ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ВЕБ-СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ФОРМИРОВАНИЯ РАБОЧИХ ПРОГРАММ ДИСЦИПЛИН НА ОСНОВЕ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

Пихтовников С.В., к.х.н., доцент,

Пихтовников А.С., аспирант

УУНиТ, г. Уфа, Россия

Аннотация. Статья продолжает работу авторов, в которой была предложена архитектура веб-системы формирования рабочих программ дисциплин на основе больших языковых моделей, и описывает ее программную реализацию. Рассмотрены программный стек, организация поиска по нормативной базе, этапы конвейера генерации и способ реализации четырех режимов. Описаны фоновое выполнение генерации, публикация готовых разделов по ходу работы и восстановление ответа модели по схеме. Показано, как реализация разделяет работу языковой модели и программные операции, которые можно проверить.

Ключевые слова: рабочая программа дисциплины, большая языковая модель, программная реализация, семантический поиск, pgvector, конвейер генерации, фоновые задачи.

В предыдущей работе авторов была предложена архитектура веб-системы автоматизированного формирования рабочих программ дисциплин (РПД), объединяющая большую языковую модель с семантическим поиском по нормативной базе, и четырехрежимный конвейер генерации [5]. Архитектура была описана на уровне компонентов, без деталей реализации, от которых на практике зависит качество документа.

Существующие системы автоматизации РПД в основном формируют структуру документа и переносят формальные параметры, а инструментальным средствам необходим программный контроль качества [3, 4]. Языковые модели способны генерировать учебный текст достаточно высокого качества [1], однако при их применении к нормативным документам основным риском остаются правдоподобные, но неверные сведения [2]. Поэтому при реализации системы принят принцип: языковая модель формирует текст, а все, что можно вычислить или проверить, вычисляется и проверяется программно.

Цель статьи состоит в описании программной реализации системы и алгоритмов, обеспечивающих этот принцип.

Система реализована на языке Python с веб-фреймворком Django. Данные хранятся в PostgreSQL с расширением pgvector, которое позволяет выполнять векторный поиск непосредственно в базе данных. Длительные операции генерации выполняются в фоновых задачах Celery с брокером Redis; Redis же служит общим кэшем, через

который веб-процесс и фоновый обработчик обмениваются состоянием задачи. Обращение к языковой модели выполняется через совместимый с OpenAI программный интерфейс, что позволяет менять модель без изменения кода. Итоговый документ формируется в формате DOCX по шаблону вуза средствами библиотек python-docx и docxtpl.

Нормативная база состоит из источников нескольких типов: ФГОС, профессиональные стандарты, квалификационные справочники, локальные акты и методические материалы. Для высшего образования загружены 472 действующих источника, разбитых на 10 487 фрагментов. Набор типов источников зависит от проекта: например, профессиональные стандарты подключаются, только если в проекте выбрана такая профессиональная основа. Каждый фрагмент хранится вместе с вектором, полученным многоязычной моделью paraphrase-multilingual-MiniLM-L12-v2 [7]; векторы рассчитываются пакетами в фоновой задаче.

При генерации текст запроса, составленный из параметров проекта, преобразуется в вектор, и база возвращает ближайшие фрагменты по косинусному расстоянию. Затем оценка фрагментов уточняется: небольшие надбавки получают источники, код направления подготовки которых совпадает с кодом проекта, и источники, прошедшие проверку. Если векторы недоступны, например при работе без расширения pgvector, используется лексическая оценка по совпадению слов запроса и фрагмента. В контекст модели передаются восемь фрагментов с наибольшей оценкой, а результаты поиска кэшируются в Redis. Такая организация реализует подход дополненной поиском генерации [6] с учетом специфики нормативных документов.

Конвейер генерации разбит на этапы, каждый из которых сообщает пользователю процент готовности и публикует готовые разделы, поэтому преподаватель видит паспорт дисциплины еще до окончания генерации (табл. 1).

Таблица 1. Этапы конвейера генерации

	Этап	**Содержание**	**Исполнитель**
1	Подбор материалов	Фрагменты нормативной базы, кандидаты литературы	Программа
2			

Ядро

Паспорт, компетенции, знания, умения, результаты обучения

Модель

3

План

Тематический план, часы, оценочные средства, методика

Модель

4

Объединение

Сборка документа, согласование часов

Программа

5

Восстановление и литература

Восстановление разделов, подбор и проверка литературы

Программа и модель

6

Проверки

Граф компетенций, шаблон, часы, экспертная проверка

Программа и модель

7

Доработка

От нуля до двух циклов исправления

Модель

Четыре режима, описанные в [5], реализованы через число циклов доработки на этапе 7: ускоренный режим пропускает доработку, режимы одного и двух циклов выполняют соответственно один и два цикла. Пошаговый режим запускает этапы 2 и 3 как отдельные фоновые задачи с остановкой после каждого: проект переходит в состояние ожидания, преподаватель проверяет и при необходимости правит разделы, после чего запускает следующий этап. Подбор литературы на этапе 5 выполняется параллельно с восстановлением разделов, так как эти операции изменяют разные части документа.

Каждый этап, выполняемый моделью, должен вернуть объект JSON по заданной схеме с перечнем обязательных ключей. Если ответ не разбирается или в нем не хватает ключей, система не прерывает генерацию, а направляет модели повторный запрос с исходным ответом, схемой и списком недостающих полей. Восстановленный ответ снова проверяется по схеме.

Таким образом, в реализации системы языковая модель отвечает за формулировки, а программные компоненты отвечают за поиск и ранжирование нормативных фрагментов, соблюдение схемы ответа, выполнение этапов в фоне и проверку результата. Пошаговый режим дополняет это разделение участием преподавателя на ключевых этапах. Такая организация делает поведение системы объяснимым и позволяет развивать отдельные компоненты, не затрагивая остальные.

Литература

1. Асенчик О.Д. Использование больших языковых моделей для создания учебных материалов по дисциплине «Базы данных» // Цифровая трансформация. 2025. Т. 31, № 4. С. 5-14.

2. Кошкина Е.А., Бордовская Н.В., Гнедых Д.С. и др. Генеративный искусственный интеллект в высшем образовании: обзор теоретических подходов и практик применения // Высшее образование в России. 2025. Т. 34, № 6. С. 36-57.

3. Майер С.Ф., Муратова Г.В. Анализ систем автоматизации формирования рабочих программ учебных дисциплин // Инженерный вестник Дона. 2021. № 11 (83). С. 274-287.

4. Рудинский И.Д., Пугачева Н.С. Автоматизация процесса разработки учебно-методической документации // Вестник науки и образования Северо-Запада России. 2020. Т. 6, № 2. С. 54-63.

5. Тархов С.В., Пихтовников С.В., Пихтовников А.С. Архитектура веб-системы автоматизированного формирования рабочих программ дисциплин на основе больших языковых моделей // Информационные технологии в образовании и науке: материалы II Международной научно-практической конференции. Бирск: УУНиТ, 2026. С. 331-334.

6. Lewis P., Perez E., Piktus A. et al. Retrieval-augmented generation for knowledge-intensive NLP tasks // Advances in Neural Information Processing Systems. 2020. Vol. 33. DOI: 10.48550/arXiv.2005.11401

7. Reimers N., Gurevych I. Making monolingual sentence embeddings multilingual using knowledge distillation // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2020. P. 4512-4525.