|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных Технологий

Кафедра Вычислительной Техники (ВТ)

**ОТЧЁТ ПО ПРАКТИЧЕСКИМ РАБОТАМ**

по дисциплине

«Проектирование интеллектуальных систем (часть 1/2)»

|  |  |
| --- | --- |
| Студент группы:  ИКБО-15-22 | Оганнисян Г.А. |
| Руководитель работы | Холмогоров В.В.. |

Москва 2025 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc199845352)

[1 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ ЦИФРОВОЙ ОЧЕРЕДИ 5](#_Toc199845353)

[**1.1 Архитектура backend-решения 5**](#_Toc199845354)

[**1.2 Технологический стек backend 6**](#_Toc199845355)

[**1.3 Модели данных и структура базы данных 8**](#_Toc199845356)

[**1.4 Программная реализация 11**](#_Toc199845357)

[2 РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ СИСТЕМЫ ЦИФРОВОЙ ОЧЕРЕДИ 12](#_Toc199845358)

[**2.1 Архитектура frontend-решения 12**](#_Toc199845359)

[**2.2 Технологический стек frontend 14**](#_Toc199845360)

[**2.3 Интерфейс пользователя 16**](#_Toc199845361)

[**2.4 Программная реализация 19**](#_Toc199845362)

[ЗАКЛЮЧЕНИЕ 21](#_Toc199845363)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 23](#_Toc199845364)

[ПРИЛОЖЕНИЯ 24](#_Toc199845365)

**ВВЕДЕНИЕ**

В современном образовательном процессе важную роль играет эффективная организация взаимодействия между студентами и преподавателями. Одним из ключевых аспектов этого взаимодействия является процесс сдачи лабораторных и практических работ. Традиционный подход к организации этого процесса часто сопряжен с рядом проблем, таких как:

* Неэффективное использование времени студентов и преподавателей из-за отсутствия структурированной очереди;
* Отсутствие прозрачности в процессе ожидания своей очереди;
* Сложности в планировании времени как для студентов, так и для преподавателей;
* Неравномерное распределение нагрузки на преподавателей в течение семестра.

Эти проблемы особенно актуальны в условиях увеличения числа студентов и ограниченного времени преподавателей. Для решения указанных проблем была разработана система цифровой очереди "Loremru - Digital Queue", которая позволяет эффективно организовать процесс сдачи лабораторных и практических работ.

Система "Loremru - Digital Queue" представляет собой веб-приложение с клиент-серверной архитектурой, которое обеспечивает:

* Автоматическое формирование очередей на основе расписания занятий;
* Возможность для студентов записываться в очередь и отслеживать свое положение в ней;
* Прозрачность процесса для всех участников;
* Оптимизацию времени ожидания студентов;
* Равномерное распределение нагрузки на преподавателей.

Целью данной практической работы является разработка и реализация системы цифровой очереди для оптимизации процесса сдачи лабораторных и практических работ в образовательных учреждениях.

Для достижения поставленной цели были определены следующие задачи:

1. Анализ существующих проблем в организации процесса сдачи работ;
2. Проектирование архитектуры системы цифровой очереди;
3. Разработка серверной части системы;
4. Разработка клиентской части системы;
5. Интеграция компонентов и тестирование системы;
6. Подготовка документации и планирование дальнейшего развития проекта.

Практическая значимость работы заключается в создании инструмента, который позволит оптимизировать процесс сдачи лабораторных и практических работ, сократить время ожидания студентов и повысить эффективность работы преподавателей.

В данном отчете представлены результаты разработки системы "Loremru - Digital Queue", включая описание архитектуры, технологического стека, программной реализации, а также результаты тестирования и планы дальнейшего развития проекта..

1. **РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ ЦИФРОВОЙ ОЧЕРЕДИ**
   1. **Архитектура backend-решения**

