Частное учреждение образования

«Колледж бизнеса и права»

ДОПУЩЕН К ЗАЩИТЕ

Заместитель директора по учебной работе

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.В.Малафей

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ДП Т.218001.401

Председатель цикловой комиссии ( )

Руководитель проекта ( С. В. Банцевич )

Консультант по экономической части ( М. А. Григораш )

Консультант по охране труда ( Н. К. Фоменко )

Обучающаяся ( В. А. Белисова )

Рецензент ( )

2025

**СОДЕРЖАНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

*ДП Т.218001.401 ПЗ*

Разраб.

Белисова В. А.

Провер.

Банцевич С. В.

Т. контр.

Н. контр.

Утверд.

Разработка веб-приложения для управления проектами

Лит.

Листов

КБП

у

[Введение 4](#_Toc194865040)

[1 Описание задачи 6](#_Toc194865041)

[1.1 Анализ предметной области 6](#_Toc194865042)

[1.2 Постановка задачи 7](#_Toc194865043)

[2 Проектирование веб-приложения 9](#_Toc194865044)

[2.1 Проектирование модели 9](#_Toc194865045)

[2.2 Требование к веб-приложению 11](#_Toc194865046)

[2.3 Структура веб-приложения 12](#_Toc194865047)

[2.4 Проектирование макета веб-приложения 13](#_Toc194865048)

[2.5 Программно-технические средства, необходимые для разработки приложения 14](#_Toc194865049)

[2.6 Защита и сохранность данных 16](#_Toc194865050)

[2.7 Организация и ведение информационной модели 17](#_Toc194865051)

[3 Реализация веб-приложения 21](#_Toc194865052)

[3.1 Описание разделов веб-приложения 21](#_Toc194865053)

[3.2 Разработка административной части приложения 22](#_Toc194865054)

[3.3 Разработка клиентской части приложения 23](#_Toc194865055)

[3.4 Описание используемых функций и процедур 24](#_Toc194865056)

[3.5 Функциональное тестирование 29](#_Toc194865057)

[4 Применение 34](#_Toc194865058)

[4.1 Назначение веб-приложения 34](#_Toc194865059)

[4.2 Программно-аппаратное обеспечение сервера и клиента 34](#_Toc194865060)

[5 Охрана труда и окружающей среды 36](#_Toc194865061)

[5.1 Правовые, нормативные, социально-экономические и организационные вопросы охраны труда 36](#_Toc194865062)

[5.2 Разработка мер по нормализации и защите от токсических веществ на производстве 37](#_Toc194865063)

[5.3 Пожарная безопасность 39](#_Toc194865064)

[5.4 Охрана окружающей среды 40](#_Toc194865065)

[6 Экономический раздел 42](#_Toc194865066)

[6.1 Расчет затрат на разработку веб-приложения 42](#_Toc194865067)

[6.2 Расчет показателей экономической эффективности разработки 44](#_Toc194865068)

[6.3 Расчет экономической эффективности у пользователя веб-приложения 45](#_Toc194865069)

[Заключение 46](#_Toc194865070)

[Список использованных источников 47](#_Toc194865071)

[ПРИЛОЖЕНИЕ А 49](#_Toc194865072)

[ПРИЛОЖЕНИЕ Б 76](#_Toc194865073)

**Введение**

В современном мире, где ценится эффективность и оперативность, важность использования программного обеспечения для управления проектами становится ключевым элементом повышения производительности и оптимизации бизнес-процессов. С увеличением сложности задач и объемов работы в компаниях, занимающихся проектной деятельностью, цифровые платформы предоставляют современным менеджерам и сотрудникам возможность эффективно управлять задачами, контролировать сроки их выполнения и обеспечивать прозрачность процессов.

Целью дипломного проекта является разработка веб-приложения для управления проектами, которое позволит автоматизировать процессы планирования, распределения и контроля задач, повысить прозрачность выполнения проектов, а также обеспечить удобство для сотрудников и менеджеров в управлении задачами и получении отчетности. Проект направлен на создание системы, которая обеспечит авторизацию и регистрацию пользователей с разграничением ролей (менеджер и сотрудник), ведение журнала заявок, распределение задач, контроль жизненного цикла заказов, учет рабочего времени, а также формирование и экспорт отчетов.

Актуальность поставленной задачи обусловлена необходимостью повышения эффективности управления проектами в условиях роста конкуренции и усложнения рабочих процессов. Внедрение веб-приложения позволит сократить временные затраты на ручное распределение задач, минимизировать ошибки, связанные с человеческим фактором, и обеспечить точный контроль сроков выполнения проектов. Кроме того, предоставление менеджерам и сотрудникам удобного инструмента для работы с задачами в формате доски Канбан, а также гибкой системы отчетности, способствует повышению общей производительности компании.

Дипломный проект включает в себя исследование предметной области, проектирование базы данных и архитектуры веб-приложения, разработку клиентской и серверной частей, создание интуитивно понятного интерфейса с адаптивной версткой, реализацию функционала авторизации, регистрации, управления задачами и отчетности, а также проведение функционального и интеграционного тестирования. В результате будет создано веб-приложение, обеспечивающее личные кабинеты для сотрудников и менеджеров, возможности просмотра, поиска, фильтрации данных, управления задачами через доску Канбан, а также формирования и экспорта отчетов по установленным формам. Проект завершается анализом применения разработанного решения в реальных условиях и рекомендациями по его внедрению.

Решение задач изложено в пояснительной записке, которая включает шесть разделов с необходимой информацией об организации и использовании веб-приложения.

В первом разделе «Описание задачи» описывается анализ предметной области и постановка задачи. Здесь проводится детальное изучение основных аспектов сферы, в которой будет функционировать разрабатываемая система, а также определяются конкретные цели и задачи проекта.

Во втором разделе «Проектирование веб-приложения» будет осуществляться разработка архитектуры и интерфейса веб-приложения. Здесь будет определена структура базы данных, разработаны модели данных и описана логика взаимодействия между различными компонентами веб-приложения. Также будет создан пользовательский интерфейс, учитывающий требования удобства использования и эстетические аспекты.

В третьем разделе «Реализация веб-приложения» будет описан процесс создания и разработки самого веб-приложения на основе предварительно разработанного проекта. Здесь будет представлена информация о выбранных технологиях и инструментах разработки, методах программирования, а также о проведенных этапах разработки функционала и компонентов приложения.

В четвёртом разделе «Применение» описывается назначение данного веб-приложения в своей области.

Пятый раздел «Охрана труда и окружающей среды» содержит информацию о правовых, нормативных, социально-экономических и организационных вопросах охраны труда, о выполнении требований безопасности к рабочим местам, об обеспечении параметров микроклимата и чистоты воздушной среды, осветительных условий, защиты от шума, защиты от воздействия электромагнитных полей, электробезопасности, пожаробезопасности и охраны окружающей среды.

В шестом экономическом разделе приводится обоснование необходимости выведения программного средства на рынок, составляется план по разработке программного средства и рассчитываются затраты на его разработку, а также рассчитывается экономический эффект у разработчика и пользователя программного средства.

В приложении А представлен текст программы.

В приложении Б представлены макеты страниц веб-приложения.

В приложении В представлены логические и физические структуры веб-приложения как клиентской части, так и административной.

В приложении Г представлены результаты работы веб-приложения.

Графическая часть состоит из:

[– диаграмма вариантов использования](#_Toc168485202);

[– диаграмма сущность-связь;](#_Toc168485204)

[– диаграмма развертывания;](#_Toc168485204)

[– структура веб-приложения](#_Toc168485205).

**1 Описание задачи**

**1.1 Анализ предметной области**

Предметной областью решаемой задачи является управление проектами в компании ООО «ФП Трэйд» (бренд «CompleteSoft»), специализирующейся на разработке заказного программного обеспечения. В рамках деятельности компании осуществляется организация работы проектных команд, включающих менеджеров и сотрудников, которые совместно выполняют задачи, связанные с созданием и сопровождением программных продуктов, в частности, для финансовых и страховых систем. Основной целью является обеспечение эффективного управления проектами, минимизация временных затрат, повышение прозрачности выполнения задач и разработка удобных, адаптивных и эффективных решений с учетом современных требований.

Работа проектной команды начинается с получения заявок на проекты, которые формируются на основе требований заказчиков и спецификаций. Заявки включают перечень задач, необходимых для реализации, таких как разработка новых функциональных модулей, доработка существующих решений, устранение ошибок, а также адаптация программного обеспечения под различные платформы, устройства и браузеры. Менеджеры (администраторы) распределяют задачи между сотрудниками с учетом их квалификации, текущей нагрузки и норм времени на выполнение задач, а также контролируют сроки и качество их исполнения. При необходимости задачи могут быть уточнены или перенаправлены с участием аналитиков или других специалистов. Сотрудники, в свою очередь, выполняют задачи, обновляют их статусы и предоставляют отчетность о затраченном времени.

Для организации эффективного управления задачами компания использует методологию Agile с применением доски Канбан, которая визуализирует текущие задачи, их статусы и ответственных исполнителей.



Рисунок 1.1 – Пример доски Канбан

Доска Канбан представляет собой инструмент, на котором задачи отображаются в виде карточек, перемещаемых между статусами («Новая», «В работе», «Завершена», «Выполнена»). Это позволяет менеджерам и сотрудникам оперативно отслеживать прогресс выполнения задач, выявлять узкие места, своевременно реагировать на изменения и приоритизировать работу, обеспечивая высокую прозрачность и эффективность процессов.

Разработка веб-приложения для управления проектами обеспечит автоматизацию и оптимизацию процессов, предоставив следующие функциональные возможности:

* авторизацию и регистрацию пользователей с ролями «менеджер» и «сотрудник»;
* ведение журнала заявок на проекты и контроль их статусов;
* планирование и распределение задач между сотрудниками с учетом перечня работ и норм времени;
* мониторинг жизненного цикла выполнения заказов и контроль сроков выполнения задач;
* учет и контроль рабочего времени сотрудников;
* отображение задач в личном кабинете сотрудника в виде доски Канбан с возможностью изменения статусов через перетаскивание;
* просмотр, поиск, сортировку и фильтрацию данных;
* добавление, удаление и обновление хранимой информации;
* формирование и экспорт отчетов установленной формы, включая статистические данные: отчеты по количеству заявок на проекты в разрезе временного периода и типа проекта, по количеству задач в разрезе сотрудника и временного периода, а также по заявкам на проекты за определенный период.

Веб-приложение будет разработано с учетом требований к адаптивной верстке, интуитивно понятного интерфейса и корректного отображения в браузерах последних версий, что обеспечит удобство использования для всех участников проектной команды.

**1.2 Постановка задачи**

Исходя из анализа предметной области управления проектами, можно выделить следующие задачи, подлежащие автоматизации:

* регистрация и авторизация пользователей с разграничением ролей (менеджер и сотрудник);
* создание и ведение журнала заявок на проекты;
* просмотр, редактирование и удаление заявок на проекты;
* классификация заявок по типу проекта и временному периоду;
* распределение задач между сотрудниками с учетом их квалификации и норм времени;
* учет рабочего времени сотрудников, затраченного на выполнение задач;
* мониторинг жизненного цикла выполнения заказов и задач;
* формирование отчетов установленной формы, включая статистические данные (по количеству заявок на проекты в разрезе временного периода и типа проекта, по количеству задач в разрезе сотрудника и временного периода, по заявкам на проекты за определенный период);
* экспорт отчетов в удобный формат (например, CSV или PDF).

Веб-приложение автоматизирует процессы: для менеджеров — создание и управление журналом заявок через личный кабинет, распределение задач с учетом нагрузки и квалификации сотрудников, контроль сроков выполнения, мониторинг жизненного цикла заказов, формирование и экспорт отчетов; для сотрудников — просмотр задач в виде доски Канбан с разбивкой по статусам («Новая», «В работе», «Завершена», «Выполнена»), изменение статусов перетаскиванием, учет рабочего времени, поиск, сортировка и фильтрация задач.

Общие преимущества: сокращение времени на ручное распределение и контроль задач, уменьшение ошибок, упрощение взаимодействия между менеджерами и сотрудниками через централизованный доступ к данным, гибкость при масштабировании системы.

В настоящее время управление проектами в компании осуществляется с использованием методологии Agile и доски Канбан, реализованной через инструменты, такие как Trello или Jira. Менеджеры вручную распределяют задачи и составляют отчеты, что требует значительных временных затрат и может приводить к ошибкам. Сотрудники используют доску Канбан для визуализации задач, но отсутствие автоматизации учета времени и отчетности снижает эффективность.

Разрабатываемое веб-приложение учитывает специфику компании, предоставляя интуитивно понятный интерфейс для ролей менеджера и сотрудника, а также гибкость в формировании и экспорте отчетов. В отличие от аналогов, таких как Trello и Jira, приложение включает учет рабочего времени, мониторинг жизненного цикла заказов и изменение статусов задач через перетаскивание, что делает его более удобным и эффективным для проектных команд.

**2 Проектирование веб-приложения**

**2.1 Проектирование модели**

Проектирование модели – ключевой этап разработки веб-приложения, на котором создается абстрактное представление его функциональности, архитектуры и взаимодействия компонентов. Этот процесс включает анализ требований, определение ролей пользователей, разработку логики взаимодействия и моделирование структуры данных. Грамотное проектирование позволяет минимизировать ошибки на этапе реализации, повысить удобство использования и обеспечить масштабируемость системы.

Проектирование веб-приложения для управления проектами включает комплексное моделирование с использованием унифицированного языка моделирования UML. Диаграмма вариантов использования, представленная в графической части на листе 1, отражает взаимодействие основных акторов системы: менеджера, который управляет заявками на проекты, распределяет задачи, контролирует их выполнение и формирует отчеты; сотрудника, который просматривает свои задачи, изменяет их статусы через доску Канбан и фиксирует рабочее время; а также системы, отвечающей за автоматическое обновление статусов, расчет времени и уведомления о дедлайнах. Диаграмма иллюстрирует основные действия пользователей: менеджер может авторизоваться, создавать и редактировать заявки на проекты, распределять задачи, контролировать сроки и формировать отчеты; сотрудник может авторизоваться, просматривать задачи, изменять их статусы перетаскиванием, вносить данные о затраченном времени и просматривать статистику. Действия представлены в виде кейсов, включающих взаимосвязанные подзадачи, например, функция «Просмотр задач» для сотрудника включает «Изменение статуса задачи» и «Учет рабочего времени». Функции «Регистрация» и «Авторизация» являются обязательными для доступа к остальным операциям.

Концептуальная модель «Сущность-связь», представленная в графической части на листе 2, описывает основные сущности системы и их взаимосвязи:

* пользователь (Users) с атрибутами ID\_Users, Full\_Name, Email, Phone\_Number, ID\_Position, Password, ID\_Role, IsAdmin;
* роль (Roles) с атрибутами ID\_Role, Role\_Name;
* должность (Positions) с атрибутами ID\_Position, Position\_Name;
* заказ (Orders) с атрибутами ID\_Order, Order\_Name, Description, Status, Start\_Date, End\_Date, ID\_Customer;
* статус (Statuses) с атрибутами Status, Status\_Name;
* задача (Tasks) с атрибутами ID\_Task, ID\_Order, Task\_Name, Description, Status, Start\_Date, End\_Date, ID\_Employee;
* журнал рабочего времени (WorkLogs) с атрибутами ID\_WorkLog, ID\_Task, ID\_Employee, Hours\_Worked, Log\_Date;
* история изменения статусов (TaskStatusHistory) с атрибутами ID\_StatusHistory, ID\_Task, Old\_Status, New\_Status, Change\_Date.  
  Сущности связаны отношениями: один пользователь может быть ответственным за множество заказов и задач, один заказ включает множество задач, каждая задача имеет один статус и может быть связана с несколькими записями в журнале рабочего времени и истории изменения статусов.

Логическая модель базы данных, соответствующая третьей нормальной форме, реализована в виде таблиц:

* Users (ID\_Users, Full\_Name, Email, Phone\_Number, ID\_Position, Password, ID\_Role, IsAdmin);
* Roles (ID\_Role, Role\_Name);
* Positions (ID\_Position, Position\_Name);
* Orders (ID\_Order, Order\_Name, Description, Status, Start\_Date, End\_Date, ID\_Customer);
* Statuses (Status, Status\_Name);
* Tasks (ID\_Task, ID\_Order, Task\_Name, Description, Status, Start\_Date, End\_Date, ID\_Employee);
* WorkLogs (ID\_WorkLog, ID\_Task, ID\_Employee, Hours\_Worked, Log\_Date);
* TaskStatusHistory (ID\_StatusHistory, ID\_Task, Old\_Status, New\_Status, Change\_Date).

Все таблицы связаны внешними ключами, обеспечивающими целостность данных. Например, поле ID\_Position в таблице Users ссылается на таблицу Positions, а поле Status в таблице Tasks – на таблицу Statuses.

Диаграмма развертывания, представленная в графической части на листе 3, описывает физическую архитектуру системы: клиентская часть (браузер пользователя с интерфейсом на React), серверная часть (Node.js с Express для обработки запросов), база данных (Microsoft SQL Server для хранения данных) и их взаимодействие через HTTP-запросы. Клиентская часть отправляет запросы к серверу, сервер обрабатывает их, взаимодействуя с базой данных, и возвращает ответы для отображения в интерфейсе.

Диаграмма классов, отражающая структуру данных и методы, может быть дополнительно разработана на основе логической модели, включая классы для каждой сущности (например, User, Order, Task) с соответствующими атрибутами и методами для управления данными. Однако в данном проекте акцент сделан на диаграммы вариантов использования, «Сущность-связь» и развертывания, как указано в требованиях.

Структура веб-приложения включает клиентскую часть (React для интерфейса с доской Канбан, формами и отчетами), серверную часть (Express для API-эндпоинтов регистрации, авторизации, управления задачами и отчетами) и базу данных (SQL Server для хранения информации). Компоненты взаимодействуют следующим образом: пользователь через браузер отправляет запросы (например, на авторизацию или создание задачи), сервер обрабатывает их, выполняя операции с базой данных, и возвращает данные для отображения в интерфейсе.

**2.2 Требование к веб-приложению**

Для обеспечения удобства, эффективности и доступности веб-приложения для управления проектами определены следующие требования, которые направлены на создание интуитивно понятного и функционального интерфейса.

Веб-приложение будет выполнено в современном минималистичном стиле, чтобы минимизировать отвлекающие элементы и сосредоточить внимание пользователей на ключевых функциях. Такой подход обоснован необходимостью упрощения взаимодействия с системой, особенно для пользователей с разным уровнем цифровой грамотности, и позволяет сократить время на освоение интерфейса.

Графический дизайн будет ориентирован на функциональность и минимализм. Использование иконок и графических элементов будет лаконичным, интуитивно понятным и направленным на упрощение восприятия информации. Например, статусы задач на доске Канбан будут выделены цветами (зеленый для «Выполнена», желтый для «В работе»), что позволит пользователям быстро оценивать состояние задач. Такой подход обоснован необходимостью ускорения работы пользователей и снижения когнитивной нагрузки.

Для обеспечения высокой читаемости и эстетической гармонии основным шрифтом выбран «Roboto» – универсальный шрифт, широко используемый в веб-интерфейсах благодаря своей четкости и поддержке различных устройств. Размеры шрифтов определены следующим образом: заголовки – 22 px для выделения основных разделов, подзаголовки – 18 px для обозначения подразделов, основной текст – 14 px для комфортного чтения. Выбор шрифта и размеров обоснован необходимостью обеспечения читаемости на устройствах с разными экранами и плотностью пикселей.

Приложение должно корректно отображаться и стабильно работать в современных браузерах, таких как Google Chrome, Mozilla Firefox, Safari и Microsoft Edge (последние версии и версии за последние два года). Также требуется обеспечить адаптивность интерфейса для корректного отображения на устройствах с различными разрешениями экранов: мобильных телефонах (от 320 px), планшетах (от 768 px) и настольных компьютерах (от 1024 px). Эти требования обоснованы необходимостью охвата широкой аудитории пользователей и обеспечения доступности приложения в разных условиях.

Контент страниц будет структурированным, лаконичным и понятным, с использованием простого и доступного языка без сложной терминологии. Например, вместо «инициировать жизненный цикл заказа» будет использовано «начать выполнение заказа». Это позволит сделать интерфейс понятным для пользователей с разным уровнем подготовки. Обоснование: минимизация времени на восприятие информации и снижение барьеров для новых пользователей.

Система управления контентом должна обеспечивать удобное добавление, редактирование и удаление данных (заявок, задач, отчетов) через интерфейс менеджера и сотрудника. Для этого предусмотрены формы с валидацией (например, проверка формата email при регистрации) и функции массового управления данными (например, удаление нескольких задач одновременно). Также система должна поддерживать автоматическое обновление данных в реальном времени, например, при изменении статуса задачи на доске Канбан. Эти требования обоснованы необходимостью упрощения управления данными и повышения оперативности работы пользователей.

Страницы приложения будут выполнены в едином стиле для всех разделов, чтобы обеспечить визуальную целостность. Элементы интерфейса выровнены по единой сетке для создания упорядоченного вида. Основные блоки информации (заявки, задачи, отчеты) четко структурированы и визуально отделены друг от друга с помощью отступов и цветовых акцентов. Навигация реализована в виде бокового или верхнего меню, обеспечивающего быстрый доступ к основным разделам (журнал заявок, доска Канбан, отчеты). Кнопки и интерактивные элементы имеют четкие границы, выделены цветом (например, синий для действий, красный для удаления) и обеспечивают визуальную обратную связь при наведении. Эти требования обоснованы необходимостью создания интуитивно понятного интерфейса, снижения времени на поиск нужных функций и повышения удобства работы.

**2.3 Структура веб-приложения**

Структура приложения представляет собой организацию его компонентов, модулей и элементов, обеспечивающую удобное и эффективное взаимодействие пользователей с системой. Четкая структура позволяет легко ориентироваться в разделах и функциях приложения, упрощая навигацию и повышая удобство использования.

Первоначальная информационная структура веб-приложения для управления проектами включает: главную страницу с формами авторизации и регистрации, личный кабинет менеджера с возможностями ведения журнала заявок на проекты (создание, редактирование, удаление заявок), распределения задач между сотрудниками с учетом перечня работ и норм времени, контроля жизненного цикла выполнения заказов, мониторинга сроков выполнения заказов и задач, формирования отчетов (статистический отчет по количеству заявок в разрезе временного периода и типа проекта, статистический отчет по количеству задач в разрезе сотрудника и временного периода, отчет по заявкам на проекты за определенный период) и их экспорта; личный кабинет сотрудника с просмотром задач в виде доски Канбан с разбивкой по статусам («Новая», «В работе», «Завершена», «Выполнена»), изменением статусов задач через перетаскивание, учетом рабочего времени (ввод данных о затраченных часах), просмотром уведомлений о дедлайнах; общие компоненты, включающие систему навигации, функции поиска, сортировки и фильтрации информации, автоматическое формирование уведомлений и отчетов.

Структура веб-приложения представлена в виде ментальной карты в графической части на листе 4.

**2.4 Проектирование макета веб-приложения**

Для разработки графических представлений макетов страниц и форм веб-приложения для управления проектами используется онлайн-сервис Figma [19], предназначенный для проектирования интерфейсов и прототипирования. Выбор Figma обоснован его широкими возможностями для создания детализированных прототипов, удобным интерфейсом и поддержкой совместной работы, что позволяет разработчикам и заказчику эффективно взаимодействовать на этапе проектирования.

Тип дизайна веб-приложения – минималистичный, что обусловлено необходимостью обеспечения удобства и скорости взаимодействия пользователей с системой. Минимализм позволяет сосредоточить внимание пользователей на ключевых функциях, таких как управление задачами и заявками, минимизируя отвлекающие элементы. Основная цветовая палитра включает три цвета: белый (фон), черный (основной текст) и синий (для интерактивных элементов, таких как кнопки и ссылки), что обеспечивает визуальную ясность и контрастность. Шрифт по умолчанию – Roboto, выбранный за его высокую читаемость и универсальность для веб-интерфейсов.

