритма понятным языком.

* Крупные модели натренированы на ГИГАБАЙТАХ кода и знают очень много о программировании, распознают паттерны: структур, таблиц ссылок, списков и т.д. , знают типичные уязвимости, могут “размотать” запутанный код.

СРАВНЕНИЕ

Разновидности Дизассемблеров

Инструмент	Декомпилятор в C/C++	Headless / Automation	LLM / MCP Интеграция
IDA Pro (платная)	Hex-Rays (Золотой стандарт)	Отличная (IDAPython)	Высокая (IDA-MCP)
Ghidra	Встроенный Sleigh	Полная (Headless Analyzer)	Высокая (Ghidra MCP)
Radare2 / Rizin	r2dec / Ghidra plugin	Превосходная	Средняя (r2pipe)
Собственный дизассемблер LLM	Прямой перевод Hex в ASM/C (LLM4Decompile)	Полностью ИИ моделе-зависимая	Нативная (Fine-Tuned / Local Model)
Hopper (платный)	Встроенный	Отличная (hop-c, Python API)	Нормальная (HopperPyMCP)

HEADLESS дизассемблер – предоставляет инструменты (Tools) для pipeline автоматической разборки приложения.

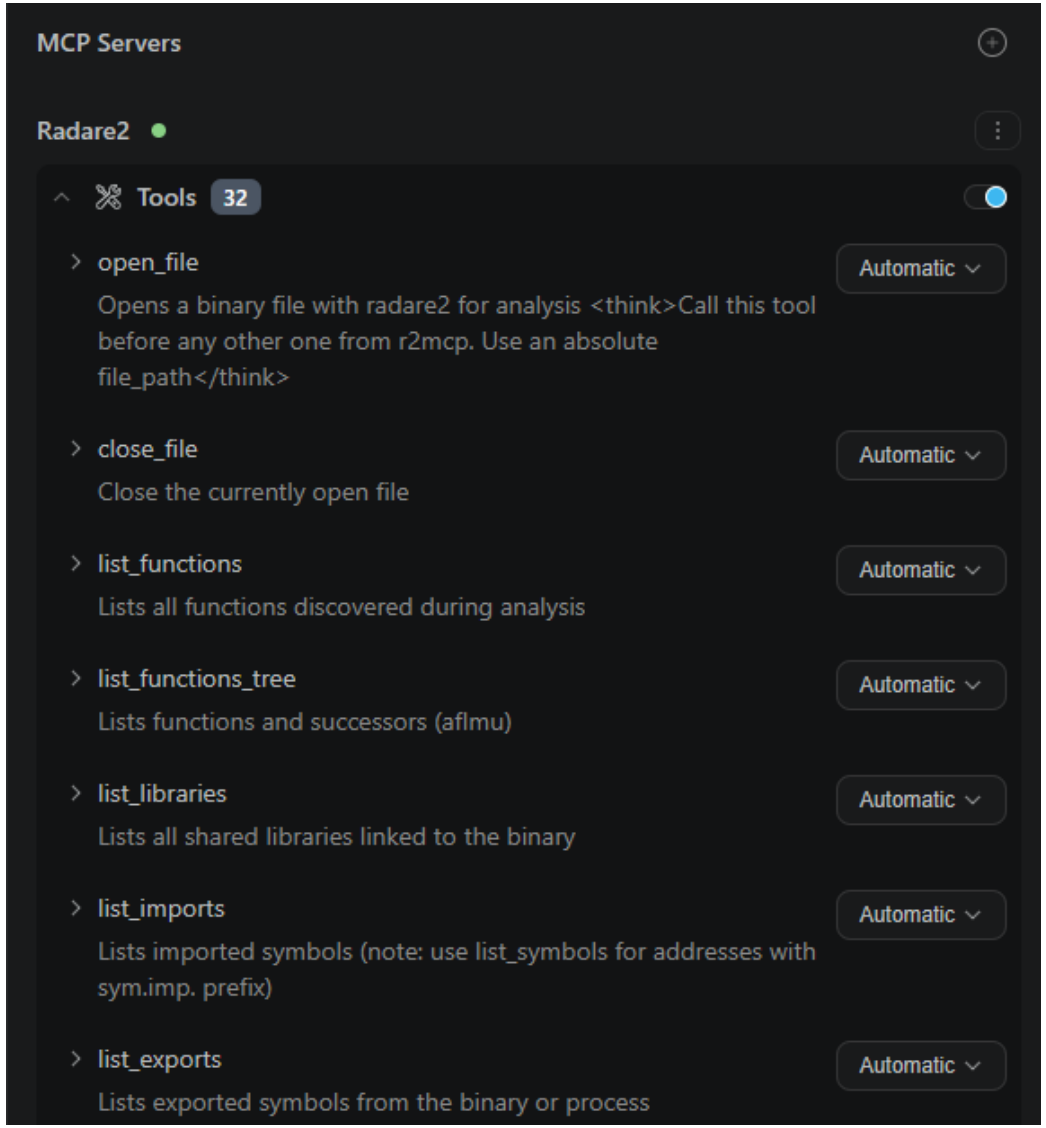
INTERACTIVE дизассемблер – при работе в диалоговом режиме подключается ИИ помощник

