

В. В. Пеньков^{1✉}, Е. Ю. Воронкин¹

Разработка веб-модуля для настройки ролей и разграничения доступа пользователей в рамках внутренней административной системы организации с применением технологий React и Node.js

¹Сибирский государственный университет геосистем и технологий,
г. Новосибирск, Российская Федерация
e-mail: penkov.voldemor@yandex.ru

Аннотация. В статье представлены результаты разработки веб-модуля для настройки ролей и разграничения доступа пользователей во внутренней административной системе организации. Рассмотрена проблема централизованного управления учетными записями, группами ролей и правами доступа. Для решения данной задачи выбрана ролевая модель разграничения доступа RBAC, позволяющая назначать разрешения пользователям не напрямую, а через группы ролей. Описаны основные требования к веб-модулю, архитектура клиентской и серверной частей, структура базы данных и механизм проверки прав доступа. Клиентская часть приложения реализована с использованием React и TypeScript, серверная часть – на Node.js и Express.js, в качестве системы управления базами данных использована PostgreSQL. Авторизация пользователей выполняется с применением JWT-токенов, а пароли хранятся в хешированном виде. В результате разработки создан веб-модуль, обеспечивающий управление пользователями, группами ролей, разрешениями, авторизацией и проверкой доступа к защищенным API-методам.

Ключевые слова: веб-модуль, разграничение доступа, RBAC, React, Node.js, PostgreSQL

V.V. Penkov^{1✉}, E. Y. Voronkin¹

Development of a web Module for configuring roles and user access control in an organization's internal administrative system using React and Node.js

¹Siberian State University of Geosystems and Technologies, Novosibirsk, Russian Federation
e-mail: penkov.voldemor@yandex.ru

Abstract. The article presents the results of developing a web module for configuring roles and controlling user access in an internal administrative system of an organization. The problem of centralized management of user accounts, role groups and access rights is considered. The role-based access control model, RBAC, was selected to solve this problem, as it allows permissions to be assigned to users through role groups rather than directly. The article describes the main requirements for the web module, the architecture of the client and server parts, the database structure and the access rights verification mechanism. The client part of the application was implemented using React and TypeScript, while the server part was developed using Node.js and Express.js. PostgreSQL was used as the database management system. User authorization is performed using JWT tokens, and passwords are stored in hashed form. As a result, a web module was developed that provides user management, role group management, permission configuration, authorization and access control for protected API methods.

Keywords: web module, access control, RBAC, React, Node.js, PostgreSQL

Введение

Современные корпоративные информационные системы включают большое количество пользователей, выполняющих разные функции в рамках рабочих процессов организации. Одним сотрудникам требуется только просмотр данных, другим – возможность создания и редактирования записей, а администраторам – управление учетными записями, группами ролей и настройками системы. В таких условиях важной задачей становится разграничение доступа к функциям приложения и данным.

При отсутствии централизованного механизма управления правами администрирование пользователей усложняется. Права могут назначаться вручную для каждой учетной записи, что повышает риск ошибок, дублирования настроек и случайной выдачи лишних полномочий. Кроме того, недостаточный контроль доступа может привести к несанкционированному изменению данных или выполнению действий, которые не должны быть доступны конкретному пользователю.

Одним из распространенных подходов к решению данной задачи является ролевая модель разграничения доступа RBAC. В данной модели разрешения назначаются не каждому пользователю отдельно, а роли или группе ролей. Пользователь включается в одну или несколько групп и получает набор прав, связанных с этими группами [1, 2]. Такой подход позволяет централизованно управлять доступом и изменять полномочия сразу для нескольких пользователей.

Целью работы является разработка веб-модуля для настройки ролей и разграничения доступа пользователей в рамках внутренней административной системы организации. Для достижения данной цели необходимо определить требования к модулю, спроектировать архитектуру приложения и базу данных, реализовать клиентскую и серверную части, а также проверить корректность основных сценариев работы.

