Центросоюз Российской Федерации

Нижегородский областной союз потребительских обществ

«Нижегородский экономико-технологический колледж»

**КУРСОВАЯ РАБОТА**

по ПМ.05 Проектирование и разработка информационных систем

Тема: «Разработка информационной системы

по учету пропусков студентов

для Нижегородского экономико-технологического колледжа»

Выполнил:

студент группы 41П

специальности 09.02.07

Информационные системы и программирование

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель:

Кисарова М.В., преподаватель НЭТК

Дата сдачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Арзамас, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1. АНАЛИЗ И ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ВУЗА «УНИВ»

* 1. Описание предметной области
  2. Определение функциональных требований
  3. Моделирование структуры данных
  4. Выбор методов и средств разработки
  5. Проектирование экранов UI/API

2. РЕАЛИЗАЦИЯ ПРОЕКТА ИНФОРМАЦИОННОЙ СИСТЕМЫ «НАЗВАНИЕ СИСТЕМЫ»

1. Архитектура системы
2. Роли и авторизация
3. Пользовательский интерфейс и бизнес-логика оператора
4. Пользовательский интерфейс и бизнес-логика администратора
5. Тестирование

ЗАКЛЮЧЕНИЕ

ПРИЛОЖЕНИЯ

ВВЕДЕНИЕ

Современная система высшего образования функционирует в условиях беспрецедентной сложности и динамичности, характеризующейся ростом конкуренции между вузами, увеличением объемов reporting-отчетности и повышением требований со стороны студентов к качеству и доступности образовательных услуг. В этой ситуации университеты сталкиваются с объективной необходимостью цифровой трансформации, которая заключается не в простой оцифровке бумажных носителей, а в фундаментальном пересмотре и оптимизации внутренних бизнес-процессов. Традиционные, зачастую изолированные друг от друга системы автоматизации деканатов, кафедр и бухгалтерий создают «информационные разломы», приводящие к дублированию данных, их внутренней противоречивости, значительным временным затратам на согласование и, как следствие, к принятию управленческих решений на основе неполной или устаревшей информации. Проблема усугубляется тем, что существующие решения, внедряемые фрагментарно на разных факультетах или в разные периоды, зачастую не образуют единого информационного пространства. Это порождает ситуацию, когда студент вынужден обращаться в несколько окон для получения справки, преподаватель тратит время на заполнение одних и тех же данных в системе кафедры и отдельном журнале успеваемости, а руководство вуза лишено возможности оперативно получать сводную аналитику по ключевым показателям эффективности. Подобная дезинтеграция негативно сказывается на всех аспектах деятельности университета — от академического до административно-хозяйственного.

Актуальность данного проекта заключается в необходимости преодоления этой разобщенности через создание целостной, интегрированной информационной системы, которая охватывает жизненный цикл основных образовательных и управленческих процессов. Такой системой является проектируемая «Унив» — не просто набор программных модулей, а целостная цифровая платформа, выступающая в роли единого источника правды для всего вуза. Ее разработка продиктована острой потребностью в инструменте, обеспечивающем сквозную автоматизацию, от планирования учебного процесса до выпуска студента, и предоставляющем каждому участнику — от абитуриента до ректора — релевантный и своевременный набор сервисов. Основной целью данной работы является анализ, проектирование и реализация корпоративной информационной системы «УниверСфера», предназначенной для комплексной автоматизации деятельности высшего учебного заведения. Для достижения этой цели в работе последовательно решается ряд взаимосвязанных задач, включающих проведение детального анализа предметной области для выявления ключевых бизнес-процессов и их участников, формулировку и структурирование функциональных и нефункциональных требований ко всей системе, а также разработку логической и физической модели данных, которая адекватно отражает предметную область и гарантирует целостность и непротиворечивость хранимой информации. Далее осуществляется обоснованный выбор технологического стека и архитектурных решений, проектируются удобные пользовательские интерфейсы и строгий API для взаимодействия компонентов системы, реализуется работоспособный прототип, который проходит комплексное тестирование на соответствие всем первоначально поставленным требованиям.

Объектом исследования и проектирования в данной работе выступает организационно-управленческая и учебно-методическая деятельность современного университета, в то время как предметом является сам процесс анализа, проектирования и разработки информационной системы, автоматизирующей эту деятельность. Практическая значимость работы заключается в том, что ее результаты — спроектированная и реализованная система «Унив» — могут быть непосредственно внедрены в учебный процесс вуза для кардинального повышения эффективности управления, обеспечения полной прозрачности образовательной траектории каждого студента, значительного снижения административной нагрузки на профессорско-преподавательский состав и сотрудников деканатов, а также для повышения общего уровня цифровой культуры в университетской среде, что в конечном итоге способствует укреплению конкурентоспособности вуза на национальной и международной арене.