Макеты страниц и форм разработаны с учетом минималистического дизайна и включают элементы, обеспечивающие удобное взаимодействие пользователей с приложением. Ниже описаны основные страницы и их структура:

* главная страница содержит форму авторизации и регистрации с полями для ввода логина и пароля, а также краткое описание приложения и его возможностей (например, «Управляйте проектами эффективно с помощью доски Канбан»). Макет главной страницы представлен в приложении Б на рисунке Б.1;
* личный кабинет менеджера включает навигационную панель с разделами: журнал заявок, распределение задач, контроль сроков и отчеты. Основной блок отображает список заявок на проекты с возможностью их создания, редактирования и удаления, а также форму распределения задач с выбором сотрудника, типа задачи и сроков выполнения. Макет личного кабинета менеджера представлен в приложении Б на рисунке Б.2;
* личный кабинет сотрудника содержит доску Канбан с задачами, разделенными по статусам («Новая», «В работе», «Завершена», «Выполнена»), с возможностью перетаскивания карточек задач для изменения статуса. Также предусмотрен блок учета рабочего времени с формой для ввода затраченных часов и уведомления о дедлайнах. Макет личного кабинета сотрудника представлен в приложении Б на рисунке Б.3;
* страница заявок включает два блока: блок фильтрации (по типу проекта, временному периоду, статусу) и список заявок с информацией о названии проекта, дате создания, статусе и ответственном менеджере. Менеджер может редактировать, удалять или добавлять новые заявки. Макет страницы заявок представлен в приложении Б на рисунке Б.4;
* страница отчетов предоставляет менеджеру возможность выбора типа отчета (статистический по заявкам, по задачам сотрудника, по временному периоду) с функцией экспорта в CSV или PDF. Отчеты отображаются в виде таблиц с возможностью фильтрации и сортировки данных. Макет страницы отчетов представлен в приложении Б на рисунке Б.5.

Цель создания макетов – визуализация интерфейса и его элементов, чтобы заказчик и разработчики могли оценить внешний вид, функциональность и удобство использования, а также внести корректировки до начала реализации. Все макеты разработаны с учетом адаптивной верстки, чтобы обеспечить корректное отображение в браузерах последних версий на устройствах с разными разрешениями экранов.

**2.5 Программно-технические средства, необходимые для разработки приложения**

Разработка веб-приложения для управления проектами требует использования современных технологий и инструментов, обеспечивающих высокую производительность, безопасность, масштабируемость и удобство работы. Выбор программно-технических средств обоснован требованиями проекта, включая поддержку многопользовательского взаимодействия, хранение данных, реализацию доски Канбан, управление задачами и формирование отчетов с возможностью экспорта.

Для разработки выбрана среда Visual Studio Code (VS Code), которая является универсальным и легковесным редактором кода с поддержкой множества расширений, таких как Prettier (для форматирования кода), ESLint (для анализа JavaScript/TypeScript), GitLens (для работы с Git) и Live Server (для локального тестирования). VS Code обеспечивает удобную работу с клиентской и серверной частями приложения, поддерживает автодополнение кода и отладку, что ускоряет процесс разработки. Выбор VS Code обоснован его кроссплатформенностью, широкой поддержкой сообщества и совместимостью с используемыми технологиями, такими как React, Node.js и TypeScript.

Разработка веб-приложения осуществляется на основе клиент-серверной архитектуры, где фронтенд отвечает за отображение интерфейса, а бэкенд – за обработку данных и бизнес-логику. Используемые технологии и фреймворки выбраны с учетом требований к адаптивности, интерактивности и масштабируемости приложения.

**Клиентская часть (фронтенд)**:

* HTML: используется для структурирования содержимого страниц, обеспечивая семантическую разметку (например, теги <header>, <nav>, <main> для навигации и контента). HTML выбран за его универсальность и поддержку всеми браузерами;
* CSS: применяется для стилизации интерфейса, включая адаптивную верстку с использованием медиа-запросов для корректного отображения на устройствах с разными разрешениями экранов. CSS обеспечивает гибкость в создании минималистичного дизайна с доской Канбан и отчетами;
* JavaScript: отвечает за динамическое обновление интерфейса, например, перетаскивание задач на доске Канбан (drag-and-drop) и фильтрацию данных в реальном времени. JavaScript выбран за его широкую поддержку и возможность создания интерактивных приложений;
* React.js: фреймворк для создания компонентного интерфейса. React позволяет разбить интерфейс на переиспользуемые компоненты (например, компонент доски Канбан, карточки задачи, формы отчетов), упрощает управление состоянием с помощью хуков (useState, useEffect) и обеспечивает высокую производительность благодаря виртуальному DOM. React выбран за его популярность, активное сообщество и поддержку масштабируемости [12];
* TypeScript: используется для типизации кода, что повышает безопасность разработки, предотвращает ошибки на этапе компиляции и упрощает поддержку кода. TypeScript выбран для улучшения читаемости и масштабируемости проекта [13].

**Серверная часть (бэкенд):**

* Node.js: среда выполнения JavaScript для серверной части. Node.js обеспечивает асинхронную обработку запросов, что важно для многопользовательского приложения с частыми операциями (например, обновление статуса задачи). Выбран за высокую производительность и совместимость с JavaScript [14];
* Express.js: фреймворк для Node.js, используемый для создания REST API. Express упрощает маршрутизацию (например, /api/tasks для получения списка задач), обработку HTTP-запросов и middleware для аутентификации (например, JWT). Выбран за минималистичность, гибкость и широкую поддержку сообщества [15].

**Для хранения данных используется** Microsoft SQL Server (SSMS) [16]: реляционная база данных для хранения информации о пользователях, заявках, задачах, рабочем времени и истории статусов. SQL Server поддерживает третью нормальную форму (3НФ), обеспечивает высокую скорость обработки запросов благодаря индексам и транзакциям, а также гарантирует целостность данных через внешние ключи. Выбран за надежность, поддержку сложных запросов и совместимость с серверной частью.

**Контроль версий и тестирование**:

* Git и GitHub: Git используется для управления версиями кода, а GitHub – для хранения репозитория и совместной работы. Выбраны за их стандартизацию и широкое распространение [17];
* Postman: инструмент для тестирования API, позволяющий проверять эндпоинты (например, регистрация, получение задач, экспорт отчетов). Выбран за удобство и поддержку автоматизации тестов [18].

Таким образом, выбранный стек технологий и инструментов обеспечивает создание интерактивного, надежного и масштабируемого веб-приложения, соответствующего требованиям к управлению проектами, адаптивности и кроссбраузерности.

**2.6 Защита и сохранность данных**

Разработка веб-приложения для управления проектами требует обеспечения безопасности данных, включая защиту от несанкционированного доступа, сохранность информации и надежность хранения. Особое внимание уделено ограничению доступа, защите данных и механизмам их сохранности.

Для обеспечения безопасности данных реализована система ролевого доступа, которая ограничивает возможности пользователей в зависимости от их роли:

* менеджер (администратор) имеет полный доступ к управлению заявками на проекты, распределению задач, контролю сроков выполнения, формированию и экспорту отчетов, а также редактированию данных сотрудников;
* сотрудник может просматривать только назначенные ему задачи на доске Канбан, изменять их статусы, вносить данные о затраченном времени и фильтровать свои задачи, но не имеет доступа к данным других сотрудников или управлению заявками.

Ограничения доступа реализованы на уровне API (например, через middleware в Express.js) и базы данных (с использованием SQL Server ролей и прав доступа), что предотвращает выполнение операций, выходящих за пределы полномочий пользователя.

Для защиты от несанкционированного использования применена многоуровневая система аутентификации и авторизации:

* вход в личный кабинет осуществляется с использованием логина (email) и пароля, который хешируется с помощью алгоритма bcrypt перед сохранением в базе данных;
* после входа генерируется JWT-токен (JSON Web Token), который используется для аутентификации каждого запроса к API. Токен имеет ограниченный срок действия (например, 1 час) и обновляется при необходимости;
* все данные, передаваемые между клиентом и сервером, шифруются с использованием HTTPS (протокол TLS/SSL), что защищает информацию от перехвата.

Для обеспечения сохранности и целостности данных Все данные хранятся в базе данных в СУБД Microsoft SQL Server с использованием SQL, что обеспечивает их сохранность и целостность. Применены механизмы резервного копирования, мониторинга целостности, разграничения прав доступа и защиты от SQL-инъекций.

Эти меры обеспечивают надежную защиту данных, минимизируют риски утечки информации и гарантируют возможность восстановления данных в случае сбоев, что соответствует требованиям проекта по обеспечению сохранности данных.

**2.7 Организация и ведение информационной модели**

Веб-приложение для управления проектами использует реляционную базу данных, реализованную в Microsoft SQL Server, для хранения данных о пользователях, заявках, задачах, статусах, рабочем времени и истории изменений. База данных спроектирована в соответствии с третьей нормальной формой (3НФ), что обеспечивает минимизацию избыточности данных и целостность информации.

База данных состоит из следующих таблиц, отражающих информационные сущности системы.

Таблица «Roles» хранит роли пользователей для разграничения доступа. Структура представлена в таблице 2.1.

Таблица 2.1 – Структура таблицы «Roles»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID\_Role | INT | 4 | Уникальный идентификатор роли |
| Role\_Name | NVARCHAR(50) | 100 | Название роли (например, «Менеджер», «Сотрудник») |

Таблица «Positions» хранит список должностей сотрудников. Структура представлена в таблице 2.2.

Таблица 2.2 – Структура таблицы «Positions»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID\_Position | INT | 4 | Уникальный идентификатор должности |
| Position\_Name | NVARCHAR(100) | 200 | Название должности (например, «Разработчик ПО») |

Таблица «Users» хранит информацию о сотрудниках компании. Структура таблицы представлена в таблице 2.3.

Таблица 2.3 – Структура таблицы «Users»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID\_Users | INT | 4 | Уникальный идентификатор пользователя |
| Full\_Name | NVARCHAR(100) | 200 | Полное имя пользователя |
| Email | NVARCHAR(100) | 200 | Адрес электронной почты (уникальный, с проверкой на @gmail.com) |
| Phone\_Number | NVARCHAR(20) | 40 | Номер телефона |
| ID\_Position | INT | 4 | Ссылка на должность |
| Password | NVARCHAR(255) | 510 | Хешированный пароль |
| ID\_Role | INT | 4 | Ссылка на роль |
| IsAdmin | BIT | 1 | Флаг администратора (1 — да, 0 — нет) |

Таблица «Statuses» содержит перечень этапов выполнения задач. Структура таблицы представлена в таблице 2.4.

Таблица 2.4 – Структура таблицы «Statuses»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Status | NVARCHAR(50) | 100 | Уникальный идентификатор статуса (например, «Новая») |
| Status\_Name | NVARCHAR(100) | 200 | Название статуса |

Таблица «Orders» хранит информацию о задачах и заказах. Структура таблицы представлена в таблице 2.5.

Таблица 2.5 – Структура таблицы «Orders»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID\_Order | INT | 4 | Уникальный идентификатор заявки |
| Order\_Name | NVARCHAR(100) | 200 | Название заявки |
| Description | NVARCHAR(255) | 510 | Описание заявки |
| Status | NVARCHAR(50) | 100 | Ссылка на статус |
| Start\_Date | DATE | 3 | Дата начала |
| End\_Date | DATE | 3 | Дата завершения |
| ID\_Customer | INT | 4 | Ссылка на менеджера, создавшего заявку |

Таблица «Tasks» Хранит информацию о задачах. Структура представлена в таблице 2.6.

Таблица 2.6 – Структура таблицы «Tasks»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID\_Task | INT | 4 | Уникальный идентификатор задачи |
| ID\_Order | INT | 4 | Ссылка на заявку |
| Task\_Name | NVARCHAR(100) | 200 | Название задачи |
| Description | NVARCHAR(255) | 510 | Описание задачи |
| Status | NVARCHAR(50) | 100 | Ссылка на статус |
| Start\_Date | DATE | 3 | Дата начала |
| End\_Date | DATE | 3 | Дата завершения |
| ID\_Employee | INT | 4 | Ссылка на сотрудника |

Таблица «WorkLogs» хранит данные о затраченном рабочем времени. Структура представлена в таблице 2.7.

Таблица 2.7 – Структура таблицы «WorkLogs»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID\_WorkLog | INT | 4 | Уникальный идентификатор записи |
| ID\_Task | INT | 4 | Ссылка на задачу |
| ID\_Employee | INT | 4 | Ссылка на сотрудника |
| Hours\_Worked | DECIMAL(5,2) | 5 | Количество затраченных часов |
| Log\_Date | DATE | 3 | Дата записи |

Таблица «TaskStatusHistory» хранит историю изменения статусов задач. Структура представлена в таблице 2.8.

Таблица 2.8 – Структура таблицы «TaskStatusHistory»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID\_StatusHistory | INT | 4 | Уникальный идентификатор записи |
| ID\_Task | INT | 4 | Ссылка на задачу |
| Old\_Status | NVARCHAR(50) | 100 | Предыдущий статус |
| New\_Status | NVARCHAR(50) | 100 | Новый статус |
| Change\_Date | DATETIME | 8 | Дата и время изменения |

Модели данных делятся на логические и физические. Логическая модель определяет структуру данных независимо от их физического хранения, включая сущности, атрибуты и отношения.

Физическая модель базы данных разрабатываемого веб-приложения представлена на рисунке 2.1

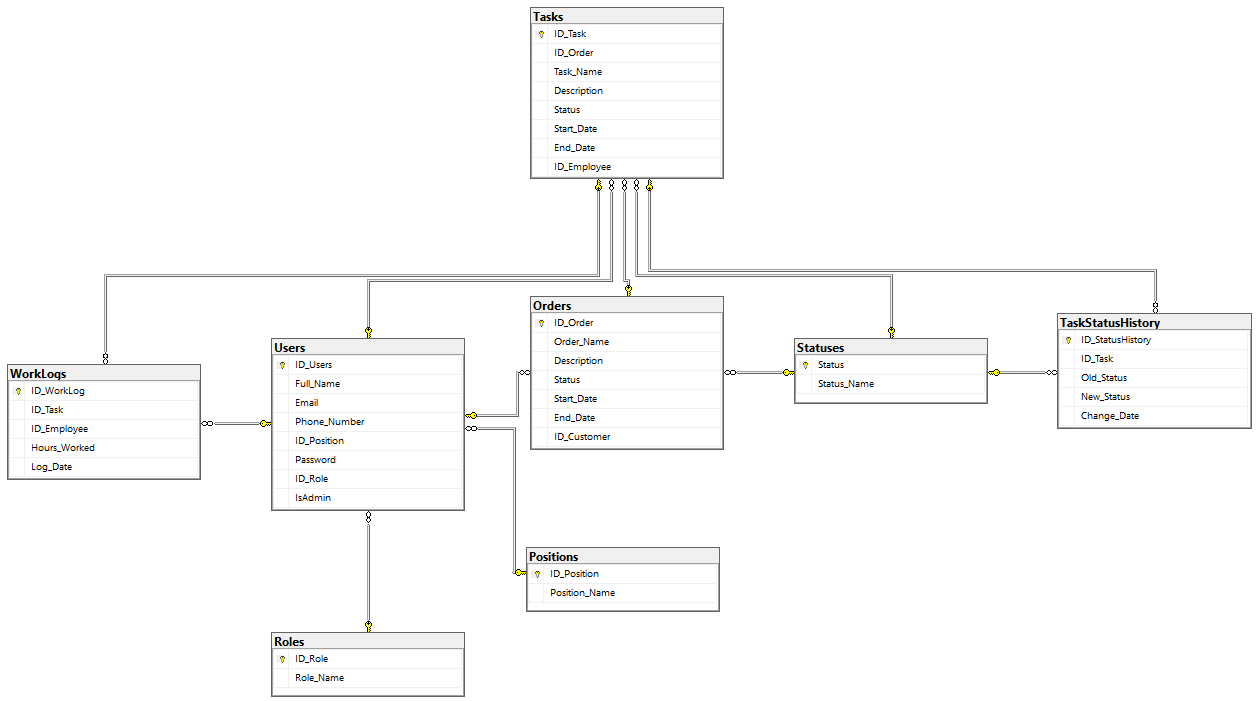


Рисунок 2.1 – Физическая модель данных

**3 Реализация веб-приложения**

**3.1 Описание разделов веб-приложения**

На главной странице представлено навигационное меню веб-приложения и форма для подачи заявки о поломке, которая едина для всех пользователей вне зависимости от их статуса. Форма включает в себя следующие поля:

* фамилия, имя, отчество (ФИО) отправителя;
* электронная почта;
* номер кабинета;
* категория заявки;
* описание.

После заполнения формы пользователь может отправить заявку для обработки.

Навигационное меню – это область интерфейса, на которой представлены навигационные опции, которые позволяют пользователям находить контент веб-приложения.

Навигационное меню предоставляет удобный доступ к основным разделам приложения в зависимости от статуса пользователя, обеспечивая ему возможность подачи заявки, управления своими данными и административными функциями (для администраторов).

Навигационное меню веб-приложения для авторизированных пользователей представлено на рисунке 9.



Рисунок 9 – Навигационное меню веб-приложения для авторизированных пользователей

Описание навигационного меню на главной странице представлено ниже:

* «Заявки» – список всех созданных сотрудниками заявок, показывает текущий статус каждой заявки (доступно только для авторизированных пользователей);
* «Профиль» – возможность выйти из системы, перейти на страницу заявок (доступно только для авторизированных пользователей), перейти в панель администратора (доступно только для суперпользователя);
* «Персонал» – возможность добавлять, просматривать и редактировать информацию о персонале (доступно только для суперпользователя)
* «Авторизация» – предоставляет возможность войти в систему, если они уже зарегистрированы (единственная доступная страница для неавторизированных пользователей, при авторизации она пропадает).

**3.2 Разработка административной части приложения**

Логическая карта – концептуальная модель системы, которая описывает ее структуру и основные компоненты на абстрактном уровне. Это высокоуровневое представление системы, независимое от конкретных технологий и деталей реализации. Логическая карта обычно включает в себя описание ключевых компонентов системы, их взаимосвязей и основных функциональных возможностей.

В файле «models.py» определены классы, которые доступны администратору:

* «Staff» – хранит информацию о сотрудниках в хозяйственной службе;
* «Material» – содержит информацию о всех материалах в базе данных;
* «Breaking» – модель заявки, содержащая информацию о отправленных заявках;
* «BreakingMaterial» – материалы, привязанные к заявке.

Веб-страницы, которые доступны администратору:

* «Index.html» – отображает главную страницу веб-приложения;
* «Journal.html» – показывает список всех заявок на поломки, их статусы, привязанные материалы;
* «Edit.html» – редактирование информации о поломке и сотруднике;
* «Material-breaking» – добавление материала к заявке;
* «Staff.html» – показывает список персонала и их данные, а также предоставляет возможность добавлять новых сотрудников;
* «Sign\_in.html» – авторизация для пользователей;
* «Profile.html» – профиль авторизированного пользователя;
* «Report.html» –отчеты на основе данных за заданный временной период.

Физическая структура – реальные компоненты и артефакты системы, такие как файлы, папки, базы данных. Это низкоуровневое представление системы, которое учитывает конкретные технологии, инфраструктуру и детали реализации. Физическая структура определяет, как компоненты системы фактически реализуются и взаимодействуют между собой в реальной среде.

Физическая структура для администратора веб-приложения представлена на рисунке 10.

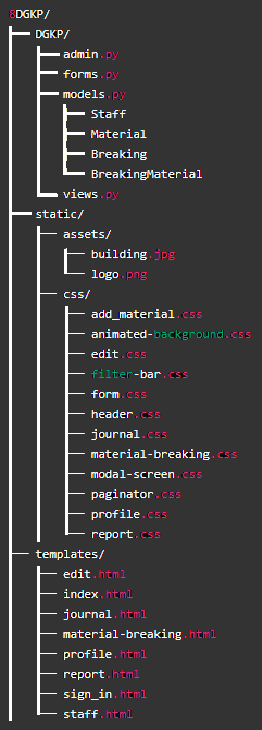


Рисунок 10 – Физическая структура для администратора веб-приложения

Архитектура развертывания административной части представлена в графической части на листе 3.

**3.3 Разработка клиентской части приложения**

Логическая карта клиентской части включает в себя описание ключевых компонентов системы, их взаимосвязей и основных функциональных возможностей.

В файле «models.py» определены классы, которые доступны клиенту веб-приложения:

* «Material» – содержит информацию о всех материалах в базе данных;
* «Breaking» – модель заявки, содержащая информацию о отправленных заявках;
* «BreakingMaterial» – материалы, привязанные к заявке.

Веб-страницы, которые доступны клиенту:

* «Index.html» – отображает главную страницу веб-приложения;
* «Journal.html» – показывает список заявок на группу пользователя;
* «Edit.html» – редактирование информации о поломке;
* «Material-breaking» – добавление материала к заявке;
* «Sign\_in.html» – авторизация для пользователей;
* «Profile.html» – профиль авторизированного пользователя.

Физическая структура для пользователя веб-приложения представлена на рисунке 11.

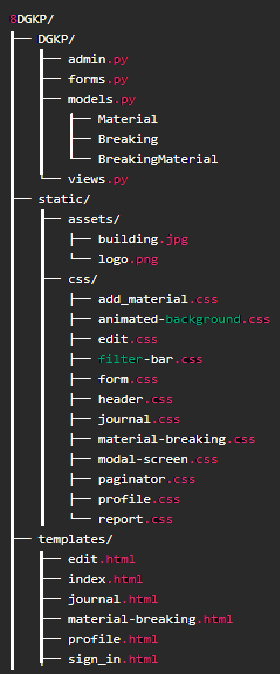


Рисунок 11 – Физическая структура для пользователя веб-приложения

Архитектура развертывания клиентской части представлена в графической части на листе 3.

**3.4 Описание используемых функций и процедур**

Функция отправки заявки «Send\_email» используется для отправки электронного уведомления пользователю о том, что его заявка успешно зарегистрирована. Она принимает в качестве параметров тему письма, текст сообщения и адрес получателя, использует параметры подключения к SMTP-серверу из настроек приложения и отправляет электронное письмо с информацией о заявке на указанный адрес получателя. При успешной отправке отображает пользователю сообщение об успешной регистрации заявки. В случае возникновения ошибки отправки выводит сообщение об ошибке. Функция вызывается при нажатии на button «Отправить» в форме на главной странице веб-приложения.

Код функции «Send\_email» представлен ниже

def send\_email(email\_subject, email\_message, email\_recipient):  
 with get\_connection( #устанвка соеднинения  
 host=settings.EMAIL\_HOST, #указание хоста  
 port=settings.EMAIL\_PORT, #указание порта  
 username=settings.EMAIL\_HOST\_USER, #SMTP логин сервера  
 password=settings.EMAIL\_HOST\_PASSWORD, #SMTP пароль сервера  
 use\_tls=settings.EMAIL\_USE\_TLS  
 ) as connection:  
 email\_from = settings.EMAIL\_HOST\_USER  
 EmailMessage(email\_subject, email\_message, email\_from, email\_recipient, connection=connection).send() #прием и отпрака данных

Функция добавления новой заявки «Index» предназначена для обработки запросов на добавление новой заявки. Она получает данные из формы, введенные пользователем, проверяет валидность полученных данных. В случае успешной валидации сохраняет данные заявки в базе данных. Отправляет электронное уведомление о регистрации заявки. Функция вызывается сразу после загрузки главной страницы веб-приложения.