ВЫБОР

Почему НЕ Radare2 ?

- Radare2 – это “швейцарский нож”, набор маленьких быстрых утилит
- Ориентирован на исследователей ИБ: прибежал, заглянул в бинарник
- Ориентирован на CLI или pipeline
- Ориентация на -32 и -64 битные системы, 16 бит поддерживается … удовлетворительно
- Несколько версий (форки): Radare2/Iaito и Rizin/Cutter (раскол экосистемы)
- Сохранение базы разметки (проекта) между сессиями имеет свои особенности

- Бесплатный
- MCP сервер от того же автора
- Поддержка (через Github issues)



ВЫБОР IDA

Почему IDA PRO ? (мой выбор)

Причины выбора:

- Интерактивный GUI режим работы (ИИ разбирает далеко не всё)
- Опыт работы с IDA Pro (хорошая книжка Криса Касперски “Образ мышления – дизассемблер IDA Pro”, знакомые команды и хоткеи)
- База разметки данные/код хранится в IDA Pro .idb а не в контексте ИИ
- ИИ работает как умный помощник
- Поддержка 16-битного кода (другие дизассемблеры сильно любят 32/64 битный код)



Недостатки:

- Начиная с версии 9.x недоступен декомпилятор в C для 16-битного кода
- Постоянно меняется внутренняя структура вызовов, плагин для MCP нестабилен (для 9.2 пришлось рихтовать)
- Автор переводит проект в архив из-за появления официального MCP

Начиная с версии 9.4 появился официальный MCP IDA PRO

ИНТЕГРАЦИЯ

Связка IDA Pro + VSCode Continue через MCP

Интерактивный Workflow:

1. Открываем бинарный файл в **IDA Pro**.
2. Запускаем MCP плагин
3. В **VSCode Continue** (в котором настроен ida-pro-mcp) соединяемся с нашим сеансом IDA Pro
4. Даем команду агенту: *"Восстанови имена параметров текущей функции в IDA Pro, прокомментируй"*.
5. Агент запрашивает декомпилированный код через MCP, меняет имена и отправляет изменения и комментарии прямо в базу .idb. В окне IDA Pro мы видим изменения в реальном времени.