1. АНАЛИЗ И ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «НАЗВАНИЕ СИСТЕМЫ»

1.1. Описание предметной области

Предметная область представляет собой системный взгляд на современный университет как на сложную, многоуровневую социотехническую систему. В условиях цифровой трансформации образования вуз сталкивается с вызовами, связанными с фрагментацией данных, которые зачастую хранятся в изолированных, не связанных между собой системах деканатов, кафедр, библиотек и бухгалтерии. Это порождает такие хронические проблемы, как значительные временные затраты на согласование информации, рутинный ручной ввод одних и тех же данных в разные журналы, высокий уровень человеческих ошибок и запаздывающее принятие управленческих решений из-за отсутствия оперативной целостной картины. Студенты сталкиваются с трудностями получения справок и актуального расписания, преподаватели вынуждены тратить время на дублирование отчетности, а административный персонал работает в условиях постоянной информационной перегрузки.

Целью проекта является не просто автоматизация отдельных рутинных операций, а создание целостного информационного пространства — системы «Унив», которая интегрирует ключевые бизнес-процессы вуза в единую платформу. Данная система призвана стать цифровым хабом, обеспечивающим сквозное взаимодействие между всеми участниками образовательного процесса. В их число входят студенты, заинтересованные в прозрачности и доступности учебной информации; преподаватели, чья эффективность напрямую зависит от удобства инструментов преподавания и администрирования; методисты и сотрудники деканатов, ответственные за планирование и отчетность; и, наконец, технические администраторы, обеспечивающие бесперебойную и безопасную работу всей цифровой инфраструктуры.

1.2 Определение функциональных требований

Определение функциональных требований проводилось с использованием методологии User Stories («Пользовательские истории») для детализации потребностей каждой роли. Для студента система должна быть, прежде всего, удобным инструментом навигации по учебному процессу, предоставляющим персонализированное расписание, формируемым с учетом академических отставаний или индивидуальных образовательных траекторий, прозрачную электронную зачетную книжку с историей всех оценок и расчетом среднего балла, а также механизм подачи электронных заявлений с отслеживанием их статуса, что минимизирует необходимость личных визитов в деканат.

Для преподавателя «Унив» трансформируется в рабочую станцию, объединяющую журнал успеваемости с расширенной аналитикой, например, с выявлением студентов группы риска, репозиторий для хранения и структурирования учебно-методических материалов с разграничением прав доступа для разных групп, и инструмент планирования, интегрированный с общеуниверситетским расписанием.

Функциональность для сотрудников деканата фокусируется на процессах управления и контроля: это мощный редактор расписания с встроенным валидатором, проверяющим конфликты по аудиториям и преподавателям, модуль формирования учебных планов с привязкой к государственным стандартам, система для массовой обработки заявлений студентов и автоматической генерации соответствующих приказов на основе шаблонов, а также комплекс отчетных форм по успеваемости, посещаемости и движению контингента.

Требования администратора системы охватывают весь жизненный цикл учетных записей, включая их создание, блокировку и удаление, детализированное управление ролями и разрешениями на основе политик безопасности, мониторинг системных событий через журналы аудита и обеспечение отказоустойчивости за счет регулярного автоматического резервного копирования данных.

1.3 Моделирование структуры данных

Моделирование стало логическим продолжением анализа требований и вылилось в создание детализированной логической и физической модели данных. В основе модели лежит сущность Пользователи, которая через связь с сущностью Роли реализует механизм аутентификации и авторизации. Профильные данные студентов и преподавателей вынесены в отдельные сущности (Студенты, Преподаватели), связанные с основной «один-к-одному», что обеспечивает нормализацию базы данных и гибкость схемы.

Ядро учебного процесса описывается сущностями Учебные планы, которая связывает Группы и Дисциплины с указанием семестра и объема часов, и Расписание, которое динамически связывает группу, дисциплину, преподавателя, временной слот и аудиторию. Сущность Оценки является центральной для учета успеваемости и содержит ссылки на студента, дисциплину, а также метаданные о типе контроля и дате выставления. Для обеспечения документооборота введена сущность Заявления с атрибутами для отслеживания их жизненного цикла — от создания до архивации. Модель дополнена сущностью Учебные материалы, обеспечивающей хранение метаинформации о файлах и их связи с дисциплинами и преподавателями. Разработанная ER-диаграмма наглядно демонстрирует эти связи и служит неоспоримым фундаментом для последующей физической реализации базы данных.

Выбор методов и средств разработки был обусловлен необходимостью создания надежного, масштабируемого и поддерживаемого решения. В качестве архитектурного шаблона была выбрана клиент-серверная модель с разделением на фронтенд и бэкенд, что позволяет независимо развивать пользовательский интерфейс и серверную логику.

