

Е. А. Калинина^{1✉}, А. А. Шаранов¹

Разработка серверной части автоматизированной системы формирования цифрового портфолио обучающегося

¹Сибирский государственный университет геосистем и технологий,
г. Новосибирск, Российская Федерация
e-mail: kalininaevgeniaqq@gmail.com

Аннотация. В статье представлены основные результаты разработки серверной части автоматизированной системы формирования цифрового портфолио обучающегося для центра дополнительного образования. Проведён анализ предметной области и существующих решений, выявивший функциональный разрыв в части сквозного сопровождения конкурсной деятельности. Обоснован выбор технологического стека на основе Node.js, Express, PostgreSQL и JWT-аутентификации. Спроектирована модульная архитектура серверного приложения, реляционная база данных, включающая двенадцать таблиц с поддержкой мягкого удаления, ролевая модель из четырёх ролей и REST API из двенадцати групп маршрутов. Описана реализация ключевых модулей: аутентификации и разграничения прав, регистрации пользователей, подачи заявок на конкурсы, публикации результатов с выгрузкой в Excel и формированием ZIP-архивов дипломов. Проведено функциональное, нагрузочное тестирование и тестирование безопасности. Нагрузочные испытания подтвердили стабильную работу системы при одновременной работе ста пользователей при среднем времени отклика 2,44 мс и нулевом количестве ошибок. Разработанная система готова к внедрению в образовательных организациях.

Ключевые слова: цифровое портфолио, REST API, Node.js, PostgreSQL, аутентификация, дополнительное образование

Е. А. Kalinina^{1✉}, A. A. Sharapov¹

Development of the server part of an automated system for forming a student digital portfolio

¹Siberian State University of Geosystems and Technologies, Novosibirsk, Russian Federation
e-mail: kalininaevgeniaqq@gmail.com

Abstract. The paper presents the main results of developing the server-side of an automated system for forming a student's digital portfolio for a supplementary education center. An analysis of the subject area and existing solutions revealed a functional gap in end-to-end support of competition activities. The technology stack based on Node.js, Express, PostgreSQL and JWT authentication is justified. A modular server architecture, a relational database comprising twelve tables with soft-delete support, a four-role access model, and a REST API with twelve route groups were designed. The implementation of key modules is described: authentication and access control, user registration, competition application submission, results publication with Excel export and ZIP archive generation. Functional, load, and security testing were conducted. Load tests confirmed stable system operation with one hundred concurrent users, an average response time of 2.44 ms and zero errors. The developed system is ready for deployment in educational organizations.

Keywords: digital portfolio, REST API, Node.js, PostgreSQL, authentication, supplementary education

Введение

Учёт учебных и творческих достижений обучающихся на протяжении десятилетий осуществлялся преимущественно в бумажном формате: дипломы хранились в папках, итоги соревнований заносились в журналы вручную. Цифровизация образования облегчила процесс хранения таких данных, однако одновременно привела к их рассредоточению по разрозненным инструментам – электронной почте, табличным редакторам, ситуативным платформам – вследствие чего единая картина достижений учащегося оказалась труднодостижимой [1].

В организациях дополнительного образования указанная проблема проявляется наиболее выражено. Проведение каждого конкурсного мероприятия сопряжено с отдельным сбором заявок, автономным ведением протоколов и ручным оформлением наград для участников. В результате педагог вынужден многократно вносить одни и те же сведения в разные системы, тогда как обучающийся лишён возможности получить консолидированный доступ к собственным достижениям [2, 3].

Настоящая работа посвящена созданию серверной части программного комплекса, способного объединить перечисленные процессы в единый контур: от подачи заявки до публикации итогов и пополнения портфолио. Для достижения цели решались задачи обследования предметной области, сравнения существующих аналогов, проектирования структуры хранения данных и программного интерфейса, а также реализации и верификации прикладных модулей.

Анализ предметной области и существующих решений

Несмотря на признанную важность портфолио как инструмента фиксации индивидуального прогресса, процесс его формирования в большинстве учреждений остаётся децентрализованным и слабо автоматизированным. Анализ рынка показал, что организаторы конкурсов в сфере дополнительного образования чаще всего прибегают к универсальным конструкторам форм: Google Формы и Яндекс Формы позволяют оперативно развернуть анкету и собрать данные в таблицу [4]. Вместе с тем подобные решения не поддерживают разграничение ролей участников процесса, а портфолио на их основе не формируется в принципе.

