

СПИСОК ЛИТЕРАТУРЫ

- 1 **Флэнаган, Д.** JavaScript. Полное руководство / Д. Флэнаган. – М. : Вильямс, 2021. – URL: <https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/> (oreilly.com in Bing) (дата обращения: 20.05.2026).
- 2 **Чиннатхамби, К.** Изучаем React / К. Чиннатхамби. – СПб. : Питер, 2019. – URL: <https://www.pearson.com/en-us/subject-catalog/p/learning-react/P2000000003465/9780134843575> (pearson.com in Bing) (дата обращения: 20.05.2026).
- 3 **Бернхардт, К.** WebAssembly: практическое руководство / К. Бернхардт. – СПб. : Питер, 2022. – URL: <https://www.cs.cmu.edu/~wasm/> (дата обращения: 20.05.2026).
- 4 **Гавнер, М.** Изучаем TypeScript 4 / М. Гавнер. – М. : ДМК Пресс, 2021. – URL: <https://link.springer.com/book/10.1007/978-1-4842-7180-0> (link.springer.com in Bing) (дата обращения: 20.05.2026).
- 5 Аналитический отчёт: Рынок труда в сфере IT. – М. : Хабр Карьера, 2025. – URL: <https://habr.com/ru/articles/810923/> (habr.com in Bing) (дата обращения: 20.05.2026).

Получено 17.06.2026

ISSN 2227-1155. Сборник студенческих научных работ.
Вып. 31. Гомель, 2026

УДК 004.8:004.91

Д. В. ЯКУБОВИЧ, И. С. КУХАРЕНКО (МТ-56)

Научный руководитель – канд. физ.-мат. наук *Н. В. РЯЗАНЦЕВА*

ИНТЕГРАЦИЯ ИНСТРУМЕНТОВ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В OBSIDIAN

Рассматривается концепция интеграции современных ИИ-агентов в систему управления знаниями Obsidian посредством протокола MCP (Model Context Protocol). Анализируются ключевые проблемы применения больших языковых моделей в долгосрочных проектах, предлагается архитектурное решение на основе структурированной базы знаний в формате Markdown. Исследуется феномен «context rot» – деградации внимания модели при переполнении контекстного окна – и обосновывается преимущество вынесения проектных знаний во внешнее хранилище. Приводятся практические рекомендации по организации базы знаний, настройке интеграции и применению подхода в командной разработке.

Большие языковые модели (LLM) стремительно проникают в повседневную практику разработчиков, исследователей и студентов. Инструменты на базе Claude, GPT и их аналогов позволяют генерировать код, анализи-

ровать документацию и автоматизировать рутинные задачи. Тем не менее при использовании ИИ-ассистентов в проектах, растянутых во времени, пользователи неизбежно сталкиваются с рядом системных ограничений.

Первое ограничение – конечность контекстного окна. Несмотря на то что к 2025–2026 годам ряд моделей поддерживает окна свыше 100 000 токенов, эффективный рабочий контекст на сложных задачах может быть значительно меньше декларируемого предела. Исследования 2025 года зафиксировали явление «context rot» – постепенной деградации внимания модели к ранним токенам по мере заполнения контекстного окна: модель начинает отдавать предпочтение свежим записям, тогда как информация из начала диалога фактически теряется.

Второе ограничение – отсутствие персистентной памяти между сессиями. Каждый новый диалог начинается «с чистого листа»: разработчик вынужден заново воспроизводить контекст, объяснять архитектуру проекта, соглашения о коде и принятые решения. На крупных проектах это превращается в значительные временные потери.

Третье ограничение носит структурный характер: линейный формат чата плохо приспособлен для хранения и навигации по многокомпонентной проектной документации. Описанные проблемы формируют запрос на новый подход: вместо накопления контекста внутри диалога – вынесение проектных знаний во внешнее, структурированное, версионизируемое хранилище. Именно такую возможность открывает интеграция ИИ-агентов с Obsidian через протокол MCP.

Obsidian – кроссплатформенное приложение для ведения заметок с открытым форматом хранения. Все данные сохраняются в виде обычных текстовых файлов формата Markdown (.md) в локальной папке – хранилище (vault). Принципиальное отличие Obsidian от облачных аналогов (Notion, Confluence) состоит в том, что данные не покидают устройство пользователя без его явного согласия, а формат хранения не привязан к конкретному поставщику сервисов. К февралю 2026 года аудитория Obsidian превысила 1,5 миллиона пользователей с приростом около 22 % год к году, что во многом объясняется совместимостью приложения с ИИ-инструментами: формат Markdown читается и генерируется языковыми моделями нативно, без специальных парсеров.