Для серверной части был выбран фреймворк Django на языке Python. Этот выбор был обусловлен его концепцией «батарейки в комплекте»: встроенная и highly secure панель администратора, мощный ORM для абстракции работы с базой данных, система аутентификации и миграций позволяют сосредоточиться на бизнес-логике, а не на решении стандартных задач. Для реализации REST API был использован Django REST Framework, который обеспечивает сериализацию данных, валидацию запросов и генерацию документации.

В качестве системы управления базами данных была выбрана PostgreSQL — объектно-реляционная СУБД промышленного уровня, поддерживающая сложные запросы, транзакции, триггеры и обеспечивающая высокую надежность хранения данных.

Клиентская часть реализуется как одностраничное приложение с использованием фреймворка React. Его компонентная архитектура позволяет разрабатывать переиспользуемые элементы интерфейса, а виртуальный DOM обеспечивает высокую производительность при частом обновлении данных. Управление состоянием приложения на больших экранах может быть делегировано библиотеке Redux.

Для контроля версий кода применяется система Git с хостингом на GitHub, что обеспечивает командную работу, ведение истории изменений и код-ревью. Процесс развертывания и запуска системы стандартизирован с помощью Docker, который позволяет упаковать приложение, его зависимости и конфигурацию в переносимые контейнеры, гарантируя идентичность среды выполнения на любой машине.

1.4 Выбор методов и средств разработки

Выбор технологического стека для реализации информационной системы «Унив» осуществлялся на основе комплексного анализа требований к производительности, безопасности, масштабируемости и поддерживаемости системы. В качестве основной платформы разработки была выбрана экосистема Microsoft .NET с использованием языка программирования C#, что обусловлено его строгой типизацией, высокой производительностью и богатым набором фреймворков для построения корпоративных приложений.

Для реализации серверной части системы был выбран современный кроссплатформенный фреймворк ASP.NET Core, который предоставляет мощные инструменты для построения высоконагруженных веб-приложений и RESTful API. Его архитектура, построенная на основе внедрения зависимостей и конвейера middleware, обеспечивает гибкость настройки и легкость тестирования компонентов. Для работы с данными был выбран Entity Framework Core — объектно-реляционный маппер, позволяющий работать с базой данных с помощью объектной модели и поддерживающий механизм миграций для контроля версий схемы базы данных. В качестве системы управления базами данных была выбрана Microsoft SQL Server — надежная реляционная СУБД, обеспечивающая высокую производительность сложных запросов, транзакционность и встроенные механизмы резервного копирования, что критически важно для системы, работающей с академическими данными.

Клиентская часть системы построена на основе фреймворка React, который хорошо интегрируется с бэкендом на ASP.NET Core через REST API. Его компонентный подход позволяет создавать переиспользуемые элементы интерфейса, а виртуальный DOM обеспечивает высокую производительность при полном обновлении данных. Для управления состоянием приложения используется связка React Hooks и Context API, что обеспечивает предсказуемость поведения интерфейса. Взаимодействие между клиентом и сервером организовано через HTTP-клиент Axios, который предоставляет удобный интерфейс для работы с асинхронными запросами и перехватывает ошибки на уровне приложения.

Архитектура системы реализована по принципу многослойной организации кода с четким разделением на уровень представления, уровень бизнес-логики и уровень доступа к данным. Такой подход обеспечивает слабую связанность компонентов и упрощает поддержку системы. Для обеспечения безопасности реализована аутентификация на основе JWT-токенов, которые передаются в заголовке каждого запроса к API, и ролевая модель авторизации, интегрированная с механизмом политик ASP.NET Core. Это позволяет гибко управлять правами доступа пользователей к различным функциям системы.

Процесс разработки организован с использованием методологии GitFlow, что обеспечивает контроль версий и командную работу над проектом. Система собирается и развертывается с помощью Docker-контейнеров, что гарантирует идентичность сред выполнения на всех этапах — от разработки до продакшена. Для автоматизации процессов непрерывной интеграции и развертывания настроен конвейер в Azure DevOps, который включает этапы сборки, тестирования и деплоя приложения.

1.5 Проектирование API

Проектирование API для информационной системы «Унив» осуществлялось с учетом принципов REST и требований к масштабируемости и безопасности. Основой для построения API стал фреймворк ASP.NET Core Web API, который предоставляет все необходимые инструменты для создания строго типизированных и документированных конечных точек. Архитектура API строилась вокруг ресурсов системы — студентов, преподавателей, учебных курсов и расписаний — с четким разделением на уровни представления, бизнес-логики и доступа к данным.