Методы и материалы

Для разработки веб-модуля была выбрана клиент-серверная архитектура. Клиентская часть отвечает за отображение пользовательского интерфейса, работу с таблицами, формами, диалоговыми окнами и панелями детальной информации. Серверная часть принимает запросы от клиента, выполняет авторизацию, проверяет права доступа, обрабатывает данные и взаимодействует с базой данных.

В качестве основы для разграничения доступа использована модель RBAC. Ее применение позволяет разделить пользователей, группы ролей и разрешения. Пользователь получает доступ к функциям системы через назначенные ему группы ролей, а каждая группа содержит определенный набор разрешений. Такой подход удобен для административных систем, поскольку позволяет управлять правами централизованно и не настраивать доступ вручную для каждой учетной записи [3, 4].

Клиентская часть веб-модуля разработана с использованием React и TypeScript. React позволяет строить интерфейс из отдельных компонентов, что удобно для стра-

ниц пользователей и групп ролей, где повторяются таблицы, формы, кнопки, элементы поиска, пагинация и панели деталей [1, 5]. TypeScript применяется для статической типизации данных, что снижает вероятность ошибок при работе с API и формами [5, 6].

Для сборки клиентского приложения используется Vite. Данный инструмент обеспечивает быстрый запуск проекта в режиме разработки и оптимизированную сборку приложения [7]. Для оформления интерфейса применяются Tailwind CSS и shadcn/ui. Tailwind CSS позволяет описывать оформление через утилитарные классы, а shadcn/ui предоставляет набор локальных компонентов, которые можно адаптировать под требования административного интерфейса [8, 9]. Для управления формами и валидацией используется React Hook Form [10].

Серверная часть разработана на Node.js с использованием Express.js. Node.js позволяет выполнять JavaScript-код на сервере, а Express.js предоставляет средства для описания маршрутов, подключения middleware и обработки HTTP-запросов [11–14]. Взаимодействие между клиентской и серверной частями организовано через REST API.

Для авторизации пользователей используется JSON Web Token. После успешного входа сервер формирует JWT-токен, который клиентская часть использует при последующих запросах к защищенным API-методам [15]. Пароли пользователей не хранятся в открытом виде. Для их защиты применяется хеширование с использованием bcrypt [16]. Такой подход повышает безопасность учетных данных и снижает риск их раскрытия при несанкционированном доступе к базе данных.

В качестве системы управления базами данных выбрана PostgreSQL. Данная СУБД подходит для хранения структурированных данных и реализации связей между пользователями, группами ролей и разрешениями [17]. Для взаимодействия серверной части с PostgreSQL используется node-postgres, позволяющий выполнять SQL-запросы из Node.js-приложения [18].

Структура базы данных включает пять основных таблиц: users, role_groups, permissions, user_role_groups и role_group_permissions. Таблица users хранит учетные записи пользователей. Таблица role_groups содержит группы ролей. Таблица permissions хранит системные разрешения. Таблицы user_role_groups и role_group_permissions используются для реализации связей «многие ко многим» между пользователями и группами ролей, а также между группами ролей и разрешениями.

Проверка доступа выполняется на серверной стороне. При обращении к защищенному API-методу сервер проверяет JWT-токен, определяет текущего пользователя, получает его группы ролей и итоговый набор разрешений. Если необходимое разрешение присутствует, операция выполняется. Если разрешение отсутствует, сервер возвращает ошибку доступа. Такой подход соответствует рекомендациям по защите API и серверной проверке авторизации [19, 20].

Результаты

В результате разработки был создан веб-модуль, включающий клиентскую и серверную части, базу данных и механизм разграничения доступа. Модуль предназначен для управления пользователями, группами ролей и правами доступа во внутренней административной системе.

На клиентской стороне реализованы основные страницы приложения: страница авторизации, страница пользователей и страница групп ролей. Страница авторизации используется для входа в систему. После успешной авторизации пользователь получает доступ к административному интерфейсу. Если токен отсутствует или недействителен, приложение перенаправляет пользователя на страницу входа.

