3.5. Углублено: MCP — протокол, который «подключает» LLM к внешнему миру

Что такое Model Context Protocol

MCP (Model Context Protocol) — открытый стандарт для подключения AI-ассистентов к внешним системам. Нынешние LLM ограничены контекстом: у них нет «природного» доступа к файлам, базам данных, облачным хранилищам или пользовательским инструментам. Каждая новая интеграция требует создания отдельного коннектора. MCP решает эту проблему, предлагая единый стандарт обмена контекстом, который можно сравнить с USB-C для AI-приложений: так же, как USB-C унифицировал подключение периферии, MCP обеспечивает единообразный способ подключения ИИ-приложений к данным и инструментам[1]. Протокол позволяет LLM-модели, работающей в Claude, ChatGPT или других средах, получать доступ к локальным файлам, базам данных, поисковым системам, калькуляторам, специальным пайплайнам или специализированным промптам[1].

Возможности MCP

- **Подключение агентов к сервисам:** MCP-агенты могут получать доступ к календарю Google Calendar, файлам в Notion и другим приложениям, что позволяет строить более персонализированные ассистенты[1].
- **Глубокая интеграция с инструментами:** с помощью MCP Claude Code может генерировать целые веб-приложения по Figma-макетам; корпоративные чат-боты могут работать с несколькими базами данных; а модели могут взаимодействовать с CAD-редакторами или 3D-принтерами[1].
- **Быстрая разработка:** разработчикам не нужно писать отдельные плагины под каждую систему: MCP сокращает время и сложность интеграции, предоставляя стандартные API и SDK, а конечные пользователи получают более функциональные ассистенты[1].

Архитектура протокола

MCP построен по схеме «клиент–сервер». В ней участвуют три роли[1]:

- **MCP Host (хост):** AI-приложение, управляющее одним или несколькими MCP-клиентами (например, Claude Code или десктоп-приложение Claude).
- **MCP Client (клиент):** компонент, который устанавливает соединение с MCP-сервером и передаёт от него контекст хосту[1].

- **MCP Server (сервер):** программа, предоставляющая контекст (файлы, инструменты, ресурсы, промпты) MCP-клиентам. Серверы могут быть локальными (встроены в приложение, используют поток STDIO) или удалёнными (поднимаются отдельно и общаются по HTTP)[1].

Каждый MCP-клиент поддерживает единственное соединение со своим MCP-сервером. Если приложение (host) подключается к нескольким серверам, оно создаёт отдельный клиент для каждого, поддерживая отношения «один-к-одному»[1]. Серверы могут работать локально (например, файловый сервер в Claude Desktop), либо удалённо — тогда используется HTTP-транспорт с двухсторонней передачей данных[1].

Слоистая модель

Протокол состоит из двух слоёв[1]:

Слой	Назначение
Data layer (данные)	Использует JSON-RPC 2.0 для обмена сообщениями между клиентом и сервером. Слой определяет примитивы (`tools`, `resources`, `prompts`, `notifications`) и управляет жизненным циклом соединения: инициализация, согласование возможностей и завершение. Сервер может предоставлять инструменты для выполнения действий LLM, ресурсы (файлы, базы данных), шаблоны промптов для общения и отправлять уведомления (например, об окончании долгой операции)[1].
Transport layer (транспорт)	Отвечает за передачу сообщений и аутентификацию. MCP поддерживает два механизма: `stdio`-транспорт (обмен через стандартные вход/выход для локальных процессов) и `Streamable HTTP` — современный HTTP-транспорт, заменивший SSE. HTTP-транспорт позволяет устанавливать удалённые соединения, поддерживает заголовки (Bearer-token, API-ключи, OAuth) и потоковую передачу данных с помощью Server Sent Events[1].

MCP — протокол с сохранением состояния (*stateful*), поэтому он предусматривает управление жизненным циклом соединений и согласование поддерживаемых возможностей. Клиенты и серверы обмениваются запросами и ответами через JSON-RPC 2.0; уведомления используются, когда ответ не нужен[1].

Примитивы протокола

Примитивы MCP определяют, какие типы контекста могут передаваться[1]:

- **Tools (инструменты):** позволяют инициировать действия. Например, сервер может предоставить функции чтения или записи файлов, выполнения SQL-запросов или отправки сообщений в Slack.
- **Resources (ресурсы):** данные, которые можно передать LLM (файлы, записи базы, изображения).
- **Prompts (промпты):** шаблоны взаимодействия, которые подсказывают LLM, как использовать предоставленные ресурсы и инструменты.

- **Notifications (уведомления):** позволяют серверу отправлять клиенту информацию о ходе выполнения длительных операций или изменениях (например, об обновлении списка файлов).

Дополнительные возможности включают вызовы `sampling/elicitation` (когда сервер просит модель сэмплировать текст) и другие вспомогательные операции.

Реализации MCP-серверов

Открытый репозиторий `modelcontextprotocol/servers` содержит ссылки на эталонные и сторонние реализации MCP-серверов и на SDK на разных языках. В README перечислены официальные SDK: C#, Go, Java, Kotlin, PHP, Python, Ruby, Rust, Swift и TypeScript[2]. Базовые серверы демонстрируют возможности MCP: `Everything` (тестовый сервер со множеством инструментов), `Fetch` (загрузка и конвертация веб-контента для LLM), `Filesystem` (работа с файлами с контролем доступа), `Git` (чтение и модификация репозиториев), `Memory` (граф знаний с долгосрочной памятью), `Sequential Thinking` (пошаговое решение задач) и `Time` (конвертация времени и часовых поясов)[2]. Многие дополнительные серверы находятся в архиве — среди них интеграции с GitHub, Slack, Google Drive, Puppeteer и PostgreSQL[2].