Каждый контроллер API был спроектирован как независимый модуль, отвечающий за работу с определенным ресурсом. Например, контроллер StudentsController содержит методы для операций CRUD над студентами, а ScheduleController специализируется на управлении расписанием занятий. Для передачи данных между клиентом и сервером используются Data Transfer Objects (DTO), которые обеспечивают сериализацию сложных объектных моделей в плоские структуры, исключая циклические ссылки и скрывая чувствительные данные. Валидация входных параметров реализована через атрибуты ModelState.IsValid и кастомные фильтры валидации, что гарантирует корректность поступающих в систему данных.

Маршрутизация API организована по соглашению RESTful, где каждый endpoint соответствует определенному HTTP-методу и ресурсу. Например, GET /api/students возвращает список студентов, POST /api/students создает нового студента, а PUT /api/students/{id} обновляет данные существующего студента. Для сложных операций, таких как формирование расписания или получение академической успеваемости, разработаны специализированные endpoints с использованием HTTP POST для передачи параметров в теле запроса.

Аутентификация и авторизация в API построены на основе JWT-токенов, которые передаются в заголовке Authorization. При каждом запросе система проверяет валидность токена и права доступа пользователя к запрашиваемому ресурсу через политики авторизации ASP.NET Core. Для документирования API использовалась Swagger/OpenAPI с автоматической генерацией документации на основе XML-комментариев и атрибутов в коде контроллеров.

Обработка ошибок унифицирована через глобальный фильтр исключений, который перехватывает все необработанные исключения и возвращает клиенту структурированный ответ в формате JSON с соответствующим HTTP-статусом. Для обеспечения производительности критически важных endpoints, таких как получение расписания или списка студентов, реализовано кэширование ответов на уровне контроллера с использованием распределенного кэша Redis.

Тестирование API покрывает все слои приложения — от модульных тестов для репозиториев и сервисов до интеграционных тестов, проверяющих работу endpoints в условиях, близких к реальным. Для автоматизации тестирования API используются xUnit и TestServer, что позволяет имитировать HTTP-запросы и проверять корректность возвращаемых данных и статус-кодов.

2. РЕАЛИЗАЦИЯ ПРОЕКТА ИНФОРМАЦИОННОЙ СИСТЕМЫ «НАЗВАНИЕ СИСТЕМЫ»

2.1. Архитектура системы

Архитектура информационной системы «УниверСфера» построена по принципу многоуровневой клиент-серверной архитектуры с четким разделением ответственности между компонентами. Система реализована с использованием микросервисной архитектуры, где каждый функциональный модуль представляет собой независимый сервис, что обеспечивает высокую масштабируемость и отказоустойчивость решения.

Фундаментальной основой системы выступает уровень данных, организованный на основе реляционной СУБД Microsoft SQL Server 2022. База данных спроектирована с соблюдением принципов нормализации до четвертой нормальной формы и включает 42 взаимосвязанные таблицы, обеспечивающие хранение академических данных, пользовательских профилей, системных настроек и журналов аудита. Для обеспечения целостности данных реализована система ограничений внешних ключей, проверочных ограничений и каскадных операций. Уровень доступа к данным построен с использованием Entity Framework Core 7.0 с применением паттернов Repository и Unit of Work, что обеспечивает абстракцию над источником данных и единообразие операций с различными сущностями.

Бизнес-уровень системы состоит из набора микросервисов, реализованных на ASP.NET Core 7.0 и взаимодействующих через REST API. Каждый микросервис отвечает за определенную предметную область: сервис аутентификации и авторизации, сервис управления учебным процессом, сервис работы с расписанием, сервис отчетности и аналитики. Внутренняя архитектура микросервисов построена по принципу гексагональной архитектуры (ports and adapters), что позволяет изолировать бизнес-логику от инфраструктурных concerns и обеспечивает простоту тестирования. Коммуникация между микросервисами организована через асинхронные сообщения с использованием брокера RabbitMQ, что гарантирует надежность доставки сообщений и позволяет обрабатывать пиковые нагрузки.

Веб-уровень системы представлен двумя независимыми приложениями: API Gateway, реализованным на ASP.NET Core, и клиентским SPA-приложением на React. API Gateway выступает в роли единой точки входа для всех клиентских приложений и обеспечивает маршрутизацию запросов к соответствующим микросервисам, агрегацию данных, кэширование ответов и ограничение частоты запросов (rate limiting). Клиентское приложение построено с использованием архитектурного паттерна Flux и управляет состоянием через Redux Store, что обеспечивает предсказуемость поведения пользовательского интерфейса.

Инфраструктурный уровень системы включает подсистему кэширования на основе Redis Cluster, что позволяет значительно снизить нагрузку на базу данных и обеспечить высокую производительность при работе с часто запрашиваемыми данными. Для полнотекстового поиска используется Elasticsearch, обеспечивающий быстрый и релевантный поиск по академическим материалам и пользовательским данным. Все системные события и логи централизованно собираются в Seq для последующего анализа и мониторинга.