Код функции «Index» представлен ниже

def index(request):  
 notification = None  
 form\_errors = []  
  
 if request.method == "POST":  
 form = Breaking\_Form(request.POST) #создание новой формы  
 if form.is\_valid(): #если форма прошла валидацию  
 try: #сбор данных для последующей отправки  
 saved\_form = form.save()  
 email\_subject = f'Уважаемый(ая) {saved\_form.breaking\_full\_name}, Ваша заявка поступила в обработку.'  
 email\_message = 'Указанные данные:\n\n'  
 email\_message += (f'Отправитель: {saved\_form.breaking\_full\_name};\n'  
 f'Дата заявки: {saved\_form.breaking\_date};\n'  
 f'Кабинет: {saved\_form.breaking\_room};\n'  
 f'Категория: {saved\_form.breaking\_category};\n'  
 f'Описание: {saved\_form.breaking\_description};\n'  
 f'\n\nС уважением 8DGKP Manager!')  
 email\_recipient = [saved\_form.breaking\_email]  
 send\_email(email\_subject, email\_message, email\_recipient)  
 messages.success(request, 'Заявка успешно оставлена!')  
 form = Breaking\_Form()  
 except Exception as e: #ошибка в случае отправки  
 messages.error(request, f'Ошибка сохранения формы или отправки электронной почты: {e}')  
 else:  
 form\_errors = [f'{form.fields[field].label}: {error}' for field, error in form.errors.items()]  
 else:  
 form = Breaking\_Form()  
  
 context = { #данные для передачи в HTML страницу  
 'notification': notification,  
 'form': form,  
 'form\_errors': form\_errors,  
 }  
 return render(request, 'index.html', context)

Функция редактирования заявки «Edit\_breaking» предоставляет пользователю возможность редактировать данные о существующей заявке. Она получает данные выбранной заявки для отображения в форме редактирования, обрабатывает отправку формы с отредактированными данными. После успешного обновления данных выводит сообщение об успешном выполнении операции. Если заявка выполнена и указана электронная почта, то отправляется электронное уведомление о завершении заявки. Функция вызывается при нажатии на button «Редактировать» на странице «Заявки».

Код функции «Edit\_breaking» представлен ниже

def edit\_breaking(request, breaking\_id):  
 breaking = get\_object\_or\_404(Breaking, id=breaking\_id)#получаем текущую заявку  
 if request.method == "POST":  
 form = Breaking\_Form\_Admin(request.POST, instance=breaking)# создание формы с данными из бд  
 if form.is\_valid(): #если форма проходит валидацию  
 form.save()  
 messages.success(request, 'Заявка успешно обновлена!')  
 email\_breaking\_full\_name = form.cleaned\_data['breaking\_full\_name']  
 email\_breaking\_date = form.cleaned\_data['breaking\_date']  
 email\_breaking\_room = form.cleaned\_data['breaking\_room']  
 email\_breaking\_category = form.cleaned\_data['breaking\_category']  
 email\_breaking\_description = form.cleaned\_data['breaking\_description']  
 email\_breaking\_recipient = form.cleaned\_data['breaking\_email']  
 if 'send\_email\_button' in request.POST:  
 email\_subject = f'Уважаемый(ая) {email\_breaking\_full\_name}, Заявка {breaking.breaking\_number} выполнена!'  
 email\_message = 'Указанные данные:\n\n'  
 email\_message += (f'Отправитель: {email\_breaking\_full\_name};\n'  
 f'Дата заявки: {email\_breaking\_date};\n'  
 f'Кабинет: {email\_breaking\_room};\n'  
 f'Категория: {email\_breaking\_category};\n'  
 f'Описание: {email\_breaking\_description};\n'  
 f'\n\nСпасибо за вашу помощь! =)\n'  
 f'С уважением 8DGKP Manager!')  
 email\_recipient = [email\_breaking\_recipient]  
 try: #все соотвествует и идет попытка отправки письма  
 send\_email(email\_subject, email\_message, email\_recipient)  
 messages.success(request, 'Email sent successfully!')  
 except Exception as e: #вывод ошибки если письмо не отправилось  
 messages.error(request, f'Error sending email: {e}')  
 return redirect('journal')  
 else:  
 messages.error(request, 'Ошибка при обновлении заявки!')  
 else:  
 form = Breaking\_Form\_Admin(instance=breaking)  
  
 context = { #передача данных в HTML шаблон  
 'form': form,  
 'breaking': breaking  
 }  
 return render(request, 'edit.html', context)

Функция «Add\_table\_with\_headers» создает таблицу с заголовками в рабочем листе файла Excel. Она принимает в себя входные параметры такие как:

* «ws» – объект рабочего листа, где будет добавлена таблица;
* «title» – название таблицы (строка);
* «headers» – список строк, представляющих заголовки столбцов;
* «data» – список списков, где каждый внутренний список представляет собой строку данных в таблице;
* «start\_date» – начальная дата отчета в формате ГГГГ-ММ-ДД (строка);
* «end\_date» – конечная дата отчета в формате ГГГГ-ММ-ДД (строка).

Функция вызывается нажатие на button «Экспорт в Excel» на странице «Отчетность».

Код функции «Add\_table\_with\_headers» представлен ниже

def add\_table\_with\_headers(ws, title, headers, data, start\_date, end\_date):  
 from openpyxl.styles import Font, Alignment, PatternFill  
  
 *# Применение стилей и цветовой палитры* header\_font = Font(color="FFFFFF", bold=True, size=12)  
 header\_fill = PatternFill(start\_color='007ACC', end\_color='007ACC', fill\_type='solid')  
 cell\_font = Font(color="000000", size=11)  
 cell\_fill = PatternFill(start\_color='D9E1F2', end\_color='D9E1F2', fill\_type='solid')  
  
 start\_date\_str = datetime.strptime(start\_date, "%Y-%m-%d").strftime('%d-%m-%Y')  
 end\_date\_str = datetime.strptime(end\_date, "%Y-%m-%d").strftime('%d-%m-%Y')  
  
 *# Adding the header phrase* header\_phrase = f"«Детская городская клиническая поликлиника №8», отчет с {start\_date\_str} по {end\_date\_str}"  
  
 ws.append([header\_phrase])  
 ws.merge\_cells(start\_row=ws.max\_row, start\_column=1, end\_row=ws.max\_row, end\_column=len(headers))  
 for cell in ws.iter\_rows(min\_row=ws.max\_row, max\_row=ws.max\_row, min\_col=1, max\_col=len(headers)):  
 for c in cell:  
 c.font = Font(color="FFFFFF", size=12, bold=True) *# Чёрный цвет, 14 шрифт, не жирный* c.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
 c.fill = PatternFill(start\_color='007ACC', end\_color='007ACC', fill\_type='solid') *# Синий фон* ws.append([]) *# Empty row  
  
 # Установка ширины столбцов* column\_widths = [50, 50, 50]  
 for i, width in enumerate(column\_widths, 1):  
 ws.column\_dimensions[chr(64 + i)].width = width  
  
 *# Определение ширины заголовка* title\_width = len(headers)  
  
 *# Добавление заголовка* ws.append([title])  
 ws.merge\_cells(start\_row=ws.max\_row, start\_column=1, end\_row=ws.max\_row, end\_column=title\_width)  
 for cell in ws.iter\_rows(min\_row=ws.max\_row, max\_row=ws.max\_row, min\_col=1, max\_col=title\_width):  
 for c in cell:  
 c.font = header\_font  
 c.fill = header\_fill  
 c.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
  
 *# Добавление заголовков таблицы* ws.append(headers)  
 for cell in ws.iter\_rows(min\_row=ws.max\_row, max\_row=ws.max\_row, min\_col=1, max\_col=len(headers)):  
 for c in cell:  
 c.font = header\_font  
 c.fill = header\_fill  
 c.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
  
 *# Добавление данных* for row\_data in data:  
 ws.append(row\_data)  
 for row in ws.iter\_rows(min\_row=ws.max\_row - len(data) + 1, max\_row=ws.max\_row, min\_col=1, max\_col=len(headers)):  
 for cell in row:  
 cell.font = cell\_font  
 cell.fill = cell\_fill  
 cell.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
  
 ws.append([]) *# Пустая строка* return ws

Функция «Report» используется в веб-приложении для генерации отчета за определенный период. Она обрабатывает HTTP-запросы и возвращает либо HTML-шаблон с данными для отчета, после чего можно экспортировать отчет с HTML страницы в формате Excel. Функция вызывается при загрузке страницы «Отчетность».

Код функции «Report» представлен ниже

def report(request):  
 start\_date = request.GET.get('start\_date')  
 end\_date = request.GET.get('end\_date')  
  
 breaking\_list = Breaking.objects.filter(breaking\_date\_\_range=[start\_date, end\_date])  
 total\_breakings = breaking\_list.count()  
 staff\_list = Staff.objects.all()  
  
 material\_list = BreakingMaterial.objects.all()  
 breaking\_materials = BreakingMaterial.objects.select\_related('breaking', 'material').all()  
 breaking\_statistics = breaking\_list.values('breaking\_status').annotate(count=Count('id'))  
  
 material\_aggregated = BreakingMaterial.objects.filter(breaking\_\_breaking\_date\_\_range=[start\_date, end\_date]).values(  
 'material\_\_material\_name').annotate(  
 total\_quantity=Coalesce(Sum('quantity', output\_field=DecimalField()), 0, output\_field=DecimalField()),  
 breaking\_numbers=Count('breaking', distinct=True)  
 )  
  
 staff\_reports = Staff.objects.annotate(  
 group\_name=F('staff\_category\_\_name'),  
 total\_breakings=Count('breaking', filter=Q(breaking\_\_breaking\_date\_\_range=[start\_date, end\_date]))  
 )  
  
 if request.method == 'POST' and 'export\_excel' in request.POST:  
 wb = Workbook()  
  
 *# Первая таблица* ws = wb.active  
 ws.title = "Отчет по заявкам"  
 add\_table\_with\_headers(ws, "Перечень заявок", ["Заявка №", "Дата", "Статус"], [  
 [breaking.breaking\_number, breaking.breaking\_date.strftime('%d.%m.%Y'), breaking.breaking\_status] for  
 breaking in breaking\_list], start\_date, end\_date)  
  
 *# Вторая таблица* materials\_ws = wb.create\_sheet(title="Отчет по материалам")  
 add\_table\_with\_headers(materials\_ws, "Использованные материалы", ["Материалы", "Количество"],  
 [[breaking\_material['material\_\_material\_name'], breaking\_material['total\_quantity']]  
 for breaking\_material in material\_aggregated], start\_date, end\_date)  
  
 *# Третья таблица* statuses\_ws = wb.create\_sheet(title="Отчет по статусам")  
 add\_table\_with\_headers(statuses\_ws, "Статусы заявок", ["Статус", "Количество"],  
 [[stat['breaking\_status'], stat['count']] for stat in breaking\_statistics], start\_date,  
 end\_date)  
  
 *# Четвертая таблица* staff\_ws = wb.create\_sheet(title="Отчет по сотрудникам")  
 add\_table\_with\_headers(staff\_ws, "Отчеты по сотрудникам", ["ФИО сотрудника", "Категория", "Количество заявок"],  
 [  
 [report.staff\_fullname, report.group\_name, report.total\_breakings] for report in  
 staff\_reports], start\_date, end\_date)  
  
 response = HttpResponse(content\_type='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')  
 response['Content-Disposition'] = 'attachment; filename="report.xlsx"'  
 wb.save(response)  
 return response  
 else:  
 context = {  
 'breaking\_list': breaking\_list,  
 'total\_breakings': total\_breakings,  
 'staff\_list': staff\_list,  
 'material\_list': material\_list,  
 'start\_date': start\_date,  
 'end\_date': end\_date,  
 'breaking\_statistics': breaking\_statistics,  
 'material\_aggregated': material\_aggregated,  
 'breaking\_materials': breaking\_materials,  
 'staff\_reports': staff\_reports  
 }

return render(request, 'report.html', context)

Код основных модулей веб-приложения представлен в приложении А. Полный код приведен в электронном документе «Полный\_код.docx».

**3.5 Функциональное тестирование**

Функциональное тестирование – это тестирование функций приложения на соответствие всем требованиям и проводится для выявления различных неполадок и недочетов программной реализации на этапе ее сдачи в эксплуатацию. Представляет собой комплекс ключевых мероприятий по проверке веб-приложения, по результатам которых устанавливается соответствие этого по требованиям.

В таблице 10 представлены тест-кейсы для проведения функционального тестирования.

Предварительное условие: запустить программу и находиться на главной странице веб-приложения. Главная страница неавторизированного пользователя представлена в приложении Б на рисунке Б.1. Главная страница для авторизированного пользователя представлена в приложении Б на рисунке Б.2. Главная страница для суперпользователя представлена в приложении Б на рисунке Б.3.

Таблица 10 – Тест-кейсы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль / функция | Шаги воспроизведения | Результат |
| 1 | Отправка заявки | 1. Ввести «Альбертов Альберт Альбертович» в поле «ФИО». 2. Ввести «alexeirubets23@gmail.com» в поле «Электронная почта» 3. Ввести кабинет «404» в поле «Кабинет». 4. Выбрать категорию «Сантехника» в поле «Категория поломки». 5. Ввести «Сломался унитаз» в поле «Описание». 6. Нажать на кнопку «Отправить» | Ожидаемый: отправка сообщения о регистрации заявки |
| Фактический: форма заявки с заполненными данными представлена в приложении Б на рисунке Б.4, фактический результат соответствует ожидаемому и представлен на рисунке Б.5, все условия соблюдены |
| 2 | Авторизация | 1. Перейти на страницу авторизации 2. Ввести «сантехник» в поле «Логин» 3. Ввести пароль «123456» в «Пароль» 4. Нажать на кнопку «Войти» | Ожидаемый: отображение профиля пользователя |
| Фактический: форма авторизации с заполненными данными представлена в приложении Б на рисунке Б.6, фактический результат соответствует ожидаемому и представлен на рисунке Б.7, все условия соблюдены |
| Продолжение таблицы 10 | | | |
| № | Модуль / функция | Шаги воспроизведения | Результат |
| 3 | Поиск заявки | 1. Перейти на страницу заявок 2. Нажать на кнопку «Параметры» 3. Ввести «404» в поле «Поиск…» 4. Нажать на кнопку «Найти» | Ожидаемый: отображение заявок с кабинетом «404» |
| Фактический: фактический результат соответствует ожидаемому и представлен на Б.8, все условия соблюдены |
| 4 | Редактирование данных | 1. Открыть страницу заявок 2. Нажать на кнопку «Параметры» 3. Ввести «Альберт» в поле «Поиск…» 4. Нажать на кнопку «Найти» 5. Нажать на кнопку «Редактировать» 6. Выбрать «В процессе» в поле «Статус» 7. Нажать на кнопку «Сохранить» | Ожидаемый: данные заявки будут изменены |
| Фактический: список заявок с найденными данными представлен в приложении Б на рисунке Б.9, фактический результат соответствует ожидаемому и представлен на рисунке Б.10, все условия соблюдены |
| 5 | Добавление материала к заявке | 1. Открыть страницу заявок 2. Нажать на кнопку «Параметры» 3. Ввести «Альберт» в поле «Поиск…» 4. Нажать на кнопку «Найти» 5. Нажать на кнопку «Материалы» 6. Выбрать «Пачка гвоздей» в поле «Материал» 7. Ввести «1» в поле «Количество» 8. Нажать на кнопку «Добавить» | Ожидаемый: отображение привязанного материала |
| Фактический: фактический результат соответствует ожидаемому и представлен на рисунке Б.11, все условия соблюдены |
| Продолжение таблицы 10 | | | |
| № | Модуль / функция | Шаги воспроизведения | Результат |
| 6 | Добавление нового материала к заявке | 1. Открыть страницу заявок 2. Нажать на кнопку «Параметры» 3. Ввести «Альберт» в поле «Поиск…» 4. Нажать на кнопку «Найти» 5. Нажать на кнопку «Материалы» 6. Нажать на кнопку «+» 7. Ввести «Труба» в «Название» 8. Нажать на кнопку «Добавить» 9. Выбрать «Труба» в поле «Материал» 10. Ввести «1» в поле «Количество» | Ожидаемый: отображение нового привязанного материала |
| Фактический: список с доступными материалами представлен в приложении Б на рисунке Б.12, фактический результат соответствует ожидаемому и представлен на рисунке Б.13, все условия соблюдены |
| 7 | Уведомление о готовности заявки | 1. Открыть страницу заявок 2. Нажать на кнопку «Параметры» 3. Ввести «Альберт» в поле «Поиск…» 4. Нажать на кнопку «Найти» 5. Нажать на кнопку «Редактировать» 6. Выбрать «Выполнено» в поле «Статус» 7. Нажать на кнопку «Сообщить о готовности» | Ожидаемый: отправка сообщения о готовности заявки |
| Фактический: форма редактирования заявки представлен на рисунке Б.14, фактический результат соответствует ожидаемому и представлен на рисунке Б.15, все условия соблюдены |
| 8 | Фильтрация данных | 1. Открыть страницу заявок 2. Нажать на кнопку «Параметры» 3. Выбрать статус «Выполнено» 4. Нажать на кнопку «Найти» | Ожидаемый: отображение выполненных заявок |
| Фактический: фактический результат соответствует ожидаемому и представлен на рисунке Б.16, все условия соблюдены |
| 9 | Выход из текущего профиля | 1. Нажать на кнопку «Профиль» 2. Нажать на кнопку «Выйти из профиля» | Ожидаемый: выход из профиля текущего пользователя |
| Фактический: фактический результат соответствует ожидаемому и представлен на рисунке Б.1, все условия соблюдены |
| Продолжение таблицы 10 | | | |
| № | Модуль / функция | Шаги воспроизведения | Результат |
| 10 | Переход на страницу персонала | 1. Перейти на страницу авторизации 2. Ввести «superuser» в поле «Логин» 3. Ввести пароль «dgkpADMIN» в «Пароль» 4. Нажать на кнопку «Войти» 5. Нажать на кнопку «Заявки» 6. Нажать на кнопку «Персонал» | Ожидаемый: отображение страницы данных персонала |
| Фактический: фактический результат соответствует ожидаемому и представлен на рисунке Б.17 все условия соблюдены |
| 11 | Формирование отчета | 1. Открыть страницу заявок 2. Нажать на кнопку «Отчетность» 3. Нажать на кнопку «Параметры» 4. Выбрать дату «05.01.2024» в поле «Начальная дата» 5. Выбрать дату «05.06.2024» в поле «Конечная дата» 6. Нажать на кнопку «Сформировать» | Ожидаемый: отображение отчетов за выбранный период времени |
| Фактический: фактический результат соответствует ожидаемому и представлен на рисунке Б.18, все условия соблюдены |
| 12 | Экспорт отчета | 1. Открыть страницу заявок 2. Нажать на кнопку «Отчетность» 3. Нажать на кнопку «Параметры» 4. Выбрать дату «05.01.2024» в поле «Начальная дата» 5. Выбрать дату «05.06.2024» в поле «Конечная дата» 6. Нажать на кнопку «Сформировать» 7. Нажать на кнопку «Экспорт в Excel» | Ожидаемый: установится Excel-документ |
| Фактический: фактический результат соответствует ожидаемому и представлен на рисунке Б.19, все условия соблюдены |

**4** **Применение**

**4.1 Назначение веб-приложения**

Веб-приложение, предназначенное для управления заявками в хозяйственной службе поликлиники, решает следующие задачи:

* автоматизация процесса подачи и обработки заявок;
* повышение эффективности работы хозяйственной службой, где сотрудники могут быстрее реагировать на заявки;
* прозрачный канал коммуникации между руководством и хозяйственной службой;
* руководство хозяйственной службы имеют доступ к подробной статистике по количеству, типам, срокам выполнения заявок, использованным материалам и другим показателям, что позволяет принимать обоснованные решения по оптимизации работы.

Целями разработки веб-приложения является автоматизация и упрощение процесса управления заявками о поломках. Это позволяет сократить время на обработку заявок, также отчеты о количестве, типах и сроках выполнения заявок, а также об использованных материалах помогут руководству хозяйственной службы оптимизировать распределение ресурсов и снизить расходы. Еще одной целью является прозрачный процесс обработки заявок, позволяя отслеживать их статус.

В область применения веб-приложения входит управление заявками о поломках в детской поликлинике. Пользователи могут зарегистрировать новые заявки, просматривать и редактировать существующие, а также получать уведомления о статусе своих заявок. Приложение ориентировано на работников поликлиники, обеспечивая им возможность эффективного взаимодействия и оперативного реагирования на возникающие проблемы.

Однако следует отметить, что веб-приложение имеет ограниченную область применения и функционал. Оно предназначено исключительно для управления заявками о поломках и не предоставляет другие функции, такие как управление медицинскими записями или организацию расписания приема. Доступ к функциям приложения имеют только авторизованные пользователи, что ограничивает его использование широкой публике.

* 1. **Программно-аппаратное обеспечение сервера и клиента**

Для корректной работы веб-приложения на сервере необходимы:

* Процессор: Intel Core i5 или аналогичный.
* Оперативная память: 8 ГБ.
* Свободное место на диске: 2 ГБ (HDD/SSD).
* Операционная система: Windows Server 2019/2022 или Linux (Ubuntu 20.04+).
* Программное обеспечение:
  + Node.js (версия 18.x или выше).
  + Microsoft SQL Server (версия 2019 или выше).
  + Git для управления версиями.  
    Эти требования обоснованы необходимостью обработки множества запросов, хранения данных в реляционной базе и обеспечения стабильной работы API.

**Требования к клиенту**  
На стороне клиента необходимы:

* Веб-браузер: Google Chrome, Mozilla Firefox, Safari или Microsoft Edge (последние версии).
* Оперативная память: 4 ГБ.
* Интернет-соединение: скорость от 1 Мбит/с.  
  Эти требования обеспечивают корректное отображение адаптивного интерфейса, быструю загрузку данных и интерактивность (например, перетаскивание задач на доске Канбан).

С учетом указанных требований веб-приложение гарантирует стабильную работу как на стороне сервера, так и на стороне клиента, обеспечивая удобство использования для всех пользователей.

**5 Охрана труда и окружающей среды**

**5.1 Правовые, нормативные, социально-экономические и организационные вопросы охраны труда**

Законодательство Республики Беларусь в области охраны труда в медицинских учреждениях, включая «8-ую детскую городскую клиническую поликлинику», определяется рядом нормативных актов. Ключевыми являются Закон Республики Беларусь «О труде». Этот нормативный документы служит основным для создания и поддержания системы управления охраной труда в детской поликлинике [4].

Важным элементом охраны труда является соблюдение стандартов безопасности, и ГОСТы играют здесь ключевую роль. Например, ГОСТ устанавливает основные принципы системы управления охраной труда, а ГОСТ 12.1.004-91 «Система стандартов безопасности труда. Пожарная безопасность. Общие требования» определяет требования к пожарной безопасности, что крайне важно для медицинских учреждений [5].

Руководитель утверждает список должностей специалистов, которые проводят стажировку, и определяет ее продолжительность. Руководители и специалисты должны проходить проверку знаний по вопросам охраны труда не реже одного раза в три года, начиная со дня назначения на должность, с участием комиссий.

Ответственность за организацию обучения и проверок знаний по охране труда лежит на главного врача и заведующих и инженера по охране труда поликлиники. Вводный инструктаж с новыми работниками всех специальностей проводит инженер по охране труда или уполномоченное лицо в специально оборудованном кабинете.

Основные задачи контроля – обеспечение безопасных условий труда, предупреждение травматизма и заболеваний, соблюдение законодательства. Методы контроля включают проверки, анализ травматизма и заболеваний, опросы работников, и изучение документации. Результаты контроля ведут к выявлению нарушений, разработке и реализации мероприятий по устранению нарушений, и привлечению к ответственности виновных лиц.

Надзор и контроль за охраной труда осуществляется внутренними и внешними органами. Главный врач, инженер по охране труда, служба охраны труда и профсоюзный комитет осуществляют внутренний контроль, включая общий контроль, проверки условий труда, контроль за соблюдением требований, утверждает положение об охране труда, назначает ответственных за охрану труда в структурных подразделениях, и обеспечивает финансирование мероприятий по охране труда. Инженер по охране труда осуществляет методическое руководство, разрабатывает инструкции, проводит вводный инструктаж, осуществляет контроль за соблюдением требований и организует обучение по охране труда.

