МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА BACKEND-АРХИТЕКТУРЫ ВЕБ-ПЛАТФОРМЫ АВТОМАТИЗИРОВАННОЙ ПРОВЕРКИ ДОКУМЕНТОВ НА СООТВЕТСТВИЕ ТРЕБОВАНИЯМ ГОСТ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Павловой Алёны Александровны

Научный руководитель
зав.каф.техн.пр.,
к.ф.-м.н., доцент _____ И. А. Батраева

Заведующий кафедрой
доцент, к. ф.-м. н. _____ С. В. Миронов

Саратов 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. В современных условиях цифровой трансформации организаций и образовательных учреждений особое значение приобретают технологии автоматизации документооборота и контроля качества подготовки документов. Практически во всех сферах деятельности используются документы, оформление которых регламентируется государственными стандартами, внутренними нормативными актами и методическими рекомендациями. Особенно актуальна данная задача для образовательных организаций, научно-исследовательских учреждений и предприятий, осуществляющих подготовку технической документации.

Подготовка курсовых работ, выпускных квалификационных работ, отчетов по научно-исследовательской деятельности и другой документации требует соблюдения большого количества требований к структуре документа, параметрам оформления текста, оформлению таблиц, рисунков, списков литературы и иных элементов. Проверка подобных требований традиционно выполняется вручную, что требует значительных временных затрат и высокой квалификации проверяющего специалиста. При увеличении количества проверяемых документов возрастает вероятность пропуска ошибок оформления, что снижает качество нормоконтроля и увеличивает трудозатраты на подготовку документации.

Существующие программные решения в большинстве случаев ориентированы на проверку орфографических, грамматических и стилистических ошибок. Наиболее известными представителями подобных систем являются Grammarly и LanguageTool. Несмотря на широкий набор инструментов для анализа текстового содержимого, данные системы практически не затрагивают вопросы проверки документов на соответствие требованиям государственных стандартов и не обеспечивают комплексный контроль структуры и оформления документа. В связи с этим возникает необходимость разработки специализированных программных средств, позволяющих автоматизировать процесс нормоконтроля документов.

Для решения данной задачи в рамках программы «Стартап как диплом» разрабатывается веб-платформа «Электронный корректор», предназначенная для автоматизированной проверки документов на соответствие требованиям ГОСТ. Платформа представляет собой комплексное программное решение,

включающее клиентскую часть, модуль интеллектуальной обработки документов и серверную инфраструктуру, обеспечивающую взаимодействие между компонентами системы.

Особое значение в рамках проекта имеет backend-платформа, поскольку именно серверная часть отвечает за обработку пользовательских запросов, хранение документов, организацию обмена данными между сервисами, управление результатами проверки и обеспечение безопасности пользовательских данных. От качества проектирования backend-архитектуры напрямую зависят производительность, масштабируемость и надежность всей системы.

Целью бакалаврской работы является разработка backend-архитектуры веб-платформы автоматизированной проверки документов на соответствие требованиям ГОСТ.

Для достижения поставленной цели были определены **следующие задачи**:

1. Выполнить анализ предметной области автоматизированного нормоконтроля документов.
2. Провести исследование существующих программных решений и определить их преимущества и недостатки.
3. Сформировать функциональные и нефункциональные требования к backend-платформе.
4. Спроектировать архитектуру серверной части системы.
5. Разработать REST API для взаимодействия клиентской и серверной частей платформы.
6. Реализовать механизм аутентификации и авторизации пользователей.
7. Организовать систему хранения документов и результатов проверки.
8. Реализовать асинхронную обработку событий с использованием брокера сообщений.
9. Выполнить контейнеризацию компонентов платформы.
10. Провести тестирование разработанной системы.

Теоретическая значимость работы заключается в исследовании современных подходов к проектированию серверных информационных систем, организации хранения документов, построению REST API и реализации механизмов асинхронного взаимодействия между компонентами программных комплексов.

Практическая значимость работы заключается в разработке backend-

платформы, обеспечивающей функционирование веб-сервиса автоматизированной проверки документов на соответствие требованиям ГОСТ и создающей основу для дальнейшего развития проекта «Электронный корректор».

Структура работы включает введение, две главы, заключение, список использованных источников и приложения. В первой главе рассматриваются вопросы анализа предметной области и проектирования архитектуры системы. Во второй главе описываются особенности реализации backend-платформы и результаты проведенного тестирования.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первая глава «Проектирование backend-архитектуры веб-платформы автоматизированной проверки документов» посвящена анализу предметной области, исследованию существующих программных решений, формированию требований к разрабатываемой системе и проектированию основных компонентов backend-платформы.