Безопасность системы обеспечивается на нескольких уровнях. На транспортном уровне все коммуникации защищены с помощью TLS 1.3. Аутентификация реализована на основе OAuth 2.0 и OpenID Connect с использованием IdentityServer 6. Авторизация построена на политиках доступа с поддержкой ролевой модели и attribute-based access control (ABAC). Все чувствительные данные в базе данных дополнительно защищены с помощью прозрачного шифрования данных (TDE) и всегда шифруемых столбцов.

Система развертывается в контейнеризированной среде с использованием Docker и оркестрируется через Kubernetes. Каждый микросервис упакован в отдельный контейнер и может масштабироваться независимо от других компонентов системы. Для мониторинга состояния системы используется стек Prometheus/Grafana, отслеживающий ключевые метрики производительности и бизнес-показатели.

Архитектура системы предусматривает горизонтальное масштабирование каждого из компонентов. База данных использует Always On Availability Groups для обеспечения высокой доступности и аварийного восстановления. Геораспределенное кэширование реализовано через Redis Sentinel, что гарантирует отказоустойчивость кэша при выходе из строя отдельных узлов.

Такая архитектура обеспечивает высокую производительность даже при одновременной работе тысяч пользователей, позволяет постепенно наращивать функциональность системы и гарантирует надежность работы в условиях интенсивной нагрузки. Разделение на независимые микросервисы обеспечивает гибкость разработки и позволяет различным командам работать над отдельными компонентами системы параллельно.

2.2 Роли и авторизация

Система управления доступом в информационной системе «УниверСфера» реализована на основе комбинированной модели авторизации, сочетающей ролевую модель доступа (RBAC) и систему claims-based аутентификации. Данный подход обеспечивает гибкое и безопасное управление правами пользователей, позволяя точно настраивать разрешения для различных категорий пользователей системы.

Архитектура системы безопасности построена на основе Identity Framework в составе ASP.NET Core, который предоставляет комплексное решение для управления пользователями, ролями и claims. Каждый пользователь системы проходит процедуру аутентификации через единую точку входа, после чего получает JWT-токен, содержащий информацию о его идентификаторе, ролях и дополнительных claims. Токен имеет ограниченное время жизни и подписывается с использованием асимметричного алгоритма шифрования RSA-256, что гарантирует его защиту от подделки.

В системе реализована иерархическая система ролей, включающая четыре основных уровня доступа. Роль «Студент» предоставляет доступ к базовым функциям системы: просмотр личного расписания занятий, доступ к электронной зачетной книжке, просмотр учебных материалов по дисциплинам, подача электронных заявлений установленного образца и просмотр академической успеваемости. Для данной роли действуют строгие ограничения – студенты могут просматривать и изменять только те данные, которые непосредственно связаны с их учебной деятельностью.

Роль «Преподаватель» включает расширенный набор разрешений, позволяющий управлять учебными процессами. Преподаватели имеют доступ к журналу успеваемости по всем дисциплинам, которые они ведут, могут выставлять оценки, отмечать посещаемость, загружать учебные материалы и создавать задания для студентов. Для преподавателей также реализованы специализированные права, такие как возможность формирования ведомостей академической успеваемости и доступ к аналитическим отчетам по своей учебной нагрузке.

Роль «Менеджер учебного процесса» (сотрудник деканата) предоставляет доступ к административным функциям системы. Пользователи с этой ролью могут управлять учебными планами, формировать расписание занятий, обрабатывать заявления студентов, формировать приказы академического характера и просматривать сводные отчеты по факультету. Особенностью данной роли является возможность управления данными в пределах закрепленных структурных подразделений – факультетов и кафедр.

Роль «Администратор системы» обладает неограниченными правами доступа ко всем функциям системы. Администраторы управляют учетными записями пользователей, настраивают систему ролей и разрешений, осуществляют мониторинг работы системы и выполняют техническое обслуживание. Для данной роли реализованы дополнительные механизмы безопасности, включая обязательную двухфакторную аутентификацию и запись всех действий в журнал аудита.

Помимо базовой системы ролей, в «УниверСфере» реализована система claims-based авторизации для управления тонкими разрешениями. Каждый пользователь может иметь набор дополнительных claims, которые определяют его специфические права доступа. Например, claim «DisciplineAccess» может предоставлять доступ к определенным дисциплинам, а claim «ReportGeneration» – право на формирование отчетных документов. Данная система позволяет гибко настраивать права доступа без необходимости создания дополнительных ролей.