Расследование несчастных случаев и профессиональных заболеваний в поликлинике направлено на выявление причин и принятие мер для предотвращения их повторения. Это включает:

* определение причин происшествия и разработку срочных мер по устранению травматичных ситуаций;
* выявление виновных лиц, ответственных за нарушения, приведшие к происшествию, а также нарушенные нормы и правила;
* применение законодательства и нормативных актов для привлечения виновных лиц к ответственности.

Важно отметить, что учет травматизма и профессиональных заболеваний осуществляется в соответствии с требованиями акта «О расследовании и учете несчастных случаев на производстве и профессиональных заболеваний». Это позволяет систематизировать данные о произошедших случаях и принимать меры для предотвращения их повторения [6]. В «8-ой детской городской клинической поликлинике» расследование и учет несчастных случаев, профессиональных заболеваний осуществляются в соответствии с утвержденными «Правилами расследования и учета несчастных случаев на производстве и профессиональных заболеваний», установленными Постановлением Совета Министров Республики Беларусь от 15.01.2004 г. №30 [7].

**5.2 Разработка мер по нормализации и защите от токсических веществ на производстве**

Общеядовитым называется действие химических веществ на организм, сопровождающееся повреждением биологических механизмов энергетического обеспечения процессов жизнедеятельности.

Основным содержанием биоэнергетических процессов в организме является непрерывный синтез в клетках и поддержание на постоянном уровне концентрации богатых энергией (макроэргических) соединений, в частности, аденозинтрифосфорной кислоты (АТФ). Источником, запасаемой в форме АТФ энергии, является биологическое окисление субстратов, образующихся в ходе метаболизма питательных веществ, поступающих из окружающей среды. Практически любой токсикант, вызывая тяжелую острую интоксикацию, в той или иной степени нарушает энергетический обмен, то есть оказывает общеядовитое действие.

Вместе с тем имеются вещества, способные первично повреждать систему энергообеспечения клеток, нарушая:

* механизмы транспорта кислорода кровью;
* механизмы биологического окисления;
* механизмы сопряжения биологического окисления и синтеза макроэргов (фосфорилирования).

Токсиканты, основным (первичным) механизмом повреждающего действия которых на организм является нарушение биоэнергетики, могут быть объединены в группу веществ общеядовитого действия.

Важными особенностями токсического процесса, развивающегося при отравлении такими веществами, являются:

* быстрота развития острой интоксикации (короткий скрытый период, бурное течение токсического процесса);
* функциональный характер нарушений со стороны вовлеченных в токсический процесс органов и систем, отсутствие грубых структурно-морфологических изменений в тканях отравленных;
* вовлечение в патологический процесс преимущественно органов и систем с интенсивным энергообменом и, прежде всего, центральной нервной системы;
* закономерный характер развития нарушений со стороны центральной нервной системы (ЦНС)
* возбуждение, переходящее в состояние гиперактивации, а затем глубокого угнетения (изменение сознания, судороги, кома и т.д.).

Как в любых медицинских учреждениях, важны меры по нормализации рабочей среды и защите от токсических веществ для обеспечения безопасности как персонала, так и пациентов. Ниже представлены основные меры, которые приняты в организации:

* вентиляция и проветривание помещений помогает избежать накопления токсичных веществ в воздухе, включает в себя установку системы вентиляции или простые меры проветривания, такие как открытие окон;
* использование личных защитных средств такие как маски, перчатки, защитные очки и халаты, если они работают с токсичными веществами;
* обучение и информирование персонал обучен в обращении с токсичными веществами и знает правила безопасности при работе с ними;
* проведение регулярного мониторинга уровня токсичных веществ в воздухе и на поверхностях может помочь в раннем выявлении проблем и принятии соответствующих мер;
* регулярная уборка помещений и обработка поверхностей с применением безопасных для здоровья средств очистки помогают минимизировать риск контакта с токсичными веществами;
* проведение регулярных медицинских обследований и мониторинга здоровья персонала для выявления возможных негативных последствий работы с токсичными веществами.

Эти меры помогают создать безопасную среду как для персонала, так и для пациентов. Если меры нормализации не принимаются, то возникает необходимость в более активной защите персонала от воздействия токсичных веществ.

**5.3 Пожарная безопасность**

Пожарная безопасность в медицинском учреждении, включая поликлинику, является одним из ключевых аспектов обеспечения безопасности персонала и пациентов. Этот аспект требует строгого соблюдения правил, разработки планов эвакуации, обучения персонала и обеспечения необходимого оборудования для быстрого реагирования в случае возникновения пожарной угрозы. Соблюдение пожарной безопасности в медицинских учреждениях, особенно в детских поликлиниках, является критически важным. В соответствии с ГОСТ 12.1.004-91 «Система стандартов безопасности труда. Пожарная безопасность. Общие требования» необходимо проводить регулярные инструктажи персонала по пожарной безопасности и осуществлять плановые проверки противопожарного оборудования.

Предотвращение пожаров начинается с осознания рисков и адекватных мер безопасности. Для этого проводятся регулярные проверки электрооборудования, проводятся тренировочные пожарные учения, осуществляется контроль за оборудованием, обучение персонала правилам эвакуации и пользования противопожарными средствами.

12.1.004-91 «Система стандартов безопасности труда. Пожарная безопасность. Общие требования» определяет общие требования к пожарной безопасности на предприятии, включая медицинские учреждения. Он содержит указания по оборудованию помещений средствами пожаротушения, правила хранения и использования горючих материалов, а также требования к эвакуационным планам и мероприятиям. В рамках стандарта важно организовать профилактические меры, такие как регулярные проверки электрооборудования, контроль за правильным хранением легковоспламеняющихся материалов, а также четкое обучение персонала правилам пожарной безопасности.

Сотрудники детской поликлиники регулярно проходят инструктажи и обучение по действиям в случае возникновения пожара, что соответствует требованиям Закона «О пожарной безопасности» [8].

Создание планов эвакуации, соответствующих ГОСТу 12.0.001-82 «Система стандартов безопасности труда. Основные положения», играет важную роль в обеспечении безопасности детей и персонала поликлиники. Они четко вывешены и регулярно обновляются в поликлинике [9].

Внимание к пожарной безопасности в детской поликлинике подразумевает соблюдение нормативных требований, регулярные инструктажи, обучение персонала и поддержание оборудования в исправном состоянии. Это необходимо для обеспечения безопасной среды как для маленьких пациентов, так и для персонала учреждения.

Детская поликлиника обеспечена первичными средствами тушения, противопожарными дверьми и средствами оповещения.

Первичные средства тушения включают в себя огнетушители различных классов, такие как порошковые, углекислотные, водные огнетушители. Они размещены на видных и доступных местах в соответствии с требованиями пожарной безопасности.

Противопожарные двери играют важную роль в предотвращении распространения огня и дыма в случае пожара. Они установлены в соответствии с нормами и стандартами безопасности, обеспечивая надежное разделение зон и эвакуацию персонала и пациентов.

Средства оповещения включают в себя системы звукового и визуального оповещения о пожаре или других чрезвычайных ситуациях. Это сирены, мегафоны, световые сигналы, которые расположены таким образом, чтобы обеспечивать быстрое и эффективное оповещение всех находящихся в здании.

Все эти средства поддерживаются в исправном состоянии, чтобы гарантировать их эффективность в случае возникновения пожара или другой чрезвычайной ситуации.

**5.4 Охрана окружающей среды**

Когда речь заходит о поддержании экологической устойчивости в детской поликлинике, соблюдение стандартов и применение современных технологий становятся необходимостью. Законы, такие как «Об охране окружающей среде» и «Об охране атмосферного воздуха являются основополагающими в установлении принципов и требований, направленных на сохранение природы и обеспечение чистоты окружающей среды [10], [11].

Для обеспечения устойчивости и минимизации влияния на окружающую среду, использование закона «Об охране атмосферного воздуха» означает регулярный контроль за состоянием атмосферного воздуха внутри помещений поликлиники. Это включает мониторинг уровня вредных веществ и установку систем очистки воздуха.

В работе детской поликлиники важно учитывать аспекты, связанные с управлением отходами. Использование закона «Об охране окружающей среде» помогает определить способы классификации и утилизации отходов, а также регламентирует их безопасное использование в соответствии с нормами и правилами.

Организация деятельности поликлиники в соответствии с вышеперечисленными законами требует разработки эффективных стратегий. Внедрение экологически устойчивых материалов и технологий, а также проведение регулярных проверок систем очистки воздуха и воды становятся важным шагом на пути к соблюдению стандартов и обеспечению безопасности окружающей среды.

В работе детской поликлиники важно следовать принципам экологической ответственности. Это включает сокращение использования пластиковых материалов, установку оборудования с энергосберегающими технологиями и организацию специализированной службы, отвечающей за контроль и внедрение экологических мероприятий.

Соблюдение законов и применение современных технологий в поликлинике помогает минимизировать отрицательное воздействие на окружающую среду. Организация деятельности в соответствии с высокими стандартами не только обеспечивает безопасность для пациентов и персонала, но и способствует сохранению природы и экологической устойчивости на долгосрочную перспективу.

**6 Экономический раздел**

**6.1** **Расчет затрат на разработку веб-приложения**

Стоимостная оценка программного продукта и определение экономического эффекта у разработчика предполагает расчет затрат, которые включают следующие элементы:

* материальные затраты;
* затраты на оплату труда;
* отчисления на социальные нужды;
* амортизация основных средств и нематериальных активов;
* прочие затраты.

В материальных затратах отражаются затраты на материалы и принадлежности, бумагу и другие материалы, необходимые для разработки программного продукта, а также затраты на электроэнергию.

Сумма затрат на канцелярские товары, необходимые для разработки программного продукта, составляет 15,70 руб.

Расчет основной заработной платы определяется исходя из численности исполнителей и трудоемкости выполнения работы, премиальных систем оплаты труда по формуле (6.1).

(6.1)

где – часовая тарифная ставка i-й исполнителя, руб.;

– количество часов работы над разработкой программного продукта, ч.;

– коэффициент премий по премиальным системам,  от 1,10 до 1,30.

Роз = 8,00 х 160 х 1,2= 1536,00 руб.

Расчет дополнительной заработной платы исполнителей включает выплаты, предусмотренные трудовым законодательством, и определяется по формуле (6.2).

, (6.2)

где Ндз – норматив дополнительной заработной платы, Ндз ≈ от 10 до 20 %.

Рдз= = 261,12 руб.

Отчисления на социальные нужды(в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле (6.3).

, (6.3)

где Нсн – норма отчислений на социальные нужды, % (согласно действующему законодательству).

Рсн = = 621,80 руб.

Амортизация основных средств и нематериальных активоврассчитываются исходя из стоимости основных средств (ОС), используемых в процессе разработки программного продукта, сроков эксплуатации оборудования (Тс) и годовой нормы амортизации (На).

Для определения затрат по данному элементу будет использоваться линейный способ начисления амортизации.

Норма амортизации вычисляется по формуле (6.4).

(6.4)

где Тс – срок службы оборудования, лет.

.

Амортизационные отчисления рассчитываются по формуле (6.5).

, (6.5)

где ОС – стоимость основных средств.

22,22руб.

Прочие затраты включают затраты на арендную плату, вознаграждения за рационализаторские предложения, выплату авторских гонораров, связь, оплату услуг рекламы и маркетинга и пр.

Прочие затраты вычисляются по формуле (6.6).

(6.6)

где ≈ 10-30 %.

На основании полученных данных рассчитывается общая сумма затрат на разработку программного продукта.

З= 15,70+1797,12+=2717,96 руб.

В результате проведенных расчетов были определены основные элементы затрат на разработку программного продукта, включая материальные затраты, затраты на оплату труда, отчисления на социальные нужды, амортизация основных средств и прочие затраты.

**6.2** **Расчет показателей экономической эффективности разработки**

Заказчик оплачивает разработчику всю сумму расходов по проекту, включая прибыль. После уплаты налогов из прибыли в распоряжении заказчика остается чистая прибыль от проекта. Ввиду того, что программный продукт разрабатывается для одного объекта, чистую прибыль можно считать в качестве экономического эффекта разработчика от реализованного программного продукта.

В дипломном проекте отпускная цена программного продукта, представляет собой не цену за единицу продукции, а цену проекта, за которую его можно продать и получить определённую выгоду.

Отпускная цена на разработку рассчитывается по формуле (6.7).

(6.7)

где ОЦ– отпускная цена, руб.;

З– затраты на разработку, руб.;

П – прибыль, руб.

ОЦ = + = 3180,01 руб.

Прибыль рассчитывается по формуле (6.8).

, (6.8)

гдеR–уровень рентабельности (от 10 до 30 %).

=462,05руб.

Стоимость проекта с учётом НДС представляет собой сумму отпускной цены и налога на добавленную стоимость, рассчитывается по формуле (6.9).

, (6.9)

где ставка НДС, % (согласно действующему законодательству).

руб.

Отпускная цена с учетом НДС рассчитывается по формуле (6.10).

(6.10)

руб.

Таким образом, разработчик программного продукта может продать заказчику программный продукт, что покроет затраты и обеспечит прибыль

**6.3** **Расчет экономической эффективности у пользователя веб-приложения**

Организационно-функциональная структура ООО «ФП Трэйд» представляет собой четко выстроенную систему, которая обеспечивает комплексный подход к управлению проектами и разработке программных продуктов высокого качества. Использование функционально-линейной модели управления позволяет компании эффективно сочетать стратегическое планирование на высшем уровне с оперативной реализацией проектов и точечным контролем на уровне отдельных подразделений.

Ключевым преимуществом такой структуры является глубокая специализация сотрудников каждого из отделов и четко определенные роли, что минимизирует пересечение зон ответственности, исключает дублирование задач и способствует рациональному распределению трудовых и временных ресурсов. Благодаря этому достигается сокращение сроков разработки, повышение качества программных решений и снижение затрат заказчика на сопровождение и поддержку внедряемых систем.

Особенно важную роль в повышении экономической эффективности играет подразделение фронтэнд-разработки. Это подразделение ориентировано на создание интуитивно понятных, удобных и адаптивных пользовательских интерфейсов, что напрямую влияет на продуктивность работы конечных пользователей систем компании. Благодаря применению современных технологий и методологий разработки обеспечивается высокий уровень удовлетворенности заказчиков, снижается необходимость дополнительных затрат на переобучение персонала и повышается скорость адаптации сотрудников к новым программным продуктам.

Экономический эффект для клиентов проявляется в оптимизации ключевых бизнес-процессов за счет внедрения специализированных решений: будь то аналитические инструменты в банковской и финансовой сфере, системы управления предприятием (ERP, CRM), автоматизация процессов страхования и инвестиций или интеграция логистических и EdTech-решений. Используя платформу Microsoft .NET и обеспечивая экспортную направленность своих услуг, компания дополнительно усиливает конкурентные преимущества и расширяет клиентскую базу как на локальном, так и на международном рынке, повышая тем самым свою финансовую устойчивость и долгосрочные перспективы развития.

**Заключение**

В ходе выполнения дипломного проекта было разработано веб-приложение для автоматизации приема и обработки заявок хозяйственной службой в УЗ «8-я городская детская клиническая поликлиника». Процесс разработки включал создание функционала регистрации новых заявок, просмотра и редактирования существующих, а также отправки уведомлений о статусе заявок клиентам. Работа над проектом также включала разработку механизма отслеживания статуса заявок, что позволяет сотрудникам поликлиники быть в курсе текущего состояния обработки заявки.

Анализируя выполнение поставленной задачи, можно отметить, что все поставленные задачи были реализованы в соответствии с требованиями и ожиданиями; веб-приложение успешно обеспечивает эффективное взаимодействие между персоналом поликлиники и клиентами, а также автоматизирует процесс управления заявками о поломках.

В процессе работы над проектом было уделено внимание пользовательскому интерфейсу. Интерфейс веб-приложения был разработан таким образом, чтобы быть интуитивно понятным и удобным для использования пользователями всех категорий. Это поможет снизить время обучения новых пользователей и повысить общую удовлетворенность от использования приложения. Уделено внимание и тестированию веб-приложения на различных устройствах и браузерах, чтобы обеспечить его корректную работу в различных средах, что позволило обеспечить корректную работу веб-приложения.

При анализе степени соответствия проектных решений заданию не было обнаружено несоответствий. Проект был выполнен в полном объеме и в срок, соответствует всем требованиями.

Таким образом, можно сделать вывод об успешном выполнении поставленной задачи и высоком уровне соответствия проектных решений её требованиям. Были учтены все аспекты охраны труда: соблюдение правовых и нормативных требований, создание безопасных и удобных условий труда для сотрудников, разработка мер по нормализации условий труда и защите от потенциальных токсических веществ, обеспечение пожарной безопасности и охраны окружающей среды.

Был произведён расчёт затрат на разработку веб-приложения, что составило 2615,09 рублей. Рассчитан показатель экономической эффективности разработки, который составил 3200,87 рублей. Также был произведён расчёт экономической эффективности у пользователя веб-приложения, где дополнительная прибыль составила 25817,86 рублей.

Дипломный проект реализован с использованием основного языка программирования «Python», веб-фреймворка «Django» и системы управления базами данных «PostgreSQL». Эти технологии обеспечивают высокую производительность, безопасность и масштабируемость веб-приложения.

**Список использованных источников**

1. Багласова, Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчетов для учащихся специальности 2-40 01 01 «Программное обеспечение технологий» / С.В. Банцевич, Т.Г. Багласова. – Минск: КБП, 2022. – 49 c.
2. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для обучающихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» / Т.Г. Багласова, С.В. Банцевич – Минск; КБП, 2024 – 49 с.
3. Григораш, М.А. Методические указания по технико-экономическому обоснованию дипломных проектов для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» / М.А. Григораш Минск; КБП, 2023 14 с.
4. Закон Республики Беларусь «О труде» [Электронный ресурс]. – Pravo.by, 2024. – Режим доступа: <https://pravo.by/document/?guid=3961&p0=HK9900296>. – Дата доступа: 16.03.2024.
5. ГОСТ 12.1.004-91 «Система стандартов безопасности труда. Пожарная безопасность. Общие требования» [Электронный ресурс]. – Белэнерго, 2024. – Режим доступа: https://energodoc.by/document/view?id=2320. – Дата доступа: 24.04.2024.
6. Акт «О расследовании и учете несчастных случаев на производстве и профессиональных заболеваний» [Электронный ресурс]. – Pravo.by, 2024. – Режим доступа: https://pravo.by/document/?guid=3961&p0=C20400030. Дата доступа: 25.04.2024.
7. «Правила расследования и учета несчастных случаев на производстве и профессиональных заболеваний» [Электронный ресурс]. – Департамент государственной инспекции труда, 2024. – Режим доступа: https://storage.git.gov.by/source/1/xNcmgCh70vUh\_-JPYlRMNzEHzhyPiQCO.pdf. Дата доступа: 26.05.2024.
8. Закон Республики Беларусь «О пожарной безопасности» [Электронный ресурс]. – Pravo.by, 2024. – https://pravo.by/document/?guid=3871&p0=v19302403. Дата доступа: 20.04.2024
9. ГОСТ 12.0.001-82 «Система стандартов безопасности труда. Основные положения» [Электронный ресурс]. – Белэнерго, 2024. – Режим доступа: https://energodoc.by/document/view?id=1571. – Дата доступа: 24.04.2024.
10. Закон Республики Беларусь «Об охране атмосферного воздуха» [Электронный ресурс]. – Pravo.by, 2024. – https://pravo.by/document/?guid=3871&p0=H10800002. Дата доступа: 20.04.2024
11. Закон Республики Беларусь «Об охране окружающей среды» [Электронный ресурс]. – Pravo.by, 2024. – https://pravo.by/document/?guid=3871&p0=v19201982. Дата доступа: 20.04.2024
12. React.js [Электронный ресурс]. – React, 2024. – Режим доступа: <https://react.dev/learn>. – Дата доступа: 01.03.2024.
13. TypeScript [Электронный ресурс]. – TypeScript, 2024. – Режим доступа: <https://www.typescriptlang.org/docs/>. – Дата доступа: 01.03.2024.
14. Node.js [Электронный ресурс]. – Node.js, 2024. – Режим доступа: <https://nodejs.org/en/docs>. – Дата доступа: 01.03.2024.
15. Express.js [Электронный ресурс]. – Express, 2024. – Режим доступа: <https://expressjs.com/en/guide/routing.html>. – Дата доступа: 01.03.2024.
16. Microsoft SQL Server [Электронный ресурс]. – Microsoft, 2024. – Режим доступа: <https://learn.microsoft.com/en-us/sql/sql-server>. – Дата доступа: 01.03.2024.
17. GitHub [Электронный ресурс]. – GitHub, 2024. – Режим доступа: <https://docs.github.com/en>. – Дата доступа: 01.03.2024.
18. Postman [Электронный ресурс]. – Postman, 2024. – Режим доступа: <https://learning.postman.com/docs/getting-started/introduction/>. – Дата доступа: 01.03.2024.
19. Figma [Электронный ресурс]. – Figma, 2024. – Режим доступа: <https://www.figma.com/resources/learn-design/>. – Дата доступа: 10.04.2024.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Текст программных модулей**

Создание базы данных:

CREATE DATABASE "8DGKP"

WITH

OWNER = postgres

ENCODING = 'UTF8'

LOCALE\_PROVIDER = 'libc'

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

Forms.py:

from django import forms  
from django.forms import ModelForm  
from dateutil.relativedelta import relativedelta  
import datetime  
from .models import \*  
  
  
class Breaking\_Form(ModelForm):  
 class Meta:  
 model = Breaking  
 fields = ['breaking\_full\_name',  
 'breaking\_room',  
 'breaking\_category',  
 'breaking\_description',  
 'breaking\_email'  
 ]  
 labels = {  
 'breaking\_full\_name': 'ФИО',  
 'breaking\_room': 'Кабинет',  
 'breaking\_category': 'Категория',  
 'breaking\_description': 'Описание',  
 'breaking\_email': 'Электронная почта'  
 }  
 widgets = {  
 'breaking\_date': forms.DateInput(attrs={'type': 'date', 'style': 'color: white;', 'class': 'date-input'}),  
 }  
  
  
class Breaking\_Form\_Admin(forms.ModelForm):  
 class Meta:  
 model = Breaking  
 fields = [  
 'breaking\_full\_name',  
 'breaking\_email',  
 'breaking\_date',  
 'breaking\_room',  
 'breaking\_category',  
 'breaking\_description',  
 'breaking\_status',  
 'breaking\_worker\_fullname',  
  
 ]  
 labels = {  
 'breaking\_full\_name': 'ФИО',  
 'breaking\_date': 'Дата',  
 'breaking\_room': 'Кабинет',  
 'breaking\_category': 'Категория',  
 'breaking\_description': 'Описание',  
 'breaking\_status': 'Статус',  
 'breaking\_worker\_fullname': 'Отвественный',  
 'breaking\_email': 'Электронная почта'  
 }  
  
  
class Material\_Form(ModelForm):  
 class Meta:  
 model = Material  
 fields = ['material\_name']  
 labels = {  
 'material\_name': 'Название'  
 }  
  
  
class BreakingMaterial\_Form(forms.ModelForm):  
 class Meta:  
 model = BreakingMaterial  
 fields = ['breaking', 'material', 'quantity']  
 labels = {  
 'breaking': 'Заявка',  
 'material': 'Материал',  
 'quantity': 'Количество'  
 }  
  
 def \_\_init\_\_(self, breaking\_id, \*args, \*\*kwargs):  
 super(BreakingMaterial\_Form, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['breaking'].initial = breaking\_id if breaking\_id else None  
  
  
class Staff\_Form(forms.ModelForm):  
 class Meta:  
 model = Staff  
 fields = [  
 'staff\_fullname',  
 'staff\_phone\_number',  
 'staff\_category'  
 ]  
 labels = {  
 'staff\_fullname': 'ФИО работника',  
 'staff\_phone\_number': 'Тел. номер',  
 'staff\_category': 'Категория'  
 }

