Содержание

Введение

Темой курсового проекта является «Проектирование и разработка системы автоматизированной проверки академических работ на соответствие стандартам оформления».

В условиях цифровой трансформации образования и возрастающей роли самостоятельной проектной деятельности студентов актуализируется задача обеспечения высокого стандарта качества представляемых академических работ. Правильное оформление курсовых и дипломных проектов — это не только формальное требование, но и важный элемент профессиональной культуры, дисциплины, а также критерий, напрямую влияющий на оценку. Однако процесс ручной проверки соответствия оформления десяткам параметров (шрифты, поля, интервалы, структура) является крайне трудоемким, подверженным субъективным ошибкам и отнимает у преподавателей время, которое могло бы быть посвящено анализу содержательной части работы.

В настоящее время в большинстве образовательных учреждений отсутствует единая автоматизированная система для решения этой задачи. Проверка осуществляется преподавателем визуально или с помощью базовых инструментов текстового редактора, что приводит к ряду системных проблем: несогласованности требований между различными кафедрами, сложностям в оперативном информировании студентов о допущенных ошибках, невозможности накопления статистики по типичным нарушениям и, как следствие, к снижению общего уровня оформления студенческих работ.

Таким образом, возникает противоречие между необходимостью обеспечения единообразного, качественного и эффективного контроля формальных требований к учебным работам и отсутствием специализированного программного инструментария для автоматизации этого процесса. Устранение данного противоречия определяет актуальность курсового проекта.

Объект исследования – процесс контроля соответствия оформления курсовых и дипломных работ в высшем или среднем профессиональном учебном заведении.

Предмет исследования – методы, алгоритмы и информационные технологии для автоматизации анализа документов и верификации их соответствия заданным стандартам оформления.

Цель курсового проекта – повышение эффективности, объективности и согласованности процесса проверки оформления учебных работ за счет проектирования и разработки веб-приложения, автоматизирующего анализ документов и предоставляющего детализированную верифицированную отчетность.

При проектировании и разработке веб-приложения будут решены следующие задачи:

сбор и анализ требований к оформлению учебных работ и потребностей целевых пользователей;

разработка проектной документации (техническое задание, спецификации) на информационную систему;

проектирование архитектуры системы, пользовательских интерфейсов и модели данных;

разработка ключевых модулей системы: модуля парсинга и анализа документов DOCX, модуля управления настраиваемыми стандартами, модуля формирования отчетов;

реализация подсистемы безопасности и разграничения доступа;

тестирование функциональности и производительности системы;

разработка руководства пользователя и администратора системы.

1. Проектирование информационной системы

1.1 Анализ предметной области и построение бизнес-процессов

Цель анализа – выявить неэффективности существующего процесса, сформулировать функциональные и нефункциональные требования, а также построить модели будущей системы, основанные на реальных потребностях преподавателей и студентов.

Процесс контроля оформления является заключительным этапом перед сдачей работы и вовлекает двух ключевых участников: студента (автора работы) и преподавателя (проверяющего, научного руководителя, рецензента). В крупных вузах эту функцию также могут выполнять методисты кафедр или администраторы образовательных программ.

Описание текущего процесса:

Подготовка стандарта: кафедра или деканат разрабатывает методические указания, часто в формате PDF- или DOC-файла, содержащие требования к оформлению. Эти требования могут меняться, что не всегда оперативно доводится до всех студентов.

Оформление работы студентом: студент форматирует свой документ в текстовом редакторе (MS Word, Google Docs), руководствуясь методичкой. Часто возникают ошибки из-за непонимания требований, использования неверных стилей или технических сложностей редактора.

Передача на проверку: готовая работа передается преподавателю по электронной почте или через файлообменник.

Ручная проверка преподавателем: преподаватель открывает документ и визуально или с помощью линейки и меню редактора проверяет параметры: сравнивает шрифт в разных разделах, измеряет поля, проверяет интервалы, нумерацию страниц, оформление заголовков, списков литературы. Обнаруженные ошибки фиксируются вручную.

Исправление и повторная сдача: студент получает замечания, исправляет работу и снова отправляет ее. Цикл может повторяться несколько раз.

Выявленные проблемы:

Высокая трудоемкость и субъективность: ручная проверка отнимает значительное время. Разные преподаватели могут по-разному интерпретировать одни и те же требования.

Неоперативность обратной связи: студент узнает об ошибках оформления лишь после завершения проверки преподавателем, что затягивает процесс.

Отсутствие единого эталона: даже при наличии методички, проверка "на глазок" не гарантирует точного соответствия всем числовым параметрам.

Сложность анализа и статистики: невозможно быстро получить сводную статистику по типичным ошибкам потока для корректировки учебного процесса.

Отсутствие истории и преемственности: результаты проверки не сохраняются в структурированном виде, что затрудняет отслеживание прогресса студента.

На основе анализа предметной области были выделены следующие ключевые роли:

Студент: основной пользователь. Заинтересован в быстрой самопроверке работы перед сдачей, получении четкого списка ошибок с указанием мест в документе и рекомендаций по исправлению.

Преподаватель: отвечает за создание, настройку стандартов, просмотр результатов проверки работ.

Администратор системы: обеспечивает техническую эксплуатацию: развертывание, резервное копирование, мониторинг, управляет учетными записями.

Сводка ключевых функциональных требований:

Для Студента: загрузка DOCX-файла, выбор стандарта для проверки, просмотр интерактивного отчета с детализацией ошибок (тип, описание, место в документе, рекомендация), история собственных проверок.