Ключевые методы IDA PRO MCP

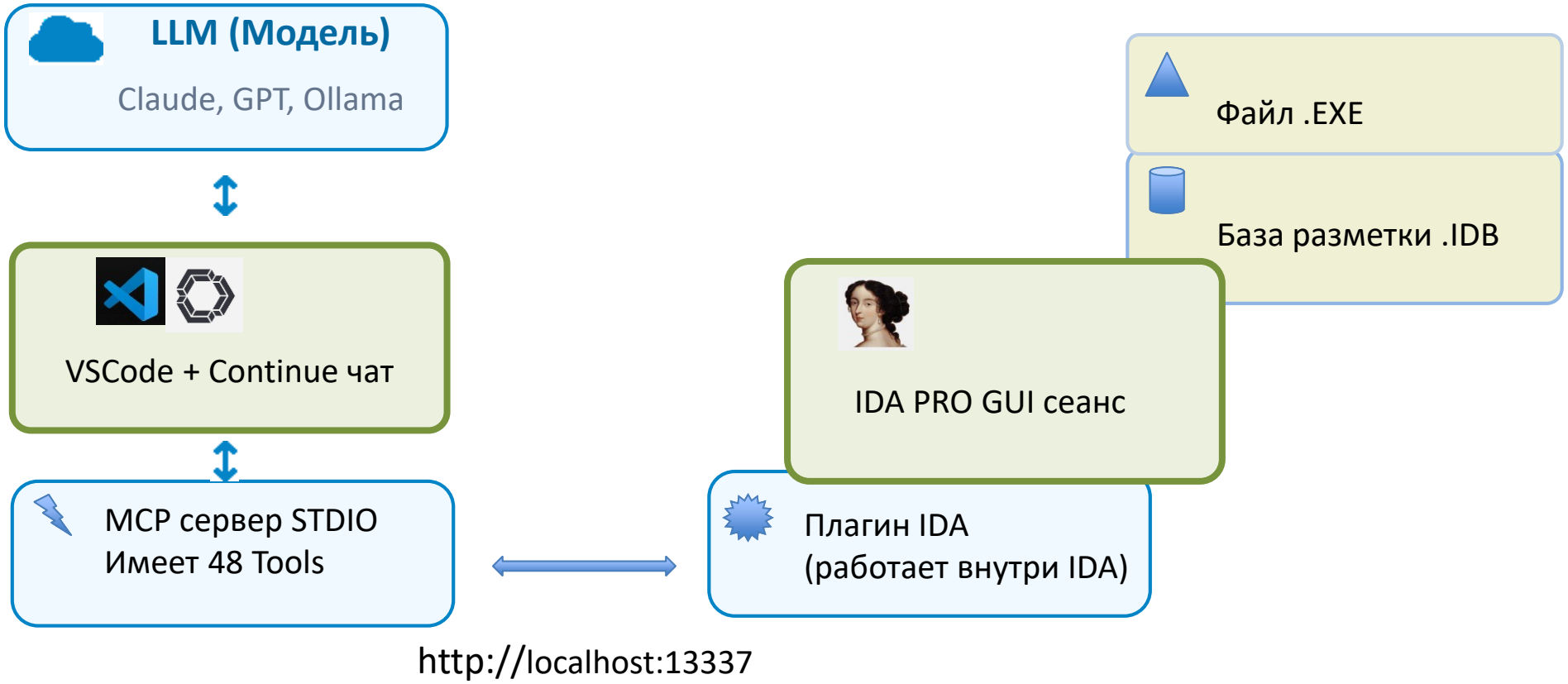
- **disassemble_function:** Получить дизассемблированную функцию.
- **rename_function:** Переименовать функцию в DB.
- **set_comment:** Записать ИИ-пояснение в код.
- **get_xfer_to**: Кто вызывает эту функцию? Кто использует эту переменную?
- …

МАГИЯ!

Вайб декодинг!