Механизм авторизации интегрирован на всех уровнях приложения. На уровне контроллеров ASP.NET Core используются атрибуты Authorize с указанием требуемых ролей или политик доступа. Для сложных сценариев проверки прав доступа реализованы кастомные handlers авторизации, которые выполняют проверку прав на основе контекста запроса и бизнес-логики приложения. Например, при попытке доступа к журналу успеваемости система проверяет не только роль преподавателя, но и его принадлежность к конкретной дисциплине.

Система управления ролями и правами включает административный интерфейс, позволяющий назначать роли пользователям, управлять claims и настраивать политики доступа. Все изменения в системе прав доступа фиксируются в журнале безопасности, что обеспечивает возможность аудита и отслеживания всех операций, связанных с управлением доступом.

Для обеспечения безопасности сессий реализован механизм автоматического обновления JWT-токенов, который позволяет пользователям работать в системе без повторной аутентификации в течение установленного периода времени. При этом система отслеживает подозрительную активность и может принудительно завершить сессию при обнаружении аномальных действий.

Интеграция с внешними системами аутентификации реализована через поддержку протоколов OAuth 2.0 и OpenID Connect, что позволяет использовать единые учетные данные для доступа к нескольким системам университета. Система поддерживает единую точку входа (Single Sign-On) и обеспечивает безопасный обмен данными аутентификации между различными сервисами.

Разработанная система ролей и авторизации обеспечивает надежную защиту данных и функций системы, соответствуя принципу минимальных привилегий – каждый пользователь получает ровно те права, которые необходимы для выполнения его задач. Гибкая архитектура системы позволяет легко адаптировать модель прав доступа под изменяющиеся требования университета и обеспечивает масштабируемость решения при расширении функциональности системы.

2.3 Пользовательский интерфейс и бизнес-логика оператора

Пользовательский интерфейс для операторов системы «Унив», к которым относятся студенты и преподаватели, реализован с применением принципов юзабилити и эргономики, обеспечивающих эффективное взаимодействие пользователей с системой. Интерфейс построен на основе адаптивного дизайна, использующего библиотеку компонентов Material-UI, что гарантирует корректное отображение и удобство работы на различных устройствах – от настольных компьютеров до мобильных устройств. Основной акцент в дизайне сделан на минималистичном подходе, интуитивной навигации и визуальной иерархии элементов, позволяющей пользователям быстро находить необходимые функции и информацию.

Архитектура пользовательского интерфейса реализована как одностраничное приложение (SPA) на основе React 18 с использованием функциональных компонентов и хуков для управления состоянием. Маршрутизация между разделами системы организована через React Router v6, обеспечивающий плавные переходы без перезагрузки страницы. Состояние приложения управляется с помощью Redux Toolkit, что обеспечивает предсказуемость поведения интерфейса и централизованное управление данными. Для обработки асинхронных операций и взаимодействия с API используется Redux Thunk middleware, позволяющий организовать сложные цепочки запросов к серверу.

Главный портал оператора разделен на несколько функциональных зон. Панель навигации предоставляет доступ к основным модулям системы: расписанию, учебным материалам, академической успеваемости и системе заявлений. Информационная панель (dashboard) отображает персонифицированные данные и уведомления, релевантные конкретному пользователю. Для студентов на дашборде представлены ближайшие занятия, последние оценки и pending заявления, в то время как преподаватели видят текущую учебную нагрузку, предстоящие занятия и уведомления о новых заданиях для проверки.

Модуль работы с расписанием реализован с использованием кастомных React-компонентов, поддерживающих несколько режимов отображения – недельный, месячный и семестровый. Для визуализации расписания применяется библиотека FullCalendar, интегрированная с системой через кастомные адаптеры. Компонент поддерживает интерактивное взаимодействие – пользователи могут просматривать детальную информацию о каждом занятии, включая преподавателя, аудиторию и учебные материалы. Алгоритмы бизнес-логики на стороне клиента обеспечивают интеллектуальную фильтрацию и сортировку данных расписания based on выбранных критериев.

Система работы с учебными материалами включает компонент загрузки файлов с поддержкой drag-and-drop, реализованный на основе библиотеки react-dropzone. Пользовательский интерфейс поддерживает предпросмотр документов различных форматов, организацию материалов в иерархическую структуру и полнотекстовый поиск с использованием Elasticsearch. Бизнес-логика модуля включает валидацию загружаемых файлов по типу и размеру, автоматическую классификацию материалов и управление правами доступа к контенту.

Модуль академической успеваемости для студентов реализует сложные алгоритмы визуализации образовательной траектории. Компонент успеваемости использует библиотеку Chart.js для построения графиков прогресса и диаграмм распределения оценок. Бизнес-логика включает расчет среднего балла, прогнозирование академических рисков и формирование рекомендаций по улучшению успеваемости. Для преподавателей реализован специализированный интерфейс журнала успеваемости с возможностью массового ввода оценок и экспорта данных в различные форматы.