Страница пользователей содержит таблицу учетных записей, поиск, сортировку, постраничную навигацию и действия над строками. Через интерфейс можно создать нового пользователя, отредактировать данные существующего пользователя, изменить статус активности, назначить группы ролей и удалить учетную запись. Для просмотра подробной информации используется боковая панель, позволяющая не покидать страницу со списком.

Страница групп ролей реализована по аналогичному принципу. Она позволяет просматривать список групп, создавать новые группы, редактировать их название, системный код, описание и набор разрешений. Через назначение разрешений группам формируется итоговый набор прав пользователей, входящих в эти группы.

В формах создания и редактирования реализована валидация данных. Проверяются обязательные поля, формат электронной почты, пароль при создании пользователя, а также корректность выбранных групп и разрешений. Ошибки отображаются рядом с соответствующими полями формы, что упрощает исправление данных пользователем.

Для работы с большим количеством записей реализована постраничная пагинация. Клиентская часть загружает только записи выбранной страницы, а общее количество записей получает от серверной части. Это снижает нагрузку на приложение и делает работу таблиц более стабильной при увеличении количества пользователей и групп ролей.

Отдельное внимание уделено обработке ошибок и пользовательских состояний. При загрузке данных отображаются индикаторы или скелетон-элементы. Если список пуст, пользователь видит поясняющее сообщение. При ошибке загрузки отображается текст ошибки и возможность повторить запрос. Для критичных действий, таких как удаление пользователя или группы ролей, используется окно подтверждения.

На серверной стороне реализованы API-методы авторизации, получения данных текущего пользователя, управления пользователями, управления группами ролей и получения списка разрешений. Серверная часть выполняет проверку токена, хеширование паролей, валидацию входных данных, обработку ошибок и проверку прав доступа к защищенным операциям.

Для первичной подготовки базы данных используется скрипт инициализации. Он добавляет базовые разрешения, группу администраторов и тестового пользователя с полным набором прав. Это позволяет быстро проверить основные сценарии: вход в систему, управление пользователями, настройку групп ролей и контроль доступа.

Тестирование веб-модуля проводилось на уровне API и пользовательского интерфейса. Для проверки серверной части использовался Postman [21]. Проверялись успешная авторизация, ошибки входа, запросы без токена, запросы с некорректным токеном, выполнение операций без необходимых прав, создание и редактирование пользователей, назначение групп ролей, создание групп и назначение разрешений.

Основные сценарии тестирования включали проверку авторизации, защиты закрытых маршрутов, операций создания, редактирования и удаления пользователей, назначения групп ролей, управления разрешениями, поиска, пагинации, отображения ошибок и переключения светлой и темной темы. По результатам тестирования критических ошибок, препятствующих использованию модуля, выявлено не было.

88

Diplom Admin

Панель управления

🔍 Пользователи

👤 Группы

🌙 Тема

admin

Пользователи

Учётные записи каталога: поиск, просмотр доступа и управление активностью учётных записей.

Создать пользователя

Всего пользователей

54

Учётные записи в каталоге

Активные

54

Могут входить в систему

Администраторы

2

Участники группы администраторов

Поиск

Статус

Сортировка

Логин, email или имя...

Все статусы

Пользователь

Обновить

Пользователь	Email	Группы	Статус	Обновлен	Действия
admin System Admin	admin@example.com	Administrators	Активен	25.04.2026, 13:46:47	...
manager Team Manager	manager@example.com	Managers	Активен	22.04.2026, 17:51:35	...
user_1 Пользователь 1	user_1@test.ru	Administrators	Активен	24.04.2026, 19:47:42	...
user_10 Пользователь 10	user_10@test.ru	Managers Доп Группа 20 Группа 21 Группа 22 Группа 23	Активен	25.04.2026, 14:21:03	...
user_11 Пользователь 11	user_11@test.ru	—	Активен	24.04.2026, 19:23:41	...
user_12 Пользователь 12	user_12@test.ru	—	Активен	24.04.2026, 19:23:41	...
user_13 Пользователь 13	user_13@test.ru	—	Активен	24.04.2026, 19:23:41	...
user_14 Пользователь 14	user_14@test.ru	—	Активен	24.04.2026, 19:23:41	...
user_15 Пользователь 15	user_15@test.ru	—	Активен	24.04.2026, 19:23:41	...
user_16 Пользователь 16	user_16@test.ru	—	Активен	24.04.2026, 19:23:41	...