Государственный портал «Навигатор дополнительного образования» решает иную задачу – каталогизацию программ – и не рассчитан на управление конкурсным процессом [5]. Платформа «ЯКласс» ориентирована на проверочные задания академического типа и не предусматривает работы с творческими конкурсными материалами [6]. Площадки «Парад талантов России» и «Конкурсы.рф» обслуживают открытые всероссийские мероприятия и не предоставляют возможности развернуть собственный экземпляр системы [7, 8].

Анализ имеющихся решений показывает, что каждое из них, как правило, направлено на автоматизацию какого-либо участка работы, тогда как сквозное сопровождение процесса остаётся нереализованным. Приём заявок, публикация итогов и ведение портфолио зачастую обеспечивается разными инструментами, между которыми отсутствует прямая передача данных. Это вынуждает либо

переносить сведения вручную, либо мириться с их рассредоточенностью. Платформы, претендующие на более полный охват задач, в большинстве случаев ориентированы на широкий круг применений и не предусматривают гибкой адаптации под условия работы отдельной организации. Ни один из рассмотренных сервисов не перекрывает весь перечень необходимых функций одновременно, что подтверждает актуальность разработки специализированного решения.

Выбор технологического стека

При выборе технологического стека принималась во внимание специфика предполагаемой нагрузки: сервер должен параллельно принимать загружаемые пользователями файлы, выполнять запросы к базе данных и генерировать документы на выгрузку. Из доступных серверных платформ предпочтение было отдано Node.js, прежде всего благодаря её однопоточной событийно-ориентированной модели с неблокирующим вводом-выводом, позволяющей обрабатывать большое число параллельных соединений без пропорционального роста расхода памяти. В качестве надстройки применён фреймворк Express –минималистичная библиотека маршрутизации, предоставляющая гибкий механизм цепочек промежуточных обработчиков (middleware), в которые естественным образом встраиваются слои аутентификации и валидации запросов [9].

Реляционная СУБД PostgreSQL выбрана с учётом двух ключевых критериев: декларативного контроля целостности данных средствами самой СУБД и наличия системы в реестре отечественного программного обеспечения Минцифры России [10]. Для управления сессиями применён беззнаковый подход: сразу после входа сервер формирует компактный JSON Web Token, самостоятельно несущий в полезной нагрузке сведения о пользователе и подписанный секретным ключом, что исключает необходимость обращаться к хранилищу сессий при каждом запросе [11]. Хранение паролей организовано с использованием адаптивной функции bcrypt на основе алгоритма Blowfish: соль, встраиваемая в хеш при каждом вычислении, и настраиваемый коэффициент сложности делают атаки по предвычисленным таблицам практически нецелесообразными [12]. Вспомогательные задачи – загрузка файлов, экспорт таблиц, архивирование, почтовые уведомления – реализованы средствами библиотек Multer [13], ExcelJS [14], Archiver [15] и Nodemailer [16] соответственно. Единообразное использование JavaScript на всех уровнях приложения снижает когнитивную нагрузку при сопровождении кода и упрощает переход разработчика между клиентской и серверной частями.

Проектирование системы

Архитектурно сервер представляет собой монолит с чёткой внутренней модульностью: каждая функциональная область изолирована в отдельный набор файлов, тогда как точка входа server.js играет роль компоновщика и не содержит бизнес-логики. Взаимодействие с клиентом осуществляется через REST API [19], что делает интерфейс независимым от реализации фронтенда. Внутри сервер поделён

на слои маршрутизации, промежуточной обработки, прикладной логики и файловой подсистемы.

Разграничение доступа строится на четырёх ролях. Администратор контролирует жизненный цикл учётных записей и рассматривает заявки на регистрацию. Модератор ведёт конкурсную программу: открывает приём работ, утверждает призёров, публикует итоги и управляет выставкой. Учитель прикреплен к конкретным ученикам и подаёт заявки исключительно от их имени. Ученик просматривает доступные конкурсы, самостоятельно подаёт работы и знакомится с накопленными наградами.