Для Преподавателя: все функции студента, а также: доступ к списку загруженных работ студентов, настройка персональных стандартов, формирование аналитических отчетов.

Для Администратора: CRUD-управление (создание, чтение, обновление, удаление).

Общие: безопасная аутентификация и авторизация, адаптивный веб-интерфейс.

Для реализации описанных функций необходимо четко определить, какие данные система получает на вход и какие данные, информацию и артефакты

она формирует на выходе. Это позволит корректно спроектировать интерфейсы, алгоритмы обработки и структуру базы данных.

Входные данные системы формируются из трех основных источников. Первый источник — это методические требования (эталон оформления), которые поступают от преподавателя или администратора. Эти требования представляют собой структурированные данные, конфигурируемые через интерфейс системы, и в итоге хранятся в базе как набор правил, определяющих параметры шрифтов, полей, интервалов, заголовков и других элементов оформления.

Второй ключевой вход — это документ для проверки, загружаемый студентом или преподавателем в формате DOCX. Вместе с файлом передаются сопроводительные данные: выбранный стандарт проверки и идентификатор пользователя.

Третий тип входных данных — это команды и запросы пользователей. Сюда входят учетные данные для входа (email и пароль), команда на запуск проверки конкретного документа по определенному стандарту, а также различные запросы на получение истории, аналитики или списков. Сюда же относятся данные из форм, заполняемых пользователями при создании стандартов или редактировании профилей.

Выходные данные системы представлены несколькими категориями. Основным продуктом является отчет о результатах проверки, доступный как интерактивная веб-страница и через API. Его содержимое включает сводную статистику по проверке (общую оценку, количество найденных ошибок) и детализированный список всех нарушений с указанием типа, описания, критичности, места в документе и рекомендации по исправлению. Для наглядности в отчете может быть реализован просмотр документа с визуальной привязкой ошибок.

Для преподавателей и администраторов система предоставляет аналитические отчеты и статистику в виде дашбордов с графиками и таблицами, отражающими типичные ошибки по потоку, успеваемость студентов и общую нагрузку на систему, с возможностью экспорта данных.

К служебным выходным данным относятся подтверждения и системная информация — уведомления о статусе операций (например, об успешном завершении проверки) и история действий пользователя, доступная в личном кабинете.

Наконец, важным выходным результатом работы системы являются структурированные данные для хранения. Все ключевые события, такие как загрузка документа или завершение проверки, фиксируются в базе данных: создаются записи в таблицах документов, результатов проверки и, что наиболее важно, формируется детальный перечень всех обнаруженных нарушений в соответствующей таблице.

1.2 Проектирование диаграммы прецедентов и диаграммы деятельности

Диаграмма прецедентов (Use Case Diagram) — это модель, которая визуализирует взаимодействие внешних пользователей (акторов) с системой. Она показывает ключевые функции (прецеденты), которые система предоставляет для достижения целей пользователей, и определяет границы системы.

На рисунке 1.1 представлена диаграмма прецедентов системы автоматизированного контроля оформления учебных работ. Диаграмма отражает, как основные акторы — Гость, Студент, Преподаватель и Администратор — взаимодействуют с системой, выполняя такие задачи, как загрузка и проверка документов, управление стандартами оформления и администрирование системы.

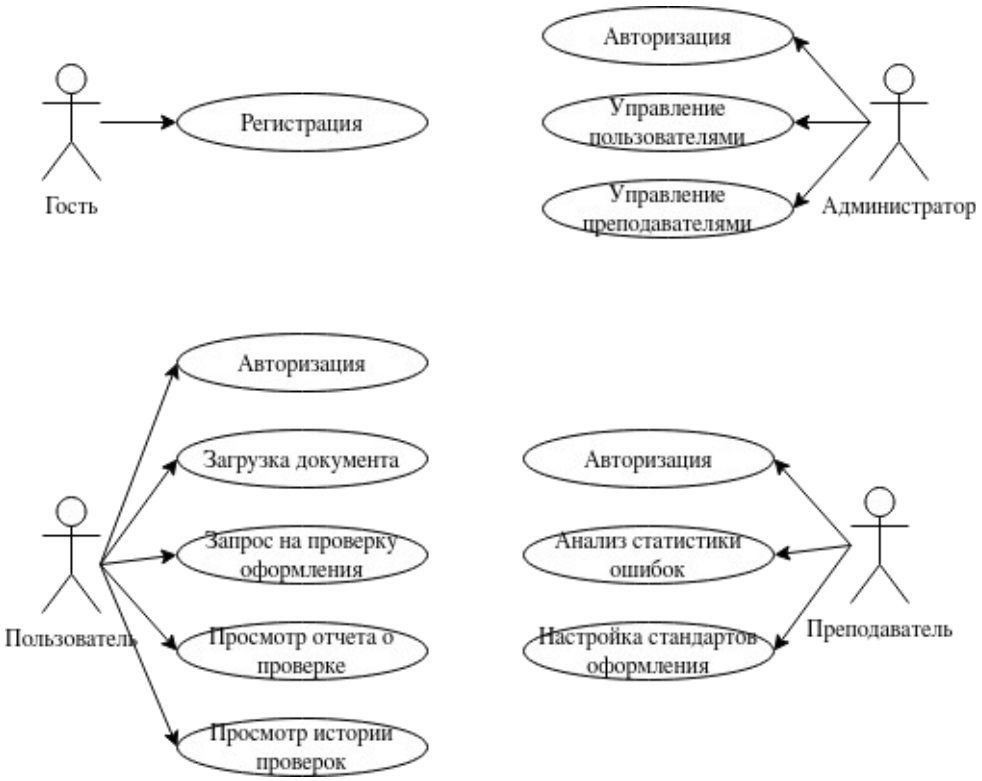

Рисунок 1.1 – Диаграмма прецедентов

Диаграмма деятельности (Activity Diagram) — это модель, которая детально описывает логику выполнения бизнес-процесса или алгоритма работы системы. Она визуализирует последовательность действий, точки принятия решений, параллельные потоки и взаимодействие различных участников процесса.