В разделе **анализа предметной области** рассмотрены особенности автоматизированного нормоконтроля документов. Отмечается, что при подготовке научных работ, технической документации, отчетов и иных документов пользователи вынуждены соблюдать большое количество требований, регламентирующих структуру документа, параметры оформления текста, правила оформления таблиц, рисунков, формул и списка использованных источников. Контроль выполнения подобных требований традиционно осуществляется вручную, что требует значительных временных затрат и зависит от квалификации проверяющего специалиста.

Показано, что автоматизированный нормоконтроль позволяет существенно сократить время проверки документов за счет использования программных алгоритмов, анализирующих структуру и параметры оформления документа. При этом система должна выполнять не только обработку текстового содержимого, но и анализ форматирования, структуры разделов, параметров страниц и других характеристик документа.

В рамках исследования был проведен **анализ существующих решений**, предназначенных для автоматизированной проверки текстовой информации. В качестве наиболее известных представителей были рассмотрены системы Grammarly и LanguageTool. Выполненное исследование показало, что данные программные продукты ориентированы преимущественно на проверку орфографии, грамматики и стилистики текста. Несмотря на высокий уровень развития лингвистических инструментов, данные решения не обеспечивают полноценную проверку документов на соответствие требованиям ГОСТ и не позволяют выполнять комплексный контроль оформления документов.

Результаты проведенного анализа позволили сформировать **основные требования к разрабатываемой backend-платформе**. К числу функциональных требований были отнесены регистрация и авторизация пользователей, загрузка документов, хранение информации о результатах проверки, управление стату-

сами обработки документов, предоставление пользователю отчетов о найденных ошибках, поддержка REST API и асинхронной обработки событий. Среди нефункциональных требований выделены производительность, масштабируемость, отказоустойчивость, безопасность хранения данных и возможность дальнейшего расширения функциональности системы.

На основании сформулированных требований была поставлена **задача проектирования серверной архитектуры платформы**, обеспечивающей обработку пользовательских запросов, хранение документов, выполнение проверок и взаимодействие между компонентами системы.

В подразделе проектирования архитектуры платформы была разработана **общая структура backend-системы**. Архитектура построена по клиент-серверной модели и предусматривает взаимодействие пользовательского интерфейса с серверной частью посредством REST API. В качестве основного backend-фреймворка выбран FastAPI, обеспечивающий поддержку асинхронной обработки запросов и высокую производительность. Для хранения структурированных данных используется PostgreSQL, а хранение файлов реализовано с использованием объектного S3-совместимого хранилища MinIO. Для организации обмена сообщениями между компонентами системы используется брокер сообщений Apache Kafka.

Особое внимание в работе уделено **проектированию REST API**. Все маршруты системы были разделены по функциональным областям. Отдельные группы endpoint-ов отвечают за регистрацию и авторизацию пользователей, работу с документами, выполнение проверок, получение результатов анализа и управление справочными сущностями системы. Для обмена данными между клиентской и серверной частями используется формат JSON, а для загрузки документов применяется формат multipart/form-data.

При проектировании системы хранения данных была разработана **структура реляционной базы данных**, включающая таблицы пользователей, документов, проверок, отчетов, ошибок, типов ошибок, поддерживаемых стандартов и статусов обработки. Центральным элементом модели является сущность документа, связывающая пользователей, проверки и результаты анализа. Для обеспечения целостности данных между таблицами были реализованы связи посредством внешних ключей.

Отдельный подраздел посвящен проектированию **системы хранения до-**

кументов. В работе обоснован выбор подхода, основанного на разделении хранения файлов и метаданных. Метаданные документов сохраняются в PostgreSQL, а сами файлы размещаются в объектном хранилище MinIO. Такой подход позволяет снизить нагрузку на базу данных, повысить производительность системы и обеспечить возможность независимого масштабирования подсистемы хранения документов.

Также в первой главе рассмотрены вопросы **проектирования контейнерной инфраструктуры**. Для развертывания платформы было принято решение использовать Docker и Docker Compose. Контейнеризация обеспечивает изоляцию компонентов системы, упрощает настройку среды разработки и позволяет ускорить процесс развертывания приложения.