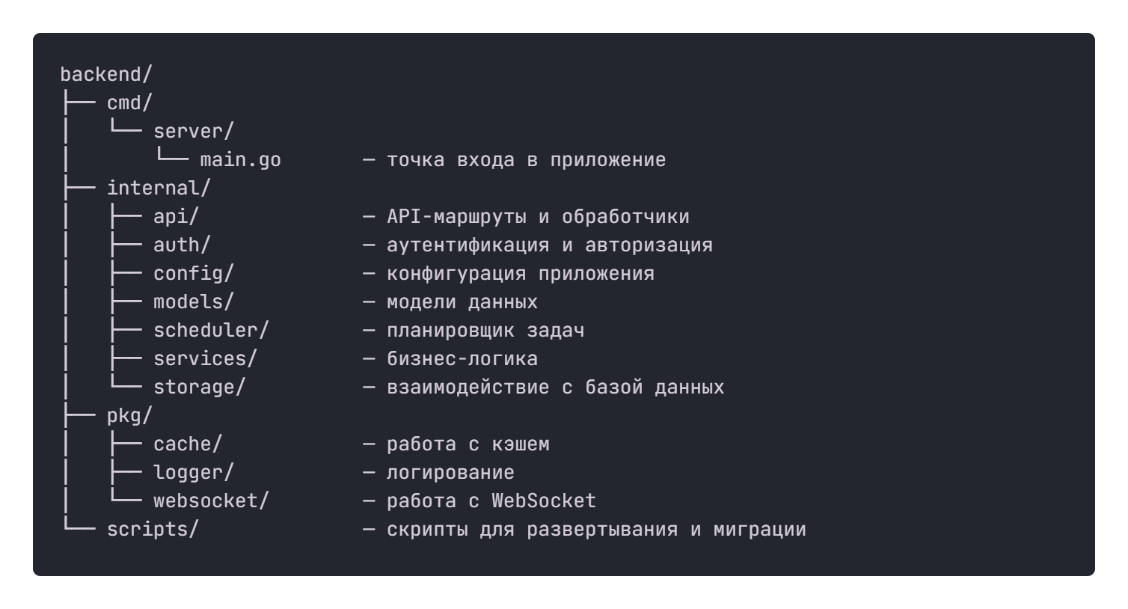
Серверная часть системы "Loremru - Digital Queue" реализована на основе микросервисной архитектуры с использованием языка программирования Go и фреймворка Gin. Данная архитектура обеспечивает высокую производительность, масштабируемость и надежность системы.

* + 1. Основные компоненты Backend

Серверная часть системы состоит из следующих основных компонентов:

* API-сервер — обрабатывает HTTP-запросы от клиентской части, реализует бизнес-логику и взаимодействует с базой данных.
* База данных — хранит информацию о пользователях, группах, расписании и очередях.
* Кэш — обеспечивает быстрый доступ к часто запрашиваемым данным.
* WebSocket-сервер — обеспечивает обновления в режиме реального времени.
* Планировщик задач — выполняет периодические задачи, такие как автоматическое открытие и закрытие очередей.
  + 1. **Структура Backend**

Серверная часть системы имеет следующую структуру (рисунок 1.1):



**Рисунок 1.1 – Структура серверной части.**

* + 1. **Архитектурные решения**

В разработке серверной части системы были применены следующие архитектурные решения:

1. **Чистая архитектура** — разделение приложения на слои (модели, сервисы, обработчики) для обеспечения независимости компонентов и упрощения тестирования.
2. **Dependency Injection** — внедрение зависимостей для обеспечения гибкости и тестируемости кода.
3. **Repository Pattern** — абстракция доступа к данным, позволяющая изменять источник данных без изменения бизнес-логики.
4. **Middleware** — промежуточное ПО для обработки запросов, такое как аутентификация, логирование и обработка ошибок.
5. **CQRS (Command Query Responsibility Segregation)** — разделение операций чтения и записи для оптимизации производительности.
   1. **Технологический стек backend**

Серверная часть системы "Loremru - Digital Queue" реализована с использованием следующих технологий:

* + 1. **Язык программирования и фреймворки**
* **Go** — современный компилируемый язык программирования, разработанный компанией Google. Go обеспечивает высокую производительность, статическую типизацию и эффективную работу с памятью, что делает его идеальным выбором для разработки высоконагруженных серверных приложений.
* **Gin** — высокопроизводительный фреймворк для разработки веб-приложений на Go. Gin обеспечивает быструю обработку HTTP-запросов, маршрутизацию, валидацию данных и другие функции, необходимые для разработки REST API.
  + 1. **База данных и кэширование**
* **PostgreSQL** — объектно-реляционная система управления базами данных. PostgreSQL обеспечивает надежное хранение данных, поддержку транзакций, сложных запросов и индексов, что делает его идеальным выбором для хранения структурированных данных системы.
* **Redis** — высокопроизводительное хранилище данных типа "ключ-значение". Redis используется для кэширования часто запрашиваемых данных, таких как информация о группах и расписании, что позволяет снизить нагрузку на базу данных и повысить производительность системы.
  + 1. **ORM и миграции**
* **GORM** — объектно-реляционный маппер для Go. GORM обеспечивает удобный интерфейс для работы с базой данных, автоматическое создание и миграцию таблиц, а также поддержку различных СУБД.
* **gorm.AutoMigrate** — инструмент для управления миграциями базы данных. AutoMigrate обеспечивает версионирование схемы базы данных, что позволяет безопасно обновлять структуру базы данных без потери данных.
  + 1. **Аутентификация и авторизация**
* **JWT (JSON Web Tokens)** — стандарт для создания токенов доступа. JWT обеспечивает безопасную передачу информации между клиентом и сервером, а также позволяет реализовать механизм аутентификации и авторизации без необходимости хранения состояния на сервере.
* **bcrypt** — алгоритм хеширования паролей. bcrypt обеспечивает безопасное хранение паролей пользователей в базе данных, защищая их от различных типов атак.
  + 1. **WebSocket и обновления в реальном времени**
* **gorilla/websocket** — библиотека для работы с WebSocket в Go. gorilla/websocket обеспечивает полнодуплексную связь между клиентом и сервером, что позволяет реализовать обновления в режиме реального времени.
  + 1. **Контейнеризация и оркестрация**
* Docker — платформа для разработки, доставки и запуска приложений в контейнерах. Docker обеспечивает изоляцию приложений и их зависимостей, что упрощает развертывание и масштабирование системы.
* Docker Compose — инструмент для определения и запуска многоконтейнерных приложений. Docker Compose обеспечивает удобное управление контейнерами, что упрощает локальную разработку и тестирование.

## Модели данных и структура базы данных

Система "Loremru - Digital Queue" использует реляционную базу данных PostgreSQL для хранения информации о пользователях, группах, расписании и очередях. Структура базы данных разработана с учетом требований нормализации и оптимизации производительности.

* + 1. **Основные сущности**

В системе определены следующие основные сущности:

* User — пользователь системы
* Group — учебная группа.
* Schedule — расписание занятий.
* Queue — очередь для сдачи работ.
  + 1. **Модели данных**
       1. **Модель User**

Модель User представляет пользователя системы и содержит следующие поля:

* ID — уникальный идентификатор пользователя.
* Email — электронная почта пользователя (используется для входа в систему).
* PasswordHash — хеш пароля пользователя.
* FirstName — имя пользователя.
* LastName — фамилия пользователя.
* Role — роль пользователя (student, teacher, admin).
* LastLogin — время последнего входа в систему.
* Groups — группы, к которым принадлежит пользователь (для студентов).
* Queues — очереди, в которых участвует пользователь.
  + - 1. **Модель Group**

Модель Group представляет учебную группу и содержит следующие поля:

* ID — уникальный идентификатор группы.
* Name — название группы.
* Faculty — факультет.
* Department — кафедра.
* Users — пользователи, принадлежащие к группе.
* Schedules — расписание занятий для группы.
  + - 1. **Модель Schedule**

Модель Schedule представляет расписание занятий и содержит следующие поля:

* ID — уникальный идентификатор расписания.
* GroupID — идентификатор группы.
* DayOfWeek — день недели (1 = Понедельник, 7 = Воскресенье).
* StartTime — время начала занятия.
* EndTime — время окончания занятия.
* Subject — предмет.
* Teacher — преподаватель.
* Room — аудитория.
* Group — группа, для которой предназначено расписание.
* Queues — очереди, связанные с данным расписанием.
  + - 1. **Модель Queue**

Модель Queue представляет очередь для сдачи работ и содержит следующие поля:

* ID — уникальный идентификатор очереди.
* ScheduleID — идентификатор расписания.
* Date — дата проведения занятия.
* IsOpen — флаг, указывающий, открыта ли очередь.
* MaxParticipants — максимальное количество участников.
* Schedule — расписание, с которым связана очередь.
* Participants — участники очереди.
  + 1. **Связи между сущностями**

В системе определены следующие связи между сущностями:

* User-Group — связь "многие ко многим" через таблицу user\_groups.
* Group-Schedule — связь "один ко многим" (одна группа может иметь несколько расписаний).
* Schedule-Queue — связь "один ко многим" (одно расписание может иметь несколько очередей).
* User-Queue — связь "многие ко многим" через таблицу queue\_participants.
  + 1. **Индексы и оптимизация**

Для оптимизации производительности базы данных были созданы следующие индексы:

* Индекс по полю Email в таблице users — для быстрого поиска пользователей по электронной почте.
* Индекс по полю GroupID в таблице schedules — для быстрого поиска расписания по группе.
* Индекс по полю ScheduleID в таблице queues — для быстрого поиска очередей по расписанию.
* Составной индекс по полям Date и IsOpen в таблице queues — для быстрого поиска открытых очередей на определенную дату.