На рисунке 1.2 представлена диаграмма деятельности для основного процесса проверки оформления учебной работы в системе.

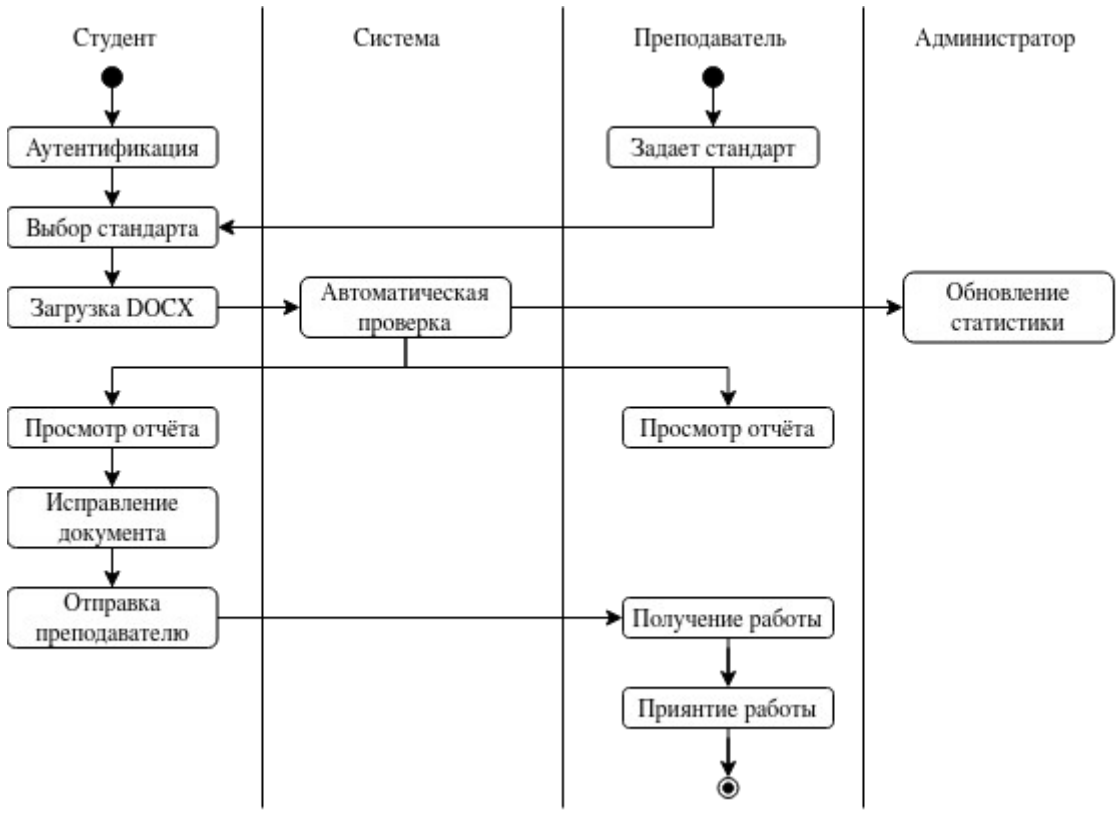

Рисунок 1.2 – Диаграмма деятельности

1.3 Проектирование схемы данных и словаря данных

Схема данных (Data Schema) — это формализованное представление структуры базы данных системы. Она определяет состав таблиц, их атрибуты (поля), типы данных, ограничения, а также ключевые связи между таблицами. Схема данных служит основой для физической реализации хранилища и обеспечивает целостность, непротиворечивость и эффективный доступ к информации.

На рисунке 1.3 представлена схема данных системы контроля оформления. Схема отображает основные сущности, такие как Пользователи, Стандарты оформления, Документы, Результаты проверок и Нарушения, а также связи между ними.

