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 И РЕАЛИЗАЦИЯ АРХИТЕКТУРЫ КЛИЕНТСКОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЯ «ЭЛЕКТРОННЫЙ КОРРЕКТОР»

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Деминой Ксении Дмитриевны

Научный руководитель
доцент, к. ф.-м. н. _____ С. В. Папшев

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

Саратов 2026

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ . . . . . 3
КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ . . . . . 5
ЗАКЛЮЧЕНИЕ . . . . . 12

# ВВЕДЕНИЕ

Развитие цифровых технологий и расширение электронного документооборота привели к значительному росту требований к качеству оформления текстовых материалов. В образовательных учреждениях, научных организациях и коммерческих компаниях ежедневно подготавливаются документы, которые должны соответствовать государственным стандартам оформления. Проверка подобных требований традиционно выполняется вручную, что требует значительных временных затрат и повышает вероятность пропуска ошибок оформления.

Существующие программные решения ориентированы преимущественно на проверку орфографических, грамматических и стилистических ошибок и практически не затрагивают вопросы проверки документов на соответствие требованиям ГОСТ. В связи с этим возникает необходимость разработки специализированных программных средств, способных выполнять автоматизированный нормоконтроль структуры и оформления документов.

Для решения данной задачи в рамках программы «Стартап как диплом» разрабатывается веб-платформа «Электронный корректор», предназначенная для автоматизированной проверки PDF- и DOCX-файлов на соответствие требованиям ГОСТ. Платформа представляет собой комплексное программное решение, включающее клиентскую часть, модуль интеллектуальной обработки документов и серверную инфраструктуру.

**Целью работы** является снижение трудозатрат при подготовке текстовых документов в соответствии с ГОСТ при поддержке клиентского интерфейса веб-приложения «Электронный корректор».

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

1. Проанализировать предметную область и существующие решения в сфере автоматизированной проверки документов.
2. Разработать требования к клиентской части системы и выбрать технологический стек и архитектурный подход для их реализации.
3. Разработать архитектуру клиентского веб-приложения и реализовать основные страницы интерфейса, включая механизмы регистрации, аутентификации и авторизации пользователей.
4. Реализовать функциональность загрузки документов и взаимодействие с серверным API, включая корректную обработку ответов и отображение

результатов проверки.

5. Разработать комплекс автоматизированных тестов, провести тестирование разработанного функционала и оценить его соответствие требованиям проекта.

**Структура и объём работы**. Бакалаврская работа состоит из введения, трёх разделов, заключения, списка использованных источников и семи приложений. Общий объём работы — 70 страниц, из них 54 страницы основного текста, включая 17 рисунков и 1 таблицу. Список использованных источников насчитывает 22 наименования.

## КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

**Первый раздел** «Аналитико-методологические основы разработки клиентского веб-приложения» посвящён анализу предметной области, обоснованию выбора технологий и описанию архитектурных решений.

В первом разделе проведён анализ существующих инструментов автоматизированной проверки документов: текстовых редакторов (MS Word, Google Docs) и онлайн-сервисов проверки текста. Традиционная проверка документов на соответствие стандартам, таким как ГОСТ, выполняется преимущественно вручную преподавателями или специалистами организаций, что требует высокой квалификации и значительных временных затрат. Сравнительный анализ показал, что существующие программные решения ограничены проверкой базового форматирования, не поддерживают работу с PDF-документами, не учитывают требования ГОСТ и не предоставляют возможности настройки пользовательских правил. Проведённый анализ выявил отсутствие готовых инструментов, сочетающих проверку ГОСТ-форматирования с поддержкой PDF и DOCX, что подтвердило целесообразность разработки и позволило сформулировать конкретные требования к функциональности и интерфейсу клиентской части. Разрабатываемое решение отличается от существующих систем тем, что обеспечивает полную автоматизацию проверки на соответствие ГОСТ, работает с PDF и DOCX, анализируя шрифты, таблицы, иллюстрации и структуру документа, предусматривает возможность кастомной проверки по пользовательским правилам и формирует подробный отчёт с предложениями по исправлению.