Ключевой особенностью Obsidian является поддержка двунаправленных ссылок между заметками. Конструкция [[имя файла]] автоматически создаёт гиперссылку и отображается в граф-представлении хранилища, позволяя строить семантически связанные сети знаний. Именно эта особенность критически важна для ИИ-агентов: вместо анализа всей кодовой базы при каждом запросе агент движется по ссылкам от корневого файла к нужным разделам документации. Расширяемость обеспечивается системой плагинов сообщества: для интеграции с ИИ ключевую роль играет плагин Local REST

API, открывающий локальный HTTP-интерфейс, через который внешние программы, в том числе MCP-серверы, могут читать, создавать и редактировать файлы хранилища.

Model Context Protocol (MCP) – открытый стандарт межпрограммного взаимодействия, разработанный компанией Anthropic в конце 2024 года. Протокол определяет унифицированный способ подключения ИИ-агентов к внешним инструментам, базам данных и API. Ключевое достоинство MCP – универсальность: один MCP-сервер может быть использован любым агентом, поддерживающим стандарт, независимо от поставщика модели. Архитектурно MCP работает по клиент-серверной модели: агент направляет структурированные запросы MCP-серверу, который транслирует их в вызовы конкретного сервиса. В случае Obsidian цепочка выглядит следующим образом: Claude Code (агент) → MCP-сервер → Local REST API → файловая система хранилища.

Важно отметить отличие данного подхода от классического RAG (Retrieval-Augmented Generation). RAG предполагает векторизацию документов и семантический поиск по эмбедингам; MCP обеспечивает прямой структурированный доступ к хранилищу по ссылкам и путям. Сравнительный анализ подходов приведён в таблице 1.

Таблица 1 – Сравнение подходов к расширению контекста ИИ-агентов

Критерий	Контекстное окно	RAG	MCP + Obsidian
Персистентность	Нет	Частичная	Да
Стоимость токенов	Высокая	Средняя	Низкая
Навигация по ссылкам	Нет	Нет	Да
Актуализация агентом	Нет	Нет	Да
Командная работа (Git)	Нет	Ограниченно	Да
Конфиденциальность	Зависит от модели	Зависит от хранилища	Локально

Исследования 2025 года подтверждают: на практике RAG и расширенные контекстные окна решают разные задачи и не исключают друг друга. RAG эффективен для поиска по большим документальным корпусам; контекстное окно – для анализа одного документа целиком; MCP с Obsidian занимает промежуточную нишу: структурированный граф небольших взаимосвязанных документов, навигируемый агентом по ссылкам.

Эффективность работы ИИ-агента напрямую определяется качеством организации базы знаний. На основе анализа практического опыта можно выделить три ключевых требования к её структуре.

Навигируемость. От корневого файла README.md должна существовать цепочка ссылок до любого дочернего документа. Агент начинает работу именно с корневого файла и движется вглубь по ссылкам, не перебирая все файлы хранилища методом полного перебора.

Актуальность. Содержимое технической документации должно точно отражать текущее состояние кодовой базы. Устаревшая документация вводит агента в заблуждение и порождает ошибки, поэтому обновление базы знаний должно быть обязательным шагом при каждом внесении изменений в код.

Лаконичность. Каждый файл должен содержать только существенную для своего раздела информацию. Практика показывает, что чрезмерно разросшиеся файлы снижают качество ответов агента, поэтому рекомендуется периодически дробить большие файлы на отдельные тематические документы.

Типовая структура хранилища для проекта средней сложности включает: README.md – корневой файл с навигационными ссылками на все разделы и инструкциями для агентов; context/ – контекстную информацию о проекте (техническое задание, требования заказчика, принятые архитектурные решения); structure/ – техническую документацию, структурно зеркалирующую кодовую базу; daily-changes/ – журнал изменений в формате «одна запись на день», который агент ведёт самостоятельно, что позволяет восстанавливать контекст при переключении между задачами.

Важным практическим приёмом является включение явных инструкций для ИИ-агента непосредственно в документы базы знаний: каждый индексный файл рекомендуется завершать разделом «Для агентов» с описанием допустимых операций, форматов записей и ограничений. Кроме того, в корне проекта создаётся файл CLAUDE.md (или AGENTS.md для других инструментов), который автоматически включается в контекст каждого запроса и содержит общий промпт, описывающий роль агента и порядок работы с хранилищем.

Типовой цикл взаимодействия при постановке задачи выглядит следующим образом. Разработчик формулирует задачу (например, «реализовать форму авторизации и соответствующий эндпоинт»). Агент открывает README.md хранилища, по навигационной карте определяет релевантные разделы и по цепочке ссылок считывает необходимую документацию – без анализа всей кодовой базы. Затем агент вносит изменения в код, соблюдая принятые в проекте соглашения, актуализирует техническую документацию и добавляет запись в журнал изменений. Контекст задачи оказывается зафиксированным в базе знаний для последующих сессий.

Описанный подход органично встраивается в командную разработку: база знаний выносится в отдельный Git-репозиторий, что обеспечивает версионирование и прозрачную историю изменений документации. Каждый

участник команды использует ИИ-агента в рамках единого согласованного контекста. Практически значимым следствием является ускорение онбординга новых разработчиков: вместо изучения кодовой базы напрямую новый участник задаёт вопросы агенту, который отвечает на основе актуальной документации.