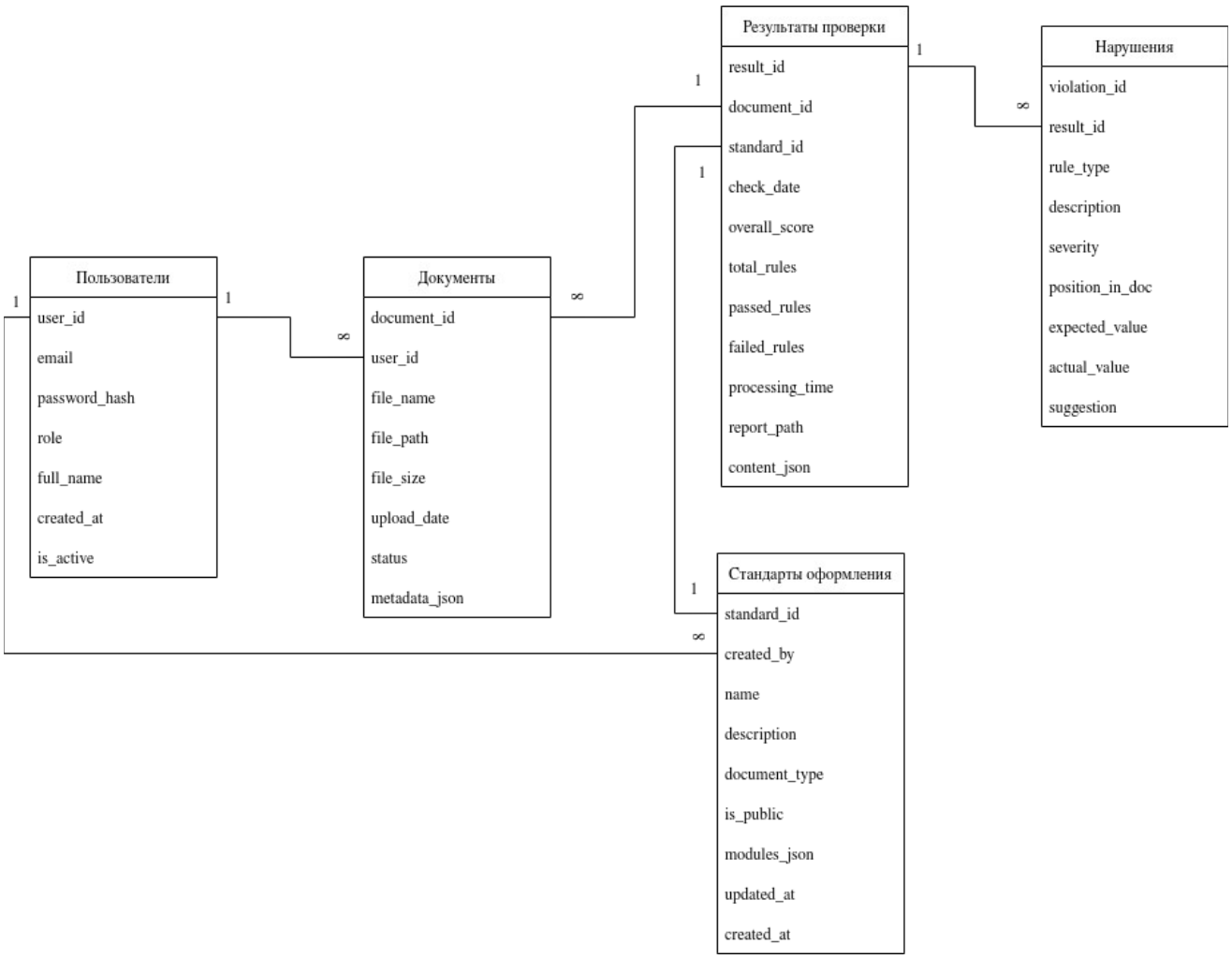

Рисунок 1.3 – Схема данных

Словарь данных (Data Dictionary) — это структурированная метабаза, содержащая подробное описание всех элементов данных, используемых в системе. Он документирует таблицы, атрибуты, их типы, ограничения, значения по умолчанию, бизнес-логику и взаимосвязи. Словарь данных служит единым источником истины о структуре информации, обеспечивая согласованность между аналитиками, разработчиками и администраторами базы данных.

Далее приведен словарь данных для системы автоматизированного контроля оформления, детализирующий состав и назначение полей в ключевых таблицах: Пользователи, Стандарты оформления, Документы, Результаты проверки и Нарушения.

Таблица 1 – Словарь данных пользователя

Атрибут	**Тип**	**Ограничения**	**Примечание**

	данных		
id	INTEGER	PRIMARY KEY	Уникальный ID пользователя
email	TEXT	NOT NULL, UNIQUE	Электронная почта
password_hash	TEXT	NOT NULL	Хэш пароля
role	TEXT	NOT NULL	Роль (student, teacher, admin)
full_name	TEXT		ФИО пользователя
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата регистрации
is_active	BOOLEAN	DEFAULT TRUE	Активен ли пользователь

Таблица 2 – Словарь данных стандарты оформления

Атрибут	**Тип данных**	**Ограничения**	**Примечание**
id	INTEGER	PRIMARY KEY	Уникальный ID стандарта
name	TEXT	NOT NULL	Название стандарта
description	TEXT		Описание стандарта
created_by	INTEGER		Кто создал стандарт (ID пользователя)
document_type	TEXT		Тип документа (курсовая, диплом и др.)
is_public	BOOLEAN	DEFAULT FALSE	Публичный стандарт
modules_json	TEXT		Конфигурация модулей в JSON
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата создания
updated_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата обновления

Таблица 3 – Словарь данных документов

Атрибут	Тип данных	Ограничения	Примечание
id	INTEGER	PRIMARY KEY	Уникальный ID документа
user_id	INTEGER		Владелец документа (ID пользователя)
file_name	TEXT		Имя файла
file_path	TEXT		Путь к файлу на диске
file_size	INTEGER		Размер файла в байтах
upload_date	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата загрузки
status	TEXT		Статус (new, processing, checked)
metadata_json	JSONB		Метаданные документа

Таблица 4 – Словарь данных результатов проверки

Атрибут	Тип данных	Ограничения	Примечание
id	INTEGER	PRIMARY KEY	Уникальный ID результата
document_id	INTEGER		ID проверяемого документа
standard_id	INTEGER		ID стандарта, по которому проверяли
check_date	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата проверки
overall_score	REAL		Общая оценка
total_rules	INTEGER		Всего правил
passed_rules	INTEGER		Пройдено правил
failed_rules	INTEGER		Провалено правил
processing_time	INTEGER		Время обработки (мс)
report_path	TEXT		Путь к файлу отчета
content_json	TEXT		Распаршенный контент для просмотра

