

АРХИТЕКТУРА И РЕАЛИЗАЦИЯ ПРОГРАММНОГО КОМПЛЕКСА ПОМОЩНИКА СТУДЕНТА С ИСПОЛЬЗОВАНИЕМ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

В. В. Кухаренко

Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», Республика Беларусь

Научный руководитель О. Д. Асенчик

Рассмотрены архитектурные и программные решения, реализованные в комплексе поддержки студентов на основе больших языковых моделей. Комплекс предназначен для сопровождения лабораторных работ по IT-дисциплинам и объединяет управление курсами, прием и интеллектуальный анализ отчетов и программного кода, генерацию персонализированных учебных материалов и формирование проверочных тестов.

Ключевые слова: большие языковые модели, LLMs, персонализированное обучение, программный комплекс, автоматическая проверка работ, анализ программного кода, генерация учебных материалов, адаптивное тестирование.

Активно развивающиеся большие языковые модели (LLMs) в последнее время рассматриваются не только как средство диалогового взаимодействия, но и как инструмент подготовки учебного контента [1]. Применение больших языковых моделей при создании материалов по учебным дисциплинам может обеспечивать высокое качество. Данный вывод важен для разработки образовательных программных систем, поскольку подтверждает возможность включения больших языковых моделей в методически контролируемый контур обучения. Одновременно обзор современных исследований фиксирует значительный потенциал таких моделей в персонализации образования и указывает на необходимость их интеграции в системы с понятной архитектурой, прозрачными ограничениями и механизмами контроля качества [2].

Для дисциплин IT-профиля особое значение имеет оперативная и содержательная обратная связь по лабораторным работам, отчетам и программному коду. При традиционной организации учебного процесса такая обратная связь формируется с задержкой и зависит от текущей загрузки преподавателя. Существующие интеллектуальные сервисы, например платформа *Khanmigo* от *Khan Academy*, демонстрируют полезность языковых моделей в учебном диалоге, однако они не решают задачу комплексного сопровождения курса, включающего управление заданиями, прием студенческих материалов, автоматический анализ результатов и формирование индивидуальных рекомендаций в единой информационной среде [3].

В связи с этим целью работы является разработка программного комплекса помощника студента, объединяющего анализ представленных работ, генерацию компенсирующих учебных материалов и формирование проверочных тестов на основе больших языковых моделей.

Разработанный программный комплекс реализован как веб-приложение с ролевой моделью доступа и построен по принципам многослойной архитектуры. В состав системы входят доменный слой, фиксирующий основные сущности учебного процесса, прикладной слой с бизнес-логикой и сценариями обработки, инфраструктурный слой, обеспечивающий доступ к данным и интеграцию с внешними сервисами, а также слой представления, реализующий пользовательский интерфейс. Такое разделение позволило локализовать ответственность компонентов и упростить мо-

дификацию отдельных подсистем без изменения всей системы. На уровне доменной модели выделены сущности курса, лабораторной работы, студенческой сдачи, результата анализа, результата проверки кода, набора тестов, конфигурации языковой модели и шаблонов запросов. Для пользователей предусмотрены роли студента, преподавателя и администратора, что обеспечивает разграничение операций создания курсов, настройки моделей, отправки работ и просмотра результатов.

Функционирование комплекса организовано вокруг асинхронного конвейера обработки работы студента. Схема процесса приведена на рис. 1.



Рис. 1. Схема обработки работы студента

После загрузки файла или указания адреса репозитория создается запись о сдаче, а материал помещается в очередь фоновой обработки. Если студент отправляет отчет в формате *TXT*, *DOCX* или *PDF*, система автоматически выбирает соответствующий модуль извлечения текста. Далее фоновая служба последовательно выполняет четыре этапа. На первом этапе формируется рецензия на отчет по шаблонам запросов и вычисляется оценка за содержание работы. На втором этапе, если студент предоставил адрес репозитория *GitHub*, выполняется проверка программного кода с использованием внешнего инструмента анализа *OpenCode*, а ее результат сохраняется отдельно. На третьем этапе по совокупности замечаний автоматически генерируются дополнительные учебные материалы, направленные на устранение выявленных пробелов. На четвертом этапе на основе сформированных материалов создается проверочный тест в формате *GIFT*, после чего вопросы сохраняются в структуре, при-

годной для последующего прохождения внутри системы и для интеграции с *Moodle*. Итоговая оценка рассчитывается как среднее значение результатов анализа отчета и проверки кода при наличии обоих компонентов.

Существенной особенностью реализации является гибкий механизм взаимодействия с *LLMs*. На уровне курса преподаватель или администратор может назначить конкретную конфигурацию модели с параметрами генерации, включая температуру, значение *top-p* и максимальный размер ответа. Система поддерживает как облачные провайдеры, так и локальные *OpenAI*-совместимые *Ollama* интерфейсы, что позволяет использовать комплекс в средах с повышенными требованиями к конфиденциальности. Для унификации вызовов применяется адаптерный слой, скрывающий различия между провайдерами, а шаблоны запросов собираются из системных заготовок, редактируемых параметров курса и переменных, подставляемых во время выполнения. Такой подход обеспечивает предметную адаптацию без изменения программного кода. Дополнительно в системе реализованы шифрование ключей доступа к внешним сервисам, учет квот использования моделей и журналирование операций, связанных с обработкой студенческих работ.