Схема базы данных включает двенадцать таблиц, распределённых по функциональным блокам: управление учётными записями (Users, Profiles, Registration_requests, Roles, Teacher_student), конкурсный контур (Competitions, Themes, Competition_applications, Application_attachments), блок итогов (Results, Degrees) и витрина выставки (Exhibition). Вместо физического удаления строк применено мягкое удаление: поле `deleted_at` фиксирует момент деактивации и позволяет сохранять историческую запись без нарушения ссылочной целостности. Ролевая модель разграничения доступа типа RBAC охватывает четыре роли; каждый маршрут API защищён двухэтапным конвейером промежуточных обработчиков – верификацией токена с последующей проверкой допустимой роли [17, 18].

Практическая реализация

Кодовая база организована по принципу разделения ответственности: директория `routes` хранит объявления маршрутов, `middleware` – повторно используемые обработчики, `utils` – вспомогательные утилиты, `uploads` – загружаемые файлы, разбитые на три поддиректории по типу контента. Подключение к PostgreSQL реализовано через пул соединений библиотеки `pg`: входящий запрос получает свободное соединение из пула и возвращает его после завершения транзакции, что устраняет накладные расходы на повторное установление TCP-соединения при каждом вызове базы данных.

При входе в систему сервер формирует JWT, вкладывая в полезную нагрузку идентификатор пользователя, идентификатор профиля, назначенную роль и логин; токен подписывается ключом из переменных окружения. При каждом запросе `middleware auth.js` извлекает токен из заголовка `Authorization` и верифицирует подпись; `middleware roleCheck.js` сопоставляет роль из токена с множеством ролей, допущенных к данному маршруту, и при несоответствии прерывает обработку с кодом 403.

Логика регистрации учитывает три ситуации: если в базе уже есть деактивированная запись с тем же номером телефона, аккаунт восстанавливается без создания дубликата; если записи нет, создаются новые строки во всех связанных таблицах; если активная запись существует, операция отклоняется с поясняющим сообщением. Экспортный модуль формирует многостраничную Excel-книгу; изображения конкурсных работ вставляются непосредственно в ячейки: сервер считывает заголовки JPEG и PNG для определения исходных размеров,

масштабирует их до 450 пикселей по большей стороне и освобождает временные файлы сразу после отправки архива клиенту.

Тестирование

Корректность реализованных сценариев проверялась вручную через Postman: для каждого маршрута формировались наборы корректных и заведомо ошибочных запросов, охватывающие граничные значения входных данных, нестандартные форматы файлов и попытки вызова маршрута с недопустимой ролью. Отдельно верифицировалась генерация Excel-выгрузок и ZIP-архивов дипломов. Нагрузочное испытание выполнено инструментом k6 [20]: виртуальные пользователи циклически выполняли четыре операции – запрос каталога конкурсов, вход в систему, чтение профиля и просмотр выставки. Измеренные показатели сведены в табл. 1.

Таблица 1. Результат нагрузочного тестирования

Показатель	Значение
Общее число запросов	15 644
Максимальное количество одновременных пользователей	100
Среднее время ответа	2,44 мс
Медианное время ответа	1,90 мс
95-й перцентиль времени ответа	3,31 мс
Количество ошибок	0
Процент успешных запросов	100 %

Нагрузочный сценарий со ста одновременными виртуальными пользователями завершился без единой ошибки: среднее время отклика составило 2,44 мс, медиана – 1,90 мс, 95-й перцентиль не превысил 3,31 мс. Тестирование защищённости охватывало три вектора. Инъекции SQL: в текстовые поля подставлялись выражения «UNION SELECT» и операторы «DROP»; параметризованные запросы к PostgreSQL полностью исключили интерпретацию введённых данных как кода. Несанкционированный доступ: запросы к защищённым маршрутам без токена возвращали статус 401, запросы с токеном неподходящей роли – статус 403. Перебор учётных данных: после пяти неверных попыток входа с одного IP-адреса система вводит временную блокировку длительностью пятнадцать минут.

Заключение

В результате выполненной работы спроектирован и реализован серверный компонент автоматизированной системы, обеспечивающей сквозное сопровождение конкурсной деятельности в центре дополнительного образования. Архитектура системы охватывает реляционную базу данных из двенадцати таблиц, ролевую модель управления доступом и REST API из двенадцати групп