Таблица 5 – Словарь данных нарушений

Атрибут	Тип данных	Ограничения	Примечание

id	INTEGER	PRIMARY KEY	Уникальный ID нарушения
result_id	INTEGER		ID результата проверки
rule_type	TEXT		Тип нарушенного правила
description	TEXT		Описание нарушения
severity	TEXT		Критичность (critical, error, warning)
position_in_doc	TEXT		Позиция в документе
expected_value	TEXT		Ожидаемое значение
actual_value	TEXT		Фактическое значение
suggestion	TEXT		Рекомендация по исправлению

1.4 Проектирование интерфейса информационной системы

Для проектирования информационной системы использовались следующие цвета

#F4F4F4 — Основной фон приложения;

#FFFFFF — Фон карточек и поверхностей;

#E5E5E5 — Второстепенный фон / нейтральный серый;

#000000 — Основной текст, границы, строгие линии;

#555555 — Второстепенный / затемненный текст;

#FF3B30 — Основной акцент;

#008000 — Цвет успеха;

#FF9500 — Цвет предупреждения.

На рисунке 1.4 представлен интерфейс для входа и регистрации пользователя.

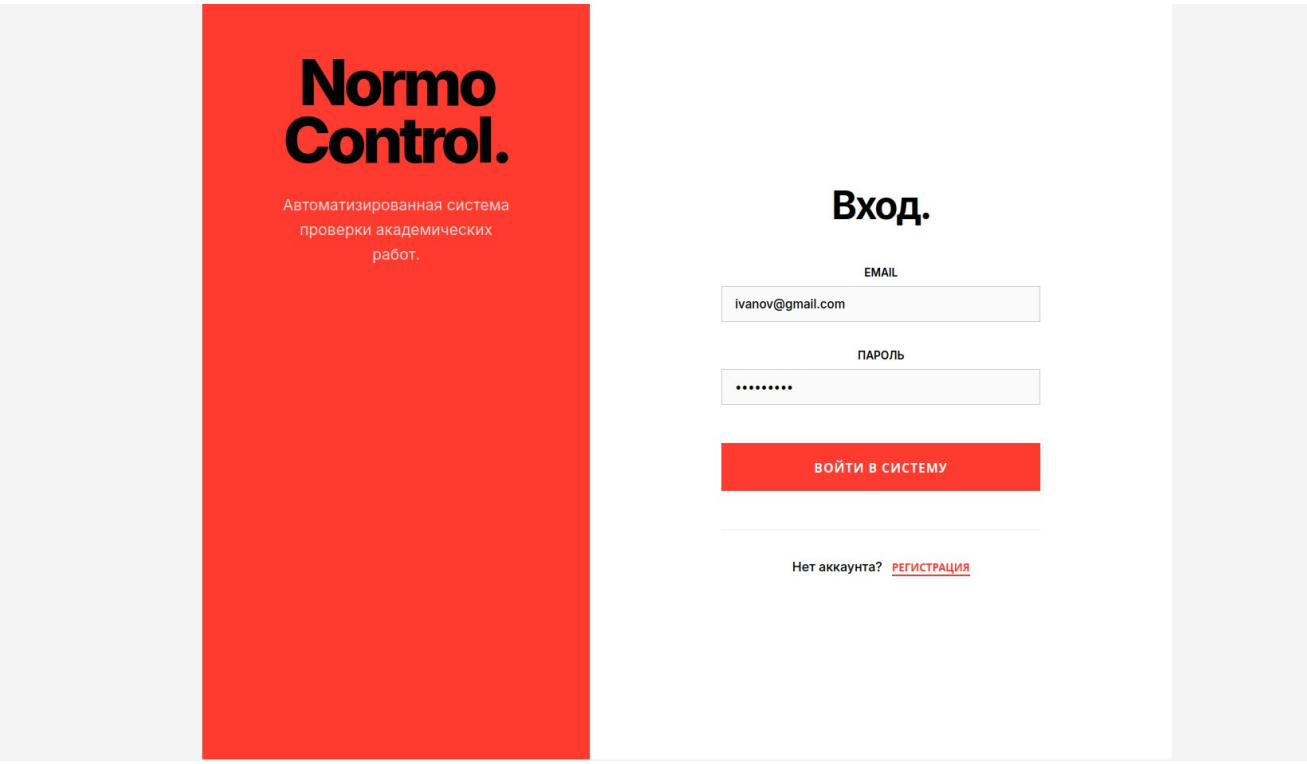

Рисунок 1.4 – Проектирование интерфейса входа и регистрации

1.5 Выбор аппаратных и технических средств для разработки информационной системы

Для разработки и реализации данной информационной системы используется комплекс современных аппаратных и программных средств, обеспечивающий надежность и масштабируемость решения.

Разработка велась на персональном рабочей станции с конфигурацией среднего уровня, достаточной для комфортной работы со средами программирования и локального тестирования контейнеров (процессор Intel Core i5, 16 ГБ оперативной памяти, SSD-накопитель объемом 512 ГБ).

Для эксплуатации системы в производственной среде используется выделенный виртуальный сервер под управлением операционной системы Ubuntu. Это позволяет обеспечить круглосуточную доступность сервиса и стабильную производительность. Конфигурация сервера включает: vCPU на базе Intel Xeon Platinum, 1 ГБ RAM, 10 ГБ NVMe накопитель и интернет-канал с пропускной способностью 50 Мбит/с.