## Программная реализация

В данном разделе представлена программная реализация серверной части системы "Loremru - Digital Queue", включая основные компоненты и их взаимодействие.

* + 1. **Точка входа в приложение**

Точкой входа в приложение является файл main.go, который инициализирует все необходимые компоненты и запускает HTTP-сервер.

* + 1. **Обработчики HTTP-запросов**

Обработчики HTTP-запросов реализуют REST API для взаимодействия с клиентской частью. Все эндпоинты документированы с использованием Swagger, что облегчает интеграцию и тестирование.

* + 1. **WebSocket для обновлений в реальном времени**

Для обеспечения обновлений в режиме реального времени используется WebSocket. Серверная часть устанавливает WebSocket-соединение с клиентской частью и отправляет уведомления о изменениях в очередях.

* + 1. **Планировщик задач**

Планировщик задач выполняет периодические задачи, такие как автоматическое открытие и закрытие очередей в соответствии с расписанием занятий.

1. **РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ СИСТЕМЫ ЦИФРОВОЙ ОЧЕРЕДИ**
   1. **Архитектура frontend-решения**

Клиентская часть системы "Loremru - Digital Queue" реализована как одностраничное приложение (SPA) на базе React и TypeScript с использованием Vite в качестве инструмента сборки. Такая архитектура обеспечивает высокую производительность, удобство разработки и отзывчивый пользовательский интерфейс.

* + 1. Основные компоненты Frontend

Клиентская часть системы состоит из следующих основных компонентов:

1. **React** — библиотека для построения пользовательских интерфейсов, обеспечивающая компонентный подход к разработке.
2. **TypeScript** — типизированный язык программирования, обеспечивающий более надежный и поддерживаемый код.
3. **Material UI** — библиотека компонентов для создания современного и отзывчивого интерфейса.
4. **React Router** — библиотека для маршрутизации в React-приложении.
5. **WebSocket Client** — компонент для установления WebSocket-соединения с сервером и получения обновлений в режиме реального времени.
   * 1. Структура Frontend

Клиентская часть системы имеет следующую структуру (рисунок 2.1):

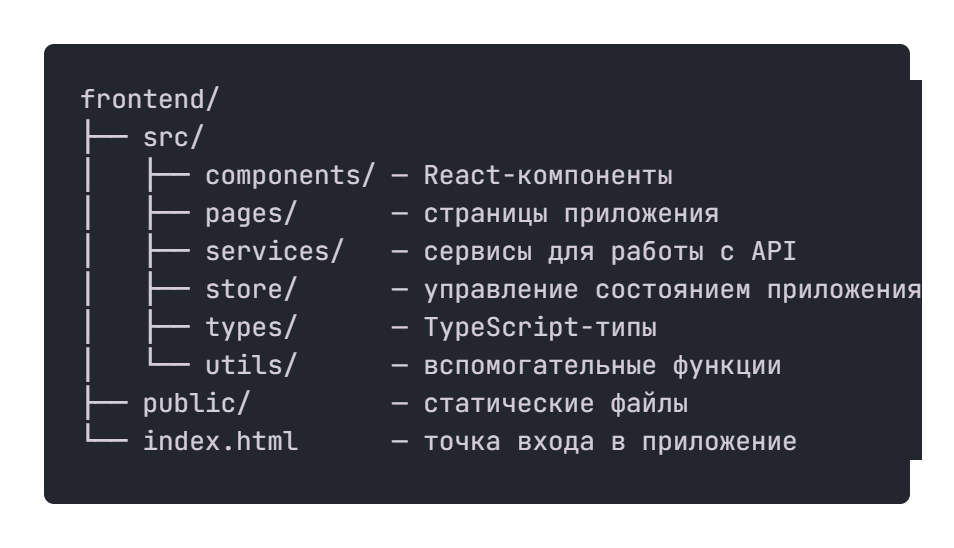


Рисунок 2.1 – Структура клиентской части.