маршрутов. Нагрузочные испытания при одновременной работе ста пользователей зафиксировали среднее время отклика 2,44 мс при полном отсутствии отказов; тестирование безопасности не выявило критических уязвимостей. Программный продукт готов к опытной эксплуатации в образовательных организациях и может быть адаптирован для смежных предметных областей, где требуется централизованный учёт достижений участников.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Архипенко, А. В. Автоматизированное электронное портфолио как метод оценки учебно-профессиональных достижений учащегося / А. В. Архипенко // Гуманизация образования. – 2024. – №2. URL: <https://cyberleninka.ru/article/n/avtomatizirovannoe-elektronnoe-portfolio-kak-metod-otsenki-uchebno-professionalnyh-dostizheniy-uchaschegosya-i-sredstvo-rovysheniya> (дата обращения: 07.05.2026). – Текст: электронный.
2. Пираков, Ф. Д. Разработка и применение системы повышения портфолио обучающегося в вузе / Ф. Д. Пираков, А. П. Клишин, Н. Л. Еремина, Е. Н. Клыжко // Вестник НГУ. Серия: Информационные технологии. – 2019. – № 4. – С. 87–100.
3. Курмангалиева, А. М. Разработка структуры информационной системы формирования портфолио обучающегося / А. М. Курмангалиева, М. М. Жаманкарин, М. К. Айткенова // Наука и реальность. – 2021. – № 2(6). – С. 228–231.
4. Google Forms : официальный сайт. – URL: <https://workspace.google.com/intl/ru/products/forms/> (дата обращения: 07.05.2026). – Текст: электронный.
5. Навигатор дополнительного образования детей : федеральный портал. – URL: <https://navigator.edu54.ru/> (дата обращения: 07.05.2026). – Текст: электронный.
6. ЯКласс : официальный сайт. – URL: <https://www.yaklass.ru/> (дата обращения: 07.05.2026). – Текст: электронный.
7. Парад талантов России : информационно-образовательный портал. – URL: <https://paradtalant.ru/> (дата обращения: 07.05.2026). – Текст: электронный.
8. Конкурсы.рф : официальный сайт. – URL: <https://конкурсы.рф> (дата обращения: 07.05.2026). – Текст: электронный.
9. Браун, И. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript / И. Браун. – Санкт-Петербург : БХВ-Петербург, 2021. – ISBN 978-5-4461-0590-8.
10. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. – Москва : Вильямс, 2005. – ISBN 978-5-8459-0788-4.
11. Jones, M. JSON Web Token (JWT) / M. Jones, J. Bradley, N. Sakimura // IETF. – 2015. – URL: <https://www.rfc-editor.org/info/rfc7519> (дата обращения: 09.05.2026). – Текст: электронный.
12. Шнейер, Б. На пути к Skein: просто и понятно про Blowfish / Б. Шнейер // Habr. – 2012. – URL: <https://habr.com/ru/articles/140394/> (дата обращения: 09.05.2026). – Текст: электронный.
13. Multer npm package documentation. – URL: <https://www.npmjs.com/package/multer> (дата обращения: 09.05.2026). – Текст: электронный.
14. ExcelJS documentation. – URL: <https://github.com/exceljs/exceljs> (дата обращения: 09.05.2026). – Текст: электронный.
15. Archiver documentation. – URL: <https://www.archiverjs.com/> (дата обращения: 09.05.2026). – Текст: электронный.
16. Nodemailer documentation. – URL: <https://nodemailer.com/> (дата обращения: 09.05.2026). – Текст: электронный.
17. Хрущёв, Д. Г. Разработка ролевой модели доступа к объектам и сервисам веб-приложения на примере системы коллективного наполнения контентом / Д. Г. Хрущёв // Молодой

учёный. – 2016. – URL: <https://cyberleninka.ru/article/n/razrabotka-rolevoy-modeli-dostupa-k-obektam-i-servisam-veb-prilozheniya-na-primere-sistemy-kollektivnogo-napolneniya-kontentom> (дата обращения: 13.05.2026). – Текст: электронный.

18. Ляшов, Е. И. Виды аутентификации в современных веб-приложениях / Е. И. Ляшов // *Universum: технические науки*. – 2024. – № 12(129). – URL: <https://cyberleninka.ru/article/n/vidy-autentifikatsii-v-sovremennyh-veb-prilozheniyah> (дата обращения: 13.05.2026). – Текст: электронный.

19. Введение в REST API — RESTful веб-сервисы // *Habr.com*. – URL: <https://habr.com/ru/articles/483202/> (дата обращения: 09.05.2026). – Текст: электронный.

20. Grafana K6 documentation. – URL: <https://grafana.com/docs/k6/latest/> (дата обращения: 13.05.2026). – Текст: электронный.

© *Е. А. Калинина, А. А. Шаранов, 2026*