С технической точки зрения серверная часть приложения построена на высокопроизводительном языке программирования Go. В основе архитектуры лежит легковесный веб-фреймворк Gin, который отвечает за

обработку HTTP-запросов и реализацию REST API. Хранение данных организовано с использованием встраиваемой реляционной системы управления базами данных SQLite, работающей с локальным файлом, что упрощает администрирование. Безопасность и управление сессиями пользователей обеспечиваются технологией JSON Web Tokens (JWT).

Клиентская часть представляет собой интерактивное веб-приложение, разработанное с использованием библиотеки React и инструмента сборки Vite для быстрой компиляции кода. Визуальная составляющая реализует принципы «Швейцарского дизайна» через кастомные CSS-стили, дополненные анимациями на базе Framer Motion и системой уведомлений React Toastify. Вся инфраструктура проекта контейнеризирована с помощью Docker и управляется через Docker Compose, что гарантирует идентичность окружения на этапах разработки и финального развертывания.

2. Разработка информационной системы

2.1 Разработка интерфейса информационной системы

Разработка визуальной оболочки информационной системы была направлена на обеспечение максимального удобства и простоты взаимодействия пользователей с функциональными возможностями программного продукта. При проектировании интерфейса основной упор был сделан на принципы эргономичности и логичности расположения элементов, что выражается в лаконичном дизайне, четкой структуре блоков и отсутствии избыточной графической информации. Такой подход позволяет пользователю сконцентрироваться на выполнении ключевых задач и минимизирует время на освоение системы.

2.2 Разработка базы данных информационной системы

Для хранения и управления данными в разрабатываемом веб-приложении используется реляционная база данных SQLite, которая обеспечивает надежное хранение структурированных данных и удобство развертывания. В ходе проектирования были созданы таблицы «users», «formatting_standards», «documents», «check_results» и «violations», каждая из которых отражает определённую сущность системы и имеет соответствующую структуру. Графическое представление этих таблиц приведено на рисунках 2.2.1-2.2.5.

2.3 Разработка функциональной части информационной системы

Реализация функциональной части системы выполнялась в рамках клиент-серверной архитектуры. Клиентская часть отвечает за визуализацию данных и взаимодействие с пользователем, в то время как серверная часть обеспечивает выполнение бизнес-логики и управление данными.

Центральным элементом функциональной части является модуль автоматизированного нормоконтроля. Процесс проверки реализован в виде алгоритмического сопоставления структуры загруженного файла с правилами, заданными в выбранном стандарте. Система выполняет парсинг метаданных каждого параграфа, извлекая параметры шрифтов, отступов и межстрочных интервалов. На основе выявленных отклонений формируется детализированный массив нарушений, рассчитывается итоговый рейтинг документа по 100-балльной шкале и генерируется интерактивный отчет для пользователя

Ключевой точкой входа в систему является модуль аутентификации и регистрации. Он необходим для идентификации субъектов доступа и разграничения их прав в системе. Интерфейс, предоставляющий пользователю возможность создать новую учетную запись, показан на рисунке 2.3.1.

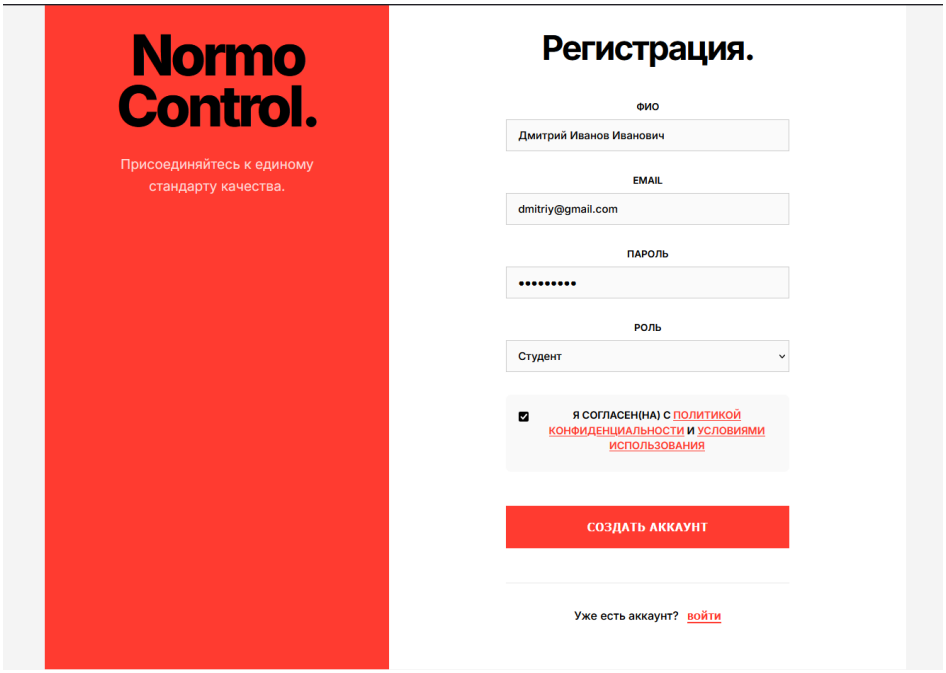

Рисунок 2.3.1 – Форма регистрации

Ниже представлен код:

```
const handleSubmit = async (e) => {
e.preventDefault();
setError('');
setValidationErrors({});
const errors = validateForm();
if (Object.keys(errors).length > 0) {
setValidationErrors(errors);
return;}
setLoading(true);
try {
```

```
await register(formData.email, formData.password, formData.fullName,
formData.role);
onSwitch();
} catch (err) {
setError(err.message || 'Ошибка регистрации');
} finally {
setLoading(false);}};
const register = async (email, password, fullName, role) => {
try {
const res = await fetch('http://localhost:8080/api/auth/register', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({
email,
password,
full_name: fullName,
role})});
const data = await res.json();
if (!res.ok) {
showToast.error(data.error || toastMessages.registerError);
throw new Error(data.error || 'Ошибка регистрации');}
showToast.success(toastMessages.registerSuccess);} catch (error) {
if (!error.message.includes('Ошибка регистрации')) {
showToast.error(toastMessages.networkError);}
throw error;}};
```

Передача данных формы регистрации на сервер представлена на рисунке 2.3.2.