Помимо этого, обоснован выбор архитектурного подхода одностраничного приложения (SPA), обеспечивающего динамическое обновление содержимого без перезагрузки страницы. Такой подход создаёт взаимодействие, при котором пользователь ощущает работу не с традиционным веб-сайтом, а с полноценным настольным приложением благодаря мгновенной реакции на действия и отсутствию задержек. Архитектура SPA подразумевает загрузку единственного HTML-документа, служащего оболочкой для всех страниц приложения, а дальнейшее взаимодействие происходит через динамическую подгрузку данных посредством HTTP-запросов. В контексте разрабатываемой платформы данный подход представляется оптимальным, поскольку основной функционал системы ориентирован на авторизованных пользователей, работающих с документами. В работе приведена обобщённая схема архитектуры SPA-приложения, отража-

ющая основные функциональные компоненты и характер их взаимодействия.

Архитектурное проектирование клиентской части опирается на принцип разделения ответственности. Компоненты представления отвечают исключительно за отображение информации и взаимодействие с пользователем. Бизнес-логика обработки данных вынесена в отдельные функции и хуки, что упрощает тестирование и позволяет повторно использовать логику в различных частях приложения. Взаимодействие с сервером организовано через специализированный слой API-сервисов, инкапсулирующий детали работы с HTTP-запросами и обработку ответов. Такое разделение позволяет изолировать изменения в одной части системы от других, что критически важно при долгосрочной поддержке проекта. Организация потока данных построена по однонаправленной модели, при которой информация передаётся от родительских компонентов к дочерним через свойства, а изменения состояния обрабатываются через функции обратного вызова. Такая архитектура делает поведение приложения предсказуемым и существенно упрощает процесс отладки. Модульная организация кода обеспечивается разделением функциональности на четыре независимых блока: модуль аутентификации, модуль работы с документами, модуль отображения результатов проверки и модуль управления профилем пользователя. Каждый модуль имеет чётко определённые точки входа и выхода, что позволяет разрабатывать и тестировать их изолированно.

Вопросы безопасности интегрированы в архитектуру на всех уровнях. Токены аутентификации хранятся в localStorage с соблюдением мер предосторожности против XSS-атак. Все входящие данные проходят валидацию на клиентской стороне для обеспечения немедленной обратной связи с пользователем. Защита маршрутов реализована через механизм проверки наличия действительного токена при попытке доступа к закрытым разделам приложения с автоматическим перенаправлением на страницу входа в случае отсутствия авторизации.

Описан технологический стек разработки клиентского приложения. Библиотека React предоставляет декларативный подход к созданию интерфейсов и использует виртуальный DOM для эффективного определения минимального набора изменений, необходимых для обновления реального DOM браузера. Компонентная архитектура React позволяет создавать переиспользуемые элементы интерфейса: форма регистрации, компонент загрузки документов, карточка профиля пользователя разработаны как самостоятельные модули, кото-

рые могут быть протестированы независимо и использованы в различных частях приложения. В проекте активно применяются React Hooks: `useState` для управления локальным состоянием, `useEffect` для выполнения побочных эффектов при монтировании или обновлении компонента, `useNavigate` для программной навигации и `useParams` для извлечения параметров из URL. TypeScript обеспечивает статическую типизацию, позволяя обнаруживать ошибки на этапе компиляции, а не во время выполнения программы. В проекте определены интерфейсы ключевых сущностей системы: `IDocument`, `IUser`, `ICheckResult`. Компонентная библиотека Material UI предоставляет набор готовых элементов интерфейса с поддержкой адаптивной вёрстки и современных стандартов доступности. Взаимодействие с серверной частью построено на использовании HTTP-клиента Axios, обеспечивающего централизованное управление конфигурацией запросов, включая автоматическое добавление токенов авторизации в заголовки. Навигация внутри приложения реализована с помощью React Router, обеспечивающего программное управление переходами и защиту маршрутов от несанкционированного доступа.

Описана методология автоматизированного тестирования веб-приложения, основанная на комплексном подходе. Обеспечение стабильной работы клиентской части невозможно без внедрения автоматизированного тестирования, поскольку ручная проверка при активной разработке не позволяет своевременно выявлять ошибки и контролировать появление регрессий. Для модульного тестирования применяется Jest, обеспечивающий выполнение тестов и контроль корректности работы отдельных функций и модулей со встроенными средствами мокирования и измерения покрытия кода. Тестирование компонентов интерфейса выполняется с помощью React Testing Library, которая позволяет оценивать поведение компонентов с точки зрения конечного пользователя, а не деталей реализации. Для сквозного тестирования используется Cypress — инструмент для автоматизированного тестирования в реальном браузере, воспроизводящий реальные действия пользователя.