* + 1. Архитектурные решения

В разработке клиентской части системы были применены следующие архитектурные решения:

1. Компонентный подход — интерфейс разделен на независимые компоненты, что обеспечивает переиспользуемость кода и упрощает тестирование.
2. Централизованное управление состоянием — использование контекста React для управления глобальным состоянием приложения.
3. Разделение бизнес-логики и представления — использование сервисов для работы с API и отделение их от компонентов представления.
4. Типизация данных — использование TypeScript для обеспечения типобезопасности и улучшения разработки.
5. Адаптивный дизайн — интерфейс адаптируется под различные размеры экранов, обеспечивая удобство использования как на десктопных, так и на мобильных устройствах.
   1. **Технологический стек frontend**

Клиентская часть системы "Loremru - Digital Queue" реализована с использованием следующих технологий:

* + 1. Язык программирования и фреймворки
* **React** — библиотека для построения пользовательских интерфейсов, разработанная компанией Facebook. React обеспечивает компонентный подход к разработке, что позволяет создавать переиспользуемые компоненты и упрощает поддержку кода.
* **TypeScript** — типизированный язык программирования, расширяющий возможности JavaScript. TypeScript обеспечивает статическую типизацию, что позволяет выявлять ошибки на этапе компиляции и улучшает процесс разработки.
* **Vite** — современный инструмент сборки, обеспечивающий быструю разработку и оптимизированную сборку для продакшена. Vite использует нативные ES-модули для разработки, что обеспечивает мгновенный запуск сервера разработки и быструю перезагрузку при изменении кода.
  + 1. Библиотеки компонентов и стилизация
* **Material UI** — библиотека компонентов для React, реализующая дизайн-систему Material Design от Google. Material UI предоставляет готовые компоненты для создания современного и отзывчивого интерфейса.
* **Emotion** — библиотека для стилизации компонентов с использованием CSS-in-JS. Emotion обеспечивает высокую производительность и удобство написания стилей.
* **Material Icons** — набор иконок, разработанный Google в соответствии с принципами Material Design. Material Icons обеспечивает единообразие и узнаваемость интерфейса.
  + 1. Маршрутизация и управление состоянием
* **React Router** — библиотека для маршрутизации в React-приложениях. React Router обеспечивает навигацию между различными страницами приложения без перезагрузки страницы.
* **React Context API** — API для управления глобальным состоянием приложения. React Context API обеспечивает передачу данных через дерево компонентов без необходимости передавать пропсы на каждом уровне.
  + 1. Работа с API и асинхронными операциями
* **Axios** — библиотека для выполнения HTTP-запросов. Axios обеспечивает удобный интерфейс для работы с REST API и поддерживает перехватчики запросов и ответов.
* **React Query** — библиотека для управления асинхронными данными в React-приложениях. React Query обеспечивает кэширование данных, автоматическую повторную выборку и управление состоянием загрузки.
  + 1. WebSocket и обновления в реальном времени
* WebSocket — протокол связи, обеспечивающий полнодуплексный канал связи через одно TCP-соединение. WebSocket используется для обеспечения обновлений в режиме реального времени.
  + 1. Валидация данных
* Yup — библиотека для валидации данных. Yup обеспечивает декларативный способ определения схем валидации и интеграцию с формами.
* Formik — библиотека для работы с формами в React. Formik упрощает процесс создания форм, валидации данных и обработки отправки форм.
  + 1. Сборка и развертывание
* **ESLint** — инструмент для статического анализа кода. ESLint обеспечивает проверку кода на соответствие стандартам и выявление потенциальных ошибок.
* **Prettier** — инструмент для форматирования кода. Prettier обеспечивает единообразное форматирование кода в проекте.
* **Docker** — платформа для разработки, доставки и запуска приложений в контейнерах. Docker обеспечивает изоляцию приложений и их зависимостей, что упрощает развертывание и масштабирование системы.

## Интерфейс пользователя

Интерфейс пользователя системы "Loremru - Digital Queue" разработан с учетом принципов Material Design и обеспечивает удобство использования как на десктопных, так и на мобильных устройствах:

* + 1. Основные экраны приложения

Клиентская часть системы включает следующие основные экраны:

* **Страница авторизации** — экран для входа в систему или регистрации нового пользователя.
* **Дашборд** — главная страница с основной информацией и доступом к ключевым функциям.
* **Расписание** — страница с отображением расписания занятий и возможностью записи в очередь.
* **Очереди** — страница с отображением очередей, в которых участвует пользователь.
* **Профиль** — страница с информацией о пользователе и возможностью управления учетной записью.
  + 1. Страница авторизации.