Одной из ключевых инноваций комплекса является интеграция агентной системы *OpenCode* [4] как специализированного средства глубокого анализа программного кода. В отличие от подхода, при котором языковая модель оценивает только переданный в запрос фрагмент текста, *OpenCode* работает с набором файлов репозитория: формирует отдельную сессию анализа, учитывает структуру проекта и взаимосвязи между файлами, после чего возвращает развернутое заключение о качестве архитектурных решений, корректности реализации и потенциальных дефектах. Благодаря синхронизации конфигурации *OpenCode* с параметрами моделей, назначенных в программном комплексе, достигается единообразие интеллектуального анализа текстовых отчетов и исходного кода. Это повышает полноту автоматической проверки и делает персональные рекомендации студенту более содержательными.

Практическая реализация комплекса ориентирована на реальные сценарии использования в учебном процессе. Преподаватель может создавать курсы и лабораторные задания, импортировать состав учебной группы из *CSV*-файла, просматривать отправленные материалы и результаты их проверки. Студент получает возможность загрузить отчет, отслеживать статус обработки и после завершения анализа ознакомиться с рецензией, замечаниями по коду, дополнительными материалами и индивидуальным тестом. Администратор управляет поставщиками и конфигурациями языковых моделей, шаблонами запросов и учетными записями пользователей.

В результате выполнения работы создан и протестирован программный комплекс, который охватывает полный учебный цикл: от публикации задания преподавателем до получения студентом развернутой обратной связи и прохождения проверочного теста. Система успешно работает с несколькими языковыми моделями; среднее время формирования рецензии на лабораторный отчет составляет от нескольких секунд до одной минуты в зависимости от выбранной модели и объема работы. Реализована выгрузка оценок в форматах, совместимых с *Moodle*, что позволяет встроить комплекс в существующую инфраструктуру учебного заведения. Значимым результатом является объединение в одной системе нескольких функций, которые обычно реализуются разрозненно: анализа отчета, проверки кода, генерации учебных материалов и формирования тестов.

Перспективами развития являются интеграция механизма контекстного поиска по учебным материалам (для более точных ответов относительно содержания курса),

расширение поддерживаемых форматов загружаемых файлов, а также интеграция тренажеров по недостаткам работ.

Литература

1. Асенчик, О. Д. Использование больших языковых моделей для создания учебных материалов по дисциплине «Базы данных» / О. Д. Асенчик // Цифровая трансформация. – 2025. – Т. 31, № 4. – С. 5–14. – DOI <https://doi.org/10.35596/1729-7648-2025-31-4-5-14>
2. Yan L. [et al.] Practical and ethical challenges of large language models in education: A systematic scoping review // British Journal of Educational Technology. – 2024. – Vol. 55, N 1. – P. 90–112. – DOI 10.1111/bjet.13370
3. Khanmigo/Khan Academy. – URL: <https://www.khanacademy.org/khan-labs> (дата обращения: 12.03.2026).
4. OpenCode. – URL: <https://opencode.ai> (дата обращения: 26.03.2026).

СИСТЕМА КОНТРОЛЯ САНИТАРНОГО СОСТОЯНИЯ ПОМЕЩЕНИЙ В УЧРЕЖДЕНИЯХ ОБРАЗОВАНИЯ

Д. О. Дубров

*Учреждение образования «Гомельский государственный технический
университет имени П. О. Сухого», Республика Беларусь*

Научный руководитель Т. Л. Романькова

Рассмотрены вопросы проектирования и реализации системы мониторинга качества воздуха в помещениях. Описан стек технологий (Go, React, PostgreSQL) и архитектурные решения.

Ключевые слова: мониторинг, качество воздуха, автоматизация, микроклимат, программный комплекс.

В современном мире автоматизация процессов играет ключевую роль в обеспечении комфортных условий жизнедеятельности, а также в оптимизации работы различных организаций и инфраструктурных объектов. Особенно актуально это для образовательных учреждений, где необходимо обеспечить не только высокий уровень организации учебного процесса, но и безопасные санитарно-гигиенические условия. Современные школы, колледжи и университеты предъявляют повышенные требования к качеству воздушной среды, стабильности температурного режима, уровню влажности и корректной освещенности. Эти параметры напрямую воздействуют на самочувствие обучающихся, их концентрацию внимания, способность усваивать материал и общую работоспособность. Несоблюдение санитарных норм может привести к снижению продуктивности учебного процесса, росту числа жалоб на состояние здоровья, а в долгосрочной перспективе – к ухудшению качества образовательной среды.

Традиционный подход к контролю санитарного состояния аудиторий основывается на периодических измерениях параметров окружающей среды и ручной фиксации полученных результатов. Обычно подобные измерения проводятся сотрудниками учреждения по заранее установленному графику – например, один или два раза в день. Однако такой подход имеет целый ряд существенных ограничений, которые снижают его эффективность в условиях современных требований к обеспечению безопасности и комфорта в образовательных учреждениях.

Целью работы является создание полноценной системы для непрерывного контроля параметров микроклимата в учебных помещениях. Разрабатываемый про-