# МЕХАНИЗМ АВТОМАТИЗИРОВАННОЙ ГЕНЕРАЦИИ ДОКУМЕНТОВНА ОСНОВЕ ДЕКЛАРАТИВНОГО СООТВЕТСТВИЯ ШАБЛОННЫХ ДАННЫХ

**Алимбекова С. Р.**, д.т.н.,

**Афонин А. А.**, аспирант,

АНО ВО «Московский университет «Синергия»,

г. Москва, Россия

**Аннотация.** В статье рассматривается архитектура механизма автоматизированной генерации документов формата .docx на основе декларативного соответствия шаблонных ключевых слов с источниками данных. Описан подход к разметке шаблонов с применением синтаксиса Jinja2, обеспечивающий поддержку условных конструкций, итерации по коллекциям и применения фильтров. Предложена схема декларативного соответствия, связывающая ключевые слова шаблона с полями реляционной базы данных через путь в точечной нотации. Рассмотрены механизмы автоматической интроспекции схемы базы данных, оптимизации запросов посредством жадной загрузки и сохранения снимка контекста генерации в поле типа JSONB.

**Ключевые слова:** шаблонизация документов, Jinja2, docxtpl, SQLAlchemy,

декларативное соответствие, образовательный документооборот.

## Введение

Перед системами документооборота образовательных организаций стоит проблема, связанная с необходимостью массового выпуска юридически значимых документов, каждый из которых отличается лишь набором индивидуальных значений.

Ручной ввод данных в подобных условиях сопряжён с повышенной вероятностью возникновения ошибок и требует значительных временных затрат. Мамонов А. А. и соавторы в рамках своего исследования приходят к выводу, что автоматизация генерации документов по шаблону обеспечивает сокращение временных затрат на 80 % при ежемесячной обработке более 500 документов [5]. Данные результаты согласуются с выводами независимых исследований, подтверждающих снижение времени создания документации на 60 % по сравнению с ручными методами [4].

Потребность в подобных решениях возникает на предприятиях различных сфер деятельности, и одним из ключевых подходов к её разрешению является создание специализированного программного обеспечения, реализующего необходимую бизнес-логику [2].

В рамках научно-образовательных учреждений данная проблематика приобретает наибольшую актуальность ввиду регулярной необходимости формирования значительного объёма документации, содержащей дублирующиеся или подлежащие преобразованию сведения, что определяет целесообразность применения технологий шаблонной генерации [3].

Предлагаемое решение основывается на двух принципах: разделении формы и содержания, при котором шаблон существует независимо от данных, и декларативном описании связи между метками в тексте документа и источниками данных. Совокупность этих принципов обеспечивает независимость как шаблонов, так и конфигурации механизма соответствия от программного кода.

**Шаблон как декларативная структура**

В реализованной системе шаблон документа представляет собой файл формата .docx, в текст которого интегрированы метки в синтаксисе Jinja2. Представление документа как шаблона-функции, в котором отсутствуют индивидуальные данные, а содержание формируется как текст с добавленными значениями по заданной структуре, концептуально согласуется с принципом разделения формы и содержания [3].

Способ разметки динамических полей является принципиальным архитектурным

выбором. Двухфазная технология предусматривает разметку шаблона Microsoft Word посредством именованных закладок (Bookmark). Принцип заключается в алгоритме, который последовательно обходит закладки по вычисляемым именам и вставляет в каждую строку данные из промежуточного текстового интерфейсного файла [1].

Визуальное разделение параметров от статического содержимого при работе в Microsoft Word можно реализовать с использованием режима разработчика и встроенного механизма ContentControls, что, в свою очередь, накладывает ограничения для рядового пользователя [2]. Использование такого подхода позволяет производить замену ключевых слов на соответствующие им данные, но возможности использования условных конструкций непосредственно в тексте документа и циклической подстановки данных из коллекций не поддерживаются этим механизмом и требуют уникальной реализации.

Выбор Jinja2 в качестве шаблонизатора обусловлен как практическими, так и техническими соображениями. В отличие от механизма ContentControls, использование шаблонизатора позволяет конечному пользователю самостоятельно создавать и редактировать шаблоны непосредственно в Microsoft Word без привлечения разработчиков для внесения правок. С технической стороны инструмент поддерживает условные конструкции, итеративный обход коллекций и применение фильтров к подставляемым значениям. Функциональность такого рода принципиально недостижима при использовании закладок или символьной подстановки.