**Второй раздел** «Проектирование и реализация клиентской части веб-приложения» содержит описание проектирования структуры компонентов, схемы маршрутизации и реализации всех функциональных модулей системы.

Во втором разделе сформулированы функциональные и нефункциональные требования к клиентской части. К функциональным требованиям отнесены

регистрация и аутентификация пользователей, загрузка документов в форматах PDF и DOCX, выбор режима проверки, отображение результатов анализа с детализацией нарушений и рекомендациями по их устранению, управление профилем и просмотр истории проверок. Нефункциональные требования включают надёжность, безопасность, удобство использования, совместимость с актуальными версиями браузеров на основе Chromium, а также в Firefox и Safari, и сопровождаемость кода. Интерфейс должен обеспечивать выполнение основного сценария — загрузки документа и получения результатов проверки — без необходимости обращения к инструкции. Приложение должно поддерживать тёмную и светлую цветовые темы с возможностью переключения, а уведомления об ошибках и успешных действиях отображаться во всплывающих сообщениях с автоматическим скрытием.

Разработана слоевая организация кодовой базы и определена схема маршрутизации приложения, включающая публичные маршруты (главная страница, страницы входа и регистрации) и защищённые маршруты, доступные только авторизованным пользователям. При отсутствии действительного токена пользователь автоматически перенаправляется на страницу входа с сохранением исходного пути для возврата после авторизации. В соответствии с таблицей 1 приведена схема маршрутов приложения:

Таблица 1 – Маршруты клиентского приложения

| **Маршрут** | **Компонент** | **Доступ** |
|---|---|---|
| / | Welcome | Публичный |
| /login | Login | Публичный |
| /check | CheckDocument | Защищённый |
| /gost-check/result/:id | CheckResult | Защищённый |
| /custom-check | CustomCheck | Защищённый |
| /custom-check/result | CustomCheckResult | Защищённый |
| /user-template-check | UserTemplateCheck | Защищённый |
| /user-template-check/result | UserTemplateResult | Защищённый |
| /profile | Profile | Защищённый |
| /profile/edit | EditProfile | Защищённый |

Модуль аутентификации включает сервис аутентификации, хук `useAuth`, компонент `ProtectedRoute`, формы входа и регистрации, а также интерцептор Axios. Сервис аутентификации инкапсулирует логику взаимодействия с API: отправку запросов на регистрацию и вход, сохранение и удаление JWT-токена в

localStorage. Хук `useAuth` управляет состоянием авторизации и предоставляет компонентам функции входа, регистрации и выхода из системы, а также флаг загрузки для отображения заглушки в период проверки токена. Интерцептор Axios автоматически добавляет токен к заголовкам всех исходящих запросов и обрабатывает ошибки авторизации: при получении ответа со статусом 401 токен удаляется из хранилища и пользователь перенаправляется на страницу входа. При загрузке файлов заголовок `Content-Type` удаляется из конфигурации, чтобы браузер самостоятельно установил корректное значение для формата `multipart/form-data`.

Модуль загрузки и проверки документов обеспечивает три режима проверки. Режим проверки по ГОСТ 7.32-2017 предусматривает загрузку документа на сервер и последующий асинхронный опрос статуса обработки через механизм polling: клиент периодически запрашивает статус до получения результата, после чего автоматически переходит на страницу результатов. Перед отправкой выполняется валидация файла по типу и размеру. Режим кастомной проверки позволяет загрузить PDF-файл с правилами ГОСТ и автоматически извлечь из него параметры для проверки целевого документа. Режим пользовательского шаблона предоставляет форму, в которой пользователь вручную задаёт параметры форматирования: название и размер шрифта, поля страницы в миллиметрах, абзацный отступ в сантиметрах и список обязательных разделов. Все поля являются необязательными — проверяются только те параметры, для которых задано значение; это поведение обеспечивается функцией `numOrNull`, преобразующей строку в число или возвращающей `null`.