Views.py:

from django.contrib.auth.decorators import login\_required  
from django.contrib import messages  
from django.db.models import Q, Sum, Count  
from django.db.models.functions import Coalesce  
from django.shortcuts import render, get\_object\_or\_404, redirect  
from reportlab.lib.pagesizes import letter  
from reportlab.pdfbase import pdfmetrics  
from reportlab.pdfbase.ttfonts import TTFont  
from .forms import \*  
from .models import Breaking  
from django.http import HttpResponse  
from urllib.parse import unquote  
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger  
from django.contrib.auth.models import Group  
from django.core.mail import EmailMessage, get\_connection  
from django.conf import settings  
from django.db.models import DecimalField  
from io import BytesIO  
from reportlab.pdfgen import canvas  
  
  
@login\_required  
def profile(request):  
 user = request.user  
 context = {  
 'user': user,  
 }  
 return render(request, 'profile.html', context)  
  
  
*# отправка по эл почте*

def send\_email(email\_subject, email\_message, email\_recipient):  
 with get\_connection(  
 host=settings.EMAIL\_HOST,  
 port=settings.EMAIL\_PORT,  
 username=settings.EMAIL\_HOST\_USER,  
 password=settings.EMAIL\_HOST\_PASSWORD,  
 use\_tls=settings.EMAIL\_USE\_TLS  
 ) as connection:  
 email\_from = settings.EMAIL\_HOST\_USER  
 EmailMessage(email\_subject, email\_message, email\_from, email\_recipient, connection=connection).send() *# передача параметров и отправка*  
  
*# главная страница*  
def index(request):  
 notification = None  
 form\_errors = []  
  
 if request.method == "POST":  
 form = Breaking\_Form(request.POST)  
 if form.is\_valid():  
 try:  
 saved\_form = form.save()  
 email\_subject = f'Уважаемый(ая) {saved\_form.breaking\_full\_name}, Ваша заявка поступила в обработку.'  
 email\_message = 'Указанные данные:\n\n'  
 email\_message += (f'Отправитель: {saved\_form.breaking\_full\_name};\n'  
 f'Дата заявки: {saved\_form.breaking\_date};\n'  
 f'Кабинет: {saved\_form.breaking\_room};\n'  
 f'Категория: {saved\_form.breaking\_category};\n'  
 f'Описание: {saved\_form.breaking\_description};\n'  
 f'\n\nС уважением 8DGKP Manager!')  
 email\_recipient = [saved\_form.breaking\_email]  
 send\_email(email\_subject, email\_message, email\_recipient)  
 messages.success(request, 'Заявка успешно оставлена!')  
 form = Breaking\_Form()  
 except Exception as e:  
 messages.error(request, f'Ошибка сохранения формы или отправки электронной почты: {e}')  
 else:  
 form\_errors = [f'{form.fields[field].label}: {error}' for field, error in form.errors.items()]  
 else:  
 form = Breaking\_Form()  
  
 context = {  
 'notification': notification,  
 'form': form,  
 'form\_errors': form\_errors,  
 }  
 return render(request, 'index.html', context)  
  
*# журнал заявок*  
@login\_required  
def journal(request):  
 user = request.user *# получение текущего пользователя*  
 user\_group\_name = None  
  
 selected\_category = request.GET.get('category', '')  
 selected\_status = request.GET.get('status', '')  
 search\_query = request.GET.get('search', '')  
  
 breaking\_materials = BreakingMaterial.objects.select\_related('breaking', 'material').all()  
 breaking\_list = Breaking.objects.all().order\_by('breaking\_date')  
  
 if user.is\_authenticated: *# проверка аутенфикации*  
 user\_groups = user.groups.all()  
 if user\_groups.exists():  
 user\_group = user\_groups.first()  
 user\_group\_name = user\_group.name  
  
 if user\_group\_name: *# проверка категории пользователя*  
 breaking\_list = breaking\_list.filter(breaking\_category\_\_name=user\_group\_name)  
  
 if selected\_category: *# выбор категории*  
 selected\_category = unquote(selected\_category.replace('+', ' '))  
 breaking\_list = breaking\_list.filter(breaking\_category\_\_name\_\_iexact=selected\_category)  
  
 if selected\_status: *# выбор статуса*  
 breaking\_list = breaking\_list.filter(breaking\_status=selected\_status)  
  
 if search\_query:  
 breaking\_list = breaking\_list.filter(  
 Q(breaking\_full\_name\_\_icontains=search\_query) |  
 Q(breaking\_date\_\_icontains=search\_query) |  
 Q(breaking\_room\_\_icontains=search\_query) |  
 Q(breaking\_description\_\_icontains=search\_query) |  
 Q(breaking\_status\_\_icontains=search\_query)  
 )  
  
 categories = set(b.breaking\_category.name for b in breaking\_list if b.breaking\_category)  
 statuses = set(b.breaking\_status for b in breaking\_list)  
  
 page = request.GET.get('page', 1)  
 paginator = Paginator(breaking\_list, 8)  
  
 try:  
 breaking\_list = paginator.page(page)  
 except PageNotAnInteger:  
 breaking\_list = paginator.page(1)  
 except EmptyPage:  
 breaking\_list = paginator.page(paginator.num\_pages)  
  
 context = {  
 'breaking\_list': breaking\_list,  
 'breaking\_materials': breaking\_materials,  
 'categories': categories,  
 'statuses': statuses,  
 'selected\_category': selected\_category,  
 'selected\_status': selected\_status,  
 }  
 return render(request, 'journal.html', context)  
  
  
@login\_required  
def edit\_breaking(request, breaking\_id):  
 breaking = get\_object\_or\_404(Breaking, id=breaking\_id) *# получаем айди поломки*  
 if request.method == "POST":  
 form = Breaking\_Form\_Admin(request.POST, instance=breaking)  
 if form.is\_valid(): *# отправка письма о готовности*  
 form.save()  
 messages.success(request, 'Заявка успешно обновлена!')  
 email\_breaking\_full\_name = form.cleaned\_data['breaking\_full\_name']  
 email\_breaking\_date = form.cleaned\_data['breaking\_date']  
 email\_breaking\_room = form.cleaned\_data['breaking\_room']  
 email\_breaking\_category = form.cleaned\_data['breaking\_category']  
 email\_breaking\_description = form.cleaned\_data['breaking\_description']  
 email\_breaking\_recipient = form.cleaned\_data['breaking\_email']  
 if 'send\_email\_button' in request.POST:  
 email\_subject = f'Уважаемый(ая) {email\_breaking\_full\_name}, Заявка {breaking.breaking\_number} выполнена!'  
 email\_message = 'Указанные данные:\n\n'  
 email\_message += (f'Отправитель: {email\_breaking\_full\_name};\n'  
 f'Дата заявки: {email\_breaking\_date};\n'  
 f'Кабинет: {email\_breaking\_room};\n'  
 f'Категория: {email\_breaking\_category};\n'  
 f'Описание: {email\_breaking\_description};\n'  
 f'\n\nСпасибо за вашу помощь! =)\n'  
 f'С уважением 8DGKP Manager!')  
 email\_recipient = [email\_breaking\_recipient]  
 try:  
 send\_email(email\_subject, email\_message, email\_recipient)  
 messages.success(request, 'Email sent successfully!')  
 except Exception as e:  
 messages.error(request, f'Error sending email: {e}')  
 return redirect('journal')  
 else:  
 messages.error(request, 'Ошибка при обновлении заявки!')  
 else:  
 form = Breaking\_Form\_Admin(instance=breaking)  
  
 context = {  
 'form': form,  
 'breaking': breaking  
 }  
 return render(request, 'edit.html', context)  
  
  
@login\_required  
def add\_material\_breaking(request, breaking\_id):  
 breaking = get\_object\_or\_404(Breaking, id=breaking\_id) *# получаем текущий айди полоки*  
 materials = BreakingMaterial.objects.filter(breaking\_id=breaking\_id) *# получаем материал*  
  
 if request.method == "POST":  
 form = BreakingMaterial\_Form(breaking\_id, request.POST)  
 if form.is\_valid():  
 material\_breaking = form.save(commit=False)  
 material\_breaking.breaking\_id = breaking\_id  
 material\_breaking.save()  
 messages.success(request, 'Материал успешно добавлен!')  
 return redirect('add\_material\_breaking', breaking\_id=breaking\_id)  
 else:  
 messages.error(request, 'Ошибка при добавлении материала!')  
 else:  
 form = BreakingMaterial\_Form(breaking\_id=breaking\_id)  
  
 if request.method == "POST":  
 mat\_form = Material\_Form(request.POST)  
 if mat\_form.is\_valid():  
 material = mat\_form.save(commit=False)  
 material.save()  
 messages.success(request, 'Добавлен новый материал!')  
 mat\_form = Material\_Form()  
 else:  
 messages.error(request, 'Ошибка добавления материала!')  
 else:  
 mat\_form = Material\_Form()  
  
 context = {  
 'form': form,  
 'materials': materials,  
 'breaking': breaking,  
 'mat\_form': mat\_form  
 }  
 return render(request, 'material-breaking.html', context)  
  
*# удаление материала*  
@login\_required  
def delete\_material(request, breaking\_id, material\_id):  
 material = get\_object\_or\_404(BreakingMaterial, id=material\_id)  
  
 if request.method == 'POST':  
 material.delete()  
 messages.success(request, 'Материал успешно удален!')  
  
 return redirect('add\_material\_breaking', breaking\_id=breaking\_id)  
  
*# таблица персонала*  
@login\_required  
def staff(request):  
 categories = Group.objects.all()  
 if request.method == "POST":  
 form = Staff\_Form(request.POST)  
 if form.is\_valid():  
 staff = form.save(commit=False)  
 staff.save()  
 messages.success(request, 'Добавлен новый персонал!')  
 form = Staff\_Form()  
 else:  
 messages.error(request, 'Ошибка добавления материала!')  
 else:  
 form = Staff\_Form()  
 staff\_list = Staff.objects.all()  
 search\_query = request.GET.get('search', '')  
 selected\_category = request.GET.get('category', '')  
  
 if search\_query:  
 staff\_list = staff\_list.filter(  
 Q(staff\_fullname\_\_icontains=search\_query) |  
 Q(staff\_phone\_number\_\_icontains=search\_query)  
 )  
  
 if selected\_category:  
 selected\_category = unquote(selected\_category.replace('+', ' '))  
 staff\_list = staff\_list.filter(staff\_category\_\_name\_\_iexact=selected\_category)  
  
 page = request.GET.get('page', 1)  
 paginator = Paginator(staff\_list, 7)  
  
 try:  
 staff\_list = paginator.page(page)  
 except PageNotAnInteger:  
 staff\_list = paginator.page(1)  
 except EmptyPage:  
 staff\_list = paginator.page(paginator.num\_pages)  
  
 context = {  
 'categories': categories,  
 'staff\_list': staff\_list,  
 'form': form  
 }  
 return render(request, 'staff.html', context)  
  
*# удаление сотруднков*  
@login\_required  
def delete\_staff(request, staff\_id):  
 staff = get\_object\_or\_404(Staff, id=staff\_id)  
  
 if request.method == 'POST':  
 staff.delete()  
 messages.success(request, 'Работник успешно удален!')  
  
 return redirect('staff')  
  
*# редактирование*  
@login\_required  
def edit\_staff(request, staff\_id):  
 staff = get\_object\_or\_404(Staff, id=staff\_id)  
 if request.method == "POST":  
 form = Staff\_Form(request.POST, instance=staff)  
 if form.is\_valid():  
 form.save()  
 messages.success(request, 'Заявка успешно обновлена!')  
 return redirect('staff')  
 else:  
 messages.error(request, 'Ошибка при обновлении заявки!')  
 else:  
 form = Staff\_Form(instance=staff)  
  
 context = {  
 'form': form,  
 'staff': staff  
 }  
 return render(request, 'edit.html', context) *# сформировываем ответ в хтмл страницы и переменных в контроллере*  
  
def report(request):  
 start\_date = request.GET.get('start\_date')  
 end\_date = request.GET.get('end\_date')  
  
 breaking\_list = Breaking.objects.filter(breaking\_date\_\_range=[start\_date, end\_date])  
 total\_breakings = breaking\_list.count()  
 staff\_list = Staff.objects.all()  
  
 material\_list = BreakingMaterial.objects.all()  
 breaking\_materials = BreakingMaterial.objects.select\_related('breaking', 'material').all()  
 breaking\_statistics = breaking\_list.values('breaking\_status').annotate(count=Count('id'))  
  
 material\_aggregated = BreakingMaterial.objects.filter(breaking\_\_breaking\_date\_\_range=[start\_date, end\_date]).values(  
 'material\_\_material\_name').annotate(  
 total\_quantity=Coalesce(Sum('quantity', output\_field=DecimalField()), 0, output\_field=DecimalField()),  
 breaking\_numbers=Count('breaking', distinct=True)  
 )  
  
 staff\_reports = Staff.objects.annotate(  
 group\_name=F('staff\_category\_\_name'),  
 total\_breakings=Count('breaking', filter=Q(breaking\_\_breaking\_date\_\_range=[start\_date, end\_date]))  
 )  
  
 if request.method == 'POST' and 'export\_excel' in request.POST:  
 wb = Workbook()  
  
 *# Первая таблица* ws = wb.active  
 ws.title = "Отчет по заявкам"  
 add\_table\_with\_headers(ws, "Перечень заявок", ["Заявка №", "Дата", "Статус"], [  
 [breaking.breaking\_number, breaking.breaking\_date.strftime('%d.%m.%Y'), breaking.breaking\_status] for  
 breaking in breaking\_list], start\_date, end\_date)  
  
 *# Вторая таблица* materials\_ws = wb.create\_sheet(title="Отчет по материалам")  
 add\_table\_with\_headers(materials\_ws, "Использованные материалы", ["Материалы", "Количество"],  
 [[breaking\_material['material\_\_material\_name'], breaking\_material['total\_quantity']]  
 for breaking\_material in material\_aggregated], start\_date, end\_date)  
  
 *# Третья таблица* statuses\_ws = wb.create\_sheet(title="Отчет по статусам")  
 add\_table\_with\_headers(statuses\_ws, "Статусы заявок", ["Статус", "Количество"],  
 [[stat['breaking\_status'], stat['count']] for stat in breaking\_statistics], start\_date,  
 end\_date)  
  
 *# Четвертая таблица* staff\_ws = wb.create\_sheet(title="Отчет по сотрудникам")  
 add\_table\_with\_headers(staff\_ws, "Отчеты по сотрудникам", ["ФИО сотрудника", "Категория", "Количество заявок"],  
 [  
 [report.staff\_fullname, report.group\_name, report.total\_breakings] for report in  
 staff\_reports], start\_date, end\_date)  
  
 response = HttpResponse(content\_type='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')  
 response['Content-Disposition'] = 'attachment; filename="report.xlsx"'  
 wb.save(response)  
 return response  
 else:  
 context = {  
 'breaking\_list': breaking\_list,  
 'total\_breakings': total\_breakings,  
 'staff\_list': staff\_list,  
 'material\_list': material\_list,  
 'start\_date': start\_date,  
 'end\_date': end\_date,  
 'breaking\_statistics': breaking\_statistics,  
 'material\_aggregated': material\_aggregated,  
 'breaking\_materials': breaking\_materials,  
 'staff\_reports': staff\_reports  
 }  
 return render(request, 'report.html', context)

Setting.py:

import os  
import sys  
from pathlib import Path  
import smtplib  
  
*# Build paths inside the project like this: BASE\_DIR / 'subdir'.*BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent  
  
BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent  
  
PROJECT\_ROOT = os.path.dirname(os.path.abspath(\_\_file\_\_))  
sys.path.insert(0, os.path.join(PROJECT\_ROOT, 'apps'))  
  
*# SECURITY WARNING: don't run with debug turned on in production!*DEBUG = False  
  
ALLOWED\_HOSTS = [] *# разрешенные хосты*  
 *# Application definition*INSTALLED\_APPS = [ *# установленные директории проекта*  
 'DGKP',  
  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'djangoProject.urls'  
  
TEMPLATES = [ *# шаблоны*  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [BASE\_DIR / 'templates']  
 ,  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'djangoProject.wsgi.application'  
  
  
DATABASES = { *# бд*  
 'default': {  
 'ENGINE': 'django.db.backends.postgresql',  
 'NAME': '8DGKP',  
 'USER': 'postgres',  
 'PASSWORD': 'dgkpADMIN',  
 'HOST': 'localhost',  
 'PORT': '5432'  
 }  
}  
  
  
*# Password validation  
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators*AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 },  
]  
  
  
LANGUAGE\_CODE = 'ru-RU' *# кодировка языка проекта*  
  
TIME\_ZONE = 'Europe/Moscow' *# часовоей пояс*  
  
USE\_I18N = True  
  
USE\_TZ = True  
  
  
STATIC\_URL = 'static/' *# папка со всеми шрифтами, картинками и т.д*  
STATICFILES\_DIRS = [  
 os.path.join(BASE\_DIR, 'static'),  
]  
STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')  
  
  
DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'  
  
*# переменные для смтп сервера*  
EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'  
EMAIL\_HOST = 'smtp.gmail.com'  
EMAIL\_USE\_TLS = False  
EMAIL\_PORT = 465  
EMAIL\_USE\_SSL = True  
EMAIL\_HOST\_USER = 'dgkpmanager@gmail.com'  
EMAIL\_HOST\_PASSWORD = 'hecc cwgw wpgd mbrf'

Urls.py:

*# импорт либ*

from django.contrib import admin  
from django.urls import path  
from django.contrib.auth import views as auth\_views  
import DGKP  
from DGKP import views  
from DGKP.models import Breaking  
*# урл пути*  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('accounts/login/', auth\_views.LoginView.as\_view(template\_name='sign\_in.html'), name='login'),  
 path('accounts/logout/', auth\_views.LogoutView.as\_view(next\_page='index'), name='logout'),  
 path('accounts/profile/', DGKP.views.profile, name='profile'),  
 path('', DGKP.views.index, name='index'),  
 path('journal', DGKP.views.journal, name='journal'),  
 path('journal/edit/<int:breaking\_id>', views.edit\_breaking, name='edit\_breaking'),  
 path('material/add/breaking/<int:breaking\_id>/', views.add\_material\_breaking, name='add\_material\_breaking'),  
 path('material/delete/breaking/<int:breaking\_id>/<int:material\_id>/', views.delete\_material,  
 name='delete\_material\_breaking'),  
 path('staff/delete/<int:staff\_id>/', views.delete\_staff,  
 name='delete\_staff'),  
 path('staff/edit/<int:staff\_id>', views.edit\_staff, name='edit\_staff'),  
 path('report', views.report, name='report'),  
 path('staff/', views.staff, name='staff'),  
 path('staff/edit/<int:staff\_id>', views.edit\_breaking, name='edit\_staff'),  
 path('send/', views.send\_email, name='send\_email')  
]

Edit.html:

{% load static %} *# загрузка статических файлов*  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Редактирование</title>  
 <link rel="stylesheet" href="{% static 'css/animated-background.css' %}">  
 <link rel="stylesheet" href="{% static 'css/edit.css' %}">  
 <meta content="width=device-width, initial-scale=1" name="viewport"/>  
  
</head>  
<body>  
<div class="bg"></div>  
<div class="bg bg2"></div>  
<div class="bg bg3"></div>  
<div class="content">  
 <div class="edit-box">  
 {% if user.is\_authenticated %}  
 <div class="login-box">  
 <h2>Редактирование</h2>  
 <form method="post">  
 {% csrf\_token %}  
 <div class="user-box">  
 {{ form.as\_p }}  
 </div>  
 <a>  
 <span></span>  
 <span></span>  
 <span></span>  
 <span></span>  
 <button type="submit">Сохранить</button>  
 {% if form.instance.breaking\_status == 'Выполнено' %}   
 {% if form.instance.breaking\_email %} *# проверка на оставленную почту*  
 <button type="submit" name="send\_email\_button">Сообщить о готовности</button> *# отправка письма*  
 {% endif %}  
 {% endif %}  
 </a>  
 </form>  
 </div>  
 {% endif %}  
 </div>  
</div>  
</body>  
</html>

Index.html:

*# загузка статик файлов*

{% load static %}  
<!DOCTYPE html>  
<html>  
<head>  
 <meta charset='utf-8'>  
 <meta http-equiv='X-UA-Compatible' content='IE=edge'>  
 <title>Главная страница</title>  
 <meta name='viewport' content='width=device-width, initial-scale=1'>  
 <link rel="stylesheet" href="{% static 'css/form.css' %}">  
 <link rel="stylesheet" href="{% static 'css/header.css' %}">  
</head>  
<body>  
<div class="header"> *# шапка*  
 <div class="inner-header flex">  
 <img src="{% static 'assets/logo.png' %}">  
 <h1>8DGKP</h1>  
 </div>  
 <div>  
 <svg class="waves" xmlns="http://www.w3.org/2000/svg" *# рисуем волны* xmlns:xlink="http://www.w3.org/1999/xlink"  
 viewBox="0 24 150 28" preserveAspectRatio="none" shape-rendering="auto">  
 <defs>  
 <path id="gentle-wave" d="M-160 44c30 0 58-18 88-18s 58 18 88 18 58-18 88-18 58 18 88 18 v44h-352z"/>  
 </defs>  
 <g class="parallax">  
 <use xlink:href="#gentle-wave" x="48" y="0" fill="rgba(255,255,255,0.7"/>  
 <use xlink:href="#gentle-wave" x="48" y="3" fill="rgba(255,255,255,0.5)"/>  
 <use xlink:href="#gentle-wave" x="48" y="5" fill="rgba(255,255,255,0.3)"/>  
 <use xlink:href="#gentle-wave" x="48" y="7" fill="#fff"/>  
 </g>  
 </svg>  
 </div>  
 <div class="nav-bar">  
 {% if user.is\_authenticated %}  
 <a class="nav-bar-a" href="{% url 'journal' %}">Заявки</a>  
 <a class="nav-bar-a" href="{% url 'profile' %}">Профиль</a>  
  
 {% else %}  
 <a class="nav-bar-a" href="{% url 'profile' %}">Авторизация</a>  
 {% endif %}  
 {% if user.is\_superuser %}  
 <a class="nav-bar-a" href="{% url 'staff' %}">Персонал</a>  
 {% endif %}  
  
 </div>  
</div>  
<div class="login-box-container">  
 <div class="login-box">  
 <h2>Обнаружили поломку?<br>Оставьте заявку!</h2>  
 <form method="post">  
 {% csrf\_token %} *# цсрф токен для предотвращения атак*  
 <div class="user-box">  
 {{ form.breaking\_full\_name }}  
 <label>ФИО:</label>  
  
 </div>  
 <div class="user-box">  
 {{ form.breaking\_email }}  
 <label>Электронная почта (необязательное)</label>  
 </div>  
 <div class="user-box">  
 {{ form.breaking\_room }}  
 <label>Кабинет:</label>  
  
 </div>  
 <div class="user-box">  
 {{ form.breaking\_category }}  
 <label class="select-cat">Категория поломки:</label>  
  
 </div>  
 <div class="user-box">  
 {{ form.breaking\_description }}  
 <label>Описание (необязательное):</label>  
  
 </div>  
 <a>  
 <span></span>  
 <span></span>  
 <span></span>  
 <span></span>  
 <button type="submit">Отправить</button>  
 </a>  
 </form>  
 </div>  
</div>  
<script> *# скрипт в случае ошибок*  
 document.addEventListener('DOMContentLoaded', function () {  
 *// Retrieve form errors from template context* var formErrors = {{ form\_errors|default:"[]"|safe }};   
  
 if (formErrors.length > 0) {  
 var alertMessage = 'Ошибка заполнения формы:\n' + formErrors.join('\n');  
 alert(alertMessage); *# ошибка*  
 }  
 });  
</script>  
</body>  
</html>

Journal.html:

*# загрузка статик файлов*

{% load static %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Журнал заявок</title>  
 <link rel="stylesheet" href="{% static 'css/journal.css' %}">  
 <link rel="stylesheet" href="{% static 'css/filter-bar.css' %}">  
 <link rel="stylesheet" href="{% static 'css/header.css' %}">  
 <link rel="stylesheet" href="{% static 'css/paginator.css' %}">  
 <meta content="width=device-width, initial-scale=1" name="viewport"/>  
</head>  
<body>  
<div class="header"> *# шапка*  
 <div class="inner-header flex">  
 <img src="{% static 'assets/logo.png' %}">  
 <h1>8DGKP</h1>  
 </div>  
 <div>  
 <svg class="waves" xmlns="http://www.w3.org/2000/svg" *# волны* xmlns:xlink="http://www.w3.org/1999/xlink"  
 viewBox="0 24 150 28" preserveAspectRatio="none" shape-rendering="auto">  
 <defs>  
 <path id="gentle-wave" d="M-160 44c30 0 58-18 88-18s 58 18 88 18 58-18 88-18 58 18 88 18 v44h-352z"/>  
 </defs>  
 <g class="parallax">  
 <use xlink:href="#gentle-wave" x="48" y="0" fill="rgba(255,255,255,0.7"/>  
 <use xlink:href="#gentle-wave" x="48" y="3" fill="rgba(255,255,255,0.5)"/>  
 <use xlink:href="#gentle-wave" x="48" y="5" fill="rgba(255,255,255,0.3)"/>  
 <use xlink:href="#gentle-wave" x="48" y="7" fill="#fff"/>  
 </g>  
 </svg>  
 </div>  
 <div class="nav-bar">  
 <a class="nav-bar-a" href="{% url 'index' %}">Главная</a>  
 {% if user.is\_authenticated %}  
 <a class="nav-bar-a" href="{% url 'profile' %}">Профиль</a>  
 {% endif %}  
 {% if user.is\_superuser %}  
 <a class="nav-bar-a" href="{% url 'staff' %}">Персонал</a>  
 <a class="nav-bar-a" href="{% url 'report' %}">Отчетность</a>  
 {% endif %}  
 </div>  
</div>  
{% if user.is\_authenticated %} *# проверка аутенфикации*  
 <div class="journal-button-container">  
 <button class="journal-button" id="toggleButton">Параметры</button>  
 </div>  
 <div id="filterFormContainer" class="filter-bar">  
 <div class="filter-box">  
 <form method="GET">  
 {% if user.is\_superuser %}  
 <select id="category" name="category">  
 <option value="">Все категории</option>  
 {% for category in categories %}  
 <option value="{{ category }}">{{ category }}</option>  
 {% endfor %}  
 </select>  
 {% endif %}  
 <select id="status" name="status">  
 <option value="">Все статусы</option>  
 {% for status in statuses %}  
 <option value="{{ status }}">{{ status }}</option>  
 {% endfor %}  
 </select>  
  
 <button type="submit">Применить фильтр</button>  
 </form>  
 </div>  
 <div class="search-box">  
 <form method="GET">  
 <input type="text" name="search" placeholder="Поиск..." value="{{ request.GET.search }}"  
 autocomplete="off">  
 <button type="submit">Найти</button>  
 </form>  
 </div>  
 </div>  
 <div class="wrapper"> *# создание таблицы*  
 <div class="table">  
 <div class="row header blue">  
 <div class="cell">  
 Номер  
 </div>  
 <div class="cell">  
 ФИО  
 </div>  
 <div class="cell">  
 Эл. почта  
 </div>  
 <div class="cell">  
 Категория  
 </div>  
 <div class="cell">  
 Дата  
 </div>  
 <div class="cell">  
 Кабинет  
 </div>  
 <div class="cell">  
 Описание  
 </div>  
 <div class="cell">  
 Отвественный  
 </div>  
 <div class="cell">  
 Статус  
 </div>  
 <div class="cell">  
 Метариалы  
 </div>  
 <div class="cell">  
 Действие  
 </div>  
 </div>  
 {% for breaking in breaking\_list %} *# # хост*  
 {% if selected\_category == "" or breaking.breaking\_category.name == selected\_category %}  
 <div class="row">  
 <div class="cell">{{ breaking.breaking\_number }}</div>  
 <div class="cell">{{ breaking.breaking\_full\_name }}</div>  
 <div class="cell">{{ breaking.breaking\_email }}</div>  
 <div class="cell">{{ breaking.breaking\_category }}</div>  
 <div class="cell">{{ breaking.breaking\_date }}</div>  
 <div class="cell">{{ breaking.breaking\_room }}</div>  
 <div class="cell">{{ breaking.breaking\_description }}</div>  
 <div class="cell">{{ breaking.breaking\_worker\_fullname }}</div>  
 <div class="cell">  
 {% if breaking.breaking\_status == 'Не выполнено' %}  
 {{ breaking.get\_breaking\_status\_display }}  
 {% elif breaking.breaking\_status == 'Выполнено' %}  
 {{ breaking.get\_breaking\_status\_display }}  
 {% else %} {{ breaking.get\_breaking\_status\_display }}  
 {% endif %}  
 </div>  
 <div class="cell">  
 {% for breaking\_material in breaking\_materials %}  
 {% if breaking\_material.breaking\_id == breaking.id %}  
 {{ breaking\_material.material.material\_name }} ({{ breaking\_material.quantity }});  
 {% endif %}  
 {% endfor %}  
 </div>  
 <div class="cell">  
 <a href="{% url 'edit\_breaking' breaking.id %}" class="action-a">Редактировать</a>  
 <a href="{% url 'add\_material\_breaking' breaking.id %}" class="action-a">Материалы</a>  
 </div>  
 </div>  
 {% endif %}  
 {% empty %}  
 <h1>Таблица пуста.</h1> *# если таблица пуста*  
 {% endfor %}  
 </div>  
  
 </div>  
 <div class="pagination"> *# пагинация страницы*  
 <span>  
 {% if breaking\_list.has\_previous %}  
 <a href="?page=1"><<</a>  
 <a href="?page={{ breaking\_list.previous\_page\_number }}">Предыдущая</a>  
 {% endif %}  
  
 <span class="current">  
 Стр. {{ breaking\_list.number }} из {{ breaking\_list.paginator.num\_pages }}  
 </span>  
  
 {% if breaking\_list.has\_next %}  
 <a href="?page={{ breaking\_list.next\_page\_number }}">Следующая</a>  
 <a href="?page={{ breaking\_list.paginator.num\_pages }}">>></a>  
 {% endif %}  
 </span>  
 </div>  
 <script> *# скрипт для доп панели*  
 document.getElementById('toggleButton').addEventListener('click', function () {  
 var filterFormContainer = document.getElementById('filterFormContainer');  
 filterFormContainer.classList.toggle('visible');  
 });  
 </script>  
{% endif %}  
</body>  
</html>

MaterialBreaking.html:

*# загрузка статик файлов*

{% load static %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Добавление материала</title>  
 <link rel="stylesheet" href="{% static 'css/animated-background.css' %}">  
 <link rel="stylesheet" href="{% static 'css/material-breaking.css' %}">  
 <link rel="stylesheet" href="{% static 'css/modal-screen.css' %}">  
 <meta content="width=device-width, initial-scale=1" name="viewport"/>  
  
</head>  
<body> *# контент*  
<div class="bg"></div>  
<div class="bg bg2"></div>  
<div class="bg bg3"></div>  
<div class="content">  
 <div class="container">  
 <div class="material-box">  
 {% if user.is\_authenticated %}  
 <div class="user-box">  
 <div class="material-left">  
 <h2 class="main-title">Добавление материала к заявке</h2>  
 {% if form.errors %} *# если ошибки вывести*  
 <ul>  
 {% for error in form.errors %}  
 <li>{{ error }}</li>  
 {% endfor %}  
 </ul>  
 {% endif %}  
 <form method="post">  
 {% csrf\_token %}  
 <div class="attribute">  
 <label for="breaking">Заявка</label>  
 <select id="id\_breaking" name="breaking" required>  
 {% for choice in form.breaking.field.choices %}  
 <option value="{{ choice.0 }}"  
 {% if choice.0 == breaking.id %}selected{% endif %}>{{ choice.1 }}</option>  
 {% endfor %}  
 </select>  
 </div>  
 <div class="attribute">  
 <label for="material">Материал</label>  
 <div class="sub-attribute">  
 <label for="id\_material"></label>  
 <select id="id\_material" name="material" required>  
 {% for choice in form.material.field.choices %}   
 <option value="{{ choice.0 }}">{{ choice.1 }}</option>  
 {% endfor %}  
 </select>  
 <a class="btn" href="#open-modal">+</a>  
 </div>  
 </div>  
 <div class="attribute">  
 <label for="quantity">Количество</label>  
 <input type="number" id="id\_quantity" name="quantity" required min="0" max="9999"  
 step="any"/>  
 </div>  
 <button type="submit">Добавить</button>  
 </form>  
 </div>  
 </div>  
 <br>  
 <div class="user-box">  
 <h2>Привязанные материалы:</h2>  
 <ul class="material-el">  
 {% for material in materials %}  
 <li>  
 {{ material.material }}: {{ material.quantity }}  
 <form method="post"  
 action="{% url 'delete\_material\_breaking' breaking\_id=breaking.id material\_id=material.id %}">  
 {% csrf\_token %}  
 <button type="submit" class="btn btn-danger">X</button>  
 </form>  
 </li>  
 {% empty %}  
 <li>Нет связанных материалов</li> *# если нет материалов у заявки*  
 {% endfor %}  
 </ul>  
 </div>  
 {% endif %}  
 </div>  
 </div>   
 <div id="open-modal" class="modal-window"> *# модальное окно*  
 <div>  
 <a href="#" class="modal-close">X</a>  
 <h1>Добавление материала</h1>  
 <form method="post">  
 {% csrf\_token %} *# цсрф токен*  
 {{ mat\_form.as\_p }}  
 <button type="submit" class="modal-button">Добавить</button>  
 </form>  
 </div>  
 </div>  
 <a class="" href="{% url 'journal' %}">Выход</a>  
</div>  
</body>  
</html>

Profile.html:

*# загрузка статик файлов*

{% load static %}  
<!DOCTYPE html>  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Профиль</title>  
 <link rel="stylesheet" href="{% static 'css/animated-background.css' %}">  
 <link rel="stylesheet" href="{% static 'css/profile.css' %}">  
 <meta content="width=device-width, initial-scale=1" name="viewport"/>  
</head>  
<body> *# контент*  
<div class="bg"></div>  
<div class="bg bg2"></div>  
<div class="bg bg3"></div>  
<div class="content">  
 <div>  
 {% if user.is\_authenticated %} *# провека аутенфикации*  
 <p>Добро пожаловать, {{ user.username }}!</p> *# приветсвие*  
 <a href="{% url 'journal' %}">Журнал заявок</a>  
 <a href="{% url 'logout' %}"> Выйти из профиля</a>  
 {% endif %}  
 </div>  
</div>  
</body>  
</html>

Add\_table\_with\_headers:

def add\_table\_with\_headers(ws, title, headers, data, start\_date, end\_date):  
 from openpyxl.styles import Font, Alignment, PatternFill  
  
 *# Применение стилей и цветовой палитры* header\_font = Font(color="FFFFFF", bold=True, size=12)  
 header\_fill = PatternFill(start\_color='007ACC', end\_color='007ACC', fill\_type='solid')  
 cell\_font = Font(color="000000", size=11)  
 cell\_fill = PatternFill(start\_color='D9E1F2', end\_color='D9E1F2', fill\_type='solid')  
  
 start\_date\_str = datetime.strptime(start\_date, "%Y-%m-%d").strftime('%d-%m-%Y')  
 end\_date\_str = datetime.strptime(end\_date, "%Y-%m-%d").strftime('%d-%m-%Y')  
  
 *# Adding the header phrase* header\_phrase = f"«Детская городская клиническая поликлиника №8», отчет с {start\_date\_str} по {end\_date\_str}"  
  
 ws.append([header\_phrase])  
 ws.merge\_cells(start\_row=ws.max\_row, start\_column=1, end\_row=ws.max\_row, end\_column=len(headers))  
 for cell in ws.iter\_rows(min\_row=ws.max\_row, max\_row=ws.max\_row, min\_col=1, max\_col=len(headers)):  
 for c in cell:  
 c.font = Font(color="FFFFFF", size=12, bold=True) *# Чёрный цвет, 14 шрифт, не жирный* c.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
 c.fill = PatternFill(start\_color='007ACC', end\_color='007ACC', fill\_type='solid') *# Синий фон* ws.append([]) *# Empty row  
  
 # Установка ширины столбцов* column\_widths = [50, 50, 50]  
 for i, width in enumerate(column\_widths, 1):  
 ws.column\_dimensions[chr(64 + i)].width = width  
  
 *# Определение ширины заголовка* title\_width = len(headers)  
  
 *# Добавление заголовка* ws.append([title])  
 ws.merge\_cells(start\_row=ws.max\_row, start\_column=1, end\_row=ws.max\_row, end\_column=title\_width)  
 for cell in ws.iter\_rows(min\_row=ws.max\_row, max\_row=ws.max\_row, min\_col=1, max\_col=title\_width):  
 for c in cell:  
 c.font = header\_font  
 c.fill = header\_fill  
 c.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
  
 *# Добавление заголовков таблицы* ws.append(headers)  
 for cell in ws.iter\_rows(min\_row=ws.max\_row, max\_row=ws.max\_row, min\_col=1, max\_col=len(headers)):  
 for c in cell:  
 c.font = header\_font  
 c.fill = header\_fill  
 c.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
  
 *# Добавление данных* for row\_data in data:  
 ws.append(row\_data)  
 for row in ws.iter\_rows(min\_row=ws.max\_row - len(data) + 1, max\_row=ws.max\_row, min\_col=1, max\_col=len(headers)):  
 for cell in row:  
 cell.font = cell\_font  
 cell.fill = cell\_fill  
 cell.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
  
 ws.append([]) *# Пустая строка* return ws

Report.html:

*# загрузка статик файлов*

{% load static %}  
<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Отчет</title>  
 <link rel="stylesheet" href="{% static 'css/header.css' %}">  
 <link rel="stylesheet" href="{% static 'css/filter-bar.css' %}">  
 <link rel="stylesheet" href="{% static 'css/report.css' %}">  
 <link rel="stylesheet" href="{% static 'css/journal.css' %}">  
 <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>  
</head>  
<body>  
<div class="header"> *# шапка*  
 <div class="inner-header flex">  
 <img src="{% static 'assets/logo.png' %}">  
 <h1>8DGKP</h1>  
 </div>  
 <div>  
 <svg class="waves" xmlns="http://www.w3.org/2000/svg" *# волны* xmlns:xlink="http://www.w3.org/1999/xlink"  
 viewBox="0 24 150 28" preserveAspectRatio="none" shape-rendering="auto">  
 <defs>  
 <path id="gentle-wave" d="M-160 44c30 0 58-18 88-18s 58 18 88 18 58-18 88-18 58 18 88 18 v44h-352z"/>  
 </defs>  
 <g class="parallax">  
 <use xlink:href="#gentle-wave" x="48" y="0" fill="rgba(255,255,255,0.7"/>  
 <use xlink:href="#gentle-wave" x="48" y="3" fill="rgba(255,255,255,0.5)"/>  
 <use xlink:href="#gentle-wave" x="48" y="5" fill="rgba(255,255,255,0.3)"/>  
 <use xlink:href="#gentle-wave" x="48" y="7" fill="#fff"/>  
 </g>  
 </svg>  
 </div>  
 <div class="nav-bar"> *# навигация*  
 <a class="nav-bar-a" href="{% url 'index' %}">Главная</a>  
 {% if user.is\_authenticated %}  
 <a class="nav-bar-a" href="{% url 'profile' %}">Профиль</a>  
 <a class="nav-bar-a" href="{% url 'staff' %}">Персонал</a>  
 <a class="nav-bar-a" href="{% url 'journal' %}">Заявки</a>  
 {% endif %}  
 </div>  
</div>  
{% if user.is\_superuser %} *# провека на суперпользователя*  
 <div class="journal-button-container">  
 <button class="journal-button" id="toggleButton">Параметры</button> *# всплывающее меню*   
 </div>  
 <div id="filterFormContainer" class="filter-bar">   
 <div class="filter-box">  
 <form method="get">  
 <label for="start\_date">Начальная дата:</label>  
 <input type="date" name="start\_date" value="{{ request.GET.start\_date }}" required>  
  
 <label for="end\_date">Конечная дата:</label>  
 <input type="date" name="end\_date" value="{{ request.GET.end\_date }}" required>  
  
 <button type="submit">Сформировать</button>  
 </form>  
 </div>  
 </div>  
 <div class="report-container">  
 <div class="first-container"> *# 1-ый отчет*  
 <h2>Перечень заявок в разрезе временного периода</h2>  
 {% for breaking in breaking\_list %}  
 <div class="item">  
 <h2>{{ forloop.counter }}</h2>  
 <h3><b>Номер:</b> {{ breaking.breaking\_number }}:</h3>  
 <p><b></b>Дата:</b> {{ breaking.breaking\_date|date:"d-m-Y" }};</p>  
 <p><b>Категория:</b> {{ breaking.breaking\_category }};</p>  
 <p><b>Отвестенный:</b> {{ breaking.breaking\_worker\_fullname }};</p>  
 <p><b>Кабинет:</b> {{ breaking.breaking\_room }};</p>  
 <p><b></b>Описание:</b> {{ breaking.breaking\_description }};</p>  
 <p><b>Статус:</b> {{ breaking.breaking\_status }};</p>  
 </div>  
 {% endfor %}  
 </div>   
 <div class="second-container"> *# 2-ой отчет*  
 <h2>Отчет по статусу заявок</h2>  
 {% for stat in breaking\_statistics %}  
 <p>{{ stat.breaking\_status }}: {{ stat.count }}</p>  
 {% endfor %}  
  
 <canvas id="breakingChart" width="250" height="150"></canvas>  
 <script>  
 var ctx = document.getElementById('breakingChart').getContext('2d');  
 var labels = [{% for stat in breaking\_statistics %}"{{ stat.breaking\_status }}"{% if not forloop.last %}, {% endif %}{% endfor %}];  
 var data = [{% for stat in breaking\_statistics %}{{ stat.count }}{% if not forloop.last %}, {% endif %}{% endfor %}];  
  
 var breakingChart = new Chart(ctx, {  
 type: 'bar',  
 data: {  
 labels: labels,  
 datasets: [{  
 label: 'Статус заявок',  
 borderWidth: 1,  
 data: data  
 }]  
 },  
 options: {  
 scales: {  
 y: {  
 beginAtZero: true  
 }  
 }  
 }  
 });  
 </script>  
 </div>  
 <div class="third-container"> *# третий отчет*  
 <h2>Отчет по использованным материалам</h2>  
 <div class="wrapper"> *# рисуем таблицу*  
 <div class="table">  
 <div class="row header blue">  
 <div class="cell">  
 Материал  
 </div>  
 <div class="cell">  
 Количество  
 </div>  
 <div class="cell">  
 Использовался  
 </div>  
 </div>  
 {% for material in material\_aggregated %}  
 <div class="row">  
 <div class="cell">  
 {{ material.material\_\_material\_name }}  
 </div>  
 <div class="cell">  
 {{ material.total\_quantity }}  
 </div>  
 <div class="cell">  
 {{ material.breaking\_numbers }}  
 </div>  
 </div>  
 {% endfor %}  
 </div>  
 </div>  
 </div>  
 <form method="post">  
 {% csrf\_token %}  
 <input type="hidden" name="export\_pdf" value="true">  
 <button type="submit">Экспорт в Excel</button> *# экспорт в эксель*  
 </form>  
 </div>  
{% endif %}  
<script> *# скрип панели*  
 document.getElementById('toggleButton').addEventListener('click', function () {  
 var filterFormContainer = document.getElementById('filterFormContainer');  
 filterFormContainer.classList.toggle('visible');  
 });  
</script>  
</body>  
</html>

Staff.html:

*# загрузка статик файлов*

{% load static %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Персонал</title>  
 <link rel="stylesheet" href="{% static 'css/header.css' %}">  
 <link rel="stylesheet" href="{% static 'css/filter-bar.css' %}">  
 <link rel="stylesheet" href="{% static 'css/journal.css' %}">  
 <link rel="stylesheet" href="{% static 'css/paginator.css' %}">  
 <meta content="width=device-width, initial-scale=1" name="viewport"/>  
</head>  
<style>  
  
</style>  
<body>  
<div class="header"> *# шапка*  
 <div class="inner-header flex">  
 <img src="{% static 'assets/logo.png' %}">  
 <h1>8DGKP</h1>  
 </div>  
 <div>  
 <svg class="waves" xmlns="http://www.w3.org/2000/svg" *# волны* xmlns:xlink="http://www.w3.org/1999/xlink"  
 viewBox="0 24 150 28" preserveAspectRatio="none" shape-rendering="auto">  
 <defs>  
 <path id="gentle-wave"  
 d="M-160 44c30 0 58-18 88-18s 58 18 88 18 58-18 88-18 58 18 88 18 v44h-352z"/>  
 </defs>  
 <g class="parallax">  
 <use xlink:href="#gentle-wave" x="48" y="0" fill="rgba(255,255,255,0.7"/>  
 <use xlink:href="#gentle-wave" x="48" y="3" fill="rgba(255,255,255,0.5)"/>  
 <use xlink:href="#gentle-wave" x="48" y="5" fill="rgba(255,255,255,0.3)"/>  
 <use xlink:href="#gentle-wave" x="48" y="7" fill="#fff"/>  
 </g>  
 </svg>  
 </div>  
 <div class="nav-bar"> *# навигация*  
 <a class="nav-bar-a" href="{% url 'index' %}">Главная</a>  
 {% if user.is\_authenticated %}  
 <a class="nav-bar-a" href="{% url 'profile' %}">Профиль</a>  
 {% endif %}  
 {% if user.is\_superuser %}  
 <a class="nav-bar-a" href="{% url 'journal' %}">Заявки</a>  
 <a class="nav-bar-a" href="{% url 'report' %}">Отчетность</a>  
 {% endif %}  
 </div>  
</div>  
{% if user.is\_superuser %} *# проверка суперпользователя*  
 <div class="journal-button-container">  
 <button class="journal-button" id="toggleButton">Параметры</button>  
 </div>  
 <div id="filterFormContainer" class="filter-bar">  
 <div class="filter-box">  
 <form method="GET">  
 {% if user.is\_superuser %}  
 <select id="category" name="category">  
 <option value="">Все категории</option>  
 {% for category in categories %}  
 <option value="{{ category }}">{{ category }}</option>  
 {% endfor %}  
 </select>  
 {% endif %}  
 <button type="submit">Применить фильтр</button>  
 </form>  
 </div>  
 <div class="search-box"> *# поиск*  
 <form method="GET">  
 <input type="text" name="search" placeholder="Поиск..." value="{{ request.GET.search }}"  
 autocomplete="off">  
 <button type="submit">Найти</button>  
 </form>  
 </div>  
 <h2>Добавление нового сотрудника</h2>  
 <form method="post">  
 {% csrf\_token %} *# цсрф токен*  
 {{ form.as\_p }}  
 <button type="submit">Добавить</button>  
 </form>  
 </div>  
 <div class="wrapper"> *# таблица*  
 <div class="table">  
 <div class="row header blue">  
 <div class="cell">  
 ФИО  
 </div>  
 <div class="cell">  
 Тел. номер  
 </div>  
 <div class="cell">  
 Категория  
 </div>  
 <div class="cell" style="justify-content: center">  
 Действие  
 </div>  
 </div>  
 {% for staff in staff\_list %} *# вывод персонала*  
 <div class="row">  
 <div class="cell" data-title="ФИО">{{ staff.staff\_fullname }}</div>  
 <div class="cell" data-title="Тел. номер">{{ staff.staff\_phone\_number }}</div>  
 <div class="cell" data-title="Категория">{{ staff.staff\_category }}</div>  
 <div class="cell" data-title="Действия">  
 <div class="button-row">  
 <div class="edit-btn">  
 <div class="button-row">  
 <a href="{% url 'edit\_staff' staff.id %}" class="action-a">Редактировать</a>  
 </div>  
 <form method="post"  
 action="{% url 'delete\_staff' staff\_id=staff.id staff\_id=staff.id %}">  
 {% csrf\_token %}  
 <a class="action-a"><button type="submit">Удалить</button></a>  
 </form>  
 </div>  
 </div>  
 </div>  
 </div>  
 {% empty %}  
 <h1>Таблица пуста.</h1>  
 {% endfor %}  
 </div>  
 </div>  
 <div class="pagination"> *# пагинация*  
 <span>  
 {% if staff\_list.has\_previous %}  
 <a href="?page=1"><<</a>  
 <a href="?page={{ staff\_list.previous\_page\_number }}">Предыдущая</a>  
 {% endif %}  
  