Страница авторизации предоставляет пользователю возможность входа в систему или регистрации нового аккаунта. Страница содержит следующие элементы:

* **Форма входа** — поля для ввода электронной почты и пароля, кнопка "Войти".
* **Форма регистрации** — поля для ввода имени, фамилии, электронной почты и пароля, кнопка "Зарегистрироваться".
* **Переключение между формами** — кнопки для переключения между формами входа и регистрации.
  + 1. Дашборд

Дашборд является главной страницей приложения и предоставляет пользователю доступ к основным функциям системы. Дашборд содержит следующие элементы:

* **Верхняя панель** — логотип системы, кнопка меню, информация о пользователе.
* **Боковое меню** — навигация по разделам приложения.
* **Карточки с информацией** — краткая информация о расписании, очередях и других важных аспектах.
* **Уведомления** — список последних уведомлений.
  + 1. Расписание

Страница расписания отображает расписание занятий для выбранной группы и предоставляет возможность записи в очередь. Страница содержит следующие элементы:

* **Выбор группы** — выпадающий список для выбора группы.
* **Выбор дня недели** — вкладки для выбора дня недели.
* **Список занятий** — список занятий на выбранный день недели с информацией о предмете, преподавателе, аудитории и времени.
* **Кнопки действий** — кнопки для записи в очередь, просмотра информации о занятии и т.д.
  + 1. Очереди

Страница очередей отображает очереди, в которых участвует пользователь, и предоставляет возможность управления своим участием в очередях. Страница содержит следующие элементы:

* **Вкладки** — вкладки для переключения между активными очередями и очередями, в которых участвует пользователь.
* **Список очередей** — список очередей с информацией о предмете, преподавателе, аудитории, времени и статусе очереди.
* **Карточка очереди** — детальная информация об очереди, включая список участников и позицию пользователя в очереди.
* **Кнопки действий** — кнопки для вступления в очередь, выхода из очереди и т.д.
  + 1. Профиль

Страница профиля отображает информацию о пользователе и предоставляет возможность управления учетной записью. Страница содержит следующие элементы:

* **Информация о пользователе** — имя, фамилия, электронная почта, роль, группа и т.д.
* **Редактирование профиля** — форма для изменения информации о пользователе.
* **Изменение пароля** — форма для изменения пароля.
* **История активности** — список последних действий пользователя в системе.
  + 1. Особенности интерфейса

Интерфейс пользователя системы "Loremru - Digital Queue" обладает следующими особенностями:

* **Адаптивный дизайн** — интерфейс адаптируется под различные размеры экранов, обеспечивая удобство использования как на десктопных, так и на мобильных устройствах.
* **Интуитивно понятная навигация** — навигация по приложению осуществляется через боковое меню на десктопных устройствах и через нижнюю панель навигации на мобильных устройствах.
* **Визуальная обратная связь** — все действия пользователя сопровождаются визуальной обратной связью, что улучшает пользовательский опыт.
* **Уведомления** — система уведомляет пользователя о важных событиях, таких как приближение очереди или изменение статуса очереди.
* **Темная тема** — поддержка светлой и темной темы оформления для комфортного использования в различных условиях освещения.

## Программная реализация

В данном разделе представлена программная реализация клиентской части системы "Loremru - Digital Queue", включая основные компоненты и их взаимодействие.

* + 1. Типы данных

Для обеспечения типобезопасности были определены TypeScript-интерфейсы для всех используемых данных, включая модели пользователей, групп, расписания и очередей.

* + 1. Сервисы для работы с API

Для взаимодействия с серверной частью были разработаны сервисы, инкапсулирующие логику работы с API, включая аутентификацию, работу с группами, расписанием и очередями.

* + 1. WebSocket для обновлений в реальном времени

Для получения обновлений в режиме реального времени был реализован сервис для работы с WebSocket, обеспечивающий подключение к серверу, обработку сообщений и автоматическое переподключение при разрыве соединения.

* + 1. Компоненты пользовательского интерфейса

Для создания пользовательского интерфейса были разработаны различные React-компоненты, включая компоненты для отображения очередей, расписания, форм авторизации и регистрации, а также компоненты для навигации и отображения уведомлений.