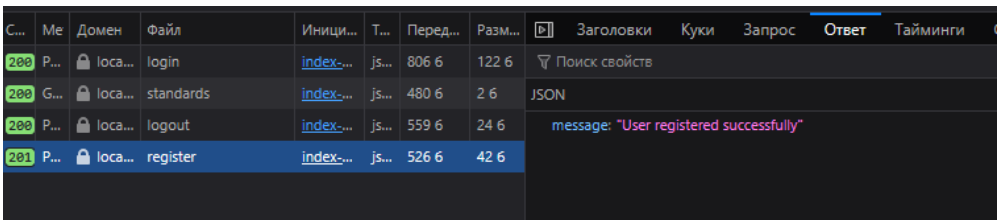

Рисунок 2.3.2 – Передача данных формы регистрации на сервер

2.4 Разработка подсистемы безопасности информационной системы

Подсистема безопасности информационной системы реализует комплексный подход к защите данных и разграничению доступа. Аутентификация пользователей осуществляется с помощью технологии JWT, при этом пароли хранятся в базе данных исключительно в виде криптографически стойких хешей (bcrypt), а сессионные токены передаются клиенту в защищенных HttpOnly Cookie, что исключает возможность их перехвата через XSS-атаки. Доступ к защищенным маршрутам API контролируется специальным промежуточным ПО (Middleware), которое проверяет валидность токена и блокирует неавторизованные запросы.

Для предотвращения SQL-инъекций взаимодействие с базой данных построено исключительно на параметризованных запросах, что делает невозможным внедрение вредоносного кода через поля ввода. Целостность данных дополнительно обеспечивается строгой серверной валидацией всех входящих параметров, отсеивающей некорректные форматы данных на этапе приема запроса. Сетевая безопасность поддерживается настройкой политики CORS, разрешающей взаимодействие с API только легитимному клиентскому приложению.

3. Тестирование информационной системы

Тестирование системы проводилось для того, чтобы убедиться в её надежности и соответствии требованиям. Основной метод проверки — ручное тестирование, в ходе которого проверялись все ключевые функции: от регистрации пользователей до проверки документов на соответствие стандартам. Главная цель — гарантировать, что система работает корректно и безопасна для использования.

4. Оценивание информационной системы для выявления возможности ее модернизации

После проведения детального анализа текущего состояния разработанной информационной системы автоматизированной проверки академических работ было выполнено оценивание ее функциональных возможностей, архитектурных решений и потенциала для дальнейшего развития. Данная оценка проводилась с целью определения направлений модернизации, которые позволят повысить эффективность образовательного процесса, снизить нагрузку на преподавательский состав и обеспечить соответствие растущим требованиям к качеству оформления студенческих работ. В процессе анализа рассматривались как технические метрики производительности алгоритмов проверки, так и отзывы пользователей, что позволило сформировать комплексное представление о текущих возможностях системы и перспективах ее развития.

В рамках проведенного анализа было установлено, что существующая система успешно справляется с базовыми задачами нормоконтроля, обеспечивая высокую точность выявления отклонений от стандартов оформления. Однако в ходе опытной эксплуатации были выявлены потенциальные точки роста, реализация которых может вывести процесс проверки работ на качественно новый уровень. Особое внимание было уделено вопросам бесшовной интеграции в существующую IT-инфраструктуру образовательного учреждения и расширению интеллектуальных возможностей анализа текста.

Основным направлением модернизации является интеграция с системами управления обучением (LMS), такими как Moodle. В настоящее время процесс проверки требует от студента ручной загрузки документа в систему, а результаты проверки существуют изолированно от электронного журнала успеваемости. Планируется внедрение двустороннего обмена данными через API, который позволит автоматически передавать проверенные работы и оценки непосредственно в личный кабинет студента и ведомость преподавателя. Это решение устранит дублирование действий при работе с разными информационными системами вуза и обеспечит автоматическую синхронизацию статусов проверки работ, создавая единое информационное пространство для участников учебного процесса.

Вторым важным направлением развития системы является разработка плагинов для текстовых редакторов (MS Word, Google Docs), что изменит парадигму взаимодействия пользователя с системой. Переход от «постфактум» проверки загруженного файла к проверке в режиме реального времени позволит студентам исправлять ошибки оформления непосредственно в процессе написания работы. Такое решение не только снизит количество итераций проверки для преподавателя, но и будет способствовать формированию у студентов устойчивых навыков работы с нормативными требованиями. Плагин обеспечит мгновенную обратную связь, подсвечивая несоответствия стандартам прямо в интерфейсе редактора.

Реализация указанных направлений модернизации позволит существенно повысить эффективность использования системы автоматизированной проверки, превратив ее из инструмента контроля в полноценного цифрового ассистента.