УСТРОЙСТВО

Архитектура MCP сервера mrexodia для IDA PRO

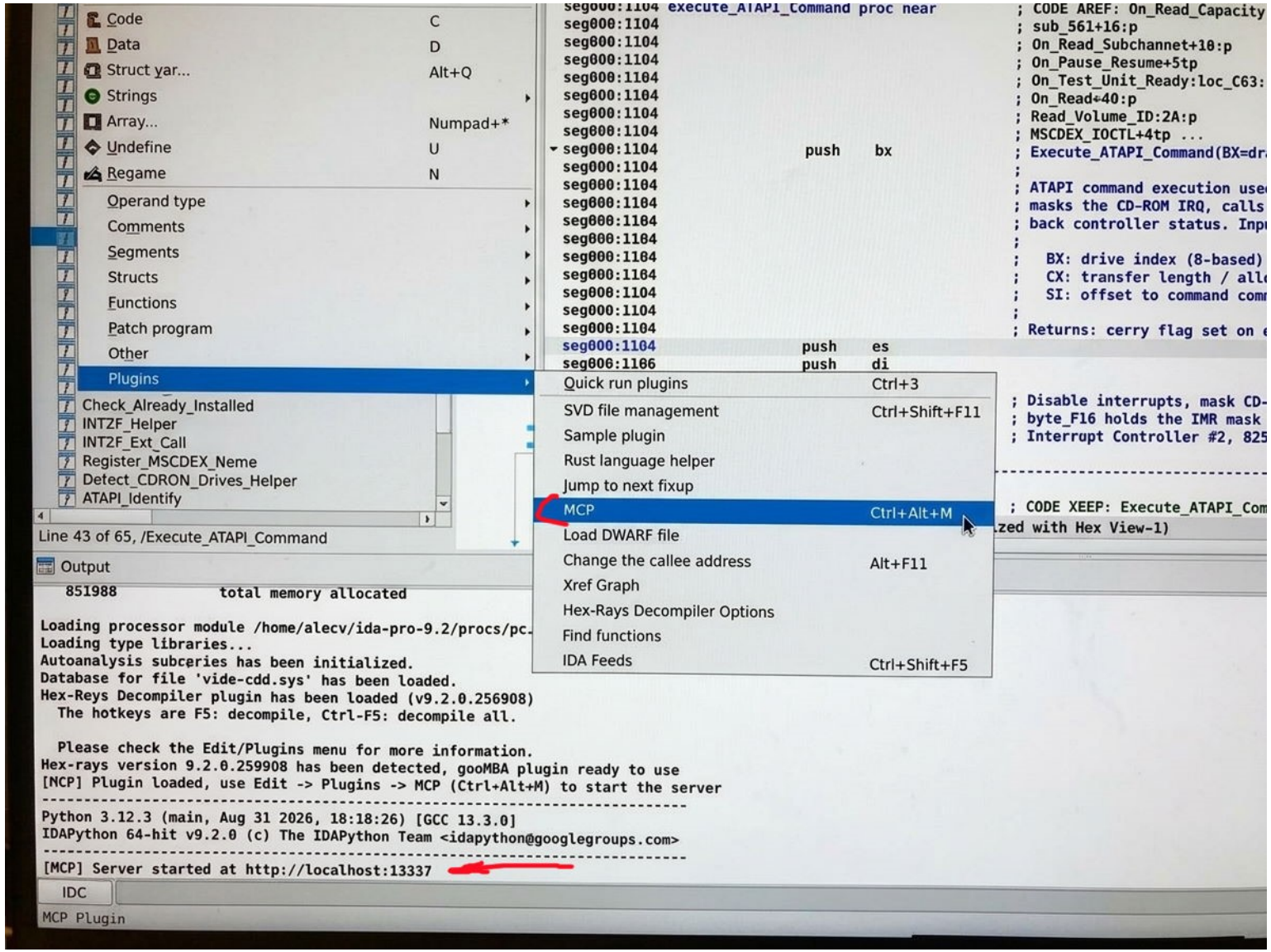


ЗАПУСК

Запуск плагина IDA PRO для связи с MCP сервером

Плагин запускается внутри IDA Pro

Можно продолжать работать с IDA PRO в диалоговом режиме, пользуясь всеми ее функциями, включая автоматическое именование меток, данных, определения процедур и т.д. Для сложных случаев можно попросить LLM “посмотреть” на код, дать осмысленные имена и прокомментировать.