Также растёт экосистема сторонних серверов. Третьи компании предлагают готовые MCP-интеграции: `21st.dev` предоставляет генерацию UI-компонентов, `2slides` — генерацию презентаций, `Paragon ActionKit` подключает 130+ SaaS-интеграций (Slack, Salesforce, Gmail), `Adfin` работает с платёжными сервисами, `AgentOps` обеспечивает трассировку и наблюдаемость для AI-агентов, а `AgentQL` помогает получать структурированные данные из неструктурированного веба[2]. Благодаря стандарту разработчики могут выбирать готовые серверы или создавать свои, используя SDK.

Кейсы и эволюция протокола

Интеграция и опыт Anthropic

Новости Anthropic об анонсе MCP подчёркивают, что модели сегодня изолированы от данных, а MCP выступает «универсальным интерфейсом» для подключения LLM к источникам информации. В релизе от ноября 2024 года компания представила протокол, поддерживающий SDK, локальные серверы в приложении Claude Desktop и каталог открытых серверов, включая Google Drive, Slack, GitHub, Git и Postgres[3]. Ранними партнёрами стали Block, Apollo и компании-разработчики инструментов (Zed, Replit, Codeium, Sourcegraph), которые используют MCP, чтобы их агенты могли получать контекст (исходный код, сообщения, задачи) для более качественного выполнения задач[3]. Anthropic видит MCP как открытый проект и приглашает разработчиков создавать новые коннекторы и инструменты[3].

Опыт Hugging Face

Hugging Face запустила собственный MCP-сервер в июле 2025 года и описала своё путешествие по его созданию. Статья отмечает, что MCP быстро развивается — за 9 месяцев после запуска вышло три ревизии протокола, в которых SSE-транспорт был заменён на `Streamable HTTP`, а также переработаны механизмы авторизации[4]. При разработке сервер должен выбрать, как клиенты будут подключаться к нему: доступны STDIO, HTTP с Server Sent Events (SSE) и Streamable HTTP[4]. STDIO используется для локальных серверов; HTTP + SSE — для удалённых соединений, однако он устарел; `Streamable HTTP` — современный

транспорт, обеспечивающий потоковую передачу с возможностью отправлять обновления (Server Push) и запросы прогресса[4]. В зависимости от выбранного транспорта сервер может поддерживать три схемы коммуникации[4]:

- **Direct Response:** стандартный запрос-ответ, подходит для простых stateless-операций.
- **Request-Scoped Streams:** временные потоковые соединения для одной операции (например, генерация видео) с возможностью отправлять прогресс и получать дополнительные данные с помощью `Sampling` и `Elicitation` запросов[4].
- **Server Push Streams:** долгоживущие соединения, позволяющие серверу инициировать сообщения (обновление ресурсов, изменение списка инструментов). Они требуют механизма поддержания связи и восстановления после разрыва[4].

Разработчики Hugging Face также выделяют выбор между *стейтлес-сервером* (каждый запрос независим) и *стейтфул-сервером* (сервер поддерживает состояние и сессионный идентификатор), что влияет на масштабируемость и поддержку повторных соединений[4]. В заключении они отмечают, что протокол уже позволил интегрировать Hugging Face Hub и Gradio Spaces, и приводят примеры применения MCP-сервера: оркестрация видеопроизводства, редактирование изображений, поиск документов и создание AI-приложений[4].

Практические рекомендации и выводы

- **Строим экосистему:** MCP — инициативный open-source проект, поэтому компании и разработчики могут вносить вклад, создавая сервера и коннекторы для своих систем. Это сокращает время интеграции и даёт LLM-агентам доступ ко внутренним данным.
- **Выбор транспорта и состояния:** для локальных интеграций достаточно STDIO, но для облачных сервисов удобнее Streamable HTTP; стоит решить, нужен ли stateful сервер (сессии и восстановление) или stateless (простая масштабируемость).
- **Использование готовых серверов:** прежде чем писать свой сервер, стоит проверить список reference- и third-party-серверов (Git, Filesystem, Memory, Time, Slack, GitHub, Google Drive, PostgreSQL и др.) — это может покрыть вашу задачу.
- **Безопасность и аутентификация:** MCP поддерживает стандартные механизмы аутентификации (Bearer-tokens, OAuth), поэтому необходимо обеспечить безопасное хранение токенов и контроль доступа.
- **Постоянное обновление:** протокол активно развивается, и клиенты должны следить за версиями и поддержкой функций в SDK. Некоторые транспортные механизмы устаревают (например, SSE), их необходимо заменить на актуальные.

MCP уже поддерживается многими инструментами (Claude Code, IDE, корпоративные агенты) и, благодаря универсальному формату подключения, становится фундаментом для LLM-агентов, которые способны не только генерировать текст, но и действовать в реальном мире.

Источники

[1] Источник: modelcontextprotocol.io
[2] Источник: raw.githubusercontent.com
[3] Источник: anthropic.com
[4] Источник: huggingface.co