* + 1. Страницы приложения

Для навигации по приложению были разработаны различные страницы, включая страницу авторизации, дашборд, страницы с расписанием, очередями и профилем пользователя.

**ЗАКЛЮЧЕНИЕ**

В рамках данной практической работы была разработана система цифровой очереди "Loremru - Digital Queue", предназначенная для оптимизации процесса сдачи лабораторных и практических работ в образовательных учреждениях..

В ходе выполнения работы были решены следующие задачи:

1. Проведен анализ существующих проблем в организации процесса сдачи работ, выявлены основные недостатки традиционного подхода, такие как неэффективное использование времени, отсутствие прозрачности и сложности в планировании.
2. Спроектирована архитектура системы цифровой очереди, основанная на клиент-серверной модели с использованием современных технологий и архитектурных решений.
3. Разработана серверная часть системы на языке Go с использованием фреймворка Gin, реализующая REST API для взаимодействия с клиентской частью, WebSocket для обновлений в режиме реального времени и интеграцию с базой данных PostgreSQL и кэшем Redis.
4. Разработана клиентская часть системы на базе React и TypeScript с использованием Material UI для создания современного и отзывчивого пользовательского интерфейса.
5. Проведена интеграция компонентов системы и комплексное тестирование на различных уровнях, включая модульное, интеграционное и системное тестирование.
6. Подготовлена документация и разработаны планы дальнейшего развития проекта, включающие расширение функциональности, интеграцию с другими системами и улучшение пользовательского опыта.

Разработанная система "Loremru - Digital Queue" обладает следующими преимуществами:

1. **Эффективность** — система оптимизирует процесс сдачи работ, сокращая время ожидания студентов и повышая эффективность работы преподавателей.
2. **Прозрачность** — система обеспечивает прозрачность процесса для всех участников, предоставляя информацию о составе и порядке очереди в режиме реального времени.
3. **Удобство использования** — система имеет интуитивно понятный интерфейс, адаптированный для различных устройств, что обеспечивает удобство использования как для студентов, так и для преподавателей.
4. **Масштабируемость** — архитектура системы позволяет легко масштабировать ее для обслуживания большого количества пользователей и интеграции с другими системами.
5. **Надежность** — система разработана с использованием современных технологий и архитектурных решений, обеспечивающих высокую надежность и отказоустойчивость.

Таким образом, разработанная система "Loremru - Digital Queue" является эффективным инструментом для оптимизации процесса сдачи лабораторных и практических работ в образовательных учреждениях. Система имеет значительный потенциал для дальнейшего развития и улучшения, что позволит сделать процесс сдачи работ еще более эффективным и удобным для всех участников образовательного процесса.

Практическая значимость работы заключается в создании готового к внедрению решения, которое может быть использовано в различных образовательных учреждениях для оптимизации учебного процесса. Система "Loremru - Digital Queue" может быть адаптирована для различных сценариев использования, таких как организация консультаций, проведение защит проектов и других мероприятий, требующих управления очередями.

**СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ**

1. Документация Go [Электронный ресурс]. — Режим доступа: <https://golang.org/doc/> (дата обращения: 01.04.2025).
2. Документация Gin Web Framework [Электронный ресурс]. — Режим доступа: <https://gin-gonic.com/en/docs/> (дата обращения: 01.04.2025).
3. Документация React [Электронный ресурс]. — Режим доступа: <https://react.dev/learn> (дата обращения: 01.04.2025).
4. Документация TypeScript [Электронный ресурс]. — Режим доступа: <https://www.typescriptlang.org/docs/> (дата обращения: 01.04.2025).

**ПРИЛОЖЕНИЯ**

Приложение А – Ссылки на исходный код серверной части.

Приложение Б – Ссылки на исходный код клиентской части.

**Приложение А**

Ссылки на исходный код серверной части

Исходный код серверной части системы "Loremru - Digital Queue" доступен в репозитории GitHub по следующим ссылкам:

1. Основной репозиторий проекта: <https://github.com/Anabol1ks/digital-queue>
2. Репозиторий серверной части (backend): <https://github.com/Anabol1ks/digital-queue-backend/>

**Приложение Б**

Ссылки на исходный код клиентской части

Исходный код клиентской части системы "Loremru - Digital Queue" доступен в репозитории GitHub по следующим ссылкам:

1. Репозиторий клиентской части(frontend):

<https://github.com/imurbestfriend/digital-queue-frontend>