 <span class="current"> *# проверка текущей страницы*  
 Стр. {{ staff\_list.number }} из {{ staff\_list.paginator.num\_pages }}.  
 </span>  
  
 {% if staff\_list.has\_next %}  
 <a href="?page={{ staff\_list.next\_page\_number }}">Следующая</a>  
 <a href="?page={{ staff\_list.paginator.num\_pages }}">>></a>  
 {% endif %}  
 </span>  
 </div>  
 <script>*# скрипт панели*  
 document.getElementById('toggleButton').addEventListener('click', function () {  
 var filterFormContainer = document.getElementById('filterFormContainer');  
 filterFormContainer.classList.toggle('visible');  
 });  
 </script>  
{% endif %}  
</body>  
</html>

Sign\_In.html:

*# загрузка статик файлов*

{% load static %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Авторизация</title>  
 <link rel="stylesheet" href="{% static 'css/form.css' %}">  
 <link rel="stylesheet" href="{% static 'css/animated-background.css' %}">  
 <meta content="width=device-width, initial-scale=1" name="viewport"/>  
</head>  
<body>  
<div class="bg"></div>  
<div class="bg bg2"></div>  
<div class="bg bg3"></div>  
<div class="content">  
 <div class="login-box"> *# контент*  
 <h2>Авторизация</h2>  
 <form method="post">  
 {% csrf\_token %} *# цсрф токен*  
 <div class="user-box">  
 {{ form.username }}  
 <label>Логин</label>  
 </div>  
 <div class="user-box">  
 {{ form.password }}  
 <label>Пароль</label>  
 </div>  
 <a>  
 <span></span>  
 <span></span>  
 <span></span>  
 <span></span>  
 <button type="submit">Войти</button>  
 <a onclick="cancelLogin()"> *# возврат на прошлую стр*  
 <button>Отмена</button>  
 </a>  
 </a>  
 </form>  
 </div>  
</div>  
<script>  
function cancelLogin() { *# скрипт на джиэс*  
 *// Add any specific cancel logic here, such as redirecting to another page* window.location.href = "{% url 'index' %}";  
}  
</script>  
</body>  
</html>