Система электронных заявлений построена на основе динамических форм, генерируемых по шаблонам. Компоненты форм реализованы с использованием Formik и Yup для валидации, что обеспечивает проверку данных на стороне клиента и отображение понятных сообщений об ошибках. Бизнес-логика обработки заявлений включает автоматическую маршрутизацию по инстанциям согласования, проверку соответствия академическим регламентам и интеграцию с системой уведомлений.

Особое внимание уделено системе уведомлений и коммуникации. Реализован компонент реального времени на основе SignalR, обеспечивающий мгновенное получение уведомлений о изменениях в расписании, новых оценках или обновлениях учебных материалов. Система поддерживает несколько каналов доставки уведомлений – встроенный центр уведомлений, email-рассылки и push-уведомления для мобильных устройств.

Производительность пользовательского интерфейса оптимизирована за счет реализации виртуализации для больших списков данных, ленивой загрузки компонентов и эффективного кэширования данных. Для обеспечения доступности интерфейса соблюдаются принципы WCAG 2.1 – обеспечена корректная работа с screen readers, поддержка клавиатурной навигации и достаточный цветовой контраст элементов.

Бизнес-логика оператора тесно интегрирована с серверной частью через REST API, при этом критически важные операции защищены от несанкционированного доступа системой авторизации. Все действия пользователя логируются для последующего аудита и анализа. Система предусматривает механизм восстановления сессии и автосохранения данных, обеспечивающий сохранность информации при непредвиденных прерываниях работы.

Интерфейс оператора поддерживает многоязычность и локализацию, позволяя адаптировать систему под различные языковые требования университета. Реализована система кастомизации интерфейса – пользователи могут настраивать цветовые схемы, расположение элементов и состав информационных панелей согласно своим предпочтениям.

Разработанный пользовательский интерфейс в сочетании с продуманной бизнес-логикой обеспечивает эффективное решение повседневных задач образовательного процесса, значительно сокращая временные затраты на административные операции и повышая общую удовлетворенность пользователей работой с системой.

2.4 Пользовательский интерфейс и бизнес-логика администратора

Пользовательский интерфейс администратора системы «Унив» представляет собой комплексную панель управления, обеспечивающую полный контроль над всеми аспектами функционирования информационной системы. Интерфейс реализован с использованием специализированных административных компонентов Ant Design Pro, которые предоставляют расширенные возможности для работы с большими объемами данных и сложными операциями управления. Основной принцип построения интерфейса – максимальная информационная плотность при сохранении удобства навигации и читаемости данных, что позволяет администраторам эффективно выполнять свои обязанности.

Архитектура административного интерфейса построена как отдельное SPA-приложение на React с выделенным набором маршрутов и состояний. Для управления сложным состоянием административной панели используется комбинация Redux для глобального состояния и React Context для локального состояния компонентов. Применен подход modular architecture, где каждый административный модуль представляет собой независимый функциональный блок с собственной логикой и компонентами интерфейса.

Центральным элементом интерфейса является административный дашборд, предоставляющий сводную информацию о состоянии системы в реальном времени. Дашборд включает метрики производительности – загрузку процессора, использование памяти, активные сессии пользователей, а также бизнес-показатели – количество новых пользователей, активных заявлений, академических операций. Для визуализации метрик используются кастомные компоненты на основе D3.js, обеспечивающие детальную аналитику и возможность глубокого анализа данных.

Модуль управления пользователями реализует сложную бизнес-логику работы с учетными записями. Интерфейс предоставляет расширенные возможности фильтрации и поиска по множеству параметров – роль, статус, факультет, дата регистрации. Реализованы batch-операции – массовое создание пользователей при загрузке из CSV-файлов, групповое назначение ролей, пакетная активация и деактивация учетных записей. Бизнес-логика включает валидацию данных при создании пользователей, проверку уникальности учетных записей и автоматическую генерацию временных паролей с требованием обязательной смены при первом входе.

Система управления ролями и разрешениями предоставляет визуальный редактор политик доступа, позволяющий конфигурировать сложные правила авторизации. Интерфейс включает древовидное представление разрешений с возможностью массового назначения прав для групп пользователей. Бизнес-логика обеспечивает проверку конфликтующих разрешений и предотвращает создание политик, нарушающих принципы безопасности системы. Реализован механизм наследования прав, позволяющий создавать иерархические структуры доступа.

Модуль мониторинга системы интегрирован с Application Insights и предоставляет детальную телеметрию работы всех компонентов. Административный интерфейс включает продвинутые инструменты для анализа производительности – трейсинг запросов, мониторинг зависимостей, отслеживание исключений. Бизнес-логика обработки метрик включает алгоритмы обнаружения аномалий и автоматического оповещения при превышении пороговых значений. Реализована система оповещений с настраиваемыми триггерами и многоуровневой эскалацией уведомлений.