Рис. 1. Интерфейс таблицы

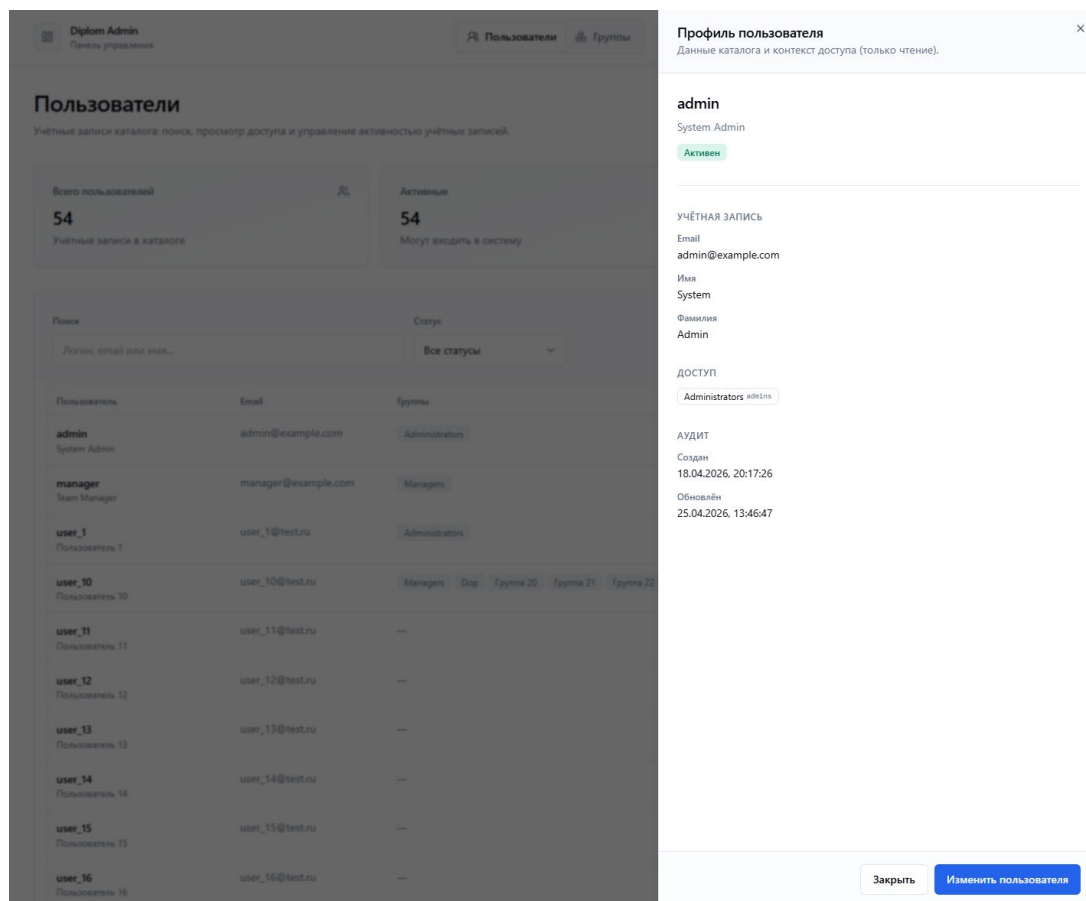


Рис. 2. Интерфейс модального окна информации

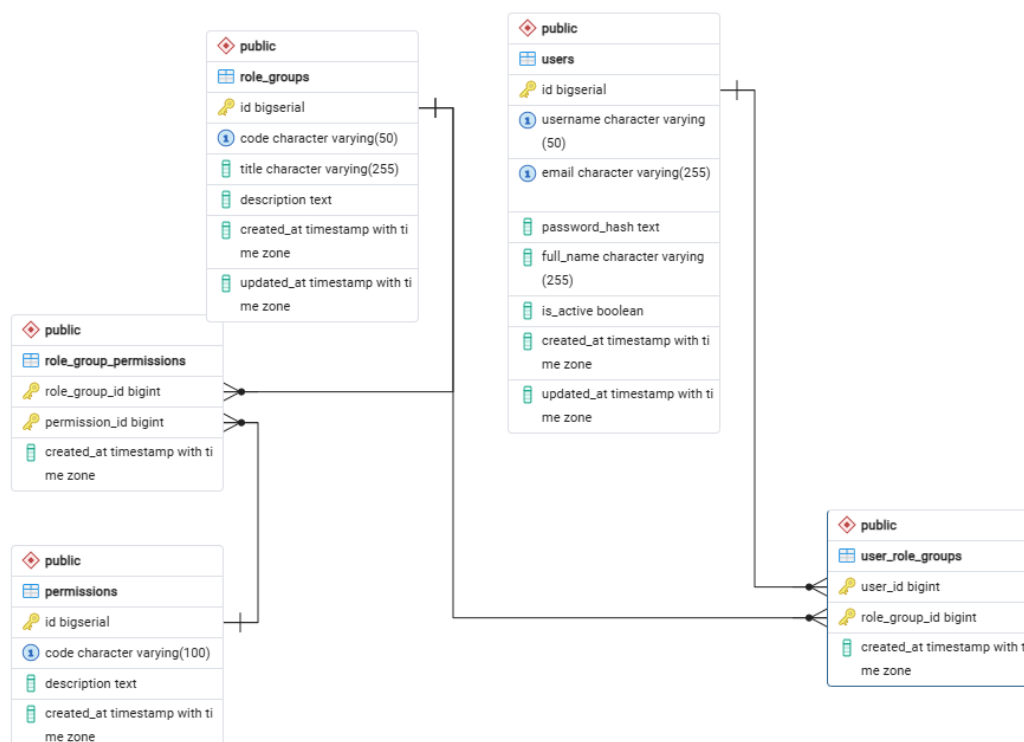


Рис. 3. Диаграмма базы данных

Обсуждение

Разработанный веб-модуль решает задачу централизованного управления доступом во внутренней административной системе. Использование RBAC позволяет избежать прямого назначения прав каждому пользователю и упростить сопровождение системы. Администратор может изменить состав разрешений у группы ролей, после чего изменения будут применены ко всем пользователям, входящим в эту группу.

По сравнению с прямым назначением прав отдельным учетным записям, ролевая модель делает структуру доступа более понятной. Пользователи объединяются в группы в соответствии с выполняемыми обязанностями, а права задаются на уровне этих групп. Такой подход снижает вероятность ошибок и упрощает добавление новых пользователей.

В то же время ролевая модель имеет ограничения. Если в системе требуется учитывать большое количество динамических условий, например подразделение пользователя, время запроса, состояние ресурса или дополнительные атрибуты объекта, может потребоваться более гибкая модель доступа. В таких случаях может применяться ABAC, однако ее реализация сложнее и требует большего количества правил проверки [19].

В разработанном модуле проверка прав выполняется как на клиентской, так и на серверной стороне. Клиентская часть использует информацию о правах для отображения или блокировки элементов интерфейса. Это улучшает пользовательский опыт, так как пользователь видит только доступные действия. Однако основная защита реализована на серверной стороне, поскольку клиентский интерфейс не может считаться надежным механизмом безопасности. Даже если пользователь попытается отправить запрос напрямую к API, сервер дополнительно проверит наличие необходимого разрешения.

Отдельным преимуществом разработанного решения является разделение клиентской и серверной частей. Такое архитектурное решение позволяет независимо развивать интерфейс и backend-логику. Например, изменение внешнего вида таблиц или форм не требует изменения базы данных, а изменение правил проверки прав доступа может быть выполнено на серверной стороне без полной переработки интерфейса.