Завершающая часть первой главы посвящена **проектированию алгоритма проверки документов на соответствие требованиям ГОСТ**. Предложенный алгоритм включает этапы получения документа, извлечения структурных и форматных характеристик, загрузки набора правил проверки, выполнения анализа и формирования итогового отчета. Для обеспечения гибкости системы правила проверки хранятся в отдельных JSON-файлах и могут изменяться без модификации программного кода приложения. Такой подход позволяет расширять набор поддерживаемых требований и адаптировать систему под различные стандарты оформления документов.

Вторая глава «Реализация и тестирование backend-платформы» посвящена практической реализации спроектированной архитектуры, разработке основных компонентов серверной части платформы и проведению тестирования реализованного программного обеспечения.

В начале главы рассматривается **реализация структуры backend- приложения**. Серверная часть платформы разработана на языке программирования Python с использованием современного web-фреймворка FastAPI. Архитектура приложения построена по модульному принципу, что позволило разделить различные функциональные области системы и повысить удобство сопровождения программного кода.

Основной точкой входа в приложение является модуль, отвечающий за создание экземпляра FastAPI-приложения, подключение middleware-компонентов и регистрацию маршрутов REST API. Для организации взаимодействия между компонентами системы использовались встроенные механизмы внедрения зави-

симостей, предоставляемые FastAPI. Такой подход позволил обеспечить слабую связанность компонентов и упростить дальнейшее расширение функциональности системы.

Отдельное внимание уделено **реализации REST API платформы**. Для каждой функциональной области были разработаны собственные маршрутизаторы, отвечающие за обработку соответствующих запросов. Реализованы маршруты регистрации и авторизации пользователей, загрузки документов, получения информации о документах, запуска проверки, получения результатов анализа и управления справочными сущностями системы.

В процессе реализации были использованы **Pydantic-схемы**, обеспечивающие валидацию входных данных и формирование структурированных ответов API. Использование схем позволило повысить надежность обработки пользовательских запросов и уменьшить вероятность возникновения ошибок при передаче данных между клиентской и серверной частями приложения.

Одним из наиболее важных компонентов платформы является **система авторизации и защиты данных пользователей**. Для решения данной задачи была реализована аутентификация на основе технологии JSON Web Token (JWT). После успешной проверки учетных данных пользователя система формирует токен доступа, содержащий информацию о пользователе и сроке действия сессии. Полученный токен используется при обращении к защищенным маршрутам платформы.

При обработке защищенных запросов backend-приложение выполняет декодирование JWT-токена, проверку его цифровой подписи и получение данных пользователя. После этого дополнительно осуществляется проверка существования учетной записи в базе данных. Использование подобного механизма обеспечивает безопасный доступ к ресурсам системы и предотвращает выполнение операций неавторизованными пользователями.

Дополнительно **реализовано разграничение прав доступа**. Пользователи платформы имеют возможность работать только со своими документами и результатами проверок, тогда как администраторы обладают расширенным набором полномочий, позволяющим выполнять управление справочными данными и контролировать работу отдельных компонентов системы.

Следующим этапом реализации стала **разработка системы хранения документов**. Для хранения структурированных данных была использована реля-

ционная система управления базами данных PostgreSQL. Работа с базой данных организована посредством ORM SQLAlchemy, что позволило описать сущности системы в виде объектных моделей и существенно упростить взаимодействие приложения с базой данных.

Хранение файлов реализовано с использованием объектного S3- совместимого хранилища MinIO. При загрузке документа backend-приложение выполняет проверку типа файла, формирует уникальное имя документа и сохраняет файл в объектное хранилище. В базе данных при этом создается запись, содержащая сведения о пользователе, пути хранения документа, размере файла и текущем статусе обработки.

Разделение хранения файлов и метаданных позволило снизить нагрузку на реляционную базу данных и обеспечить возможность независимого масштабирования подсистемы хранения документов. Кроме того, данный подход соответствует современным принципам построения распределенных информационных систем.

Для организации взаимодействия между компонентами платформы была реализована **подсистема асинхронной обработки событий**. В качестве брокера сообщений использовался Apache Kafka. Основной задачей Kafka в разработанной системе является обработка событий изменения состояния документов в процессе их проверки.

При выполнении операций загрузки документа, запуска проверки или изменения статуса backend-приложение публикует сообщения в специализированный **Kafka-топик**. Полученные сообщения обрабатываются отдельным consumer-компонентом, который выполняет обновление состояния документа независимо от основного потока обработки HTTP-запросов.