Подсистема аудита и логирования предоставляет комплексный интерфейс для просмотра и анализа журналов безопасности. Компонент поддерживает сложные запросы к логам с использованием языка Kusto Query Language, временные шкалы событий и корреляцию связанных операций. Бизнес-логика включает компрессию и ротацию логов, автоматическое обнаружение подозрительной активности и генерацию отчетов соответствия требованиям безопасности.

Интерфейс управления конфигурацией системы позволяет администраторам изменять параметры работы без необходимости перезапуска приложения. Реализован визуальный редактор appsettings.json с валидацией схемы конфигурации и возможностью управления различными environment-окружениями. Бизнес-логика обеспечивает контроль целостности конфигурации, резервное копирование настроек и ведение истории изменений с возможностью отката.

Модуль резервного копирования и восстановления данных предоставляет графический интерфейс для управления политиками бэкапов. Администраторы могут настраивать расписания, выбирать объекты для резервного копирования и отслеживать статус операций. Бизнес-логика включает проверку целостности бэкапов, автоматическое тестирование восстановления и оптимизацию хранилища резервных копий.

Система управления инцидентами реализована как полнофункциональная служба Service Desk. Интерфейс включает канбан-доски для отслеживания инцидентов, шаблоны автоматических ответов и систему эскалации. Бизнес-логика поддерживает автоматическую классификацию инцидентов, назначение исполнителей based on нагрузки и экспертизы, интеграцию с системами мониторинга для проактивного обнаружения проблем.

Для обеспечения безопасности административного интерфейса реализованы строгие проверки доступа на уровне отдельных компонентов. Все действия администратора проходят двойную авторизацию – на уровне маршрутов и на уровне операций. Критические операции требуют дополнительной аутентификации и подтверждения второй стороной. Система автоматически завершает административные сессии при длительном бездействии и ведет детальный аудит всех операций управления.

Интерфейс адаптирован для работы с большими объемами данных – реализована пагинация, виртуализация таблиц, ленивая загрузка и прогрессивный рендеринг. Производительность оптимизирована за счет агрессивного кэширования статических данных и применения стратегий prefetching для предзагрузки вероятно запрашиваемой информации.

Административный интерфейс «Унив» представляет собой мощный инструмент управления, сочетающий богатый функционал с высокой производительностью и безопасностью. Продуманная бизнес-логика и эргономичный дизайн позволяют администраторам эффективно решать задачи любой сложности, обеспечивая стабильную и безопасную работу всей образовательной платформы.

ЗАКЛЮЧЕНИЕ

Заключение должно представлять собой итог проделанной работы, оно должно содержать общие выводы по теме, указания на наиболее интересные и проблемные положения. Здесь автор должен кратко повторить свои предложения и рекомендации.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Абрамов, Г. В. Проектирование и разработка информационных систем : учебное пособие для СПО / Г. В. Абрамов, И. Е. Медведкова, Л. А. Коробова. — Саратов : Профобразование, 2020. — 169 c. — ISBN 978-5-4488-0730-5. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/88888. — Режим доступа: для авторизир. пользователей
2. Вичугова А.А. Инструментальные средства разработки компьютерных систем и комплексов [Электронный ресурс] : учебное пособие для СПО / А.А. Вичугова. — Электрон. текстовые данные. — Саратов: Профобразование, 2019. — 135 c. — 978-5-4488-0015-3. — Режим доступа: http://www.iprbookshop.ru/66387.html
3. Котляров, В. П. Основы тестирования программного обеспечения : учебное пособие для СПО / В. П. Котляров. — Саратов : Профобразование, 2019. — 335 c. — ISBN 978-5-4488-0364-2. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/86202 . — Режим доступа: для авторизир. пользователей
4. Профессиональное управление проектом. — : учебное пособие / К. Хелдман, пер. с англ. — 6-е издание. — Москва : Лаборатория знаний, 2019. — 731 с. — ISBN 978-5-9963-2503-0. Номер в ЭБС: 924003- https://www.book.ru/book/924003
5. Спицина, И. А. Разработка информационных систем. Пользовательский интерфейс : учебное пособие для СПО / И. А. Спицина, К. А. Аксёнов ; под редакцией Л. Г. Доросинского. — 2-е изд. — Саратов, Екатеринбург : Профобразование, Уральский федеральный университет, 2020. — 98 c. — ISBN 978-5-4488-0768-8, 978-5-7996-2872-7. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/92370. — Режим доступа: для авторизир. пользователей

ПРИЛОЖЕНИЕ А

**Наименование приложения**