Практическое применение разработанного модуля возможно во внутренних административных системах организаций. Он может использоваться как отдельный модуль управления доступом или как часть более крупной корпоративной информационной системы. В дальнейшем решение может быть расширено за счет добавления журнала действий пользователей, аудита изменений, восстановления пароля, интеграции с внешними сервисами авторизации и более детального анализа назначенных прав.

Заключение

В ходе работы был разработан веб-модуль для настройки ролей и разграничения доступа пользователей во внутренней административной системе органи-

зации. Модуль обеспечивает управление учетными записями, группами ролей и разрешениями, а также выполняет авторизацию и проверку прав доступа к защищенным API-методам.

Для клиентской части использованы React, TypeScript, Vite, Tailwind CSS, shadcn/ui и React Hook Form. Серверная часть реализована на Node.js и Express.js. Для хранения данных применена PostgreSQL. Авторизация пользователей выполнена с использованием JWT-токенов, а пароли защищены с помощью bcrypt.

В результате разработки создана структура базы данных, соответствующая ролевой модели разграничения доступа. Пользователи могут состоять в нескольких группах ролей, а группы могут содержать несколько разрешений. Такая структура позволяет гибко управлять полномочиями пользователей и расширять систему без существенной переработки базы данных.

Проведенное тестирование подтвердило корректность основных сценариев работы веб-модуля: авторизации, защиты маршрутов, управления пользователями, назначения групп ролей, управления группами, назначения разрешений, поиска, пагинации и обработки ошибок. Разработанное решение может быть использовано во внутренних административных системах для повышения удобства администрирования и безопасности управления доступом.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Sandhu, R. The NIST Model for Role-Based Access Control: Towards a Unified Standard / R. Sandhu, D. Ferraiolo, R. Kuhn // National Institute of Standards and Technology : сайт. – URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=916402 (дата обращения: 03.05.2026). – Текст : электронный.
2. Ferraiolo, D. F. Role-Based Access Controls / D. F. Ferraiolo, D. R. Kuhn // National Institute of Standards and Technology : сайт. – URL: <https://csrc.nist.gov/files/pubs/conference/1992/10/13/rolebased-access-controls/final/docs/ferraiolo-kuhn-92.pdf> (дата обращения: 03.05.2026). – Текст : электронный.
3. React. Документация : официальный сайт. – URL: <https://react.dev> (дата обращения: 03.05.2026). – Текст : электронный.
4. React Reference Overview // React : официальный сайт. – URL: <https://react.dev/reference/react> (дата обращения: 03.05.2026). – Текст : электронный.
5. TypeScript. Документация : официальный сайт. – URL: <https://www.typescriptlang.org/docs/> (дата обращения: 03.05.2026). – Текст : электронный.
6. TypeScript Handbook // TypeScript : официальный сайт. – URL: <https://www.typescriptlang.org/docs/handbook/intro.html> (дата обращения: 03.05.2026). – Текст : электронный.
7. Vite. Документация : официальный сайт. – URL: <https://vite.dev/guide/> (дата обращения: 03.05.2026). – Текст : электронный.
8. Tailwind CSS. Styling with utility classes // Tailwind CSS : официальный сайт. – URL: <https://tailwindcss.com/docs/styling-with-utility-classes> (дата обращения: 03.05.2026). – Текст : электронный.
9. shadcn/ui. Introduction // shadcn/ui : официальный сайт. – URL: <https://ui.shadcn.com/docs> (дата обращения: 03.05.2026). – Текст : электронный.
10. React Hook Form. Get Started // React Hook Form : официальный сайт. – URL: <https://react-hook-form.com/get-started> (дата обращения: 03.05.2026). – Текст : электронный.
11. Node.js. Документация : официальный сайт. – URL: <https://nodejs.org> (дата обращения: 03.05.2026). – Текст : электронный.