На странице результатов проверки пользователь видит общую оценку соответствия документа и список найденных нарушений с указанием типа, описания, страницы и приоритета каждого нарушения, а также рекомендации по их устранению. Для кастомной проверки дополнительно отображается итоговый балл соответствия в числовом и процентном выражении. После получения результата он сохраняется в localStorage в двух форматах: краткая запись для отображения в истории проверок на странице профиля и полный результат для возможности повторного открытия. Время анализа фиксируется в момент получения результата и также сохраняется в localStorage.

Модуль профиля пользователя включает отображение персональных данных, загрузку аватара, смену пароля, статистику проверок и редактирование

профиля. Для вычисления среднего времени анализа разработан специализированный хук, считывающий данные о времени из localStorage и вычисляющий среднее значение по всем сохранённым проверкам. История проверок отображает ранее выполненные операции с возможностью повторного просмотра результатов. При загрузке данных пользователя выполняется запрос к эндпоинту профиля; полученные данные отображаются на странице и используются для предзаполнения формы редактирования.

**Третий раздел** «Тестирование и оценка результатов» посвящён верификации разработанного программного обеспечения.

В третьем разделе описано модульное и компонентное тестирование, выполненное с применением Jest и React Testing Library. Разработаны тесты для сервисного слоя, пользовательских хуков и компонентов интерфейса. Тесты сервисного слоя проверяют корректность формирования запросов к API и обработку ответов сервера, включая граничные случаи — обработку ошибок сети и некорректных данных. Тесты хуков верифицируют логику управления состоянием аутентификации: корректность установки флага авторизации при наличии токена, очистку состояния при выходе и вычисление статистики проверок. Тесты компонентов проверяют корректность отображения данных при различных состояниях — загрузке, успешном получении данных и возникновении ошибки — а также реакцию интерфейса на действия пользователя. Для изоляции тестов от внешних зависимостей применялось мокирование HTTP-запросов и браузерных API, в частности localStorage и window.location.

Всего разработано 92 теста в рамках 21 тестового файла, все тесты пройдены успешно.

Сквозное тестирование выполнено с применением Cypress. Разработаны три файла E2E-тестов, охватывающих основные пользовательские сценарии: регистрацию и вход в систему, загрузку документа и получение результатов проверки, работу с профилем пользователя. E2E-тесты воспроизводят реальное поведение пользователя в браузере, включая заполнение форм, навигацию между страницами и проверку отображаемых данных. Тестирование охватывает как позитивные сценарии — успешное выполнение операций, так и негативные — обработку ошибок валидации и недоступности сервера. По результатам тестирования подтверждена корректность работы всех ключевых компонентов системы. Разработанный комплекс тестов обеспечивает контроль качества при дальней-

шем развитии приложения и позволяет своевременно выявлять регрессии при внесении изменений в код.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была спроектирована и реализована клиентская часть веб-приложения «Электронный корректор», предназначенного для автоматизированной проверки текстовых документов на соответствие требованиям государственных стандартов оформления.

Проведён анализ предметной области и существующих решений, по результатам которого подтверждена целесообразность разработки и сформулированы функциональные и нефункциональные требования к клиентской части системы. Обоснован выбор технологического стека: библиотека React 19, язык TypeScript, компонентная библиотека Material UI, HTTP-клиент Axios и маршрутизатор React Router.

Спроектирована и реализована архитектура клиентского SPA-приложения, основанная на принципах модульности и разделения ответственности. Разработаны четыре функциональных модуля: модуль аутентификации с JWT-токенами и защитой маршрутов, модуль загрузки и проверки документов с поддержкой трёх режимов проверки и асинхронного опроса статуса, модуль отображения результатов проверки, а также модуль профиля пользователя со статистикой и историей проверок.

Разработан комплекс автоматизированных тестов: 92 unit-теста в 21 тестовом файле и 3 файла сквозных E2E-тестов. Тестирование подтвердило корректность работы всех ключевых компонентов системы.

По итогам участия в программе «Стартап как диплом» работе была дана высокая рекомендательная оценка. Разработанный клиентский интерфейс может быть применён в образовательных учреждениях и организациях, практикующих проверку документов на соответствие стандартам оформления.