АРХИТЕКТУРА ИИ-АГЕНТА С ИСПОЛЬЗОВАНИЕМ MCP И RAG ДЛЯ УПРАВЛЕНИЯ ПРОДУКТИВНОСТЬЮ

Хорошун Н.А., студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Куприянова Д.В. – маг. техн. наук, старший преподаватель

В работе рассматривается проектирование архитектуры интеллектуального ИИ-агента, объединяющего протокол Model Context Protocol (MCP) и метод Retrieval-Augmented Generation (RAG) для персонализированного управления продуктивностью пользователя. Описана реализация системы на базе микросервисной архитектуры с применением векторного поиска pgvector и LLM Gemini.

Современные системы управления продуктивностью, такие как трекеры времени и менеджеры задач, ограничиваются сбором количественных метрик: длительность сессий, количество выполненных задач, процент фокусного времени. Однако они не способны учитывать качественный контекст пользовательской деятельности – содержание рефлексивных заметок, историю взаимодействия с ИИ-ассистентом, индивидуальные паттерны продуктивности. Традиционные чат-боты на основе больших языковых моделей (LLM) работают в рамках ограниченного контекстного окна и не имеют доступа к долговременной памяти о пользователе, что снижает релевантность генерируемых рекомендаций [1]. В связи с этим актуальной задачей является проектирование архитектуры ИИ-агента, способного объединять структурированные и неструктурированные данные пользователя для формирования персонализированных рекомендаций в реальном времени.

В основу предлагаемого решения положены два ключевых подхода: Retrieval-Augmented Generation (RAG) и Model Context Protocol (MCP). RAG – это метод расширения возможностей LLM путём извлечения релевантных фрагментов из внешней базы знаний перед генерацией ответа [2]. MCP – открытый протокол, стандартизирующий взаимодействие LLM с внешними источниками данных и инструментами через унифицированный интерфейс «клиент–сервер», что позволяет агенту динамически подключать новые возможности без модификации ядра системы [3].

Серверная часть состоит из набора микросервисов, взаимодействующих через REST API и брокер сообщений Redpanda. Данные о телеметрии использования приложений собираются клиентом и передаются через потоковую обработку в аналитическое хранилище ClickHouse для агрегаций.

ИИ-агент реализован как отдельный сервис и использует Gemini API в качестве LLM. Подсистема RAG построена на расширении pgvector [4] для PostgreSQL, что позволяет хранить 768-мерные векторные представления непосредственно в реляционной СУБД без необходимости развертывания отдельной векторной базы данных. Для индексации используется алгоритм HNSW (Hierarchical Navigable Small World), обеспечивающий поиск ближайших соседей со сложностью O(log n). При поступлении запроса пользователя система выполняет семантический поиск по трём категориям источников: рефлексивные заметки из сессий фокусировки, описания задач и встроенная база знаний о методологиях продуктивности. Найденные фрагменты ранжируются по косинусному сходству и включаются в системный промпт LLM, что позволяет генерировать ответы с учётом индивидуального контекста пользователя.

Интеграция MCP обеспечивает модульное расширение возможностей агента. Через MCP-серверы агент получает доступ к инструментам управления задачами, запуска фокус-сессий, просмотра аналитики и синхронизации с Google Calendar. Каждый MCP-сервер предоставляет стандартизированное описание своих возможностей, что позволяет LLM автономно выбирать необходимый инструмент на основе намерения пользователя. Такая архитектура позволяет добавлять новые интеграции без изменения логики самого агента.

Таким образом, предложенная архитектура ИИ-агента обеспечивает три ключевых результата: семантический доступ к долговременной памяти пользователя посредством RAG устраняет ограничение контекстного окна LLM; стандартизация инструментного взаимодействия через MCP обеспечивает масштабируемость без модификации ядра агента; совмещение реактивного и проактивного режимов позволяет системе как отвечать на запросы, так и самостоятельно инициировать рекомендации на основе анализа поведенческих паттернов пользователя.

Список использованных источников:

1. Zhao W. X. A Survey of Large Language Models / W. X. Zhao [et al.] // arXiv preprint arXiv:2303.18223. – 2023.

2. Lewis P. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / P. Lewis [et al.] // Advances in Neural Information Processing Systems. – 2020. – Vol. 33. – P. 9459–9474.

3. Model Context Protocol Specification [Электронный ресурс]. – Режим доступа: https://modelcontextprotocol.io. – Дата доступа: 25.03.2026.

4. pgvector: Open-source vector similarity search for Postgres [Электронный ресурс]. – Режим доступа: https://github.com/pgvector/pgvector. – Дата доступа: 25.03.2026.