Исходя из возможностей,предоставляемых инструментом, ключевые слова были разделены на три функциональных класса:

· Скалярные переменные. Используются для подстановки одиночных значений (например, {{ org_full_name }}, {{ student_full_name }}
).

· Скалярные переменные с параметрами склонения.Дополнительные параметры позволяют изменять склонения подставляемых данных при помощи реализованного морфологического анализатора (например, {{sig_full_name|m('gent')}}
).

· Блочные конструкции. Обеспечивают итеративный обход коллекций данных с подстановкой по заданному образцу. Применяются при формировании таблиц, списков и других схожих структур с переменным числом строк.

Файл формата .docx можно представить в формате ZIP-архива. Полученный архив содержит информацию о структуре и форматировании документа в виде различных наборов XML-файлов. Извлечение ключевых слов выполняется путём анализа этого XML-содержимого. Здесь проявляется характерная особенность формата: один визуально непрерывный фрагмент текста нередко оказывается разбитым на несколько элементов <w:r> с различными атрибутами форматирования. Из этого следует, что прямой поиск ключевых слов по XML-дереву ненадёжен, так как метка, записанная в редакторе как единый текст, в структуре файла может быть разнесена по нескольким узлам. Для корректного распознавания меток текст каждого абзаца предварительно конкатенируется. Проблема механизмов восстановления разбитых меток является общей для большинства существующихинструментов, но они не предоставляют соответствующего функционала [5].

## Сопоставление ключевых слов с источниками данных

Центральной идеей системы является декларативное описание связи между тегом шаблона и источником данных, что исключает зависимость конфигурации от программного кода. Задача установления соответствия между полями шаблона и источниками данных решается в смежных исследованиях по-разному:

- определение связей между функциями документа-приёмника и документов-источников [3],
- позиционное соответствие строк интерфейсного файла и индексов закладок шаблона [1],
- посредством интеграции с CSV-источниками, когда столбцы таблицы динамически соотносятся с тегами шаблона с автоматической валидацией типов данных [5].

Последний подход эффективен для однородных наборов данных фиксированной структуры, однако не обеспечивает работу с иерархическими объектами и вычисляемыми значениями.

Для каждого тега шаблона администратор системы указывает путь к полю базы данных в точечной нотации, например organization.full_name или application_items.student.full_name. Пути строятся относительно корневой сущности (Application), от которой через отношения ORM достигается любая связанная сущность.

Для исключения ошибок при конфигурировании путей к данным применяется механизм интроспекции схемы базы данных, который на этапе инициализации выполняет рекурсивный обход моделей SQLAlchemy. Обход начинается с сущности Application и формирует структурированный реестр допустимых маршрутов, адаптированный для визуализации в пользовательском интерфейсе. Данный инструмент сопоставления данных также поддерживает системные параметры, независимые от контекста отдельной заявки, включая идентификатор текущего пользователя и временные метки, а также обеспечивает агрегатные вычисления над коллекциями данных (COUNT, SUM, MIN, MAX, AVG). Результирующая конфигурация сериализуется и сохраняется с типом данных JSONB в структуре шаблона документа.

**Процесс генерации и сохранения снимка данных**

Формирование документа начинается с извлечения из базы данных записи заявки и всех связанных с ней объектов по уникальному идентификатору. При работе с большим количеством взаимосвязанных объектов каждое обращение к связанной сущности инициирует отдельный запрос к базе данных, что приводит проблеме N+1 запросов. Для её устранения на уровне ключевых сущностей настроен механизм жадной загрузки, позволяющий получать данные нескольких связанных объектов в рамках одного запроса. Дополнительно система шаблонной генерации анализирует конфигурацию соответствия, автоматически выстраивает дерево зависимостей и формирует соответствующие стратегии загрузки joinedload и selectinload для необходимых задействованных отношений .

Необходимость такой оптимизации напрямую обусловлена характером процесса подстановки. При заполнении шаблона каждое уникальное ключевое слово вычисляется в результате обхода указанных объектов по заданному пути в точечной нотации, а каждая коллекция разворачивается в список объектов. Если ключевое слово

содержит морфологические параметры, соответствующие данные дополнительно обрабатываются анализатором. Итогом является словарь вида {tag_name: resolved_value} , передаваемый шаблонизатору. Разрешение каждого уникального ключевого слова сопровождается обращением к базе данных и, как следствие, временными затратам на выполнение этих запросов. Мамонов А. А. установил, что при росте числа ключевых слов с 1500 до 5500 время формирования документа увеличивается с 0,12864 до 9,95829 секунд [5]. Применяемые методы оптимизации позволяют существенно снизить нагрузку на базу данных и удержать время ответа системы в приемлемых для пользователя пределах.