Использование **асинхронного обмена сообщениями** позволило уменьшить нагрузку на серверное приложение и обеспечить более устойчивую работу платформы при увеличении количества одновременно выполняемых операций.

Для упрощения развертывания системы была выполнена **контейнеризация компонентов платформы** с использованием Docker и Docker Compose. Каждый сервис системы запускается в отдельном контейнере, что обеспечивает изоляцию компонентов и упрощает сопровождение инфраструктуры.

В состав контейнеризированной среды входят FastAPI-приложение, MinIO, PostgreSQL, Kafka, Zookeeper и Kafka UI. Все сервисы объединены общей сетью

Docker, что обеспечивает их взаимодействие посредством внутренних сетевых адресов. Использование Docker Compose позволило автоматизировать процесс запуска платформы и сократить время настройки рабочего окружения.

Значительная часть второй главы посвящена **реализации алгоритма автоматизированной проверки документов на соответствие требованиям ГОСТ**. Алгоритм включает этапы извлечения данных из документа, загрузки набора правил и последовательного выполнения проверок.

Для обеспечения независимости алгоритма от конкретного формата документа реализовано промежуточное представление данных, содержащее информацию о структуре документа, параметрах оформления, настройках шрифта, размерах полей страницы, абзацных отступах и других характеристиках, используемых при проверке.

Проверка документа выполняется на основе набора правил, хранящихся в формате JSON. Для каждого правила определяется тип проверки, после чего вызывается соответствующий обработчик. Реализованы проверки наличия обязательных структурных элементов документа, соответствия параметров оформления установленным требованиям, проверки числовых диапазонов и анализа текстового содержимого отдельных разделов документа.

Особое внимание уделено проверке раздела «Введение». В отличие от большинства формальных проверок данный этап предполагает анализ содержательной части текста и поиск ключевых смысловых элементов, характеризующих актуальность работы, цель исследования и поставленные задачи.

Результатом работы алгоритма является набор объектов проверки, содержащих сведения о выполненных проверках, найденных нарушениях и рекомендациях по их устранению. На основании этих данных формируется итоговый отчет, предоставляемый пользователю через интерфейс платформы.

Завершающая часть второй главы посвящена тестированию разработанной backend-платформы. Основной целью тестирования являлась проверка корректности функционирования отдельных компонентов системы и подтверждение возможности их совместного использования в рамках единой программной платформы.

Для проверки работоспособности REST API использовался **инструмент Postman**, позволяющий выполнять отправку HTTP-запросов и анализировать ответы серверного приложения. В ходе тестирования были проверены основные

пользовательские сценарии работы с системой, включая регистрацию пользователей, авторизацию, получение JWT-токенов, загрузку документов, запуск проверки и получение результатов анализа.

Проведенные испытания показали, что backend-приложение корректно обрабатывает входящие запросы, выполняет проверку пользовательских данных и формирует ответы в соответствии со спецификацией REST API. Также была подтверждена корректность работы механизма аутентификации и авторизации пользователей на основе JWT-токенов.

Отдельный этап тестирования был посвящен **проверке системы хранения документов**. В процессе испытаний выполнялась загрузка документов в объектное хранилище MinIO, сохранение метаданных в базе данных PostgreSQL и последующее получение информации о загруженных файлах. Особое внимание уделялось проверке корректности формирования путей хранения документов и контролю прав доступа пользователей к собственным файлам.

Результаты тестирования подтвердили стабильную работу подсистемы хранения документов и корректное взаимодействие backend-приложения **с объектным хранилищем**. Используемый подход обеспечил надежное разделение хранения файлов и структурированных данных без потери целостности информации.

Для проверки механизма асинхронной обработки событий было проведено тестирование взаимодействия backend-приложения с брокером сообщений Apache Kafka. В ходе испытаний выполнялась публикация событий изменения состояния документов в Kafka-топик, после чего consumer-компонент осуществлял обработку поступивших сообщений и обновление соответствующих записей в базе данных.

Мониторинг сообщений осуществлялся с использованием инструмента Kafka UI. Проведенные испытания подтвердили корректность передачи сообщений между компонентами системы, успешную обработку событий и обновление статусов документов без участия основного потока обработки HTTP-запросов. Использование Kafka позволило обеспечить независимость процессов обработки событий и повысить устойчивость системы к увеличению нагрузки.