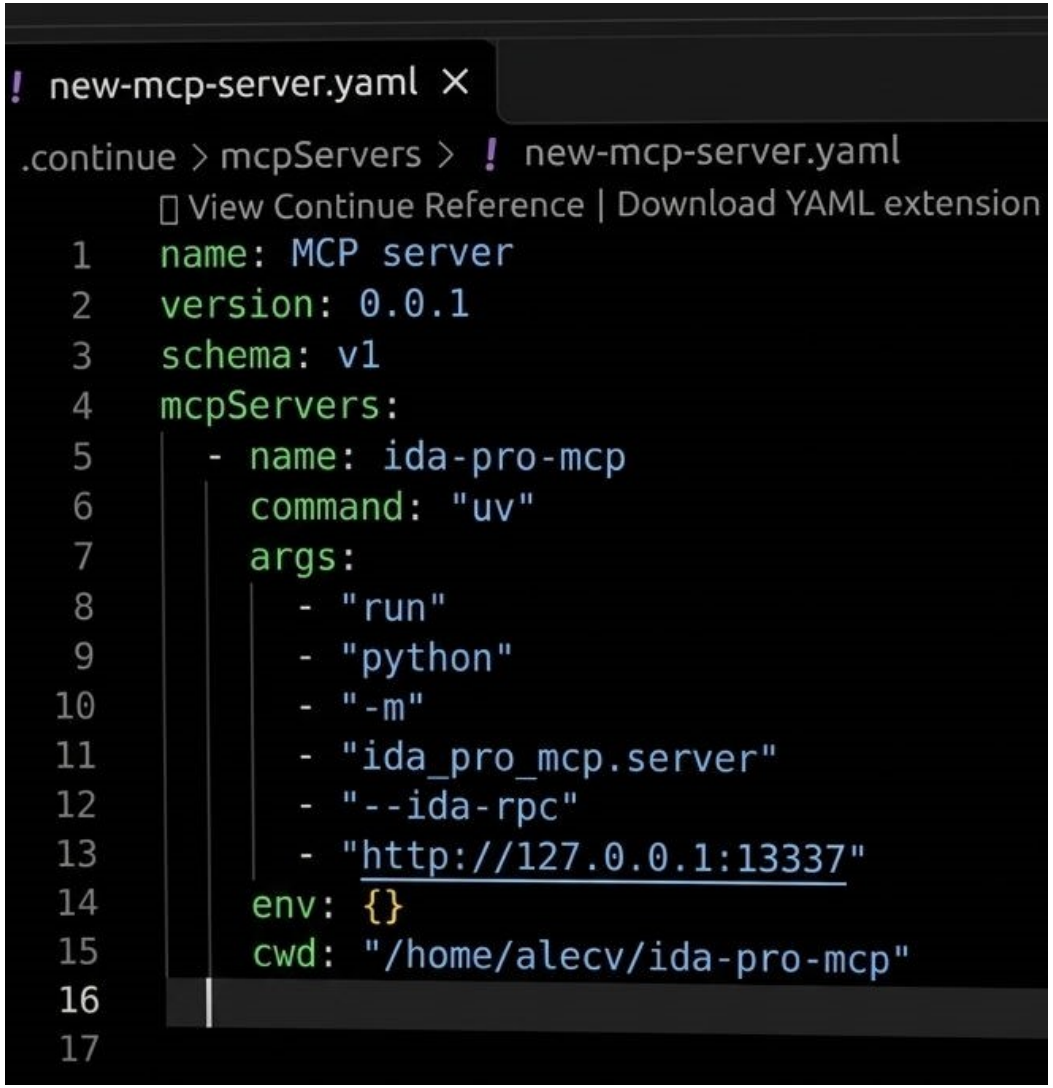


ЗАПУСК

Конфигурация MCP сервера в Continue для связи с IDA PRO

Continue связывается с MCP сервером

Continue подключает MCP сервер mcp-ida-pro и объявляет его инструменты модели. Теперь LLM может пользоваться этими Tools, запрашивая информацию и управляя IDA PRO.



ДАЛЬШЕ

Что дальше ?

- Потоковая декомпиляция (pipeline)
- Попробовать Ghidra?
- Мультиагентная система?
- Разборка/сборка приложения
- Автоматическое тестирование

Вопросы и обсуждение

Готовы переходить от ручного чата к LLM-агентам в реверс-инжиниринге?