**ПРИЛОЖЕНИЕ Б**

**(справочное)**

**Результаты работы приложения**

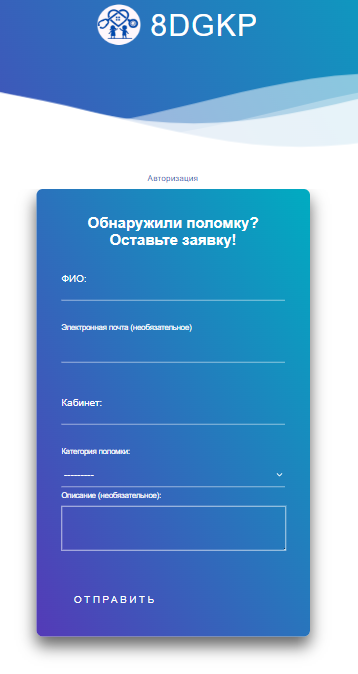


Рисунок Б.1 – Главная страница для неавторизированного пользователя

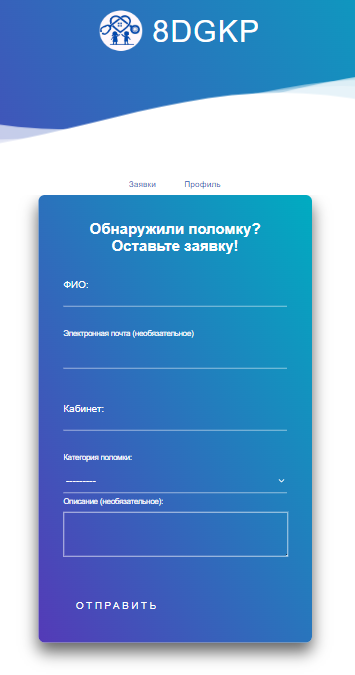


Рисунок Б.2 – Главная страница для авторизированного пользователя

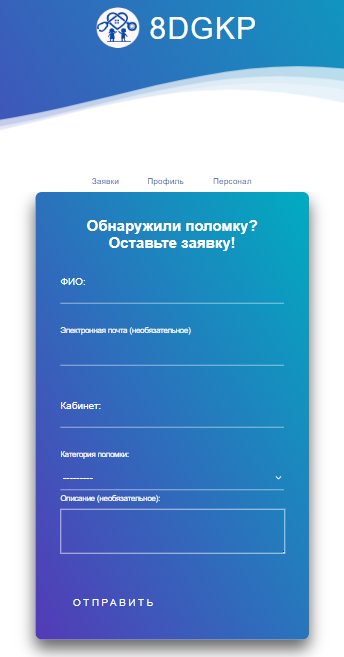


Рисунок Б.3 – Главная страница для суперпользователя

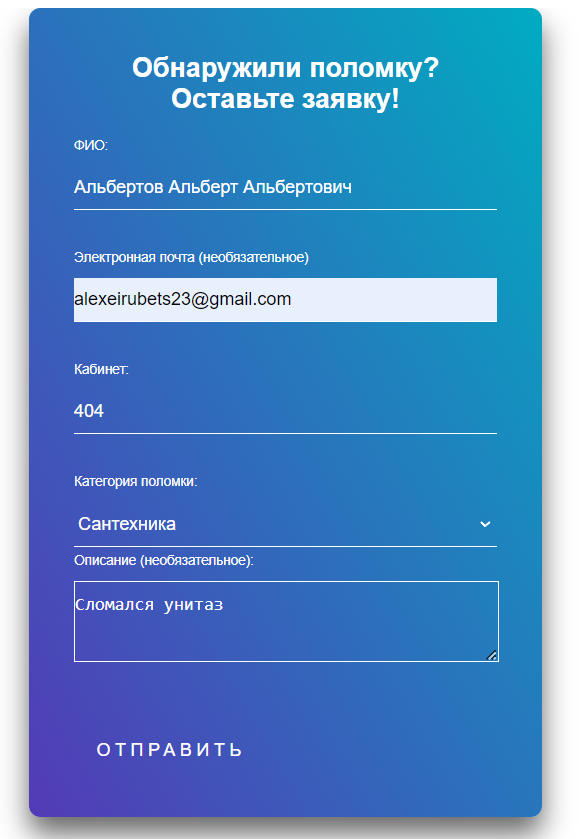


Рисунок Б.4 – Заполоненная форма подачи заявки

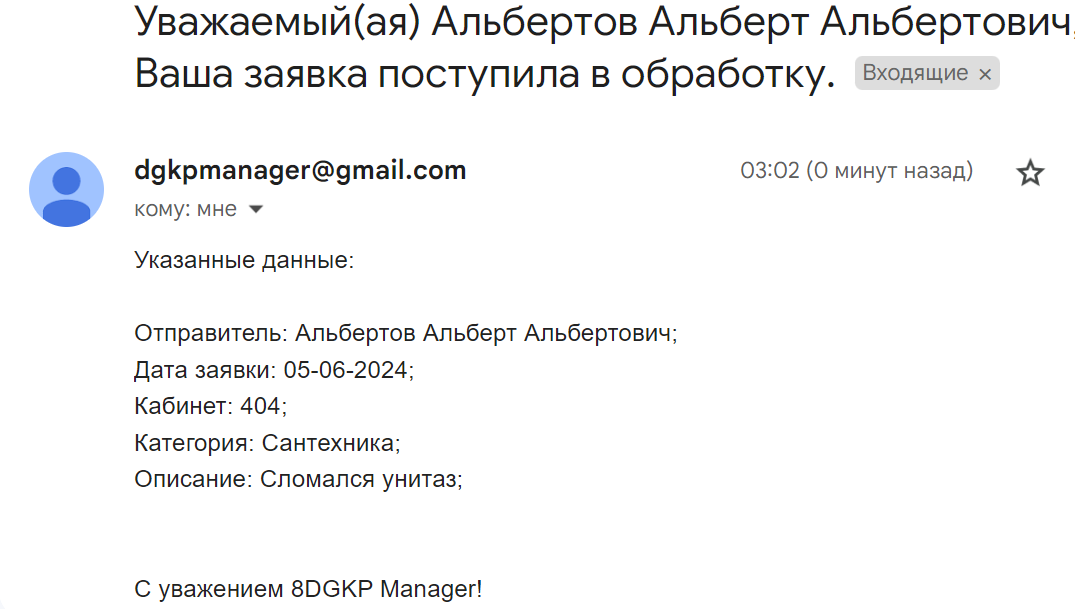


Рисунок Б.5 – Результат подачи заявки

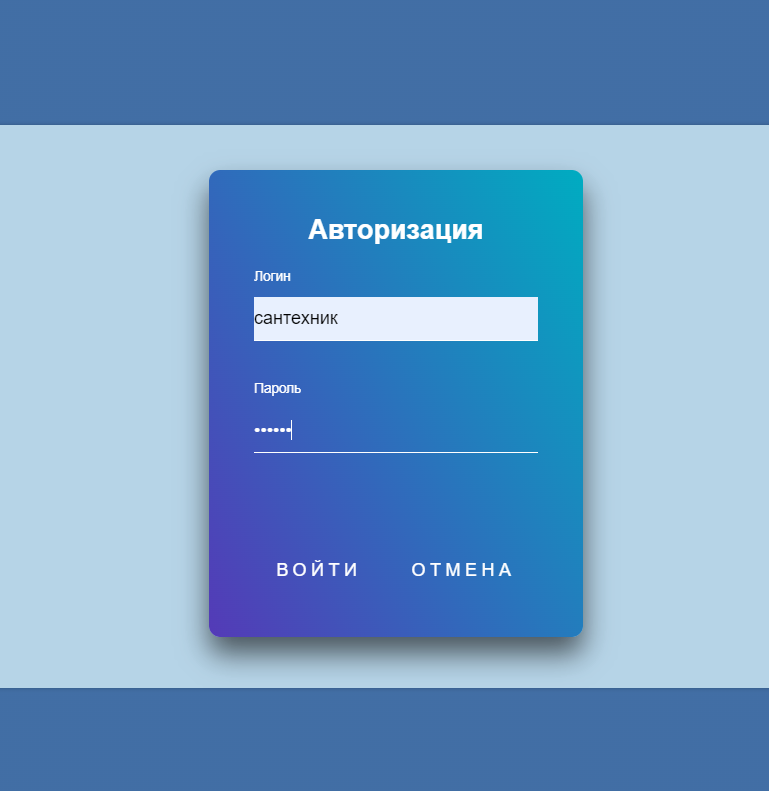


Рисунок Б.6 – Заполненная форма авторизации

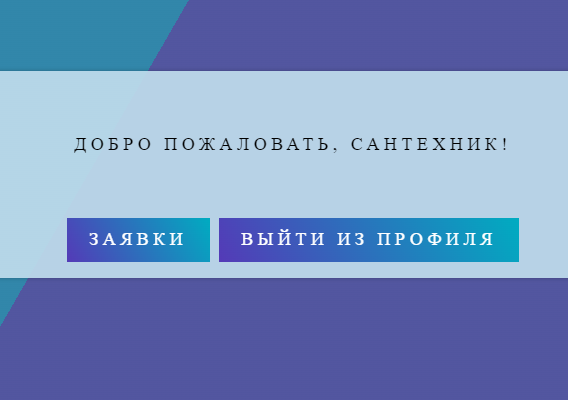


Рисунок Б.7 – Результат авторизации

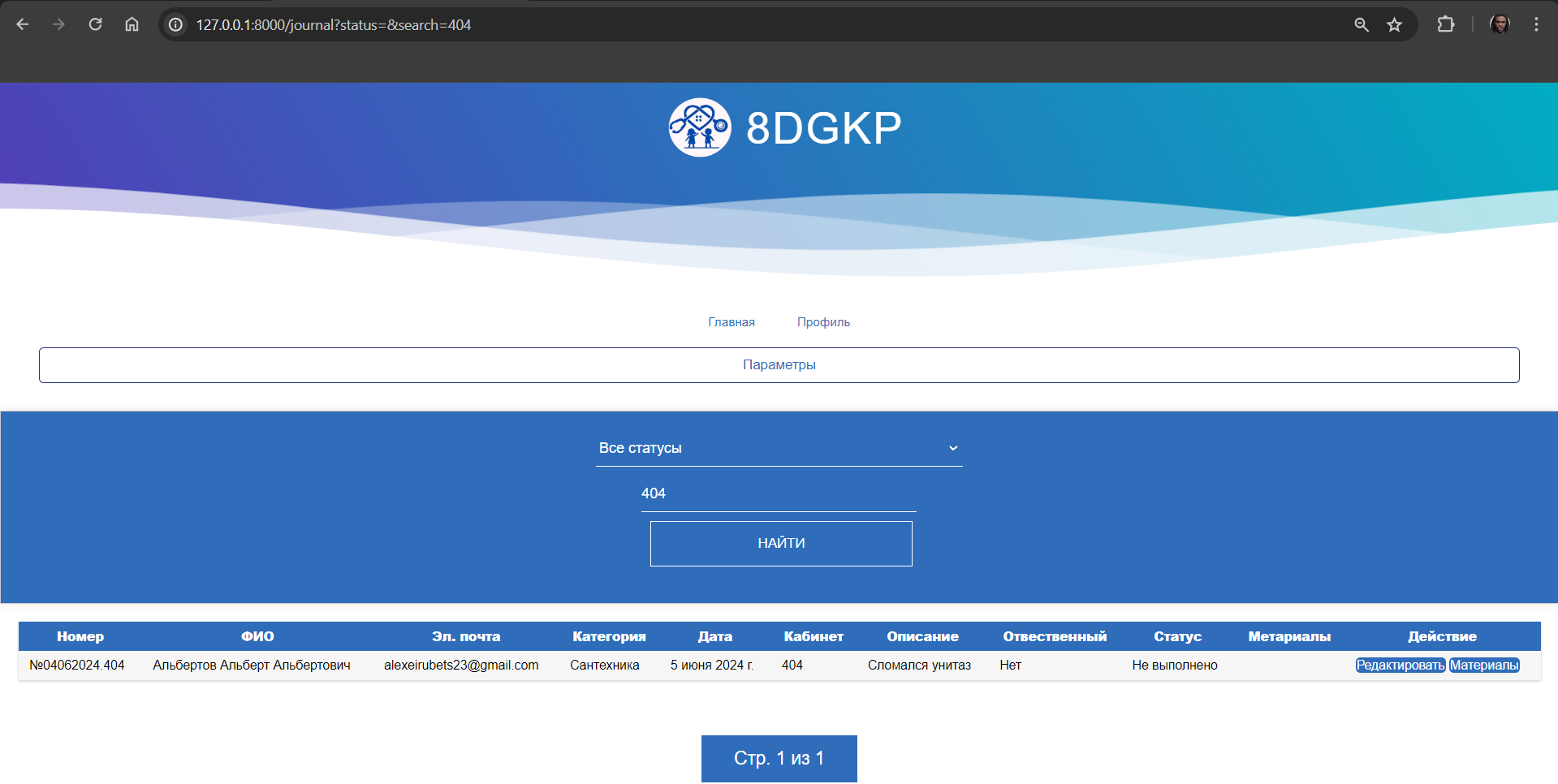


Рисунок Б.8 – Результат поиска заявок

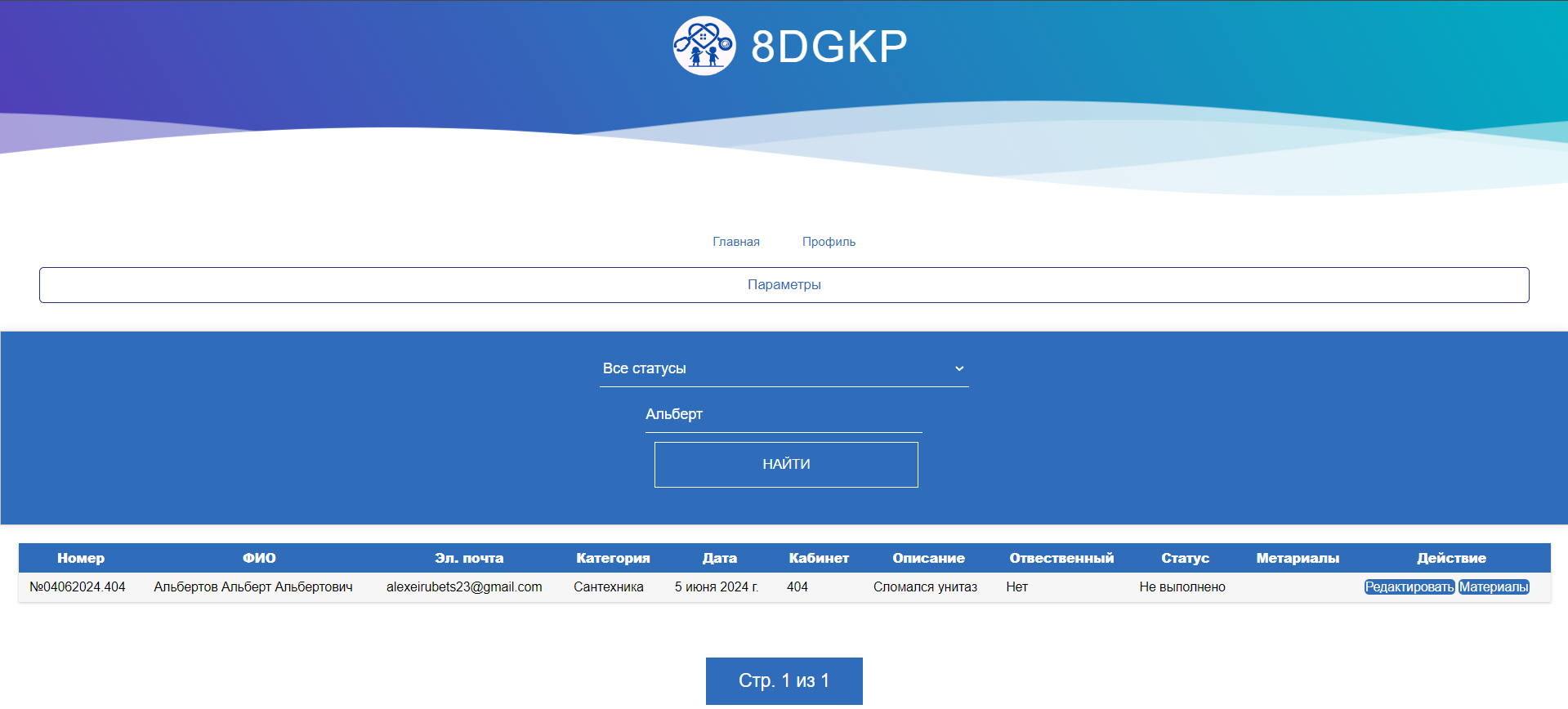


Рисунок Б.9 – Список найденных заявок



Рисунок Б.10 – Результат редактирования данных

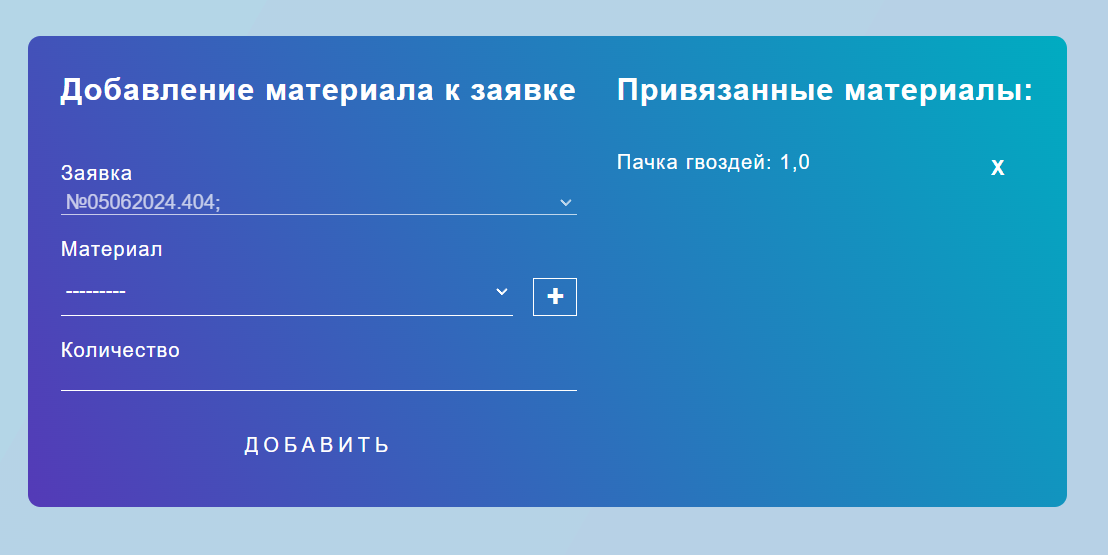


Рисунок Б.11 – Результат добавления материала

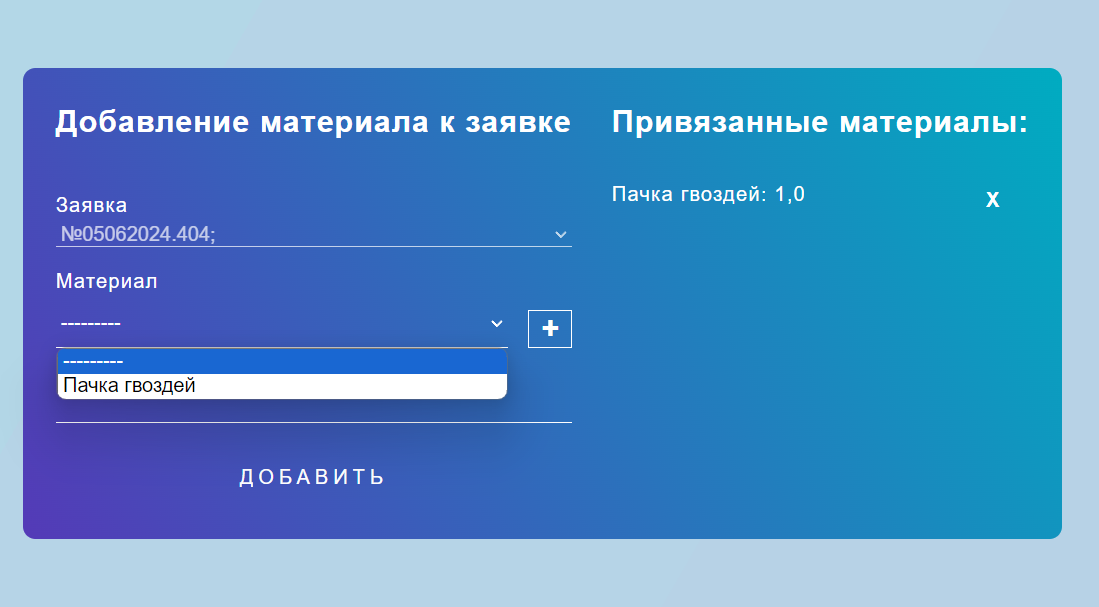


Рисунок Б.12 – Список доступных материалов

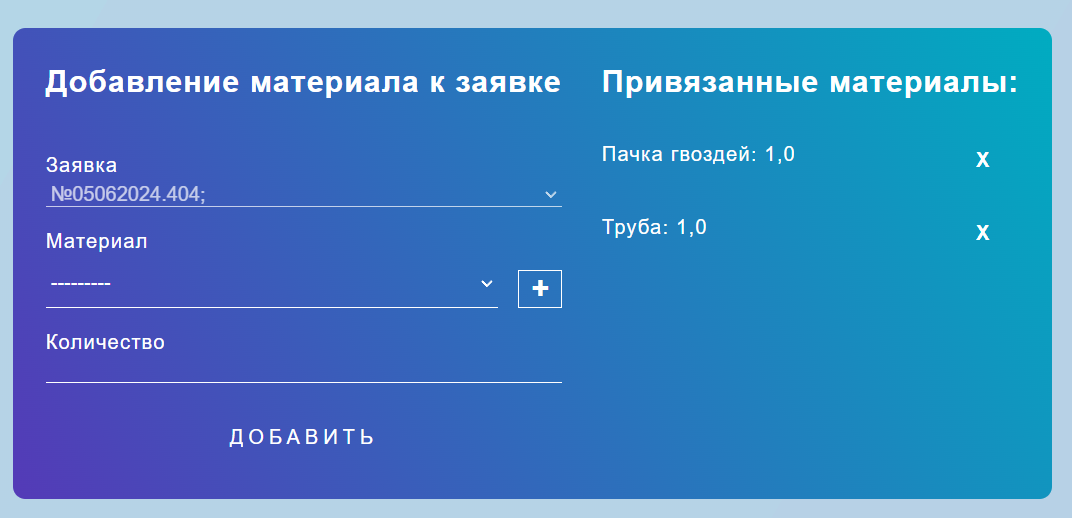


Рисунок Б.13 – Результат добавления нового материала

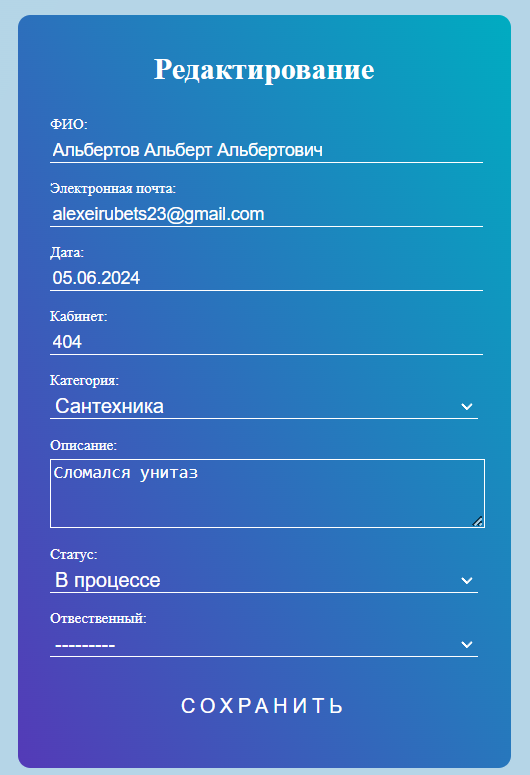


Рисунок Б.14 – Форма редактирования заявки

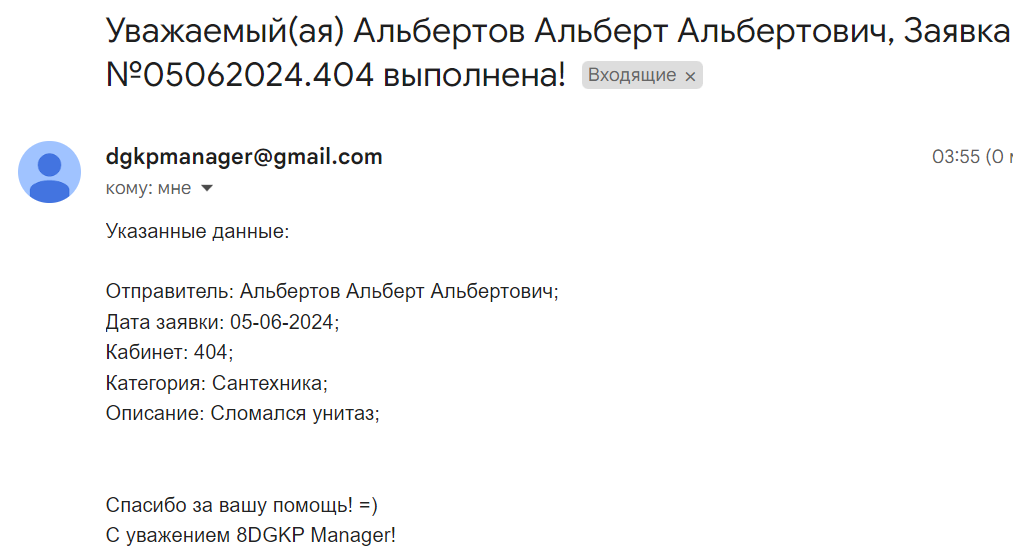


Рисунок Б.15 – Результат уведомления о готовности заявки

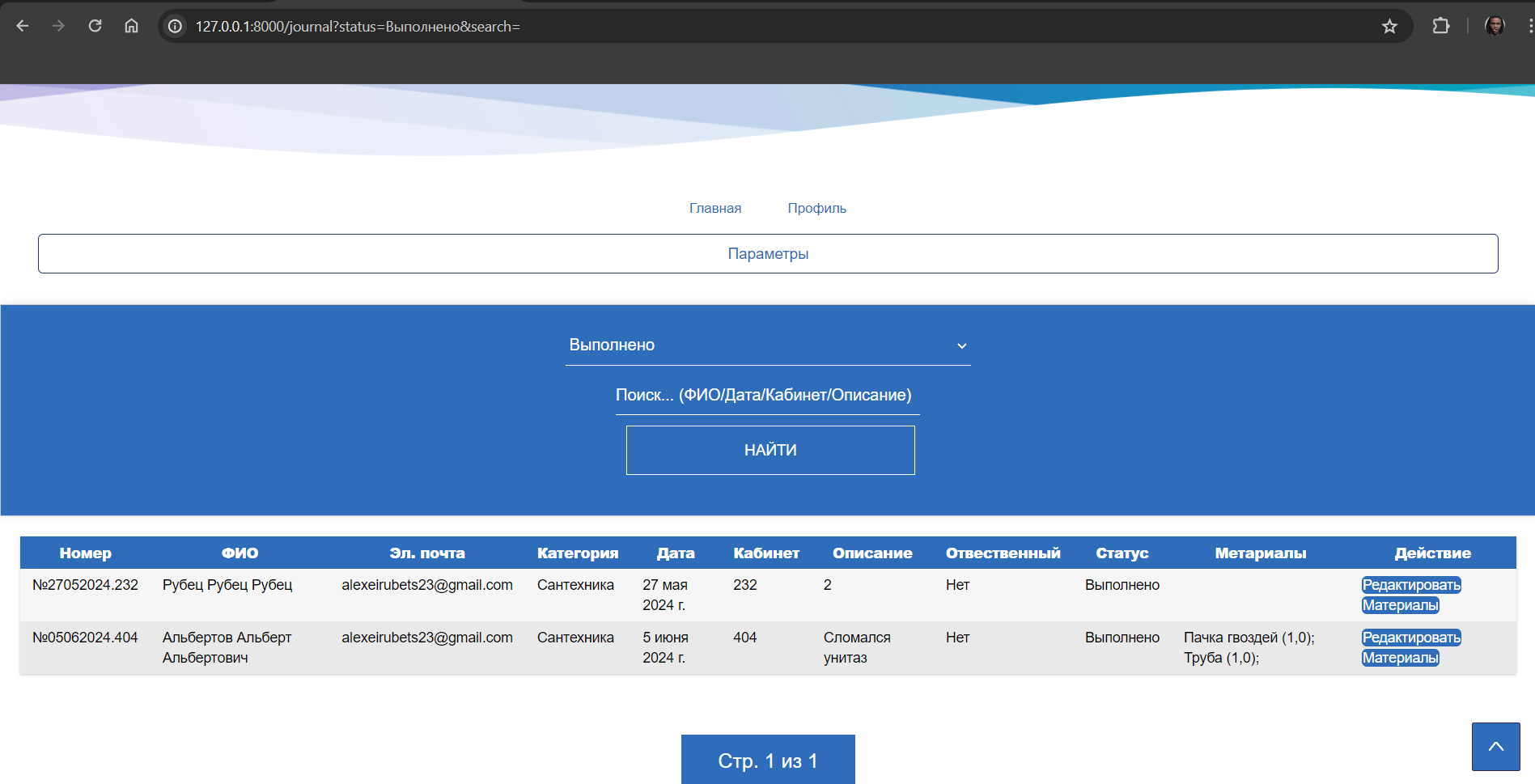


Рисунок Б.16 – Результат фильтрации данных

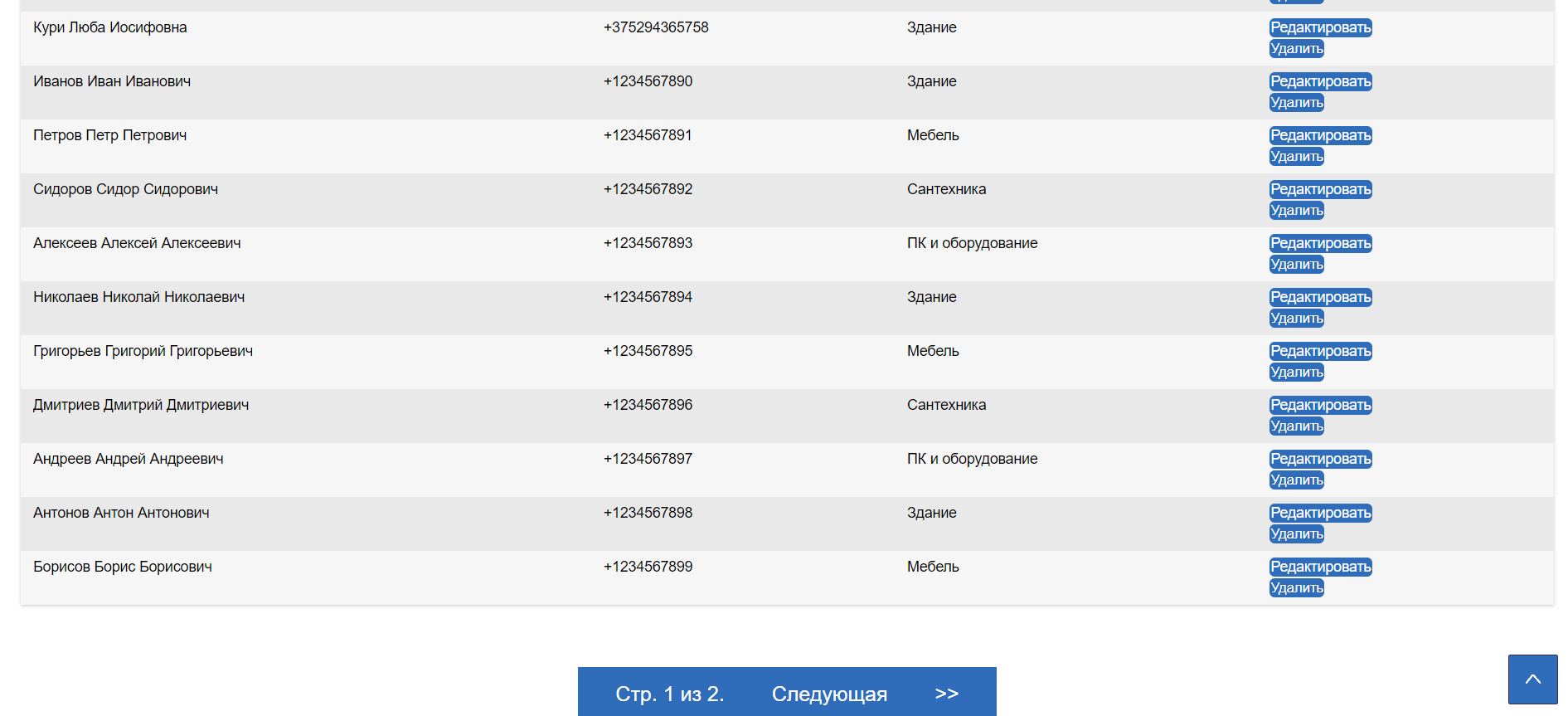


Рисунок Б.17 – Результат перехода на страницу персонала

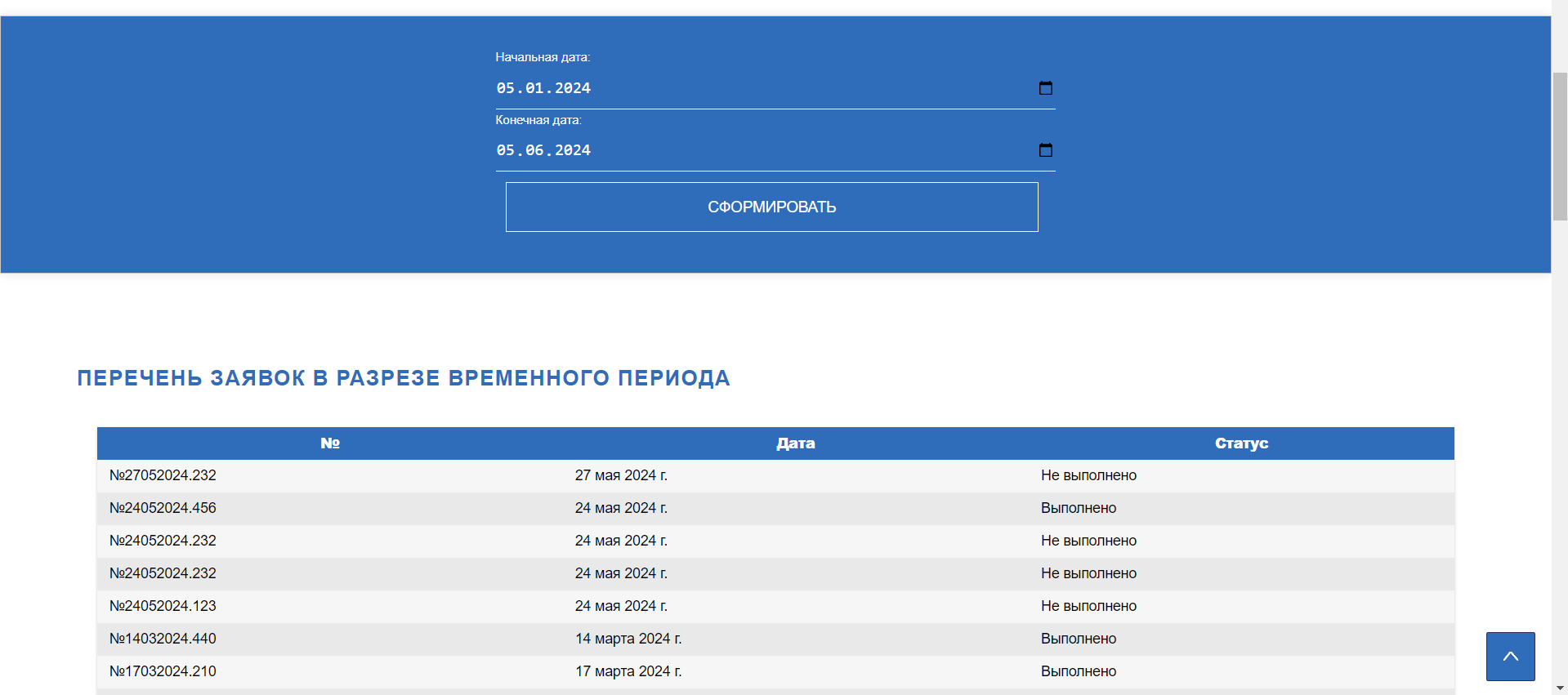


Рисунок Б.18 – Результат формирования отчета

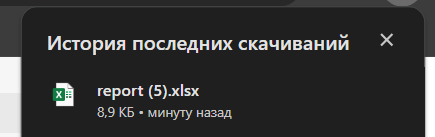


Рисунок Б.19 – Результат экспорта отчета

КБП

У

Т. Контр.

Разраб. Разраб.

Белисова В.А.

Провер. Провер.

Реценз.

Утверд.

Банцевич С.В.

Н. Контр.

Концептуальная модель диаграммы «Сущность-связь»

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 2

Листов 3

Инв.№подл.

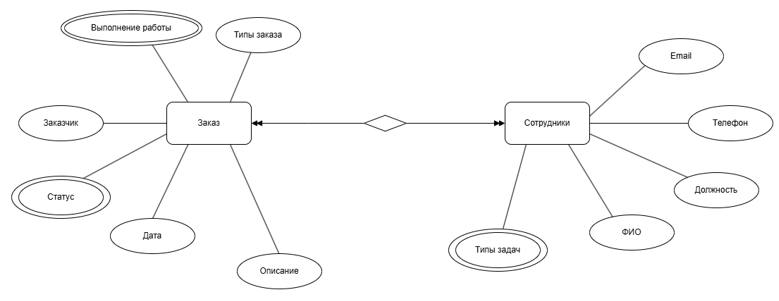
Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.218001.401 ГЧ



КБП

У

Т. Контр.

Разраб. Разраб.

Белисова В.А.

Провер. Провер.

Реценз.

Утверд.

Н. Контр.

Диаграмма вариантов использования

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 3

Листов 3

Инв.№подл.

Подп. и дата

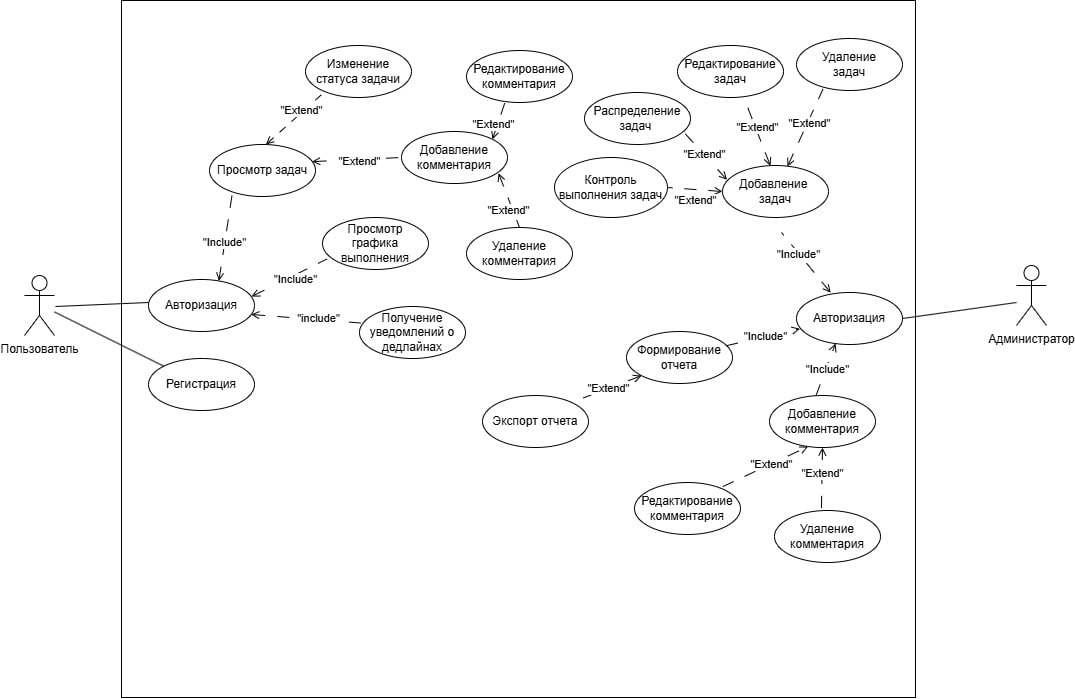
Взам.инв.№

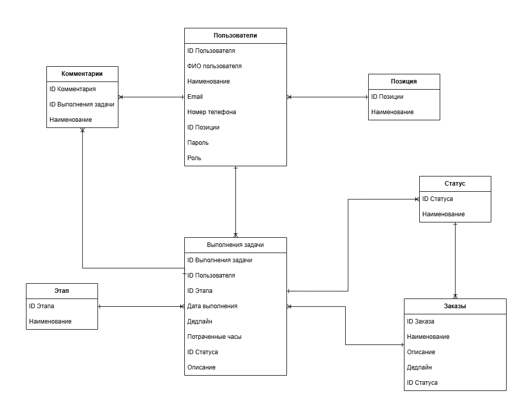
Инв.№дубл.

Подп. и дата

КП Т.218001.401 ГЧ

Банцевич С.В.





КБП

У

Т. Контр.

Разраб. Разраб.

Белисова В.А.

Провер. Провер.

Реценз.

Утверд.

Н. Контр.

Логическая модель базы данных диаграммы «сущность-связь»

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 3

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.218001.401 ГЧ

Банцевич С.В.

КБП

У

Т. Контр.

Разраб. Разраб.

Белисова В.А.

Провер. Провер.

Реценз.

Утверд.

Н. Контр.

Диаграмма вариантов использования

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 3

Листов 3

Инв.№подл.

Подп. и дата

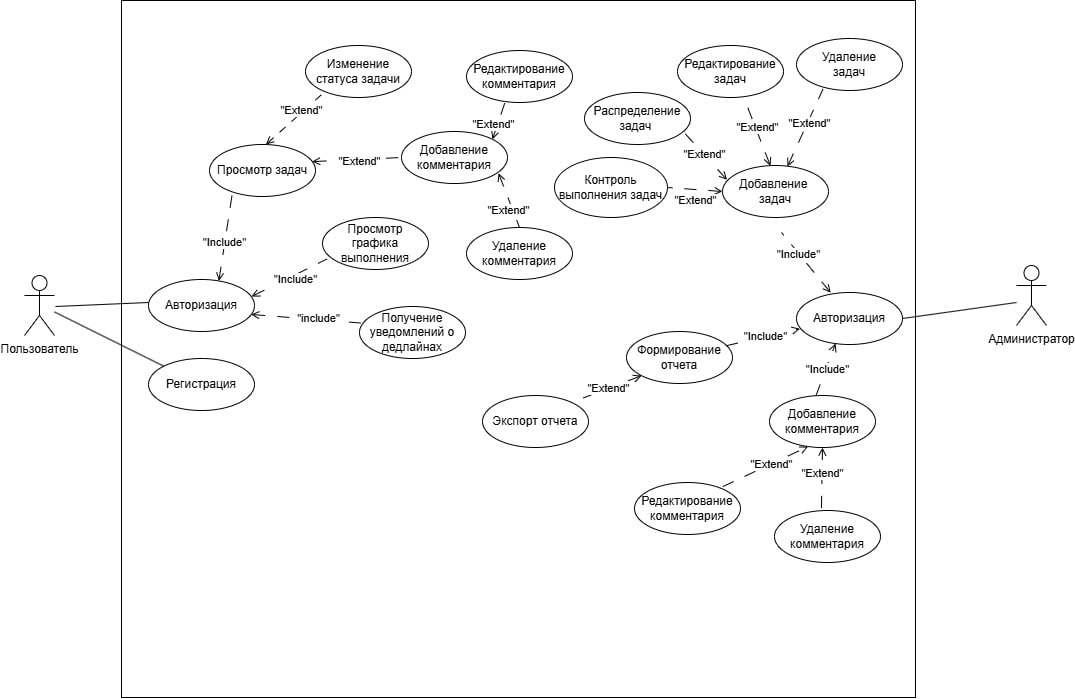
Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.218001.401 ГЧ

Банцевич С.В.



Банцевич С.В.

КБП

У

Т. Контр.

Разраб. Разраб.

Белисова В.А.

Провер. Провер.

Реценз.

Утверд.

Н. Контр.

Логическая модель базы данных диаграммы «сущность-связь»

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 3

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.218001.401 ГЧ