12. Express. Документация : официальный сайт. – URL: <https://expressjs.com> (дата обращения: 03.05.2026). – Текст : электронный.
13. Express.js. Routing // Express : официальный сайт. – URL: <https://expressjs.com/en/guide/routing.html> (дата обращения: 03.05.2026). – Текст : электронный.
14. Express.js. Using middleware // Express : официальный сайт. – URL: <https://expressjs.com/en/guide/using-middleware.html> (дата обращения: 03.05.2026). – Текст : электронный.
15. JSON Web Token Introduction // JWT.io : сайт. – URL: <https://jwt.io/introduction> (дата обращения: 03.05.2026). – Текст : электронный.
16. bcrypt // npm : сайт. – URL: <https://www.npmjs.com/package/bcrypt> (дата обращения: 03.05.2026). – Текст : электронный.
17. PostgreSQL. Документация : официальный сайт. – URL: <https://www.postgresql.org/docs/> (дата обращения: 03.05.2026). – Текст : электронный.
18. node-postgres. Документация : сайт. – URL: <https://node-postgres.com/> (дата обращения: 03.05.2026). – Текст : электронный.
19. Authorization Cheat Sheet // OWASP Cheat Sheet Series : сайт. – URL: https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html (дата обращения: 03.05.2026). – Текст : электронный.
20. REST Security Cheat Sheet // OWASP Cheat Sheet Series : сайт. – URL: https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html (дата обращения: 03.05.2026). – Текст : электронный.
21. Postman documentation overview // Postman Docs : официальный сайт. – URL: <https://learning.postman.com/docs/introduction/overview/> (дата обращения: 03.05.2026). – Текст : электронный.
22. Прохоренок, Н. А. JavaScript и Node.js для веб-разработчиков / Н. А. Прохоренок, В. А. Дронов. – Санкт-Петербург : БХВ-Петербург, 2022. – 768 с.
23. Фленов, М. В. JavaScript. Подробное руководство / М. В. Фленов. – Санкт-Петербург : БХВ-Петербург, 2020. – 720 с.
24. Кирсанов, Д. И. CSS: каскадные таблицы стилей / Д. И. Кирсанов. – Санкт-Петербург : БХВ-Петербург, 2020. – 352 с.
25. Built-in React Hooks // React : официальный сайт. – URL: <https://react.dev/reference/react/hooks> (дата обращения: 03.05.2026). – Текст : электронный.
26. Introduction to Node.js // Node.js : официальный сайт. – URL: <https://nodejs.org/learn/getting-started/introduction-to-nodejs> (дата обращения: 03.05.2026). – Текст : электронный.
27. React Router. Документация : официальный сайт. – URL: <https://reactrouter.com/> (дата обращения: 03.05.2026). – Текст : электронный.
28. dotenv // npm : сайт. – URL: <https://www.npmjs.com/package/dotenv> (дата обращения: 03.05.2026). – Текст : электронный.
29. Cross-Origin Resource Sharing (CORS) // MDN Web Docs : сайт. – URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS> (дата обращения: 03.05.2026). – Текст : электронный.
30. cors // npm : сайт. – URL: <https://www.npmjs.com/package/cors> (дата обращения: 03.05.2026). – Текст : электронный.
31. Zod. Документация : официальный сайт. – URL: <https://zod.dev/> (дата обращения: 03.05.2026). – Текст : электронный.
32. WebStorm : официальный сайт. – URL: <https://www.jetbrains.com/webstorm/> (дата обращения: 03.05.2026). – Текст : электронный.
33. Authentication Cheat Sheet // OWASP Cheat Sheet Series : сайт. – URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата обращения: 03.05.2026). – Текст : электронный.

34. Session Management Cheat Sheet // OWASP Cheat Sheet Series : сайт. – URL: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html (дата обращения: 03.05.2026). – Текст : электронный.

35. Password Storage Cheat Sheet // OWASP Cheat Sheet Series : сайт. – URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (дата обращения: 03.05.2026). – Текст : электронный..

© В. В. Пеньков, Е. Ю. Воронкин, 2026