Дополнительно было выполнено **тестирование контейнерной инфраструктуры платформы**. Проверялась корректность запуска сервисов с использованием Docker Compose, доступность контейнеров внутри общей сети

и взаимодействие между компонентами системы. В процессе испытаний были успешно протестированы соединения между FastAPI-приложением, PostgreSQL, MinIO и Kafka.

Результаты тестирования показали, что контейнеризированная инфраструктура обеспечивает стабильную работу платформы и упрощает процесс развертывания программного комплекса в различных средах эксплуатации.

Отдельное внимание было уделено **тестированию алгоритма проверки документов** на соответствие требованиям ГОСТ. Для проведения испытаний использовались документы, содержащие как корректные параметры оформления, так и намеренно внесенные нарушения требований стандарта. В процессе проверки алгоритм успешно выявлял отсутствие обязательных структурных элементов документа, нарушения параметров форматирования, некорректные значения полей страницы и другие несоответствия установленным требованиям.

Полученные результаты подтвердили работоспособность реализованного алгоритма и возможность его использования для автоматизированного нормоконтроля документов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была разработана backend-архитектура веб-платформы автоматизированной проверки документов на соответствие требованиям ГОСТ. В рамках проведенного исследования были рассмотрены особенности автоматизированного нормоконтроля документов, проанализированы существующие программные решения и определены основные требования к разрабатываемой системе.

Проведенный анализ предметной области показал, что существующие программные продукты ориентированы преимущественно на проверку текстового содержимого документов и не обеспечивают комплексного контроля оформления в соответствии с требованиями государственных стандартов. Это подтверждает актуальность создания специализированных систем автоматизированного нормоконтроля, способных выполнять анализ структуры документа, параметров оформления и соответствия установленным нормативным требованиям.

В процессе работы была спроектирована архитектура серверной части платформы «Электронный корректор», обеспечивающая обработку пользовательских запросов, хранение документов, выполнение проверок и взаимодействие между компонентами системы. В качестве технологической основы были выбраны современные программные решения, позволяющие обеспечить надежность, масштабируемость и возможность дальнейшего развития платформы.

В ходе реализации разработано backend-приложение на основе фреймворка FastAPI. Создан REST API для взаимодействия клиентской и серверной частей системы, реализованы механизмы регистрации и авторизации пользователей, организовано хранение документов и результатов проверки. Для обеспечения безопасности данных внедрена система аутентификации на основе JWT-токенов и реализовано разграничение прав доступа пользователей.

Для хранения структурированных данных использована реляционная система управления базами данных PostgreSQL, а хранение документов организовано с использованием объектного S3-совместимого хранилища MinIO. Такой подход позволил разделить хранение файлов и метаданных, повысить эффективность работы системы и обеспечить возможность независимого масштабирования отдельных компонентов платформы.

Одним из важных результатов работы стала реализация механизма асин-

хронной обработки событий с использованием Apache Kafka. Использование брокера сообщений позволило организовать обработку изменений состояния документов независимо от основного потока выполнения запросов, что способствует повышению устойчивости системы и улучшению её производительности при увеличении нагрузки.

В рамках работы также был реализован алгоритм автоматизированной проверки документов на соответствие требованиям ГОСТ. Алгоритм выполняет извлечение структурных и форматных характеристик документа, анализирует полученные данные на основании набора правил и формирует подробный отчет о выявленных нарушениях. Для обеспечения гибкости системы правила проверки вынесены в отдельные JSON-файлы, что позволяет расширять перечень поддерживаемых требований без изменения программного кода приложения.

После завершения этапа реализации было проведено комплексное тестирование основных компонентов платформы. Проверены корректность работы REST API, механизмов авторизации, системы хранения документов, асинхронной обработки событий и алгоритма проверки документов. Результаты тестирования подтвердили работоспособность разработанной системы и корректность взаимодействия между её компонентами.

Практическая значимость работы заключается в создании backend- платформы, которая может использоваться в качестве основы для дальнейшего развития веб-систем автоматизированного нормоконтроля документов. Разработанное решение обеспечивает возможность хранения документов, выполнения автоматизированных проверок, формирования отчетов и организации взаимодействия между компонентами распределенной информационной системы.

Выпускная квалификационная работа выполнена в рамках программы «Стартап как диплом». Разработанная backend-платформа является составной частью проекта «Электронный корректор» и обеспечивает реализацию ключевых функций серверной инфраструктуры системы. Полученные результаты могут быть использованы при дальнейшем развитии платформы, расширении набора поддерживаемых стандартов и внедрении дополнительных модулей интеллектуального анализа документов.