Заключение

В ходе работы над курсовым проектом была успешно решена основная задача — разработка web-приложения для автоматизированной проверки академических работ на соответствие стандартам оформления. Все поставленные цели были достигнуты, что позволило создать целевое, стабильное и удобное приложение, учитывающее специфику образовательного процесса и требования к оформлению документации.

Основные результаты и реализованный функционал:

Для студентов:

Безопасный доступ в личный кабинет с возможностью просмотра истории проверок;

Загрузка работ в систему с мгновенным получением результатов анализа;

Подробный отчет об ошибках: система не только указывает на наличие несоответствия, но и выделяет проблемные места в тексте документа, что позволяет студенту самостоятельно исправить недочеты без привлечения преподавателя.

Для преподавателей:

Просмотр расширенной статистики по студентам и группам, включая распределение баллов и типичные ошибки;

Доступ к детальной истории проверок каждого студента, что позволяет отслеживать прогресс исправления работы;

Снижение рутинной нагрузки: преподаватель избавлен от необходимости с линейкой вымерять отступы и шрифты, сосредотачиваясь на проверке смыслового содержания работ.

Для администраторов:

Полное управление пользователями: регистрация, блокировка и редактирование учетных записей студентов и преподавателей;

Управление базой стандартов: гибкий редактор правил оформления позволяет оперативно вносить изменения в требования без переписывания программного кода системы;

Мониторинг общей статистики использования системы для оценки нагрузки и эффективности.

Ключевые преимущества приложения:

Объективность и точность: автоматизированная проверка исключает «человеческий фактор», гарантируя беспристрастную оценку оформления работы в строгом соответствии с заложенными алгоритмами;

Скорость и эффективность: время проверки одной работы сокращается с десятков минут до нескольких секунд, обеспечивая мгновенную обратную связь для студента;

Удобство и масштабируемость: микросервисная архитектура и веб-интерфейс позволяют использовать систему с любого устройства без установки дополнительного ПО, а гибкая настройка стандартов делает решение применимым для проверки различных типов работ (курсовые, ВКР, рефераты);

Безопасность данных: реализована надежная система аутентификации и разделения прав доступа, защищающая как личные данные пользователей, так и интеллектуальную собственность (загруженные работы).

Недостатки и перспективы развития:

Ограниченность форматов: в текущей версии основной акцент сделан на проверку готовых DOCX-документов; требуется расширение поддержки исходных форматов для более глубокого анализа структуры;

Отсутствие интеграции: на данном этапе система работает автономно; перспективным направлением является внедрение API для прямой связи с корпоративными LMS (например, Moodle) для автоматического выставления оценок;

Статический анализ: система проверяет работу уже после её написания; развитие направления плагинов для текстовых редакторов позволило бы исправлять ошибки в режиме реального времени.

В заключение можно сказать, что разработанная информационная система представляет собой готовое к внедрению решение, которое кардинально модернизирует устаревший процесс ручного нормоконтроля. Система ликвидирует необходимость многократных очных консультаций по оформлению, минимизирует субъективность оценки и значительно повышает качество подготавливаемых студентами документов. Внедрение

автоматизированной проверки создает надежную цифровую основу для повышения эффективности всего образовательного процесса на кафедре.

Список используемой литературы

1. React : офиц. сайт. Менло-Парк, 2013 – . Обновляется в течение суток. URL: https://react.dev/

2. Vite : офиц. сайт. 2020 – . Обновляется в течение суток. URL: https://vitejs.dev/guide/

3. The Go Programming Language : офиц. сайт. Маунтин-Вью, 2009 – . Обновляется в течение суток. URL: https://go.dev/doc/

4. Gin Web Framework : офиц. сайт. 2014 – . Обновляется в течение суток. URL: https://gin-gonic.com/docs/

5. SQLite : офиц. сайт. Шарлотт, 2000 – . Обновляется в течение суток. URL: https://www.sqlite.org/docs.html

6. JSON Web Tokens (JWT) : сайт. 2013 – . Обновляется в течение суток. URL: https://jwt.io/introduction

7. React Router : офиц. сайт. 2014 – . Обновляется в течение суток. URL: https://reactrouter.com/en/main

8. Chart.js : офиц. сайт. 2013 – . Обновляется в течение суток. URL: https://www.chartjs.org/docs/latest/

9. React-pdf : сайт. 2017 – . Обновляется в течение суток. URL: https://github.com/wojtekmaj/react-pdf

10. Docker : офиц. сайт. Пало-Альто, 2013 – . Обновляется в течение суток. URL: https://docs.docker.com/

11. OWASP Top Ten : сайт. 2003 – . Обновляется по мере выхода релизов. URL: https://owasp.org/www-project-top-ten/

12. MDN Web Docs : сайт / учредитель Mozilla Foundation. Маунтин-Вью, 2005 – . Обновляется в течение суток. URL: https://developer.mozilla.org/

13. Go Cryptography (x/crypto) : пакет расширений : офиц. сайт. 2012 – . Обновляется в течение суток. URL: https://pkg.go.dev/golang.org/x/crypto

14. React Toastify : офиц. сайт. 2017 – . Обновляется в течение суток. URL: https://fkhadra.github.io/react-toastify/introduction/

15. GORM : офиц. сайт. 2013 – . Обновляется в течение суток. URL: https://gorm.io/docs/index.html

Приложение А – Инструкция пользователя и администратора

Для корректной работы системы на рабочем компьютере или сервере должна быть установлена платформа контейнеризации Docker и инструмент Docker Compose.

Приложение Б – Электронный формат пояснительной записки и информационной системы