Юридически значимые документы, как правило, имеют жёстко регламентированный вид, включая шрифты, таблицы и стили [5]. Прямые манипуляции с XML-структурой файла сопряжены с риском нарушения целостности разметки, поэтому в системе применяется библиотека docxtpl, обеспечивающая корректную интеграцию шаблонизатора Jinja2 с форматом .docx. Обращение к высокоуровневым инструментам вместо низкоуровневой работы с XML характерно и для других систем аналогичного назначения, что свидетельствует об устойчивости данного подхода [2].

Наряду с формированием документа система сохраняет снимок контекста генерации. Бинарное содержимое файла и данные, на основе которых он был сформирован, хранятся раздельно. В базе данных фиксируется сериализованный словарь соответствия ключевых слов в формате JSONB и ссылка на расположение файла. Такое разделение согласуется с принципом специализации хранилищ, применяемым в аналогичных системах [4]. Даже при утрате или недоступности файла содержимое документа остаётся верифицируемым, а повторная генерация по сохранённому контексту гарантирует воспроизводимость результата независимо от последующих изменений в исходных данных.

## Заключение

В статье описана система генерации документов, реализующая декларативный подход к связыванию ключевых слов шаблона с источниками данных. Конфигурация соответствия хранится отдельно от программного кода, что обеспечивает независимость её изменения от разработчиков системы. Применение синтаксиса Jinja2 предоставляет необходимые возможности работы с коллекциями данных и условными конструкциями, которые не поддерживаются в подходах на основе закладок и символьной подстановки. Механизм интроспекции схемы базы данных позволяет администраторам самостоятельно описывать пути к данным, опираясь на визуализированный реестр допустимых маршрутов. Сохранение снимка контекста генерации в формате JSONB обеспечивает идемпотентность результата и возможность последующего аудита содержимого документа.

Наиболее перспективным путём развития инструмента является интеграция отдельного модуля семантической проверки соответствия данных документа, а также расширение возможностей морфологического модуля за счёт внедрения ИИ-моделей. Существующий анализатор опирается на детерминированные правила, тогда как нейросетевой подход способен корректнее обрабатывать нестандартные словоформы. Искусственный интеллект также позволит решить принципиальную задачу автоматизации подготовки шаблонов на основе уже готовых документов. Внедрение такого функционала существенно сократит временные затраты на подготовку шаблонов и корректировку итоговых документов.

## Литература

1. Большаков, С. А. Двухфазная технология формирования выходных документов в информационных системах на основе MS WORD / С. А. Большаков, С. Б. Спиридонов // Наука и образование: научное издание МГТУ им. Н.Э. Баумана. – 2015. – № 8. – С. 252-268. – DOI 10.7463/0815.0786571. – EDN UJFAJL.

2. Карышев, А. А. Разработка web-сервиса для автоматизированной генерации документов на основе docx-шаблонов / А. А. Карышев, В. Р. Афанасьев // Известия Тульского государственного университета. Технические науки. – 2017. – № 5. – С. 290-297. – EDN ZBCKQP.

3. Коробова, И. Л. Разработка шаблона последовательности функций и связей для документов научно-образовательного учреждения / И. Л. Коробова, Н. В. Майстренко, Д. Д. Бараева // Вестник Тамбовского государственного технического университета. – 2017. – Т. 23, № 2. – С. 241-246. – DOI 10.17277/vestnik.2017.02.pp.241-246. – EDN YSEIEV.

4. Ларионов М. С. Разработка сервиса формирования электронных документов по шаблону // Форум молодых ученых. 2025. №5 (105). URL: https://cyberleninka.ru/article/n/razrabotka-servisa-formirovaniya-elektronnyh-dokumentov-po-shablonu (дата обращения: 18.04.2026).

5. Мамонов А. А. Салпагаров С. И., Матюшкин Д. В., Миронов Д. А., Кройтор О. К. Разработка системы автоматизированной подготовки документов с использованием библиотеки APACHE POI // КИО. 2025. №2. URL: https://cyberleninka.ru/article/n/razrabotka-sistemy-avtomatizirovannoy-podgotovki-dokumentov-s-ispolzovaniem-biblioteki-apache-poi (дата обращения: 18.04.2026).