Интеграция с ИИ-агентами придаёт новое измерение концепции «второго мозга» – персональной внешней системы управления знаниями, восходящей к методологии Zettelkasten Никласа Лумана и её современным интерпретациям. Хранилище знаний становится не только инструментом памяти для человека, но и рабочей средой для ИИ: вместо объяснения ситуации в каждом новом диалоге разработчик предоставляет агенту доступ к структурированной базе знаний, которую агент читает, дополняет и использует как рабочую память. Выбор формата Markdown оказался дальновидным задолго до эры LLM: текстовые файлы – наиболее универсальный формат, читаемый любой моделью.

К преимуществам подхода относятся: решение проблемы context rot за счёт загрузки знаний только по мере необходимости; кратная экономия токенов в крупных репозиториях; персистентная память, не привязанная к конкретному поставщику ИИ; командная совместимость через Git; «живая» документация, снижающая затраты на онбординг; конфиденциальность благодаря локальному хранению данных и возможности работы с локальными моделями (Ollama, LM Studio).

Вместе с тем подход имеет ограничения. Эффективность агента прямо зависит от качества документации: недостаточно строгий контроль над записями приводит к накоплению неточных данных, вводящих агента в заблуждение. Проектирование структуры хранилища и первоначальное наполнение базы знаний требуют значительного времени, что делает подход нецелесообразным для небольших краткосрочных проектов. При командной работе синхронизация через Git требует внимания к разрешению конфликтов, а некоторые MCP-серверы создают инфраструктурную зависимость, требуя запущенного приложения Obsidian в момент работы агента.

В заключение следует отметить, что интеграция инструментов искусственного интеллекта в Obsidian через протокол MCP представляет собой практически реализуемый ответ на системные ограничения современных ИИ-ассистентов: конечность и деградацию контекстного окна, отсутствие персистентной памяти между сессиями и сложность командного взаимодействия с агентом. Центральная идея – разделение контекста и диалога: проектные знания выносятся во внешнее структурированное хранилище, к которому агент обращается по навигации по ссылкам. Направлениями дальнейших исследований являются разработка метрик качества ИИ-генерируемой документации, автоматизация первоначального наполнения базы знаний и изучение применимости архитектуры в образовательном процессе.

СПИСОК ЛИТЕРАТУРЫ

- 1 Obsidian – официальная документация // Obsidian. – URL: <https://obsidian.md/> (дата обращения: 01.05.2026).
- 2 Model Context Protocol – спецификация протокола // Anthropic. – URL: <https://modelcontextprotocol.io/> (date of access: 01.06.2026).
- 3 Разработка с Obsidian + Claude. Практический гайд // Habr. – URL: <https://habr.com/ru/articles/1030316/> (дата обращения: 01.05.2026).
- 4 LLM Context Window Limitations in 2026 // Atlan. – URL: <https://atlan.com/know/llm-context-window-limitations/> (date of access: 01.06.2026).
- 5 Mastering Personal Knowledge Management with Obsidian and AI // Eric J. Ma. – URL: <https://ericmj1.github.io/blog/2026/3/6/mastering-personal-knowledge-management-with-obsidian-and-ai/> (date of access: 01.06.2026).
- 6 Obsidian AI Second Brain: Complete Guide 2026 // NxCode. – URL: <https://www.nxcode.io/resources/news/obsidian-ai-second-brain-complete-guide-2026> (date of access: 01.06.2026).
- 7 **Forte, T.** Building a Second Brain: A Proven Method to Organize Your Digital Life and Unlock Your Creative Potential / T. Forte. – New York : Atria Books, 2022. – 272 p.
- 8 From RAG to Context: A 2025 year-end review of RAG // RAGFlow. – URL: <https://ragflow.io/blog/rag-review-2025-from-rag-to-context> (date of access: 01.05.2026).

Получено 26.06.2026

ISSN 2227-1155. Сборник студенческих научных работ.
Вып. 31. Гомель, 2026

UDC 339.138.8:745.5

Y. M. YUNCHYTS (ГЧ-21)

Research Supervisor – E. L. BATURINA, PhD

COLOR AS A BRANDING TOOL = ЦВЕТ КАК ИНСТРУМЕНТ БРЕНДИНГА

This article explains modern trends in how companies use color in branding, the choices brand make, and facts about how people react to color that are often overlooked. It examines the psychological reasons for people's color responses and how brands can use these reactions, when creating their visual identity and marketing.

Color is a key tool in branding. It helps brands influence people's feelings, how they see the company, and their desire to buy something from this brand. It makes brand memorable and unique in people's minds. This psychological impact strengthens brand identity.

Historical symbolism of color influences how we read shades today. Michel Pastoureau shows that colors build their meanings over centuries. They are not